CHAPITRE III

Réseaux d'ondelettes (approche fondée sur la transformée continue)

I. INTRODUCTION.

Le terme *ondelette* désigne une fonction qui oscille pendant un " temps donné" (si la variable est le temps) ou sur un intervalle de longueur finie (si la variable est de type spatial). Au delà, la fonction décroît très vite vers zéro.

Historiquement, les premières ondelettes (introduites par Haar dans les années 1930) constituaient une base de fonctions orthogonales. Les ondelettes de Haar présentent la particularité de ne pas être dérivables.

Plus récemment, de nouvelles fonctions ondelettes ont été introduites [Meyer85, Meyer90], qui constituent également une base de fonctions orthogonales, et qui, de plus, sont dérivables. Elles ont été notamment mises en œuvre dans le cadre de l'analyse multirésolution de signaux [Mallat89]. Ces ondelettes ne peuvent s'exprimer sous une forme analytique simple. Pour cette raison, elles sont peu adaptées pour l'approximation de fonctions. Nous n'utiliserons donc pas les ondelettes orthogonales dans ce mémoire.

Les structures obliques (*frames* en anglais) ont été introduites par J. Morlet dans le but de trouver des bases de fonctions (non nécessairement orthogonales) pour représenter des signaux. Ces structures obliques ont été ensuite l'objet des travaux de I. Daubechies [Daubechies90] qui a développé un support théorique aux résultats de J. Morlet. Les structures obliques ont des expressions analytiques simples, et toute fonction de carré sommable peut être approchée, avec la précision voulue, par une somme finie d'ondelettes issues d'une structure oblique. Cette propriété est équivalente à celle de l'approximation universelle pour les réseaux de fonctions dorsales. Pour toutes ces raisons, nous nous sommes intéressés uniquement, dans notre travail, à des structures obliques d'ondelettes.

Dans ce chapitre, nous présentons tout d'abord les fonctions ondelettes et la transformée en ondelettes. Deux approches sont à considérer : la transformée en ondelettes continue et la transformée en ondelettes discrète, comme illustré par la Figure 1.

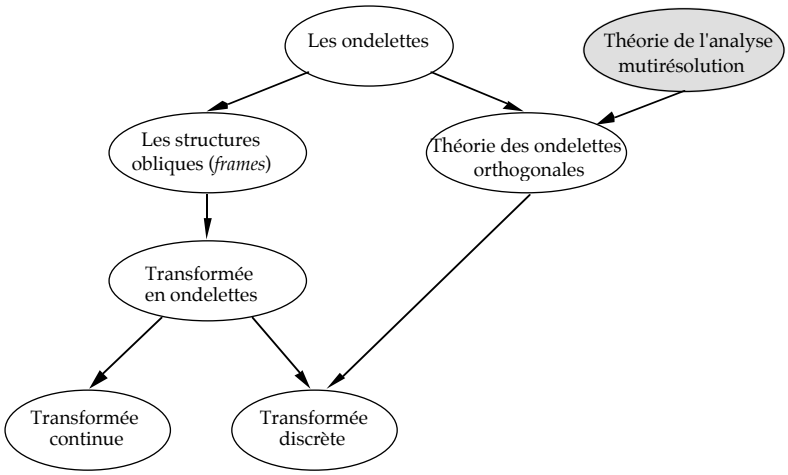


Figure 1.

Le présent chapitre est consacré aux ondelettes utilisées pour la transformée continue et aux réseaux de telles ondelettes. Nous décrivons en détail la technique de modélisation statique par réseaux d'ondelettes, et nous introduisons la modélisation dynamique par ces réseaux ; nous montrons qu'il est possible de considérer soit des réseaux entrée-sortie, soit des réseaux d'état.

II. RÉSEAUX ISSUS DE LA TRANSFORMÉE EN ONDELETTES CONTINUE.

De manière analogue à la théorie des séries de Fourier, les ondelettes sont principalement utilisées pour la décomposition de fonctions. La décomposition d'une fonction en ondelettes consiste à l'écrire comme une somme pondérée de fonctions obtenues à partir d'opérations simples effectuées sur une fonction principale appelée *ondelette–mère*. Ces opérations consistent en des translations et dilatations de la variable. Selon que ces translations et dilatations sont choisies de manière continue ou discrète, on parlera d'une transformée en ondelettes continue ou discrète.

II.1 La transformée en ondelettes continue.

Une transformée en ondelettes est dite continue lorsque les paramètres structurels des fonctions utilisées (c'est-à-dire les translations et les dilatations) peuvent prendre n'importe quelle valeur de l'ensemble des réels $\boldsymbol{R}$ (les dilatations doivent néanmoins être positives).

Soit ϕ une ondelette–mère, x la variable, m_j le paramètre de translation et d_j le paramètre de dilatation. L'ondelette ϕ_j de la famille de ϕ ayant pour paramètres m_j et d_j a pour expression :

$$\phi_j(x) = \frac{1}{\sqrt{d_j}}\,\phi\left(\frac{x \pm m_j}{d_j}\right) \qquad (1)$$

avec $m_j \in \boldsymbol{R}$ et $d_j \in \boldsymbol{R}^*_+$.

On constitue ainsi une famille d'ondelettes engendrée à partir de l'ondelette-mère. On la note Ω. On a alors la définition suivante :

$$\boldsymbol{\Omega} = \left\{ \frac{1}{\sqrt{d_j}} \phi\left(\frac{x \pm m_j}{d_j}\right), m_j \in R \text{ et } d_j \in R^*_+ \right\} \tag{2}$$

Comme les réseaux d'ondelettes auxquels nous allons nous intéresser sont issus de la transformée en ondelettes continue, nous allons présenter brièvement celle–ci.

Soient f et g deux fonctions ; on définit leur produit scalaire par l'intégrale suivante :

$$<f, g> = \int_R f(x)\, g(x)\, dx \tag{3}$$

Pour que la transformée en ondelettes d'une fonction existe, il faut que cette fonction appartienne à l'ensemble des fonctions de carré sommable que l'on note par $L^2(\boldsymbol{R})$. Autrement dit, il faut que son carré soit fini. Cette condition se traduit par :

$$\int_R f^2(x)\, dx < \infty \tag{4}$$

Dans ces conditions, la transformée en ondelette continue de la fonction f est définie comme le produit scalaire de f et de ϕ [Cohen96] :

$$W(m, d) = \frac{1}{\sqrt{d}} \int_R f(x)\, \phi\left(\frac{x \pm m}{d}\right) dx \tag{5}$$

La famille Ω doit constituer une *structure oblique* de l'ensemble $L^2(\boldsymbol{R})$ et y être dense. Cette propriété est assurée par l'existence de deux constantes $c > 0$ et $C < \infty$ telles que, pour toute fonction f pour laquelle il existe une transformée en ondelettes, on ait l'inégalité suivante :

$$c\,|f|^2 \le \sum_{\phi_j \in \Omega} \left| < \phi_j, f > \right|^2 \le C\,|f|^2 \tag{6}$$

De ce fait, toute combinaison linéaire d'un nombre fini d'éléments de la famille Ω est dense dans $L^2(\boldsymbol{R})$. Ceci garantit également que cette famille de fonctions possède la propriété d'approximation universelle définie dans le chapitre I du présent mémoire [Zhang92].

La reconstruction de la fonction f à partir de sa transformée est possible dans le cas où l'intégrale suivante est convergente :

$$C_\phi = \int_0^\infty \frac{|\hat{\phi}(\omega)|^2}{\omega} d\omega \tag{7}$$

où $\hat{\phi}$ est la transformée de Fourier de ϕ. Cette dernière condition est également appelée critère d'admissibilité pour une ondelette. Dans ce cas, f peut être reconstruite à partir de la relation suivante :

$$f(x) = \frac{1}{C_\phi} \int_{\mathbb{R}} \int_{\mathbb{R}_+} W(m, d) \frac{1}{\sqrt{d}} \phi\left(\frac{x \pm m}{d}\right) \mathrm{d}d \, \mathrm{d}m \tag{8}$$

La condition (7) est très intéressante dans la mesure où elle donne des informations sur les propriétés que doit vérifier une ondelette mère (si l'on souhaite que la reconstruction de la fonction transformée soit possible). En particulier, on doit avoir $\phi(0)= 0$. En remplaçant ω par 0 dans la définition de la transformée de Fourier de ϕ, on voit que cette condition est équivalente à :

$$\int_R \phi(x)\, dx = 0 \tag{9}$$

Donc, une ondelette est une fonction à support de longueur finie et d'intégrale nulle. Ainsi, les gaussiennes radiales ne peuvent pas être considérées comme des ondelettes.

II.2 De la transformée inverse aux réseaux d'ondelettes.

La relation (8) donne l'expression d'une fonction f de carré sommable sous la forme d'une intégrale sur toutes les dilatations et toutes les translations possibles de l'ondelette mère. Supposons que l'on ne dispose que d'un nombre fini N_w d'ondelettes ϕ_j obtenues à partie de l'ondelette mère ϕ. On peut alors considérer la relation

$$f(x) \approx \sum_{j=1}^{N_w} c_j \, \phi_j(x) \tag{10}$$

comme une approximation de la relation (8). La somme finie de la relation (10) est donc une approximation d'une transformée inverse. Elle peut être vue aussi comme la décomposition d'une fonction en une somme pondérée d'ondelettes, où chaque poids c_j est proportionnel à $W(m_j, d_j)$. Si l'on cherche à réaliser une approximation d'une fonction définie sur un domaine fini (donc de carré sommable), la transformée en ondelette de cette fonction existe, et sa reconstruction est possible.

Dans le cadre de la modélisation boîte noire, la fonction que l'on veut approcher (la fonction de régression de la grandeur à modéliser) n'est pas connue : on ne dispose que des points de mesure, en nombre fini. On peut alors chercher à obtenir une approximation de la fonction de régression sous la forme (10), où les coefficients c_j, ainsi que les paramètres m_j et d_j des ondelettes, doivent être estimés à partir des données disponibles.

C'est dans cette perspective qu'a été proposée l'idée de réseaux d'ondelettes. Ces réseaux ont été introduits pour la première fois à la même époque dans [Zhang92, Pati93].

III. DÉFINITION DES ONDELETTES MULTIDIMENSIONNELLES ET DES RÉSEAUX D'ONDELETTES.

III.1 Ondelettes multidimensionnelles.

Dans le paragraphe précédent, nous avons présenté les ondelettes à une dimension. Dans le cadre de la modélisation, il est fréquent d'avoir affaire à des processus multivariables ; il est donc utile d'introduire la notion d'ondelette multidimensionnelle.

On peut définir une ondelette multidimensionnelle comme le produit d'ondelettes monodimensionnelles : on dit alors que les ondelettes sont séparables. Dans ce cas, l'expression d'une ondelette multidimensionnelle est :

$$\Phi_j(x) = \prod_{k=1}^{N_i} \phi(z_{jk}) \quad \text{avec} \quad z_{jk} = \frac{x_k \pm m_{jk}}{d_{jk}} \tag{11}$$

où x_k est la k-ième composante du vecteur d'entrée x, et z_{jk} la composante centrée par m_{jk} et dilatée d'un facteur d_{jk}. Il a été montré dans [Kuga95] que ces ondelettes multidimensionnelles sont des structures obliques de $L^2(\boldsymbol{R}^{Ni})$.

III.2 Réseaux d'ondelettes.

Dans le présent travail, nous considérons des réseaux d'ondelettes de la forme suivante :

$$y = \psi(x) = \sum_{j=1}^{N_w} c_j \Phi_j(x) + \sum_{k=0}^{N_i} a_k x_k \quad \text{avec} \quad x_0=1 \tag{12}$$

où y est la sortie du réseau et $x = \{x_1, x_2, ..., x_{N_i}\}$ le vecteur des entrées ; il est souvent utile de considérer, outre la décomposition en ondelettes proprement dite, que la sortie peut avoir une composante affine par rapport aux variables, de coefficients a_k (k = 0, 1, ... , N_i). Pour la simplicité de l'exposé, nous ne considérerons que des réseaux à une sortie ; la généralisation à des réseaux à plusieurs sorties ne présente pas de difficulté.

Par analogie avec les réseaux de fonctions dorsales discutés dans le chapitre II, on peut représenter une ondelette de manière analogue à un neurone, comme indiqué sur la figure 2.

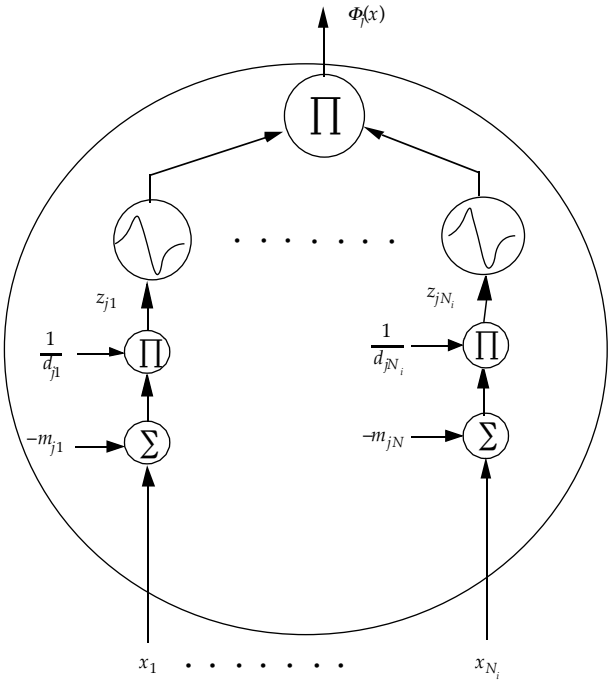


Figure 2. Représentation graphique d'une ondelette multidimensionnelle séparable.

Le réseau peut être considéré comme constitué de trois couches. Une première couche avec N_i entrée(s), une couche cachée constituée par N_w ondelettes et un sommateur (ou neurone linéaire) de sortie recevant les sorties pondérées des ondelettes multidimensionnelles et la partie affine. Ce réseau est illustré par la figure 3.

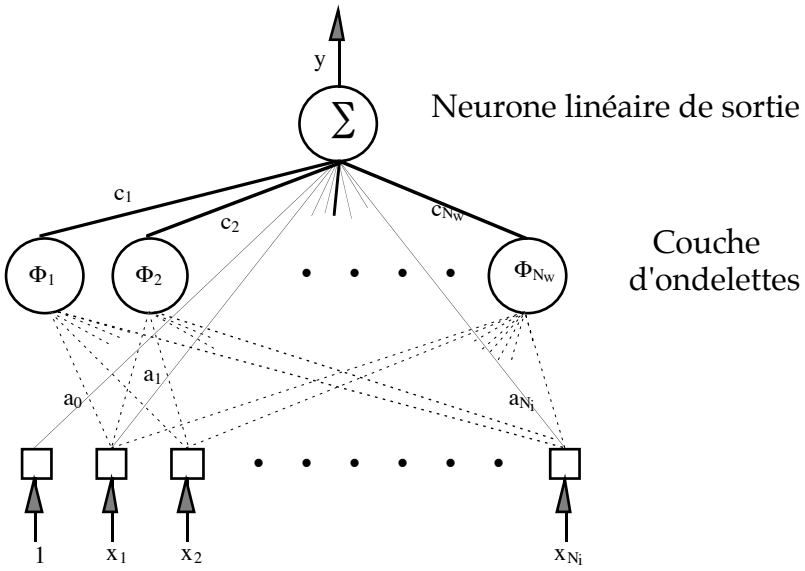


Figure 3. Représentation graphique d'un réseau d'ondelettes.

Ainsi, en se référant à la classification faite dans le paragraphe IV du chapitre I, les réseaux d'ondelettes sont des réseaux de fonctions non linéaires paramétrées, où le vecteur Θ_i est constitué par les translations et les dilatations de l'ondelette multidimensionnelle.

Plusieurs choix d'ondelettes sont possibles. En effet, plusieurs familles d'ondelettes existent. Les ondelettes les plus connues (et aussi les plus anciennes) sont certainement celles qui constituent le système de Haar que l'on présentera dans le chapitre suivant, dans le contexte d'ondelettes orthogonales. Les fonctions du système de Haar n'étant pas dérivables, il n'est pas possible d'appliquer aux réseaux de telles ondelettes les algorithmes d'estimation des paramètres présentés dans le chapitre I.

Les ondelettes que nous allons utiliser pour la construction de réseaux sont celles issues des travaux de I. Daubechies. On parle dans ce cas d'ondelettes de la famille de Daubechies. Ces fonctions sont dérivables et possèdent la propriété d'approximation universelle en vertu de la relation (6).

Une ondelette-mère que nous utilisons dans ce mémoire et que l'on retrouve dans [Zhang92] est la dérivée première de la fonction gaussienne. C'est l'une des ondelettes les plus utilisées [Torré95]. Elle est définie par :

$$\phi(x) = \pm x\, e^{-\frac{1}{2}x^2} \quad \text{et} \quad \phi_j(x) = \pm \frac{1}{\sqrt{d_j}} \left(\frac{x \pm m_j}{d_j} \right) \exp\left(\pm \frac{1}{2} \left(\frac{x \pm m_j}{d_j} \right)^2 \right) \tag{13}$$

Le graphe de cette fonction est représenté sur la figure suivante :

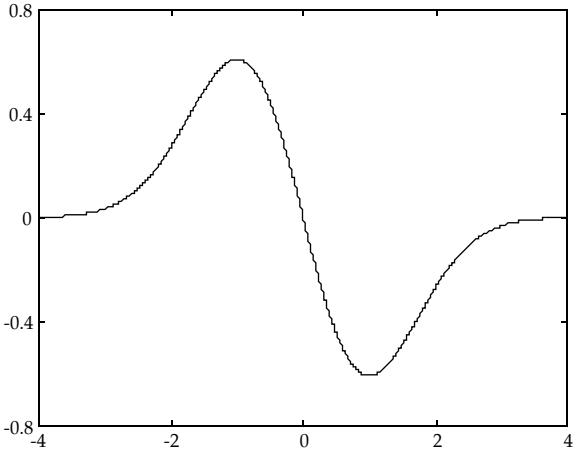


Figure 4. Graphe d'une ondelette.

Cette ondelette peut être considérée comme une forme dérivable des ondelettes du système de Haar, comme la tangente hyperbolique utilisée comme fonction d'activation des réseaux de neurones (présentés au chapitre II de ce mémoire) est une forme dérivable de la fonction signe.

Une autre ondelette-mère que l'on rencontre souvent dans la bibliographie (par exemple dans [Cannon95, Baron97]) est la dérivée seconde de la fonction gaussienne. Son expression est :

$$\phi(x) = \left(x^2 \pm 1\right) e^{-\frac{1}{2}x^2} \tag{14}$$

Elle est appelée "ondelette chapeau mexicain". Son graphe est le suivant :

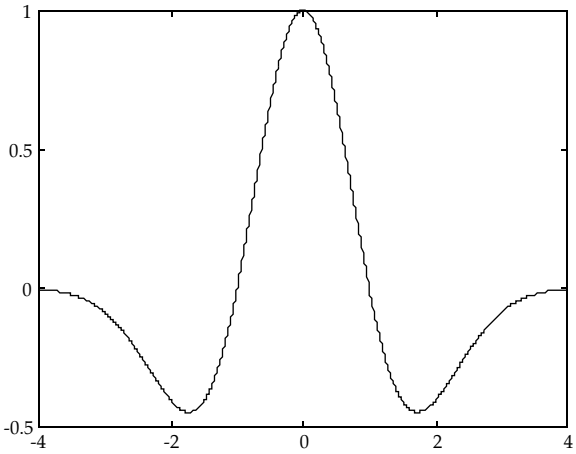


Figure 5. Graphe de l'ondelette "chapeau mexicain".

III.3 Réseaux d'ondelettes et réseaux de neurones.

La principale ressemblance entre les réseaux de neurones à fonctions dorsales, étudiés au chapitre II du présent mémoire, et les réseaux d'ondelettes, réside dans le fait que les deux réseaux calculent une combinaison linéaire, à paramètres ajustables, de fonctions non linéaires dont la forme dépend de paramètres ajustables (translations et dilatations).

Les différences essentielles entre ces deux types de réseaux sont les suivantes :

- contrairement aux fonctions dorsales, les ondelettes sont des fonctions qui décroissent rapidement , et tendent vers zéro dans toutes les directions de l'espace. Elles sont donc locales si d_j est petit ;
- contrairement aux fonctions dorsales, la forme de chaque ondelette monodimensionnelle est déterminée par deux paramètres ajustables (translation et dilatation) qui sont des paramètres structurels de l'ondelette ;
- chaque ondelette monodimensionnelle possède deux paramètres structurels, donc, pour chaque ondelette multidimensionnelle, le nombre de paramètres ajustables est le double du nombre de variables.

Pour comparer la complexité des réseaux, deux éléments sont importants : le nombre de paramètres ajustables et le nombre d'opérations élémentaires à effectuer (tableau 1).

Nous utiliserons les notations suivantes :

Nombre d'entrées : N_i (entrée constante non comprise).
Nombre de fonctions : N_w (pour les ondelettes), N_c (pour les fonctions dorsales).
Nombre de sorties : Une.

	Réseaux de fonctions dorsales	Réseaux de fonctions ondelettes
Nombre de fonctions.	N_c	N_w
Nombre de paramètres.	$\left(N_i + 2\right)\left(N_c + 1\right) \pm 1$	$2\,N_i\,N_w + N_w + N_i + 1$
Nombre d'opérations pour le calcul de la sortie.	$N_c\left(2\,N_i + 3\right) + N_i + 1$	$3\,N_w\left(N_i + 2\right) + N_i + 1$

Tableau 1. Une comparaison entre réseaux d'ondelettes et de fonctions dorsales.

On entend par "opération" les opérations mathématiques élémentaires, c'est-à-dire une addition, une multiplication ou une division. Étant données les propriétés de la fonction exponentielle, et pour chacun des deux types de réseaux, il y a autant de fonctions exponentielle à calculer que de neurones cachés ou d'ondelettes multidimensionnelles dans le réseau.
Lequel des deux types de réseaux est plus économique en termes de nombre d'opérations nécessaires pour le calcul de la sortie ? La réponse peut être obtenue en faisant la différence entre les deux résultats de la dernière ligne du tableau ci-dessus. En effet, à nombre de fonctions égales (c'est-à-dire $N_w = N_c$), la différence entre les nombre d'opérations pour les deux types de réseaux est égale à N_w (N_i + 3).
Le nombre d'opérations effectuées lors du calcul de la sortie avec un réseau d'ondelettes est donc toujours supérieur à celui effectué par un réseau de fonctions dorsales ayant le même nombre d'entrées et de fonctions.

IV. APPRENTISSAGE DES RÉSEAUX D'ONDELETTES NON BOUCLÉS.

IV.1 Calcul du gradient de la fonction de coût.

Les coefficients du réseau peuvent être divisés en deux classes :

– les paramètres structurels des fonctions, c'est-à-dire les translations et les dilatations ;

– les coefficients de pondérations c_j et les coefficients a_k de la partie affine.

Deux possibilités s'offrent à nous pour la construction du réseau :

– choisir les paramètres structurels dans un ensemble de valeurs discrètes ;

– considérer ces paramètres comme ceux d'un réseau de neurones classique et utiliser une technique d'optimisation pour en faire une estimation.

Discrétiser le domaine des translations et des dilatations signifie qu'on effectue la construction de réseaux d'ondelettes suivant une approche fondée sur la transformée en ondelettes discrète. Cette question sera étudiée en détail dans le chapitre suivant.

Dans ce qui suit, nous allons adopter la seconde possibilité, et faire appel aux techniques d'optimisation non linéaire décrite dans le chapitre I de ce document.

Rappelons que l'apprentissage consiste en la minimisation de la fonction de coût suivante :

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

avec

$$y^n = \psi\left(x^n, \theta\right) = \sum_{j=1}^{N_w} c_j \Phi_j\left(x^n, m_j, d_j\right) + \sum_{k=0}^{N_i} a_k x_k^n \quad \text{avec} \quad x_0^n = 1 \tag{16}$$

où y_p^n est la sortie désirée correspondant à l'exemple n, y^n est la sortie du réseau d'ondelettes pour l'exemple n, et

$$x^n = \left\{x_1^n, \ldots, x_{N_i}^n\right\} \tag{17}$$

est le vecteur des entrées.

θ est le vecteur regroupant l'ensemble des paramètres ajustables :

$$\theta = \left\{m_{jk}, d_{jk}, c_j, a_k, a_0\right\} \quad j = 1, \ldots, N_w \quad \text{et} \quad k = 1, \ldots, N_i \tag{18}$$

Les techniques d'optimisation utilisées nécessitent le calcul du vecteur gradient de la fonction de coût par rapport au vecteur des paramètres ajustables. Son expression est :

$$\frac{\partial J}{\partial \theta} = \pm \sum_{n=1}^{N} e^n \frac{\partial y^n}{\partial \theta} \tag{19}$$

où $\frac{\partial y^n}{\partial \theta}$ est la valeur du gradient de la sortie du réseau par rapport aux paramètres θ au point $x=x^n$:

$$\frac{\partial y^n}{\partial \theta} = \left.\frac{\partial y}{\partial \theta}\right|_{x = x^n} \tag{20}$$

Calculons à présent la dérivée de la sortie par rapport à chacun des paramètres du réseau.

• Pour les coefficients directs $\{a_k\}$:

$$\frac{\partial y^n}{\partial a_k} = x_k^n \quad k = 1\ ,\ \ldots\ ,\ N_i \tag{21}$$

• Pour les pondérations des ondelettes $\{c_j\}$:

$$\frac{\partial y^n}{\partial c_j} = \Phi_j(x^n) \quad k = 1\ ,\ \ldots\ ,\ N_i \ \text{ et } \ j = 1\ ,\ \ldots\ ,\ N_w \tag{22}$$

• Pour les translations $\{m_{jk}\}$:

$$\frac{\partial y^n}{\partial m_{jk}} = \pm \frac{c_j}{d_{jk}} \left.\frac{\partial \Phi_j}{\partial z_{jk}}\right|_{x = x^n} \quad k = 1\ ,\ \ldots\ ,\ N_i \ \text{ et } \ j = 1\ ,\ \ldots\ ,\ N_w \tag{23}$$

• Pour les dilatations $\{d_{jk}\}$:

$$\frac{\partial y^n}{\partial d_{jk}} = \pm \frac{c_j}{d_{jk}} z_{jk}^n \left.\frac{\partial \Phi_j}{\partial z_{jk}}\right|_{x = x^n} \quad k = 1\ ,\ \ldots\ ,\ N_i \ \text{ et } \ j = 1\ ,\ \ldots\ ,\ N_w \tag{24}$$

$\left.\frac{\partial \Phi_j}{\partial z_{jk}}\right|_{x = x^n}$ est la valeur de la dérivée partielle de l'ondelette multidimensionnelle par rapport à la variable z_{jk} au point $x=x^n$. Étant donné la relation (11), cette dérivée partielle vaut :

$$\left.\frac{\partial \Phi_j}{\partial z_{jk}}\right|_{x = x^n} = \phi\left(z_{j1}^n\right)\phi\left(z_{j2}^n\right) \ldots \phi'\left(z_{jk}^n\right) \ldots \phi\left(z_{jN_i}^n\right) \tag{25}$$

avec $\phi'\left(z_{jk}^n\right)$ la dérivée au point $x=x^n$ de l'ondelette scalaire, c'est-à-dire :

$$\phi'_{z_{jk}^n} = \left.\frac{d\phi(z)}{dz}\right|_{z = z_{jk}^n} \tag{26}$$

IV.2 Initialisation des paramètres du réseau.

Une fonction ondelette monodimensionnelle est définie sur tout l'ensemble $\boldsymbol{R}$, mais l'essentiel de sa contribution s'étend sur un intervalle centré autour de la valeur de la translation et dont la longueur dépend du paramètre de dilatation.

Dans le cas de réseaux de neurones à fonctions dorsales, l'initialisation des paramètres du réseau est généralement effectuée de manière aléatoire, de telle manière que le potentiel de chaque neurone caché soit suffisamment petit pour que les sorties des neurones se trouvent dans la partie linéaire de la sigmoïde. Les ondelettes étant des fonctions à décroissance rapide, une initialisation aléatoire des paramètres de translation et de dilatation serait très inefficace : en effet, si les translations sont initialisées à l'extérieur du domaine contenant les exemples, ou si les dilatations choisies sont trop petites, la sortie de l'ondelette

est pratiquement nulle, de même que sa dérivée. L'algorithme d'adaptation des paramètres étant fondé sur une technique de gradient, il est inopérant. Une attention particulière doit donc être portée à cette phase d'initialisation des paramètres.

Nous proposons ici une procédure d'initialisation simple, qui prend en considération le domaine où sont répartis les exemples de l'ensemble d'apprentissage.

Soit $[\alpha_k, \beta_k]$ l'intervalle contenant les $k^{\text{ème}}$ composantes des vecteurs d'entrée des exemples. On initialise les translations m_{jk} ($j = 1, \ldots, N_w$) au centre de l'intervalle $[\alpha_k, \beta_k]$:

$$m_{jk} = \frac{\alpha_k + \beta_k}{2} \quad \text{avec} \quad j = 1, \ldots, N_u \tag{27}$$

Les paramètres de dilatation sont choisis de telle manière que les variations de l'ondelette s'étendent sur tout l'intervalle $[\alpha_k, \beta_k]$. Cette condition est remplie avec le choix suivant :

$$d_{jk} = 0{,}2(\alpha_k \pm \beta_k) \quad \text{avec} \quad j = 1, \ldots, N_w \tag{28}$$

Cette procédure est valable notamment pour l'ondelette mère illustrée par la figure 4 que nous allons utiliser dans nos exemples.

Reste la question de l'initialisation des coefficients de pondération des ondelettes (c_j avec $j = 1, \ldots, N_w$) et ceux de la partie affine a_k avec $k = 1, \ldots, N_i$. L'initialisation de ces coefficients est moins importante, pour le déroulement de l'apprentissage, que celle des paramètres structurels ; ils sont initialisés de manière aléatoire, uniformément répartis dans l'intervalle$[-10^{-2} ; +10^{-2}]$.

Cette procédure ne nécessite pratiquement pas de calcul ; elle est très simple à mettre en œuvre. Le fait que, pour $j = 1, \ldots, N_w$ toutes les translations soient initialisées à la même valeur (ainsi que les dilatations) peut laisser penser qu'elles vont évoluer de manière identique si l'on effectue plusieurs apprentissages successifs. Une telle situation est évitée par le fait que les pondérations de chacune des ondelettes du réseau sont initialisées différemment.

Néanmoins, cette procédure d'initialisation présente un inconvénient : elle utilise peu les propriétés des ondelettes. En effet, on peut imaginer que l'on puisse mettre au point une technique d'initialisation qui utilise plus l'information apportée par les paramètres structurels, afin que la fonction de coût soit au voisinage d'un minimum avant d'effectuer l'apprentissage proprement dit. Une telle procédure est proposée dans le chapitre suivant ; elle est utilisable pour des réseaux d'ondelettes issus de la transformée en ondelettes discrète.

IV.3 Exemple de modélisation statique.

IV.3.1 Présentation du processus simulé.

Pour mettre en pratique les réseaux d'ondelettes non bouclés que nous venons de présenter, nous nous proposons d'étudier la modélisation statique d'un processus à une entrée.

Le processus est simulé à partir de la fonction définie sur l'intervalle [–10, +10] par :

$$f(x) = \begin{cases} \pm 2{,}186\, x \pm 12{,}864 \quad \text{si} \quad x \in [\pm 10, \pm 2[\\ 4{,}246\, x \quad \text{si} \quad x \in [\pm 2, 0[\\ 10 \exp\left(\pm 0{,}05\, x \pm 0{,}5\right) \sin\left(x\left(0{,}03\, x + 0{,}7\right)\right) \quad \text{si} \quad x \in [0, 10] \end{cases} \tag{29}$$

Le graphe de cette fonction est représenté sur la figure 6 :

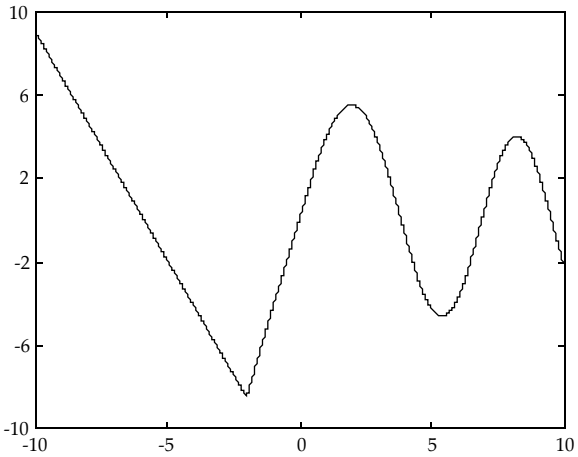


Figure 6. Sortie du processus pour l'intervalle de l'entrée considéré.

IV.3.2 Modélisation avec 100 exemples.

La séquence d'apprentissage est constituée de 100 exemples choisis de manière aléatoire, uniformément répartis, dans l'intervalle considéré. La séquence d'estimation de la performance du modèle est formée de 1000 exemples régulièrement répartis. On utilise les deux algorithmes de BFGS et de Levenberg–Marquardt (présentés dans le chapitre I de ce document). Dans le cas de l'utilisation de la procédure BFGS, une phase de gradient simple avec pas asservi est préalablement appliquée. Pour chaque réseau, on effectue cent apprentissages en modifiant à chaque fois le germe de l'initialisation aléatoire des pondérations $\{c_j\}$ des ondelettes et des coefficients $\{a_k\}$ de la partie affine du réseau. Rappelons que l'initialisation des translations et des dilatations est déterministe (suivant la procédure exposée au paragraphe précédent) : elle est donc identique pour tous les apprentissages. Nous avons testé quatre architectures, à 4, 6, 8 et 10 ondelettes.

Le tableau 2 présente, pour chacune de ces quatre architectures :

- le meilleur EQMP obtenu à l'issue de cent apprentissages avec l'algorithme de BFGS,
- l'EQMA correspondant.

Nombre d'ondelettes.	EQMA	EQMP
4	$7{,}9\ 10^{-3}$	$8{,}3\ 10^{-3}$
6	$1{,}3\ 10^{-3}$	$1{,}4\ 10^{-3}$
8	$5{,}7\ 10^{-4}$	$9{,}1\ 10^{-4}$
10	$1{,}0\ 10^{-4}$	$2{,}4\ 10^{-4}$

Tableau 2. Résultats obtenus avec l'algorithme de BFGS.

La Figure 7 présente les histogrammes des EQMA et des EQMP, pour les 100 apprentissages d'un réseau de 10 ondelettes effectués avec l'algorithme de BFGS. On observe une dispersion des EQMA et des EQMP, due à l'existence de minima locaux de la fonction de coût. Nous montrerons dans le paragraphe suivant que ce problème est très atténué si l'on utilise un plus grand nombre d'exemples pour l'apprentissage.

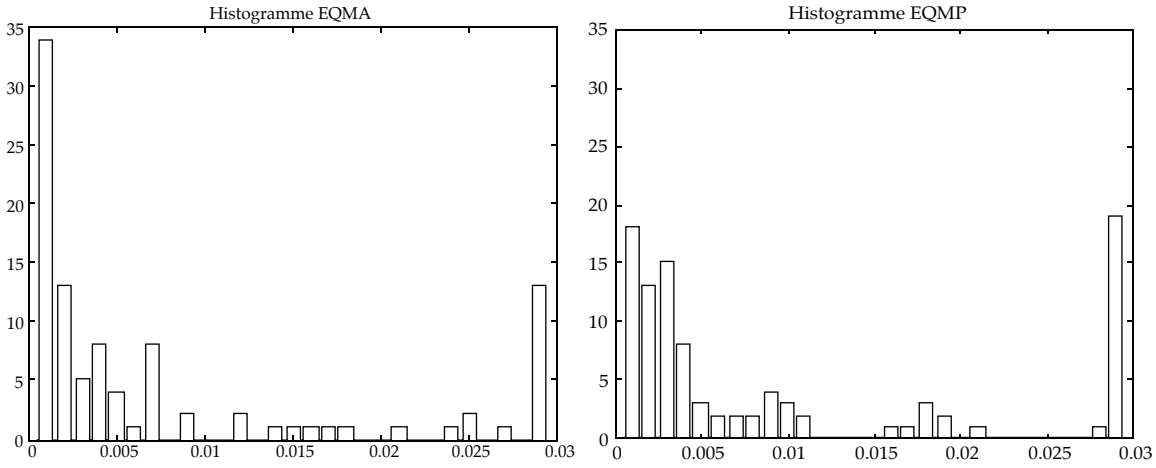


Figure 7. Histogrammes des EQMA et EQMP pour 100 apprentissages.

Les résultats obtenus dans les mêmes conditions en utilisant l'algorithme de Levenberg–Marquardt sont portés sur le tableau 3.

Nombre d'ondelettes.	EQMA	EQMP
4	$8{,}1\ 10^{-3}$	$7{,}8\ 10^{-3}$
6	$2{,}0\ 10^{-3}$	$2{,}3\ 10^{-3}$
8	$1{,}4\ 10^{-4}$	$3{,}2\ 10^{-4}$
10	$3{,}9\ 10^{-5}$	$1{,}9\ 10^{-4}$

Tableau 3. Résultats obtenus avec l'algorithme de Levenberg–Marquardt.

Les meilleurs résultats fournis par les deux algorithmes sont équivalents.

IV.3.3 Modélisation avec 300 exemples.

Nous utilisons cette fois un ensemble d'apprentissage comprenant 300 exemples uniformément répartis dans l'intervalle [–10, +10] et nous effectuons de nouveau 100 apprentissages comme précédemment.

Le tableau 4 présente, pour chacune des quatre architectures considérées :

- le meilleur EQMP obtenu à l'issue de cent apprentissages avec l'algorithme de BFGS,
- l'EQMA correspondant.

Nombre d'ondelettes.	EQMA	EQMP
4	$6{,}8\ 10^{-3}$	$6{,}6\ 10^{-3}$
6	$9{,}3\ 10^{-4}$	$1{,}2\ 10^{-3}$
8	$5{,}1\ 10^{-4}$	$6{,}4\ 10^{-4}$
10	$7{,}5\ 10^{-5}$	$1{,}1\ 10^{-4}$

Tableau 4. Résultats obtenus avec l'algorithme de BFGS.

La Figure 8 présente les histogrammes des EQMA et des EQMP, pour les 100 apprentissages d'un réseau de 10 ondelettes effectués avec l'algorithme de BFGS.

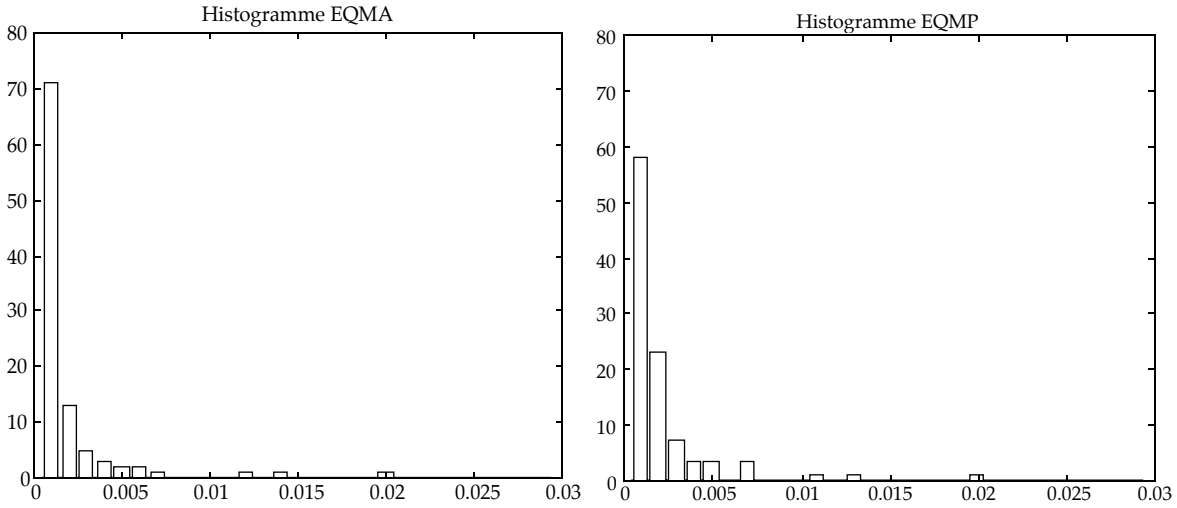


Figure 8. Histogrammes des EQMA et EQMP pour 100 apprentissages.

On constate que les résultats sont beaucoup moins dispersés que ceux qui sont présentés sur la Figure 7. Le fait que la distribution des minima locaux est d'autant plus large que le nombre d'exemples est petit n'est pas spécifique des ondelettes ; il a fait l'objet d'une étude dans [Stoppi97].

Les résultats obtenus dans les mêmes conditions en utilisant l'algorithme de Levenberg–Marquardt sont portés sur le tableau 5.

Nombre d'ondelettes.	EQMA	EQMP
4	$7{,}0\ 10^{-3}$	$6{,}5\ 10^{-3}$
6	$1{,}2\ 10^{-3}$	$1{,}3\ 10^{-3}$
8	$1{,}5\ 10^{-4}$	$2{,}6\ 10^{-4}$
10	$3{,}4\ 10^{-5}$	$5{,}0\ 10^{-5}$

Tableau 5. Résultats obtenus avec l'algorithme de Levenberg–Marquardt.

Là encore, les meilleurs résultats sont analogues à ceux qui ont été obtenus avec l'algorithme de BFGS. Les fréquences d'obtention des meilleurs minima sont voisines.
Cet exemple sera repris dans le chapitre IV, où nous illustrerons la mise en œuvre d'une procédure de sélection pour l'initialisation des translations et dilatations des ondelettes.

IV.3.4 Influence des termes directs

Les résultats présentés dans les deux paragraphes précédents étaient relatifs à des réseaux décrits par la relation (12), dans laquelle apparaissent des "termes directs" (coefficients $\{a_k, k\neq 0\}$) qui réalisent une fonction linéaire des entrées du réseau. Pour évaluer l'influence de ces termes, nous présentons ici les résultats obtenus par apprentissage de réseaux sans termes directs ($a_k = 0$, $k = 1, \ldots N_i$). Nous considérerons uniquement l'apprentissage avec 300 exemples.
Le tableau 6 présente les résultats obtenus après apprentissage par l'algorithme de BFGS, et le tableau 7 ceux obtenus par l'algorithme de Levenberg-Marquardt.

Nombre d'ondelettes.	EQMA	EQMP
4	$4{,}0\ 10^{-2}$	$3{,}6\ 10^{-2}$
6	$5{,}3\ 10^{-3}$	$5{,}4\ 10^{-3}$
8	$8{,}5\ 10^{-4}$	$1{,}2\ 10^{-3}$
10	$3{,}9\ 10^{-4}$	$4{,}7\ 10^{-4}$

Tableau 6. Résultats obtenus avec l'algorithme de BFGS.

Nombre d'ondelettes.	EQMA	EQMP
4	$2{,}6\ 10^{-2}$	$2{,}4\ 10^{-2}$
6	$1{,}4\ 10^{-3}$	$1{,}9\ 10^{-3}$
8	$3{,}7\ 10^{-4}$	$4{,}8\ 10^{-4}$
10	$3{,}3\ 10^{-4}$	$4{,}0\ 10^{-4}$

Tableau 7. Résultats obtenus avec l'algorithme de Levenberg–Marquardt.

On observe que les EQM sont systématiquement supérieures à celles que l'on obtient avec des réseaux comportant des termes directs.

IV.3.5 Quelques figures.

La figure suivante illustre la disposition des ondelettes en fin d'apprentissage pour le réseau de 10 ondelettes optimisé avec l'algorithme de Levenberg–Marquardt.

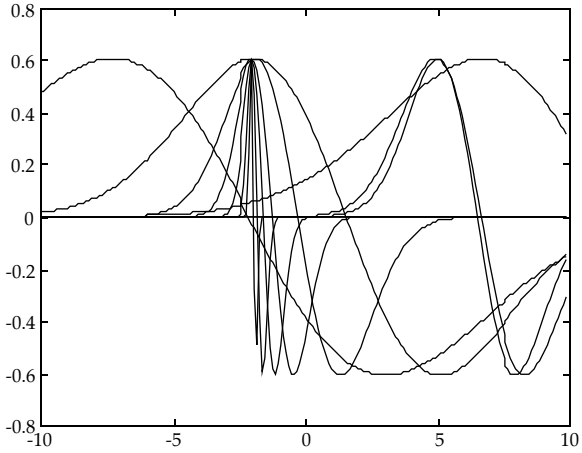
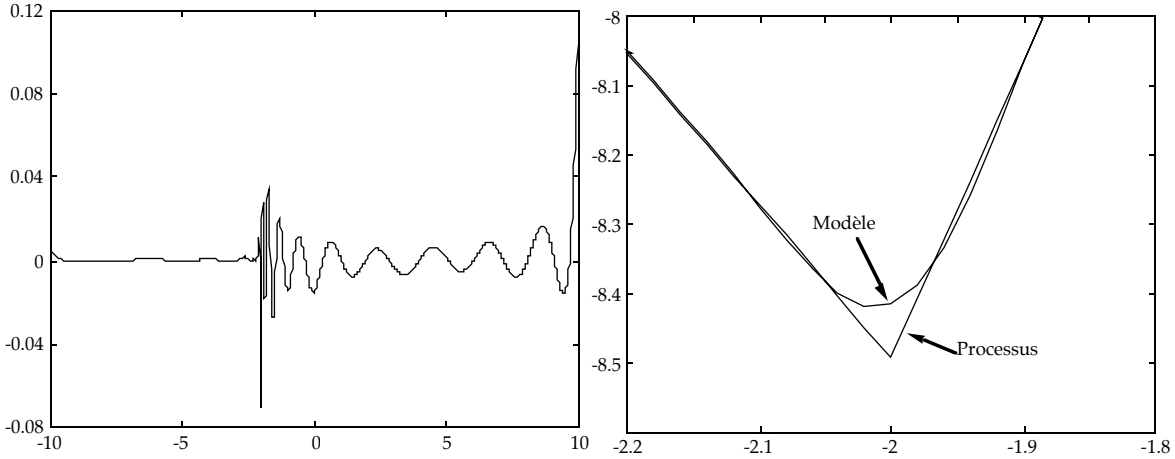


Figure 9. Ondelettes en fin d'apprentissage.

On peut observer que tous les centres ne sont pas à l'intérieur du domaine où est définie la fonction *f*, mais l'intersection du support de chacune des ondelettes avec le domaine est non nulle. Afin d'utiliser une seule échelle, les ondelettes sont représentées avec leurs sorties non pondérées par les coefficients c_j.



(a) (b)

Figure 10. Erreur de modélisation (a) et détail de la sortie du modèle et du processus autour du point anguleux (b).

La figure 10 illustre l'erreur de modélisation (a) commise par le réseau de 10 ondelettes. L'erreur est principalement commise au niveau du point anguleux (b) qui est certainement la seule difficulté pour l'approximation de cette fonction.

V. MODÉLISATION DYNAMIQUE ENTRÉE–SORTIE ET RÉSEAUX D'ONDELETTES.

Comme nous venons de le voir, la construction de réseaux d'ondelettes non bouclés pour la modélisation statique de processus tire son origine de la transformée en ondelettes inverse. On se propose d'étendre l'utilisation des réseaux d'ondelettes à la modélisation dynamique de processus.

Considérons un modèle-hypothèse de la forme :

$$y_p(n) = f\left(y_p(n\text{-}1)\ ,\ \dots\ ,\ y_p(n\text{-}N_s),\ u(n\text{-}1)\ ,\ \dots\ ,\ u(n\text{-}N_e),\ w(n)\ ,\ \dots\ ,\ w(n\text{-}N_n)\right) \qquad (30)$$

où u est une entrée externe appliquée au processus et y_p sa sortie. N_s est l'ordre du modèle. $\{w(n)\}$ est une séquence de variables aléatoires de moyenne nulle et de variance σ^2. f est une fonction paramétrée inconnue dont il s'agit d'estimer les paramètres à l'aide d'une séquence d'apprentissage. Chaque exemple correspond à un instant de mesure. Pour une séquence d'apprentissage de N exemples, nous avons $n = 1, \dots, N$.

Des hypothèses supplémentaires sont généralement faites sur la façon dont le bruit agit. Un choix adéquat du prédicteur associé peut alors être effectué. Différents exemples de modèles-hypothèses ainsi que les prédicteurs optimaux qui leur sont associés sont présentés dans le paragraphe III.3 du chapitre I.

Rappelons que :

- si l'hypothèse de l'existence d'un bruit additif de sortie a été retenue, ou si, en l'absence de bruit, on désire obtenir un modèle de simulation du processus, les entrées d'état du prédicteur, durant son apprentissage, sont les sorties passées du prédicteur ; si le prédicteur est réalisé par un réseau, celui-ci est bouclé pendant l'apprentissage ;
- si l'hypothèse de l'existence d'un bruit d'état additif a été retenue, ou si, en l'absence de bruit, on envisage d'utiliser le prédicteur pour prédire la sortie une seule période d'échantillonnage plus tard, les entrées d'état du prédicteur, durant son apprentissage, sont les sorties du processus ; si le prédicteur est une réseau de fonctions, celui-ci est non bouclé pendant l'apprentissage.

A notre connaissance, les réseaux d'ondelettes bouclés n'ont jamais été étudiés auparavant. On trouve dans la conclusion de la référence [Zhang92] (qui traite des réseaux d'ondelettes fondés sur la transformée en ondelette continue)

un commentaire à ce propos. Les auteurs soulignent qu'une investigation des performances de tels réseaux est une voie à explorer.

V.1 Apprentissage de réseaux de type entrée-sortie.

Le schéma d'apprentissage que nous adoptons est semblable à celui qui est utilisé dans le cas de réseaux de neurones à fonctions sigmoïdes. Nous utilisons la notion de copie du réseau. On désigne par "copie numéro n" la partie statique du réseau canonique qui calcule $y(n)$.

V.1.1 Apprentissage de prédicteurs non bouclés.

Pour paramétrer un réseau constituant un prédicteur non bouclé pendant l'apprentissage, on se ramène à la notation utilisée pour les réseaux statiques.

Soit $x^n \in R^{N_i}$ le vecteur d'entrée de la copie n. Ses différentes composantes sont les suivantes :

Pour $k = 1$, ... , N_e : $x_k^n = u(n \pm k)$ sont les entrées externes. N_e est le nombre de ces entrées.

Pour $k = N_e$+1, ... , N_e+ N_s : $x_k^n = y_p(n \pm k + N_e)$ sont les entrées d'état, qui sont les sorties mesurées sur le processus (entrées d'état).

Comme nous l'avons rappelé dans le paragraphe précédent, les valeurs des entrées d'état pour chaque copie sont forcées aux sorties correspondantes du processus. Le prédicteur est dirigé par le processus, d'où le nom d'apprentissage "dirigé" [Nerrand92, Nerrand93] ou *"teacher–forced"* [Jordan85].

Nous avons ainsi $N_e + N_s = N_i$, le nombre d'entrées d'un réseau d'ondelettes pour la modélisation statique.

L'apprentissage par la méthode du gradient s'effectue de la même manière que dans le cas de la modélisation statique.

V.1.2 Apprentissage de prédicteurs bouclés.

Pour un prédicteur bouclé pour l'apprentissage, le calcul du gradient de la fonction de coût doit tenir compte du fait que le réseau est bouclé.

Pour $k = 1, \ldots, N_e : x_k^n = u(n \pm k)$ sont les entrées externes.

Pour $k = N_e$+1, ... , N_e+ N_s : $x_k^n = y(n \pm k + N_e)$ sont les N_s valeurs passées des sorties de la copie n–1.

Pour $k = N_e$+N_s+1 , ... , N_e+2N_s : $x_k^n = y(n \pm k + N_e + N_s + 1)$ sont les sorties de la copie n.

La figure 11 illustre la configuration du réseau pour l'exemple de l'instant n (c'est-à-dire la copie numéro n).

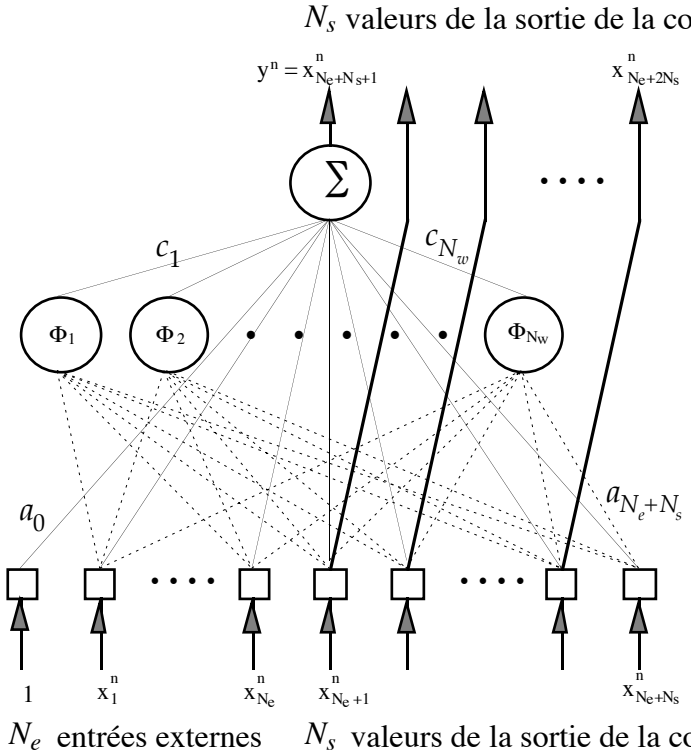


Figure 11. La copie numéro n du réseau prédicteur entrée-sortie bouclé.

Ici, seules les valeurs des entrées d'état de la première copie sont prises égales aux valeurs correspondantes de la sortie du processus. Pour les copies suivantes, ces entrées prennent les valeurs des variables d'état en sortie de la copie précédente. Le prédicteur est "*semi-dirigé*" par le processus. Pour cette raison, l'algorithme est dit *"semi-dirigé"* [Nerrand92] (*"backpropagation through time"* [Rumelhart86]).

Rappelons que la fonction de coût à minimiser au cours de l'apprentissage est la même que dans le cas de la modélisation statique, c'est-à-dire $J(\theta) = \frac{1}{2}\sum_{n=1}^{N}\left(y_p^n \pm y^n(\theta)\right)^2$, où θ est le vecteur des paramètres ajustables.

On désigne par θ^n le vecteur des paramètres de la copie n du réseau :

$$\theta^n = \left\{ m_{jk}^n,\ d_{jk}^n,\ c_j^n,\ a_k^n,\ a_0^n \right\} \quad avec \quad j = 1, \ldots, N_w \text{ et } k = 1, \ldots, N_e+N_s \tag{31}$$

Il est nécessaire de distinguer les paramètres θ_i^n et $\theta_i^{n'}$ de deux copies différentes n et n' bien qu'ils aient les mêmes valeurs : en effet, les composantes du gradient $\frac{\partial J}{\partial \theta_i^n}$ et $\frac{\partial J}{\partial \theta_i^{n'}}$ sont différentes. Rappelons que $\frac{\partial J}{\partial \theta} = \sum_{n=1}^{N} \frac{\partial J}{\partial \theta^n}$.

Ces notations étant définies, nous abordons le calcul du gradient $\frac{\partial J}{\partial \theta}$ pour les réseaux d'ondelettes bouclés. Deux approches sont possibles :

- calcul par rétropropagation,
- calcul dans le sens direct.

V.1.3 Calcul du gradient par rétropropagation.

Le vecteur gradient est décomposé de la manière suivante :

$$\frac{\partial J}{\partial \theta} = \sum_{n=1}^{N} \frac{\partial J}{\partial \theta^n} = \sum_{n=1}^{N} \frac{\partial J}{\partial y^n} \frac{\partial y^n}{\partial \theta^n} \tag{32}$$

La quantité $\frac{\partial y^n}{\partial \theta^n}$ est la dérivée de la sortie de la copie n par rapport aux coefficients de la même copie. Les expressions de cette dérivée pour chacune des composantes du vecteur θ sont données par les relations (20) à (24).

Reste donc à calculer $\frac{\partial J}{\partial y^n}$ pour $n = 1, \ldots, N$.

Afin de présenter les expressions de façon plus claire, on introduit des variables intermédiaires que l'on note par q_k^n et que l'on définit par :

$$q_k^n = \pm \frac{\partial J}{\partial x_k^n} \quad \text{avec} \quad k = N_e+1 , \ \ldots \ , N_e+2N_s \tag{33}$$

Ce sont les dérivées partielles de la fonction $-J$ par rapport aux variables d'état en entrée et en sortie de la copie numéro n.

Pour la présentation du calcul du gradient par rétropropagation, on considère séparément la dernière copie, de numéro N, celles dont le numéro est compris entre $N-1$ et 2, et enfin la première copie.

<u>Pour la copie N</u>, nous avons :

Pour la sortie :

$$\pm \frac{\partial J}{\partial y^N} = q_{sortie}^N = q_{N_e+N_s+1}^N = e^N \tag{34}$$

Pour les autres variables d'état (en sortie du réseau) :

$$q_k^N = 0 \quad \text{avec} \quad k = N_e+N_s+2 , \ \ldots \ , N_e+2N_s \tag{35}$$

Pour les variables d'état (en entrée du réseau) :

$$\pm \frac{\partial J}{\partial x_k^N} = q_k^N = \left(a_k^N + \sum_{j=1}^{N_w} \frac{c_j^N}{d_{jk}^N} \frac{\partial \Phi_j^N}{\partial z_{jk}^N} \right) q_{sortie}^N \quad \text{avec} \quad k = N_e+1 , \ \ldots \ , N_e+N_s \tag{36}$$

<u>Pour les copies de $n=N-1$ à 2</u>, nous avons :

Pour la sortie :

$$\pm \frac{\partial J}{\partial y^n} = q^n_{sortie} = e^n + q^{n+1}_{N_e+1} \tag{37}$$

Pour les autres variables d'état (en sortie du réseau) :

$$q^n_k = q^{n+1}_{k \pm N_s} \quad \text{avec} \quad k = N_e+N_s+2 \, , \, \ldots \, , \, N_e+2N_s \tag{38}$$

Pour les variables d'état en entrée du réseau :

$$\pm \frac{\partial J}{\partial x^n_k} = q^n_k = q^n_{k+N_s+1} + \left(a^n_k + \sum_{j=1}^{N_w} \frac{c^n_j}{d^n_{jk}} \frac{\partial \Phi^n_j}{\partial z^n_{jk}} \right) q^n_{sortie} \quad \text{avec} \quad k = N_e+1 \, , \, \ldots \, , \, N_e+N_s \tag{39}$$

Pour la copie n=1, nous avons :

Pour la sortie :

$$\pm \frac{\partial J}{\partial y^1} = q^1_{sortie} = e^1 + q^2_{N_e+1} \tag{40}$$

V.1.4 Calcul du gradient dans le sens direct.

L'utilisation de l'algorithme de Levenberg–Marquardt pour l'apprentissage de réseaux bouclés nécessite la mise en oeuvre du calcul du gradient dans le sens direct. En effet, cet algorithme demande le calcul du Hessien de la fonction de coût J (ou d'une approximation de celui-ci). Rappelons que cette approximation s'exprime de la façon suivante :

$$\tilde{H} = \sum_{n=1}^{N} \frac{\partial e^n}{\partial \theta} \left(\frac{\partial e^n}{\partial \theta} \right)^T = \sum_{n=1}^{N} \frac{\partial y^n}{\partial \theta} \left(\frac{\partial y^n}{\partial \theta} \right)^T \tag{41}$$

Les dérivées partielles sont calculées à partir de :

$$\frac{\partial y^n}{\partial \theta} = \sum_{m=1}^{n} \frac{\partial y^n}{\partial \theta^m} \tag{42}$$

Afin d'obtenir les quantités de la relation (42) avec un calcul du gradient par rétropropagation, il est nécessaire d'effectuer N rétropropagations. De ce fait, le calcul dans le sens direct est plus économique.

La relation précédente exprime que le calcul du gradient de la sortie de la copie numéro n du réseau par rapport au vecteur des paramètres θ doit prendre en considération les dérivées de cette sortie par rapport à chacune des copies du vecteur des paramètres d'indices inférieurs ou égaux. Décomposons l'expression (42) :

$$\sum_{m=1}^{n} \frac{\partial y^n}{\partial \theta^m} = \sum_{m=1}^{n\pm 1} \frac{\partial y^n}{\partial \theta^m} + \frac{\partial y^n}{\partial \theta^n} \tag{43}$$

Le deuxième terme est donné par les relations (21) à (24) ; il suffit donc de calculer le premier.

La quantité $\frac{\partial y^n}{\partial \theta^m}$ peut s'écrire de la manière suivante, où y^{n-l} est la variable d'état numéro l en entrée de la $n^{ème}$ copie :

$$\frac{\partial y^n}{\partial \theta^m} = \sum_{l=1}^{N_s} \frac{\partial y^n}{\partial y^{n\pm l}} \frac{\partial y^{n\pm l}}{\partial \theta^m} \quad \text{avec } m = 1, \ldots, n\pm 1 \text{ et } n \pm l \geq m \tag{44}$$

On obtient ainsi :

$$\sum_{m=1}^{n\pm 1} \frac{\partial y^n}{\partial \theta^m} = \sum_{l=1}^{N_s} \frac{\partial y^n}{\partial y^{n\pm l}} \left(\sum_{m=1}^{n\pm 1} \frac{\partial y^{n\pm l}}{\partial \theta^m} \right) \quad \text{avec} \quad n \pm l \geq m \tag{45}$$

Remarquons que le second facteur peut s'écrire :

$$\sum_{m=1}^{n\pm 1} \frac{\partial y^{n\pm l}}{\partial \theta^m} = \sum_{m=1}^{n\pm l} \frac{\partial y^{n\pm l}}{\partial \theta^m} = \frac{\partial y^{n\pm l}}{\partial \theta} \tag{46}$$

En introduisant ce dernier résultat dans la relation (45), on obtient :

$$\sum_{m=1}^{n\pm 1} \frac{\partial y^n}{\partial \theta^m} = \sum_{l=1}^{N_s} \frac{\partial y^n}{\partial y^{n\pm l}} \frac{\partial y^{n\pm l}}{\partial \theta} \tag{47}$$

En reprenant la relation (43), on aboutit à :

$$\frac{\partial y^n}{\partial \theta} = \sum_{l=1}^{N_s} \frac{\partial y^n}{\partial y^{n\pm l}} \frac{\partial y^{n\pm l}}{\partial \theta} + \frac{\partial y^n}{\partial \theta^n} \tag{48}$$

La quantité $\frac{\partial y^n}{\partial y^{n\pm l}}$ est la dérivée de la sortie de la copie n par rapport à la sortie calculée de la copie $n-l$, qui est donc la $l^{\text{ème}}$ variable d'état en entrée de la copie n. Elle est donnée par :

$$\frac{\partial y^n}{\partial y^{n\pm l}} = a^n_{N_e+l} + \sum_{j=1}^{N_w} \frac{c^n_j}{d^n_{j,N_e+l}} \frac{\partial \Phi^n_j}{\partial z^n_{j,N_e+l}} \quad \text{avec} \quad l=1, \ldots, N_s \tag{49}$$

La relation (48) permet de calculer la dérivée de la sortie y^n de la copie n par rapport à θ en fonction de celles calculées aux N_s copies précédentes.

V.2 Exemple.

V.2.1 Présentation du processus.

On se propose d'étudier un exemple de modélisation dynamique à l'aide d'un réseau d'ondelettes bouclé afin de mettre en œuvre les algorithmes et procédures présentés au paragraphe précédent.
Le processus est simulé à partir d'une équation aux différences ayant pour expression :

$$y_p(k) = \frac{y_p(k\pm1)\, y_p(k\pm2)\, y_p(k\pm3)\, u(k\pm2)\left(y_p(k\pm3) \pm 1\right) + u(k\pm1)}{1 + y_p^2(k \pm 2) + y_p^2(k \pm 3)} \tag{50}$$

où $u(.)$ est l'entrée externe et $y_p(.)$ la sortie du processus. Afin de simuler le processus, il est indispensable de choisir une séquence pour l'entrée externe u. La séquence des entrées externes est une séquence pseudo-aléatoire de distribution uniforme entre –1 et 1. Les séquences d'apprentissage et d'estimation de la performance sont constituées chacune de 1000 points. Étant donné que le processus n'est pas bruité, on peut effectuer indifféremment une modélisation avec un réseau bouclé ou non. On choisit la première possibilité.

Dans le domaine des entrées choisi, c'est-à-dire [–1, +1], les sorties sont comprises dans le même intervalle.

V.2.2 Étude du gain statique.

Un régime statique est atteint si pour $u(k-1)=u(k-2)=\alpha$ constante, on a $y_p(k)=y_p(k-1)=y_p(k-2)=y_p(k-2)=y_p(k-3)=\beta$ constante. Le gain statique est le rapport de la sortie à l'entrée :

$$G_{\text{statique}} = \frac{\beta}{\alpha} \tag{51}$$

En utilisant les égalités précédentes et le modèle du processus donné par la relation (50), on obtient pour expression du gain statique en fonction de la sortie :

$$G_{\text{statique}}(\beta) = \frac{\beta^3\left(\beta \pm 1\right) + 1}{1 + 2\,\beta^2} \tag{52}$$

Dans le domaine des entrées que nous avons choisi pour la construction de nos deux séquences, le graphe du gain statique est le suivant :

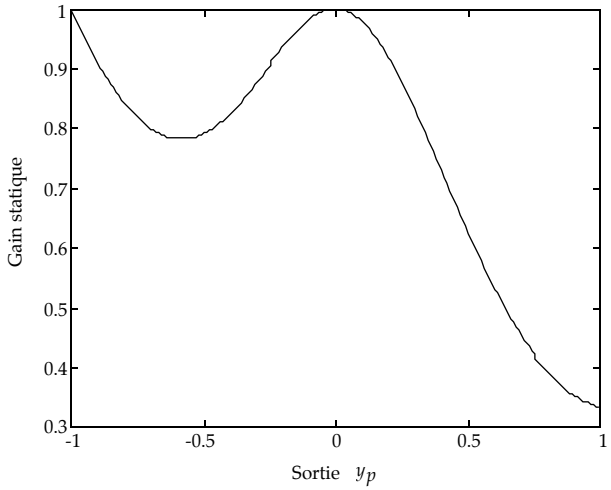


Figure 12. Gain statique dans le domaine de sortie [-1 ; +1]

On constate que pour de faibles amplitudes de la sortie (proches de zéro) le gain statique est proche de l'unité. Précisons que cette étude ne nous donne pas d'information sur la stabilité du modèle. En fait, des essais de simulation montrent que le modèle est instable si des entrées d'amplitudes supérieures à 1 sont appliquées.

V.2.3 Modélisation du processus.

On se propose d'utiliser quatre architectures de réseaux formés respectivement de 5, 10 et 15 ondelettes. Une modélisation linéaire est également effectuée (réseau ne contenant aucune ondelette). Pour chaque architecture, on effectue 50 apprentissages en modifiant à chaque fois le germe de l'initialisation aléatoire des paramètres. Le réseau retenu est celui dont performance estimée est la meilleure. Les résultats obtenus en utilisant l'algorithme de BFGS sont représentés dans le tableau suivant :

Nb. d'ondelettes.	EQMA	EQMP
0	$8{,}1\ 10^{-3}$	$9{,}3\ 10^{-3}$
5	$8{,}7\ 10^{-6}$	$1{,}3\ 10^{-5}$
10	$1{,}5\ 10^{-6}$	$2{,}3\ 10^{-6}$
15	$1{,}3\ 10^{-7}$	$2{,}9\ 10^{-7}$

Tableau 8. Résultats obtenus avec l'algorithme de BFGS.

On effectue également des apprentissages dans les mêmes conditions en utilisant cette fois l'algorithme de Levenberg–Marquardt avec le calcul du gradient dans le sens direct (comme présenté plus haut dans ce chapitre). Les résultats obtenus sont reportés sur le tableau 9.

Nb. d'ondelettes.	EQMA	EQMP
0	$8{,}1\ 10^{-3}$	$9{,}3\ 10^{-3}$
5	$6{,}6\ 10^{-6}$	$9{,}4\ 10^{-6}$
10	$1{,}1\ 10^{-7}$	$3{,}1\ 10^{-7}$
15	$1{,}2\ 10^{-8}$	$4{,}8\ 10^{-8}$

Tableau 9. Résultats obtenus avec l'algorithme de Levenberg–Marquardt.

Les EQM obtenues ici avec les grands réseaux (10 et 15 ondelettes) sont plus faibles que celles obtenues par l'algorithme de BFGS. Sur cet exemple, nous avons donc un meilleur comportement de l'algorithme de Levenberg–Marquardt au prix d'un temps de calcul beaucoup plus important.

VI. MODÉLISATION D'ÉTAT ET RÉSEAUX D'ONDELETTES.

L'état d'un modèle est l'ensemble minimal des N_s valeurs nécessaires à l'instant k pour calculer sa sortie à l'instant k+1, les valeurs des entrées étant données jusqu'à k. N_s est l'ordre du modèle. La représentation d'état est la représentation la plus générale du comportement dynamique d'un processus. La représentation entrée-sortie en est un cas particulier.

Un modèle d'état à temps discret est constitué

- d'un système de N_s équations récurrentes du 1er ordre exprimant l'état à l'instant k+1 en fonction de l'état et des entrées à l'instant k,
- d'une équation d'observation, qui exprime la sortie en fonction de l'état, ou plus généralement, de l'état et des entrées.

Un modèle-hypothèse prend la forme suivante :

$$\begin{cases} x_p(k+1) = f\left(x_p(k), u(k)\right) \\ y_p(k) = g\left(x_p(k)\right) \end{cases} \tag{53}$$

où $x_p(k) \in R^{N_s}$ est le vecteur d'état à l'instant k, $u(k) \in R^{N_e}$ est le vecteur des entrées et $y_p(k)$ la sortie du processus à l'instant k. f et g sont des fonctions à variable vectorielle.

Dans le cadre de cette étude, on s'intéressera aux cas où $u(k)$ est une entrée scalaire et où le modèle ne possède qu'une sortie (modèles mono-entrée mono-sortie ou SISO pour Single Input, Single Output).

Dans le cas d'un modèle entrée-sortie, les variables d'état sont les sorties, donc elles sont nécessairement mesurées, ce qui n'est pas le cas pour un modèle d'état.

Dans la suite de la présente étude de modélisation par réseaux d'état, nous ne considérons que des modèles sans bruit ou des modèles avec un bruit additif de sortie.

VI.1 Modèles d'état sans bruit, avec états non mesurables.

Si les états d'un modèle boîte noire ne sont pas mesurables, seul le comportement entrée–sortie peut être modélisé. Dans ce cas, les états obtenus n'ont pas forcément une signification physique, contrairement au cas où ils sont mesurables. En changeant le nombre de fonctions dans le réseau, ou son initialisation, les modèles obtenus peuvent posséder une performance équivalente du point de vue du comportement entrée–sortie, bien que les séquences des variables d'état soient différentes. Le prédicteur associé est obligatoirement bouclé et son expression est la suivante :

$$\begin{cases} x(k+1) = \psi_1(x(k), u(k)) \\ y(k+1) = \psi_2(x(k), u(k)) \end{cases} \tag{54}$$

Ce prédicteur est illustré par la figure suivante :

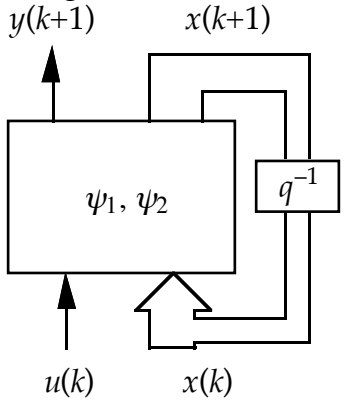


Figure 13. Prédicteur d'état bouclé.

Dans le cas où la sortie *y(k)* n'est pas fonction de *u(k)*, un réseau associé comme celui de la figure 13 est envisageable en supprimant la connexion correspondante.

Dans les exemples que nous étudierons dans la suite de ce mémoire, les états sont généralement non mesurables et les séquences d'apprentissage et d'estimation de la performance dont on dispose sont formées uniquement des mesures de l'entrée externe et de la sortie du processus : seul l'apprentissage d'un prédicteur d'état bouclé est possible. Dans la suite de ce mémoire, on ne considérera donc que des prédicteurs d'état bouclés comme celui illustré par la figure 13.

VI.2 Apprentissage de réseaux d'état bouclés.

VI.2.1 Structure du réseau d'état.

Le réseau non bouclé de la forme canonique d'un réseau d'ondelettes bouclé comprend :

• Une couche d'entrée possédant N_e entrées externes et N_s variables d'état. Le nombre total des entrées est alors $N_i = N_e + N_s$.

• Une couche cachée constituée par N_w ondelettes multidimensionnelles.

• Une couche de sortie comportant un neurone linéaire donnant la sortie du réseau $y(n) = y^n$ et N_s neurones linéaires d'état donnant chacun la valeur de l'état correspondant pour l'instant considéré.

La notion de copie de réseau utilisée dans le cadre de la modélisation entrée–sortie est généralisée. Ici, comme expliqué plus haut, seuls les réseaux d'état bouclés seront utilisés et l'apprentissage met en jeu un grand réseau constitué par N copies en cascade (N est toujours la taille de la séquence d'apprentissage). Le vecteur d'état en entrée de la copie numéro n est le vecteur de sortie de la copie précédente. Pour la première copie :

- si les états sont mesurables le vecteur est identique au vecteur des entrées du processus ;
- si les états ne sont pas mesurables, et en l'absence de toute information sur l'état initial du processus, on force les entrées d'état de la première copie à zéro.

Dans les exemples que nous avons traités, nous nous trouvons dans cette dernière situation.

Suivant la description donnée plus haut sur la structure du réseau, le vecteur θ des paramètres est composé des éléments suivants :

$$\theta = \{m_{jk}, d_{jk}, c_{kj}, c_0\} \tag{55}$$

• Les translations m_{jk} et les dilatations d_{jk} avec k=1 , ... , $N_e + N_s$ et j=1 , ... , N_w.

• Les pondérations et les coefficients directs que l'on note par c_{kj}. Ce choix d'indices signifie qu'il s'agit du coefficient de la liaison entre la fonction (ou le neurone d'entrée) numéro j et le neurone de sortie (ou le neurone d'état) numéro k. Pour les pondérations nous avons :
$j= N_e+N_s+1$, ... , $N_e+N_s+N_w$ et $k=N_e+N_s+N_w+1$, ... , $N_e+2N_s+N_w+1$.
Pour les coefficients directs, nous avons :
j=1, ... , N_e+N_s et $k=N_e+N_s+N_w+1$, ... , $N_e+2N_s+N_w+1$.

• Un terme constant sur le neurone linéaire de la sortie que l'on note par c_0.

• Le nombre de composantes du vecteur θ est alors :

$$2N_w(N_e+N_s)+(N_s+1)(N_e+N_s+N_w)+1.$$

On note par x_k^n la variable d'état numéro k en entrée de la copie numéro n du réseau si $k=N_e+1$, ... , N_e+N_s et en sortie de cette copie si $k=N_e+N_s+N_w+2$, ... ,

N_e+2N_s+N_w+1. La figure 14 montre l'architecture de la copie n. Notons que les noeuds sont numérotés dans l'ordre de 1 à N_e+2N_s+N_w+1 alors que les ondelettes le sont de 1 à N_w.

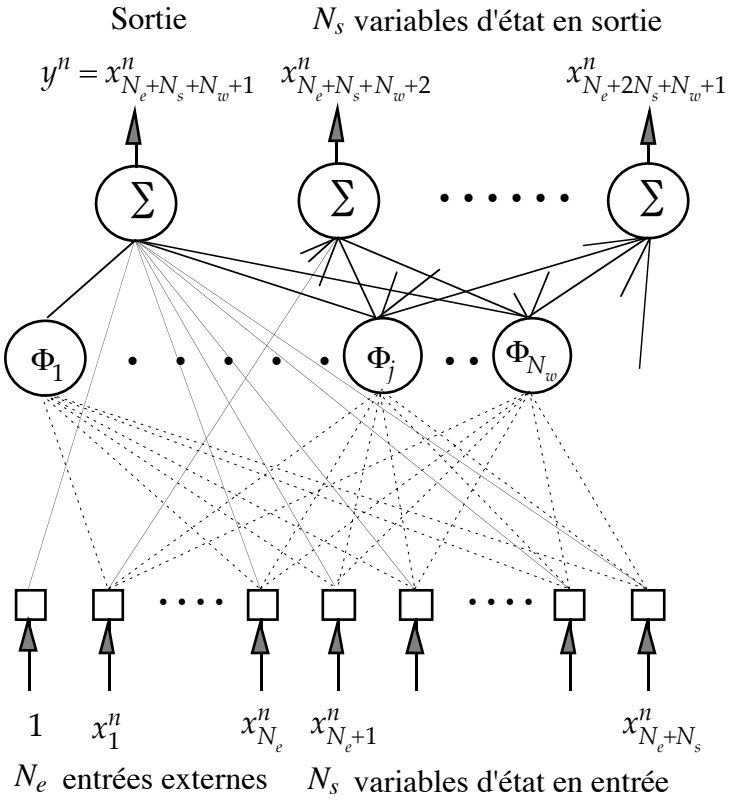


Figure 14. Illustration de la copie numéro n du réseau d'état.

La sortie de la copie n du réseau a pour expression :

$$y^n = \sum_{j=1}^{N_w} c_{\alpha,\, j+N_e+N_s}\, \Phi_j\left(x^n\right) + \sum_{k=0}^{N_e+N_s} c_{\alpha,k}\, x^n_k \quad \text{avec} \quad \alpha = N_e+N_s+N_w+1 \quad \text{et} \quad x^n_0=1 \qquad (56)$$

avec $x^n = \left\{x^n_1, x^n_2, \ldots, x^?_{N_e+N_s}\right\}$.

La variable d'état numéro k en sortie du réseau est calculée à partir de la relation suivante :

$$x^n_k = \sum_{j=N_e+N_s+1}^{N_e+N_s+N_w} c_{k,\,j}\, \Phi_j\left(x^n\right) + \sum_{j=1}^{N_e+N_s} c_{k,\,j}\, x^n_j \quad \text{avec} \quad k = N_e+N_s+N_w+2\,,\ \ldots\,,\ N_e+2N_s+N_w+1 \qquad (57)$$

Notons que nous avons omis les termes directs sur les neurones linéaires d'état. En effet, ces termes ont ici moins d'importance que pour le neurone linéaire de la sortie du réseau.

La variable d'état en entrée est prise égale à celle de la variable d'état correspondante en sortie de la copie précédente. On peut écrire pour la copie n :

$$x^n_k = \begin{cases} x^{n\pm1}_{k\,+N_s+N_w+1} & \text{si } n \geq 2 \\ 0 & \text{si } n = 1 \end{cases} \qquad \text{avec} \quad k = N_e+1\,,\ \ldots\,,\ N_e+N_s \qquad (58)$$

VI.2.2 Calcul du gradient par rétropropagation.

De la même façon que pour les réseaux de type entrée–sortie bouclés et non bouclés, nous devons calculer le gradient de la fonction de coût par rapport au vecteur des paramètres. L'apprentissage est également fondé sur des méthodes de gradient telles que celles présentées au chapitre I de ce mémoire.

Les deux approches déjà mentionnées (calcul du gradient par rétropropagation à travers les copies ou dans le sens direct) sont envisageables. Nous les présenterons toutes les deux. Seule la première sera mise en œuvre. En effet, dans le cas des réseaux d'état, le calcul du gradient dans le sens direct implique un volume de calculs plus important que le calcul par rétropropagation. Ce dernier se divise en deux étapes :

- calcul du gradient de la fonction de coût J par rapport à la sortie et aux variables d'état en entrée et en sortie de chacune des N copies,
- calcul du gradient de J par rapport au vecteur θ des paramètres.

VI.2.2.1 Calcul du gradient de J par rapport à la sortie et aux variables d'état.

La fonction de coût utilisée pour l'apprentissage est toujours : $J(\theta) = \frac{1}{2} \sum_{n=1}^{N} \left(y_p^n \pm y^n(\theta) \right)^2$.

On distingue trois calculs différents suivant qu'il s'agit de la première copie, de la dernière ou des autres.

<u>Pour la copie N,</u> nous avons :

Pour la sortie :

$$\frac{\partial J}{\partial y^N} = \pm e^N \tag{59}$$

Pour les variables d'état en sortie, $k=N_e+N_s+N_w+2$, ... , $N_e+2N_s+N_w+1$:

$$\frac{\partial J}{\partial x_k^N} = 0 \tag{60}$$

Pour les variables d'état en entrée, $k=N_e+1$, ... , N_e+N_s :

$$\frac{\partial J}{\partial x_k^N} = \frac{\partial J}{\partial y^N} \frac{\partial y^N}{\partial x_k^N} = \pm e^N \left(c_{\alpha,k} + \sum_{j=1}^{N_w} \frac{c_{\alpha, N_e+N_s+j}}{d_{jk}} \frac{\partial \Phi_j(x^N)}{\partial z_{jk}} \right) \tag{61}$$

avec $\alpha = N_e+N_s+N_w+1$.

<u>Pour les copies de $n=N-1$ à 2</u>, nous avons :

Pour la sortie :

$$\frac{\partial J}{\partial y^n} = \pm e^n \tag{62}$$

Pour les variables d'état en sortie, $k=N_e+N_s+N_w+2$, ... , $N_e+2N_s+N_w+1$:

$$\frac{\partial J}{\partial x_k^n} = \frac{\partial J}{\partial x_{k\pm N_s\pm N_w\pm 1}^{n+1}} \tag{63}$$

Pour les variables d'état en entrée, $k=N_e+1$, ... , N_e+N_s :

$$\frac{\partial J}{\partial x_k^n} = \frac{\partial J}{\partial y^n}\frac{\partial y^n}{\partial x_k^n} = \pm e^n \left(c_{\alpha,k} + \sum_{j=1}^{N_w} \frac{c_{\alpha,\,N_e+N_s+j}}{d_{jk}} \frac{\partial \Phi_j(x^n)}{\partial z_{jk}} \right) + \sum_{j=N_e+N_s+N_w+2}^{N_e+2N_s+N_w+1} c_{j,k} \frac{\partial J}{\partial x_j^n} \tag{64}$$

Pour la copie n=1, nous avons :

Pour la sortie :

$$\frac{\partial J}{\partial y^1} = \pm e^1 \tag{65}$$

Pour les variables d'état en sortie, $k=N_e+N_s+N_w+2$, ... , $N_e+2N_s+N_w+1$:

$$\frac{\partial J}{\partial x_k^1} = \frac{\partial J}{\partial x_{k\pm N_s\pm N_w\pm 1}^{2}} \tag{66}$$

Pour les variables d'état en entrée, $k=N_e+1$, ... , N_e+N_s :

le calcul des $\dfrac{\partial J}{\partial x_k^1}$ n'est pas utile.

VI.2.2.2 Calcul du gradient de J par rapport aux paramètres du réseau.

Disposant des dérivées calculées précédemment, il est à présent possible de déterminer le gradient de la fonction de coût par rapport à chacune des composantes du vecteur des paramètres ajustables.

Pour les coefficients directs sur la sortie :

$$\frac{\partial J}{\partial c_{\alpha j}} = \sum_{n=1}^{N} \frac{\partial J}{\partial y^n}\frac{\partial y^n}{\partial c_{\alpha j}^n} = \pm \sum_{n=1}^{N} e^n x_j^n \quad \text{avec } j = 1 , \dots , N_e+N_s \text{ et } \alpha = N_e+N_s+N_w+1 \tag{67}$$

Pour les coefficients directs sur les états :

$$\frac{\partial J}{\partial c_{k,j}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_k^n}\frac{\partial x_k^n}{\partial c_{k,j}^n} = \sum_{n=1}^{N} \frac{\partial J}{\partial x_k^n} x_j^n \tag{68}$$

avec $j = 1 , \dots , N_e+N_s$ et $k = N_e+N_s+N_w+2 , \dots , N_e+2N_s+N_w+1$

Pour les pondérations sur la sortie :

$$\frac{\partial J}{\partial c_{\alpha,j+N_e+N_s}} = \sum_{n=1}^{N} \frac{\partial J}{\partial y^n} \frac{\partial y^n}{\partial c^n_{\alpha,j+N_e+N_s}} = \pm \sum_{n=1}^{N} e^n \, \Phi_j(x^n) \tag{69}$$

avec $j = 1, \ldots, N_w$ et $\alpha = N_e+N_s+N_w+1$

Pour les pondérations sur les états :

$$\frac{\partial J}{\partial c_{k,N_e+N_s+j}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x^n_k} \frac{\partial x^n_k}{\partial c^n_{k,N_e+N_s+j}} = \sum_{n=1}^{N} \frac{\partial J}{\partial x^n_k} \Phi_j(x^n) \tag{70}$$

avec $j = 1, \ldots, N_w$ et $k = N_e+N_s+N_w+2, \ldots, N_e+2N_s+N_w+1$

Pour le terme constant sur le neurone de sortie :

$$\frac{\partial J}{\partial c_0} = \sum_{n=1}^{N} \frac{\partial J}{\partial y^n} \frac{\partial y^n}{\partial c^n_0} = \pm \sum_{n=1}^{N} e^n \tag{71}$$

Pour les translations, j= 1, ... N_w et k=*1* , ... , N_e+N_s :

$$\frac{\partial J}{\partial m_{jk}} = \sum_{n=1}^{N} \frac{\partial J}{\partial m^n_{jk}} = \sum_{n=1}^{N} \left(\frac{\partial J}{\partial y^n} \frac{\partial y^n}{\partial m^n_{jk}} + \sum_{l=N_e+N_s+N_w+2}^{l=N_e+2N_s+N_w+1} \frac{\partial J}{\partial x^n_l} \frac{\partial x^n_l}{\partial m^n_{jk}} \right) \tag{72}$$

En remplaçant les dérivées par leurs expressions (déjà calculées), on obtient :

$$\frac{\partial J}{\partial m_{jk}} = \sum_{n=1}^{N} \frac{\partial J}{\partial m^n_{jk}} = \sum_{n=1}^{N} \left(e^n \frac{c_{\alpha,N_e+N_s+j}}{d_{jk}} \frac{\partial \Phi_j(x^n)}{\partial z_{jk}} \pm \sum_{l=N_e+N_s+N_w+2}^{N_e+2N_s+N_w+1} \frac{\partial J}{\partial x^n_l} \frac{c_{l,N_e+N_s+j}}{d_{jk}} \frac{\partial \Phi_j(x^n)}{\partial z_{jk}} \right) \tag{73}$$

Enfin, une factorisation permet d'alléger cette expression :

$$\frac{\partial J}{\partial m_{jk}} = \sum_{n=1}^{N} \frac{\partial J}{\partial m^n_{jk}} = \sum_{n=1}^{N} \frac{1}{d_{jk}} \frac{\partial \Phi_j(x^n)}{\partial z_{jk}} \left(c_{\alpha,N_e+N_s+j} \, e^n \pm \sum_{l=N_e+N_s+N_w+2}^{N_e+2N_s+N_w+1} c_{l,N_e+N_s+j} \frac{\partial J}{\partial x^n_l} \right) \tag{74}$$

Pour les dilatations, j=1 , ... N_w et k=1 , ... , N_e+N_s :

$$\frac{\partial J}{\partial d_{jk}} = \sum_{n=1}^{N} \frac{\partial J}{\partial d^n_{jk}} = \sum_{n=1}^{N} \left(\frac{\partial J}{\partial y^n} \frac{\partial y^n}{\partial d^n_{jk}} + \sum_{l=N_e+N_s+N_w+2}^{N_e+2N_s+N_w+1} \frac{\partial J}{\partial x^n_l} \frac{\partial x^n_l}{\partial d^n_{jk}} \right) \tag{75}$$

En remplaçant les dérivées par leurs expressions (déjà calculées), on obtient :

$$\frac{\partial J}{\partial d_{jk}} = \sum_{n=1}^{N} \frac{\partial J}{\partial d^n_{jk}} = \sum_{n=1}^{N} \frac{z^n_{jk}}{d_{jk}} \frac{\partial \Phi_j(x^n)}{\partial z_{jk}} \left(c_{\alpha,N_e+N_s+j} \, e^n \pm \sum_{l=N_e+N_s+N_w+2}^{N_e+2N_s+N_w+1} c_{l,N_e+N_s+j} \frac{\partial J}{\partial x^n_l} \right) \tag{76}$$

VI.2.2.3 *Commentaire sur le choix des variables d'état.*

Lors de la conception d'un modèle d'état d'un processus, il arrive que le cahier de charges exige qu'une des composantes du vecteur d'état soit la sortie du processus. Dans un tel cas, le calcul du gradient de *J* présenté ci-dessus se trouve légèrement modifié. La principale modification à apporter se situe au niveau du calcul du gradient de *J* par rapport à la sortie du réseau. Suivant la notation adoptée pour la présentation des relations ci–dessus, si la sortie est considérée comme une variable d'état du modèle, on la notera de la manière suivante :

$$y^n = x^n_{N_e+N_s+N_w+1} \tag{77}$$

La relation (64) devient dans ce cas :

$$\frac{\partial J}{\partial y^n} = \frac{\partial J}{\partial x^n_{N_e+N_s+N_w+1}} = \pm e^n + \frac{\partial J}{\partial x^{n+1}_{N_e+1}} \tag{78}$$

Remarquons que cette relation est très semblable à la relation (37) qui concerne le calcul du gradient dans le cas d'un réseau entrée–sortie.

Nous présentons en annexe de ce mémoire les modifications à apporter aux équations précédentes (de 59 à 76). Le calcul du gradient exposé dans cette annexe permet aisément le passage d'un modèle où la sortie est une variable d'état au cas où tous les états sont indépendants de la sortie.

VI.2.3 Calcul du gradient dans le sens direct.

La motivation pour le calcul du gradient de la fonction coût dans le sens direct est la même que dans le cas de la modélisation entrée-sortie : le calcul du Hessien approché nécessite la connaissance de grandeurs qui ne sont pas fournies par le calcul par rétropropagation.

Le principe du calcul de $\frac{\partial y^n}{\partial \theta}$ en prenant en considération toutes les copies est donné par la relation (42). On considère le problème de l'évaluation du second membre de la relation (43) mais cette fois dans le cadre d'un réseau d'état. On a :

$$\frac{\partial y^n}{\partial \theta^m} = \sum_{k=N_e+1}^{N_e+N_s} \frac{\partial y^n}{\partial x^n_k} \frac{\partial x^n_k}{\partial \theta^m} \text{ avec } m < n \tag{79}$$

En faisant la sommation de toutes les équations de la forme de (79) pour m allant de 1 à n–1, on obtient :

$$\sum_{m=1}^{n-1} \frac{\partial y^n}{\partial \theta^m} = \sum_{k=N_e+1}^{N_e+N_s} \frac{\partial y^n}{\partial x^n_k} \left(\sum_{m=1}^{n-1} \frac{\partial x^n_k}{\partial \theta^m} \right) \tag{80}$$

Étant donné que nous avons $x^n_k = x^{n\pm1}_{k+N_s+N_w+1}$, nous pouvons écrire les égalités suivantes :

$$\sum_{m=1}^{n\pm1} \frac{\partial x_k^n}{\partial \theta^m} = \sum_{m=1}^{n\pm1} \frac{\partial x_{k+N_s+N_w+1}^{n\pm1}}{\partial \theta^m} = \frac{\partial x_{k+N_s+N_w+1}^{n\pm1}}{\partial \theta} \tag{81}$$

En injectant ce résultat dans la relation (80), on aboutit à :

$$\sum_{m=1}^{n\pm1} \frac{\partial y^n}{\partial \theta^m} = \sum_{k=N_e+1}^{N_e+N_s} \frac{\partial y^n}{\partial x_k^n} \frac{\partial x_{k+N_s+N_w+1}^{n\pm1}}{\partial \theta} \tag{82}$$

Ce dernier résultat aboutit à l'écriture de la relation qui nous permet de calculer la dérivée de la sortie par rapport au vecteur θ :

$$\frac{\partial y^n}{\partial \theta} = \sum_{k=N_e+1}^{N_e+N_s} \frac{\partial y^n}{\partial x_k^n} \frac{\partial x_{k+N_s+N_w+1}^{n\pm1}}{\partial \theta} + \frac{\partial y^n}{\partial \theta^n} \tag{83}$$

Nous allons analyser sommairement la complexité de ce calcul.

$\frac{\partial y^n}{\partial \theta^n}$ est la dérivée de la sortie par rapport au vecteur des paramètres de la même copie. Les différentes composantes de ce vecteur sont données par les relations (20) à (24).

$\frac{\partial y^n}{\partial x_k^n}$ est la dérivée de la sortie par rapport à une variable d'état en entrée de la même copie. Le calcul est immédiat et est donné par la relation suivante :

$$\frac{\partial y^n}{\partial x_k^n} = \sum_{j=1}^{N_w} \frac{c_{\alpha,N_e+N_s+j}}{d_{jk}} \frac{\partial \Phi_j\left(x^n\right)}{\partial z_{jk}} + c_{\alpha,k} \tag{84}$$

où $\alpha = N_e+N_s+N_w+1$ est l'indice du neurone de sortie.

$\frac{\partial x_{k+N_s+N_w+1}^{n\pm1}}{\partial \theta}$ est plus délicat à évaluer puisqu'il s'agit d'une quantité qui implique le calcul des dérivées par rapport à chacune des copies d'indice inférieur ou égal. Étant donné que les variables d'états sont indépendantes de la sortie, il faut trouver une relation récurrente du type de (48) pour calculer à chaque copie les dérivées des états par rapport aux paramètres de manière économique. Par analogie avec la relation (48), nous proposons la relation suivante :

$$\frac{\partial x_l^{n\pm1}}{\partial \theta} = \sum_{k=N_e+1}^{N_e+N_s} \frac{\partial x_l^{n\pm1}}{\partial x_k^{n\pm1}} \frac{\partial x_{k+N_s+N_e+1}^{n\pm2}}{\partial \theta} + \frac{\partial x_l^{n\pm1}}{\partial \theta^{n\pm1}} \tag{85}$$

avec $n \geq 2$ et $l = N_e+N_s+N_w+2, \ldots, N_e+2N_s+N_w+1$

En résumé, l'algorithme de calcul du Hessien approché à l'aide des relations présentées ci-dessus est le suivant :

Pour n allant de 1 à N faire :

Pour m allant de 1 à N_s faire :

Exécution de la relation (85) en utilisant les résultats de cette même relation à l'étape précédente.

Fin de la boucle sur m.

Exécution de la relation (83).

Fin de la boucle sur n.

Construction du Hessien approché à partir de la relation (41).

Commentaire.

Les relations que nous venons d'établir pour l'apprentissage de réseaux d'ondelettes nécessitent un volume de calcul plus important que les relations équivalentes relatives aux réseaux de neurones à sigmoïdes.

Pour l'étude des exemples présentés dans le chapitre V, nous avons mis en œuvre le calcul du gradient par rétropropagation. De ce fait, l'apprentissage de nos réseaux d'état ne peut se faire qu'à l'aide de l'algorithme de BFGS.

V I.2.4 Initialisation des paramètres du réseau.

Un problème intéressant qui se pose est celui de l'initialisation des paramètres du réseau d'ondelettes dans le cas d'une modélisation d'état.

Deux cas peuvent se présenter :
- les états sont mesurables,
- les états ne sont pas mesurables.

Dans le cas où les états sont mesurables, la question peut être résolue de la même façon que dans le cas des réseaux pour la modélisation statique. En effet, les domaines de toutes les entrées étant connus (entrées externes et états), le calcul des translations et des dilatations initiales est immédiat (suivant la procédure proposée).

Dans le cas où les états ne sont pas mesurables, situation que nous avons choisi d'étudier dans nos exemples, le domaine des entrées du réseau (plus particulièrement les états) ne sont pas connus avant l'apprentissage. Le calcul des translations et des dilatations initiales est donc rendu délicat.

Nous avons initialisé ces paramètres en faisant l'hypothèse que les états sont dans un domaine de longueur comparable à celui de la sortie et centré en zéro. En fait, le but de cette procédure et d'éviter une initialisation aléatoire et de placer les ondelettes initialement dans le domaine de la variable d'entrée. Étant donné que les coefficients des neurones d'état sont initialisés à de petites valeurs, cette condition est vérifiée : en début d'apprentissage, les valeurs des états sont à l'intérieur du support des ondelettes.

VII. LE PROBLÈME MAÎTRE–ÉLÈVE ET LES RÉSEAUX D'ONDELETTES.

Le problème dit "maître–élève" consiste à engendrer des données à l'aide d'un réseau de fonctions "maître" dont les poids sont fixés, puis à retrouver ce réseau par apprentissage d'un réseau "élève" qui possède la même architecture. Ainsi, on est assuré que la fonction de régression recherchée (le réseau "maître") fait partie de la famille de fonctions du modèle (le réseau "élève"). L'intérêt de ce problème est qu'il permet de tester l'efficacité des algorithmes d'apprentissage, et notamment d'évaluer l'influence des minima locaux : en effet, si l'algorithme d'apprentissage converge en un temps raisonnable, et s'il ne trouve pas un minimum local, le réseau obtenu après apprentissage doit être le réseau maître, aux erreurs d'arrondi près.

Le système d'apprentissage pour un tel problème peut être illustré de la manière suivante :

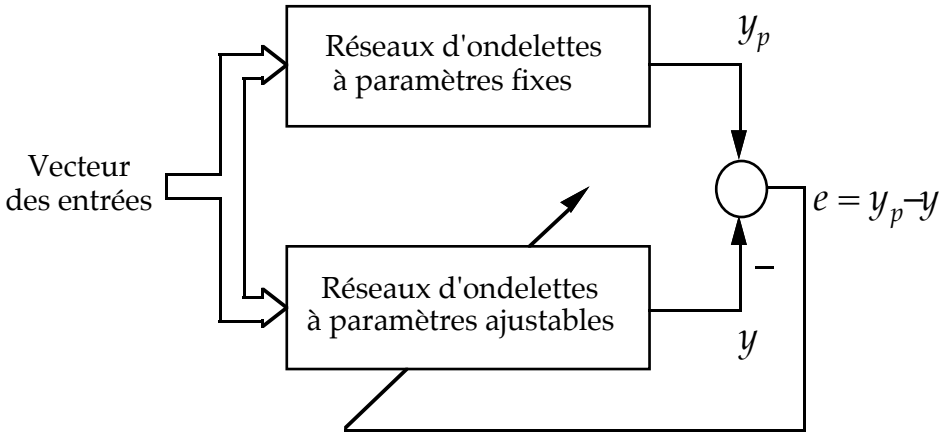


Figure 15. Système d'apprentissage pour le problème du "maître–élève".

Contrairement aux réseaux statiques présentés dans ce chapitre, on a supprimé la partie affine du réseau (sauf le terme constant sur le neurone linéaire de sortie que l'on conserve). La relation (12) donnant l'expression de la sortie du réseau est dans ce cas donnée par :

$$y = \sum_{j=1}^{N_w} c_j \Phi_j(x) + a_0 \tag{86}$$

Nous allons tout d'abord établir le résultat suivant : *l'existence des minima locaux de la fonction de coût ne dépend que de l'architecture du réseau maître, et ne dépend pas de la valeur des paramètres de celui-ci*. Nous préciserons ensuite les conditions des expériences numériques que nous avons menées, puis nous en décrirons et commenterons les résultats.

VII.1 Minima locaux de la fonction de coût.

Considérons un réseau maître réalisant la fonction $f(x, \theta_0)$, où x et θ_0 sont les vecteurs des variables et des paramètres respectivement, et un réseau élève $f(x, \theta)$. La fonction de coût minimisée pendant l'apprentissage est :

$$J = \frac{1}{2} \sum_{n=1}^{N} \left[f(x_n, \theta_0) \pm f(x_n, \theta) \right]^2 \tag{87}$$

où x_n désigne le vecteur des variables pour l'exemple n, et où N est le nombre d'éléments de l'ensemble d'apprentissage.
Le gradient de la fonction de coût a pour expression :

$$\frac{\partial J}{\partial \theta} = \pm \sum_{n=1}^{N} \left[f(x_n, \theta_0) \pm f(x_n, \theta) \right] \frac{\partial f(x_n, \theta)}{\partial \theta} \tag{88}$$

Pour un minimum (local ou global), obtenu pour un vecteur de paramètres θ_m, on a donc :

$$\left[\frac{\partial J}{\partial \theta} \right]_{\theta = \theta_m} = \sum_{n=1}^{N} \left[f(x_n, \theta_0) \pm f(x_n, \theta_m) \right] \left[\frac{\partial f(x_n, \theta)}{\partial \theta} \right]_{\theta = \theta_m} = 0 \ . \tag{89}$$

Supposons que l'on fasse varier le vecteur des paramètres du réseau maître, et cherchons quelle variation il faut faire subir au vecteur θ_m pour qu'il corresponde toujours à un minimum. Il suffit pour cela d'écrire la différentielle totale du gradient :

$$\begin{aligned} \mathrm{d}\left[\frac{\partial J}{\partial \theta} \right]_{\theta = \theta_m} &= \sum_{n=1}^{N} \left[\frac{\partial f(x_n, \theta)}{\partial \theta} \right]_{\theta = \theta_0} \left[\frac{\partial f(x_n, \theta)}{\partial \theta} \right]_{\theta = \theta_m} \mathrm{d}\theta_0 + \\ \sum_{}^{N} \left\{ \left[f(x_n, \theta_0) \pm f(x_n, \theta_m) \right] \left[\frac{\partial^2 f(x_n, \theta)}{\partial \theta^2} \right] \right. & \left. \pm \left[\frac{\partial f(x_n, \theta)}{\partial \theta} \right]^2_{\theta \quad \theta} \right\} \mathrm{d}\theta_m = 0 \ . \end{aligned} \tag{90}$$

On peut donc écrire :

$$\frac{\mathrm{d}\theta_m}{\mathrm{d}\theta_0} = \pm \frac{\displaystyle\sum_{n=1}^{N} \left[\frac{\partial f(x_n, \theta)}{\partial \theta} \right]_{\theta = \theta_0} \left[\frac{\partial f(x_n, \theta)}{\partial \theta} \right]_{\theta = \theta_m}}{\displaystyle\sum_{n=1}^{N} \left\{ \left[f(x_n, \theta_0) \pm f(x_n, \theta_m) \right] \left[\frac{\partial^2 f(x_n, \theta)}{\partial \theta^2} \right]_{\theta = \theta_m} \pm \left[\frac{\partial f(x_n, \theta)}{\partial \theta} \right]^2_{\theta = \theta_m} \right\}} \tag{91}$$

sous réserve que le dénominateur ne soit pas nul (il est nul si la fonction f est constante).

Pour les paramètres θ^l dont f dépend linéairement, on a $d\theta^l_m / d\theta^l_0 = 1$. Si le modèle est linéaire par rapport à tous les paramètres, il y a évidemment un seul minimum $\theta_m = \theta_0$.
Ainsi, si l'on connaît les minima de la fonction de coût pour une valeur donnée θ_0 des paramètres du réseau maître et pour un ensemble d'apprentissage donné, on peut, en principe, en déduire, par intégration de l'équation (91), les valeurs des minima pour toute autre valeur des paramètres du réseau maître. Changer les valeurs des paramètres du réseau maître ne change donc pas le nombre de minima locaux, mais seulement leur position dans l'espace des paramètres.
En conséquence, pour étudier l'influence des minima locaux sur l'apprentissage, nous choisirons, pour une architecture donnée, une seule valeur des paramètres.

VII.2 Choix de la séquence d'apprentissage.

Pour qu'un apprentissage soit efficace, il est nécessaire que la séquence d'apprentissage soit suffisamment riche pour représenter le comportement du processus. D'autre part, le nombre d'exemples constituant la séquence doit être très supérieur à celui des paramètres ajustables. Le réseau le plus volumineux que nous ayons considéré est constitué de 5 entrées et de 5 ondelettes, soit 56 paramètres.

Pour tous les exemples étudiés, nous avons considéré une séquence d'apprentissage formée par 2000 exemples. Nous faisons l'hypothèse que cette séquence représente suffisamment le processus (réseau maître) à apprendre dans le domaine des entrées choisi.

VII.3 Choix du domaine des entrées et des paramètres du réseau maître.

La question du choix du domaine des entrées est très importante puisqu'elle détermine le domaine dans lequel on veut modéliser le processus.

Dans le cas de réseaux d'ondelettes, ce choix est très lié à celui des paramètres du réseau maître. En effet, si l'on fait un choix pour le domaine des entrées, il faut choisir les paramètres du réseau maître de telle manière que les supports des ondelettes aient une intersection non nulle avec ces domaines. Sinon, la sortie y_p sera partout nulle.

Nous avons choisi pour les entrées des valeurs aléatoires suivant une distribution gaussienne centrée réduite (de moyenne nulle et de variance 1). Une analyse simple des entrées obtenues montre qu'elles sont toutes comprises dans un intervalle que l'on peut encadrer par [–4, 4]. Cette information est utile pour le choix des paramètres du réseau maître, comme nous le verrons dans le paragraphe suivant.

On choisit les translations des ondelettes du réseau maître de manière aléatoire (suivant une distribution uniforme) dans le domaine des entrées. Les dilatations sont également choisies de manière uniformément distribuée mais cette fois-ci dans le domaine [0.6, 2.6]. Cet intervalle est centré autour de la valeur $\frac{0.6+2.6}{2}=1.6$. Or, en reprenant la procédure d'initialisation utilisée pour l'apprentissage des réseaux d'ondelettes, on peut remarquer que cette valeur est celle des dilatations initiales (étant donné l'intervalle [–4, 4]). Ce choix est motivé par le fait qu'il correspond à des ondelettes dont les supports sont de l'ordre de la longueur de l'intervalle comprenant les entrées (c'est-à-dire [–4, 4]).

VII.4 Choix de l'algorithme et de l'initialisation du réseau.

Comme nous l'avons indiqué plus haut, nous avons le choix entre l'algorithme de BFGS et celui de Levenberg–Marquardt, afin d'éviter d'introduire, comme paramètre supplémentaire de notre étude, le nombre d'itérations de gradient simple à effectuer avant le démarrage de l'algorithme du second ordre.

L'initialisation du réseau s'effectue sur la base de la procédure présentée au paragraphe IV.2 de ce chapitre. Étant donné que les termes directs ont été retirés (voir relation 86), seules les pondérations des ondelettes et le terme constant sur le neurone linéaire de sortie sont initialisés de manière aléatoire.

VII.5 Approche adoptée pour l'étude du problème.

On se propose d'étudier les performances des réseaux d'ondelettes sur le problème maître–élève en prenant en considération l'influence du nombre des entrées et du nombre d'ondelettes dans la couche cachée.

Pour chaque architecture, nous choisissons un vecteur de paramètres pour le réseau maître, et nous effectuons vingt apprentissages avec vingt initialisations différentes pour les pondérations. Nous estimons qu'un apprentissage est un succès lorsque le vecteur des paramètres trouvé correspond exactement à celui du réseau maître (aux erreurs d'arrondi près).

VII.6 Résultats et commentaires.

Le tableau 10 présente, pour chaque architecture utilisée (caractérisée par le nombre d'entrées et le nombre d'ondelettes), le nombre d'apprentissages effectués avec succès avec un ensemble d'apprentissage constitué d'exemples pour lesquels les entrées suivent une loi gaussienne centrée réduite.

		Nombre d'ondelettes				
		1	2	3	4	5
Nombre d'entrées	1	20	20	0	0	0
	2	20	20	20	0	0
	3	20	20	16	0	0
	4	20	0	0	0	0
	5	20	0	0	0	0

Tableau 10. Résultats du problème maître–élève sur les réseaux d'ondelettes.

On observe que, au-delà de 3 entrées et 3 ondelettes, il devient pratiquement impossible de retrouver le réseau maître : l'apprentissage aboutit à des minima locaux de la fonction de coût. Les résultats obtenus avec d'autres distributions des entrées sont tout-à-fait analogues. Pour les réseaux à sigmoïdes, des expériences similaires ont montré que, au contraire, la probabilité de succès est d'autant plus grande que le réseau est plus grand [Stoppi97]. Les minima locaux semblent donc être plus gênants pour l'apprentissage des réseaux d'ondelettes que pour celui des réseaux de neurones. Il faut noter néanmoins que, dans un problème pratique, les données sont toujours entachées de bruit : on ne cherche donc pas à annuler l'erreur comme dans le cas du problème maître-élève, mais à trouver un minimum tel que la variance de l'erreur de modélisation soit égale à celle du bruit.

VIII. CONCLUSION.

Dans ce chapitre, nous avons présenté la modélisation statique et dynamique de processus à l'aide de réseaux d'ondelettes fondés sur la transformée en ondelette continue. Nous avons montré que les ondelettes peuvent être considérées comme des fonctions paramétrées (à paramètres continus), et qu'une combinaison linéaire d'ondelettes dont les centres et les dilatations sont ajustables peut, au même titre qu'un réseau de neurones,

constituer un modèle non linéaire de processus. Les paramètres de ce modèle peuvent être estimés à partir d'observations, de telle manière que la sortie du modèle approche la fonction de régression de la grandeur à modéliser.

La sortie du modèle n'étant pas linéaire par rapport aux dilatations et aux translations, l'estimation des paramètres doit être effectuée à l'aide d'algorithmes itératifs. Les ondelettes étant locales, le problème de l'initialisation des dilatations et translations est très important. Nous avons proposé une procédure d'initialisation simple qui prend en considération cette propriété.

Nous avons également montré que les réseaux d'ondelettes peuvent être utilisés pour la modélisation dynamique de processus, et peuvent constituer soit des modèles entrée-sortie, soit des modèles d'état. Pour ces deux types de modèles, nous avons établi les procédures de calcul du gradient, par rétropropagation et dans le sens direct. Les expressions obtenues nous ont montré que la complexité des calculs est plus importante que dans le cas de réseaux à fonctions dorsales.

Enfin, nous avons présenté une étude du problème "maître-élève" pour des réseaux d'ondelettes. Nous avons prouvé que, dans un tel cas, le nombre de minima ne dépend pas de la valeur des paramètres du réseau maître, mais seulement de son architecture et de l'ensemble d'apprentissage. Les résultats obtenus montrent que le nombre de minima locaux de la fonction de coût croît rapidement avec le nombre d'ondelettes et avec le nombre de variables du modèle. Nous présenterons dans le chapitre V des exemples de modélisation dynamique à l'aide de réseaux d'ondelettes.