Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud Mammeri, Tizi-Ouzou

Faculté du Génie Electrique et Informatique

Département d'Automatique

MEMOIRE DE MAGISTER

Spécialité : Automatique

Option: Automatique des Systèmes Continus et Productique

Présenté par :

M. HAMMOUCHE Sofiane

Thème :

Identification d'un modèle fractionnaire à l'aide des réseaux de neurones.

Soutenu le 26/06/2012, devant le jury d'examen composé de :

M. DIAF Moussa	**Professeur à l'UMMTO**	**Président**
M. MELLAH Rabah	**M.C, Classe A à l'UMMTO**	**Rapporteur**
M. LAGHROUCHE Mourad	**Professeur à l'UMMTO**	**Examinateur**
M. GUERMAH Saïd	**M.C, Classe B à l'UMMTO**	**Examinateur**
M^{me}. DJAMAH Tounsia	**M.C, Classe B à l'UMMTO**	**Examinatrice**

Remerciements

Le travail présenté dans ce mémoire a été effectué au sein du Laboratoire de Conception et Conduite des Systèmes de Production (L2CSP) de l'Université Mouloud Mammeri de Tizi-Ouzou (UMMTO).

Je tiens tout d'abord à exprimer mes sincères remerciements à mon promoteur M. MELLAH Rabah, Maître de conférences Classe A à l'UMMTO pour la confiance qu'il m'a accordée, sa disponibilité, ses conseils et son aide assez précieuse, tout au long de ma thèse.

Je remercie M. DIAF MOUSSA, Professeur à l'UMMTO, pour avoir accepté d'examiner ce travail et de présider le jury de soutenance de ce mémoire.

Mes vifs remerciements vont également à M^{me}. DJAMAH Tounsia, Maître de conférences Classe B à l'UMMTO, pour les conseils qu'elle m'a prodigués et pour avoir accepté de participer au jury de soutenance.

Je remercie très vivement M. LAGHROUCHE Mourad, Professeur à l'UMMTO et M. GUERMAH Saïd, Maître de conférences classe B à l'UMMTO, qui ont accepté de rapporter sur ma thèse et m'ont fait l'honneur d'être membres du jury de soutenance.

Mes remerciements s'adressent également à tous les membres du Laboratoire de Conception et Conduite des Systèmes de Production (L2CSP) et particulièrement M. MANSOURI Rachid, et tous les collègues pour l'excellente ambiance de travail qu'ils ont créé.

Enfin, je remercie toute ma famille, et tous mes amis, qui de prés ou de loin m'ont supporté, soutenu et encouragé tout au long de ces années.

Sommaire

Sommaire

Sommaire

Introduction générale

Les systèmes décrits par des modèles d'ordre fractionnaires, utilisant des équations différentielles fractionnaires basées sur la dérivée non entière ont suscité l'intérêt de la communauté scientifique. Les ingénieurs ont seulement compris l'importance des équations différentielles d'ordre non entier, que durant les trois dernières décennies, surtout lorsque ils ont observé que la description de quelques systèmes est plus exacte, que lorsque la dérivée fractionnaire est utilisée (Boroomand et Bagher, 2009). En effet, Nous assistant récemment à la modélisation de nombreux phénomènes physiques par des systèmes d'ordre fractionnaire, motivée par l'application de ces derniers à divers domaines des sciences de l'ingénieur comme la modélisation des processus de diffusion : thermique (Battaglia et al, 2000 ; Djamah et al, 2008), acoustique, électrochimique (Sabatier et al, 2006), électromagnétiques, etc.

En outre, les procédés réels sont souvent complexes, ainsi il est difficile voir impossible d'associer des lois physiques décrivant leur dynamique, alors « la modélisation expérimentale » appelée communément « identification » constitue une alternative à la modélisation classique, et dans ce cas, on parle de modèle de comportement ou boite noire.

L'objectif majeur d'un modèle mathématique représentant aussi fidèlement que possible le comportement d'un processus à étudier à partir des données expérimentales, étant alors l'approximation de ces dernières par un système d'équations mathématiques dont les paramètres à identifier n'ont pas toujours de signification physique particulière, mais servent uniquement à l'ajustement du modèle, dans un domaine de fonctionnement limité. En pratique les boites sont plutôt grises, et le modèle est construit à partir des données expérimentales, tout en tenant compte de la connaissance disponible à priori.

Les systèmes fractionnaires sont caractérisés par la propriété de mémoire longue et de dimension infinie (systèmes à paramètres distribués). Contrairement à la dérivée entière qui ne fait intervenir qu'un nombre limité de valeurs passées de la fonction à dériver, la dérivée non entière nécessite la connaissance de tout le passé de la fonction qui donne la dimension infinie. Cette caractérisation impose l'utilisation du modèle de dimension infinie pour modéliser un système fractionnaire par des modèles entiers. Dans ce contexte, l'identification de tels systèmes par des modèles entiers s'avers inadaptée, et requiert un nombre important de paramètres pénalisant ainsi le modèle obtenu.

Par ailleurs, le développement des méthodes d'identifications basées sur les modèles d'ordre non entiers apparait donc comme une exigence, mais ils se révèlent donc nettement plus complexes que dans le cas entier, du fait de l'estimation des ordres fractionnaires aux

mêmes titres que les paramètres du modèle (Djamah et al, 2010). Ces considérations nécessitent le recours à d'autres outils plus performants tels que les réseaux de neurones sous ses différents aspects, pour répondre au problème de la complexité de la dynamique d'un système fractionnaire, dans lequel s'inscrit le thème de ce mémoire de magistère qui consiste à identifier un modèle fractionnaire à l'aide des réseaux de neurones.

En effet les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle, dont chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Ainsi les réseaux de neurones artificiels, constituent une approche permettant d'aborder sous des angles nouveaux les problèmes de perception, de mémoire, d'apprentissage et de raisonnement. Ils s'avèrent aussi des alternatives très prometteuses pour contourner certaines des limitations des ordinateurs classiques. Grâce à leur traitement parallèle de l'information et de leur mécanisme inspirés de des cellules nerveuses (neurones). Ils infèrent des propriétés émergentes permettant de solutionner des problèmes jadis qualifiés de complexes.

Notre travail a pour objectif principal de mettre en œuvre deux architectures neuronales pour l'identification de modèles fractionnaires, permettant d'assurer des simulations faciles et rapides. La première architecture appelée « réseaux de neurones multicouches retardé en entrée et en sortie » consiste à mettre en œuvre un réseau de neurones récurrent qui permet de reproduire le caractère fractionnaire du système, une fois est entrainé à partir de plusieurs couples entrée/sortie. Quant à la seconde, appelée « réseaux de neurones à espace d'état », utilise la représentation d'état des systèmes pour reconstituer le comportement des états et de la sortie des systèmes. Cette méthode utilise deux réseaux de neurones, dont le premier est entrainé à partir du signal d'entrée et des états du système pour reproduire l'évolution des états du système. Ces derniers sont ensuite utilisés pour entrainer un autre réseau de neurones, afin de reproduire le comportement de la sortie du système (Zamarreno, 2000). Ainsi, nous avons organisé notre mémoire comme suit :

Dans le chapitre 1, la présentation de l'état de l'art des systèmes à dérivée d'ordre non entier est entreprise. Elle est constituée d'une étude permettant de comprendre les formalismes mathématiques mis en jeu pour l'obtention des modèles appropriés aux systèmes fractionnaires.

Le chapitre 2, présente les réseaux de neurones sous deux aspects : réseau de neurones multicouches de type perceptron et réseau de neurones à espace d'état. L'apprentissage de ces derniers est pris en charge par la méthode de Levenberg-Marquardt.

Le chapitre 3, aborde la mise en œuvre de quatre architectures neuronales, pour l'identification et la prédiction à un pas des modèles d'ordre entier et non entier. La fin de ce chapitre est consacrée aux simulations.

Dans le chapitre 4, nous exploitons deux bases d'échantillonnage entrée/sortie, obtenues expérimentalement sur des systèmes fractionnaires réels, pour valider les performances des architectures neuronales multicouches bouclée et non bouclée, proposées au troisième chapitre.

Nous terminons notre mémoire par une conclusion générale récapitulant nos principaux résultats et quelques perspectives.

Chapitre 1

Description d'un système d'ordre non entier et son modèle d'état

1. Introduction

Quand on introduit la notion de dérivée, on se rend compte vite qu'on peut appliquer le concept de dérivée à la fonction de dérivée elle-même, ainsi qu'introduire la dérivée seconde, puis les dérivées successives d'ordre entier. L'intégration est une opération inverse de la dérivée, qui peut être éventuellement considérée comme une dérivée d'ordre ''moins un''. En outre, on peut aussi se demander si ces dérivées d'ordres successifs ont un équivalent d'ordre fractionnaire (non entier).

On pourrait penser que cette recherche de dérivation fractionnaire est une question mathématique ''pure'' sans intérêt pour l'ingénieur. Pourtant un exemple simple de la mécanique des fluides montre comment la dérivée d'ordre un demi apparait tout naturellement quand on veut expliciter un flux de chaleur sortant latéralement d'un écoulement fluide en fonction de l'évolution temporelle de la source interne. La dérivée d'ordre un demi étant introduite, on doit être vigilant quant à sa définition précise dans les situations les plus générales. Il en est de même pour la définition de la dérivée d'ordre fractionnaire α, où α est typiquement un nombre réel entre zéro et un (Dubois et al, 2008).

2. Outils mathématiques de base

Pour comprendre comment, à partir de la formule de Cauchy, on est arrivé à définir la dérivation fractionnaire, il convient de rappeler et d'introduire dans ce début de chapitre quelques définitions mathématiques de base.

2.1. Transformation et convolution de Laplace

La transformée de Laplace est un opérateur très utilisé en analyse des systèmes. Le passage au domaine de Laplace présente l'avantage de résoudre de manière algébrique des

opérations mathématiques fondamentales telles que le calcul différentiel et la convolution, plus complexes dans le domaine temporel.

Soit une fonction causale $f(t)$ positive pour $t \geq 0$, la transformée de Laplace de $f(t)$ est définie par :

$$L\{f(t)\} = F(s) = \int_0^\infty e^{-st} f(t)dt \quad (1.1)$$

La transformée de Laplace de sa i éme dérivée, notée $f^{(i)}(t)$, est donnée par (Gong et al, 2010) :

$$L\{f^{(i)}(t)\} = s^i F(s) - \sum_{k=0}^{i-1} s^{i-k-1} f^{(k)}(0) = s^i F(s) - \sum_{k=0}^{i-1} s^k f^{(i-k-1)}(0) \quad (1.2)$$

La convolution de Laplace de deux fonctions causales $f(t)$ et $g(t)$ s'exprime par :

$$f(t) * g(t) = \int_0^t f(t-\tau)g(\tau)\,d\tau = g(t) * f(t) \quad (1.3)$$

Généralement, ce produit de convolution est difficile à résoudre, on lui préfère sa transformée de Laplace exprimée par le produit des transformées de Laplace respectives des deux fonctions $f(t)$et $g(t)$:

$$L\{f(t) * g(t)\} = F(s)G(s) \quad (1.4)$$

2.2. La fonction Gamma

La fonction Gamma (Γ) est la généralisation aux nombres réels de la fonction factorielle, définie pour les nombres entiers positifs, elle est donnée par :

$$\Gamma(x) = \int_0^\infty y^{x-1} e^{-y} dy, \quad x > 0 \quad (1.5)$$

Propriétés :

A partir de l'expression (1.5), on déduit que :

$$\Gamma(1) = 1 \quad (1.6)$$

Ainsi que :

$$\Gamma(x+1) = x * \Gamma(x) \quad (1.7)$$

Et pour $x \in \mathbb{N}$, on a : $\Gamma(x+1) = x!$ (1.8)

La fonction Gamma permet également de définir la fonction causale $\Phi_x(t)$ comme suit :

$$\Phi_x(t) = \frac{t^{x-1}}{\Gamma(x)} \qquad x \epsilon \mathbb{R}_+^* \quad (1.9)$$

Cette fonction est utilisée pour donner un sens alternatif aux deux concepts : dérivation/intégration fractionnaire.

2.3. La fonction de Mittag-Leffler

La fonction de Mittag-Leffler à deux paramètres est définie par le développement en série suivant :

$$E_{\gamma,\beta}(v) = \sum_{k=0}^{\infty} \frac{v^k}{\Gamma(\gamma k+\beta)}, \qquad ,\gamma,\beta \in \mathbb{R}_+^* \tag{1.10}$$

Pour $\beta = 1$, on obtient la fonction de Mittag-Leffler dite à un paramètre :

$$E_{\gamma}(v) = \sum_{k=0}^{\infty} \frac{v^k}{\Gamma(\gamma k+1)} \tag{1.11}$$

La fonction exponentielle est déduite de $E_{\gamma}(v)$ en posant $\gamma = 1$:

$$E_1(v) = \sum_{k=0}^{\infty} \frac{v^k}{\Gamma(k+1)} \tag{1.12}$$

Comme $k \in \mathbb{N}$ la fonction Gamma équivaut à la fonction factorielle ($\Gamma(k+1) = k!$), ce qui donne :

$$E_1(v) = \sum_{k=0}^{\infty} \frac{v^k}{k!} \tag{1.13}$$

Cette expression n'est autre que le développement en série de la fonction exponentielle e^v.

3. Opérateurs d'ordre fractionnaires

Le calcul fractionnaire est une généralisation de l'intégration et de la différentiation à l'opérateur fondamental d'ordre non entier ${}_{t_0}D_t^{\alpha}$ ou t_0 et t son des limites de l'opération. L'opérateur intégro-différentiel continu est défini comme :

$${}_{t_0}D_t^{\alpha} = \begin{cases} \frac{d^{\alpha}}{dt^{\alpha}} & \alpha > 0\,, \\ 1 & \alpha = 0, \\ \int_{t_0}^{t} (d\tau)^{-\alpha} & \alpha < 0, \end{cases} \tag{1.14}$$

où $\alpha \in \mathbb{R}$ est l'ordre de l'opération.

4. L'intégration fractionnaire

L'approche de Riemann-Liouville appropriée à la définition de l'intégration fractionnaire s'appuie sur la formule de Cauchy qui calcule i fois l'intégrale répétée d'une fonction causale $f(t)$. On la note $I^i f(t)$ (i est un nombre entier positif) :

$$I^i f(t) = \frac{1}{(i-1)!} \int_0^t (t-\tau)^{i-1} f(\tau) d\tau \qquad i \in \mathbb{N}^* \tag{1.15}$$

On déduit de cette expression que $I^i f(t)$ est nulle à $t = 0$, ainsi que ses dérivées d'ordre 1,2 ,3, . . . , i-1. Par convention on requière également la causalité de la fonction $f(t)$.

La généralisation de la formule de Cauchy à un ordre n réel positif, implique le remplacement de la fonction factorielle par la fonction Gamma, comme suit :

$$I^n f(t) = \frac{1}{\Gamma(n)} \int_0^t (t-\tau)^{n-1} f(\tau) d\tau \qquad n \in \mathbb{R}^* \tag{1.16}$$

Propriétés

- Selon la définition (1.14) l'intégration à un ordre $n = 0$ est l'opérateur identité :

  $$I^0 f(t) = f(t) \tag{1.17}$$

- On vérifie la propriété de semi groupe qui généralise celle de l'intégration usuelle (d'ordre non entier) :

$$I^n I^m = I^{n+m} \qquad n\ et\ m \in \mathbb{R}_+ \tag{1.18}$$

Qui implique la propriété de commutativité :

$$I^n I^m = I^m I^n \tag{1.19}$$

- Une propriété supplémentaire de l'intégrale de Riemann-Liouville apparaît en exploitant la fonction $\Phi_n(t)$, pour obtenir cette relation :

$$\Phi_n(t) * f(t) = \int_0^t \frac{(t-\tau)^{n-1}}{\Gamma(n)} f(\tau) d\tau \tag{1.20}$$

Comme cette expression est équivalente à celle donnée en (1.16), ceci montre que l'intégration fractionnaire de $f(t)$ s'exprime également par la convolution de Laplace des fonctions $f(t)$ et $\Phi_n(t)$ comme suit :

$$I^n f(t) = \Phi_n(t) * f(t) = \frac{1}{\Gamma(n)} \int_0^t (t-\tau)^{n-1} f(\tau) d\tau \tag{1.21}$$

L'expression (1.20) permet également de déduire la transformée de Laplace de l'opérateur d'intégration fractionnaire, sachant que la transformée de Laplace de $\Phi_n(t)$ se calcule au moyen du lemme suivant :

***Lemme* 1 :** *La transformée de Laplace de la fonction* t^n $(n \epsilon \mathbb{R}_+)$ *est donnée par*

$$L\{t^n\} = s^{-1-n}\Gamma(n+1) \tag{1.22}$$

Soit alors :

$$L\{\Phi_n(t)\} = L\left\{\frac{t^{n-1}}{\Gamma(n)}\right\} = s^{-n} \tag{1.23}$$

En effet, la transformée de Laplace de l'opérateur d'intégration fractionnaire s'obtient par :

$$L\{I^n f(t)\} = L\{\Phi_n(t) * f(t)\} = L\{\Phi_n(t)\}L\{f(t)\} = s^{-n}F(s) \tag{1.24}$$

Il existe plusieurs définitions mathématiques pour l'intégration et la dérivation d'ordre fractionnaire. Néanmoins, ces définitions ne mènent pas toujours à des résultats identiques, mais sont équivalentes pour un large panel de fonctions.

4.1. Définition de Riemann-Liouville (R-L)

Définition 1 : Soit $\alpha \in \mathbb{R}_+$ avec $t_0 \in \mathbb{R}$, et f une fonction localement intégrable définie sur $[t_0 + \infty[$. L'intégral d'ordre α de f de borne inferieur t_0 est définie par :

$$_{t_0}^{RL}I_t^{\alpha} f(t) \equiv \frac{1}{\Gamma(\alpha)} . \int_{t_0}^t (t-\tau)^{\alpha-1} f(\tau)\, d\tau \tag{1.25}$$

Avec $t \geq t_0$ et $\Gamma(.)$ est la fonction gamma d'Euler définie précédemment par l'expression (1.5).

Définition 2 : (Oustaloup et al, 2008; Gong et al, 2010)

Soit $\alpha \in \mathbb{R}$ avec $t_0 \in \mathbb{R}$, n un entier positif et f une fonction localement intégrable définie sur $[t_0 \ + \infty[$. La dérivée d'ordre α de f de borne inférieure t_0 est définie par :

$$ {}^{RL}_{t_0}D^{\alpha}_{t} f(t) \equiv \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} . \int_{t_0}^{t} (t-\tau)^{n-\alpha-1} f(\tau)\, d\tau \tag{1.26} $$

où le nombre entier n est choisie de telle manière que : $(n-1) < \alpha < n$.

Cette dérivée d'ordre fractionnaire peut aussi se réécrire comme suit :

$$ {}^{RL}_{t_0}D^{\alpha}_{t} f(t) \equiv \frac{d^n}{dt^n} \left\{ I^{(n-\alpha)} f(t) \right\} \tag{1.27} $$

4.2. Définition de Grünwald-Letnikov (G-L)

La dérivée d'ordre fractionnaire de d'ordre $\alpha > 0$ de G-L est donnée par :

$$ {}^{GL}_{t_0}D^{\alpha}_{t} f(t) \equiv \lim_{h \to 0} \frac{1}{h^{\alpha}} \sum_{k=0}^{\left[\frac{t-t_0}{h}\right]} (-1)^k \binom{\alpha}{k} f(t - k.h) \tag{1.28} $$

Où $[.]$ dénote la partie entière d'un nombre réel, h est la période d'échantillonnage et les coefficients $\binom{\alpha}{k}$ sont donnés par :

$$ \binom{\alpha}{k} = \frac{\Gamma(\alpha+1)}{\Gamma(k+1).\Gamma(\alpha-k+1)} \tag{1.29} $$

La définition de Grünwald-Letnikov de l'intégration d'ordre fractionnaire se traduit par l'expression suivante :

$$ {}^{GL}_{t_0}I^{\alpha}_{t} f(t) \equiv {}^{GL}_{t_0}D^{-\alpha}_{t} f(t) \equiv \lim_{h \to 0} h^{\alpha} \sum_{k=0}^{\left[\frac{t-t_0}{h}\right]} (-1)^k \binom{-\alpha}{k} f(t - k.h) \tag{1.30} $$

Cette définition de Grünwald-Letnikov sera utilisée tout au long de ce mémoire pour la simulation et l'évaluation numériques de systèmes fractionnaires.

4.3. Définition de Caputo

Caputo a introduit une autre formulation de la dérivée d'ordre fractionnaire, exprimée par :

$$ {}^{C}_{t_0}D^{\alpha}_{t} f(t) \triangleq I^{n-\alpha}\, D^{n} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_{t_0}^{t} \frac{f^{(n)}(\tau)}{(t-\tau)^{\alpha-n+1}}\, d\tau \tag{1.31} $$

avec n est un entier positif vérifiant l'inégalité $(n-1) < \alpha < n$.

5. Quelques propriétés de la dérivation non entière

Les principales propriétés des opérateurs d'ordre fractionnaires sont les suivantes :

- Si $f(t)$ est une fonction analytique de t, alors sa dérivée d'ordre fractionnaire $D^{\alpha} f(t)$ est une fonction analytique de t et α.

- Pour $\alpha = n$, ou n est un entier, l'opérateur $D^{\alpha} f(t)$ donne le méme résultat que la différentiation classique d'ordre entier n.
- Pour $\alpha = 0$, l'opération $D^{\alpha} f(t)$ est l'opérateur identité :

$$D^{0} f(t) = f(t) \tag{1.32}$$

- La différentiation et l'intégration d'ordre fractionnaire sont des opérations linéaires, vérifiant cette propriété :

$$D^{\alpha} af(t) + D^{\alpha} bg(t) = aD^{\alpha} f(t) + bD^{\alpha} g(t) \tag{1.33}$$

6. Représentation des systèmes d'ordre fractionnaire

Les systèmes fractionnaires sont en général représentés par des équations différentielles non entières, mais d'autres descriptions peuvent être utilisées, telle que la *représentation diffusive*. Dans notre cas nous considérons un système d'ordre non entier, linéaire à temps continu causal et invariant dans le temps décrit par l'approche classique, et représenté comme dans le cas entier par trois modèles (Oustaloup, 1995).

- Equation différentielle généralisée.
- Fonction de transfert fractionnaire.
- Représentation d'état fractionnaire.

6.1. Equation différentielle généralisée

De manière générale, un système d'ordre non entier mono variable, linéaire à temps invariant est décrit par une équation différentielle généralisée de la forme :

$$y(t) + \sum_{i=1}^{n} a_i D^{\alpha_i} y(t) = \sum_{j=1}^{m} b_j D^{\beta j} u(t) + b_0 u(t) \tag{1.34}$$

Où $u(t) \in \mathbb{R}$ et $y(t) \in \mathbb{R}$ désignent respectivement l'entrée et la sortie du système, a_i, $b_j \in \mathbb{R}$; α_i , $\beta_j \in \mathbb{R}^{+}$.

Lorsque les ordres de dérivation α_i, β_j sont tous multiples d'un même nombre réel α tel que $\alpha_i = i\,\alpha$; β_j =j α, le système non entier est dit d'ordre commensurable.

6.2. Fonction de transfert d'ordre non entier

L'application de la transformée de Laplace à l'équation (1.34) permet de déduire la fonction de transfert :

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0 + \sum_{j=1}^{m} b_j s^{\beta j}}{1 + \sum_{i=1}^{n} a_i s^{\alpha_i}} \tag{1.35}$$

Dans le cas d'un système commensurable, cette fonction de transfert s'écrit :

$$G(s) = \frac{b_0+\sum_{j=1}^{m} b_j s^{j\alpha}}{1+\sum_{i=1}^{n} a_i s^{i\alpha}} \tag{1.36}$$

Par contre, dans le cas général des systèmes non entiers multivariables, ayant l entrées et q sorties, ils sont décris par un système d'équations différentielles d'ordre non entier, dont la matrice de fonction de transfert a pour expression :

$$G(s) = \begin{pmatrix} G_{11}(s) & \cdots & G_{1l}(s) \\ \vdots & \ddots & \vdots \\ G_{q1}(s) & \cdots & G_{ql}(s) \end{pmatrix} \tag{1.37}$$

où chaque $G_{ij}(s)$ est une fonction de transfert de la forme (1.35)

6.3. Représentation d'état des systèmes fractionnaires

Le modèle d'état d'un système d'ordre fractionnaire est défini comme dans le cas entier par deux équations :

- Une équation d'état : dans laquelle chaque variable d'état $x_i(t)$est dérivée à un ordre non entier α_i. Dans ce cas on parle de la représentation d'état généralisée. Dans le cas des systèmes commensurables ; tous les états $x_i(t)$ sont dérivés à un même ordre non entier α.
- Une équation de sortie : qui est une combinaison linéaire des états, comme dans le cas entier.

Il en résulte un modèle d'état, écrit sous la forme compacte suivante :

$$\begin{cases} D^{(\alpha)}(x) = Ax + Bu \\ y = Cx + Du \end{cases} \tag{1.38}$$

où: $$D^{(\alpha)}(x) = [D^{\alpha_1}x_1, D^{\alpha_2}x_2 \ldots \ D^{\alpha_n}x_n]^T \tag{1.39}$$

avec : $x \in \mathbb{R}^n, u \in \mathbb{R}^l, y \in \mathbb{R}^q, A \in \mathbb{R}^{n\times n}, \in \mathbb{R}^{n\times l}, C \in \mathbb{R}^{q\times n}, D \in \mathbb{R}^{q\times l}$.

Dans le cas d'un système commensurable le modèle d'état (1.38) peut être reformulé par l'expression suivante :

$$\begin{cases} D^{\alpha}(x) = Ax + Bu \\ y = Cx + Du \end{cases} \tag{1.40}$$

Avec :

$$D^{\alpha}(x) = D^{\alpha}[x_1, x_2 \ \ldots \ x_n]^T \tag{1.41}$$

7. Réalisation du modèle d'état fractionnaire

7.1. Cas des systèmes commensurables

Etant donné un système non entier mono variable linéaire invariant représenté par sa fonction de transfert $G(s)$, donnée sous la forme :

$$G(s) = \frac{b_m s^{m\alpha} + b_{m-1} s^{(m-1)\alpha} + \ldots + b_1 s^{\alpha} + b_0}{s^{n\alpha} + a_{n-1} s^{(n-1)\alpha} + \ldots + a_1 s^{\alpha} + a_0} \quad (1.42)$$

Pour calculer le modèle d'état qui lui correspond, on procède en trois étapes :

Etape 1 : grâce au changement de variable $p = s^{\alpha}$, le modèle non entier $G(s)$ peut être remplacé par un modèle entier $G(p)$, qui s'écrit sous la forme :

$$G(p) = \frac{b_m p^{m} + b_{m-1} p^{(m-1)} + \ldots + b_1 p + b_0}{p^{n} + a_{n-1} p^{(n-1)} + \ldots + a_1 p + a_0} \quad (1.43)$$

Etape 2 : calculer le modèle d'état entier correspondant à $G(p)$. On peut obtenir toutes les formes particulières utilisées dans la théorie des systèmes d'ordre entier (forme commandable, observable, Jordan …). On obtient, le modèle de la forme :

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \quad (1.44)$$

Etape 3 : remplacer dans le modèle d'état (1.44) la dérivée entière d'ordre un par la dérivée non entière d'ordre α, ainsi le modèle d'état correspondant à la fonction de transfert (1.43) obtenu est de la forme (Djamah, 2009) :

$$\begin{cases} D^{\alpha} x = Ax + Bu \\ y = Cx + Du \end{cases} \quad (1.45)$$

La méthode reste applicable dans le cas des systèmes commensurables multivariables.

7.2. Cas des systèmes d'ordre non entier généralisés

- ***Cas où $G(s)$ a un numérateur constant* :**

La fonction de transfert $G(s)$ s'écrit sous la forme :

$$G(s) = \frac{Y(s)}{U(s)} = \frac{Z(s)}{U(s)} * \frac{Y(s)}{Z(s)} = \frac{1}{s^{\alpha_n} + a_{n-1} s^{\alpha_{n-1}} + \ldots + a_1 s^{\alpha_1} + a_0} * b_0 \quad (1.46)$$

On suppose, sans perte de généralité, que $\alpha_n > \alpha_{n-1} > \cdots > \alpha_2 > \alpha_1$. L'équation différentielle associée à $\frac{Z(s)}{U(s)}$ est donnée par : (la variable t est omise pour ne pas surcharger les expressions).

$$D^{\alpha_n} z = -a_{n-1} D^{\alpha_{n-1}} z - \cdots - a_1 D^{\alpha_1} z - a_0 z + u \quad (1.47)$$

Faisons le changement de variable pour obtenir le vecteur d'état :

$$\begin{cases} x_1 = z \\ x_2 = D^{\alpha_1} x_1 = D^{\alpha_1} z \\ x_3 = D^{\alpha_2-\alpha_1} x_2 = D^{\alpha_2-\alpha_1}(D^{\alpha_1} z) = D^{\alpha_2} z \\ \quad . \\ \quad . \\ \quad . \\ x_i = D^{\alpha_{i-1}-\alpha_{i-2}} x_{i-1} = D^{\alpha_{i-1}-\alpha_{i-2}}(D^{\alpha_{i-2}-\alpha_{i-3}} x_{i-2}) = \cdots = D^{\alpha_{i-1}} z \\ \quad . \\ \quad . \\ \quad . \\ x_n = D^{\alpha_{n-1}-\alpha_{n-2}} x_{n-1} = D^{\alpha_{n-1}} z \end{cases} \tag{1.48}$$

$$D^{(\alpha)}(x) = \begin{cases} D^{\alpha_1} x_1 = x_2 \\ D^{\alpha_2-\alpha_1} x_2 = x_3 \\ D^{\alpha_3-\alpha_2} x_3 = x_4 \\ \quad . \\ \quad . \\ \quad . \\ D^{\alpha_i-\alpha_{i-1}} x_i = x_{i+1} \\ \quad . \\ \quad . \\ \quad . \\ D^{\alpha_n-\alpha_{n-1}} x_n = D^{\alpha_n-\alpha_{n-1}}(D^{\alpha_{n-1}-\alpha_{n-2}} x_{n-1}) = \ldots = D^{\alpha_n} z \end{cases} \tag{1.49}$$

La dernière composante de la dérivée du vecteur d'état $D^{(\alpha)}(x)$ $(D^{(\alpha_n)} z)$ s'écrit en fonction des autres dérivées de $z(t)$ (équation (1.47)), et s'exprime comme suit :

$$D^{\alpha_n-\alpha_{n-1}} x_n = -a_0 x_1 - a_1 x_2 - \cdots - a_{n-2} x_{n-1} - a_{n-1} x_n + u \tag{1.50}$$

Le modèle d'état correspondant au modèle fonction de transfert (1.46) est finalement donnée par :

$$\begin{cases} D^{(\alpha)}(x) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & & 0 & 0 \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u \\ y = \begin{bmatrix} b_0 & 0 & 0 & \ldots & 0 & 0 \end{bmatrix} x \end{cases} \tag{1.51}$$

Avec

$$\begin{cases} x = [x_1, x_2 \ldots x_n]^T \\ D^{(\alpha)}(x) = \left[D^{\alpha_1} x_1, D^{(\alpha_2-\alpha_1)} x_2, \cdots D^{(\alpha_n-\alpha_{n-1})} x_n\right]^T \end{cases} \tag{1.52}$$

De la même manière, on peut obtenir une forme similaire à la forme canonique observable des systèmes entiers.

- ***Cas où le numérateur de $G(s)$ a un polynôme :***

La fonction de transfert $G(s)$, supposée propre s'écrit dans ce cas sous la forme :

$$G(s) = \frac{b_m s^{\beta m} + b_{m-1} s^{\beta m-1} + \ldots + b_1 s^{\beta 1} + b_0}{s^{\alpha n} + a_{n-1} s^{\alpha n-1} + \ldots + a_1 s^{\alpha 1} + a_0} \quad (1.53)$$

Supposant que $\alpha_n > \alpha_{n-1} > \cdots > \alpha_2 > \alpha_1$ et $\beta_m > \beta_{m-1} > \cdots > \beta_2 > \beta_1$.

La méthode générale permettant de calculer la représentation d'état de la fonction de transfert $G(s)$ (1.53) est présentée dans ce qui suit :

Soit $\tilde{\alpha}$ le vecteur constitué de la concaténation des ordres non entiers α_i et β_i de $G(s)$:

$$\tilde{\alpha} = [\tilde{\alpha}_{n+m} \ \tilde{\alpha}_{n+m-1} \ \tilde{\alpha}_{n+m-2} \ \ldots \ \tilde{\alpha}_3 \ \ \tilde{\alpha}_2 \ \ \tilde{\alpha}_1 \ ] \quad (1.54)$$

Tel que $\tilde{\alpha}_{n+m} > \tilde{\alpha}_{n+m-1} > \cdots > \tilde{\alpha}_2 > \tilde{\alpha}_1$.

Les ordres $\tilde{\alpha}_i$ étant quelconques, par analogie à la section précédente pour le choix des variables d'état, nous obtenant le modèle d'état suivant :

$$\begin{cases} D^{(\tilde{\alpha})}(x) = \begin{bmatrix} 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & & 0 & 0 \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & & 0 & 1 \\ -\tilde{a}_1 & -\tilde{a}_2 & -\tilde{a}_3 & \ldots & -\tilde{a}_{n+m-1} & -\tilde{a}_{n+m} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u \\ y = [\ \tilde{c}_1 \quad \tilde{c}_2 \quad \tilde{c}_3 \quad \ldots \quad \tilde{c}_{n+m-1} \quad \tilde{c}_{n+m} \ ] x \end{cases} \quad (1.55)$$

Avec :

$$D^{(\tilde{\alpha})}(x) = \left[D^{\tilde{\alpha}_1} x_1, D^{(\tilde{\alpha}_2 - \tilde{\alpha}_1)} x_2, \cdots \ D^{(\tilde{\alpha}_{n+m} - \tilde{\alpha}_{n+m-1})} x_{n+m}\right]^T \quad (1.56)$$

La fonction de transfert $G(s)$ contient $(m+1)$ termes au numérateur et $(n+1)$ termes au dénominateur. En tenant compte que le nombre de variables d'état du système non entier est égal à la somme de la dimension du polynôme du numérateur et celle du polynôme dénominateur, nous obtenons donc un vecteur d'état augmenté de dimension $(n+m)$, contrairement aux systèmes entiers, où le nombre de variables de leur modèle d'état est fixé par la dimension du polynôme du dénominateur de sa fonction de transfert.

Les éléments de la matrice A non nuls sont les coefficients correspondant aux ordres du dénominateur et les éléments non nuls du vecteur C sont ceux du numérateur de $G(s)$. Il suffit donc de les trier de sorte à faire ressortir m termes pour lesquels les ordres non entiers correspondent à ceux des numérateurs de $G(s)$et n termes pour lesquels les ordres non entiers correspondent à ceux du dénominateur de $G(s)$. La procédure de sélection des éléments $\tilde{a}_i$ et $\tilde{c}_i$ s'effectue comme suit :

$$\begin{cases} \tilde{a}_{n+m} = a_n \qquad\qquad\qquad\qquad \tilde{c}_1 = b_0 \\ si \quad \tilde{\alpha}_i = \beta_j \quad alors \quad \tilde{c}_{i+1} = b_j \quad et \ \tilde{a}_{n+m-i} = 0 \quad i = 1, \ldots n+m-1 \\ si \quad \tilde{\alpha}_i = \alpha_j \quad alors \quad \tilde{c}_{i+1} = 0 \quad et \ \tilde{a}_{n+m-i} = a_{n-j} \quad i = 1, \ldots n+m-1 \end{cases} \quad (1.57)$$

8. Passage du modèle d'état au modèle fonction de transfert

Le passage du modèle d'état fractionnaire au modèle fonction de transfert se fait de façon analogue au cas entier en utilisant la formule suivante :

$$G(s) = C\left[\left(s^{(\alpha)}I_n - A\right)^{-1}\right]B + D \tag{1.58}$$

Avec $s^{(\alpha)}I_n = diag[s^{\alpha_1}, s^{\alpha_2}, \dots, s^{\alpha_n}]$

Dans le cas particulier d'une forme canonique commandable, la fonction de transfert obtenue s'écrit sous la forme :

$$G(s) = \frac{c_1 s^{\beta_1} + c_2 s^{\beta_2} + \dots + c_n s^{\beta_n}}{a_1 s^{\tilde{\alpha}_1} + a_2 s^{\tilde{\alpha}_2} + \dots + a_n s^{\tilde{\alpha}_n} + s^{\tilde{\alpha}_{n+1}}} \tag{1.59}$$

Avec

$$\beta_1 = 0, \beta_i = \sum_{j=1}^{i-1} \alpha_j \quad pour \quad i = 2, \dots, n$$

et

$$\tilde{\alpha}_1 = 0, \quad \tilde{\alpha}_i = \sum_{j=1}^{i-1} \alpha_j \quad pour \ i = 2, \dots, n+1$$

Comme dans le cas entier, les éléments des matrices C et A constituent respectivement les coefficients du numérateur et du dénominateur de la fonction de transfert correspondante. Les ordres de dérivation du numérateur et du dénominateur sont des combinaisons linéaires des ordres de dérivation des vecteurs d'état.

$$G(s) = \frac{c_1 + c_2 s^{\beta_2} + \dots + c_n s^{\beta_n}}{a_1 + a_2 s^{\tilde{\alpha}_2} + \dots + a_n s^{\tilde{\alpha}_n} + s^{\tilde{\alpha}_{n+1}}} \tag{1.60}$$

L'équation caractéristique du système, correspond comme dans le cas entier au polynôme du dénominateur de la fonction de transfert et elle est donnée par :

$$S^{(\alpha)}I_n - A = 0 \tag{1.61}$$

9. Stabilité des systèmes factionnaires

Dans la théorie de la stabilité des systèmes linéaires à temps invariant et à dérivée d'ordre entier, nous savons bien qu'un système est stable si les racines du polynôme caractéristique sont à parties réelles strictement négatives, donc situées dans la moitié gauche du plan complexe. Par ailleurs, dans le cas des systèmes fractionnaires linéaires à temps invariant, la définition de la stabilité est différente de celle des systèmes d'ordre entier. En effet, les systèmes fractionnaires ou d'ordre non entier peuvent bel et bien avoir des racines dans la moitié droite du plan complexe et être stables (N'Doye, 2011).

La stabilité des systèmes fractionnaires commensurables a été étudiée dans (Matignon, 1996-b ; Matignon, 1996-a), où des conditions nécessaires et suffisantes ont été obtenues donnant lieu au théorème suivant :

Théorème: (Matignon, 1996-b ; Sabatier et al, 2008) considérons le système linéaire fractionnaire d'ordre commensurable suivant :

$$\begin{cases} D^{\alpha}x(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \\ x(0) = x_0 \end{cases} \qquad 0 < \alpha < 2 \qquad (1.62)$$

$x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, y(t) \in \mathbb{R}^p$. Soit $\sigma(A)=\{\lambda_1, \dots, \lambda_n\}$, le système (1.62) avec comme entrée $u(t) = 0$ est stable si et seulement si

$$|\arg(\lambda_i)| > \frac{\alpha\pi}{2}, \qquad \lambda_i \in \sigma(A), \qquad i = 1 \dots n \qquad (1.63)$$

D'après ce théorème sur la stabilité, il en découle les différentes régions de stabilité illustrées par les Figures 1.1 et 1.2.

Si $1 < \alpha < 2$, alors la relation (1.63) décrit une région convexe du plan complexe comme le montre la Figure (1.2).

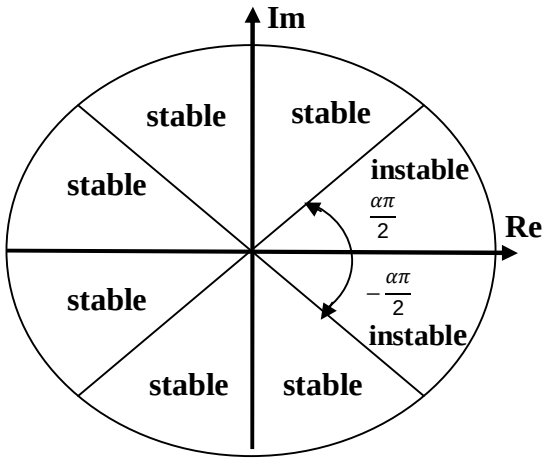


Figure 1.1 : Région de stabilité des systèmes d'ordre fractionnaires avec $0 < \alpha < 1$

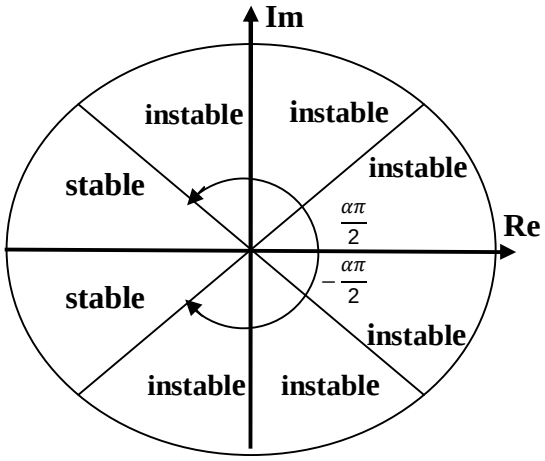


Figure 1.2 : Région de stabilité des systèmes d'ordre fractionnaires avec $1 < \alpha < 2$.

10. Commandabilité et observabilité des systèmes fractionnaires :

Afin d'étudier la commandabilité, l'observabilité, nous considérons le système linéaire fractionnaire décrit par l'équation suivante :

$$\begin{cases} D^{\alpha}x(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \\ x(0) = x_0 \end{cases} \qquad 0 < \alpha < 2 \qquad (1.64)$$

$x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, y(t) \in \mathbb{R}^p$.

Pour vérifier si le système linéaire (1.64) est commandable ou observable, il existe deux critères dits de Kalman et d'Hautus.

Définition : (Matignon, 1996-b ; Matignon et Andréa-Novel, 1996; Matignon, et Andréa-Novel, 1997 ; Vinagre et al, 2002).

Le système fractionnaire (1.64) d'ordre $0 < \alpha < 2$, est commandable (ou la paire (A, B) est commandable) si et seulement si l'une des deux conditions équivalentes suivantes est vérifiée.

- *Critère de Kalman*

$$rang[A \;\; AB \;\; \ldots \;\; A^{n-1}B] = n = \dim(x) \qquad (1.65)$$

- *Critère d'Hautus*

$$rang([\sigma I_n - A \quad B]) = n = \dim(x) \qquad \forall \sigma \epsilon \mathbb{C} \qquad (1.66)$$

Définition : (Matignon, 1996-b ; Matignon et Andréa-Novel, 1996; Matignon et Andréa-Novel, 1997 ; Vinagre, et al, 2002).

Le système fractionnaire (1.64) d'ordre $0 < \alpha < 2$, est observable (ou la paire (A, C) est observable) si et seulement si l'une des deux conditions équivalentes suivantes est vérifiée.

- *Critère de Kalman*

$$rang\left(\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}\right) = n = \dim(x) \tag{1.67}$$

- *Critère d'Hautus*

$$rang\left(\begin{bmatrix} \sigma I_n - A \\ C \end{bmatrix}\right) = n = \dim(x) \qquad \forall \sigma \epsilon \mathbb{C} \tag{1.68}$$

11. Méthodes d'approximation du dérivateur généralisé

Les systèmes d'ordre fractionnaire sont à mémoire illimitée (dimension infinie). Par souci de réalisabilité physique et parce que les outils de calcul et de simulation ne traitent pas (encore) directement de la dérivation/intégration d'ordre non entier, l'implémentation d'un système d'ordre fractionnaire requière, au préalable, son approximation par une fonction de transfert d'ordre entier, donc de dimension finie. Durant les vingt dernières années plusieurs algorithmes numériques ont été développés en utilisant les modèles rationnels continus et discrets dans le temps rapprochant ainsi les systèmes fractionnaires (Vinagre et al, 2000 ; Hamdaoui, 2006).

Le dérivateur généralisé constitue l'élément principal pour la synthèse des systèmes fractionnaires, c'est pourquoi l'approximation de ces derniers, passe nécessairement par celle du dérivateur généralisé (Aït messaoud, 2007). Plusieurs méthodes d'approximation ont été proposées, parmi elles on peut citer, la méthode d'Oustaloup, de Charef, de Carlson, et la méthode de Matsuda, etc. (Charef, 1992; Vinagre et al, 2000).

Pour les besoins de ce mémoire, on ne présentera qu'une seule méthode d'approximation, qui est la méthode d'Oustaloup.

11.1. Méthode d'approximation d'Oustaloup (1991)

L'approximation d'Oustaloup d'un dérivateur généralisé, dont l'action différentielle couvre tout l'espace des fréquences, repose sur une distribution récursive d'une infinité de zéros et de pôles réels négatifs. Dans le cadre d'une synthèse réaliste (pratique) fondée sur un nombre fini de zéros et de pôles, il convient de réduire le comportement différentiel généralisé sur un intervalle fréquentiel borné, choisi selon les besoins de l'application. Le transfert résultant dit "dérivateur généralisé borné en fréquences" est ensuite approximé par une

fonction de transfert d'ordre entier. (Oustaloup, 1991 ; Oustaloup, 1995; Oustaloup, 1999; Benchellal, 2008).

La fonction de transfert du dérivateur généralisé est donnée par :

$$D(s) = \left(\frac{s}{w_c}\right)^n, \qquad n \in \mathbb{R}. \tag{1.69}$$

w_c : La fréquence de coupure,

n : L'ordre de dérivation non entier.

Une troncature à la fois du coté des basses et des hautes fréquences consiste à limiter sur l'intervalle fréquentiel $[w_A \;\; w_B]$, centré géométriquement sur w_c, le comportement différentiel du transfert $D(s)$. En fait, la troncature sera réellement effectuée, pour plus de précision, sur un intervalle de fréquence plus large $[w_b \;\; w_h]$, tel que :

$w_b \ll w_A$ et $w_h \gg w_B$

On obtient alors la fonction de transfert fractionnaire qui est bornée en fréquence,

$$D_b(s) = \left(c_0 \frac{1+s/w_b}{1+s/w_h}\right)^n \tag{1.70}$$

pour assurer un gain unitaire à la fréquence w_c on pose :

$$c_0 = \frac{w_b}{w_c} = \frac{w_c}{w_h} \tag{1.71}$$

Le dérivateur borné en fréquence, étant lui aussi un transfert fractionnaire, qui doit être approximé par une fonction de transfert d'ordre entier, construite au moyen de zéros et de pôles distribués récursivement :

$$D_b(s) = \lim_{N\to\infty} D_N(s) \tag{1.72}$$

Avec

$$D_N(s) = \left(\frac{w_c}{w_h}\right)^n \prod_{i=-N}^{N} \frac{\left(1+\frac{s}{z_i}\right)}{\left(1+\frac{s}{p_i}\right)} \tag{1.73}$$

$p_i \; et \; z_i$: sont une paire de pôle et de zéro au nombre total de 2N+1, distribuées récursivement comme suit :

$$\begin{cases} \frac{p_i}{z_i} = \alpha > 0, \; \frac{z_{i+1}}{p_i} = \eta > 0 \\ \frac{z_{i+1}}{z_i} = \frac{p_{i+1}}{p_i} = \alpha\eta > 1 \end{cases} \tag{1.74}$$

Comme le montre la Figure 1.3 l'approximation est réalisée par un lissage des diagrammes asymptotiques de Bode de $D_N(s)$ pour conduire à ceux de $D_b(s)$.

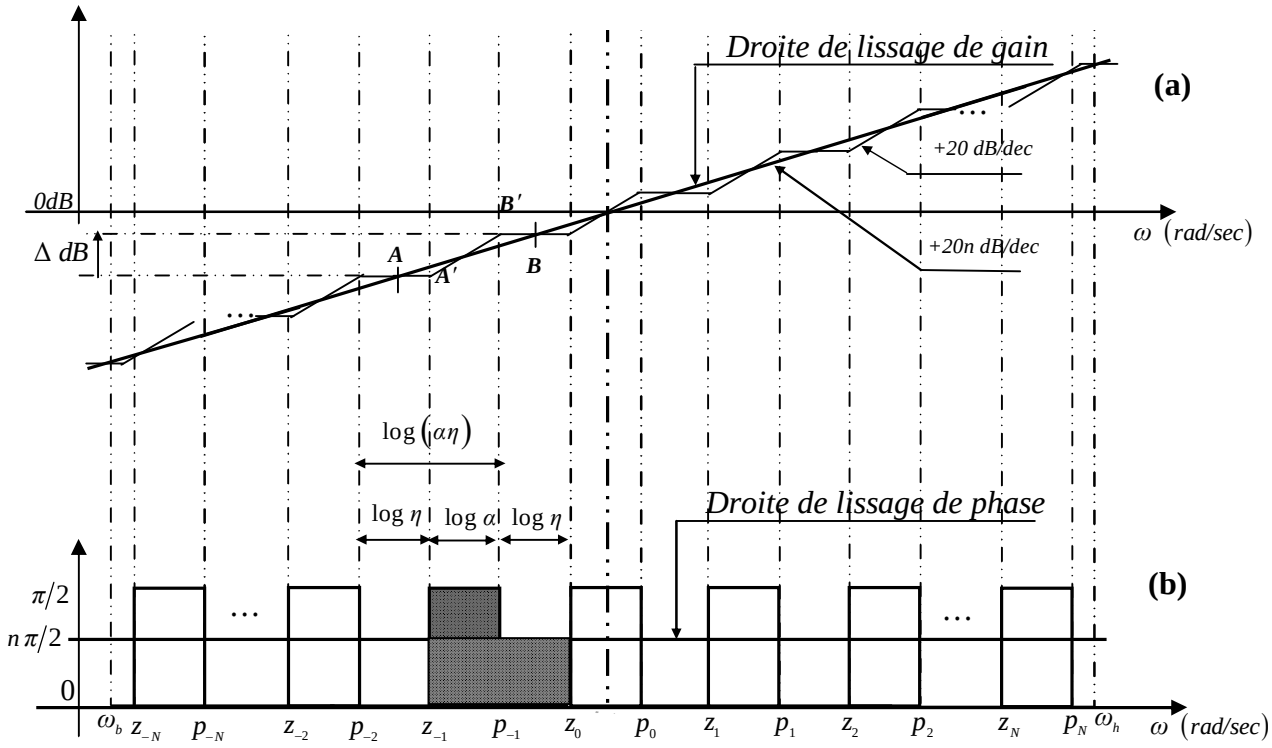


Figure 1.3 : Diagrammes asymptotique de Bode de $D_b(s)$. et de $D_N(s)$ pour $n\epsilon]0,1[$.

Exemple illustratif

Soit un dérivateur généralisé fractionnaire d'ordre 0.5, qui est donné par la fonction de transfert suivante :

$$D(s) = \left(\frac{s}{1}\right)^{0.5} \tag{1.75}$$

Si on choisi $w_b=10^{-2}$, $w_h=10^2$ et un nombre de cellules =10,

L'approximation de (1.75) par la méthode d'approximation d'Oustaloup nous donne :

$$D_N = \frac{\boldsymbol{num}}{\boldsymbol{den}} \tag{1.76}$$

Avec

$$\boldsymbol{num} = 10s^{10} + 832.6\,s^9 + 1.974\,10^4\,s^8 + 1.672\,10^5\,s^7 + 5.416\,10^5\,s^6 6.859\,10^5 s^5 + 3.417\,10^5 s^4 + 6.657\,10^4 s^3 + 4958\,s^2 + 132\,s^1 + 1$$

$$\boldsymbol{den} = s^{10} + 132\,s^9 + 4958\,s^8 + 6.657\,10^4 s^7 + 3.417\,10^5\,s^6 + 6.859\,10^5 s^5 + 5.41610^5\,s^4 + 1.672\,10^5\,s^3 + 1.974\,10^4\,s^2 832.6 + s^1 + 10$$

On constate que l'ordre du dérivateur approximé par la méthode d'Oustaloup, obtient une fonction de transfert d'ordre entier, mais de grande de dimension.

La Figure 1.4 représente, les diagrammes de Bode du dérivateur généralisé $D(s)$ (en trait rouge), et ceux du dérivateur approximé D_N (en trait bleu). Les deux diagrammes se superposent sur un intervalle fréquentiel centré sur $w_c = 1$, beaucoup plus large pour

l'amplitude que pour la phase. Toutefois, l'écart entre les diagrammes respectifs d'amplitude et de phase s'accentue en dehors de l'intervalle fréquentiel d'approximation.

On constate aussi que les courbes d'amplitude et de phase du dérivateur approximé sont respectivement de pente de $0.5 \times 20\ db/dec$ et de phase de 0.5×90.

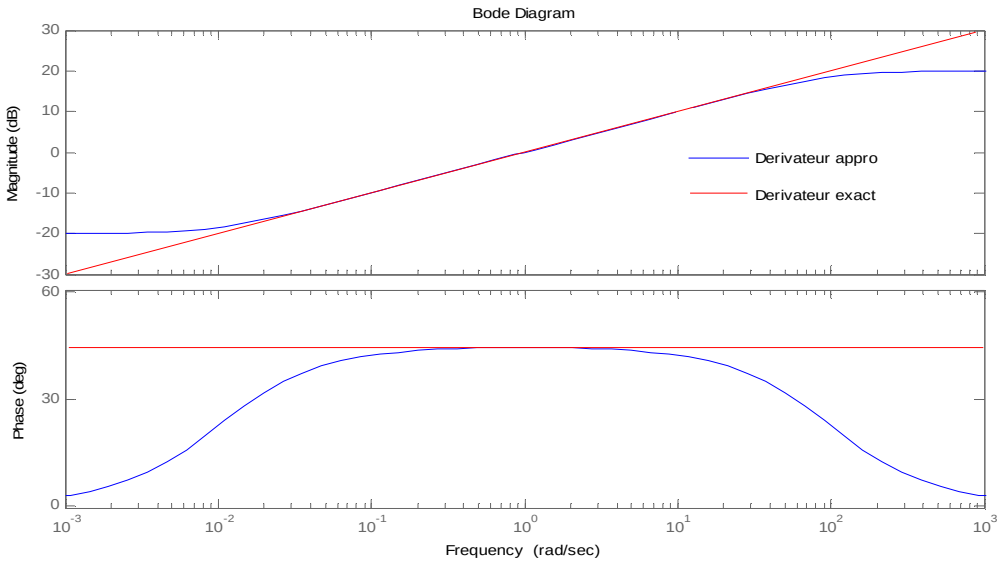


Figure 1.4 : Diagramme de Bode de D(s) et de $D_N(S)$.

Conclusion :

Dans ce chapitre, nous avons présenté un état de l'art sur les systèmes à dérivée non entière. En effet, on a introduit la théorie de la dérivation non entière à partir de quelques rappels sur les fonctions de Gamma d'Euler et Mittag-Leffler et sur les différentes définitions et propriétés de la dérivée fractionnaire. Ensuite, on a abordé les différentes représentations des systèmes fractionnaires, comme l'équation différentielle généralisée, le modèle fonction de transfert non entier ainsi que sa représentation d'état. Le critère de stabilité ainsi que les conditions de commandabilité et d'observabilité ont été traité dans ce chapitre. Enfin on a clôturé ce chapitre par une méthode d'approximation du dérivateur généralisé.

Bien entendu, les systèmes fractionnaires ne bénéficient pas encore d'un arsenal théorique, à l'image des systèmes d'ordre entier. On souligne aussi la lourdeur des expressions analytiques, car l'algorithme de calcul de la dérivée généralisée nécessite la prise en compte, à chaque itération, de tout le passé de la fonction. Cela met en évidence la propriété de dimension infinie des systèmes fractionnaires et pose ainsi le problème, toujours soulevé, de leur simulation et de leur identification, puisque la précision de la réponse dépend fortement de la capacité de mémorisation (limitée) du calculateur.

Une autre manière de contourner cette difficulté consiste à approximer les systèmes fractionnaires par des modèles neuronaux ayant le même comportement, et c'est justement le propos du chapitre suivant.

Chapitre 2

Réseaux de neurones multicouches et Réseaux de neurones à espace d'état

1. Introduction

Le terme réseaux de neurones regroupe une vaste catégorie de modèles d'apprentissage ayant tous en commun le fait de s'inspirer du fonctionnement du système nerveux : tous ces modèles mettent en jeu des unités appelées neurones dont les interactions sont gouvernées par des poids les reliant (Amoura, 2010) les synapses. L'apprentissage d'un réseau de neurones consistant donc à faire varier ces poids jusqu'à obtenir les valeurs désirées en sortie.

De part leur robustesse, leur flexibilité et leur grande capacité de généralisation, les réseaux de neurones ont su s'imposer comme l'un des modèles majeurs en apprentissage machine. Afin de donner au lecteur une bonne appréciation de ce que sont les réseaux de neurones, nous avons décidé de parler de divers modèles. Nous verrons tout d'abord les réseaux à propagation avant à travers le perceptron multicouches (MLP). Par la suite, nous présenterons deux types de réseaux de neurones récurrents, à savoir le réseau de neurones MLP bouclé d'entrée et le réseau de neurones à espace d'état (SSNN), qui occupent une place centrale dans ce mémoire.

L'objectif de ce chapitre est la présentation de deux architectures neuronales utilisées dans l'identification et la modélisation des modèles dynamiques. La première architecture est appelée « Réseau de neurones multicouches (MLP) bouclé d'entrée », elle consiste à mettre en œuvre un réseau de neurones à partir de plusieurs couples d'entrées/sorties. Quant à la seconde, appelée « Réseau de neurones à espace d'état» utilise la représentation d'état des systèmes pour reconstituer le comportement des états et la sortie du système. Cette méthode utilise deux réseaux de neurones, dont le premier est entrainé à partir

du signal d'entrée et des états du système pour reproduire l'évolution des états du système. Ces derniers sont ensuite utilisés pour entrainer un autre réseau de neurones, afin de reproduire le comportement de la sortie du système.

2. Le neurone biologique

Le neurone biologique est une cellule vivante spécialisée dans le traitement des signaux électriques.

Les neurones sont reliés entre eux par des liaisons appelées axones. Ces axones vont eux même jouer un rôle important dans le comportement logique de l'ensemble. Ces axones conduisent les signaux électriques de la sortie d'un neurone vers l'entrée (synapse) d'un autre neurone.

Les neurones font une sommation des signaux reçus en entrée et en fonction du résultat obtenu, vont fournir un courant en sortie.

La structure d'un neurone se compose en trois parties (Voir la Figure 2.1) :

- *Le noyau* : ou cellule d'activation nerveuse, au centre du neurone.
- *L'axone* : attaché au noyau qui est électriquement actif, ce dernier conduit l'impulsion générée par le neurone.
- *Dendrites* : électriquement passives, elles reçoivent les impulsions d'autres neurones.

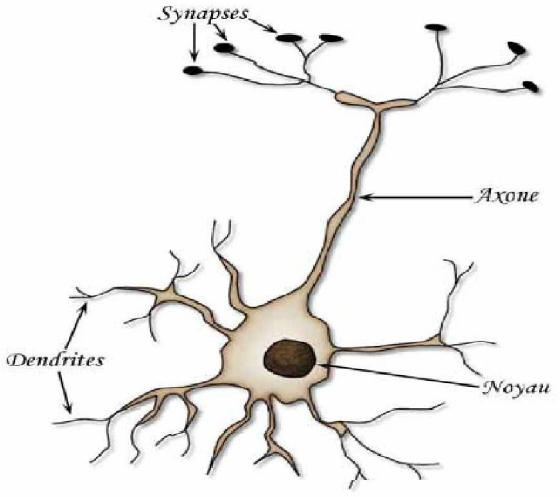


Figure 2.1: Le neurone biologique

3. Le neurone formel (artificiel)

Le neurone artificiel (ou cellule) et un processus élémentaire. Il reçoit un nombre variable d'entrées en provenance de neurones appartenant à un niveau situé en amont (on parlera de neurones ‘’amonts’’). A chacune des entrées est associé un poids *W* représentatif de

la force de connexion. Chaque processus élémentaire est doté d'une sortie unique, qui se ramifie ensuite pour alimenter un nombre variable de neurones appartenant à un niveau situé en aval (on parlera de neurones ''avals''). A chaque connexion est associé un poids. (Figure2.2)

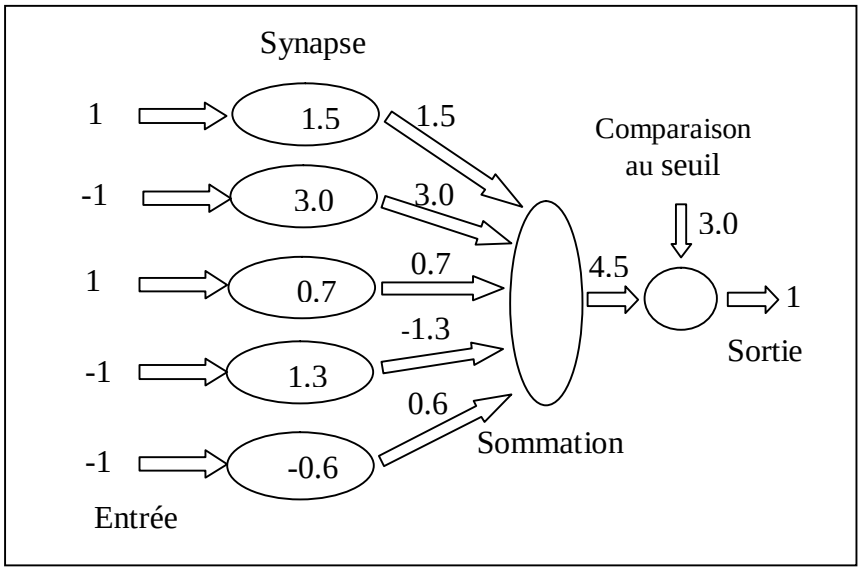


Figure 2.2 : Le neurone formel.

4. Modélisation d'un neurone formel

Les réseaux de neurones formels sont à l'origine d'une tentative de modélisation mathématique du cerveau humain. Les premiers travaux datent de 1943 et sont l'œuvre de MM. Mac Culloch et Pitts. Ils représentent un modèle assez simple pour les neurones et explorent les possibilités de ce modèle.

La modélisation consiste à mettre en œuvre un système de réseau neuronal sous un aspect non pas biologique mais artificiel, cela suppose d'après le principe biologique, on aura une correspondance pour chaque élément composant le neurone biologique, donc une modélisation pour chacun d'entre eux.

On pourra résumer cette modélisation par le Tableau (2.1), qui nous permettra de voir clairement la transition entre le neurone biologique et le neurone formel.

Neurone biologique	**Neurone artificiel**
synapses	Poids de connexions
Axones	Signal de sortie
Dendrites	Signal d'entrée
Noyau	Fonction d'activation

Tableau 2.1 : Analogie entre le neurone biologique et le neurone formel.

4.1. Les entrées

Elles peuvent être :

- Booléennes.
- Binaires (0,1) ou bipolaires (-1,1).
- Réelles.

4.2. Fonction d'activation

Cette fonction permet de définir l'état interne du neurone en fonction de son entrée totale, citons à titre d'exemple quelques fonctions souvent utilisées :

a. Fonction binaire à seuil

Fonction Heaviside définie par la fonction (2.1), qui est illustrée par la Figure 2.3

$$h(x) = \begin{cases} 1 & si\ x \geq 0 \\ 0 & sinon \end{cases} \quad (2.1)$$

Fonction signe définie par la fonction (2.2) et qui est illustrée par la Figure 2.4

$$sgn(x) = \begin{cases} +1 & si\ x \geq 0 \\ -1 & sinon \end{cases} \quad (2.2)$$

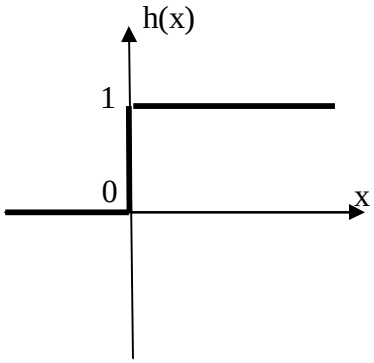


Figure 2.3 : Fonction de Heaviside

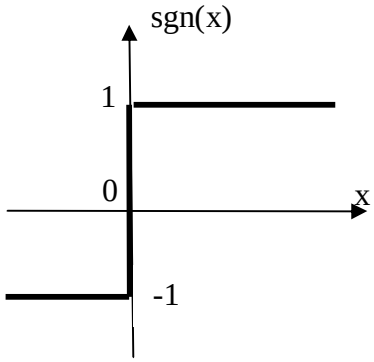


Figure 2.4 : Fonction signe

Le seuil introduit une non-linéarité dans le comportement du neurone, cependant il limite la gamme des réponses possibles à deux valeurs.

b. Fonction linéaire

C'est l'une des fonctions d'activations les plus simples, sa fonction est définie par :

$$F(x) = x \quad (2.3)$$

Elle est représentée par la Figure 2.5

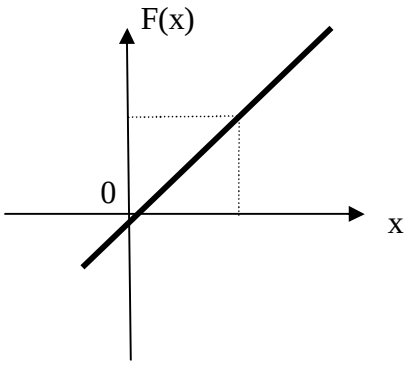


Figure 2.5 : Fonction linéaire.

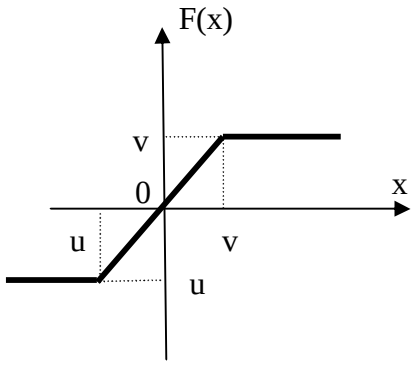


Figure 2.6 : Fonction linéaire à seuil.

c. Fonction linéaire à seuil ou multi-seuils

On peut la définir comme suit :

$$F(x) = \begin{cases} x & x \in [u\ v] \\ v & si\ x \geq v \\ u & si\ x \leq u \end{cases} \tag{2.4}$$

Cette fonction représente un compromis entre la fonction linéaire et la fonction seuil : entre ses deux barres de saturations, elle confère au neurone une gamme de réponses possibles. En modulant la pente de la linéarité, on affecte la plage de neurone. (Figure 2.6)

d. Fonction sigmoïde

Elle est l'équivalent continu de la fonction linéaire. Etant continue, elle est dérivable, d'autant plus que sa dérivée est simple à calculer, elle est définie par la fonction (2.5) :

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.5}$$

Le tracé de cette fonction est donné par la Figure 2.7

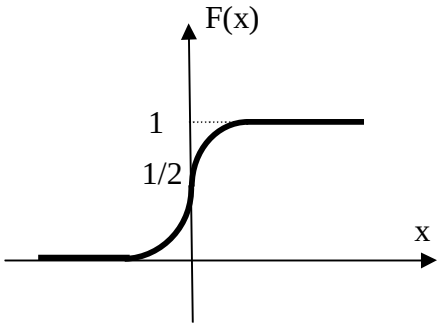


Figure 2.7 : Fonction sigmoïde

4.3. Fonction de sortie

Elle calcule la sortie d'un neurone en fonction de son état d'activation. En général cette fonction est considérée comme la fonction identité. Elle peut être :

- Binaire (0,1) ou bipolaire (-1,1).
- Réelle.

5. Réseaux de neurones

Un neurone réalise simplement une fonction non linéaire, paramétrée, de ses entrées. L'intérêt des neurones réside dans les propriétés qui résultent de leur association en réseaux, c'est-à-dire de la *composition* des fonctions non linéaires réalisées par chacun des neurones.

On distingue essentiellement deux familles de réseau de neurones :

5.1. Réseaux de neurones non bouclés :

Un réseau de neurones non bouclé peut donc être imaginé comme un ensemble de neurones « connectés » entre eux, l'information circulant des entrées vers les sorties sans « retour en arrière ». On peut alors représenter le réseau par un graphe *acyclique* dont les nœuds sont les neurones et les arêtes les « connexions » entre ceux-ci. Si l'on se déplace dans le réseau, à partir d'un neurone quelconque, en suivant les connexions et en respectant leurs sens, on ne peut pas revenir au neurone de départ.

La représentation de la topologie d'un réseau par un graphe est très utile, notamment pour les réseaux bouclés, comme on le verra dans la section « Réseaux de neurones bouclés ». Les neurones qui effectuent le dernier calcul de la composition de fonctions sont les *neurones de sortie* ; ceux qui effectuent des calculs intermédiaires sont les *neurones cachés* (voir la Figure 2.8).

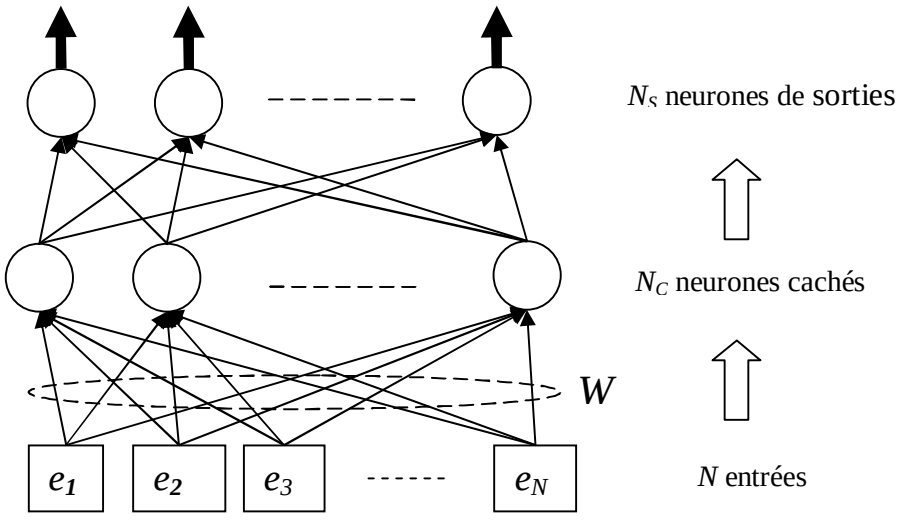


Figure 2.8 : Un réseau de neurones non bouclé à N entrées, une couche de Nc neurones cachés et Ns neurones de sortie.

Un réseau de neurones non bouclé réalise donc une fonction non linéaire ϕ de ses entrées paramétrées par les coefficients W du réseau :

$$Y(k) = \phi(E(k); W) \tag{2.6}$$

Où $Y(k) \in \mathbb{R}^{N_S}$ est le vecteur de sorties à l'instant k, $E(k) \in \mathbb{R}^{N}$ est le vecteur des entrées et ϕ: $\mathbb{R}^{N} \rightarrow \mathbb{R}^{N_S}$ est la fonction non linéaire réalisée par les neurones du réseau.

Le temps ne joue aucun rôle fonctionnel dans un réseau de neurones non bouclé : si les variables sont indépendantes du temps, les sorties le sont également. Le temps nécessaire pour le calcul de la fonction réalisée par chaque neurone est négligeable et, fonctionnellement, on peut considérer ce calcul comme instantané. Pour cette raison, les réseaux non bouclés sont souvent appelés « réseaux statiques », par opposition aux réseaux bouclés ou « dynamiques ».

- *Propriété d'approximation universelle* des RNFs

Toute fonction $\rho(x)$ à valeur bornées, continue ou non, dans un intervalle fermé $[X_A, X_B]$ peut être approchée, à ε près, avec la fonction réalisée par un RNF à une couche d'un nombre fini de neurones cachés à fonction d'activation sigmoïde exponentielle. Plus ceci est vrai pour toute fonction d'activation universelle, c'est-à-dire toute fonction non polynomiale. En particulier, les fonctions dont les limites en $-\infty$ et $+\infty$ sont finies et distinctes, dites sigmoïdales, sont universelles. La précision peut toujours être améliorée en augmentant le nombre de neurones cachés. C'est ce que l'on appelle la propriété *d'approximation universelle* des RNFs (Hornik et al, 1994).

5.2. Réseaux de neurones bouclés :

L'architecture la plus générale, pour un réseau de neurones, est celle des « réseaux bouclés », dont le graphe des connexions est *cyclique* : lorsque l'on se déplace dans le réseau en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ (un tel chemin est désigné sous le terme de « cycle »). La sortie d'un neurone du réseau peut donc être fonction d'elle même ; ceci n'est évidemment concevable que si la notion de *temps* est explicitement prise en considération.

Ainsi, à chaque connexion d'un réseau de neurones bouclé (ou à chaque arête de son graphe) est attaché, outre un paramètre comme pour les réseaux non bouclés, un *retard*, multiple entier (éventuellement nul) de l'unité de temps choisie. Une grandeur, à un instant donné, ne pouvant pas être fonction de sa propre valeur au même instant, tout cycle du graphe du réseau doit contenir au moins une arête dont le retard n'est pas nul.

Un réseau de neurones bouclé à temps discret réalise une (ou plusieurs) équation(s) aux différences non linéaires, par composition des fonctions réalisées par chacun des neurones et des retards associés à chacune des connexions.

A la fin des années 1980, les RNFs bouclés ont commencé à être utilisés comme modèles dynamiques de processus, et comme filtres non linéaires.

Tout réseau de neurones bouclés est un système dynamique non linéaire que l'on peut mettre sous la forme d'une représentation d'état faisant intervenir un (ou des) réseau(x) de neurones non bouclé(s) :

$$\begin{cases} x(k+1) = f(x(k), E(k), W) \\ Y(k) = g(x(k), E(k), W) \end{cases} \quad (2.7)$$

Où k est l'instant discret, $E(k)$ est le vecteur des entrées externes du système, $Y(k)$ est le vecteur des sorties, et $x(k)$ est le vecteur des variables d'état, à l'instant k. les fonctions f et g sont réalisées par les neurones des réseaux, interconnectés avec les paramètres W. Cette représentation d'état est appelée *forme canonique*, et est particulièrement utile pour une formulation simple des algorithmes d'apprentissage. Elle est illustrée par la Figure 2.9.

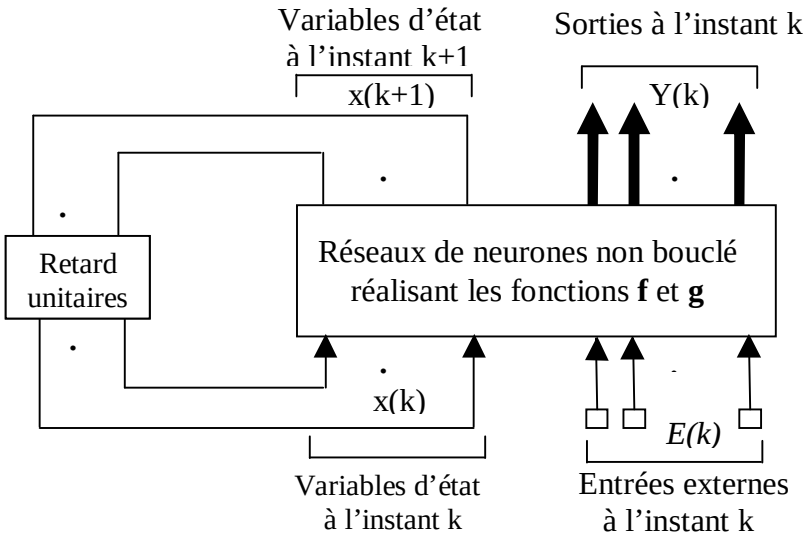


Figure 2.9: Forme canonique d'un réseau de neurones bouclé.

5.2.1. Propriétés des réseaux bouclés

Nous présentons ici la propriété d'approximation universelle pour les réseaux bouclés. Considérons le modèle d'état suivant, supposé représenter exactement le comportement dynamique d'un processus :

$$\begin{cases} x_p(k+1) = \phi(x_p(k), E(k)) \\ Y_p(k) = \gamma\left(x_p(k), E(k)\right) \end{cases} \quad (2.8)$$

Où $E(k)$ est le vecteur des entrées externes du modèle, $Y_p(k)$ est le vecteur des sorties, $x_p(k)$ est le vecteur des variables d'état, ϕ et γ sont des fonctions vectorielles non linéaires. Comme il est possible d'approcher n'importe quelle fonction avec un réseau de neurones non bouclé, on peut espérer qu'un modèle de type (2.8) puisse être approché par un réseau bouclé (2.7), les fonctions ϕ et γ étant réalisées par des réseaux non bouclés.

Plus précisément, la propriété d'approximation universelle pour les réseaux bouclés peut s'énoncer de la manière suivante :

- Pour tout système dynamique (2.8)
- Pour toute précision désirée ε, sur une séquence de taille finie [0; T],
- Pour des entrées bornées $\{E(k)\}$ appartenant à $\mathbb{E} \subset \mathbb{R}^{N_E}$ et un état initial $x_p(0) \in \mathbb{X} \subset \mathbb{R}^{N_X}$,

Il existe un réseau de neurones bouclé (2.7) qui approche le comportement entrée-sortie du système avec la précision ε sur l'intervalle [0 ; T] et sur les ensembles $\mathbb{E}$ et $\mathbb{X}$.

Cette propreté n'est donc pas globale, mais relative à la séquence de taille finie et à l'état initiale considéré : elle ne garantie pas à elle seule que l'approximateur possède des caractéristiques dynamiques du processus qu'il est censé approcher, sa stabilité par exemple.

A cette réserve près, les RNFs se prêtent potentiellement bien à la réalisation de modèles dynamiques.

Plusieurs motivations peuvent pousser l'ingénieur ou le chercheur à concevoir un modèle dynamique :

- utiliser le modèle comme « simulateur » pour prévoir l'évolution d'un processus dont la modélisation de connaissance est trop complexe ou trop incertaine ;
- utiliser le modèle comme simulateur d'un processus dont la modélisation de connaissance est possible, mais conduit à des équations différentielles, ou aux dérivées partielles, dont la résolution numérique est lourde et ne peut répondre à des contraintes de fonctionnement en temps réel : on peut alors créer un ensemble d'apprentissage, et concevoir un réseau de neurones qui fournit de très bonnes solutions dans des temps de calcul beaucoup plus courts. L'architecture de ce réseau peut avantageusement être inspirée des équations différentielles du modèle de connaissance : on conçoit alors un « modèle semi physique » ou modèle « boîte grise » ;
- utiliser le modèle comme prédicteur à très court terme (une période d'échantillonnage) afin de l'intégrer à un système de commande.

Par conséquent, le réseau de neurones récurrent peut être utilisé comme un modèle simulateur, ou comme un modèle prédicteur à un pas :

- *Les modèles de simulation* : un modèle de simulation peut prédire la sortie du processus qu'il représente à long terme, il est utilisé de manière indépendante du processus, pour décrire une dynamique aussi proche que possible à celle de ce dernier. De tels modèles

sont utilisés pour valider la conception d'un système avant sa fabrication, pour la prévision à long terme…etc.

Dans ce cas, le modèle neuronal ne peut pas utiliser les sorties mesurées sur le processus comme des entrées, en revanche, ses entrées seront les sorties prédites par lui-même aux instants précédents, ce qui fait que l'estimation des paramètres de tels modèles est non adaptatif (Oussar, 1998).

- *Les modèles prédicteur à un pas* : un prédicteur à un pas est utilisé en parallèle avec le processus dont il est le modèle, en effet une partie des entrées du modèle sont les sorties mesurées du processus. Il prédit la sortie du processus qui vient juste après un pas d'échantillonnage; il est ainsi par exemple possible de prévoir la sortie du processus à l'instant k+1 connaissant les sorties de ce dernier à l'instant k (Dreyfus et al, 2002).

6. Le réseau de neurones de type perceptron multicouches

L'idée principale du perceptron multicouches (noté MLP pour Multi Layer Perceptron) est de grouper des neurones en couches. La première couche est reliée aux entrées, puis ensuite chaque couche est reliée à la couche précédente. C'est la dernière couche qui produit les sorties du MLP. Il a été ainsi démontré qu'un perceptron multicouche avec une seule couche cachée pourvue d'un nombre suffisant de neurones, peut approximer n'importe quelle fonction avec la précision souhaitée.

Néanmoins, cette propriété ne permet pas de choisir, pour un type de fonction donnée, le nombre de neurones optimal dans la couche cachée. Autrement dit ce résultat ne mène pas vers une technique de construction d'architecture.

La Figure 2.10, représente la structure d'un MLP à une seule couche cachée.

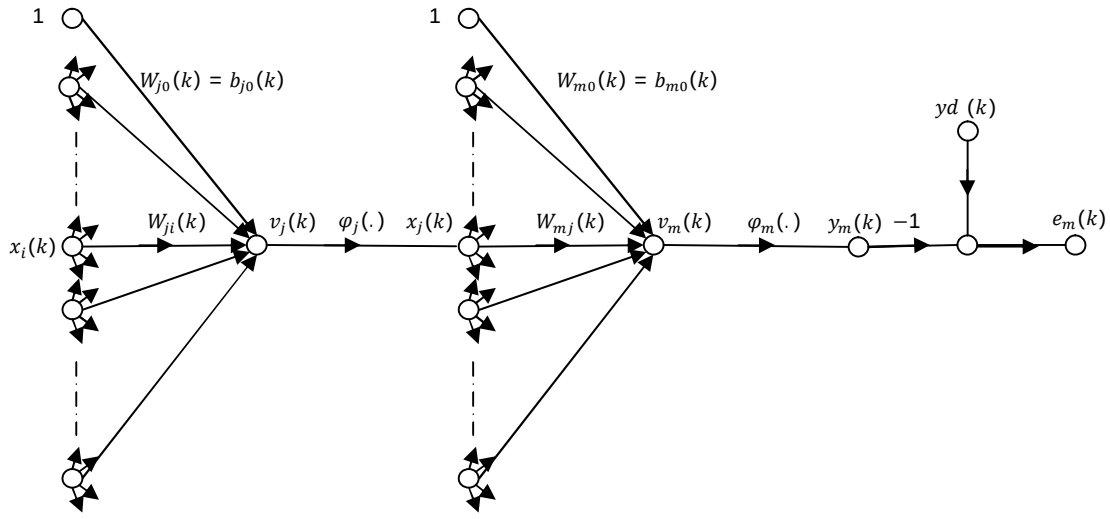


Figure 2.10 : Structure du MLP à une seule couche cachée.

Dans le cas où une seule couche cachée est présente, les fonctions discriminantes réalisées par un tel réseau de neurones sont de la forme :

$$y_m(k) = \varphi_m(v_m) = \varphi_m\left(\sum_{j=0}^{n_j} W_{mj}(k)x_j(k)\right)$$
$$= \varphi_m\left(\sum_{j=0}^{n_j} W_{mj}(k)\varphi_j\left(\sum_{i=0}^{n_i} W_{ji}(k)x_i(k)\right)\right)$$
$$= \varphi_m\left(\sum_{j=1}^{n_j} W_{mj}(k)\varphi_j\left(\sum_{i=1}^{n_i} W_{ji}(k)x_i(k) + b_{j0}(k)\right) + b_{m0}(k)\right) \quad (2.9)$$

avec

$x_i(k)$, $x_j(k)$ et $y_m(k)$: représentent respectivement les valeurs à la sortie des neurones de la couche d'entrée, de la couche cachée et de la couche de sortie,

$\varphi_m(.) et\ \varphi_j(.)$: les fonctions d'activation des neurones de la couche de sortie et de la couche cachée (généralement, φ_m est une fonction linéaire et φ_j est une fonction non linéaire),

$b_{m0}(k)\ et\ b_{j0}(k)$: Les biais des neurones de la couche de sortie et de la couche cachée.

$n_i\ et\ n_j$: nombre des neurones de la couche d'entrée et de la couche de sortie.

W_{ji} : poids synaptique (ou de connexion) entre le neurone amont i, et le neurone aval j.

$yd\ (k)$: représente la valeur de la sortie désirée du système à l'instant k.

6.1. Modèles-hypothèses entrée-sortie et prédicteurs associés

Les Modèles-hypothèses d'état sont les plus généraux, mais nous commençons par présenter les modèles entrée-sortie, qui facilitent la compréhension de la modélisation en présence du bruit.

6.1.a. Modèles-hypothèses entrée-sortie avec bruit de boucle

Nous introduisons en premier lieu le modèle-hypothèse avec bruit dans la boucle, ou bruit d'état, habituellement appelé NARX (Non linéaire AutoRégressif avec entrée eXogène, ou eXtérieure), et qui constitue une extension non linéaire du modèle linéaire ARX. Ce Modèle-hypothèse postule l'existence d'une fonction η telle que :

$$y_P(k) = \eta\left(y_P(k-1), \dots, y_P(k-n), u(k-1), \dots, u(k-m)\right) + V(k) \quad (2.10)$$

Où les $\{V(k)\}$ sont les réalisations d'un bruit pseudo blanc d'espérance mathématique nulle. En adoptant ce modèle, on suppose donc que u est la seule entrée externe, que les paramètres m et n ont une valeur suffisante. On fait d'autre part l'hypothèse que la sortie à l'instant k dépend du bruit au même instant par l'intermédiaire du terme additif, mais également du bruit à tous les instants précédents par l'intermédiaire de $y_P(k), \dots, y_P(k-n)$: le bruit intervient dans la boucle.

Le prédicteur théorique associé au modèle-hypothèse avec bruit de boucle est non bouclé :

$$y(k+1) = \eta\big(y_P(k), \dots, y_P(k-n+1), u(k), \dots, u(k-m+1)\big) \quad (2.11)$$

D'après (2.10), ce prédicteur fournit bien l'espérance mathématique conditionnelle de la sortie du processus $y_P(k+1)|k$. Il est optimal au sens de la variance de l'erreur de prédiction $e(k+1) = y_P(k+1) - y(k+1)$ est minimale, puisqu'elle est égale à $V(k+1)$, qui est par définition imprédictible.

Afin d'obtenir une estimation non biaisée, il faut donc mettre en œuvre un prédicteur neuronale ayant les mêmes entrées que le prédicteur théorique, par exemple un RNF non bouclé de paramètres $\boldsymbol{W}$:

$$y(k+1) = h(y_P(k), \dots, y_P(k-n+1), u(k), \dots, u(k-m+1), \boldsymbol{W}) \quad (2.12)$$

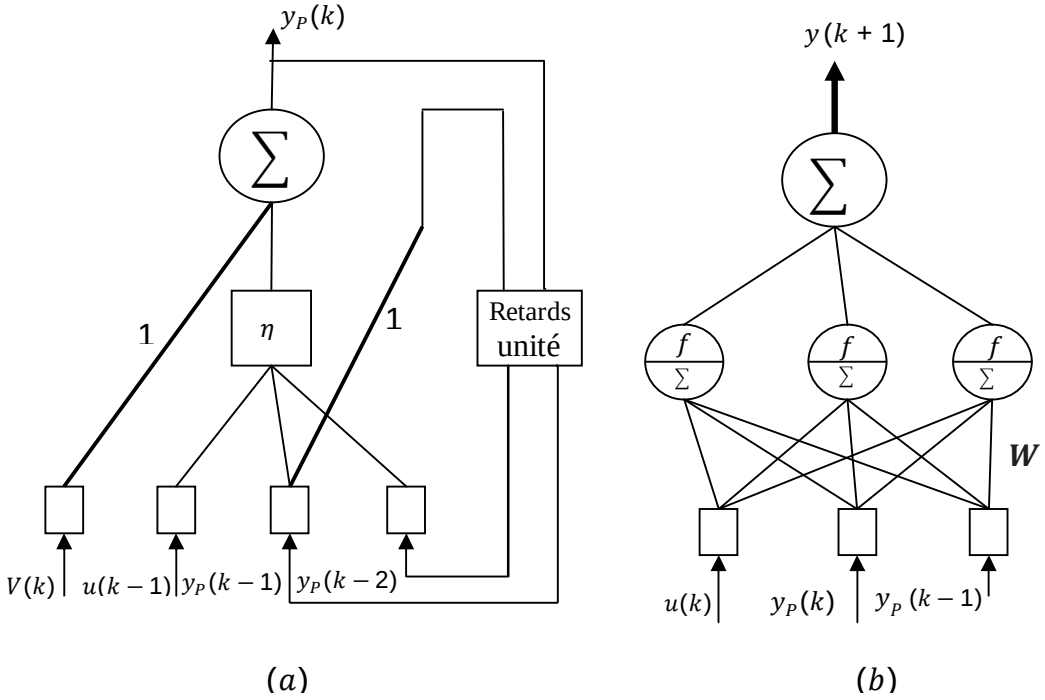


Figure 2.11 :

(a) Modèle-hypothèse entrée-sortie avec bruit dans la boucle (NARX) d'ordre 2 (n=2) et avec m=1, **(b)** Prédicteur neuronal MLP non bouclé d'entrées définies à partir du prédicteur théorique associé.

Ainsi, bien que le modèle-hypothèse soit récursif (bouclé), le prédicteur est non récursif (non bouclé), et l'estimation des paramètres W, s'effectue comme en modélisation statique avec un réseau de neurones non bouclé (voir Figure 2.11). Si le modèle-hypothèse est vrai, si le réseau du prédicteur est suffisamment complexe, si les séquences d'apprentissage $\{u(k), y_P(k)\}$ sont suffisamment riches, est si l'on met en œuvre un algorithme d'apprentissage performant, alors la fonction h sera une bonne approximation (plus

précisément une approximation non biaisée) de la fonction η dans le domaine exploré par l'ensemble d'apprentissage. Le prédicteur neuronal sera alors une bonne approximation du prédicteur théorique.

6.1.b. Modèles-hypothèses entrée-sortie avec bruit de sortie

Un deuxième modèle-hypothèse postule un bruit additif sur la sortie, par exemple pour rendre compte du bruit de mesure provenant d'un capteur :

$$\begin{cases} x_P(k) = \eta(x_P(k-1), \dots, x_P(k-n), u(k-1), \dots, u(k-m)) \\ y_P(k) = x_P(k) + V(k) \end{cases} \quad (2.13)$$

Les $\{V(k)\}$ sont les réalisations d'un bruit pseudo blanc centré. Ce modèle-hypothèse est connu en anglais, pour les modèles linéaires, sous le nom de « *Output Error Model* ». Contrairement au modèle NARX, c'est la sortie du modèle qui est affectée par le bruit à l'instant k : le bruit n'intervient pas dans la boucle.

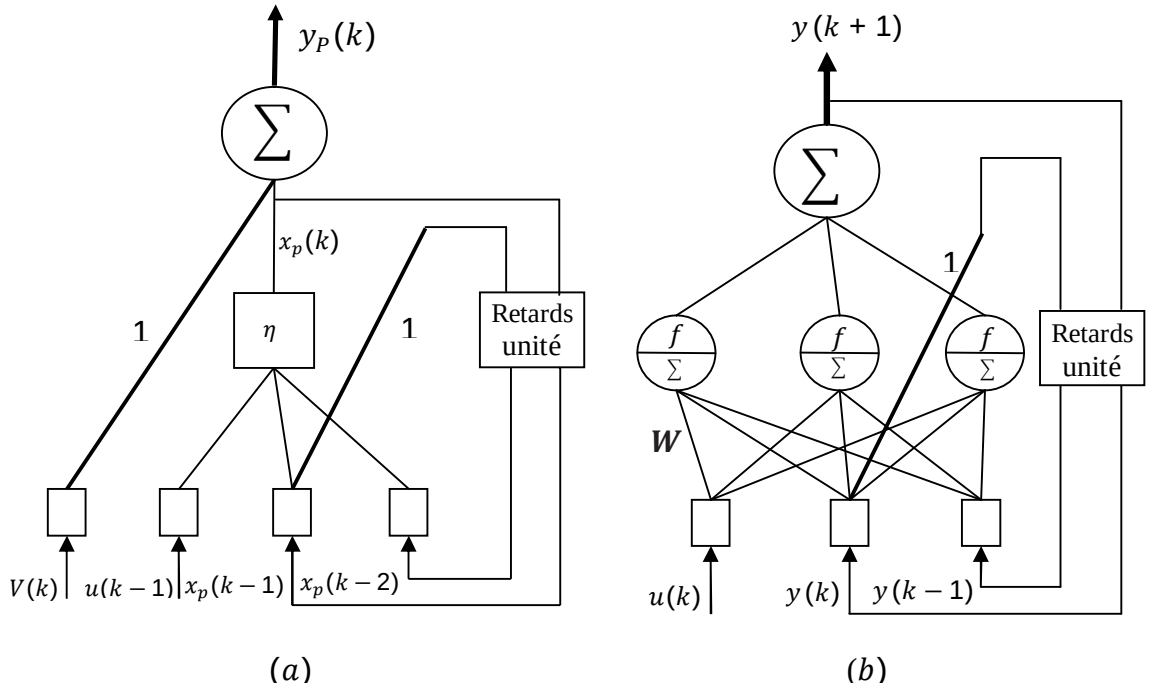


Figure 2.12

(b) Modèle-hypothèse entrée-sortie avec bruit additif de sortie (Output Error Model) d'ordre 2 (n=2) et avec m=1 ; **(b)** Prédicteur neuronal MLP bouclé d'entrées définies à partir du prédicteur théorique associé.

Le prédicteur théorique associé au modèle-hypothèse avec bruit de sortie est un prédicteur bouclé :

$$y(k+1) = \eta\big(y(k), \dots, y(k-n+1), u(k), \dots, u(k-m+1)\big) \quad (2.14)$$

Supposant que les n erreurs de prédiction précédant celle de l'instant k soient égale au bruit :

$y_P(k-i+1) - y(k-i+1) = V(k-i+1)$, autrement dit $y(k-i+1) = x_P(k-i+1)$ pour i=1 à n. Alors on a : $e(k+1) = y_P(k+1) - y(k+1) = V(k+1)$. Pour des conditions initiales différentes, si le prédicteur est stable, l'erreur tend asymptotiquement vers le bruit.

Pour obtenir un modèle prédictif non biaisé, il faut donc mettre en œuvre un prédicteur neuronal bouclé comme le prédicteur théorique :

$$y(k+1) = h(y(k), \dots, y(k-n+1), u(k), \dots, u(k-m+1), \boldsymbol{W}) \qquad (2.15)$$

Où la fonction h est réalisée par un réseau MLP non bouclé de paramètres W(voir Figure 2.12). Si le modèle-hypothèse est vrai, si le réseau du prédicteur est suffisamment complexe, si les séquences d'apprentissage sont suffisamment riches, est si l'algorithme d'apprentissage est performant, le prédicteur neuronal obtenu est une bonne approximation du prédicteur théorique associé dans le domaine exploré par l'ensemble d'apprentissage.

7. Réseau de neurones à espace d'état :

Les réseaux de neurones à espace d'état (State-Space Neural Network, ou SSNN) est un paradigme de réseau de neurones qui permet d'assurer la stabilité, et d'analyser le comportement dynamique avec les poids synaptiques du réseau.

Il possède une architecture de réseau de neurones récurrent (RNR), dont la structure reflète exactement une modélisation dans l'espace d'état d'un système dynamique non linéaire. (Zamarreno et al, 2000).

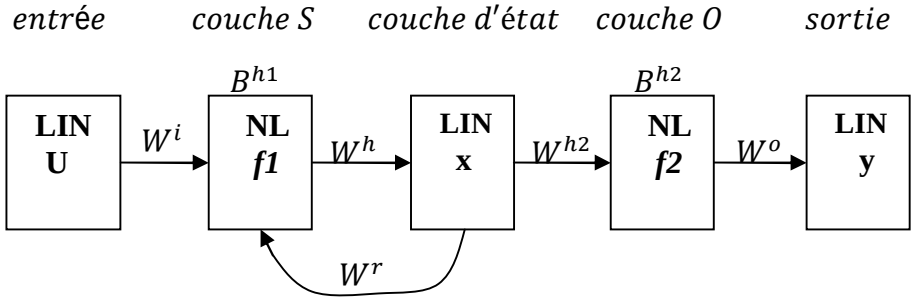


Figure 2.13 : schéma bloc d'un réseau de neurones à espace d'état.

LIN : éléments linéaires,

f1, f2 : fonctions non linéaires.

Le schéma bloc représenté par la Figure 2.13, est équivalent à la représentation d'état d'un système non linéaire déterministe qui a pour représentation mathématique :

$$\begin{cases} x(k+1) = \boldsymbol{F}(x(k), u(k)) \\ y(k) = \boldsymbol{G}(x(k)) \end{cases} \qquad (2.16)$$

Où $x\epsilon\mathbb{R}^s$ est le vecteur des états, $u\epsilon\mathbb{R}^n$ est le vecteur d'entrée, $y\epsilon\mathbb{R}^m$ est le vecteur de sortie, $\boldsymbol{F} : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}^s$, et $\mathbf{G} : \mathbb{R}^s \to \mathbb{R}^m$, sont deux fonctions non linéaires statiques.

La Figure 2.13 représente un réseau de neurones à espace d'état, ce réseau de neurones récurrent spécial est composé de cinq couches :

- *La couche d'entrée :* cette couche fournit les entrées de commandes à chaque neurone de la couche suivante, chaque entrée de commande est pondérée par le poids W^i .
- *La couche cachée (S) :* cette couche représente la fonction non linéaire qui représente le comportement des états, la lettre **(S)** vient du mot « State ».
- *La couche d'état :* chaque neurone de cette couche représente un état
- *La couche cachée (O) :* c'est la couche avant la couche de sortie, elle représente la fonction non linéaire qui relie la sortie du réseau aux états. La lettre **(O), «** vient du mot Output ».
- *La couche de sortie :* relie les signaux de la couche cachée (O) à chaque neurone de sortie, ces sorties sont finalement les sorties du SSNN.

Un réseau de neurones de type SSNN comporte deux avantages principaux :

1. Etant un modèle neuronal, il a la flexibilité de représenter n'importe quelle fonction non linéaire.
2. Etant un modèle d'espace d'état, le nombre de connexions extérieures est minimal, donc le nombre de paramètre à ajuster est minimal. Les entrées /sorties du modèle neuronal sont les entrées/sorties du système physique.

Il a été prouvé que n'importe quelle fonction non linéaire peut être estimée par un réseau de neurones contenant une seule couche cachée, composée de neurones dont la fonction de transfert sigmoïdale est bouclée. Ainsi deux réseaux de neurones interconnectés peuvent représenter le système d'équation (2.16), l'un représente la fonction qui donne le comportement d'état et l'autre représente la fonction qui relie les sorties aux états. Ainsi le modèle (2.16) peut être représenté comme suit (Zamarreno et al, 2000) :

$$\begin{cases} \hat{x}(k+1) = \mathrm{W}^{\boldsymbol{h}}.f_1(W^r.\hat{x}(k) + W^i u(k) + B^{h1}) \\ \hat{y}(k) = \mathrm{W}^{\boldsymbol{o}}.f_2(W^{h2}.\hat{x}(k) + B^{h2}) \end{cases} \tag{2.17}$$

Où

$-\ \mathrm{W}^{\boldsymbol{h}}$**,** $W^r, W^i, \mathrm{W}^{\boldsymbol{o}}$**,** et W^{h2} sont des matrices de dimensions $s \times h, h \times s, h \times n, m \times h2$, et $h2 \times s$ respectivement.

– B^h, B^{h2}sont deux vecteurs de biais de dimension h et $h2$, qui ont pour rôle d'augmenter ou d'abaisser l'entrée nette de la fonction d'activation.

– $f1$ et $f2$ sont deux fonctions non linéaires, généralement elles sont sigmoïdales.

La Figure 2.14 est une réalisation de l'équation (2.17), elle montre clairement les couches du réseau ainsi que les liens entre les neurones.

La différence entre ce type de réseaux et les réseaux de neurones classiques, tels que le réseau multicouches précédent, c'est que le réseau multicouches apprend et estime une fonction ou une relation entre un espace d'entrée (par exemple l'espace des entrée de commande *u(k)* d'un système) et un espace de sortie (par exemple l'espace de sorties *y(k)* d'un système) sans tenir compte du comportement interne du système.

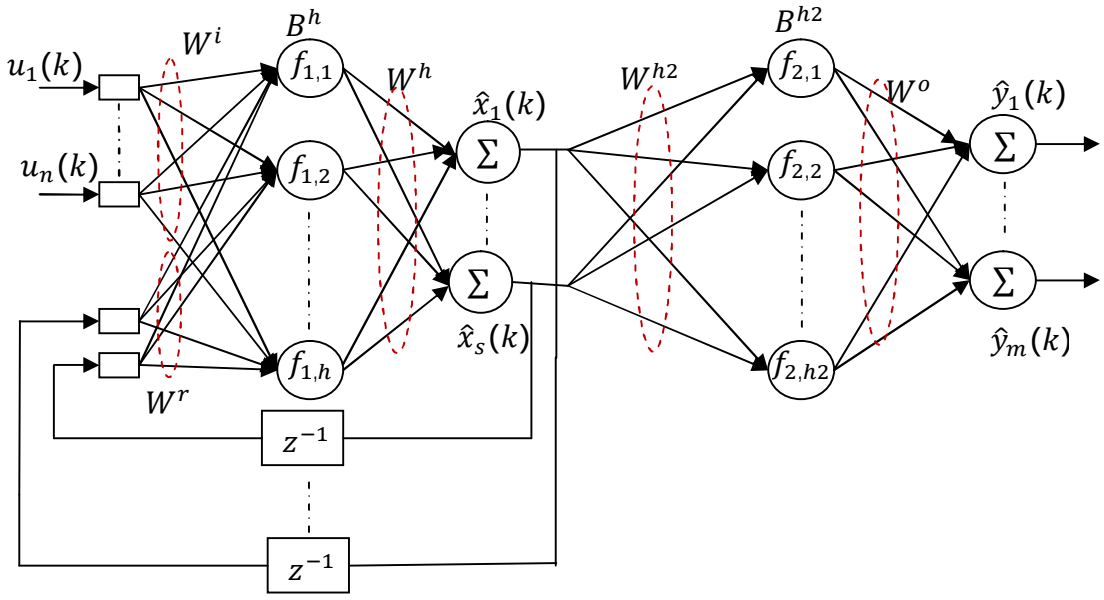


Figure 2.14 : Réseau de neurones à espace d'état.

Par contre la structure d'un SSNN cherche à estimer les deux sous fonctions qui représentent les transformations non linéaires de la représentation d'état. C'est-à-dire apprendre d'abord une relation entre un espace d'entrée (entrée de commande *u(k)*), et un espace interne (les variables d'états *x(k)*), et ensuit apprendre une autre relation entre un espace interne (les variables d'états *x(k)*), et l'espace de sortie (sortie *y(k)*) ; dans le cas où les grandeurs internes ne sont pas accessibles (mesurables), il sera alors difficile d'estimer les sous fonctions.

7.1. Etude du cas d'un système linéaire (cas particulier) :

Dans le cas particulier où le système est linéaire, c'est-à-dire les fonctions **F** et **G** sont linéaire, et lorsque le bais est égal à zéros, alors l'équation (2.17) devient :

$$\begin{cases} \hat{x}(k+1) = \mathbf{W^h}.(W^r.\hat{x}(k) + W^i u(k)) \\ \hat{y}(k) = \mathbf{W^o}.(W^{h2}.\hat{x}(k)) \end{cases} \quad (2.18)$$

En posant $\mathbf{W^h}.W^r$=Φ, $\mathbf{W^h}.W^i$=Γ, $\mathbf{W^o}.W^{h2}$ = Ψ, et en effectuant les remplacements nécessaires dans l'équation (2.18), on obtient :

$$\begin{cases} \hat{x}(k+1) = \Phi.\hat{x}(k) + \Gamma.u(k) \\ \hat{y}(k) = \Psi.\hat{x}(k) \end{cases} \quad (2.19)$$

Le modèle de (2.19) est de la forme d'un modèle linéaire déterministe. Φ, Γ et Ψ sont des matrices dont les éléments sont les poids des connexions du réseau SSNN.

Les SSNN peuvent être utilisés pour l'identification d'une large gamme des systèmes dynamiques linéaires et non linéaires, dans notre cas, on va les appliquer uniquement pour les systèmes dynamiques linéaires.

8. Apprentissage

L'apprentissage est vraisemblablement la propriété la plus intéressante des réseaux de neurones. C'est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré.

L'apprentissage neuronal fait appel à des exemples de comportement. Durant cette phase de fonctionnement, le réseau adapte sa structure (le plus souvent, les poids des connexions) afin de fournir sur ses neurones de sortie les valeurs désirées. Cet apprentissage nécessite des exemples désignés aussi sous l'appellation d'échantillon d'apprentissage ainsi qu'un algorithme d'apprentissage.

Après initialisation des poids du réseau (en générale des valeurs aléatoires), il y a présentation des exemples au réseau et calcul des sorties correspondantes. Une valeur d'erreur ou de correction est calculée et une correction des poids est appliquée.

Au niveau des algorithmes d'apprentissage, il a été défini deux grandes classes selon que l'apprentissage est dit supervisé ou non supervisé (Mang-Hui, 2005). Cette distinction repose sur la forme des exemples d'apprentissages. Dans le cas de l'apprentissage supervisé, les exemples sont des couples (entrée, sortie associée) alors que l'on ne dispose que de valeurs (entrée) pour l'apprentissage non supervisé.

8.1. Apprentissage non supervisé

L'apprentissage est qualifié de non supervisé lorsque seules les valeurs d'entrée sont disponibles. Dans ce cas les exemples présentés à l'entée provoquent une auto-adaptation du réseau afin de produire des valeurs de sorties qui soient proches en réponse à des valeurs d'entrée similaires (de même nature).

8.2. Apprentissage supervisé

L'apprentissage est dit supervisé lorsque les exemples sont constitués de couples de valeurs du type : (valeur d'entrée / valeur de sortie). Tout le problème de l'apprentissage supervisé consiste, étant donné un ensemble d'apprentissage E de N couples (entrées / sorties désirées) (u_i, y_i) i=1,2,…. N, à déterminer le vecteur des poids W d'un réseau F_W capable de mettre ces informations en correspondance, c'est à dire un réseau tel que :

$$F_W(u_i) = y_i \ , \qquad \text{i=1,2, …. , N.} \tag{2.20}$$

Pour l'apprentissage supervisé des réseaux on peut appliquer les méthodes d'apprentissage de premier ou de deuxième ordre basées sur le calcul du gradient:

- *Méthodes du premier ordre :* les méthodes dites du premier ordre sont basées sur le calcul du gradient de la fonction du coût, donnée par la relation (2.21), le minimum de cette fonction est atteint si son gradient (autrement dit sa dérivée) est nul. On compte parmi ces méthodes, la méthode du gradient simple, la méthode du gradient stochastique …

$$J(W) = \sum_{k=1}^{N} J^k\ (W) = \sum_{k=1}^{N} \frac{1}{2}\left(e^k(W)\right)^2 = \sum_{k=1}^{N} \frac{1}{2}\left(yd^k - y^k(W)\right)^2 \tag{2.21}$$

Où : $J^k(W)$ est le coût partiel relatif à l'exemple k.

- *Méthodes du deuxième ordre :* les méthodes du deuxième ordre sont ainsi appelées parce qu'elles prennent en considération la dérivé seconde de la fonction de coût (2.21). Parmi ces méthodes, on compte la méthode de Newton, la méthode de Levenberg Marquardt (Lucea, 2006).

À présent, nous allons illustrer deux méthodes d'apprentissage supervisées, en premier lieu on va voir une méthode du premier ordre, dite algorithme de rétro-propagation du gradient, en second lieu on exposera une méthode du deuxième ordre, appelée Méthode de Levenberg-Marquardt (LM).

8.2.a. Algorithme de rétro-propagation du gradient

L'algorithme d'apprentissage par rétro-propagation du gradient de l'erreur est un algorithme itératif de premier ordre qui a pour objectif de trouver les poids des connections minimisant l'erreur quadratique moyenne (2.21), commise par le réseau sur l'ensemble d'apprentissage.

Cet algorithme d'apprentissage du réseau MLP non bouclé, qui est donné par la Figure 2.10, peut se résumer comme suit :

1. initialiser l'ensemble des vecteurs de poids $W_{ji}(0), W_{mj}(0)$, et le paramètre d'apprentissage η à des valeurs aléatoires,
2. appliquer le vecteur $x_i(k)$ en entrée du réseau,
3. calculer la sortie $y_m(k)$ donnée par la relation (2.9),
4. calculer l'erreur $e_m(k) = yd\,(k) - y_m(k)$,
5. calculer l'ensemble des nouveaux poids $W_{mj}(k+1)$ et $W_{ji}(k+1)$,

$$W_{mj}(k+1) = W_{mj}(k) + \eta\delta_m(k)x_j(k) \tag{2.22}$$

avec

$$\delta_m(k) = e_m(k)\varphi'_m(v_m) \tag{2.23}$$

et

$$W_{ji}(k+1) = W_{ji}(k) + \eta\delta_j(k)x_i(k) \tag{2.24}$$

avec

$$\delta_j(k) = \varphi'_j(v_j)\sum_{m=0}^{n_m}\delta_m(k)\,W_{mj} \tag{2.25}$$

Où n_m est le nombre de neurones de la couche de sortie.

6. incrémenter k → k+1 et aller à l'étape 2.

L'utilisation de l'algorithme du gradient peut conduire à des blocages en cas d'un minimum local. Ainsi son efficacité dépend, en effet, d'un grand nombre de paramètres que doit fixer l'utilisateur : le pas du gradient, le paramètre d'apprentissage η, les paramètres des fonctions d'activations, l'initialisation des poids et les biais, l'architecture du réseau, le nombre de neurones par couche, etc.

En revanche, l'algorithme du gradient est très efficace loin du minium local, lorsque la norme du gradient a une grande valeur. Toutefois, la convergence devient particulièrement lente lorsque le vecteur paramètre W s'approche d'un minimum : on a alors intérêt à utiliser un algorithme du (quasi) second ordre (Personnaz et Rivals, 2003).

8.2.b. Méthode de Levenberg-Marquardt (LM)

L'algorithme de Levenberg-Marquardt est un algorithme itératif, consiste à modifier les paramètres proportionnellement au gradient de la fonction du coût total. Cet algorithme adopte une direction de déplacement qui n'est plus la direction du gradient, mais une transformation linéaire appropriée du gradient du coût total, en modifiant les paramètres du réseau selon la formule suivante (Personnaz et Rivals, 2003 ; Dreyfus, 2007) :

$$W(k+1) = W(k) - \left[H\big(W(k)\big) + \lambda_k I\right]^{-1}\nabla J(W(k)) \tag{2.26}$$

Le gradient $\nabla J(W(k))$ du coût quadratique $J(W)$ qui est donné par la relation (2.21), est calculé comme suit :

$$\nabla J(W(k)) = \frac{\partial J}{\partial W} = \sum_{k=1}^{N} \frac{\partial J^k}{\partial W} = \sum_{k=1}^{N} e^k \frac{\partial e^k}{\partial W} = -\sum_{k=1}^{N} e^k \frac{\partial y^k}{\partial W} = -Ja^T e \qquad (2.27)$$

Ja est la matrice Jacobienne ou le Jacobien.

λ_k est un paramètre constant strictement supérieur à zéro.

H est la matrice hessienne ou la dérivée seconde par rapport au vecteur paramètre W. C'est une matrice carrée symétrique d'ordre q.

$$H(W) = \frac{\partial^2 J}{\partial W \partial W^T} \qquad (2.28)$$

$$[H]_{i,j} = \frac{\partial^2 J}{\partial W_i \partial W_j} = \sum_{k=1}^{N} \left(\frac{\partial e^k}{\partial W_i} \frac{\partial e^k}{\partial W_j} + \frac{\partial^2 e^k}{\partial W_i \partial W_j} e^k \right) \quad \text{pour} \quad \text{i, j=1 à q} \qquad (2.29)$$

On néglige souvent le second terme de (2.29), qui est effectivement petit si la surface est proche d'une quadrique. Donc pour un RNF possédant une seule sortie, on a ainsi l'expression approchée du Hessien (Mahul, 2005) :

$$H(W) = Ja^T Ja \qquad (2.30)$$

Il faut noter que l'expression de la matrice hessienne de la fonction de coût ne s'applique que si la fonction à optimiser est la fonction de coût des moindres carrés ; la méthode de Levenberg Marquardt ne peut donc pas s'appliquer à l'optimisation de n'importe quelle fonction de coût, notamment à la minimisation de la fonction de coût d'entropie croisée pour la classification. (Personnaz et Rivals, 2003)

Parmi les nombreuses méthodes d'asservissement du paramètre λ_k de l'algorithme de Levenberg-Marquardt, voici une méthode simple et robuste. (Voir Figure 2.15)

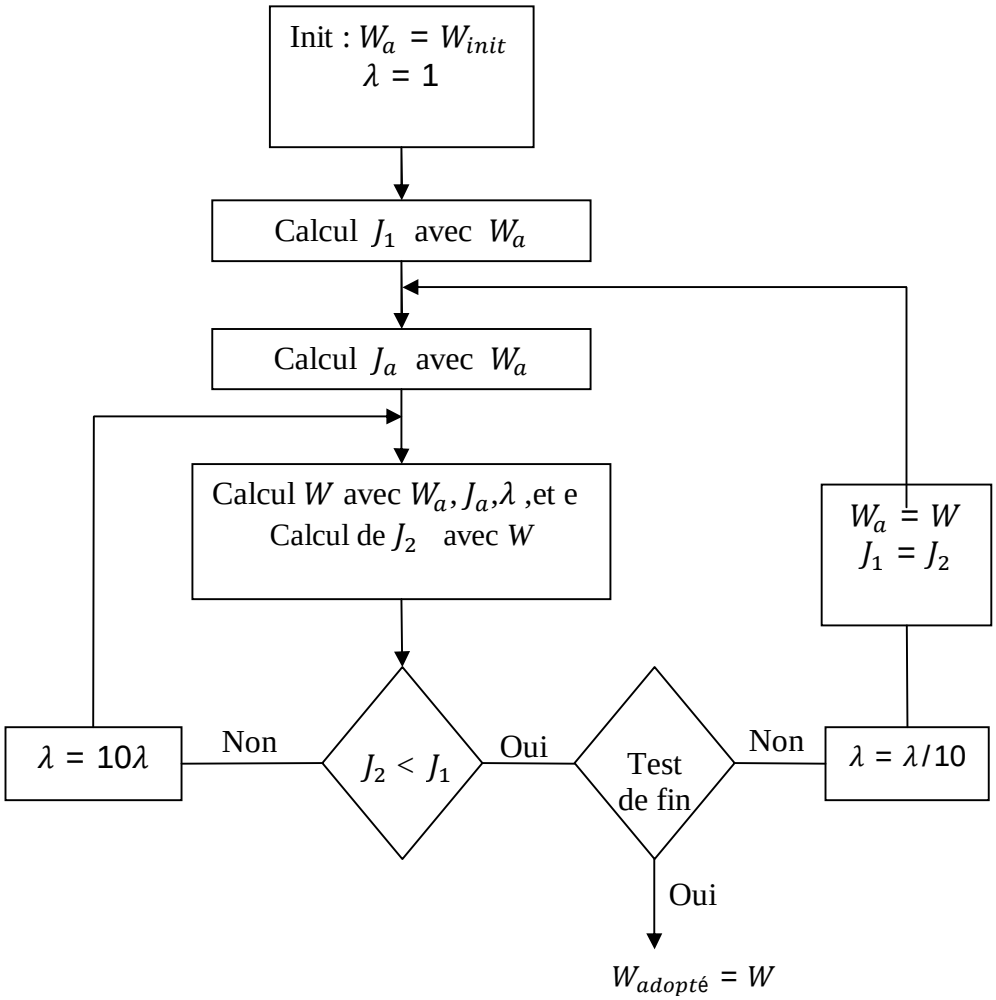


Figure 2.15: Asservissement économique du paramètre λ de l'algorithme de Levenberg-Marquardt.

1. Si la modification des paramètres provoque la diminution du cout J_2, on accepte cette modification, et l'on se rapproche de la direction de newton en diminuant λ (par exemple en le divisant par 10). Et on calcule le Jacobien avec les nouveaux paramètres, etc.
2. Si la modification des paramètres provoque l'augmentation du coût J_2, on rejette cette modification et l'on se rapproche de la direction du gradient en augmentant λ (par exemple en le multipliant par 10). Ensuite on calcule une nouvelle modification des paramètres avec le même gradient, si J_2 ne décroit pas on recommence l'opération.

Il été démontré dans plusieurs études que l'algorithme de Levenberg-Marquardt et plus performant et plus rapide dans les calculs que l'algorithme du gradient, par conséquent il sera utilisé tout au long de ce mémoire dans l'étape d'apprentissage des réseaux de neurones.

9. Amélioration de la généralisation (ou le dilemme biais/variance)

Un problème qui apparaît lors d'un apprentissage est le problème du sur apprentissage. Si le réseau de neurone apprend par cœur, il donnera de mauvais résultats quand on lui présentera des données un peu différentes. Des méthodes existent pour optimiser la phase d'apprentissage afin que le phénomène de sur ou sous apprentissage disparaisse, dont la technique de l'early stopping et de la régularisation (Int, 2005).

9.1. Régularisation

La technique de régularisation consiste à imposer des contraintes, donc à apporter une information supplémentaire, sur l'évolution des poids du réseau de neurones. Par exemple, on peut volontairement pénaliser les poids trop grands selon la formule :

$$F_{erreur} = \frac{1}{N}\sum(e_i)^2 \qquad \text{et} \qquad F_{poids} = \frac{1}{n}\sum(w_i)^2 \tag{2.31}$$

donc on impose

$$F_{new} = \gamma F_{erreur} + (1-\gamma)F_{poids} \tag{2.32}$$

où γ est un paramètre d'optimisation. Mais le problème réside dans le choix de la valeur de ce paramètre.

La régularisation bayesienne, qui suppose que les poids et les biais suivent des distributions spécifiques (les paramètres sont estimés au fur et à mesure de l'apprentissage) donne en général des résultats très satisfaisants.

9.2. Early stopping

Cette technique consiste à diviser les données disponibles en trois lots distincts. Le premier lot sert à entraîner le réseau de neurones. Le second lot sert à la validation du réseau. L'erreur de validation doit normalement diminuer au cours du processus d'apprentissage (la variance diminue). Mais quand le réseau commence à apprendre par cœur (le biais augmente, alors l'erreur de validation recommence à croître), on arrête alors la phase d'apprentissage. Quant au troisième lot, il sert à vérifier que la généralisation est correcte.

9.3. Normalisation des données

Afin d'améliorer la performance des réseaux neuronaux multicouches, il est préférable de normaliser les données d'entrée et de sortie de telle sorte qu'elles se trouvent dans l'intervalle [-1 +1].

9.4. Recherche de l'information

Avant de vouloir utiliser un réseau de neurones en tant qu'approximation de fonction, il est nécessaire de faire des études de sensibilités afin de déterminer les paramètres pertinents qui doivent être gardés, et de supprimer les autres, qui ne feraient que diminuer la performance du réseau. Une autre solution peut aussi consister à faire une analyse en composante principale sur le jeu de données (réduction de l'information).

Conclusion :

Ce chapitre est dédié à l'étude des réseaux de neurones artificiels, En premier lieu, on a rapporté quelques définitions de bases sur le neurone biologique ainsi que sur le neurone formel ou artificiel. Ensuite une étude comparative a été effectuée sur ces deux derniers.

Le réseau de neurones artificiels est un ensemble de neurones formels interconnectés les un aux autres par des liaisons pondérées, et selon la façon dont ces dernières sont effectuées on peut distinguer deux grandes familles de réseaux de neurones : le réseau bouclé (ou dynamique) et le réseau non bouclé (ou statique). Ce dernier réalise des fonctions non linéaires alors que les réseaux bouclés (ou récurrent), réalise des équations aux déférences (ou équation récurrentes) non linéaires.

Ce chapitre introduit deux types de réseaux de neurones récurrents à savoir, le réseau de neurones multicouches MLP bouclé d'entrée, et le réseau de neurones à espace d'état (SSNN) ; une étude détaillée à été portée sur deux derniers. Le réseau de neurones multicouches MLP bouclé d'entrée est un réseau de type récurrent, par conséquent il réalise des équations aux différences (ou équation récurrentes) non linéaires. La structure des réseaux de neurones à espace d'états se rapproche beaucoup de la représentation d'état classique, ce qui lui procure la possibilité de mieux représenter les systèmes dynamique.

Ce chapitre se termine par une présentation de deux méthodes d'apprentissage supervisées de premier ordre et de second ordre, ainsi que quelques techniques améliorant la généralisation du réseau de neurones.

Chapitre 3

Identification par réseaux de neurones des modèles d'ordre non entier

1. Introduction

La fonction de transfert réelle d'un procédé industriel est pratiquement impossible à déterminer avec précision. Il est alors nécessaire d'identifier ce procédé à partir de l'analyse de ses entrées/sorties, en vue d'obtenir un modèle mathématique représentant son comportement dynamique. A priori, les réseaux de neurones récurrents seraient des candidats idéaux pour la réalisation de ce genre de tâches notamment par ce qu'ils sont capables de maintenir une représentation non linéaire de l'historique des entrées au sein de leurs dynamiques internes.

Les systèmes décrits par des modèles fractionnaires, utilisant des équations différentielles fractionnaires basées sur la dérivée non entière ont suscité l'intérêt de la communauté scientifique, en raison du nombre de plus en plus important des phénomènes physiques appropriés au comportement fractionnaire. Comme on la souligné précédemment ces phénomènes sont caractérisés par les propriétés de mémoire longue et de dimension infinie. Cette caractéristique impose l'utilisation de modèle à dimension infini pour modéliser un système fractionnaire par des modèles entiers. Dans ce contexte, l'identification de tels systèmes par des modèles entiers s'avers inadaptée, et requiert un nombre important de paramètres pénalisant ainsi le modèle obtenu. Le développement des méthodes d'identifications basées sur les modèles non entiers apparait donc comme une nécessité, mais ils se révèlent donc nettement plus complexes que dans le cas entier, du fait de l'estimation des ordres fractionnaires aux mêmes titres que les paramètres du modèle (Djamah et al, 2010).

Ainsi les réseaux de neurones artificiels, peuvent constituer une nouvelle approche permettant d'aborder sous des angles nouveaux les problèmes d'identification et de modélisation des systèmes fractionnaire.

L'objectif de ce chapitre concerne l'identification et la prédiction à un pas d'échantillonnage à l'aide des réseaux de neurones, la dynamique des modèles fractionnaires, synthétisés soit à partir de la fonction de transfert ou bien à partir des formes d'état d'ordre non entier. Pour se faire, nous sommes basés sur deux architectures neuronales différentes. La première architecture est appelée « Réseau de neurone multicouches MLP bouclé d'entrée ». Quant à la seconde, appelée « Réseau de neurones à espace d'état», qui utilise la représentation d'état des systèmes pour reconstituer le comportement des états et la sortie du système.

2. Identification

L'identification d'un système a pour intérêt évident en commande de processus : c'est souvent sur la base d'un modèle qu'il est possible de concevoir une stratégie de conduite, hors ligne ou en ligne. Or, si l'identification des systèmes linéaires a été largement étudiée et semble ne plus poser de problèmes insurmontables pour la majorité des cas, il n'en est pas de même pour les systèmes non-linéaires et les systèmes d'ordre non entier. En revanche, les réseaux de neurones offrent d'intéressants avantages lorsqu'on les utilise pour l'approximation de fonctions. Il est dès lors nature de penser à appliquer ces techniques en identification de processus.

Généralement, on peut distinguer deux techniques d'identification à base de réseau de neurones: l'identification directe et l'identification inverse.

2.1. Identification directe

Le principe de l'identification directe est illustré par la Figure 3.1. Pour ce type d'identification le réseau de neurones identificateur RNI est utilisé en parallèle avec un processus de type boite noire. La sortie estimée du réseau $\hat{y}$ est comparée à la sortie réelle du processus y ; l'erreur e qui en résulte sert à ajuster les paramètres du réseau neuronal.

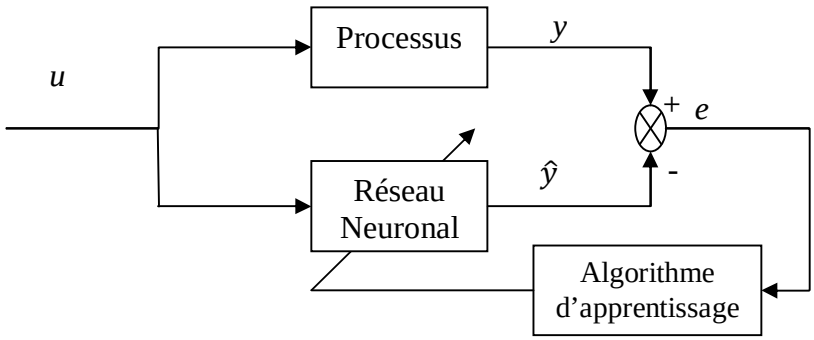


Figure 3.1 : Schéma d'identification directe d'un processus avec réseau de neurones.

2.2. Identification inverse

Dans cette technique, l'entrée *u* du processus est comparée avec la sortie estimée $\hat{u}$ de l'identificateur neuronal RNI, et la sortie du processus est injectée comme entrée du réseau de neurones. Le principe de cette identification est illustré par la Figure 3.2.

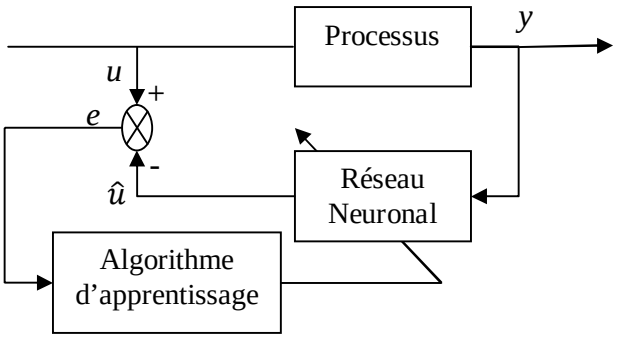


Figure 3.2 : Schéma d'identification inverse d'un processus avec réseau de neurones.

Après un apprentissage hors-ligne du modèle inverse, le réseau de neurones identificateur peut être configuré afin d'assurer un contrôle direct du processus.

3. Application des réseaux de neurones à l'identification d'un modèle d'ordre entier

On a vu précédemment que, les réseaux de neurones servent aujourd'hui à toutes sortes d'applications dans divers domaines. Différentes structures de réseaux et algorithmes d'apprentissages, ont été ainsi développés pour des réseaux de taille plus modeste.

Dans ce paragraphe il s'agit d'appliquer la méthode l'identification neuronale directe à des modèles dynamiques d'ordre entiers, en utilisant des réseaux de neurones récurrents. En effet, dans la première partie on va appliquer le réseau de neurones multicouches retardé en entrée et en sortie, ensuite, dans la seconde partie on va mettre en application le réseau de neurones de type SSNN.

Les objectifs de cette identification sont :

1. Trouver des simulateurs neuronaux à des systèmes dynamiques linéaires de différents ordres entiers, c'est-à-dire de trouver des modèles neuronaux qui reproduisent au mieux le comportement du système sur un horizon infini.
2. d'évaluer les performances d'apprentissage du réseau.
3. d'évaluer la qualité d'estimation du réseau.

3.1. Identification par un réseau de neurones multicouches retardé en entrée et en sortie

Les réseaux de neurones multicouches sont utilisés dans différents domaines tels que : la reconnaissance de formes, le traitement du signal, filtrage, la reconnaissance vocale, l'aide à la décision ainsi que la simulation. Ils ont été aussi appliqués dans le domaine de l'automatique, soit pour la commande de système, contrôle du pendule inversé, soit pour l'identification de procédés.

Dans ce qui suit, on va identifier à l'aide d'un « réseau de neurones multicouches retardé en entrée et en sortie », deux modèles dynamiques d'ordre entier. Le premier modèle est un modèle fonction de transfert linéaire du premier ordre, et le second modèle est une fonction de transfert linéaire du quatrième ordre.

3.1.1. Identification d'un modèle linéaire du premier ordre :

Prenons l'exemple d'un système linéaire stable d'ordre un, qui est donné par la fonction de transfert suivante :

$$G(s) = \frac{0.5}{s+0.3} \tag{3.1}$$

À l'heure actuelle, l'immense majorité des applications des réseaux de neurones sont réalisées par des systèmes numériques (ordinateurs conventionnels ou circuits numériques spécialisés pour le traitement du signal) : il est donc naturel de se placer dans le cadre des systèmes *à temps discret,* régis par des « équations aux différences » (ou « équations récurrentes », d'où le terme de « réseaux récurrents »). Ces équations jouent le même rôle, en temps discret, que les équations différentielles en temps continu.

Si on discrétise le modèle donné par l'équation (3.1), en choisissant un pas d'échantillonnage de 0.2 seconde, on obtient la fonction de transfert en Z suivante :

$$G(z) = \frac{Y(z)}{U(z)} = \frac{0.09706}{z-0.9418} \tag{3.2}$$

Laquelle nous donne l'équation aux différences ci-dessous :

$$y(k) = 0.9418.\, y(k-1) + 0.09706.\, u(k-1) \tag{3.3}$$

Architecture choisie :

Souvent, l'entrée et la sortie sont représentatives de la dynamique d'un système, dans ce cas il faut en tenir compte explicitement pour réaliser l'apprentissage. Par conséquent, l'architecture du modèle neuronal multicouches, bouclé choisie, comporte : 2 entrées (l'entrée de commande et la sortie précédente estimée par le réseau), 3 neurones dans la couche cachée, 1

neurone dans la couche de sortie, les fonctions d'activation utilisées sont des fonctions linéaires ; cette architecture est montrée par la Figure 3.3 ci-dessous :

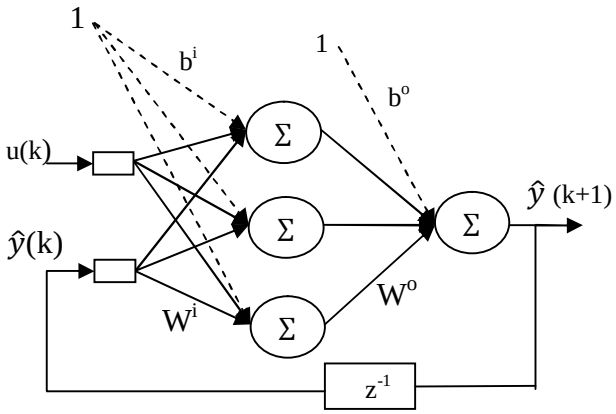


Figure 3.3 : Architecture du réseau de neurones multicouches retardé en sortie.

Les détails concernant l'apprentissage sont les suivants :

- La séquence de commande utilisée pour l'apprentissage est une suite de créneaux, d'amplitude aléatoire, comme le montre la Figure 3.4.a.
- La séquence de sortie utilisée pour l'apprentissage est générée à partir de l'équation aux différences qui est donnée par l'équation (3.3).
- La séquence de commande utilisée pour l'estimation de la performance du réseau (séquence de test) est une suite de créneaux d'amplitudes aléatoire complètement différente de la séquence d'apprentissage, on a choisi la duré des créneaux plus longue afin de vérifier la convergence au régime permanent, la Figure 3.4.b montre l'allure de cette entrée.

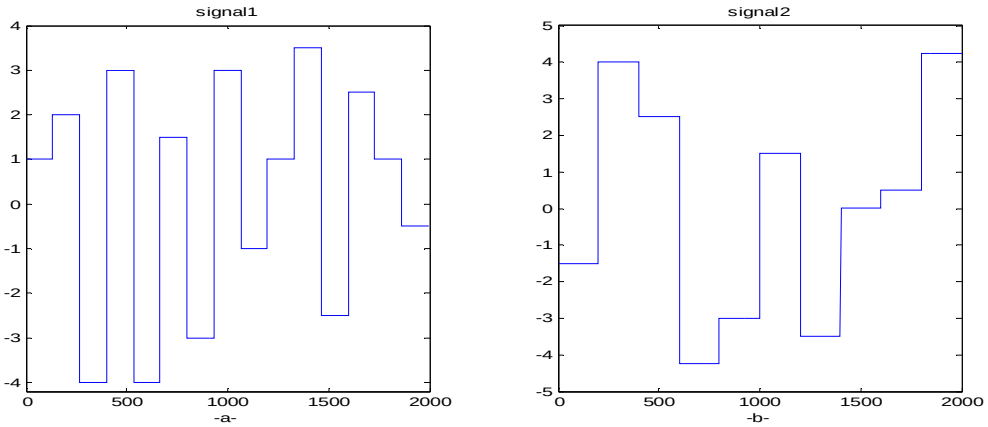


Figure 3.4 : Signal d'entrée ou de commande
-a- Séquence d'apprentissage (signal1) **-b-** Séquence du test (signal2).

La Figure 3.5 montre les résultats d'apprentissage et du test en utilisant un apprentissage hors ligne avec l'algorithme de L-Marquardt. Le pas d'apprentissage est fixé au début à 0.001.

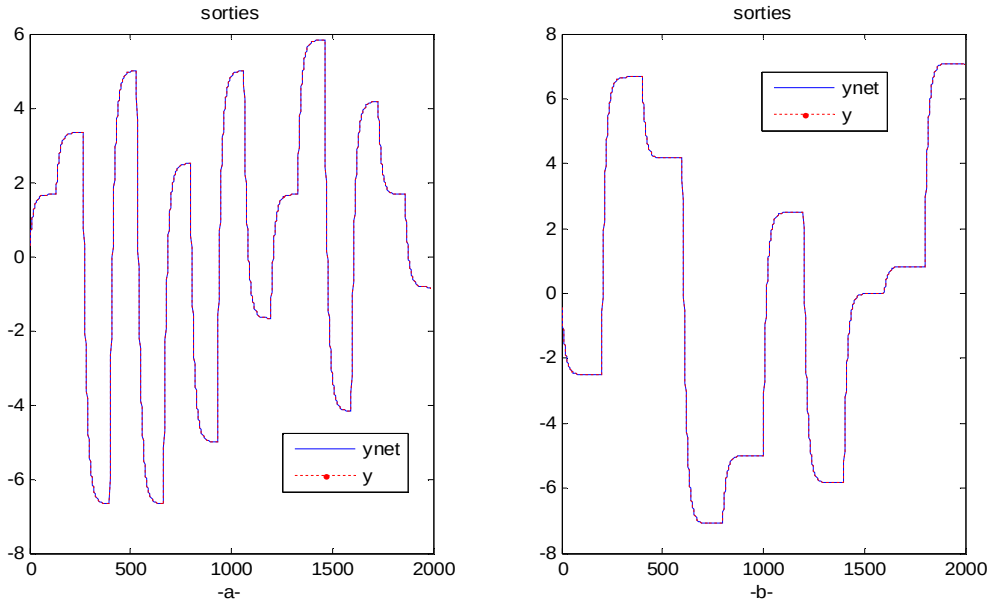


Figure 3.5 : Résultats de la sortie obtenus lors de l'apprentissage et du test
-a- Séquence d'apprentissage -b- Séquence de test.

Le Tableau (3.1) ci-dessous, rassemble les erreurs quadratiques moyennes, d'apprentissages notées (EQMA) et de tests notées (EQMT).

EQMA : Erreur quadratique moyenne de la sortie y sur l'ensemble d'apprentissage ;

EQMT : Erreur quadratique moyenne de la sortie sur l'ensemble du test.

	EQMA	EQMT	Itérations
Levenberg Marquardt	$7.0658\ 10^{-27}$	$4.2785\ 10^{-6}$	4

Tableau 3.1 : Erreur quadratique moyenne entre la sortie réelle et la sortie estimée.

D'après la Figures 3.5, on constate que les courbes de sortie estimée par le réseau de neurones et celle de sortie réelle du modèle, coïncident parfaitement, que soit dans la phase d'apprentissage, ou du test. A partir du Tableau (3.1), nous constatons également que les performances obtenues sont très satisfaisantes, bien qu'on a utilisé uniquement 3 neurones dans la couche cachée, et après seulement 4 itérations (Séquence d'apprentissage). Éventuellement on constate que l'EQMT est supérieure à celle de l'apprentissage, Cela s'explique par un léger sur-apprentissage des données d'entrée durant la séquence d'apprentissage du réseau de neurones.

3.1.2. Identification d'un modèle linéaire du quatrième ordre

Prenons maintenant, l'exemple d'un modèle linéaire d'ordre quatre, qui est donné par la fonction de transfert suivante :

$$G(s) = \frac{-5}{s^4+2.s^3+3.s^2+4.s+0.8} \tag{3.4}$$

La réponse indicielle de la fonction de transfert (3.4), représente une dynamique oscillatoire amortie avec un gain négatif, comme le montre la Figure 3.7, La transformée en Z bloquée de la relation (3.4), avec une période d'échantillonnage de 0,3 seconde, nous donne la fonction de transfert en Z suivante:

$$G(z) = \frac{-0.00149.z^3-0.01444.z^2-0.01281.z-0.00104}{z^4-3.306.z^3+4.245.z^2-2.483.z+0.5488} \tag{3.5}$$

La fonction de transfert (3.5) nous donne l'équation aux différences ci-dessous :

$$\begin{aligned} y(k) = {} & 3.306.y(k-1) - 4.245.y(k-2) + 2.483.y(k-3) \\ & -0.5488.y(k-4) \pm 0.00149.u(k-1) - 0.01444.u(k-2) \\ & -0.01281.u(k-3) - 0.00104.u(k-4) \end{aligned} \tag{3.6}$$

Architecture choisie :

L'architecture du « modèle neuronal multicouches retardé en entrée et en sortie » choisie est montrée par la Figure 3.6 ci-dessous. Cette dernière comporte : 8 entrées (4 entrées de commande et 4 sorties précédentes estimées par le réseau), 3 neurones linéaires dans la couche cachée, 1 neurone linéaire dans la couche de sortie.

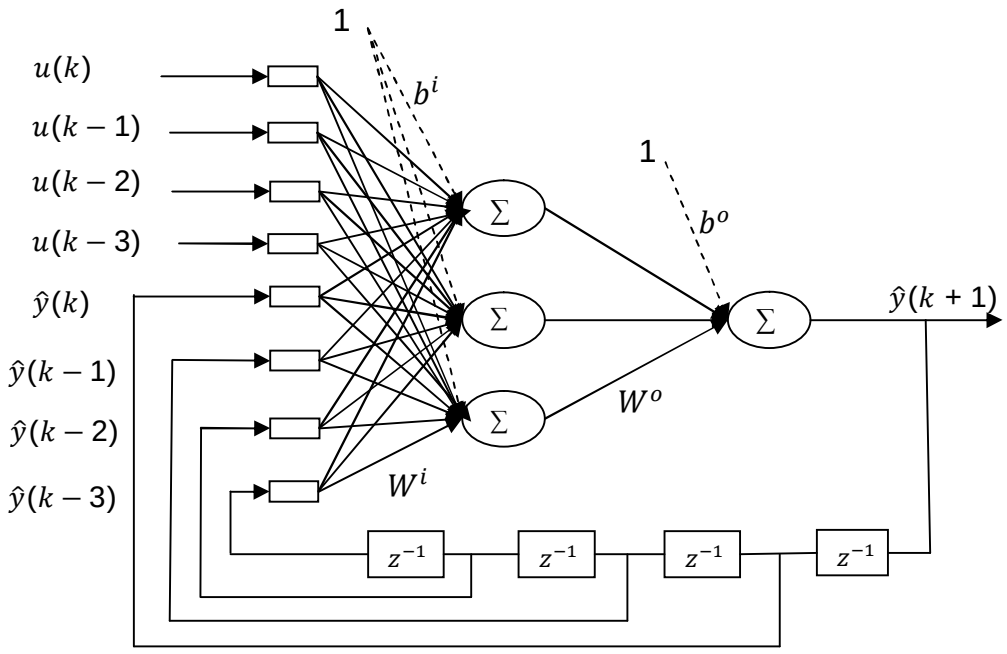


Figure 3.6 : Architecture du réseau de neurones multicouches retardé en entrée et en sortie.

Le signal d'entrée utilisé pour identifier le système est illustré par la Figure 3.4.a.

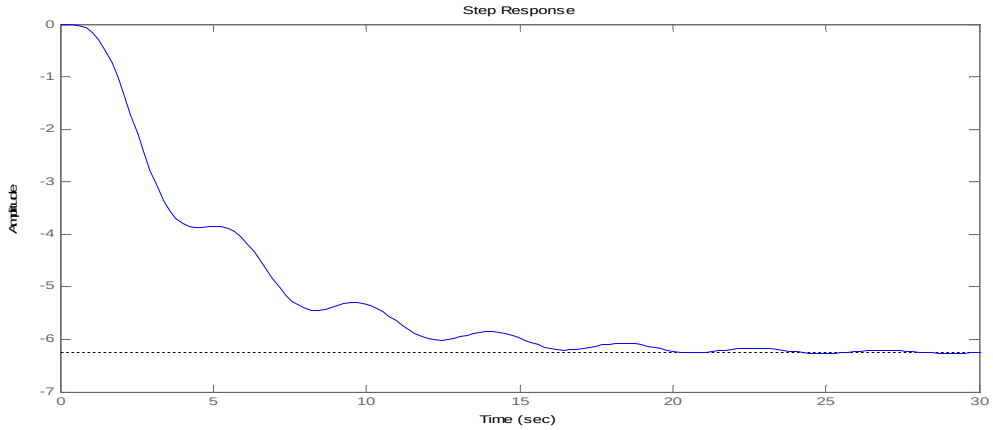


Figure 3.7 : Réponse indicielle de la fonction de transfert (3.4)

La Figure 3.8 montre les résultats d'apprentissage et du test, relatifs à l'apprentissage hors ligne avec l'algorithme de L-Marquardt, dont Le pas d'apprentissage est fixé au début à 0.001.

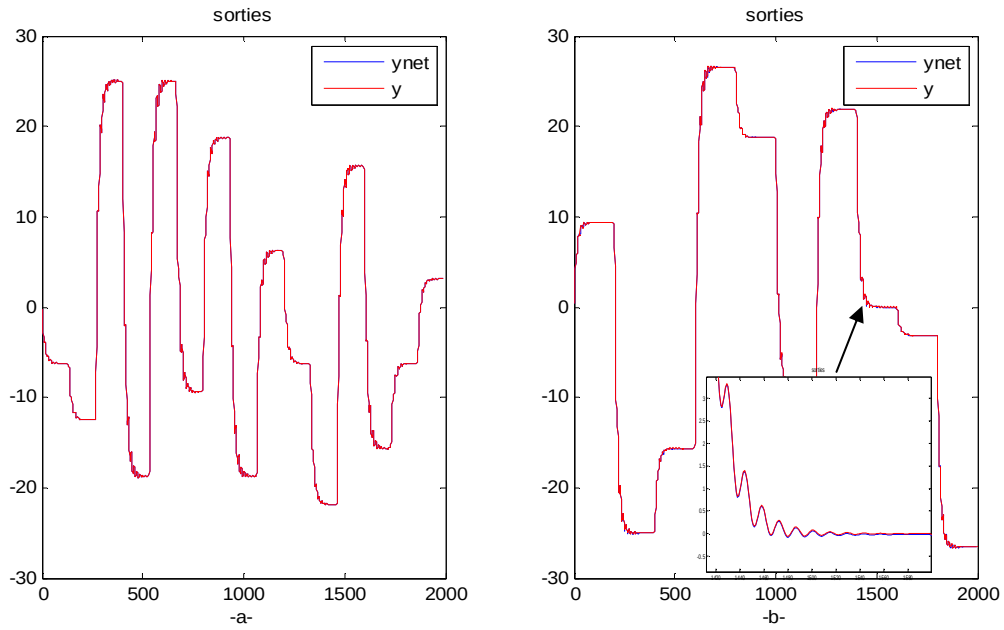


Figure 3.8 : Résultats de la sortie obtenus lors de l'apprentissage et du test
-a- Séquence d'apprentissage, **-b-** Séquence de test.

Le Tableau (3.2) ci-dessus rassemble les erreurs quadratiques moyennes, d'apprentissages notées (EQMA) et de tests notées (EQMT).

	EQMA	EQMT	Itérations
Levenberg Marquardt	$2.6885\ 10^{-21}$	$4.0482\ \ 10^{-4}$	8

Tableau 3.2: Erreur quadratique moyenne entre la sortie réelle et la sortie estimée.

D'après la Figure 3.8 on remarque une très bonne poursuite de la trajectoire désirée par la courbe de sortie estimée par le réseau, que se soit en phase d'apprentissage ou en phase du test. À partir du Tableau (3.2), on peut conclure que les erreurs de poursuite sont bornées par de faibles valeurs malgré qu'on a utilisé uniquement, 3 neurones dans la couche cachée, et après seulement 8 itérations. En comparant les Tableaux (3.1) et (3.2), on constate effectivement une légère augmentation des erreurs quadratiques avec le deuxième modèle, ceci s'explique physiquement par le fait que l'ordre du modèle à identifier a augmenté, ce qui complique la poursuite de sa dynamique.

En conclusion, on peut dire que le réseau de neurones multicouches retardé en entrée et en sortie, est capable d'identifier n'importe quel modèle dynamique linéaire d'ordre entier. La dynamique du système est ''captée'' à l'aide de lignes à retard. En revanche, la qualité de cette identification se verra diminuée au fur et à mesure que l'ordre du modèle dynamique à identifier augmente.

3.2. Identification par un réseau de neurones SSNN

Comme dans le cas précédent, on va identifier à l'aide d'un « réseau de neurones SSNN », deux modèles dynamiques d'ordre entier. Le premier modèle est un modèle d'état linéaire de premier ordre, et le second modèle est aussi un modèle d'état linéaire de quatrième ordre. Dans le but de comparer les résultats et les performances obtenus avec le réseau de neurones SSNN et ceux du réseau multicouches précédent, on choisira les mêmes exemples de modèles dynamiques utilisés précédemment.

3.2.1. Identification d'un modèle d'état linéaire d'ordre 1 :

Considérons le système donné par le modèle d'état linéaire du premier ordre suivante :

$$\begin{cases} x(k+1) = 0.9418.x(k) + 0.0971.u(k) \\ y(k) = 1.x(k) \end{cases} \tag{3.7}$$

L'équation (3.7), représente l'équation d'état discrète du modèle fonction de transfert (3.1), échantillonné avec une période d'échantillonnage Te = 0.2 sec.

L'architecture du modèle neuronal (SSNN) choisie, comporte : une entrée de commande $u(k)$, 2 neurones linéaires dans la première couche cachée, un neurone d'état linéaire, 2 neurones linéaires dans la deuxième couche cachée, et enfin un neurone linéaire dans la couche de sortie. On notera cette architecture par (1, 2, 1, 2 ,1) qui est montrée par la Figure 3.9.

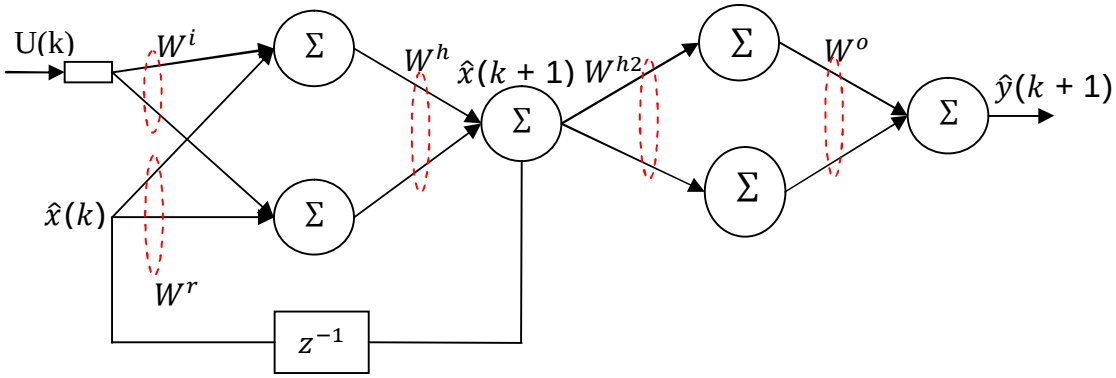


Figure 3.9: Architecture du SSNN (1, 2, 1, 2, 1)

A partir de cette architecture neuronale, on écrira la représentation d'état suivante :

$$\begin{cases} \hat{x}(k+1) = \Phi.\hat{x}(k) + \Gamma.u(k) \\ \hat{y}(k) = \Psi.\hat{x}(k) \end{cases} \quad (3.8)$$

Tel que : $\Phi = W^h * W^r$, $\Gamma = W^h * W^i$, $\Psi = W^o * W^{h2}$

Les résultats d'apprentissage et de test obtenus en utilisant un apprentissage hors ligne avec l'algorithme de L-Marquardt sont illustrés par la Figure 3.10. A partir de ces résultats, on constate une parfaite adéquation entre les courbes simulées et celles estimées ainsi que, un rejet des pics dus au changement de consigne, ce qui permet de conclure que les performances de poursuite sont très peu affectées par ces changements. Cela est dû à la faible sensibilité aux changements des données d'entrées du réseau.

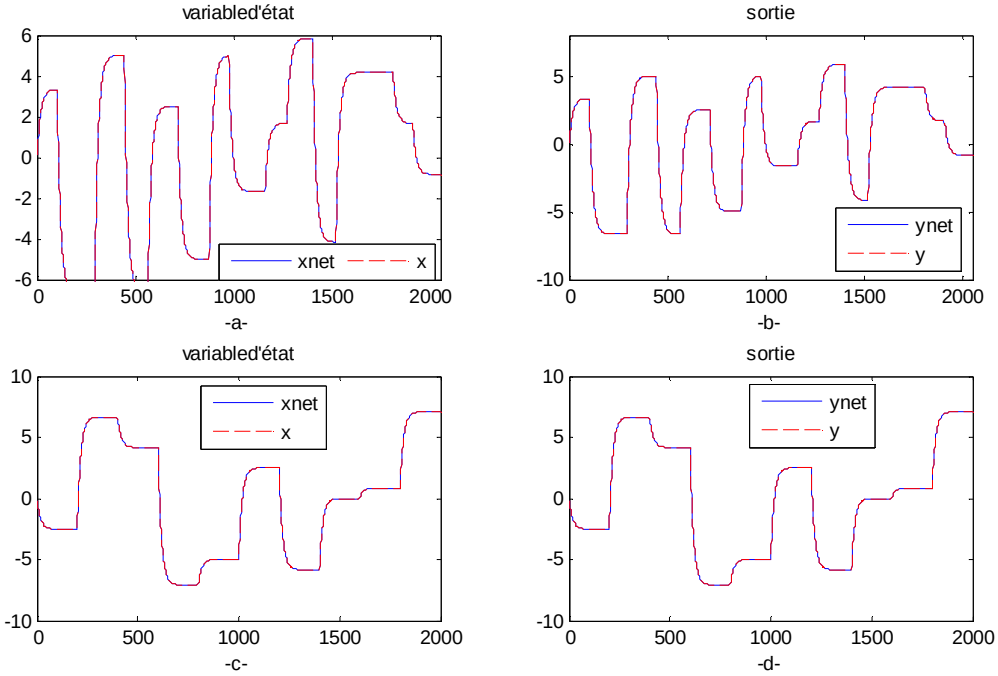


Figure 3.10 : Résultats obtenus avec le SSNN
-a- et -b- séquences d'apprentissages, **-c- et -d-** séquences de tests.

Le Tableau (3.3) ci-dessous rassemble les erreurs quadratiques moyennes, d'apprentissages notées (EQMA) et de tests notées (EQMT).

	EQMAx	EQMAy	EQMTx	EQMTy
Levenberg-Marquardt	$5.2591\ 10^{-13}$	$1.3849\ 10^{-23}$	$3.1318\ 10^{-15}$	$3.1317\ 10^{-15}$

Tableau 3.3 : erreur quadratique moyenne entre l'état (la sortie) estimé et l'état (la sortie) réel.

D'après le Tableau (3.3), nous constatons que les performances de poursuite obtenues sont excellentes, du fait que, les EQMTx et l'EQMTy sont de l'ordre de 10^{-15}, et cela est obtenu avec l'utilisation de deux neurones seulement dans chaque couche cachée, et après 4 itérations.

La comparaison des Tableaux (3.1) et (3.3), montre que l'utilisation de l'architecture SSNN a amélioré considérablement l'EQMTy, en la faisant passer de 10^{-6} à 10^{-15}.

Les matrices des poids obtenues après l'étape d'apprentissage sont les suivantes :

$W^i = \begin{bmatrix} -0.5380 \\ -0.0343 \end{bmatrix}$; $W^r = \begin{bmatrix} 0.1151 \\ 1.1538 \end{bmatrix}$; $W^h = [-0.2338 \quad 0.8396]$; $W^{h2} = \begin{bmatrix} -1.5007 \\ -0.5991 \end{bmatrix}$;
$W^o = [-0.5280 \quad -0.3466]$;

$\Phi = W^h * W^r = 0.9418, \Gamma = W^h * W^i = 0.0971, \Psi = W^o * W^{h2} = 1.0000$

En remplaçant les valeurs de Φ, Γ et Ψ dans l'équation (3.8) on obtient l'équation d'état suivante :

$$\begin{cases} \hat{x}(k+1) = 0.9048.\hat{x}(k) + 0.0952.u(k) \\ \hat{y}(k) = 2.\hat{x}(k) \end{cases} \tag{3.9}$$

Les matrices poids obtenues (Φ, Γ et Ψ) sont identiques aux matrices d'états du modèle réel.

3.2.2. Identification d'un modèle d'état linéaire d'ordre 4 :

Dans le but de compliquer encore d'avantage l'identification, on prend l'exemple d'un modèle d'état du quatrième ordre, ayant pour représentation d'état l'équation suivante :

$$\begin{cases} x(k+1) = \begin{bmatrix} 0.4462 & -0.3961 & -0.2218 & -0.08583 \\ 0.4291 & 0.8754 & -0.07422 & -0.02907 \\ 0.1454 & 0.5745 & 0.9844 & -0.006153 \\ 0.007691 & 0.04403 & 0.1494 & 0.9998 \end{bmatrix} x(k) + \begin{bmatrix} 0.4291 \\ 0.1454 \\ 0.03076 \\ 0.001192 \end{bmatrix} u(k) \\ y(k) = [0 \quad 0 \quad 0 \quad -1.25]\ x(k) \end{cases} \tag{3.10}$$

L'équation (3.10), représente l'équation d'état discrète de modèle fonction de transfert (3.4), échantillonné avec une période d'échantillonnage de 0.3 sec.

La Figure 3.11 montre l'architecture du modèle neuronal SSNN (1, 4, 4, 2, 1) qui correspond à ce système. Cette architecture est composée de : une entrée de commande $u(k)$, 4 neurones linéaires dans la première couche cachée, 4 neurones d'états linéaires, 2 neurones

linéaires dans la deuxième couche cachée, et enfin un neurone linéaire dans la couche de sortie.

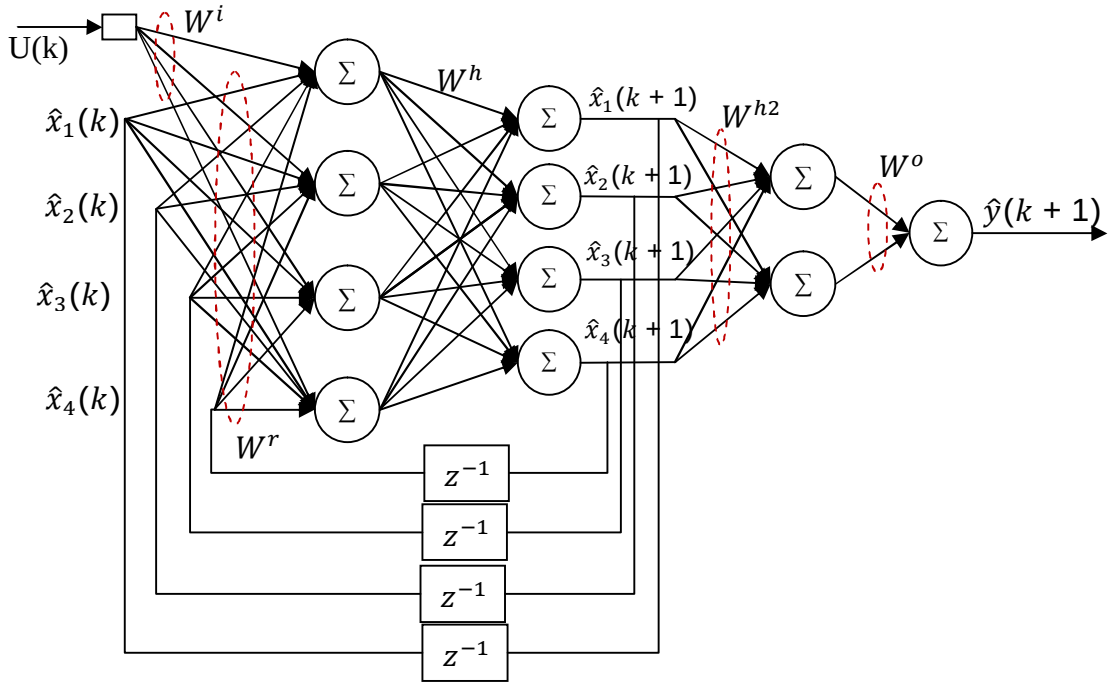
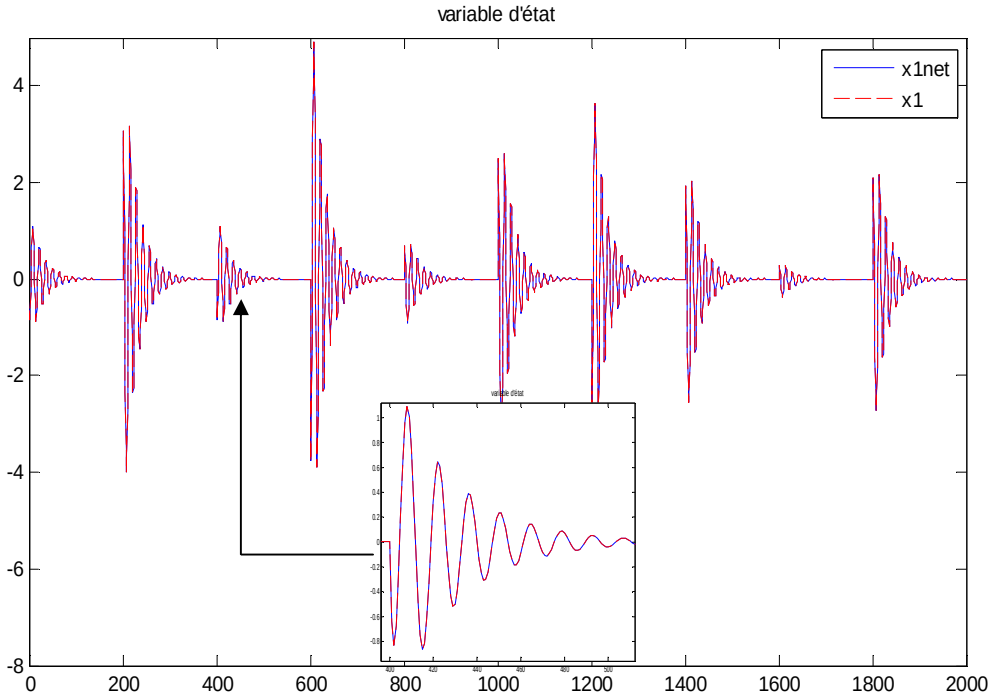
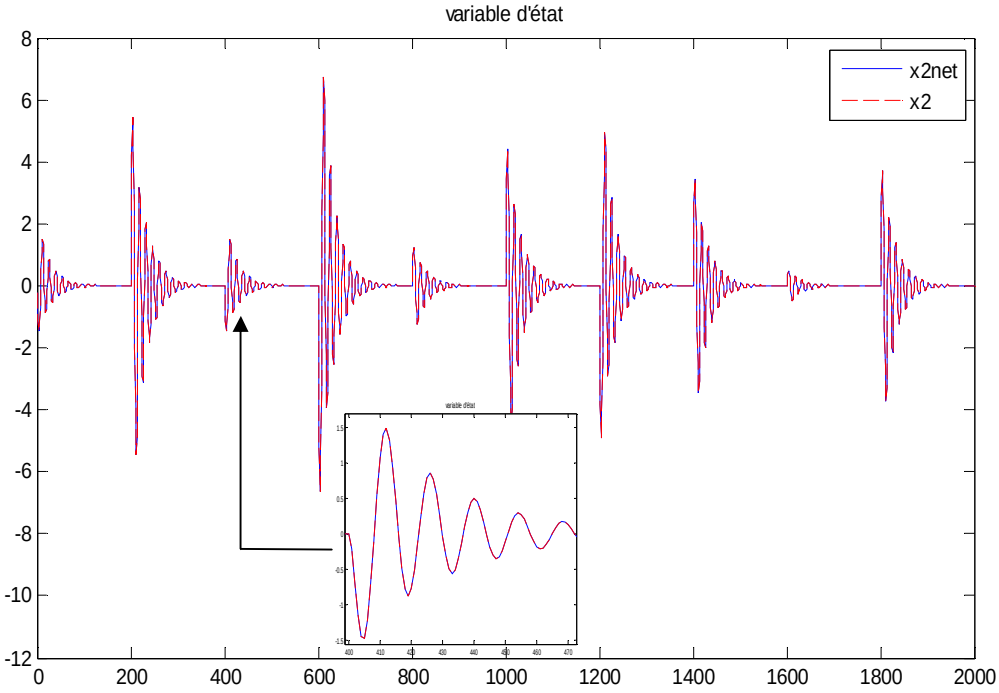


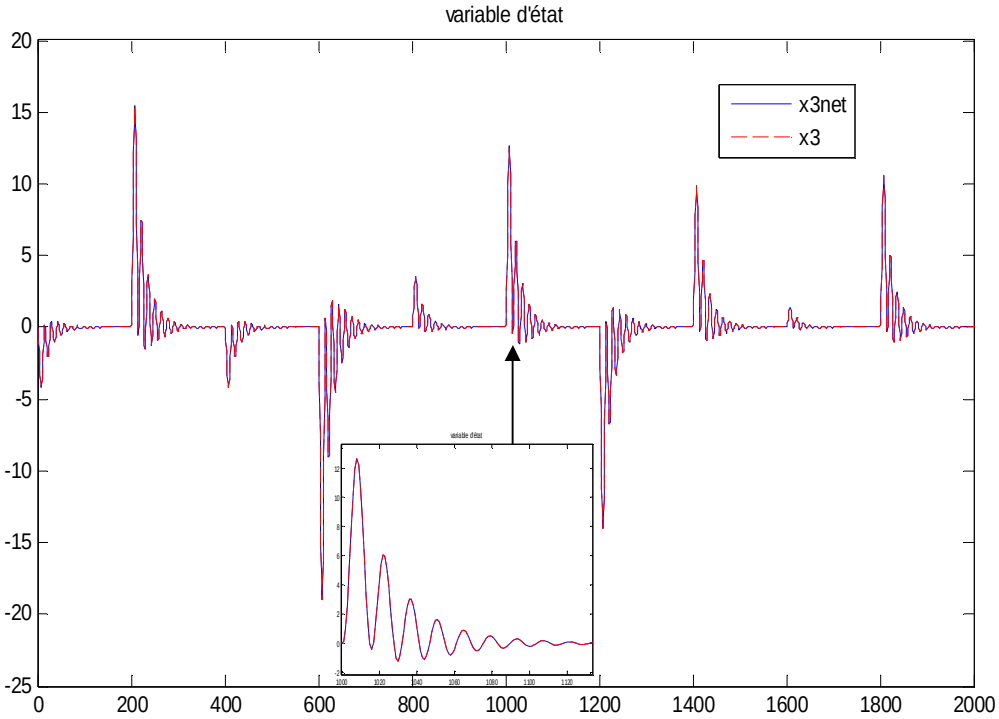
Figure 3.11 : Architecture du SSNN (1, 4, 4, 2, 1).

Les résultats :

La Figure 3.12 montre les résultats du test du SSNN (1, 4, 4, 2, 1) après 6 itérations en utilisant un apprentissage hors ligne avec l'algorithme de L-Maquardt.




variable d'état
x2net
x2


variable d'état
x3net
x3

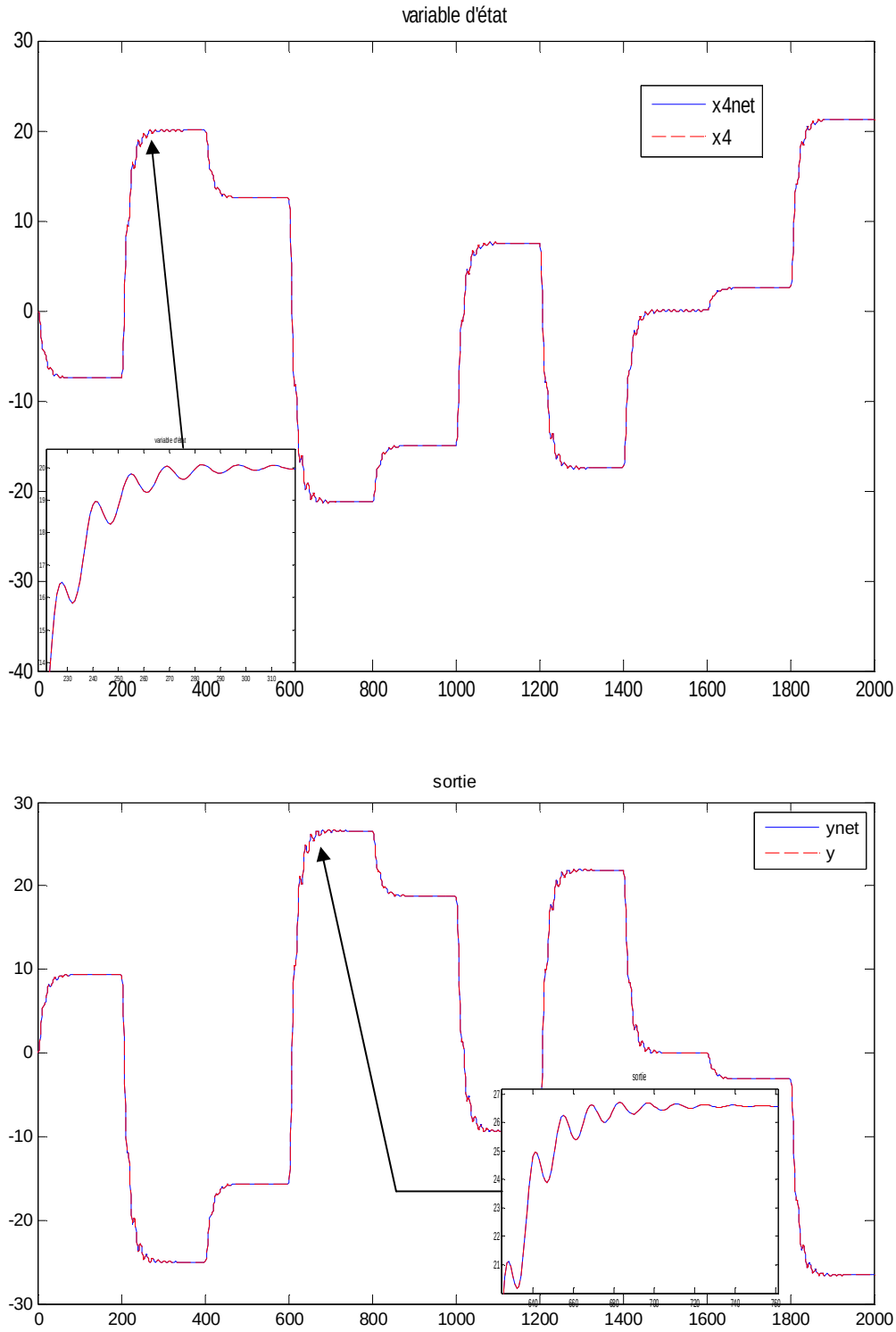


Figure 3.12: résultats du test obtenus avec le SSNN (1, 4, 4, 2, 1)

Le Tableau (3.4) et le Tableau (3.5) ci-dessous rassemblent les erreurs quadratiques moyennes, d'apprentissages notées (EQMA) et de tests notées (EQMT).

	EQMAx1	EQMAx2	EQMAx3	EQMAx4	EQMAy
L-M	$4.0205\ 10^{-9}$	$8.8058\ 10^{-9}$	$1.8122\ 10^{-8}$	$1.1818\ 10^{-9}$	$1.7439\ 10^{-24}$

Tableau 3.4 : Erreurs quadratique moyenne d'apprentissage entre les états estimés et les états réels, la sortie estimée et la sortie réelle.

	EQMTx1	EQMTx2	EQMTx3	EQMTx4	EQMTy
L-M	$1.6403\ 10^{-10}$	$1.3892\ 10^{-9}$	$2.3289\ 10^{-10}$	$3.0783\ 10^{-10}$	$4.8099\ 10^{-10}$

Tableau 3.5 : Erreurs quadratique moyenne du test entre les états estimés et les états réels, la sortie estimée et la sortie réelle.

D'après la Figure 3.12 et les Tableaux (3.4) et (3.5), et malgré une dynamique très oscillatoire des d'état, les performances obtenues sont excellentes, et cela est obtenu avec seulement 4 neurones dans la première couche cachée et de 2 neurones dans la deuxième couche cachée. En comparant le Tableau (3.5) avec le Tableau (3.2) on constate une nette amélioration de l'EQMTy, puisque elle passe de 10^{-4} à 10^{-10}, grâce à l'utilisation du réseau SSNN.

Les matrices poids obtenues après apprentissage avec la méthode de L- Marquardt sont les suivantes :

$$W^i = \begin{bmatrix} 0.5987 \\ -0.5633 \\ 0.2464 \\ 0.0581 \end{bmatrix},\ W^r = \begin{bmatrix} -0.1294 & -2.1378 & 5.6936 & -0.1692 \\ -1.1399 & -0.8029 & 3.8718 & -0.7796 \\ -0.2295 & -1.6021 & 1.9418 & 0.7672 \\ 0.5957 & 1.6038 & -1.5000 & 0.9142 \end{bmatrix};$$

$$W^h = \begin{bmatrix} 0.3269 & -0.5447 & -0.2252 & -0.3089 \\ 0.3428 & -0.0897 & -0.5455 & 0.4129 \\ 0.3114 & 0.1702 & -0.3604 & 0.4985 \\ -0.1881 & 0.1797 & 0.7319 & 0.5979 \end{bmatrix};\ W^{h2} = \begin{bmatrix} -0.6682 & -0.1920 & 0.3197 & 1.0647 \\ 2.4839 & 0.7139 & -1.1884 & 2.4555 \end{bmatrix};$$

$$W^0 = [-0.7245 \quad -0.1949];$$

Là aussi le modèle neuronal peut être mis sous la représentation d'état avec les valeurs numériques obtenues après apprentissage, $\Phi = W^{\boldsymbol{h}}.W^r$; $\Gamma = W^{\boldsymbol{h}}.W^i$; $\Psi = W^{\boldsymbol{o}}.W^{h2}$

$$\begin{cases} \hat{x}(k+1) = \begin{bmatrix} 0.4462 & -0.3961 & -0.2218 & -0.08583 \\ 0.4291 & 0.8754 & -0.07422 & -0.02907 \\ 0.1454 & 0.5745 & 0.9844 & -0.006153 \\ 0.007691 & 0.04403 & 0.1494 & 0.9998 \end{bmatrix}.\hat{x}(k) + \begin{bmatrix} 0.4291 \\ 0.1454 \\ 0.03076 \\ 0.001192 \end{bmatrix}.u(k) \\ \hat{y}(k) = [\ 0 \quad 0 \quad 0 \quad -1.25\ ].\hat{x}(k) \end{cases} \tag{3.11}$$

La représentation d'état obtenue avec les poids optimaux du réseau SSNN qui est donné par (3.11), est exactement la même que celle de la représentation du système de l'équation (3.10) ; donc un réseau de neurone à espace d'état (SSNN) peut reproduire parfaitement le comportement de n'importe quel système linéaire.

D'après les résultats obtenus, on peut conclure que pour l'identification des modèles linéaires entiers, le réseau SSNN est beaucoup plus performant que le réseau multicouches retardé en entrée et en sortie, du fait que le réseau SSNN en plus de son identification de la

sortie du modèle, il arrive à identifier aussi les différentes variables d'états. Ce qui lui permet d'accéder à la dynamique internes du modèle.

4. Application des réseaux de neurones à l'identification de modèles d'ordre non entier

Dans ce paragraphe il s'agit de proposer deux architectures neuronales récurrentes ensuite les appliquer à l'identification des modèles d'ordre fractionnaires. Ces deux architectures, seront inspirées à partir des deux architectures neuronales, appliquées auparavant pour identifier des modèles d'ordre entier.

Dans la première partie on va appliquer le réseau de neurones de type multicouches retardé en entrée et en sortie pour identifier un modèle fonction de transfert d'ordre fractionnaire, ensuite, dans la seconde partie on va appliquer le réseau de neurones de type SSNN pour identifier un modèle d'équation d'état d'ordre non entier.

Comme précédemment les objectifs de cette identification sont :

1. Trouver des simulateurs neuronaux à des systèmes fractionnaires, c'est-à-dire de trouver des modèles neuronaux qui reproduisent au mieux le comportement du système fractionnaire sur un horizon infini.
2. Evaluer les performances d'apprentissage des deux architectures neuronales récurrentes.
3. D'évaluer la qualité d'estimation déduite des deux architectures neuronales récurrentes.

4.1. Identification par un réseau de neurones multicouches retardé en entrée et en sortie

On a vu que le « réseau de neurones multicouches retardé en entrée et en sortie », parvient facilement à identifier les différents modèles dynamiques d'ordre entier, et reproduire fidèlement leurs dynamiques et pour des entrées quelconques et avec une bonne précision. Malheureusement, on a vu précédemment que dès qu'on augmente la dimension du modèle, la performance du « réseau multicouches retardé en entrée et en sortie » diminue avec, ce qui réduit considérablement la possibilité d'identifier les systèmes fractionnaires avec cette architecture, étant donné que ces derniers sont de dimension infinie, c'est-à-dire à mémoire infinie, par conséquent on est ramené à effectuer quelques modifications dans cette architecture, afin que cette dernière puisse représenter la dynamique fractionnaire des systèmes d'ordre non entier.

Comme précédemment, on va identifier à l'aide de ce réseau de neurones deux modèles dynamiques d'ordre fractionnaire. Le premier modèle est un modèle fonction de transfert linéaire de dimension un, et le second est de dimension deux.

4.1.1. Identification d'un modèle fractionnaire linéaire de dimension 1

Prenons l'exemple d'un système linéaire stable d'ordre non entier strictement propre, de dimension un, donné par la fonction de transfert suivante :

$$G(s) = \frac{0.5}{s^{1.25}+1} \tag{3.12}$$

Les détails concernant l'apprentissage sont les suivants :

- La séquence de commande utilisée pour l'apprentissage est une suite de créneaux d'amplitude et de durée aléatoire, la séquence comporte 3000 pas d'échantillonnage; elle est représentée par la Figure 3.13.a.
- La séquence de sortie utilisée pour l'apprentissage et du test, est générée à partir de la simulation de la fonction de transfert fractionnaire (3.12).
- La séquence de commande utilisée pour l'évaluation de la performance du réseau (séquence du test) est une suite de dix créneaux d'amplitude aléatoire, la séquence comporte aussi 3000 pas d'échantillonnage, elle est représentée par la Figure 3.13.b.

La simulation du modèle (3.12) et réalisée avec la méthode d'approximation de Grünwald-Letnikov avec un pas d'échantillonnage de Te= 0.05 secondes.

Le temps de simulation du modèle (3.12), en lui appliquant la séquence d'apprentissage et la séquence du test, égale à 711.3 secondes (11.85 minutes).

Sachant que cette simulation est réalisée avec un micro ordinateur qui possède les caractéristiques suivante :

- Microprocesseur Pentium (R) 4CPU 2.80GHz.
- RAM de 480 Mo.

Architecture proposée:

Du fait que, les systèmes fractionnaires possèdent une mémoire longue, il est alors nécessaire de tenir compte des sorties et des entrées précédentes du réseau. Par conséquent, l'architecture du « modèle neuronal multicouches retardé en entrée et en sortie proposé» est donnée par la Figure 3.14, cette dernière comporte : 10 entrées (5 entrées précédentes de l'entrée de commande et 5 sorties précédentes du réseau), 3 neurones dans la couche cachée, 1 neurone dans la couche de sortie ; tous les neurones utilisés dans la couche cachée et dans la couche de sortie possèdent des fonctions linéaires.

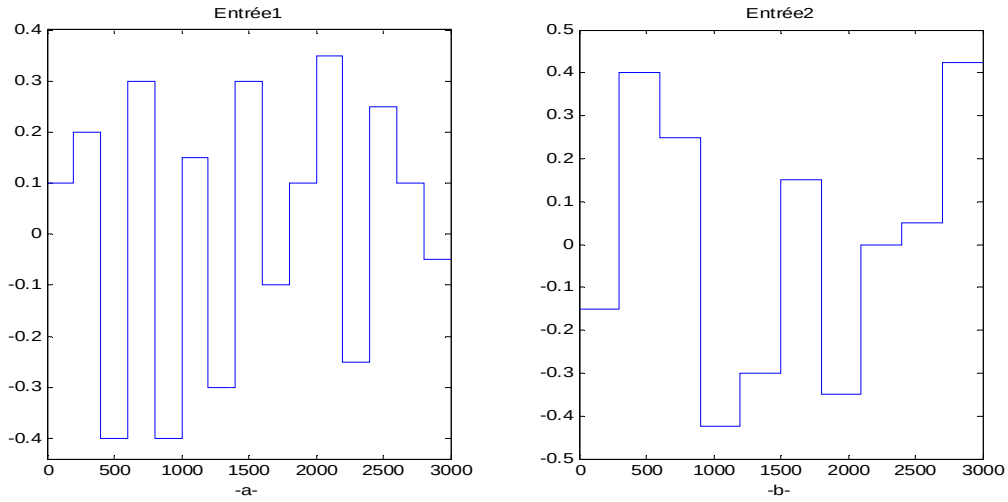


Figure 3.13 : Signal d'entrée ou de commande.
-a- Séquence d'apprentissage, **-b-** Séquence du test.

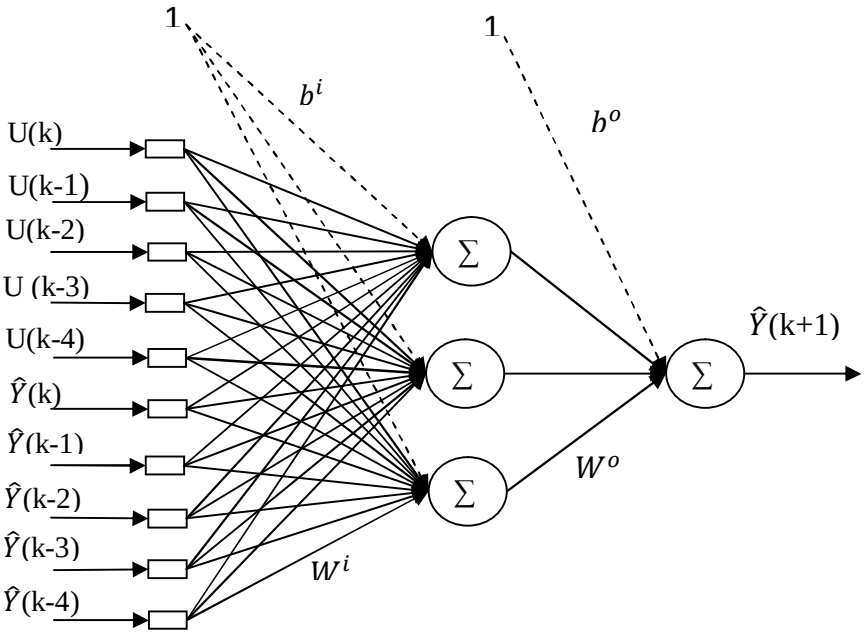


Figure 3.14 : Architecture du réseau de neurones multicouches retardé en entrée et en sortie.

Le temps de simulation nécessaire pour simuler le modèle (3.12) avec ce réseau de neurones en lui appliquant en entrée la séquence de test, est égale seulement à 8.21 secondes.

La Figure 3.15, montre les résultats d'apprentissage et de validation avec à un apprentissage hors ligne en utilisant l'algorithme de L-Marquardt.

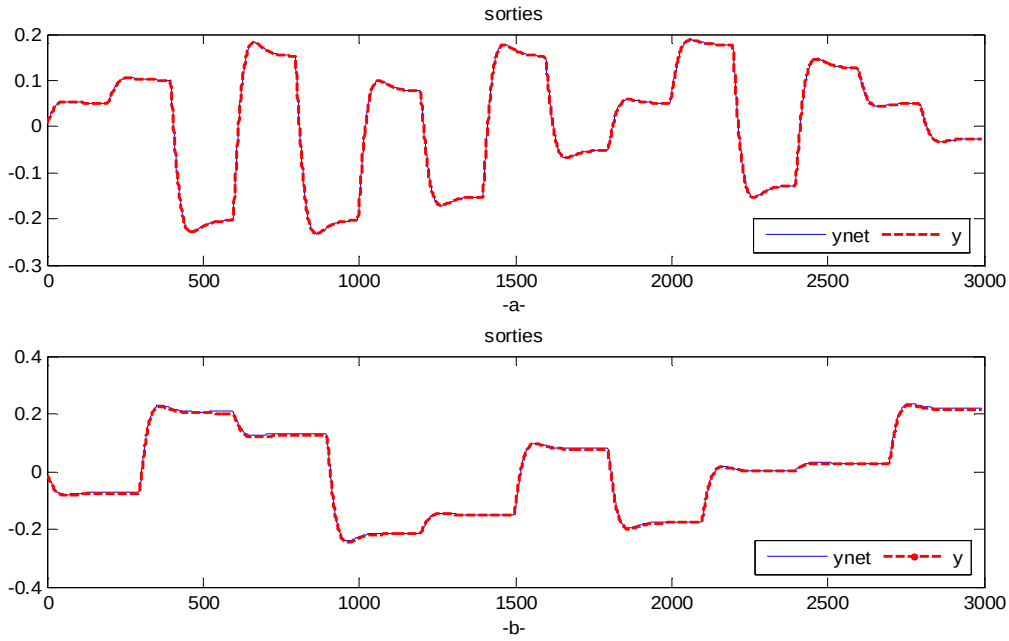


Figure 3.15 : Résultats de la sortie obtenus lors de l'apprentissage et de test
-a- Séquence d'apprentissage. **-b-** Séquence de test.

La Figure 3.16 montre l'allure des erreurs d'apprentissage et de validation entre la sortie désirée et la sortie estimée.

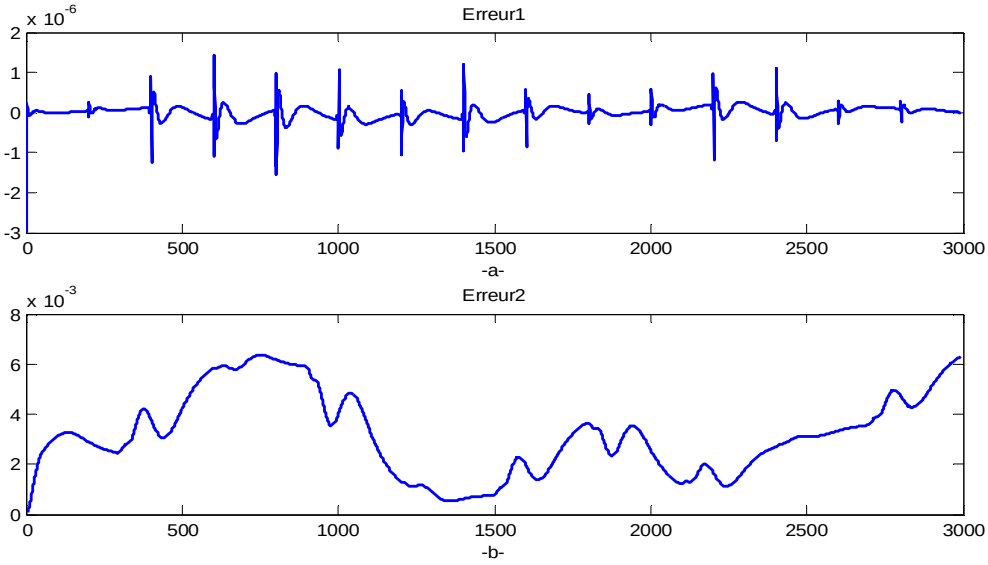


Figure 3.16: Erreurs obtenus lors de l'apprentissage et de test
-a- Erreur d'apprentissage. **-b-** Erreur de test.

La Figure 3.16, illustre les profils des erreurs de poursuite en phase d'apprentissage et de test. Ces résultats montrent des qualités de poursuite satisfaisantes. En effet, nous constatons d'une part que les erreurs de poursuite sont bornées par de faibles valeurs et d'autre part ces courbes présentent quelques fluctuations qui sont dues aux changements

brusques des valeurs du signal d'entrée. Il est toutefois essentiel de noter que les phénomènes non modélisés dégradent la qualité des résultats.

Le Tableau (3.6) ci-dessous rassemble les erreurs quadratiques moyennes, d'apprentissages notées (EQMA) et de validations notées (EQMT).

	EQMA	EQMT	Itérations
Levenberg-Marquardt	$1.6988*10^{-14}$	$1.2661*10^{-5}$	10000

Tableau 3.6 : Erreur quadratique moyenne entre la sortie réelle et la sortie estimée.

D'après la Figure 3.15, on constate que la sortie estimée par le réseau de neurones suit très bien la sortie du modèle fractionnaire (3.12), même dans la phase de validation. On constate également l'absence des pics dus aux changements brusque de l'entrée. À partir du Tableau (3.6), on peut déduire aussi que les performances de poursuites obtenues avec 3 neurones dans la couche cachée, sont très bonnes, néanmoins, l'apprentissage a nécessité 10000 itérations.

Bien qu'on a choisit une structure neuronale très simple, qui utilise uniquement 5 valeurs passées de l'entrée et de la sortie estimée par le réseau de neurones, ce dernier a réussi à identifier la dynamique très complexe du modèle fractionnaire. En effet, ce réseau de neurones, est parvenu durant la phase d'apprentissage à réajuster ses poids de connexion de tel sort à compenser l'absence de tout l'historique du modèle fractionnaire.

4.1.2. Identification d'un modèle fractionnaire linéaire de dimension 2

Prenons maintenant un deuxième exemple plus complexe, qui est donné par la relation (3.13) ; il s'agit d'un modèle d'ordre non entier, non commensurable, strictement propre et de dimension deux:

$$G(s) = \frac{s^{1.25}+4.s^{1.23}+8}{s^{2.48}+2.s^{1.25}+1} \tag{3.13}$$

La simulation du modèle (3.13), est effectuée également avec la méthode d'approximation de Grünwald-Letnikov, en utilisant un pas d'échantillonnage Te= 0.05 sec.

Les entrées de commande sont identiques à celles appliquées précédemment (voir la Figure 3.13).

Le temps de simulation du modèle (3.13), en lui appliquant en entrée la séquence d'apprentissage et la séquence du test, est égale à 1071.36 secondes (17 minutes).

Architecture proposée:

L'architecture du « modèle neuronal multicouches retardé en entrée et en sortie proposée», est illustrée par la Figure 3.17 ci-dessous, qui comporte : 10 entrées (5

entrées de commande retardées et 5 sorties précédentes calculées par le réseau), 3 neurones dans la couche cachée, 1 neurone dans la couche de sortie; tous les neurones utilisés possèdent une fonction de transfert linéaire.

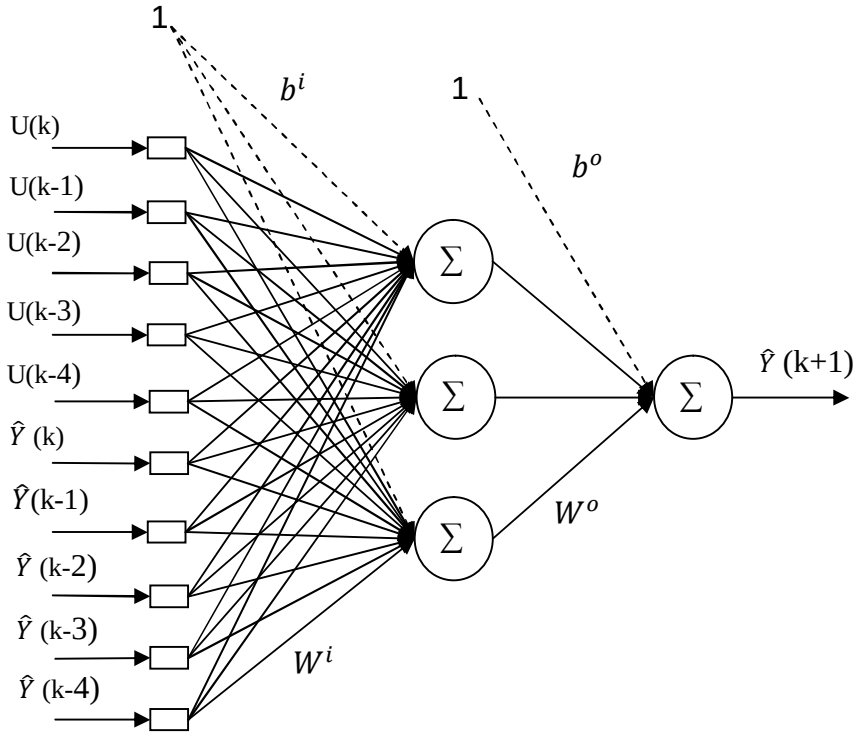


Figure 3.17 : Architecture du réseau de neurones multicouches retardé en entrée et en sortie.

L'application de cette architecture à l'identification d'un modèle fractionnaire donné par l'équation (3.13) permet d'obtenir les performances montrées sur les Figures (3.18 et 3.19). La Figure 3.18 représente les résultats d'apprentissage et du test en utilisant un apprentissage hors ligne avec l'algorithme de L-Marquardt. Quant à la Figure (3.19) montre l'allure des erreurs d'apprentissage et du test entre la sortie désirée et la sortie estimée. Le Tableau (3.7) rassemble les erreurs quadratiques moyennes, d'apprentissages notées (EQMA) et de tests notées (EQMT).

Le temps de simulation nécessaire pour simuler le modèle (3.13) avec ce réseau de neurones en lui appliquant en entrée, la séquence du test, est égale seulement à 8.25 secondes.

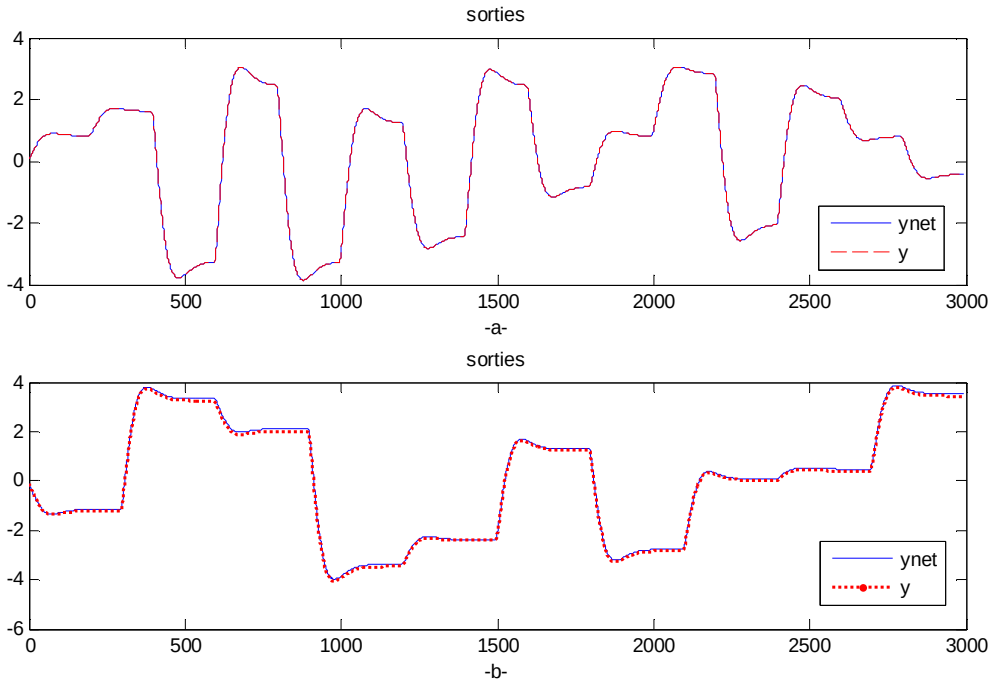


Figure 3.18 : Résultats de la sortie obtenus lors de l'apprentissage et du test
-a- Séquence d'apprentissage. **-b-** Séquence de test.

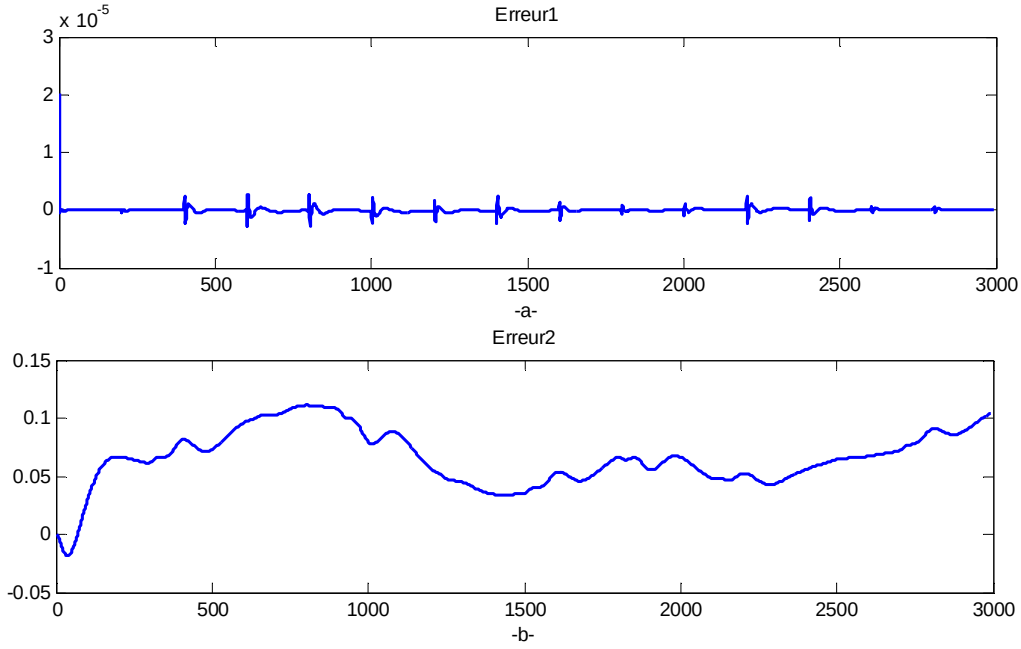


Figure 3.19 : Erreurs obtenus lors de l'apprentissage et du test
-a- Erreur d'apprentissage. **-b-** Erreur du test.

	EQMA	EQMT	Itérations
Levenberg Marquardt	$2.6516*10^{-13}$	0.0049	4171

Tableau 3.7: Erreur quadratique moyenne entre la sortie réelle et la sortie estimée.

On conclue que après une période d'adaptation que :

- Ces résultats graphiques permettent une très bonne poursuite de la sortie du modèle non entier par la sortie du réseau de neurones dans la phase d'apprentissage, et aussi une bonne poursuite dans la séquence de validation mais avec un très léger décalage entre les deux courbes au niveau du régime permanant.
- La dynamique des erreurs de poursuite dans les phases appropriées à l'apprentissage et au test, présentent des fluctuations causées par un changement brusque de la valeur du signal d'entrée.
- Les profils des erreurs de poursuite sont bornés par de faibles valeurs.
- Malgré l'utilisation de seulement 3 neurones linéaires dans la couche cachée, le réseau neurones multicouches est parvenu à des résultats satisfaisants. En revanche, on comparant le Tableau (3.7) et le Tableau (3.6), on constate une augmentation de l'erreur. Ceci s'explique par le fait que la dimension du modèle a augmenté, ce qui complique un petit peu l'identification de la dynamique du modèle par le réseau le neurones multicouches retardé en entrée et en sortie.
- Malgré une grande complexité de la dynamique des modèles fractionnaires, leur identification avec des « réseaux récurrents multicouches retardé en entrée et en sortie », s'est avéré finalement, simple et efficace, puisque, la structure du réseau utilise seulement une seule couche cachée composée de quelques neurones et un seul neurone dans la couche de sortie. Les fonctions d'activations des neurones étant linéaires.
- On constate aussi que le temps de simulation du modèle fractionnaire avec le nouveau : « réseau de neurones multicouches retardé en entrée et en sortie », se fait en un temps beaucoup plus court, comparé à celui de la méthode d'approximation de Grünwald-Letnikov, ce qui est un grand avantage pour la simulation de ce type de modèles.

4.2. Identification par un réseau de neurones SSNN

On a vu précédemment, que la structure des réseaux de neurones à espace d'état, convient parfaitement pour identifier des systèmes linéaires décrits par une représentation d'état classique (d'ordre entier). Par conséquent, on va s'inspirer de cette structure neuronale, afin de réaliser une nouvelle architecture adaptée à l'identification d'un système décrit par un modèle d'état d'ordre non entier.

Afin de comparer les résultats et les performances du réseau de neurones SSNN proposé, avec ceux du « réseau multicouches retardé en entrée et en sortie» proposé pour

identifier les modèles fractionnaire, on va utiliser les mêmes modèles fractionnaires (3.12) et (3.13).

4.2.1. Identification d'un modèle d'état non entier à 1 variable l'état:

Prenant le même modèle fractionnaire de dimension 1, déjà donné par la relation (3.12). Sa représentation d'état s'exprime comme suite :

$$\begin{cases} D^{1.25}x(t) = -1.x(t) + 2.u(t) \\ y(t) = 0.25.x(t) \end{cases} \tag{3.14}$$

Cette représentation d'état peut être vérifiée en utilisant la relation donnée par (1.58).

Ce modèle d'état non entier possède une seule variable d'état, dérivée à d'ordre fractionnaire 1.25. La simulation du modèle d'état (3.14) est réalisée également par la méthode d'approximation de Grünwald-Letnikov, en utilisant un pas d'échantillonnage Te= 0.05 sec.

Architecture proposée :

D'après le modèle d'état fractionnaire (3.14), on constate que l'opérateur fractionnaire s'opère uniquement sur l'équation d'état du modèle, c'est-à-dire sur les d'états et les entrées de commande, alors, il est essentiel de tenir compte de l'historique des variables d'état ainsi que des entrées de commande dans la conception de l'architecture du réseau SSNN. Cette architecture s'inspire elle aussi du réseau SSNN.

L'architecture du modèle neuronal (SSNN) proposée : comporte 5 entrées de commande, 3 neurones linéaires dans la première couche cachée, un neurone d'état linéaire, 2 neurones linéaires dans la deuxième couche cachée, et enfin un seul neurone linéaire dans la couche de sortie. La couche d'entrée tient compte aussi de l'historique de l'état du modèle neuronale. On notera cette architecture par (1, 3, 1, 2 ,1). L'entrée de commande et la variable d'état sont retardées cinq fois, comme le montre la Figure 3.20.

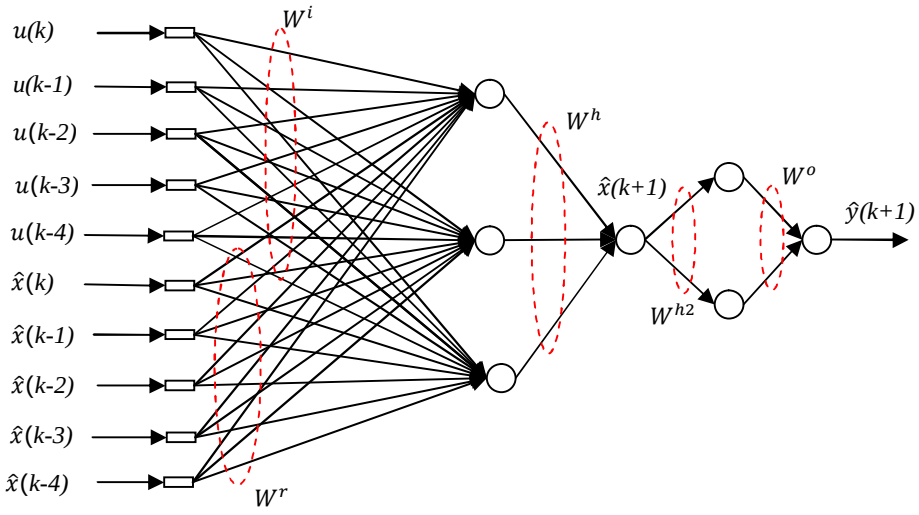


Figure 3.20: Architecture du SSNN (1, 3, 1, 2, 1) modifié.

Les signaux de commande sont identiques à ceux appliqués dans le chapitre précédent (voir la Figure 3.13).

La Figure 3.21, montre le résultat d'apprentissage pour le réseau de neurones SSNN (1, 3, 1, 2, 1) modifié, en utilisant le signal d'apprentissage de la Figure 3.13-a-

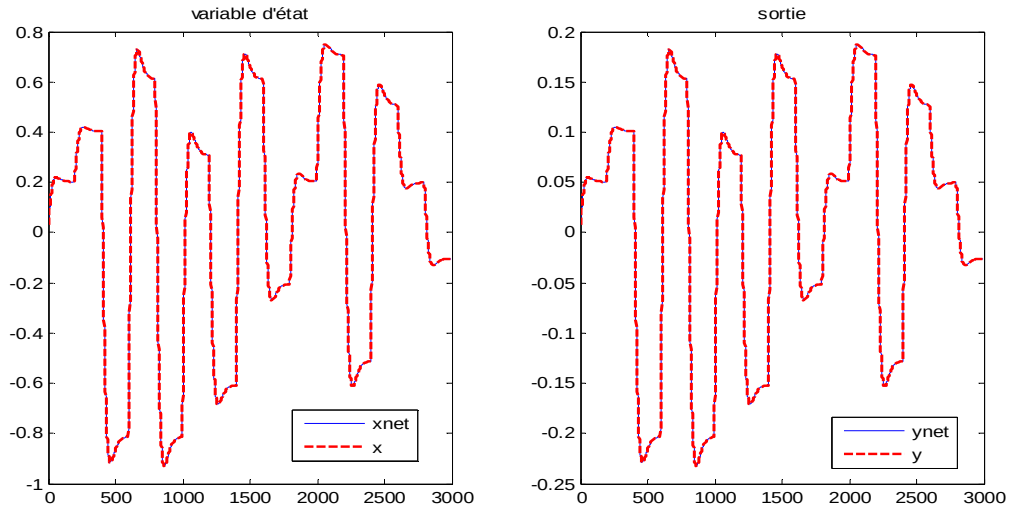


Figure 3.21 : Résultats d'apprentissage obtenus avec le SSNN (1, 3, 1, 2, 1) modifié.

La Figure 3.22, illustre les essais en simulation (test) du réseau de neurones (1, 3, 1, 2, 1) en utilisant le signal de commande, donné par la Figure 3.13-b-.

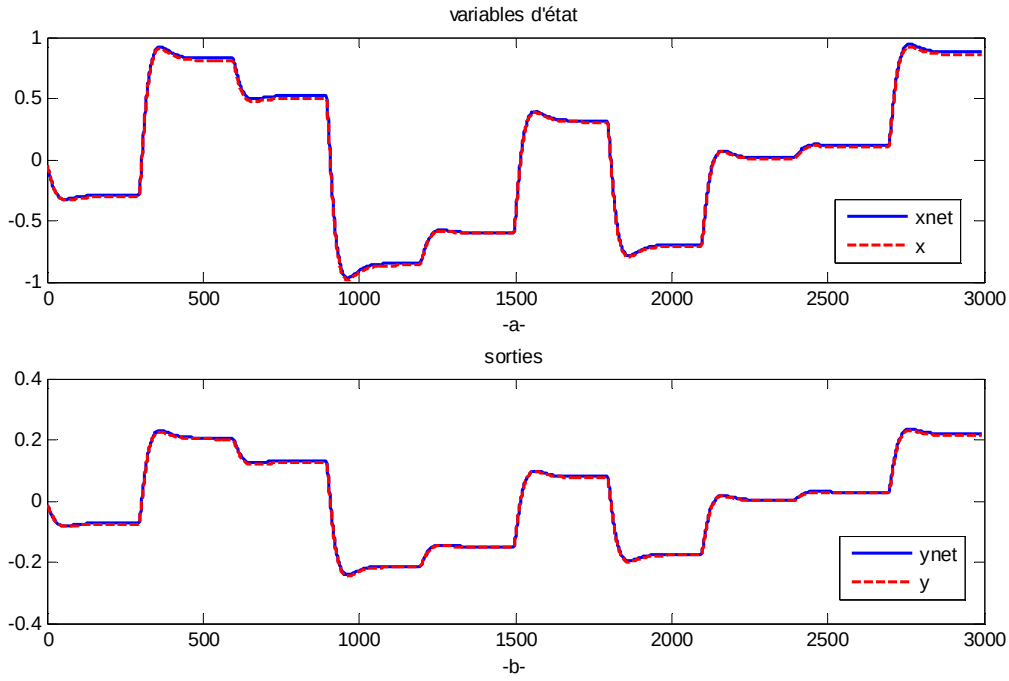


Figure 3.22 : Résultats du test obtenus avec le SSNN (1, 10, 1, 2, 1) modifié.
-a- Variables d'état -b-Sorties.

D'après les Figures 3.21 et 3.22 on constate que les courbes des états et des sorties (estimées et réelles) sont presque confondues dans les phases appropriées à l'apprentissage et au test et qu'elles ne présentent pas de pics au changement de consigne.

Le Tableau (3.8) rassemble les erreurs quadratiques moyennes d'apprentissage et du test entre l'état estimé et l'état réel, la sortie estimée et la sortie réelle. Les résultats obtenus sont très bons ; en comparant les deux Tableaux (3.8) et (3.6), on constate que les performances de poursuites obtenues avec le réseau SSNN modifié en 12 itérations, n'ont pas été atteintes par le réseau de neurones multicouches en 10000 itérations. Ce qui démontre clairement la performance du réseau SSNN modifié.

	EQMAx	EQMAy	EQMTx	EQMTy	Itérations
L-M	$3.2042*10^{-14}$	$5.3243*10^{-34}$	$1.9574*10^{-4}$	$1.2234*10^{-5}$	12

Tableau 3.8 : Erreurs quadratiques moyennes d'apprentissage et de test entre l'état estimé et l'état réel, la sortie estimée et la sortie réelle.

Le temps de simulation nécessaire pour simuler le modèle (3.14) avec ce réseau neurones en lui appliquant en entrée, la séquence du test, est égale seulement à 10.49 secondes.

4.2.2. Identification d'un modèle d'état non entier à 2 variables d'états:

Maintenant prenant l'exemple d'un modèle d'état non entier avec deux variables d'états, représenté par la relation suivante :

$$\begin{cases} \begin{bmatrix} D^{1.25}x_1(t) \\ D^{1.23}x_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} . \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} . u(t) \\ y(t) = [2 \quad 1] . \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \end{cases} \tag{3.15}$$

L'équation (3.15) représente le modèle d'état du modèle fonction de transfert d'ordre non entier déjà donné par l'équation (3.13), qui peut être aussi vérifié par la relation (1.58).

La représentation d'état généralisée (3.15) comporte deux variables d'états qui sont dérivées respectivement aux ordres non entiers 1.25 et 1.23.

Architecture proposée :

La Figure 3.23, montre l'architecture du modèle neuronal (SSNN) proposé (1,3, 2, 2, 1), qui comporte des entrées de commande $u(k)$, 3 neurones linéaires dans la première couche cachée, deux neurones d'états linéaires, 2 neurones linéaires dans la deuxième couche cachée, et enfin un neurone linéaire dans la couche de sortie. Les deux variables d'états ainsi que l'entrée de commande sont retardées cinq fois, ensuite appliquées à l'entrée du réseau.

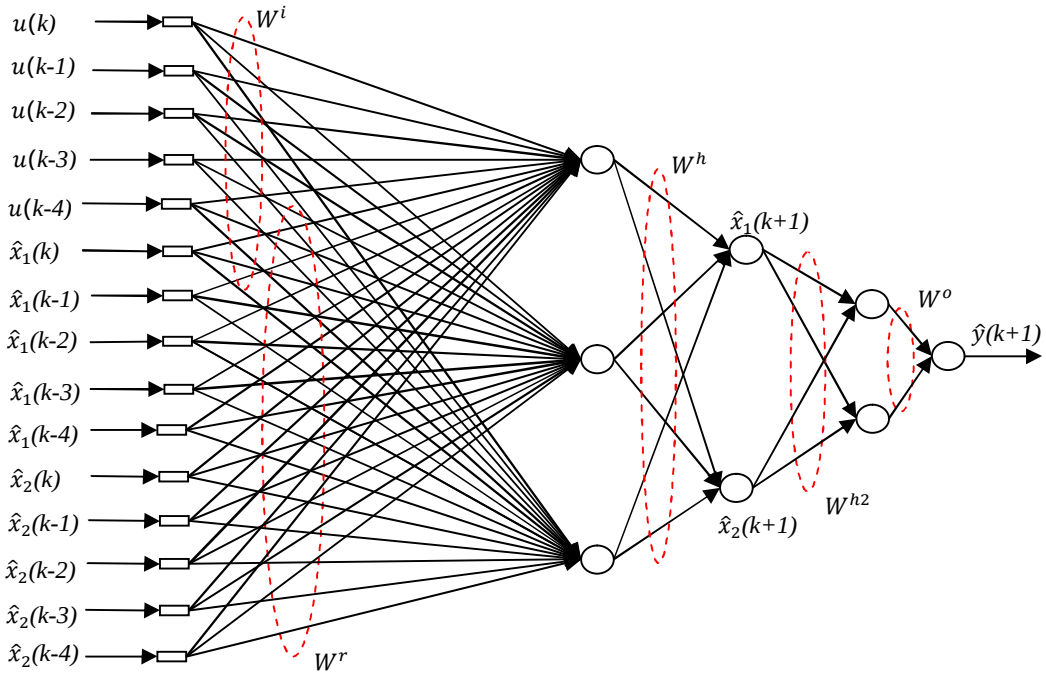


Figure 3.23: Architecture du SSNN (1, 3, 2, 2,1) modifié.

Après simulation de la représentation d'état précédente (3.15), en utilisant toujours l'approximation de Grünwald-Letnikov avec une période d'échantillonnage Te= 0.05 sec et en appliquant le même signal de commande utilisé précédemment (Figure 3.13-a-), on obtient

les courbes données par la Figure 3.24 qui montre l'évolution des variables d'états et de la sortie en fonction du temps.

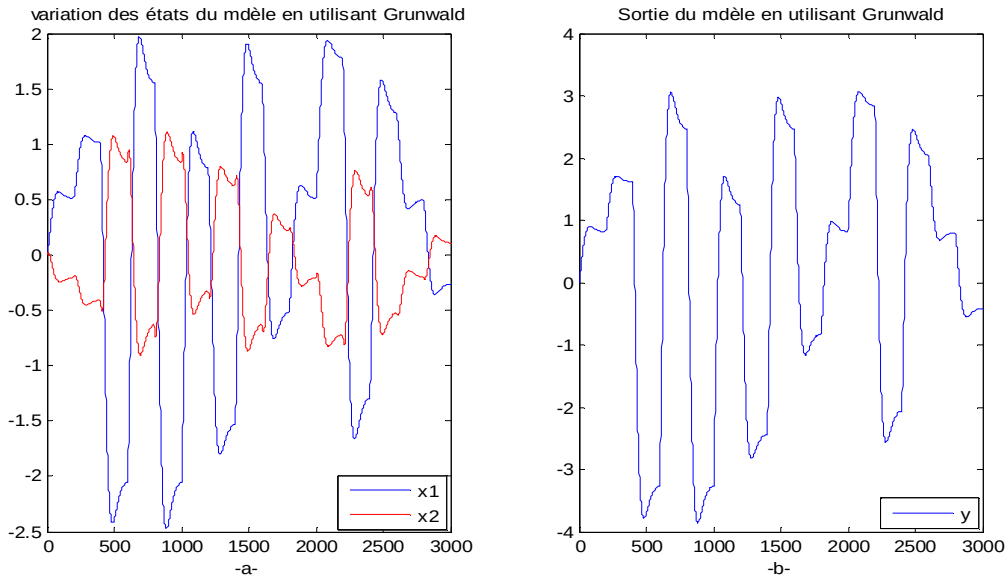


Figure 3.24 : Résultats de simulation du modèle d'état non entier
-a- Variables d'états, **-b-** Variable de sortie

La Figure 3.25, montre le résultat d'apprentissage du réseau de neurones SSNN (1, 3, 2, 2, 1) en utilisant l'entrée1 comme signal d'apprentissage, qui est illustré par la Figure 3.13-a-.

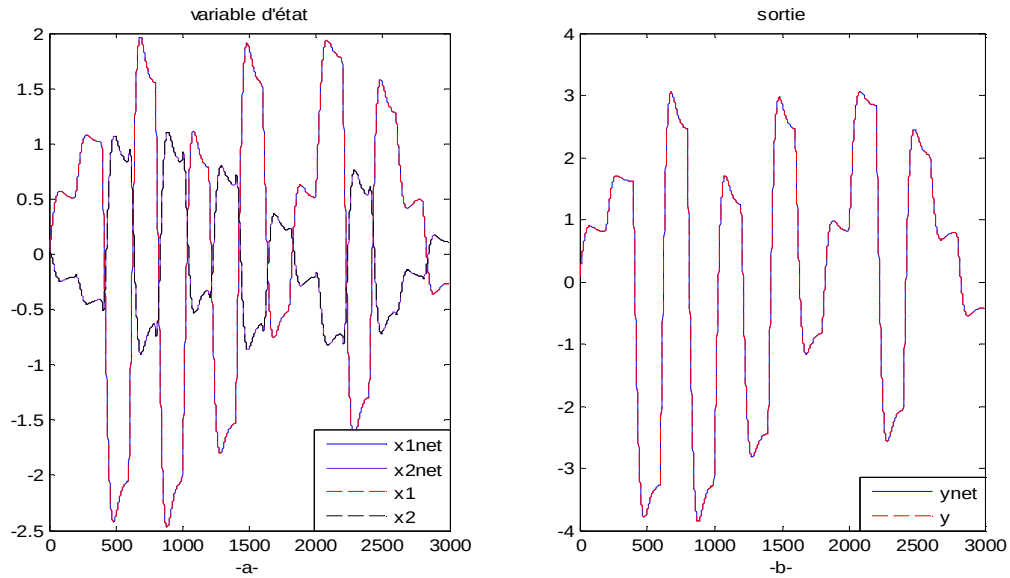


Figure 3.25 : Résultats d'apprentissage obtenus avec le SSNN (1, 3, 2, 2, 1) modifié.
-a- Variables d'états. -b- Variable de sortie.

La Figure 3.26, montre le résultat du test du réseau de neurones SSNN (1, 3, 2, 2, 1) en utilisant l'entrée 2 comme signal du test, qui est illustré par la Figure 3.13-b-.

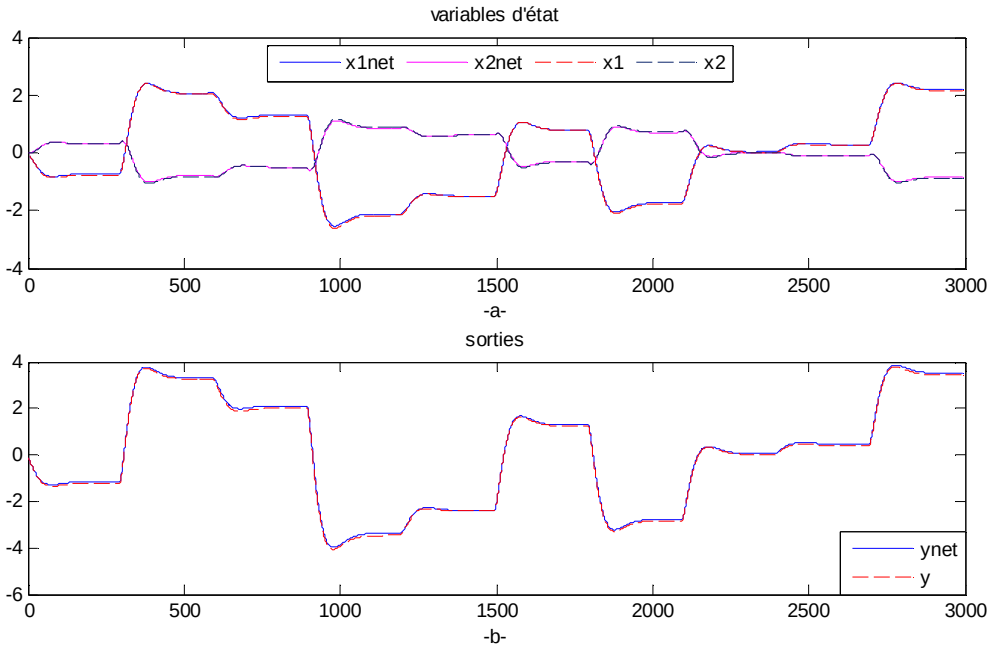


Figure 3.26 : Résultat du test obtenu avec le SSNN (1,3, 2, 2,1) modifié.
-a- Variables d'états -b- variable de sortie.

Le Tableau (3.9) rassemble les erreurs quadratiques moyennes d'apprentissage et du test entre les états estimés et les états réels, la sortie estimée et la sortie réelle.

	EQMAx1	EQMAx2	EQMAy	EQMTx1	EQMTx2	EQMTy	Itération
L-M	$1.36*10^{-10}$	$1.5*10^{-11}$	$7.17*10^{-24}$	$1.1*10^{-3}$	$3.66*10^{-4}$	$4.4*10^{-3}$	91

Tableau 3.9 : Erreurs quadratiques moyennes d'apprentissage et du test entre les états estimés et les états réels, la sortie estimée et la sortie réelle.

Le temps de simulation nécessaire pour simuler le modèle (3.15) avec ce réseau neurones en lui appliquant en entrée, la séquence du test, est égale seulement à 11.26 secondes.

A partir des Figures 3.25 et 3.26, on peut conclure une parfaite adéquation entre les courbes simulées et celles estimées par le réseau de neurones, dans les étapes appropriées à l'apprentissage et à la validation. En comparant les deux Tableaux (3.9) et (3.7), on constate que les performances de poursuite obtenues avec le réseau SSNN en 91 itérations, sont meilleures que celles obtenues avec le réseau de neurones multicouches en 4171 itérations. Ce qui démontre encore une autre fois la performance du réseau SSNN proposé.

Par conséquent, la prise en compte du comportement interne du système fractionnaire (variables d'états), améliore considérablement le temps de convergence ainsi que l'identification neuronale du système non entier.

En comparant le temps de simulation avec celui de la méthode d'approximation de Grünwald-Letnikov, on a constaté également que le temps de simulation est beaucoup plus court avec le réseau de neurones à espace d'état SSNN proposé.

5. Application des réseaux de neurones à la prédiction à un pas d'un modèle non entier

On se propose maintenant d'utiliser un réseau de neurones statique afin de prédire à chaque instant d'échantillonnage t = k.T, la valeur future d'un signal.

Dans ce paragraphe, il s'agit d'appliquer deux types d'architectures neuronales statiques à la prédiction à un pas des modèles d'ordre fractionnaires, ces architectures sont basées elles aussi sur les deux architectures neuronales, appliquées auparavant pour identifier les modèles d'ordre entier.

En effet, dans la première partie on va appliquer le réseau de neurones multicouches statique pour la prédiction à un pas d'un modèle fonction de transfert d'ordre non entier, ensuite, dans la seconde partie on va appliquer le réseau de neurones de type SSNN prédicteur pour la prédiction à un pas d'un modèle d'équation d'état d'ordre non entier.

5.1. Prédicteur à un pas d'un modèle non entier avec un réseau multicouches prédicteur

Afin de prédire la sortie d'un système fractionnaire à un pas d'échantillonnage avec un réseau de neurones multicouches statique, on prend le même exemple du modèle fractionnaire utilisé précédemment et le même fichier de données entrées /sorties, données respectivement par la fonction de transfert (3.13) et la Figure 3.13.

Architecture proposée:

L'architecture du modèle neuronal statique multicouches proposée pour la prédiction à un pas d'échantillonnage du modèle fractionnaire comporte : 11 entrées (une entrée de commande et 10 valeurs passées de la sortie du modèle à identifier), 5 neurones non linéaires dans la couche cachée, 1 neurone linéaire dans la couche de sortie; tous les neurones utilisés dans la couche cachée possèdent une fonction de transfert tangente hyperbolique (th), comme le montre la Figure 3.27 ci-dessous :

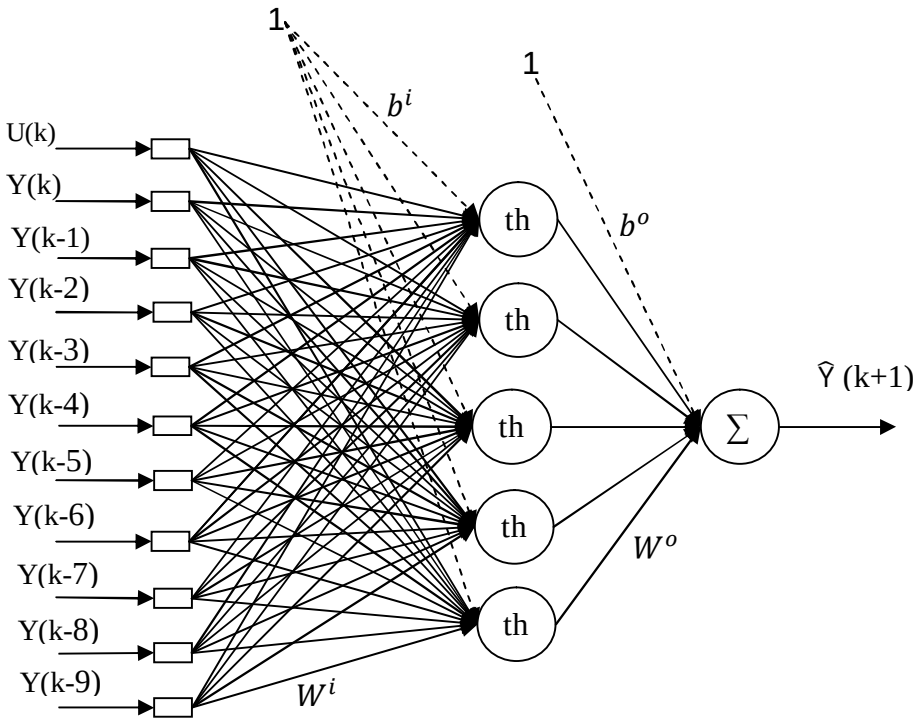


Figure 3.27 : Architecture du réseau de neurones multicouches prédicteur à un pas.

La Figure 2.28 illustre les résultats d'apprentissage et du test en utilisant un apprentissage hors ligne avec l'algorithme de L-Marquardt.

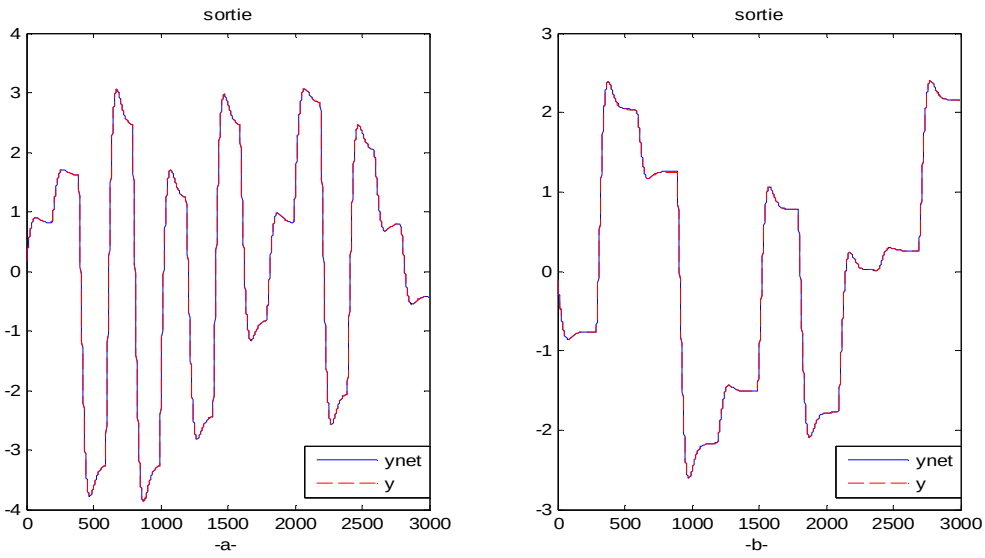


Figure 2.28 : Résultats de la sortie obtenus lors de l'apprentissage et du test
-a- Séquence d'apprentissage. **-b-** Séquence de test.

Le Tableau (3.10), rassemble les erreurs quadratiques moyennes d'apprentissage et du test entre la sortie estimée et la sortie réelle.

	EQMA	EQMT	Itérations
Levenberg Marquardt	$6.6379*10^{-7}$	$9.1011*10^{-6}$	976

Tableau 3.10: Erreur quadratique moyenne entre la sortie réelle et la sortie estimée par le réseau multicouches prédicteur à un pas.

D'après les Figures 2.28-a- et 2.28-b- on constate une très bonne approximation du système, ainsi qu'une absence des pics causés par les variations brusques de l'entrée. A partir du Tableau (3.10), on constate aussi que les performances de poursuites obtenues sont très bonnes. Tandis que le nombre d'itérations nécessaires pour l'apprentissage du réseau est de 976 itérations. Ce qui démontre que le réseau de neurones multicouches prédicteur proposé, a réussi à prédire la sortie d'un modèle fractionnaire de dimension deux.

5.2. Prédiction à un pas d'un modèle fractionnaire avec un réseau SSNN prédicteur

On a vu précédemment, que la présence des d'états dans l'architecture du réseau de neurones, contribuent à une meilleure identification de la dynamique du modèle fractionnaire. Par conséquent, on va s'inspirer de l'architecture SSNN pour concevoir un réseau de neurone qui permet de prédire un pas la sortie d'un modèle fractionnaire.

Afin de comparer les résultats obtenus, avec le réseau multicouches prédicteur à un pas, on choisit le même exemple de modèle d'état fractionnaire donné par la relation (3.15).

Architecture proposée :

La Figure 3.29 montre l'architecture du modèle neuronal SSNN prédicteur proposé (21, 10, 2, 2, 1), l'architecture du modèle neuronal (SSNN) est constituée de 21 entrées (une entrée de commande $u\ (k)$et 10 valeurs passées de chaque variables d'état du modèle) ,10 neurones non linéaires (th) dans la première couche cachée, deux neurones d'états linéaires, 2 neurones linéaires dans la deuxième couche cachée, et enfin un neurone linéaire dans la couche de sortie. Ce réseau peut prédire la sortie et les états à l'instant *k*, connaissant les valeurs passés des états du système.

Contrairement à l'architecture SSNN, qui utilise les d'état estimées par le réseau de neurones, cette architecture, nécessite les valeurs passées des d'états du modèle.

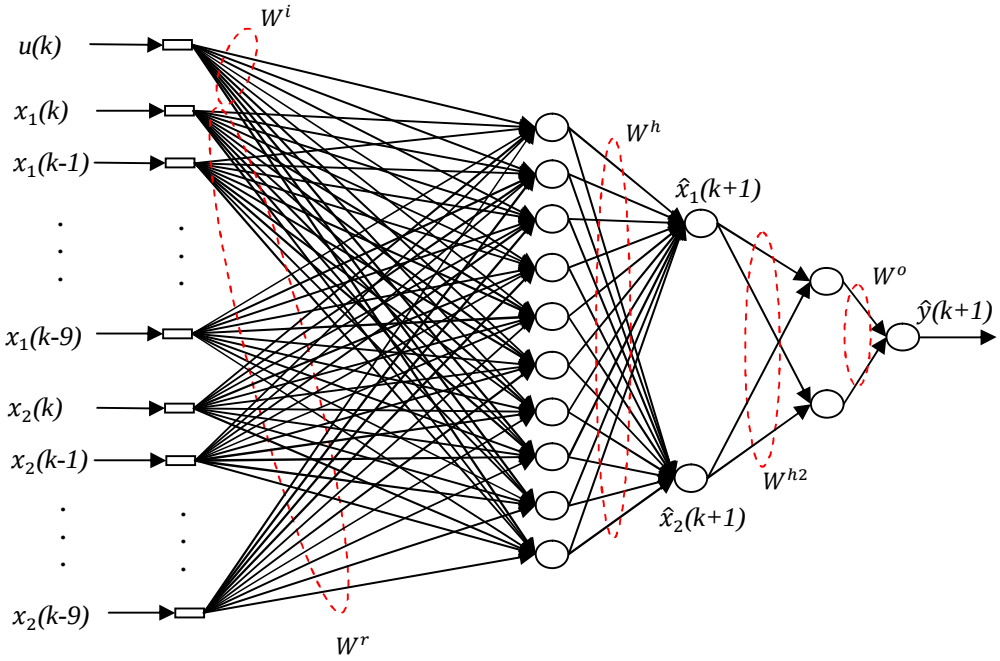


Figure 3.29: Architecture du SSNN (21, 10, 2, 2,1) prédicteur à un pas.

La Figure 3.30 montre le résultat d'apprentissage pour le réseau de neurones SSNN (21, 10, 2, 2, 1) prédicteur à un pas, en utilisant le signal d'apprentissage de la Figure 3.13-a-.

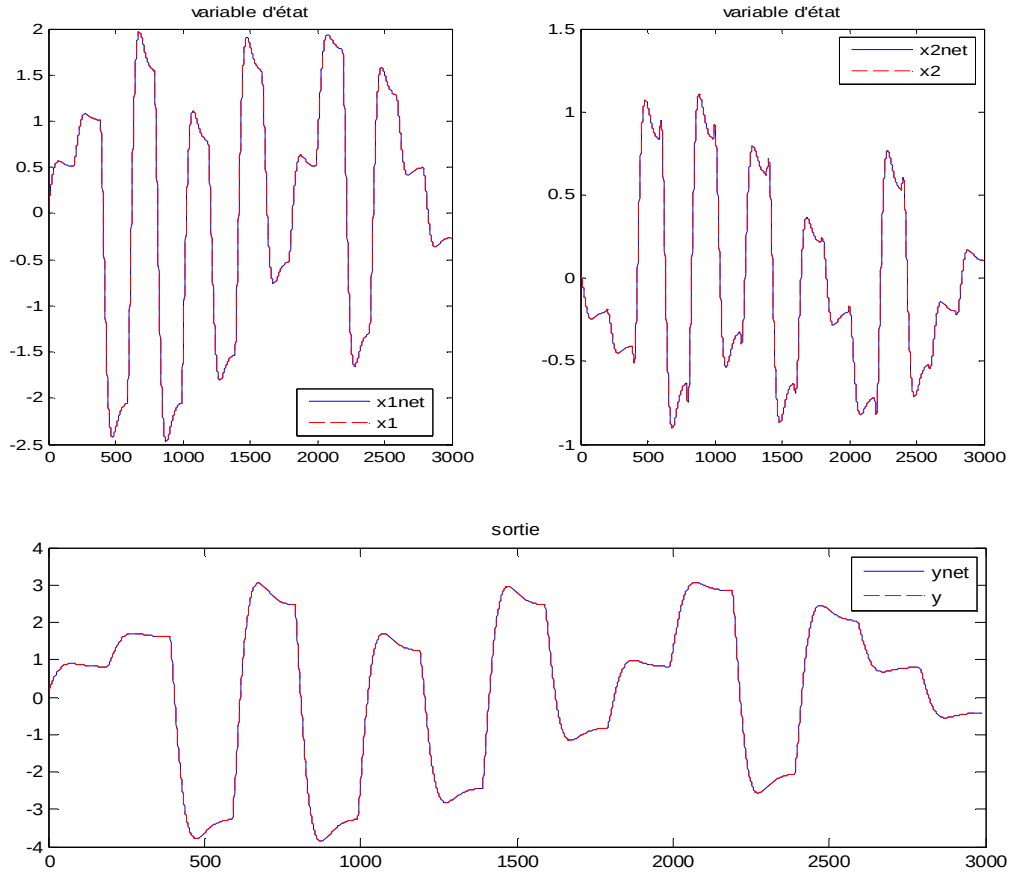


Figure 3.30 : Résultats d'apprentissage obtenus avec le SSNN (21, 10, 2, 2, 1) prédicteur.

La Figure 3.31 quant à elle, montre le résultat du test pour le réseau de neurones SSNN (21, 10, 2, 2, 1) prédicteur en utilisant l'entrée 2 comme signal du test, illustré par de la Figure 3.13-b-.

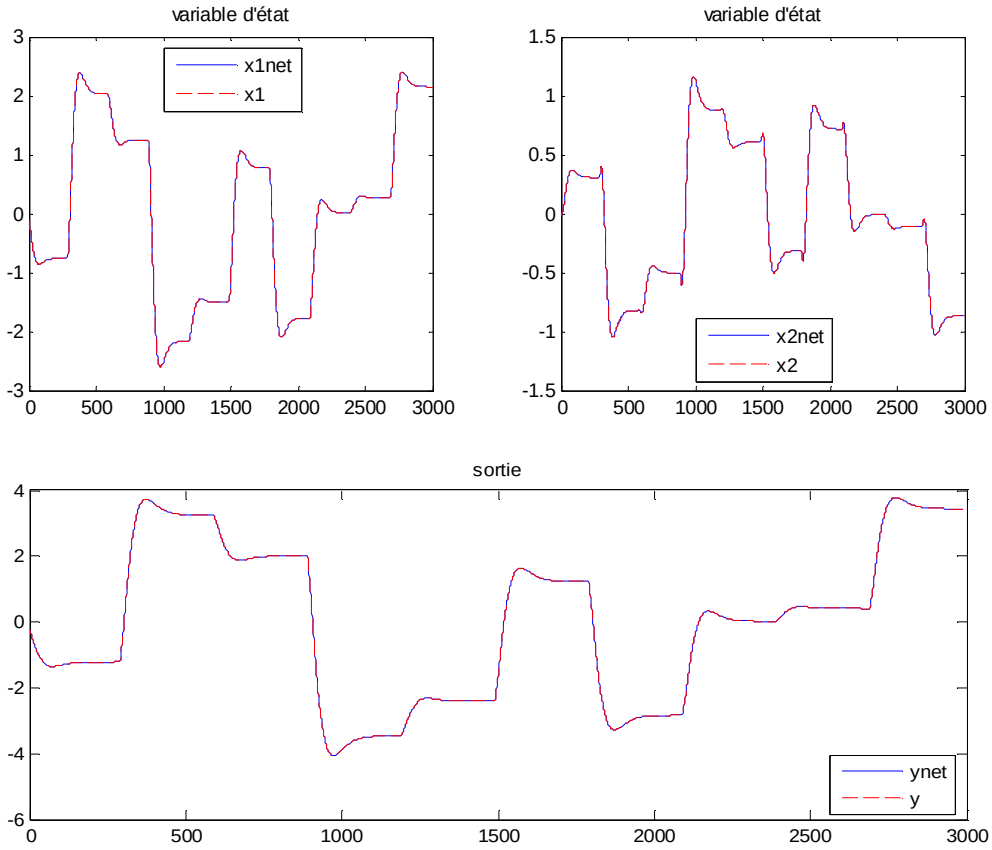


Figure 3.31 : Résultats de test obtenus avec le SSNN (21,10, 2, 2,1) prédicteur.

Les Figures 3.30 et 3.31, montrent une parfaite adéquation entre les courbes simulées et les courbes estimées. On constate que les performances de poursuite des variables d'états et de la sortie sont excellentes.

Le Tableau (3.11) montre que les performances de poursuite des d'états et de la sortie sont excellentes. En comparant les deux Tableaux (3.11) et (3.10), on constate que l'EQMTy est égale à 9.10^{-10}, c'est-à-dire qu'elle est 10 000 fois inferieure à celle obtenue avec un réseau de neurones multicouches prédicteur à un pas d'échantillonnage.

	EQMAx1	EQMAx2	EQMAy	EQMTx1	EQMTx2	EQMTy
L-M	$1.24*10^{-11}$	$3.01*10^{-10}$	$1.63*10^{-28}$	$1.52*10^{-10}$	$3.01*10^{-10}$	$9.35*10^{-10}$

Tableau 3.11 : Erreurs quadratiques moyennes d'apprentissage et du test. entre les états estimés et les états réels, la sortie estimée et la sortie réelle.

On conclusion on peut dire que le réseau le SSNN prédicteur à un pas, est plus performant que le réseau multicouches prédicteur à un pas d'échantillonnage. L'introduction des d'états dans l'architecture du réseau a amélioré considérablement la qualité d'identification du réseau prédicteur à un pas d'échantillonnage.

Conclusion :

Les travaux présentés dans ce chapitre portent sur l'identification des modèles dynamiques entiers et fractionnaires à l'aide des réseaux de neurones dynamiques. On a entamé ce chapitre par l'identification de modèles d'ordre entier, en utilisant deux types de réseau récurrent, à savoir « le réseau de neurones multicouches retardé en entrée et en sortie » et « le réseau de neurones SSNN ». L'identification a abouti à des résultats très satisfaisants, néanmoins on a constaté la performance du réseau SSNN, puisque, ce dernier a non seulement identifié la dynamique, mais aussi a réussit à déterminer les différentes matrices du modèle d'état, grâce à sa structure neuronale qui se rapproche de la représentation d'états classique.

Ensuite, on a abordé l'identification neuronale des modèles d'ordres non entiers généralisés, à cet effet, on a proposé deux réseaux récurrents : le réseau de neurones multicouche modifié, retardé en entrée et en sortie, ainsi que les réseaux de neurones SSNN modifié, ces derniers s'inspirent beaucoup des deux architectures neuronales précédents. Contrairement au cas entier, où le nombre des lignes à retard du réseau, dépend principalement de l'ordre du système ; le nombre des lignes à retard des réseaux proposés, ne dépend en aucun cas de l'ordre modèle de connaissance, étant donné que ce dernier est de dimension infini. Les réseaux de neurones proposés, ont identifié les différents modèles fractionnaires, en reproduisant leurs dynamiques avec une bonne précision. Par conséquent, ils peuvent être utilisés comme simulateurs neuronaux pour les systèmes linéaires d'ordre non entiers.

A la fin de ce chapitre, on a traité la prédiction à un pas d'échantillonnage des modèles d'ordres fractionnaires. A cet effet, deux architectures neuronales statiques ont été proposées : « le réseau de neurone multicouches prédicteur à un pas d'échantillonnage » et « le réseau SSNN prédicteur à un pas ». Les résultats obtenus avec ces deux structures sont excellents. Par conséquent, les deux réseaux de neurones prédicteur précédents sont des prédicteur à un pas idéaux capables de prédire à un pas la sortie de n'importe quel système fractionnaire.

Dans ce chapitre, on a démontré que le réseau SSNN et plus performent que le réseau de neurones multicouches, dans l'identification ainsi que dans la prédiction à un pas des modèles d'ordre fractionnaires, grâce à son identification du comportement interne du

modèle. Toutefois, le réseau SSNN présente un inconvénient majeur, qui est celui de la disponibilité des variables d'états à la mesure.

Le temps de simulation des modèles fractionnaires avec les réseaux de neurones, s'est avéré beaucoup plus court que celui des méthodes de simulations mathématiques classiques, qui elles, tiennent en compte de tous le passé du système, ce qui augmente la duré de leur simulation.

Chapitre 4

Tests et résultats sur une base de données d'un système fractionnaire réel

1. Introduction

Le chapitre précédent a été consacré exclusivement à l'identification et à la prédiction à un pas des modèles théoriques d'ordre entier et non entier, à l'aide des réseaux de neurones multicouches récurrents et les réseaux de neurones à espace d'états (SSNN).

Cependant, ce présent chapitre est consacré exclusivement à l'identification des systèmes fractionnaires réels, et cela en exploitant deux bases de données, obtenues expérimentalement sur deux systèmes thermiques non entiers.

L'objectif de ce chapitre est la validation des résultats théoriques, obtenus dans le chapitre précédent avec le « réseau de neurones multicouches retardé en entrée et en sortie proposé » ainsi qu'avec le « réseau de neurones multicouches prédicteur à un pas d'échantillonnage ».

2. Introduction aux systèmes thermiques

L'étude des systèmes thermiques à géométrie simple révèle, à travers l'établissement des solutions analytiques, un comportement relevant de la dérivation d'ordre non entier. Ce comportement apparaît naturellement dans les systèmes régis par l'équation de diffusion, notamment entre le flux de chaleur et la température dans le cas de l'équation de la chaleur (Battaglia, 2002 ; Benchellal, 2008).

2.1. Equation de diffusion de la chaleur

Il est bien connu que l'évolution de la température dans un milieu conducteur de la chaleur est régie par une équation aux dérivées partielles dite équation de la chaleur.

Le transfert de chaleur idéalisé à une dimension est régi par l'équation aux dérivées partielles appelée équation de la chaleur (Djamah, 2009) :

$$\frac{\partial T(x,t)}{\partial t} = \frac{\alpha_c}{x^p}\frac{\partial}{\partial x}\left(x^p \frac{\partial T(x,t)}{\partial x}\right) \quad \text{pour} \quad 0 < x < \infty \ et \ t > 0 \tag{4.1}$$

Avec $\alpha_c = \frac{\lambda_c}{\rho c_p}$ (4.2)

Où

α_c représente le coefficient de diffusivité,

x est l'abscisse du point de mesure,

λ_c est la conductivité thermique du milieu, supposé constante,

c_p représente la chaleur massique du milieu,

ρ représente la densité massique du milieu.

La géométrie du matériau considéré est caractérisée par la variable p telle que :

- $p = 0$ pour une géométrie plane ;
- $p = 1$ pour une géométrie cylindrique ;
- $p = 2$ pour une géométrie sphérique.

2.2. Etude du transfert de chaleur dans un mur plan

Dans ce paragraphe on va montrer sur un exemple simple, comment la dérivée non entière apparait d'une façon naturelle dans un système thermique a géométrie plane (Poinot et Trigeassou, 2004 ; Benchellal, 2008).

Pour cela considérons le cas classique du transfert de chaleur à travers un mur plan (Figure 4.1). On considère un mur uniforme, homogène, de longueur L, chauffé sur la face A par un flux de chaleur $\phi(0,t)$, tandis que la face B transmet la chaleur au milieu ambiant à travers une résistance thermique R.

La température dans le mur est une fonction $T(x,t)$ de l'abscisse x, avec $0 \leq x \leq$ L. Cette température est considérée uniforme en chaque point d'un plan parallèle aux faces A et B. Soit $\phi(x,t)$ le flux de chaleur qui traverse le mur à l'abscisse x. La diffusion de la chaleur sur ce milieu est exprimée par l'´equation de la chaleur (4.3), généralisation de la loi de Fick (4.4).

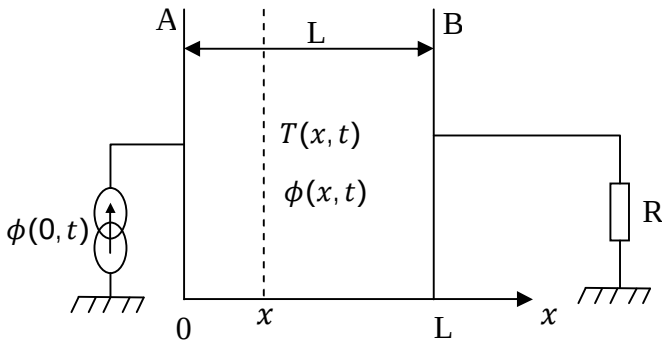


Figure 4.1: Transfert de chaleur dans un mur plan.

$$\frac{\partial T(x,t)}{\partial t} = \alpha_c \frac{\partial^2 T(x,t)}{\partial x^2} \quad (4.3)$$

$$\phi(x,t) = -\lambda_c \frac{\partial T(x,t)}{\partial x} \quad (4.4)$$

L'´equation (4.4) exprime que la propagation de la chaleur correspond à un flux proportionnel au gradient de la température. La chaleur va se propager de la face la plus chaude vers la face la plus froide pour équilibrer les températures par conduction.

Les équations (4.3) et (4.4) spécifient la relation entre $\phi(x,t)$ et $T(x,t)$, considérés respectivement comme l'entrée et la sortie du système lorsque $x = 0$, ce qui permet de définir l'interface de diffusion. Pour ce faire, on considère les conditions aux limites suivantes :

- ***Sur la surface A :*** le flux de chaleur $\phi(0,t)$ est imposé. Il représente l'entrée $u(t)$ du système.

$$\phi(0,t) = u(t) \quad (4.5)$$

La sortie est alors $y(t) = T(0,t)$, (4.6)

- ***Sur la surface B :***

$$\phi(L,t) = \frac{T(L,t)}{R} \quad (4.7)$$

R est la résistance thermique entre le mur et le milieu ambiant. La condition initiale de la température est :

$$T(x,0) = 0 \quad (4.8)$$

La température du milieu ambiant est supposée constante et égale à zéro ($T(L,t)$= 0). La sortie de ce système est la température $T(0,t) = y(t)$ sur la face A.

La modélisation théorique de cette interface est équivalente à la détermination de la fonction de transfert $H(s)$ entre $Y(s)$ et $U(s)$ (où $Y(s)$et $U(s)$ sont les transformées de Laplace de $y(t)$ et $u(t)$). Nous utiliserons donc la technique de la transformée de Laplace appliquée à l'équation aux dérivées partielles.

$$\mathcal{L}\big(T(x,t)\big) = T(x,s) = \int_0^\infty T(x,t)e^{-st}dt \quad (4.9)$$

Avec $$\mathcal{L}\left(\frac{\partial T(x,t)}{\partial t}\right) = sT(x,s) - T(x,0) \quad (4.10)$$

Et

$$\mathcal{L}\left(\frac{\partial^2 T(x,t)}{\partial x^2}\right) = \frac{\partial^2}{\partial x^2}\,\mathcal{L}\big(T(x,t)\big) = \frac{\partial^2}{\partial x^2}\,T(x,s) \quad (4.11)$$

Alors la relation (4.3) s'écrit

$$sT(x,s) - T(x,0) = \alpha_c \frac{\partial^2}{\partial x^2}\,T(x,s) \quad (4.12)$$

Ou encore

$$\alpha_c \frac{\partial^2}{\partial x^2} T(x,s) - sT(x,s) = -T(x,0) \tag{4.13}$$

qui représente une équation différentielle du deuxième ordre (par rapport à x) où s est un coefficient. Considérons cette équation différentielle sans second membre :

$$\alpha_c \frac{\partial^2}{\partial x^2} T(x,s) - sT(x,s) = 0 \tag{4.14}$$

Elle admet pour équation caractéristique :

$$\alpha_c {\lambda_c}^2 - s = 0 \quad \Rightarrow \quad \lambda_c = \pm\sqrt{\frac{s}{\alpha_c}} \tag{4.15}$$

La solution générale de l'´equation différentielle est de la forme :

$$T(x,s) = L_1(s)e^{\lambda_{c_1} x} + L_2(s)e^{\lambda_{c_2} x} \tag{4.16}$$

Soit

$$T(x,s) = L_1(s)e^{x\sqrt{\frac{s}{\alpha_c}}} + L_2(s)e^{-x\sqrt{\frac{s}{\alpha_c}}} \tag{4.17}$$

Les termes $L_1(s)$ et $L_2(s)$ dépendent des conditions aux limites. Leur calcul se fait à partir de $\phi(0,t)$et $\phi(L,t)$ pour cela, on doit formuler $\phi(x,s)$, En considérant l'équation (4.4), qui nous permet d'écrire :

$$\phi(x,s) = -\lambda_c \frac{\partial T(x,s)}{\partial x} \tag{4.18}$$

Par conséquent la dérivée de la relation (4.17) par rapport à x, nous donne

$$\phi(x,s) = -\lambda_c \left[\sqrt{\frac{s}{\alpha_c}} L_1(s)e^{x\sqrt{\frac{s}{\alpha_c}}} - \sqrt{\frac{s}{\alpha_c}} L_2(s)e^{-x\sqrt{\frac{s}{\alpha_c}}}\right] \tag{4.19}$$

Pour déterminer $L_1(s)$ et $L_2(s)$, écrivons (4.19) en x = 0 :

$$\phi(0,s) = -\lambda_c \sqrt{\frac{s}{\alpha_c}} (L_1(s) - L_2(s)) \tag{4.20}$$

Pour $x = L$, on a :

$$\phi(L,s) = -\lambda_c \left[\sqrt{\frac{s}{\alpha_c}} L_1(s)e^{L\sqrt{\frac{s}{\alpha_c}}} - \sqrt{\frac{s}{\alpha_c}} L_2(s)e^{-L\sqrt{\frac{s}{\alpha_c}}}\right] \tag{4.21}$$

Définissons $x_1 = L_1(s)e^{L\sqrt{\frac{s}{\alpha_c}}}$ et $x_2 = L_2(s)e^{-L\sqrt{\frac{s}{\alpha_c}}}$

Soit $$\phi(L,s) = \frac{T(L,s)}{R} \tag{4.22}$$

D'après la relation (4.19), on peut écrire :

$$-\lambda_c R\sqrt{\frac{s}{\alpha_c}}[x_1 - x_2] = x_1 + x_2 \tag{4.23}$$

Posons : $$z = \lambda_c R\sqrt{\frac{s}{\alpha_c}} \tag{4.24}$$

Alors on obtient $$L_1(s) = L_2(s)e^{-2\sqrt{\frac{s}{\alpha_c}}L}\frac{z-1}{z+1} \quad (4.25)$$

En combinant (4.25) et (4.20), on a :

$$L_2(s) = \frac{\phi(0,s)}{\lambda_c\sqrt{\frac{s}{\alpha_c}}}\frac{1}{\left(1-ae^{-2\sqrt{\frac{s}{\alpha_c}}L}\right)} \text{ avec } a = \frac{z-1}{z+1} \quad (4.26)$$

On peut définir par la suite $L_1(s)$:

$$L_1(s) = \frac{\phi(0,s)}{\lambda_c\sqrt{\frac{s}{\alpha_c}}}\frac{ae^{-2\sqrt{\frac{s}{\alpha_c}}L}}{\left(1-ae^{-2\sqrt{\frac{s}{\alpha_c}}L}\right)} \quad (4.27)$$

Finalement on obtient grâce à la relation (4.17) :

$$T(x,s) = \frac{\phi(0,s)}{\lambda_c\sqrt{\frac{s}{\alpha_c}}}\left[\frac{ae^{-2\sqrt{\frac{s}{\alpha_c}}L}}{\left(1-ae^{-2\sqrt{\frac{s}{\alpha_c}}L}\right)}e^{x\sqrt{\frac{s}{\alpha_c}}} + \frac{1}{\left(1-ae^{-2\sqrt{\frac{s}{\alpha_c}}L}\right)}e^{-x\sqrt{\frac{s}{\alpha_c}}}\right] \quad (4.28)$$

$$\phi(x,s) = -\phi(0,s)\left[\frac{ae^{-2\sqrt{\frac{s}{\alpha_c}}L}}{\left(1-ae^{-2\sqrt{\frac{s}{\alpha_c}}L}\right)}e^{x\sqrt{\frac{s}{\alpha_c}}} + \frac{1}{\left(1-ae^{-2\sqrt{\frac{s}{\alpha_c}}L}\right)}e^{-x\sqrt{\frac{s}{\alpha_c}}}\right] \quad (4.29)$$

On s'intéresse au transfert reliant la température à l'interface ($T(0,s)$) au flux de chaleur ($\phi(0,s)$). On obtient ainsi :

$$H(s) = \frac{T(0,s)}{\phi(0,s)} = \frac{\lambda_c R\sqrt{\frac{s}{\alpha_c}}+1+(\lambda_c R\sqrt{\frac{s}{\alpha_c}}-1)e^{-2\sqrt{\frac{s}{\alpha_c}}L}}{\lambda_c\sqrt{\frac{s}{\alpha_c}}\left(\lambda_c R\sqrt{\frac{s}{\alpha_c}}+1-(\lambda_c R\sqrt{\frac{s}{\alpha_c}}-1)e^{-2\sqrt{\frac{s}{\alpha_c}}L}\right)} \quad (4.30)$$

Pour étudier le comportement asymptotique du modèle théorique obtenu $H(s)$, considérons le cas ou le flux $\phi(0,t)$ est un échelon de valeur ϕ. Alors $\phi(0,s) = \frac{\phi}{s}$ et

$$T(0,s) = H(s) * \frac{\phi}{s} \quad (4.31)$$

Lorsque $t \to \infty$ (équivalent à $w \to 0$) on obtient :

$$T(0,\infty) = y(\infty) = \phi * R \quad (4.32)$$

Le mur se comporte alors comme une résistance thermique de valeur nulle (conduction idéale). A l'inverse, pour faire apparaître le comportement transitoire, faisons tendre $t \to 0$ (équivalent à $w \to \infty$). On obtient :

$$H(s) \approx \frac{\sqrt{\alpha_c}}{\lambda_c s^{0.5}} \quad (4.33)$$

Aux fréquences infinies (ou de manière équivalente aux temps très faibles) le "mur" se comporte donc comme un système fractionnaire, dont l'ordre n égal à 0.5.

3. Identification d'un système expérimental

3.1. Description du premier système expérimental

Le système expérimental considéré est une sphère de laiton de trois centimètres de rayon (voir Figure 4.2). Un transistor de puissance est placé au centre de la sphère afin de générer un flux de chaleur. Un capteur de température est fixé sur la source de chaleur, donc à l'interface du système de diffusion. La sphère est placée dans une enceinte où la température d'un fluide en circulation est imposée. L'entrée du système est la puissance fournie par le transistor, tandis que la sortie est la température mesurée par le capteur au centre de la sphère en laiton.

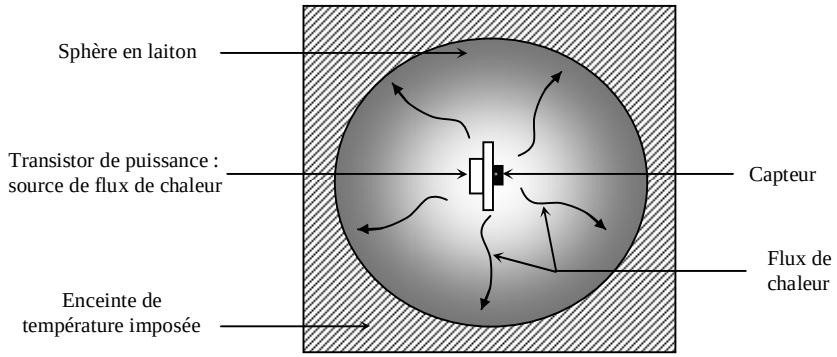


Figure 4.2: Pilote thermique.

Les valeurs des données d'entrée et de sortie sont mesurées par un système d'acquisition de données avec une période d'échantillonnage *Te* = 1 *sec*, l'entrée étant une séquence binaire pseudo-aléatoire de dimension *k*=2000. L'excitation et la réponse du système sont représentées par la Figure 4.3. Le fichier des données entrées /sorties a été divisé en deux parties :

- La première partie qui contient 1264 échantillons a été consacrée pour l'apprentissage du réseau.
- La deuxième partie a été utilisée pour la validation, elle contient 736 échantillons.

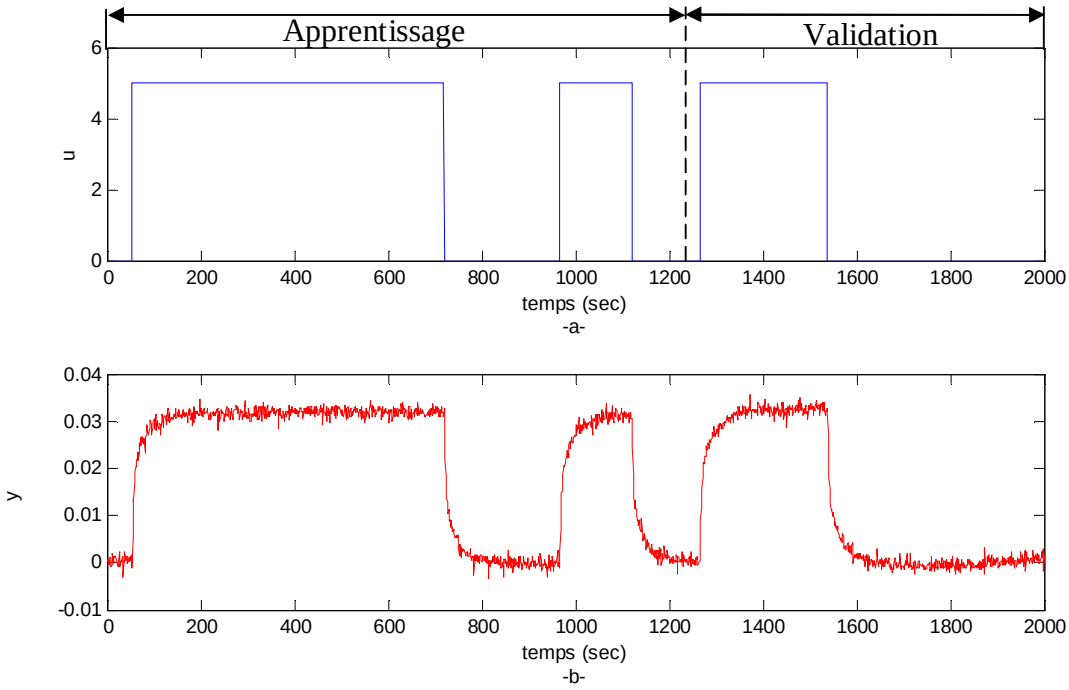


Figure 4.3 : Excitation et réponse du système réel
-a- entée du système réel. **-b-** sortie du système réel.

Architecture proposée

Puisque la base de données contient uniquement des mesures d'entrée/sortie (variables d'états non mesurables), l'architecture appliquée dans ce cas est une architecture multicouches retardée en entrée et en sortie comme le montre la Figure 4.4 ci-dessous :

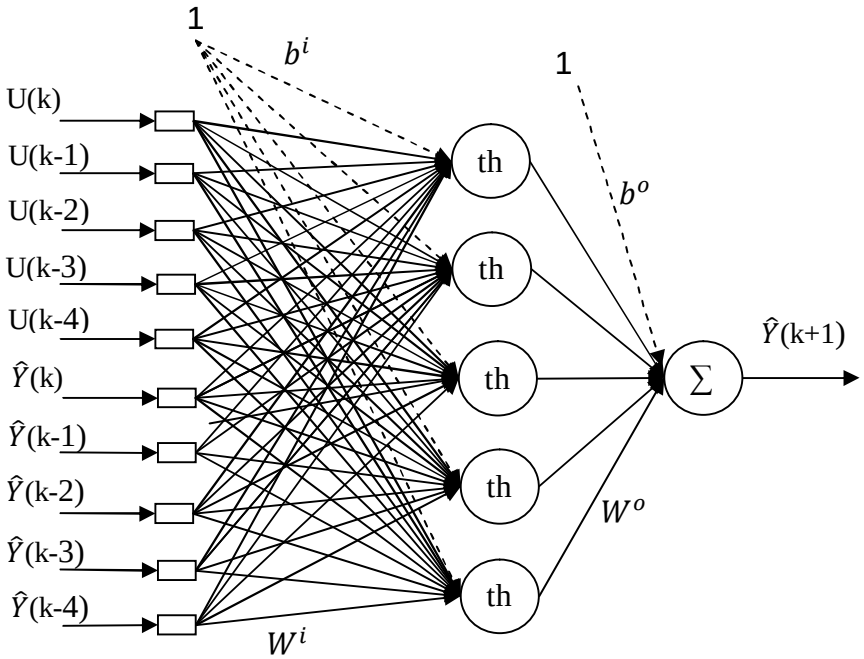


Figure 4.4: Architecture du réseau de neurones multicouches retardé en entrée et en sortie.

L'architecture du modèle neuronal multicouches appliqué comporte : 10 entrées (5 entrées de commande et 5 valeurs passées de la sortie du réseau), 5 neurones dans la couche cachée, 1 neurone dans la couche de sortie; tous les neurones utilisés dans la couche cachée possèdent une fonction de transfert tangente hyperbolique (th), le neurone de sortie est un neurone linéaire.

Les résultats de l'apprentissage et de validation sont illustrés par la Figure 4.5.

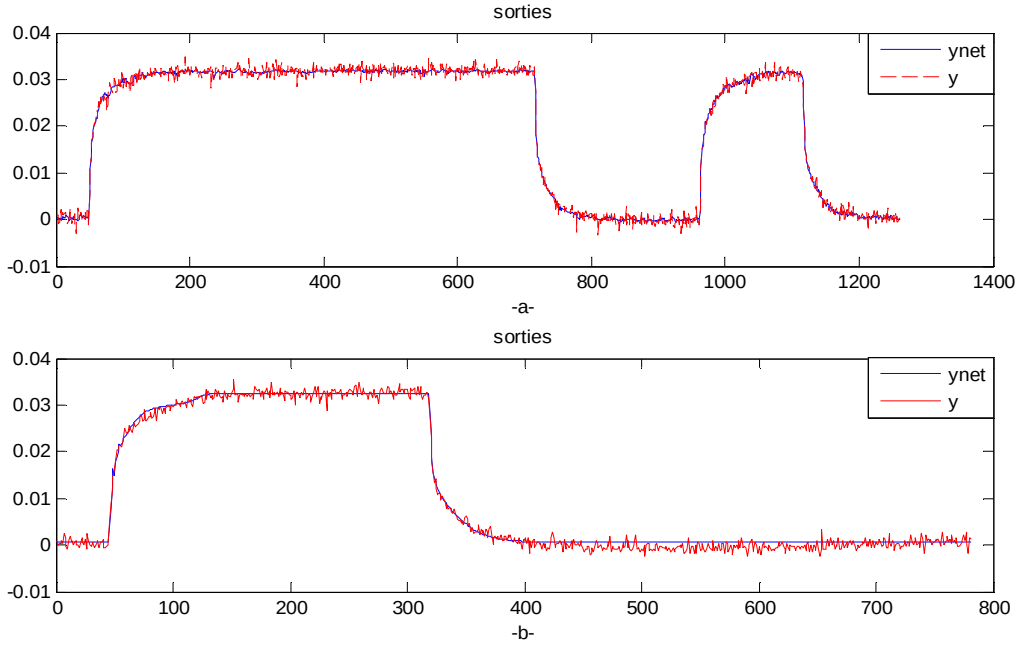


Figure 4.5: Résultats de la sortie obtenus lors de l'apprentissage et du test
-a- Séquence d'apprentissage. -b- Séquence de test.

Le Tableau (4.1) rassemble les EQMA et EQMT.

	EQMA	EQMT	Itérations
Levenberg -Marquardt	$8.7638*10^{-7}$	$1.0969*10^{-6}$	51

Tableau 4.1: Erreur quadratique moyenne d'apprentissage et du test entre la sortie estimée et la sortie réelle.

La Figure 4.6 montre l'allure des erreurs d'apprentissage et du test entre la sortie désirée et la sortie estimée.

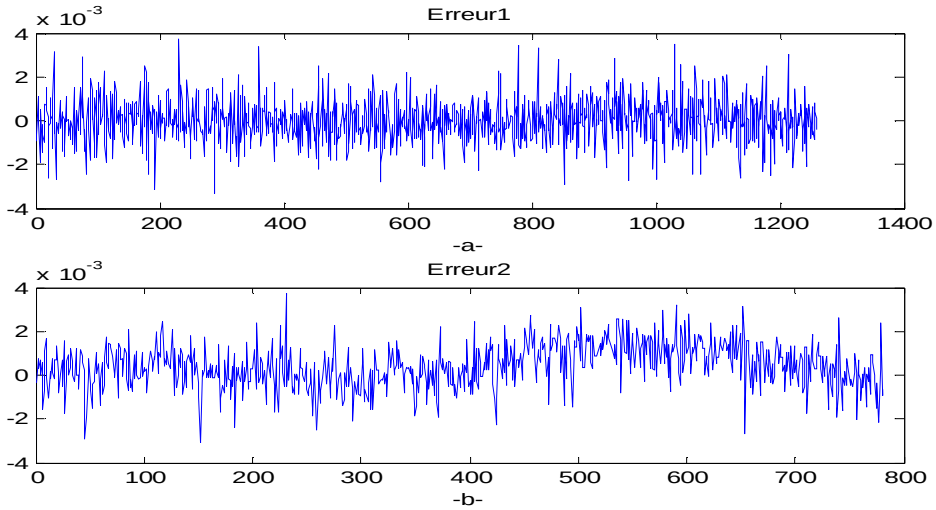


Figure 4.6: Erreurs obtenus lors de l'apprentissage et de test
-a- Erreur d'apprentissage. **-b-** Erreur de test.

D'après la Figure 4.5, on constate que la courbe de sorties estimée par notre réseau suit bien la sortie du système expérimental dans l'étape appropriée à l'apprentissage et à la validation. D'après le Tableau (4.1), on estime que les performances de poursuite obtenus sont satisfaisantes, malgré que l'entrée utilisée pour l'apprentissage, n'est pas riche ni en fréquence, et ni en amplitude. A partir de la Figure 4.6 on peut conclure d'une part que les erreurs de poursuite sont bornées par de faibles valeurs, malgré qu'on a utilisé uniquement 5 neurones dans la couche cachée et après seulement 51 itérations, et que l'EQMT varie peu par rapport à l'EQMA. Par conséquent, on peut conclure que notre réseau de neurones multicouches retardé en entrée et en sortie, a bien identifié la dynamique du procédé réel.

4. Prédiction à un pas d'échantillonnage d'un système expérimental

4.1. Description du deuxième système expérimental

Ce deuxième système expérimental est une barre en aluminium isolée, représentée par la Figure 4.7, l'entrée étant le flux de chaleur appliqué à l'une des extrémités, et la sortie étant la température mesurée à une distance x = 0.5 cm de l'extrémité chauffée (Figure 4.8);

La température est mesurée en utilisant une sonde en platine, le flux thermique est généré par une résistance chauffante, colée à l'une des extrémités de la tige. Le flux thermique maximal est de 20W. Ce flux thermique est contrôlé par un calculateur à travers un convertisseur PWM (MLI), en variant la tension de l'entrée entre 0 et 12 Volts (voir la Figure 4.9).

A l'instant initial, le système est à la température ambiante, et on suppose que l'échange thermique avec l'extérieur est nul (Malti et Sabatier, 2004).

Comme dans l'exemple précédant du transfert de chaleur dans un mur plan, ce système est régi lui aussi par les équations (4.3) et (4.4), car il possède une géométrie plane. Ainsi, le développement de ces équations permet d'aboutir à l'équation (4.16).

Si maintenant on s'intéresse au transfert reliant la température ($T(x,s)$) au flux de chaleur ($\phi(s)$). Et en tenant compte des conditions initiales :

$$T(x,s) = 0, \quad 0 < x < \infty, \quad t = 0 \tag{4.34}$$

On obtient alors la fonction de transfert suivante:

$$H(x,s) = \frac{T(x,s)}{\phi(s)} = \frac{1}{\sqrt{s}\sqrt{\lambda_c \rho c_p}} e^{-x\sqrt{\frac{s}{\alpha_c}}} \tag{4.35}$$

L'équation (4.35) représente un modèle fonction de transfert d'ordre non entier retardé, d'ordre 0.5.

Figure 4.7: système expérimental

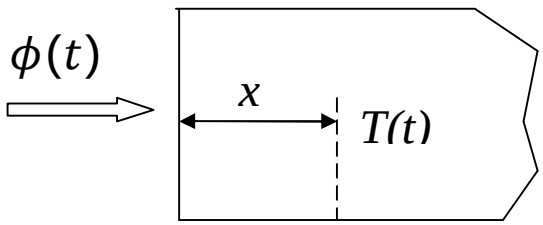


Figure 4.8 : Représentation schématisée du système expérimental.

Les valeurs des données d'entrée et sortie sont mesurées par un système d'acquisition de données avec une période d'échantillonnage *Te* = 0.5 *sec*, l'entrée étant une séquence binaire pseudo-aléatoire de dimension *k*=4421. L'excitation et la réponse du système sont représentées en Figure 4.9.

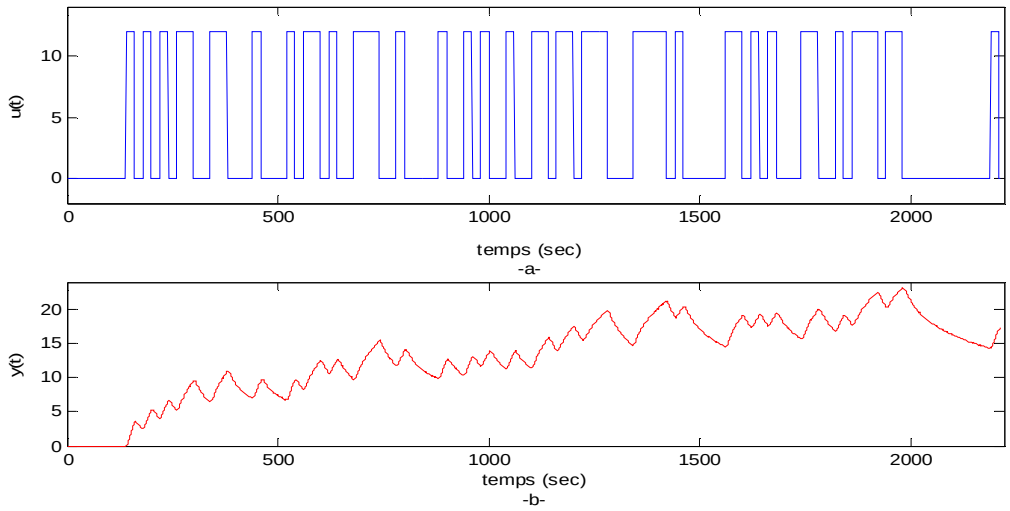


Figure 4.9 : Excitation et réponse du système réel
-a- entée du système réel.**-b-** sortie du système réel.

Le fichier de données correspondant à ce système expérimental, contient seulement les couples de mesure entrées / sorties. Ceci exige l'application du réseau de neurones multicouches prédicteur à un pas, qui lui nécessite seulement les couples d'entrées /sorties, dont l'architecture est illustrée par la Figure 4.10.

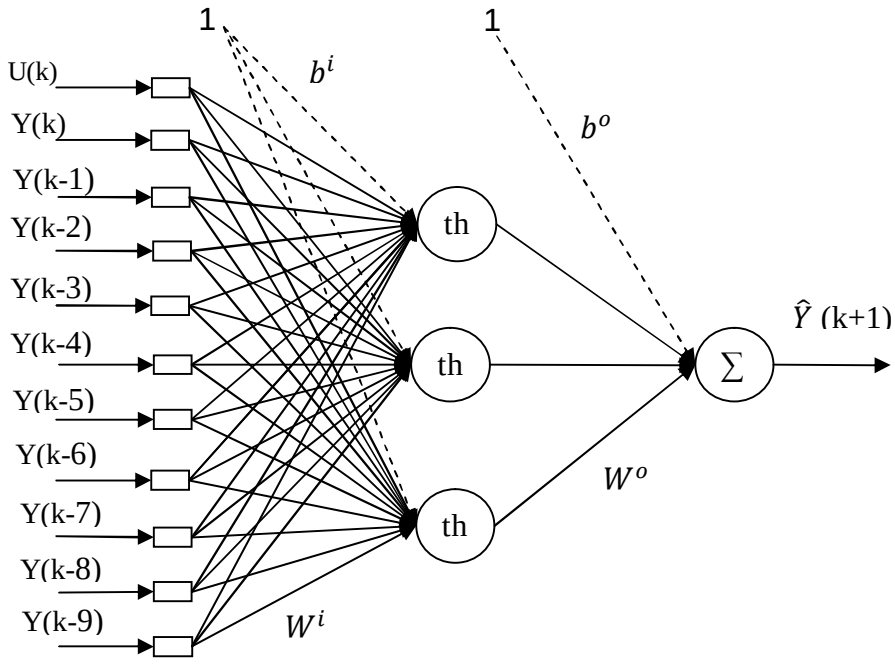


Figure 4.10: Architecture du réseau de neurones multicouches prédicteur à un pas.

Cette architecture appropriée au modèle neuronal multicouches choisie comporte : 11 entrées (1 de commande et 10 valeurs passées de la sortie du système), 3 neurones dans la couche cachée, 1 neurone dans la couche de sortie; tous les neurones utilisés dans la couche

cachée possèdent une fonction de transfert tangente hyperbolique (th), le neurone de sortie est un neurone linéaire.

On partage la base de données du système expérimental (Figure 4.10), en deux parties. La première partie ($0 \leq t \leq 1350$) est réservée à l'apprentissage du réseau, quant à la deuxième partie ($1350 < t \leq 2210$) est consacrée à la validation du réseau.

Les résultats de l'apprentissage et de validation sont illustrés par la Figure 4.11.

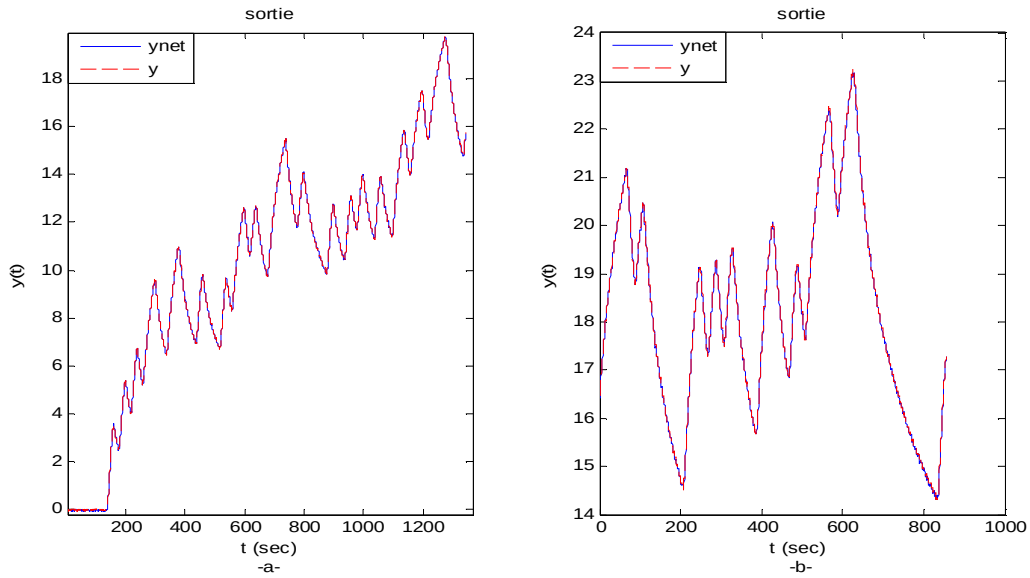


Figure 4.11: Résultats de la sortie obtenus lors de l'apprentissage et du test **-a-** Séquence d'apprentissage. **-b-** Séquence de test.

Le Tableau (4.2), rassemble les EQMA et les EQMT entre les sorties estimées et les sorties du système réel.

	EQMA	EQMT	Itérations
Levenberg Marquardt	0.0010	0.0014	3000

Tableau 4.2: Erreur quadratique moyenne d'apprentissage et du test entre la sortie estimée et la sortie réelle.

D'après la Figure 4.11, on constate une très bonne adéquation entre les courbes estimées et les courbes désirées dans l'étape appropriée à l'apprentissage ainsi que dans l'étape de validation. D'après le Tableau (4.2), on constate que les performances de poursuite sont également acceptables, du moment que les erreurs quadratiques moyennes sont de l'ordre de 10^{-3}.

Par conséquent, le réseau de neurones multicouches prédicteur permet lui aussi de prédire la sortie du système expérimental à l'instant d'échantillonnage suivant.

Conclusion

Dans ce chapitre on a introduit quelques notions de base sur les systèmes thermiques, comme l'équation de la chaleur et la loi de Fick, avec les quelles on a modélisé un système thermiques de géométrie plane; cette étude nous a permet de constater effectivement que les systèmes thermiques à géométrie simple, possèdent un comportement relevant de la dérivation d'ordre non entier.

Pour identifier un système thermique réel à géométrie simple, on a appliqué le réseau de neurones multicouches récurrent, initialement proposé pour identification des modèles fractionnaires théoriques. Bien que ce système fractionnaire soit un procédé réel, notre réseau de neurones multicouches retardé en entrée et en sortie, a parvenu à identifier le système expérimental.

Ensuite, pour prédire la sortie à un pas d'échantillonnage d'un deuxième système thermique réel, régit par un modèle fonction de transfert fractionnaire, retardé et instable, nous avons utilisé le réseau de neurones multicouches prédicteur, proposé au troisième chapitre. Ce réseau de neurones a également réussi à prédire la sortie du système expérimental. Forcément les résultats d'identification obtenus avec les deux systèmes fractionnaires réels sont moins bons que ceux obtenus avec les systèmes théoriques, du faite que, d'une part, le fichier de données utiliser est entaché de bruit, et d'autre part le signal d'entrée appliqué pour l'apprentissage n'est pas assez riche en amplitude. Mais ils restent toujours acceptables.

L'utilisation des deux fichiers de données expérimentales entrées/ sorties, nous ont permet de valider les deux structures neuronales multicouches. Par conséquent le réseau de neurones multicouches retardé en entrée et en sortie, pourra être utilisé comme simulateur à horizon infini des systèmes fractionnaires réels stables et pour n'importe qu'elle entrée. Ainsi que le réseau de neurones prédicteur, comme prédicteur à un pas de n'import quels systèmes fractionnaires.

En revanche, la validation pratique des résultats obtenus avec les deux réseaux de neurones à espace d'état, proposés pour identifier des systèmes fractionnaires théoriques, n'a pas pus ce faire, à cause de l'indisponibilité des variables d'états dans les deux fichiers de données expérimentales.

Conclusion générale :

Les travaux présentés dans ce mémoire ont porté sur l'identification à l'aide des réseaux de neurones de modèles fractionnaires linéaire. La modélisation complète d'un système fractionnaire est conditionnée par un temps de calcul important et une capacité de mémoire infinie. Ces propriétés rendent la tâche du concepteur du système de modélisation très difficile. En revanche des réseaux de neurones permettent l'identification de la dynamique des systèmes fractionnaires avec moins de temps de calcul et une capacité mémoire très faible, grâce à leur propriété d'approximation universelle ainsi que du faite de la prise en compte, uniquement de quelques valeurs passées. Deux nouvelles architectures de réseaux de neurones récurrents ont été proposées. La première architecture, appelée « réseau de neurones multicouches bouclé d'entrée», consiste à mettre en œuvre un réseau de neurones récurrent qui permet de reproduire le caractère fractionnaire du système, une fois ce réseau est entrainé à partir de plusieurs couples entrée/sortie. Quant à la seconde, appelée « réseaux de neurones à espace d'état », utilise la représentation d'état du système pour reconstituer le comportement des états et de la sortie du système.

La démarche proposée comporte :

Une étude détaillée sur les réseaux de neurones artificiels, plus précisément, deux réseaux de neurones récurrents, à savoir le « réseau de neurones multicouches retardé en entrée et en sortie » et le « réseau de neurones à espace d'état ». Le premier réseau s'apprête très bien à la modélisation des équations aux différences non linéaire, alors que le deuxième réseau quant à lui, se rapproche beaucoup de la représentation d'état classique. Deux méthodes d'apprentissages supervisées des réseaux de neurones ont été présentées.

L'identification de deux modèles linéaires différents, d'ordre entier, à l'aide des deux architectures neuronales présentées au deuxième chapitre, a donnée des résultats très satisfaisants. Toutefois, on constate que la performance obtenue avec le « réseau de neurones multicouches retardé en entrée et en sortie » varie inversement par rapport à l'ordre du modèle, Ce qui écarte l'application de cette architecture pour identifier des modèles de dimension infinie. A cet effet, deux nouvelles structures neuronales ont été proposées pour l'identification des modèles fractionnaires. L'application de ces deux architectures à l'identification des modèles fractionnaires de dimension un et de dimension deux, a donnée des résultats satisfaisants. En revanche, ayant accès aux états internes du modèle fractionnaire, le « SSNN modifié » présente une meilleure performance, que le nouveau « réseau multicouches retardé en entrée et en sortie ». Deux autres structures neuronales ont été mises

en œuvre pour la prédiction à un pas d'échantillonnage de la sortie d'un modèle fractionnaire de dimension deux. Les résultats de prédiction obtenus sont excellents.

L'utilisation des deux fichiers de données expérimentaux, a permet de valider les résultats d'identification et de prédiction à un pas, obtenus précédemment avec les deux structures neuronales multicouches. Le « réseau de neurones multicouches retardé en entrée et en sortie » et le « réseau de neurones multicouches prédicteur à un pas » proposés, se sont montrés robustes vis-à-vis de bruits de mesure. Par conséquent on peut dire que les deux réseaux de neurones multicouches proposés sont capables d'identifier la dynamique complexe d'un système fractionnaire réel, en utilisant seulement une seule couche cachée contenant quelques neurones non linéaires.

En conclusion, on peut dire que les réseaux de neurones ont encore une autre fois démontrée leurs aptitudes à identifier des systèmes très complexes, à l'image des systèmes fractionnaires, grâce au réajustement de leurs poids de connexion, de tel sort à compenser l'absence de l'historique du modèle fractionnaire. La dynamique du système est ''captée'' à l'aide de lignes à retard. En outre, les réseaux de neurones artificiels présentent aussi l'avantage de simuler des systèmes fractionnaires en un temps record par rapport aux méthodes de simulation classiques. Les poids de connections obtenus avec les réseaux de neurones n'ont pas de sens physiques comme dans le cas des systèmes linéaires d'ordre entier.

Les résultats de simulation et expérimentales effectuées au moyen des logiciels que nous avons développés sous environnement Matlab nous ont permis grâce aux courbes graphiques obtenues de valider les hypothèses et les choix faits concernant les architectures neuronales. Par ailleurs, ce travail nous a permis à travers les multiples outils de la simulation et l'utilisation de l'algorithme de L-Marquardt, pour l'apprentissage des réseaux de proposer la modélisation complète d'un système fractionnaire.

Quant aux perspectives que nous pouvons lancer au terme de ce travail, elles se résument en quatre points essentiels :

- Valider les résultats du réseau SSNN modifié, sur une base de données d'un système réels d'ordre fractionnaire.
- Utiliser les nouvelles structures neuronales, pour identifier des systèmes fractionnaires non linéaires et multivariables.
- Utilisation d'un autre algorithme d'apprentissage, tel que les techniques de recherche aléatoires et les méthodes métaheuristiques (Dréo et al, 2003).

- Application du réseau de neurones gamma, pour l'identification des systèmes fractionnaires (Khelil et Benyettou, 2007).

Références bibliographiques

Ait messaoud, L. (2007). Contribution à la commande des systèmes par des régulateurs d'ordre non entier, Application à la commande de la machine asynchrone. Mémoire de Magister en Automatique, UMMTO, Tizi-Ouzou.

Amoura ,K. (2010). Réseaux de neurones et espace d'état pour l'identification et la prédiction. Mémoire de Magister en Automatique, UMMTO, Tizi-Ouzou.

Battaglia, J.L., Le Lay, L., Batsale, J.C., Oustaloup, A., et CoisO. (2000). Heat flow estimate Through inverted not integer identification models. In Int. J. of Thermal Science. Vol.39, n± 3, pp 374-389.

Battaglia, J. L. (2002). Méthodes d'identification de modèles à dérivées d'ordre non entier et de réduction modale : Application à la résolution de problèmes thermiques Industriels. Habilitation à Diriger des Recherches, Université de Bordeaux I.

Benchellal, A. (2008). Modélisation des interfaces de diffusion à l'aide d'opérateurs d'intégration fractionnaires. Thèse de Doctorat, Université de Poitiers, France.

Boroomand, A. ,et Bagher, M. M. (2009). Fractional-Based Approach in Neural Networks for Identification Problem. Chinese Control and Decision Conference (CCDC), pp. 2319- 2322.

Charef, A., Sun, H. H., Tsao, Y. Y. and Onaral, B. (Sept 1992). Fractal system as represented by singularity function. IEEE Transactions on Automatic Control, Vol. 37,n° 9.

Djamah, T., Djennoune, S., Bettayeb, M. (2008). Identification of diffusion processes using Fractional non commensurate order models. 2008 5th International Multi-Conference on Systems, Signals and Devices.

Djamah, T. (2009). Identification des systèmes par des modèles d'ordre fractionnaire. Thèse de doctorat en Automatique, UMMTO, Tizi-Ouzou.

Djamah, T., Djennoune, S., Bettayeb, M. (2-4 Juin 2010). Identification fractionnaire multivariables. 6^{éme} conférence Internationale Francophone d'Automatique. Nancy, France.

Dreyfus, G., Martinez, J.M., Samuelides, M., Gordon, M.B., Badran, F., Thiria, S., Hérault,L. (2002). Réseaux de neurones: méthodologie et applications, Eyrolles, Paris.

Dreyfus, G. ,Martinez, J-M., Samuelides, M. ,Gordon, M- B. ,Badran, F. ,Thiria,S. (2007). Apprentissage statistique, Editions EYROLLES.

Dréo, J., Pétrowsky, A., Siarry, P., Taillard, E. (2003). Métaheuristiques pour l'optimisation difficile. Groupe Eyrolles.

Dubois, F., Galucio, A-C. et Point, N. (2008). Introduction à la dérivation fractionnaire Théorie et applications. Editions T.I. Doc. AF 510, *Paris, France.*

Dugowson , S. (1994) . Les différentielles métaphysiques : histoire et philosophie de la généralisation de l'ordre de dérivation. PHD thesis, Université Paris 13, Villetaneuse, France.

Gong, R., Yang, Y., Tian, Y., Zhang, G., Sun, J., Chen, L. (2010). Design and Implement of Neural Network Based Fractiona1 order PI^{α} Controller. Sixth International Conference on Natural Computation (ICNC 2010).

Hamdaoui, K. (2006). Implémentation analogique et numérique de dérivateur et intégrateur d'ordre fractionnaire. Mémoire de Magister en Traitement du signal, Université de Constantine.

Hornik, k., Stinchcombe, M., White, H. and Auer, P. (1994). Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives. Neural computation 6, 1262-1275.

(Int, 2005). Introduction aux réseaux de neurones, Master, 2005.
http://wwwobs.univ-bpclermont.fr/atmos/enseignement/cours-Master-2A/cours_RN_2006.pdf

Khelil, H., Benyettou, A. (March 25-29, 2007). Application du Réseau de Neurones Gamma à la Reconnaissance de la Parole. SETIT 2007 4th International Conference: Sciences of Electronic, Technologies of Information and Telecommunications 2007–Tunisia.

Lucea, M., (2006). Modélisation dynamique par réseaux de neurones et machines à vecteurs supports : contribution à la maîtrise des émissions polluantes de véhicules automobiles. Thèse de Doctorat, Université de Paris 6, Paris.

Mahul, A. (2005). Apprentissage de la qualité de service dans les réseaux multiservices : Application au routage optimal sous contraintes. Thèse de Doctorat à l'Université de Sciences Pour l'Ingénieur de Clermont Ferrand.

Malti, R., Sabatier, J. (2004). Time-domain system identification using fractional models: A benchmark. IFAC Workshop, Fractional Differentiation and its Applications -FDA'04, Bordeaux, France.

Mang-Hui, W., (2005). Member,IEEE. Extension Neural Network-Type 2 and its Applications. IEEE transactions on neural networks, vol. 16,No. 6, pp 1352-1361.

Matignon, D. (1996-a). Recent results in fractional differential systems theory. Technical-Report 96C004, École Nationale Supérieure des Télécommunications, France.

Matignon, D. (1996-b). Stability results for fractional differential equations with applications to control processing. In Proc. IEEE-IMACS Syst. Man Cyber. Conf., Lille, France.

Matignon, D. and Andréa-Novel, B. (1996). Some results on controllability and observability of finite-dimensional fractional differential systems. In Proc. Mathematical Theory of Networks and Systems Symposium, Lille, France.

Matignon, D. and Andréa-Novel, B (1997). Observer-based for fractional differential systems. In Proc. IEEE Conf. Decision & Contr., San Diego, USA.

N'Doye, I. (2011). Généralisation du lemme de Gronwall-Bellman pour la stabilisation des systèmes fractionnaires. Thèse de doctorat en Automatique, l'Université Henri. Poincaré -Nancy 1.

Ossar, Y., (1998). Réseaux d'ondelettes et réseaux de neurones pour la modélisation statique et dynamique de processus. Thèse de Doctorat, Université de Paris 6, Paris.

Oustaloup, A. (1991), La commande CRONE. Editions HERMES, Paris.

Oustaloup, A. (1995). La Dérivation Non Entière : Théorie, Synthèse et Applications. Editions HERMES, Paris.

Oustaloup, A. (1999). La commande CRONE, du scalaire au multivariable. Editions HERMES, Paris.

Oustaloup, A., Sabatier, .J, Lanusse, P., Malti, R., Melchior, P., Moreau, X., Moze, M.(2008). An overview of the CRONE approach in system analysis, modeling and identification, observation and control. Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea.

Parizeau**,** M. (2004). Réseaux De Neurones, GIF-21140 et GIF- 64326, Université de Laval.

Personnaz, L. and Rivals, I. (2003). Réseaux de neurones formels pour la modélisation, la commande, et La classification. CNRS EDITIONS, Paris.

Poinot, T., Trigeassou, J-C. (2004). Identification of Fractional Systems Using an Output-Error Technique. *Nonlinear Dynamics* 38**:** 133–154, 2004. ©2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

Sabatier, J., Aoun, M., Oustaloup, A., Gregoire, G., Ragot, F. et Roy, P. (2006). Estimation of lead acid battery state of charge with a novel fractional model. In*proc IFAC workshop on fractional differentiation and its applications*. pp 362-67, Porto, Portugal.

Sabatier, J., Moze, M. and Farges, C. (2008). On stability of fractional order systems. In Proc. IFAC Workshop on Fractional Differentiation and its Application, Ankara, Turkey.

Vinagre, B. M., Podlubny, I., Hernàndez, A., Feliu, V. (2000). Some approximations of Fractional order operators used in control theory and applications. Fractional calculus App. Anal. Vol. 3, No 3, pp 231-248.

Vinagre, B. M., Monje, C. A. and Caldero, A. J. (2002). Fractional order systems and fractional order actions, tutorial workshop. In Proc. IEEE Conf. Decision & Contr., Las Vegas, USA.

Zamarreno, J.M., Vega, P., Garcia, L.D., Francisco, M., (2000). State-space neural network for modelling, prediction and control. Control Engineering Practice, vol. 8, no. 9, pp. 1063-1075.