Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mouloud MAMMERI de Tizi-Ouzou

Faculté de Génie Électrique et Informatique

Département d'Automatique

MÉMOIRE DE MAGISTER

En Automatique

Présenté par :

M^{elle}. Lynda AMIMER

Ingénieur UMMTO

Thème

Modélisation et Commande des Systèmes Non Linéaires Fractionnaires par des Réseaux de Neurones Fractionnaires

Mémoire soutenu le 11 novembre 2015 devant le jury composé de :

Salah HADDAD	*Professeur*	UMMTO	Président
Rachid MANSOURI	*Professeur*	UMMTO	Rapporteur
Ahmad MAIDI	*Maître de Conférences classe A*	UMMTO	Examinateur
Amar SI AMMOUR	*Maître de Conférences classe A*	UMMTO	Examinateur

Remerciement

Ce travail a été effectué au Laboratoire de Conception et Conduite des Systèmes de Production (L2CSP), de l'université Mouloud Mammeri de Tizi-Ouzou.

Je tiens à remercier, en premier lieu, mon promoteur M^r Rachid MANSOURI, Professeur à l'UMMTO pour m'avoir fait profiter de son enthousiasme, de sa rigueur scientifique, de son expérience et pour m'avoir fait confiance tout au long de Mémoire.

Je remercie également M^r Salah HADDAD, Professeur à l'université Mouloud Mammeri de Tizi-Ouzou, M^r Ahmad MAIDI, Maître de conférences classe A à l'université Mouloud Mammeri de Tizi-Ouzou, et M^r Amar SI-AMMOUR, Maître de conférences classe A à l'université Mouloud Mammeri de Tizi-Ouzou, d'avoir accepté de participer à ce jury.

Je tiens aussi à remercier en particulier M^r Sofiane HAMMOUCHE, Maître assistant classe B, pour son aide et son précieux soutien.

Mes remerciements s'adressent également à tous les membres du Laboratoire de Conception et Conduite des Systèmes de Production (L2CSP), en particulier son directeur M^r Mohamed Aidene, Professeur à l'université Mouloud Mammeri de Tizi-Ouzou, pour leur sympathie et l'excellente ambiance de travail qu'ils ont créée.

La réalisation de ce Mémoire ne serait être possible sans le soutien inconditionnel des proches tout au long de ce travail.

Notations

$I^k[f(t)]$:	$(k \in \mathbb{N})$, l'intégration répétée k fois de la fonction $f(t)$
$I^\alpha[f(t)]$:	$(\alpha \in \mathbb{R})$, l'intégration non entière d'ordre α de la fonction $f(t)$
${}_a^{RL}D_t^\alpha f(t)$:	dérivée d'ordre α de la fonction $f(t)$ nulle pour $t \leq a$ selon la définition de Riemann-Liouville
${}_a^{C}D_t^\alpha f(t)$:	dérivée d'ordre α de la fonction $f(t)$ nulle pour $t \leq a$ selon la définition de Caputo
$D^\alpha f(Kh)$:	désigne la valeur de la dérivée $\alpha^{ème}$ de $f(t)$ à l'instant Kh
D^α :	opérateur de dérivation d'ordre non entier α
$\Gamma(\alpha)$:	$(\alpha \in \mathbb{R}^* \backslash \mathbb{Z}_-)$ Fonction d'Euler Gamma
$\Phi_\alpha(t)$:	facteur d'oubli
$\mathcal{L}$:	symbole de la transformation de Laplace
s :	opérateur de Laplace
$\begin{pmatrix} n \\ j \end{pmatrix}$:	$(n \in \mathbb{N})$, désigne le binôme de Newton
$\begin{pmatrix} \alpha \\ j \end{pmatrix}$:	$(\alpha \in \mathbb{R}_+)$, désigne le binôme de Newton généralisé à des ordres non entiers
h :	$h \in \mathbb{R}^{*+}$ période d'échantillonnage
Δ_h^n :	opérateur de différence d'ordre entier 'n' de la fonction $f(t)$ avec un pas d'échantillonnage h
Δ_h^α :	opérateur de différence généraliser d'ordre non entier 'α' de la fonction $f(t)$ avec un pas d'échantillonnage h

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Un système est un ensemble d'éléments interconnectés, les uns aux autres, afin de réaliser une fonction donnée. Si le système a une caractéristique d'évolution dans le temps, on parle de système dynamique. Pour la compréhension et la maîtrise du comportement des systèmes, on fait souvent appelle à la modélisation.

La modélisation consiste à trouver un modèle paramétré dont le comportement dynamique approche celui du système, cette représentation est utilisée pour la simulation des systèmes dans le but de la conception et la commande des systèmes, la détection des anomalies de fonctionnement et le diagnostic des pannes [18].
Principalement, on a deux grandes classes de modèles qui sont : le modèle de connaissance et le modèle de comportement (expérimental) [44]. Ce classement est établi à propos des informations qui sont utilisées pour leurs constructions.

La conception des *modèles de connaissance* est basée sur l'analyse de systèmes physiques, chimiques, biologiques,...etc, en appliquant les lois générales fondées sur des principes et lois de la mécanique, l'électromagnétisme, de la thermodynamique ...etc.

Toutefois, dans la pratique, on rencontre souvent des obstacles pour la construction des modèles de connaissance soit à cause de la complexité des processus, soit à cause des phénomènes qui sont mal connus. De plus, afin de simplifier les équations, on néglige quelques paramètres physiques, cela permet d'obtenir des modèles moins précis, ce qui nous pousse à la conception des modèles basés sur la mesure des entrées/sorties des systèmes à modéliser, c'est ce qu'on appelle les modèles à comportement.

La procédure de la conception des *modèles à comportement*, également appelée identification, consiste à poser une structure mathématique paramétrique qui représente une relation entre l'entrée et la sortie. Pour identifier les paramètres de l'équation, on utilise les mesures disponibles. On applique alors des méthodes d'optimisation, afin d'estimer les paramètres du modèle choisi,

tout en essayant d'obtenir la meilleure précision [45, 46].

Dans le but de modéliser les systèmes dynamiques, il existe plusieurs représentations mathématiques à savoir les équations différentielles, la représentation par fonctions de transfert, la représentation d'état et la représentation par réseaux de neurones que nous utiliserons dans notre travail.

Avant d'entamer la description de chaque modèle, on définit deux grandes classes de systèmes dynamiques qui sont les systèmes linéaires, et les systèmes non linéaires. Chaque classe a ses propres modèles mathématiques.

Du point de vue mathématique, un système est *linéaire* lorsque son comportement vérifie le principe de superposition. Du point de vue physique, un système est linéaire s'il est décrit par des équations différentielles linéaires d'ordre fini et à coefficients constants.

Le comportement d'un système linéaire et invariant dans le temps est régi par une équation différentielle linéaire à coefficients constants ayant la forme générale :
$a_n D^n y(t) + a_{n-1} D^{n-1} y(t) + ... + a_1 Dy(t) + a_0 y(t) = b_m D^m u(t) + b_{m-1} D^{m-1} u(t) + ... + b_1 Du(t) + b_0 u(t)$
$u(t)$ étant l'entrée du système et $y(t)$ sa sortie.
n est l'ordre du système linéaire qui est par définition l'ordre de son équation différentielle.

La *fonction de transfert* d'un système linéaire à temps invariant est le rapport de la transformée de Laplace de la sortie sur la transformée de Laplace de l'entrée avec les conditions initiales nulles. On peut aussi définir la fonction de transfert comme étant la transformée de Laplace de la réponse impulsionnelle du système avec les conditions initiales nulles.

En appliquant la transformée de Laplace à l'équation différentielle, avec les conditions initiales nulles, le transfert entre $U(s)$ et $Y(s)$ est donné par :

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{(m-1)} s^{(m-1)} + ... + b_1 s + b_0}{a_n s^n + a_{(n-1)} s^{(n-1)} + ... + a_1 s + a_0} = \frac{N(s)}{D(s)} \quad (1)$$

Le modèle d'état est une autre alternative pour décrire un système linéaire.

On appelle **état d'un système**, un ensemble de variables qui, étant connues à l'instant initial, permettent de décrire l'évolution de ce système.

Soit un système multi-entrées (m entrées), multi-sorties (p sorties), dont le modèle est décrit par une ou plusieurs équations différentielles linéaires à coefficients constants. Ce modèle peut s'écrire sous la forme d'un système d'équations différentielles matricielles du premier ordre :

$$\begin{cases} \dot{x}(t) = A.x(t) + B.u(t) & \text{Equation de la commande} \\ y(t) = C.x(t) + D.u(t) & \text{Equation de l'observation} \end{cases} \tag{2}$$

Avec : $x \in \mathbb{R}^n$ est le vecteur d'état qui représente les n variables d'état, $y \in \mathbb{R}^q$ est le vecteur de sortie, représentant les q mesures, $u \in \mathbb{R}^p$ est un vecteur de commandes qui représente les p commandes. A, B, C et D sont des matrices à coefficients constants.
Cette représentation d'état n'est pas unique et dépend du vecteur d'état choisi.

Par définition, un *système non linéaire* est un système qui n'est pas linéaire, c'est-à-dire (au sens physique) ne peut pas être décrit par des équations différentielles linéaires à coefficients constants.

Les systèmes non linéaires peuvent être modélisés par différentes structures non linéaires, à savoir les équations différentielles non linéaires et la représentation d'état non linéaire.

L'équation différentielle non linéaire est définie par :

$$D^n y(t) = g(D^{n-1}y(t), D^{n-2}y(t), ..., Dy(t), u(t)) \tag{3}$$

n est l'ordre de l'équation différentielle non linéaire, $g(.)$ est la fonction non linéaire.

Aussi le système non linéaire peut être décrit par la représentation d'état non linéaire de la forme

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)) \\ y(t) = g(x(t), u(t)) \end{cases} \tag{4}$$

avec : $f(.)$, $g(.)$ étant les fonctions non linéaires

Les chercheurs ont proposé un autre moyen de modéliser des systèmes en général et des systèmes dynamiques en particulier, il s'agit des réseaux de neurones formels. L' origine de ce modèle est l'étude du cerveau humain qui a commencé depuis des milliers d'années. Le premier pas vers les réseaux de neurones artificiels a débuté en 1943, lorsque le neurophysiologiste Warren McCulloch et le jeune mathématicien Walter Pitts ont écrit un article [2] sur le fonctionnement d'un neurone.

La modélisation par des réseaux de neurones a une grande puissance de généralisation qui lui permet de représenter une large gamme de systèmes dynamiques linéaires et non linéaires [3].

Dans la pratique, il existe des systèmes très complexes et leurs modélisations par des modèles entiers marquent de faibles précisions. Pour cela plusieurs chercheurs ont introduit le calcul

factionnaire dans les modèles classiques [5, 17, 15]. Un modèle d'ordre fractionnaire est la généralisation d'un modèle entier.

La dérivée d'ordre fractionnaire est caractérisée par la propriété de mémoire longue, le calcul de cette dérivée nécessite la connaissance de tout le passé de la fonction, ce qui implique la caractéristique de dimension infinie. Contrairement à la dérivée entière qui ne fait intervenir qu'un nombre limité de valeurs du passé de la fonction à dériver.

Les réseaux de neurones artificiels ont été introduits dans plusieurs travaux, pour la modélisation et la commande de processus [9], ils sont utilisés comme outil de résolution des équations différentielles d'ordre fractionnaire [13]. Les réseaux de neurones de type "feed-forward" ont été utilisés pour la réalisation du correcteur PID d'ordre fractionnaire (FOPID).[4].

Dans le but de modéliser les systèmes non linéaires fractionnaires, on a besoin de sélectionner une structure au modèle qui permet de représenter fidèlement le comportement du système. Dans ce contexte, on propose d'introduire le calcul fractionnaire au réseau de neurones. Deux structures peuvent être envisagées. La première consiste à concevoir de nouvelles architectures des réseaux de neurones, en introduisant l'aspect fractionnaire dans les fonctions d'activation [13]. La deuxième structure que nous utilisons dans notre travail est l'association de réseau de neurones formel qui représente les systèmes non linéaires, avec une bonne précision, et un bloc d'intégrateurs fractionnaires qui permet de représenter l'aspect fractionnaire du système, on l'appellera modèle Réseau de Neurones-Intégrateurs Fractionnaires, on le notera (RdNs-IsFs) [1].

L'objectif de ce mémoire est l'utilisation des RdNs-IsFs pour la modélisation des systèmes non linéaires d'ordre fractionnaire.

Dans notre étude, nous nous plaçons dans le cadre de modèles à temps discret qui se prête bien à l'utilisation des réseaux de neurones qui présentent l'aspect non linéaire. Nous avons utilisé la méthode de Grünwald-Letnikov pour le calcul des dérivées non entières pour représenter l'aspect fractionnaire du modèle.

Le mémoire est organisé comme suit :

Le chapitre 1, représente le modèle réseau de neurones classique. Nous développerons ses différentes composantes et architectures, et les différentes méthodes d'apprentissages qui sont l'algorithme de rétropropagation de l'erreur et l'algorithme Levenberg-Marquardt utilisant la régularisation bayésienne.

Le chapitre 2, est dédié au calcul d'ordre fractionnaire. Nous présenterons les différentes méthodes d'approximation de la dérivée d'ordre fractionnaire. Ce qui nous permet de mettre en oeuvre les systèmes d'ordre fractionnaire. Dans notre cas, nous prendrons la représentation d'état non linéaire d'ordre fractionnaire comme système de référence pour l'apprentissage des neurones.

Dans le chapitre 3, Nous présenterons le modèle Réseau de Neurones-Intégrateurs Fractionnaires qui permet de modéliser les systèmes non linéaires d'ordre fractionnaire. Nous proposerons l'algorithme à suivre pour la conception de ce modèle qui est une combinaison d'un réseau de neurones artificiel et un ensemble d'intégrateurs d'ordre fractionnaire. Deux(02) exemples d'application seront donnés pour illustrer la puissance de la structure choisie pour la modélisation des systèmes non linéaires d'ordre fractionnaire.

Nous terminerons ce mémoire par une conclusion générale récapitulant les résultats obtenus et quelques perspectives.

Chapitre 1

Réseaux de Neurones Artificiels

Chapitre 1

Réseaux de Neurones artificiels

1.1 Introduction

Les réseaux de neurones artificiels sont une inspiration du système nerveux humain. Leur développement est issu d'une volonté de l'homme de comprendre et d'imiter les capacités du cerveau, afin de synthétiser des systèmes qui remplacent l'homme dans la réalisation des tâches complexes, comme la mémorisation, l'apprentissage, l'intelligence, la généralisation...etc.

Le réseau de neurones artificiel est composé de plusieurs opérateurs non linéaires, les neurones formels, qui fournissent un signal de sortie en fonction d'un certain nombre de signaux d'entrées diversement pondérés. Ces neurones peuvent être connectés de diverses manières afin de constituer une technique de traitement de données bien comprise et maîtriser selon l'architecture neuronale, afin de résoudre les problèmes qui intéressent différents aspects scientifiques, comme le contrôle et la commande de processus, la modélisation, la classification, la prédiction, le diagnostic, la reconnaissance de forme,...etc.

Dans ce chapitre, nous introduisons la notion des réseaux de neurones artificiels, on expose en détail les différents composants d'un réseau y compris les neurones artificiels, leurs fonctions d'activation ainsi que leurs différentes façons d'organisation et de connexion, qui nous permettent d'avoir plusieurs structures de réseau de neurones artificiel. Ce dernier doit se doter d'une technique, qui lui assure la récolte d'informations à partir du monde extérieur ou simplement d'une règle d'apprentissage qui permet au réseau de s'adapter à l'environnement, par l'ajustement et l'actualisation de ses poids, qui représentent la force de connexion.

On peut noter que le réseau de neurones artificiel est un modèle qui peut représenter une

large gamme des systèmes non linéaires, car il possède des caractéristiques spécifiques, telles que la capacité d'apprentissage et d'adaptation et la capacité de généralisation.

1.2 Historique

En 1943 McCulloch et Pitts ont proposé des neurones formels [2, 6] mimant les neurones biologiques, capables de mémoriser les fonctions booléennes simples. Les réseaux de neurones artificiels réalisés à partir de ce type de neurones sont ainsi inspirés du système nerveux. Ils sont conçus pour reproduire certaines caractéristiques des mémoires biologiques par le fait qu'ils sont :

- Massivement parallèles.
- Capables d'apprendre.
- Capables de mémoriser l'information dans les connexions entre les neurones.
- Capables de traiter des informations incomplètes.

En 1949, Hebb a mis en évidence l'importance du couplage synaptique dans l'apprentissage par le renforcement ou la dégénérescence des liaisons inter-neuronales lors de l'interaction du cerveau avec le milieu extérieur.

Le premier modèle opérationnel est le perceptron inspiré du modèle visuel. Il a été proposé en 1958 par Rosenblatt [7]. Les limites du perceptron monocouche du point de vue performance ont été montrées en 1969 par les mathématiciens Minsky et Papert.

Les travaux de Hopfield en 1982 [47], ont montré que des réseaux de neurones artificiels étaient capables de résoudre des problèmes d'optimisation. Et ceux de Kohonen (1982) [48] ont montré qu'ils étaient capables de résoudre des tâches de classification et de reconnaissance.

Aujourd'hui, les réseaux de neurones artificiels ont des nombreuses applications dans le secteur du contrôle comme : la commande de processus, le diagnostic et l'asservissement de robots.

1.3 Notion de neurone biologique

La compréhension et la modélisation du cerveau ont permis d'isoler le composant cellulaire de base du cerveau appelé neurone [8], qui est l'unité de traitement de l'information dont les constituants sont les synapses, les dendrites, le noyau et l'axone comme le montre clairement la figure 1.1 :

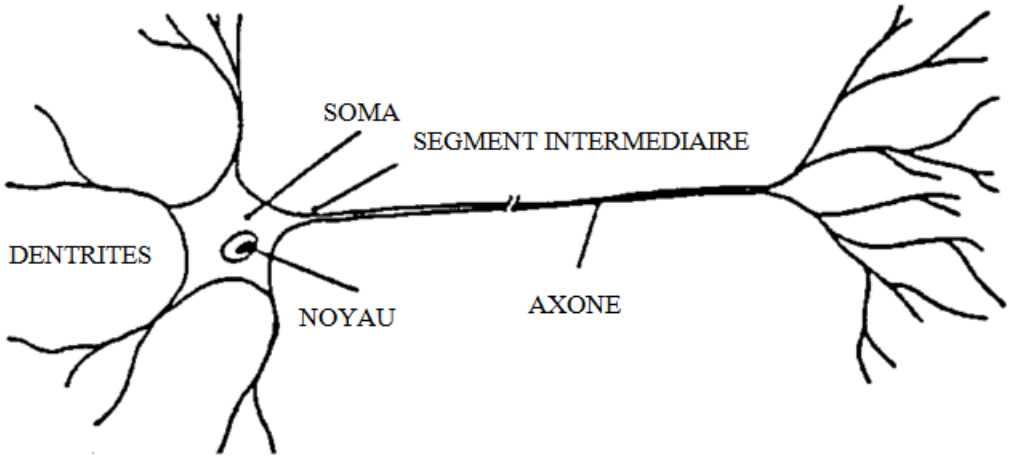


FIGURE 1.1: Neurone biologique

Chaque neurone reçoit un ensemble de potentiels excitateurs ou inhibiteurs, par l'intermédiaire des synapses qui le relient aux autres neurones, les dendrites calculent une somme pondérée de leurs entrées, selon le niveau d'activation obtenu, le noyau génère ou non un potentiel d'action qui se propage le long de l'axone. Ainsi, ce modèle biologique simple sert de base au modèle mathématique du neurone formel.

1.4 Neurone formel

Un neurone est un automate indépendant MISO [11] (multi inputs, single output), dont l'état de sortie, est une valeur scalaire Y_k.

En général, on considère que chaque neurone fournit une information additive aux unités de calcul (neurones) qui lui sont connectées. La valeur nette ou règle de propagation de neurone k, k_{Net} est simplement, la somme pondérée des sorties des différents neurones en amont avec lesquels il est connecté, plus une valeur d'offset (biais). Chaque neurone formel a la structure de la figure 1.2 ; il est constitué de :

- **Entrées :** sont directement les entrées du système ou peuvent provenir d'autres neurones.
- **Biais :** sont les entrées qui sont toujours mises à 1 et qui permettent d'ajouter la flexibilité au réseau en variant le seuil de déclenchement du poids de biais lors de l'apprentissage.
- **Poids :** sont les facteurs multiplicateurs qui affectent l'influence de chaque entrée sur la sortie du neurone.

- **Noyau :** intègre toutes les entrées et le biais et calcule la sortie du neurone selon une fonction d'activation qui est souvent non linéaire pour donner une plus grande flexibilité d'apprentissage.
- **Sortie :** la sortie du réseau de neurones peut être distribuée vers d'autres neurones.

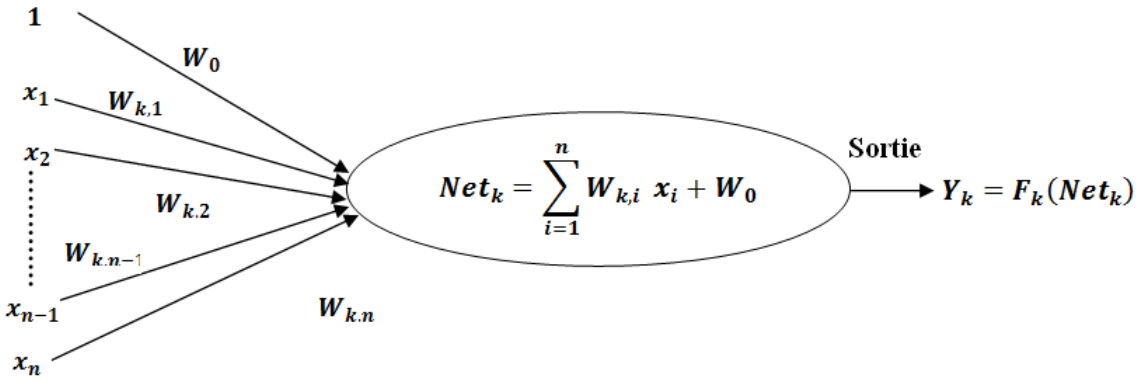


FIGURE 1.2: Structure générale d'un neurone formel.

Les paramètres du neurone sont :

- x_n :la $n^{ème}$ entrée du $k^{ème}$ neurone.
- $W_{k,j}$: poids associé à la $j^{ème}$ entrée du neurone k.
- $Net_k = \sum_{i=1}^{n} W_{k,i}x_i + W_0$: règle de propagation.
- W_0 : valeur d'offset ou biais interne.
- F_k : fonction d'activation.
- Y_k : sortie du neurone k.

1.4.1 Fonctions d'activation

L'utilisation d'une fonction d'activation non linéaire, permet au Réseau de neurones Artificiels (R.N.A) de modéliser des équations dont la sortie n'est pas une combinaison linéaire des entrées, cette caractéristique confie au R.N.A une grande capacité de modélisation fortement appréciées pour résoudre des problèmes non linéaires. Les fonctions d'activation souvent utilisées sont :

1. **Fonction binaire à seuil**
   - Fonction Heaviside montrée par la figure 1.3(a) et définie par :

$$h(x) = \begin{cases} 1 & si \quad \text{x} > 0 \\ 0 & sinon \end{cases} \tag{1.1}$$

   - Fonction Signe montrée par la figure 1.3(b). Elle est définie par :

$$sgn(x) = \begin{cases} 1 & si \quad x > 0 \\ \text{-1} & sinon \end{cases} \tag{1.2}$$

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

2. **Fonction linéaire**

   C'est l'une des fonctions d'activations les plus simples, cette fonction est représentée par la figure 1.3(c), sa fonction est définie par :

$$F(x) = x \tag{1.3}$$

3. **Fonction linéaire à seuil ou multi-seuils**

   On peut la définir comme suit :

$$F(x) = \begin{cases} x & x \in [u, v] \\ v & si \quad x < v \\ u & si \quad x > u \end{cases} \tag{1.4}$$

   Cette fonction représente un compromis entre la fonction linéaire et la fonction seuil. Son graphe est représenté par la figure 1.3(d) : entre ses deux barres de saturation, elle confère

au neurone une gamme de réponses possibles. En modulant la pente de la linéarité, on affecte la plage de réponse du neurone.

4. **Fonction log sigmoïde**

   Elle est l'équivalent continu de la fonction linéaire représentée par la figure 1.3(e). Etant continue, elle est dérivable, d'autant plus que sa dérivée est simple à calculer, la fonction log sigmoïde est définie par :

$$logsig(x) = \frac{1}{1 + e^{-x}} \tag{1.5}$$

   et sa dérivée est :

$$\frac{d}{dx}\left(logsig(x)\right) = logsig(x)(1 - logsig(x)) \tag{1.6}$$

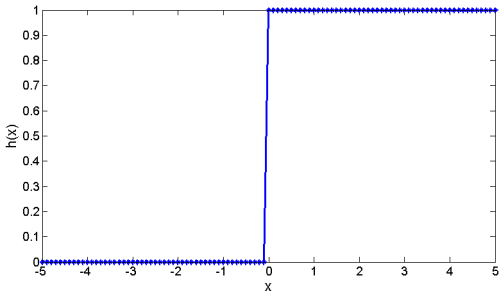
5. **Fonction tangente sigmoïde**

   la fonction tangente sigmoïde est celle montrée par la figure 1.3(f). Elle est définie par :

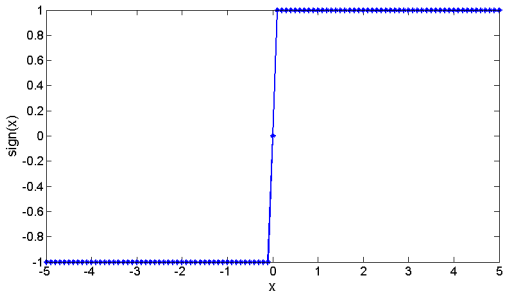
$$tansig(x) = \frac{2}{(1 + e^{-2x})} - 1 \tag{1.7}$$

   et sa dérivée est définie comme suit :

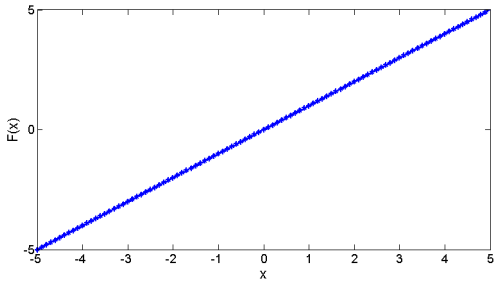
$$\frac{d}{dx}\left(tansig(x)\right) = \frac{4e^{-2x}}{(e^{-2x} + 1)^2} \tag{1.8}$$



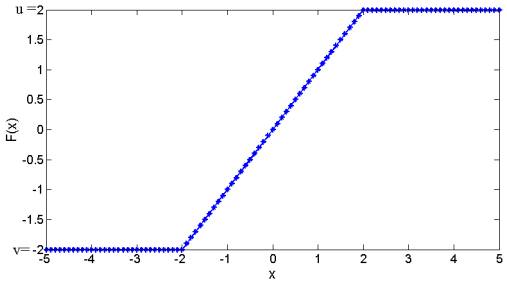
(a) Fonction Heaviside



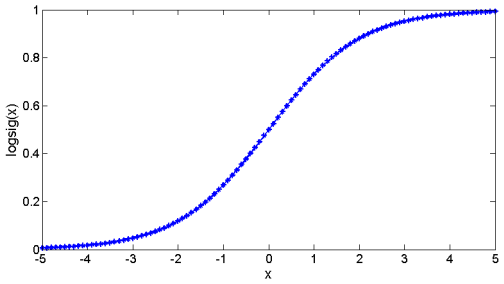
(b) Fonction signe



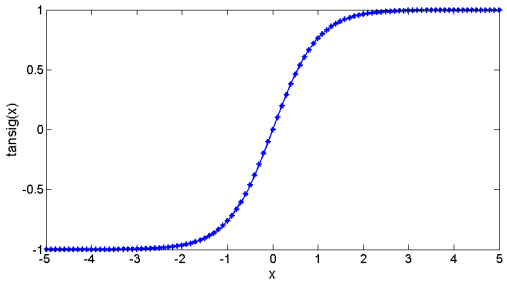
(c) Fonction linéaire



(d) fonction linéaire à seuil



(e) Fonction log sigmoïde



(f) Fonction tangente sigmoïde

FIGURE 1.3: Différentes fonctions d'activation

1.5 Différents types de réseaux de neurones

Il existe une panoplie impressionnante d'architectures de réseaux de neurones. Lorsque les neurones sont branchés entre eux d'une façon complète, la difficulté principale devient alors de trouver un algorithme efficace à son apprentissage.

1.5.1 Architecture multicouche

Dans ce type d'architecture, on distingue deux familles de réseaux : les réseaux bouclés et les réseaux non bouclés.

Réseaux de neurones non bouclés

Un réseau est dit non bouclé (static/feedforward) si et seulement si son graphe est acyclique[10].

Un réseau non bouclé réalise une fonction non linéaire de ses entrées. Il existe deux architectures classiques :

- Les réseaux de neurones complètement connectés.
- Les réseaux de neurones à couches (RNC) appelés aussi perceptrons multicouches (PMC).

Dans un réseau de neurones complètement connecté, chaque neurone est commandé par tous les neurones d'indices inférieurs, la figure 1.4 présente un exemple d'un réseau de neurones complètement connecté. S'il y a plusieurs sorties, elles ne sont pas connectées entre elles.

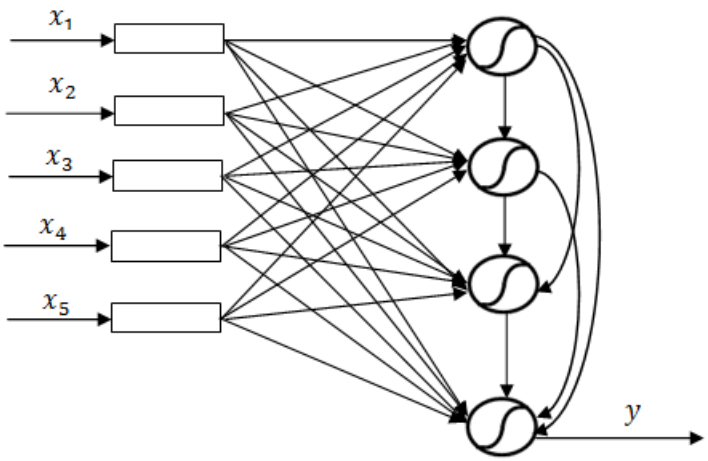


FIGURE 1.4: Réseau de Neurones non bouclé complètement connecté

Dans un réseau de neurones à couches, il y a une couche d'entrée, une ou plusieurs couches

de neurones cachés et une couche de neurones de sortie, la figure 1.5 présente l'architecture d'un réseau de neurones non bouclé et ses différentes couches. Chaque neurone est commandé par tous les neurones de la couche précédente.

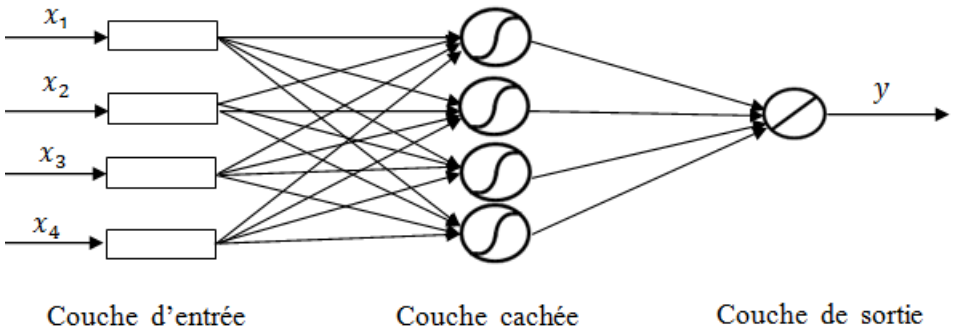


FIGURE 1.5: Réseau de Neurones non bouclé à couches.

Réseaux de neurones bouclés

Les réseaux de neurones bouclés peuvent avoir une topologie de connexions quelconque, comprenant notamment des boucles qui ramènent aux entrées la valeur d'une ou plusieurs sorties. La figure 1.6 montre un exemple d'un réseau de neurones bouclé.[16]

Pour qu'un tel système soit causal, il faut évidemment qu'à toute boucle soit associé un retard, un réseau de neurones bouclé est donc un système dynamique, régi par des équations différentielles. Comme la majorité des applications sont réalisées par des programmes d'ordinateurs, on se place dans le cadre des systèmes à temps discret, où les équations différentielles sont remplacées par des équations aux différences.

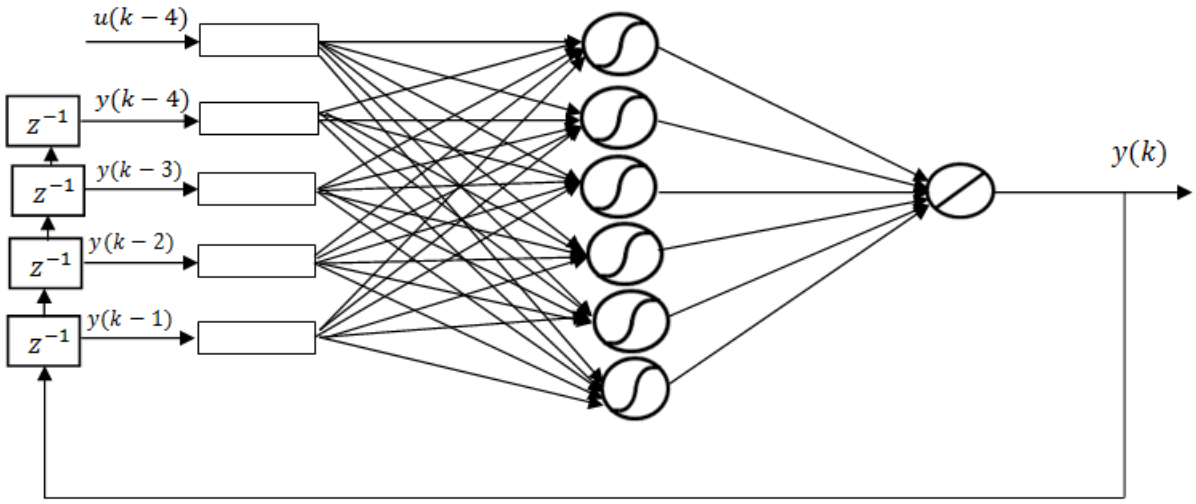


FIGURE 1.6: Réseau de neurones bouclé

1.6 Architecture des réseaux

Les connexions entre les neurones qui composent le réseau décrivent la topologie du modèle. Elle peut être quelconque, mais le plus souvent il est possible de distinguer une certaine régularité (réseau à connexion complète).

1.6.1 Réseau monocouche

La structure d'un réseau monocouche est telle que des neurones organisés en entrée soient entièrement connectés à d'autres neurones organisés en sortie par une couche.

1.6.2 Réseau multicouches

Dans ce cas, les neurones sont arrangés par couches. Il n'y a pas de connexion entre neurones d'une même couche, les connexions ne se font qu'avec les neurones de couches avals. Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante. Ceci permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc de définir les concepts de neurone d'entrée et de neurone de sortie. Par extension, on appelle couche d'entrée l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelées les couches cachées. La figure 1.7 illustre l'exemple de R.N.A multicouches.

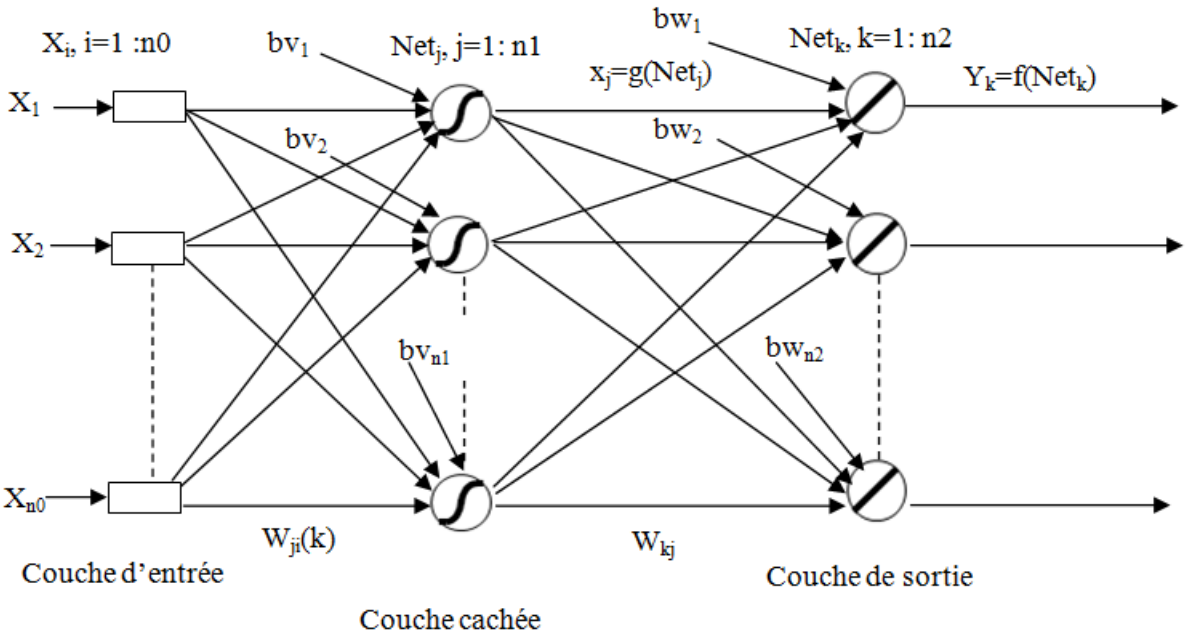


FIGURE 1.7: Réseau de Neurones multicouches à n0 entrées, n1 neurones cachés, n2 neurones de sortie.

- X_i, X_j, Y_k : représentent respectivement le vecteur d'entrée « i » de la couche d'entrée, la valeur de la sortie du neurone « j » de la couche cachée et celle du neurone « k » de la couche de sortie.
- Net_j et Net_k : représentent respectivement le potentiel des neurones de la couche cachée et potentiel des neurones de la couche de sortie.
- W_{ji} : poids synaptique entre le neurone amont i et le neurone aval j.
- W_{kj} : poids synaptique entre le neurone amont j et le neurone aval k.
- $f(.)$ et $g(.)$: représentent les fonctions d'activation de la couche de sortie et de la couche cachée (généralement $g(.)$ est une fonction non linéaire et $f(.)$ est une fonction linéaire).
- bv_j et bw_k : représentent les biais de la couche cachée et de la couche cachée de sortie.
- n_0, n_1 et n_2 : représentent respectivement le nombre d'entrées, le nombre de neurones de la couche cachée et le nombre de neurones de la couche de sortie.

1.7 Modèles des réseaux de neurones

1.7.1 Modèle de Hopfield

Le modèle de Hopfield fut présenté en 1982 [47]. Ce modèle très simple est basé sur le principe des mémoires associatives. C'est d'ailleurs la raison pour laquelle ce type de réseau est dit associatif (par analogie avec le pointeur qui permet de récupérer le contenu d'une case mémoire).
Le modèle de Hopfield utilise l'architecture des réseaux entièrement connectés et récurrents (dont les connexions sont non orientées et où chaque neurone n'agit pas sur lui-même). Les sorties sont en fonction des entrées et du dernier état pris par le réseau.

1.7.2 Modèle de Kohonen

Ce modèle a été présenté par Kohonen en 1982 [48] en se basant sur des constatations biologiques.

Il a pour objectif de présenter des données complexes et appartenant généralement à un espace discret de grandes dimensions dont la topologie est limitée à une ou deux dimensions.

Les cartes de Kohonen sont réalisées à partir d'un réseau à deux couches, une en entrée et une en sortie. Il faut noter que les neurones de la couche d'entrée sont entièrement connectés à la couche de sortie.

Les neurones de la couche de sortie sont placés dans un espace d'une ou de deux dimensions en général, chaque neurone possède donc des voisins dans cet espace. Chaque neurone de la couche de sortie possède des connexions latérales récurrentes dans sa couche (le neurone inhibe les neurones éloignés et laisse agir les neurones voisins).

1.7.3 Modèle perceptron

Le mécanisme perceptron fut inventé par le psychologue FRANK Rosenblat à la fin des années 50. Il représentait sa tentative d'illustrer certaines propriétés fondamentales des systèmes intelligents en général. Le réseau dans ce modèle est formé de trois couches : Une couche d'entrée (la rétine), fournissant des données à une couche intermédiaire, chargée des calculs, cela en fournissant la somme des impulsions qui lui viennent des cellules auxquelles elle est connectée, et elle répond généralement suivant une loi définie avec un seuil, elle-même connectée à la couche de

sortie (couche de décision), représentant les exemples à mémoriser. Seule cette dernière couche renvoie des signaux à la couche intermédiaire, jusqu'à ce que leurs connexions se stabilisent.

1.7.4 Modèle Adaline

Le modèle d'adaline (Adaptatif Linear Neurone) de Widrow et Hoff est un réseau à trois couches : une d'entrée, une couche cachée et une couche de sortie. Ce modèle est similaire au modèle de perceptron, seule la fonction d'activation change, mais reste toujours linéaire : $F(x) = x$.

1.8 Apprentissage

L'apprentissage 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.
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'échantillons d'apprentissage ainsi qu'un algorithme d'apprentissage.

Après initialisation des poids du réseau (en général 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 existe deux grandes classes selon que l'apprentissage est dit supervisé en anglais «supervised Learning » ou non supervisé en anglais « unsupervised Learning ».

Dans le cas de l'apprentissage supervisé qui est représenté par le schéma bloc de la figure 1.8 les données sont des couples (Entrée, Sortie associée). Ce type d'apprentissage est destiné à reproduire un processus quelconque (chimique, mécanique, financier...) dont on connaît seulement quelques variables et résultats correspondants.

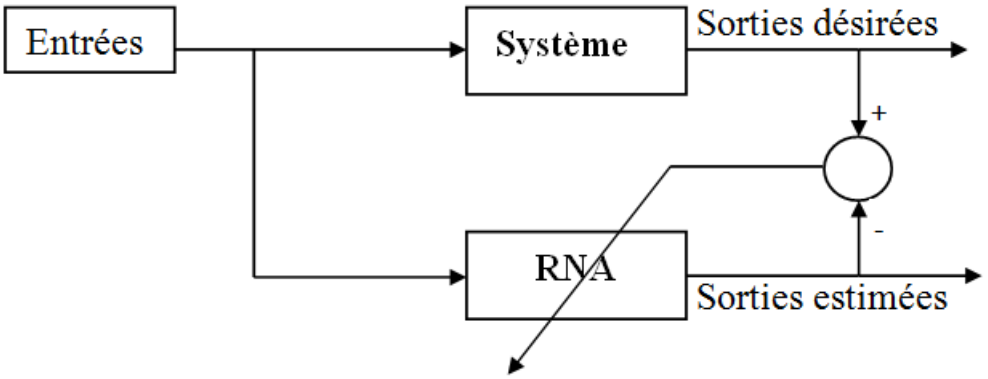


FIGURE 1.8: Apprentissage supervisé

Alors que l'on ne dispose que des valeurs (Entrées) pour l'apprentissage non supervisé c'est ce que montre la figure 1.9. Ce type d'apprentissage est utilisé par exemple en classification lorsque les classes aux quelles doivent appartenir les données ne sont pas connues a priori.

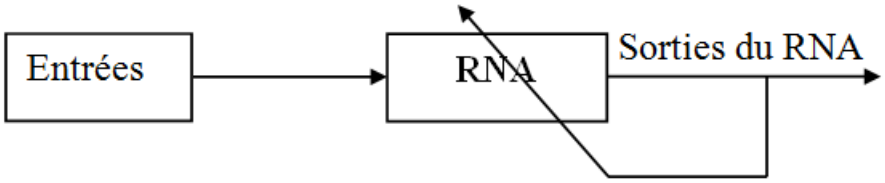


FIGURE 1.9: Apprentissage non supervisé

1.8.1 Rétro-propagation du gradient de l'erreur

Cet algorithme est utilisé dans les réseaux de type feedforward [12], ce sont des réseaux de neurones à couches, ayant une couche d'entrée, une couche de sortie, et au moins une couche cachée. Il n'y a pas de récursivité dans les connexions, et pas de connexions entre neurones de la même couche. Le principe de la rétro-propagation consiste à présenter au réseau un vecteur d'entrées, de procéder au calcul de la sortie par propagation à travers les couches, de la couche d'entrée vers la couche de sortie en passant par les couches cachées. Cette sortie obtenue est comparée à la sortie désirée, une erreur est alors obtenue. A partir de cette erreur, est calculé le gradient de l'erreur qui est à son tour propagé de la couche de sortie vers la couche d'entrée, d'où le terme de rétro-propagation. Cela permet la modification des poids du réseau et donc l'apprentissage. L'opération est réitérée pour chaque vecteur d'entrée et cela jusqu'à ce que le critère d'arrêt soit vérifié.

Déroulement de l'algorithme de rétro-propagation

On considère un réseau de neurones multicouches ; une couche d'entrée, une couche cachée et une couche de sortie. n_0 entrées, n_1 neurones avec une fonction d'activation non linéaire (fonction sigmoïde) dans la couche cachée et n_2 neurones linéaires dans la couche de sortie.

1. **Propagation**

   Après l'initialisation des poids, on calcule les sorties du réseau en propageant les valeurs de x_i de couche en couche, la figure 1.10 illustre cette opération.

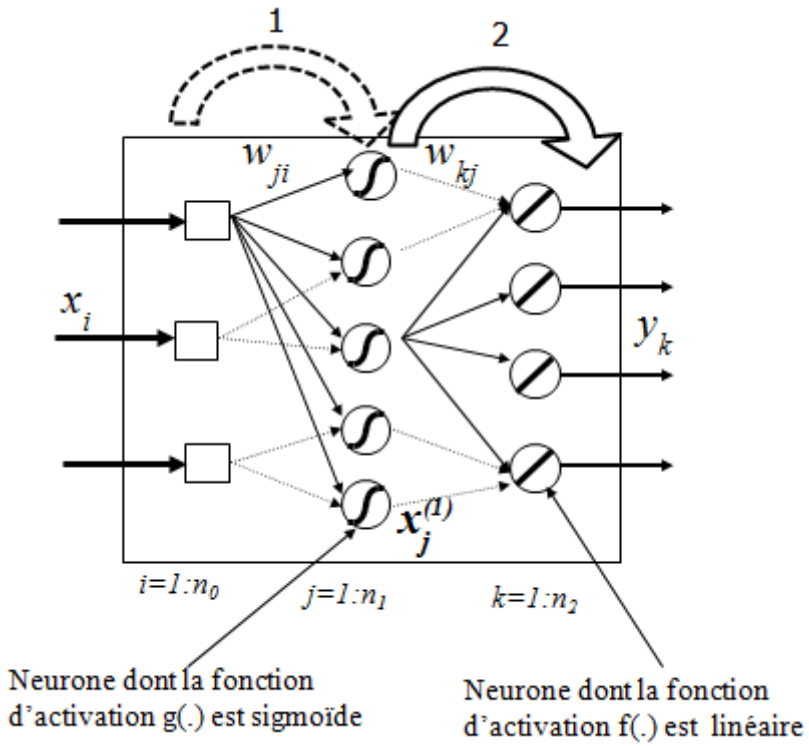


FIGURE 1.10: Propagation des entrées

– **Etape 1 :**

La valeur de la sortie de chaque neurone de la couche cachée $x_j^{(1)}$ ($j = 1...n_1$), dépend de la somme des entrées x_i ($i = 1...n_0$) pondérées par les poids entre la couche d'entrée et la couche cachée W_{ji}. La fonction d'activation non linéaire $g(.)$ détermine l'état du neurone.

$$\begin{cases} a_j^{(1)} = \sum_{i=1}^{n_0} W_{ji} x_i \\ x_j^{(1)} = g(a_j^{(1)}) \end{cases} \tag{1.9}$$

$g(.)$ étant la fonction d'activation des neurones de la couche cachée.

– **Etape 2 :**

De même on calcule la valeur de la sortie de chaque neurone de la couche de sortie $y_k(k = 1...n_2)$, Elle dépend de la somme des sorties de la couche cachée qu'on a calculée dans l'étape 1 ($x_j^{(1)}$) pondérées par les poids entre la couche cachée et la couche de sortie W_{kj}

$$\begin{cases} a_k^{(2)} = \sum_{j=1}^{n_1} W_{kj} x_j^{(1)} \\ y_k = f(a_k^{(2)}) \end{cases} \tag{1.10}$$

$f(.)$ étant la fonction d'activation des neurones de la couche de sortie.

Fonction de coût

– On présente un exemple $x = [x_1, x_2...x_{n_0}]$ et un vecteur de sortie désiré $y^{des} = [y_1^{des}, y_2^{des}...y_{n_2}^{des}]$

– On calcule le vecteur de sortie du réseau de neurones, en appliquant les étapes 1 et 2. $y = [y_1, y_2...y_{n_2}]$

– On calcule l'erreur entre la sortie réelle et la sortie désirée par

$$e_k = y_k^{des} - y_k \tag{1.11}$$

– Coût associé à l'exemple

$$J_{(exemple)} = \frac{1}{2} \sum_{k=1}^{n_2} e_k^2 \tag{1.12}$$

– Coût global pour exemples n, c'est-à-dire $x_1, x_2...x_{n_0}$ sont des vecteurs à n dimensions, il est calculé par

$$J = \sum_{l=1}^{n} J_{(exemple^{(l)})} \tag{1.13}$$

2. **Rétro-propagation**

1. Calcul du gradient

Mise à jour des poids W_{ji} et W_{kj} :

$$W_{nouvelle} = W_{ancienne} + \Delta W \tag{1.14}$$

Avec

$$\Delta W = -\lambda \frac{\partial J}{\partial W} \tag{1.15}$$

Ou $\dfrac{\partial J}{\partial W}$ est le gradient de la fonction de coût par rapport aux poids du réseau de neurones, λ est le pas du gradient de descente

2. calcul de $\frac{\partial J}{\partial W_{ji}}$ et $\frac{\partial J}{\partial W_{kj}}$

   (a) Calcul du gradient de la fonction de coût, par rapport aux poids de connexion entre la couche cachée et la couche de sortie W_{kj} représenté par la figure 1.11, $\frac{\partial J}{\partial W_{kj}}$

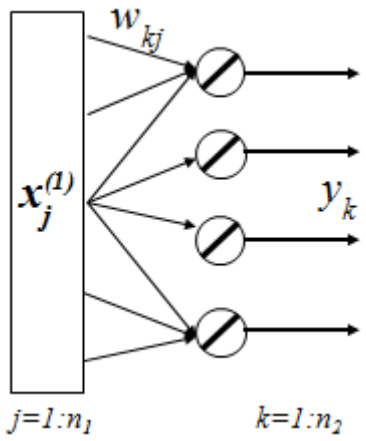


FIGURE 1.11: Rétropropagation de la deuxième couche

$$\frac{\partial J}{\partial W_{kj}} = \frac{\partial J}{\partial y_k}.\frac{\partial y_k}{\partial a_k^{(2)}}.\frac{\partial a_k^{(2)}}{\partial W_{kj}} \tag{1.16}$$

Avec :

$$\begin{cases} J = \frac{1}{2}\sum_{k=1}^{n_2}(y_k^{des} - y_k)^2 \\ y_k = f(a_k^{(2)}) \\ a_k^{(2)} = \sum_{k=1}^{n_1} W_{kj} x_j^{(1)} \end{cases} \tag{1.17}$$

Donc

$$\begin{cases} \frac{\partial J}{\partial y_k} = \sum_{k=1}^{n_2} -(y_k^{des} - y_k) \\ \frac{\partial y_k}{\partial a_k} = f'(a_k^{(2)}) \\ \frac{\partial a_k^{(2)}}{\partial W_{kj}} = x_j^{(1)} \end{cases} \tag{1.18}$$

L'équation (1.16) devient :

$$\frac{\partial J}{\partial W_{kj}} = \sum_{k=1}^{n_2} -(y_k^{des} - y_k).f'(a_k^{(2)}).x_j^{(1)} \tag{1.19}$$

(b) Calcul du gradient de la fonction de coût, par rapport aux poids de connexion entre la couche d'entrée et la couche cachée W_{ji} représenté par la figure 1.12, $\frac{\partial J}{\partial W_{ji}}$

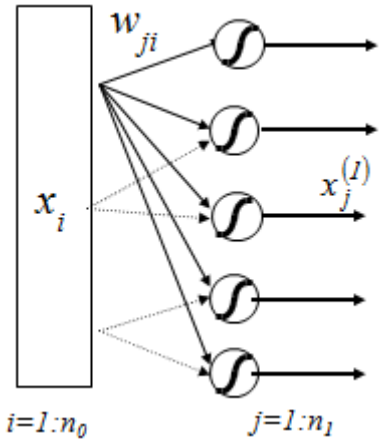


FIGURE 1.12: Rétropropagation de la première couche

$$\frac{\partial J}{\partial W_{ji}} = \frac{\partial J}{\partial x_j^{(1)}} . \frac{\partial x_j^{(1)}}{\partial a_j^{(1)}} . \frac{\partial a_j^{(1)}}{\partial W_{ji}} \tag{1.20}$$

Avec

$$\begin{cases} \frac{\partial J}{\partial x_j^{(1)}} = \sum_{k=0}^{n_2} \frac{\partial J}{\partial a_k^{(2)}} . \frac{\partial a_k^{(2)}}{\partial x_j^{(1)}} \\ x_j^{(1)} = g(a_j^{(1)}) \\ a_j^{(1)} = \sum_{i=1}^{n_0} W_{ji} x_i \end{cases} \tag{1.21}$$

donc

$$\begin{cases} \frac{\partial J}{\partial x_j^{(1)}} = \sum_{k=0}^{n_2} -(y_k^{des} - y_k) f'(a_k^{(2)}) . W_{kj} \\ \frac{\partial x_j^{(1)}}{\partial a_j^{(1)}} = g^{'}(a_j^{(1)}) \\ \frac{\partial a_j^{(1)}}{\partial W_{ji}} = x_i \end{cases} \tag{1.22}$$

L'équation (1.20)devient

$$\frac{\partial J}{\partial W_{ji}} = \left(\sum_{k=0}^{n_2} -(y_k^{des} - y_k) f'(a_k^{(2)}) . W_{kj} \right) g^{'}(a_j^{(1)}) x_i \tag{1.23}$$

Etapes d'apprentissage de l'algorithme de rétro-propagation

On peut résumer les étapes d'apprentissage de cet algorithme comme suit :

1. Initialisation de poids synaptiques d'une manière appropriée.
2. Introduction d'un exemple d'apprentissage (entrée, sortie).
3. Propager le signal d'entrée à travers le réseau, puis calculer la sortie.
4. Calculer l'erreur.
5. Application de l'algorithme de rétropropagation du gradient.
6. Répéter les étapes 2 à 5 jusqu'à la fin des exemples d'apprentissage.

– **Choix du critère à minimiser :**

Dans le cas de la rétro-propagation de l'erreur, le critère à minimiser est une erreur quadratique. L'application de l'algorithme du gradient nécessite la dérivabilité de la fonction d'activation. Le critère de minimisation d'erreur est représenté dans la formule (1.24).

$$J = \frac{1}{2}\sum_{i=1}^{n} e_i^2 \tag{1.24}$$

Méthodes d'optimisation

– **Méthode du gradient**

La méthode du gradient est un algorithme itératif qui consiste en recherche du minimum du critère à partir de ses dérivées par rapport à chacun des paramètres. La procédure est présentée par l'équation aux récurrences (1.25).

$$W^{i+1} = W^i - \lambda_i J'(W^i) \tag{1.25}$$

Avec

λ_i étant le pas du gradient de descente.

$J'(W^i)$ étant le gradient du critère par rapport aux paramètres.

– Lorsque le point initial est situé loin de l'optimum, la convergence est rapide elle devient lente en se rapprochant du minimum.
– L'utilisation de cet algorithme peut conduire à des minimums locaux.
– pour éviter cet inconvénient, on a intérêt à utiliser l'algorithme du second ordre.

– **Méthode de Gauss-Newton**

C'est une méthode itérative qui consiste en recherche du minimum du critère en utilisant la dérivée première et seconde du critère par rapport à chacun des paramètres, cette recherche est présentée par l'équation récurrente (1.26)

$$W^{i+1} = W^i - \lambda_i [J^{''}(W^i)]^{-1} J'(W^i) \tag{1.26}$$

Avec

$[J^{''}(W^i)]$ étant la matrice Hessienne (les dérivées secondes du critère par rapport aux paramètres).

Cette méthode est efficace au voisinage de l'optimum, mais se montre fragile dans sa convergence par rapport au choix des conditions initiales.

– **Méthode de Levenberg-Marquardt**

C'est une méthode itérative qui combine entre les deux méthodes précédentes dans le but de lever les inconvénients qui sont présentés par ces deux méthodes.

L'équation aux récurrences (1.27) présente la procédure itérative de l'algorithme.

$$W^{i+1} = W^i - [J^{''}(W^i) + \lambda_i I]^{-1} J'(W^i) \tag{1.27}$$

– **Méthode de régularisation bayésienne**

La régularisation bayésienne est une modification de l'algorithme d'apprentissage de Levenberg-Marquardt, afin de produire des réseaux de neurones ayant une bonne généralisation, et de réduire la difficulté de déterminer l'architecture optimale du réseau.

Dans toutes les méthodes d'optimisation, le problème consiste à trouver les valeurs des poids w minimisant la fonction de coût.

$$J = \frac{1}{2} \sum_{i=1}^{n} e_i^2 \tag{1.28}$$

Si on applique l'apprentissage sur la minimisation de perte, ça nous génère le problème de sur-apprentissage "trop de paramètres", pour cela on introduit la notion de régularisation qui nous permet de réduire la capacité donc de réduire le sur-apprentissage. On pénalise certaines valeurs de nos paramètres, ça nous permet d'exprimer certaines préférences sur des valeurs des paramètres qu'on préfère avoir comme résultat de notre apprentissage.

On ajoute la somme carrée des paramètres, ou bien le carré de la norme des paramètres.

$$J = \frac{1}{2}\sum_{i=1}^{n} e_i^2 + \frac{\lambda}{2}\|W\|^2 \tag{1.29}$$

avec $\|W\|^2 = W^T W = W_0^2 + W_1^2 + W_2^2 + ... + W_m^2$

m étant le nombres de paramètres.

Le but est de trouver les paramètres du réseau de neurones qui minimisent la fonction de coût de l'équation (1.29).

Dans ce travail, nous allons utiliser la méthode de régularisation bayésienne, avec la commande "trainbr" implémentée dans Matlab.

1.9 Conclusion

Dans le but d'utiliser le réseau de neurones comme un modèle qui représente le comportement de systèmes non linéaires, nous avons présenté une étude détaillée sur les réseaux de neurones artificiels. Nous avons présenté l'élément essentiel du réseau de neurones qui est le neurone formel. Ce dernier n'est qu'une modélisation d'un neurone biologique. Nous avons conclu aussi que la manière de la connexion entre les neurones nous permet d'obtenir plusieurs types d'architectures. Dans l'objectif d'entraîner les paramètres du réseau de neurones, nous avons présenté les algorithmes d'apprentissage. Dans notre travail, nous allons utiliser le réseau de neurones multi-couches, dont les paramètres sont entraînés par la méthode de régularisation bayésienne.

Chapitre 2

Notions sur le calcul d'ordre fractionnaire

Chapitre 2

Notions sur le calcul d'ordre fractionnaire

2.1 Introduction

Le calcul de la dérivée et de l'intégral d'ordre fractionnaire est une généralisation de la dérivée et de l'intégral d'ordre entier, il remonte à la correspondance entre Leibniz et L'Hospital en 1695. En raison de la complexité des calculs et le manque de théorèmes, il n'a pas attiré l'attention de beaucoup de chercheurs pendant une longue période. Récemment, il a été considéré comme outil précieux dans la modélisation de nombreux phénomènes dans divers domaines de l'ingénierie, de la physique et de l'économie, tels que la polarisation diélectrique, ondes électromagnétiques, les systèmes viscoélastiques[24], diffusion de chaleur[23, 18], de la biologie, de la finance...etc. Aujourd'hui, il a acquis des intérêts de chercheurs dans divers domaines de plus en plus, il est devenu l'un des sujets principaux. La dérivée d'ordre fractionnaire offre un excellent outil pour la description de la mémoire et processus dynamiques.

La notion de l'intégral d'ordre fractionnaire est la conséquence de la formule de Cauchy [17], qui permet de calculer $'k'$ fois l'intégrale de la fonction $f(t)$ sur un intervalle$[0, \infty[$.

La formule de Cauchy est définie par

$$g(t) = I^k\left[f(t)\right] = \int_0^t dt_k \int_0^{t_k} dt_{k-1} ... \int_0^{t_3} dt_2 \int_0^{t_2} f(t_1)dt_1 = \frac{1}{(k-1)!}\int_0^t (t-\tau)^{(k-1)} f(\tau)d\tau \quad (2.1)$$

k étant un nombre entier

2.2 Intégration d'ordre non entier

2.2.1 Définition de Riemann-Liouville

Dans le but de généraliser la formule de Cauchy, Riemann a remplacé la fonction factorielle par la fonction Gamma qui est la généralisation à l'ordre réel de la fonction factorielle.
Donc à l'ordre α (avec $\alpha \in \mathbb{R}^{*+}$), la formule de Cauchy devient

$$g(t) = I^{\alpha}[f(\tau)] = \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(\tau) d\tau \tag{2.2}$$

avec :

– $\Gamma(\alpha)$ est la fonction d'Euler, définie par

$$\Gamma(\alpha) = \int_0^{\infty} v^{\alpha-1} e^{-v} dv \tag{2.3}$$

La fonction d'Euler possède les propriétés suivantes :

$$\begin{aligned} &\Gamma(1) = 1 \\ &\Gamma(\alpha+1) = \alpha! \qquad (\forall \alpha \in \mathbb{N}^*) \\ &\Gamma(\alpha+1) = \alpha\Gamma(\alpha) \qquad (\forall \alpha \in \mathbb{R}^{*+}) \end{aligned} \tag{2.4}$$

On peut définir une autre fonction, en utilisant la fonction Gamma, $\Phi_\alpha(t)$ comme suit :

$$\Phi_\alpha(t) = \frac{t^{\alpha-1}}{\Gamma(\alpha)} \tag{2.5}$$

donc l'expression (2.2) devient :

$$g(t) = I^{\alpha}\left[f(\tau)\right] = \int_0^t \Phi_\alpha(t-\tau) f(\tau) d\tau \tag{2.6}$$

$\Phi_\alpha(t)$ est le facteur d'oubli [26], il permet de moduler, pour des valeurs de α non entiers, la pondération de la fonction d'une façon différente. Ainsi, les valeurs plus récentes ont plus de poids que les valeurs anciennes de $f(t)$. La fonction $\Phi_\alpha(t) = 1$ dans le cas entier ($\alpha = 1$).[17]

2.3 Dérivation d'ordre non entier

La dérivation d'ordre non entier est la généralisation de la dérivée entière classique à des ordres non entiers quelconques. Cette généralisation peut être obtenue par deux manières différentes. On obtient alors deux définitions, la définition de Riemann-Liouville et la définition de Caputo.

Une autre définition de la dérivée d'ordre non entier est obtenue à partir de la généralisation de la dérivée entière usuelle qui est la définition de Grünwald-Letnikov.

2.3.1 Définition de la dérivée d'ordre non entier de Riemann-Liouville

Le calcul de la dérivée d'ordre non entier α, compris entre $n-1$ et n ($n \in \mathbb{N}$), de la fonction $f(t)$ selon la définition de Riemann-Liouville est obtenu en deux étapes [49]

- **Première étape :** Calculer l'intégral d'ordre non entier $(n-\alpha)$ de la fonction $f(t)$.
- **Deuxième étape :** Calculer la dérivée d'ordre entier (n) du résultat de la première étape.

La formule mathématique de cette définition est

$$ {}_{a}^{RL}D_{t}^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \frac{d^n}{dt^n} \left[\int_a^t (t-\tau)^{n-\alpha-1} f(\tau) d\tau \right] \tag{2.7} $$

La notation ${}_{a}^{RL}D_{t}^{\alpha} f(t)$ signifie la dérivée d'ordre non entier α de la fonction $f(t)$ entre a et t selon la définition de Riemann-Liouville.

2.3.2 Définition de la dérivée d'ordre non entier de Caputo

le calcul de la dérivée d'ordre non entier d'une fonction $f(t)$ selon la définition de Caputo est également obtenue en deux étapes [50]

- **Première étape :** Calculer la dérivée d'ordre entier (n) de la fonction $f(t)$.
- **Deuxième étape :** Calculer l'intégral d'ordre non entier $(n-\alpha)$ du résultat de la première étape.

cette définition est représentée par la formule mathématique suivante :

$$ {}_{a}^{C}D_{t}^{\alpha} f(t) = \frac{1}{\Gamma(n-\alpha)} \int_a^t (t-\tau)^{n-\alpha-1} f^{(n)}(\tau) d\tau \tag{2.8} $$

$f^{(n)}(\tau)$ étant la dérivée d'ordre entier n de la fonction $f(\tau)$.
La notation ${}_{a}^{C}D_{t}^{\alpha} f(t)$ signifie la dérivée d'ordre non entier α de la fonction $f(t)$ entre a et t selon la définition de Caputo.

Comparaison entre les deux définitions de la dérivée d'ordre non entière selon Caputo et Rieman-Liouville

La différence entre la définition de la dérivée d'ordre non entier selon Caputo et la définition de la dérivée d'ordre non entier selon Reiman-Liouville est résumée dans trois poins essentiels.

- La dérivée d'ordre non entier de la fonction $f(t)$, selon la définition de Caputo, exige que la fonction $f(t)$ ainsi que ses n dérivées successives soient nulles pour $t \leq 0$. Par contre, la définition de Rieman-Liouville exige seulement la causalité de $f(t)$.

- Pour la résolution des équations différentielles d'ordre non entier, la solution s'exprime en fonction des valeurs initiales d'ordre non entier $(y(0), D^{\alpha}y(0)\ldots)$, alors que la solution obtenue, utilisant la définition de Caputo, s'exprime en fonction des valeurs initiales d'ordre entier, qui sont plus perceptibles dans le domaine de la science physique par rapport à leurs dérivées d'ordre non entier.
- La dérivée d'ordre non entier de la constante est une fonction non nulle qui dépend de la variable t selon la définition de Reiman-Liouville , alors que sa dérivée non entier est nulle selon la définition du Caputo.

$$ {}_{a}^{RL}D_{t}^{\alpha}C = \frac{C\,(t-a)^{-\alpha}}{\Gamma\,(1-\alpha)} \qquad\qquad {}_{a}^{C}D_{t}^{\alpha}C = 0 \tag{2.9}$$

2.3.3 Définition de la dérivée d'ordre non entier de Grünwald-Letnikov

La dérivée généralisée d'une fonction $f(t)$ peut également être obtenue de façon plus naturelle en utilisant la définition entière usuelle. C'est la définition proposée par Grünwald [20]. Elle est plus adéquate au calcul numérique de la dérivation non entière. En effet, partant de la dérivée première :

$$D^{1}\left[f(t)\right] = \lim_{h\to 0}\frac{f(t)-f(t-h)}{h} = \lim_{h\to 0}\frac{\Delta_{h}^{1}\left[f(t)\right]}{h} \tag{2.10}$$

h étant la période d'échantillonnage.

La dérivée seconde donne :

$$D^{2}\left[f(t)\right] = \lim_{h\to 0}\frac{f(t)-2f(t-h)+f(t-2h)}{h^{2}} = \lim_{h\to 0}\frac{\Delta_{h}^{2}\left[f(t)\right]}{h^{2}} \tag{2.11}$$

La dérivée d'ordre n ($n \in \mathbb{N}$) de la fonction $f(t)$ est défini par la série :

$$D^{n}\left[f(t)\right] = \lim_{h\to 0}\frac{1}{h^{n}}\sum_{j=0}^{n}\left((-1)^{j}\begin{pmatrix} n \\ j \end{pmatrix} f(t-jh)\right) = \lim_{h\to 0}\frac{\Delta_{h}^{n}\left[f(t)\right]}{h^{n}} \tag{2.12}$$

où la notation $\begin{pmatrix} n \\ j \end{pmatrix}$ représente le binôme de Newton dont l'expression est donnée par

$$\begin{pmatrix} n \\ j \end{pmatrix} = \frac{n!}{j!(n-j)!} \tag{2.13}$$

$\Delta_{h}^{n}\left[f(t)\right]$ étant l'opérateur de la différence d'ordre $'n'$ de la fonction $f\,(t)$ avec un pas d'échantillonnage $h \in \mathbb{R}^{*+}$.[21, 22]

Avec

$$\Delta_h^n [f(t)] = \sum_{j=0}^{n} \left((-1)^j \begin{pmatrix} n \\ j \end{pmatrix} f(t - jh) \right) \tag{2.14}$$

posant $t = Kh$

$$\Delta_h^n [f(Kh)] = \sum_{j=0}^{n} \left((-1)^j \begin{pmatrix} n \\ j \end{pmatrix} f((K - j)h) \right) \tag{2.15}$$

La dérivée généralisée d'ordre α est donnée comme suit :

$$D^\alpha [f(t)] = \lim_{h \to 0} \frac{1}{h^\alpha} \sum_{j=0}^{\infty} \left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} f(t - jh) \right) = \lim_{h \to 0} \frac{\Delta_h^\alpha [f(t)]}{h^\alpha} \tag{2.16}$$

Dans le cas où la fonction $f(t)$ est causale, en posant t = Kh, cette condition se traduit par $f(Kh - jh) = 0$ pour $Kh - jh < 0$ c'est à dire pour $j > K$, donc l'intervalle de la somme de l'équation (2.16) sera réduit de $[0, \infty[$ à $[0, K]$, on obtient la dérivée d'ordre non entier de la fonction $f(t)$ (avec $t = Kh$) à l'ordre α selon la définition Grünwald-Letnikov [25]

$$D^\alpha [f(Kh)] = \frac{1}{h^\alpha} \sum_{j=0}^{K} \left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} f((K - j)h) \right) \tag{2.17}$$

Donc à l'ordre α et pour $f(t)$ causale, l'équation (2.15) devient

$$\Delta_h^\alpha [f(Kh)] = \sum_{j=0}^{K} \left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} f((K - j)h) \right) \tag{2.18}$$

Avec

- $\Delta_h^\alpha [f(Kh)]$ est l'opérateur de la différence d'ordre non entier (α) de la fonction $f(Kh)$
- $\alpha \in \mathbb{R}^{*+}$ (l'ensemble des nombres réels strictement positives) est l'ordre non entier.
- $h \in \mathbb{R}^{*+}$est le pas d'échantillonnage, dans tout ce qui suit on prend $h = 1$.
- $K \in \mathbb{N}$

Avec $\begin{pmatrix} \alpha \\ j \end{pmatrix}$ est le binôme de Newton généralisé à des ordres non entiers :

$$\begin{pmatrix} \alpha \\ j \end{pmatrix} = \begin{cases} 1 & pour \quad j = 0 \\ \dfrac{(\alpha)(\alpha - 1)...(\alpha - j + 1)}{j!} & pour \quad j > 0 \end{cases} \tag{2.19}$$

Pour $h = 1$, l'équation (2.18) devient

$$\Delta^\alpha [f(K)] = \sum_{j=0}^{K} \left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} f(K - j) \right) \tag{2.20}$$

L'équation (2.20) est la différenciation d'ordre non entier de la fonction $f(K)$.

2.3.4 Quelques propriétés de la dérivation d'ordre non entier

Les principales propriétés des dérivées et intégrales d'ordre fractionnaire sont données par[14]

Linéarité

La dérivation non entière est un opérateur linéaire. Ainsi, si $f(t)$ et $g(t)$ sont deux fonctions continues et λ et μ sont des nombres réels, on a :

$$D^{\alpha}\left[\lambda.f(t)+\mu.g(t)\right]=\lambda.D^{\alpha}\left[f(t)\right]+\mu.D^{\alpha}\left[g(t)\right] \tag{2.21}$$

Opérateur identité

Pour $\alpha=0$ l'opération $D^{\alpha}\left[f(t)\right]$ est l'opérateur identité :

$$D^{0}\left[f(t)\right]=f(t) \tag{2.22}$$

Loi additive d'indexe

$$D^{\alpha}\left[D^{\beta}f(t)\right]=D^{\beta}\left[D^{\alpha}f(t)\right]=D^{\alpha+\beta}\left[f(t)\right] \tag{2.23}$$

où α et $\beta\in\mathbb{R}^{*+}$

Généralisation

La dérivée d'ordre non entier est une généralisation de de la dérivée entière, lorsque $\alpha=n$ ($n\in\mathbb{N}$), l'opération $D^{\alpha}f(t)$ donne le même résultat que la dérivée classique d'ordre entier.

2.4 Modélisation des systèmes d'ordre fractionnaire

Comme dans le cas entier, il existe différents modèles qui représentent les systèmes d'ordre fractionnaire on cite principalement :

2.4.1 Équation différentielle d'ordre fractionnaire

Comme dans le cas entier, un système d'ordre non entier, peut être représenté par une équation différentielle généralisée suivante :

$$\sum_{i=0}^{n}a_{i}D^{\alpha_{i}}y(t)=\sum_{j=0}^{m}b_{j}D^{\beta_{j}}u(t) \tag{2.24}$$

$u(t)$, $y(t)$ désignent respectivement l'entrée et la sortie du système à l'instant t, D^{α} est l'opérateur de la dérivée d'ordre α. a_i, b_j sont les coefficients de l'équation différentielle et $\alpha_i, \beta_j \in \mathbb{R}^+$

Définition

Un système non entier est dit d'ordre commensurable, lorsque tous les ordres de dérivation α_i et β_j de son équation différentielle sont multiples du même nombre non entier α. Dans ce cas, l'équation différentielle généralisée (2.24) s'écrit sous forme :

$$\sum_{i=0}^{n} a_i D^{i.\alpha} y(t) = \sum_{j=0}^{m} b_j D^{j.\alpha} u(t) \tag{2.25}$$

2.4.2 Fonction de transfert d'ordre fractionnaire

L'utilisation de la transformée de Laplace de l'équation (2.24), en considérant les conditions initiales nulles, permet de déduire la fonction de transfert d'ordre non entier suivante :

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^{\beta_m} + b_{m-1} s^{\beta_{m-1}} + \ldots + b_0 s^{\beta_0}}{a_n s^{\alpha_n} + a_{n-1} s^{\alpha_{n-1}} + \ldots + a_0 s^{\alpha_0}} \tag{2.26}$$

Dans le cas d'un système commensurable d'ordre α, la fonction de transfert généralisée de l'équation (2.26) devient :

$$G(s) = \frac{b_m s^{m.\alpha} + b_{m-1} s^{(m-1)\alpha} + \ldots + b_0}{a_n s^{n.\alpha} + a_{n-1} s^{(n-1)\alpha} + \ldots + a_0} = \frac{b_m (s^m)^{\alpha} + b_{m-1} (s^{(m-1)})^{\alpha} + \ldots + b_0}{a_n (s^n)^{\alpha} + a_{n-1} (s^{(n-1)})^{\alpha} + \ldots + a_0} \tag{2.27}$$

2.4.3 Représentation d'état d'ordre fractionnaire

La représentation d'état d'ordre fractionnaire est définie comme dans le cas entier, on remplace la dérivée entière d'ordre 1 par la dérivée fractionnaire d'ordre α.

Systèmes continus

Les systèmes continus d'ordre fractionnaire sont donnés par des représentations d'état d'ordre fractionnaire de différentes manières selon le cas linéaire ou non linéaire.

– Dans le cas linéaire, la représentation d'état est donnée par

$$\begin{cases} D^{\alpha} x(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{2.28}$$

$$D^{\alpha} x(t) = \begin{bmatrix} D^{\alpha} x_1(t) & D^{\alpha} x_2(t) & \ldots & D^{\alpha} x_n(t) \end{bmatrix}^T \tag{2.29}$$

$x(t)$: vecteur d'état de dimension $n \times 1$.

$D^{\alpha}x(t)$: vecteur de la dérivée d'ordre α (avec $\alpha \in \mathbb{R}$).

A partir de la représentation d'état (2.28), on peut obtenir la fonction de transfert correspondante, en appliquant la transformé de Laplace et en considérant les conditions initiales nulles. On obtient :

$$G(s) = C\left[(s^{\alpha}I_n - A)^{-1}\right]B \tag{2.30}$$

– Dans le cas non linéaire, la représentation d'état est donnée par

$$\begin{cases} D^{\alpha}x(t) = f(x(t), u(t)) \\ y(t) = \Psi(x(t)) \end{cases} \tag{2.31}$$

$f(.)$ et $\Psi(.)$ étant les fonctions non linéaires.

D^{α} étant l'opérateur de la dérivée d'ordre α.

Systèmes discrets

De même que pour le cas continu, les systèmes discrets sont représentés dans le cas linéaire ou non linéaire.

– Le système linéaire d'ordre fractionnaire est représenté par le modèle d'état linéaire d'ordre fractionnaire à temps discret comme suit : [27]

$$\begin{cases} \Delta^{\alpha}x\left(k+1\right) = Ax\left(k\right) + Bu\left(k\right) \\ y(k) = Cx(k) + Du(k) \end{cases} \tag{2.32}$$

Avec

$$\Delta^{\alpha}x(k+1) = \left[\begin{array}{cccc} \Delta^{\alpha}x_1(k+1) & \Delta^{\alpha}x_2(k+1) & ... & \Delta^{\alpha}x_n(k+1) \end{array}\right]^T \tag{2.33}$$

Δ^{α} étant l'opérateur de la différence d'ordre α, $x(k) \in \mathbb{R}^n$, $u(k) \in \mathbb{R}^m$, $y(k) \in \mathbb{R}^p$ sont respectivement les vecteurs d'état, d'entrée et de sortie et $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times m}$

– Le système non linéaire d'ordre fractionnaire est représenté par le modèle d'état non linéaire d'ordre fractionnaire à temps discret comme suit :

$$\begin{cases} \Delta^{\alpha}x(k+1) = f(x(k), u(k)) \\ y(k) = \Psi(x(k)) \end{cases} \tag{2.34}$$

$f(.)$ et $\Psi(.)$ étant les fonctions non linéaires.

2.5 Simulation des systèmes d'ordre fractionnaire

La simulation des systèmes d'ordre fractionnaire consiste à simuler l'élément de base qui est l'opérateur d'intégration ou de dérivation d'ordre fractionnaire. Actuellement, le principe consiste souvent à approximer l'opérateur de dérivation par un transfert entier. A cet effet, plusieurs méthodes d'approximations sont introduites pour la modélisation de cet opérateur. On distingue essentiellement deux classes selon que la transmittance obtenue est continue ou discrète.

2.5.1 Approche continue

Dans cet approche, il existe plusieurs techniques d'approximation qui sont l'approximation d'Oustaloup [33, 32, 31] et l'approximation de Charef [30]. D'autres méthodes d'approximation utilisent des techniques d'interpolation, on peut mentionner la méthode de Carlson [28] qui se base sur un processus itératif de Newton, et la méthode de Matsuda [29] qui utilise le principe du développement en fractions continues (CFE : Continued Fraction Expansions), permet d'approximer la réponse du dérivateur généralisé sur un intervalle de fréquences espacées de façon logarithmique. Il existe aussi des techniques d'identification fréquentielles. La démarche consiste donc à identifier la transmittance d'ordre entier à partir de la réponse fréquentielle "idéale" du dérivateur généralisé. On peut citer l'algorithme de Lévy [34] qui a été utilisé par Duarté dans [35], l'algorithme "Vector Fitting" de Gustavsen [36] mentionné dans [65], ainsi que l'approche proposée par Xue et Chen [38] et dont le principe consiste à minimiser la norme de l'erreur d'approximation.

2.5.2 Approche discrète

Dans le cas discret, l'approximation s'effectue selon deux approches distinctes, les méthodes de discrétisation directe et les méthodes de discrétisation indirecte.

– **Méthodes de discrétisation directe :** Parmi ces méthodes, nous pouvons citer la méthode de Grünwald-Letnikov qui est une définition de la dérivée usuelle d'une fonction. Cette méthode permet de faire l'approximation des équations différentielles généralisées par des équations aux récurrences. Ainsi il existe d'autres méthodes d'approximation qui sont basées sur la discrétisation classique dans le domaine fréquentiel Euler, Simpson, El-Alaoui [39, 40, 41]

– **Méthodes de discrétisation indirecte :** La discrétisation indirecte s'effectue en deux étapes

1. L'approximation continue.
2. La discrétisation de cette dernière au moyen des méthodes usuelles [39]. Cette approche est basée sur des méthodes numériques.

2.5.3 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 (afin d'assurer un comportement à phase minimale). 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 transmittance d'ordre entier.

La fonction de transfert du dérivateur généralisé est donnée par

$$D(s) = \left(\frac{s}{w_c}\right)^{\alpha} \tag{2.35}$$

Avec

w_c étant la pulsation de coupure, $\alpha \in \mathbb{R}$ étant l'ordre de dérivateur généralisé.

Une troncature à la fois du coté des basses et des hautes fréquences consiste à limiter sur l'intervalle fréquentiel $\left[\begin{array}{cc} w_A & w_B \end{array}\right]$, centré géométriquement sur w_c , le comportement différentiel du transfert $\dfrac{s}{w_c}$. En fait, la troncature sera réellement effectuée, pour plus de précision, sur un intervalle de fréquence plus large $\left[\begin{array}{cc} w_b & w_h \end{array}\right]$, tel que :

$$w_b \ll w_A \quad et \quad w_h \gg w_B \tag{2.36}$$

La fonction de transfert fractionnaire bornée en fréquence est donnée par

$$D_b = \left(C_0 \frac{1 + \dfrac{s}{w_b}}{1 + \dfrac{s}{w_h}}\right)^{\alpha} \tag{2.37}$$

pour assurer un gain unitaire à la fréquence on pose

$$C_0 = \frac{w_b}{w_c} = \frac{w_c}{w_h} \tag{2.38}$$

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

$$D_b(s) = \lim_{N\to\infty} D_N(s) \tag{2.39}$$

Avec

$$D_N(s) = \left(\frac{w_c}{w_h}\right)^{\alpha} \Pi_{i=-N}^{N} \frac{\left(1+\frac{s}{z_i}\right)}{\left(1+\frac{s}{p_i}\right)} \tag{2.40}$$

p_i et z_i étant le pôle et le zéros de rang i, et le nombre total de paires (pôle,zéros) étant $2N+1$ sont distribuées récursivement comme suit :

$$\begin{cases} \dfrac{p_i}{z_i} = \alpha > 0, \dfrac{z_{i+1}}{p_i} = \eta > 0 \\ \dfrac{z_{i+1}}{z_i} = \dfrac{p_{i+1}}{p_i} = \alpha\eta > 1 \end{cases} \tag{2.41}$$

La figure(2.1) montre l'approximation qui a été réalisée par le lissage du diagramme asymptotique de Bode.

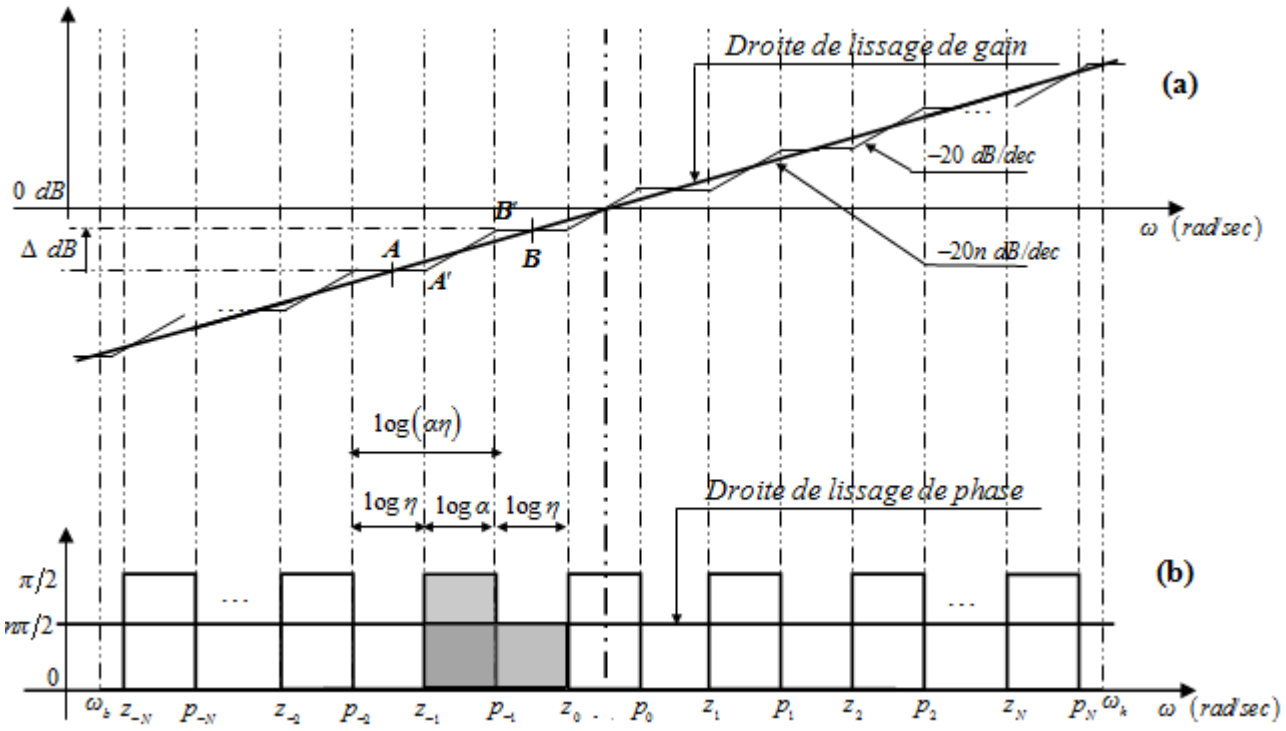


FIGURE 2.1: Diagrammes asymptotiques de Bode de $D_b(s)$ et de $D_N(s)$ pour $\alpha \in]0,1[$

2.6 Système à modéliser

Le but de ce travail est la modélisation de systèmes non linéaires d'ordre fractionnaire représentés par une équation aux différences non linéaire d'ordre fractionnaire de la forme

$$\Delta^{\alpha_n} y(k+n) = g(\Delta^{\alpha_{(n-1)}} y(k+n-1), ..., \Delta^{\alpha_1} y(k+1), y(k), u(k)) \tag{2.42}$$

Avec

$\alpha_n > \alpha_{n-1} > ... > \alpha_1$ (α_1, α_2,$\cdots$, $\alpha_n \in \mathbb{R}^{*+}$)

On prend en considération le système qui est décrit par l'équation aux différences d'ordre commensurable α de la forme :

$$\Delta^{n\alpha} y(k+n) = g(\Delta^{(n-1)\alpha} y(k+n-1)..., \Delta^{\alpha} y(k+1), y(k), u(k)) \tag{2.43}$$

Avec

- n : est un entier positif ($n \in \mathbb{N}$).
- $\alpha \in \mathbb{R}^{*+}$
- $\Delta^{n\alpha} y(k+n)$: est la différence d'ordre $n \times \alpha$ de la réponse du système à l'instant $(k+n)$
- y(i) : est la valeur de la sortie à l'instant « i »
- u(j) : est la valeur de l'entrée à l'instant « j »
- g(.) : est la fonction non linéaire du système.

L'expression (2.43) peut être réécrite sous forme d'une représentation d'état non linéaire d'ordre fractionnaire en effectuant les changements de variables suivants :

$$\begin{cases} x_1(k) = y(k) \\ x_2(k) = \Delta^{\alpha} x_1(k+1) = \Delta^{\alpha} y(k+1) \\ x_3(k) = \Delta^{\alpha} x_2(k+1) = \Delta^{\alpha}(\Delta^{\alpha} x_1(k+2)) = \Delta^{2\alpha} y(k+2) \\ x_4(k) = \Delta^{\alpha} x_3(k+1) = \Delta^{\alpha}(\Delta^{\alpha} x_2(k+2)) = \Delta^{3\alpha} y(k+3) \\ \vdots \\ x_n(k) = \Delta^{\alpha} x_{n-1}(k+1) = \Delta^{\alpha}(\Delta^{\alpha} x_{n-2}(k+2)) = \Delta^{(n-1)\alpha} y(k+n-1) \end{cases} \tag{2.44}$$

$$\Delta^{\alpha} x_n(k+1) = \Delta^{n\alpha} y(k+n) = g(x_1(k), x_2(k), \ldots, x_n(k), u(k)) \tag{2.45}$$

On obtient la représentation d'état non linéaire d'ordre fractionnaire :

$$\begin{cases} \Delta^{\alpha} x_1(k+1) = x_2(k) \\ \Delta^{\alpha} x_2(k+1) = x_3(k) \\ \Delta^{\alpha} x_3(k+1) = x_4(k) \\ \vdots \\ \Delta^{\alpha} x_{n-1}(k+1) = x_n(k) \\ \Delta^{\alpha} x_n\left(k+1\right) = g(x_1\left(k\right), x_2\left(k\right), \ldots, x_n\left(k\right), u\left(k\right)) \\ \\ y(k) = x_1(k) \end{cases} \tag{2.46}$$

2.7 Exemples d'application

Deux exemples d'application de systèmes non linéaires d'ordre fractionnaire sont représentés par les modèles d'état non linéaires d'ordre fractionnaire, les résultats de simulation seront utilisés comme base de données pour la conception des modèles RdNs-IsFs.

2.7.1 Exemple 1

Dans cet exemple, on considère le système non linéaire d'ordre fractionnaire. Pour la simulation du système, on utilise la méthode de Grünwald-Letnikov. Le système ne possède qu'une seule variable d'état avec une non linéarité cubique :

$$\begin{cases} \Delta^{\alpha} x(k+1) = -0.1x^3(k) + u(k) \\ y(k) = x(k) \end{cases} \tag{2.47}$$

L'entrée utilisée est un créneau de différentes amplitudes représenté par la figure 2.2.

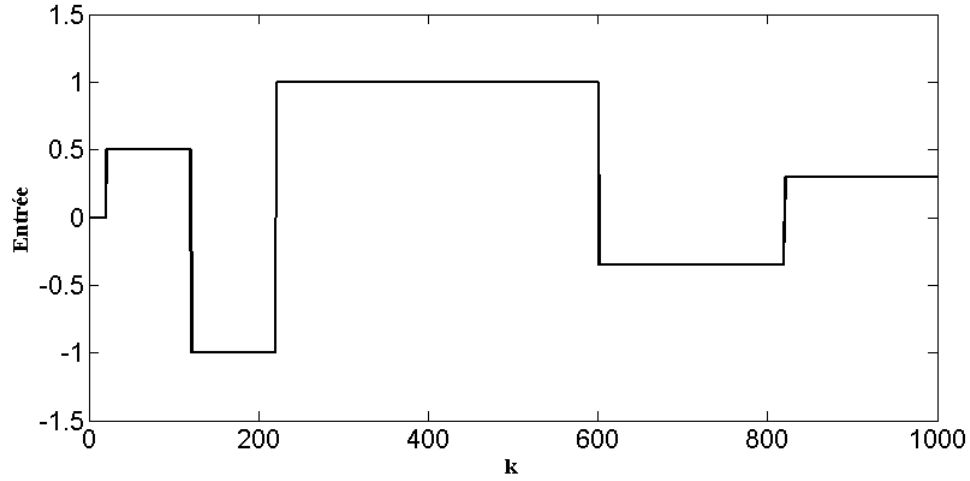


FIGURE 2.2: Signal d'entrée

Lorsqu'on utilise la définition de la dérivée d'ordre fractionnaire de Grünwald-Letnikov, pour la simulation du système, l'équation (2.47) devient :

$$\begin{cases} \Delta^{\alpha}x(k+1) = -0.1x^3(k) + u(k) \\ x(k+1) = \Delta^{\alpha}x(k+1) - \sum_{j=1}^{k+1}\left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} x(k-j+1)\right) \\ y(k) = x(k) \end{cases} \tag{2.48}$$

Pour différentes valeurs de α ($\alpha = 0.1$, 0.3, 0.5 et 0.8.), et pour la condition initiale nulle, les résultats de simulation du système (2.48) sont montrés par la figure 2.3.

Les résultats de simulation de la figure 2.3 montrent qu'à chaque fois qu'on change la valeur de α, la dynamique du système change ; pour $\alpha = 0.8$ le système est plus rapide que pour $\alpha = 0.1$.

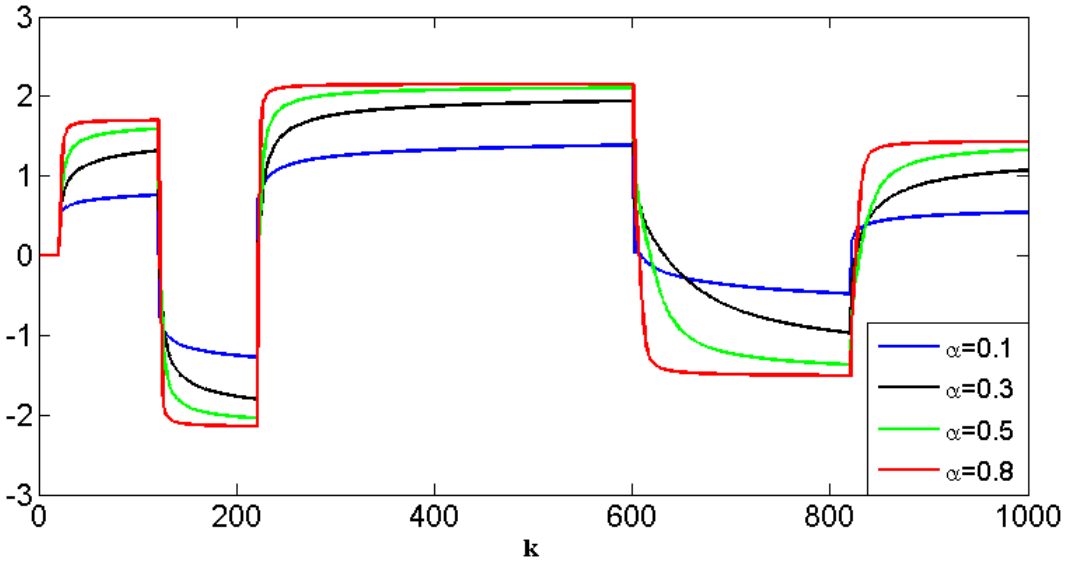


FIGURE 2.3: Réponse du système (2.47) à l'entrée U pour différentes valeurs de α.

2.7.2 Exemple 2

Soit le système non linéaire d'ordre fractionnaire, représenté par l'équation aux récurrences non linéaire d'ordre $2.\alpha$ ($\alpha \in \mathbb{R}$) avec une non linéarité cubique suivante

$$\begin{cases} \Delta^{2\alpha} x_1(k+2) = -0.1 x_2^3(k) + u(k) \\ \text{Avec} \\ x_2(k) = \Delta^{2\alpha} y(k+2) \end{cases} \tag{2.49}$$

Le système (2.49) peut être réécrit sous forme d'une représentation d'état discrète à deux variables d'état avec une non linéarité cubique. Le système est soumis à une entrée u(k) qui est un créneau de différentes amplitudes :

$$\begin{cases} \Delta^{\alpha} x_1(k+1) = x_2(k) \\ \Delta^{\alpha} x_2(k+1) = -0.1 x_2^3(k) + u(k) \\ y(k) = x_1(k) \end{cases} \tag{2.50}$$

Pour la simulation du système on utilise une nouvelle fois la définition de Grünwald-Letnikov.

On prend les conditions initiales nulles ($x_1(0) = 0$ et $x_2(0) = 0$).

$$\begin{cases} \Delta^{\alpha} x_1(k+1) = x_2(k) \\ x_1(k+1) = \Delta^{\alpha} x_1(k+1) - \sum_{j=1}^{k+1} \left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} x_1(k-j+1) \right) \\ \Delta^{\alpha} x_2(k+1) = -0.1 x_2^3(k) + u(k) \\ x_2(k+1) = \Delta^{\alpha} x_2(k+1) - \sum_{j=1}^{k+1} \left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} x_2(k-j+1) \right) \\ y(k) = x_1(k) \end{cases} \tag{2.51}$$

Pour une même entrée que l'exemple précédent, la réponse du système est montrée par la figure2.4

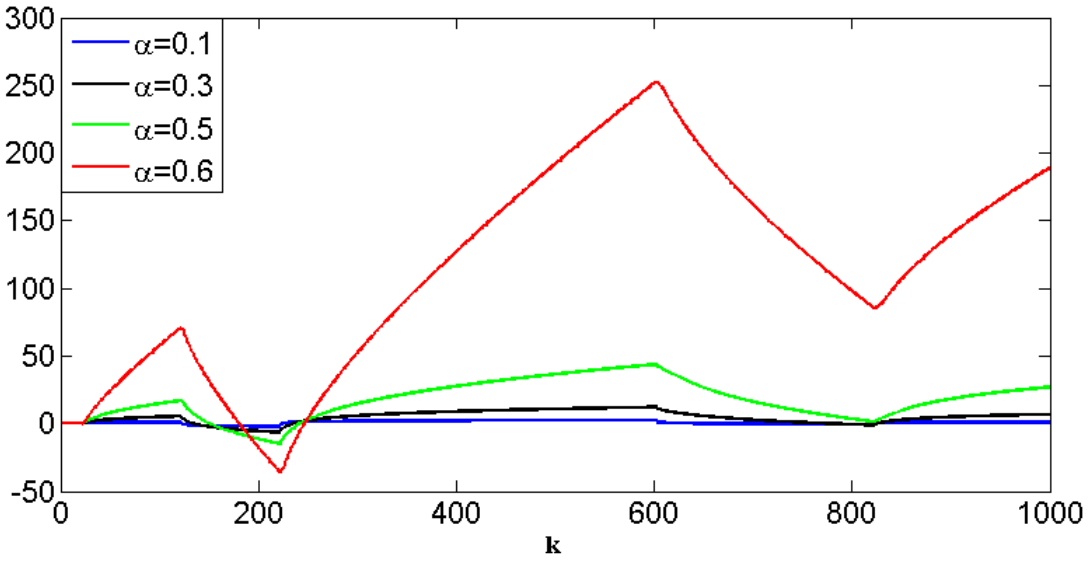


FIGURE 2.4: Réponse du système (2.49) à l'entrée U pour différentes valeurs de α

Pour valider les résultats de simulation des deux exemples précédents obtenus en utilisant la définition de Grünwald-Letnikov, l'approximation d'Oustaloup est effectuée pour des modèles continus. On doit donc écrire les modèles continus correspondant aux modèles discrets (3.39) et (2.50)

Nous présentons dans ce qui suit, les résultats de simulation pour l'exemple 1. Le modèle ici est donné par

$$\begin{cases} D^{\alpha}x(t) = -0.1x^{3}(t) + u(t) \\ y(t) = x(t) \end{cases} \tag{2.52}$$

Pour simuler ce système on a besoin d'un dérivateur généralisé d'ordre 0.5 sur l'intervalle de fréquences $\left[\begin{array}{cc} 10^{-4} & 10^{4} \end{array}\right]$

On prend le nombre de cellules égale à 20. La figure 2.5 représente le schéma de simulation du système (2.52).

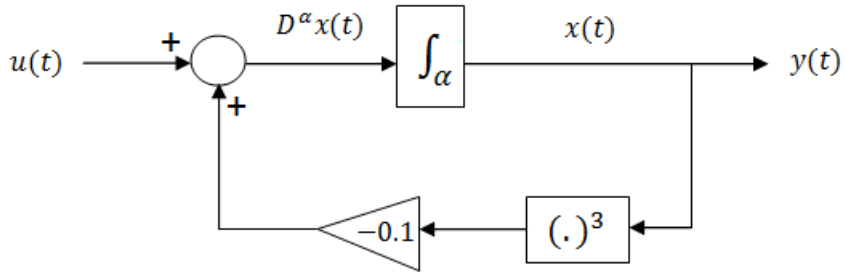


FIGURE 2.5: Schéma de simulation du premier système en utilisant l'approximation d'Oustaloup

La figure 2.6 montre les résultats de simulation obtenus en utilisant les deux méthodes,la méthode d'approximation d'Oustaloup et la méthode de Grünwald-Letnikov.

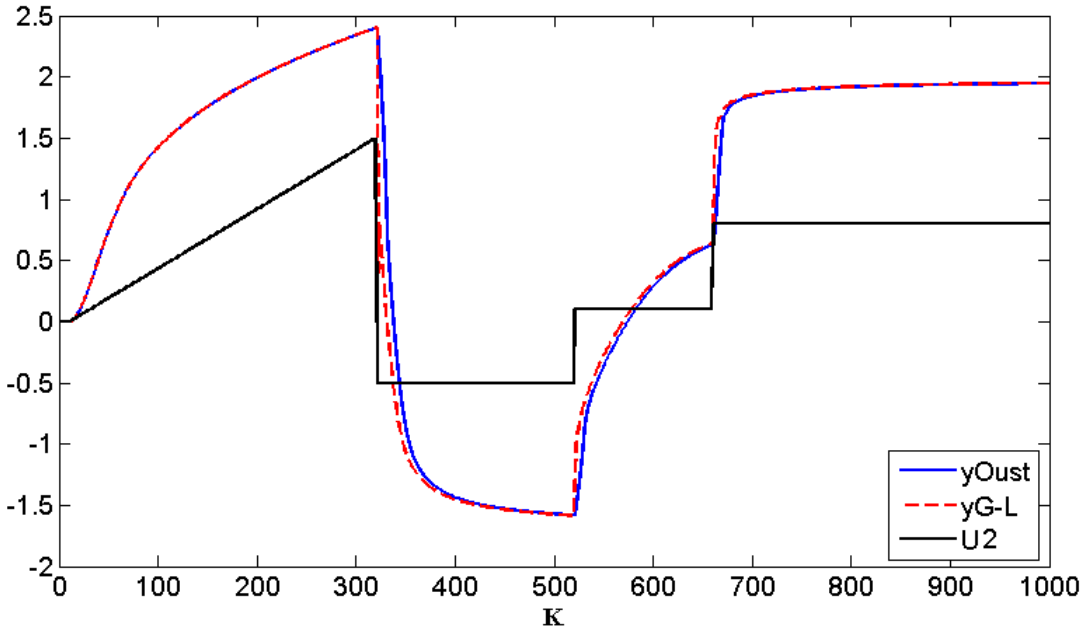


FIGURE 2.6: Réponse du système 1 à l'entrée U2 pour $\alpha = 0.5$ en utilisant l'approximation d'Oustaloup et celle de Grünwald-Letnikov

La réponse du système lorsqu'on a utilisé la méthode d'approximation de Grünwald-Letnikov est presque confondue avec celle lorsqu'on a utilisé la méthode d'approximation d'oustaloup. La présence de l'erreur entre les deux réponses vient des approximations qui diffèrent d'une méthode à l'autre.

2.8 Conclusion

Dans ce chapitre nous avons présenté les généralités sur le calcul non entier. Les différentes définitions de calcul de la dérivation et de l'intégration d'ordre fractionnaire ainsi que les définitions de Reimann-Liouville, Caputo, Grünwald-Letnikov ont été données. Nous avons aussi développé la modélisation des systèmes avec des modèles d'ordre fractionnaire. Nous avons également présenté deux exemples de simulation pour les systèmes non linéaires d'ordre fractionnaire. Pour la simulation des systèmes, nous avons utilisé la définition de Grünwald-Letnikov et pour la validation, nous avons utilisé la méthode d'Oustaloup. Ces deux exemples seront utilisés pour valider la représentation des systèmes par des réseaux de neurones que nous développons dans ce travail.

Chapitre 3

Modélisation des systèmes Non Linéaires Fractionnaires par des Réseaux de Neurones

Chapitre 3

Modélisation des systèmes Non Linéaires Fractionnaires par des Réseaux de Neurones

3.1 Introduction

Les réseaux de neurones ont des propriétés intéressantes pour modéliser la dynamique des systèmes non linéaires. Indépendamment de l'ordre du système, plusieurs travaux de recherche ont proposé des réseaux de neurones d'ordre entier pour modéliser des systèmes non linéaires entiers [42, 43] et des systèmes fractionnaires [12].

Lorsqu'on utilise un réseau de neurones standard (classique), pour modéliser un système non linéaire fractionnaire, la structure du réseau de neurones est compliquée et la précision peut être insuffisante. Deux solutions peuvent alors être envisagées pour résoudre le problème. La première consiste de concevoir de nouvelles architectures de réseau de neurones qui tient compte l'aspect fractionnaire. Cela peut être obtenu en introduisant l'aspect fractionnaire dans les fonctions d'activation par exemple ; c'est un domaine de recherche qui reste ouvert et où quelques résultats ont été proposés [13]. La deuxième solution que nous utilisons dans notre travail consiste en une combinaison d'un réseau de neurone artificiel standard, permettant de reconstituer l'aspect non linéaire du système et d'un bloc constitué d'un ensemble d'intégrateurs d'ordre fractionnaire, permettant de reconstituer l'aspect fractionnaire du système [1].

3.2 Réseaux de neurones associés à un bloc d'ordre fractionnaire

La figure 3.1, montre la structure générale de la combinaison d'un bloc de réseau de neurones artificiel entier (R.N.A) et d'un bloc de représentation d'état d'ordre fractionnaire (REDLF). Ce dernier a une structure particulière, de sorte qu'il se comporte comme un bloc des intégrateurs d'ordre fractionnaire. Pour faire l'approximation d'intégrateurs d'ordre fractionnaire, on utilise la définition de Grünwald-Letnikov. Le système non linéaire d'ordre fractionnaire à modéliser est de la forme

$$\begin{cases} \Delta^{\alpha} x(k+1) = f(x(k), u(k)) \\ x(k+1) = \Delta^{\alpha} x(k+1) - \sum_{j=1}^{k+1} \begin{pmatrix} \alpha \\ j \end{pmatrix} x(k+1-j) \\ y(k) = \Psi(x(k)) \end{cases} \tag{3.1}$$

$f(.)$ et $\Psi(.)$ étant les fonctions non linéaires.

Le système non linéaire d'ordre fractionnaire est représenté par une représentation d'état non linéaire d'ordre fractionnaire. Les bases de données utilisées pour la conception des réseaux de neurones sont les résultats de simulations obtenus par la méthode de Grünwald-Letnikov.

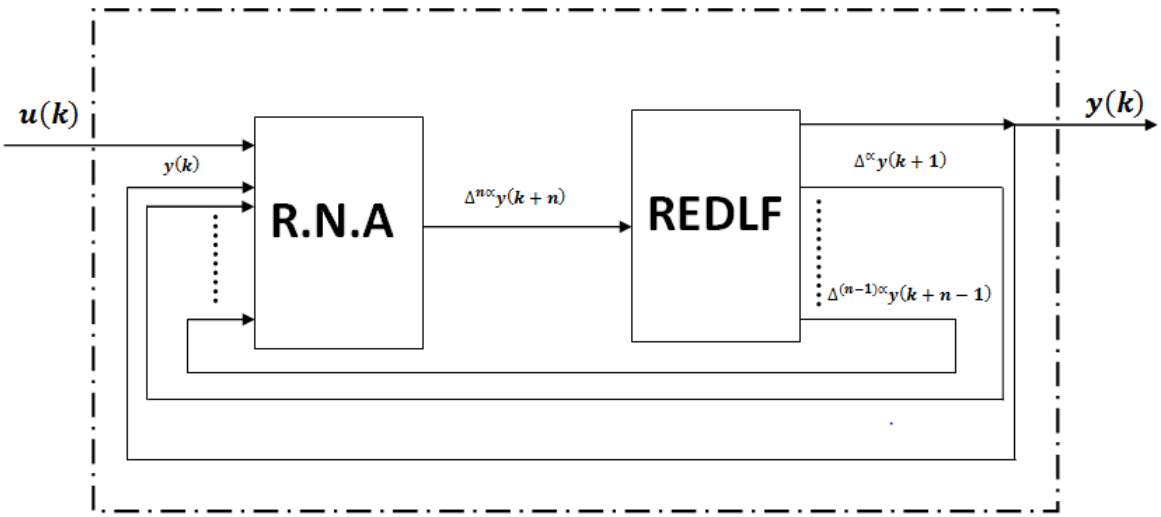


FIGURE 3.1: Structure générale du réseau de neurones associé à un bloc d'intégrateur fractionnaire à temps discret

3.2.1 Bloc Réseau de Neurones Artificiel (R.N.A)

Le réseau de neurones utilisé est un réseau de type feed-forward dont la structure comporte

- **Couche d'entrée :** Les signaux d'entrées sont :
  - signal d'entrée du système $u(k)$
  - signal de sortie $y(k)$
  - les signaux des dérivées précédentes de $\Delta^{(n-1)\alpha}y(k+n-1)$ au $\Delta^{\alpha}y(k+1)$
- **Couches cachées :** sont composées d'une ou plusieurs couches de neurones dont leurs fonctions d'activations sont 'tansig'.
- **Couche de sortie :** est composée de neurones dont les fonctions d'activation sont des fonctions linéaires. Le nombre de neurones de la couche de sortie dépend du nombre de sorties désirées, dans notre cas, nous avons besoin d'une seule sortie qui est la différence d'ordre $n\alpha$ du vecteur de sortie $(\Delta^{n\alpha}y(k+n))$ (avec n est le nombre de variables d'état, α est l'ordre de la différence d'ordre fractionnaire) de la sortie.

3.2.2 Bloc Représentation d'Etat Discrète Linéaire d'ordre Fractionnaire (REDLF)

C'est un ensemble d'intégrateurs d'ordre fractionnaire α, donné par une représentation d'état linéaire d'ordre fractionnaire à temps discret

$$\begin{cases} \Delta^{\alpha}x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Du(k) \end{cases} \tag{3.2}$$

les matrices de la représentation d'état de l'équation (3.2) sont

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.3}$$

Avec : $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times 1}$, $C \in \mathbb{R}^{n\times n}$ et $D \in \mathbb{R}^{n\times 1}$

$x(k)$ étant le vecteur des variables d'état tel que

$$x(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ \cdots \\ x_n(k) \end{bmatrix} = \begin{bmatrix} y(k) \\ \Delta^{\alpha}y(k+1) \\ \Delta^{2\alpha}y(k+2) \\ \cdots \\ \Delta^{n\alpha}y(k+n) \end{bmatrix} \tag{3.4}$$

$u(k)$ est l'entrée de la représentation d'état 3.2. Dans la structure de la figure 3.1, l'entrée du bloc (REDLF) est la différence fractionnaire d'ordre $(n\alpha)$ de la sortie du système $\Delta^{n\alpha}y(k)$.
Les sorties du bloc sont les variables d'état (la matrice C est une matrice identité).

Les sorties du bloc (REDLF (3.2.2)) seront bouclées aux entrées du bloc (R.N.A (3.2.1)).

Pour la bonne construction du modèle, réseaux de neurones associé-bloc d'intégrateur d'ordre fractionnaire, on suit les étapes de l'algorithme suivant :

3.2.3 Algorithme de la structure proposée

1. On choisit une structure pour le réseau de neurones (nombre de couches, nombre de neurones dans chaque couche, les fonctions d'activation).

   Les entrées du réseau sont :
   - l'entrée du système $u(k)$.
   - la sortie du système $y(k)$.
   - les dérivées d'ordre fractionnaires de la sortie

     $(\Delta^{\alpha}y(k+1), \Delta^{2\alpha}y(k+2), \Delta^{3\alpha}y(k+3), ..., \Delta^{(n-1)\alpha}y(k+n-1))$

     Dans notre travail, nous utilisons les résultats de simulation du modèle représentation d'état non linéaire d'ordre fractionnaire du système comme système de référence.

     La sortie du réseau est :
   - la dérivée fractionnaire d'ordre $n\alpha$ de la sortie estimée du système
2. On fait l'apprentissage pour ce réseau de neurones d'une manière classique (en choisissant un des algorithmes d'apprentissage). On obtient un réseau de neurones fixe (on fixe les valeurs des poids et celles des biais du réseau).
   - Ce réseau de neurones représente la partie non linaire du système.
3. Après un bon apprentissage du réseau de neurones, on place, à la sortie de ce réseau, le bloc contenant les intégrateurs fractionnaires et à la sortie de ce bloc, on obtient la sortie estimée et ses dérivées fractionnaires

$$(\Delta^{\alpha}y(k+1), \Delta^{2\alpha}y(k+2), \Delta^{3\alpha}y(k+3), ..., \Delta^{(n-1)\alpha}y(k+n-1)) \tag{3.5}$$

4. Au lieu d'utiliser la sortie du système et ses dérivées fractionnaires, qui ne sont pas disponibles, à l'entrée du réseau, on utilise les valeurs de la sortie estimée et ses dérivées d'ordre fractionnaire obtenues à la sortie du bloc de REDLF. C'est-à-dire on introduit un bouclage de la sortie du bloc vers l'entrée du réseau de neurones comme le montre la figure (3.1).

Pour la modélisation des systèmes non linéaires d'ordre fractionnaire, on suit les étapes de l'algorithme proposé. On applique le modèle choisi pour le système à une seule variable d'état (Exemple (2.7.1)), puis le système à deux variables d'état (Exemple(2.7.2)).

3.3 Résultats obtenus pour le système à un état

Rappelons que le système non linéaire d'ordre fractionnaire considéré est donné par

$$\begin{cases} \Delta^{\alpha} x(k+1) = -0.1x^3(k) + u(k) \\ y(k) = x(k) \end{cases} \tag{3.6}$$

Pour simuler ce système, nous avons utilisé la définition de Grünwald-Letnikov pour le calcul de la dérivée d'ordre fractionnaire. Nous prenons la condition initiale $x(0) = 0$. Le système d'équation (3.6) devient

$$\begin{cases} \Delta^{\alpha} x(k+1) = -0.1x^3(k) + u(k) \\ x(k+1) = \Delta^{\alpha} x(k+1) - \sum_{j=1}^{k+1} \left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} x(k-j+1) \right) \\ y(k) = x(k) \end{cases} \tag{3.7}$$

3.3.1 Conception du Réseau de Neurones

Le modèle non linéaire sera représenté par un réseau de neurones associé à un ensemble d'intégrateurs fractionnaires. Dans ce qui suit nous présenterons le détail de conception du réseau de neurones.

1. **Architecture du réseau de neurones**

   Nous utilisons un réseau de neurones à 3 couches, 2 entrées et 1 sortie.

   Le réseau de neurones est constitué de deux neurones avec des fonctions d'activation non linéaires (tansig) dans la couche cachée et un neurone linéaire dans la couche de sortie. La figure 3.2 montre la structure du réseau de neurones qui modélise l'aspect non linéaire.

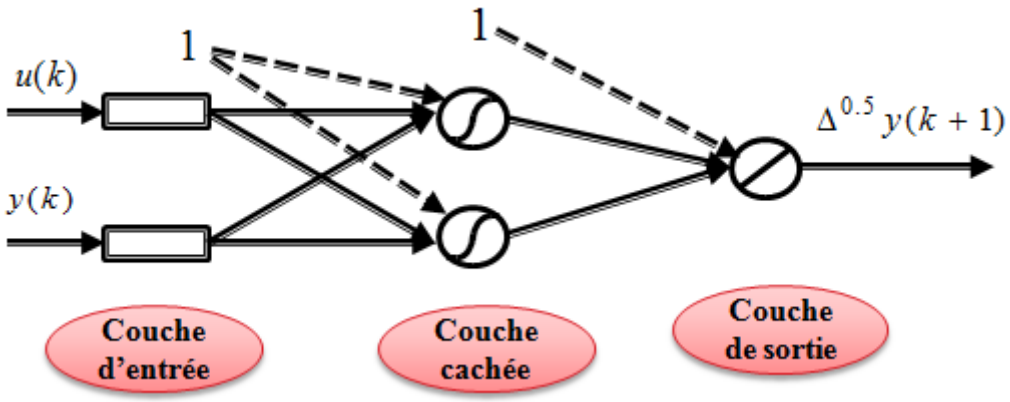


FIGURE 3.2: Structure du réseau de neurones utilisé

Les vecteurs d'entrées pour ce réseau sont de la forme :

$$\begin{cases} \left[\begin{array}{cccc} u(0) & u(1) & ... & u(k) \end{array} \right] \\ \left[\begin{array}{cccc} y(0) & y(1) & ... & y(k) \end{array} \right] \end{cases} \tag{3.8}$$

et le vecteur de sortie a la forme

$$\left[\begin{array}{cccc} \Delta^{0.5}y(1) & \Delta^{0.5}y(2) & ... & \Delta^{0.5}y(k+1) \end{array} \right] \tag{3.9}$$

2. **Phase d'apprentissage**

On fixe les paramètres du réseau de neurones qui sont les poids et les biais. On utilise les signaux d'apprentissage $u(k)$, $y(k)$ et $\Delta^{0.5}y(k)$ du système qui sont représentés dans la figure 3.3, avec $u(k)$ est un signal sinusoïdal, $y(k)$ est la réponse du système et $\Delta^{0.5}y(k)$ est la dérivée d'ordre 0.5 de la réponse du système.

La figure 3.4 représente la structure d'apprentissage du réseau de neurones, on injecte le même signal d'entrée au système et le modèle réseau de neurones, et la sortie du système sera injectée comme une deuxième entrée du réseau. On calcule la dérivée d'ordre 0.5 de la réponse du système, par une des méthodes d'approximation de calcul de la dérivée d'ordre fractionnaire (dans notre travail nous utilisons la méthode d'approximation de Grünwald-Letnikov), ensuite on calcule l'erreur entre la dérivée d'ordre 0.5 de la réponse du système et la sortie du réseau de neurones, en appliquant un algorithme d'apprentissage pour l'adaptation des paramètres du réseau qui minimise l'erreur entre la réponse du système ($\Delta^{0.5}y(k+1)$) et la sortie du réseau qui est $\Delta^{0.5}y_{est}(k+1)$.

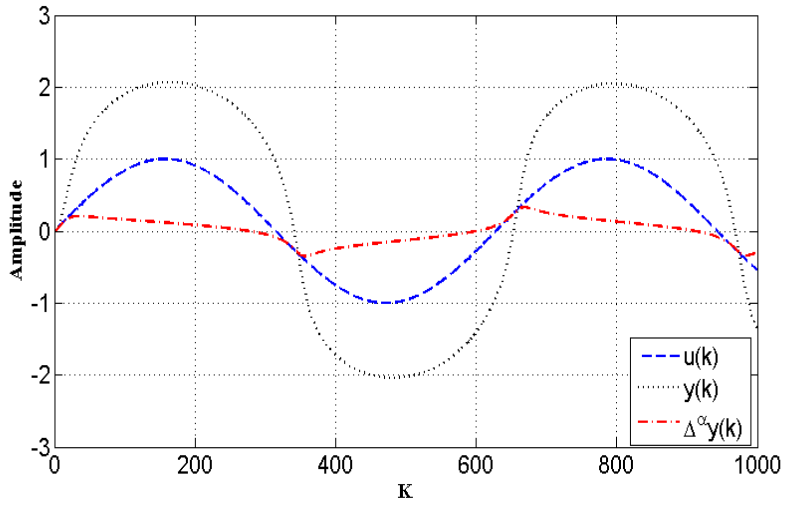


FIGURE 3.3: Signaux d'apprentissage

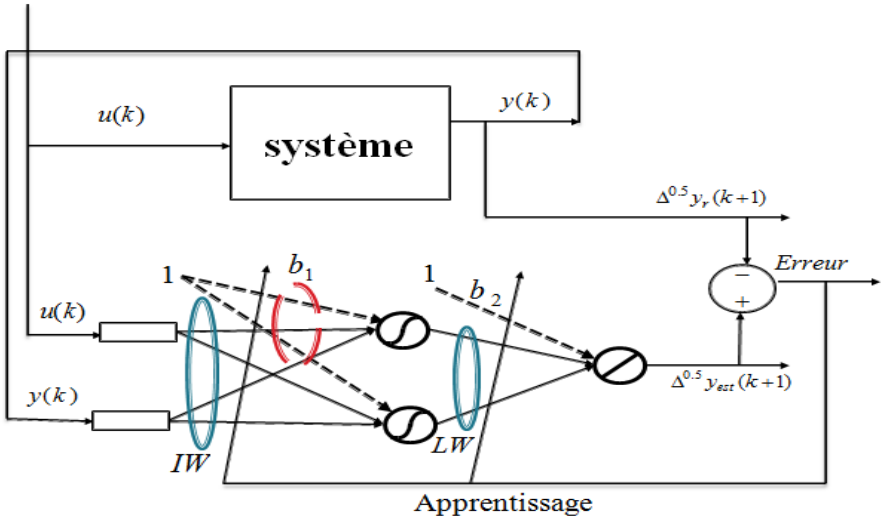
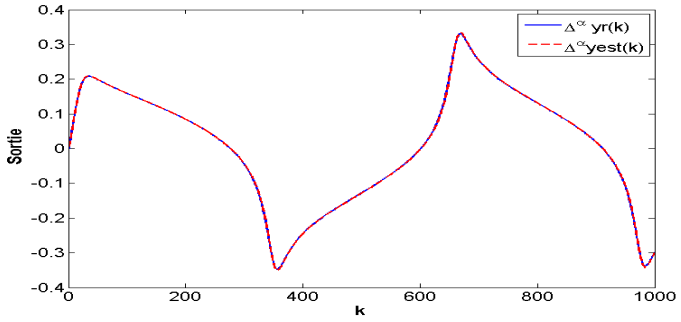
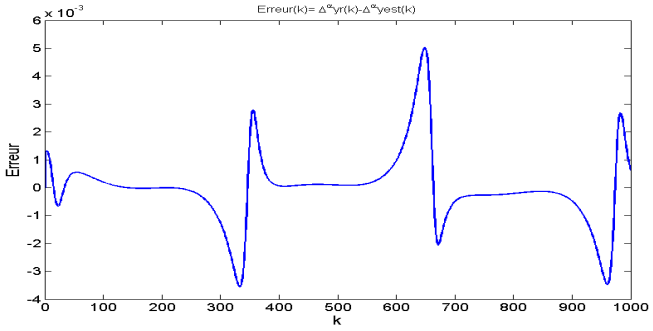


FIGURE 3.4: Structure d'apprentissage du réseau de neurones

La figure 3.5(a) montre que le signal de la dérivée d'ordre 0.5 de la sortie du système et le signal de la sortie du réseau de neurones sont confondus. La figure 3.5(b) présente l'évolution de l'erreur entre ces deux signaux ; on remarque que cette erreur tend vers zéro avec une faible valeur de l'erreur quadratique moyenne de $1.57\,10^{-6}$. Néanmoins, l'erreur présente quelques pics qui s'expliquent par le changement brusque de l'amplitude de la réponse du système. On constate que le réseau de neurones est bien entraîné.



(a) Dérivée d'ordre 0.5 de la réponse du système et la sortie du réseau de neurones



(b) Erreur entre la dérivée réelle de la réponse du système et la sortie du RNA

FIGURE 3.5: Résultats de l'apprentissage

Finalement, les paramètres définitifs du réseau de neurones sont :

– Les paramètres (poids) de connexion entre la couche d'entrée et les autres couches du réseau de neurones sont :

$$IW = \begin{bmatrix} [2 \times 2] \\ [] \end{bmatrix} \tag{3.10}$$

$$IW\{1\} = \begin{bmatrix} -0.0018 & 0.2468 \\ -0.0026 & 0.1033 \end{bmatrix} \tag{3.11}$$

$$IW\{2\} = [] \tag{3.12}$$

IW (Input Weight) : vecteur colonne de cellules contenant les poids d'entrée (les poids qui relient la couche d'entrée et les couches qui suivent).

$IW\{1\}$: est la matrice des poids qui relient les entrées de la couche d'entrée et les neurones de la première couche qui suit(dans cet exemple, c'est la couche cachée). La taille de la matrice est (2×2) c'est-à-dire deux (02) entrées qui sont reliées à deux (02) neurones de la couche cachée.

$IW\{2\}$: est la matrice des poids qui relient les entrées de la couche d'entrée et les neurones de la deuxième couche (couche de sortie). On remarque que cette matrice est vide, car il n'existe pas de connexions entre la couche d'entrée et la couche de sortie.

– Les paramètres (poids) de connexion entre les couches cachées et la couche de sortie du réseau de neurones sont :

$$LW = \begin{bmatrix} [] & [] \\ [1 \times 2] & [] \end{bmatrix} \tag{3.13}$$

$$LW\{1,1\} = LW\{1,2\} = LW\{2,2\} = [] \tag{3.14}$$

$$LW\{2,1\} = \begin{bmatrix} 0.6434 & -1.5370 \end{bmatrix} 10^3 \tag{3.15}$$

LW (Layer Weight) : matrice de cellule des matrices des poids entre les couches(les couches cachées et la couche de sortie).

$LW\{1,1\}, LW\{1,2\}, LW\{2,2\}$: Ces matrices sont vides car il n' existe pas de connexions qui relient, respectivement, entre les neurones de la couche cachée, les neurones de la couche de sortie vers les neurones de la couche cachée et entre les neurones de la couche de sortie.

$LW\{2,1\}$: c'est la matrice des poids de connexion qui relient les neurones de la couche cachée aux neurones de la couche de sortie.

– Les paramètres (biais) des couches cachées et la couche de sortie du réseau de neurones

$$b = \begin{bmatrix} [2 \times 1] \\ -1.9273 \end{bmatrix} \tag{3.16}$$

$$b\{1\} = \begin{bmatrix} 0.0010 \\ -0.0008 \end{bmatrix} \tag{3.17}$$

$$b\{2\} = [-1.9273] \tag{3.18}$$

b : c'est un vecteur colonne de cellules contenant les biais des couches du réseau de neurone, dans ce cas la taille du vecteur est (2×1), car on a une couche cachée et une couche de sortie.

$b\{1\}$: est le vecteur de biais des neurones de la couche cachée. La taille de ce vecteur est (2×1) car on a deux (02) neurones dans la couche cachée.

$b\{2\}$: est le vecteur de biais des neurones de la couche de sortie. Dans ce cas, ce vecteur est un scalaire car on a un seul neurone dans la couche de sortie.

3. **Phase de test**

Pour tester le réseau de neurones entier, on injecte d'autres signaux à l'entrée du réseau obtenu dans la phase d'apprentissage, l'entrée test est le signal $'U1'$ de la figure 3.6, la réponse du système obtenue est donnée sur la figure 3.7. On compare ensuite la sortie du réseau de neurones à la dérivée d'ordre 0.5 du système.

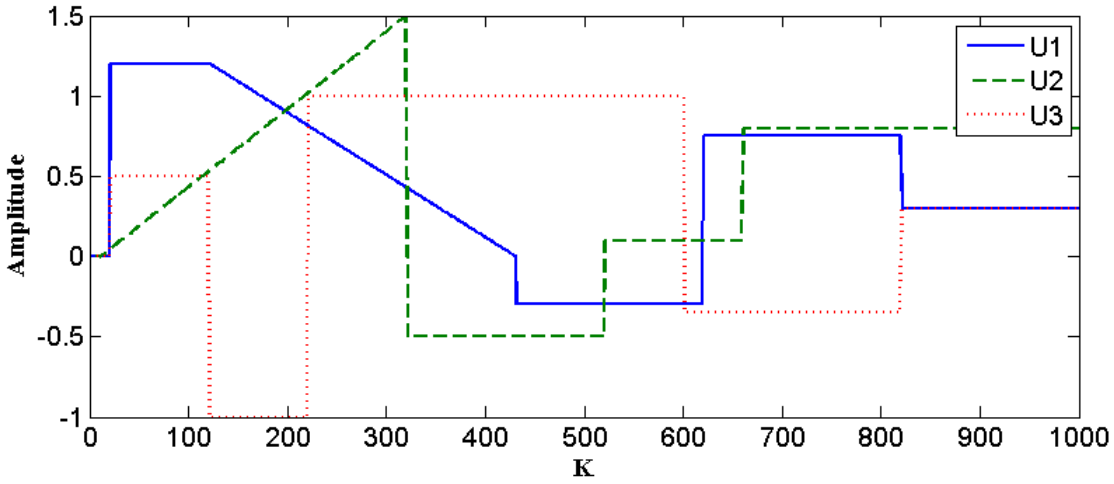


FIGURE 3.6: Signaux tests

Les signaux tests sont des signaux qui sont différents du signal d'apprentissage.

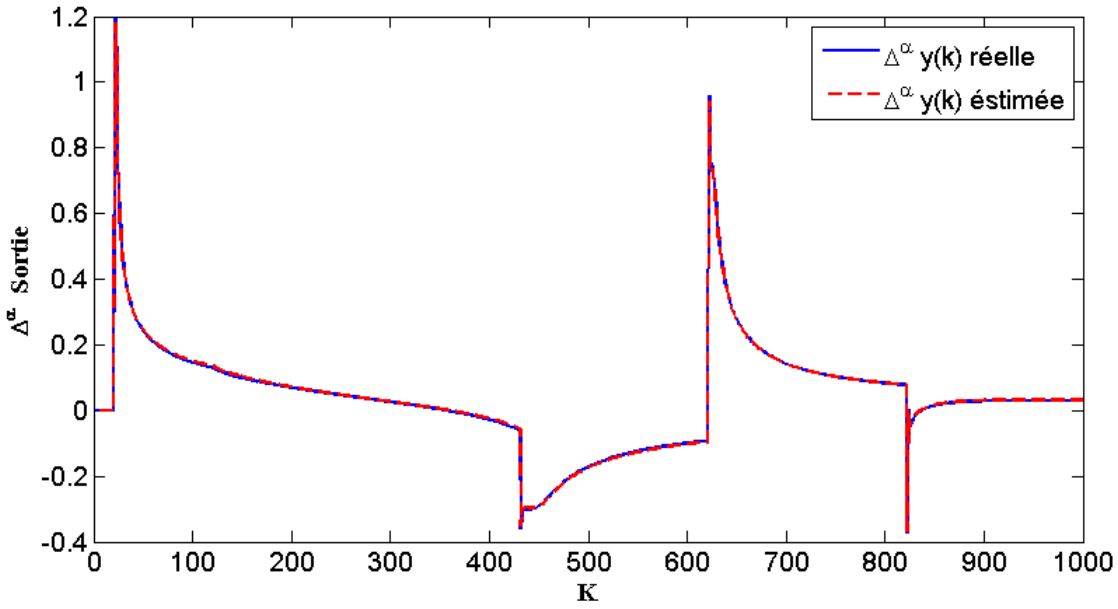


FIGURE 3.7: Dérivée 0.5 de la sortie du système et la réponse du réseau de neurones pour l'entrée $U1$

De la figure 3.7 on remarque que la sortie du réseau de neurones est confondue avec la dérivée d'ordre 0.5 de la réponse du système. L'erreur quadratique moyenne est 8.10^{-6}. On conclut que le réseau de neurones est bien entraîné.

3.3.2 Association du réseau de neurones avec le bloc d'intégrateurs fractionnaires

Dans notre travail, l'objectif est d'obtenir la réponse du système, et non pas sa dérivée d'ordre 0.5. Pour cela, on relie la sortie du réseau à un bloc de représentation d'état d'ordre fractionnaire suivant :

$$\begin{cases} \Delta^{0.5}x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Du(k) \end{cases} \tag{3.19}$$

dont les matrices sont

$$A = [0], B = [1], C = [1], D = [0]. \tag{3.20}$$

L'ordre de ce bloc est $\alpha = 0.5$

Ce bloc joue le rôle de l'intégrateur d'ordre fractionnaire dans la structure de la figure 3.8.

On injecte une entrée $U1$ représentée par la figure 3.6 à l'entrée du réseau de neurones et on prend la condition initiale nulle pour la deuxième entrée du réseau de neurones.

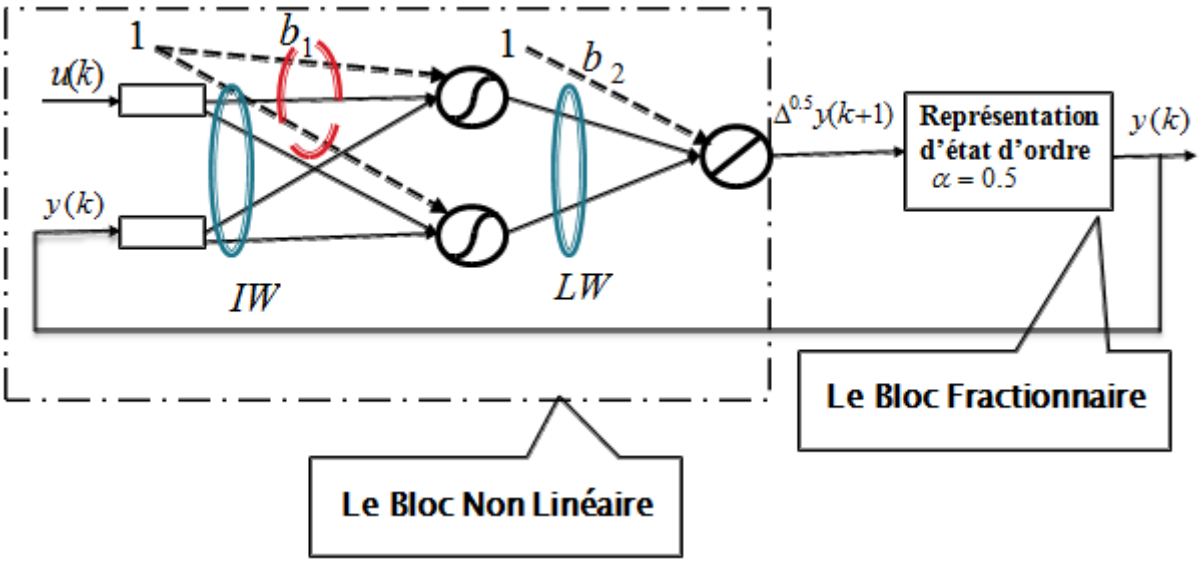
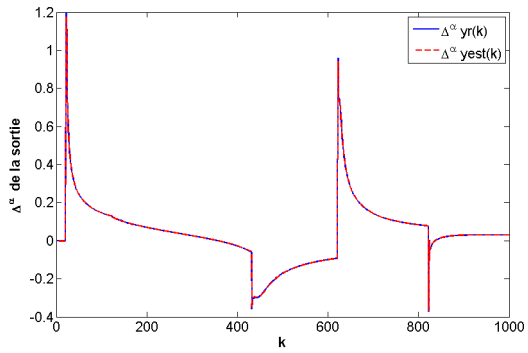
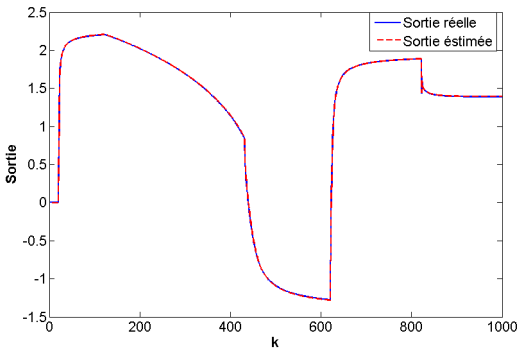


FIGURE 3.8: Structure du RdNs-IsFs

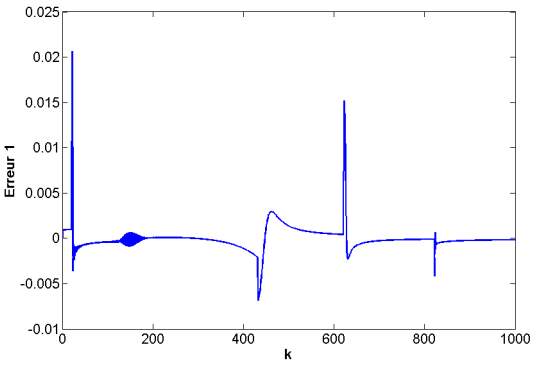
Les résultats de simulation sont représentés par la figure 3.9. le signal de la dérivée d'ordre 0.5 de la réponse du système et le signal de la sortie du réseau de neurones entier de la figure 3.9(a) sont confondus avec une erreur quadratique moyenne de $2.07\,10^{-6}$. La figure 3.9(c) montre l'évolution de l'erreur de ces deux signaux. Donc la non linéarité qui est représentée par le réseau de neurone entier est bien maîtrisée. Les figures 3.9(b) et 3.9(d) montrent respectivement la réponse du modèle RdNs-IsFs avec la réponse réelle et l'évolution de l'erreur entre ces deux réponses. On remarque que le modèle a une bonne précision avec une erreur quadratique moyenne de $3.24\,10^{-5}$.



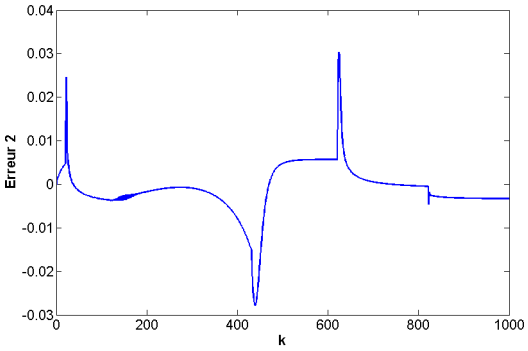
(a) Dérivée d'ordre 0.5 de la réponse du système et la sortie du réseau de neurones



(b) Réponse du système et la sortie du RdNs-IsFs



(c) Erreur de prédiction entre la dérivée d'ordre 0.5 de la réponse du système et la sortie du réseau de neurones



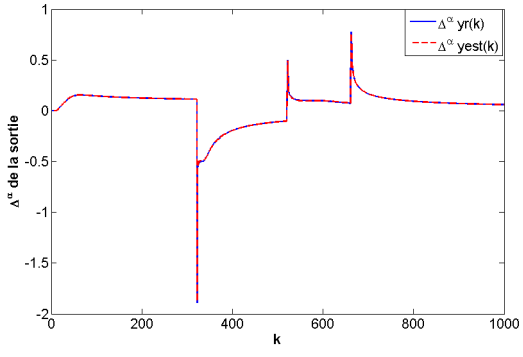
(d) Erreur de prédiction entre la réponse du système et la sortie du RdNs-IsFs

FIGURE 3.9: Résultats de simulation pour l'entrée U_1

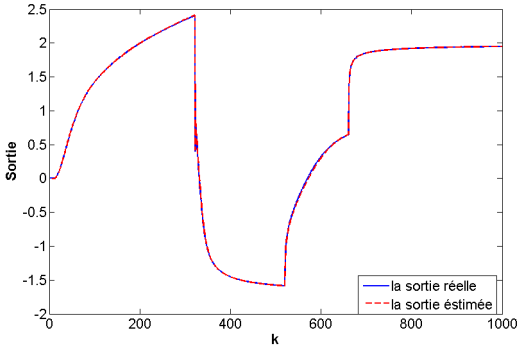
Pour la validation du modèle RdNs-IsFs, on injecte d'autres signaux d'entrées $U2$ puis $U3$ de la figure 3.6.

Les résultats de simulation sont représentés dans les figures 3.10 et 3.11

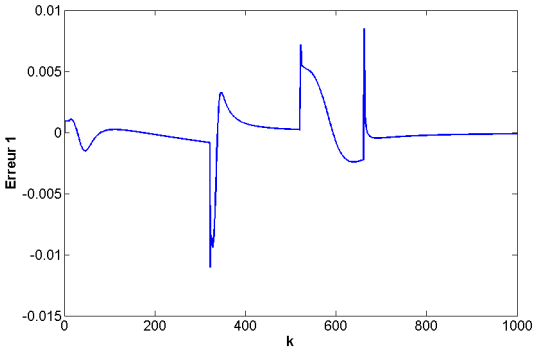
– **Résultats de simulation du modèle RdNs-IsFs pour une entrée test $U2$**



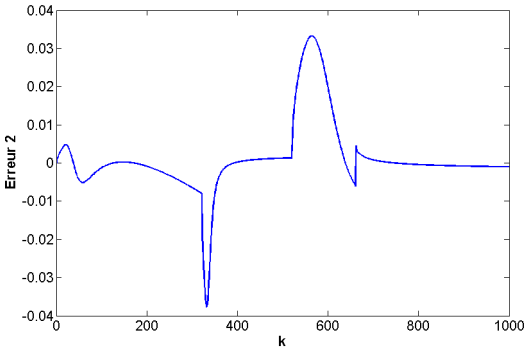
(a) Dérivée d'ordre 0.5 de la réponse du système et la sortie du réseau de neurones



(b) Réponse du système et la sortie du RdNs-IsFs



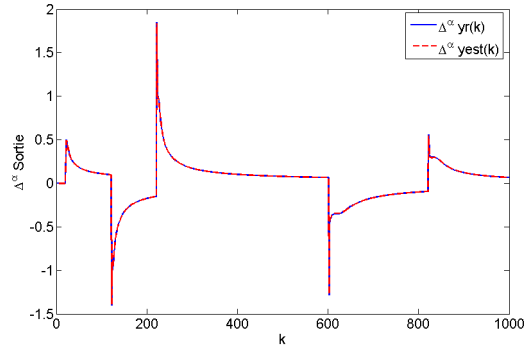
(c) Erreur de prédiction entre la dérivée d'ordre 0.5 de la réponse du système et la sortie du réseau de neurones



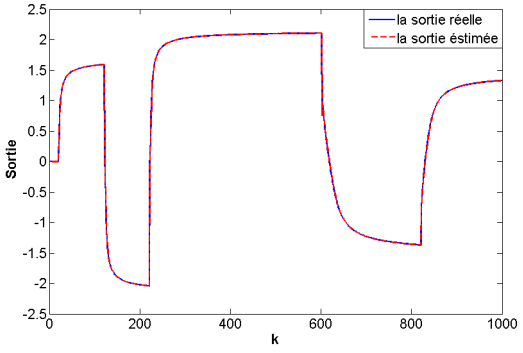
(d) Erreur de prédiction entre la réponse du système et la sortie du RdNs-IsFs

FIGURE 3.10: Résultats de simulation pour l'entrée U_2

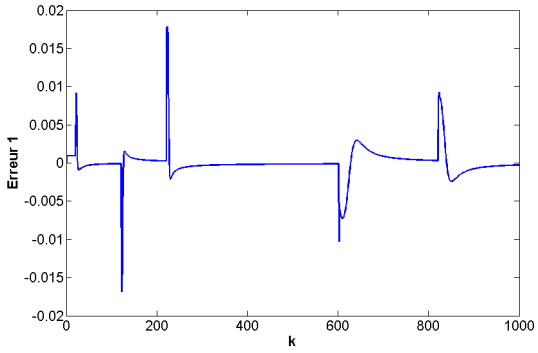
– **Résultats de simulation du modèle RdNs-IsFs pour une entrée test $U3$**



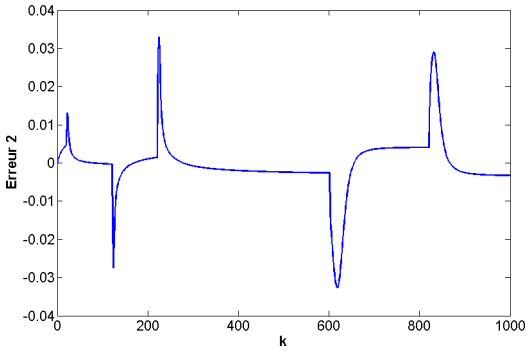
(a) Dérivée d'ordre 0.5 de la réponse du système et la sortie du réseau de neurones



(b) Réponse du système et la sortie du RdNs-IsFs



(c) Erreur de prédiction entre la dérivée d'ordre 0.5 de la réponse du système et la sortie du réseau de neurones



(d) Erreur de prédiction entre la réponse du système et la sortie du RdNs-IsFs

FIGURE 3.11: Résultats de simulation pour l'entrée U_3

On remarque que la réponse du système et la sortie du modèle RdNs-IsFs sont confondues pour l'entrée $U2$ voir figure 3.10(b), et de même pour l'entrée $U3$ figure 3.11(b).

Le modèle RdNs-IsFs représenté par la figure 3.8 et dont les paramètres sont fixés dans la phase d'apprentissage (3.15), représente avec une bonne précision le système non linéaire d'ordre fractionnaire de l'exemple 2.7.1.

3.4 Test de Robustesse du Réseau de Neurones vis à vis de l'ordre du système fractionnaire

Afin de montrer que le réseau de neurones représente l'aspect non linéaire et que le bloc d'intégrateur d'ordre fractionnaire représente seulement l'aspect fractionnaire du système non linéaire d'ordre fractionnaire, on entraîne le réseau de neurones, qui est représenté par la figure 3.4, du système d'ordre α_1 et on fait le test pour le système d'ordre α_2.

3.4.1 Test lorsque $\alpha_1 \geq \alpha_2$

On entraîne le réseau de neurones à deux entrées et une sortie. Les entrées du réseau sont l'entrée et la sortie du système. La sortie du réseau est la dérivée d'ordre $\alpha_1 = 0.8$ de la sortie estimée.

On utilise un réseau de neurones de la structure à trois couches (une couche d'entrée qui est composée de deux (02) entrées, une couche cachée a deux (02) neurones dont les fonctions d'activation sont tangentes sigmoïdes(tansig) et une couche de sortie à un seul neurone linéaire).

Après l'apprentissage du réseau, nous avons obtenu les paramètres suivants

– Les paramètres (poids) de connexion entre la couche d'entrée et les autres couches

$$IW = \begin{bmatrix} [2 \times 2] \\ [] \end{bmatrix} \tag{3.21}$$

$$IW\{1\} = \begin{bmatrix} 0.0004 & -0.2838 \\ 0.0027 & -0.1189 \end{bmatrix} \tag{3.22}$$

$$IW\{2\} = [] \tag{3.23}$$

– Les paramètres (poids) de connexion entre les couches cachées et la couche de sortie

$$LW = \begin{bmatrix} [] & [] \\ [1 \times 2] & [] \end{bmatrix} \tag{3.24}$$

$$LW\{1,1\} = LW\{1,2\} = LW\{2,2\} = [] \tag{3.25}$$

$$LW\{2,1\} = \begin{bmatrix} -0.8340 & 1.9905 \end{bmatrix} 10^3 \tag{3.26}$$

– Les paramètres (biais) des couches cachées et la couche de sortie

$$b = \begin{bmatrix} [2 \times 1] \\ 35.0069 \end{bmatrix} \tag{3.27}$$

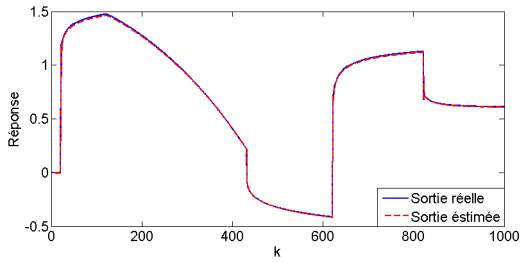
$$b\{1\} = \begin{bmatrix} -0.0090 \\ -0.0214 \end{bmatrix} \tag{3.28}$$

$$b\{2\} = [35.0069] \tag{3.29}$$

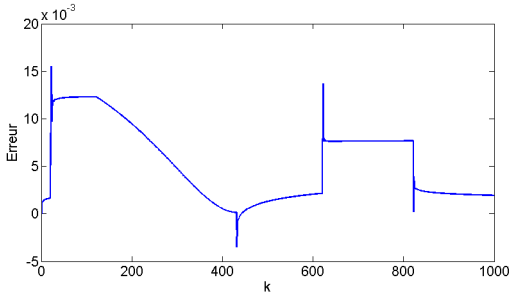
Nous associons le réseau de neurones obtenu avec un bloc d'intégrateur fractionnaire d'ordre α_2 (avec $\alpha_2 = 0.1, 0.2, 0.3, 0.5, 0.8$) de même structure que la figure 3.8.

On teste le modèle pour les systèmes ayant la même non linéarité avec différents ordres fractionnaires $\alpha_2 = 0.1, 0.2, 0.3, 0.5, 0.8$.

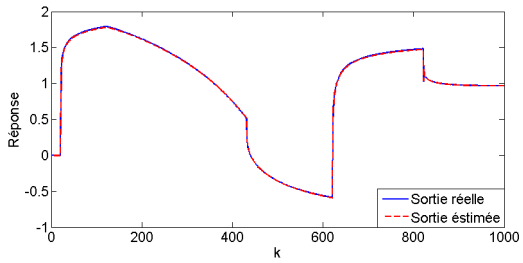
Les résultats de simulation obtenus sont représentés dans la figure 3.12



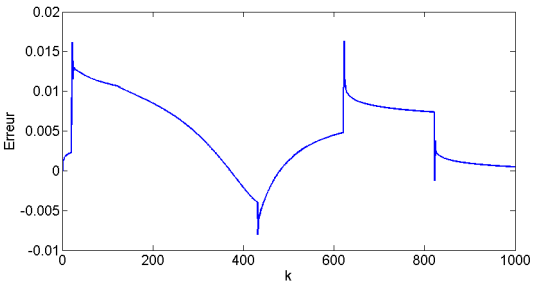
(a) Réponse du système et la sortie du RdNs-IsFs d'ordre $\alpha_2 = 0.1$



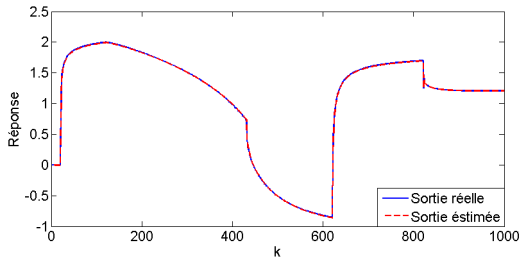
(b) Erreur entre la sortie réelle et la sortie estimée du système non linéaire d'ordre $\alpha_2 = 0.1$



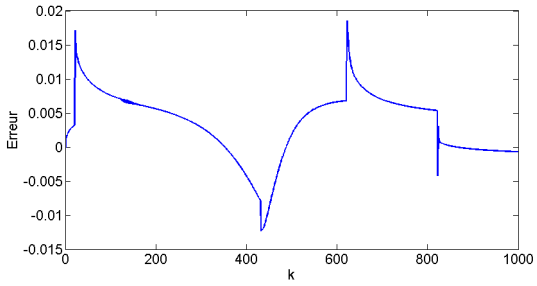
(c) Réponse du système et la sortie du RdNs-IsFs d'ordre $\alpha_2 = 0.2$



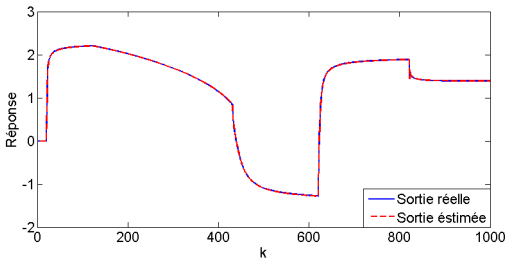
(d) Erreur entre la sortie réelle et la sortie estimée du système non linéaire d'ordre $\alpha_2 = 0.2$



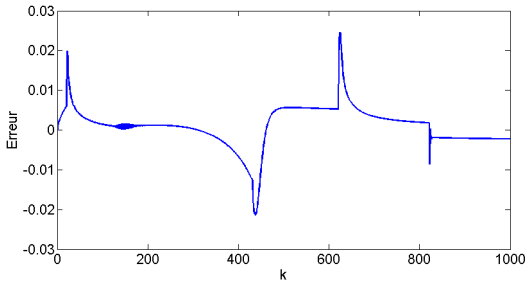
(e) Réponse du système et la sortie du RdNs-IsFs d'ordre $\alpha_2 = 0.3$



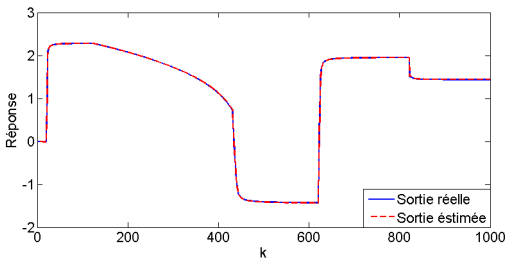
(f) Erreur entre la sortie réelle et la sortie estimée du système non linéaire d'ordre $\alpha_2 = 0.3$



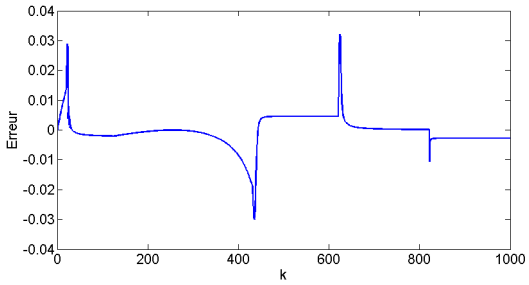
(g) Réponse du système et la sortie du RdNs-IsFs d'ordre $\alpha_2 = 0.5$



(h) Erreur entre la sortie réelle et la sortie estimée du système non linéaire d'ordre $\alpha_2 = 0.5$



(i) Réponse du système et la sortie du RdNs-IsFs d'ordre $\alpha_2 = 0.8$



(j) Erreur entre la sortie réelle et la sortie estimée du système non linéaire d'ordre $\alpha_2 = 0.8$

FIGURE 3.12: Résultats de simulation des modèles, RdNs-IsFs, pour différentes valeurs de α_2 avec le même réseau de neurones entraîné pour $\alpha_1 = 0.8$ de systèmes de même non linéarité de différents ordres fractionnaires α_2 =0.1, 0.2, 0.3, 0.5, 0.8

L'erreur quadratique moyenne entre la réponse du système et la sortie du Réseau de Neurones-Intégrateurs Fractionnaires pour chaque valeur de α_2 est représentée dans le tableau (3.1).

α_2	0.1	0.2	0.3	0.5	0.8
EQM	$4.36.10^{-5}$	$4.21.10^{-5}$	$4.43.10^{-5}$	$2.61.10^{-5}$	$2.55.10^{-5}$

TABLE 3.1: Erreur quadratique moyenne pour différents systèmes de différents ordres fractionnaires

La figure 3.12 et les valeurs de l'erreur quadratique moyenne pour chaque α_2 représentées dans le tableau (3.1) montrent la bonne précision des modèles qui représentent les systèmes qui ayant un même aspect non linéaire avec des ordres des dérivées fractionnaires différentes.

3.4.2 Test lorsque $\alpha_1 \leq \alpha_2$

On utilise un réseau de neurones de la structure à trois couches (une couche d'entrée qui se compose de deux (02) entrées, une couche cachée à trois (03) neurones dont les fonctions d'activation sont tangentes sigmoïdes(tansig) et une couche de sortie à un seul neurone linéaire).

Les entrées du réseau sont l'entrée et la sortie du système. La sortie du réseau est la dérivée d'ordre $\alpha_1 = 0.1$ de la sortie estimée.

Après l'apprentissage du réseau, nous avons obtenu les paramètres suivants

– Paramètres (poids) de connexion entre la couche d'entrée et les autres couches

$$IW = \begin{bmatrix} [3 \times 2] \\ [] \end{bmatrix} \tag{3.30}$$

$$IW\{1\} = \begin{bmatrix} -0.1742 & 1.0733 \\ -0.2512 & -0.1224 \\ 0.3931 & -1.3204 \end{bmatrix} \tag{3.31}$$

$$IW\{2\} = [] \tag{3.32}$$

– Paramètres (poids)de connexion entre les couches cachée et la couche de sortie

$$LW = \begin{bmatrix} [] & [] \\ [1 \times 3] & [] \end{bmatrix} \tag{3.33}$$

$$LW\{1,1\} = LW\{1,2\} = LW\{2,2\} = [] \tag{3.34}$$

$$LW\{2,1\} = \begin{bmatrix} -2.2812 & -4.2406 & 2.5730 \end{bmatrix} \tag{3.35}$$

– Paramètres (biais) des couches cachées et de la couche de sortie

$$b = \begin{bmatrix} [3 \times 1] \\ 0.6663 \end{bmatrix} \tag{3.36}$$

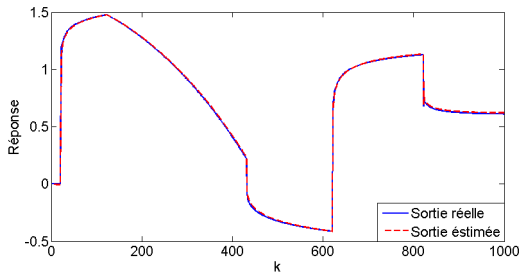
$$b\{1\} = \begin{bmatrix} -1.8131 \\ 0.0767 \\ -2.1105 \end{bmatrix} \tag{3.37}$$

$$b\{2\} = [0.6663] \tag{3.38}$$

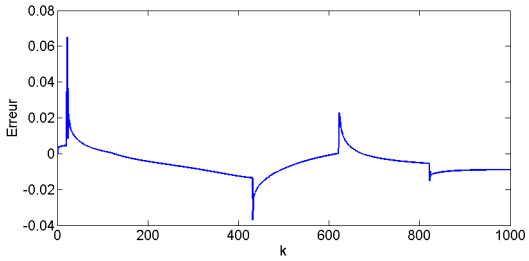
Nous associons le réseau de neurones (entraîné pour $\alpha_1 = 0.1$) avec un bloc d'intégrateurs fractionnaires d'ordre α_2 (avec $\alpha_2 = 0.1, 0.2, 0.3, 0.5, 0.8$) de même structure que la figure 3.8.

On teste le modèle pour les systèmes de même non linéarité avec différentes dérivées d'ordre fractionnaire $\alpha_2 = 0.1, 0.2, 0.3, 0.5, 0.8$.

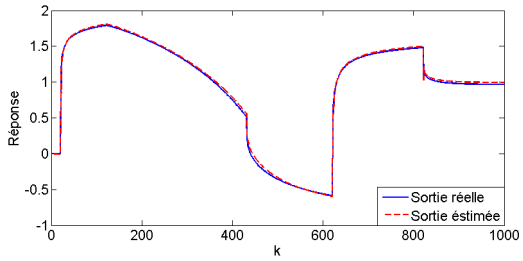
Les résultats de simulation obtenus sont représentés dans la figure 3.13



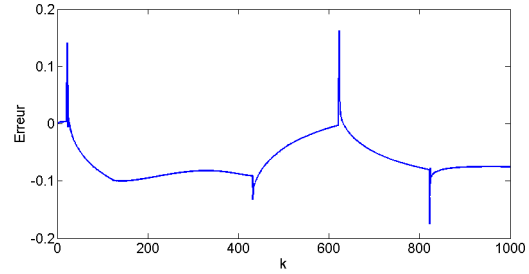
(a) Réponse du système et la sortie du RdNs-IsFse $\alpha_2 = 0.1$



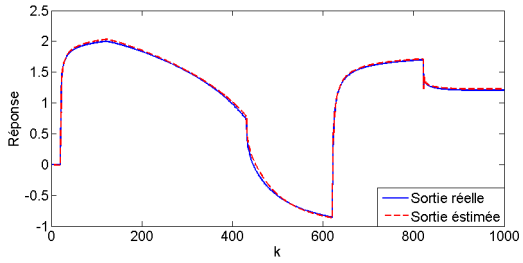
(b) Erreur entre la sortie réelle et la sortie estimée du système fractionnaire d'ordre $\alpha_2 = 0.1$



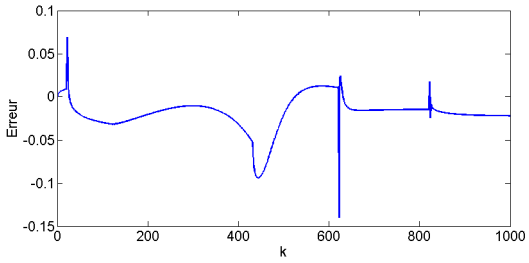
(c) Réponse du système et la sortie du r RdNs-IsFs d'ordre $\alpha_2 = 0.2$



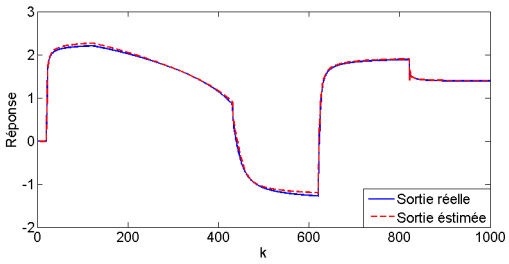
(d) Erreur entre la sortie réelle et la sortie estimée du système fractionnaire d'ordre $\alpha_2 = 0.2$



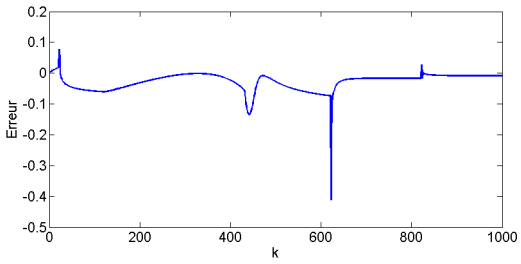
(e) Réponse du système et la sortie du RdNs-IsFs d'ordre $\alpha_2 = 0.3$



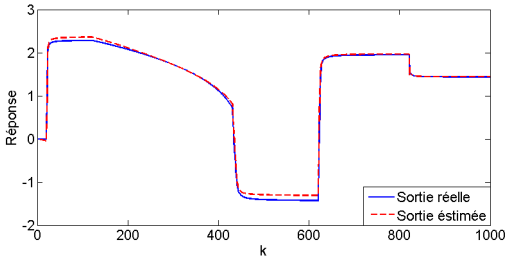
(f) Erreur entre la sortie réelle et la sortie estimée du système fractionnaire d'ordre $\alpha_2 = 0.3$



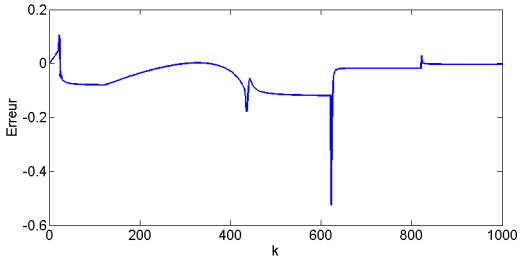
(g) Réponse du système et la sortie du RdNs-IsFs d'ordre $\alpha_2 = 0.5$



(h) Erreur entre la sortie réelle et la sortie estimée du système fractionnaire d'ordre $\alpha_2 = 0.5$



(i) Réponse du système et la sortie du RdNs-IsFs d'ordre $\alpha_2 = 0.8$



(j) Erreur entre la sortie réelle et la sortie estimée du système fractionnaire d'ordre $\alpha_2 = 0.8$

FIGURE 3.13: Résultats de simulation des modèles, RdNs-IsFs, pour différentes valeurs de α_2 avec le même réseau de neurones entrainé pour $\alpha_1 = 0.1$ de systèmes de même non linéarité de différents ordres fractionnaires α_2 =0.1, 0.2, 0.3, 0.5, 0.8

.

L'erreur quadratique moyenne entre la sortie réelle et celle estimée pour différentes valeurs de α_2 est donnée par le tableau (3.2).

α_2	0.1	0.2	0.3	0.5	0.8
EQM	$6.97\,10^{-5}$	$4.93\,10^{-4}$	$7.16\,10^{-4}$	$0.16\,10^{-2}$	$0.42\,10^{-2}$

TABLE 3.2: Erreur quadratique moyenne pour différents systèmes de différents ordres fractionnaires

Les résultats de simulations de la figure 3.13 et l'erreur quadratique moyenne pour différentes valeurs de α_2 donnée par le tableau 3.2 montrent que le modèle RdNs-IsFs avec les paramètres du réseau de neurones sont entraînés pour la dérivée d'ordre $\alpha_1 = 0.1$ peut représenter les systèmes

de même non linéarité avec différents ordres fractionnaires (α_2 =0.1, 0.2, 0.3, 0.5, 0.8).

Le modèle RdNs-IsFs avec le réseau de neurones entrainé pour $\alpha_1 = 0.1$ est moins précis que le modèle dont son réseau de neurones est entrainé pour $\alpha_1 = 0.8$.

3.5 Résultats obtenus pour le système à deux (02) états

Le deuxième système est représenté par :

$$\begin{cases} \Delta^{2\alpha} x_1(k+2) = -0.1\, x_2^3(k) + u(k) \\ \qquad\qquad \text{Avec} \\ x_2(k) = \Delta^{2\alpha} y(k+2) \end{cases} \tag{3.39}$$

le système 3.39 peut être représenté par la représentation d'état de la forme

$$\begin{cases} \Delta^{\alpha} x_1(k+1) = x_2(k) \\ \Delta^{\alpha} x_2(k+1) = -0.1\, x_2^3(k) + u(k) \\ y(k) = x_1(k) \end{cases} \tag{3.40}$$

Pour la simulation du système on utilise la définition du Grünwald-Letnikov.

$$\begin{cases} \Delta^{\alpha} x_1(k+1) = x_2(k) \\ x_1(k+1) = \Delta^{\alpha} x_1(k+1) - \sum_{j=1}^{k+1} \left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} x_1(k-j+1) \right) \\ \Delta^{\alpha} x_2(k+1) = -0.1 x_2^3(k) + u(k) \\ x_2(k+1) = \Delta^{\alpha} x_2(k+1) - \sum_{j=1}^{k+1} \left((-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} x_2(k-j+1) \right) \\ y(k) = x_1(k) \end{cases} \tag{3.41}$$

On prend les conditions initiales $x_1(0) = 0$ et $x_2(0) = 0$.

On modélise le système de cet exemple par un RdNs-IsFs discret qui est composé de deux blocs. Le premier représente l'aspect non linéaire et le deuxième représente l'aspect fractionnaire.

3.5.1 Conception du Réseau de Neurones

Dans cette section, nous expliquerons les étapes à suivre pour la conception du réseau de neurones utilisé. Pour le cas du système à deux variables d'état.

1. **Architecture du réseau de neurones**

   Le réseau de neurones de type MLP à 3 couches ; Une couche d'entrée à 3 entrées, Une couche cachée à 2 neurones dont leur fonction d'activation est "tansig" et une couche de sortie à un seul neurone avec une fonction d'activation linéaire.

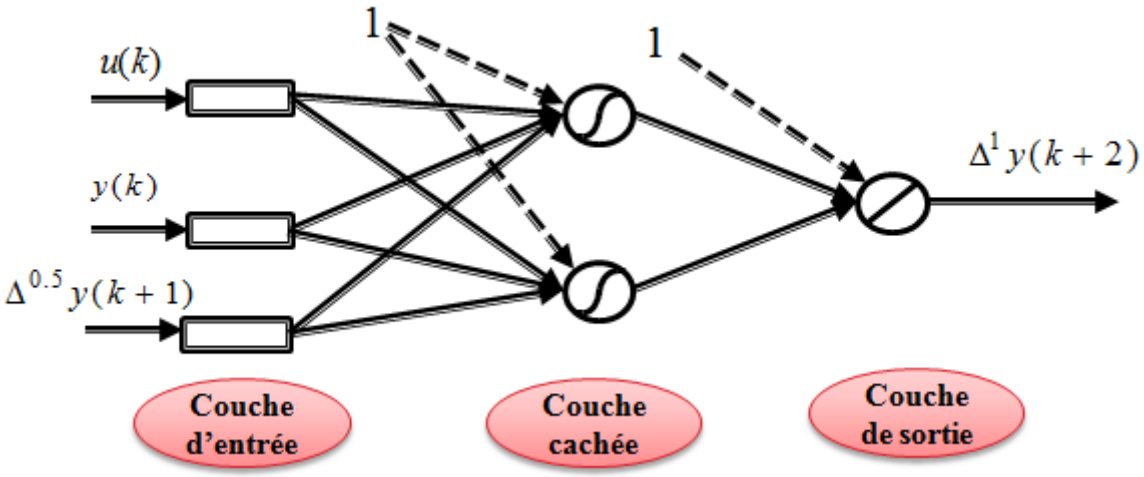


FIGURE 3.14: Structure du réseau de neurones qui modélise l'aspect non linéaire pour le système non linéaire d'ordre fractionnaire à deux variables

Les vecteurs d'entrée sont :

$$\begin{cases} \left[\begin{array}{cccc} u(0) & u(1) & ... & u(k) \end{array} \right] \\ \left[\begin{array}{cccc} \Delta^{0.5}y(1) & \Delta^{0.5}y(2) & ... & \Delta^{0.5}y(k+1) \end{array} \right] \\ \left[\begin{array}{cccc} y(0) & y(1) & ... & y(k) \end{array} \right] \end{cases} \tag{3.42}$$

et le vecteur de sortie a la forme suivante :

$$\left[\begin{array}{cccc} \Delta^{1}y(2) & \Delta^{1}y(3) & ... & \Delta^{1}y(k+2) \end{array} \right] \tag{3.43}$$

2. **Phase d'apprentissage**

On fixe les paramètres du réseau de neurones qui sont les poids et les biais, nous utilisons les signaux d'apprentissage $u(k)$, $y(k)$, $\Delta^{0.5}y(k)$ et $\Delta^{1}y(k)$ du système. $u(k)$ est un signal sinusoïdal, $y(k)$ est la réponse du système non linéaire d'ordre fractionnaire à deux variables d'état, $\Delta^{0.5}y(k)$ est la dérivée d'ordre 0.5 de la sortie et$\Delta^{1}y(k)$ est la dérivée première de la réponse du système.

La structure de l'apprentissage du réseau de neurones est représentée dans la figure 3.15. On injecte trois (03) entrées dans la couche d'entrée du réseau. Ces entrées sont l'entrée du système $u(k)$, la réponse du système $y(k)$ et la dérivée d'ordre 0.5 de la réponse du système $\Delta^{0.5}y(k)$. La sortie estimée du réseau de neurones est la dérivée d'ordre 1 de la réponse du système $\Delta^{1}y(k)$. On applique l'algorithme d'apprentissage et on adapte des paramètres du

réseau de neurones minimisant l'erreur entre la sortie du réseau et la dérivée d'ordre 1 de la réponse du système.

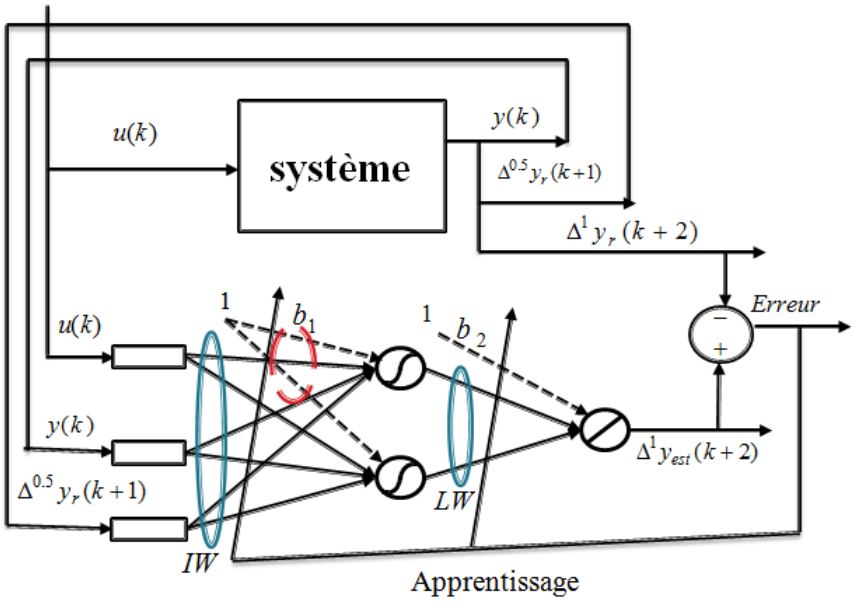


FIGURE 3.15: Structure d'apprentissage du réseau de neurones

Après l'apprentissage, nous avons obtenu les paramètres suivants :

les poids d'entrée

$$IW = \begin{bmatrix} [2 \times 3] \\ [] \end{bmatrix} \tag{3.44}$$

$$IW\{1\} = \begin{bmatrix} 0.0120 & -0.0017 & -0.2760 \\ 0.0120 & 0.0030 & 0.4972 \end{bmatrix} \tag{3.45}$$

$$IW\{2\} = [] \tag{3.46}$$

$$LW = \begin{bmatrix} [] & [] \\ [1 \times 2] & [] \end{bmatrix} \tag{3.47}$$

les poids des couches

$$LW\{1,1\} = LW\{1,2\} = LW\{2,2\} = [] \tag{3.48}$$

$$LW\{2,1\} = \begin{bmatrix} 151.5169 & 84.1872 \end{bmatrix} \tag{3.49}$$

les biais des neurones du réseau

$$b = \begin{bmatrix} [2 \times 1] \\ -1.9273 \end{bmatrix} \tag{3.50}$$

$$b\{1\} = \begin{bmatrix} -0.0130 \\ 0.0095 \end{bmatrix} \tag{3.51}$$

$$b\{2\} = [1.1806] \tag{3.52}$$

Après avoir fixé les paramètres du réseau de neurones, nous relions à la sortie un bloc d'intégrateurs d'ordre fractionnaire.

3.5.2 Association du réseau de neurones avec bloc d'intégrateurs fractionnaires

Le bloc fractionnaire est une représentation d'état d'ordre fractionnaire qui est formulée comme suit :

$$\begin{cases} \Delta^{0.5}x(k+1) = Ax(k) + Bu(k) \\ y(k) = Cx(k) + Du(k) \end{cases} \tag{3.53}$$

avec :

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{3.54}$$

Ce bloc joue le rôle d'un double intégrateur d'ordre non entier 0.5.

Les sorties du bloc fractionnaire seront rebouclées vers l'entrée du réseau de neurones, comme le montre la figure 3.16. On obtient ainsi la structure permettant de représenter le système linéaire d'ordre fractionnaire de l'équation 3.41.

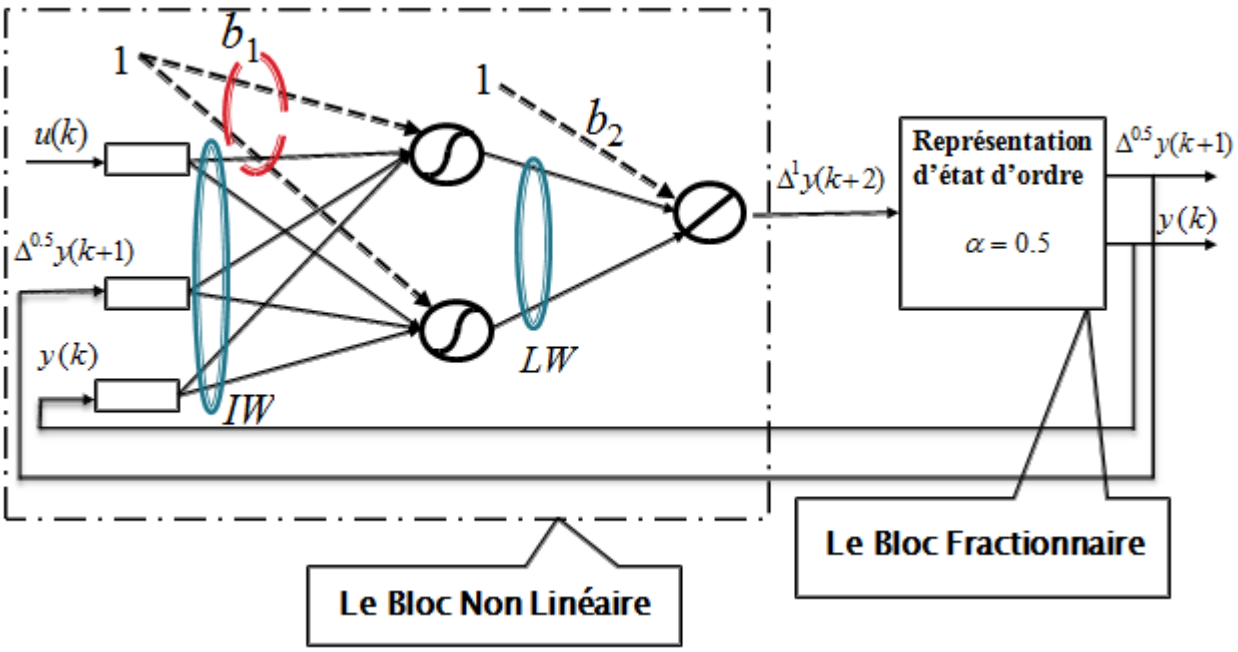
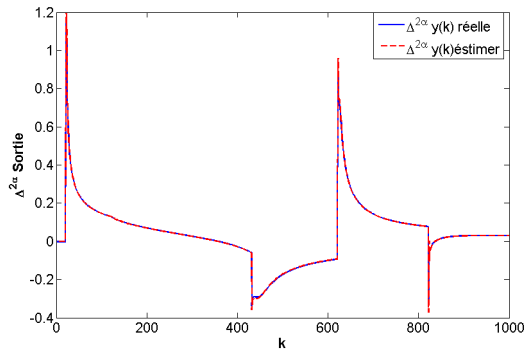


FIGURE 3.16: Structure du modèle RdNs-IsFs

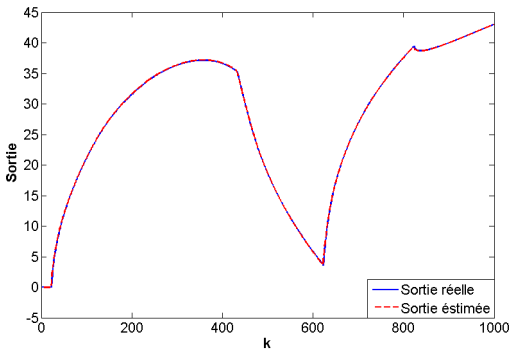
3. **Phase test**

Comme pour le premier système, dans cette phase on injecte les entrées tests pour valider le modèle conçu.

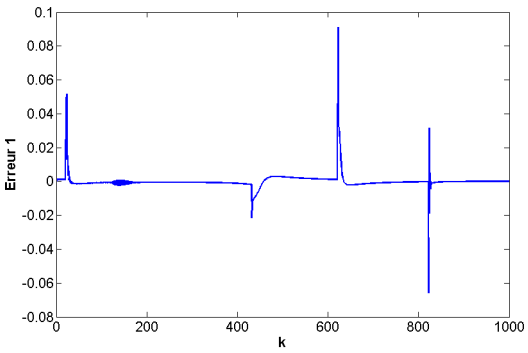
– **Résultats de simulation du modèle RdNs-IsFs pour l'entrée test $U1$ représenté par la figure 3.6**



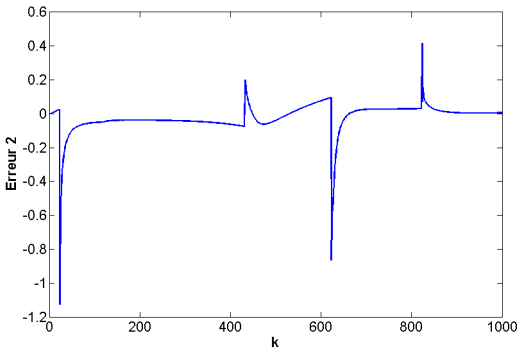
(a) Dérivée d'ordre 1 de la réponse du système et la sortie su réseau de neurones



(b) Réponse du système et la sortie du RdNs-IsFs



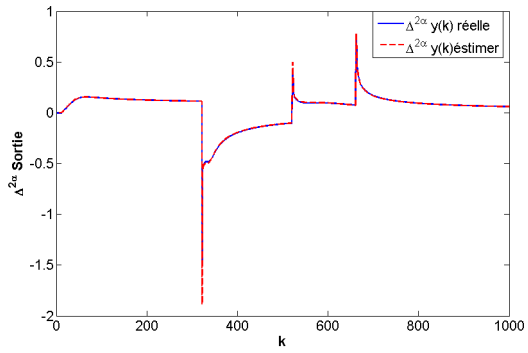
(c) Erreur de prédiction entre la dérivée d'ordre 1 de la réponse du système et la sortie du réseau de neurones



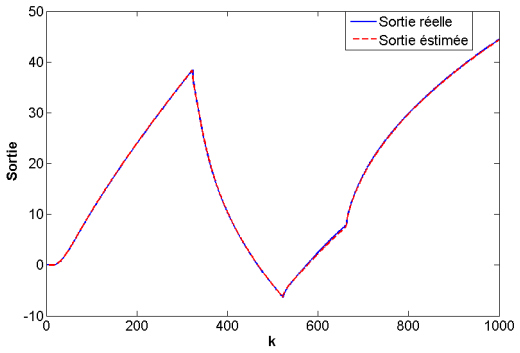
(d) Erreur de prédiction entre la réponse du système et la sortie du RdNs-IsFs

FIGURE 3.17: Résultats de simulation pour l'entrée U_1

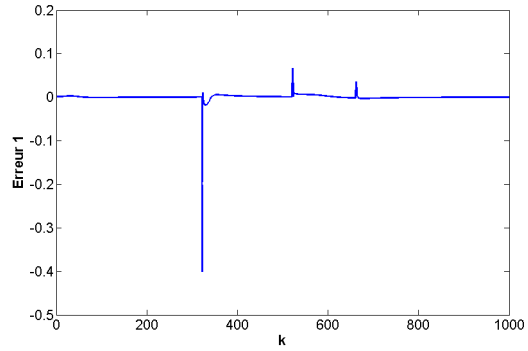
– **Résultats de simulation du modèle RdNs-IsFs pour l'entrée test $U2$ représenté par la figure 3.6**



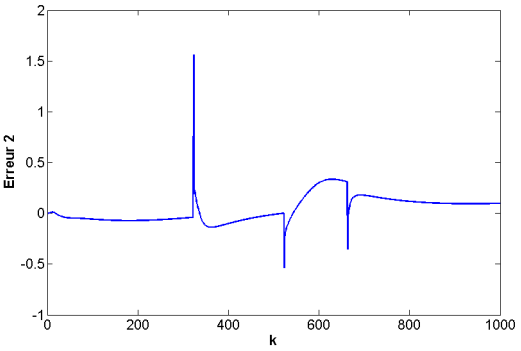
(a) Dérivée d'ordre 1 de la réponse du système et la sortie su réseau de neurone



(b) Réponse du système et celle du RdNs-IsFs



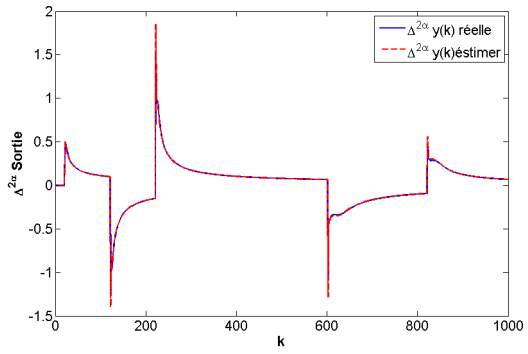
(c) Erreur de prédiction entre la dérivée d'ordre 1 de la réponse du système et la sortie du réseau de neurones



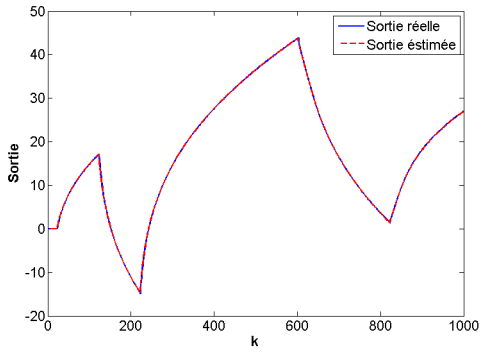
(d) Erreur de prédiction entre la réponse du système et la sortie du RdNs-IsFs

FIGURE 3.18: Résultats de simulation pour l'entrée U_2

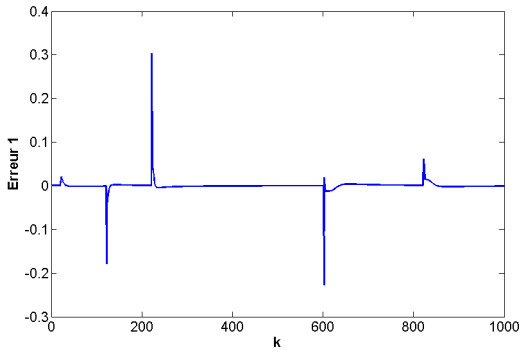
– **Résultats de simulation du modèle RdNs-IsFs pour l'entrée test $U3$ représentée par la figure 3.6**



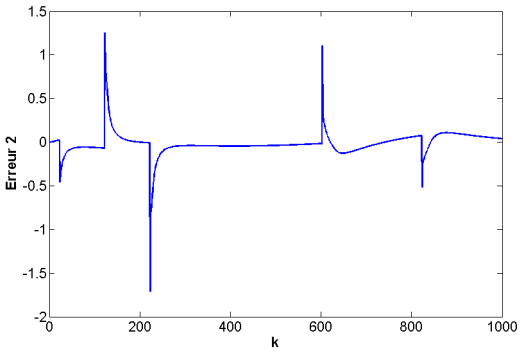
(a) Dérivée d'ordre 1 de la réponse du système et la sortie su réseau de neurone



(b) Réponse du système et la sortie du RdNs-IsFs



(c) Erreur de prédiction entre la dérivée d'ordre 1 de la réponse du système et la sortie du réseau de neurones



(d) Erreur de prédiction entre la réponse du système et la sortie du RdNs-IsFs

FIGURE 3.19: Résultats de simulation pour l'entrée U_3

Les résultats de simulation montrent la bonne précision du modèle, pour les entrées U_1, U_2 et U_3. La dérivée d'ordre 1 de la réponse du système et la sortie du système pour chaque entrée sont confondues comme le montrent les figures 3.17(a), 3.18(a) et 3.19(a). Pour montrer la précision entre la dérivée d'ordre 0.5 du système et la sortie du réseau, nous avons tracé l'erreur entre ces deux signaux pour les entrées U_1, U_2 et U_3 respectivement. Elles sont illustrées dans les figures 3.17(c), 3.18(c) et 3.19(c).

Notre objectif dans ce travail est de modéliser le système non linéaire d'ordre fractionnaire par un RdNs-IsFs, les résultats de simulations montrent la précision du modèle conçu puisque la sortie réelle du système est confondue avec la sortie du modèle pour les différentes entrées U_1, U_2 et U_3 et l'erreur d'estimation sont représentées dans les figures (3.17(d), 3.18(d), 3.19(d)). Ces figures montrent la précision du modèle. En revanche, on remarque la présence des pics qui se justifient par le changement brusque du signal d'entrée.

– Pour illustrer les choix des structures des réseaux de neurones utilisées dans les exemples d'application du système 1, on calcule l'erreur quadratique moyenne entre la sortie du RdNs-IsFs et la réponse système non linéaire fractionnaire pour différentes valeurs de α en fonction du nombre de neurones de la couche cachée, pour différents signaux d'entrée.

U_1					
α / NN dans la cc	0.1	0.2	0.3	0.5	0.8
2 Neurones	$1.33\,10^{-4}$	$\mathbf{7.19\,10^{-5}}$	$\mathbf{5.98\,10^{-5}}$	$\mathbf{3.24\,10^{-5}}$	$\mathbf{2.55\,10^{-5}}$
3 Neurones	$\mathbf{6.97\,10^{-5}}$	$8.52\,10^{-5}$	$6.62\,10^{-5}$	$7.74\,10^{-5}$	$2.27\,10^{-4}$
6 Neurones	$9.85\,10^{-5}$	$8.18\,10^{-5}$	$9.04\,10^{-5}$	$3.20\,10^{-4}$	$5.78\,10^{-4}$
U_2					
2 Neurones	$5.17\,10^{-4}$	$\mathbf{7.60\,10^{-5}}$	$\mathbf{8.90\,10^{-5}}$	$\mathbf{8.97\,10^{-5}}$	$\mathbf{8.42\,10^{-5}}$
3Neurones	$\mathbf{8.36\,10^{-5}}$	$9.12\,10^{-5}$	$3.02\,10^{-4}$	$4.06\,10^{-4}$	$1.82\,10^{-4}$
6 Neurones	$8.68\,10^{-4}$	$4.72\,10^{-4}$	$7.46\,10^{-4}$	$1.10\,10^{-3}$	$2.80\,10^{-3}$
U_3					
2Neurones	$6.27\,10^{-4}$	$\mathbf{1.02\,10^{-4}}$	$\mathbf{1\,10^{-4}}$	$\mathbf{5.21\,10^{-5}}$	$\mathbf{3.87\,10^{-5}}$
3Neurones	$\mathbf{4.47\,10^{-4}}$	$1.10\,10^{-4}$	$3.88\,10^{-4}$	$4.87\,10^{-4}$	$5.74\,10^{-4}$
6 Neurones	$0.13\,10^{-2}$	$5.95\,10^{-4}$	$7.96\,10^{-4}$	$0.12\,10^{-2}$	$0.59\,10^{-2}$

TABLE 3.3: Récapitulatif des erreurs quadratiques moyennes entre les RdNs-IsFs et les systèmes à une seule variable d'état de même non linéarité et dont l'ordre de la dérivée fractionnaire est différent avec différentes structures pour le réseau de neurones.

Avec

NN dans la cc : Nombre de neurones dans la couche cachée.

Le tableau (3.3) montre la bonne précision du modèle RdNs-IsFs. Néanmoins, on remarque des petites différences dans les résultats selon l'architecture du réseau de neurones classique choisi et le système à modéliser. En effet, pour le système fractionnaire non linéaire dont l'ordre de la dérivée est 0.1, l'erreur quadratique moyenne est faible lorsque nous avons utilisé, dans le bloc réseaux de neurones classique, trois (03) neurones dans la couche cachée, par contre pour les autres systèmes où l'ordre de la dérivée d'ordre fractionnaire est (0.2, 0.3, 0.5 et 0.8), l'erreur quadratique moyenne est faible lorsque nous avons utilisé deux (02) neurones dans la couche

cachée du bloc réseau de neurones classique. Lorsque nous avons utilisé six (06) neurones dans l'architecture du bloc (R.N.A), les modèles sont moins précis ; ce qui explique le phénomène de sur-apprentissage du réseau de neurones, Si on ajoute plus de neurones, on aura la diminution de la précision du modèle.

Pour les trois (03) signaux test (U_1, U_2 et U_3), les valeurs qui sont écrites en gras et en grands caractères dans le tableau (3.3) représentent les meilleurs résultats pour chaque signal test. Aussi l'erreur quadratique moyenne entre la réponse désirée et celle estimée change, pour le système non linéaire d'ordre α lorsqu'on utilise des signaux d'entrée différents. Cela s'explique par la différence du degré de la complexité d'un signal d'entrée à l'autre.

– On fait les mêmes tests que le système 1 pour choisir une meilleure structure du réseau de neurones. On calcule l'erreur quadratique moyenne entre la sortie du RdNs-IsFs et la réponse système non linéaire fractionnaire pour différentes valeurs de α en fonction du nombre de neurones de la couche cachée, pour différents signaux d'entrée.

α / NN dans la cc	0.1	0.2	0.3	0.5	0.8
U_1					
2 Neurones	$\mathbf{0.15\,10^{-2}}$	$\mathbf{0.36\,10^{-2}}$	$\mathbf{0.53\,10^{-2}}$	$\mathbf{0.89\,10^{-2}}$	2.88
3 Neurones	$0.15\,10^{-2}$	$0.48\,10^{-2}$	$1.12\,10^{-2}$	$1.04\,10^{-2}$	1.03
6 Neurones	$0.16\,10^{-2}$	$0.74\,10^{-2}$	$2.10\,10^{-2}$	$1.18\,10^{-2}$	$\mathbf{0.47}$
U_2					
2 Neurones	$\mathbf{0.22\,10^{-2}}$	$\mathbf{0.30\,10^{-2}}$	$\mathbf{0.38\,10^{-2}}$	$\mathbf{1.94\,10^{-2}}$	$\mathbf{1.18\,10^{-1}}$
3 Neurones	$0.23\,10^{-2}$	$0.40\,10^{-2}$	$0.54\,10^{-2}$	$7.87\,10^{-2}$	1.32
6 Neurones	$0.22\,10^{-2}$	$0.57\,10^{-2}$	$1.27\,10^{-2}$	$1.80\,10^{-1}$	0.94
U_3					
2 Neurones	$\mathbf{0.35\,10^{-2}}$	$\mathbf{0.57\,10^{-2}}$	$\mathbf{0.86\,10^{-2}}$	$\mathbf{1.89\,10^{-2}}$	0.53
3 Neurones	$0.38\,10^{-2}$	$0.69\,10^{-2}$	$1.64\,10^{-2}$	$2.42\,10^{-2}$	$\mathbf{0.29}$
6 Neurones	$0.38\,10^{-2}$	$1.20\,10^{-2}$	$2.34\,10^{-2}$	$4.05\,10^{-2}$	0.34

TABLE 3.4: Récapitulatif des erreurs quadratiques moyennes entre les RdNs-IsFs et les systèmes à deux variables d'état de même non linéarité et dont l'ordre de la dérivée fractionnaire est différent avec différentes structures pour le réseau de neurones.

Le tableau (3.4) montre les erreurs quadratiques moyennes entre la réponse du système non linéaire d'ordre fractionnaire à deux variables d'état et celle du modèle RdNs-IsFs pour différentes valeurs de α. On remarque que ces valeurs sont moins faibles comparant avec les valeurs qui sont représentées dans le tableau (3.3). La raison est que l'amplitude de la réponse du deuxième exemple pour différentes valeurs de α est plus importante par rapport à l'amplitude de la réponse du premier exemple.

Dans cet exemple les modèles représentant une bonne précision dont l'ordre de dérivée fractionnaire ($\alpha = 0.1$, 0.2, 0.3, 0.5) sont les modèles qui comprennent un bloc réseau de neurones avec deux neurones dans sa couche cachée (les valeurs des erreurs quadratiques moyennes les plus faibles sont représentées dans le tableau (3.4) en gras). Ces modèles représentent une bonne précision quel que soit le signal d'entrée.

Pour le système non linéaire d'ordre $\alpha = 0.8$ à deux variables d'état, l'erreur quadratique moyenne est faible. Dans ce cas, le choix de la structure du bloc réseau de neurones classique est un peu difficile. Car pour l'entrée U_1, nous avons obtenu une meilleure valeur de l'EQM lorsque nous avons utilisé un RdNs-IsFs dont la structure du bloc réseau de neurones entier contient une couche cachée a six (06)neurones pour l'entrée U_2 une couche cachée de deux (02) neurones et pour l'entrée U_3 le meilleur résultat est obtenu, lorsque nous avons utilisé une couche cachée de trois (03) neurones dans le bloc réseau de neurones entier. Mais il faut qu'on choisisse une seule structure qui donne une meilleure précision quel que soit le signal d'entrée injecté au modèle.

Dans ce cas, nous utilisons un RdNs-IsFs dont le bloc réseau de neurones d'ordre entier comprend six neurones dans la couche cachée.

3.6 Conclusion

Ce chapitre permet d'exposer une nouvelle structure de réseaux de neurones artificiels, afin de modéliser les systèmes non linéaires d'ordre fractionnaire. Nous avons expliqué en détail la méthode de conception des modèles de la structure proposée. Puis nous avons appliqué la démarche de la conception du modèle RdNs-IsFs sur des exemples numériques. Dans la première partie nous avons modélisé le système non linéaire d'ordre fractionnaire à une seule variable d'état, nous avons obtenu de bons résultats de simulation avec de faibles erreurs. De même, dans la deuxième partie, nous avons modélisé un système non linéaire d'ordre fractionnaire à deux variables d'état et les résultats de simulation montrent la bonne précision du modèle obtenu.

Conclusion générale

La conception des modèles qui permettent de décrire les systèmes non linéaires d'ordre fractionnaire avec une bonne précision est l'objectif majeur de ce travail.

La démarche suivie dans ce mémoire est la suivante :

Nous avons commencé d'abord par une étude générale sur la modélisation et un état de l'art sur les principaux travaux qui ont été effectués pour la conception des modèles, puis nous avons présenté une étude détaillée sur les réseaux de neurones classiques dans le premier chapitre, plus précisément, les réseaux de neurones multicouches et ses différentes méthodes d'apprentissage.

Nous avons représenté en détail l'algorithme de rétro-propagation de l'erreur. Cette méthode se prête bien pour comprendre la manière d'adaptation des paramètres du réseau à savoir les poids et les biais. Néanmoins, cette méthode risque de tomber dans les minimums locaux. Dans notre travail, nous avons utilisé la méthode de régularisation bayésienne qui est une modification de l'algorithme d'apprentissage de Levenberg-Marquardt et ce dans le but d'éviter de tomber dans le problème de sur-apprentissage. Cette méthode est implémentée dans la toolbox réseau de neurones de **Matlab**, avec la commande ***trainbr***.

Le deuxième chapitre introduit les méthodes de calculs de la dérivée d'ordre fractionnaire. Les modèles d'ordre fractionnaire présentent la propriété d'une mémoire longue mais pour leur simulation, il n'existe pas d'outils directs. Pour ce faire, on est obligé de les approximer par des modèles entiers de grande dimension. Dans notre cas, nous avons utilisé la méthode d'approximation discrète de Grünwald-Letnikov.

Pour la modélisation du système non linéaire d'ordre fractionnaire, nous avons pris on considération la représentation d'état non linéaire d'ordre fractionnaire comme système de référence. Dans le troisième chapitre, nous avons conçu un modèle de réseau de neurones-Intégrateurs fractionnaires qui utilise une base de données des systèmes pour l'apprentissage des paramètres du réseau de neurones classique. Le choix du signal d'entrée pour l'apprentissage du réseau est

important. Nous avons utilisé un signal sinusoïdal qui présente une base de données riche aux amplitudes. Pour le choix de la structure du réseau de neurones (nombre de couches cachées, nombre de neurones dans chaque couche), elle est faite par des essais. C'est l'inconvénient majeur des réseaux de neurones car il n'existe pas une théorie pour le choix de la structure. La sortie du réseau est liée à un intégrateur d'ordre fractionnaire décrit par une représentation d'état d'ordre fractionnaire linéaire.

Les résultats de simulation obtenus avec la structure du Réseau de Neurones-Intégrateur Fractionnaires conçu sont très satisfaisants malgré la complexité des systèmes à modéliser. Dans le cadre de ce travail, nous pouvons proposer les perspectives suivantes

1. Validation des modèles RdNs-IsFs avec la structure proposée, sur une base de données d'un système réel non linéaire d'ordre fractionnaire.
2. Conception d'une autre structure de modèle de réseau de neurones qui comprend l'aspect fractionnaire, et en remplaçant les fonctions d'activation non linéaires par des fonctions d'activation non linéaires d'ordre fractionnaire.
3. Programmation et implantation des fonctions d'activation d'ordre fractionnaire et les introduire dans les commandes de **MATLAB** pour les réseaux de neurones. Ceci permettra de faciliter la tâche de programmation des modèles de réseaux de neurones d'ordre fractionnaire.

Bibliographie

[1] D. Sierociuk, G. Sarwas, A. Dzielinski. Discrete fractional order artificial neural network, acta mechanica et automatica, vol.5 no.2, Institute of Control and Industrial Electronics, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland, 2011.

[2] W. Arren S. Mcculloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biology Vol. 52, No. l/2. pp. 99-115, 1990.

[3] M. Nouressadat. Etude des performances des réseaux de neurones dynamiques à représenter des systèmes réels : une approche dans l'espace d'état, Mémoire de magister en Contrôle, Université de Setif 1, 2009.

[4] M. Önder Efe. Neural network assisted $PI^{\lambda}D^{\mu}$ control, department of Pilotage, Turkish Scientific Council (TÜBITAK) Contract 107E137. University of Turkish Aeronautical Association, Akköprü, Ankara, Turkey.

[5] I. Podlubny. Fractional differential equations . Academie Press, San diego, 1999.

[6] P. Wira. Réseaux de neurones artificiels : architectures et applications, université de Haute Alsace. Laboratoire MIPS (Modélisation. Intelligence. Processus. Système), 2009.

[7] F. Rosenblatt. Two theorems of statistical separabiliiy in the perceptron,pp.419-472, 1960.

[8] A. Mouraud. Approche distribuée pour la simulation événementielle de réseaux de neurones impulsionnels, l'Université des Antilles et de la Guyane, 2009.

[9] I. Rivals. Modélisation et commande de processus par réseaux de neurones ; application au pilotage d'un véhicule autonome. thèse de Doctorat en Physique, l'université Paris 6, 1995.

[10] I. Rivals. Les réseaux de neurones formels Pour le pilotage de robots mobiles, Laboratoire d'électronique de l'ESPCI, FLUX, revue de l'Association amicale Les Ingénieurs Supelec : La robotique mobile ; la fonction achats-logistique, N°178, septembre-octobre 1996, ISSN 0766-3536. École Supérieure de Physique et de Chimie Industrielles, 1996.

[11] K.Arbatni. Réseaux de neurones appliqués à l'analyse et à la modélisation non linéaire du signal ECG, Mémoire de magister en Traitement de signal l'Université Mentouri. Constantine, 2007.

[12] S. Hammouche. Identification d'un modèle fractionnaire à l'aide des réseaux de neurones,Mémoire de magister en automatique, Université Mouloude Mammeri,Tizi-Ouzou, 2012.

[13] R. Muhammad Asif Zahoor, I. M. Qureshi and Junaid Ali Khan. Swarm intelligence optimized neural networks for solving fractional differential equations. ICIC International ISSN 1349-4198 pp.6301-6318. Department of Electronic Engineering International Islamic University H-10, Islamabad, Pakistan, Department of Electrical Engineering Air University Air Headquarters, E-9, Islamabad Pakistan, 2011.

[14] K. Bettou. Analyse et réalisation de correcteurs analogiques d'ordre fractionnaire, thèse de Doctorat en sciences, Université Mentouri de Constantine, 2011.

[15] Concepción A. Monje, YangQuan Chen, Blas M. Vinagre, Dingyü Xue, Vicente Feliu. Fractional-order Systems and Controls, Fundamentals and Applications. Springer London Dordrecht Heidelberg New York, 2010.

[16] G. Dreyfus. Les réseaux de neurones, Mécanique Industrielle et Matériaux n°51 (septembre 1998), École Supérieure de Physique et de Chimie Industrielles de la Ville de Paris (ESPCI), Laboratoire d'Électronique 10, rue Vauquelin 75005 PARIS, 1998.

[17] R. Mansouri. Contribution à l'analyse et la synthèse des systèmes d'ordre fractionnaire par la représentation d'état, Thèse de Doctorat en Electrotechnique, Université Mouloude Mammeri, Tizi-Ouzou, 2008.

[18] T. Djamah. Identification des systèmes par des modèles d'ordre fractionnaire. Thèse de Doctorat en Automatique,Université Mouloude Mammeri, Tizi-Ouzou, 2009.

[19] L. Ait messaoud. Contribution à la Commande des Systèmes par Régulateurs d'Ordre Non Entier : Application à la Commande de la Machine Asynchrone. Mémoire de Magistère en Automatique, Université Mouloude Mammeri, Tizi-Ouzou, 2007.

[20] N. Alauldin. Solutions of dynamic fractional order differential algebraic equations system, Science College, Eng and Tech. Journal, Vol.29, N°3, 2011.

[21] D. Baleanu, Z.B. Güvenc, J.A. Tenreiro machado. New tends in nanothechnology and fractional calculus application. Springer Dordrecht Heidelberg London New York, 2010.

[22] M. Bettayeb, S. Djennoune, S. Guermah. Structural properties of linear discrete-time fractional-ordre systems.17th IFAC world congress, Seoul, South Corea, 6-11 July, 2008.

[23] M. Viberg, B. Ottersten, B. Wahlberg and L. Ljung. A statistical perspective on state space modeling using subspace methods.Proc.30th IEEE Conference on Decision and control, pp 1337-1342.B righton, 1991.

[24] R.L. Bagley and P.J. Torvik. Fractional calculs : a different approach to the analysis of viscoelastic damped structures.AIAA Journal. Vol $21n°5$. PP.741-748, 1983.

[25] R. K. Biswas and S. Sen. Fractional optimal control prolems : a pseudo-state-space approach. Journal of Vibration and Control, 17 : 1034-1041, 2010.

[26] R. Orjuela, R. Malti, M. Moze et A. Oustaloup. Prise en compte des conditions initiales lors de la simulation de fonctions de transfert non entières. In : Proceedings de la Conférence Internationale Francophone d'Automatique (CIFA 2006) 30, 31 Mai et 1er juin 2006 Bordeaux.

[27] K.Tadeuzs. Selected problems of fractional systems theory, Springer-Verlag Berlin Heidelberg, 2011.

[28] G. E. Carlson and C. Halijak. Approximation of fractional capacitors by a regular Newton process. IEEE Transactions on Circuits and Systems, vol.11, n°2, pp.210-213, 1964.

[29] K. Matsuda and H. Fujii. Optimised wave absorbing control : analytical and experimental results. J. Guidance Control and Dynamics, vol.16, n°6, pp.1146-1153, 1993.

[30] A. Charef, H. H. Sun, Y. Y. Tsao, and B. Onaral. Fractal system as represented by singularity function". IEEE Transactions on Automatic Control, vol.37, n°9, pp.14651470, 1992.

[31] A. Oustaloup, F. Levron, B. Mathiew, and F. Nanot. Frequency-band complex noninteger differentiator : characterization and synthesis". IEEE Transactions on Circuits and Systems I, vol.47, n°1, pp.25-39, 2000.

[32] A. Oustaloup. La commande CRONE. Editions HERMES, Paris, 1991.

[33] A. Oustaloup. La Dérivation non entière : théorie, synthèse et applications. Editions HERMES, Paris, 1995.

[34] E. Levy. Complex curve fitting. IRE Transactions on Automatic Control, vol.4, pp.37-43, 1959.

[35] D. Valério and J. S. Da costa. Levy's identification method extended to commensurate fractional order transfer functions. In Fifth EUROMECH Nonlinear Dynamics conference. Eindhoven : EUROMECH, 2005.

[36] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. IEEE Transactions on Power Delivery, vol.14, n°3, pp.1052-1061, 1999.

[37] R.Mansouri, M. Bettayeb, T. Djamah and S. Djennoune. System identification in frequency domain by fractional vector fitting algorithm. Second International Conference on Modelling, Simulation and Applied Optimization, ICMSAO'07, pp.24-27 Mars 2007, Abu Dhabi, E.A.U, 2007.

[38] D.Y.Xue and Y. Q. Chen. Sub-optimum rational approximation to fractional order linear systems. Proceedings of the ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, long Beach, California, USA, 2005.

[39] Y. Q. Chen and K. L. Moore. Discretization schemes for fractional order differentiators and integrators. IEEE Transactions on Circuits and Systems I, vol.49, n°3, pp.363-367, 2002.

[40] B. M. Vinagre, Y. Q. Chen, and I. Petras. Two direct Tustin discretization methods for fractional order differentiator / integrator". Journal of Franklin Institute, pp.349-362, 2003.

[41] Y. Q. Chen, B. M. Vinagre, and I. Podlubny. Using continued fraction to discretize fractional order derivatives. Kluwer Academic Publishers, 2003.

[42] K. Hunt, G. Irwin, K. Warwick. Neural network engineering in dynamic control systems, Springer, 1995.

[43] J. Kalkkuhl, K. Hunt,R. Zbikowski, A. Dzielinski. Applications of neural adaptive control technology, World Scientific, 1997.

[44] P. Borne, G. Dauohin-tanguy, J.P. Richard, F. Rotella, I. Zambitakiss. Modélisation et identification des processus, 1995.

[45] L.Ljung. System identification :theory for the user.Prentice.Hall, London, 1978.

[46] I.D. Landau. Identification et commande des systèmes. Série automatique.2^{eme} édition. Herms Paris, 1993.

[47] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. Division of Chemistry and Biology, California Institute of Technology, Pasadena, California 91125 ; and Bell Laboratories, Murray Hill, New Jersey 07974 Contributed by John J. Hopfweld, January 15, 1982.

[48] T. Kohonen. Self-organized formation of topologically correct feature maps, Department of Technical Physics, Helsinki University of Technology, Espoo, Finland, 1982.

[49] S.G.Samko, A.A.Kilbas et O.I.Marichev Fractional Integrals and Derivatives. Gordon and Breach Science Publishers, 1993.

[50] M. Caputo. Linear models of dissipation whose q is almost frequency independent. Geophysical journal of the royal astronomical society. vol. 2, n° 13, pp 529-539, 1967.

Annexe

Annexe

Définition

Matrice de cellules "cell array", c'est une écriture sous Matlab, est une matrice qui peut recevoir n'importe quel type de données.

Exemple

$$A = \begin{bmatrix} & a & & 1 \\ \begin{bmatrix} 2 & 8 \\ 5 & 3 \end{bmatrix} & & 6 & \end{bmatrix} \tag{55}$$

La matrice A de taille (2×2) se compose des éléments de différents types.

$A\{1,1\} = a$ est une lettre

$A\{1,2\} = 1$ est un scalaire

$A\{2,1\} = \begin{bmatrix} 2 & 8 \\ 5 & 3 \end{bmatrix}$ est une matrice de taille (2×2)

$A\{2,2\} = 6$ est un scalaire

Cette représentation mathématique est utilisée pour la représentation des poids de connexion qui relient les neurones de différentes couches du réseau de neurones artificiel.