Une commande neuronale adaptative basée sur des émulateurs neuronal et multimodèle pour les systèmes non linéaires MIMO et SIMO

Nesrine Bahri

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Gabès

THÈSE de DOCTORAT

en

Co-tutelle

entre

l'Université de Gabès (Tunisie)

et

l'Université du Havre Haute-Normandie (France)

présentée à

l'École Nationale d'Ingénieurs de Gabès

pour l'obtention du titre de

Docteur en Génie Électrique de l'École Nationale d'Ingénieurs de Gabès

et

Docteur en Génie Info, Automatique et Traitement du Signal de l'Université du Havre

par

Nesrine BAHRI

UNE COMMANDE NEURONALE ADAPTATIVE BASÉE SUR DES EMULATEURS NEURONAL ET MULTIMODÈLE POUR LES SYSTÈMES NON LINÉAIRES MIMO ET SIMO

Soutenue le 30 Septembre 2015 devant la commission d'examen :

M. Mohamed Naceur ABDELKRIM	**Professeur ENIG**	**Président**
M. Christophe AUBRUN	**Professeur Univ-Lorraine**	**Rapporteur**
M. Mohamed CHTOUROU	**Professeur ENIS**	**Rapporteur**
M. Fabrice DRUAUX	**Maître de Conférences Univ-Havre**	**Membre**
M. Dimitri LEFEBVRE	**Professeur Univ-Havre**	**Directeur de Thèse**
M. Ridha BEN ABDENNOUR	**Professeur ENIG**	**Directeur de Thèse**

Thèse préparée à l'Unité de Recherche CONPRI de Gabès et au Groupe de Recherche GREAH du Havre

À mon père

À ma mère

À tous ceux qui me sont chères...

Remerciements

Les travaux présentés dans cette thèse sont effectués en cotutelle entre l'Unité de Recherche COmmande Numérique des PRrocédés Industriels, CONPRI de l'Ecole Nationale d'Ingénieurs de Gabès, Tunisie, sous la Direction de Monsieur Ridha BEN ABDENNOUR et le Groupe de Recherche en Electrotechnique et Automatique du Havre, GREAH de l'Université du Havre, France. Avant de présenter ces travaux, je tiens à remercier tous ceux et celles qui ont participé à l'élaboration et à la réussite de ma thèse.

Je tiens à exprimer ma sincère gratitude et ma profonde reconnaissance à mon Directeur de thèse Monsieur Ridha BEN ABDENNOUR, Professeur à l'Ecole Nationale d'Ingénieurs de Gabès (ENIG) et Directeur de l'Unité de Recherche CONPRI, pour m'avoir accueilli dans son unité de recherche et m'avoir encadré. Mon estime à son égard n'a fait que croître au fil du temps, depuis l'année 2009 quand j'ai commencé à travailler sous sa direction dans le cadre de mon Projet de Fin d'Etudes et de mon Mastère jusqu'à présent. Par ses remarques pertinentes et son souci de précision et de clareté, il m'a aidé à structurer et formuler ma pensée. Sa bonne humeur communicative, son intuition sans faille et sa bienveillance constante m'ont permis d'atteindre ce résultat. Qu'il trouve ici le témoignage de ma profonde reconnaissance.

Mes très sincères remerciements à mon Directeur de thèse, Monsieur Dimitri LEFEBVRE, Professeur à l'Université du Havre, pour sa gentillesse, sa disponibilité et sa patience au cours de ces années de thèse. Merci pour la confiance qu'il m'a accordé, pour son vif intérêt pour le bon déroulement de mes séjours au Havre et de l'avancement de mes travaux. Je lui suis reconnaissante pour son soutien humain et moral et ses aides durant ces années de thèse.

Je remercie vivement Monsieur Fabrice DRUAUX, Maitre de Conférences à l'Université du Havre, d'avoir co-encadré cette thèse. Je lui apporte ma plus sincère gratitude pour le temps précieux qu'il m'a consacré tout au long de mes séjours au Havre. Sa gentillesse et sa disponibilité ont notablement contribué à faciliter ma tâche.

Je tiens également à adresser mes plus vifs remerciements à Madame Asma Atig, Maitre Assistante à l'ISSIG, le Co-encadreur de cette thèse, pour son soutien et ses encouragements. Je lui suis reconnaissante pour ses aides particulièrement humaines.

Il m'est impossible d'omettre de remercier Madame Majda LTAIEF, Maitre de Conférences à l'ENIG et Monsieur Anis Messaoud, Maitre Assistant à l'ENIG. Leurs qualités de chercheur, leurs qualités humaines et leur regard bienveillant ont été pour moi une aide inestimable.

Je tiens également à adresser mes plus vifs remerciements aux membres du jury qui m'ont fait l'honneur d'examiner l'ensemble de ces recherches. Je remercie, donc, Monsieur Mohamed Naceur ABDELKRIM, Professeur à l'ENIG, de m'avoir fait l'honneur de présider mon jury de thèse. Mes remerciement s'adressent aussi aux rapporteurs de cette thèse, Monsieur Christophe AUBRUN, Professeur à l'Université de Lorraine, et Monsieur Mohamed CHTOUROU, Professeur à l'ENIS, pour tout le temps consacré à la lecture minutieuse de ce manuscrit de thèse et leur acceptation de l'assistance à la soutenance de ma thèse.

Toute ma reconnaissance va à tous mes collègues de l'Unité CONPRI, pour leur conseils éclairés, leurs informations coopératives et l'ambiance sympathique qu'ils ont su créer.

Je voudrais remercier tous les membres du GREAH pour leur soutien exemplaire et l'ambiance qu'ils ont toujours su faire régner durant mes séjours au Havre. Je vais citer particulièrement Marwa et Mouheb pour leur sympathie et avec lesquels j'ai partagé des beaux moments.

Un grand Merci pour ma chère cousine Maryem et son mari Aymen pour mon chère frère Nihed et sa femme Hiba pour leur soutien humain et moral durant mes séjours en France.

Finalement, je tiens à témoigner ma profonde gratitude à mes parents, mon fiancé, toute ma famille et mes amis pour leur soutien quotidien, leur encouragement et leur immense disponibilité. Ils ont été force d'esprit et d'exemple d'amour pour accomplir ce travail.

Enfin, je tiens à remercier tous ceux qui de près ou de loin, ont contribué à l'accomplissement de ce travail.

Liste de Publications

Articles publiés dans des revues internationales avec comité de lecture

A1 - BAHRI Nesrine, ATIG Asma, BEN ABDENNOUR Ridha, DRUAUX Fabrice & LEFEBVRE Dimitri. "Multimodel and neural emulators for non-linear systems : application to an indirect adaptive neural control". *International Journal of Modelling, Identification and Control (IJMIC). Vol. 17, No. 4, pp.348-359, 2012.*

A2 - BAHRI Nesrine, ATIG Asma, BEN ABDENNOUR Ridha, DRUAUX Fabrice & LEFEBVRE Dimitri. "A Systematic Design of Emulators for Multivariable Non Square and Nonlinear Systems". *International Journal of Automation and Computing (IJAC). Accepted (under press).*

A3 - BAHRI Nesrine, ATIG Asma, BEN ABDENNOUR Ridha, DRUAUX Fabrice & LEFEBVRE Dimitri. "Multivariable Adaptive Neural Control Based on Multimodel Emulator for Nonlinear Square MIMO Systems". *Transactions on Systems, Signals & Devices (TSSD). Vol. 10, No. 1, pp.1-20, 2015.*

A4 - BAHRI Nesrine, MESSAOUD Anis & BEN ABDENNOUR Ridha. "A Multimodel Emulator For Non Linear System Controls". *International Journal of Sciences and Techniques of Automatic control & computer engineering (IJ-STA). Vol. 5, No. 1, pp.1500-1515, 2011.*

Conférences internationales avec comité de lecture

C1 - BAHRI Nesrine, DRUAUX Fabrice, ATIG Asma, BEN ABDENNOUR Ridha & LEFEBVRE Dimitri. "An adaptive neural controller based on neural emulator for single-input multi-output nonlinear systems". 14th *European Control Conference - ECC'15, Linz, Austria, July 2015.*

C2 - BAHRI Nesrine, ATIG Asma, BEN ABDENNOUR Ridha, DRUAUX Fabrice & LEFEBVRE Dimitri. "Multivariable adaptive neural control based on multimodel emulator for nonlinear square MIMO systems". 11th *IEEE International Multi-Conference on Systems, Signals & Devices - SSD'2014, Castelldefels-Barcelona, Spain, February 2014.*

C3 - BAHRI Nesrine, ATIG Asma, BEN ABDENNOUR Ridha, DRUAUX Fabrice & LEFEBVRE Dimitri. "Emulation of Multivariable non square and nonlinear systems". 14th *IEEE International conference on Sciences and Techniques of Automatic control & computer engineering - STA'2013, Sousse, Tunisia, December 2013.*

Notations

Principaux symboles

Tanh(x)	Tangente hyperbolique.
Tanh'(x)	Dérivée de la tanh(x) par rapport x.
δ	Symbole de Kronecker.
B_l	Base de modèles partiels représentant la sortie d'indice l.
$M_{l,i}$	Le i^{ème} modèle local appartenant à une base B_l.

Variables

N_e, N_c	Nombre total des neurones de l'émulateur et du correcteur.
N_{IN}, N_{OUT}	Nombre total d'entrées et des sorties d'un système multivariable.
N	Nombre de neurones d'entrées ou de sorties de l'émulateur ou du correcteur pour un système MIMO carré.
$s_i(k)$	Etat du i^{ème} neurone de l'émulateur.
$o_i(k)$ et $o_{i_l}(k)$	Etat du i^{ème} neurone du correcteur et du correcteur partiel d'indice l.
$x_i(k)$	Entrée du i^{ème} neurone de l'émulateur.
$u_i(k)$	i^{ème} signal de commande.
$w_{ij}(k)$	Poids de connexion du neurone j vers le neurone i de l'émulateur.
$\phi_{ij}(k)$ et $\phi_{ij_l}(k)$	Poids de connexion du neurone j vers le neurone i du correcteur et du correcteur partiel d'indice l.
$1/\tau_e(k)$	Constante de temps des neurones de l'émulateur.
$1/\tau_c(k)$ et $1/\tau_{c_l}(k)$	Constante de temps des neurones du correcteur et du correcteur partiel d'indice l.
$e_e(k)$	Erreur quadratique instantanée d'émulation.
$e_c(k)$	Erreur quadratique instantanée de commande.
$\eta_e(k)$	Facteur d'adaptation de l'émulateur.
$\eta_c(k)$ et $\eta_{c_l}(k)$	Facteur d'adaptation du correcteur et du correcteur partiel d'indice l.
$P_{lij}(k)$, $J_{lk}(k)$, $Q_{kij}(k)$	Fonctions de sensibilité.
$v_l(k)$, $r_l(k)$	Petites perturbations des sorties $\hat{y}_l$ du neurone l.
ε_e, ε_c et ε_{c_l}	Termes de démarrage de l'algorithme (émulateur, correcteur et correcteurs partiels).
ΔT	Pas de calcul.
$y_{n_l}(k)$	Sortie du neurone d'indice l de l'émulateur.
$y_l(k)$	Sortie d'indice l du système.
$y_{c_l}(k)$	Consigne d'indice l.
$y_{ml}(k)$	Sortie du multimodèle représentant la sortie d'indice l.
$y_{l,i}(k)$	Sortie du i^{ème} modèle local de la base B_l.

N_H	Nombre de mesures effectuées sur le système simulé.
$\xi(k)$	Variable d'indexation.
$\mu_{l,i}(\xi(k))$	Fonctions de pondération ou d'activation.
$w_{l,i}(\xi(k))$	Fonctions gaussiennes.
J_{ml}	Critère global pour l'optimisation non linéaire.
θ_l	Vecteur des paramètres de la base B_l.
$G_l(\theta_l)$ et $H_l(\theta_l)$	Le vecteur gradient et la matrice hessienne.
$\vartheta_{l,j}$	Vecteur de régression.
$Pot_{l,j}$	Le potentiel calculé pour la classification des vecteurs de regression.
$c_{l,ie}$	Centre de classe sélectionné comme centre d'une fonction de pondération.
$\sigma_{l,ie}$	Dispersions des fonctions de pondération.
$\upsilon_l{}'(k)$	Validité normalisée du correcteur neuronal partiel d'indice l.

Abréviations

RTRL	Real-Time Recurrent Learning algorithm.
EN	Emulateur Neuronal.
CN	Correcteur Neuronal.
SISO	Single-input and Single-output.
MIMO	Multi-input and Multi-output.
MISO	Multi-input and Single-output.
SIMO	Single-input and Multi-output.
$NMSE$	Normalized Mean Square Error.
RNA(s)	Réseaux de Neurones Artificiels.
MLP	Multi Layer Perceptron.
MSE	Mean Square Error.
VAF	Variance-Accounted-For.
MRAC	Model Reference Adaptive Control.
SCNI	Système de Commande Neuronale Indirecte.
RC	Réseau de commande.
RE	Réseau d'émulation.
T-S	Takagi-Sugeno.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Face aux restrictions environnementales, opérationnelles et énergétiques et à la complexification des procédés industriels, la conception d'une commande efficace pour les systèmes non linéaires multivariables est une question intéressante de la théorie des systèmes. En effet, ces procédés sont souvent caractérisés par un grand nombre d'entrées et de sorties reliées par des intéractions non linéaires et sont souvent soumis à l'influence de signaux d'entrées supplémentaires appelés perturbations. De plus, les procédés peuvent avoir une structure SIMO (Single Input-Multi Output) qui ne dispose que d'un seul actionneur avec deux ou plusieurs sorties. Cette dernière restriction provoque de grandes difficultés pratiques pour la commande de ces procédés. En effet, vu la dynamique des systèmes SIMO, certaines conditions d'inversibilité ne sont pas satisfaites. Par conséquent, pour résoudre les problèmes liés à la commande des systèmes industriels complexes plusieurs approches de commande sont proposées dans la littérature, mais devant les problèmes rencontrés, les résultats sont restreints à certaines classes de systèmes et souffrent de fortes contraintes [12; 13; 58; 63].

Grace à leurs capacités à approcher des fonctions non linéaires, les réseaux de neurones artificiels présentent une alternative prometteuse pour la représentation et la commande des systèmes non linéaires multivariables. Dans ce contexte, une commande neuronale adaptative indirecte est développée pour les systèmes non linéaires multivariables carrés [3; 73]. Le correcteur adaptatif utilisé dans ce schéma de commande est conçu à la base des réseaux de neurones récurrents. L'adaptation en ligne des paramètres de correcteur est la question clé de la commande adaptative. Ces schémas sont, souvent, fondés sur des algorithmes qui nécessitent une évaluation de la variation des sorties par rapport à la variation des entrées. Pour résoudre ce problème, un second réseau de neurones est utilisé comme émulateur de la dynamique du procédé non linéaire à commander. Les paramètres des réseaux de commande et d'émulation s'adaptent en utilisant un algorithme inspiré de l'algorithme Real Time Recurrent Learning (RTRL). L'avantage principal de cet algorithme est qu'il ne nécessite aucun modèle dynamique du procédé. Les résultats obtenus nous ont guidé pour poursuivre l'étude de ce schéma de commande dans les travaux de recherche proposés dans ce mémoire.

Dans ce travail, on poursuit l'étude et le développement du schéma de commande neuronale adaptative en discret. Malgré que les paramètres de l'émulateur neuronal et du correcteur s'adaptent à partir des conditions initiales nulles, on a montré que les performances de l'émulation et de la commande offertes en utilisant l'émulateur neuronal dépendent fortement d'un terme choisi arbitrairement pour assurer le démarrage de l'adaptation des paramètres de l'émulateur avec des conditions initiales nulles. Pour résoudre ce problème, une première contribution majeure est proposée et qui consiste à proposer un émulateur multimodèle pour la représentation et la commande des systèmes non linéaires MIMO et SIMO.
L'approche multimodèle connaît un regain d'intérêt ces dernières années. Elle doit sa popularité à la simplicité de la représentation des systèmes non linéaires obtenue par décomposition de l'espace de fonctionnement en plusieurs sous espaces permettant de décrire un système par plusieurs modèles locaux de structure simple. Le comportement global du système est ensuite représenté en considérant judicieusement la contribution relative de chaque sous-modèle au moyen d'une fonction de pondération associée à chaque zone de fonctionnement.
Deux structures multimodèles peuvent être distinguées dans la bibliographie. La première à états couplés connue aussi sous l'appellation de multimodèle de Takagi-Sugeno et la seconde à états découplés connue sous l'appellation de multimodèle découplé.
Vue la généralité et la flexibilité de la structure découplée du multimodèle, dûe au découplage des sous modèles, une première contribution consiste à proposer un émulateur multimodèle découplé pour l'émulation et la commande des systèmes non linéaires multivariables. Des améliorations sont apportées pour la synthèse des émulateurs multimodèles en proposant une méthode systématique pour le décomposition de l'espace de fonctionnement.

Une deuxième contribution majeure de notre travail consiste à proposer un schéma de commande neuronale adaptative utilisant des émulateurs neuronal et multimodèle de structures SIMO pour la commande des systèmes non linéaires SIMO. Ce schéma de commande neuronale adaptative de structure non carrée est développé en s'inspirant du schéma carré. En effet, le correcteur global s'obtient par une fusion d'un ensemble de correcteurs neuronaux de structures carrées. Cette tâche peut être assurée par l'utilisation des degrés de validité dont le calcul nécessite certainement quelques informations ; à savoir les sorties du procédé à commander et les sorties désirées.

Le mémoire est organisé comme suit :
Dans le premier chapitre, on introduit la commande neuronale adaptative comme un moyen performant pour la commande des systèmes non linéaires multivariables carrés. La première partie de ce chapitre présente une étude bibliographique de quelques approches

de commande des systèmes non linéaires multivariables. Dans cette partie on ne traite pas en profondeur chacune de ces techniques mais on expose rapidement leurs avantages et inconvénients. Pour résoudre les problèmes exposés, on introduit aussi la commande à base des Réseaux de Neurones Artificiels (RNAs). La structure du schéma de commande neuronale basé sur un émulateur neuronal est présentée dans une deuxième partie. Les structures et les algorithmes d'adaptation des paramètres du correcteur et de l'émulateur neuronaux sont développés dans cette partie. Afin de mettre en évidence l'apport en performances de ce schéma de commande, deux exemples de simulation des systèmes non linéaires MIMO sont présentés. A la fin de ce chapitre, on montre que l'utilisation d'un réseau de neurones pour l'émulation peut affecter les performances en boucle fermée.

Dans le deuxième chapitre, et pour résoudre les problèmes rencontrés lors de l'utilisation de l'émulateur neuronal, on propose un émulateur multimodèle pour la modélisation et la commande des systèmes non linéaires multivariables. Une première partie de ce chapitre aborde le problème soulevé par l'identification des systèmes non linéaires MIMO. Les différentes étapes nécessaires à l'élaboration de la structure (d'état ou polynômiale) d'un émulateur multimodèle découplé en se basant sur une technique itérative d'optimisation non linéaire sont présentées. Dans une deuxième partie, l'émulateur multimodèle obtenu est exploité dans le schéma de commande neuronale adaptative. Les modifications apportées par l'utilisation de l'émulateur proposé et qui sont nécessaires pour l'adaptation des paramètres du correcteur sont présentées. Des simulations sur des procédés non linéaires multivariables sont réalisées pour mettre en valeur les performances de l'utilisation de l'émulateur multimodèle en termes de modélisation et de commande en boucle fermée.
Dans la troisième partie, on propose une méthode basée sur la classification de données d'identification pour la décomposition de l'espace de fonctionnement et pour assurer une génération systématique des fonctions de pondération qui déterminent les degrés d'activation des sous-modèles constituant le multimodèle. Cette nouvelle méthode est proposée pour remplacer une méthode classique basée sur l'exploitation des caractéristiques statiques du système étudié pour la décomposition de l'espace de fonctionnement. Des simulations numériques sont présentés pour la mise en évidence de l'intérêt et de l'efficacité de la méthode développée.

Dans le troisième chapitre, un nouveau schéma de commande neuronale adaptative non carré est proposé pour la commande des systèmes non linéaires SIMO. Ce schéma consiste à considérer un ensemble de correcteurs neuronaux partiels carrés. La commande globale appliquée effectivement au système est obtenue par fusion des commandes partielles issues de chaque correcteur partiel. Ce schéma nécessite l'utilisation d'un émulateur pour représenter la dynamique du procédé étudié. Dans une première partie, un émulateur neuronal SIMO est proposé pour la représentation et la commande des systèmes non linéaires

SIMO. Les structures et les algorithmes d'adaptation des paramètres de cet émulateur et des correcteurs partiels sont présentés. Une nouvelle méthode pour l'estimation en ligne des validités des commandes partielles est aussi avancée. Des simulations numériques sont présentées pour mettre en évidence l'efficacité du schéma de commande proposé et pour souligner les problèmes qui peuvent apparaître lors de l'utilisation d'un émulateur neuronal.
Dans une deuxième partie, un émulateur multimodèle SIMO est utilisé dans le schéma de commande non carré pour remplacer l'émulateur neuronal. L'efficacité de cet émulateur pour la représentation et la commande des systèmes non linéaires SIMO est prouvé par deux simulations numériques.

Une conclusion et des perspectives achèvent le présent rapport.

Chapitre I

Commande neuronale adaptative pour les systèmes non linéaires multivariables carrés

Sommaire

I.1 Introduction

La présence de comportements non linéaires dans la dynamique des systèmes complexes rend les modèles linéarisés inefficaces pour la conception d'un contrôleur performant dans de nombreuses situations. Ainsi, la commande non linéaire multivariable a fait l'objet d'une recherche constante sur plusieurs décennies. Plusieurs approches de commande sont proposées dans la littérature, mais devant les problèmes rencontrés, les résultats sont restreints à certaines classes de systèmes et souffrent de fortes contraintes. Une alternative prometteuse pour la commande multivariable (MIMO : Multi Input multi Output) des systèmes dynamiques non linéaires est la mise en œuvre des réseaux de neurones artificiels.

Dans ce chapitre on commence par une description de quelques approches de commandes des systèmes non linéaires. Ensuite, on introduit une commande adaptative non linéaire basée sur un émulateur neuronal pour les systèmes multivariables carrés.

I.2 Sur la commande des systèmes non linéaires multivariables

Parmi les nombreuses techniques de modélisation et de commande des systèmes non linéaires multivariables, certaines ont fait l'objet de théories poussées mais néanmoins relativement complexes. Notre objectif dans ce paragraphe est de ne pas traiter en profondeur chacune d'elles mais de faire un exposé rapide sur ces techniques en présentant leurs avantages et leurs inconvénients. Une attention particulière est portée à la modélisation, l'émulation et à la commande neuronales.

I.2.1 Commandes non linéaires

Les commandes linéarisantes des systèmes non linéaires multivariables ont fait leur apparition dans les années 80 avec les travaux d'Isidori [27] et ont fait par la suite l'objet de plusieurs applications [14; 39; 57]. Ces techniques de linéarisation consistent à transformer, partiellement ou complètement, des systèmes non linéaires par une approximation linéaire autour d'un point de fonctionnement. Ces méthodes permettent une mise en œuvre simple et efficace des lois de commande linéaires mais elles restent limitées uniquement autour d'un point de fonctionnement et ne permettent pas la commande des systèmes non linéaires dans la totalité de leurs zones de fonctionnement.

Cette restriction a poussé les chercheurs à proposer d'autres techniques de commande telle que la commande par séquencement de gains "gain scheduling control" [33; 59]. Cette technique consiste à linéariser le système localement autour de plusieurs points de fonc-

tionnement et d'associer un contrôleur linéaire à chaque zone. Bien que cette méthode permette de commander le système dans la totalité de la zone de fonctionnement, des problèmes majeurs liés à la stabilité peuvent apparaître à cause des changements rapides des gains des correcteurs.

Plus récemment, des méthodes de conception de commande non linéaires avancées, y compris la commande robuste H_∞, la linéarisation par bouclage (Feedback linearization) [27; 51; 61] et l'inversion dynamique non linéaire [18; 25] ont été développées et appliquées avec succès aux problèmes de commande [2]. Ces méthodes avancées de conception de commandes non linéaires nécessitent souvent une connaissance explicite d'un modèle analytique du système dynamique, qui est souvent, imprécis ou indisponible.

I.2.2 Commandes par les méthodes de Lyapunov

La résolution des problèmes cités précédemment a poussé les chercheurs à développer des méthodes de commande permettant la synthèse des correcteurs non linéaires pour la commande des systèmes non linéaires multivariables tous en couvrant un domaine de fonctionnement plus étendu. Ces commandes sont basées sur les méthodes dites de Lyapunov. L'idée consiste à choisir une fonction de Lyaponuv qui dépend de l'erreur de commande et à déterminer l'expression de la commande appropriée qui garantie la décroissance asymptotique de cette fonction ce qui permet d'assurer la stabilité asymptotique du système en boucle fermée. Parmi de nombreuses techniques basées sur la méthode de Lyapunov on peut citer, à titre indicatif, les commandes par mode glissant et le "Backstepping" (classique ou adaptative).

La commande par mode glissant consiste, dans un premier temps, à forcer le système à atteindre une surface dite "*surface de glissement*" qui représente la dynamique désirée. Dans un second temps, elle doit assurer le maintien de la trajectoire de phase sur la surface et le glissement du point représentatif du mouvement le long de cette surface pour atteindre l'origine du plan de phase, point d'équilibre du système étudié. Cette commande est largement utilisée dans la littérature grâce à sa simplicité de mise en œuvre et sa robustesse vis à vis des variations paramétriques et des perturbations externes [17; 44; 45]. Cependant, elle présente un phénomène indésirable dit de réticence ou de broutement (Chattering en anglais) qui provoque des variations indésirables de la commande. Pour résoudre ce problème, plusieurs solutions sont apportées parmi lesquelles on peut citer : l'introduction des commandes avec des gains décroissants [76], la commande par mode glissant d'ordre supérieur [62] et la commande par mode glissant à bande limite [56].

Plus récemment la technique du "Backstepping" (commande stabilisante non linéaire) est apparue comme une alternative [23; 30; 31]. Il s'agit d'un algorithme récursif qui permet

un choix adéquat des fonctions de Lyapunov. L'idée fondamentale consiste à synthétiser une commande d'une manière recursive où certaines composantes du vecteur d'état sont considérées comme des "commandes virtuelles" et des lois de commande intermédiaires sont élaborées.
Bien que cet algorithme permet de garantir la stabilité du système, il présente une complexité qui rend son exploitation difficile en pratique de plus son utilisation est limité pour les systèmes triangulaires.

Malheureusement d'une manière générale et pour toutes les commandes basées sur les méthodes de Lyapunov, il n'existe pas des procédures générales permettant de trouver une fonction de Lyapunov candidate capable de prouver la stabilité de la boucle fermée pour n'importe quel système non linéaire.

I.2.3 Commandes floues

La logique floue a pris naissance en 1965, depuis les travaux de Lotfi Zadeh [72] et elle a suscité un intérêt croissant au cours de ces dernières décennies [13; 65]. En se basant sur l'expertise des opérateurs humains, la logique floue fournit un moyen simple et facile pour décomposer la tâche de modélisation et de commande en un groupe de tâches locales. Les différentes étapes de conception d'un contrôleur flou sont les suivantes : la fuzzification qui permet de passer de variables numériques à des variables symboliques, l'inférence qui grâce à la base de règles permet de relier les sous-ensembles flous d'entrée (mesures) aux sous-ensembles flous de sortie (commande) et la défuzzification qui permet la conversion des valeurs symboliques aux valeurs numériques.

Beaucoup de travaux portent sur cette stratégie de commande et fournissent différents types de contrôleurs qui sont classés en trois groupes : les systèmes flous linguistiques ou systèmes de Mamdani, les systèmes flous relationnels et les systèmes à conséquence fonctionnelle ou encore connus sous le nom de systèmes flous de type Takagi-Sugeno-Kang (TSK). Le lecteur désireux de mieux comprendre la logique floue peut consulter l'ouvrage de Titli et Foulloy [68] où ses principes sont rappelés et les nombreux domaines d'application sont détaillés (commande, mesure...).

Malgré le succès visible remporté par la logique floue dans le domaine de la commande des systèmes non linéaires complexes, de nombreuses questions fondamentales restent à résoudre. Parmi celles-ci on peut citer l'analyse de la stabilité et la conception systématique des correcteurs. En effet, un choix judicieux des conclusions et des règles des correcteurs flous n'est pas forcément triviale et les performances obtenues dépendent fortement de l'expertise.

I.2.4 Commandes par les Réseaux de Neurones Artificiels (RNAs)

La conception des modèles neuronaux a commencé depuis 1943 avec les travaux de McCulloch et Pitts. Ces deux chercheurs ont étudié les capacités d'interconnexion de plusieurs composants en se basant sur le modèle d'un neurone biologique. D'une manière générale, un RNA peut être défini comme un réseau complexe constitué d'unités de calcul élémentaires (les neurones formels) interconnectés. Chaque neurone reçoit des informations (qui peuvent être les entrées du modèle dans la cas de modélisation, les sorties désirées ou les erreurs de commande dans le cas de conception d'un contrôleur neuronal ou aussi les sorties d'autres neurones), les traite et envoie le résultat du traitement vers d'autres neurones.
Les RNAs peuvent être distingués selon les valeurs des entrées et des sorties (réelles ou binaires), selon les fonctions d'activation de leurs neurones (linéaire, non linéaire, polynomiale, exponentielle, gaussienne...) selon la procédure d'apprentissage (supervisé ou non supervisé) ou selon la structure connexionniste (réseaux directs et récurrents).
Les réseaux directs sont les premiers qui ont été utilisés en approximation des fonctions non linéaires. Ils sont, jusqu'à maintenant, les plus exploités grace à leur structure simple et à la facilité d'implémentation de leurs algorithmes d'apprentissage. Parmi ce type de RNAs on peut citer le Perceptron Multi-Couche (MPC) et le Réseau à Fonctions de Base Radiale (RBF : Radial Basis Function). Les réseaux récurrents présentent une structure plus performante et plus proche du modèle biologique. Parmi ces réseaux on peut citer le Réseau d'Elman Modifié (REM) et le réseau de Jordan [3].

Les réseaux de neurones sont caractérisés par leur capacité d'approcher les dynamiques non linéaires incertaines et de générer des lois de commande robustes et adaptatives. Après une période de stagnation, ces avantages ont attiré l'attention des automaticiens et ont poussé les chercheurs à exploiter les réseaux de neurones pour palier aux problèmes rencontrés dans le domaine de modélisation et de commande des systèmes non linéaires. En effet, au début des années 90, l'identification et la commande des systèmes non linéaires par des réseaux de neurones ont été proposées d'une manière systématique pour la première fois par Narendra et Parthasarathy [49]. Par la suite plusieurs autres chercheurs ont continué le travail et ont fait référence aux travaux effectués par Narendra dans le domaine de la commande neuronale. Récemment une commande neuronale adaptative est proposée [3; 5; 6; 74; 75]. L'efficacité de cette approche est prouvée pour la commande des systèmes non linéaires monovariables et multivariables carrées. L'avantage principale de cette stratégie est que la dynamique du système à commander n'est pas nécessairement connue. A partir des conditions initiales nulles, les paramètres du système de commande proposé s'adaptent de manière autonome, afin de capturer les dynamiques inconnues du procédé et de suivre la trajectoire désirée.

Dans ce qui suit on va développer ce schéma de commande tout en présentant ses avantages et ses inconvénients.

I.3 Commande adaptative neuronale basée sur un émulateur neuronal pour les systèmes non linéaires multivariables carrés

Dans cette partie on s'intéresse à une commande neuronale adaptative indirecte développée pour les systèmes non linéaires multivariables carrés [3; 5; 6; 74; 75]. Le point clé de ce schéma de commande est que les paramètres du contrôleur neuronal adaptatif varient au cours du temps, il est donc nécessaire d'ajuster ces paramètres en ligne. Cependant, l'adaptation des poids du contrôleur neuronal est fondée sur des algorithmes qui nécessitent une connaissance précise de la dynamique du système et qui a besoin d'évaluer la dérivée partielle d'une fonction coût par rapport aux paramètres du contrôleur. Le calcul de cette dérivée requière à son tour le calcul de la dérivée partielle de toutes les sorties du systèmes par rapport à ses entrées. Cependant, cette dernière ne peut pas être évaluée analytiquement à moins que la dynamique du procédé ne soit connue. Afin de résoudre ce problème un réseau de neurones est utilisé comme émulateur de la dynamique du système non linéaire à commander. Cet émulateur permet d'avoir une expression analytique de la dérivée partielle de toutes les sorties du système par rapport aux poids du contrôleur. Comme le montre la figure I.1, la structure de la commande neuronale adaptative contient deux réseaux de neurones récurrents totalement déconnectés l'un de l'autre : un réseau d'émulation neuronale et un réseau de commande. Tous les neurones des deux réseaux sont mis à jour simultanément en fonction de l'état des deux réseaux calculé à l'instant précédent (cette mise à jour est désignée par les traits pointillés).

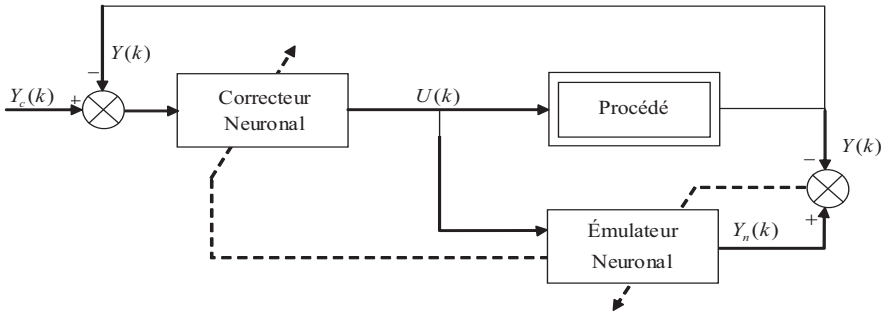


Figure I.1: Structure de la Commande Adaptative Neuronale basée sur un Emulateur Neuronal.

On désigne respectivement par N_{IN} et N_{OUT} le nombre d'entrées et de sorties du système à étudier, où IN est l'ensemble des indices des entrées du système à étudié et OUT est l'ensemble des indices de ses sorties. On considère des systèmes SISO et MIMO carrés, on note alors $N = N_{IN} = N_{OUT}$.

Les vecteurs $Y(k) = [y_1(k), ..., y_l(k), ..., y_N(k)]^T$ et $Y_n(k) = [y_{n_1}(k), ..., y_{n_l}(k), ..., y_{n_N}(k)]^T$ sont respectivement le vecteur des sorties réelles du système et celui des sorties de l'émulateur neuronal. Le vecteur $U(k) = [u_1(k), ..., u_l(k), ..., u_N(k)]^T$ est le vecteur des commandes issues du contrôleur neuronal et appliquées au système et $Y_c(k) = [y_{c_1}(k), ..., y_{c_l}(k), ..., y_{c_N}(k)]^T$ est le vecteur des sorties désirées.

Dans ce qui suit, les structures et les algorithmes d'adaptation de l'émulateur neuronal et du contrôleur neuronal sont présentés.

I.3.1 Emulateur Neuronal : Structure et algorithme d'adaptation

L'émulateur neuronal est constitué d'un nombre de neurones $N_e = N_{IN} + N_{OUT} = 2N$, ces neurones sont séparés en neurones d'entrée et neurones de sortie pour asssurer le découplage entre les entrées et les sorties du système. La figure I.2 montre que cet émulateur a une structure totalement connectée.

On attire l'attention du lecteur sur le fait qu'on ne s'interesse qu'au comportement instantané du réseau d'émulation nécessaire pour l'adaptation des paramètres du contrôleur neuronal, aucune mémorisation de la dynamique du système n'est nécessaire. Par conséquent, un nombre réduit de neurones permet l'émulation des systèmes complexes.

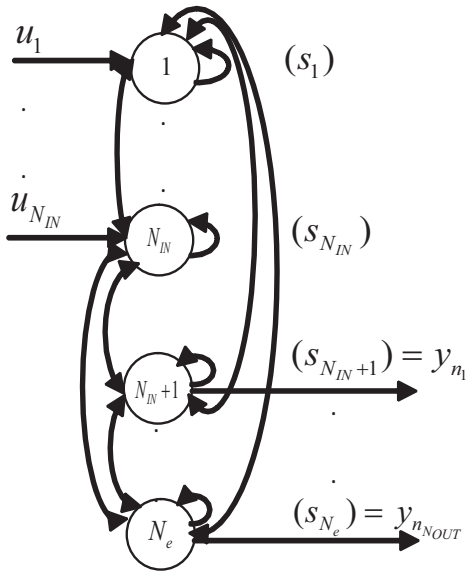


Figure I.2: Structure totalement connectée de l'émulateur neuronal ($N_e = 2N$).

L'activation de ces neurones évolue en temps discret selon l'équation suivante :

$$s_i(k+1) = e^{-|\tau_e(k)|\Delta T} s_i(k) + \left(1 - e^{-|\tau_e(k)|\Delta T}\right) \tanh\left(\sum_{j=1}^{N_e} w_{ij}(k) s_j(k) + x_i(k)\right) \tag{I.1}$$

où k, ΔT, $w_{ij}(k)$ et $1/|\tau_e(k)|$ représentent respectivement le temps discret, le pas de calcul, les poids de connexion du neurone j vers le neurone i et la constante de temps des neurones de l'émulateur. $s_i(k)$ est l'état du $i^{\text{ème}}$ neurone dont l'entrée est $x_i(k)$. Si $i \in \{1, ..., N\}$ $x_i(k) = u_i(k)$, si $i \in \{N+1, ..., 2N\}$ $x_i(k) = 0$ et $y_{n_{i-N}}(k) = s_i(k)$ avec y_{n_i} qui représente l'estimation de la $i^{\text{ème}}$ sortie du système.

On attire l'attention du lecteur sur le fait que pour qu'un réseau d'émulation neuronale puisse être utile pour un contrôleur neuronal adaptatif, il est essentiel qu'il puisse s'adapter en permanence à d'éventuelles évolutions de l'environnement du système à commander (incertitude, perturbation...). A cet effet, un algorithme inspiré de la méthode d'apprentissage récurrent temps réel (Real-Time Recurrent Learning algorithm : RTRL) est utilisé pour l'adaptation des paramètres de l'émulateur neuronal. L'avantage principal de cet algorithme est qu'il ne nécessite aucune connaissance de la dynamique du système à commander. A partir des conditions initiales nulles, les paramètres de l'émulateur s'adaptent de manière autonome pour capturer la dynamique instantanée du système. A chaque instant k, on considère l'erreur quadratique entre la sortie réelle du système et celle du réseau émulateur donné par l'équation suivante :

$$e_e(k) = \frac{1}{2} \sum_{l=1}^{N} \left(y_{n_l}(k) - y_l(k)\right)^2 \tag{I.2}$$

où $y_l(k)$ est la sortie réelle du système et y_{n_l} la sortie de l'émulateur neuronal. L'algorithme RTRL permet l'adaptation itérative des poids du réseau émulateur $w_{ij}(k)$ en utilisant le gradient de l'erreur quadratique d'émulation selon l'équation (I.3) :

$$\Delta w_{ij}(k) = -\left|\eta_e(k)\right| \Delta T \frac{\partial e_e(k-1)}{\partial w_{ij}} = -\left|\eta_e(k)\right| \Delta T \sum_{l=1}^{N} \left((y_{n_l}(k-1) - y_l(k-1)) \frac{\partial y_{n_l}(k-1)}{\partial w_{ij}} \right) \tag{I.3}$$

où $\eta_e(k)$ est le facteur d'adaptation de l'émulateur dont dépend la vitesse de convergence, la stabilité et la précision de l'algorithme. Ce paramètre indique l'importance relative de la correction à apporter aux poids. Il doit être adaptatif pour prendre en compte l'évolution dynamique du procédé.

Les dérivées partielles des sorties $y_{n_l}(k)$ du système commandé par rapport aux poids w_{ij}, $(i,j) \in \{1,...,N_e\}^2$ se calculent en utilisant des fonctions de sensibilité notées $P_{lij}(k) = \partial y_{n_l}(k)/\partial w_{ij}$, $l \in \{1,...,N\}$. Ces fonctions sont considérées comme des petites perturbations des sorties du neurone l, dûes à des petites variations ∂w_{ij}. Elles sont calculées à partir des dérivées partielles de l'équation (I.1), qui décrit le comportement dynamique des neurones constituant l'émulateur, par rapport aux poids et vérifient l'équation suivante, pour $d = 1,...,N_e$:

$$\begin{aligned} P_{dij}(k+1) &= e^{-|\tau_e(k)|\Delta T} P_{dij}(k) \\ &+ (1 - e^{-|\tau_e(k)|\Delta T}) \left(\tanh' \left(\sum_{m=1}^{N_e} w_{dm}(k) s_m(k) + x_d(k) \right) \right) \left(\delta_i^d s_j(k) + \sum_{m=1}^{N_e} w_{dm}(k) P_{mij}(k) \right) \end{aligned} \tag{I.4}$$

où δ_i^d est le symbole de Kronecker et $\tanh'$ est la dérivé de la fonction $\tanh(x)$ par rapport à x.

Puisque l'adaptation est effectuée itérativement, la correction des paramètres du réseau émulateur dépend fortement du choix du facteur d'adaptation et des valeurs initiales des différents paramètres. Pour surmonter la difficulté du choix de ces paramètres et obtenir un émulateur capable de représenter la dynamique de différents systèmes non linéaires avec un minimum de contraintes initiales, les paramètres $\eta_e(k)$ et $\tau_e(k)$ doivent s'adapter automatiquement. Un algorithme basé sur la minimisation de la fonction coût $e_e(k)$ et similaire à celui utilisé pour l'adaptation des poids est utilisé pour l'adaptation de ces paramètres en utilisant les équations suivantes :

$$\Delta \eta_e(k) = -\Delta T \frac{\partial e_e(k-1)}{\partial \eta_e} = -\Delta T \sum_{l=1}^{N} \left((y_{n_l}(k-1) - y_l(k-1)) \frac{\partial y_{n_l}(k-1)}{\partial \eta_e} \right) \tag{I.5}$$

$$\Delta \tau_e(k) = -\left|\eta_e(k)\right| \Delta T \frac{\partial e_e(k-1)}{\partial \tau_e} = -\left|\eta_e(k)\right| \Delta T \sum_{l=1}^{N} \left((y_{n_l}(k-1) - y_l(k-1)) \frac{\partial y_{n_l}(k-1)}{\partial \tau_e} \right) \tag{I.6}$$

Les fonctions de sensibilité $v_l^{\eta_e}(k) = \partial y_{n_l}(k)/\partial \eta_e$ et $v_l^{\tau_e}(k) = \partial y_{n_l}(k)/\partial \tau_e$ sont considérées comme des petites perturbations des sorties y_{n_l} des neurones d'ordre l, dûes respectivement à des petites variations de $\eta_e(k)$ et $\tau_e(k)$. Suivant cette approximation et pour des raisons de simplification, ces fonctions évoluent de la même manière à partir des conditions initiales nulles, on considère alors $v_l(k) = v_l^{\eta_e}(k) = v_l^{\tau_e}(k)$. La variation de ces fonctions est donnée par l'équation suivante, pour $d = 1, ..., N_e$:

$$v_d(k+1) = e^{-|\tau_e(k)|\Delta T} v_d(k)$$

$$+(1 - e^{-|\tau_e(k)|\Delta T}) \left(\tanh' \left(\sum_{m=1}^{N_e} w_{dm}(k) s_m(k) + x_d(k) \right) \right) \left(\sum_{m=1}^{N_e} (w_{dm}(k) v_m(k)) \right) + \frac{\varepsilon_e}{|\tau_e(k)|} \tag{I.7}$$

La constante ε_e est choisie arbitrairement. Cette constante est introduite pour assurer le démarrage et l'évolution autonome de l'algorithme à partir des conditions initiales nulles pour tous les paramètres du réseau neuronal. Par conséquent, les performances de l'émulation neuronale ne dépendent pas des valeurs initiales des paramètres, mais dépendent de la valeur de ε_e.

I.3.2 Correcteur neuronal adaptatif : Structure et algorithme d'adaptation

Deux types de structures sont proposées et étudiées pour le correcteur neuronal : une structure réduite constituée d'un nombre de neurones $N_c = N$ neurones et une structure complète constituée de $N_c = 2N$ neurones [3; 73]. Il a été montré que la structure complète est plus efficace pour la commande des systèmes non linéaires. En effet, elle assure une adaptation plus performante et permet une convergence rapide des sorties réelles vers les sorties désirées [3]. Pour ces raisons, on considère la structure complète dans ce travail. La structure du réseau neuronal de commande est similaire à celle choisie pour l'émulation. Le réseau neuronal est donc formé de $N_c = 2N$ neurones totalement connectés (figure I.3). Les N premières entrées du correcteur sont les N erreurs $(y_{c_l}(k) - y_l(k))$ entre les consignes $y_{c_l}(k)$ et les sorties mesurées $y_l(k)$ et les N autres entrées sont les N consignes $y_{c_l}(k)$. Les commandes appliquées au système et à l'émulateur sont les N premières sorties du contrôleur.

L'activation des neurones du correcteur évolue en temps discret selon les équations suivantes, $i = 1, ..., N_c$:

$$o_i(k+1) = e^{-|\tau_c(k)|\Delta T} o_i(k) + (1 - e^{-|\tau_c(k)|\Delta T}) \tanh \left(\sum_{j=1}^{N_c} \phi_{ij}(k) o_j(k) + z_i(k) \right) \tag{I.8}$$

$o_i(k)$ est l'état du $i^{\text{ème}}$ neurone, si $i \in 1, ..., N$ $o_i(k) = u_i(k)$ représente le $i^{\text{ème}}$ signal de commande, $1/\tau_c(k)$ et $\phi_{ij}(k)$ sont respectivement la constante de temps du correcteur et

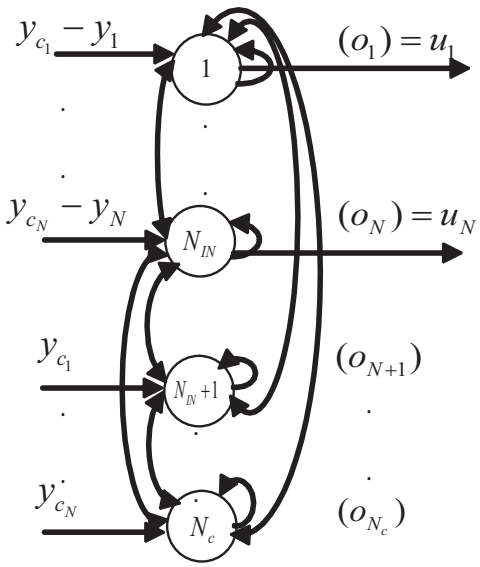


Figure I.3: Structure complète et totalement connectée du correcteur neuronal ($N_c = 2N$).

les poids de connexion du neurone j vers le neurone i. $z_i(k) = y_{c_i}(k) - y_i(k)$ si $i \in \{1, ..., N\}$ et $z_i(k) = y_{c_i}(k)$ si $i \in \{N+1, ..., 2N\}$.
L'adaptation des poids du correcteur est basée sur une procédure d'optimisation qui consiste à minimiser l'erreur quadratique instantannée de commande définie par la relation suivante :

$$e_c(k) = \frac{1}{2} \sum_{l=1}^{N} (y_{c_l}(k) - y_l(k))^2 \tag{I.9}$$

Un algorithme d'adaptation, analogue à celui utilisé pour l'adaptation des poids du réseau émulateur, inspiré de la descente du gradient est utilisé pour la mise à jour des poids de réseau de commande :

$$\Delta\phi_{ij}(k) = -\left|\eta_c(k)\right| \Delta T \frac{\partial e_c(k-1)}{\partial \phi_{ij}} \tag{I.10}$$

où $\eta_c(k)$ est le facteur d'adaptation du correcteur. En remplaçant $e_c(k)$ par son expression, l'équation (I.10) devient :

$$\Delta\phi_{ij}(k) = \left|\eta_c(k)\right| \Delta T \sum_{l=1}^{N} \left((y_{c_l}(k-1) - y_l(k-1)) \frac{\partial y_l(k-1)}{\partial \phi_{ij}} \right) \tag{I.11}$$

Vu qu'il n'y a aucune expression analytique reliant les poids du correcteur et les sorties du système, un émulateur neuronal est donc nécessaire pour représenter la dynamique non linéaire du système et approcher la relation $\frac{\partial y_l(k-1)}{\partial \phi_{ij}}$ par $\frac{\partial y_{n_l}(k-1)}{\partial \phi_{ij}}$.
Sachant que les commandes dépendent des poids du réseau de commande et les sorties de l'émulateur neuronal dépendent des commandes, la dérivée partielle $\frac{\partial y_{n_l}(k)}{\partial \phi_{ij}}$ s'exprime

sous la forme suivante :

$$\frac{\partial y_{n_l}(k)}{\partial \phi_{ij}} = \sum_{d=1}^{N_c} \left(\frac{\partial y_{n_l}(k)}{\partial o_d} \frac{\partial o_d(k)}{\partial \phi_{ij}} \right) = \sum_{d=1}^{N_c} \left(J_{ld}(k) Q_{dij}(k) \right) \tag{I.12}$$

où $J_{ld}(k)$ et $Q_{dij}(k)$ sont des fonctions de sensibilité du réseau neuronal.
Les fonctions de sensibilité $Q_{dij}(k)$ sont calculées à partir de la dérivée partielle de l'équation (I.8), qui représente le comportement dynamique des neurones du correcteur, par rapport aux poids $\phi_{ij}(k)$ en utilisant l'équation suivante, $d = 1, ..., N_c$:

$$Q_{dij}(k+1) = e^{-|\tau_c(k)|\Delta T} Q_{dij}(k) + (1 - e^{-|\tau_c(k)|\Delta T}) \varphi_d(k) \psi_d(k) \tag{I.13}$$

Pour $d \in \{1, ..., N\}$, $z_d(k) = y_{c_d}(k) - y_d(k)$ et les relations $\varphi_d(k)$ et $\psi_d(k)$ sont données par :

$$\begin{aligned} \varphi_d(k) &= \tanh' \left(\sum_{h=1}^{N_c} \phi_{dh}(k) o_h(k) + (y_{c_d}(k) - y_d(k)) \right) \\ \psi_d(k) &= \left(\delta_i^d o_j(k) + \sum_{h=1}^{N_c} \phi_{dh}(k) Q_{hij}(k) - \sum_{h=1}^{N_c} J_{dh}(k) Q_{hij}(k) \right) \end{aligned} \tag{I.14}$$

Pour $d \in \{N+1, ..., 2N\}$, $z_d(k) = y_{c_d}(k)$. Ces fonctions sont alors données par :

$$\begin{aligned} \varphi_d(k) &= \tanh' \left(\sum_{h=1}^{N_c} \phi_{dh}(k) o_h(k) + y_{c_d}(k) \right) \\ \psi_d(k) &= \left(\delta_i^d o_j(k) + \sum_{h=1}^{N_c} \phi_{dh}(k) Q_{hij}(k) \right) \end{aligned} \tag{I.15}$$

A partir de la dérivée partielle par rapport à $o_d(k)$, $d \in 1, ..., N_c$, de l'équation (I.1), qui représente le comportement dynamique du réseau émulateur, on obtient les fonctions de sensibilité $J_{ld}(k)$ qui sont calculées par l'équation suivante pour $l = 1, ..., N_e$:

$$\begin{aligned} J_{ld}(k+1) &= e^{-|\tau_e(k)|\Delta T} J_{ld}(k) \\ &+ \left(1 - e^{-|\tau_e(k)|\Delta T}\right) \left(\tanh' \left(\sum_{h=1}^{N_e} w_{lh}(k) s_h(k) + x_l(k) \right) \right) \left(\delta_d^l + \sum_{h=1}^{N_e} w_{lh}(k) J_{hd}(k) \right) \end{aligned} \tag{I.16}$$

En utilisant un algorithme analogue à celui utilisé pour l'adaptation des paramètres $\eta_e(k)$ et $\tau_e(k)$ de l'émulateur neuronal les paramètres du correcteur $\eta_c(k)$ et $\tau_c(k)$ sont adaptés en minimisant la fonction coût $e_c(k)$. En utilisant l'équation I.9, l'adaptation de ces paramètres est formulée par :

$$\Delta \eta_c(k) = -\Delta T \frac{\partial e_c(k-1)}{\partial \eta_c} = \Delta T \sum_{l=1}^{N} \left((y_{c_l}(k-1) - y_l(k-1)) \frac{\partial y_l(k-1)}{\partial \eta_c} \right) \tag{I.17}$$

$$\Delta \tau_c(k) = -\left|\eta_c(k)\right| \Delta T \frac{\partial e_c(k-1)}{\partial \tau_c} = \left|\eta_c(k)\right| \Delta T \sum_{l=1}^{N} \left((y_{c_l}(k-1) - y_l(k-1)) \frac{\partial y_l(k-1)}{\partial \tau_c} \right) \tag{I.18}$$

Pour approcher respectivement les dérivées $\frac{\partial y_l(k-1)}{\partial \eta_c}$ et $\frac{\partial y_l(k-1)}{\partial \tau_c}$ et par analogie aux fonctions de sensibilité $v_l^{\eta_e}(k)$ et $v_l^{\tau_e}(k)$, on définit les fonctions de sensibilité $r_l^{\eta_c}(k)$ et $r_l^{\tau_c}(k)$:

$$r_l^{\eta_c}(k) = \frac{\partial y_{n_l}(k)}{\partial \eta_c} \tag{I.19}$$

$$r_l^{\tau_c}(k) = \frac{\partial y_{n_l}(k)}{\partial \tau_c} \tag{I.20}$$

Ces fonctions sont considérées comme des petites perturbations des sorties $y_{n_l}(k)$ dûes respectivement à des petites variations de η_c et τ_c. Pour des raisons de simplification, on considère $r_l^{\eta_c}(k) = r_l^{\tau_c}(k) = r_l(k)$. Les fonctions $r_d(k)$ sont alors calculées, en utilisant la dynamique de l'émulateur neuronal décrite par l'équation I.1, comme suit, pour $d = 1, ..., N_e$:

$$\begin{aligned} r_d(k+1) &= e^{-|\tau_e(k)|\Delta T} r_d(k) \\ &+ (1 - e^{-|\tau_e(k)|\Delta T}) \left(\tanh' \left(\sum_{m=1}^{N_e} w_{dm}(k) s_m(k) + x_d(k) \right) \right) \left(\sum_{m=1}^{N_e} w_{dm}(k) r_m(k) \right) + \frac{\varepsilon_c}{|\tau_e(k)|} \end{aligned} \tag{I.21}$$

La constante ε_c est choisie pour assurer le démarrage et l'évolution de l'algorithme d'adaptation à partir des conditions initiales nulles.

I.4 Simulations numériques

Pour illustrer les performances et montrer les limites des stratégies d'émulation et de commande présentées, deux exemples de simulations pour deux systèmes non linéaires sont présentés.

Afin d'évaluer la qualité d'émulation de chaque sortie du système, deux indices de performance peuvent être définis. Le premier indice proposé est la moyenne quadratique de l'erreur (Mean Square Error MSE_l) donnée par :

$$MSE_l = \frac{1}{N_H} \sum_{k=1}^{N_H} \left(y_{n_l}(k) - y_l(k) \right)^2 \tag{I.22}$$

Le deuxième indice proposé est la VAF_l (variance-accounted-for) définie dans [10]. Cet indice sert à comparer la séquence de la sortie réelle du système à la séquence de la sortie de l'émulateur. Pour chaque séquence de sortie $y_l(k), k = 1, ..., N_H$, il est défini par :

$$VAF_l = \max \left\{ 1 - \frac{\text{var}\left\{ y_{n_l}(k) - y_l(k) : \ k = 1...N_H \right\}}{\text{var}\left\{ y_{n_l}(k) : \ k = 1...N_H \right\}}, 0 \right\} \times 100\% \tag{I.23}$$

I.4.1 Emulation et commande neuronales adaptatives d'un système non linéaire SISO

On considère, dans cet exemple, un système non linéaire SISO décrit par l'équation discrète suivante [8] :

$$y(k) = 0.4u(k-1) + 0.12y(k-1) + \frac{0.4u(k-1)y(k-2)(u(k-1)+2.5)}{1+u(k-1)^2+y(k-2)^2} \tag{I.24}$$

On commence par évaluer l'émulation neuronale de ce système, on considère alors une émulation en boucle ouverte. La structure et la dynamique du système sont complètement inconnues pour le processus d'adaptation neuronale. Selon la dynamique du système, le pas de calcul retenu est $\Delta T = 0.1s$. Le terme ε_e est choisi arbitrairement pour assurer le démarrage et l'évolution de l'algorithme d'émulation à partir des conditions initiales nulles pour tous les paramètres. La séquence d'entrée utilisée pour l'émulation de ce système est donnée par :

$$\begin{cases} u(k) = 0.8\, sin(\frac{2\pi}{250}k) & 1 \leq k \leq 775 \\ \\ u(k) = 0.7\, sin(\frac{2\pi}{250}k) + 0.1\, sin(\frac{2\pi}{25}k) & k > 775 \end{cases} \tag{I.25}$$

Un bon choix de ε_e conduit à de bonnes performances en émulation. En effet, pour $\varepsilon_e = 100$ les résultats d'émulation neuronale sont donnés par la figure I.4. L'émulateur neuronal fournit, dans ce cas, une estimation satisfaisante de la sortie du processus. L'erreur d'émulation tracée à l'échelle logarithmique (figure I.5) montre une bonne qualité d'émulation.

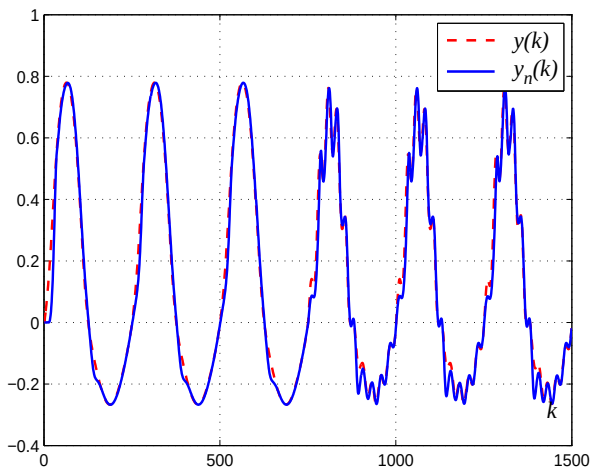


Figure I.4: Variations de la sortie réelle et de l'émulateur neuronal ($\varepsilon_e = 100$).

En calculant les indices définis par les équations (I.22) et (I.23) pour cette valeur de ε_e on a enregistré un $MSE = 1.510^{-3}$ et un $VAF = 98.87\%$. Il est à noter que cette valeur

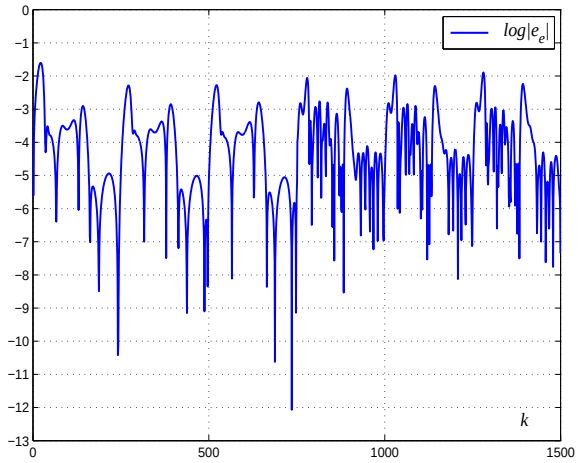


Figure I.5: Erreur d'émulation à l'échelle logarithmique ($\varepsilon_e = 100$).

de ε_e est trouvée suite à plusieurs essais afin d'assurer une erreur d'émulation minimale. En effet, un choix arbitraire de ce paramètre peut conduire à des mauvaises performances en émulation. Si on prend par exemple $\varepsilon_e = 1$, on obtient les résultats illustrés sur les figures I.6 et I.7. Les réponses du système et de l'émulateur, représentées sur la figure I.6, montrent que l'émulateur neuronal s'adapte à la variation de la sortie du système avec une erreur d'émulation relativement importante. L'erreur d'émulation, tracée à l'échelle logarithmique sur la figure I.7, permet d'évaluer une qualité peu satisfaisante d'émulation.

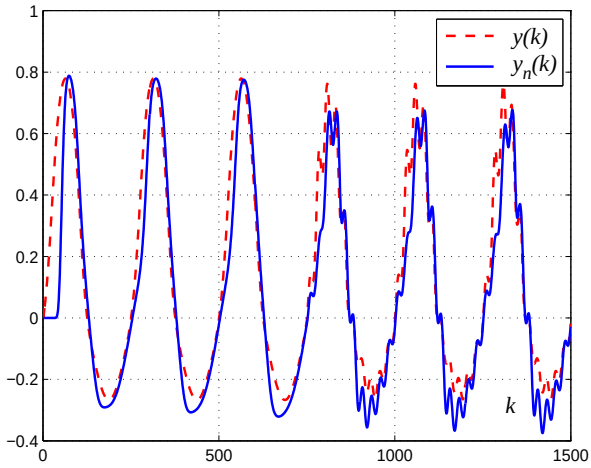


Figure I.6: Variations de la sortie réelle et de l'émulateur neuronal ($\varepsilon_e = 1$).

L'erreur MSE enregistrée avec cette valeur de ε_e est égale à 1.510^{-2} et l'indice VAF est égal à 89.95%. Ces simulations montrent clairement que la qualité d'émulation neuronale est fortement liée au choix du paramètre de démarrage de l'émulateur. Il est bien

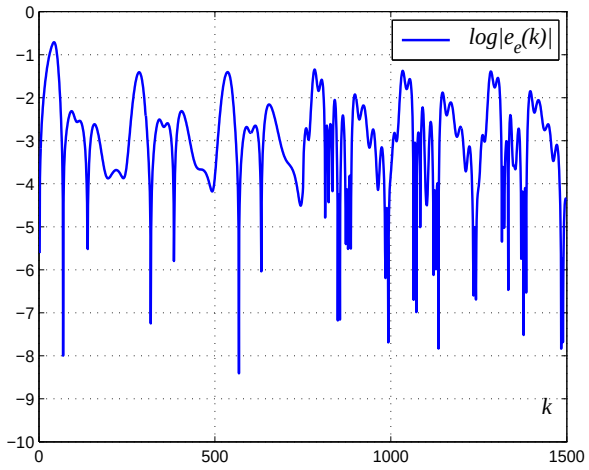


Figure I.7: Erreur d'émulation à l'échelle logarithmique ($\varepsilon_e = 1$).

évident que cette dépendance peut affecter les performances de la commande neuronale basée sur un émulateur neuronal.

L'émulateur neuronal est ensuite exploité dans le schéma de commande neuronale adaptative. Un premier choix des termes de démarrage du correcteur neuronal et de l'émulateur neuronal est considéré tels que $\varepsilon_c = 135$ et $\varepsilon_e = 130$, sous la condition $\varepsilon_e < \varepsilon_c$ qui assure la rapidité de l'adaptation du correcteur [3]. Les simulations proposées considèrent une phase de poursuite pour $k \in [1, 2500]$ et également une phase de régulation pour $k \in [2500, 5000]$. Au cours de la phase de régulation et pour $k = 3000$ (pendant 500 itérations) une perturbation de type échelon d'amplitude 10% de la valeur d'entrée du système de commande, affecte la sortie du système. Les résultats de simulation sont donnés par les figures I.8 et I.9. Ces figures montrent que l'émulateur neuronal permet, dans ce cas, au correcteur neuronal d'assurer de bonnes performances en termes de poursuite et de régulation.

Comme on l'a déjà mentionné, le choix de ε_e n'est pas systématique. Par conséquent, un choix non judicieux peut conduire à des très mauvaises performances en boucle fermée. Par exemple, pour $\varepsilon_c = 135$ et $\varepsilon_e = 100$, on obtient les résultats enregistrés sur les figures I.10 et I.11. Les variations de la sortie réelle et de la sortie désirée (figure I.10) montrent des oscillations importantes au démarrage. Dans la phase de régulation, les perturbations affectant l'observation ont conduit à une forte variance de la sortie du système. L'erreur $e_c(k)$ entre les sorties désirée et réelle, mesurée à l'échelle logarithmique et tracée sur la figure I.11, est jugé relativement importante.

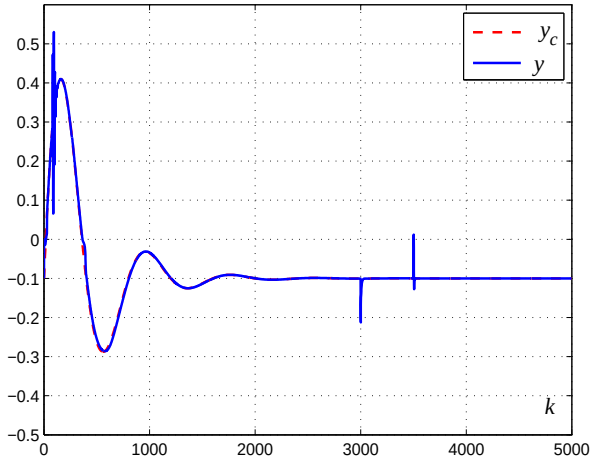


Figure I.8: Variations de la sortie réelle et désirée ($\varepsilon_e = 130$, $\varepsilon_c = 135$).

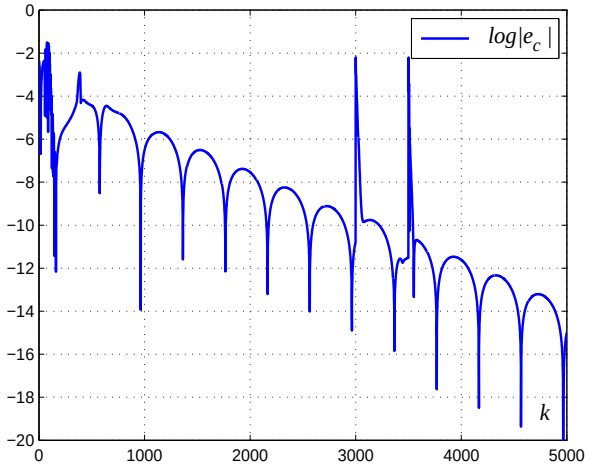


Figure I.9: Erreur de commande à l'échelle logarithmique ($\varepsilon_e = 130$, $\varepsilon_c = 135$).

I.4.2 Emulation et commande neuronales adaptatives d'un système non linéaire MIMO carré

Dans ce paragraphe, un exemple de simulation numérique est présenté pour vérifier l'efficacité de l'émulateur neuronal pour les systèmes non linéaires MIMO carrés [10]. On considère le système non linéaire à deux entrées et deux sorties défini par l'équation

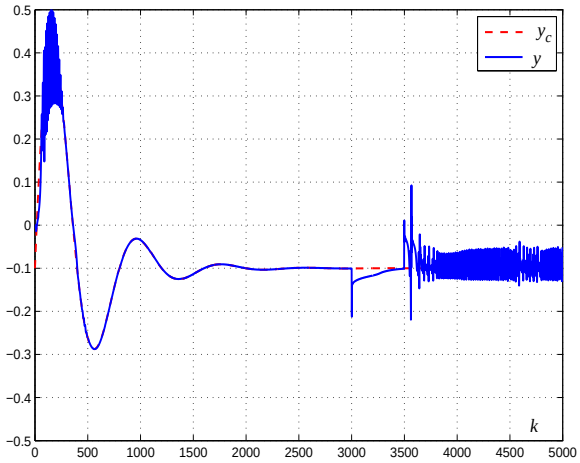


Figure I.10: Variations de la sortie réelle et désirée ($\varepsilon_e = 100$, $\varepsilon_c = 135$).

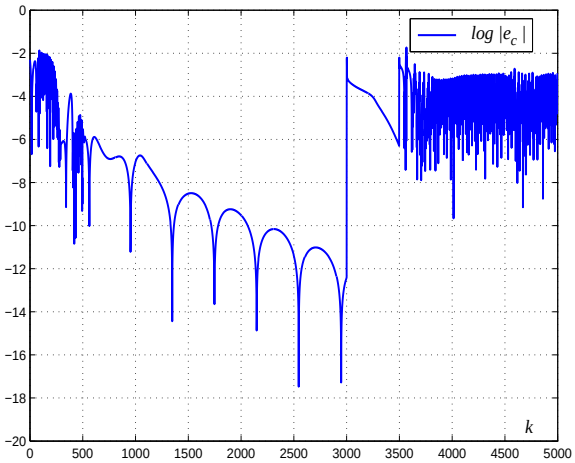


Figure I.11: Erreur de commande à l'échelle logarithmique ($\varepsilon_e = 100$, $\varepsilon_c = 135$).

suivante :

$$\begin{aligned} y_1(k) &= \frac{1}{2}((1.1 - 0.1z_{11}(k-1) - z_{21}(k-1))y_1(k-1) \\ &\quad + z_{11}(k-1)u_1(k-1) + z_{21}(k-1)u_2(k-1)) \\ y_2(k) &= \frac{1}{2}((1.5 - z_{12}(k-1) - 0.2z_{22}(k-1))y_2(k-1) \\ &\quad + z_{12}(k-1)u_1(k-1) + z_{22}(k-1)u_2(k-1)) \end{aligned} \tag{I.26}$$

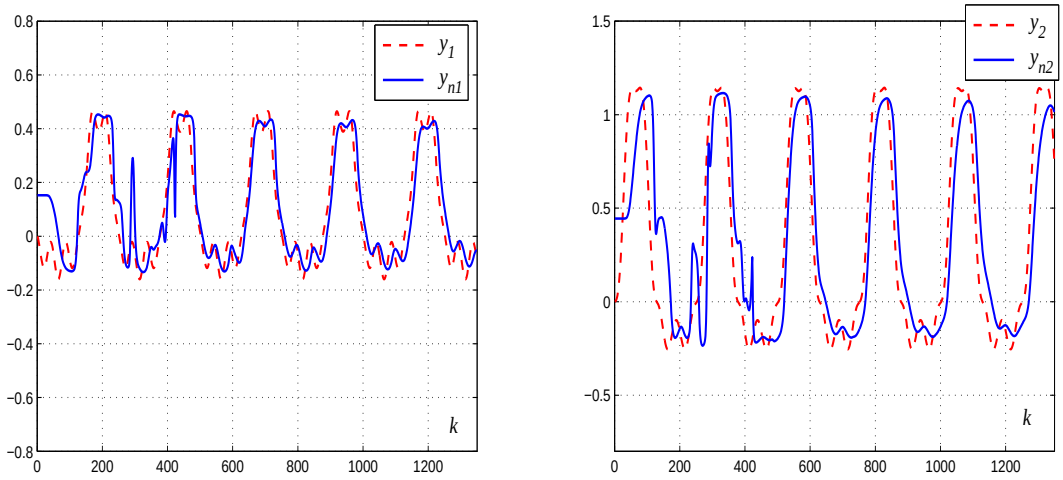
avec :

$$z_{11}(k-1) = \frac{0.4 - 0.6y_1(k-1)}{1 + 0.2y_1(k-1)}\ ;\ z_{21}(k-1) = \frac{0.6u_2(k-1)}{1 + 0.2y_1(k-1)}$$

$$z_{12}(k-1) = \frac{0.5u_1(k-1)}{1 + 0.5y_2(k-1)}\ ;\quad z_{22}(k-1) = \frac{0.7 - 0.07y_2(k-1)}{1 + 0.1y_2(k-1)}$$

Un émulateur neuronal à quatre neurones est exploité pour la représentation de ce système non linéaire. Ce nombre de neurones ne dépend que du nombre d'entrées et du nombre de sorties ($N_e = 2N = 4$). En considérant la dynamique du système, le pas de calcul est pris égal à $0.1s$.La séquence des entrées de commande et les sorties correspondantes prennent des valeurs normalisées dans l'intervalle $[-1, 1]$.

Les simulations montrent encore que la qualité d'émulation neuronale est fortement liée au choix du paramètre de démarrage ε_e. En effet, un choix judicieux de $\varepsilon_e = 0.4$ conduit à une estimation précise des sorties. Les résultats d'émulation donnés sur la figure I.12 laissent apparaitre une représentation relativement précise de la dynamique du système.



(a) Première sortie du système

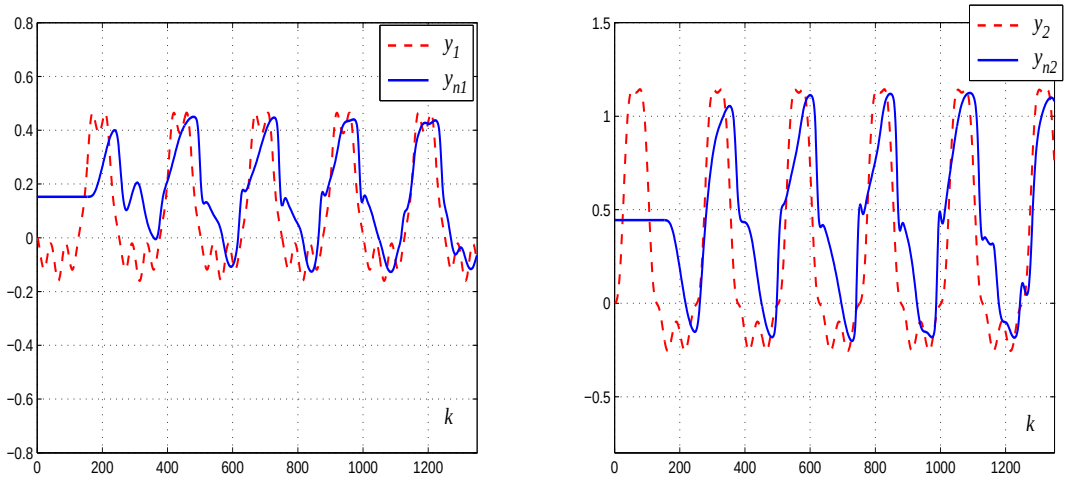
(b) Deuxième sortie du système

Figure I.12: Variations des sorties réelles et celles de l'émulateur neuronal ($\varepsilon_e = 0.4$).

On note, par ailleurs, qu'un choix arbitraire de ε_e ($\varepsilon_e = 0.02$ par exemple) a pu dégrader considérablement les performances de l'émulation neuronale en terme de précision (voir figure I.13).

Pour confirmer encore une fois la dépendance des performances d'émulation du paramètre de démarrage ε_e, les deux indices définis par les équations (I.22) et (I.23) sont calculés pour un choix arbitraire et un choix judicieux de ce paramètre. Ils sont résumés dans le tableau I.1.

Pour illustrer les performances en boucle fermée de la commande neuronale adaptative basée sur un émulateur neuronal pour les systèmes multivariables, on considère le système non linéaire MIMO formulé par la relation (I.26). Les deux objectifs de poursuite et de régulation sont considérés par des simulations numériques.
Une première simulation considère, une phase de poursuite pour $k \in [1, 2500]$ et une phase de régulation pour $k \in [2500, 4500]$. Pendant la phase de régulation, des perturbation de

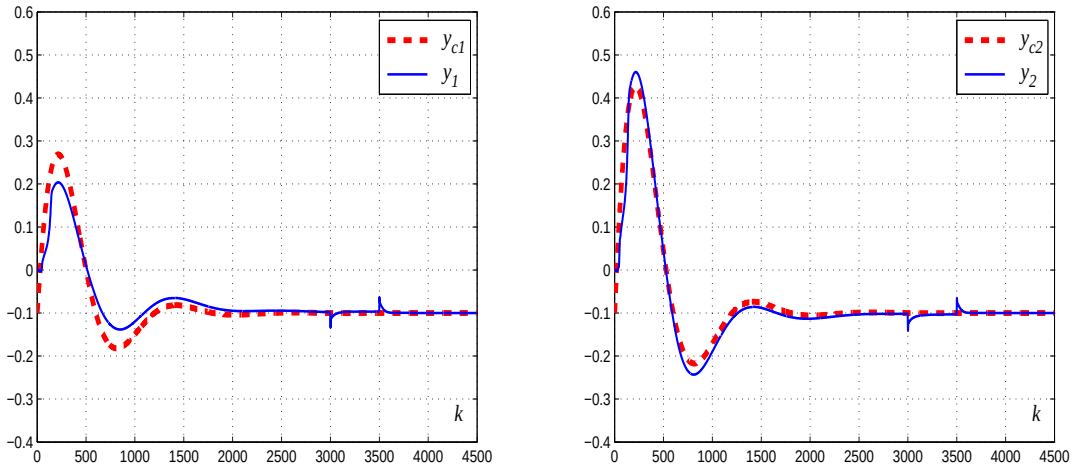


(a) Première sortie du système

(b) Deuxième sortie du système

Figure I.13: Variations des sorties réelles et celles de l'émulateur neuronal ($\varepsilon_e = 0.02$).

type échelon d'amplitude 10% de la valeur d'entrée du système de commande, affectent les sorties du système durant l'intervalle $k \in [3000, 3500]$. En utilisant l'émulateur neuronal avec la valeur $\varepsilon_e = 2$ et un terme de démarrage ε_c choisi arbitrairement ($\varepsilon_c = 4$ par exemple), on obtient les résultats donnés par la figures I.14. Ces résultats montrent que



(a) Première sortie du système.

(b) Deuxième sortie du système.

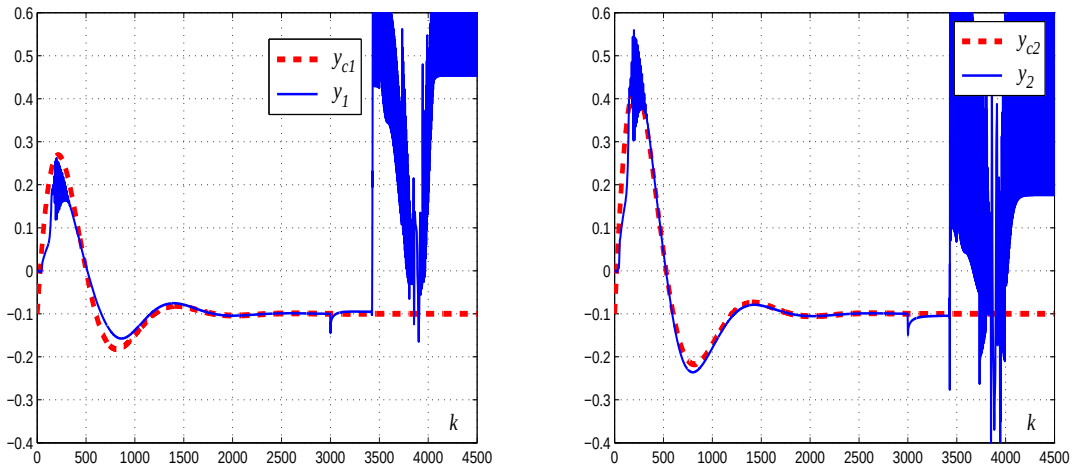
Figure I.14: Commande neuronale adaptative basée sur un émulateur neuronal ($\varepsilon_e = 2$, $\varepsilon_c = 4$) : variation des sorties réelles et désirées.

l'émulateur neuronal permet d'offrir de bonnes performances en boucle fermée en termes de poursuite et de rejet de perturbations.

	Emulateur Neuronal $\varepsilon_e = 0.02$	Emulateur Neuronal $\varepsilon_e = 0.4$
MSE_1	$2.51\,10^{-2}$	$8.4\,10^{-3}$
MSE_2	$1.32\,10^{-1}$	$7.04\,10^{-2}$
VAF_1	32.2%	80.74%
VAF_2	29.25%	70.67%

Tableau I.1: Les indices MSE_l et VAF_l $(l = 1, 2)$ calculés pour les deux sorties du système, pour un choix arbitraire et un choix judicieux de ε_e lors d'une émulation en boucle ouverte.

Pour montrer l'influence du choix du paramètre de démarrage de l'émulateur neuronal sur les performances en boucle fermée, on considère maintenant une autre valeur de $\varepsilon_e = 0.4$ tout en gardant la même valeur de $\varepsilon_c = 4$. Les résultats obtenus sont illustrés sur la figure I.15. Pour les deux sorties du système, des oscillations importantes sont enregistrées



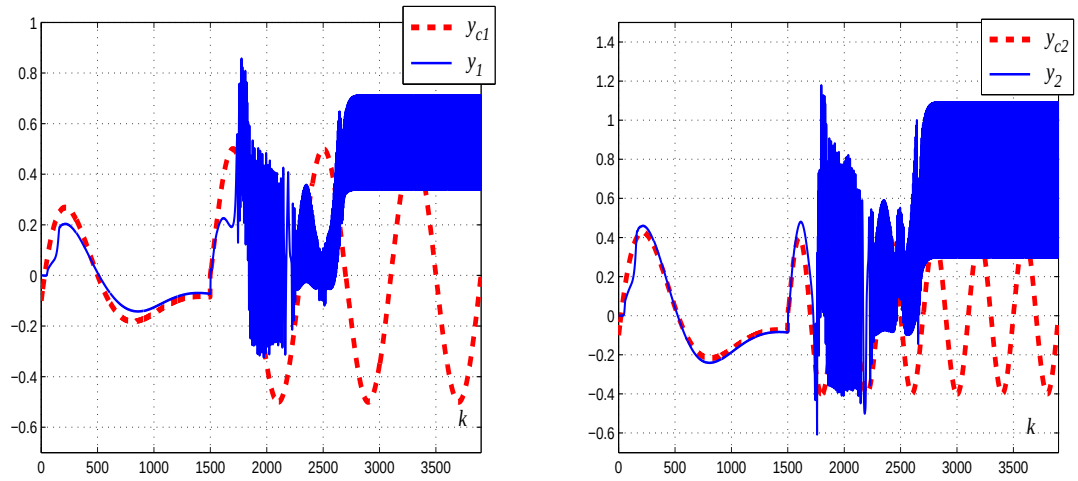
(a) Première sortie du système

(b) Deuxième sortie du système

Figure I.15: Commande neuronale adaptative basée sur un émulateur neuronal ($\varepsilon_e = 0.4$, $\varepsilon_c = 4$) : variation des sorties réelles et désirées.

au démarrage de l'algorithme d'adaptation ce qui affecte la qualité de poursuite. Cette figure montre, par ailleurs, que dans la phase de régulation, les perturbations affectant le système commandé et ont conduit à une variance importantes des sorties.

On considère, maintenant, une seconde simulation où seulement une phase de poursuite est considérée. Dans ce cas, les sorties désirées présentent une dynamique relativement rapide à partir de l'instant $k = 1500$. Les paramètres de démarrage sont choisis dans ces conditions égaux à $\varepsilon_e = 0.4$ et $\varepsilon_c = 2$. Les variations des consignes et des sorties réelles, dans ce cas, sont données sur la figure I.16. Cette figure renseigne sur des très mauvaises performances en poursuite. En effet, dans ce cas l'émulateur neuronal ne peut pas s'adapter rapidement au changement de la dynamique des sorties.



(a) Première sortie du système

(b) Deuxième sortie du système

Figure I.16: Commande neuronale adaptative basée sur un émulateur neuronal ($\varepsilon_e = 0.4$, $\varepsilon_c = 2$) : variation des sorties réelles et désirées (cas d'une dynamique rapide des sorties désirées).

Les résultats de simulation obtenus pour des systèmes non linéaires SISO ou MIMO illustrent clairement que l'utilisation d'un émulateur neuronal peut ne pas conduire, dans tous les cas, à des bonnes performances. On note également que l'émulation neuronal nécessite une adaptation en ligne ce qui impose une charge calculatoire relativement importante. Dans ce contexte, une solution peut être avancée en proposant un émulateur multimodèle pour l'émulation et la commande des systèmes non linéaires multivariables.

I.5 Conclusion

Dans ce chapitre, une stratégie de commande neuronale adaptative pour les systèmes non linéaires multivariables carrés est présentée. Le schéma développé est composé de deux réseaux de neurones récurrents : un émulateur et un correcteur. L'émulateur neuronal est utilisé pour émuler la dynamique non linéaire du système à commander et permet de calculer les dérivées des sorties par rapport aux commandes qui sont nécessaires pour

l'adaptation des paramètres du correcteur.

Les paramètres des deux réseaux, pris initialement tous nuls, sont adaptés par l'algorithme RTRL. Pour assurer cette adaptation à partir des valeurs initiales nulles, deux paramètres appelés paramètres de démarrage (ε_e pour l'émulateur et ε_c pour le contrôleur) sont initialisés avec des valeurs arbitrairement choisies non nulles. Bien que ce schéma ait montré sa capacité à émuler et commander plusieurs systèmes non linéaires SISO et MIMO carrés et ait été appliqué avec succès en temps réel à des procédés chimiques et thermiques [6; 4; 75], on a montré que l'utilisation d'un émulateur neuronal peut affecter les performances de la commande en boucle fermée. En effet, les simulations numériques montrent que les performances en poursuite et en régulation dépendent fortement du choix du paramètre de démarrage ε_e. De plus l'utilisation d'un émulateur neuronal nécessite une adaptation en ligne de ces paramètres ce qui augmente la charge calculatoire et complique le calcul particulièrement dans le cas des systèmes non linéaires multivariables.

Afin de résoudre ce problème, une première contribution de cette thèse consiste a développer un émulateur multimodèle pour l'émulation et la commande des systèmes non linéaires multivaribles. Cet émulateur est le sujet du chapitre suivant.

Il s'avère aussi important de signaler que la structure de la commande neuronale basée sur un émulateur neuronal et tel qu'elle est présentée dans ce chapitre ne permet pas la commande des systèmes non linéaires multivariables non carrés. Une nouvelle structure de commande neuronale adaptative pour une classe de systèmes non linéaires multivariables non carrés (sous actionnés) est le sujet des chapitres suivants.

Chapitre II

Un Emulateur Multimodèle pour la commande neuronale adaptative des systèmes non linéaires multivariables carrés

Sommaire

II.1 Introduction

La synthèse d'une loi de commande nécessite une connaissance précise de la dynamique du système à étudier. Une modélisation précise s'avère alors une phase préliminaire. Dans ce contexte, plusieurs travaux de recherche ont été dédiés à la représentation des systèmes non linéaires par une approche multimodèle. La motivation de cette approche découle du fait qu'il est souvent difficile de concevoir un modèle qui tienne compte de toute la complexité du système à étudier. En effet, la stratégie de l'approche multimodèle coïncide avec la démarche usuelle qui consiste à subdiviser un problème complexe en un ensemble de sous problèmes simples, maniables et pouvant être résolus séparément. L'idée consiste à représenter le comportement d'un système complexe par un ensemble de modèles partiels, souvent de structure simple, caractérisant le fonctionnement du système dans ses différentes zones de fonctionnement et possédant chacun un domaine de validité bien défini. Un mécanisme d'interpolation entre ces modèles permet de balayer tout l'espace de fonctionnement. On a ainsi un modèle global qui peut représenter le système dans la totalité de son domaine de fonctionnement.

Dans ce chapitre, on propose de mettre en œuvre une procédure d'identification hors ligne pour la conception d'un émulateur multimodèle pour la représentation et la commande des systèmes non linéaires multivariables. Cette procédure d'identification concerne la recherche d'une structure optimale et une estimation précise des paramètres du multimodèle. Dans une première partie de ce chapitre, on présente les différentes étapes nécessaires à l'élaboration d'un émulateur multimodèle découplé. Ensuite, l'émulateur obtenu est exploité par le schéma de commande adaptative neuronale, développé dans le premier chapitre, afin de surmonter les problèmes rencontrés lors de la mise en œuvre d'un émulateur neuronal. Finalement, on propose une méthode basée sur la classification pour la décomposition de l'espace de fonctionnement du système et la génération systématique des paramètres de synthèse du multimodèle. Quelques exemples de simulation sont présentés pour la mise en évidence de l'intérêt et de l'efficacité des méthodes proposées.

II.2 Un état de l'art sur la représentation multimodèle

L'utilisation de l'approche multimodèle pour la modélisation et la commande des systèmes dynamiques non stationnaires ou non linéaires est relativement récente, elle date des travaux de Johansen et Foss en 1992 [29]. Parmi les premières publications offrant une présentation complète et générale des approches multimodèles on peut citer le livre de Murray Smith [46] et les travaux de Gasso [20].
Cette approche doit sa popularité à la simplicité qu'elle apporte par la décomposition de

l'espace de fonctionnement d'un système en plusieurs sous espaces. Plusieurs travaux se sont intéressés à la modélisation, l'émulation, l'estimation et la commande des systèmes non stationnaires et non linéaires monovariables par l'approche multimodèle. A titre indicatif, on invite le lecteur à consulter ces travaux [48; 67; 53; 1; 38; 64] qui portent une attention particulière à cette approche.

Dans ce chapitre on propose une structure multimodèle pour l'émulation des systèmes non linéaires multivariables. La stratégie proposée consiste à décomposer le système MIMO (Multi Input-Multi Output) en un sous ensemble de systèmes MISO (Multi Input-Single Output), ce qui permet de réduire la complexité de l'approche globale [50; 55; 32]. Un multimodèle MISO différent est donc élaboré pour caractériser chaque sortie du système. Cette stratégie d'identification permet de réduire le nombre de paramètres à identifier lors de la procédure d'optimisation de chaque multimodèle ce qui permet d'une part d'éviter la complexité de l'approche globale et d'autre part de faciliter la mise en œuvre des schémas de commande de ces systèmes. Pour chaque sortie $y_l(k)$ du système, l'approche multimodèle permet de remplacer la recherche d'un modèle unique M_l souvent complexe par la recherche d'une base B_l, $l = 1, ..., N_{OUT}$ de modèles locaux $M_{l,i}$ qui peuvent décrire localement le comportement du système sur son domaine de fonctionnement. Ces modèles n'ont, souvent rien à voir avec le système réel. En effet, ils peuvent être de structures différentes, d'ordre plus faible et, sont en général, linéaires.
Afin de générer le modèle global qui couvre la totalité du domaine de fonctionnement du système, l'approche multimodèle implique la determination d'un mécanisme d'interpolation par le calcul des fonctions appelées fonctions de pondération ou d'activation notées $\mu_{l,i}(\xi(k))$. Ces fonctions déterminent le degré d'activation du $i^{ième}$ modèle local associé à la sortie l : elles prennent une valeur suffisamment importante (proche de 1) si la variable d'indexation $\xi(k)$ correspondante est proche du centre de la zone de validité du modèle, et tend vers zéros au fur et à mesure que la variable d'indexation s'en éloigne. Un choix judicieux de la technique d'interpolation, permet d'approcher avec une précision suffisante le comportement du système dans la totalité de son domaine de fonctionnement. En effet, Au cours du temps, ces fonctions on été construites de différentes façons : en utilisant des fonctions de type booléen, des fonctions à dérivées discontinues (fonctions triangulaires ou trapézoïdales) ou des fonctions à dérivés continues (fonctions gaussiennes ou sigmoïdes) [20]. Une autre façon de construire les fonctions de pondération est d'utiliser les bornes des variables d'indexation si celles-ci sont disponibles [47]. Quelle que soit la méthode utilisée pour générer les fonctions de pondération, celles ci doivent satisfaire les propriétés de somme convexe suivante :

$$\begin{aligned} &\sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k)) = 1, \quad \forall\, l = 1...N_{OUT}, \\ &0 \leq \mu_{l,i}(\xi(k)) \leq 1 \end{aligned} \tag{II.1}$$

où N_{ml} est le nombre de modèles locaux associés à une base B_l.

Pour l'obtention d'un multimodèle, différentes techniques sont proposées dans la littérature. Ainsi, une première catégorie de méthodes se base sur la linéarisation autour de différents points de fonctionnement du système à modéliser ou sur la transformation polytopique convexe [47; 46; 66]. Ces deux méthodes sont basées sur des modèles mathématiques existants ce qui exige la connaissance d'une description phénoménologique et d'un modèle mathématique non linéaire du système à étudier. Toutefois, la complexité importante des systèmes non linéaires et/ou la connaissance insuffisante des phénomènes présents dans ces systèmes rend l'utilisation d'une telle démarche difficile dans la pratique. Dans une deuxième catégorie, on trouve les méthodes d'identification et d'estimation des paramètres à partir des données expérimentales (fichier entrées/sorties du système). Cette méthode permet de représenter le plus fidèlement possible le comportement externe du système sans chercher à obtenir un modèle de connaissance capable de décrire son comportement interne. Dans ce qui suit, on privilégie cette dernière méthode pour l'élaboration d'un émulateur multimodèle pour les systèmes non linéaires multivariables.

Il est important de noter qu'un choix judicieux de la structure des sous modèles et des fonctions d'activation permet d'approcher avec une précision suffisante n'importe quel comportement non linéaire dans un large domaine de fonctionnement. En effet, différentes représentations des modèles locaux sont envisageables ; à savoir la représentation sous forme d'une équation de régression entrée-sortie ou une représentation d'état. Le choix entre l'une ou l'autre de ces deux représentations obéit, bien évidement, à l'objectif fixé a priori (commande, observation, supervision...). Il est intéressant dans certaines applications telles que l'identification et la commande des systèmes MIMO, d'utiliser une représentation d'état qui permette de mettre facilement en évidence les sous-modèles. Cette représentation est simple et plus générale que la représentation polynômiale. D'autre part, plusieurs structures permettant d'interconnecter les différents sous-modèles peuvent être envisagées afin de générer la sortie globale du multimodèle. Deux structures multimodèles peuvent être distinguées : la première à états couplés connue aussi sous l'appellation de multimodèle de Takagi-Sugeno (T.S.) et la seconde à états découplés connue sous l'appellation de multimodèle découplé.

II.2.1 Structure couplée : Multimodèle de Takagi-Sugeno

La structure à états couplés a été introduite dans les années 80 par les travaux de Takagi et Sugeno dans un contexte de modélisation floue. Une étude précise de la relation entre les modèles flous de Takagi-Sugeno et la description par des modèles linéaires locaux a été effectuée par Hunt en 1996 [26]. Le multimodèle de T.S. a été depuis largement développé afin de représenter les systèmes non linéaires monovariables et multivariables

[28; 69; 20; 55; 50]. Cette structure multimodèle peut être mise soit sous une forme polynômiale (II.2), soit sous une forme de représentation d'état (II.4).
Sous la forme polynômiale, chaque multimodèle MISO est donné par :

$$y_{ml}(k) = \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k)) \left(-A_{l,i}(q^{-1})y_l(k) + B_{l,i}(q^{-1})U(k)\right) \tag{II.2}$$

avec :

$U(k)$: le vecteur de commande.

$y_{ml}(k)$: la sortie du multimodèle MISO représentant la $l^{ième}$ sortie du système.

et avec :

$$\begin{aligned} B_{l,i}(q^{-1}) &= \left[\begin{array}{cccccc} B_{l,i}^{1}(q^{-1}) & \dots & B_{l,i}^{\alpha}(q^{-1}) & \dots & B_{l,i}^{N_{IN}}(q^{-1}) \end{array}\right]; \\ A_{l,i}(q^{-1}) &= \sum_{j=1}^{n_{A_{l,i}}} a_{l,ji}q^{-j} \\ \text{avec } B_{l,i}^{\alpha}(q^{-1}) &= \sum_{j=1}^{n_{B_{l,i}^{\alpha}}} b_{l,ji}^{\alpha}q^{-j} \text{ et } \alpha \in [1, N_{IN}] \end{aligned} \tag{II.3}$$

où :

q^{-1} représente l'opérateur retard.

$B_{l,i}(q^{-1})$ et $A_{l,i}(q^{-1})$ désignent respectivement un vecteur des polynômes et un polynôme définissant les sous-modèles d'indice i de chaque base B_l .

Bien que les modèles entrée-sortie puissent approcher une large classe de systèmes non linéaires, ils ne conviennent pas pour un grand nombre de stratégies de commande. Beaucoup de méthodes actuelles de commande non linéaire sont fondées sur une représentation d'état.
Dans ce cas, chaque multimodèle MISO est donnée par :

$$\begin{aligned} X_{l,i}(k+1) &= A_{l,i}(\theta_{l,i})X_l(k) + B_{l,i}(\theta_{l,i})U(k) \\ X_l(k+1) &= \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k))X_{l,i}(k+1) \\ y_{ml}(k) &= \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k))C_{l,i}(\theta_{l,i})X_l(k) \end{aligned} \tag{II.4}$$

avec :

$A_{l,i}(\theta_{l,i})$, $B_{l,i}(\theta_{l,i})$ et $C_{l,i}(\theta_{l,i})$ la matrice d'évolution, la matrice de commande et la matrice d'observation de chaque sous-modèle.

$\theta_{l,i}$: le vecteur qui regroupe les paramètres des matrices $A_{l,i}$, $B_{l,i}$ et $C_{l,i}$.

$X_{l,i}(k)$: les variables d'état intermédiaires pour chaque multimodèle.

$X_l(k)$: les variables d'état communes aux sous-modèles d'une base B_l .

$U(k)$: le vecteur de commande.

$y_{ml}(k)$: la sortie du multimodèle MISO représentant la $l^{ième}$ sortie du système.

D'après l'équation (II.4), on peut clairement constater que, pour chaque multimodèle, l'état global $X_l(k)$ couple tout les états intermédiaires $X_{l,i}(k)$ par le mélange des équations dynamiques des sous-modèles et que les fonctions de pondération $\mu_{l,i}(\xi(k))$ effectuent un mélange des paramètres des sous-modèles en fonction de la zone de fonctionnement du système non linéaire. D'un point de vue structurel, on peut remarquer que tous les sous-modèles constituant ce multimodèle partagent le même vecteur d'état. La complexité des sous-modèles est, par conséquent, constante quelle que soit la complexité du système dans les différentes zones de fonctionnement. Le multimodèle ainsi obtenu risque alors d'être sur-paramétré et sa complexité se trouve inutilement augmentée.

II.2.2 Structure découplée : Multimodèle découplé

Le multimodèle découplé a été introduit par Filev en 1991 [19] en proposant un multimodèle issu de l'agrégation des sous-modèles sous la forme d'une structure à états découplés. Cette structure a été par la suite utilisée pour la modélisation, la commande et l'estimation d'état des systèmes non linéaires monovariables [54; 41; 7; 8]. Comme dans le cas d'une structure couplée, les sous-modèles de la structure découplée peuvent exploiter une équation de régression Entrée-Sortie (II.5) ou une représentation d'état (II.6). L'identification de cette structure en utilisant les deux méthodes de représentation est représentée dans ce travail.
Une représentation sous la forme polynômiale de chaque multimodèle MISO peut être donnée par :

$$\begin{aligned} y_{l,i}(k) &= -A_{l,i}(q^{-1})y_{l,i}(k) + B_{l,i}(q^{-1})U(k) \\ y_{ml}(k) &= \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k))y_{l,i}(k) \end{aligned} \tag{II.5}$$

avec :

$U(k)$: le vecteur de commande.

$y_{ml}(k)$: la sortie du multimodèle MISO représentant la $l^{ième}$ sortie du système.

$B_{l,i}(q^{-1})$ et $A_{l,i}(q^{-1})$: désignent respectivement un vecteur des polynômes et un polynôme définissant les sous-modèles d'indice i de chaque base B_l définis par l'équation (II.3) .

Une représentation d'état de chaque multimodèle MISO peut, dans ce cas, être donnée par :

$$\begin{aligned} X_{l,i}(k+1) &= A_{l,i}(\theta_{l,i})X_{l,i}(k) + B_{l,i}(\theta_{l,i})U(k) \\ y_{l,i}(k) &= C_{l,i}(\theta_{l,i})X_{l,i}(k) \\ y_{ml}(k) &= \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k))y_{l,i}(k) \end{aligned} \tag{II.6}$$

$X_{l,i}(k)$: les variables d'état du $i^{ème}$ sous-modèle de chaque multimodèle MISO.

$U(k)$: le vecteur de commande.

$y_{ml}(k)$: la sortie du multimodèle MISO représentant la $l^{ième}$ sortie du système.

En examinant l'équation (II.6), on peut remarquer que contrairement au multimodèle couplé, cette structure de multimodèle ne présente pas un état global $X_l(k)$: la prise en compte des contributions des sous modèles s'effectue à travers la somme pondérée des sorties des sous-modèles, c'est-à-dire le rôle des fonctions de pondération $\mu_{l,i}(\xi(k))$, dans ce cas, est de pondérer la sortie de chaque sous-modèle sans mélanger ses paramètres.
C'est le découplage entre les sous-modèles qui motive l'intérêt de cette structure. Il est, en effet, possible d'envisager une transposition plus facile des techniques d'analyse de systèmes linéaires au multimodèle et d'utiliser des sous-modèles de structure complètement différente (de type linéaire ou non linéaire et de dimensions différentes).

II.3 Un émulateur multimodèle découplé

Classiquement, l'élaboration d'un multimodèle impose de passer par les trois étapes fondamentales suivantes : le choix de la variable de décision, la décomposition de l'espace de fonctionnement et finalement l'identification structurelle et paramétrique des sous-modèles. Dans le but d'obtenir un multimodèle simple et capable de fournir une bonne caractérisation du système à identifier, on développe dans ce paragraphe les différentes étapes précitées.

II.3.1 Choix de la variable de décision

La variable de décision appelée aussi variable d'indexation ou de prémisse $\xi(k)$ est une variable vectorielle qui intervient dans la construction des fonctions de pondération $\mu_{l,i}(\xi(k))$. Ce vecteur peut contenir une ou plusieurs variables internes ou externes du système. En effet, $\xi(k)$ peut être choisie comme le vecteur de sortie, le vecteur d'entrée et/ou le vecteur d'état mesurable.
Pour réduire la complexité de la procédure d'estimation paramétrique, dans le présent travail, on peut choisir le vecteur des commandes du système $U(k)$ comme une variable de décision.

II.3.2 Décomposition de l'espace de fonctionnement

Il est admis que la connaissance a priori des fonctions de pondérations n'est pas souvent possible. En effet, afin de garantir la précision de l'identification multimodèle, la plupart des travaux optent pour l'optimisation simultanée des paramètres des fonctions de pondérations et des paramètres des modèles locaux [67; 1; 22; 21]. D'autres techniques de décomposition de l'espace de fonctionnement des systèmes non-linéaires, peuvent être utilisées, parmi celles-ci on trouve l'approche ascendante développée par Nelles et *al.* [50]

et l'approche descendante développée par Gasso et *al.* [22; 21]. A ce niveau, on invite le lecteur à consulter la thèse d'Orjuela [53] où une bibliographie riche est présentée pour exposer ces différentes méthodes de partitionnement de l'espace de fonctionnement.
Dans ce même contexte, les méthodes basées sur la classification en utilisant la carte de Kohonen et la classification de Chiu ont été proposées pour la décomposition de l'espace de fonctionnement pour des systèmes non linéaires mono-variables [36; 37; 43; 42; 64].
Dans le présent travail, ce problème est contourné, dans un premier temps, en se fixant a priori les fonctions de pondération et le partitionnement de l'espace de fonctionnement du système. Le positionnement des différentes fonctions poids résulte d'une analyse heuristique des caractéristiques statiques qui constituent une information a priori sur le système à identifier. La complexité du problème à traiter est ainsi réduite et les seuls paramètres à identifier sont ceux des sous-modèles. Dans un deuxième temps, une méthode systématique est proposée pour la décomposition de l'espace de fonctionnement et la génération systématique des paramètres de synthèse du multimodèle. Cette méthode basée, principalement, sur une procédure de classification de données est présentée à la fin de ce chapitre.
Après avoir choisi la variable de décision et les fonctions de pondérations, l'étape relative à l'estimation paramétrique d'un multimodèle découplé peut être formulé.

II.3.3 Procédure d'identification paramétrique

Cette section est consacrée à la procédure d'identification paramétrique des multimodèles découplés MISO [11; 9]. Le multimodèle obtenu, après une étape de validation servira comme émulateur du comportement dynamique du système non linéaire multivariable étudié.

Selon un choix entre l'une de deux représentations d'état ou polynômiale, la structure découplée utilisée pour chaque multimodèle d'indice l ($l = 1, ..., N_{OUT}$) est donnée par l'une des relations suivantes :

$$\text{Représentation polynômiale} \quad \begin{cases} y_{l,i}(k) = -A_{l,i}(q^{-1})y_{l,i}(k) + B_{l,i}(q^{-1})U(k) \\ y_{ml}(k) = \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k))y_{l,i}(k) \end{cases} \tag{II.7}$$

$$\text{Représentation d'état} \quad \begin{cases} X_{l,i}(k+1) = A_{l,i}(\theta_{l,i})X_{l,i}(k) + B_{l,i}(\theta_{l,i})U(k) \\ y_{l,i}(k) = C_{l,i}(\theta_{l,i})X_{l,i}(k) \\ y_{ml}(k) = \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k))y_{l,i}(k) \end{cases} \tag{II.8}$$

Dans toute la suite le vecteur de commande du système $U(k)$ sert de variable de décision car il conduit le système vers les différents modes de fonctionnement. Le vecteur $\xi(k)$ est, alors, défini comme suit :

$$\xi(k) = [\xi_1(k), ..., \xi_e(k), ...\xi_N(k)]^T = [u_1(k), ..., u_e(k), ..., u_{N_{IN}}(k)]^T \tag{II.9}$$

La figure II.1 donne l'architecture d'un multimodèle découplé MISO en utilisant une représentation d'état.

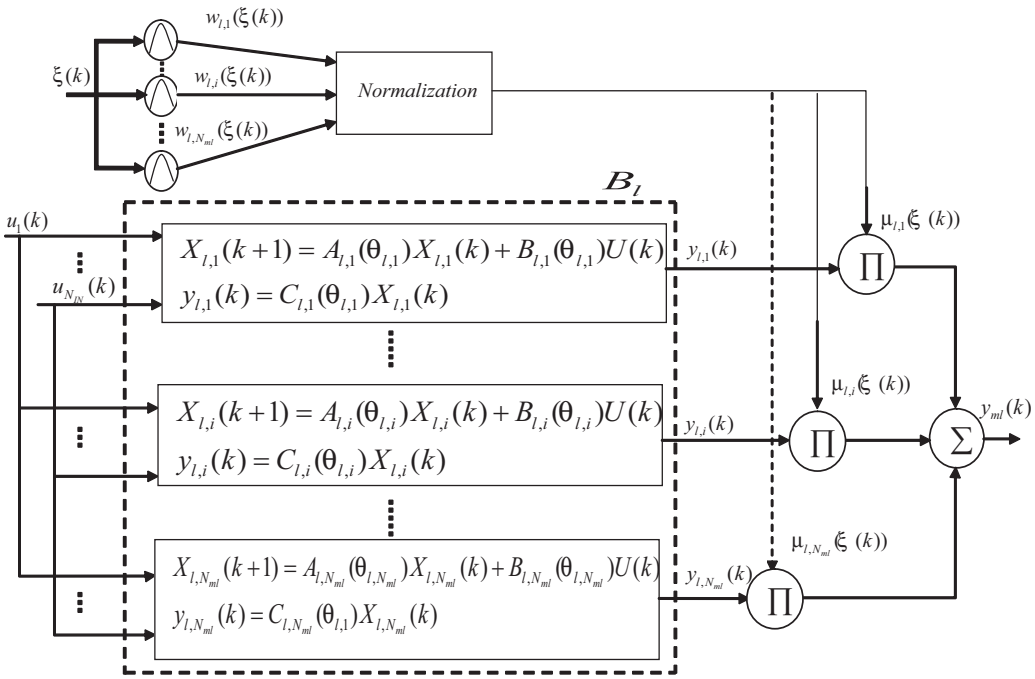


Figure II.1: Structure de l'émulateur multimodèle MISO représentant une sortie y_l d'un système non linéaire multivaribale (sous-modèle d'état).

Les fonctions d'activation $\mu_{l,i}(\xi(k))$ sont construites par la normalisation de fonctions gaussiennes $w_{l,i}(\xi(k))$ qui sont des fonctions continuement dérivables. Elles sont obtenues selon les équations (II.10) et (II.11) :

$$w_{l,i}(\xi(k)) = \prod_{e=1}^{N_{IN}} \exp(-\frac{(\xi_e(k) - c_{l,ie})^2}{\sigma_{l,ie}^2}) \tag{II.10}$$

$$\mu_{l,i}(\xi(k)) = \frac{w_{l,i}(\xi(k))}{\sum_{p=1}^{N_{ml}} w_{l,p}(\xi(k))} \tag{II.11}$$

$c_{l,ie}$ et $\sigma_{l,ie}$ désignent respectivement les centres et les dispersions correspondant à la $i^{ème}$ zone de fonctionnement. Le choix du nombre de modèles N_{ml}, des paramètres $c_{l,ie}$ et $\sigma_{l,ie}$ a une grande influence sur la précision de l'émulateur. Ce choix va être effectué en exploitant la caractéristique statique.

Pour les deux représentations, l'identification paramétrique consiste à déterminer pour chaque modèle local i, de la base B_l qui construit le $l^{i\text{ème}}$ multimodèle MISO, le vecteur de paramètre $\theta_{l,i}$ donné par :

$$\theta_{l,i} = [\theta_{l,i_1} \quad \dots \quad \theta_{l,i_p} \quad \dots \quad \theta_{l,i_{p_{l,i}}}]^T \tag{II.12}$$

avec :

$p_{l,i}$: le nombre maximal de paramètres appartenant au sous-modèle d'indice i.

θ_{l,i_p}, $p = 1, ..., p_{l,i}$: les paramètres scalaires à identifier.

La procédure d'optimisation adoptée, nécessite le regroupement des blocs colonnes $\theta_{l,i}$ dans un vecteur colonne θ_l. Chaque base de sous-modèles B_l est donc entièrement définie par le vecteur de paramètres suivant :

$$\theta_l = [\theta_{l,1}^T \quad ... \quad \theta_{l,i}^T \quad ... \quad \theta_{l,N_{ml}}^T]^T \tag{II.13}$$

D'après les equations (II.7) et (II.8), on peut remarquer que la structure du modèle du système est non linéaire par rapport aux paramètres des sous modèles. Afin d'estimer les paramètres des modèles partiels, on recourt, donc, à des techniques numériques d'optimisation non linéaire.

L'identification paramétrique en temps différé nécessite la connaissance d'une séquence d'entrée $U(k)$ et d'une séquence de sortie $Y(k)$ du système non linéaire. Elle se base sur l'optimisation d'un critère quadratique en utilisant un algorithme d'optimisation non linéaire.
Le multimodèle doit assurer une adéquation globale entrée/sortie du système afin d'être capable de prédire correctement le comportement du système dans le domaine de validité donné. Cependant, en fonction de son contexte d'exploitation, il peut être également souhaitable d'assurer une adéquation locale du multimodèle et du système (les sous-modèles doivent fournir une caractérisation du comportement du système dans chaque zone de fonctionnement). Pour répondre à ces contraintes, trois critères (global, local ou combiné) peuvent être utilisés pour la procédure d'identification paramétrique. Dans le présent cas, un critère global défini par l'équation suivante est retenu :

$$J_{ml} = \frac{1}{2}\sum_{k=1}^{N_H}(y_{ml}(k) - y_l(k))^2 = \sum_{k=1}^{N_H} e_{ml}(k) \tag{II.14}$$

où $y_{ml}(k)$ est la sortie du multimodèle MISO représentant la $l^{ième}$ sortie du système, $y_l(k)$ est la $l^{ième}$ sortie du système non linéaire et N_H est le nombre de mesures.

Plusieurs algorithmes peuvent être utilisés pour assurer la minimisation itérative de ce critère, on peut citer les plus utilisés : l'algorithme du gradient, l'algorithme de Gauss-Newton et l'algorithme de Levenberg-Marquardt . L'algorithme du gradient est toujours stable mais très coûteux en temps de calcul. L'algorithme de Gauss-Newton présente une convergence rapide (contrairement à l'algorithme du gradient) mais il présente un risque important d'instabilité si les paramètres initiaux sont loin de l'optimum. De plus, rien n'assure que l'algorithme converge vers un minimum global. Le succès de cette méthode

d'optimisation est, donc, tributaire d'un choix judicieux des paramètres initiaux. Dans ce travail, on a opté pour l'algorithme de Levenberg-Marquardt [40] qui combine judicieusement les deux autres méthodes d'optimisation. La mise à jour du vecteur de paramètres est obtenue à partir de la formule de récurrence suivante :

$$\theta_l(it+1) = \theta_l(it) - \Delta_l(it)(H_l(\theta_l) + \lambda_l(it)I)^{-1}G_l(\theta_l) \tag{II.15}$$

où it est l'itération courante, I est la matrice identité de dimension appropriée, $G_l(\theta_l)$ et $H_l(\theta_l)$ désignent respectivement le vecteur gradient et la matrice hessienne. $\Delta_l(it)$ est un coefficient de relaxation introduit pour minimiser le critère dans la direction du vecteur $H_l(\theta_l)^{-1}G_l(\theta_l)$. $\lambda_l(it)$ est un scalaire (paramètre de régularisation). Si sa valeur est proche de zéro alors l'algorithme est proche de celui de Gauss-Newton et si elle tend vers l'infini ($\lambda_l(it) \gg 0$) alors l'algorithme est proche de celui du gradient. Cet algorithme peut tirer, à la fois, profit de la stabilité de l'algorithme du gradient et de la rapidité de convergence de l'algorithme de Gauss-Newton. A chaque itération les valeurs de $\lambda_l(it)$ et $\Delta_l(it)$ sont réglées, le plus souvent au moyen d'une heuristique basée sur l'évolution du critère.
L'estimation paramétrique de chaque multimodèle MISO nécessite le calcul d'un vecteur gradient et d'une matrice hessienne. Le vecteur gradient $G_l(\theta_l)$ s'obtient en dérivant le critère global par rapport aux paramètres θ_l, soit :

$$G_l(\theta_l) = \frac{\partial J_{ml}}{\partial \theta_l} = \sum_{k=1}^{N_H} (y_{ml}(k) - y_l(k)) \frac{\partial y_{ml}(k)}{\partial \theta_l} \tag{II.16}$$

La matrice hessienne $H_l(\theta_l)$ s'obtient en dérivant deux fois le critère global par rapport aux paramètres θ_l, soit :

$$H_l(\theta_l) = \frac{\partial^2 J_{ml}}{\partial \theta_l \partial \theta_l^T} = \sum_{k=1}^{N_H} \frac{\partial y_{ml}(k)}{\partial \theta_l} \left(\frac{\partial y_{ml}(k)}{\partial \theta_l} \right)^T \tag{II.17}$$

avec :

$$\frac{\partial y_{ml}(k)}{\partial \theta_l} = \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k)) \frac{\partial y_{l,i}(k)}{\partial \theta_l} \tag{II.18}$$

Le calcul des fonctions $\frac{\partial y_{l,i}(k)}{\partial \theta_l}$ appelées fonctions de sensibilité de premier ordre exige le calcul du Jacobian de chaque $y_{l,i}(k)$ qui n'est autre que le vecteur qui comporte toutes les dérivées premières de $y_{l,i}(k)$ par rapport à tous les paramètres du multimodèle correspondant.

La démarche adoptée pour calculer les fonctions de sensibilité diffère selon le choix retenu de la structure multimodèle.
Pour chaque base B_l, si les sous-modèles sont représentés par une équation polynômiale (équation (II.7), les fonctions de sensibilité seront calculées comme suit :

$$\begin{aligned} \frac{\partial y_{l,i}(k)}{\partial \theta_{l,i_p}} &= -\frac{\partial A_{l,i}(q^{-1})}{\partial \theta_{l,i_p}} y_{l,i}(k) - A_{l,i}(q^{-1}) \frac{\partial y_{l,i}(k)}{\partial \theta_{l,i_p}} + \frac{\partial B_{l,i}(q^{-1})}{\partial \theta_{l,i_p}} U(k), \\ & l = 1, ..., N_{OUT},\ i = 1, ..., N_{ml} \text{ et } p = 1, ..., p_{l,i} \end{aligned} \tag{II.19}$$

Si les sous-modèles sont sous la forme d'une représentation d'état, le calcul des fonctions de sensibilité consiste à dériver les sorties des sous-modèles de l'équation (II.8) par rapport aux paramètres θ_{l,i_p}. Il convient alors de distinguer deux sortes de fonctions de sensibilité. Les premières s'explicitent par dérivation partielle des équations de sortie des sous-modèles comme suit :

$$\begin{gathered}\frac{\partial y_{l,i}(k)}{\partial \theta_{l,i_p}} = \frac{\partial C_{l,i}}{\partial \theta_{l,i_p}} X_{l,i}(k) + C_{l,i} \frac{\partial X_{l,i}(k)}{\partial \theta_{l,i_p}}, \\ l = 1, ..., N_{OUT},\ i = 1, ..., N_{ml} \text{ et } p = 1, ..., p_{l,i}\end{gathered} \tag{II.20}$$

L'état étant inconnu à l'instant présent, la seconde classe des fonctions de sensibilité est calculée, alors, par dérivation partielle des états de l'équation (II.8) par rapport à chaque paramètre θ_{l,i_p} comme suit :

$$\frac{\partial X_{l,i}(k+1)}{\partial \theta_{l,i_p}} = \frac{\partial A_{l,i}}{\partial \theta_{l,i_p}} X_{l,i}(k) + A_{l,i} \frac{\partial X_{l,i}(k)}{\partial \theta_{l,i_p}} + \frac{\partial B_{l,i}}{\partial \theta_{l,i_p}} U(k) \tag{II.21}$$

A ce niveau, il s'avère important de signaler que le découplage entre les dynamiques des sous-modèles constituant chaque base B_l entraîne un découplage entre les fonctions de sensibilité des différents sous-modèles. Ceci entraîne la simplification des équations (II.19), (II.20) et (II.21) lors de la mise en œuvre de l'algorithme. Par conséquent :

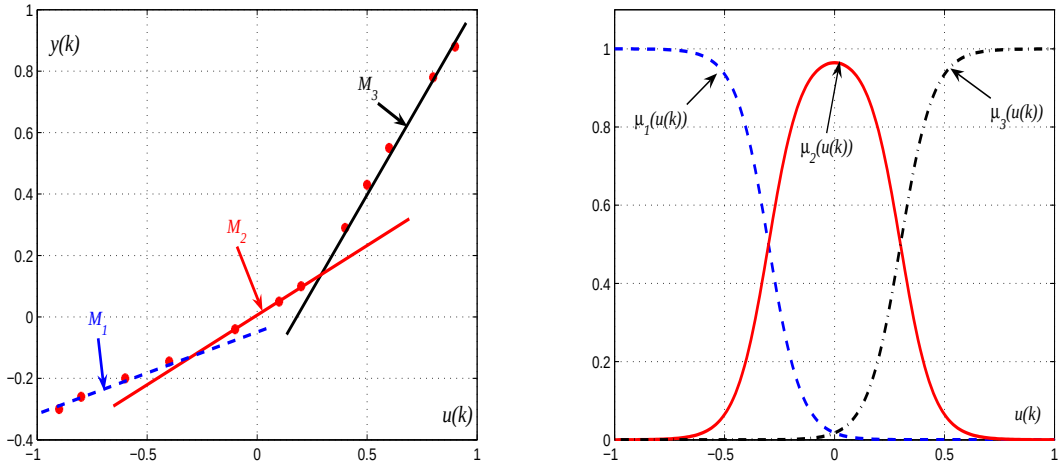
$$\frac{\partial y_{l,i}(k)}{\partial \theta_{l,j}} = 0_{1 \times p_{l,j}} \text{ pour } i \neq j \ \ i = 1, ..., N_{ml} \text{ et } j = 1, ..., N_{ml} \tag{II.22}$$

II.3.4 Simulations numériques

Les deux exemples retenues dans le premier chapitre sont repris dans cette section pour permettre à la fois l'évaluation de la précision du multimodèle élaboré dans ce chapitre et sa comparaison à l'émulateur neuronal.

II.3.4.1 Une émulation multimodèle d'un système non linéaire SISO

Pour mettre en évidence l'apport en performances de l'émulateur multimodèle proposé pour la représentation d'un système non linéaire SISO, on reprend l'exemple considéré précédemment qui est défini par l'équation (I.24).
L'identification du multimodèle est assurée par la minimisation d'un critère global (défini par la relation (II.14)). L'entrée d'excitation $u(k)$ du système est constituée par la concaténation de créneaux d'amplitudes variables ($u(k) \in [-1, 1]$). En exploitant la caractéristique statique du système, on peut opter pour un partitionnement homogène de l'espace de fonctionnement en trois zones de fonctionnement et retenir par la suite une base de trois sous-modèles ($N_m = 3$). Les paramètres de synthèse des fonctions de pondération $\mu_i(u(k))$ retenus sont les suivants : $c_1 = -0.6$, $c_2 = 0$, $c_3 = 0.6$ et $\sigma = 0.3$. La caractéristique statique du système ainsi que les fonctions de pondération utilisées pour partitionner son espace de fonctionnement sont illustrées sur la figure II.2.



(a) La caractéristique statique du système

(b) Les fonctions de pondération retenues

Figure II.2: Le partionnement de l'espace de fonctionnement.

On note que, en général, les modèles locaux doivent avoir une structure aussi simple que possible. Seule la phase de validation de la base de modèles peut autoriser l'augmentation de la complexité de la structure de base. Dans ce cas, une base de trois modèles représentés sous la forme d'une équation polynômiale d'ordre deux fournit une précision suffisante. Les sous-modèles se présentent, alors, sous la forme suivante :

$$y_i(k) = -A_i(q^{-1})y_i(k) + B_i(q^{-1})u(k) \tag{II.23}$$

avec :

$$A_i(q^{-1}) = \sum_{j=1}^{2} a_{ji}q^{-j} \text{ et } B_i(q^{-1}) = \sum_{j=1}^{2} b_{ji}q^{-j}$$

Les figures II.3, II.4 et II.5 illustrent l'approximation des paramètres du multimodèle polynomial découplé. Ces figures montrent que ces variables convergent vers les valeurs optimales permettant d'avoir la base des modèles locaux caractérisés par les polynômes suivants :

* Model M_1

$$A_1(q^{-1}) = -0.6527q^{-1} - 0.0076q^{-2},$$

$$B_1(q^{-1}) = 0.4351q^{-1} \text{ - } 0.3239q^{-2},$$

* Model M_2

$$A_2(q^{-1}) = 0.2617q^{-1} - 0.2318q^{-2},$$

$$B_2(q^{-1}) = 0.3194q^{-1} + 0.1146q^{-2},$$

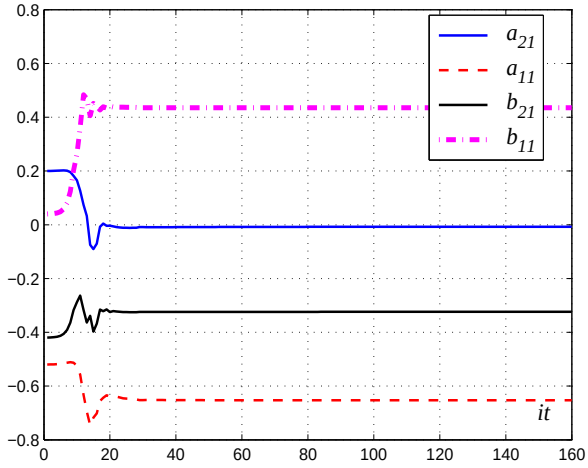


Figure II.3: Variations des paramètres du Modèle M_1.

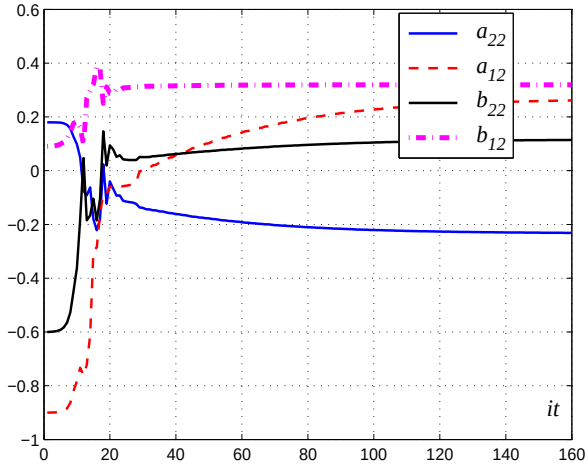


Figure II.4: Variations des paramètres du Modèle M_2.

* Model M_3

$$A_3(q^{-1}) = -1.2365q^{-1} + 0.3380q^{-2},$$

$$B_3(q^{-1}) = 0.3903q^{-1} - 0.2945q^{-2},$$

Pour la validation de l'émulateur multimodèle obtenu on utilise la même séquence d'entrée que celle utilisée, dans le premier chapitre, pour l'émulateur neuronal et définie par l'équation (I.25). Les résultats de validation de l'émulation multimodèle donnés sur la figure II.6 montrent que le multimodèle découplé permet une bonne adéquation entre la sortie du système non linéaire et celle de l'émulateur. L'erreur d'émulation, tracée à l'échelle logarithmique sur la figure II.7 laisse apparaitre une précision d'émulation satis-

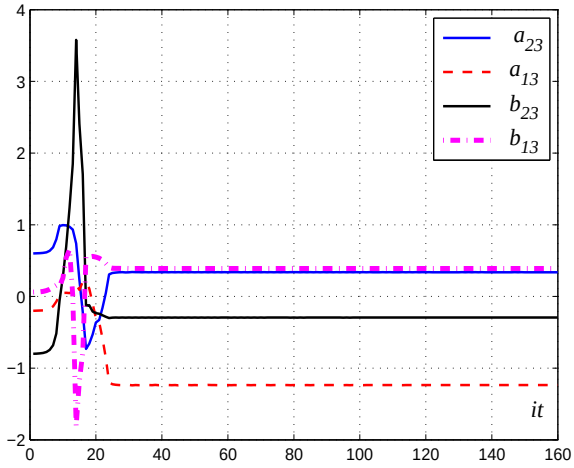


Figure II.5: Variations des paramètres du Modèle M_3.

faisante. Les performances de l'émulateur multimodèle proposé peuvent être comparées

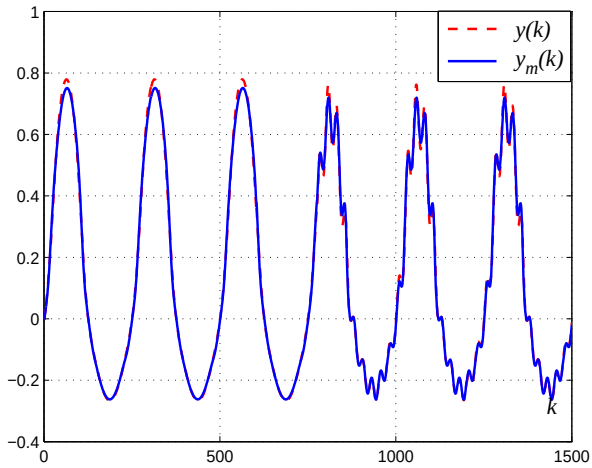


Figure II.6: Variations de la sortie réelle et de l'émulateur multimodèle.

à celles offertes par l'émulateur neuronal. Sachant que les performances de l'émulation neuronale dépendent du paramètre de démarrage ε_e, les indices des performances MSE et VAF sont calculés pour différentes valeurs de ε_e (figure II.8).

Cette figure montre que l'indice MSE minimale est atteint pour $\varepsilon_e = 100$ (correspond à une valeur maximale de VAF). Cette valeur de ε_e est donc retenue comme un bon choix du paramètre de démarrage.

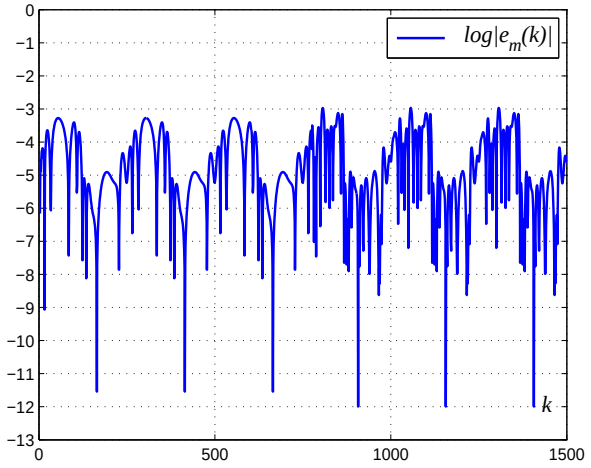
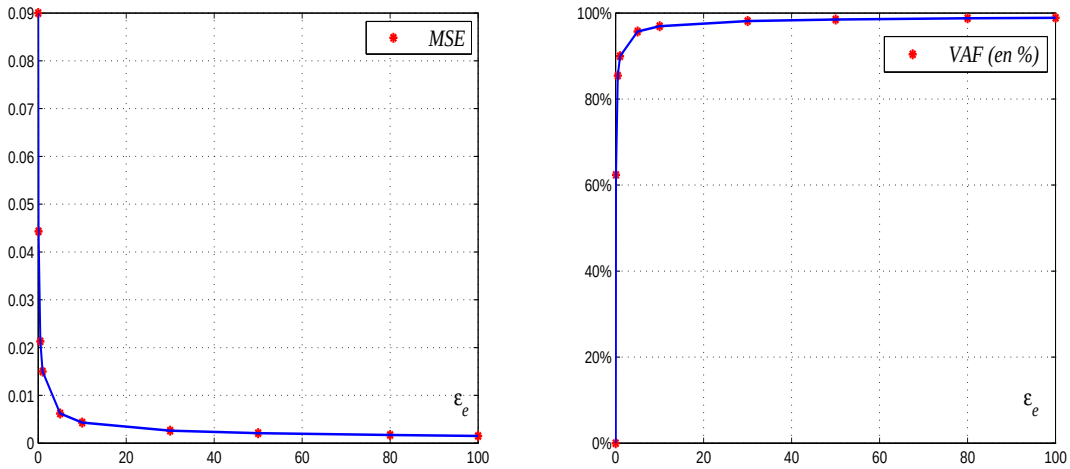


Figure II.7: Erreur d'émulation à l'échelle logarithmique.



(a) Variation de l'erreur MSE en fonction de ε_e

(b) Variation de la VAF en fonction de ε_e

Figure II.8: Variations des indices de performances en fonction de ε_e (Emulation neuronale).

Le tableau II.1 résume les indices des performances MSE et VAF calculés dans le cas de deux émulateurs. Ce tableau confirme que les performances de l'émulation neuronale dépendent du choix du paramètre de démarrage ε_e. L'émulateur multimodèle permet d'obtenir des résultats relativement meilleurs que ceux obtenus par un émulateur neuronal avec un bon choix du paramètre de démarrage.

	Emulateur Multimodèle	Emulateur Neuronal
MSE	$2.09\,10^{-4}$	$1.5\,10^{-2}(\varepsilon_e = 1)$ $1.5\,10^{-3}(\varepsilon_e = 100)$
VAF	99.8214%	89.95%$(\varepsilon_e = 1)$ 98.87%$(\varepsilon_e = 100)$

Tableau II.1: Les indices MSE et VAF dans les cas des émulateurs multimodèle et neuronal.

II.3.4.2 Une émulation multimodèle d'un système non linéaire MIMO carré

Dans ce paragraphe, un exemple de simulation numérique est présenté pour mettre en évidence l'efficacité de l'émulateur multimodèle dans le cas d'un système non linéaire MIMO [10; 9]. On considère le système non linéaire à deux entrées et deux sorties défini par l'équation (I.26). Ce système peut être représenté par deux multimodèles MISO. L'identification de ces deux multimodèles est réalisée à la base d'un critère global et par l'application des commandes riches en fréquence au système non linéaire considéré. Ces entrées $u_l(k)$, $l = 1, 2$ qui servent, par ailleurs, comme des variables de décision pour les fonctions de pondération ($\xi_l(k) = u_l(k)$), sont constituées par la somme de signaux sinusoidaux d'amplitudes et de fréquences différentes.
L'exploitation des caractéristiques statiques du système (figure II.9) permet un partitionnement homogène de l'espace de fonctionnement. Chaque multimodèle MISO est représenté par une base de quatre sous-modèles ($N_{ml} = 4$, $l = 1, 2$). Les paramètres de synthèse retenus pour chaque bases sont les suivants :

$$y_{m1}\begin{cases} c_{1,11} = c_{1,21} = -1, \;\; c_{1,31} = c_{1,41} = 0.6, \;\; \sigma_{1,1} = 0.8 \\ \\ c_{1,12} = c_{1,32} = -1.2, \;\; c_{1,22} = c_{1,42} = 1.4, \;\; \sigma_{1,2} = 0.6 \end{cases}$$

$$y_{m2}\begin{cases} c_{2,11} = c_{2,21} = -1.2, \;\; c_{2,31} = c_{2,41} = 1.4, \;\; \sigma_{2,1} = 0.6 \\ \\ c_{2,12} = c_{2,32} = -0.8, \;\; c_{2,22} = c_{2,42} = 0.8, \;\; \sigma_{2,2} = 0.3 \end{cases}$$

Les fonctions de pondération associées à chaque zone de fonctionnement sont, donc, formulées par les expressions (II.10) et (II.11) et sont représentées sur la figure II.10.
Dans ce cas, deux bases construites par des modèles d'état locaux du premier ordre fournissent suffisamment de précision. En effet, les résultats de validation de l'émulation multimodèle, donnés sur la figure II.11, montrent l'évolution des sorties du système non

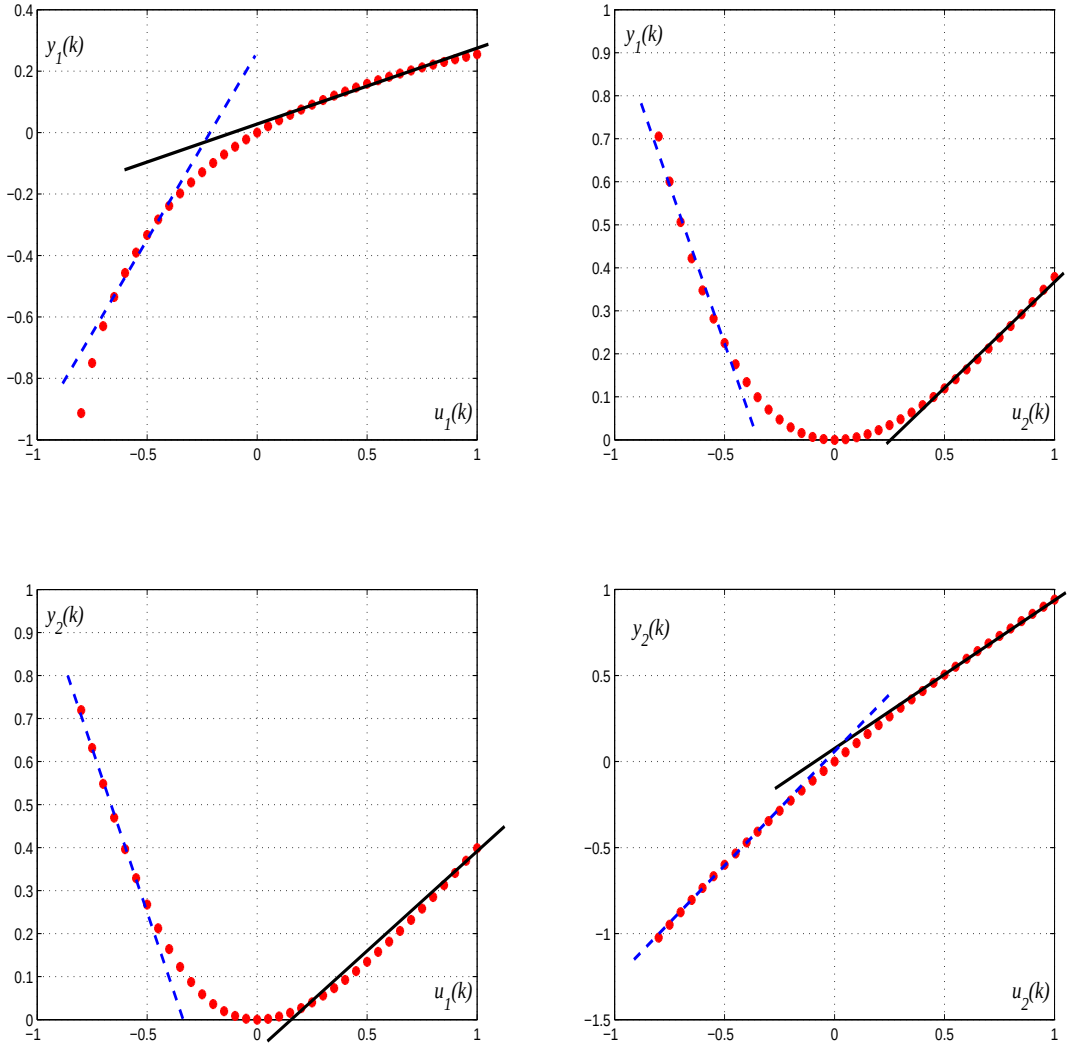
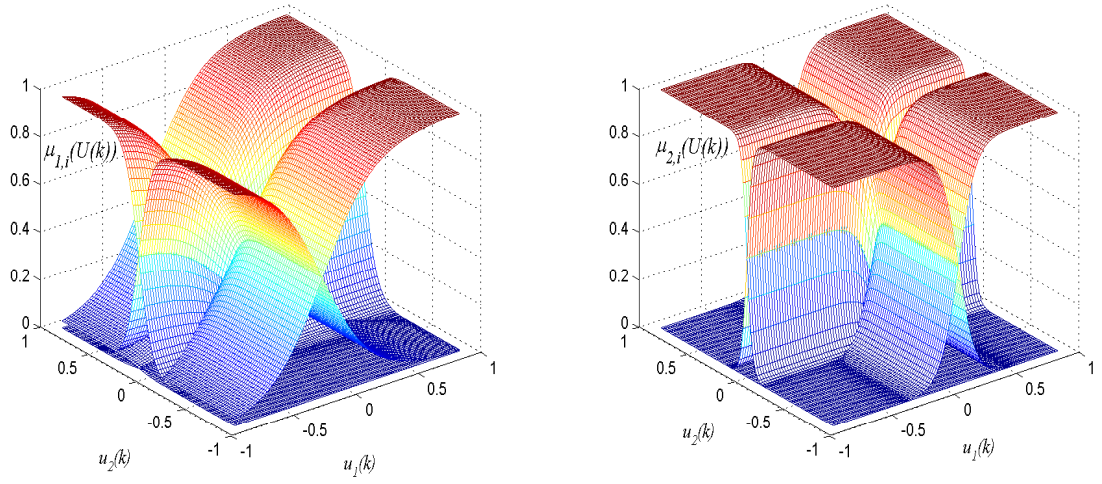


Figure II.9: Les caractéristiques statiques du système MIMO.

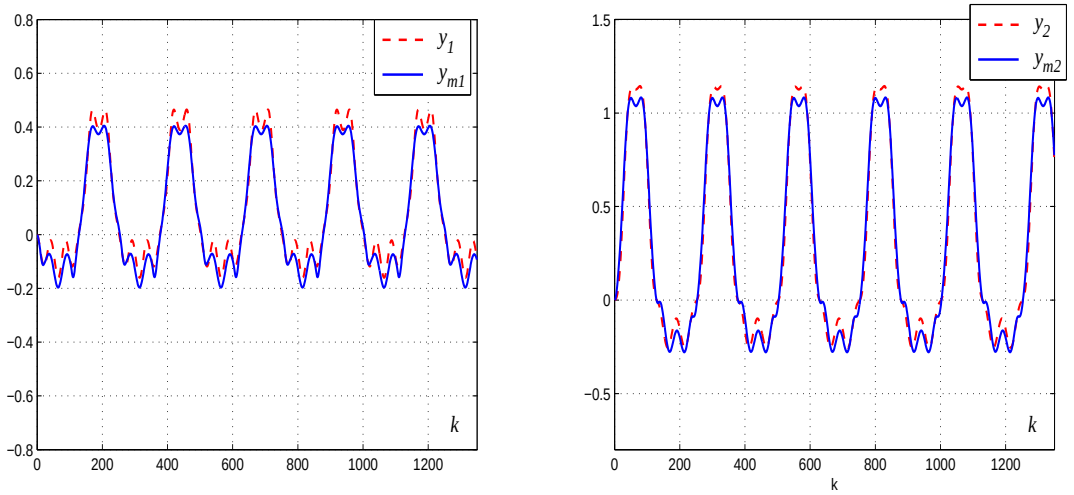
linéaire et les sorties du multimodèle ($y_l(k)$ et $y_{ml}(k)$, $l = 1, 2$). Les résultats de simulation confirment que l'émulateur multimodèle découplé a pu offrir une précision de modélisation satisfaisante.



(a) Les fonctions de pondération retenues pour le premier multimodèle

(b) Les fonctions de pondération retenues pour le deuxième multimodèle

Figure II.10: Les fonctions de pondération retenues pour les deux multimodèles MISO.

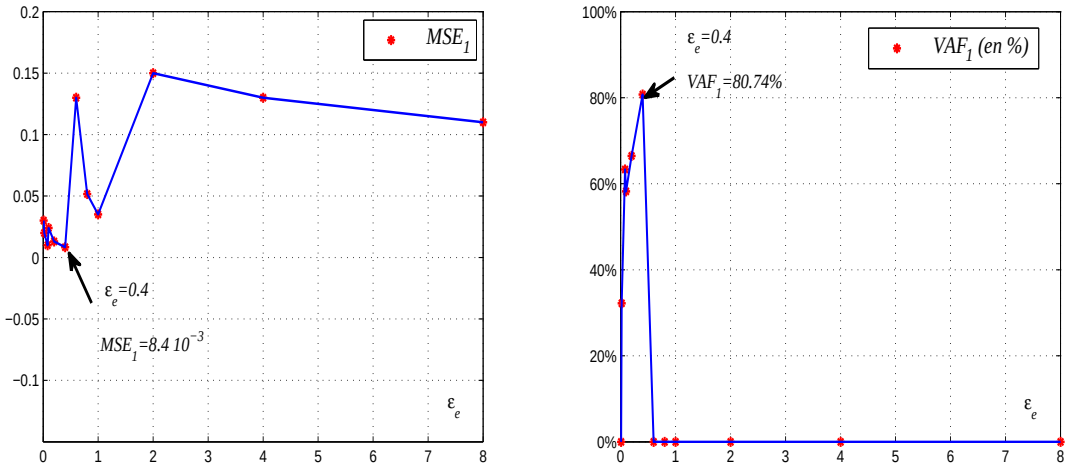


(a) Première sortie du système

(b) Deuxième sortie du système

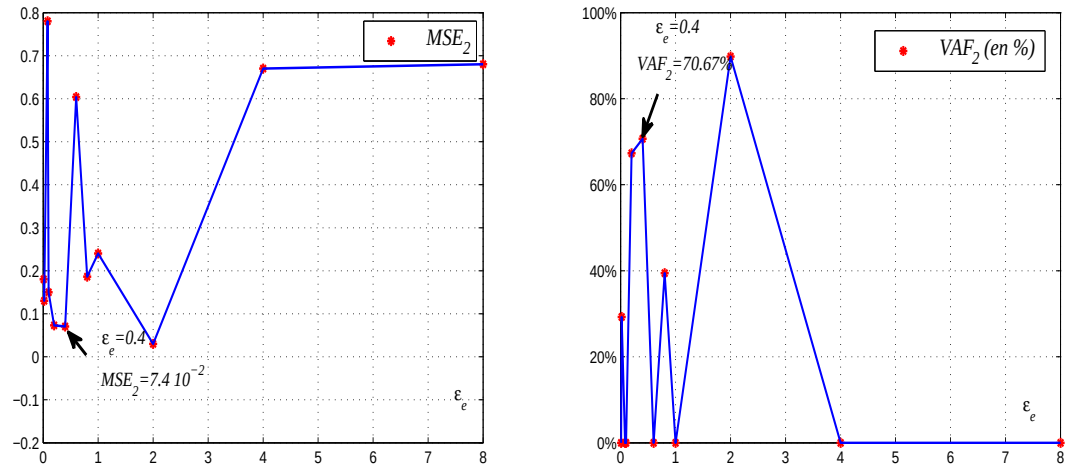
Figure II.11: Variations des sorties réelles et celles de l'émulateur multimodèle.

Ces résultats sont aussi comparés à ceux obtenus par l'utilisation d'un émulateur neuronal. L'évolution des indices de performances MSE_l et VAF_l $(l = 1, 2)$ (figures II.12 et II.13) en fonction du paramètre de démarrage ε_e a conduit à un choix judicieux de la valeur de $\varepsilon_e = 0.4$.



(a) Variation de l'erreur MSE_1 en fonction de ε_e (b) Variation de la VAF_1 en fonction de ε_e

Figure II.12: Variations des indices de performances en fonction de ε_e(première sortie du système, émulation neuronale).



(a) Variation de l'erreur MSE_2 en fonction de ε_e (b) Variation de la VAF_2 en fonction de ε_e

Figure II.13: Variations des indices de performances en fonction de ε_e (deuxième sortie du système, émulation neuronale).

Le tableau II.2 résume, dans ce cas, les valeurs des indices de performances dans le cas des émulateurs multimodèle et neuronal. Ce tableau montre des diminutions des erreurs MSE_l et des augmentations d'environ 18% et 28% des VAF_l $(l = 1, 2)$ calculées pour les deux sorties du système en utilisant un émulateur multimodèle. Ces résultats confirment l'efficacité de l'émulateur multimodèle proposé par rapport à l'émulateur neuronal.

	Emulateur Multimodèle	Emulateur Neuronal
MSE_1	$1.4\,10^{-3}$	$8.4\,10^{-3}\,(\varepsilon_e = 0.4)$
MSE_2	$3.2\,10^{-3}$	$7.04\,10^{-2}\,(\varepsilon_e = 0.4)$
VAF_1	98.219%	80.74% $(\varepsilon_e = 0.4)$
VAF_2	98.8205%	70.67% $(\varepsilon_e = 0.4)$

Tableau II.2: MSE_l et VAF_l $(l = 1, 2)$ calculées pour les deux sorties du système dans les cas des émulations multimodèle et neuronal.

II.4 Une commande neuronale adaptative basée sur un émulateur multimodèle

Afin de résoudre les problèmes causés par l'émulateur neuronal en boucle fermée, l'émulateur multimodèle proposé est utilisé dans le schéma de commande adaptative neuronal pour représener la dynamique des systèmes non linéaires multivariables.

II.4.1 Adaptation des paramètres du correcteur neuronal

Dans ce qui suit, on considère, toujours, des systèmes non linéaires multivariables carrés (soit $N = N_{IN} = N_{out}$). Soit l'émulateur multimodèle découplé résultant de l'estimation paramétrique et décrit soit par des modèles polynômiaux (relation (II.7)) soit par des modèles d'état (relation (II.8)). Cet émulateur est inséré dans le schéma de commande de la figure I.1 à la place de l'émulateur neuronal pour approcher les dérivés partielles des sorties du système par rapport aux paramètres du contrôleur neuronal.
En utilisant l'émulateur multimodèle, l'expression (I.11) décrivant l'adaptation des poids du réseau neuronal de commande est remplacé par l'équation suivante :

$$\Delta\phi_{ij}(k) = |\eta_c(k)|\,\Delta T \sum_{l=1}^{N} \left((y_{c_l}(k-1) - y_l(k-1)) \frac{\partial y_{ml}(k-1)}{\partial \phi_{ij}} \right) \tag{II.24}$$

Les dérivées partielles des sorties du multimodèle par rapport aux poids du réseau neuronal de commande $\partial y_{ml}(k)/\partial\phi_{ij},\ l = 1, ..., N$ sont données par :

$$\frac{\partial y_{ml}(k)}{\partial \phi_{ij}} = \sum_{d=1}^{N} \left(\frac{\partial y_{ml}(k)}{\partial o_d} \frac{\partial o_d(k)}{\partial \phi_{ij}} \right) = \sum_{d=1}^{N} \left(\frac{\partial y_{ml}(k)}{\partial o_d} Q_{dij}(k) \right) \tag{II.25}$$

Les fonctions de sensibilité $Q_{dij}(k)$ qui calculent les dérivées partielles des sorties du correcteur par rapport aux poids $\phi_{ij}(k)$ sont données par l'équation suivante, $d = 1, ..., N_c$, avec $N_c = 2N$:

$$Q_{dij}(k+1) = e^{-|\tau_c(k)|\Delta T} Q_{dij}(k) + (1 - e^{-|\tau_c(k)|\Delta T})\varphi_d(k)\psi_d(k) \tag{II.26}$$

Pour $d \in \{1, ..., N\}$, les entrées du contrôleur sont définies par $z_d(k) = y_{c_d}(k) - y_d(k)$ et les relations $\varphi_d(k)$ et $\psi_d(k)$ vérifient les équations suivantes :

$$\begin{aligned} \varphi_d(k) &= \tanh'\left(\sum_{h=1}^{N_c} \phi_{dh}(k) o_h(k) + (y_{c_d}(k) - y_d(k))\right) \\ \psi_d(k) &= \left(\delta_i^d o_j(k) + \sum_{h=1}^{N_c} \phi_{dh}(k) Q_{hij}(k) - \frac{\partial y_{md}(k)}{\partial \phi_{ij}}\right) \end{aligned} \tag{II.27}$$

Pour $d \in \{N+1, ..., 2N\}$, $z_d(k) = y_{c_d}(k) - y_d(k)$ alors les relations $\varphi_d(k)$ et $\psi_d(k)$ sont données :

$$\begin{aligned} \varphi_d(k) &= \tanh'\left(\sum_{h=1}^{N_c} \phi_{dh}(k) o_h(k) + y_{c_d}(k)\right) \\ \psi_d(k) &= \left(\delta_i^d o_j(k) + \sum_{h=1}^{N_c} \phi_{dh}(k) Q_{hij}(k)\right) \end{aligned} \tag{II.28}$$

On rappelle à ce niveau que les N premières sorties du correcteur neuronal sont les N commandes appliquées directement au système non linéaire considéré et à l'émulateur multimodèle (si $d \in 1, ..., N$; $o_d(k) = u_d(k)$), les fonctions de sensibilité $J_{ld}(k)$, $(l = 1, ..., 2N,\ d = 1, ..., 2N)$ définies par l'équation (I.16) pour calculer $\frac{\partial y_l(k)}{\partial o_d}$ sont, donc, remplacées par les termes $\frac{\partial y_{ml}(k)}{\partial u_d}$, $(l = 1, ..., N,\ d = 1, ..., N)$ qui sont calculés, en dérivant l'équation qui donne les sorties multimodèle, par l'équation suivante :

$$\frac{\partial y_{ml}(k)}{\partial u_d} = \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k)) \frac{\partial y_{l,i}(k)}{\partial u_d} \tag{II.29}$$

On peut remarquer que les dérivées partielles des sorties du multimodèle sont calculées seulement par rapport aux N premières sorties du correcteur. En effet, les sorties du multimodèle ne dépendent pas des N dernières sorties du correcteur ce qui permet de simplifier le calcul des lois de commande.

Les équations d'adaptation des paramètres du correcteur $\eta_c(k)$ et $\tau_c(k)$ définies respectivement par (I.17) et (I.18) sont remplacées, dans le cas d'un émulateur multimodèle, par les équations suivantes :

$$\Delta\eta_c(k) = \Delta T \sum_{l=1}^{N} \left((y_{c_l}(k-1) - y_l(k-1)) \frac{\partial y_{ml}(k-1)}{\partial \eta_c}\right) \tag{II.30}$$

$$\Delta\tau_c(k) = |\eta_c(k)|\, \Delta T \sum_{l=1}^{N} \left((y_{c_l}(k-1) - y_l(k-1)) \frac{\partial y_{ml}(k-1)}{\partial \tau_c}\right) \tag{II.31}$$

avec :

$$\begin{aligned}\frac{\partial y_{ml}(k)}{\partial \eta_c} &= \sum_{d=1}^{N} \frac{\partial y_{ml}(k)}{\partial o_d} \frac{\partial o_d(k)}{\partial \eta_c} \\ \frac{\partial y_{ml}(k)}{\partial \tau_c} &= \sum_{d=1}^{N} \frac{\partial y_{ml}(k)}{\partial o_d} \frac{\partial o_d(k)}{\partial \tau_c}\end{aligned} \tag{II.32}$$

Pour approcher les dérivées $\frac{\partial o_d(k)}{\partial \eta_c}$ et $\frac{\partial o_d(k)}{\partial \tau_c}$, les fonctions de sensibilité $R_d^{\eta_c}(k)$ et $R_d^{\tau_c}(k)$ sont définies. Ces fonctions sont considérées comme des petites perturbations des états du neurone d'indice d dûes respectivement à des petites variations $\partial\eta_c$ et $\partial\tau_c$ [10]. Pour des raisons de simplification, on considère $R_d^{\eta_c}(k) = R_d^{\tau_c}(k) = R_d(k)$. Ces fonctions sont calculées à partir de la dérivée partielle de l'équation (I.8). Elles sont, alors, données par la relation suivante, pour $d = 1, ..., N_c$:

$$R_d(k+1) = e^{-|\tau_c(k)|\Delta T} R_d(k) + (1 - e^{-|\tau_c(k)|\Delta T})\varphi_d(k)\chi_d(k) + \frac{\varepsilon_c}{|\tau_c(k)|} \tag{II.33}$$

où ε_c est le paramètre qui assure le démarrage et l'évolution autonome de l'algorithme à partir des conditions initiales nulles.
Pour $d \in \{1, ..., N\}$, $z_d(k) = y_{c_d}(k) - y_d(k)$ et $\chi_d(k)$ est calculé comme suit :

$$\begin{aligned}\varphi_d(k) &= \tanh'\left(\sum_{h=1}^{N_c} \phi_{dh}(k) o_h(k) + (y_{c_d}(k) - y_d(k))\right) \\ \chi_d(k) &= \sum_{h=1}^{N_c} \phi_{dh}(k) R_h(k) - \frac{\partial y_{md}(k)}{\partial \eta_c}\end{aligned} \tag{II.34}$$

Pour $d \in \{N+1, ..., 2N\}$, $z_d(k) = y_{c_d}(k)$ et :

$$\begin{aligned}\varphi_d(k) &= \tanh'\left(\sum_{h=1}^{N_c} \phi_{dh}(k) o_h(k) + y_{c_d}(k)\right) \\ \chi_d(k) &= \sum_{h=1}^{N_c} \phi_{dh}(k) R_h(k)\end{aligned} \tag{II.35}$$

II.4.2 Simulations numériques

Pour surmonter les problèmes causés par le paramètre de démarrage lors de l'utilisation d'un émulateur neuronal en boucle fermée, l'émulateur multimodèle est utilisé pour la commande des deux systèmes non linéaires considérés précédemment.

II.4.2.1 Commande neuronale basée sur un émulateur multimodèle d'un système non linéaire SISO

Dans cette partie on reprend l'exemple du système non linéaire SISO défini par l'équation (I.24). L'émulateur multimodèle élaboré précédemment (défini par les sous-modèles M_1, M_2 et M_3) est exploité dans le schéma de commande neuronale. Pour une étude comparative entre les émulateurs neuronal et multimodèle dans le cas d'une commande neuronale adaptative, on reprend les mêmes conditions de simulation considérées, dans

le premier chapitre. Une phase de poursuite pour $k \in [1, 2500]$ et, également, une phase de régulation pour $k \in [2500, 5000]$ sont considérées. Au cours de la phase de régulation et pour $k = 3000$ (pendant 500 itérations) une perturbation de type échelon et d'amplitude 10% de la valeur d'entrée du système de commande, est appliquée sur la sortie du système. Les résultats de la commande neuronale basée sur un émulateur multimodèle avec $\varepsilon_c = 135$ sont donnés sur les figures II.14 et II.15. La figure II.14 laisse apparaître de bonnes performances en poursuite. Un rejet de perturbation satisfaisant est aussi enregistré dans la phase de régulation. La figure II.15 confirme que les performances se sont nettement améliorées en utilisant un émulateur multimodèle. En effet l'erreur $e_c(k)$ entre les sorties désirées et réelles, mesurée à l'échelle logarithmique, est dans ce cas inférieure à celle enregistrée quand un émulateur neuronal est utilisé (figure I.9).

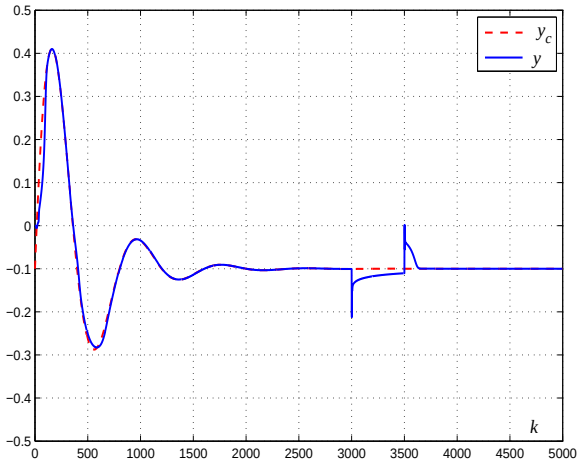


Figure II.14: Variation de la sortie réelle et de la sortie désirée (Commande neuronale adaptative basée sur un émulateur multimodèle $\varepsilon_c = 135$).

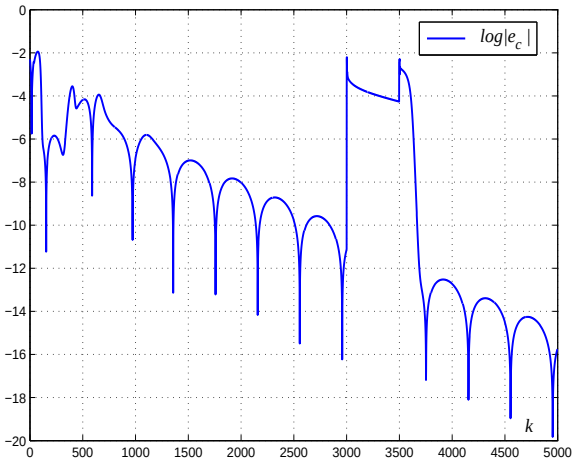


Figure II.15: Erreur de commande à l'échelle logarithmique (Commande neuronale adaptative basée sur un émulateur multimodèle $\varepsilon_c = 135$).

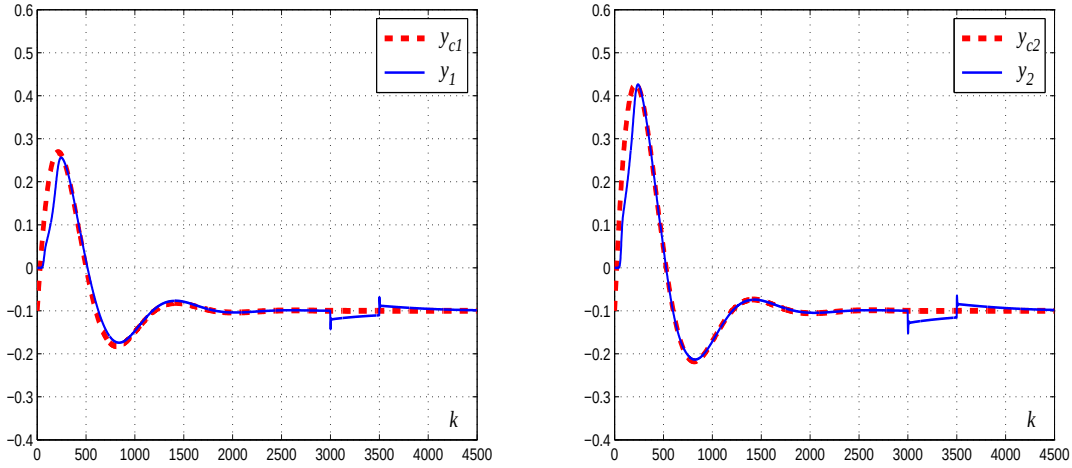
II.4.2.2 Commande neuronale basée sur un émulateur multimodèle d'un système non linéaire MIMO carré

Pour mettre en évidence encore une fois, l'efficacité d'un correcteur neuronal adaptatif basé sur l'émulateur multimodèle, un système non linéaire multivariable, est considéré (relation (I.26)) [10; 9]. Les performances obtenues seront comparées avec celles obtenues dans le cas où un emulateur neuronal (EN) est utilisé. Les deux aspects poursuite et régulation sont aussi pris en compte.

Une première simulation comprend pour les deux sorties, une phase de poursuite pour $k \in [1, 2500]$ et une phase de régulation pour $k \in [2500, 4500]$. Pendant la phase de régulation, des perturbations de type échelon sont appliquées sur les sorties du système à partir de $k = 3000$ jusqu'à $k = 3500$. Le terme de démarrage ε_c est choisi arbitrairement. Pour $\varepsilon_c = 4$, on obtient les résultats donnés par la figure II.16. Cette figure montre que de bonnes performances sont obtenues dans la phase de poursuite. Un rejet de perturbation satisfaisant, est également, enregistré dans la phase de régulation.

Afin de comparer les performances des deux correcteurs neuronaux basés respectivement sur un émulateur multimodèle et un émulateur neuronal, les erreurs MSE_l $(l = 1, 2)$ et les erreurs quadratiques moyennes normalisées $NMSE_l$ $(l = 1, 2)$, définies pour chaque sortie du système par (II.36), sont calculées. Le tableau II.3 résume les valeurs obtenues dans les deux cas.

$$NMSE_l = \frac{\sum_{k=1}^{N_H} \left(y_{c_l}(k) - y_l(k)\right)^2}{\sum_{k=1}^{N_H} \left(y_{c_l}(k)\right)^2} \tag{II.36}$$



(a) Première sortie du système

(b) Deuxième sortie du système

Figure II.16: Variation des sorties réelles et désirées (commande neuronale adaptative basée sur un émulateur multimodèle $\varepsilon_c = 4$).

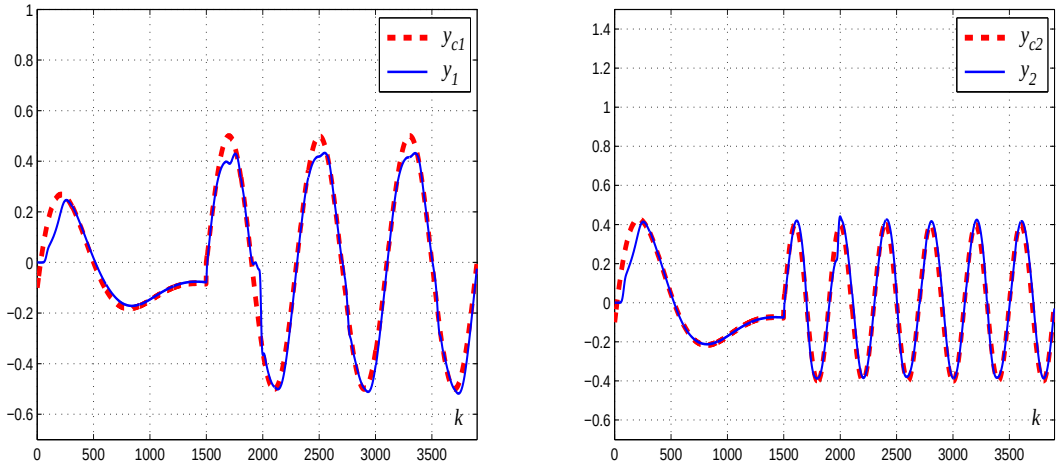
	Emulateur Multimodèle ($\varepsilon_c = 4$)	Emulateur Neuronal ($\varepsilon_c = 4$, $\varepsilon_e = 0.4$)
MSE_1	$4.55\,10^{-4}$	$8.66\,10^{-2}$
MSE_2	$7.32\,10^{-4}$	$7.24\,10^{-2}$
$NMSE_1$	0.0324	6.1725
$NMSE_2$	0.0346	3.4169

Tableau II.3: Les erreurs MSE et $NMSE$ calculées pour les deux sorties du système (schémas de commande basés sur les émulateurs multimodèle et neuronal).

Ce tableau montre une diminution d'environ 99% et 98% des erreurs $NMSE$ calculées respectivement pour la première et la deuxième sortie du système en utilisant l'émulateur multimodèle dans le schéma de commande neuronal. Ces résultats confirment que les performances enregistrées en utilisant un émulateur multimodèle sont relativement meilleures que celles obtenues en utilisant un émulateur neuronal.

Une seconde simulation est considérée avec seulement une phase de poursuite. Dans ce

cas, les sorties désirées présentent une dynamique relativement rapide à partir de l'instant $k = 1500$. Une valeur arbitraire de $\varepsilon_c = 2$ est retenue. Les résultats de simulation sont illustrés sur la figure II.17. Cette figure montre, comme attendu, que les résultats enregistrés dans le cas de la stratégie avancée dans ce travail sont relativement meilleurs par rapport à ceux obtenus avec l'émulateur neuronal. En effet, contrairement à l'émulateur neuronal, l'émulateur multimodèle ne nécessite pas d'adaptation en ligne ce qui permet une adaptation plus rapide aux changements de fréquences des consignes.



(a) Première sortie du système

(b) Deuxième sortie du système

Figure II.17: Variation des sorties réelles et désirées dans le cas d'une dynamique rapide des sorties désirées ($\varepsilon_c = 2$; émulateur multimodèle).

Le tableau II.4 résume les erreurs MSE_l et $NMSE_l$, $l = 1, 2$ calculées pour ce dernier cas de commande. Ce tableau montre une diminution d'environ 98% et 99% des erreurs $NMSE$ calculées respectivement pour la première et la deuxième sortie du système en exploitant l'émulateur multimodèle dans le schéma de commande neuronale. Ces résultats confirment encore une fois les conclusions avancées précédemment.

	Emulateur Multimodèle ($\varepsilon_c = 2$)	Emulateur Neuronal ($\varepsilon_c = 2$, $\varepsilon_e = 0.4$)
MSE_1	$2\,10^{-3}$	$1.89\,10^{-1}$
MSE_2	$2.1\,10^{-3}$	$2.79\,10^{-1}$
$NMSE_1$	0.0236	2.2155
$NMSE_2$	0.0296	4.2351

Tableau II.4: Les erreurs MSE et $NMSE$ calculées pour les deux sorties du système (schémas de commande basés sur un émulateur neuronal et multimodèle : cas d'une dynamique rapide des sorties désirées).

Les résultats de simulation confirment que l'utilisation d'un émulateur multimodèle permet de résoudre les problèmes causés par un mauvais choix du paramètre de démarrage (ε_e) de l'émulateur neuronal et d'éviter les efforts nécessaires pour la recherche d'une valeur convenable de ce paramètre. L'émulateur proposé permet, aussi, une adaptation plus rapide aux changements de fréquences des consignes. Cependant, il convient de noter à ce stade qu'il y a deux facteurs qui rendent l'identification de l'émulateur multimodèle difficile. En effet, un nombre relativement réduit de données et un signal d'entrée relativement peu riche peut dégrader la qualité de modélisation ainsi que les performances en boucle fermée. Ce problème peut être aisément résolu par l'utilisation d'un signal d'excitation suffisamment riche pour exciter les différentes zones de fonctionnement. Ce signal doit aussi être choisi judicieusement de signal d'excitation en fonction de trajectoires de consigne souhaitées.
De plus, il est à noter que la méthode utilisée pour le partitionnement de l'espace de fonctionnement a une influence sur les capacités de l'émulateur multimodèle à approcher avec précision le comportement dynamique des systèmes non linéaires MIMO. La méthode basée sur l'exploitation des caractéristiques statiques nécessite des efforts supplémentaires pour trouver les valeurs convenables des paramètres de synthèse des fonctions de pondération. Pour résoudre ce problème, on propose dans la suite une méthode de classification qui permet la determination systématique de ces paramètres.

II.5 Détermination systématique des paramètres de synthèse de l'émulateur multimodèle pour les systèmes non linéaires multivariables

Dans cette partie on présente une méthode qui permet le partitionnement systématique de la totalité de domaine de fonctionnement du système non linéaire MIMO par exploitation d'une procédure de classification. En considérant les données numériques qui vont être utilisés pour l'identification des multimodèles, cette procédure permet de sélectionner systématiquement les centres relatifs à toutes les zones de fonctionnement et permet de calculer les fonctions de pondérations (équations (II.10) et (II.11)) en utilisant la méthode de classification de Chiu [16]. Les centres sélectionnés sont utilisés par la suite pour le calcul des dispersions en se basant sur un calcul des distances qui séparent les centres voisins.
Après la determination des centres et des dispersions, le partitionnement du domaine de fonctionnement est alors effectué. L'identification des paramètres des modèles partiels constituant chaque base B_l est réalisée en exploitant ces paramètres de synthèse et en mettant en oeuvre la procédure d'identification.

II.5.1 Procédure de classification : détermination des centres des fonctions de pondération

La méthode de Chiu appelée aussi méthode des montagnes [70; 71] consiste à sélectionner les centres des classes parmi l'ensemble de données persistantes d'identification qui doit être suffisamment riches pour couvrir la totalité du domaine de fonctionnement du système non linéaire multivariable considéré. Dans ce travail, on utilise cette méthode pour classifier un ensemble de vecteur de régression défini pour chaque sortie du système multivariable par $\vartheta_{l,j} = [y_l(j), y_l(j-1), u_1(j-1), ..., u_e(j-1), ..., u_{N_{IN}}(j-1)]^T, j = 1, ..., N_H$, $l = 1, ..., N_{OUT}$ et $e = 1, ..., N_{IN}$.
On commence par associer à chaque vecteur de regression $\vartheta_{l,j}$, un premier potentiel $Pot_{l,j}(1)$ défini par l'expression suivante :

$$Pot_{l,j}(1) = \sum_{h=1}^{N_H} e^{\left(\frac{-4\left\|\vartheta_{l,j}-\vartheta_{l,h}\right\|^2}{r_{l,a}^2}\right)} \tag{II.37}$$

où $r_{l,a}$ est un scalaire positif qui gère la décroissance du potentiel. En effet, chaque potentiel $Pot_{l,j}(1)$ est calculé en fonction des distances entre le vecteur de regression correspondant $\vartheta_{l,j}$ et tous les autres vecteurs de regression. Le potentiel décroît, alors, exponentiellement lorsque $\vartheta_{l,h}$ s'éloigne de $\vartheta_{l,j}$.
Après le calcul de tous les potentiels, le premier vecteur "centre de classe" noté $\vartheta^*_{l,1}$ est

sélectionné : c'est le vecteur dont le potentiel donné par l'équation II.37 est maximal. Ce potentiel maximal est noté $Pot^*_{l,1}$.

Ensuite et pour éviter de sélectionner le premier centre et ses voisins comme un autre centre de classe, la procédure de classification modifie la valeur de tous les potentiels en fonction de la distance entre le vecteur régression courant et celui sélectionné comme un premier centre, par la formule suivante :

$$Pot_{l,j}(2) = Pot_{l,j}(1) - Pot^*_{l,1} e^{\left(\frac{-4\left\|\vartheta_{l,j}-\vartheta^*_{l,1}\right\|^2}{r^2_{l,b}}\right)} \tag{II.38}$$

$r_{l,b}$ est un paramètre positif choisi strictement supérieur à $r_{l,a}$ pour favoriser l'opération relative à la sélection d'autres classes nettement différentes de la première et de ses proches voisines. En effet, les vecteurs de regression proches du premier centre sélectionné, auront des potentiels considérablement réduits et ils sont, donc, peu susceptibles d'être choisis comme un nouveau centre.

A partir de la deuxième itération, la procédure de sélection peut être généralisée comme suit :
On sélectionne de la même façon le ($i^{ème} - 1$) vecteur "centre" noté $\vartheta^*_{l,i-1}$ avec le maximum de potentiel $Pot^*_{l,i-1}$, et on modifie les potentiels de chaque vecteur régression par l'expression suivante :

$$Pot_{l,j}(i) = Pot_{l,j}(i-1) - Pot^*_{l,i-1} e^{\left(\frac{-4\left\|\vartheta_{l,j}-\vartheta^*_{l,i-1}\right\|^2}{r^2_{l,b}}\right)} \tag{II.39}$$

Le choix entre l'arrêt de la procédure de sélection ou la sélection de nouveaux vecteurs des centres de classes obéit à un ensemble de tests où deux paramètres positifs ε_1 et ε_2 qui obéissent à l'inégalité $\varepsilon_1 > \varepsilon_2$ sont introduits.
L'algorithme suivant résume cet ensemble de tests qui doit être vérifié à chaque étape :
Algorithme :

Si $Pot^*_{l,j} > \varepsilon_1 Pot^*_{l,1}$

La sélection est autorisée : accepter $\vartheta^*_{l,j}$ comme un centre de classe et continuer,

Sinon si $Pot^*_{l,j} < \varepsilon_2 Pot^*_{l,1}$

La sélection achevée : rejeter $\vartheta^*_{l,j}$ et la procédure de classification est terminée,

Sinon

Définir $d_{\min}$: la plus petite distance entre $\vartheta^*_{l,j}$ et tous les centres déjà sélectionnés.

Si $(\frac{d_{\min}}{r_{l,a}} \geq 1 - \frac{Pot^*_{l,j}}{Pot^*_{l,1}})$

Accepter $\vartheta^*_{l,j}$ comme un centre de classe et continuer,

Sinon

Rejeter $\vartheta^*_{l,j}$ et mettre le potentiel correspondant à 0.
Sélectionner le vecteur de regression ayant le prochain maximum des potentiels comme un nouveau $\vartheta^*_{l,j}$ et refaire le test

Fin Si

Fin Si.

La distance $d_{\min}$ est définie par l'équation suivante :

$$d_{\min} = Min\left(Min\left(\left|\vartheta^*_{l,j} - \vartheta^*_{l,1}\right|\right), ..., Min\left(\left|\vartheta^*_{l,j} - \vartheta^*_{l,j-1}\right|\right)\right) \tag{II.40}$$

où $\vartheta^*_{l,j}$ est le vecteur de régression centre de classe courant et $\vartheta^*_{l,1}, \vartheta^*_{l,2}, \ldots, \vartheta^*_{l,j-1}$ sont les derniers vecteurs de régression centres sélectionnés.
Une fois que tous les vecteurs de régression "centres de classes" sont sélectionnés, et selon la variable de décision à prendre en considération, les centres des fonctions de pondération sont choisis. Comme on l'a mentionné précédemment, le vecteur d'entrée servira de variables de décision ($\xi(k) = U(k)$) et les centres $c_{l,ie}$ des fonctions de pondération sont, alors, les N_{IN} derniers éléments des vecteurs de regression "centres de classes" :

$$c_{l,ie} = u^*_e,\ l = 1, ..., N_{OUT},\ e = 1, ..., N_{IN} \text{ et } i = 1, ..., N_{ml}$$

où N_{ml} est le nombre de sous modèles (appartenant à chaque base B_l) générés systématiquement : c'est le nombre de vecteurs de regression sélectionnés comme vecteurs des centres de classe.
Il reste maintenant à déterminer les dispersions correspondantes.

II.5.2 Calcul des dispersions des fonctions de pondération

Les dispersions $\sigma_{l,ie}$ correspondantes aux centres sélectionnés sont calculées en fonction des distances séparant ces centres, moyennant l'une des expressions suivantes :
Expression 1 :
Les dispersions $\sigma_{l,ie}$ peuvent être ajustées selon la moyenne des distances aux n plus proches voisins [46] :

$$\sigma_{l,ie} = \alpha_l \frac{1}{n_l} \sum_{k=1}^{n} |c_{l,ie} - c_{l,ke}| \tag{II.41}$$

$c_{l,ie}$ est le centre courant, $c_{l,ke}$ désignent les n_l plus proches centres voisins à $c_{l,ie}$. α_l un coefficient de réglage qui définit le degré de recouvrement entre les fonctions de pondération.
Expression 2 :

Les dispersions $\sigma_{l,ie}$ sont choisies proportionnelles à la distance qui sépare le centre $c_{l,ie}$ et le plus proche centre voisin [60] :

$$\sigma_{l,ie} = \frac{1}{\alpha_l} \min_{k=1,\dots,n} (|c_{l,ie} - c_{l,ke}|) \tag{II.42}$$

où α_l est un facteur positif définissant le degré de recouvrement entre les fonctions de pondération.

II.5.3 Simulations numériques

Afin de mettre en évidence l'intérêt et l'efficacité de la méthode de génération systématique des paramètres de synthèse de l'émulateur multimodèle, deux exemples de simulation sont présentés dans cette section. Une comparaison entre l'approche classique exploitant les caractéristiques statiques et la technique proposée est présentée pour des systèmes non linéaires multivariables carrés.
Pour les deux exemples considérés, l'élaboration des émulateurs multimodèles est réalisée en temps différé et par recours au critère global (équation (II.14)). Les entrées du système considéré, qui servent de variables de décision ($\xi(k) = U(k)$), sont constituées par la concaténation de créneaux d'amplitudes variables ($u_l(k) \in [-1, 1]$, $l = 1, ..., N_{IN}$).

II.5.3.1 Un émulateur multimodèle à paramètres de synthèse optimisés pour un système non linéaire MIMO carré

Le système non linéaire multivariable, considéré dans ce cas, est un système à deux entrées et deux sorties et est défini comme suit :

$$\begin{aligned} y_1(k) &= 0.2y_1(k-1) + u_2(k-2)^3 + u_1(k-2)^2 - 0.2y_1(k-1)u_1(k-2) - 0.1u_2(k-1) \\ y_2(k) &= 0.05y_2(k-1) + u_1(k-2)^3 + u_2(k-1)^2 + 0.4y_2(k-2)u_2(k-1) \end{aligned} \tag{II.43}$$

L'élaboration des deux émulateurs multimodèles en utilisant les deux approches de décomposition de l'espaces de fonctionnement (classique et avancée) est réalisée en utilisant la même base des données d'identification (Entrée-Sorties). La méthode classique, utilisée précédemment, exploite les caractéristiques statiques du système non linéaire. En effet, à chaque zone linéaire est associée un centre $c_{l,ie}$. Le nombre de sous-modèles et les fonctions de pondérations associées sont obtenus par une combinaison entre les centres obtenus (équations II.10 et II.11). Les dispersions associées sont, aussi, choisies, en se basant sur ces mêmes caractéristiques statiques.
Les caractéristiques statiques du système non linéaire multivariable données sur la figure II.18 permettent d'élaborer, pour chaque sortie , un multimodèle MISO constitué de six sous-modèles.

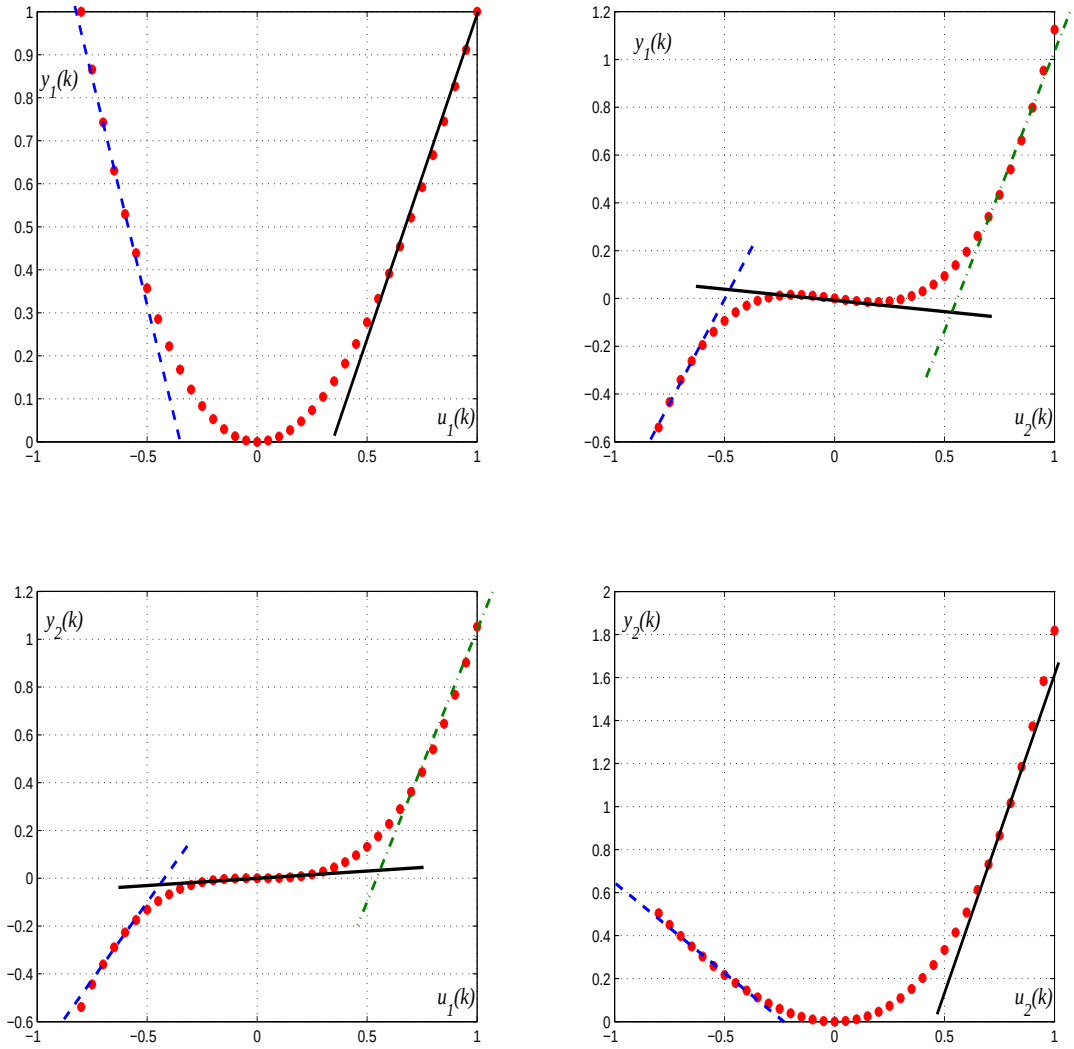
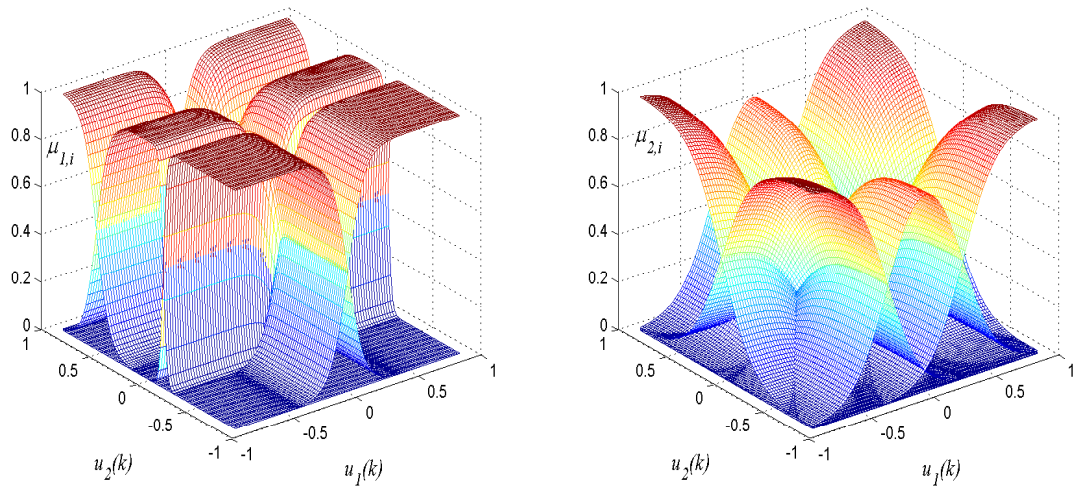


Figure II.18: Les caractéristiques statiques du système non linéaire MIMO carré.

La mise en oeuvre de la procédure de classification pour générer l'ensemble de vecteurs de regression $\vartheta_{l,j} = [y_l(j), y_l(j-1), u_1(j-1), u_2(j-1)]^T$, $l = 1, 2$ donne naissance à trois classes pour chaque sortie du système. Une combinaison entre les centres obtenus respectivement par rapport à la première et à la deuxième commande du système permet de représenter chaque sortie du système par une base de neuf sous-modèles (équations II.10 et II.11). Dans ce cas, on peut également remarquer que la méthode systématique génère des modèles additionnels par rapport à la méthode classique. Ceci permet d'améliorer la précision d'émulation. Les centres et les dispersions obtenus par les deux méthodes de décomposition de l'espace de fonctionnement sont résumés sur le tableau II.5. Les figures II.19 et II.20 donnent respectivement les fonctions de pondération obtenues en utilisant les paramètres de synthèse présentés sur le tableau II.5.

	Méthode basée sur les caractéristiques statiques	Méthode basée sur la procédure de classification
$c_{1,ie}$ et $\sigma_{1,ie}$	$c_{1,11} = -0.6$, $\sigma_{1,11} = 0.2$ $c_{1,21} = 0.6$, $\sigma_{1,21} = 0.2$ $c_{1,12} = -0.8$, $\sigma_{1,12} = 0.1$ $c_{1,22} = 0$, $\sigma_{1,22} = 0.1$ $c_{1,32} = 0.8$, $\sigma_{1,32} = 0.1$	$c_{1,11} = -0.6657$, $\sigma_{1,11} = 1.0088$ $c_{1,21} = -0.0352$, $\sigma_{1,21} = 1.0606$ $c_{1,31} = 0.6601$, $\sigma_{1,31} = 1.1125$ $c_{1,12} = -0.5783$, $\sigma_{1,12} = 0.5252$ $c_{1,22} = 0.0053$, $\sigma_{1,22} = 0.5170$ $c_{1,32} = 0.5706$, $\sigma_{1,32} = 0.5088$
$c_{2,ie}$ et $\sigma_{2,ie}$	$c_{2,11} = -0.6$, $\sigma_{2,11} = 0.2$ $c_{2,21} = 0$, $\sigma_{2,21} = 0.2$ $c_{2,31} = 0.6$, $\sigma_{2,31} = 0.2$ $c_{2,12} = -0.5$, $\sigma_{2,12} = 0.2$ $c_{2,22} = 0.5$, $\sigma_{2,22} = 0.2$	$c_{2,11} = -0.5814$, $\sigma_{2,11} = 0.2185$ $c_{2,21} = -0.0352$, $\sigma_{2,21} = 0.2505$ $c_{2,31} = 0.6710$, $\sigma_{2,31} = 0.2825$ $c_{2,12} = -0.5626$, $\sigma_{2,12} = 0.2839$ $c_{2,22} = 0.0053$, $\sigma_{2,22} = 0.2898$ $c_{2,32} = 0.5966$, $\sigma_{2,32} = 0.2957$

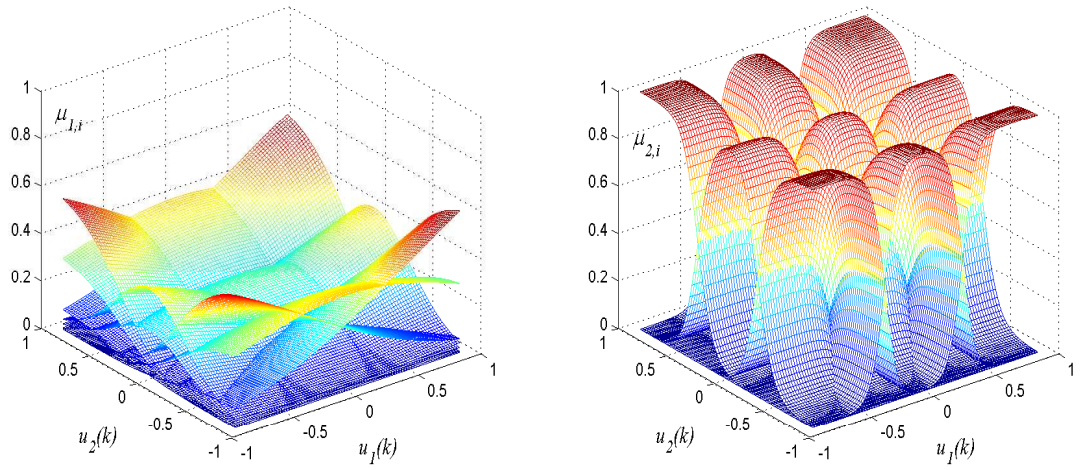
Tableau II.5: Les dispersions et les centres déterminés systématiquement et par l'approche classique.



(a) Les fonctions de pondération pour le premier multimodèle

(b) Les fonctions de pondération pour le deuxième multimodèle

Figure II.19: Les fonctions de pondération déterminées par exploitation des caractéristiques statiques du système non linéaire MIMO carré.

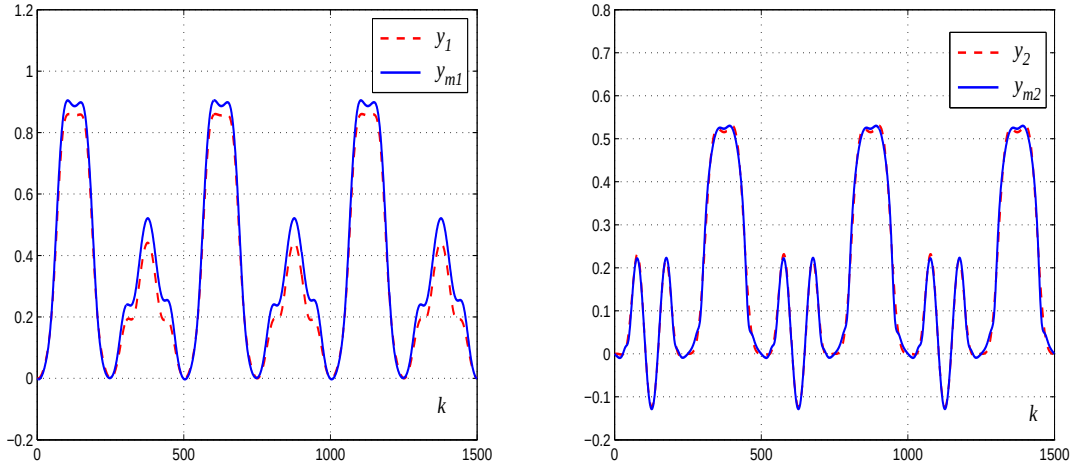


(a) Les fonctions de pondération pour le premier multimodèle

(b) Les fonctions de pondération pour le deuxième multimodèle

Figure II.20: Les fonctions de pondération déterminées systématiquement par classification (système non linéaire MIMO carré).

La figure II.21 illustre les résultats de validation du multimodèle obtenu en utilisant les paramètres optimisés. Cette figure, donnant les évolutions des sorties multimodèle et réelle, montre que l'émulateur multimodèle peut décrire avec précision suffisante le comportement du système non linéaire multivariable considéré.



(a) Première sortie du système

(b) Deuxième sortie du système

Figure II.21: Variation des sorties du système non linéaire MIMO carré et celles de l'émulateur multimodèle (paramètres de synthèse optimisés).

En conservant les mêmes conditions de validation, les deux indices de validation MSE_l et VAF_l, $l = 1, 2$ sont calculés pour les multimodèles obtenus en utilisant les deux méthodes de décomposition de l'espace de fonctionnement. Les résultats obtenus dans le cas de la méthode proposée sont nettement meilleurs relativement au cas où la méthode classique est utilisée. Le tableau II.6 , confirme les constatations précitées.

	Méthode basée sur les caractéristiques statiques	Méthode basée sur la procédure de classification
MSE_1	$1.8\,10^{-2}$	$2\,10^{-3}$
MSE_2	$1.1\,10^{-2}$	$3.46\,10^{-4}$
VAF_1	97.88%	99.15%
VAF_2	97.15%	99.38%

Tableau II.6: Les indices MSE_l et VAF_l $(l = 1, 2)$ calculés pour les deux multimodèles obtenus respectivement par les deux méthodes de décomposition de l'espace de fonctionnement (système non linéaire MIMO carré).

II.6 Conclusion

Principalement deux contributions sont présentées dans ce chapitre. La première consiste à proposer un émulateur multimodèle découplé pour la représentation et la commande neuronale adaptative des systèmes non linéaires multivariables. En utilisant cet émulateur, chaque sortie du système est prédite en utilisant une bibliothèque de modèles qui résulte d'une procédure d'identification hors ligne du système multivariable non linéaire par une approche multimodèle. Les performances en boucle ouverte et en boucle fermée, offertes en utilisant l'émulateur proposé sont comparées à celles obtenues avec un émulateur neuronal. L'avantage principal du schéma utilisant un émulateur multimodèle est de réduire considérablement les équations de l'émulateur neural, qui deviennent de plus en plus complexes dans le cas MIMO, en l expressions des variations des sorties du système à commander par rapport aux entrées. Aucun paramètre d'initialisation, et aucune adaptation en ligne ne sont nécessaires. Cela permet de réduire la complexité de calcul du système de commande neuronale adaptative multivariable et permet, par conséquent, de résoudre le problème lié au choix de paramètre de démarrage de l'émulateur neuronal. Les résultats des simulations montrent clairement que l'émulateur multimodèle proposé conduit à de bonnes performances en boucle fermée relativement au cas où un émulateur neuronal est appliqué.

La deuxième contribution est l'élaboration d'une méthode basée sur la classification de données d'identification pour la décomposition de l'espace de fonctionnement du système et la génération systématique des fonctions de pondération. Cette approche systématique présente un avantage important qui consiste à éviter tout les efforts nécessaires pour trou-

ver des valeurs optimales des paramètres de synthèses en utilisant la méthode classique basée sur l'exploitation des caractéristiques statiques du système non linéaire pour assurer la décomposition de son espace de fonctionnement. Plusieurs simulations numériques ont pu mettre en évidence que l'émulateur multimodèle synthétisé à la base d'une partitionnement systématique de l'espace de fonctionnement conduit, dans ce cas, à de bonnes performances relativement au cas où la méthode classique est utilisée.

Chapitre III

Une commande neuronale adaptative pour les systèmes non linéaires SIMO

Sommaire

III.1 Introduction

Face aux restrictions environnementales, opérationnelles et énergétiques, qui ont contribué à l'augmentation de la complexité des systèmes industriels, la conception d'une commande efficace pour les systèmes non linéaires multivariables est une question intéressante en théorie de commande non linéaire. En particulier, la commande des systèmes multivariables SIMO (Single Input-Multi Output) a émergé comme un domaine de recherche intéressant dans les dernières années. Ces systèmes présentent une classe importante des systèmes non linéaires et apparaissent pour représenter une large classe de systèmes industriels. En fait, les systèmes SIMO sont rencontrés dans un large éventail d'applications, notamment la robotique [24; 52; 58], les systèmes électriques [15; 35] et les systèmes hydrauliques [34].

En raison de leur large gamme d'applications, la commande des systèmes SIMO reste un défi en termes de représentation, d'observation et de commande des systèmes non linéaires. Un système SIMO ne dispose que d'un seul actionneur avec deux ou plusieurs sorties, le correcteur à concevoir doit fournir une seule commande pour manipuler plusieurs sorties en même temps. Cette restriction provoque de grandes difficultés pratiques lors du traitement de ces systèmes. En effet, vu la dynamique des systèmes SIMO, certaines conditions d'inversibilité ne sont pas satisfaites. De plus, ces systèmes ne sont pas complètement linéarisables par bouclage statique. Par la suite, de nombreuses méthodes récentes et traditionnelles de commande non linéaire ne sont pas directement applicables aux systèmes SIMO.

De nombreuses méthodes de commande ont été étudiées pour surmonter ces problèmes, telle que la commande par retour d'état [24; 12; 15], la commande par mode glissant [58], la commande par modèle interne [34] et la commande basée sur la linéarisation partielle par bouclage statique [63]. Récemment, une commande adaptative utilisant deux réseaux de neurones pour la conception du correcteur a été proposée pour une classe de systèmes non linéaires SIMO [35].

Dans le présent chapitre, on propose un schéma de commande neuronale adaptative non carré dédié aux systèmes non linéaires SIMO. Ce schéma est inspiré de celui proposé précédemment pour la commande des systèmes non linéaires MIMO. Cette nouvelle contribution consiste à construire une famille de correcteurs neuronaux carrés, les paramètres de chaque correcteur partiel sont ajustés en ligne en minimisant une erreur de commande correspondante pour fournir un ensemble de commandes partielles. Une autre contribution consiste à proposer une nouvelle technique pour l'estimation en ligne des validités de commandes locales qui fusionnent ces commandes pour obtenir la commande globale à

appliquer effectivement au système SIMO. Comme l'adaptation des poids des correcteurs partiels nécessite l'utilisation d'un émulateur, on propose dans une première partie de ce chapitre d'utiliser un émulateur neuronal non carré pour représenter la dynamique des systèmes étudiés. Dans une deuxième partie, on propose un émulateur multimodèle non carré pour remplacer l'émulateur neuronal.

III.2 Une commande neuronale adaptative basée sur un émulateur neuronal

Le schéma de commande adaptative neuronale proposé pour les systèmes non linéaires multivariables SIMO est composé d'un émulateur non carré (qui peut être neuronal ou multimodèle) et d'un ensemble de correcteurs neuronaux partiels carrés. Le schéma général de la stratégie de commande proposée est donné sur la figure III.1.

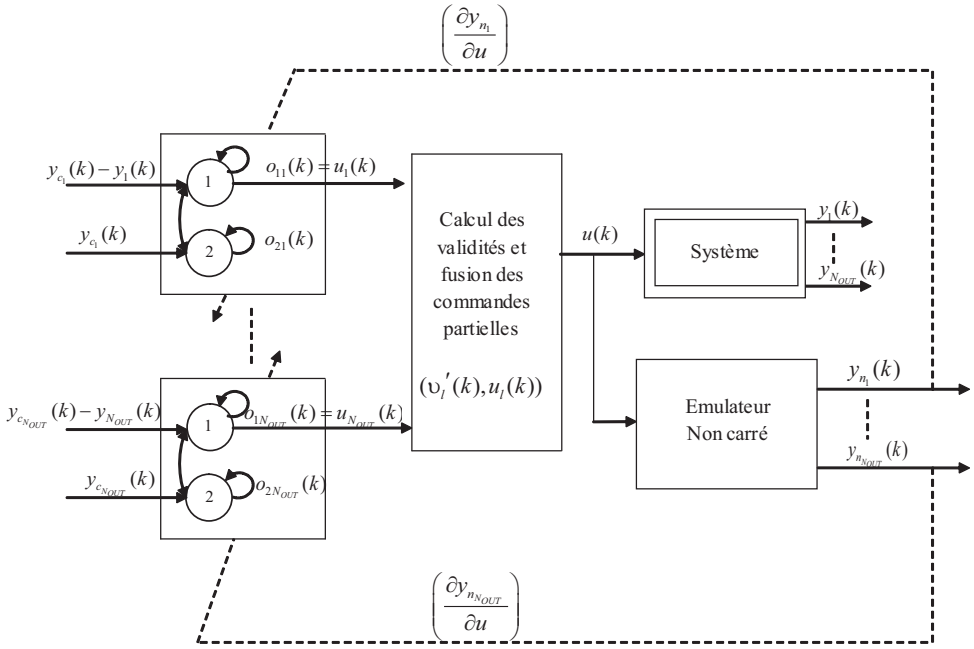


Figure III.1: Structure du système de commande adaptatif neuronal non carré pour les systèmes non linéaires multivariables SIMO.

Les systèmes SIMO étudiés ont une seule entrée et plusieurs sorties, le nombre d'entrées est alors $N_{IN} = 1$ et le nombre de sorties est égal à N_{OUT}. On définit $y_l(k)$ et $y_{n_l}(k)$, $(l = 1, ..., N_{OUT})$ respectivement les sorties réelles du système SIMO considéré et les sorties estimées par l'émulateur. $u(k)$ est la seule entrée du système.
L'idée de base de la structure de commande proposée est de développer pour chaque sortie, un correcteur neuronal partiel totalement déconnecté de ses voisins. Chaque correcteur

partiel fourni une commande adaptative partielle en minimisant une erreur quadratique instantanée entre la sortie $y_l(k)$ du système et la sortie désirée correspondante $y_{c_l}(k)$. Par conséquent, le correcteur neuronal non carré est formé par N_{OUT} correcteurs partiels notés CN_l avec $l = 1, ..., N_{OUT}$. Le signal de commande global $u(k)$, effectivement appliqué au système, est une fusion des commandes partielles $u_l(k)$ pondérées par les degrés de validité respectifs $\upsilon_l{}'(k)$. Ces degrés de validité sont calculés par un bloc de décision qui évalue, en fonction des erreurs $(y_{c_l}(k) - y_l(k))$ correspondantes, les degrés de contributions des différents correcteurs pour garantir les performances souhaitées en boucle fermée.

Afin de fournir les lois de commandes partielles, les paramètres de chaque correcteur partiel doivent être mis à jour. Cette adaptation nécessite l'évaluation de la dérivée partielle de la sortie correspondante par rapport à la commande $u(k)$, ceci justifie la nécessité d'utiliser un émulateur pour avoir une expression analytique de cette dérivée partielle.
Dans cette section, un émulateur neuronal non carré est utilisé, la structure de cet émulateur, la structure des correcteurs partiels et la stratégie proposée pour le calcul des degrés de validité sont développées.

III.2.1 Un emulateur neuronal non carré : structure et algorithme d'adaptation

L'émulateur neuronal proposé dans cette partie a une structure non carrée totalement connectée représentée sur la figure III.2.

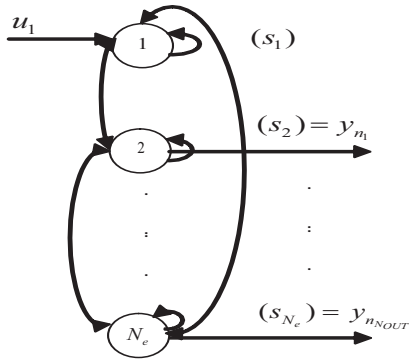


Figure III.2: Structure totalement connectée de l'émulateur neuronal non carré ($N_e = N_{OUT} + 1$).

Le nombre de neurones de l'émulateur ne dépend que du nombre des entrées et des sorties du système, ainsi, l'émulateur neuronal est construit avec un nombre de neurones $N_e = N_{OUT}+1$. Ces neurones sont séparés en deux groupes : le premier groupe comporte un seul neurone d'entrée relié à l'entrée du système $u(k)$ et le second groupe comporte N_{OUT}

neurones de sorties qui donnent les sorties estimées y_{n_l}, $l = 1, ..., N_{OUT}$. Par conséquent, l'activation de ces neurones évolue selon l'équation suivante :

$$\begin{aligned} s_1(k+1) &= e^{-|\tau_e(k)|\Delta T} s_1(k) + \left(1 - e^{-|\tau_e(k)|\Delta T}\right) \tanh\left(\sum_{j=1}^{N_e} w_{1j}(k) s_j(k) + u(k)\right) \\ s_i(k+1) &= e^{-|\tau_e(k)|\Delta T} s_i(k) + \left(1 - e^{-|\tau_e(k)|\Delta T}\right) \tanh\left(\sum_{j=1}^{N_e} w_{ij}(k) s_j(k)\right), \ i = 2, ..., N_e \end{aligned} \tag{III.1}$$

où $w_{ij}(k)$ et $1/|\tau_e(k)|$ représentent respectivement les poids de connexion du neurone j vers le neurone i et la constante de temps des neurones. $s_i(k)$ est l'état du $i^{\text{ème}}$ neurone. Pour l'adaptation des poids et des paramètres de l'émulateur, l'algorithme RTRL est utilisé de la même manière pour l'adaptation des paramètres de l'émulateur neuronal de structure carrée (présenté dans le premier chapitre figure I.2). A chaque instant k, on définit l'erreur quadratique instantanée d'émulation par :

$$e_e(k) = \frac{1}{2} \sum_{l=1}^{N_{OUT}} \left(y_{n_l}(k) - y_l(k)\right)^2 \tag{III.2}$$

Les poids du réseau émulateur s'adaptent d'une manière itérative en utilisant le gradient de l'erreur d'émulation selon l'équation :

$$\Delta w_{ij}(k) = -\left|\eta_e(k)\right| \Delta T \sum_{l=1}^{N_{OUT}} \left((y_{n_l}(k-1) - y_l(k-1)) \frac{\partial y_{n_l}(k-1)}{\partial w_{ij}} \right) \tag{III.3}$$

où $\eta_e(k)$ est le facteur d'adaptation. Le calcul des dérivées partielles des sorties $y_{n_l}(k)$ de l'émulateur neuronal, qui remplacent les sorties du système commandé, par rapport aux poids w_{ij}, $(i,j) \in \{1, ..., N_e\}^2$ se fait en utilisant les fonctions de sensibilité notées $P_{hij}(k)$, $h \in \{2, ..., N_e\}$:

$$\begin{aligned} P_{1ij}(k+1) &= e^{-|\tau_e(k)|\Delta T} P_{1ij}(k) \\ &+ (1 - e^{-|\tau_e(k)|\Delta T}) \left(\tanh'\left(\sum_{m=1}^{N_e} w_{1m}(k) s_m(k) + u(k)\right)\right) \left(\delta_i^1 s_j(k) + \sum_{m=1}^{N_e} w_{1m}(k) P_{mij}(k)\right) \end{aligned} \tag{III.4}$$

$$\begin{aligned} P_{hij}(k+1) &= e^{-|\tau_e(k)|\Delta T} P_{hij}(k) \\ &+ (1 - e^{-|\tau_e(k)|\Delta T}) \left(\tanh'\left(\sum_{m=1}^{N_e} w_{hm}(k) s_m(k)\right)\right) \left(\delta_i^h s_j(k) + \sum_{m=1}^{N_e} w_{hm}(k) P_{mij}(k)\right) \end{aligned} \tag{III.5}$$

En se basant sur la même méthode utilisée pour l'adaptation des poids de l'émulateur, un algorithme basé sur la minimisation de la fonction coût $e_e(k)$ est utilisé pour l'adaptation des paramètres $\eta_e(k)$ et $\tau_e(k)$ en utilisant les équations suivantes :

$$\Delta \eta_e(k) = -\Delta T \sum_{l=1}^{N_{OUT}} \left((y_{n_l}(k-1) - y_l(k-1)) \frac{\partial y_{n_l}(k-1)}{\partial \eta_e} \right) \tag{III.6}$$

$$\Delta\tau_e(k) = -\left|\eta_e(k)\right| \Delta T \sum_{l=1}^{N_{OUT}} \left((y_{n_l}(k-1) - y_l(k-1)) \frac{\partial y_{n_l}(k-1)}{\partial \tau_e} \right) \tag{III.7}$$

Par analogie aux fonctions de sensibilité $P_{lij}(k)$, un autre ensemble de fonctions de sensibilité $v_l^{\eta_e}(k)$ et $v_l^{\tau_e}(k)$ est introduit pour calculer les dérivées partielles de y_{n_l} respectivement par rapport à η_e et τ_e. Ces fonctions vont évoluer de la même manière à partir des conditions initiales nulles. Donc, et pour des raisons de simplification, on retient $v_l^{\eta_e}(k) = v_l^{\tau_e}(k) = v_l(k)$. Ces fonctions vérifient les équations suivantes pour $h \in \{2, ..., N_e\}$:

$$\begin{aligned} v_1(k+1) &= e^{-|\tau_e(k)|\Delta T} v_1(k) \\ &+ (1 - e^{-|\tau_e(k)|\Delta T}) \left(\tanh' \left(\sum_{m=1}^{N_e} w_{1m}(k) s_m(k) + u(k) \right) \right) \left(\sum_{m=1}^{N_e} (w_{1m}(k) v_m(k)) \right) + \frac{\varepsilon_e}{|\tau_e(k)|} \end{aligned} \tag{III.8}$$

$$\begin{aligned} v_h(k+1) &= e^{-|\tau_e(k)|\Delta T} v_h(k) \\ &+ (1 - e^{-|\tau_e(k)|\Delta T}) \left(\tanh' \left(\sum_{m=1}^{N_e} w_{hm}(k) s_m(k) \right) \right) \left(\sum_{m=1}^{N_e} (w_{hm}(k) v_m(k)) \right) + \frac{\varepsilon_e}{|\tau_e(k)|} \end{aligned} \tag{III.9}$$

Comme dans le cas de l'émulateur neuronal carré, la constante ε_e est choisie arbitrairement pour assurer le démarrage et l'évolution autonome de l'algorithme à partir des conditions initiales nulles pour tous les paramètres du réseau neuronal.

III.2.2 Correcteurs neuronaux partiels : structure et algorithme d'adaptation

Chaque correcteur neuronal partiel CN_l est développé avec un réseau de neurones récurrent totalement connecté et de structure carrée. Ce réseau est, donc, différent de celui utilisé pour l'émulation du système. En effet, chaque correcteur est constitué de deux neurones : l'entrée du premier neurone est l'erreur $y_{c_l}(k) - y_l(k)$ entre la $l^{ième}$ sortie du système et la sortie désirée correspondante, l'entrée du deuxième neurone est la $l^{ième}$ sortie désirée $y_{c_l}(k)$. La sortie du premier neurone est utilisée pour le calcul de la commande et elle représente la loi de commande partielle $u_l(k)$. L'activation des états des neurones de chaque correcteur partiel $o_{d_l}(k)$ pour $(d = 1, 2)$ et $(l = 1, ..., N_{OUT})$ est calculée par les équations suivantes :

$$o_{d_l}(k+1) = e^{-\left|\tau_{c_l}(k)\right|\Delta T} o_{d_l}(k) + \left(1 - e^{-\left|\tau_{c_l}(k)\right|\Delta T}\right) D_{d_l}(k) \tag{III.10}$$

avec :

$$\begin{aligned} D_{1_l}(k) &= \tanh\left(\sum_{s=1}^{2} \phi_{1s_l}(k) o_{s_l}(k) + y_{c_l}(k) - y_l(k)\right) \\ D_{2_l}(k) &= \tanh\left(\sum_{s=1}^{2} \phi_{2s_l}(k) o_{s_l}(k) + y_{c_l}(k)\right) \end{aligned} \tag{III.11}$$

où $\phi_{rs_l}(k)$ $((r,s) \in (1,2)^2)$ et $1/\tau_{c_l}(k)$ sont respectivement les poids de connexion du neurone s vers le neurone r et la constante de temps de chaque correcteur partiel.
Pour mettre à jour les poids des correcteurs, un algorithme inspiré de la descente du gradient est utilisé. Pour chaque correcteur partiel, l'objectif est de minimiser l'erreur quadratique instantanée entre la $l^{ième}$ sortie désirée et la $l^{ième}$ sortie mesurée définie comme suit :

$$e_{c_l}(k) = \frac{1}{2}(y_{c_l}(k) - y_l(k))^2 \tag{III.12}$$

L'adaptation des poids de chaque correcteur partiel NC_l est donc donnée par l'équation (III.13) :

$$\Delta\phi_{rs_l}(k) = |\eta_{c_l}(k)|\, \Delta T\, (y_{c_l}(k-1) - y_l(k-1))\, \frac{\partial y_l(k)}{\partial \phi_{rs_l}} \tag{III.13}$$

où, $\eta_{c_l}(k)$ est le facteur d'adaptation du $l^{ième}$ correcteur partiel NC_l. Pour calculer la dérivée partielle de chaque sortie du système à commander $y_l(k)$ par rapport aux poids du correcteur neuronal partiel correspondant ϕ_{rs_l}, un émulateur neuronal est utilisé. Dans ce cas chaque sortie $s_{l+1}(k)$ $(l = 1, ..., N_{OUT})$ de l'émulateur neuronal est utilisée pour estimer la sortie correspondante $y_l(k)$ et approcher la relation $\frac{\partial y_l(k)}{\partial \phi_{rs_l}}$ par $\frac{\partial y_{n_l}(k)}{\partial \phi_{rs_l}}$ en utilisant l'équation (III.14) suivante :

$$\frac{\partial y_{n_l}(k)}{\partial \phi_{rs_l}} = \frac{\partial s_{l+1}(k)}{\partial \phi_{rs_l}} = \frac{\partial s_{l+1}(k)}{\partial u} \frac{\partial u(k)}{\partial \phi_{rs_l}} \tag{III.14}$$

Selon la structure du schéma de commande proposé, pour calculer le terme $\frac{\partial u(k)}{\partial \phi_{rs_l}}$, on introduit de nouveaux termes qui calculent la dérivée partielle de la commande globale à appliquer au système par rapport aux états des neurones de chaque correcteur partiel $o_{d_l}(k)$ comme suit :

$$\frac{\partial y_{n_l}(k)}{\partial \phi_{rs_l}} = \frac{\partial s_{l+1}(k)}{\partial \phi_{rs_l}} = \sum_{d=1}^{2} \frac{\partial s_{l+1}(k)}{\partial u} \frac{\partial u(k)}{\partial o_{dl}} \frac{\partial o_{dl}(k)}{\partial \phi_{rs_l}} \tag{III.15}$$

Puisque la commande globale $u(k)$ à appliquer au système ne dépend que des sorties des premiers neurones de chaque correcteur partiel $o_{1_l}(k)$ qui représentent les lois de commande partielles $u_l(k)$ pour $l = 1, ..., N_{OUT}$, les termes $\frac{\partial u(k)}{\partial o_{dl}}$ pour $d = 2$ sont nuls et l'équation (III.15) peut se réecrire :

$$\frac{\partial y_{n_l}(k)}{\partial \phi_{rs_l}} = \frac{\partial s_{l+1}(k)}{\partial u} \frac{\partial u(k)}{\partial u_l} \frac{\partial u_l(k)}{\partial \phi_{rs_l}} = J_{l+1}(k) \upsilon_l{}'(k) Q_{1rs_l}(k) \tag{III.16}$$

où $J_i(k) = \dfrac{\partial s_i(k)}{\partial u}$ et $Q_{drs_l}(k) = \dfrac{\partial u_l(k)}{\partial \phi_{rs_l}}$ sont respectivement les fonctions de sensibilité de l'émulateur neuronal et celles de chaque correcteur partiel. Les fonctions $\upsilon_l'(k) = \dfrac{\partial u(k)}{\partial u_l}$ sont les degrés de validité correspondant à chaque commande partielle $u_l(k)$. La stratégie de calcul des validités est décrite dans la partie suivante.

Les fonctions de sensibilité de chaque correcteur neuronal partiel $Q_{drs_l}(k)$ pour $d = 1, 2$ sont calculées à partir de la dérivée partielle de l'équation (III.10) par rapport aux poids du correcteur correspondant $\phi_{rs_l}(k)$ en utilisant l'équation suivante :

$$Q_{drs_l}(k+1) = e^{-|\tau_{c_l}(k)|\Delta T} Q_{drs_l}(k) + \left(1 - e^{-|\tau_{c_l}(k)|\Delta T}\right) \varphi_{d_l}(k)\psi_{d_l}(k) \tag{III.17}$$

avec :

$$\begin{aligned}
\varphi_{1_l}(k) &= \tanh'\left(\sum_{p=1}^{2} \phi_{1p_l}(k) o_{p_l}(k) + y_{c_l}(k) - y_l(k)\right) \\
\varphi_{2_l}(k) &= \tanh'\left(\sum_{p=1}^{2} \phi_{2p_l}(k) o_{p_l}(k) + y_{c_l}(k)\right) \\
\psi_{1_l}(k) &= \left(\delta_r^1 o_{s_l}(k) + \sum_{p=1}^{2} \phi_{1p_l}(k) Q_{pij_l}(k) - \frac{\partial y_{n_l}(k)}{\phi_{rs_l}}\right) \\
\psi_{2_l}(k) &= \left(\delta_r^2 o_{s_l}(k) + \sum_{p=1}^{2} \phi_{2p_l}(k) Q_{pij_l}(k)\right)
\end{aligned} \tag{III.18}$$

Les fonctions de sensibilité de l'émulateur neuronal $J_i(k)$ pour $i = 1, ..., N_e$ sont calculées dans ce cas à partir de la dérivée partielle de l'équation (III.1) qui représente le comportement dynamique de l'émulateur neuronal non carré par rapport à la commande globale appliquée au système $u(k)$ en utilisant l'équation suivante pour $h = 1, ..., N_e$:

$$J_h(k+1) = e^{-|\tau_e(k)|\Delta T} J_h(k) + \left(1 - e^{-|\tau_e(k)|\Delta T}\right) \beta_h(k) \tag{III.19}$$

avec :

$$\begin{aligned}
\beta_1(k) &= \alpha_1(k)\left(\sum_{m=1}^{N_e} w_{1m}(k) J_m(k) + 1\right) \\
\beta_h(k) &= \alpha_h(k)\left(\sum_{m=1}^{N_e} w_{hm}(k) J_m(k)\right), \text{ for } h = 2, ..., N_e
\end{aligned} \tag{III.20}$$

L'adaptation du facteur d'adaptation $\eta_{c_l}(k)$ et de la constante de temps $\tau_{c_l}(k)$ de chaque correcteur partiel est assurée en utilisant un algorithme analogue à celui utilisé pour l'adaptation des paramètres du réseaux émulateur. Cette adaptation est donnée par les équations :

$$\Delta\eta_{c_l}(k) = \Delta T(y_{c_l}(k-1) - y_l(k-1)) \frac{\partial y_{n_l}(k-1)}{\partial \eta_{c_l}} \tag{III.21}$$

$$\Delta\tau_{c_l}(k) = |\eta_{c_l}(k)|\, \Delta T(y_{c_l}(k-1) - y_l(k-1)) \frac{\partial y_{n_l}(k-1)}{\partial \tau_{c_l}} \tag{III.22}$$

A ce niveau, pour chaque correcteur partiel, un autre ensemble de fonctions de sensibilité $r_{h_l}^{\eta_{c_l}}(k)$ et $r_{h_l}^{\tau_{c_l}}(k)$ devrait être calculé pour approcher les dérivées partielles des sorties

$y_{n_l}(k)$ respectivement par rapport aux paramètres $\eta_{c_l}(k)$ et $\tau_{c_l}(k)$ correspondant à chaque correcteur. Ces fonctions sont considérées comme des petites perturbations des sorties $y_{n_l}(k)$ dûes respectivement à des petites variations de η_{c_l} et τ_{c_l}. Pour des raisons de simplification, on définit $r_{h_l}(k) = r_{h_l}^{\eta_{c_l}}(k) = r_{h_l}^{\tau_{c_l}}(k)$. Les fonctions $r_{h_l}(k)$ sont calculées en utilisant l'équation III.1 pour $h = 1, ..., N_e$, $l = 1, ..., N_{OUT}$ comme suit :

$$r_{1_l}(k+1) = e^{-|\tau_e(k)|\Delta T} r_{1_l}(k)$$

$$+(1 - e^{-|\tau_e(k)|\Delta T}) \left(\tanh' \left(\sum_{m=1}^{N_e} w_{1m}(k) s_m(k) + u(k) \right) \right) \left(\sum_{m=1}^{N_e} (w_{1m}(k) r_{m_l}(k)) \right) + \frac{\varepsilon_{c_l}}{|\tau_e(k)|} \tag{III.23}$$

$$r_{h_l}(k+1) = e^{-|\tau_e(k)|\Delta T} r_{h_l}(k)$$

$$+(1 - e^{-|\tau_e(k)|\Delta T}) \left(\tanh' \left(\sum_{m=1}^{N_e} w_{hm}(k) s_m(k) \right) \right) \left(\sum_{m=1}^{N_e} (w_{hm}(k) r_{m_l}(k)) \right) + \frac{\varepsilon_{c_l}}{|\tau_e(k)|} \tag{III.24}$$

Pour chaque correcteur partiel, une constante ε_{c_l} est utilisée pour assurer le démarrage et l'évolution de l'algorithme d'adaptation des paramètres correspondant à partir des conditions initiales nulles.

III.2.3 Calcul des degrés de validité

A chaque instant k, les lois de commandes partielles $u_l(k)$ sont combinées pour générer la commande neuronale globale $u(k)$ en utilisant une stratégie de fusion appropriée. Dans cette partie on propose une nouvelle stratégie de fusion en introduisant une nouvelle formule pour le calcul des validités basées sur les erreurs de poursuites $(y_{c_l}(k) - y_l(k))$ comme suit :

$$\upsilon_l(k) = (y_{c_l}(k) - y_l(k))^2 + \upsilon_l(k-1) \left(1 - \frac{\sum_{f=1}^{N_{OUT}} (y_{c_f}(k) - y_f(k))^2)}{\sum_{f=1}^{N_{OUT}} \max((y_{c_f}(k) - y_f(k))^2)} \right) \tag{III.25}$$

Cette formule est proposée pour éviter un problème d'oscillation qui apparaît si nous utilisons chaque erreur quadratique de commande $(y_{c_l}(k) - y_l(k))^2$ comme un degré de validité relatif à chaque commande partielle. En effet, la validité $\upsilon_l(k)$ conserve sa valeur précédente $\upsilon_l(k-1)$ lorsque toutes les erreurs convergent vers zéro en même temps. Dans le cas contraire, chaque degré de validité est égale à la somme de l'erreur quadratique de commande correspondant et la valeur précédente de chaque degré de validité pondéré

par un autre terme en fonction de la normalisation de toutes les erreurs quadratiques de commande. Pratiquement, il est recommandé d'utiliser l'expression normalisée suivante :

$$\upsilon_l{}'(k) = \frac{\upsilon_l(k)}{\sum\limits_{f=1}^{N_{OUT}} \upsilon_f(k)} \tag{III.26}$$

Finalement, la commande globale appliquée effectivement au système non linéaire SIMO est calculée, à chaque instant k, en utilisant les validités normalisées comme suit :

$$u(k) = \sum_{l=1}^{N_{OUT}} \upsilon_l{}'(k) u_l(k) \tag{III.27}$$

III.3 Simulations numériques

III.3.1 Mise en évidence de l'apport en performances du schéma de commande neuronale adaptative non carrée basée sur un émulateur neuronal

III.3.1.1 Cas d'un système non linéaire SIMO à une entrée et deux sorties

Pour valider la méthode proposée pour le calcul des degrés de validité et mettre en évidence l'apport en performances du schéma de commande neuronale non carrée proposé pour les systèmes non linéaires SIMO, on considère le système non linéaire à une entrée et deux sorties décrit par :

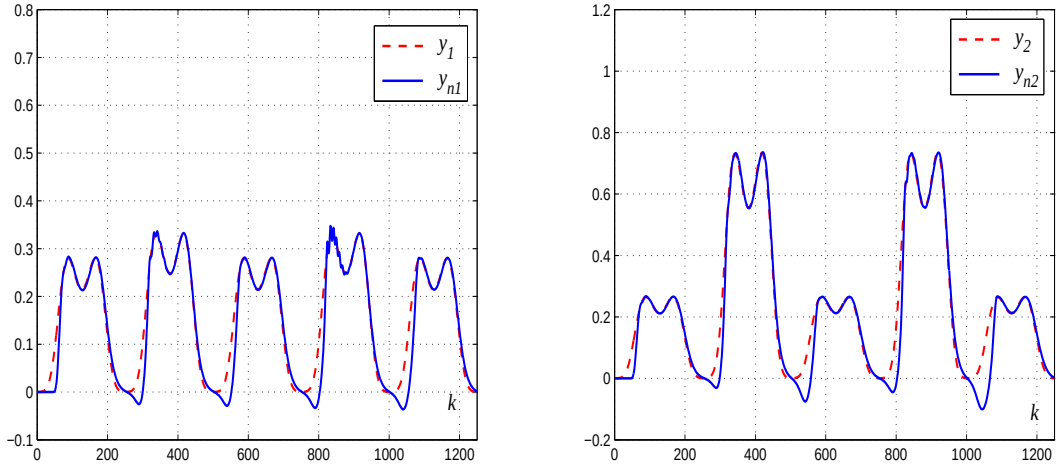
$$\begin{aligned} y_1(k) &= \frac{1}{2}((1.1 - 0.1 z_{11}(k-1)) y_1(k-1) + z_{11}(k-1) u(k-1)) \\ y_2(k) &= \frac{1}{2}((1.5 - z_{12}(k-1) - 0.2 z_{22}(k-1)) y_2(k-1) + z_{12}(k-1) u(k-1)) \end{aligned} \tag{III.28}$$

avec :

$$z_{11}(k-1) = \frac{0.6u(k-1) - 0.06y_1(k-1)}{1 + 0.2y_1(k-1)}$$

$$z_{12}(k-1) = \frac{0.5u(k-1)}{1 + 0.5y_2(k-1)};\ z_{22}(k-1) = \frac{-0.07y_2(k-1)}{1 + 0.1y_2(k-1)}$$

Afin d'évaluer l'efficacité de l'émulateur neuronal non carré, on considère dans un premier temps une émulation en boucle ouverte. Puisque le nombre de neurones de l'émulateur dépend uniquement du nombre d'entrées et de sorties du système étudié, un réseau neuronal à trois neurones est capable d'émuler le système non linéaire considéré ($N_e = N_{OUT} + 1 = 3$). La simulation est menée avec un pas de calcul $\Delta T = 0.5s$. Après plusieurs essais, le terme de démarrage ε_e est choisi égal à 0.1 ; c'est la valeur qui conduit à des résultats d'émulation satisfaisants. En effet, les résultats de l'émulation neuronale sont



(a) Première sortie du système

(b) Deuxième sortie du système

Figure III.3: Variations des sorties réelles et celles de l'émulateur neuronal ($\varepsilon_e = 0.1$).

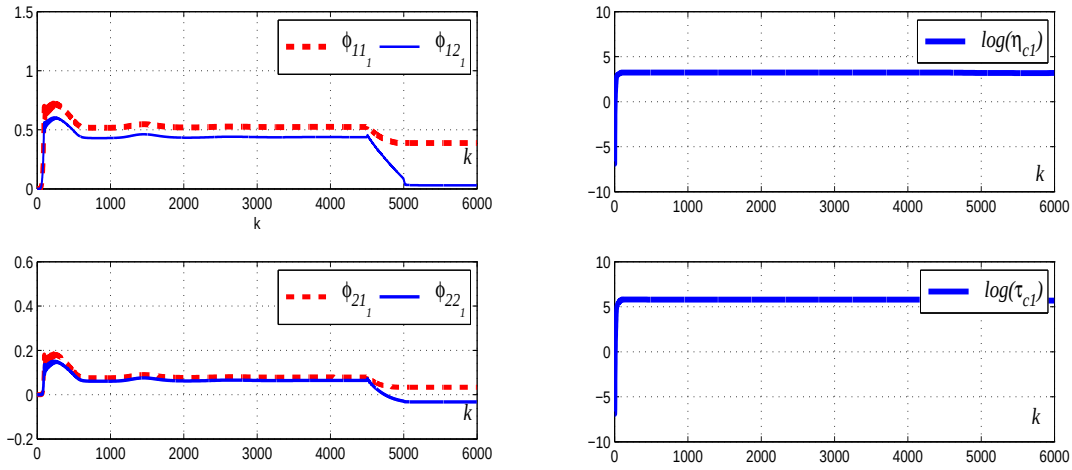
représentés sur la figure III.3. Cette figure montre que l'émulateur neuronal fournit, dans ce cas, une estimation satisfaisante des sorties du système. Pour évaluer les performances de l'émulateur, les deux indices MSE et VAF définis respectivement par les équations (I.22) et (I.23) sont calculés pour les deux sorties du système. Les valeurs obtenues sont résumées dans le tableau III.1. Ces valeurs confirment les conclusions antérieures.

	Emulateur Neuronal ($\varepsilon_e = 0.1$)
MSE_1	$1.3\,10^{-3}$
MSE_2	$3.5\,10^{-3}$
VAF_1	92.45%
VAF_2	94.59%

Tableau III.1: Les indices MSE_l et VAF_l ($l = 1, 2$) calculés pour les deux sorties du système SIMO dans le cas d'une émulation neuronale.

Afin de souligner l'efficacité du schéma de commande proposé, le correcteur neuronal adaptatif non carré est mis en oeuvre pour le système non linéaire SIMO considéré. Le correcteur neuronal global est formé avec deux correcteurs partiels constitué chacun par un réseau à deux neurones. Pour choisir les paramètres de démarrage des deux correc-

teurs partiels et de l'émulateur neuronal de nombreux essais sont réalisés pour retenir les meilleures valeurs. Un choix judicieux de ces paramètres conduit, alors, à retenir $\varepsilon_{c_l} = 2$ pour $l = 1, 2$ et $\varepsilon_e = 1$. Les simulations proposées visent le problème conjoint de poursuite et de régulation. Elles comprennent une phase de poursuite pour $k \in [0, 4000]$ et une phase de régulation pour $k \in [4000, 6000]$. Au cours de la phase de régulation une perturbation de type échelon et d'amplitude 10% de la valeur d'entrée du système de commande, est appliquée sur les sorties du système durant l'intervalle $[4500, 5000]$. Les résultats de simulation sont donnés par les figures III.4, III.5, III.6, III.7, III.8 et III.9.
Les variations des paramètres des deux correcteurs partiels sont données sur les figures III.4 et III.5. On constate que les amplitudes des poids des deux correcteurs sont différentes. Ainsi, on peut conclure que les paramètres de chaque correcteur s'adaptent en fonction de la dynamique de la sortie correspondante.



(a) Amplitude des poids

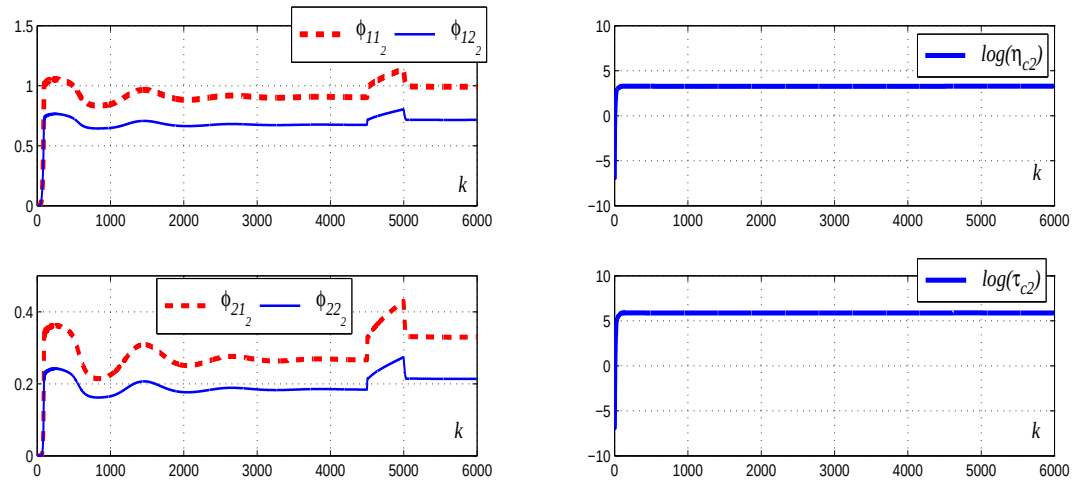
(b) $\eta_{c_1}(k)$ et $\tau_{c_1}(k)$ à l'échelle logarithmique

Figure III.4: Commande neuronale adaptative basée sur un émulateur neuronal : variations des paramètres du premier correcteur neuronal partiel ($\varepsilon_e = 1$, $\varepsilon_{c_1} = 2$).

A chaque instant k, on calcule les degrés de validité $\upsilon_{1_N}(k)$ et $\upsilon_{2_N}(k)$ (figure III.7) de chaque commande partielle en utilisant respectivement les relations (III.29) et (III.30).

$$\begin{aligned} \upsilon_1(k) &= (e_{c_1}(k))^2 + \upsilon_1(k-1)\left(1 - \frac{(e_{c_1}(k))^2 + (e_{c_2}(k))^2}{8}\right) \\ \upsilon_2(k) &= (e_{c_2}(k))^2 + \upsilon_2(k-1)\left(1 - \frac{(e_{c_1}(k))^2 + (e_{c_2}(k))^2}{8}\right) \end{aligned} \tag{III.29}$$

$$\begin{aligned} \upsilon_1{}'(k) &= \frac{\upsilon_1(k)}{\upsilon_1(k) + \upsilon_2(k)} \\ \upsilon_2{}'(k) &= \frac{\upsilon_2(k)}{\upsilon_1(k) + \upsilon_2(k)} \end{aligned} \tag{III.30}$$

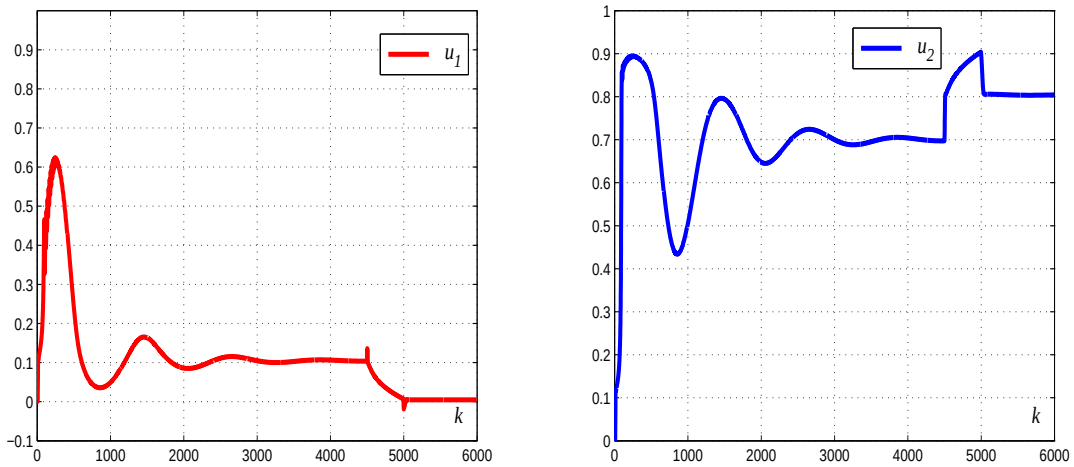


(a) Amplitude des poids

(b) $\eta_{c_2}(k)$ et $\tau_{c_2}(k)$ à l'échelle logarithmique

Figure III.5: Commande neuronale adaptative basée sur un émulateur neuronal : variations des paramètres du deuxième correcteur neuronal partiel ($\varepsilon_e = 1$, $\varepsilon_{c_2} = 2$).

La commande effective appliquée au système, donnée sur la figure III.8, est une fusion des deux lois de commande élémentaires (figure III.6) pondérées chacune par le degré de validité correspondant.



(a) La sortie du premier correcteur partiel

(b) La sortie du deuxième correcteur partiel

Figure III.6: Commande neuronale adaptative basée sur un émulateur neuronal : variations des lois de commande partielles ($\varepsilon_e = 1$, $\varepsilon_{c_1} = 2$, $\varepsilon_{c_2} = 2$).

La figure III.9 donne les variations des sorties désirées et des sorties réelles du système non linéaire SIMO. Cette figure montre que le correcteur neuronal non carré proposé

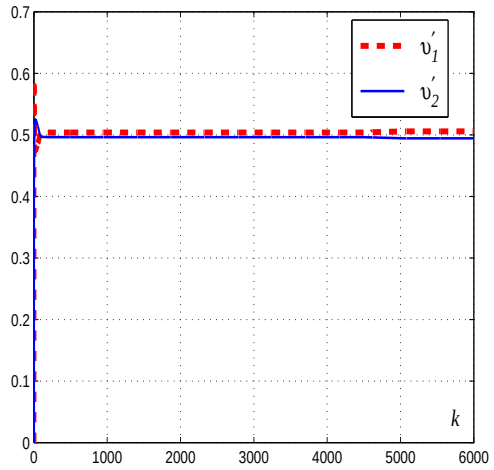


Figure III.7: Commande neuronale adaptative basée sur un émulateur neuronal : variations des degrés de validité normalisés ($\varepsilon_e = 1$, $\varepsilon_{c_1} = 2$, $\varepsilon_{c_2} = 2$).

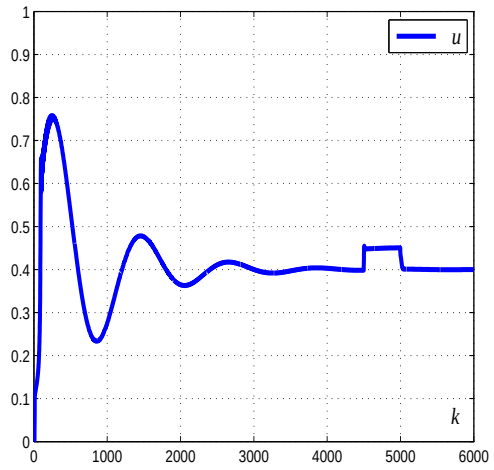
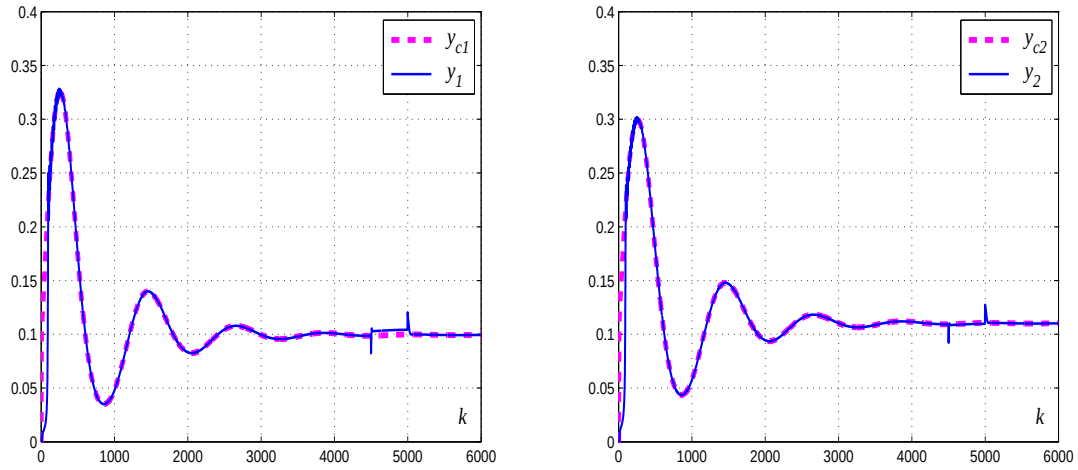


Figure III.8: Commande neuronale adaptative basée sur un émulateur neuronal : Variations de la loi de commande appliquée au système non linéaire SIMO ($\varepsilon_e = 1$, $\varepsilon_{c_1} = 2$, $\varepsilon_{c_2} = 2$).

conduit à de bonnes performances en termes de poursuite et de régulation. En effet, on a pu enregistrer une poursuite parfaite des consignes imposées. Le système de commande a pu assurer, par ailleurs, une bonne rejection des perturbations affectant les sorties du système SIMO considéré.



(a) Première sortie du système

(b) Deuxième sortie du système

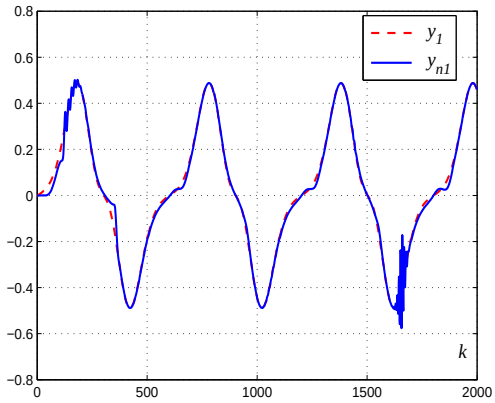
Figure III.9: Commande neuronale adaptative basée sur un émulateur neuronal ($\varepsilon_e = 1$ $\varepsilon_{c_1} = 2$ et $\varepsilon_{c_2} = 2$) : variations des sorties réelles et désirées.

III.3.1.2 Cas d'un système non linéaire SIMO à une entrée et trois sorties

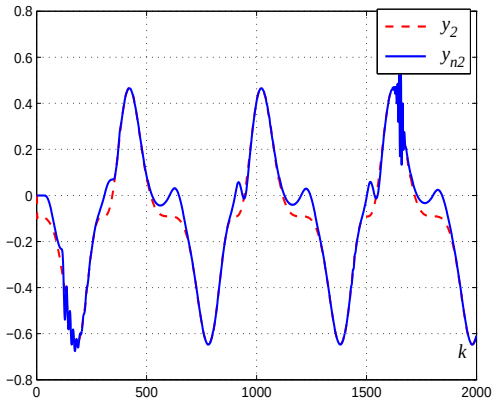
On considère dans ce paragraphe un deuxième exemple où le système non linéaire a une entrée et trois sorties. Ce système est défini par les relations suivantes :

$$\begin{aligned} y_1(k) &= 0.2y_1(k-1) + 0.5u(k-2)^3 + 0.1u(k-2) \\ y_2(k) &= -(0.1y_2(k-1) + u(k-2)^3) - 0.1 \\ y_3(k) &= 0.9(0.1y_3(k-1) + u(k-2)^3) \end{aligned} \tag{III.31}$$

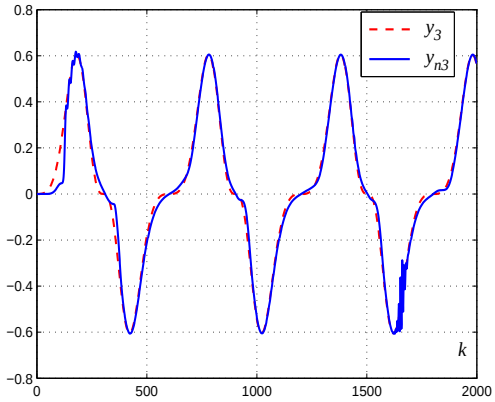
Un réseau neuronal à quatre neurones est utilisé dans ce cas pour l'émulation du système non linéaire considéré ($N_e = N_{OUT} + 1 = 4$). La simulation est menée avec un pas de calcul $\Delta T = 0.5s$. Un choix judicieux du terme de démarrage $\varepsilon_e = 0.1$ est retenu suite à plusieurs essais afin d'assurer une erreur d'émulation minimale. Les résultats de l'émulation neuronale sont représentés sur la figure III.10. Cette figure montre que l'émulateur neuronal fournit, dans ce cas, une estimation satisfaisante des sorties du système.



(a) Première sortie du système



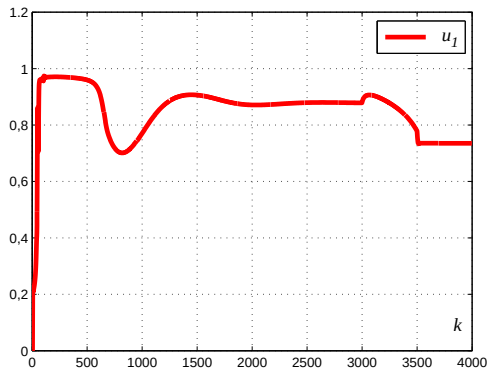
(b) Deuxième sortie du système



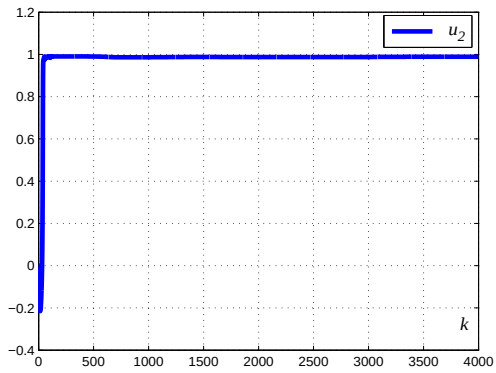
(c) Troisième sortie du système

Figure III.10: Variations des sorties réelles et celles de l'émulateur neuronal ($\varepsilon_e = 0.1$).

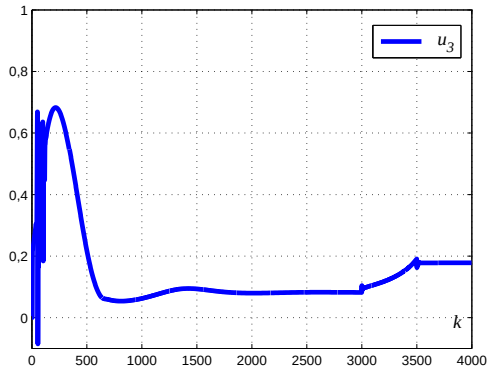
Pour souligner l'efficacité du schéma de commande proposé pour ce système, le correcteur neuronal adaptatif non carré est mis en œuvre pour le système non linéaire SIMO considéré. Le correcteur neuronal global est construit avec trois correcteurs partiels constitué chacun par un réseau de deux neurones. Après quelques essais, les termes de démarrage des correcteurs partiels et de celui de l'émulateur neuronal retenus sont égaux $\varepsilon_{c_1} = 30$, $\varepsilon_{c_2} = 25$, $\varepsilon_{c_3} = 30$ et $\varepsilon_e = 10$. La différences entre les valeurs choisies pour $\varepsilon_{c_l}, l = 1, 3$ est dûe à la différence entre les dynamiques des trois sorties du système étudié. Les simulations réalisées considèrent conjointement les problèmes de poursuite et de régulation. Ces simulations comprennent une phase de poursuite pour $k \in [0, 2000]$ et une phase de régulation pour $k \in [2000, 4000]$. Au cours de la phase de régulation une perturbation de type échelon et d'amplitude 10% de la valeur d'entrée du système de commande, est appliquée sur les sorties du système durant l'intervalle $[3000, 3500]$. Les variations des commandes partielles sont données sur la figure III.11.



(a) La sortie du premier correcteur partiel



(b) La sortie du deuxième correcteur partiel



(c) La sortie du troisième correcteur partiel

Figure III.11: Commande neuronale adaptative basée sur un émulateur neuronal : variations des lois de commande partielles ($\varepsilon_e = 10$, $\varepsilon_{c_1} = 30$, $\varepsilon_{c_2} = 25$ et $\varepsilon_{c_3} = 30$).

A chaque instant k, les degrés de validité de chaque commande partiel (donnés sur la figure III.12) sont calculés en utilisant respectivement les relations suivantes :

$$\begin{aligned} \upsilon_1(k) &= (e_{c_1}(k))^2 + \upsilon_1(k-1)\left(1 - \frac{(e_{c_1}(k))^2 + (e_{c_2}(k))^2 + (e_{c_3}(k))^2}{12}\right) \\ \upsilon_2(k) &= (e_{c_2}(k))^2 + \upsilon_2(k-1)\left(1 - \frac{(e_{c_1}(k))^2 + (e_{c_2}(k))^2 + (e_{c_3}(k))^2}{12}\right) \\ \upsilon_3(k) &= (e_{c_3}(k))^2 + \upsilon_3(k-1)\left(1 - \frac{(e_{c_1}(k))^2 + (e_{c_2}(k))^2 + (e_{c_3}(k))^2}{12}\right) \end{aligned} \tag{III.32}$$

$$\upsilon_l{}'(k) = \frac{\upsilon_l(k)}{\upsilon_1(k) + \upsilon_2(k) + \upsilon_3(k)},\ l = 1, ..., 3 \tag{III.33}$$

La loi de commande adaptative neuronale appliqué au système est donnée sur la figure III.13. Elle est obtenue par la fusion des lois élémentaires :

$$u(k) = \sum_{l=1}^{N_{OUT}} \upsilon_1{}'(k) u_l(k),\ l = 1, ..., 3 \tag{III.34}$$

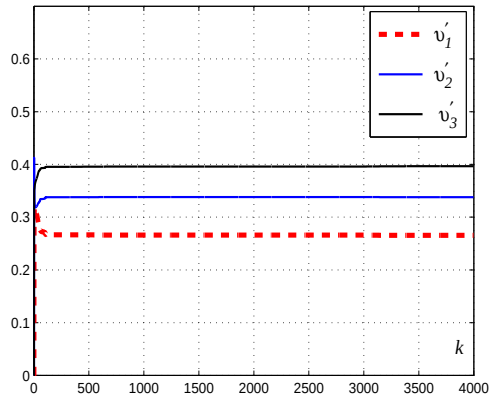


Figure III.12: Commande neuronale adaptative basée sur un émulateur neuronal : variations des degrés de validité normalisés ($\varepsilon_e = 10$, $\varepsilon_{c_1} = 30$, $\varepsilon_{c_2} = 25$ et $\varepsilon_{c_3} = 30$).

La figure III.14 donne les variations des sorties désirées et des sorties réelles du système

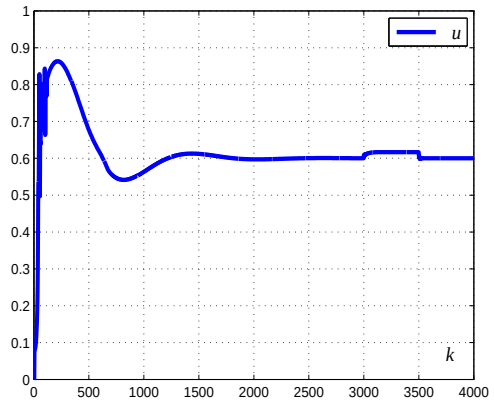
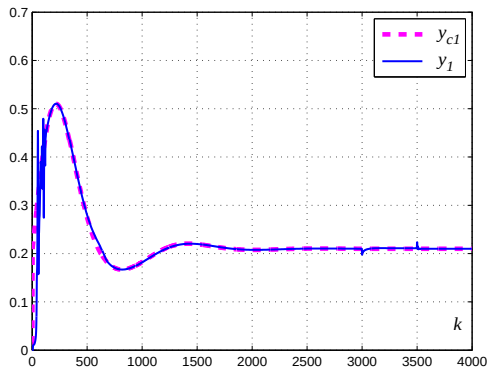
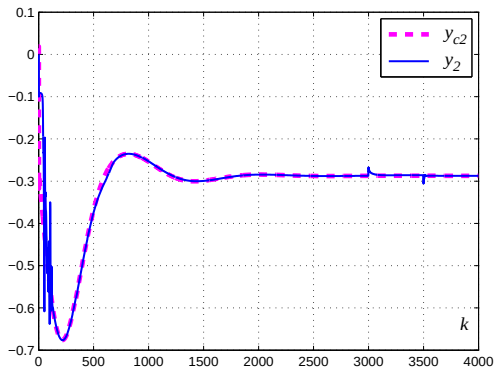


Figure III.13: Commande neuronale adaptative basée sur un émulateur neuronal : variation de la loi de commande appliquée au système non linéaire à 1 entrée et 3 sorties ($\varepsilon_e = 10$, $\varepsilon_{c_1} = 30$, $\varepsilon_{c_2} = 25$ et $\varepsilon_{c_3} = 30$).

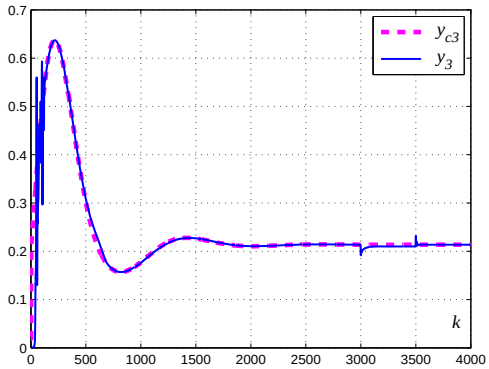
non linéaire considéré. Cette figure confirme que le correcteur proposé pour la commande des systèmes non linéaires SIMO (1 entrée-3 sorties) assure de bonnes performances en poursuite et en régulation. En effet, on a pu enregistrer une poursuite satisfaisante des consignes imposées et une bonne rejection des perturbations affectant les sorties du système.



(a) Première sortie du système



(b) Deuxième sortie du système



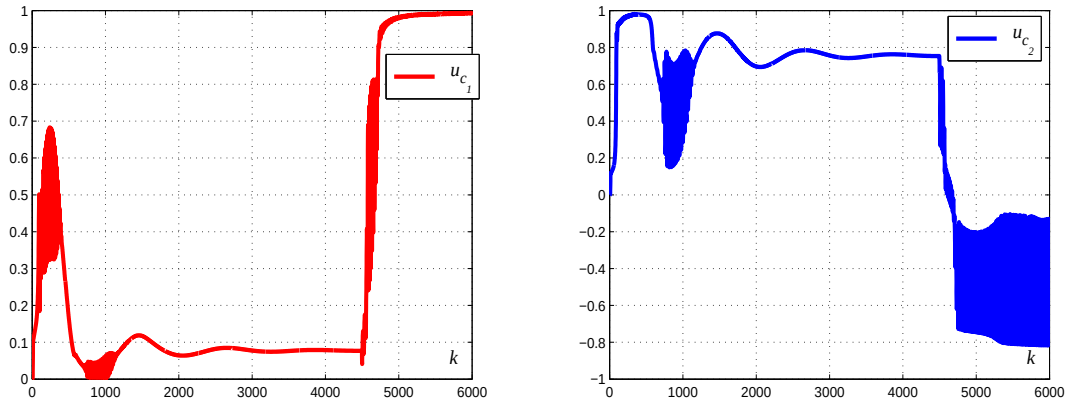
(c) Troisième sortie du système

Figure III.14: Commande neuronale adaptative basée sur un émulateur neuronal ($\varepsilon_e = 10$, $\varepsilon_{c_1} = 30$, $\varepsilon_{c_2} = 25$ et $\varepsilon_{c_3} = 30$) : variations des sorties réelles et désirées.

Les résultats de simulations représentés pour les deux exemples de systèmes non linéaires SIMO confirment que le schéma de commande neuronale adaptative non carrée basé sur un émulateur neuronal conduit à de bonnes performances en boucle fermée. L'avantage principal de l'utilisation de l'émulateur neuronal dans le schéma de commande proposé réside dans le fait qu'un réseau de neurones de structure simple composé d'un nombre réduit de neurones permet d'émuler la dynamique des systèmes non linéaires SIMO avec une précision suffisante. De plus, grâce à l'utilisation du paramètre de démarrage ε_e tout les paramètres de l'émulateur neuronal s'adaptent à partir des conditions initiales nulles. Cependant, le choix de ce paramètre est très sensible. En effet, ce paramètre doit être soigneusement choisi pour ne pas dégrader les performances de la commande en boucle fermée. Dans la partie suivante, on présente quelques simulations pour montrer l'effet d'un choix arbitraire de ε_e.

III.3.2 Les problèmes liés à l'utilisation d'un émulateur neuronal dans le schéma de commande proposé

Dans cette partie on reprend l'exemple de simulation du système à une entrée et deux sorties défini par l'équation III.28. Pour un choix arbitraire du paramètre de démarrage ε_e (par exemple avec $\varepsilon_e = 0.4$) et en utilisant $\varepsilon_{c_1} = \varepsilon_{c_2} = 2$, on obtient les résultats donnés sur les figures III.15, III.16, III.17 et III.18. Les variations des lois de commandes partielles et globale données respectivement sur les figures III.15 et III.17 laissent apparaître des variances relativement importantes des signaux de commandes.



(a) La sortie du premier contrôleur partiel (b) La sortie du deuxième contrôleur partiel

Figure III.15: Variations des lois de commande partielles ($\varepsilon_e = 0.4$, $\varepsilon_{c_1} = 2$, $\varepsilon_{c_2} = 2$).

La commande globale appliquée au système est obtenue par fusion des commandes partielles pondérées par les validitées correspondantes (figure III.16).

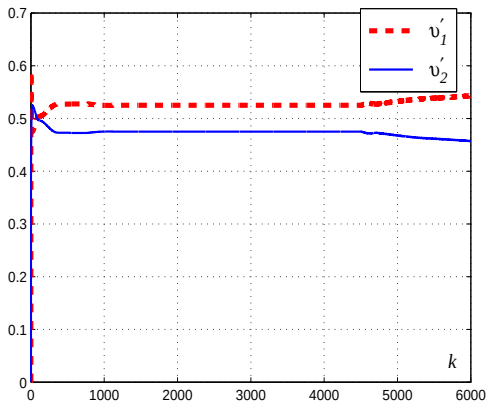


Figure III.16: Variations des degrés de validité normalisés ($\varepsilon_e = 0.4$, $\varepsilon_{c_1} = 2$, $\varepsilon_{c_2} = 2$).

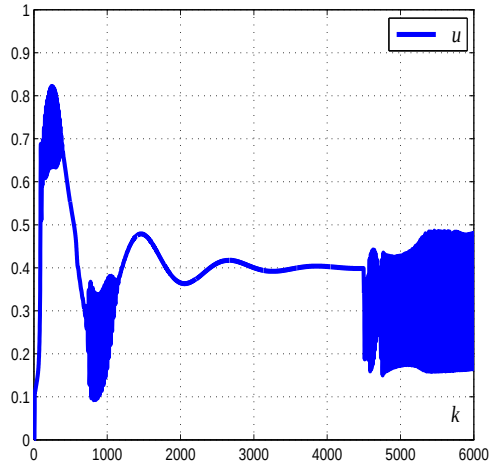
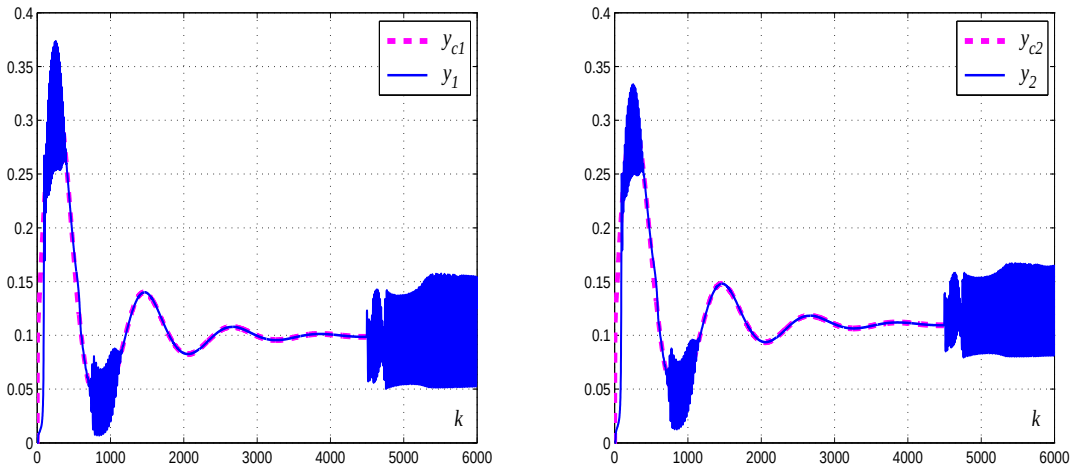


Figure III.17: Variations de la loi de commande appliquée au système non linéaire SIMO ($\varepsilon_e = 0.4$, $\varepsilon_{c_1} = 2$, $\varepsilon_{c_2} = 2$).

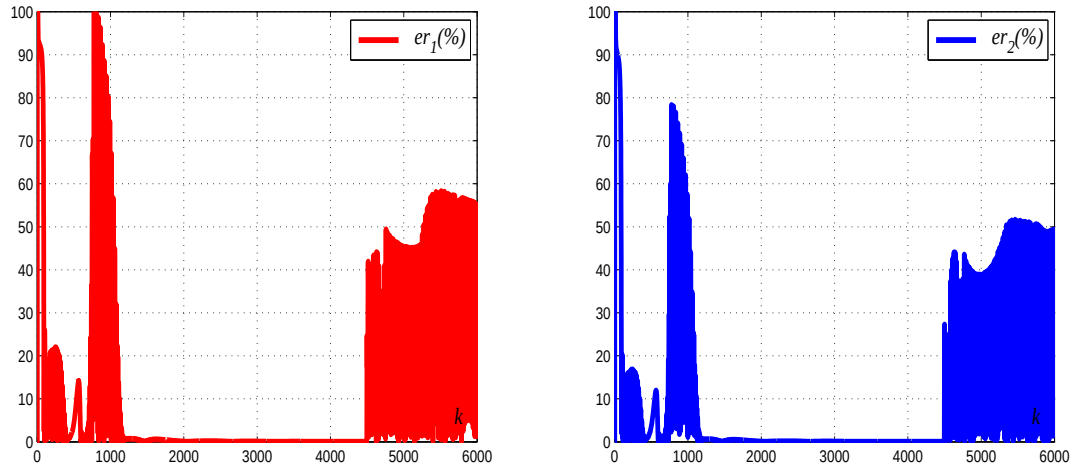
La figure III.18, donnant les sorties réelles et désirées, montre de très mauvaises performances en poursuite et en régulation. En effet, des oscillations importantes apparaissent pour les deux sorties du système. La figure III.19, donnant les erreurs relatives entre les sorties réelles et les sorties désirées, montre des valeurs relativement importantes.



(a) Première sortie du système

(b) Deuxième sortie du système

Figure III.18: Commande neuronale adaptative basée sur un émulateur neuronal ($\varepsilon_e = 0.4$, $\varepsilon_{c_1} = 2$, $\varepsilon_{c_2} = 2$) : variations des sorties réelles et désirées.



(a) Erreur relative (première sortie du système) (b) Erreur relative (deuxième sortie du système)

Figure III.19: Commande neuronale adaptative basée sur un émulateur neuronal ($\varepsilon_e = 0.4$, $\varepsilon_{c_1} = 2$, $\varepsilon_{c_2} = 2$) : variations des erreurs relatives entre les sorties réelles et les sorties désirées.

III.4 Adaptation des paramètres des correcteurs partiels dans le cas d'une émulation multimodèle pour les système non linéaires SIMO

Pour surmonter les problèmes liés à un choix non judicieux du paramètre de démarrage de l'émulation, un émulateur multimodèle pour la commande neuronale des systèmes non linéaires SIMO est proposé dans cette section. La structure de l'émulateur multimodèle proposée dans le chapitre précédent permet la représentation des systèmes non linéaires SIMO. En insérant l'émulateur multimodèle découplé, résultant de l'estimation paramétrique et décrit soit par des modèles polynômiaux (relation (II.7)) soit par des modèles d'état (relation (II.8)), dans le schéma de commande de la figure III.1 quelques modifications doivent être apportées sur l'algorithme d'adaptation des paramètres des correcteurs partiels.

En utilisant l'émulateur multimodèle, l'expression (III.13) décrivant l'adaptation des poids de chaque correcteur partiel NC_l est remplacée par l'équation suivante :

$$\Delta\phi_{rs_l}(k) = |\eta_{c_l}(k)|\,\Delta T(y_{c_l}(k-1) - y_l(k-1))\frac{\partial y_{ml}(k)}{\partial \phi_{rs_l}} \tag{III.35}$$

où $y_{ml}(k)$ est la sortie de l'émulateur multimodèle utilisée pour estimer la sortie correspondante du système $y_l(k)$ et approcher les dérivées partielles des sorties du système par rapport aux poids de chaque correcteur partiel $\frac{\partial y_l(k)}{\partial \phi_{rs_l}}$ par l'expression $\frac{\partial y_{ml}(k)}{\partial \phi_{rs_l}}$. Ces

expressions, qui remplacent l'équation (III.16), sont données par :

$$\frac{\partial y_{ml}(k)}{\partial \phi_{rs_l}} = \frac{\partial y_{ml}(k)}{\partial u}\frac{\partial u(k)}{\partial o_{1_l}}\frac{\partial o_{1_l}(k)}{\partial \phi_{rs_l}} = \upsilon_l{}'(k) Q_{1rs_l}(k)\frac{\partial y_{ml}(k)}{\partial u} \tag{III.36}$$

où $\upsilon_l{}'(k) = \dfrac{\partial u(k)}{\partial u_l}$ sont les degrés de validité correspondant à chaque commande partielle $u_l(k)$. Puisque les sorties des premiers neurones de chaque correcteur partiel représentent les lois de commande élémentaires, ces degrés remplacent les termes $\dfrac{\partial u(k)}{\partial o_{1_l}}$. Les fonctions $Q_{drs_l}(k) = \dfrac{\partial u_l(k)}{\partial \phi_{rs_l}}$ sont les fonctions de sensibilité de chaque correcteur partiel. Ces fonctions de sensibilité sont calculées, pour $d = 1, 2$, à partir de la dérivée partielle de l'équation (III.10) par rapport aux poids du correcteur correspondant $\phi_{rs_l}(k)$ en utilisant l'équation suivante :

$$Q_{drs_l}(k+1) = e^{-|\tau_{c_l}(k)|\Delta T} Q_{drs_l}(k) + \left(1 - e^{-|\tau_{c_l}(k)|\Delta T}\right)\varphi_{d_l}(k)\psi_{d_l}(k) \tag{III.37}$$

avec :

$$\begin{aligned}
\varphi_{1_l}(k) &= \tanh'\left(\sum_{p=1}^{2}\phi_{1p_l}(k)o_{p_l}(k) + y_{c_l}(k) - y_l(k)\right)\\
\varphi_{2_l}(k) &= \tanh'\left(\sum_{p=1}^{2}\phi_{2p_l}(k)o_{p_l}(k) + y_{c_l}(k)\right)\\
\psi_{1_l}(k) &= \left(\delta_r^1 o_{s_l}(k) + \sum_{p=1}^{2}\phi_{1p_l}(k)Q_{pij_l}(k) - \frac{\partial y_{ml}(k)}{\phi_{rs_l}}\right)\\
\psi_{2_l}(k) &= \left(\delta_r^2 o_{s_l}(k) + \sum_{p=1}^{2}\phi_{2p_l}(k)Q_{pij_l}(k)\right)
\end{aligned} \tag{III.38}$$

Les fonctions de sensibilité de l'émulateur neuronal $J_h(k)$, $(h = 1, ..., N_e)$ définies par les équations (III.19) et (III.20) sont remplacées, dans ce cas, par les dérivées partielles des sorties du multimodèle par rapport à la commande globale $\dfrac{\partial y_{ml}(k)}{\partial u}$, $(l = 1, ..., N_{OUT})$ qui sont calculées par l'équation suivante :

$$\frac{\partial y_{ml}(k)}{\partial u} = \sum_{i=1}^{N_{ml}} \mu_{l,i}(\xi(k))\frac{\partial y_{l,i}(k)}{\partial u} \tag{III.39}$$

En utilisant l'émulateur multimodèle dans le schéma de commande neuronale proposé, les équations d'adaptation des facteurs $\eta_{c_l}(k)$ et des constantes de temps $\tau_{c_l}(k)$ de chaque correcteur partiel, définies dans le cas où un émulateur neuronal est utilisé par (III.21) et (III.22), sont remplacées par les équations suivantes :

$$\Delta\eta_{c_l}(k) = \Delta T(y_{c_l}(k-1) - y_l(k-1))\frac{\partial y_{ml}(k-1)}{\partial \eta_{c_l}} \tag{III.40}$$

$$\Delta\tau_{c_l}(k) = |\eta_{c_l}(k)|\,\Delta T(y_{c_l}(k-1) - y_l(k-1))\frac{\partial y_{ml}(k-1)}{\partial \tau_{c_l}} \tag{III.41}$$

Les dérivées partielles des sorties multimodèles par rapport aux paramètres des correcteurs partiels sont données par :

$$\begin{aligned}\frac{\partial y_{ml}(k)}{\partial \eta_{c_l}} &= \frac{\partial y_{ml}(k)}{\partial u}\frac{\partial u(k)}{\partial u_l}\frac{\partial u_l(k)}{\partial \eta_{c_l}} = \upsilon_l{}'(k)R_{1_l}^{\eta_{c_l}}(k)\frac{\partial y_{ml}(k)}{\partial u}\\ \frac{\partial y_{ml}(k)}{\partial \tau_{c_l}} &= \frac{\partial y_{ml}(k)}{\partial u}\frac{\partial u(k)}{\partial u_l}\frac{\partial u_l(k)}{\partial \tau_{c_l}} = \upsilon_l{}'(k)R_{1_l}^{\tau_{c_l}}(k)\frac{\partial y_{ml}(k)}{\partial u}\end{aligned} \tag{III.42}$$

Les fonctions $R_{1_l}^{\eta_{c_l}}(k)$ et $R_{1_l}^{\tau_{c_l}}(k)$ sont définies pour approcher les dérivées partielles $\frac{\partial u_l(k)}{\partial \eta_{c_l}}$ et $\frac{\partial u_l(k)}{\partial \tau_{c_l}}$. Ces fonctions de sensibilité sont calculées pour les deux neurones de chaque correcteur partiel. En effet, pour chaque correcteur d'indice l ces fonctions sont considérées comme des petites perturbations des états du neurone d'indice d dûes respectivement à des petites variations $\partial\eta_{c_l}$ et $\partial\tau_{c_l}$. Pour des raisons de simplification, on considère $R_{d_l}^{\eta_{c_l}}(k) = R_{d_l}^{\tau_{c_l}}(k) = R_{d_l}$ $(d = 1, 2)$. Ces fonctions sont calculées à partir de la dérivée partielle de l'équation (III.10) en tenant compte de (III.11). Elles sont, alors, données par la relation suivante, pour $d = 1, 2$:

$$R_{d_l}(k+1) = e^{-|\tau_{c_l}(k)|\Delta T}R_{d_l}(k) + (1 - e^{-|\tau_{c_l}(k)|\Delta T})\varphi_{d_l}(k)\chi_{d_l}(k) + \frac{\varepsilon_{c_l}}{|\tau_{c_l}(k)|} \tag{III.43}$$

avec :

$$\begin{aligned}\varphi_{1_l}(k) &= \tanh'\left(\sum_{p=1}^{2}\phi_{1p_l}(k)o_{p_l}(k) + (y_{c_l}(k) - y_l(k))\right)\\ \varphi_{2_l}(k) &= \tanh'\left(\sum_{p=1}^{2}\phi_{2p_l}(k)o_{p_l}(k) + y_{c_l}(k)\right)\\ \chi_{1_l}(k) &= \sum_{p=1}^{2}\phi_{1p_l}(k)R_{p_l}(k) - \frac{\partial y_{ml}(k)}{\partial \eta_{c_l}}\\ \chi_{2_l}(k) &= \sum_{p=1}^{2}\phi_{2p_l}(k)R_{p_l}(k)\end{aligned} \tag{III.44}$$

III.5 Simulations numériques : mise en évidence de l'apport en performance du schéma de commande neuronale adaptative proposée basée sur un émulateur multimodèle

Pour montrer l'efficacité de l'émulation multimodèle pour la représentation des systèmes non linéaires SIMO et pour mettre en évidence l'apport en performances de l'utilisation d'un émulateur multimodèle dans le schéma de commande neuronale non carré, on reprend les deux systèmes non linéaires SIMO définis précédemment respectivement par les équations III.28 et III.31.

III.5.1 Cas d'un système non linéaire SIMO à une entrée et deux sorties

III.5.1.1 Un émulateur multimodèle non carré

L'ensemble de données d'identification est utilisé pour générer l'ensemble de vecteurs de regression $\vartheta_{l,j} = [y_l(j), y_l(j-1), u(j-1)]^T$, $l = 1, 2$ qui va être traité par la procédure de classification présentée dans le chapitre précédent pour fournir les vecteurs de regression centres de classes $\vartheta^*_{l,j}$. Cette méthode donne naissance à trois classes pour chaque sortie du système. Puisque l'entrée du système $u(k)$ est considérée comme variable d'indexation, les centres et les dispersions obtenus en utilisant l'expression (II.42) sont résumés sur le tableaux III.2.

	Les paramètres de synthèse de l'émulateur multimodèle
$c_{1,i}$ et $\sigma_{1,i}$	$c_{1,1} = -0.6974$, $\sigma_{1,1} = 0.6556$ $c_{1,2} = -0.0418$, $\sigma_{1,2} = 0.6556$ $c_{1,3} = 0.6941$, $\sigma_{1,3} = 0.7359$
$c_{2,i}$ et $\sigma_{2,i}$	$c_{2,1} = -0.7419$, $\sigma_{2,1} = 0.6535$ $c_{2,2} = -0.0884$, $\sigma_{2,2} = 0.6535$ $c_{2,3} = 0.6745$, $\sigma_{2,3} = 0.7629$

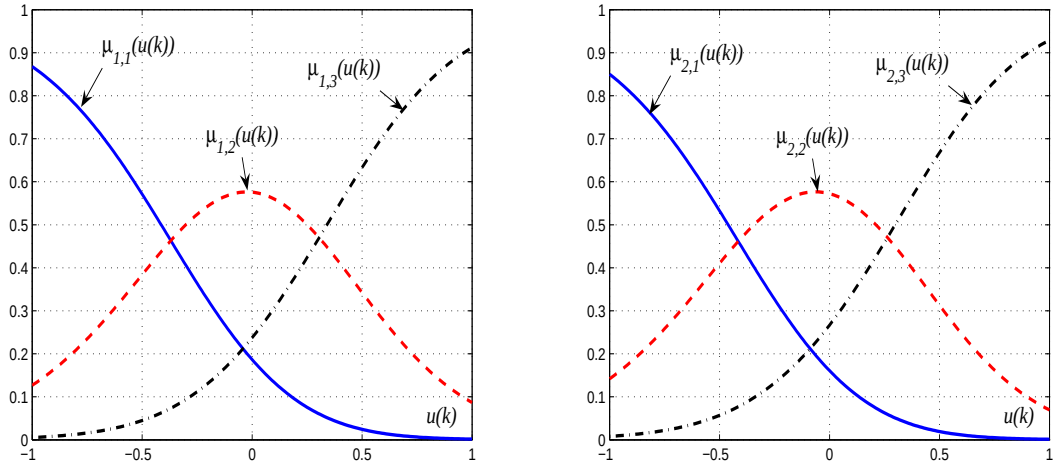
Tableau III.2: Les centres et les dispersions des fonctions de pondération déterminés systématiquement pour l'émulation du systèmes non linéaires SIMO.

La figure III.20 donne les fonctions de pondération obtenues en utilisant les paramètres de synthèse présentés dans le tableau III.2.

D'après ces résultats, chaque sortie du système peut être représentée par un multimodèle SISO constitué de trois sous-modèles. Dans ce cas, deux bases (B_l, $l = 1, 2$) de trois modèles d'état du premier ordre sont retenues pour représenter chaque sortie du système. Les sous-modèles de chaque base se présentent, alors, sous la forme suivante :

$$\begin{aligned} x_{l,i}(k+1) &= a_{l,i}x_{l,i}(k) + b_{l,i}u(k) \\ y_{l,i}(k) &= x_{l,i}(k) \end{aligned} \tag{III.45}$$

avec : $\theta_{l,i} = \begin{bmatrix} a_{l,i} & b_{l,i} \end{bmatrix}$ le vecteur des paramètres de chaque sous-modèle appartenant à une base B_l.



(a) Les fonctions de pondération (premier multimodèle)

(b) Les fonctions de pondération (deuxième multimodèle)

Figure III.20: Les fonctions de pondération obtenues suite à l'application de la procédure de classification.

Les modèles partiels résultant de l'application de l'algorithme d'identification développé dans le chapitre précédent sont, donc, décrits par les vecteurs de paramètres $\theta_{l,i}$ suivants :

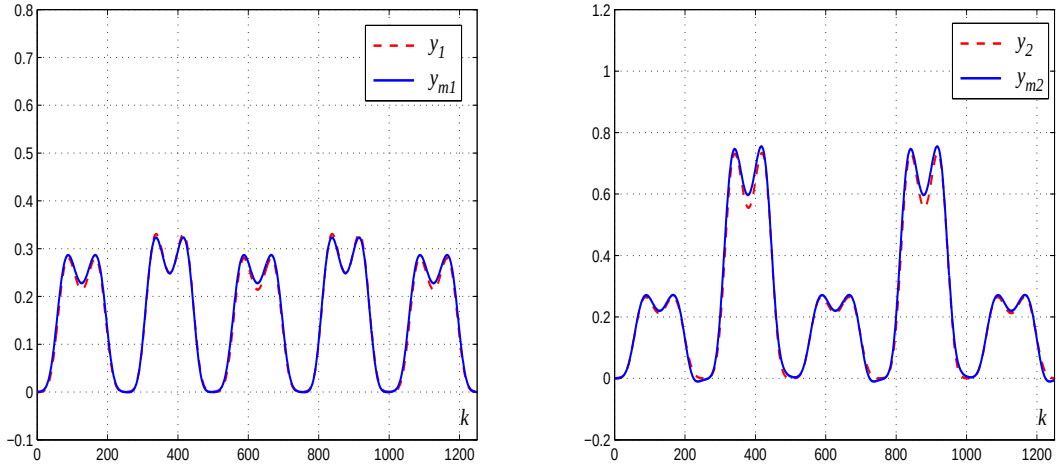
* Les sous modèles de la base B_1 représentant la première sortie du système

$$\begin{aligned} \theta_{1,1} = \begin{bmatrix} 0.6811 & -0.2193 \end{bmatrix}, \theta_{1,2} = \begin{bmatrix} 0.2428 & 0.0396 \end{bmatrix}, \\ \theta_{1,3} = \begin{bmatrix} 0.6501 & 0.1847 \end{bmatrix}, \end{aligned} \tag{III.46}$$

* Les sous modèles de la base B_2 représentant la deuxième sortie du système

$$\begin{aligned} \theta_{2,1} = \begin{bmatrix} 0.8827 & -0.1992 \end{bmatrix}, \theta_{2,2} = \begin{bmatrix} 0.1779 & 0.1077 \end{bmatrix}, \\ \theta_{2,3} = \begin{bmatrix} 0.8086 & 0.0904 \end{bmatrix}, \end{aligned} \tag{III.47}$$

La figure III.21 illustre les résultats de validation du multimodèle obtenu en utilisant la même séquence d'entrée utilisée pour valider l'émulateur neuronal. Par comparaison aux résultats obtenus en utilisant l'émulateur neuronal (figure III.3), ces résultats montrent que l'émulateur multimodèle offre une meilleure précision de modélisation.



(a) Première sortie du système

(b) Deuxième sortie du système

Figure III.21: Variation des sorties du système non linéaire SIMO et celles de l'émulateur multimodèle.

Les indices de performances MSE_l et VAF_l, $l = 1, 2$ sont calculés pour confirmer l'apport en performances de la méthode proposée par comparaison à la méthode neuronale. Le tableau III.3 résume ces indices dans le cas des émulateurs neuronal et multimodèle.

	Emulateur Neuronal ($\varepsilon_e = 20$)	Emulateur Multimodèle
MSE_1	$1.6\,10^{-3}$	$6.57\,10^{-5}$
MSE_2	$4.8\,10^{-3}$	$3.08\,10^{-4}$
VAF_1	91.33%	99.66%
VAF_2	93.51%	99.60%

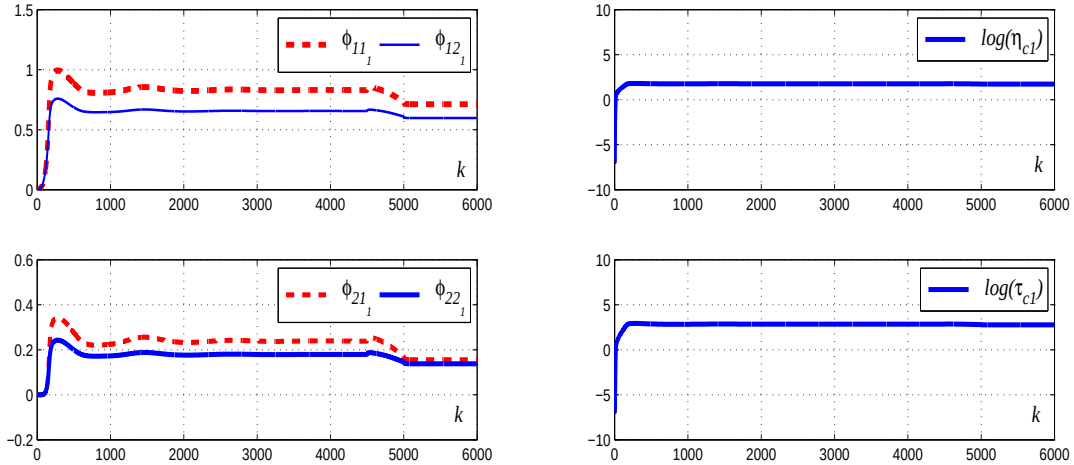
Tableau III.3: Les indices MSE_l et VAF_l ($l = 1, 2$) calculés pour les deux sorties du système SIMO dans les cas des émulations multimodèle et neuronale.

Ce tableau montre que l'utilisation de l'émulateur multimodèle permet une diminution des erreurs MSE_l et une augmentation d'environ 8% et 6% des VAF_l ($l = 1, 2$). Ces résultats permettent d'affirmer la bonne précision des bases élaborées et confirment l'efficacité de l'émulateur multimodèle proposé par rapport à l'émulateur neuronal.

III.5.1.2 Commande neuronale adaptative SIMO basée sur un émulateur multimodèle

Dans cette partie, une étude comparative entre l'utilisation des émulateurs neuronal et multimodèle dans le schéma de commande neuronale adaptative non carré est présentée. On reprend alors les mêmes conditions de simulation considérées dans la section III.3.1.1. Dans ce cas seulement les paramètres de démarrage des correcteurs partiels sont choisis pour assurer des bonnes performances en terme de poursuite et de régulation. En exploitant l'émulateur développé précédemment et basé sur les multimodèles (III.46) et (III.47), les paramètres de démarrage des correcteurs partiels retenus ($\varepsilon_{c_1} = \varepsilon_{c_2} = 40$) ont conduit aux résultats de commande donnés sur les figures III.22, III.23, III.24, III.25, III.26 et III.27.

Les variations des paramètres des deux correcteurs partiels sont données sur les figures III.22 et III.23. On constate que les amplitudes des poids des deux correcteurs s'adaptent toujours en fonction de la dynamique de la sortie correspondante.

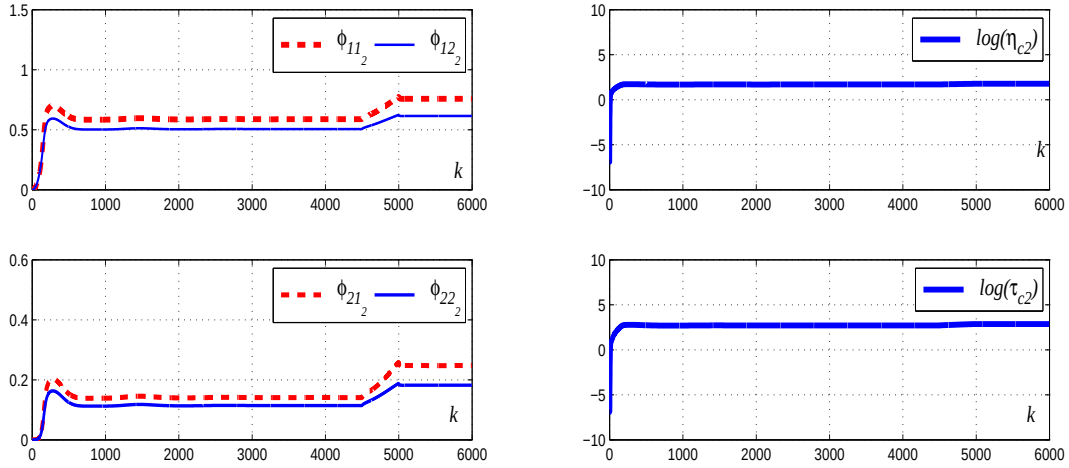


(a) Amplitude des poids

(b) $\eta_{c_1}(k)$ et $\tau_{c_1}(k)$ à l'échelle logarithmique

Figure III.22: Commande neuronale adaptative basée sur un émulateur multimodèle : variations des paramètres du premier correcteur neuronal partiel ($\varepsilon_{c_1} = 40$).

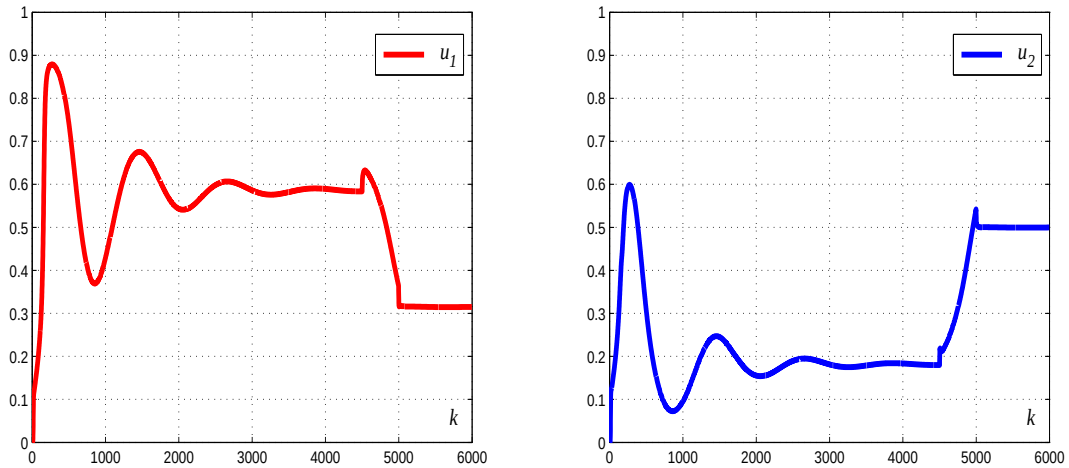
La figure III.24 donne la variation des commandes partielles enregistrés dans ce cas. A chaque instant k, les degrés de validité $\upsilon'_1(k)$ et $\upsilon'_2(k)$ (figure III.25) sont calculés en utilisant les relations (III.29) et (III.30).



(a) Amplitude des poids

(b) $\eta_{c_2}(k)$ et $\tau_{c_2}(k)$ à l'échelle logarithmique

Figure III.23: Commande neuronale adaptative basée sur un émulateur multimodèle : variations des paramètres du deuxième correcteur neuronal partiel ($\varepsilon_{c_2} = 40$).



(a) La sortie du premier contrôleur partiel

(b) La sortie du deuxième contrôleur partiel

Figure III.24: Commande neuronale adaptative basée sur un émulateur multimodèle : variations des lois de commande partielles ($\varepsilon_{c_1} = 40$ et $\varepsilon_{c_2} = 40$).

Le signal de commande effectif appliqué au système, obtenu par la fusion des deux lois de commande élémentaires pondérées chacune par le degré de validité correspondant, est donné sur la figure III.26. Par comparaison à la figure III.17, cette figure montre qu'en utilisant un émulateur multimodèle, la variance de commande se réduit considérablement et ne provoque aucune sollicitation de l'actionneur.

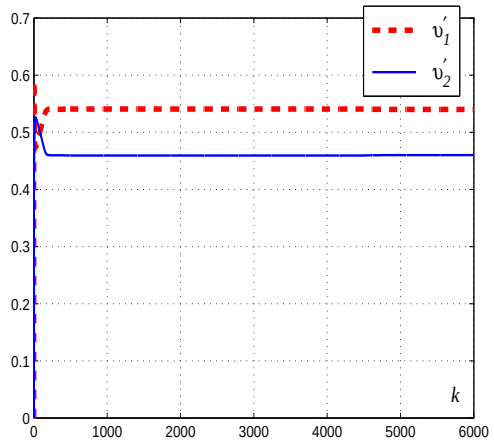


Figure III.25: Commande neuronale adaptative basée sur un émulateur multimodèle : variations des degrés de validité normalisés ($\varepsilon_{c_1} = 40$ et $\varepsilon_{c_2} = 40$).

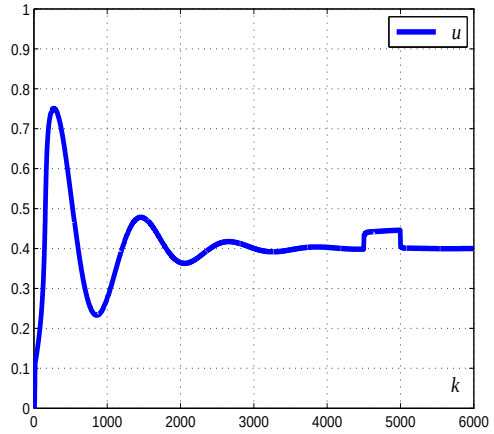
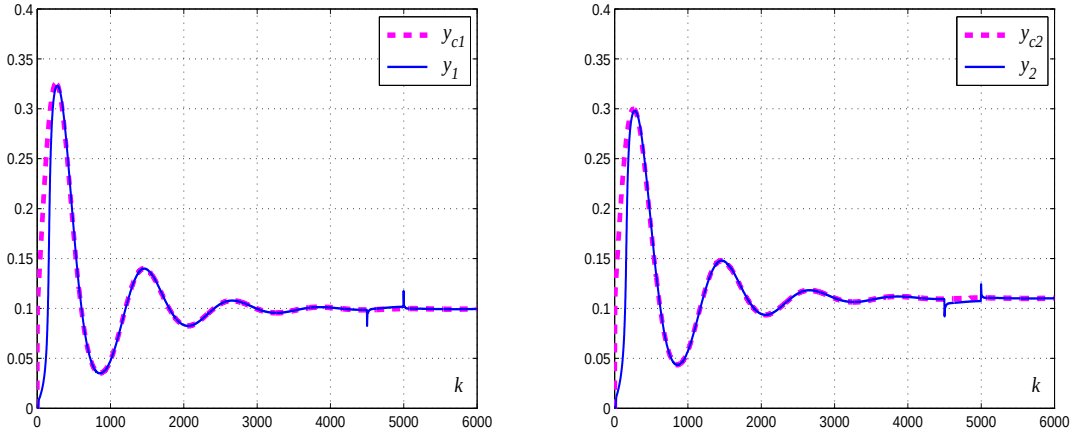


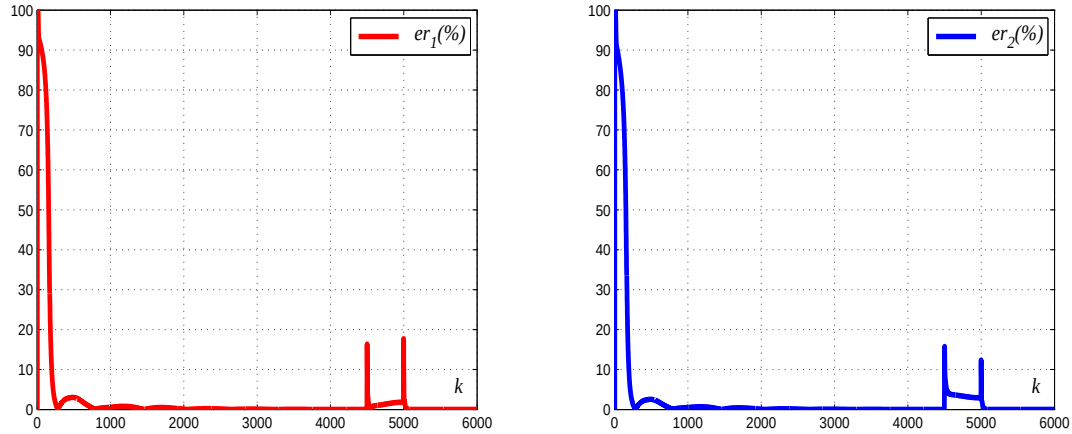
Figure III.26: Commande neuronale adaptative basée sur un émulateur multimodèle : variation de la loi de commande appliquée au système non linéaire SIMO ($\varepsilon_{c_1} = 40$ et $\varepsilon_{c_2} = 40$).

La variation des sorties réelles du système et des consignes est donnée sur la figure III.27. Il est clair que les performances enregistrées dans ce dernier cas sont nettement meilleures que celles obtenues avec un émulateur neuronal (figure III.18). En effet, cette figure montre de bonnes performances en poursuite. Un rejet de perturbation très satisfaisant est également enregistré dans la phase de régulation.
La figure III.28, donnant les erreurs relatives entre les sorties réelles et les sorties désirées, confirme les constatations précitées. En effet, les erreurs enregistrées dans le cas où un émulateur multimodèle est utilisé, sont relativement négligeable.



(a) Première sortie du système (b) Deuxième sortie du système

Figure III.27: Commande neuronale adaptative basée sur un émulateur multimodel ($\varepsilon_{c_1} = 40$ et $\varepsilon_{c_2} = 40$) : variations des sorties réelles et des consignes.



(a) Erreur relative (première sortie du système) (b) Erreur relative (deuxième sortie du système)

Figure III.28: Commande neuronale adaptative basée sur un émulateur multimodel ($\varepsilon_{c_1} = 40$ et $\varepsilon_{c_2} = 40$) : erreurs relatives entre les sorties réelles et les consignes.

Les résultats de simulation confirment que l'utilisation d'un émulateur multimodèle dans le schéma de commande neuronale SIMO conduit à des très bonnes performances en termes de poursuite et de régulation. L'émulateur multimodèle permet de résoudre les problèmes causés par un mauvais choix du paramètre de démarrage (ε_e) de l'émulateur neuronal et d'éviter les efforts nécessaires pour la recherche d'une valeur convenable de ce paramètre.

III.5.2 Cas d'un système non linéaire SIMO à une entrée et trois sorties

Pour la mise en évidence de l'efficacité du schéma de commande neuronale non carrée basée sur un émulateur multimodèle, on reprend dans cette section l'exemple du système non linéaire à une entrée et trois sorties défini précédemment par la relation III.31.

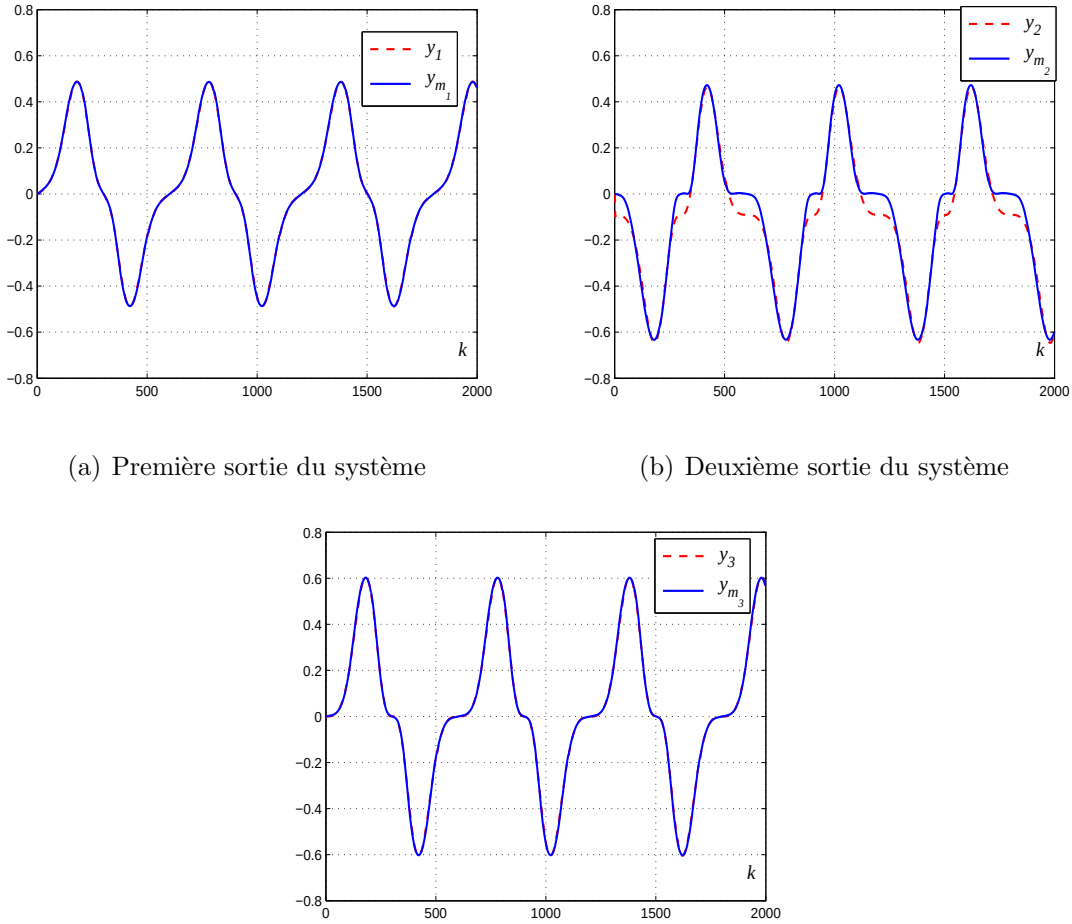
III.5.2.1 Un émulateur multimodèle SIMO (1 entrée-3 sorties)

En utilisant la méthode de classification développée précédemment, l'ensemble de données d'identification est traité pour générer l'ensemble de vecteurs de regression $\vartheta_{l,j} = [y_l(j), y_l(j-1), u(j-1)]^T$, $l = 1, 3$. Cette méthode donne naissance à cinq classes pour chaque sortie du système. L'entrée du système $u(k)$ est toujours considérée comme variable d'indexation, les centres et les dispersions obtenues sont, donc, résumés dans le tableaux III.4.

	Les paramètres de synthèse de l'émulateur multimodèle
$c_{1,i}$ et $\sigma_{1,i}$	$c_{1,1} = -0.8377$, $\sigma_{1,1} = 0.2456$ $c_{1,2} = -0.5920$, $\sigma_{1,2} = 0.2456$ $c_{1,3} = 0$, $\sigma_{1,3} = 0.5920$ $c_{1,4} = 0.5932$, $\sigma_{1,4} = 0.2445$ $c_{1,5} = 0.8377$, $\sigma_{1,5} = 0.2445$
$c_{2,i}$ et $\sigma_{2,i}$	$c_{2,1} = -0.8501$, $\sigma_{2,1} = 0.2594$ $c_{2,2} = -0.5907$, $\sigma_{2,2} = 0.2594$ $c_{2,3} = 0$, $\sigma_{2,3} = 0.5907$ $c_{2,4} = 0.5907$, $\sigma_{2,4} = 0.2594$ $c_{2,5} = 0.8501$, $\sigma_{2,5} = 0.2594$
$c_{3,i}$ et $\sigma_{3,i}$	$c_{3,1} = -0.8345$, $\sigma_{3,1} = 0.2498$ $c_{3,2} = -0.5847$, $\sigma_{3,2} = 0.2498$ $c_{3,3} = 0$, $\sigma_{3,3} = 0.5847$ $c_{3,4} = 0.5847$, $\sigma_{3,4} = 0.2498$ $c_{3,5} = 0.8345$, $\sigma_{3,5} = 0.2498$

Tableau III.4: Les centres et les dispersions des fonctions de pondération déterminés systématiquement pour l'émulation du systèmes non linéaires SIMO à une entrée et trois sorties.

Chaque sortie du système est représentée par un multimodèle SISO de cinq sous-modèles d'état de premier ordre. Les résultats de validation du multimodèle obtenus en utilisant la même séquence d'entrée utilisée pour valider l'émulateur neuronal (section III.3.1.2) sont donnés sur la figure III.29. Par comparaison aux résultats obtenus en utilisant l'émulateur neuronal (figure III.10), on peut aisément constater que l'émulateur multimodèle offre une meilleure précision de modélisation.



(a) Première sortie du système

(b) Deuxième sortie du système

(c) Troisième sortie du système

Figure III.29: Variation des sorties réelles du système non linéaire SIMO à une entrée et trois sorties et celles de l'émulateur multimodèle.

Les indices de performances MSE_l et VAF_l, $l = 1, 3$ sont calculés pour confirmer l'apport en performances, en terme de modélisation, de l'émulateur multimodèle proposé par comparaison à l'approche de modélisation neuronale. Le tableau III.5 résume ces indices dans le cas des émulateurs neuronal et multimodèle.

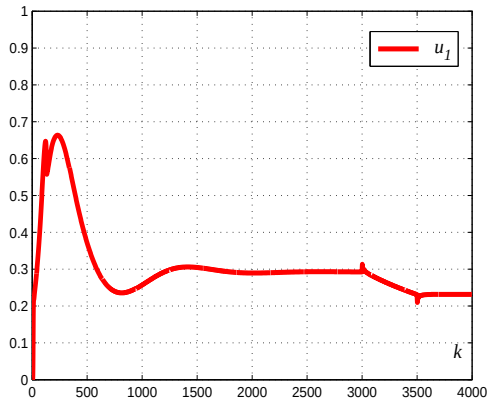
	Emulateur Neuronal ($\varepsilon_e = 0.1$)	Emulateur Multimodèle
MSE_1	$5.32\,10^{-4}$	$1.6\,10^{-5}$
MSE_2	$2.8\,10^{-3}$	$2.4\,10^{-3}$
MSE_3	$1.3\,10^{-3}$	$4.47\,10^{-5}$
VAF_1	99.28%	99.98%
VAF_2	97.8%	98.14%
VAF_3	98.74%	99.96%

Tableau III.5: Les indices MSE_l et VAF_l ($l = 1, 3$) calculés pour les trois sorties du système SIMO dans les cas des émulations multimodèle et neuronale.

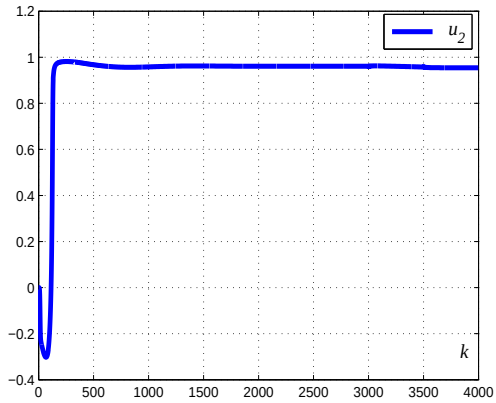
III.5.2.2 Commande neuronale adaptative SIMO basée sur un émulateur multimodèle

L'émulateur multimodèle obtenu est maintenant exploité dans le schéma de commande neuronale SIMO proposé. Dans les mêmes conditions de simulation décrites dans la section III.3.1.2 et avec un choix des des trois termes de démarrage des correcteurs $\varepsilon_{c_1} = 50$, $\varepsilon_{c_2} = 45$ et $\varepsilon_{c_3} = 50$ on obtient les résultats illustrés sur les figures III.30, III.31, III.32 et III.33.

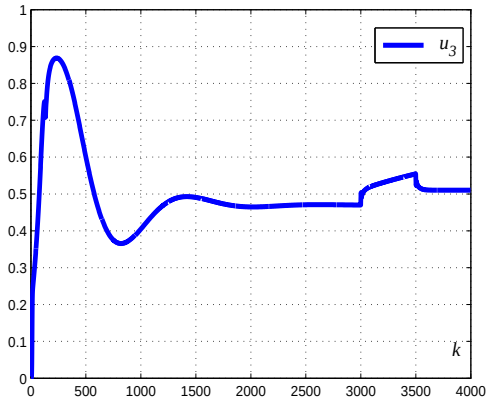
La figure III.30 donne la variation des commandes partielles enregistrée dans ce cas. Les degrés de validité correspondant à chacune de ces trois commandes, calculés en utilisant les relations III.32 et III.33, sont représentés sur la figure III.31. La commande effective appliquée au système est donnée sur la figure III.32. Ces figures montrent des résultats comparables à ceux obtenus par l'utilisation d'un émulateur neuronal.



(a) La sortie du premier contrôleur partiel



(b) La sortie du deuxième contrôleur partiel



(c) La sortie du troisième contrôleur partiel

Figure III.30: Commande neuronale adaptative basée sur un émulateur multimodèle : variations des lois de commande partielles ($\varepsilon_{c_1} = 50$, $\varepsilon_{c_2} = 45$ et $\varepsilon_{c_3} = 50$).

La figure III.33 donne les variations des sorties désirées et des sorties réelles obtenues en utilisant un émulateur multimodèle pour la commande du système à une entrée et trois sorties. Cette figure montre des bonnes performances en poursuite. Le schéma de commande neuronale SIMO a permis, dans ce cas, de rejeter rapidement la perturbation affectant les sorties du système SIMO (1 entrée-3 sorties).

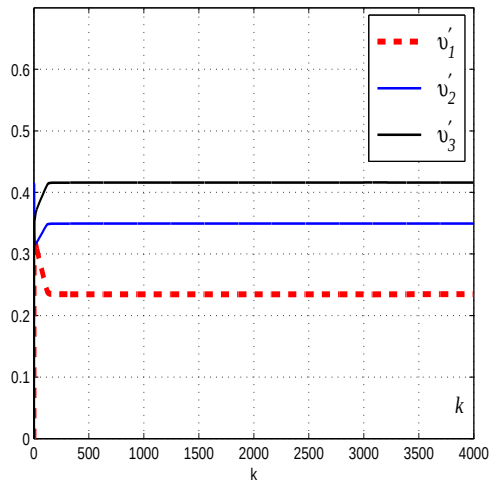


Figure III.31: Commande neuronale adaptative basée sur un émulateur multimodèle : les degrés de validité normalisés ($\varepsilon_{c_1} = 50$, $\varepsilon_{c_2} = 45$ et $\varepsilon_{c_3} = 50$).

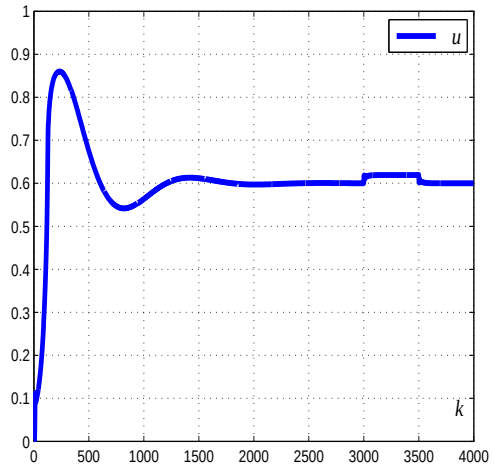
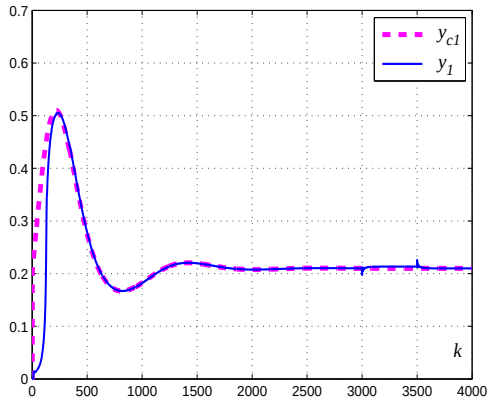
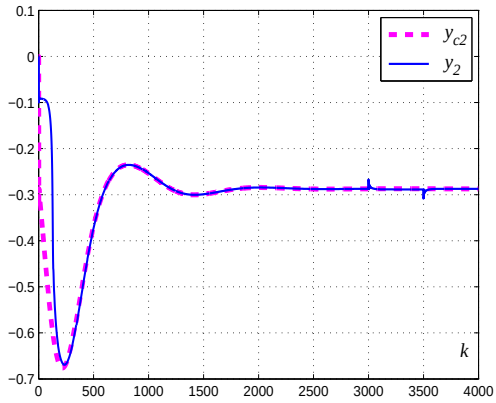


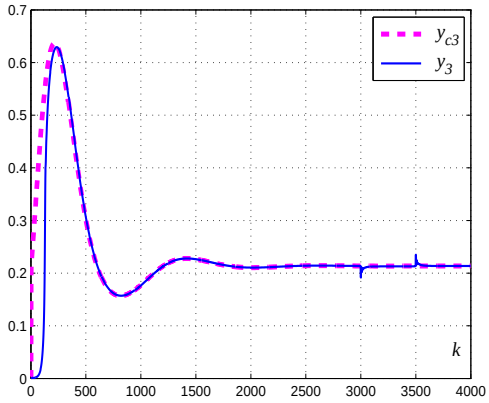
Figure III.32: Commande neuronale adaptative basée sur un émulateur multimodèle : variation de la loi de commande appliquée au système non linéaire SIMO ($\varepsilon_{c_1} = 50$, $\varepsilon_{c_2} = 45$ et $\varepsilon_{c_3} = 50$).



(a) Première sortie du système



(b) Deuxième sortie du système



(c) Troisième sortie du système

Figure III.33: Commande neuronale adaptative SIMO basée sur un émulateur multimodèle ($\varepsilon_{c_1} = 50$, $\varepsilon_{c_2} = 45$ et $\varepsilon_{c_3} = 50$) : variations des sorties réelles et désirées.

III.6 Conclusion

Dans ce chapitre, une commande neuronale adaptative est proposée pour résoudre les problèmes de commande des systèmes non linéaires SIMO. Principalement, trois contributions sont présentées. Dans une première partie, un nouveau schéma de commande neuronal non carré est obtenu par la fusion d'un ensemble de correcteurs neuronaux carrés. Ce schéma nécessite l'utilisation d'un émulateur qui permet le calcul des dérivées partielles des sorties par rapport au signal de commande qui sont nécessaires pour l'adaptation des paramètres des correcteurs partiels. Pour résoudre ce problème, un émulateur neuronal non carré est proposé comme une première solution. Une nouvelle formule est, aussi, proposée pour calculer les degrés de validité. Ces validités sont utilisées pour évaluer les contributions des correcteurs partiels et calculer la commande globale à appliquer au système SIMO considéré.
L'avantage principal du système de commande proposé est lié à la simplicité de la structure des correcteurs partiels qui facilite la mise en oeuvre de la loi de commande adéquate. Les résultats de simulation illustrent clairement que l'émulateur neuronal conduit à une estimation satisfaisante des sorties du système SIMO et elles montrent également que la stratégie de commande proposée conduit à des bonnes performances en boucle fermée.
Bien que ce schéma ait montré sa capacité à commander les systèmes non linéaires SIMO, des simulations numériques montrent que l'utilisation d'un émulateur neuronal peut affecter les performances de la commande en boucle fermée. Dans une deuxième partie de ce chapitre, et pour résoudre ce problème, un schéma d'une commande neuronale non carré basé sur un émulateur multimodèle est proposé. Les résultats de simulation montrent clairement que l'utilisation d'un émulateur multimodèle dans le schéma de commande des systèmes non linéaires SIMO conduit à des bonnes performances relativement au cas où un émulateur neuronal est utilisé.

Conclusion générale

Les travaux de recherche proposés dans cette thèse apportent des contributions aux méthodes de représentation et de commande des systèmes non linéaires multivariables MIMO et SIMO dont les dynamiques sont inconnues.
Un schéma de commande adaptative composé de deux réseaux de neurones récurrents est présenté. Le premier réseau sert d'émulateur de la dynamique du système étudié et le deuxième sert de correcteur. Les deux réseaux s'adaptent par l'algorithme RTRL à partir des conditions initiales nulles grâce à deux paramètres de démarrage. Des simulations numériques ont montré que l'utilisation d'un émulateur neuronal peut affecter les performances en boucle fermée à cause d'une mauvaise initialisation du paramètre de démarrage et à cause d'une adaptation relativement lente par rapport à la rapidité du changement des consignes.

Un émulateur multimodèle découplé est, donc, proposé pour remplacer l'émulateur neuronal et représenter la dynamique du système considéré. En utilisant cet émulateur, chaque sortie du système est prédite en utilisant une base de modèles partiels qui résultent d'une procédure d'identification hors ligne du système non linéaire multivariable. La contribution de chaque modèle partiel est estimée par une fonction d'activation (ou de pondération). Les performances offertes par cet émulateur en termes de représentation et de commande sont nettement améliorées par rapport à celles de l'émulateur neuronal. L'avantage principal du schéma utilisant un émulateur multimodèle est aussi de réduire considérablement les calculs à effectuer en ligne. Aucun paramètre d'initialisation, et aucune adaptation en ligne ne sont nécessaires. Cela permet de résoudre le problème lié au choix des paramètres de démarrage de l'émulateur neuronal.
Dans ce cadre, on a proposé, par ailleurs, une nouvelle méthode basée sur la classification de données d'identification pour la décomposition de l'espace de fonctionnement du système et la génération systématique des fonctions de pondération. Cette approche a permis d'éviter tous les efforts à déployer pour la recherche des valeurs optimales des paramètres de synthèses. Plusieurs simulations numériques sont réalisées et elles ont pu mettre en évidence que l'émulateur multimodèle conduit à de bonnes performances en boucle ouverte et en boucle fermée.

Le développement d'un schéma de commande neuronale adaptative non carré et basé sur des émulateurs neuronal et multimodèle SIMO est également proposé pour résoudre les problèmes liés à la commande des systèmes non linéaires SIMO. Cette nouvelle stratégie de commande consiste à développer un ensemble de correcteurs neuronaux partiels avec des structures carrées. La commande effective à appliquer au système SIMO est une fusion des commandes issues des correcteurs partiels. Une nouvelle formule est, également, proposée pour calculer les degrés de validité utilisées pour évaluer les contributions des correcteurs partiels. L'avantage principal du système de commande proposé est lié à la simplicité de la structure des correcteurs partiels qui facilite la mise en oeuvre de la loi de commande adéquate. Les simulations numériques proposées ont montré l'efficacité de ce schéma de commande et elles ont permis d'établir une comparaison entre les performances des émulateurs neuronal et multimodèle SIMO.

Les résultats proposés dans cette thèse s'ouvrent sur de multiples perspectives.

Afin d'étendre le schéma de commande neuronale SIMO proposé dans cette thèse à d'autres domaines d'application, il est important de généraliser tout d'abord ce schéma pour les systèmes non linéaires multivariables où le nombre de variables de commande est supérieur à un. Ces systèmes sous-actionnés apparaissent dans une large classe d'applications. Le sous-actionnement est une propriété qui peut être due principalement à l'une des raisons suivantes :

- Il peut être dû à la dynamique propre du système (les avions, les satellites, les hélicoptères, les véhicules sous-marins, les motrices électriques...).
- Le sous-actionnement peut aussi être introduit volontairement pour certaines raisons pratiques ou pour réduire le nombre d'actionneurs et, par conséquent, le coût de construction du prototype (les robots à liaisons flexibles, les navires, les robots mobiles...).
- La panne d'actionneurs pour les systèmes pleinement actionnés conduit au sous-actionnement, comme dans le cas d'un avion, d'un navire de surface, des engins spatiaux. Dans ces cas, les pannes peuvent provoquer des catastrophes.
- Le sous-actionnement peut aussi être imposé artificiellement pour créer des systèmes non linéaires complexes d'ordre inférieur dans le but d'obtenir un aperçu sur la commande des systèmes sous-actionnés d'ordre supérieur (le Pendubot, l'Acrobot, le système Beam and Ball...)

Pour réaliser cette généralisation, deux idées peuvent être envisagées. La première idée consiste à dupliquer le nombre de multi-correcteurs et de développer un schéma de commande avec N_{IN} (le nombre de variables de commande) correcteurs SIMO. La deuxième idée consiste à garder la même structure du correcteur partiel SIMO proposé et de trouver une nouvelle formule pour calculer les validités des correcteurs partiels et une nouvelle

méthode pour évaluer les contributions de ces correcteurs et obtenir les N_{IN} commandes à appliquer effectivement au système non linéaire sous-actionné.

Le schéma étendu et même le schéma proposé dans le troisième chapitre de cette thèse, peuvent être considérés comme la pierre angulaire de la synthèse d'une commande neuronale adaptative des systèmes sous-actionnés et particulièrement, les cas du Pendubot et d'un navire sous actionné.

Il sera aussi important d'exploiter le schéma de commande sous actionné comme une commande tolérante aux défauts lorsque le sous-actionnement est dû à une panne de l'un des actionneurs. Une méthode qui permette la commutation systématique du schéma de commande neuronal carré (proposé dans les deux premiers chapitres) vers un schéma de commande non carré sous-actionné pourra être étudiée.

Bibliographie

[1] A. Akhenak. *Conception d'observateurs non linéaires par approche multimodèle : application au diagnostic.* Thèse de Doctorat de l'Institut National Polytechnique de Lorraine, Decembre 2004.

[2] A. Astolfi, D. Karagiannis, and R. Ortega. *Nonlinear and Adaptive Control with Applications.* Springer, Berlin, 2008.

[3] A. Atig. *Sur la Commande Neuronale Adaptative des Systèmes Non Linéiares.* Université de Gabès, Ecole Nationale d'Ingénieurs de Gabès, 2012.

[4] A. Atig, F. Druaux, D. Lefebvre, K. Abderrahim, and R. Ben Abdennour. Neural emulator and controller with decoupled adaptive rates for nonlinear systems : application to chemical reactors. *International Journal on Sciences and Techniques of Automatic Control and Computer Engineering (IJ-STA)*, 5(1) :1500–1515, 2010.

[5] A. Atig, F. Druaux, D. Lefebvre, K. Abderrahim, and R. Ben Abdennour. A new neural adaptive control based on neural emulation of complex square systems. *International Review of Automatic Control, IREACO*, 3(6) :612–623, 2010.

[6] A. Atig, F. Druaux, D. Lefebvre, K. Abderrahim, and R. Ben Abdennour. Adaptive control design using stability analysis and tracking errors dynamics for nonlinear square MIMO systems. *Engineering Applications of Artificial Intelligencecial Intelligence (EAAI), Elsevier*, 25 :1450–1459, 2012.

[7] N. Bahri, A. Atig, R. Ben Abdennour, F. Druaux, and D. Lefebvre. A multimodel emulator for non linear system controls. *International Journal on Sciences and Techniques of Automatic Control and Computer Engineering (IJ-STA)*, 5(1) :1500–1515, 2011.

[8] N. Bahri, A. Atig, R. Ben Abdennour, F. Druaux, and D. Lefebvre. Multimodel and neural emulators for non-linear systems : Application to an indirect adaptive neural control. *International Journal of Modelling, Identification and Control*, 17 :348–359, 2012.

[9] N. Bahri, A. Atig, R. Ben Abdennour, F. Druaux, and D. Lefebvre. Multivariable adaptive neural control based on multimodel emulator for nonlinear square MIMO systems. *Transactions on Systems, Signals & Devices*, 10(1) :1–20, 2015.

[10] N. Bahri, A. Atig, R. Ben Abdennour, F. Druaux, and D. Lefebvre. Multivariable adaptive neural control based on multimodel emulator for nonlinear square MIMO systems. In *11 International Multi-Conference on Systems, Signals and Devices (SSD)*, Barcelone, Spain, 2014.

[11] N. Bahri, A. Atig, R. Ben Abdennour, F. Druaux, and D. Lefebvre. Emulation of multivariable non square and nonlinear systems. In *14 International Conference on Sciences and Techniques of Automatic control and computer engineering (STA'13)*, Sousse, 2013.

[12] T. Bakhtiar and S. Hara. H2 regulation performance limitations for SIMO linear time-invariant feedback control systems. *Automatica*, 44 :pp. 659–670, 2008.

[13] O. Begovich and E N. Sanchez. Takagi-Sugeno fuzzy scheme for real-time trajectory tracking of an underactuated robot. *IEEE Transactions on Control Systems Technology*, 10(1) :14–20, 2002.

[14] A. Charara, J. De Miras, and B. Caron. Nonlinear control of a magnetic levitation system without premagnetisation. *IEEE Transactions on Control Systems Technology, Special Issue on Magnetic Bearing Control*, 5(4) :513–523, September 1996.

[15] G. Chen, J. Chen, and R. H. Middleton. Optimal tracking performance for SIMO systems. *IEEE Transactions on Automatic Control*, 47 :1770–1775, 2002.

[16] S. L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2 :267–278, 1994.

[17] K. Dehri. *Sur le rejet des perturbations harmoniques par les régimes glissants.* Université de Gabès, Ecole Nationale d'Ingénieurs de Gabès, Tunisie, 2012.

[18] S. Devasia, D. Chen, and B. Paden. Nonlinear inversion-based output tracking. *IEEE Transactions on Automatic Control*, 41(7) :930–942, July 1996.

[19] D. Filev. Fuzzy modeling of complex systems. *International Journal of Approximate Reasonning*, 5(3) :281–290, 1991.

[20] K. Gasso. *Identification des systèmes dynamiques non-linéaires : Approche multi-modèle.* Thèse de Doctorat de l'Institut National Polytechnique de Lorraine, Décembre 2000.

[21] K. Gasso, G. Mourot, and J. Ragot. Identification of an output error Takagi-Sugeno model. *IEEE International Conference on Systems, Man and Cybernetics,*, 1 :14–19, 2000.

[22] K. Gasso, G. Mourot, and J. Ragot. Structure identification in multiple model representation : Elimination and merging of local models. *IEEE Conference on Decision and Control*, 3 :2992–2997, Orlando, Florida, 2001.

[23] S. S. Ge and C. Wang. Direct adaptive NN control of a class of nonlinear systems. *IEEE Transactions on Neural Networks*, 13(1) :214–221, 2002.

[24] S. Hara, T. Bakhtiar, and M. Kanno. The best achievable H_2 tracking performances for SIMO feedback control systems. *Journal of Control Science and Engineering (JCSE)*, 7 :1–12, 2007.

[25] J. Huang. Asymptotic tracking of a nonminimum phase nonlinear system with nonhyperbolic zero dynamics. *IEEE Transactions on Automatic Control*, 45(3) :542–546, March 2000.

[26] K. J. Hunt, R. Haas, and R. Murray-Smith. Extending the functional equivalence of radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, 7(3) :776–781, 1996.

[27] A. Isidori. *Nonlinear Control Systems.* Springer Verlag Berlin, Heidelberg, 1989.

[28] T. A. Johansen and A. B. Foss. Constructing NARMAX using ARMAX models. *International Journal of Control*, 58(5) :1125–1153, 1993.

[29] T. A. Johansen and A. B. Foss. Non linear local model representation for adaptive systems. *IEEE International Conference on Intelligent Control and Instrumentation*, 2 :677–682, Singapore, 1992.

[30] I. Kanellakopoulos, P.V. Kokotovik, and R. Morse. Systematic design of adaptive controllers for feedback linearisable systems. *IEEE Transactions on Automatic Control*, 36(11) :1241–1253, 1991.

[31] M. Krstic, I. Kanellakopoulos, and P. Kokotovic. *Nonlinear and Adaptive Control Design.* A John Wiley & Sons Interscience Publication, New York, USA 1995.

[32] I. Lagrat, H. Ouakka, and I. Boumhidi. Adaptive control of a class of nonlinear systems based on Takagi-Sugeno fuzzy models. *Proceeding Information and Communication Technologies International Symposium, ICTIS 07, Fez, Morrocco*, pages 594–597, 2007.

[33] D. A Lawrence and W. J. Rugh. Gain scheduling dynamic linear controllers for a nonlinear plant. *Automatica*, 31 :381–390, 1995.

[34] X. Litrico. Robust IMC flow control of SIMO dam-river open-channel systems. *IEEE transaction on control systems technology*, 10(3) :432–437, 2002.

[35] B. Liu, H. b. He, and S. Chen. Adaptive dual network design for a class of SIMO systems with nonlinear time-variant uncertainties. *Automatica*, 36(4) :564–572, 2010.

[36] M. Ltaief, K. Abderrahim, R. Ben Abdennour, and M. Ksouri. Contributions to the multimodel approach : Systematic determination of a models' base and validities estimation. *International Journal of Automation and Systems Engineering (JASE)*, 2(3) :39–51, 2008.

[37] M. Ltaief, A. Messaoud, and R. Ben Abdennour. An optimal systematic determination of models' base for multimodel representation : Real time application. *International Journal of Automation and Computing (IJAC)*, 11(6) :644–652, 2014.

[38] M. Ltaif, K. Abderrahim, R. Ben Abdennour, and M. Ksouri. Systematic determination of a model's base for the multimodel approach : Experimental validation. *WSEAS Transactions on Electronics*, 1(2) :331–336, 2004.

[39] J. Lévine, J. Lottin, and J.C Ponsart. A nonlinear approach to the control of magnetic bearings. *IEEE Transactions on Control Systems Technology, Special Issue on Magnetic Bearing Control*, 5(4) :524–544, 1996.

[40] D. Marquard. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2) :431–441, 1963.

[41] A. Messaoud. *Sur la Représentation et la Commande prédictive Multimodèle des Systèmes Complexes.* Thèse de Doctorat de l'Ecole Nationale d'Ingénieurs de Gabèse, Université de Gabès, 2010.

[42] A. Messaoud, M. Ltaief, and R. Ben Abdennour. Supervision based on partial predictors for a multimodel generalized predictive control : Experimental validation on a semi-batch reactor. *International Journal of Modelling, Identification and Control*, 6(4) :333–340, 2009.

[43] A. Messaoud, M. Ltaief, and R. Ben Abdennour. Partial predictors for the supervision of a multimodel direct generalized predictive control. In *American Control Conference*, Seattle, Washington, USA, 2008.

[44] M. Mihoub. *Contributions à l'observation des systèmes complexes en régime glissant.* Thèse de Doctorat de l'Ecole nationale d'ingénieurs de Gabès, Université de Gabès, Tunisie, 2010.

[45] M. Mihoub, A. S. Nouri, and R. Ben Abdennour. Real-time application of discrete second order sliding mode control to a chemical reactor. *Control Engineering Practice*, 17(9) :1089–1095, 2009.

[46] R. Murray-Smith and T. A. Johansen. *Multiple Model Approaches to Modelling and Control*. Taylor & Francis, London, 1997.

[47] A. M. Nagy. *Analyse et synthèse de multimodèles pour le diagnostic. Application à une station d'épuration*. Thèse de Doctorat de l'Institut National Polytechnique de Lorraine Nancy-Université, Novembre 2010.

[48] K. Narendra and J. Balakrishnan. Adaptive control using multiple models. *IEEE Transactions on Automatic Control*, 42(2) :171–187.

[49] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1) :4–27, 1991.

[50] O. Nelles. *Nonlinear System Identification*. Springer-Verlag, Berlin Heidelberg, 2001.

[51] R. Olfati-Saber. *Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles*. Massachusetts Institute of Technology, 2001.

[52] R. Olfati-Saber. Normal forms for underactuated mechanical systems with symmetry. *IEEE Transactions on automatic control*, 47(2) :305–308, 2002.

[53] R. Orjuela. *Contribution à l'estimation d'état et au diagnostic des systèmes représentés par des multimodèles*. Thèse de Doctorat de l'Institut National Polytechnique de Lorraine, Novembre 2008.

[54] R. Orjuela, B. Marx, D. Maquin, and J. Ragot. Design of robust H_∞ observers for nonlinear systems using a multiple model. In 17th *IFAC World Congress, Séoul, Corée du Sud.*, 2008.

[55] H. Ouakka. *Contribution à l'Identification et la Commande Floue d'une Classe de Systèmes Non Linéaires*. Thèse de Doctorat de la Faculté des Sciences Dhar El Mehraz de Fes, Université Sidi Mohamed Ben Abdellah, Juin 2009.

[56] R. Ouiguini, R. Bouzid, and Y. Sellami. Une commande robuste par mode glissant flou appliquée à la poursuite de trajectoire d'un robot mobile non holonome. In *Conférence Internationale sur les Systèmes de Télécommunications, d'Electronique Médicale et d'Automatique*, Tlemcen, Algérie, 2003.

[57] J. C. Ponsart. *Asservissements numériques de paliers magnétiques. Application aux pompes à vide*. Thèse de Doctorat de l'Université de Savoie, Décembre 1996.

[58] D. Qian, J. Yi, and D. Zhao. Hierarchical sliding mode control for a class of SIMO under-actuated systems. *Control and Cybernetics*, 37(1) :159–175, 2008.

[59] J. S. Shamma and M. Athans. Analysis of gain scheduled control for nonlinear plants. *IEEE Transactions on Automatic Control*, 35(8) :898–907, 1990.

[60] A. Shigeo. *Neural networks and fuzzy systems theory and applications.* Kluwer Academic Publishers USA, 1997.

[61] J. Slotine and W. Li. *Applied Nonlinear Control.* Prentice Hall, New Jersey, 1991.

[62] M. Smaoui, X. Brun, and D. Thomasset. Etude d'un différentiateur robuste via mode glissants d'ordre supérieur : application en électropneumatique. In *Journées Doctorales et Nationales du GDR MACS,*, Lyon, France, 2005.

[63] M.W. Spong. The swing up control problem for the acrobot. *IEEE Control Systems Magazine*, 15 :49–55, 1995.

[64] S. Talmoudi, K. Abderrahim, R. Ben Abdennour, and M. Ksouri. Experimental validation of a systematic determination approach of a model's base. *WSEAS Transaction of Electronics*, 1(2) :410–416, 2004.

[65] K. Tanaka and H. O. Wang. *Fuzzy Control Systems Design and Analysis : A Linear Matrix Inequality Approach.* Wiley-Interscience Publication, 2001.

[66] M. C. M. Texeira and S. H. Zak. Stabilizing controller design for uncertain nonlinear systems using fuzzy models. *IEEE Transactions on Fuzzy Systems*, 7(2) :133–142, April 1999.

[67] L. Thiaw. *Identification de systèmes dynamiques non-linéaires par réseaux de neurones et multimodèles.* Thèse de Doctorat de l'Université Paris XII, 2008.

[68] A. Titli and L. Foulloy. *La logique floue.* Observatoire Francais des techniques avancées, Masson, 1994.

[69] V. Verdut. *Non linear systems Identification : A state space approach.* Thèse de Doctorat University of Twente, Hollande, 2002.

[70] R. R. Yager and D. P. Filev. Approximate clustering via the mountain method. *IEEE Transactions on Systems, Man and Cybernetics*, 24(8) :1279–1284, 1994.

[71] R. R. Yager and D. P. Filev. Generation of fuzzy rules by mountain clustering. *Journal of Intelligent and Fuzzy Systems*, 2 :209–2019, 1994.

[72] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3(1) :28–44, 1973.

[73] S. Zerkaoui. *Commande neuronale adaptative des systèmes non linéaires.* Thèse de Doctorat de l'Université du Havre, Le Havre, France, 2007.

[74] S. Zerkaoui, F. Druaux, E. Leclercq, and D. Lefebvre. Stable adaptive control with recurrent neural networks for square MIMO nonlinear systems. *Engineering Applications of Artificial Intelligence*, 12(4-5) :702–717, 2009.

[75] S. Zerkaoui, F. Druaux, E. Leclercq, and D. Lefebvre. Indirect neural control for plant-wide systems : application to the tennessee eastman challenge process. *Computers and Chemical Engineering*, 34 :232–243, 2010.

[76] M. Zhiong, H.R. Wu, and M. Palaniswami. An adaptive tracking controller using neural network for a class of nonlinear systems. *IEEE Transactions on Neural Networks*, 9(5) :947–955, 1998.

Résumé

Dans le présent travail, une commande neuronale adaptative est proposée pour les systèmes non linéaires MIMO et SIMO. Pour le calcul en temps réel des variations des sorties en fonction des variations des entrées, des émulateurs neuronaux sont développés. L'approche multimodèle est, aussi, proposée pour la représentation et l'émulation des systèmes non linéaires multivariables. Dans ce contexte et pour améliorer les performances de l'émulation, une méthode systématique pour la détermination des paramètres de synthèse du multimodèle est également avancée.
Un nouveau schéma de commande neuronale non carré, basé sur des émulateurs (neuronal et multimodèle), est proposé pour les systèmes non linéaires SIMO. Dans ce dernier cas, des nouvelles formules sont proposées pour le calcul des degrés de validité des commandes partielles. L'apport notable en performances des différentes stratégies développées est évalué en poursuite et en régulation par des simulations numériques et sur des systèmes non linéaires MIMO et SIMO.

Mots-clés: *Systèmes Non Linèaires, Systèmes SIMO et MIMO, Réseaux de Neurones, Commande Adaptative Indirecte, Multimodèle Découplé, Optimisation Non Linéaire, Classification.*

Abstract

In this present work, an adaptive neural control is proposed for nonlinear MIMO and SIMO systems. For online calculation of outputs variations in function of inputs variations, neural emulators are developed. The multimodel approach is, also, proposed for the representation and emulation of nonlinear multivariable systems. In this context and in order to improve the emulation performance, a systematic method for the determination of multimodel synthesis parameters is also advanced.
A new non-square scheme of an adaptive neural control based on emulators (neural and multimodel) is proposed for nonlinear SIMO systems. In this last case, new formulas are proposed to compute the validity degrees of partial controls. The obtained good performance in regulation and tracking aspects of the different proposed strategies is evaluated in numerical simulations and in the case of nonlinear MIMO and SIMO systems.

Keywords: *Non Linear Systems, SIMO and MIMO Systems, Neural Networks, Indirect Adaptive Control, Uncoupled Multimodel, Non Linear Optimisation, Classification.*