Identification de systèmes dynamiques non linéaires par réseaux de neurones et multimodèles

Lamine Thiaw

Université Paris XII Val de Marne

Laboratoire Images, Signaux et Systèmes Intelligents

Université Paris XII

École Doctorale Sciences et ingénierie : matériaux, modélisation et environnement (SIMME)

Doctorat

Spécialité : Genie Informatique, Automatique et Traitement du Signal

Auteur : **Lamine THIAW**

Identification de systèmes dynamiques non-linéaires par réseaux de neurones et multimodèles

Thèse soutenue le : 28 janvier 2008

Jury :

Patrick GARDA	Rapporteur
Thierry POINOT	Rapporteur
Hichem MAAREF	Examinateur
Gustave SOW	Invité
Rachid MALTI	Examinateur
Kurosh MADANI	Directeur de thèse

LiSSi - Université Paris12 Val-de-Marne (Bâtiment P2 - Marie Curie 2ème étage) 61, avenue du Général de Gaulle 94010, Créteil cedex
Tél : 01.45.17.14.94 Fax : 01.45.17.14.92 http ://lissi.univ-paris12.fr

Remerciements

Je tiens à remercier Monsieur Patrick GARDA, Professeur à l'Université de Pierre et Marie Curie et Monsieur Thierry POINOT, Professeur à l'École Supérieure d'Ingénieurs de Poitiers, pour avoir accepté de rapporter ce travail de thèse. Je les remercie également pour les conseils qu'ils m'ont donnés lors de nos rencontres. Merci également à Monsieur Hichem MAAREF, Professeur à l'Université d'Evry Val d'Essonne pour l'intérêt qu'il porte à mon travail en acceptant de faire partie de mon jury de thèse qu'il a accepté d présider.

Mes très sincères remerciements à :

- mon directeur de thèse, Monsieur Kurosh MADANI, Professeur à l'Université de Paris 12, qui a bien voulu m'accueillir dans son Laboratoire et guidé mes pas jusqu'au terme de ce travail. Je le suis reconnaissant pour sa disponibilité, ses conseils et tout le soutien qu'il m'a apporté.
- mon co-encadreur de thèse, Monsieur Rachid MALTI, maître de conférence à l'Université de Bordeaux 1, pour sa disponibilité et le soutien qu'il m'a apporté. Je garderai en mémoire son goût de la perfection et la rigueur scientifique qu'il m'a toujours imposée.
- Monsieur Gustave SOW, Responsable du Laboratoire d'Energies Renouvelable de l'École Supérieure Polytechnique de Dakar, pour avoir accepté de faire partie de mon jury, malgré la distance et un emploi de temps très chargé. J'ai toujours pu trouver auprès de lui le soutien nécessaire pour mener à bien mes activités de recherche.
- Abdennasser CHEBIRA, Amine CHOHRA, Véronique AMARGER, Christophe SABOURIN, pour leur soutien tout au long de ce travail de thèse.
- mes collègues doctorants et docteurs, plus particulièrement : Saliou, Sofianne, Nadia, Moustapha, Ivan, Arash, Weiwei, Mathieu, Samira, Mohammed, pour leur soutien exemplaire et l'ambiance très chaleureuse qu'ils ont toujours su faire régner.
- tous mes collègues de l'Ecole Supérieure Polytechnique de Dakar,
- M. Soussou SAMBOU, enseignant-chercheur à l'Université Cheikh Anta Diop de Dakar, et à travers lui l'OMVS, pour avoir mis à ma disposition des données sur le fleuve Sénégal qui ont servi à la validation de certains modèles.

J'adresse une mention spéciale à mon épouse **Khady** ainsi qu'à toute ma famille et à tous mes amis pour leur soutien inestimable.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Ce travail de thèse est issu d'une collaboration entre le Laboratoire Intelligence dans les Instrumentations et les Systèmes (I^2S, qui a intégré le Laboratoire Images, Signaux et Systèmes Intelligents - LiSSi) de l'Université Paris 12 et le Laboratoire d'Energies Renouvelables (LER) de l'École Supérieure Polytechnique de Dakar. Les séjours alternés entre le LiSSi et le LER (séjour en France de quatre mois par an) ont été financés par l'ÉGIDE (Centre Français pour l'Accueil et les Échanges Internationaux). Le thème de recherche rentre dans le cadre de la modélisation expérimentale, l'objectif étant la mise en place de méthodologies d'identification pour la commande, la simulation ou le diagnostic de systèmes et également pour la modélisation de phénomènes environnementaux. Par ailleurs cette thèse a fait l'objet d'une collaboration avec le Laboratoire de l'Intégration du Matériau au Système (IMS) de l'Université de Bordeaux 1.

L'étude d'un système dynamique non linéaire, quelle que soit sa nature (industrielle, environnementale, financière, etc.) et quel que soit l'objectif visé (commande, optimisation du fonctionnement, prédiction, analyse du comportement, etc.), nécessite la mise en place d'une représentation capable de reproduire son comportement. Cette représentation, communément appelée modèle, permet dans certaines applications de simuler le comportement du système, notamment lorsque l'expérimentation est coûteuse.

Parmi les différents types de modélisation qui existent, la modélisation mathématique connaît plus de succès grâce au progrès de l'informatique qui a permis une avancée significative des méthodes de calcul numérique. La modélisation mathématique d'un système est une représentation mathématique sous forme d'une relation liant les différentes variables régissant son fonctionnement. Cependant, la détermination de ces variables et de la structure de la relation qui les lie constitue souvent un problème majeur.

Pour certains systèmes, il est possible d'établir ces relations à partir de connaissances physiques, chimiques, biologiques ou autres : une telle représentation est appelée modèle de connaissance, modèle boîte blanche ou modèle théorique. Il est cependant très difficile, voire parfois impossible, d'établir de tels modèles pour des systèmes complexes.

La démarche la plus courante est l'établissement d'un modèle boîte noire (modélisation expérimentale), basé sur les informations recueillies sur le fonctionnement du système, notamment les mesures faites sur les variables. Le choix des relations qui expliquent le fonctionnement du système est alors guidé par deux objectifs contradictoires :

- la relation doit être suffisamment complexe pour représenter le plus fidèlement possible le système ;
- la relation doit être suffisamment simple de sorte que l'estimation paramétrique ne soit pas très coûteuse en temps de calcul et que la variance des paramètres estimés reste faible.

La difficulté d'une modélisation de type boîte noire est de trouver un compromis entre ces deux objectifs contradictoires.

Plusieurs approches ont été proposées pour la représentation des systèmes dynamiques non linéaires : méthode de linéarisation du système, représentation de Volterra-Wiener [1], représentations de Hammerstein et de Wiener ([2]), etc. Plus récemment, d'autres techniques ont vu le jour notamment celles basées sur les Réseaux de Neurones Artificiels (RNA) [3, 4, 5, 6] et les multimodèles [7, 8, 9, 10].

Inspirés du fonctionnement du vivant, les RNA tentent d'acquérir les propriétés fondamentales que sont : une organisation parallèle, un mode de calcul et une mémoire distribués, une capacité d'apprentissage, de généralisation et d'adaptation. Plusieurs travaux [11, 12, 13, 14] ont établi que le perceptron multi-couches (« *Multi-Layer Perceptron, MLP* »), une implantation des RNA très utilisée dans les problèmes de regression non linéaires, est un approximateur universel . Il permet en fait la représentation de tout système non linéaire avec une précision arbitraire, si sa structure est convenablement choisie. Cependant, il n'existe pas de règles permettant de choisir pour un système donné, la structure du réseau MLP capable de le représenter. Une contribution à l'identification structurelle de réseaux MLP est apportée dans ce travail de thèse à travers l'utilisation du critère d'information d'Akaike (*AIC*) [15] dans le choix du nombre de neurones.

La représentation des systèmes dynamiques non linéaires par multimodèles est une technique relativement récente dont la formalisation mathématique date des travaux de

Johansen et Foss [16]. Elle regroupe en son sein plusieurs concepts de modélisation connus sous d'autres appellations telles que modèles de Takagi-Sugeno, systèmes multi-experts, etc. C'est une technique basée sur la stratégie « de diviser pour régner » qui consiste, dans le cas de la modélisation, à décomposer un système complexe en plusieurs sous-systèmes pour lesquels des modèles locaux simples peuvent être élaborés. Chaque modèle local tente alors de représenter le système dans un domaine de fonctionnement bien défini. Le modèle global du système est obtenu par une combinaison des modèles locaux.

L'implantation conventionnelle des approches neuronale et multimodèle, bien que performante, souffre de certaines difficultés liées à la complexité de l'estimation paramétrique (notamment pour les systèmes bouclés) et à la complexité de l'identification structurelle (pour les systèmes à fortes non linéarités). En effet, l'estimation paramétrique des implantations neuronales des modèles bouclés fait appel à l'algorithme de « Back Propagation Through Time »(voir [17]) ou à celui de « Real-Time Recurrent Learning » (voir [18]). Ces algorithmes basés sur la technique d'optimisation non linéaire par la méthode de descente du gradient présentent plusieurs inconvénients dont les principaux sont l'attraction vers des minima locaux et la lenteur de convergence. En ce qui concerne l'approche multimodèle, une des difficultés majeures est la décomposition de l'espace de fonctionnement du système à modéliser en différents sous-espaces pour lesquels des modèles locaux sont élaborés. Récemment plusieurs auteurs ([8, 19, 9, 20, 21]) ont utilisés des méthodes de décomposition basées sur la classification floue, permettant ainsi de regrouper des données présentant une certaine « similarité ». Un modèle local (affine) est ensuite élaboré pour chaque groupe de données. Tous les modèles locaux ont ainsi la même structure (affine). Or, la démarche devrait logiquement conduire à une architecture de multimodèle où la structure de chaque modèle local est déterminée par le groupe de données qui définit son domaine de validité. Il se pose alors un problème d'identification structurelle des modèles locaux.

Les problèmes ainsi soulevés ont mené à ce travail de thèse qui vise les objectifs suivants.

Simplifier l'apprentissage des modèles bouclés : l'implantation d'architectures multimodèles récurrentes avec un algorithme d'apprentissage adapté constitue un des apports de ce travail de thèse, publié dans une revue internationale (voir [8]). L'architecture proposée permet l'identification des modèles non linéaires tels que les modèles NARMAX ou NOE.

Implanter une structure simple de modèle local non linéaire : une architecture de

multimodèle avec des modèles locaux polynômiaux (de degré supérieur à 1), plus aptes à appréhender les non linéarités locales que des modèles locaux affines classiquement utilisés, est proposée et constitue un autre apport de ce travail qui a fait l'objet d'une communication [22]. Cette structure permet de réduire le nombre de modèles locaux dans le cas d'un système présentant de fortes non linéarités. Elle permet également l'utilisation de techniques d'optimisation linéaire pour l'estimation paramétrique.

Implanter une architecture multimodèle hétérogène : une architecture multimodèle hétérogène, constituée de modèles locaux de structures différentes (polynômiale de degrés différents, polynômiale et/ou neuronale) est proposée, permettant d'adapter la structure de chaque modèle local au groupe de données qui détermine son domaine de validité. Cette architecture est similaire à celle d'un réseau multi-experts mais utilise une méthode d'apprentissage plus simple.

L'ensemble des méthodologies proposées dans ce travail a abouti à la mise en œuvre d'un outil logiciel d'identification de systèmes non linéaires par multimodèles et réseaux de neurones MLP. A partir d'une base de données du système contenant les informations entrées-sorties recueillies, le logiciel permet l'identification structurelle (détermination de l'architecture multimodèle ou neuronale choisie) et l'estimation paramétrique du modèle du système.

Ce rapport de thèse est composé de quatre chapitres répartis comme suit.

Chapitre 1 - Ce chapitre présente la problématique de la représentation des systèmes dynamiques non linéaires. Différents types de modèles y sont présentés, les méthodes d'estimation paramétrique et les méthodes de sélection de modèles y sont également abordées. Ces méthodes sont utilisées pour l'identification structurelle et paramétrique des architectures MLP et multimodèle présentées aux chapitres 2 et 3.

Chapitre 2 - Ce chapitre traite de l'identification de systèmes dynamiques non linéaires par réseaux de neurones de type MLP. La méthodologie d'identification structurelle des réseaux MLP proposée dans ce travail est présentée et appliquée à des architectures MLP non récurrentes et récurrentes. L'influence du bruit (bruit de sortie, bruit d'état ou bruit de sortie et bruit d'état) sur le choix de la structure de modèle non linéaire est également abordée.

Chapitre 3 - Dans ce chapitre, l'identification de systèmes dynamiques non linéaires par multimodèles est présentée. Le principe de la multi-modélisation y est décrit et différentes méthodes de décomposition de l'espace de fonctionnement de systèmes y

sont présentées et comparées. Les principaux apports de ce travail de thèse, notamment la description des multimodèles avec des modèles locaux polynômiaux [22], l'algorithme d'apprentissage des multimodèles récurrents [8, 23] et les multimodèles hétérogènes sont également présentés. Les différentes méthodologies proposées sont appliquées à l'identification de systèmes dynamiques non linéaires simulés.

Chapitre 4 - Ce chapitre est consacré à une étude comparative entre les architectures MLP et multimodèle sur la base d'un exemple académique et de deux systèmes réels :

- prédiction de la concentration de CO_2 dans le processus de Box-Jenkins [24] : la sortie, (concentration de CO_2 résultant de la combustion d'un mélange de méthane et d'air) doit être expliquée en fonction de la variation du débit de méthane injecté dans le four ;
- prédiction de débit dans le fleuve Sénégal à partir des mesures de débits effectuées à la station de Bakel (sur le fleuve Sénégal) et aux stations amont situées à Fadougou (sur la rivière Falémé) et à Oualia (sur la rivière Bakoye).

La conclusion générale et les perspectives viennent clore la thèse.

Chapitre 1

Représentation des Systèmes Dynamiques Non Linéaires

Sommaire

1.1 Introduction

Dans plusieurs domaines de la science, il est souvent nécessaire de disposer des représentations de systèmes afin de maîtriser ou d'optimiser leur fonctionnement, voire de mieux comprendre leur évolution. Ces systèmes peuvent être artificiels (systèmes industriels), naturels (système écologique, biologique, etc.), sociaux ou financiers. La modélisation est une représentation du système dans le but de satisfaire un besoin préalablement défini. La représentation qu'on va faire du système, donc le modèle à établir, dépend principalement des objectifs visés. Dans les sciences pour l'ingénieur, on utilise principalement le modèle physique ou le modèle mathématique. Le travail qui fait l'objet de cette thèse est basé sur la modélisation mathématique. L'étude se limite aux systèmes dynamiques non linéaires à temps discret de type *MISO*[1] (multi-entrées, mono-sortie). Pour ces systèmes, la sortie est une fonction non linéaire des variables caractéristiques.

Après une description sommaire du principe de la modélisation mathématique, nous abordons ici différentes structures de modèles de systèmes dynamiques non linéaires, les types les plus courants de modèles non linéaires, les critères d'estimation paramétrique et de sélection de modèles ainsi que les méthodes d'estimation paramétrique des modèles non linéaires.

1.2 Principe de la modélisation mathématique

La modélisation mathématique est une représentation qui traduit le fonctionnement d'un système à travers des relations mathématiques liant les différentes variables du système. Elle peut se faire de deux façons différentes :

Modélisation théorique : la représentation du système est faite à partir des lois (physiques, chimiques, biologiques, etc.) régissant le fonctionnement du système. Il est donc nécessaire d'avoir une connaissance complète du système. Cette modélisation peut présenter des difficultés lorsqu'elle est appliquée à des systèmes complexes. Les modèles de ce type sont appelés modèles de connaissance ou modèles de type « boîte blanche ».

Modélisation expérimentale : la représentation est faite sur la base de données re-

[1] *Multiple Inputs, Single Output*. L'étude présentée dans ce travail peut également s'appliquer à des systèmes multi-entrées, multi-sorties (*Multiple Inputs, Multiple Outputs - MIMO*).

cueillies sur le système à modéliser. Cette représentation ne requiert aucune connaissance du système. Les modèles de ce type sont appelés modèles expérimentaux ou de type « boîte noire ». Ils sont représentés en général sous la forme d'une relation de type « entrée-sortie ».

Dans certains cas, des connaissances *a priori* sur le système permettent de fixer la structure du modèle. La combinaison de ces connaissances *a priori* et des données expérimentales recueillies permet d'aboutir à une représentation du système communément appelée modèle de type « boîte grise ».

Dans notre travail, nous utilisons la modélisation expérimentale. L'établissement de ce type de modèle est une procédure itérative comportant cinq phases [25] (voir figure 1.1) :

Extraction de données : durant cette phase, des mesures sont effectuées sur les variables sensées caractériser le système. Ces variables peuvent être des variables externes qui agissent sur le système (entrées de commande ou perturbations mesurables), des variables internes qui traduisent l'état du système (variables d'état), ou la réponse du système (variable de sortie). Il existe souvent des perturbations non mesurables qui agissent sur le système (en entrée ou en sortie) rendant plus difficile sa modélisation.

Choix de la structure du modèle : il s'agit de définir d'une façon formelle la relation expliquant le fonctionnement du système. Cette relation correspond à une famille de fonctions mathématiques dont une seule correspond au modèle recherché.

Choix du critère d'estimation paramétrique : c'est le choix de la fonction objectif (fonction coût) dont l'optimisation (minimisation) permet de déterminer la structure du modèle de façon unique. Ce critère est fonction de l'écart entre la sortie du système et celle du modèle. Le critère quadratique est généralement choisi.

Estimation paramétrique : de la famille de fonctions obtenues du choix de la structure du modèle, il faut en déterminer une qui représente convenablement le système. Il s'agit alors de trouver la valeur des paramètres permettant la satisfaction d'un critère de performance donné (optimisation de la fonction objectif).

Validation du modèle : c'est une procédure qui permet d'évaluer l'exactitude (ou la fidélité) du modèle. Pendant cette phase, le modèle est testé avec des données non utilisées pendant la phase d'identification.

Il existe deux grandes catégories de modèles :

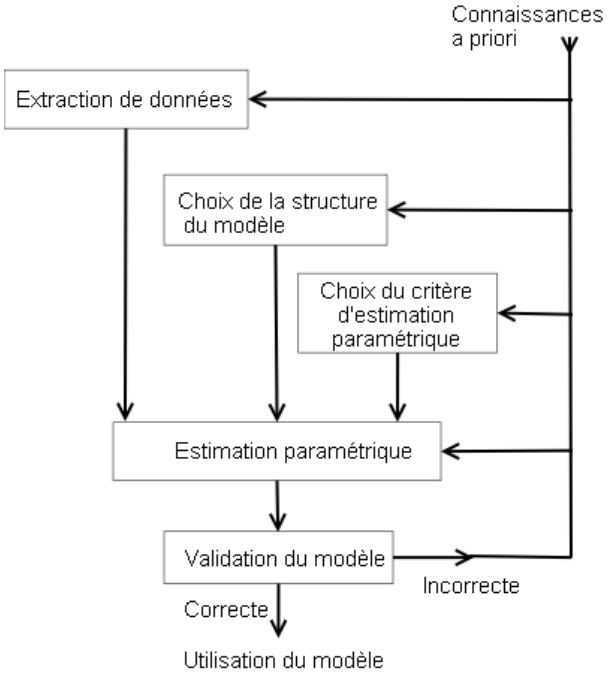

FIG. 1.1 – Procédure d'identification d'un système.

– Les modèles de prédiction qui fonctionnent en parallèle avec le système et pour lesquels la sortie du système à l'instant $t + h$ est estimée à partir des entrées et des sorties disponibles jusqu'à l'instant t, h étant le pas de prédiction. Si $h = 1$ le modèle correspond à un prédicteur à *1 pas*. Ces modèles sont très utilisés dans l'industrie, la prédiction météorologique, la prédiction du taux de pollution, la prédiction des cours en bourse, etc.
– Les modèles de simulation qui fonctionnent indépendamment du système et pour lesquels la sortie inconnue est estimée en se basant sur les entrées externes du système. Ces modèles permettent entre autre : l'extrapolation du fonctionnement du système dans des conditions dont la réalisation expérimentale est difficile, dangereuse voire coûteuse, la synthèse de correcteur pour un système de commande, etc.

Les modèles que nous étudions dans ce travail font partie de l'une ou l'autre de ces deux catégories.

1.3 Structures de modèles non linéaires

La construction d'un modèle « boîte noire » d'un système repose sur l'hypothèse selon laquelle il existe une relation déterministe liant les entrées du système à sa sortie. D'une manière générale, le modèle prédictif[2] de comportement d'un système dynamique non linéaire peut s'écrire sous la forme :

$$y_s(t+h) = F_s\Big(\underline{u}(t), \underline{\tilde{y}}_s(t)\Big) + e(t+h) \tag{1.1}$$

où :

$y_s(t+h)$ est la sortie du système à l'instant $t+h$, h étant le pas de prédiction ;
$F_s(\cdot)$ est une fonction non linéaire déterministe inconnue, appelée prédicteur théorique ;
$\underline{u}(t)$ est un vecteur dont les composantes sont des éléments des entrées externes du système à l'instant courant t et/ou aux instants antérieurs :

$$\begin{aligned}\underline{u}(t) \;=\; \Big[&u_1(t), u_1(t-1), \ldots, u_1(t-n_{u_1}+1), \ldots,\\ &u_k(t), u_k(t-1), \ldots, u_k(t-n_{u_k}+1), \ldots,\\ &u_{n_i}(t), u_{n_i}(t-1), \ldots, u_{n_i}(t-n_{u_{n_i}}+1)\Big]^T\end{aligned}$$

avec n_{u_k} l'ordre de l'entrée u_k, $k = 1, \ldots, n_i$, n_i étant le nombre d'entrées ;
$\underline{\tilde{y}}_s(t)$ est un vecteur dont les composantes sont liées à l'état du système à l'instant courant t et/ou aux instants antérieurs ; on peut avoir par exemple :

$$\underline{\tilde{y}}_s(t) = \Big[y_s(t), y_s(t-1), \ldots, y_s(t-n_{y_s}+1)\Big]^T$$

avec n_{y_s} l'ordre de la sortie y_s ;
$e(t+h)$ est une variable aléatoire de moyenne nulle et de variance σ^2 représentant le bruit.

La représentation formelle donnée par la relation (1.1) intègre les connaissances *a priori* du système ainsi que des hypothèses concernant son comportement : connaissance des variables descriptives, caractère statique ou dynamique, linéaire ou non linéaire, présence ou absence de perturbations.

Le modèle est statique si la sortie à l'instant t ne dépend que des entrées à l'instant t, il est dynamique si la sortie dépend aussi des entrées et/ou sorties antérieures. Le modèle

[2] L'appellation modèle prédictif vient surtout du fait que l'identification est basée sur la minimisation de l'erreur de prédiction. Le modèle peut être utilisé en tant que modèle de simulation.

est linéaire si $y_s(t+h)$ est une combinaison linéaire de $\underline{u}(t)$ et de $\underline{\tilde{y}}_s(t)$. Dans le cas contraire, il est non linéaire. Le modèle est récurrent si $\underline{\tilde{y}}_s(t)$ comporte des variables d'état estimés à l'instant courant t et/ou aux instants antérieurs. A l'inverse, si la sortie du modèle ne dépend que des entrées et/ou des sorties mesurées, le modèle est non récurrent.

L'estimation paramétrique des modèles non linéaires récurrents est un problème majeur en identification. L'un des principaux apports de ce travail de thèse est la mise en place d'une méthodologie d'estimation paramétrique de modèles non linéaires récurrents présentée dans le chapitre 3.

En considérant la sortie du système comme une variable aléatoire $Y_s(t)$ dont une réalisation est $y_s(t)$, on peut écrire [26] :

$$Y_s(t+h) = E\Big[Y_s(t+h)|t\Big] + e(t+h) \tag{1.2}$$

où $E\Big[Y_s(t+h)|t\Big]$ est l'espérance mathématique conditionnelle de $Y_s(t+h)$, lorsqu'on dispose de toutes les informations disponibles jusqu'à l'instant courant t, et $e(t+h)$ correspond à la partie non prédictible de $Y_s(t+h)$ à l'instant t (erreur de modélisation). La sortie y du système, donnée par le modèle $F_s(\cdot)$, est :

$$y(t+h) = E\Big[Y_s(t+h)|t\Big] = F_s\Big(\underline{u}(t), \underline{\tilde{y}}_s(t)\Big) \tag{1.3}$$

La fonction $F_s(\cdot)$ est approchée par la fonction $F(\cdot)$ de structure connue paramétrée par un vecteur $\underline{\theta}$. La sortie $\hat{y}$ estimée par ce prédicteur réel est :

$$\hat{y}(t+h) = F\Big(\underline{u}(t), \underline{\tilde{y}}_s(t), \underline{\theta}\Big) = F\Big(\underline{\varphi}(t), \underline{\theta}\Big) \tag{1.4}$$

où $\underline{\varphi}(t) = [\underline{u}(t)^T, \underline{\tilde{y}}_s(t)^T]^T$ est le vecteur de régression ou d'information constitué de l'ensemble des variables explicatives du système dans la zone de validité du modèle. Il est obtenu par la concaténation des éléments des vecteurs $\underline{u}(t)$ et $\underline{\tilde{y}}_s(t)$. Chaque élément de $\underline{\varphi}(t)$ est appelé régresseur.

L'erreur de prédiction du modèle est obtenue par :

$$\varepsilon(t) = y_s(t) - \hat{y}(t) \tag{1.5}$$

La détermination des régresseurs est un problème majeur rencontré en modélisation expérimentale. L'idéal est de ne sélectionner que les variables caractéristiques des non-linéarités du système. Ces variables étant inconnues, plusieurs démarches sont proposées pour la sélection des entrées [27, 11]. L'une d'elle, comme expliquée dans [11], consiste à

choisir un ensemble de variables d'entrée le plus grand possible permettant d'obtenir le « modèle complet ». Les performances de ce modèle sont comparées à celles des modèles dont les variables d'entrée constituent des sous-ensembles des variables du « modèle complet ». Le modèle qui présente les meilleures performances est choisi. Avec cette démarche le nombre de modèle croît de façon exponentielle avec le nombre de variables, ce qui complique la mise en œuvre. Des stratégies sous-optimales (décrites dans [11]) mais plus simples à mettre en œuvre sont souvent utilisées en pratique. Il s'agit de la stratégie d'élimination (*« stepwise backward regression »*) et de la stratégie de construction (*« stepwise forward regression »*).

Dans ce travail, nous ne traitons pas le problème de la sélection des entrées et considérons que les variables permettant de décrire les systèmes à étudier sont connues.

La description du système donnée par l'équation (1.4) conduit à deux questions fondamentales : l'identification structurelle de la fonction $F(\cdot)$ et l'estimation paramétrique. L'identification structurelle consiste à choisir une structure de $F(\cdot)$ adéquate pour la description du système. L'estimation paramétrique de $F(\cdot)$ consiste à estimer la valeur $\hat{\underline{\theta}}$ de $\underline{\theta}$ qui minimise une norme de l'écart entre la sortie réelle du système y_s et celle du modèle $\hat{y}$ pour l'ensemble des observations :

$$\hat{\underline{\theta}} = \arg \min_{\underline{\theta}} \left(\left\| y_s(t+h) - F(\underline{\varphi}(t), \underline{\theta}) \right\| \right) \tag{1.6}$$

La valeur $\hat{\underline{\theta}}$ détermine la structure choisie pour $F(\cdot)$ de façon unique.

La variété des modèles de représentation des systèmes dynamiques non linéaires repose sur le choix de la structure de la fonction $F(\cdot)$.

1.3.1 Modèle de Volterra-Wiener

Le modèle de Volterra-Wiener [1], l'un des plus anciens, est basé sur une décomposition en série, soit pour un système mono-entrée :

$$y(t) = \sum_{i=1}^{\infty} y_i(t) \tag{1.7}$$

$$y_i(t) = \sum_{\tau_1=0}^{t-1} \cdots \sum_{\tau_i=0}^{t-1} h_i(\tau_1, \ldots, \tau_i) u(t-\tau_1) u(t-\tau_i) \tag{1.8}$$

où $h_i(\tau_1, \ldots, \tau_i)$ est la réponse impulsionnelle d'ordre i du système.

Cette représentation n'a pas connu beaucoup de succès en identification, vu le nombre très élevé de paramètres qu'elle engendre.

1.3.2 Modèle de Hammerstein et de Wiener

Les modèles de Hammerstein et de Wiener (voire figues 1.2 et 1.3) sont constitués de blocs distincts de modèles dynamiques linéaires et de modèles statiques non linéaires, interconnectés en série et/ou en parallèle [2].

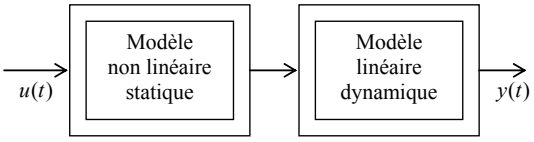

FIG. 1.2 – Modèle de Hammerstein.

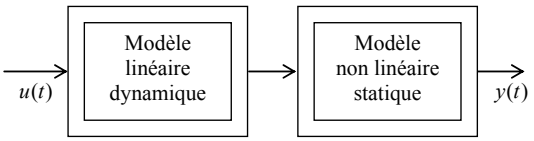

FIG. 1.3 – Modèle de Wiener.

1.3.3 Modèles flous

Les modèles flous ont connu un développement important ces dernières années ([9, 28, 29]). Ils permettent une représentation des systèmes non linéaires en intégrant les imprécisions du raisonnement humain. Un modèle flou est constitué d'un ensemble de règles comportant des variables linguistiques. Les règles sont de la forme :

si *prémisse* ***alors*** *conséquence*

Par exemple :

si *la vitesse est faible* ***alors*** *appuyer sur l'accélérateur*

La variable *vitesse* est la variable de prémisse. Afin d'évaluer son degré de véracité, on utilise des ensembles flous correspondant à des qualificatifs de la variable. Pour l'exemple

considéré, les qualificatifs de la variable de prémisse vitesse peuvent être : *faible*, *moyenne* ou *élevée*. Le degré d'appartenance d'une variable de prémisse à un ensemble flou est donné par une fonction d'appartenance. Pour l'exemple considérée, ce serait une fonction indiquant le degré d'appartenance de la variable de prémisse *vitesse* à l'ensemble flou *faible* (ou aux autres ensembles flous). Les fonctions d'appartenance peuvent être de différentes formes : triangulaire, trapèzoïdale, gaussienne, sigmoïde, etc.

Un modèle flou est constitué principalement de quatres parties :

- la fuzzification : transforme les valeurs numériques de l'entrée en un ensemble flou ;
- la base de connaissance : comprend une base de règles permettant de décrire le système et une base de données comportant les paramètres des fonctions d'appartenance et ceux des parties conséquences des règles ;
- le moteur d'inférence : permet de déterminer, pour une entrée donnée, l'ensemble flou de sortie à l'aide de la base de connaissance ;
- la défuzzification : transforme l'ensemble flou de sortie en une valeur numérique.

Il existe principalement trois types de modèles flous [21, 30] : modèles flous linguistiques proposés par Zadeh en 1973 et Mamdani en 1977, le modèle à relation flou de Pedrycz (1941) et le modèle de Takagi-Sugeno (voir [31]).

Dans le modèle de Takagi-Sugeno, qui a connu beaucoup de succès notamment dans l'industrie, la partie conséquence de chaque règle (r_i) est une fonction affine des variables de prémisse :

$$r_i : \ \textbf{\textit{si}}\ (x_{i1}\ est\ A_{i1},\ x_{i2}\ est\ A_{i2},\ \ldots,\ x_{in}\ est\ A_{in})\ \textbf{\textit{alors}}\ y_i = \underline{\varphi}^T\ \theta_{i1} + \theta_{i0}$$

où :

A_{ij} est l'ensemble flou correspondant à la variable de prémisse x_{ij}, $j = 1, \ldots, n$, n étant le nombre de variables ;

$\underline{\varphi} = [x_{i1}, x_{i2}, \cdots, x_{in}]^T$ vecteur composé des variables de prémisse ;

θ_{i1} vecteur de paramètres et θ_{i0} constante scalaire.

La sortie globale du modèle est obtenue par l'expression :

$$y = \frac{\sum_{i=1}^{M} v_i(\underline{\varphi}) y_i}{\sum_{j=1}^{M} v_j(\underline{\varphi})} \tag{1.9}$$

où v_i est le degré de véracité de la règle r_i.
L'expression (1.9) peut se mettre sous la forme :

$$y = \sum_{i=1}^{M} \omega_i y_i \tag{1.10}$$

$\omega_i = \dfrac{v_i(\underline{\varphi})}{\sum_{j=1}^{M} v_j(\underline{\varphi})}$ étant le degré de véracité normalisé de la règle r_i.

Nous verrons dans le chapitre 3 que le modèle de Takagi-Sugeno appartient à la classe des multimodèles.

1.3.4 Modèles multi-experts

Un modèle multi-experts (ou mélange d'expert - « *mixture of experts* » en anglais) est composé de plusieurs modèles ou réseaux d'experts et d'un réseau de déclenchement. Ce dernier permet d'évaluer la sortie du modèle par une combinaison linéaire des sorties des réseaux experts. La sortie de chaque réseau expert est pondérée par la probabilité que la sortie du modèle soit égale à celle de l'expert. Ces modèles appartiennent à la classe des multimodèles et sont présenté au chapitre 3.

1.3.5 Modèle à base de réseaux de neurones

Initialement étudiés en vue de modéliser le comportement du cerveau humain, les modèles à base de réseaux de neurones sont aujourd'hui des outils de calculs mathématiques sophistiqués utilisés dans des domaines très divers. Un modèle neuronal est constitué de plusieurs unités de calcul élémentaires (les neurones artificiels), fonctionnant en parallèle. Chaque neurone reçoit des informations (qui peuvent être les entrées du modèle ou les sorties d'autres neurones), les traite, et envoie le résultat du traitement vers d'autres neurones. Le ou les neurones de sorties permet(tent) de reproduire le comportement du système à modéliser. Ces types de réseaux sont capables de représenter des systèmes très complexes. Il existe différents types de réseaux de neurones artificiels dont les Perceptrons Multi-Couches, utilisés dans ce travail de thèse, et les Réseaux de Fonctions à Base Radiale [32] qui présentent une certaine similitude avec les multimodèles.

1.3.5.1 Perceptrons Multi-Couches

Les réseaux de neurones de type Perceptrons Multi-Couches (*Multi Layer Perceptron - MLP*) sont des réseaux à propagation avant, composés d'une ou plusieurs couches cachées et d'une couche de sortie. Chaque couche du réseau est composée de neurones artificiels. La première couche cachée reçoit l'information provenant des entrées. L'information est traitée et transmise vers les couches suivantes jusqu'à la dernière. Les MLP sont connus comme étant des approximateurs universels et sont très utilisés dans des problèmes de régression non linéaire. Ils font l'objets d'une étude détaillée dans le chapitre 2.

1.3.5.2 Réseaux de fonctions à base radiale

Les Réseaux de Fonctions à Base Radiale (*Radial Basis Function - RBF*) ont une architecture semblable à celle des MLP, mais ne comporte généralement que deux couches : une couche cachée et une couche de sortie. L'architecture d'un réseau RBF est représentée sur la figure 1.4.

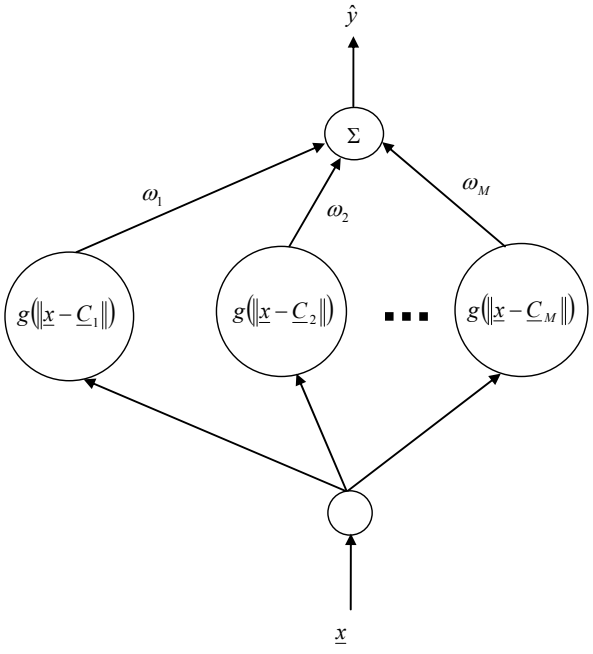

FIG. 1.4 – Réseau de neurones RBF.

Un RBF permet de réaliser une décomposition sur une base de fonctions radiales [33] :

$$\hat{y} = \sum_{i=1}^{M} \omega_i \; g\Big(\big\|\underline{x} - \underline{C}_i\big\|_{\Sigma_i}\Big) \tag{1.11}$$

où :

$\hat{y}$ est la sortie du réseau ;

g est la fonction d'activation à base radiale ;

ω_i représente le poids de la connexion entre le neurone i ($i = 1, \cdots, M$) et le neurone de sortie ;

$\underline{x}$ est le vecteur d'entrée ;

$\underline{C}_i$ est un vecteur désignant le centre de la fonction d'activation du neurone i ;

Σ_i est une matrice qui détermine l'étendue et l'orientation de la fonction d'activation dans l'espace des variables de régression ;

$\big\|\underline{x} - \underline{C}_i\big\|_{\Sigma_i}$ désigne la distance entre le point courant $\underline{x}$ et le centre $\underline{C}_i$:

$$\big\|\underline{x} - \underline{C}_i\big\|_{\Sigma_i} = \big(\underline{x} - \underline{C}_i\big)^T \Sigma_i^{-1} \big(\underline{x} - \underline{C}_i\big)$$

Il existe plusieurs types de fonction à base radiale, mais la fonction gaussienne est la plus utilisée pour les RBF. Pour un neurone i de la couche cachée du réseau, la fonction d'activation pour une entrée $\underline{x}$ est :

$$g\Big(\big\|\underline{x} - \underline{C}_i\big\|_{\Sigma_i}\Big) = g_i\big(\underline{x}\big) = \exp\Big[\big(\underline{x} - \underline{C}_i\big)^T \Sigma_i^{-1} \big(\underline{x} - \underline{C}_i\big)\Big]$$

Les réseaux RBF sont, comme les MLP, des approximateurs universels [34] et sont utilisés dans des applications telles que la classification, la reconnaissance de forme ou la prédiction (voir par exemple [33, 35, 36, 37]). Ils sont faciles à estimer mais généralisent moins bien que les MLP et leur taille croît vite avec la dimension de l'entrée.

Nous verrons au chapitre 3 que l'architecture du réseau RBF représentée sur la figure 1.4 est un cas particulier d'une architecture multimodèle.

1.3.6 Machines à vecteurs supports

Les machines à vecteurs supports (*Support Vector Machines - SVM*) sont des outils d'apprentissage statistiques introduits par Vapnik [38] en 1995. Les SVM sont généralement utilisées dans des problèmes de classification. L'approche permet de définir des surfaces complexes dans des espaces de dimensions importantes, avec des représentations très

concises. Si les méthodes classiques d'apprentissage se basent sur la minimisation de l'erreur d'apprentissage (risque empirique), le principal avantage des SVM est la possibilité de déterminer une marge d'erreur (le risque structurel) valable pour la validation. Basés sur le principe des SVM, les *Support Vector Regression - SVR* permettent de traiter des problèmes de régression (linéaire ou non linéaire). Récemment plusieurs travaux ont été consacrés à l'utilisation des SVR pour l'approximation de fonctions et pour la prédiction [39, 40, 41]. Cependant un des inconvénients de cette approche est le coût de calcul souvent élevé.

1.3.7 Multimodèles

La modélisation par multimodèles consiste à décomposer un système complexe en plusieurs sous-systèmes pour lesquels on peut élaborer des modèles locaux simples. Chaque modèle local tente alors de représenter le système dans un domaine de fonctionnement bien défini. Le modèle global du système est obtenu par une combinaison des modèles locaux. Les multimodèles apparaissent ainsi comme une généralisation de plusieurs types de modèles tels que les modèles flous, les modèles multi-experts, etc. Les multimodèles sont présentés dans le chapitre 3.

1.4 Modèles dynamiques non linéaires

Les connaissances *a priori* du système à modéliser, prises en compte dans la représentation (1.4), intègrent le choix des variables explicatives et la manière dont le bruit agit sur le système. Le bruit peut agir sur la sortie, sur l'état ou sur la sortie et l'état à la fois. Ces informations sont spécifiées lors de la constitution du vecteur de régression qui permet ainsi de déterminer la classe du modèle non linéaire. On distingue différentes classes de modèles non linéaires. Le choix d'un de ces modèles est indépendant du choix de la structure de la fonction $F(\cdot)$. Ainsi, un même modèle non linéaire peut par exemple être représenté avec une structure neuronale ou multimodèle. Le choix de la structure doit permettre d'obtenir un modèle parcimonieux (faible erreur avec un minimum de paramètres). Tout au long de ce travail, les performances des structures MLP et multimodèles sont étudiées à travers l'implantation de quelques modèles non linéaires qui sont présentons ici.

1.4.1 Modèle NFIR

Le modèle NFIR (*Nonlinear Finite Impulse Response*) permet de représenter des systèmes non linéaires dont la sortie ne dépend que des entrées à l'instant courant et/ou aux instants antérieurs.

$$\begin{aligned} y_s(t+1) &= F_s\Big(u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ &\quad u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ &\quad u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2)\Big) \\ &\quad + e(t+1) \end{aligned} \tag{1.12}$$

où :

$d_{u_k} \geq 1$ est le retard correspondant à l'entrée u_k ;

$n_{u_k} \geq 1$ est la mémoire sur l'entrée u_k, avec $k = 1, \ldots, n_i$, n_i nombre d'entrées.

Le pas de prédiction h défini dans (1.1) peut être déterminé par :

$$h = \min(d_{u_k}), \quad k = 1, \ldots, n_i$$

Le prédicteur réel du modèle NFIR est la fonction non linéaire $F(\cdot)$ de structure connue, paramétrée par un vecteur $\underline{\theta}$ qui, convenablement choisi, permettrait d'obtenir une erreur de prédiction ayant les mêmes caractéristiques statistiques que le bruit e :

$$\begin{aligned} \hat{y}(t+1) &= F\Big(u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ &\quad u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ &\quad u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2), \underline{\theta}\Big) \end{aligned} \tag{1.13}$$

Puisque le vecteur de régression du modèle NFIR ne dépend que des entrées externes, le modèle est non récurrent.

1.4.2 Modèle NARX

Le modèle NARX (*Nonlinear AutoRegressive with eXogenous input*) permet de représenter des systèmes dynamiques non linéaires dont la sortie dépend des entrées passées et des sorties mesurées passées. Il permet également la représentation de systèmes dynamiques non linéaires avec « bruit d'état » :

$$\begin{cases} x(t+1) &= F_s\Big(u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & \quad u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & \quad u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2), \\ & \quad x(t-d_x+1), x(t-d_x), \ldots, x(t-d_x-n_x+2)\Big) + e(t+1) \\ y_s(t) &= x(t) \end{cases} \tag{1.14}$$

où :

x désigne l'état du système et y_s sa sortie ;

$d_x \geq 1$ et $n_x \geq 1$ représentent le retard et l'ordre de la variable x respectivement.

Le prédicteur réel $F(\cdot)$ du model NARX est :

$$\begin{aligned} \hat{y}(t+1) &= F\Big(u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & \quad u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & \quad u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2), \\ & \quad y_s(t-d_{y_s}+1), y_s(t-d_{y_s}), \ldots, y_s(t-d_{y_s}-n_{y_s}+2), \underline{\theta}\Big) \end{aligned} \tag{1.15}$$

où :

$d_{y_s} \geq 1$ est le retard correspondant à la sortie mesurée y_s ;

$n_{y_s} \geq 1$ est l'ordre de la partie autorégressive de y_s.

Le pas de prédiction h défini dans (1.1) peut être déterminé par :

$$h = \min(d_{u_k}, d_{y_s})$$

Le modèle NARX (appelé aussi modèle à Erreur d'Equation ou « *Equation Error* » en anglais) est un modèle non récurrent, tous les régresseurs étant des mesures faites sur le système (le vecteur de régression est constitué des entrées passées u_k et des sorties mesurées passées de y_s).

1.4.3 Modèle NOE

Le modèle NOE (*Nonlinear Output Error*) permet de représenter des systèmes dynamiques non linéaires par estimation de la sortie à un instant t à partir des entrées antérieures et des sorties antérieures estimées. Il permet également la représentation de

systèmes dynamiques non linéaires avec « bruit de sortie » :

$$\left\{\begin{array}{rcl} x(t+1) & = & F_s\Big(u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & & u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & & u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2), \\ & & x(t-d_x+1), x(t-d_x), \ldots, x(t-d_x-n_x+2)\Big) \\ y_s(t) & = & x(t)+e(t) \end{array}\right. \tag{1.16}$$

Le prédicteur réel $F(\cdot)$ du model NOE est :

$$\begin{array}{rcl} \hat{y}(t+1) & = & F\Big(u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & & u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & & u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2), \\ & & \hat{y}(t-d_{\hat{y}}+1), \hat{y}(t-d_{\hat{y}}), \ldots, \hat{y}(t-d_{\hat{y}}-n_{\hat{y}}+2), \underline{\theta}\Big) \end{array} \tag{1.17}$$

où $d_{\hat{y}}$ et $n_{\hat{y}}$ sont le retard et l'ordre associé à $\hat{y}$.

Le pas de prédiction est :

$$h = \min(d_{u_k}, d_{\hat{y}})$$

Le vecteur de régression est constitué des entrées passées u_k et des sorties passées estimées $\hat{y}$, ce qui fait du modèle NOE un modèle récurrent. C'est un modèle qui convient bien à la simulation puisqu'il n'utilise pas les sorties réelles du système.

1.4.4 Modèle NARMAX

NARMAX désigne *Nonlinear AutoRegressive Moving Average with eXogenous input*. Ce modèle permet de représenter un système dynamique non linéaire par estimation de la sortie à un instant t à partir des entrées passées, des sorties mesurées passées et des erreurs de prédiction passées. Le modèle NARMAX est aussi utilisé pour représenter un

système dynamique non linéaire en présence de « bruit d'état et bruit de sortie » :

$$\begin{aligned} y_s(t+1) \;=\; & F_s\Big(u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2), \\ & y_s(t-d_{y_s}+1), y_s(t-d_{y_s}), \ldots, y_s(t-d_{y_s}-n_{y_s}+2), \\ & e(t-d_e+1), e(t-d_e), \ldots, e(t-d_e-n_e+2)\Big) \\ & + e(t+1) \end{aligned} \tag{1.18}$$

où $d_e \geq 1$ et $n_e \geq 1$ sont le retard et la mémoire correspondant à la variable aléatoire e.

Les valeurs $e(t-d_e+1),\ e(t-d_e), \ldots,\ e(t-d_e-n_e+2)$ n'étant pas mesurables, ce prédicteur est irréalisable. Cependant en supposant que ces valeurs peuvent être estimées par celles de l'erreur de prédiction $\varepsilon(t) = y_s(t) - \hat{y}(t)$, on peut obtenir le prédicteur réel du modèle NARMAX :

$$\begin{aligned} \hat{y}(t+1) \;=\; & F\Big(u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & u_{n_k}(t-d_{u_{n_k}}+1), u_{n_k}(t-d_{u_{n_k}}), \ldots, u_{n_k}(t-d_{u_{n_k}}-n_{u_{n_k}}+2), \\ & y_s(t-d_{y_s}+1), y_s(t-d_{y_s}), \ldots, y_s(t-d_{y_s}-n_{y_s}+2), \\ & \varepsilon(t-d_\varepsilon+1), \varepsilon(t-d_\varepsilon), \ldots, \varepsilon(t-d_\varepsilon-n_\varepsilon+2), \underline{\theta}\Big) \end{aligned} \tag{1.19}$$

où $d_\varepsilon \geq 1$ et $n_\varepsilon \geq 1$ représentent le retard et la mémoire correspondant à l'erreur de prédiction ε.

Le pas de prédiction est :

$$h = \min(d_{u_k}, d_{y_s}, d_\varepsilon)$$

Le vecteur de régression du modèle NARMAX est constitué des entrées passées u_k, des sorties mesurées passées de y_s et des erreurs de prédiction ε. C'est donc un modèle récurrent.

Le modèle NARMAX est un excellent outil pour l'analyse, la modélisation et la prédiction de séries temporelles [42, 43, 44, 45].

1.5 Choix des critères d'estimation paramétrique et de sélection de modèle

La fonction non linéaire $F(\cdot)$ définie par l'équation (1.4) peut être choisie parmi les structures présentées au paragraphe §1.3 : $F(\cdot)$ peut être un modèle flou, un modèle neuronal, un multimodèle, etc. Quelle que soit la structure adoptée pour la fonction $F(\cdot)$, on aboutit toujours à une famille de fonctions paramétrées par $\underline{\theta}$. Une procédure d'identification paramétrique doit permettre d'estimer $\underline{\theta}$ de façon à ce qu'un critère de performance J (ou fonction de coût) du modèle soit optimal. Cependant, dans la mesure où plusieurs possibilités existent pour représenter un seul et même système, il est nécessaire de disposer d'un outil de comparaison, c'est-à-dire, d'un critère de sélection de modèles. Cet outil, basé sur des critères de performances à atteindre, permet de sélectionner le meilleur modèle.

Par ailleurs, le choix de la structure de modèle s'accompagne de la détermination de sa topologie interne (par exemple, nombre de règles pour un modèle flou, nombre de neurones cachés pour un modèle neuronal ou nombre de modèles locaux pour un multimodèle). L'utilisation d'un critère de sélection de modèles permet aussi de déterminer la meilleure topologie pour une structure de modèle donnée.

1.5.1 Choix du critère d'estimation paramétrique

L'estimation des paramètres d'un modèle se fait en optimisant un critère de performance dans le but d'approcher la sortie du système par celle du modèle. Ce critère de performance est une fonction de l'écart (ou résidu) ε entre la sortie réelle du système et celle du modèle :

$$\varepsilon(t) = y_s(t) - \hat{y}(t) \tag{1.20}$$

Le critère quadratique est le plus utilisé. Il s'exprime par :

$$J = \frac{1}{2}\sum_{t=1}^{k} \varepsilon(t)^2 \tag{1.21}$$

où k désigne le nombre d'observations considérées.

Dans certains cas, si la variance du bruit varie fortement pour certaines observations,

il est nécessaire d'utiliser le critère quadratique pondéré défini par :

$$J = \frac{1}{2}\sum_{t=1}^{k} \omega(t)\,\varepsilon(t)^2 \tag{1.22}$$

où $\omega(t)$ est la fonction de pondération affectée au résidu $\varepsilon(t)$, de façon à réduire l'influence des points situés dans les zones de forte variance.

1.5.2 Choix du critère de sélection de modèle

La structure du modèle détermine sa complexité en terme de nombre de paramètres nécessaires pour décrire le système et de procédure d'estimation de ces paramètres. L'objectif est d'obtenir un modèle qui soit, d'une part, aussi simple que possible comportant un minimum de paramètres, et d'autre part, aussi précis que possible en disposant d'une bonne capacité de généralisation.

L'estimation de la capacité de généralisation d'un modèle nécessite la disposition d'une base de données de test n'ayant pas été utilisée dans la phase d'apprentissage. Cela réduit la base d'apprentissage et peut donc conduire à des pertes d'informations. Des méthodes de sélection de modèles basées sur la parcimonie (moins de paramètres et plus de précision) ont été élaborées ([46, 47, 48]).

La méthode de validation croisée consiste à diviser la base d'apprentissage en k sous-ensembles de tailles identiques et à procéder à k étapes d'identification-validation. A chaque étape, $k-1$ sous-ensembles sont utilisés pour l'identification et le dernier sous-ensemble pour la validation. L'erreur quadratique moyenne de généralisation (Mean Squared Generalization Error - *MSGE*) est ensuite évaluée par :

$$MSGE = \frac{1}{k}\sum_{j=1}^{k}\sum_{t=1}^{N_j}\Big(y_s(t) - y_j(t)\Big)^2 \tag{1.23}$$

où $y_j(\cdot)$ est le modèle identifié à l'itération j et N_j la taille du sous-ensemble de validation à l'itération j.

La structure de modèle permettant d'avoir la plus petite *MSGE* est sélectionnée. Parmi les différents modèles de cette structure, le modèle $y_j(\cdot)$ ayant la plus petite erreur sur les données de test est choisi.

D'autres critères de sélection de modèles ayant la particularité de ne pas réduire la base d'apprentissage ont été élaborés [15, 49, 50]. Ces critères font intervenir deux

termes dont l'un permet de mesurer la qualité de l'estimation paramétrique (valeur du critère d'estimation) et l'autre de pénaliser la complexité du modèle estimé (nombre de paramètres utilisés). En effet si le modèle comporte trop de paramètres (sur-ajustement), il aura tendance à modéliser le bruit. Si par contre le modèle comporte peu de paramètres (sous-ajustement), il ne sera pas à mesure d'expliquer le système. Parmi ces critères de sélection, on peut noter :

Critère d'information d'Akaike (*Akaike Information Criterion - AIC*) : Ce critère détermine la complexité du modèle en minimisant la fonction d'information théorique définie par :

$$AIC = N \ln J + 2n_\theta \tag{1.24}$$

où N est le nombre de données d'observation sur le système et n_θ est le nombre de paramètres dans le modèle. Le terme $2n_\theta$ permet d'introduire une pénalité sur le nombre de paramètres pour les modèles complexes. Le modèle retenu est celui qui minimise le critère AIC (plus de précision avec moins de paramètres).

Critère de l'erreur de prédiction finale d'Akaike (*Akaike's Final Prediction Error Criterion - FP*E) : ce critère est défini par :

$$FPE = \frac{1}{2}\frac{N + n_\theta}{N - n_\theta} J \tag{1.25}$$

Le terme $(N + n_\theta)/(N - n_\theta)$ décroît avec n_θ. Il représente une pénalité pour la sur-paramétrisation.

Critère de la longueur de description minimum de Rissanen : (*Rissanen's Minimum Description Length Criterion - MDL*) : l'idée est de trouver « la plus simple description possible » des données. Le critère est défini par :

$$MDL = \ln J + \frac{n_\theta}{N} \ln N \tag{1.26}$$

La « plus simple description » correspond au minimum du critère MDL.

Ces critères de sélection sont basés sur des considérations statistiques avec les hypothèses suivantes [51, 52] :

- le vrai modèle du système est correctement spécifié ou sur-paramétré par les modèles candidats ;
- les paramètres du modèle sont identifiables ;
- les données de validation possèdent les mêmes propriétés statistiques du deuxième ordre que les données d'estimation.

Le critère MDL est surtout utilisé si peu de données sont disponibles. L'expression (1.25) du critère FPE montre que le terme de pénalité n'est pas très significatif si le nombre de paramètres n_θ est très petit devant le nombre d'observations N. Pour ces raisons, le critère de sélection AIC est utilisé tout au long de cette thèse. Mais dans le cadre de la toolbox développée, l'utilisateur a le choix entre plusieurs critères de sélection de modèles.

1.6 Estimation des paramètres de modèles

L'identification paramétrique résulte de la minimisation du critère J défini par la relation (1.21). Cependant deux cas sont à considérer :

- *premier cas :* l'ensemble des données (base d'apprentissage D_N constituée de N données d'observation sur le système) nécessaires à l'identification est disponible et l'estimation est faite sur cet ensemble simultanément : on parle alors de méthodes globales de minimisation. L'évaluation du critère J se fait sur toute la base d'apprentissage D_N. Le problème d'optimisation décrit par l'équation (1.6) peut se mettre sous la forme suivante :

$$\underline{\hat{\theta}} = \arg\min_{\underline{\theta}} \left(J(\underline{\theta}, D_N) \right) \tag{1.27}$$

  où :

$$J(\underline{\theta}, D_N) = \frac{1}{2} \sum_{t=1}^{N} \varepsilon(t, \underline{\theta})^2 \tag{1.28}$$

$$\varepsilon(t, \underline{\theta}) = y_s(t) - \hat{y}(t, \underline{\theta}) \tag{1.29}$$

  L'inconvénient de cette approche est la nécessité de disposer de l'ensemble des éléments du vecteur de régression $\underline{\varphi}(t)$ (voir relation (1.4)) avant de pouvoir identifier le modèle. Cette démarche ne s'applique pas aux modèles récurrents qui, en dehors des données issues des mesures, font intervenir des données issues du modèle en cours d'identification. L'approche ne convient pas non plus à l'identification en temps réel de systèmes pour lesquels les données nécessaires ne sont pas entièrement disponibles.
- *deuxième cas :* certaines données nécessaires à l'identification des modèles récurrents ne sont pas entièrement disponibles (par exemple sorties estimés ou erreurs de prédiction). Il convient alors de faire une estimation récursive qui permet une estimation des paramètres au fur et à mesure que les données sont disponibles.

La minimisation du critère J se fait de proche en proche sur des sous-ensembles réduits de D_N où toutes les données nécessaires sont disponibles. Cette approche est aussi adaptée à l'identification en temps réel de systèmes.

Remarque *: En général, lorsque les unités de mesure diffèrent d'une variable à l'autre, il est recommandé de normaliser les variables afin d'atténuer l'effet d'échelle et obtenir des paramètres de même ordre de grandeur.*

Les paragraphes suivants décrivent différentes méthodes d'estimation paramétrique des modèles non linéaires.

1.6.1 Méthodes d'estimation globale

1.6.1.1 Optimisation linéaire

Un modèle peut être non linéaire par rapport au vecteur de régression mais linéaire par rapport aux paramètres. Dans ce cas l'équation (1.4) peut s'écrire sous la forme :

$$\hat{y}(t+h) = F\Big(\underline{\varphi}(t), \underline{\theta}\Big) = \phi\Big(\underline{\varphi}(t)\Big)\underline{\theta} = \underline{\Phi}(t)^T\,\underline{\theta} \tag{1.30}$$

où $\phi(\cdot)$ est une fonction non linéaire du vecteur de régression $\underline{\varphi}(t)$ telle que

$$\underline{\Phi}(t) = \phi\Big(\underline{\varphi}(t)\Big)$$

La valeur optimale $\underline{\hat{\theta}}_{opt}$ minimisant le critère (1.21) est obtenue par l'estimateur des moindres carrés :

$$\underline{\hat{\theta}}_{opt} = \left(\Phi_I^T\Phi_I\right)^{-1}\Phi_I^T\underline{y}_s \tag{1.31}$$

où :

Φ_I est la matrice d'information :

$$\Phi_I = \begin{pmatrix} \underline{\Phi}(1)^T \\ \vdots \\ \underline{\Phi}(N)^T \end{pmatrix}$$

$\underline{y}_s$ est le vecteur de sortie :

$$\underline{y}_s = \begin{pmatrix} y_s(1) \\ \vdots \\ y_s(N) \end{pmatrix}$$

Dans le cas où la matrice $\Phi_I^T \Phi_I$ est mal conditionnée (le rapport entre la plus grande valeur propre et la plus petite est élevé), son inversion pose problème et il n'est pas possible d'estimer $\underline{\theta}$ par (1.31). On utilise dans ce cas une technique de régularisation (voir [53, 54]) qui consiste à modifier le critère à minimiser en y ajoutant un terme de pénalité permettant d'améliorer le conditionnement de la matrice $\Phi_I^T \Phi_I$.

On peut par exemple utiliser le critère modifié [7] :

$$J_{reg}(\underline{\theta}, D_N) = J(\underline{\theta}, D_N) + \frac{\lambda}{2} \underline{\theta}^T K \underline{\theta} \tag{1.32}$$

où λ est un coefficient de régularisation et K une matrice permettant de coder les pénalités. On utilise très couramment la « *ridge regularization* » ou « *weight decay* » correspondant à $K = I$, I étant la matrice identité. La valeur optimale $\underline{\hat{\theta}}_{opt}$ est alors donnée par :

$$\underline{\hat{\theta}}_{opt} = \left(\Phi_I^T \Phi_I + \lambda I\right)^{-1} \Phi_I^T y_s \tag{1.33}$$

1.6.1.2 Optimisation non linéaire

Pour les modèles non linéaires par rapport aux paramètres, l'estimation paramétrique se fait par une méthode itérative d'optimisation non linéaire. Cette technique est basée sur la recherche d'une direction de l'espace des paramètres suivant laquelle le critère $J(\underline{\theta}, D_N)$ diminue, et la modification pas à pas de $\underline{\theta}$ dans cette direction. L'équation de modification du vecteur des paramètres $\underline{\theta}$ est de la forme générale :

$$\underline{\hat{\theta}}_{k+1} = \underline{\hat{\theta}}_k - \mu_k \underline{d}_k \tag{1.34}$$

où :

$\underline{\hat{\theta}}_k$ est l'estimée de $\underline{\theta}$ à l'itération k ;

μ_k est le pas de recherche ;

$\underline{d}_k$ est la direction de recherche dans l'espace des paramètres.

Il existe différentes méthodes d'optimisation selon le choix de la direction de recherche $\underline{d}_k$. Nous présentons ici certaines de ces méthodes qui font intervenir le gradient ou le hessien du critère J. Pour ces méthodes, il est donc nécessaire que le critère J possède des dérivées premières ou des dérivées secondes selon le cas.

Méthode du gradient à pas constant : cette méthode basée sur la technique de descente du gradient à pas constant permet d'obtenir une convergence vers un minimum

local. Elle ne permet cependant pas de garantir une convergence vers le minimum global. La relation (1.34) se met sous la forme suivante :

$$\underline{\hat{\theta}}_{k+1} = \underline{\hat{\theta}}_k - \mu \underline{G}_k \tag{1.35}$$

où :

μ est le pas de recherche (constant) ;

$\underline{G}_k$ est le gradient du critère J estimé à $\underline{\hat{\theta}}_k$:

$$\underline{G}_k = \nabla(J_k) = \left.\frac{\partial J}{\partial \underline{\theta}}\right|_{\underline{\theta}=\underline{\hat{\theta}}_k}$$

Il est cependant nécessaire de choisir judicieusement le pas de descente μ. Une faible valeur de μ conduit à une convergence lente vers le minimum alors qu'une valeur élevée induit des oscillations autour du minimum.

Méthode du gradient à pas variable : Cette méthode permet de pallier au problème que pose le choix d'un pas de recherche constant. Elle consiste à adapter la variation du pas de recherche de façon à descendre rapidement quand on est loin du minimum et lentement quand on s'y approche. On parvient ainsi à augmenter la vitesse de convergence de l'algorithme tout en évitant des oscillations autour du minimum.

$$\underline{\hat{\theta}}_{k+1} = \underline{\hat{\theta}}_k - \mu_k \underline{G}_k \tag{1.36}$$

Parmi les algorithmes les plus utilisés, on peut citer les méthodes de dichotomie, les méthodes de minimisation de Nash [55] et de Wolfe et Powell [56].

Méthode de Newton : Cette méthode exploite la « courbure » du critère J pour atteindre le minimum plus rapidement. Le pas est choisi égal à 1 alors que la direction de recherche est calculée par la relation :

$$\underline{d}_k = H_k^{-1} G_k$$

avec H_k désignant le hessien de J calculé à $\underline{\hat{\theta}}_k$:

$$H_k = \left.\frac{\partial^2 J}{\partial \underline{\theta}\, \partial \underline{\theta}^T}\right|_{\underline{\theta}=\underline{\hat{\theta}}_k}$$

La relation (1.34) se met sous la forme :

$$\underline{\hat{\theta}}_{k+1} = \underline{\hat{\theta}}_k - H_k^{-1} \underline{G}_k \tag{1.37}$$

Le principal inconvénient de cette méthode est la difficulté que présente le calcul du hessien H et son inversion à chaque itération :

$$H = \frac{\partial^2 J}{\partial \underline{\theta}\, \partial \underline{\theta}^T} = \sum_{k=1}^{N} \varepsilon(k) \frac{\partial^2 \hat{y}(k)}{\partial \underline{\theta}\, \partial \underline{\theta}^T} + \sum_{k=1}^{N} \frac{\partial \hat{y}(k)}{\partial \underline{\theta}} \frac{\partial \hat{y}(k)}{\partial \underline{\theta}^T}$$

De plus, si le hessien n'est pas défini positif pour $\underline{\hat{\theta}}_k$, rien ne garantit la convergence vers un minimum ou vers un maximum. Afin d'éviter une divergence de l'algorithme, on utilise μ comme coefficient de relaxation permettant une modification des paramètres dans le sens de la diminution du critère :

$$\underline{\hat{\theta}}_{k+1} = \underline{\hat{\theta}}_k - \mu H_k^{-1} \underline{G}_k$$

Le coefficient de relaxation est déterminé suivant une heuristique analogue à celle utilisée pour le choix du pas de recherche dans l'algorithme du gradient à pas variable.

Méthode de Gauss-Newton : la simplification de la méthode de Newton conduit à la méthode de Gauss-Newton dans laquelle le hessien H est approché par une matrice H_a obtenue en négligeant la dérivée d'ordre 2 de $\hat{y}$:

$$H_a = \sum_{k=1}^{N} \frac{\partial \hat{y}(k)}{\partial \underline{\theta}} \frac{\partial \hat{y}(k)}{\partial \underline{\theta}^T}$$

La méthode garantit la convergence vers un minimum puisque le hessien approché H_a est défini positif, en revanche elle ralentit la convergence au voisinage du minimum.

Méthode de Levenberg-Marquardt : dans cette méthode, le choix de la direction de recherche est issu d'un compromis entre la méthode du gradient régularisée et celle de Newton. Elle utilise le hessien régularisé H_r en lieu et place du hessien approché H_a (qui peut être mal conditionné) de la méthode de Gauss-Newton :

$$H_r = H_a + \lambda_k I$$

où λ_k est le coefficient de régularisation dont la valeur est itérativement modifiée en fonction de l'évolution du critère. La valeur de λ_k est augmentée si le citère diminue, elle est diminuée si le critère augmente.

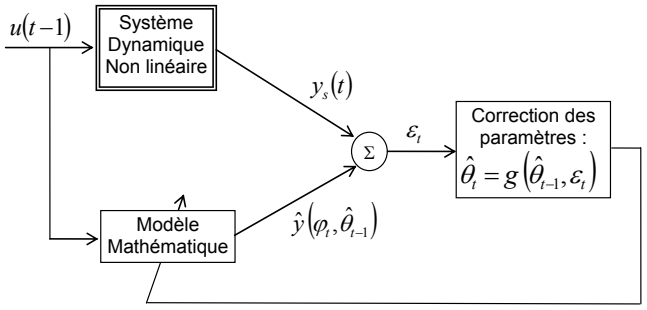

FIG. 1.5 – Principe de l'identification récursive des paramètres d'un modèle.

La relation (1.34) devient :

$$\underline{\hat{\theta}}_{k+1} = \underline{\hat{\theta}}_k - [H_a + \lambda_k I]^{-1} \underline{G}_k \tag{1.38}$$

Si λ_k est grand, on retrouve la méthode du gradient avec un pas de recherche $\mu_k = 1/\lambda_k$. Si λ_k est petit, la méthode se réduit à celle de Gauss-Newton.

1.6.2 Méthodes d'estimation récursive

Le principe de l'estimation récursive consiste à évaluer à chaque instant l'écart ε entre la sortie du modèle $\hat{y}$ et celle du système y_s et d'apporter une correction sur les paramètres $\hat{\theta}$ de façon à réduire l'écart à l'instant suivant. Ce principe est décrit sur la figure 1.5. Nous présentons ici les méthodes du gradient récursif et des moindres carrés récursifs. Ces méthodes sont utilisées pour les systèmes linéaires, mais nous montrons au chapitre 3 qu'elles sont applicables à l'estimation paramétrique de multimodèles récurrents pour l'identification de systèmes dynamiques non linéaires.

1.6.2.1 Méthode du gradient récursif

Le principe de la méthode consiste à modifier les paramètres du modèle à chaque instant k afin de minimiser le critère à l'instant suivant $(k+1)$:

$$J(k+1) = \frac{1}{2}\Big[y_s(k+1) - \hat{y}(k+1)\Big]^2 = \frac{1}{2}\Big[\varepsilon(k+1)\Big]^2 \tag{1.39}$$

où $\varepsilon(k+1)$ est l'erreur de prédiction *a posteriori* (erreur à l'instant $k+1$ calculée avec les paramètres estimés à l'instant $k+1$).

Les paramètres sont ajustés à chaque instant conformément à :

$$\underline{\hat{\theta}}_{k+1} = \underline{\hat{\theta}}_k - \mu \frac{\partial J(k+1)}{\partial \underline{\hat{\theta}}_{k+1}} \tag{1.40}$$

L'estimée de $\underline{\hat{\theta}}_k$ par (1.40) dépend de la structure du modèle. Si le modèle est linéaire par rapport aux paramètres conformément à (1.30), alors :

$$\varepsilon(k+1) = y_s(k+1) - \hat{y}(k+1) = y_s(k+1) - \underline{\Phi}^T(k)\ \underline{\hat{\theta}}_{k+1} \tag{1.41}$$

et

$$\frac{\partial J(k+1)}{\partial \underline{\hat{\theta}}_{k+1}} = \frac{\partial \varepsilon(k+1)}{\partial \underline{\hat{\theta}}_{k+1}} \varepsilon(k+1) = \underline{\Phi}(k)\, \varepsilon(k+1) \tag{1.42}$$

ce qui donne :

$$\underline{\hat{\theta}}_{k+1} = \underline{\hat{\theta}}_k + \mu\, \underline{\Phi}(k)\, \varepsilon(k+1) \tag{1.43}$$

$\varepsilon(k+1)$ est fonction de $\underline{\hat{\theta}}_{k+1}$. En utilisant l'erreur de prédiction *a priori* $\tilde{\varepsilon}$ (erreur à l'instant $k+1$ calculée avec les paramètres estimés à l'instant k) définie par :

$$\tilde{\varepsilon}(k+1) = y_s(k+1) - \hat{y}_{\underline{\hat{\theta}}_k}(k+1) = y_s(k+1) - \underline{\Phi}^T(k)\ \underline{\hat{\theta}}_k \tag{1.44}$$

L'équation (1.41) devient :

$$\varepsilon(k+1) = y_s(k+1) - \underline{\Phi}^T(k)\underline{\hat{\theta}}_k + \underline{\Phi}^T(k)\underline{\hat{\theta}}_k - \underline{\Phi}^T(k)\ \underline{\hat{\theta}}_{k+1} \tag{1.45}$$

soit :

$$\varepsilon(k+1) = \tilde{\varepsilon}(k+1) - \underline{\Phi}^T(k)\left[\underline{\hat{\theta}}_{k+1} - \underline{\hat{\theta}}_k\right] \tag{1.46}$$

En utilisant l'équation (1.43) on obtient :

$$\varepsilon(k+1) = \tilde{\varepsilon}(k+1) - \underline{\Phi}^T(k)\, \mu\, \underline{\Phi}(k)\, \varepsilon(k+1) \tag{1.47}$$

d'où :

$$\varepsilon(k+1) = \frac{\tilde{\varepsilon}(k+1)}{1 + \underline{\Phi}^T(k)\, \mu\, \underline{\Phi}(k)} \tag{1.48}$$

On obtient finalement l'équation de récurrence permettant d'estimer $\underline{\hat{\theta}}_{k+1}$ à partir de $\underline{\hat{\theta}}_k$:

$$\underline{\hat{\theta}}_{k+1} = \underline{\hat{\theta}}_k - \frac{\mu\, \underline{\Phi}(k)}{1 + \underline{\Phi}^T(k)\, \mu\, \underline{\Phi}(k)} \tag{1.49}$$

1.6.2.2 Méthode des Moindres Carrés Récursifs

La méthode des moindres carrés récursifs est basée sur la minimisation d'un critère quadratique sur l'ensemble des observations disponibles jusqu'à l'instant courant k. On considère le cas d'un système linéaire pour lequel le modèle peut être représenté sous la forme de l'équation (1.30). A l'instant k, le critère $J(k)$ s'écrit :

$$J(k) = \frac{1}{2}\sum_{t=1}^{k}[\varepsilon(t)]^2 = \frac{1}{2}\sum_{t=1}^{k}\left[y_s(t) - \underline{\Phi}^T(t-1)\,\hat{\underline{\theta}}_k\right]^2 \tag{1.50}$$

La minimisation du critère conduit à l'équation suivante :

$$\frac{\partial J(k)}{\partial \hat{\underline{\theta}}_k} = -\sum_{t=1}^{k}\left[y_s(t) - \underline{\Phi}^T(t-1)\,\hat{\underline{\theta}}_k\right]\underline{\Phi}(t-1) = 0 \tag{1.51}$$

A partir de l'équation (1.51), on peut établir la relation :

$$\hat{\underline{\theta}}_k = \left[\sum_{t=1}^{k}\underline{\Phi}(t-1)\,\underline{\Phi}^T(t-1)\right]^{-1}\sum_{t=1}^{k}y_s(t)\underline{\Phi}(t-1) \tag{1.52}$$

En posant $A_k = \left[\sum_{t=1}^{k}\underline{\Phi}(t-1)\,\underline{\Phi}^T(t-1)\right]^{-1}$, la relation (1.52) s'exprime aussi par :

$$\hat{\underline{\theta}}_k = A_k\sum_{t=1}^{k}y_s(t)\underline{\Phi}(t-1) \tag{1.53}$$

Cette relation permet d'écrire :

$$\hat{\underline{\theta}}_{k+1} = A_{k+1}\sum_{t=1}^{k+1}y_s(t)\underline{\Phi}(t-1) \tag{1.54}$$

Le terme $B = \sum_{t=1}^{k+1} y_s(t)\underline{\Phi}(t-1)$ dans (1.54) peut être réécrit comme suit :

$$\begin{aligned} B &= \sum_{t=1}^{k} y_s(t)\underline{\Phi}(t-1) + y_s(k+1)\underline{\Phi}(k) \\ &= \sum_{t=1}^{k} y_s(t)\underline{\Phi}(t-1) + y_s(k+1)\underline{\Phi}(k) + \underline{\Phi}(k)\underline{\Phi}^T(k)\hat{\underline{\theta}}_k - \underline{\Phi}(k)\underline{\Phi}^T(k)\hat{\underline{\theta}}_k \\ &= A_k^{-1}\,\hat{\underline{\theta}}_k + \underline{\Phi}(k)\,\underline{\Phi}^T(k)\,\hat{\underline{\theta}}_k + \underline{\Phi}(k)\left[y_s(k+1) - \underline{\Phi}^T(k)\,\hat{\underline{\theta}}_k\right] \\ &= \left[A_k^{-1} + \underline{\Phi}(k)\,\underline{\Phi}^T(k)\right]\hat{\underline{\theta}}_k + \underline{\Phi}(k)\left[y_s(k+1) - \underline{\Phi}^T(k)\,\hat{\underline{\theta}}_k\right] \\ &= A_{k+1}^{-1}\hat{\underline{\theta}}_k + \underline{\Phi}(k)\left[y_s(k+1) - \underline{\Phi}^T(k)\,\hat{\underline{\theta}}_k\right] \end{aligned}$$

On obtient alors la relation :

$$\sum_{t=1}^{k+1} y_s(t)\underline{\Phi}(t-1) = A_{k+1}^{-1}\hat{\underline{\theta}}_k + \underline{\Phi}(k)\,\tilde{\varepsilon}(k+1) \tag{1.55}$$

avec $\tilde{\varepsilon}(k+1) = y_s(k+1) - \underline{\Phi}^T(k)\,\hat{\underline{\theta}}_k$ étant l'erreur de prédiction *a priori* (erreur à l'instant $k+1$ évaluée avec les paramètres estimés à l'instant k).

A partir de la relation (1.54), on peut écrire :

$$\sum_{t=1}^{k+1} y_s(t)\underline{\Phi}(t-1) = A_{k+1}^{-1}\hat{\underline{\theta}}_{k+1} \tag{1.56}$$

En utilisant (1.55) et (1.56), on obtient :

$$\hat{\underline{\theta}}_{k+1} = \hat{\underline{\theta}}_k + A_{k+1}\,\underline{\Phi}(k)\,\tilde{\varepsilon}(k+1) \tag{1.57}$$

A_{k+1} dans la relation (1.57) peut être calculé comme suit :

$$\begin{aligned} [A_{k+1}]^{-1} &= \sum_{t=1}^{k+1} \underline{\Phi}(t-1)\,\underline{\Phi}^T(t-1) \\ &= \sum_{t=1}^{k} \underline{\Phi}(t-1)\,\underline{\Phi}^T(t-1) + \underline{\Phi}(k)\,\underline{\Phi}^T(k) \\ &= [A_k]^{-1} + \underline{\Phi}(k)\,\underline{\Phi}^T(k) \end{aligned}$$

Lemme : Soit A une matrice régulière de dimension $n \times n$ et $\underline{\varphi}$ un vecteur de dimension n. Alors :

$$(A^{-1} + \underline{\varphi}\,\underline{\varphi}^T)^{-1} = A - \frac{A\,\underline{\varphi}\,\underline{\varphi}^T\,A}{1 + \underline{\varphi}^T\,A\,\underline{\varphi}}$$

En utilisant ce lemme, A_{k+1} peut être calculé avec la relation suivante :

$$A_{k+1} = A_k - \frac{A_k \, \underline{\Phi}(k) \, \underline{\Phi}^T(k) \, A_k}{1 + \underline{\Phi}^T(k) \, A_k \, \underline{\Phi}(k)} \tag{1.58}$$

Le vecteur de paramètres $\underline{\hat{\theta}}$ peut être mis à jour de façon récursive à chaque instant en utilisant les relations (1.57) et (1.58).

1.7 Conclusion

Dans ce chapitre, nous avons présenté les différents aspects liés à la représentation des systèmes dynamiques non linéaires. Différents modèles non linéaires ainsi que différentes structures de représentations permettant leur implantation ont également été présentés. Les méthodes d'estimation paramétrique les plus fréquemment utilisées ont aussi été présentées. Ces méthodes sont utilisées dans les chapitres suivants.

Le principe de la sélection des modèles, basée sur des critères de performances mesurant la parcimonie ou la capacité de généralisation, a également été abordé.

Dans les chapitres suivants, les structures neuronales et multimodèles seront développées. Des méthodologies améliorant l'identification des systèmes dynamiques non linéaires seront mises en œuvre.

Chapitre 2

Identification de Systèmes Dynamiques Non Linéaires par Réseaux de Neurones

Sommaire

2.1 Introduction

Inspirés du fonctionnement du cerveau humain, les Réseaux de Neurones Artificiels (RNA) occupent aujourd'hui une place prépondérante dans plusieurs domaines des Sciences pour l'Ingénieur. Les progrès accomplis dans la compréhension du fonctionnement du cerveau humain ont contribué dans une large mesure au développement des RNA. Cependant les scientifiques sont toujours impressionnés par l'architecture du système neuronal humain, sa complexité, son efficacité et sa rapidité dans le traitement de certains problèmes complexes (par exemple ceux liés à la perception) que le plus puissant des ordinateurs actuels ne peut résoudre avec la même efficacité.

Les domaines d'application des RNA sont très nombreux : contrôle et commande de processus ([3, 57]), identification ([58, 4, 59]), classification ([60]), prédiction ([61, 62, 63]), diagnostic ([64, 65]), reconnaissance de forme ([66]), etc. Il existe plusieurs types de RNA (voir annexe A). Les RNA de type Perceptrons Multi-Couches (*Multi Layer Perceptron - MLP*) sont les plus utilisés, notamment dans les problèmes de régression non linéaire qui sont à la base de ce travail. Ils font donc l'objet d'une présentation dans ce chapitre dans le but d'une part, de pouvoir comparer les performances de l'architecture neuronale MLP à celle d'un multimodèle (voir chapitre 3), et d'autre part de pouvoir les intégrer dans une architecture multimodèle hétérogène.

Après une brève présentation de l'architecture des réseaux MLP, l'apprentissage des réseaux MLP non récurrents et récurrents est abordé, puis les implantations neuronales des différents modèles non linéaires exposés au chapitre 1 sont présentées. Une procédure d'identification structurelle des réseaux MLP (détermination de la topologie) est également proposée. Un exemple illustratif de l'importance du choix de la structure de modèle non linéaire en fonction du bruit agissant sur le système est aussi présenté.

2.2 Architecture d'un réseau MLP

Un réseau MLP comporte une ou plusieurs couches cachées à fonction d'activation de type sigmoïde ainsi qu'une couche de sortie. Les neurones de la couche caché c reçoivent des informations des neurones (ou unités) de la couche $c-1$ et sont connectés aux neurones de la couche $c+1$. Il n'existe pas de connexions entre les neurones d'une même couche. Chaque neurone de la couche de sortie réalise une fonction non linéaire des entrées du

réseau. Le potentiel v_i d'un neurone i et son activation O_i à un instant t sont donnés par :

$$v_i(t) = \sum_{j=1}^{p} \omega_{ij} x_j(t) - b_i \tag{2.1}$$

$$O_i(t) = f_i(v_i(t)) \tag{2.2}$$

où :

x_j représente l'entrée j du réseau si le neurone i appartient à la première couche cachée, ou dans le cas contraire, la sortie O_j du neurone j de la couche cachée précédant celle du neurone i ;

b_i est une constante scalaire appelée biais ;

$f_i(\cdot)$ est la fonction d'activation du neurone i. Elle peut être différente selon que le neurone soit un neurone caché ou un neurone de sortie. En général, tous les neurones cachés ont la même fonction d'activation qui est souvent une fonction sigmoïde (par exemple $f(v) = 1/(1 + \exp(-v))$. Les neurones de sortie ont également la même fonction d'activation, en général linéaire (par exemple $f(v) = v$).

Les MLP sont utilisés dans plusieurs types d'applications telles que : classification (y compris pour des classes non linéairement séparables), approximation de fonctions, identification, etc. C'est ce type de réseau que nous utilisons dans ce travail pour l'identification des systèmes dynamiques non linéaires. La figure 2.1 représente un schéma simplifié d'un réseau MLP à deux entrées, comportant une couche cachée à trois neurones et une couche de sortie à deux neurones. Le MLP à une couche cachée avec des fonctions d'activation sigmoïde et une couche de sortie constituée d'un neurone linéaire possède la propriété d'approximateur universel parcimonieux [11, 12, 13, 14].

2.3 Processus d'apprentissage dans les réseaux MLP

Les réseaux MLP utilisent un mode d'apprentissage supervisé. Dans ce mode d'apprentissage, un ensemble de données constitué des entrées du système à modéliser et des sorties correspondantes est présenté au réseau qui doit adapter ses paramètres suivant un algorithme d'apprentissage de façon à ce que la différence entre la sortie du système et celle du modèle soit suffisamment faible.

Les techniques d'apprentissage les plus utilisées dans les réseaux MLP sont l'algorithme de rétropropagation du gradient (Back Propagation - BP, [67]) pour les MLP

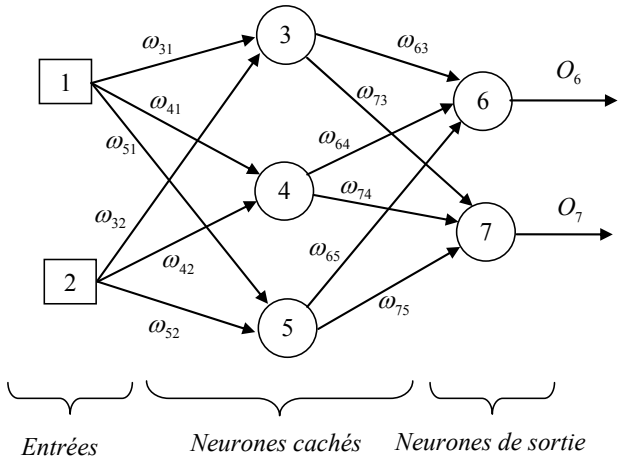

FIG. 2.1 – Schéma simplifié d'un perceptron multi-couche à deux entrées, une couche cachée et une couche de sortie.

non récurrents et l'algorithme de rétro propagation à travers le temps (Back Propagation Through Time - BPTT, voir [17]) pour les MLP récurrents. Nous présentons ici ces deux techniques d'apprentissage.

2.3.1 Algorithme d'apprentissage dans un MLP non récurrent

Considérons un réseau MLP comportant plusieurs couches cachées et une couche de sortie à plusieurs neurones. A un instant t donné, l'erreur entre la sortie désirée $d_i(t)$ et la sortie $O_i(t)$ produite par le neurone de sortie d'indice i s'exprime par :

$$e_i(t) = d_i(t) - O_i(t) \tag{2.3}$$

L'erreur quadratique moyenne du réseau est :

$$\varepsilon(t) = \frac{1}{2}\sum_{i\in S}[e_i(t)]^2 \tag{2.4}$$

où S est l'ensemble des indices des neurones de sortie.

L'objectif de l'apprentissage est de minimiser le critère J défini par la relation (1.28) dans laquelle $\underline{\theta}$ représente le vecteur des poids du réseau d'éléments ω_{ij}. L'algorithme de rétro propagation du gradient est un algorithme itératif permettant de modifier les paramètres, afin de minimiser l'erreur quadratique moyenne du réseau sur l'ensemble des

données d'apprentissage, suivant une technique de descente du gradient (voir 1.35) :

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \mu \frac{\partial \varepsilon(t)}{\partial \omega_{ij}(t)} \tag{2.5}$$

Le gradient de l'erreur $\frac{\partial \varepsilon(t)}{\partial \omega_{ij}(t)}$ peut être calculé de deux façon différentes suivant que le neurone i soit un neurone de sortie ou un neurone caché.

2.3.1.1 Evaluation du gradient pour un neurone de sortie

Pour un neurone de sortie, l'erreur $e_i(t)$ peut être évaluée par l'expression (2.3) qui conduit à :

$$\frac{\partial \varepsilon(t)}{\partial \omega_{ij}(t)} = \frac{\partial \varepsilon(t)}{\partial e_i(t)} \frac{\partial e_i(t)}{\partial O_i(t)} \frac{\partial O_i(t)}{\partial v_i(t)} \frac{\partial v_i(t)}{\partial \omega_{ij}(t)} \tag{2.6}$$

A partir des équations (2.4), (2.3), (2.2) et (2.1) il vient respectivement :

$$\frac{\partial \varepsilon(t)}{\partial e_i(t)} = e_i(t) \tag{2.7}$$

$$\frac{\partial e_i(t)}{\partial O_i(t)} = -1 \tag{2.8}$$

$$\frac{\partial O_i(t)}{\partial v_i(t)} = f_i'\Big(v_i(t)\Big) \tag{2.9}$$

$$\frac{\partial v_i(t)}{\partial \omega_{ij}(t)} = O_j(t) \tag{2.10}$$

L'évaluation du gradient de l'erreur donne alors :

$$\frac{\partial \varepsilon(t)}{\partial \omega_{ij}(t)} = -e_i(t) f_i'\Big(v_i(t)\Big) O_j(t) \tag{2.11}$$

soit :

$$\frac{\partial \varepsilon(t)}{\partial \omega_{ij}(t)} = \delta_i(t) O_j(t) \tag{2.12}$$

où $\delta_i(t)$ est le gradient local défini par :

$$\delta_i(t) = e_i(t) f_i'\Big(v_i(t)\Big) \tag{2.13}$$

2.3.1.2 Evaluation du gradient pour un neurone caché

Pour un neurone caché, il n'est pas possible d'évaluer l'erreur $e_i(t)$. Pour un neurone i de la dernière couche cachée, le gradient de l'erreur peut être déterminé par :

$$\frac{\partial \varepsilon(t)}{\partial \omega_{ij}(t)} = \frac{\partial \varepsilon(t)}{\partial O_i(t)} \frac{\partial O_i(t)}{\partial v_i(t)} \frac{\partial v_i(t)}{\partial \omega_{ij}(t)} \tag{2.14}$$

$$\begin{aligned}\frac{\partial \varepsilon(t)}{\partial O_i(t)} &= \frac{\partial}{\partial O_i(t)}\Big[\sum_{k\in S}[e_k(t)]^2\Big] \\ &= \sum_{k\in S} e_k(t)\frac{\partial e_k(t)}{\partial O_i(t)} \\ &= \sum_{k\in S} e_k(t)\frac{\partial e_k(t)}{\partial v_k(t)}\frac{\partial v_k(t)}{\partial O_i(t)} \\ &= \sum_{k\in S} e_k(t)\Big(-f_k'\Big(v_k(t)\Big)\Big)\omega_{ki}(t)\end{aligned} \tag{2.15}$$

D'où finalement :

$$\frac{\partial \varepsilon(t)}{\partial \omega_{ij}(t)} = -f_i'\Big(v_i(t)\Big)O_j(t)\sum_{k\in S} e_k(t)\Big(f_k'\Big(v_k(t)\Big)\Big)\omega_{ki}(t) \tag{2.16}$$

Par conséquent, le gradient local d'un neurone caché d'indice i s'exprime par :

$$\delta_i(t) = f_i'\Big(v_i(t)\Big)\sum_{k\in S} e_k(t)\Big(f_k'\Big(v_k(t)\Big)\Big)\omega_{ki}(t) \tag{2.17}$$

où

$$\delta_i(t) = f_i'\Big(v_i(t)\Big)\sum_{k\in S}\delta_k(t)\omega_{ki}(t) \tag{2.18}$$

Connaissant les gradients locaux des neurones de la dernière couche cachée, on peut calculer pas à pas les gradients locaux des neurones des couches cachées précédentes en utilisant de façon récursive la relation (2.18).

L'algorithme de rétro propagation du gradient peut ainsi se résumer en deux étapes :

- Pendant la première étape, l'information (les entrées) transite de l'entrée du réseau vers sa sortie, les poids des connexions étant maintenus fixes. On calcule alors les erreurs à la sortie du réseau.
- Pendant la deuxième étape, l'information (les erreurs à la sortie du réseau) transite de la sortie vers l'entrée et permet la modification des poids par la relation :

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \mu\delta_i(t)O_j(t) \tag{2.19}$$

où les gradients locaux $\delta_i(t)$ sont calculés par les relations (2.13) pour les neurones de sortie et (2.18) pour les neurones cachés.

Le principal inconvénient de l'algorithme de rétro propagation est qu'il n'y a pas de garantie de convergence vers le minimum global. De plus, le choix du pas de recherche

μ (voir 1.35) constitue une tâche difficile. Pour accélérer la vitesse de convergence de l'algorithme, il est proposé dans [67] de modifier (2.19) en y ajoutant un terme d'inertie (momentum) qui est d'autant plus grand que la différence entre deux valeurs successives de ω_{ij} est importante :

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha\Delta\omega_{ij}(t-1) + \mu\delta_i(t)O_j(t) \tag{2.20}$$

où

$\Delta\omega_{ij}(t-1) = \omega_{ij}(t) - \omega_{ij}(t-1)$
α une constante positive de valeur faible ($0 \leq \alpha < 1$).

2.3.2 Algorithme d'apprentissage dans un MLP récurrent

Deux algorithmes sont souvent utilisés pour l'apprentissage des réseaux de neurones récurrents (voir [68] pour plus de détails) :

Backpropagation Through Time (BPTT) **:** il s'agit de construire un réseau non bouclé qui reproduit le réseau bouclé sur sa trajectoire temporelle. A chaque instant, on dispose d'un exemplaire (ou copie) du réseau. La variable d'état (la variable rebouclée) utilisée comme entrée dans une copie correspondant à un instant t est obtenue de la copie du réseau à l'instant $t-1$. On obtient ainsi un réseau non bouclé et la modification des poids se fait suivant l'algorithme de rétro propagation du gradient. Les figures 2.2 et 2.3 illustrent un réseau récurrent et le réseau non récurrent obtenu par déroulement de ce dernier en trois copies. Il faut cependant noter qu'à chaque instant, toutes les copies du réseau doivent avoir les mêmes poids (les poids sont partagés). L'application stricte de cette méthode impose qu'il y'ait autant de copies que d'observations, puisqu'à chaque instant doit correspondre une copie. Ceci augmente considérablement le temps de calcul. C'est pour cette raison que le dépliement du réseau est limité à un nombre réduit d'instants correspondant à la largeur d'une « fenêtre temporelle » qu'on fait glisser de la première observation à la dernière. Une description de l'algorithme de BPTT est présentée dans [17]. Dans ce travail, l'algorithme de BPTT est implémenté pour l'identification de modèles récurrents par un réseau MLP.

Real Time Recurrent Learning (RTRL) **:** avec cette méthode, le gradient des états antérieurs par rapport aux poids courants est approché par le cumul des gradients par rapport aux anciens poids obtenus aux instants antérieurs [18]. A chaque instant

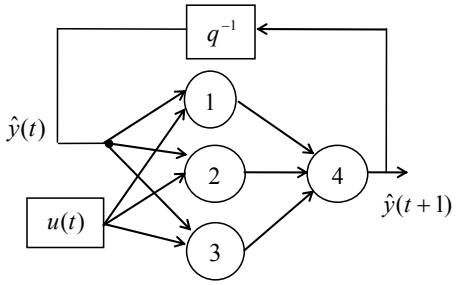

FIG. 2.2 – Exemple d'un réseau MLP bouclé.

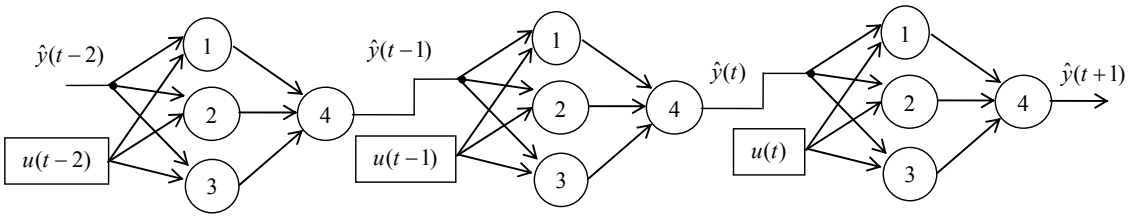

FIG. 2.3 – Obtention d'un réseau MLP non bouclé par dépliement du réseau MLP bouclé en trois copies.

k, il faut minimiser le critère :

$$J(k) = \sum_{t=1}^{k} \varepsilon(t) \tag{2.21}$$

$$\varepsilon(t) = \frac{1}{2}\Big[d(t) - O(t)\Big]^2 \tag{2.22}$$

où d représente la sortie désirée et O la sortie du réseau. La modification du poids ω_{ij} à l'instant k est :

$$\Delta\omega_{ij}(k) = \sum_{t=1}^{k} \Delta\omega_{ij}(t) \tag{2.23}$$

avec $\Delta\omega_{ij}(t) = -\mu\frac{\partial\varepsilon(t)}{\partial\omega_{ij}}$.

Cet algorithme est très utilisé dans des applications temps réel.

2.4 Identification de systèmes dynamiques non linéaires par un MLP

En général, un système dynamique est décrit par des équations différentielles s'il est à temps continu ou par des équations aux différences s'il est à temps discret. Dans la

pratique, il est rare qu'un système complexe puisse être entièrement décrit par un modèle de connaissance. On a souvent recours aux modèles entrées-sortie de type « boîtes noires », pour lesquels aucune connaissance sur le système n'est nécessaire, mais des mesures sur les variables régissant le fonctionnement du système sont indispensables et en quantité suffisante. Le problème de l'identification devient alors un problème de régression non linéaire.

Dans ce travail, l'étude est limitée à des MLP possédant une seule couche cachée avec des fonctions d'activation sigmoïde et une couche de sortie ayant un neurone à fonction d'activation linéaire, vue que cette structure a la propriété d'approximateur universel parcimonieux [11]. Cependant, la principale difficulté reste la détermination du nombre de neurones dans la couche cachée. Un algorithme combinant l'identification structurelle du réseau (détermination de la topologie) et l'identification paramétrique (détermination des poids synaptiques) est présenté. Le critère AIC de sélection de modèle est utilisé pour déterminer la meilleure architecture MLP. Les méthodologies développées sont illustrées par l'identification de systèmes dynamiques simulés. Les variables explicatives des systèmes sont supposées connues et le problème de leur détermination n'est pas abordé.

2.4.1 Représentation de systèmes dynamiques non linéaires par un réseau MLP

Les systèmes dynamiques non linéaires étudiés au paragraphe §1.4 peuvent être représentés par des réseaux de neurones de type MLP. La structure du réseau dépend de la classe du modèle définie par son vecteur de régression.

Pour un modèle NFIR (voir (1.12)), le vecteur de régression est :

$$\begin{aligned}\underline{\varphi}(t) \;=\; & \big[u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2)\big]^T\end{aligned}$$

L'implantation de ce modèle avec une structure MLP est représentée sur la figure 2.4.

Pour un modèle NARX (voir (1.14)), le vecteur de régression est :

$$\begin{aligned}\underline{\varphi}(t) \;=\; & \big[u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2), \\ & y_s(t-d_{y_s}+1), y_s(t-d_{y_s}), \ldots, y_s(t-d_{y_s}-n_{y_s}+2)\big]^T\end{aligned}$$

La figure 2.5 représente l'implantation d'un modèle NARX avec une structure MLP.

Pour un modèle NOE (voir (1.17)), le vecteur de régression est :

$$\begin{aligned}\underline{\varphi}(t) \;=\; & \big[u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & u_{n_i}(t-d_{u_{n_i}}+1), u_{n_i}(t-d_{u_{n_i}}), \ldots, u_{n_i}(t-d_{u_{n_i}}-n_{u_{n_i}}+2), \\ & \hat{y}(t-d_{\hat{y}}+1), \hat{y}(t-d_{\hat{y}}), \ldots, \hat{y}(t-d_{\hat{y}}-n_{\hat{y}}+2)\big]^T\end{aligned}$$

L'implantation d'un modèle NOE avec une structure MLP est représentée sur la figure 2.6.

Pour un modèle NARMAX (voir (1.18)), le vecteur de régression est :

$$\begin{aligned}\underline{\varphi}(t) \;=\; & \big[u_1(t-d_{u_1}+1), u_1(t-d_{u_1}), \ldots, u_1(t-d_{u_1}-n_{u_1}+2), \ldots, \\ & u_k(t-d_{u_k}+1), u_k(t-d_{u_k}), \ldots, u_k(t-d_{u_k}-n_{u_k}+2), \ldots, \\ & u_{n_k}(t-d_{u_{n_k}}+1), u_{n_k}(t-d_{u_{n_k}}), \ldots, u_{n_k}(t-d_{u_{n_k}}-n_{u_{n_k}}+2), \\ & y_s(t-d_{y_s}+1), y_s(t-d_{y_s}), \ldots, y_s(t-d_{y_s}-n_{y_s}+2), \\ & \varepsilon(t-d_{\varepsilon}+1), \varepsilon(t-d_{\varepsilon}), \ldots, \varepsilon(t-d_{\varepsilon}-n_{\varepsilon}+2)\big]^T\end{aligned}$$

La figure 2.7 représente l'implantation d'un modèle NARMAX avec une structure MLP.

Si pour un système donné les connaissances *a priori* ne permettent pas de connaître la structure du réseau MLP à utiliser, chacune des structures présentées ici doit être considérée. Un critère de sélection de modèle permet ensuite de déterminer le réseau MLP qui convient le mieux.

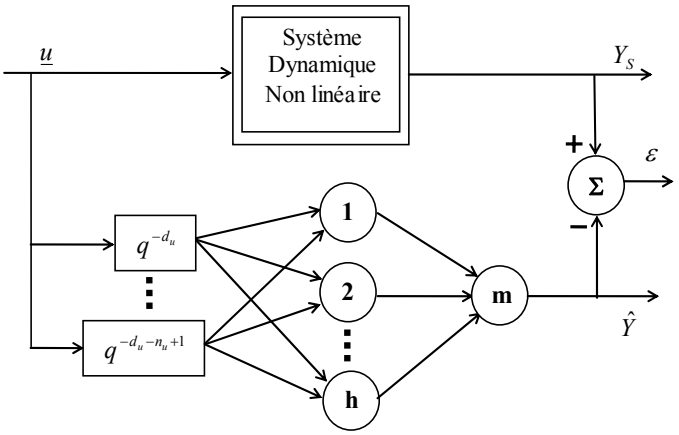

FIG. 2.4 – Exemple d'implantation d'un modèle NFIR par un réseau MLP. Les décalages temporels sur les entrées sont indiqués par $q^{-d_u}, \ldots, q^{-d_u-n_u+1}$.

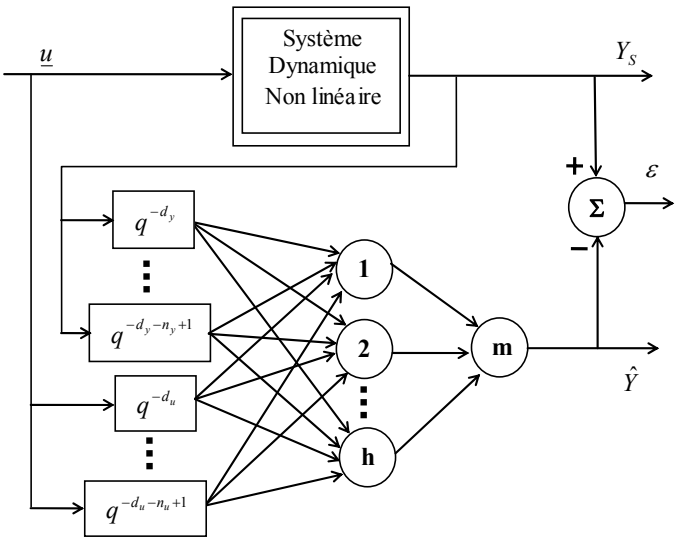

FIG. 2.5 – Exemple d'implantation d'un modèle NARX par un réseau MLP. Les décalages temporels sur les variables sont indiqués par $q^{-d_u}, \ldots, q^{-d_u-n_u+1}$ pour l'entrée et $q^{-d_y}, \ldots, q^{-d_y-n_y+1}$ pour la sortie mesurée.

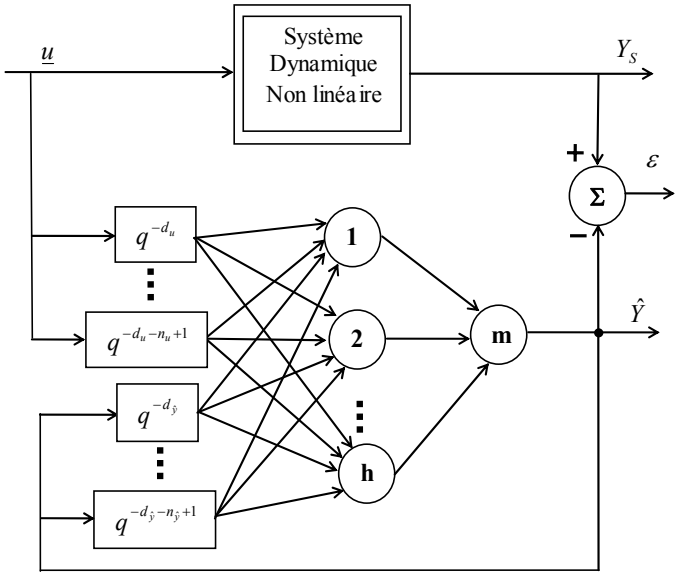

FIG. 2.6 – Exemple d'implantation d'un modèle NOE par un réseau MLP récurrent. Les décalages temporels sur les variables sont indiqués par $q^{-d_u}, \ldots, q^{-d_u-n_u+1}$ pour l'entrée et $q^{-d_{\hat{y}}}, \ldots, q^{-d_{\hat{y}}-n_{\hat{y}}+1}$ pour la sortie estimée.

2.4.2 Identification structurelle et paramétrique d'un MLP

Il existe plusieurs approches pour déterminer la topologie d'un réseau de neurones :

- approche empirique par essai-erreur ;
- approches ascendantes qui consistent à augmenter le nombre de connexions et de neurones dans le réseau (« *Cascade-Correlation* » [69], « *Upstart* » [70]) ;
- approches descendantes qui consistent à élaguer des connexions et des neurones dans le réseau pendant l'apprentissage (« *Weight Elimination* » [71]) ou après l'apprentissage ([72]) ;
- approches évolutives utilisant les algorithmes génétiques [71].

La méthode d'identification structurelle et paramétrique du réseau MLP que nous présentons ici est basée sur la démarche illustrée dans la figure 1.1 (voir chapitre 1). Le choix du modèle est fait avec le critère de sélection AIC. Le critère d'estimation paramétrique est le critère quadratique. L'algorithme d'estimation paramétrique est l'algorithme de rétro propagation du gradient (BP) si le réseau MLP est non récurrent, ou l'algorithme de rétro propagation du gradient à travers le temps (BPTT) si le réseau est récurrent. La méthode de détermination du nombre de neurones cachés proposée ici est une approche ascendante qui consiste à augmenter itérativement le nombre de neurones

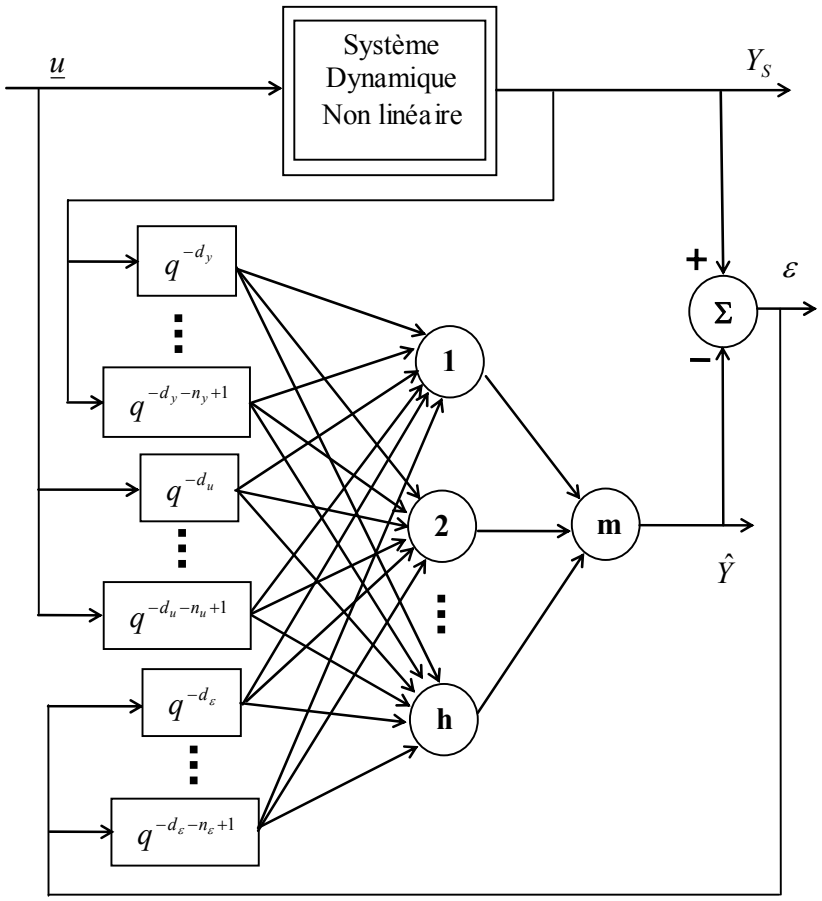

FIG. 2.7 – Exemple d'implantation d'un modèle NARMAX par un réseau MLP récurrent. Les décalages temporels sur les variables sont indiqués par $q^{-d_u}, \ldots, q^{-d_u-n_u+1}$ pour l'entrée, $q^{-d_y}, \ldots, q^{-d_y-n_y+1}$ pour la sortie mesurée et $q^{-d_\varepsilon}, \ldots, q^{-d_\varepsilon-n_\varepsilon+1}$ pour l'erreur de prédiction.

cachés, à estimer les paramètres et à évaluer, pour chaque topologie, la valeur du critère de sélection AIC. Tant que le critère s'améliore, on poursuit la procédure. Si à une itération donné le critère se dégrade, on retient la structure précédente.

L'augmentation du nombre de neurones cachés contribue à la diminution de l'erreur d'apprentissage, mais se traduit par une augmentation du nombre de paramètres du réseau, donc à un risque de surajustement dû à l'apprentissage du bruit. Le réseau aura alors une mauvaise capacité de généralisation. Une éventuelle dégradation du critère AIC suite à une augmentation du nombre de neurones cachés signifie que la diminution de l'erreur est moins importante par rapport l'augmentation du nombre de paramètres. L'approche est illustrée par l'organigramme représenté sur la figure 2.8. Il est utilisé au

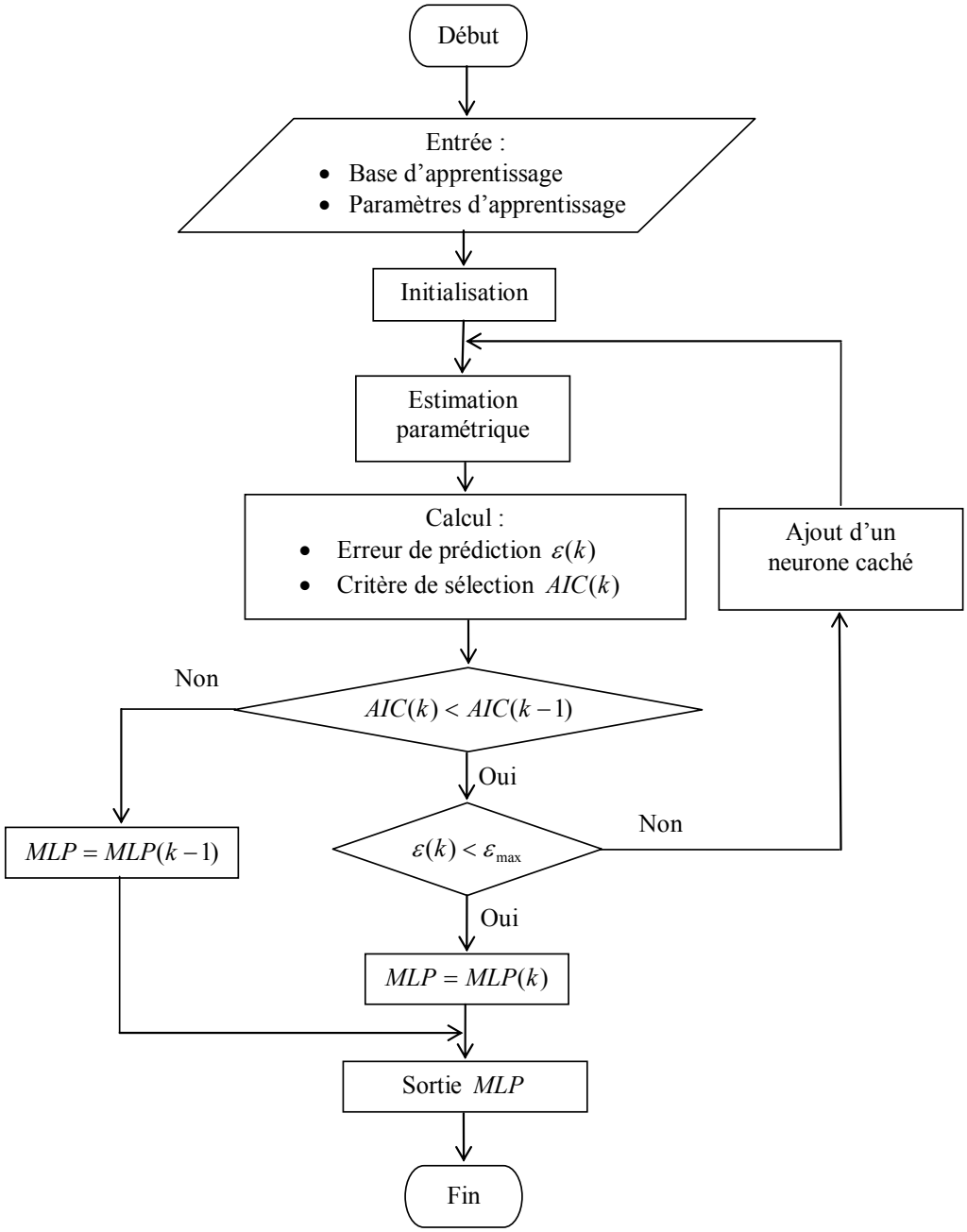

FIG. 2.8 – Organigramme d'identification structurelle et paramétrique d'un MLP à une couche cachée.

paragraphe suivant pour l'identification de modèles dynamiques non linéaires.

2.5 Exemple d'identification de système dynamique non linéaire par un réseau MLP

Considérons le modèle dynamique non linéaire décrit par Narendra et Parthasarathy [59] :

$$y_s(t+1) = \frac{y_s(t)y_s(t-1)y_s(t-2)u(t-1)\Big[y_s(t-2)-1\Big] + u(t)}{1+y_s(t-1)^2+y_s(t-2)^2} \tag{2.24}$$

où $u(t) \in [-1,\ 1]$ est l'entrée du système à l'instant t et $y_s(t)$ la sortie correspondante.

Les données sont générées pour plusieurs cas de figures : en absence de bruit, en présence de bruit de sortie additif, en présence de bruit d'état et en présence de bruit de sortie et de bruit d'état. Pour chaque cas de figure, les performances des structures NARX, NOE et NARMAX, notamment le critère AIC en apprentissage et l'erreur de validation, sont comparées afin de vérifier l'adéquation entre la structure du modèle non linéaire et le système à identifier. En effet, l'influence du bruit sur l'apprentissage se traduit par une mauvaise capacité de généralisation du modèle. Or, un choix judicieux de la structure de modèle non linéaire permet de minimiser l'influence du bruit sur l'estimation paramétrique.

L'exemple permet également de valider la méthodologie proposée pour l'identification structurelle et paramétrique des MLP à une couche cachée présentée à la figure 2.8.

Les performances des modèles en apprentissage et en validation sont évaluées par la racine carrée de l'erreur quadratique moyenne (*Root Mean Square Error - RMSE*) :

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\big(y_s(t)-\hat{y}(t)\big)^2} \tag{2.25}$$

où N est le nombre d'observations et $\hat{y}(\cdot)$ la sortie du modèle.

Les données sont générées pour t variant de 1 à 1600, l'entrée étant constituée de créneaux d'amplitudes et de largeurs variables. La première moitié des données ainsi générées est utilisée pour l'identification et l'autre moitié pour la validation.

La figure 2.9 représente l'entrée d'identification et la sortie non bruitée correspondante.

Pour un modèle *NARX* (voir paragraphe §1.4), les régresseurs sont des mesures

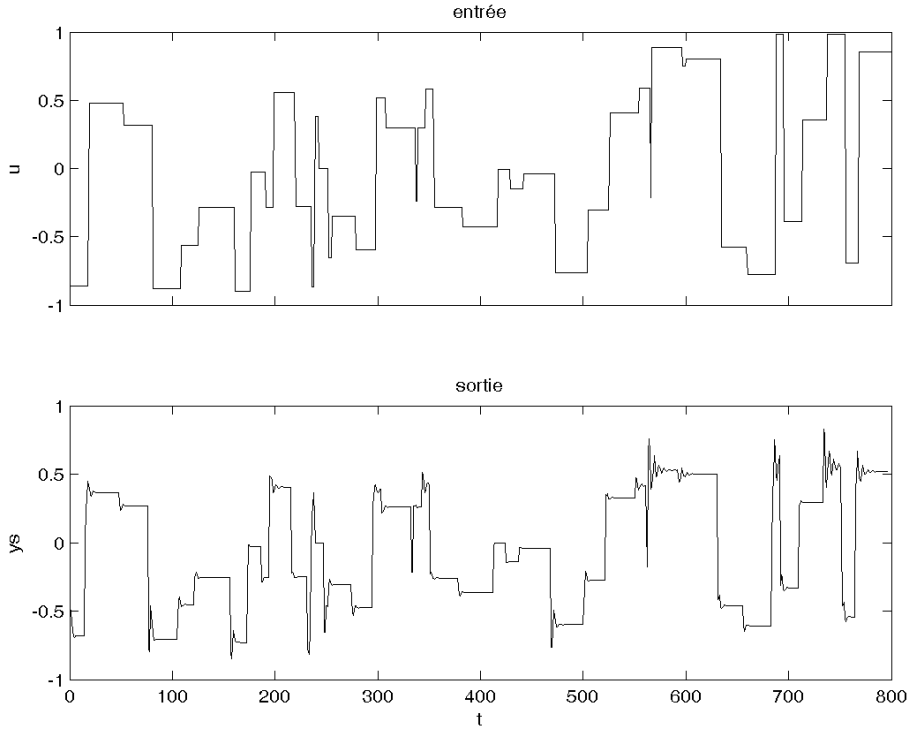

FIG. 2.9 – Entrée du système étudié et sortie non bruitée correspondante.

faites sur le système. Le vecteur de regression utilisé ici est :

$$\varphi_{NARX}(t) = \Big[u(t)\ u(t-1)\ y_s(t)\ y_s(t-1)\ y_s(t-2)\Big]$$

L'ensemble de la base d'apprentissage étant entièrement disponible, l'identification est faite en utilisant un apprentissage supervisé tel que présenté au paragraphe §2.3.1. La sortie du MLP *NARX* s'exprime :

$$y_{NARX}(t+1) = F\Big(\varphi_{NARX}(t), \underline{\theta}_{NARX}\Big)$$

où :

F est un réseau MLP à une couche cachée à fonction d'activation sigmoïde et un neurone de sortie linéaire ;

$\underline{\theta}_{NARX}$ vecteur de paramètres du réseau MLP NARX.

Pour la structure *NOE* (voir paragraphe §1.4), le vecteur de régression suivant est utilisé :

$$\varphi_{NOE}(t) = \Big[u(t)\ u(t-1)\ \hat{y}(t)\ \hat{y}(t-1)\ \hat{y}(t-2)\Big]$$

La sortie du MLP *NOE* est :

$$y_{NOE}(t+1) = F\Big(\varphi_{NOE}(t), \underline{\theta}_{NOE}\Big)$$

où $\underline{\theta}_{NOE}$ est le vecteur de paramètres du réseau MLP NOE.

Pour la structure *NARMAX* (voir paragraphe §1.4), le vecteur de régression utilisé est :

$$\varphi_{NARMAX}(t) = \Big[u(t)\ u(t-1)\ y_s(t)\ y_s(t-1)\ y_s(t-2)\ \varepsilon(t)\Big]$$

La sortie du réseau MLP *NARMAX* est :

$$y_{NARMAX}(t+1) = F\Big(\varphi_{NARMAX}(t), \underline{\theta}_{NARMAX}\Big)$$

où $\underline{\theta}_{NARMAX}$ est le vecteur de paramètres du réseau MLP NARMAX.

Les vecteurs de régression des structures *NOE* et *NARMAX* comportent des variables issues du modèle et par conséquent il est impossible de disposer de l'ensemble des données d'apprentissage au début de la phase d'identification. Il est alors nécessaire d'utiliser un algorithme d'apprentissage adaptatif permettant de modifier les paramètres au fur et à mesure que les variables inconnues sont évaluées (voir §2.3.2). Dans notre approche, l'utilisation d'un algorithme d'apprentissage adaptatif est uniquement due à l'indisponibilité des données d'apprentissage car dépendant de la sortie du modèle. Pendant la phase d'utilisation du réseau, les poids synaptiques ne sont pas modifiés, contrairement à l'usage courant d'un algorithme d'apprentissage adaptatif qui permet la modification des paramètres du réseau à chaque fois que c'est nécessaire, même pendant la phase d'utilisation (la base d'apprentissage est infinie).

2.5.1 En absence de bruit

Les données sont générées avec la relation (2.24). Les résultats obtenus en apprentissage et en validation sont présentés au tableau 2.1. Ces résultats montrent que le modèle *NARX* est plus parcimonieux et possède une capacité de généralisation meilleure.

La figure 2.10 représente l'entrée, la sortie du système et des modèles NARX, NOE et NARMAX en validation. La figure 2.11 représente les résidus obtenus pour chacun des modèles en validation. Les fonctions d'autocorrélation des résidus des modèles ainsi que les marges correspondant à un intervalle de confiance à 99% sont représentées sur la figure 2.12.

Classe de modèle	Architecture	AIC	*RMSE* app.	*RMSE* valid.
NARX	[5 3 1]	-6864	$1.3\ 10^{-2}$	$1.5\ 10^{-2}$
NOE	[5 3 1]	-5912	$2.4\ 10^{-2}$	$2.5\ 10^{-2}$
NARMAX	[6 3 1]	-3179	$13.2\ 10^{-2}$	$18.0\ 10^{-2}$

TAB. 2.1 – Résultats obtenus pour les modèles MLP de type *NARX*, *NOE* et *NARMAX* en absence de bruit.

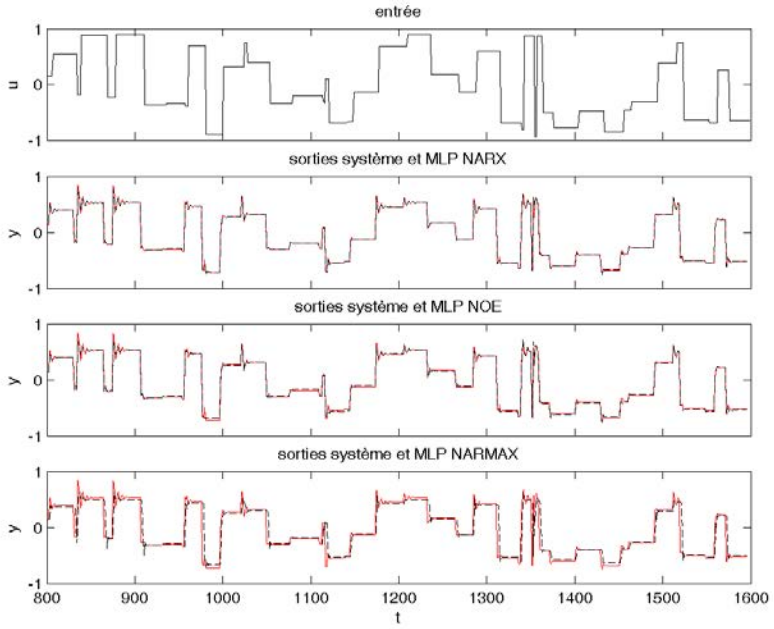

FIG. 2.10 – Identification en absence de bruit : entrée, sortie du système (trait plein) et sortie des modèles (trait discontinu) MLP NARX, MLP NOE et MLP NARMAX en validation.

2.5.2 En présence de bruit de sortie additif

La sortie bruitée $y_{s_{ba}}$, est générée en ajoutant un bruit à la sortie du système y_s de l'équation (2.24) :

$$y_{s_{ba}}(t) = y_s(t) + e_1(t)$$

où le bruit $e_1(t)$ est une réalisation d'une variable aléatoire qui suit une distribution normale centrée de variance 0.2, $\mathcal{N}(0, 0.2)$. Le rapport signal/bruit est de 7.5 dB. Le système ainsi obtenu est identifié avec les modèles MLP NARX, MLP NOE et MLP

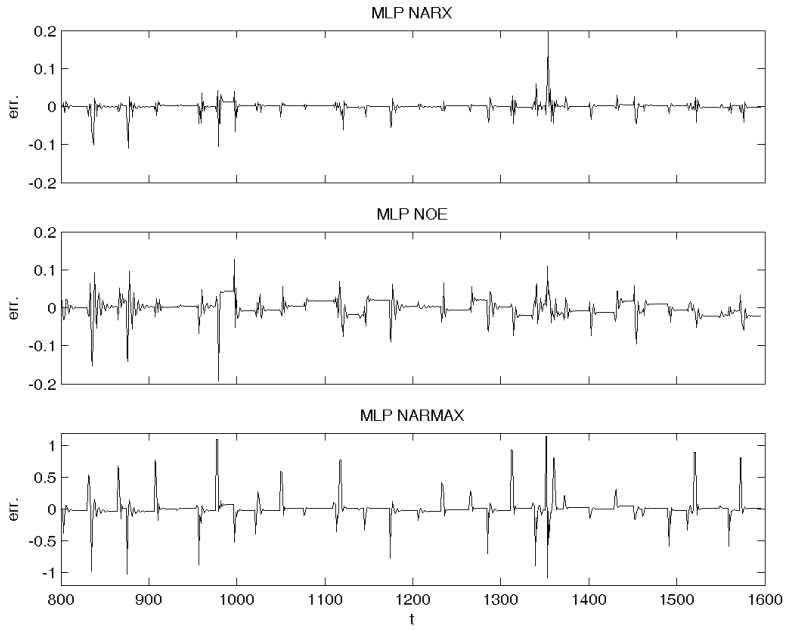

FIG. 2.11 – Résidus des modèles en validation en absence de bruit.

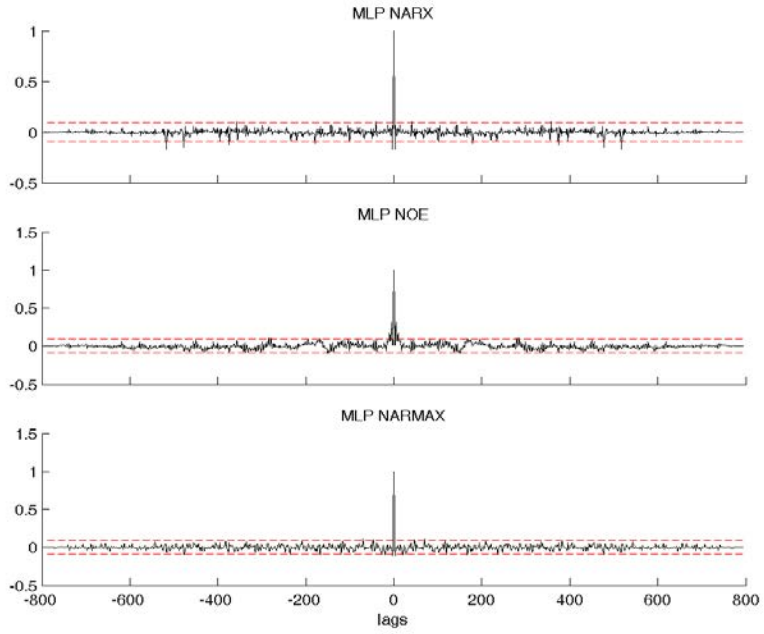

FIG. 2.12 – Fonctions d'autocorrélation des résidus des modèles en validation en absence de bruit.

NARMAX. En général, un modèle de type NOE est mieux adapté pour représenter un système dynamique non linéaire en présence de bruit de sortie (voir §1.4). La figure 2.13 représente l'évolution du critère AIC en apprentissage et de la $RMSE$ en validation (sur des signaux non bruités) en fonction du nombre de neurones cachés pour les structures NOE, NARX et NARMAX identifiées en présence de bruit de sortie.

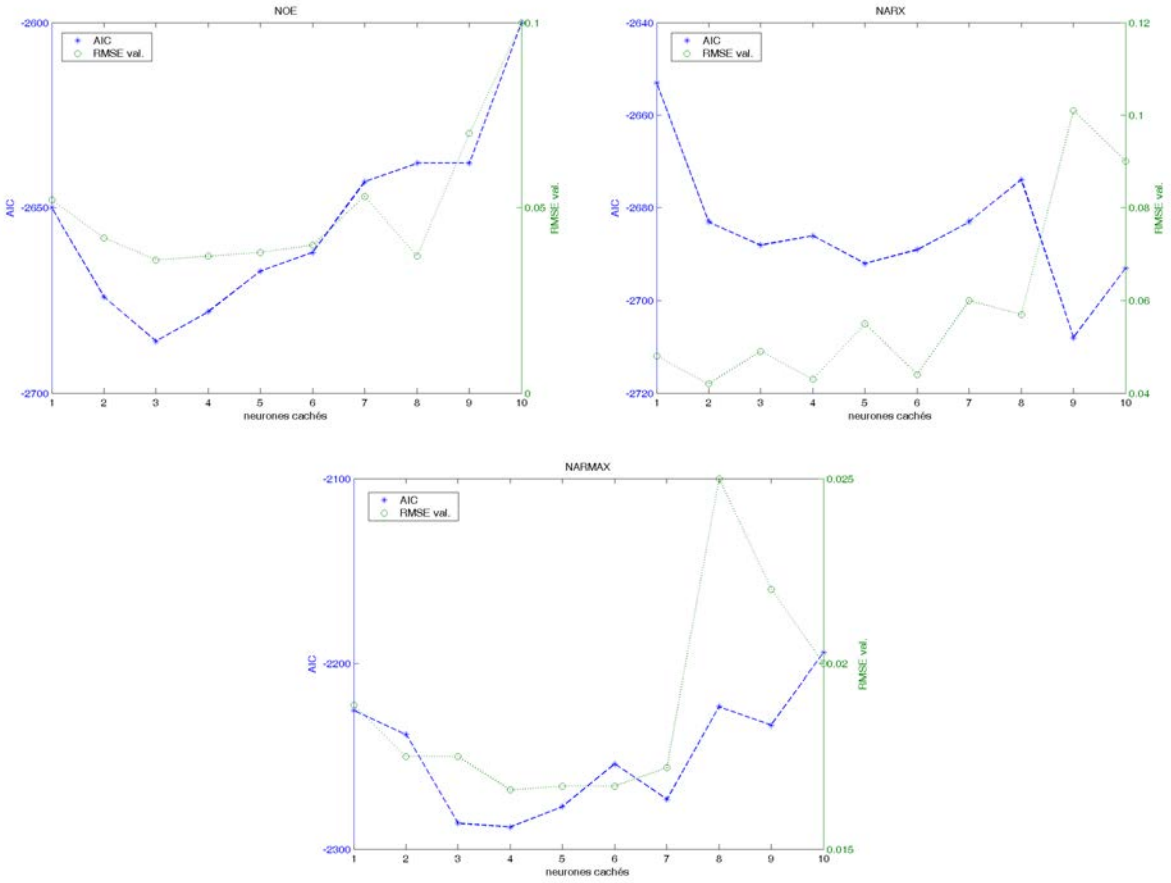

FIG. 2.13 – Performances des structures MLP NOE, MLP NARX et MLP NARMAX en présence de bruit de sortie : Evolution du critère AIC en apprentissage et de la $RMSE$ en validation en fonction du nombre de neurones cachés.

L'évolution des courbes AIC et $RMSE$ du modèle MLP NOE montrent qu'en augmentant progressivement le nombre de neurones cachés, le critère AIC s'améliore puis se dégrade. La plus faible valeur de la $RMSE$ de validation correspond au minimum du critère AIC. Les courbes montrent également que le critère AIC commence à se dégrader quand le nombre de neurones cachés atteint la valeur 3, le modèle MLP NOE présentant la meilleure capacité de généralisation, ce qui est conforme à l'hypothèse d'un bruit de

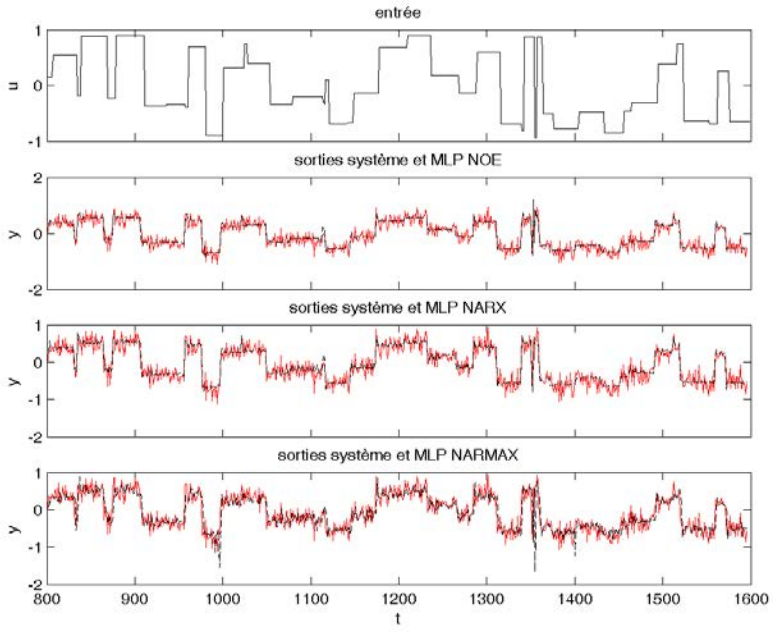

FIG. 2.14 – Identification en présence de bruit de sortie : entrée, sortie bruitée du système (trait plein) et sortie des modèles (trait discontinu) MLP NOE, MLP NARX et MLP NARMAX en validation.

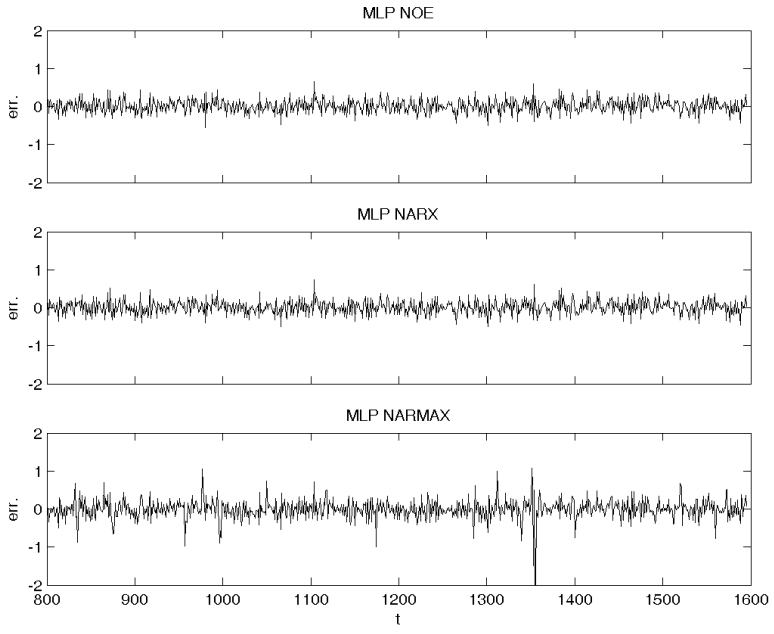

FIG. 2.15 – Influence d'un bruit de sortie additif : résidus des modèles en validation.

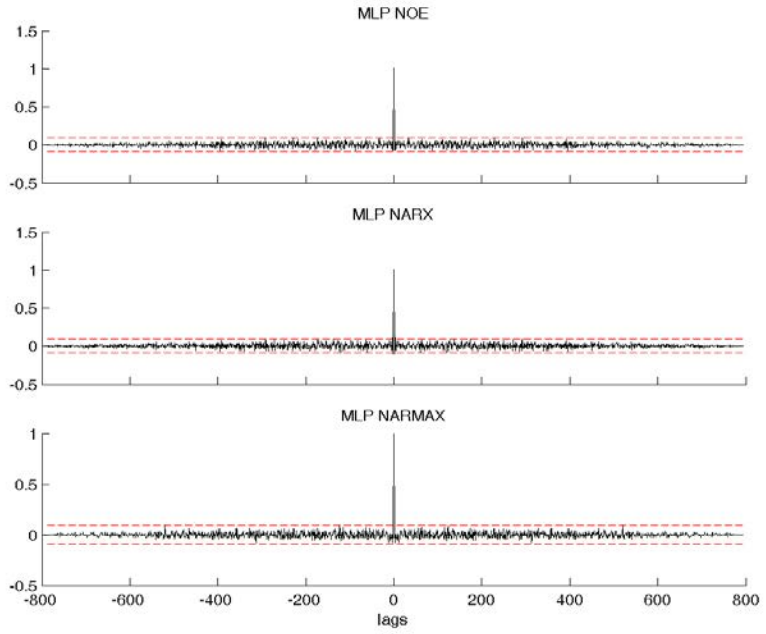

FIG. 2.16 – Fonctions d'autocorrélation des résidus des modèles en validation en présence de bruit de sortie.

sortie. En effet le modèle MLP NOE permet d'observer l'état du système qui correspond ici à la sortie non bruitée, d'où une bonne capacité de généralisation.

L'application aux modèles MLP NOE, MLP NARX et MLP NARMAX de l'algorithme d'identification structurelle présenté sur la figure 2.8 a permis d'obtenir les résultats présentés au tableau 2.2. Le modèle MLP NOE à trois neurones cachés se présente comme la meilleure structure. La figure 2.14 représente l'entrée, la sortie du système et des modèles NOE, NARX et NARMAX en validation. La figure 2.15 représente les résidus obtenus pour chacun des modèles en validation. Les fonctions d'autocorrélation des résidus des

Classe de modèle	Architecture	AIC	*RMSE* app.	*RMSE* valid.
NOE	[5 3 1]	-2667	0.180	0.175
NARX	[5 3 1]	-2692	0.179	0.177
NARMAX	[6 4 1]	-2281	0.229	0.257

TAB. 2.2 – Résultats obtenus pour les modèles MLP NOE, NARX et NARMAX en présence de bruit de sortie additif.

modèles sont représentées sur la figure 2.16.

2.5.3 En présence de bruit d'état

La sortie bruitée $y_{s_{be}}$ est obtenue à partir de l'équation (2.24) par l'expression :

$$\begin{cases} y_s(t+1) = & \dfrac{y_s(t)y_s(t-1)y_s(t-2)u(t-1)\left[y_s(t-2)-1\right]+u(t)}{1+y_s(t-1)^2+y_s(t-2)^3} + e_2(t+1) \\ y_{s_{be}}(t+1) = & y_s(t+1) \end{cases}$$

où le bruit $e_2(t)$ est une réalisation d'une variable aléatoire qui suit une distribution normale $\mathcal{N}(0, 0.17)$. Le rapport signal/bruit est de 7.3 dB. En général, un modèle de structure NARX est plus approprié à la représentation de systèmes dynamiques non linéaires en présence de bruit d'état (voir §1.4).

La figure 2.17 représente l'évolution du critère AIC en apprentissage et de la $RMSE$ en validation (sur des signaux non bruités) en fonction du nombre de neurones cachés pour les structures NARX, NOE et NARMAX identifiées en présence de bruit d'état. Pour le modèle MLP NARX, le critère AIC se dégrade à partir de 3 neurones cachés puis s'améliore, alors que la capacité de généralisation se dégrade au delà de 7 neurones cachés. En effet, l'augmentation du nombre de neurones cachés entraîne la prise en compte du bruit dans l'apprentissage ; l'erreur d'apprentissage tend à diminuer, ce qui améliore le critère AIC, mais le modèle perd sa capacité de généralisation. Le modèle MLP NOE possède une bonne capacité de généralisation qui tend à se dégrader avec l'augmentation du nombre de neurones cachés. Le modèle MLP NARMAX, avec des valeurs de $RMSE$ beaucoup plus élevées, convient moins à la représentation du système.

Les résultats obtenus de l'utilisation de la procédure d'identification structurelle présentée sur la figure 2.8 sont donnés au tableau 2.3. Le modèle MLP NARX se présente comme la structure la plus parcimonieuse (plus faible valeur du critère AIC) et possède la meilleure capacité de généralisation.

La figure 2.18 représente l'entrée, la sortie du système et des modèles MLP NARX, MLP NOE et MLP NARMAX en validation. La figure 2.19 représente les résidus obtenus par les modèles en validation.

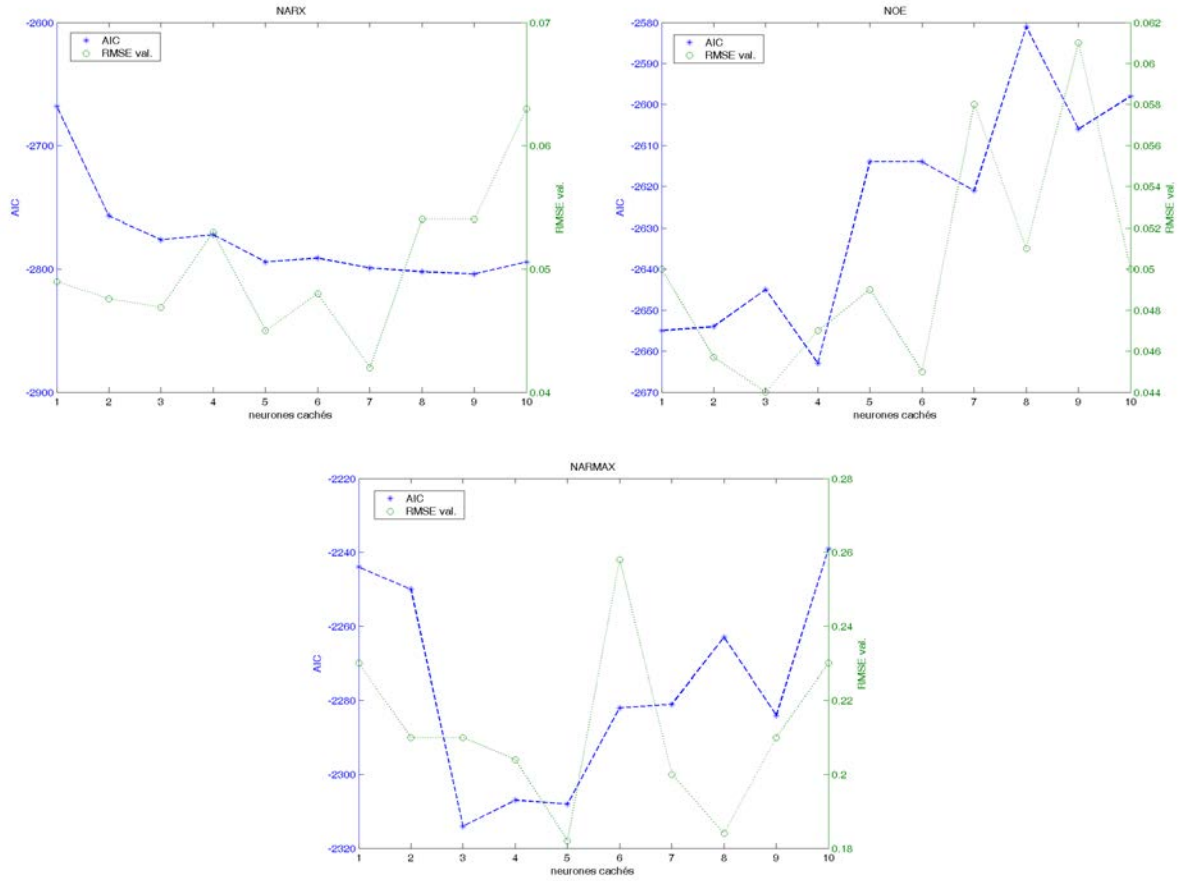

FIG. 2.17 – Performances des structures MLP NARX, MLP NOE et MLP NARMAX en présence de bruit d'état : Evolution du critère AIC en apprentissage et de la $RMSE$ en validation en fonction du nombre de neurones cachés.

Classe de modèle	Architecture	AIC	$RMSE$ app.	$RMSE$ valid.
NARX	[5 3 1]	-2774	0.170	0.176
NOE	[5 1 1]	-2656	0.186	0.184
NARMAX	[6 3 1]	-2316	0.226	0.278

TAB. 2.3 – Résultats obtenus pour les modèles MLP NARX, NOE et NARMAX en présence de bruit d'état.

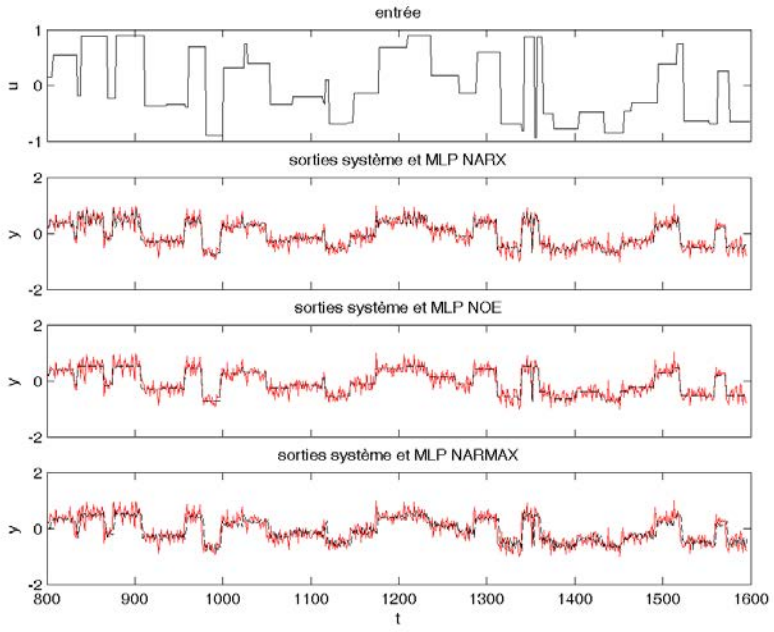

FIG. 2.18 – Identification en présence de bruit d'état : entrée, sortie bruitée du système (trait plein) et sorties des modèles (trait discontinu) MLP NARX, MLP NOE et MLP NARMAX en validation.

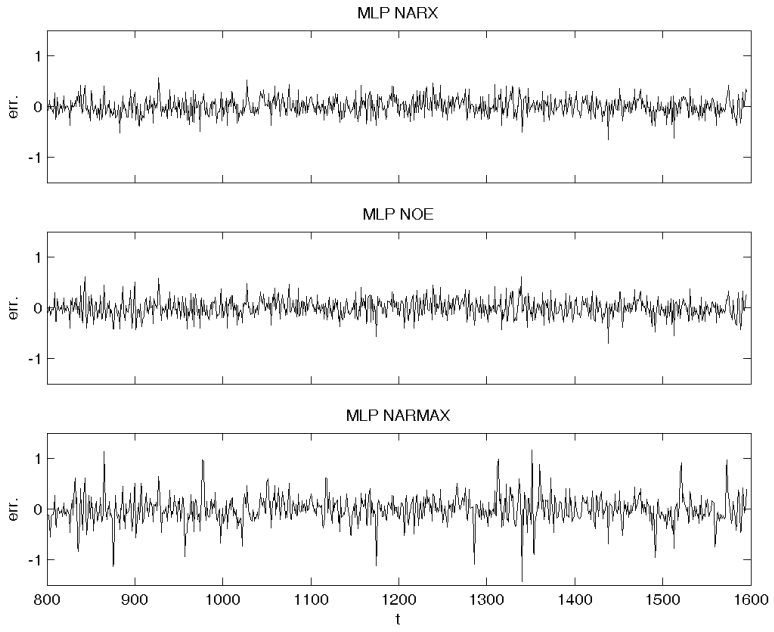

FIG. 2.19 – Influence d'un bruit d'état : résidus des modèles en validation.

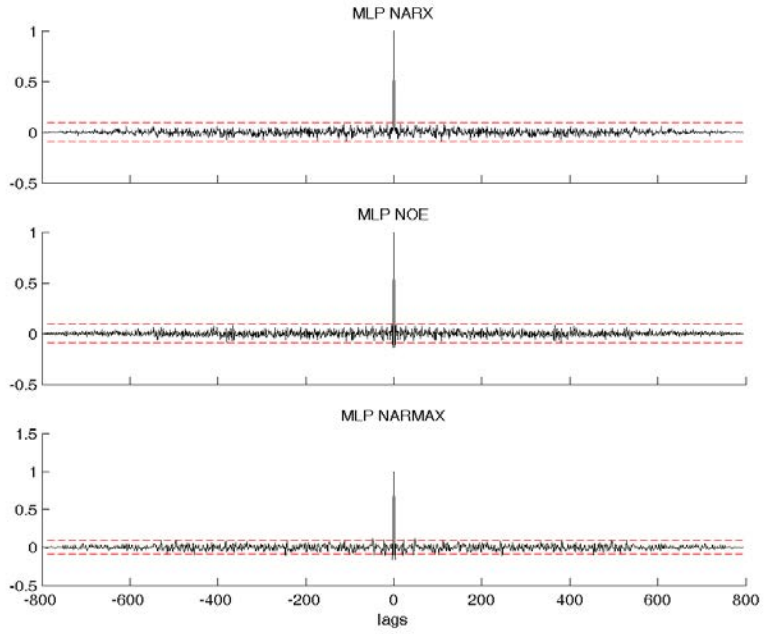

FIG. 2.20 – Fonctions d'autocorrélation des résidus des modèles en validation en présence de bruit d'état.

2.5.4 En présence de bruit de sortie et de bruit d'état

Le système avec bruit de sortie et bruit d'état $y_{s_{bse}}$ est généré par l'équation :

$$\begin{cases} y_s(t+1) = & \dfrac{y_s(t)y_s(t-1)y_s(t-2)u(t-1)\left[y_s(t-2)-1\right]+u(t)}{1+y_s(t-1)^2+y_s(t-2)^3} + e_2(t+1) \\ y_{s_{bse}}(t+1) = & y_s(t+1) + e_3(t+1) \end{cases}$$

où :

$e_3(t)$ est une réalisation d'une variable aléatoire qui suit une loi de distribution normale $\mathcal{N}(0, 0.2)$;

$e_2(t)$ est une réalisation d'une variable aléatoire qui suit une distribution normale $\mathcal{N}(0, 0.17)$.

Le rapport signal/bruit est de 4.3 dB. Les performances des modèles NARMAX, NOE et NARX sont comparées. D'après l'étude présentée au paragraphe §1.4, un modèle NARMAX est en général plus adapté à représenter un tel système.

La figure 2.21 représente l'évolution du critère AIC en apprentissage et de la $RMSE$ en validation en fonction du nombre de neurones cachés pour les structures MLP NARMAX, MLP NARX et MLP NOE identifiées en présence de bruit de sortie et de bruit

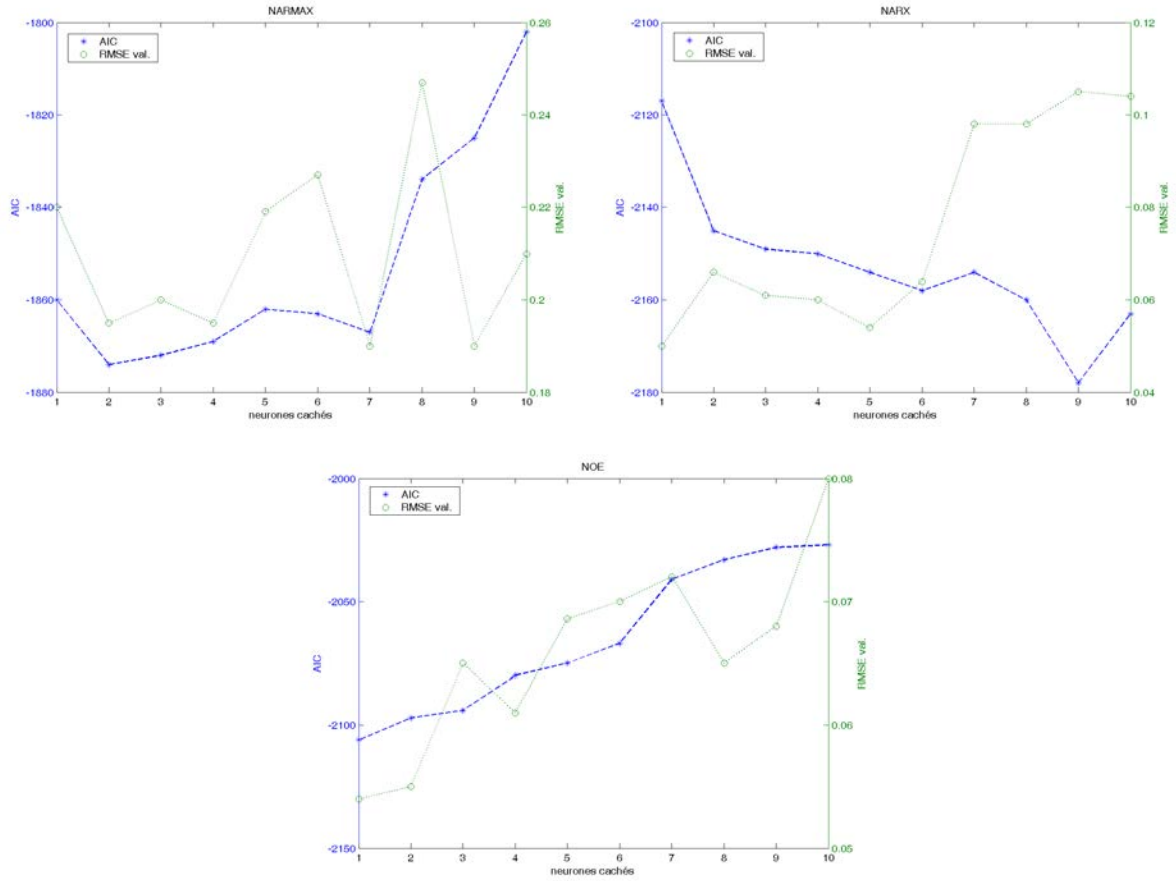

FIG. 2.21 – Performances des structures MLP NOE, MLP NARX et MLP NARMAX en présence de bruit de sortie et de bruit d'état : évolution du critère AIC en apprentissage et de la $RMSE$ en validation en fonction du nombre de neurones cachés.

d'état. Les résultats montrent que l'erreur de généralisation des modèles MLP NOE et MLP NARX croît avec l'augmentation du nombre de neurones cachés, ce qui n'est pas le cas du modèle MLP NARMAX.

L'application aux modèles MLP NARMAX, MLP NARX et MLP NOE de l'algorithme d'identification structurelle présenté sur la figure 2.8 a permis d'obtenir les résultats présentés au tableau 2.4. L'algorithme d'identification permet de déterminer la meilleure structure MLP NARMAX, mais les résultats montrent que les structures MLP NOE et MLP NARX possèdent des capacités de généralisation légèrement meilleures.

La figure 2.22 représente l'entrée, la sortie du système et celle des modèles MLP NARMAX, MLP NARX et MLP NOE en validation. La figure 2.23 représente les résidus obtenus par les modèles en validation et la figure 2.24, les fonctions d'autocorrélation des

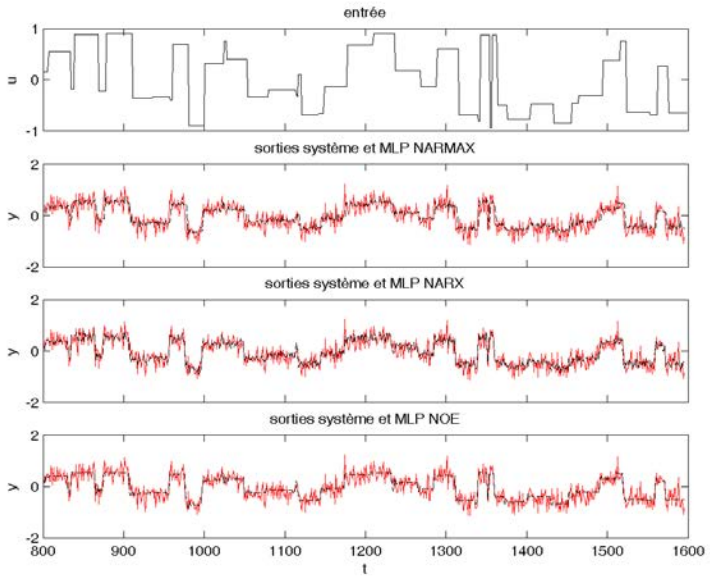

FIG. 2.22 – Identification en présence de bruit d'état et de bruit de sortie : entrée, sortie bruitée du système (trait plein) et sorties des modèles (trait discontinu) MLP NARMAX, MLP NARX et MLP NOE en validation.

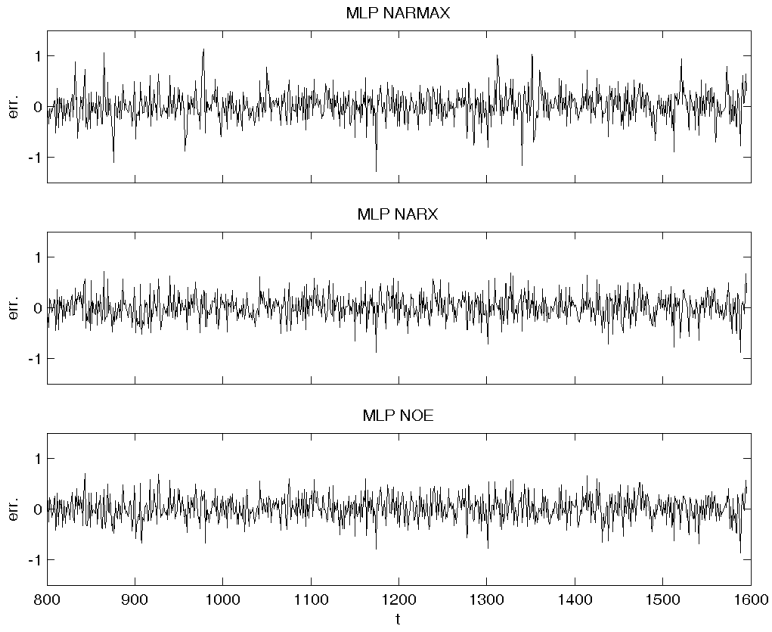

FIG. 2.23 – Influence d'un bruit de sortie et d'un bruit d'état : résidus des modèles en validation.

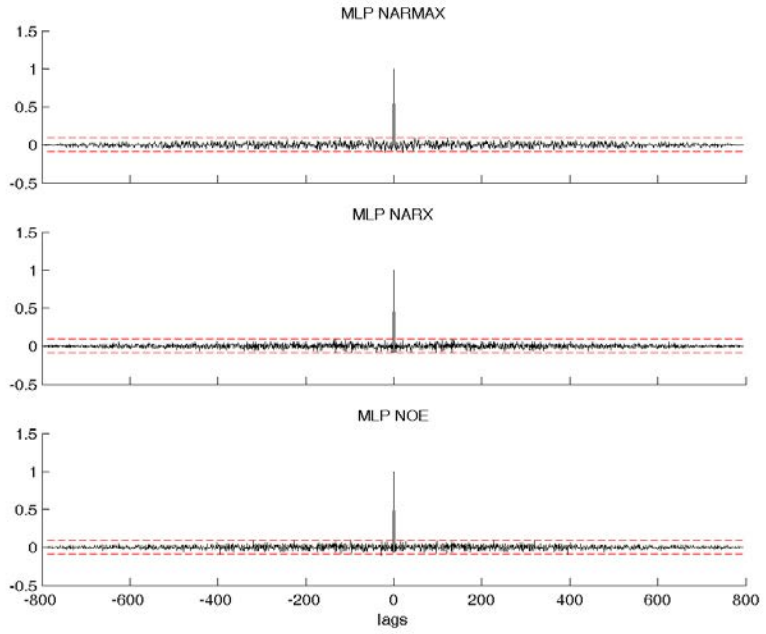

FIG. 2.24 – Fonctions d'autocorrélation des résidus des modèles en validation en présence de bruit de sortie et de bruit d'état.

résidus.

Classe de modèle	Architecture	AIC	*RMSE* app.	*RMSE* valid.
NARMAX	[6 2 1]	-1874	0.301	0.300
NARX	[5 6 1]	-2159	0.244	0.253
NOE	[5 1 1]	-2106	0.264	0.250

TAB. 2.4 – Résultats obtenus pour les modèles MLP NARMAX, NARX et NOE en présence de bruit de sortie et de bruit d'état.

2.6 Conclusion

Dans ce chapitre, nous avons étudié différentes structures de modèles non linéaires et montré qu'il est possible de les représenter avec un réseau de neurones de type MLP : un MLP non récurrent permet d'implanter un modèle NARX, tandis qu'un MLP récurrent permet d'implanter les modèles NARMAX et NOE. L'importance du choix de la structure

de modèle non linéaire a été illustré à travers l'étude d'un système en présence de bruit de sortie, de bruit d'état puis de bruit de sortie et de bruit d'état.

Si les connaissances *a priori* n'indiquent pas la manière dont le bruit intervient sur le système (bruit d'état, bruit de sortie ou bruit d'état et de sortie), il est nécessaire de tester les différentes structures de modèles non linéaires. La sélection d'une structure de modèle ne doit être faite que sur la base de la capacité de généralisation puisque certaines structures peuvent avoir des critères de performances meilleurs que d'autre en apprentissage, et disposer d'une capacité de généralisation médiocre. La méthode d'identification structurelle proposée dans ce chapitre permet de déterminer le nombre de neurones cachés d'un réseau MLP sur la base de l'analyse du critère AIC évalué pendant l'apprentissage.

Les algorithmes d'apprentissage utilisés pour l'identification des modèles MLP bien que performants, souffrent de certaines difficultés dont un temps de calcul important (surtout pour le MLP récurrent qui nécessite plusieurs copies du réseau non récurrent correspondant), le risque d'obtenir un minimum local pour le critère à minimiser, des problèmes de vitesse de convergence et de stabilité liés au choix du pas d'apprentissage. Ces difficultés sont inhérentes à l'optimisation par la technique de descente du gradient.

Dans le chapitre 3, nous présentons l'approche multimodèle permettant de contourner les difficultés précédemment mentionnées et d'intégrer des modèles MLP dans une architecture hétérogène.

Chapitre 3

Identification de Systèmes Dynamiques Non Linéaires par multimodèles

Sommaire

3.1 Introduction

L'approche multimodèle est une technique relativement récente utilisée pour la modélisation de systèmes dynamiques non linéaires. Elle connaît un regain d'intérêt ces dernières années [9, 10, 29, 73, 74, 75], notamment dans des applications telles que la prédiction de séries temporelles, la commande ou la simulation. La formalisation mathématique des représentations multimodèles date des travaux de Johansen et Foss en 1992 [16], qui réunissaient en un seul concept plusieurs techniques de modélisation connues sous différentes appellations, telles que le modèle de Takagi-Sugeno (T-S) [31] ou les systèmes multi-experts.

Le multimodèle doit sa popularité à la simplicité de 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. La sortie du modèle est alors obtenue par combinaison des sorties des modèles locaux. L'approche permet d'appréhender le comportement local du système dans chaque zone de fonctionnement et évite la mise en place d'un modèle unique qui peut s'avérer très complexe. Dans la plupart des travaux, des modèles locaux de structure affine sont utilisés en raison de leur simplicité. Cependant, si le système comporte de fortes non linéarités, le nombre de modèles locaux peut être très important, ce qui augmente la complexité du modèle.

Nous proposons de remédier à ce problème par l'implantation de modèles locaux de structure polynômiale qui permettent de mieux appréhender le comportant local du système dans chaque zone de fonctionnement. Les implantations des architectures multimodèles concernent principalement des structures non récurrentes. Nous proposons aussi dans ce travail des architectures de multimodèles récurrentes et présentons un algorithme d'apprentissage de ces multimodèles. Cette contribution a fait l'objet d'une publication dans une revue internationale [8]. Nous montrons que ces architectures ont beaucoup d'avantages par rapport aux réseaux de neurones présentés au chapitre précédent. Nous proposons enfin une architecture multimodèle avec des modèles locaux hétérogènes, une généralisation de l'implantation conventionnelle dans laquelle les modèles locaux ont la même structure, en général affine. L'utilisation des modèles locaux de type MLP a fait également l'objet d'un publication dans une revue internationale [10].

La première partie du chapitre présente l'approche multimodèle, les architectures de multimodèles non récurrentes et récurrentes à modèles locaux polynômiaux. Les deuxième

et troisième parties du chapitre traitent de l'identification structurelle et paramétrique d'un multimodèle. Une étude des multimodèles à modèles locaux hétérogènes est présentée dans la dernière partie.

3.2 Présentation de l'approche multimodèle

Un multimodèle est une représentation d'un système, composée d'un ensemble de modèles locaux à validité limitée dans une zone bien définie de l'espace de fonctionnement. La validité locale d'un modèle pour une observation donnée est spécifiée par un degré d'activation du modèle qui indique en même temps le degré d'appartenance de l'observation (ou de la variable d'indexation correspondante) à la zone de validité du modèle. Les zones de validité sont des sous-espaces obtenus par subdivision de l'espace de fonctionnement global du système suivant des variables appelées variables d'indexation. Le degré d'activation d'un modèle local prend une valeur suffisamment importante (proche de 1) si la variable d'indexation correspondant à l'observation est proche du « centre » de la zone de validité du modèle, et tend progressivement vers zéro au fur et à mesure que la variable d'indexation s'en éloigne. La figure 3.1 schématise un exemple d'architecture multimodèle (cette représentation correspond à celle d'un modèle NFIR).

Avec une représentation multimodèle, le modèle général d'un système dynamique non linéaire peut se mettre sous la forme :

$$\hat{y}(t+h) = \sum_{i=1}^{M} \omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big)\ f_i\Big(\underline{\varphi}(t), \underline{\theta}_i\Big) \tag{3.1}$$

où :

M est le nombre de modèles locaux ;

$\omega_i(\cdot)$ est le degré d'activation du modèle local $f_i(\cdot)$, avec les conditions :

$$\omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big) \in [0,\ 1]\ , \quad \sum_{i=1}^{M} \omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big) = 1 \quad \forall t$$

$\underline{\xi}(t)$ est un vecteur contenant les variables d'indexation qui déterminent l'appartenance de l'observation $\underline{\varphi}(t)$ à une zone de fonctionnement. Les variables d'indexation peuvent être les entrées du système, les entrées et la sortie du système, ou les composantes du vecteur de régression ;

$\underline{\beta}_i$ est un vecteur de paramètres caractérisant la zone de validité du modèle local $f_i(\cdot)$;

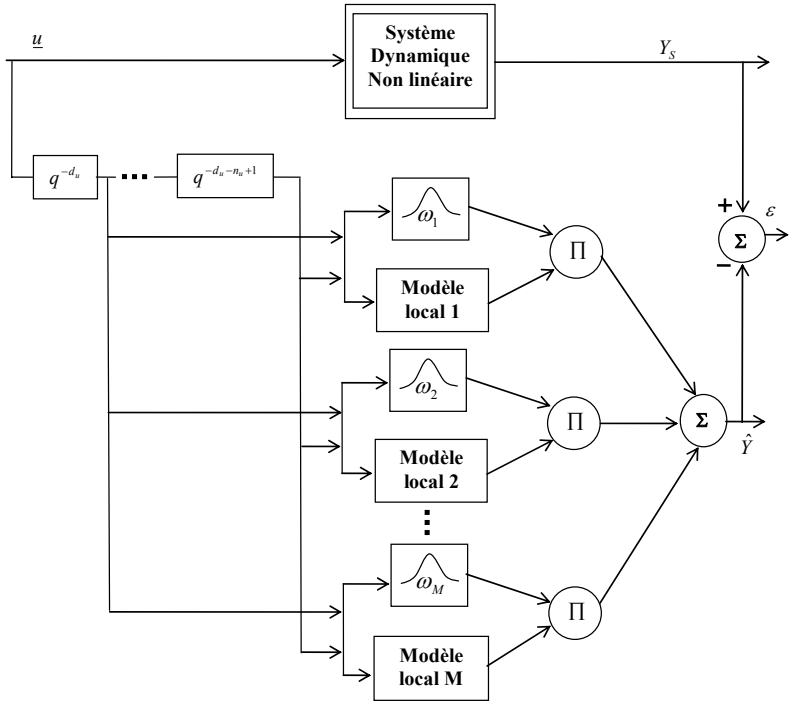

FIG. 3.1 – Exemple d'architecture multimodèle (NFIR). Un ensemble d'opérateurs de décalage temporel est utilisé pour la construction du vecteur de régression.

$\underline{\theta}_i$ est un vecteur de paramètres caractérisant le modèle local $f_i(\cdot)$;
$f_i\Big(\underline{\varphi}(t), \underline{\theta}_i\Big) = \hat{y}_i(t+h)$ est la sortie prédite par le modèle local i.

La détermination des paramètres des modèles locaux se fait par minimisation d'un critère de performance. Les degrés d'activation des modèles locaux peuvent être déterminés de différentes façons selon la méthode utilisée pour la décomposition de l'espace de fonctionnement (voir §3.3). Il existe des méthodes de décomposition déterministes pour lesquelles les frontières entre les zones de validité des modèles locaux sont nettes, et des méthodes de décomposition floues pour lesquelles ces frontières ne sont pas définies. Dans tous les cas, les degrés d'activation des modèles locaux peuvent permettre un recouvrement entre les zones, une observation pouvant appartenir à plusieurs zones, à des degrés différents. Dans le cas des méthodes de décomposition déterministes, le degré d'activation d'un modèle local $f_i(\cdot)$ est calculé par :

$$\omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big) = \frac{\rho_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big)}{\sum_{k=1}^{M} \rho_k\Big(\underline{\xi}(t), \underline{\beta}_i\Big)} \tag{3.2}$$

où $\rho_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big)$ est une fonction d'appartenance spécifiant l'appartenance de $\underline{\xi}(t)$ à la zone du modèle local $f_i(\cdot)$ caractérisée par le vecteur $\underline{\beta}_i$. Il existe différents types de fonctions d'appartenance : fonctions d'allure gaussienne, fonctions sigmoïdes, fonctions trapèzoïdales, etc. Dans ce travail, nous nous sommes focalisés sur des fonctions d'appartenance d'allure gaussienne. Une telle architecture constitue une généralisation de l'architecture RBF présentée sur la figure 1.4. En effet, un réseau RBF est un multimodèle où les modèles locaux sont des constantes (poids ω_i dans (1.11)) et les fonctions d'appartenance des gaussiennes (les valeurs de $g\left(\|\underline{x} - \underline{C}_i\|_{\Sigma_i}\right)$ dans (1.11) correspondent au degrés d'activation non normalisés des modèles locaux).

En se basant sur une décomposition en série de Taylor, Johansen et Foss ont donné les conditions suivant lesquelles l'approximation de toute fonction continue non linéaire multivariable peut être réalisée avec une précision arbitraire et un nombre fini de modèles locaux M [44]. Ces conditions sont :

- les mesures expérimentales sont bornées,
- le vrai modèle du système est $(p+1)$ fois dérivable et sa dérivée d'ordre $(p+1)$ est bornée ($p \in \mathbb{N}^*$),
- les fonctions de validité ont un support localisé et leurs centres couvrent de façon dense l'espace de régression,
- les modèles locaux f_i sont les p premiers termes du développement en série de Taylor du vrai modèle autour des points de fonctionnement.

Exemple

Considérons le problème d'approximation de la fonction non linéaire statique définie par :

$$y = 1 + exp(-t^2)sin(\pi t)$$

où $t \in [-2,\ 2]$. Dans cet exemple, $\underline{\varphi}(t) = \underline{\xi}(t) = t$.

En décomposant arbitrairement l'interval de variation de t en 3 partitions :

$$P_1 = [-2,\ -0.4] \quad P_2 = [-0.4,\ 0.4] \quad P_3 = [0.4,\ 2]$$

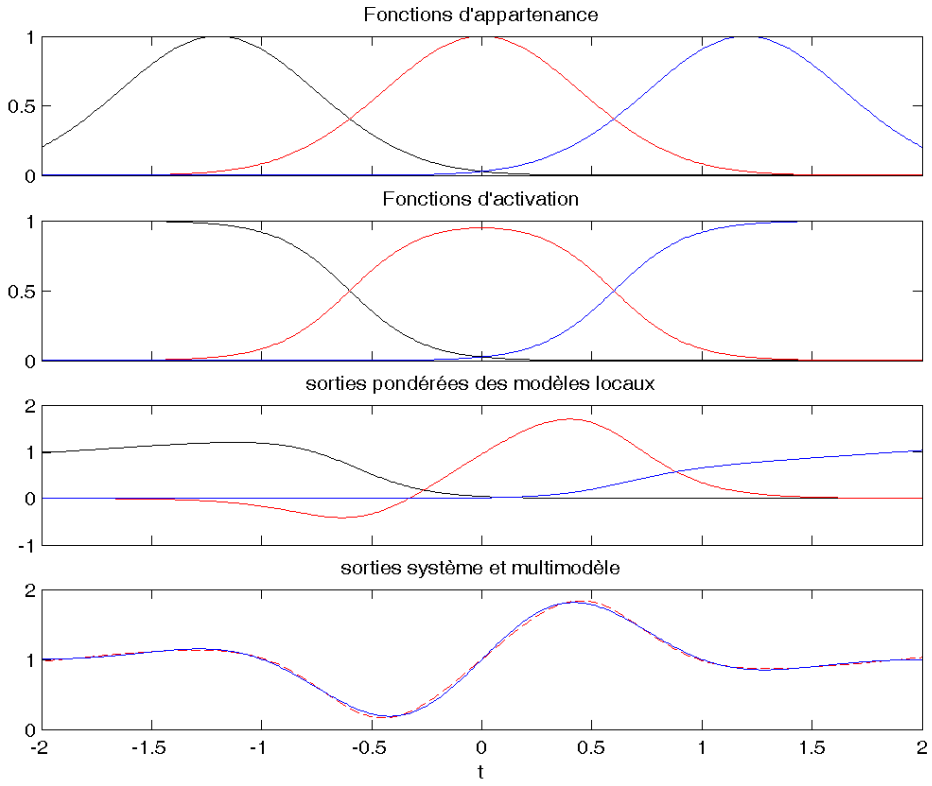

FIG. 3.2 – Principe de la modélisation par multimodèle. La sortie du multimodèle (trait discontinu) est obtenu par la somme pondérée des sorties de 3 modèles locaux.

et en utilisant les modèles locaux et les fonctions d'appartenance suivants :

$$\begin{cases} y_1 = 0.32t + 1.61 & \rho_1 = \exp\left(-\dfrac{(t+1.2)^2}{2 \times 0.447^2}\right) \\ y_2 = 3.03t + 1 & \rho_2 = \exp\left(-\dfrac{t^2}{2 \times 0.447^2}\right) \\ y_3 = 0.32t + 0.39 & \rho_3 = \exp\left(-\dfrac{(t-1.2)^2}{2 \times 0.447^2}\right) \end{cases}$$

la sortie y peut être estimée avec une erreur faible ($RMSE = 0.02$) par :

$$\hat{y} = \sum_{i=1}^{3} \omega_i \, y_i$$

où les ω_i sont calculés par (3.2). La figure 3.2 montre les degrés d'activation des modèles locaux, les sorties des modèles locaux ainsi que la sortie du multimodèle et celle du système.

Cet exemple académique simple permet de voir les différents problèmes que pose la modélisation par approche multimodèle : Quel est le nombre de modèle locaux permettant de représenter le système ? Quelle doit être la structure des modèles locaux ? Comment définir les zones de validité des différents modèles locaux ? Comment déterminer les paramètres des modèles locaux et de leur zones de validité ?

3.2.1 Représentation de systèmes dynamiques non linéaires par une architecture multimodèle

Avec une architecture multimodèle, il est possible de représenter les systèmes dynamiques non linéaires étudiés au paragraphe 1.4. L'architecture du multimodèle dépend alors de la classe du modèle à identifier et peut être non récurrente (modèles NFIR ou NARX) ou récurrente (modèles NOE, NARMAX).

La figure 3.1 correspond à l'implantation multimodèle d'un modèle NFIR. Le vecteur de régression d'un tel modèle ne comporte que les entrées passées du système.

La figure 3.3 représente l'implantation d'un modèle NARX avec une structure multimodèle. La fonction vectorielle $g(\cdot)$ utilisée dans schéma permet de former le vecteur de régression en utilisant l'opérateur de décalage temporel q^{-1}. Le vecteur de régression du modèle NARX comporte les entrées passées et les sorties mesurées passées.

L'implantation multimodèle d'un modèle NOE est représentée sur la figure 3.4. Le vecteur de régression du modèle NOE est composé des entrées passées et les sorties simulées passées.

Pour un modèle NARMAX, le vecteur de régression est constitué des entrées passées, des sorties mesurées passées et des erreurs de prédiction. L'implantation multimodèle d'un modèles NARMAX est représentée sur la figure 3.5.

Pour les modèles non récurrents, il est possible de disposer de l'ensemble des données nécessaires à l'identification (base de données expérimentales). L'estimation paramétrique se fait alors suivant une optimisation linéaire ou non linéaire en fonction de la structure des modèles locaux.

L'implantation de multimodèles récurrents et l'établissement des techniques d'apprentissage correspondantes constituent une des contributions de ce travail de thèse. Pour un modèle récurrent, certaines données nécessaires à l'estimation paramétrique sont issues

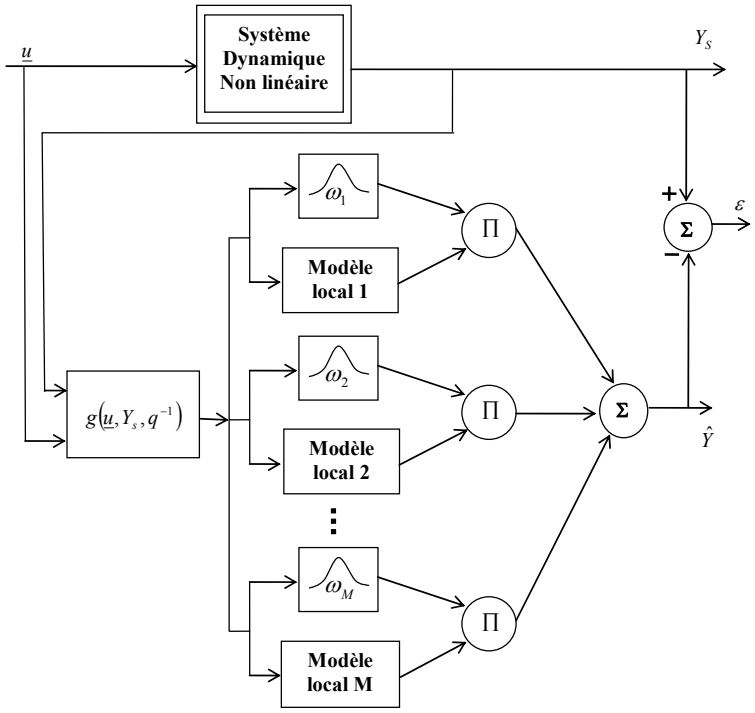

FIG. 3.3 – Multimodèle NARX.

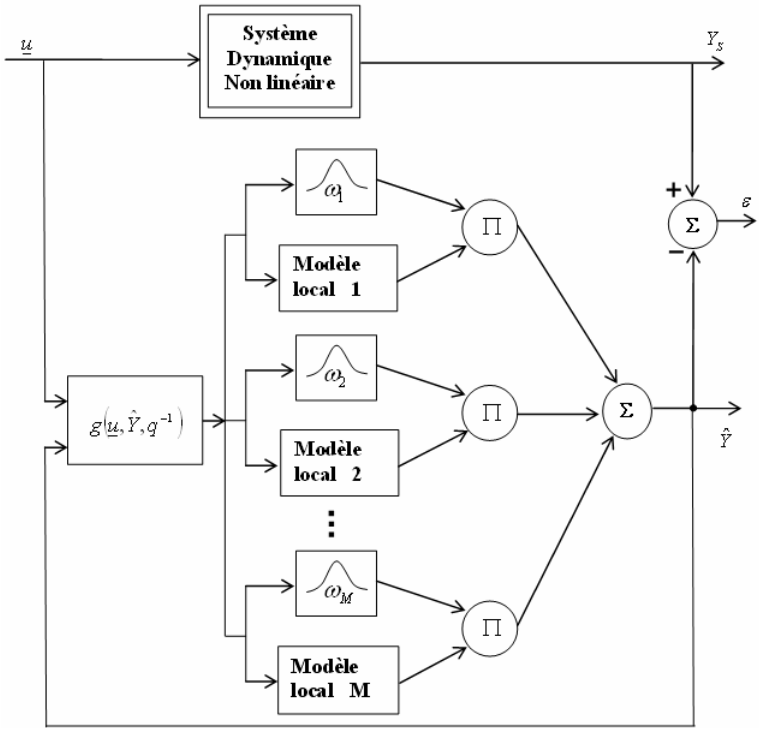

FIG. 3.4 – Multimodèle NOE.

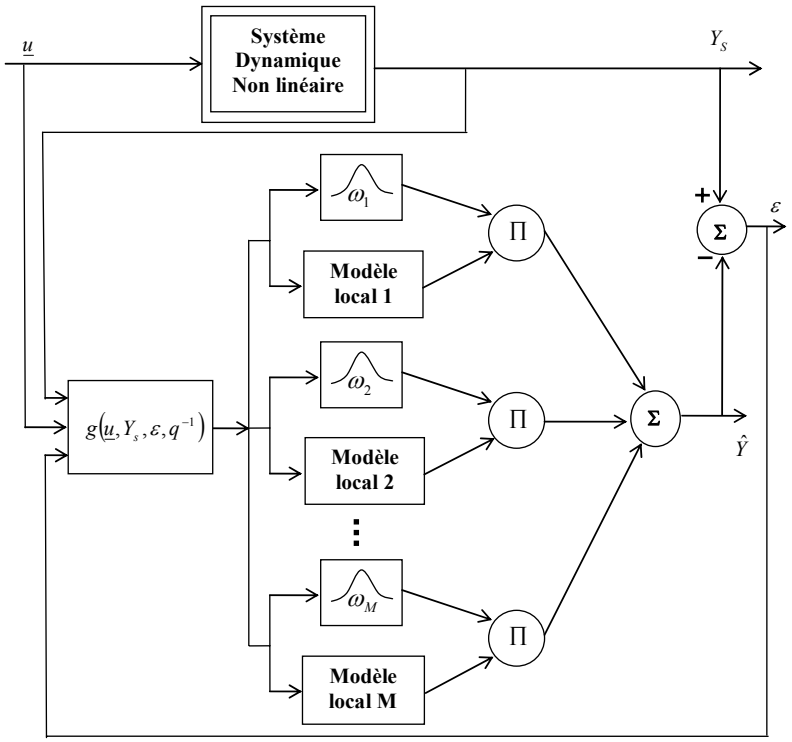

FIG. 3.5 – Multimodèle NARMAX.

des sorties du multimodèle et ne sont obtenues qu'au fur et mesure que se déroule le processus d'identification. Il est alors nécessaire d'utiliser une méthode d'estimation récursive. L'estimation paramétrique des multimodèles récurrents est présentée au paragraphe §3.4.

3.3 Identification structurelle d'un multimodèle

L'identification structurelle d'un multimodèle consiste en la détermination de la structure des modèles locaux et la définition des zones de fonctionnement (ou zones de validité) de chaque modèle local.

Les modèles locaux peuvent être de différentes structures. On utilise en général des modèles locaux de structure simple, comme par exemple des modèles linéaires. Il est aussi possible d'utiliser des modèles locaux non linéaires ce qui permet de mieux appréhender les non linéarités locales du système et de réduire le nombre de modèles locaux.

La spécification des zones de fonctionnement consiste à définir les sous-espaces d'activation de chaque modèle local. Une observation peut appartenir à plusieurs zones de

fonctionnement mais à des degrés divers. Le degré d'activation d'un modèle local pour une observation donnée dépend du degré d'appartenance de l'observation au sous-espace où est défini le modèle local.

3.3.1 Choix des modèles locaux

La spécification de la structure de chaque modèle local consiste à déterminer la relation qui explique le modèle dans son domaine de validité. En général tous les modèles locaux ont la même structure, mais il est tout a fait possible d'utiliser une structure hétérogène de modèles locaux comme nous le proposons au paragraphe §3.5. Nous n'abordons dans ce travail que les modèles locaux de type entrées-sortie avec une structure polynômiale ou neuronale. Les raisons de ce choix sont diverses. L'intérêt d'utiliser des modèles locaux de structure polynômiale est que ces modèles sont linéaires par rapport aux paramètres. Cette structure permet également d'introduire des non linéarités dans les modèles locaux lorsque le degré du polynôme est supérieur à 1, ce qui permet de réduire leur nombre. Quant à la structure neuronale, elle se justifie par la propriété d'approximation du MLP à une couche cachée (voir §2.4). La combinaison de ces deux structures dans un multimodèle hétérogène est abordée au paragraphe §3.5.

3.3.1.1 Modèles locaux affines

Comme suggéré dans [44], chaque modèle local peut être défini comme les p premiers termes du développement en série de Taylor du vrai modèle (inconnu) $F_s(\cdot)$ autour d'un point de la zone de validité du modèle local. Les modèles locaux affines ($p = 1$) sont les plus utilisés grace à leur facilité d'implantation. L'expression générale des modèles affines est de la forme :

$$f_i\Big(\underline{\varphi}(t), \underline{\theta}_i\Big) = \Big[\underline{\varphi}(t)^T \ 1\Big] \ \underline{\theta}_i \tag{3.3}$$

L'inconvénient avec cette structure est que le nombre de modèles locaux peut être très élevé si le système à modéliser est complexe. Par ailleurs, une telle structure ne convient pas à l'utilisation de certaines méthodes de décomposition de l'espace de fonctionnement des systèmes (voir §3.5.5).

3.3.1.2 Modèles locaux polynômiaux

Pour parer à l'inconvénient que présentent les modèles locaux affines, une structure polynômiale est proposée. Elle consiste à effectuer un développement en série de Taylor du vrai modèle à un ordre p supérieur à 1. Les modèles locaux sont obtenus en effectuant la transformation non linéaire suivante :

$$\underline{\varphi}_p(t) = g_p\Big(\underline{\varphi}(t)\Big)$$

où $g_p(\cdot)$ est la transformation non linéaire du vecteur $\underline{\varphi}(t)$ en un vecteur $\underline{\varphi}_p(t)$ composé de l'ensemble des combinaisons possibles des produits (p produits aux maximum) des éléments de $\underline{\varphi}(t)$ (distincts ou non). Nous proposons ici une procédure simple pour construire le vecteur $\underline{\varphi}_p(t)$.

Soit

$$\underline{\varphi}(t) = [\varphi_1 \quad \varphi_2 \quad \cdots \quad \varphi_{n_\varphi}]^T \tag{3.4}$$

où n_φ représente la dimension du vecteur $\varphi(t)$.

On peut construire les vecteurs suivants :

$$\begin{aligned}
V_{1,1} &= [\varphi_1 \quad \varphi_2 \quad \cdots \quad \varphi_{n_\varphi}] \\
V_{1,2} &= [\varphi_2 \quad \cdots \quad \varphi_{n_\varphi}] \\
&\cdots \\
V_{1,n_\varphi} &= [\varphi_{n_\varphi}] \\
V_{2,1} &= [\varphi_1 V_{1,1} \quad \varphi_2\ V_{1,2} \quad \cdots \quad \varphi_{n_\varphi} V_{1,n_\varphi}] \\
V_{2,2} &= [\varphi_2\ V_{1,2} \quad \cdots \quad \varphi_{n_\varphi} V_{1,n_\varphi}] \\
&\cdots \\
V_{2,n_\varphi} &= [\varphi_{n_\varphi} V_{1,n_\varphi}] \\
&\cdots \\
V_{p-1,1} &= [\varphi_1 V_{p-2,1} \quad \varphi_2\ V_{p-2,2} \quad \cdots \quad \varphi_{n_\varphi} V_{p-2,n_\varphi}] \\
V_{p-1,2} &= [\varphi_2\ V_{p-2,2} \quad \cdots \quad \varphi_{n_\varphi} V_{p-2,n_\varphi}] \\
&\cdots \\
V_{p-1,n_\varphi} &= [\varphi_{n_\varphi} V_{p-2,n_\varphi}] \\
V_{p,1} &= [\varphi_1 V_{p-1,1} \quad \varphi_2\ V_{p-1,2} \quad \cdots \quad \varphi_{n_\varphi} V_{p-1,n_\varphi}]
\end{aligned}$$

$V_{1,1}$ contient les éléments de $\underline{\varphi}(t)$;
$V_{2,1}$ contient tous les produits des éléments de $\underline{\varphi}(t)$ 2 à 2 ;

$V_{p,1}$ contient tous les produits des éléments de $\underline{\varphi}(t)$ p à p ;
On obtient alors pour le vecteur $\underline{\varphi}_p(t)$ la relation :

$$\underline{\varphi}_p(t) = [V_{1,1} \quad V_{2,1} \quad \cdots \quad V_{p,1}]^T \tag{3.5}$$

Par exemple si $\underline{\varphi}(t) = [\varphi_1 \quad \varphi_2 \quad \varphi_3]^T$ et $p = 2$, on obtient :

$$\begin{aligned} V_{1,1} &= [\varphi_1 \quad \varphi_2 \quad \varphi_3] \\ V_{2,1} &= [\varphi_1^2 \quad \varphi_1\,\varphi_2 \quad \varphi_1\,\varphi_3 \quad \varphi_2^2 \quad \varphi_2\,\varphi_3 \quad \varphi_3^2] \end{aligned}$$

et d'après la relation (3.5) :

$$\underline{\varphi}_2(t) = [\varphi_1 \quad \varphi_2 \quad \varphi_3 \quad \varphi_1^2 \quad \varphi_1\,\varphi_2 \quad \varphi_1\,\varphi_3 \quad \varphi_2^2 \quad \varphi_2\,\varphi_3 \quad \varphi_3^2]^T$$

On s'aperçoit cependant que le nombre de paramètres n_{φ_p} de $\underline{\varphi}_p(t)$ peut être très important si le vecteur de régression $\underline{\varphi}(t)$ comporte plusieurs variables ou si l'ordre p est élevé.

Pour des raisons de simplicité, nous utilisons la notation V_k à la place de $V_{k,1}$. Les modèles locaux s'expriment alors par la relation :

$$f_i\Big(\underline{\varphi}(t), \underline{\theta}_i\Big) = \sum_{k=1}^{p} V_k \underline{\theta}_{i_k} + \theta_{i0} = \underline{\phi}_{p_e}(t)^T\, \underline{\theta}_i \tag{3.6}$$

où :

$\underline{\theta}_{i_k}$, $k = 1 \cdots p$ est un vecteur de paramètres ;
$\underline{\theta}_{i_0}$ est une constante scalaire ;
$\underline{\phi}_{p_e}(t) = [\underline{\varphi}_p(t)^T \; 1]^T$ est le vecteur de régression augmenté ;
$\underline{\theta}_i = [\theta_{i_0} \quad \underline{\theta}_{i_1}^T \quad \cdots \quad \underline{\theta}_{i_p}^T]^T$ est le vecteur des paramètres du modèle local i.

L'avantage d'une telle représentation est que les modèles locaux sont linéaires par rapport aux paramètres bien que chaque sous-modèle soit non linéaire par rapport aux entrées, ce qui facilite considérablement l'estimation paramétrique (voir §3.4).

L'équation (3.1) s'écrit plus simplement :

$$\hat{y}(t+h) = \sum_{i=1}^{M} \underline{\Phi}_i(t)^T\, \underline{\theta}_i = \underline{\Phi}(t)^T\, \underline{\theta} \tag{3.7}$$

où :

$\underline{\Phi}_i(t) = \omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big)\underline{\phi}_{p_e}(t)$;

$\underline{\Phi}(t) = \left[\omega_1\left(\underline{\xi}(t), \underline{\beta}_1\right)\underline{\phi}_{p_e}(t)^T \cdots \omega_i\left(\underline{\xi}(t), \underline{\beta}_i\right)\underline{\phi}_{p_e}(t)^T \cdots \omega_M\left(\underline{\xi}(t), \underline{\beta}_M\right)\underline{\phi}_{p_e}(t)^T\right]^T$ est le vecteur de régression global pondéré ;
$\underline{\theta} = \left[\underline{\theta}_1^T \ldots \underline{\theta}_i^T \ldots \underline{\theta}_M^T\right]^T$ représente la concaténation des vecteurs de paramètres des modèles locaux.

Certaines méthodes de décomposition de l'espace de fonctionnement du système (voir le partitionnement par classification floue au paragraphe §3.3.2.3) permettent de déterminer les paramètres $\underline{\beta}_i$ de chaque zone sans optimisation des paramètres $\underline{\theta}_i$.

3.3.1.3 Modèles locaux neuronaux de type MLP

Les MLP à une couche cachée ont la propriété d'approximateurs universels parcimonieux [11]. Dans des applications où l'aspect représentation locale d'un système dans une zone de fonctionnement spécifique est primordial par rapport à la représentation globale dans tout l'espace de fonctionnement, des modèles locaux neuronaux paraissent les plus convenables du fait de leur propriété d'approximation. Une telle architecture correspond aux systèmes multi-experts, où chaque expert est un réseau de neurones spécialisé dans une tâche. Ces structures sont abordées en détail au paragraphe §3.5.

3.3.2 Partitionnement de l'espace de fonctionnement

Il existe principalement trois techniques de partitionnement : partitionnement en grille, partitionnement basé sur un arbre de décision (partitionnement hiérarchique orthogonal aux axes [76, 77, 78] et partitionnement hiérarchique oblique par rapport aux axes [79, 80]) et partitionnement par classification floue [21, 81, 82]. Les modes de partitionnement les plus utilisés (partitionnement en grille, partitionnement hiérarchique orthogonal aux axes et partitionnement par classification floue) sont présentés ici.

3.3.2.1 Partitionnement en grille

Ce mode de partitionnement consiste à faire un maillage de l'espace de fonctionnement du système en subdivisant l'intervalle de variation de chaque variable ξ_j du vecteur d'indexation $\underline{\xi}(t)$ en p_j partitions (p_j intervalles) P_{kj}, $k = 1, \ldots, p_j$, $j = 1, \ldots, n_\xi$, n_ξ étant la dimension du vecteur d'indexation $\underline{\xi}$:

$$\underline{\xi}(t) = [\xi_1, \ldots, \xi_j, \ldots, \xi_{n_\xi}]$$

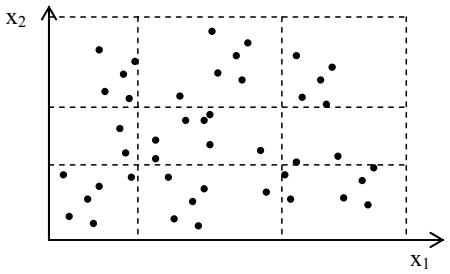

FIG. 3.6 – Partitionnement en grille d'un espace bidimensionnel.

Le degré d'appartenance μ_{kj} d'une variable ξ_j à une partition P_{kj} est spécifié par une fonction d'appartenance $\mu(\cdot)$ appelée également fonction de validité individuelle. Cette fonction peut être choisie comme étant une gaussienne :

$$\mu_{kj} = \mu(\xi_j, P_{kj}) = \exp\Big(- \frac{(\xi_j - c_{kj})^2}{2\sigma_j^2} \Big) \tag{3.8}$$

où c_{kj} est le centre de la partition P_{kj} et σ_j la dispersion. La dispersion σ_j doit être commune à toutes les partitions d'une variable ξ_j afin d'éviter le phénomène de réactivation de modèles locaux [83].

Pour chaque modèle local i ($i = 1, \ldots, M$), la zone de validité Z_i comporte une partition P_{k_ij} de chaque variable ξ_j. Un vecteur de paramètres $\underline{\beta}_i$ dépendant de la structure des fonctions de validité individuelle est associé à la zone Z_i. Par exemple pour des fonctions de validité d'allure gaussienne, $\underline{\beta}_i$ contient les centres et les dispersions pour chacune des partitions formant la zone Z_i. La fonction de validité $\rho_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big)$ d'un modèle local dans la zone de validité Z_i est obtenue par le produit des n_ξ fonctions de validité individuelles correspondantes à chaque partition P_{k_ij} :

$$\rho_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big) = \prod_{j=1}^{n_\xi} \mu_{k_ij} \tag{3.9}$$

Les degrés d'activation des modèles locaux $\omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big)$ sont obtenus de la normalisation des fonctions de validité $\rho_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big)$:

$$\omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big) = \frac{\rho_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big)}{\sum_{k=1}^{M} \rho_k\Big(\underline{\xi}(t), \underline{\beta}_i\Big)}$$

La principale difficulté avec le mode de partitionnement en grille est la détermination du nombre de partition pour chaque variable ainsi que les points de découpage des partitions. On doit en plus s'assurer de disposer de suffisamment de données dans chaque zone obtenue afin que l'estimation des paramètres du modèle local soit possible. Un autre inconvénient du mode de partitionnement en grille est le nombre élevé de modèles locaux qu'il engendre et qui correspond à :

$$M = \prod_{j=1}^{n_\xi} p_j$$

Ce nombre augmente très rapidement avec le nombre de variables et avec le nombre de partitions par variable. Le mode de partitionnement en grille n'est donc approprié que pour les systèmes de faible dimension et est souvent combiné avec une procédure de fusion de modèles locaux [7]. La figure 3.6 montre un exemple de partitionnement en grille d'un espace de fonctionnement bidimensionnel avec comme variable d'entrées x_1 et x_2.

3.3.2.2 Partitionnement hiérarchique orthogonal aux axes

C'est un partitionnement suivant un arbre de décision utilisé par un certain nombre d'auteurs [76, 77, 78]. A partir d'une seule zone, l'espace est successivement décomposé de façon à obtenir à chaque étape un sous-espace de plus qu'à l'étape précédente. La procédure se fait de façon itérative et parallèlement à l'estimation paramétrique. Elle est arrêtée si le multimodèle obtenu satisfait un critère de performance donné ou si la décomposition ne permet plus d'améliorer les performances. La procédure est décrite par l'algorithme 3.1.

Les degrés d'activation $\omega_i(\cdot)$ sont calculés de la même façon que dans le cas du partitionnement en grille. Le principal avantage est un nombre plus réduit de zones de fonctionnement.

La figure 3.7 montre un exemple de partitionnement hiérarchique orthogonal aux axes d'un espace de fonctionnement bidimensionnel avec comme variables d'entrée x_1 et x_2. Le principe de la procédure de décomposition est illustrée sur la figure 3.8.

La démarche permet d'éviter une exploration exhaustive de toutes les possibilités de décomposition, augmentant ainsi la vitesse de convergence de l'algorithme. La décomposition de l'espace de fonctionnement ne se fait que suivant les variables caractéristiques des non linéarités du système, car ce sont elles qui contribuent majoritairement à la diminu-

Algorithme 3.1 : Algorithme de décomposition hiérarchique orthogonale aux axes.

Choisir une structure de multimodèle avec une seule zone de fonctionnement (un seul modèle local) ; estimer les paramètres et évaluer les performances.
Choisir un critère d'arrêt.
tant que *critère d'arrêt non satisfait* **faire**
  Pour chaque variable du vecteur d'indexation $\underline{\xi}$, générer une structure multimodèle par décomposition d'une de ses partitions en deux sous-partitions (la structure obtenue possède donc un modèle local de plus que la précédente) ;
  Estimer les paramètres des multimodèles obtenus ;
  Évaluer les performances des multimodèles ;
  Sélectionner le multimodèle ayant la meilleure performance.
fin

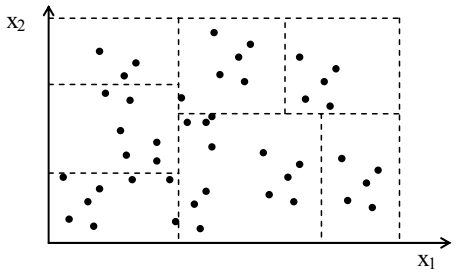

FIG. 3.7 – Partitionnement hiérarchique orthogonal aux axes d'un espace bidimensionnel.

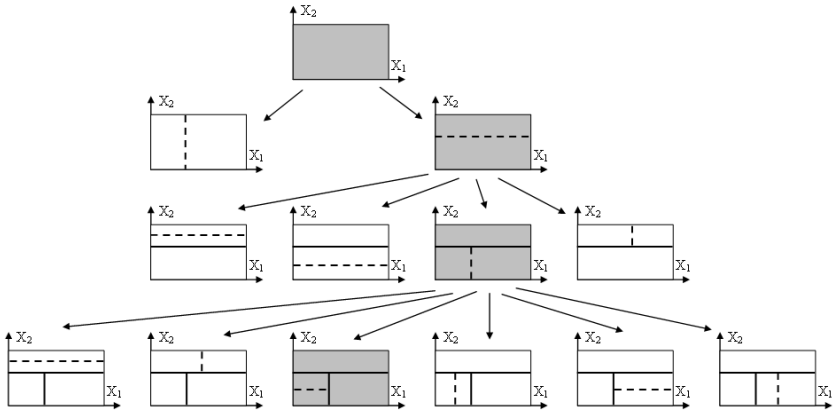

FIG. 3.8 – Principe du partitionnement hiérarchique orthogonal aux axes. A chaque étape la décomposition se fait à partir de la meilleure structure (figure grisée).

tion de l'erreur. On rencontre également avec ce mode de partitionnement le problème de la détermination des points de découpage des partitions. Une démarche permettant l'optimisation du point de découpage est proposée dans [7]. Dans notre implantation, la partition est découpée soit au milieu soit au point median de la partition.

Par ailleurs, le partitionnement hiérarchique orthogonal aux axes permet de faire la sélection des variables d'entrée, un problème majeur qui se pose en identification. En effet, la méthode permet d'identifier les variables caractéristiques des non linéarités du système car c'est suivant ces variables que s'effectue le découpage de l'espace de fonctionnement. Par conséquent, à la fin de la procédure de décomposition, un examen de l'espace de fonctionnement permet d'identifier ces variables caractéristiques. Un nouveau multimodèle peut ainsi être établi avec ces variables.

3.3.2.3 Partitionnement par classification floue

La technique de partitionnement par classification floue est de plus en plus utilisée en identification [9, 20, 84]. Le principe du partitionnement par classification floue est de retrouver, dans un ensemble de données, différents groupes (ou classes) qui se distinguent naturellement les uns des autres. La démarche consiste à maximiser les similarités intra-groupes et à minimiser les similarités inter-groupes, en autorisant un recouvrement entre les groupes. Dans le cas de la multi-modélisation, les groupes constitués correspondent aux zones de validité des modèles locaux. Une observation peut appartenir simultanément à plusieurs groupes à des degrés divers. Chaque groupe est caractérisé par son « centre ». La « distance » entre une observation et le « centre » d'une zone définit le degré d'appartenance de l'observation à la zone considérée. Le degré de validité d'un modèle local pour une observation est donné par le degré d'appartenance de cette observation à la zone du modèle local.

Le partitionnement par classification floue permet la réduction du temps de calcul en évitant la détermination des points de découpage des partitions des variables pour la constitution des zones de validité des modèles locaux, comme c'est le cas avec les modes de partitionnement déjà présentés. Les algorithmes des « *fuzzy-c-means* » [85], de « *Gustafson-Kessel* » [86] et de « *subtractive clustering* » [87] sont parmi les plus utilisés pour la classification floue.

Algorithme des « *fuzzy-c-means* » : Introduit par Bezdek [85], c'est l'algorithme le plus utilisé dans la classification floue.

Soit X la matrice représentant l'ensemble des N observations x_j $(j = 1, \ldots, N)$ disponibles sur n_v variables v_k $(k = 1, \ldots, n_v)$:

$$X = [\underline{x}_1 \quad \underline{x}_2 \quad \cdots \quad \underline{x}_j \quad \cdots \quad \underline{x}_N] \in \mathbf{R}^{n_v \times N}$$

avec

$$\underline{x}_i = [v_1 \quad v_2 \quad \cdots \quad v_k \quad \cdots \quad v_{n_v}]^T \in \mathbf{R}^{n_v \times 1}$$

On suppose que les données observées peuvent être groupées en n_c catégories ou classes :

$$C = [\underline{c}_1 \quad \underline{c}_2 \quad \cdots \quad \underline{c}_i \quad \cdots \quad \underline{c}_{n_c}] \in \mathbf{R}^{n_v \times n_c}$$

C est la matrice des vecteurs prototypes ou centres.

On définit la matrice U appelée matrice d'appartenance dont chaque élément μ_{ij} représente le degré d'appartenance de l'observation $\underline{x}_j$ à la classe $\underline{c}_i$. Les éléments de la matrice U doivent satisfaire les contraintes de normalisation suivantes :

$$\mu_{ij} \in [0,\ 1], \quad \forall j = 1, \ldots N, \quad \forall i = 1, \ldots n_c \tag{3.10}$$

$$\sum_{i=1}^{n_c} \mu_{ij} = 1, \quad \forall j = 1, \ldots, N \tag{3.11}$$

L'objectif de l'algorithme des « *fuzzy-c-means* » est de minimiser la fonction coût définie par :

$$J(X, U, C) = \sum_{i=1}^{n_c} \sum_{j=1}^{N} \mu_{ij}^m D_{ijA}^2 \tag{3.12}$$

où :

$m \in [1, +\infty[$ est l'exposant flou qui caractérise le recouvrement entre les classes. Si m est proche de 1, le recouvrement est faible (les classes sont nettes). Si $m \to +\infty$, le recouvrement est fort ;

D_{ijA} est une norme sur $\mathbf{R}^{n_v}$ permettant de mesurer la distance entre l'observation $\underline{x}_j$ et le centre $\underline{c}_i$:

$$D_{ijA}^2 = \|\underline{x}_j - \underline{c}_i\|_A^2 = (\underline{x}_j - \underline{c}_i)^T A (\underline{x}_j - \underline{c}_i) \tag{3.13}$$

La matrice définie positive $A \in \mathbf{R}^{n_v \times n_v}$ caractérise la métrique. Si A est la matrice identité, la distance D_{ijA} est une distance euclidienne et les classes obtenues sont dites sphériques. D'autres normes peuvent être utilisées pour le calcul de la distance D_{ijA}.

A chaque classe i de centre $\underline{c}_i$ est associé un modèle local dont le degré d'activation pour une observation $\underline{\varphi}(j)$ donnée, indexée par $\underline{\xi}(j)$, est :

$$\omega_i\Big(\underline{\xi}(j), \underline{c}_i\Big) = \mu_{ij} \tag{3.14}$$

L'algorithme des « *fuzzy-c-means* » est un algorithme itératif et se présente suivant le schéma 3.2 (l indique l'itération courante).

Algorithme 3.2 : Algorithme des « *fuzzy-c-means* »

initialisation : fixer le nombre de groupes $(2 \leq n_c < N)$; choisir la métrique A ; fixer une valeur du paramètre m (en général on choisit $m = 2$) ; initialiser de façon aléatoire la matrice U de telle sorte que les contraintes (3.10) et (3.11) soient vérifiées ; fixer un critère d'arrêt (par exemple δ, variation minimale de U entre deux itérations consécutives).

répéter

Calculer les centres des classes :

$$\underline{c}_i^{(l)} = \frac{\sum_{j=1}^{N} \left(\mu_{ij}^{(l-1)}\right)^m \underline{x}_j}{\sum_{j=1}^{N} \left(\mu_{ij}^{(l-1)}\right)^m}$$

Calculer les distances :

$$D_{ijA}^2 = \|\underline{x}_j - \underline{c}_i\|_A^2 = (\underline{x}_j - \underline{c}_i)^T A (\underline{x}_j - \underline{c}_i) \qquad 1 \leq i \leq n_c,\ 1 \leq j \leq N$$

Mettre à jour la matrice d'appartenance :

si $D_{ijA} > 0$ **alors**

$$\mu_{ij}^{(l)} = \frac{1}{\sum_{k=1}^{n_c} \left(D_{ijA}/D_{kjA}\right)^{2/(m-1)}} \qquad 1 \leq i \leq n_c,\ 1 \leq j \leq N$$

sinon

si $D_{ijA} > 0$ *et* $\mu_{ij}^{(l)} \in [0,1]$ *avec* $\sum_{i=1}^{n_c} \mu_{ij}^{(l)} = 1$ **alors** $\mu_{ij}^{(l)} = 0$

fin

jusqu'à $\|U^{(l)} - U^{(l-1)}\| < \delta$

L'inconvénient majeur de ce mode de partitionnement est qu'il faut au préalable indiquer le nombre de groupes nécessaires, ce qui n'est pas souvent connu. Dans ce travail, l'algorithme des « *fuzzy-c-means* » est combiné à une démarche itérative permettant de déterminer le nombre adéquat de groupes. Elle consiste, à partir d'un nombre minimal de

groupes (2 par exemple), de rajouter à chaque itération un nouveau groupe et d'évaluer ensuite les performances (erreur ou critère de sélection, voir 1.5.2) du multimodèle obtenu. Tant que les performances des structures ainsi générées s'améliorent, on continue la procédure. Cette dernière est arrêtée si l'objectif est atteint ou si les performances se dégradent.

Algorithme de « *Gustafson-Kessel* » : L'algorithme de « *Gustafson-Kessel* » [86] est une extension de l'algorithme des « *fuzzy-c-means* », permettant l'évaluation des distances dans chaque groupe par une métrique individuelle adaptée. Cela permet de détecter des groupes de formes et d'orientations différentes. Pour un groupe i, la métrique est donnée par la relation :

$$A_i = \alpha_i |F_i|^{1/n_v} F_i^{-1} \tag{3.15}$$

avec α_i désignant le volume du groupe (en général 1) et F_i la matrice de « variance-covariance » floue du groupe i définie par :

$$F_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^m (\underline{x}_j - \underline{c}_i)(\underline{x}_j - \underline{c}_i)^T}{\sum_{j=1}^{N} (\mu_{ij})^m} \tag{3.16}$$

La fonction coût est définie par :

$$J(X, U, C, \{A_i\}) = \sum_{i=1}^{n_c} \sum_{j=1}^{N} \mu_{ij}^m D_{ijA_i}^2 \tag{3.17}$$

L'algorithme de « *Gustafson-Kessel* » est représenté sur le schéma 3.3 (l indiquant l'itération courante).

L'inconvénient de cet algorithme est qu'il peut conduire à des problèmes de calcul numérique lors de l'évaluation des matrices de « variance-covariance » floues lorsque les groupes contiennent peu de données ou lorsque les données sont linéairement corrélées. Ces matrices sont singulières et leur inversion pose problème. Une méthode permettant de contourner ce problème est présentée dans [84].

Algorithme du « *subtractive clustering* » : La méthode de regroupement par « *subtractive clustering* »(ou regroupement par soustraction) a été introduite par Chiu [87].

Algorithme 3.3 : Algorithme de « *Gustafson-Kessel* »

initialisation : fixer le nombre de groupes ($2 \leq n_c < N$) ; fixer une valeur du paramètre m ; initialiser de façon aléatoire la matrice U de telle sorte que les contraintes (3.10) et (3.11) soient vérifiées ; fixer un critère d'arrêt (par exemple δ, variation minimale de U entre deux itérations consécutives).

répéter

Calculer les centres des classes :

$$\underline{c}_i^{(l)} = \frac{\sum_{j=1}^{N} \left(\mu_{ij}^{(l-1)}\right)^m \underline{x}_j}{\sum_{j=1}^{N} \left(\mu_{ij}^{(l-1)}\right)^m}$$

Calculer les matrices de « variance-covariance » floues :

$$F_i = \frac{\sum_{j=1}^{N} (\mu_{ij})^m (\underline{x}_j - \underline{c}_i)(\underline{x}_j - \underline{c}_i)^T}{\sum_{j=1}^{N} (\mu_{ij})^m}$$

Calculer les distances :

$$D_{ijA_i}^2 = \|\underline{x}_j - \underline{c}_i\|_{Ai}^2 = (\underline{x}_j - \underline{c}_i)^T \left(\alpha_i |F_i|^{1/n_v} F_i^{-1}\right)(\underline{x}_j - \underline{c}_i) \quad 1 \leq i \leq n_c,\ 1 \leq j \leq N$$

Mettre à jour la matrice d'appartenance :

si $D_{ijA} > 0$ **alors**

$$\mu_{ij}^{(l)} = \frac{1}{\sum_{k=1}^{n_c} \left(D_{ijA}/D_{kjA}\right)^{2/(m-1)}} \qquad 1 \leq i \leq n_c,\ 1 \leq j \leq N$$

sinon

si $D_{ijA} > 0$ *et* $\mu_{ij}^{(l)} \in [0,1]$ *avec* $\sum_{i=1}^{n_c} \mu_{ij}^{(l)} = 1$ **alors** $\mu_{ij}^{(l)} = 0$

fin

jusqu'à $\|U^{(l)} - U^{(l-1)}\| < \delta$

Contrairement aux techniques de regroupement par « *fuzzy-c-means* » et « *Gustafson-Kessel* », elle n'est pas basée sur la minimisation d'une fonctionnelle et son principal avantage est que le nombre de groupes (qui est en général inconnu) n'a pas à être spécifié. Ici chaque point de la base de données est un centre potentiel et l'algorithme décèle les points autour desquels il y a plus de concentration de données. Pour cela, il est défini pour chaque observation $\underline{\varphi}(k)$ un potentiel P_k en fonction de sa distance par rapport aux autres observations. Ces distances sont calculées à l'aide des variables d'indexation

correspondant à ces observations :

$$P_k = \sum_{j=1}^{N} \exp\left(-\frac{4}{r_a^2} \times \left\|\underline{\xi}(k) - \underline{\xi}(j)\right\|^2\right) \tag{3.18}$$

où :

$\|\cdot\|$ est la distance euclidienne ;

r_a est une constante positive appelée rayon du groupe.

Le potentiel d'un point est d'autant plus élevé qu'il y a de données dans son voisinage. Désignant par P_1^* le potentiel le plus élevé correspondant au premier centre $\underline{c}_1$, les potentiels de tous les autres points sont réévalués en excluant l'influence de $\underline{c}_1$:

$$P_j^{(2)} = P_j^{(1)} - P_1^* \exp\left(-\frac{4}{r_b^2} \times \left\|\underline{c}_1 - \underline{\xi}(j)\right\|^2\right) \tag{3.19}$$

où $P_j^{(1)}$ et $P_j^{(2)}$ sont les valeurs du potentiel P_j aux itérations 1 et 2 respectivement et r_b est une constante positive qui définit le voisinage (distance minimale entre deux groupes) : $r_b = \eta\, r_a$, et η une constante positive appelée « *squash factor* ». On détermine ensuite le point de plus grand potentiel P_2^* parmi les P_j recalculés par l'expression (3.19). Ce point est accepté comme centre ($\underline{c}_2$) si l'une des conditions suivantes est vérifiée :

$$P_2^* > \varepsilon_{sup} P_1^* \tag{3.20}$$

$$\frac{d_{min}}{r_a} + \frac{P_2^*}{P_1^*} \geq 1 \tag{3.21}$$

où ε_{sup} est une constante positive appelée « *accept ratio* » et d_{min} est la distance minimale entre le centre $\underline{c}_2$ et tous les autres centres trouvés (à cette étape, $\underline{c}_1$ seul).

Les autres centres sont obtenus par itération de l'algorithme. A l'itération k, les potentiels des points sont réévalués par l'expression :

$$P_j^{(k)} = P_j^{(k-1)} - P_{k-1}^* \exp\left(-\frac{4}{r_b^2} \times \left\|\underline{c}_{k-1} - \underline{\xi}(j)\right\|^2\right) \tag{3.22}$$

Si le point de potentiel maximal P_k^* ne vérifie aucune des conditions (3.20) et (3.21), P_k^* est fixé à 0 et le point de plus grand potentiel est sélectionné parmi les points restants.

Ces étapes sont itérées jusqu'à ce que la condition suivante soit vérifiée :

$$P_k^* < \varepsilon_{inf} P_1^* \tag{3.23}$$

où ε_{inf} est une constante positive appelée « *reject ratio* ».

Algorithme 3.4 : Algorithme du « *subtractive clustering* »

Initialiser les paramètres $r_a,\ \eta,\ \varepsilon_{sup},\ \varepsilon_{inf}$

pour $i = 1$ ***à*** N **faire**

Initialiser le potentiel P_i de $\underline{\xi}(i)$ par l'équation :

$$P_i = \sum_{j=1}^{N} \exp\left(-\frac{4}{r_a^2} \times \left\|\underline{\xi}(i) - \underline{\xi}(j)\right\|^2\right)$$

fin

Accepter $\underline{\xi}_1^*$ comme centre tel que $P_1^* = \max_i(P_i)$

l=1 //compteur de centres de groupes

$P_m^* = P_1^*$

tant que $P_m^* \geq \varepsilon_{inf} P_1^*$ **faire**

incrémenter l

pour $k = 1$ ***à*** N **faire**

Mettre à jour chaque potentiel P_k de $\underline{\xi}(k)$en excluant l'influence du point de potentiel le plus élevé (P_{l-1}^*) :

$$P_k = P_k - P_{l-1}^* \exp\left(-\tfrac{4}{r_b^2} \times \left\|\underline{\xi}_{l-1}^* - \underline{\xi}(k)\right\|^2\right)$$

fin

Poser $P_m^* = \max_k(P_k)$

Tester si le point $\underline{\xi}(h)$ de potentiel P_m^* est un centre : calculer la plus petite des distances entre $\underline{\xi}(h)$ et tous les autres centres trouvés précédemment :

$d_{min} = \min_{i=1,\ldots,(l-1)} \left\|\underline{\xi}(h) - \underline{\xi}_i^*\right\|$

si $P_m^* > \varepsilon_{sup} P_1^*$ *ou* $\frac{d_{min}}{r_a} + \frac{P_m^*}{P_1^*} \geq 1$ **alors**

Accepter $\underline{\xi}(h)$ comme centre : $\underline{\xi}_l^* = \underline{\xi}(h)\ ,\quad P_l^* = P_m^*$

sinon

Rejeter $\underline{\xi}(h) : P_h = 0\ ,\quad P_l^* = P_{l-1}^*$

fin

fin

Une description de l'algorithme du « *subtractive clustering* » est représentée sur le schéma 3.4.

Des valeurs empiriques des paramètres utilisés sont suggérées dans la littérature ([87, 88]) :

$$\eta = 1.5 \qquad \varepsilon_{sup} = 0.5 \qquad \varepsilon_{inf} = 0.15 \qquad 0.15 \leq r_a \leq 0.3 \tag{3.24}$$

Cet algorithme a le mérite de permettre la détermination du nombre de groupes (ou de modèles locaux) nécessaires à l'implantation de l'architecture multimodèle. L'appartenance d'une observation $\underline{\varphi}(k)$ (désignée par la variable d'indexation $\underline{\xi}(k)$) à un groupe de centre $\underline{c}_i$ est donnée par la relation :

$$\rho_i\left(\underline{\xi}(k)\right) = \exp\left(-\frac{4}{r_a^2} \times \left\|\underline{c}_i - \underline{\xi}(k)\right\|^2\right) \tag{3.25}$$

Les degrés d'activation $\omega_i\left(\xi(t)\right)$ des modèles locaux sont ensuite calculés par la normalisation des degrés d'appartenance $\rho_i\Big(\xi(t)\Big)$ conformément à (3.2).

3.4 Estimation paramétrique d'un multimodèle

Nous avons vu que l'identification structurelle d'un multimodèle permet de spécifier la structure des modèles locaux et des zones de fonctionnement. Cette étape conduit à l'établissement d'une famille de fonctions de la forme (3.1). Cette famille de fonctions est paramétrée par le vecteur de paramètres $\underline{\theta}_i$, définissant la structure du modèle local i, et le vecteur de paramètres $\underline{\beta}_i$, caractérisant la zone de validité de ce même modèle local.

L'estimation paramétrique consiste à déterminer pour chaque modèle local i le vecteur de paramètres :

$$\underline{\Theta}_i = [\underline{\theta}_i^T,\ \underline{\beta}_i^T]^T$$

Cependant, dans le cas du partitionnement par classification floue, les paramètres $\underline{\beta}_i$ sont déterminés pendant la décomposition.

Le multimodèle est donc entièrement défini par le vecteur de paramètres :

$$\underline{\Theta} = [\underline{\Theta}_1^T, \ldots, \underline{\Theta}_i^T, \ldots, \underline{\Theta}_M^T]^T \tag{3.26}$$

L'estimation paramétrique (appelée aussi apprentissage) se fait sur la base de l'optimisation (minimisation) d'une fonctionnelle liant les entrées et sorties du système à l'ensemble des paramètres qui caractérisent le modèle. L'estimation des paramètres d'un multimodèle peut se faire suivant deux types de critères d'apprentissage :

- un critère d'apprentissage local (J_L) visant à minimiser l'écart entre la sortie du système et celle de chaque modèle local. Ainsi, l'espace de fonctionnement du système étant décomposé en plusieurs zones, chaque modèle local représente le

système dans sa zone de validité. Le critère local associé au modèle local i est :

$$J_i = \frac{1}{2}\sum_{t=1}^{N} \omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big) \left[y_s(t) - f_i\Big(\underline{\varphi}(t), \underline{\theta}_i\Big)\right]^2 \tag{3.27}$$

Pour le multimodèle, le critère d'apprentissage local est :

$$J_L = \sum_{i=1}^{M} J_i \tag{3.28}$$

– un critère d'apprentissage global (J_G) visant à minimiser l'écart entre la sortie du système et celle du multimodèle :

$$J_G = \frac{1}{2}\sum_{t=1}^{N}\Big(y_s(t) - \hat{y}(t)\Big)^2 = \frac{1}{2}\sum_{t=1}^{N}\left(y_s(t) - \sum_{i=1}^{M} \omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big) f_i\Big(\underline{\varphi}(t), \underline{\theta}_i\Big)\right)^2 \tag{3.29}$$

Dans ce cas, la qualité de l'approximation locale n'est pas pris en considération.

L'optimisation de ces critères dépend du type de modèle à identifier (modèle récurrent ou non-récurrent), de la structure des modèles locaux (modèles locaux linéaires ou non par rapport aux paramètres) et de la structure des zones de fonctionnement.

Nous avons vu qu'avec une structure polynômiale, les modèles locaux peuvent se mettre sous la forme (voir 3.7) :

$$f_i\Big(\underline{\varphi}(t), \underline{\theta}_i\Big) = \underline{\Phi}_i(t)^T\ \underline{\theta}_i \tag{3.30}$$

Les critères J_L et J_G sont donc quadratiques par rapport aux paramètres des modèles locaux, mais ne le sont pas par rapport aux paramètres des degrés d'activation (paramètres des zones de validité des modèles locaux). L'optimisation se fera selon que les paramètres des degrés d'activation sont fixés ou pas. Nous présentons ici l'estimation paramétrique d'un multimodèle (non-récurrent puis récurrent) dont les modèles locaux ont une structure polynômiale.

3.4.1 Estimation paramétrique d'un multimodèle non-récurrent

Deux méthodes sont proposées selon que les paramètres des fonctions d'activation sont fixes ou non.

3.4.1.1 Paramètres des fonctions d'activation fixes

Dans le cas d'un multimodèle non-récurrent, on peut disposer de l'ensemble des données nécessaires à l'identification. On utilise alors une méthode globale de minimisation, consistant à minimiser le critère sur l'ensemble des données simultanément. Avec la structure des modèles locaux proposée (voir (3.6)), l'estimation par la méthode des moindres carrées ordinaires (pour le critère global) ou par la méthodes des moidres carrés pondérés (pour le critère local) donne respectivement les expressions suivantes pour les paramètres des modèles locaux :

$$\underline{\hat{\theta}} = (\Phi_g^T \, \Phi_g)^{-1} (\Phi_g^T \, \underline{Y}_s) \tag{3.31}$$

$$\underline{\hat{\theta}}_i = (\Phi_{eg}^T \, W_i \, \Phi_{eg})^{-1} (\Phi_{eg}^T \, W_i \, \underline{Y}_s) \tag{3.32}$$

où :

$\underline{\hat{\theta}}$ est l'estimation de $\underline{\theta} = [\underline{\theta}_1^T, \ldots, \underline{\theta}_i^T, \ldots, \underline{\theta}_M^T]^T$;
$\underline{\hat{\theta}}_i$ est l'estimation $\underline{\theta}_i$ (vecteur de paramètres du modèle local i) ;
$\Phi_g = \Big[\Phi(t)\Big]_{t=1}^{t=N}$ est la matrice de régression globale pour toutes les observations ;
$\underline{Y}_s = \Big[y_s(t)\Big]_{t=1}^{t=N}$ est le vecteur des sorties du système pour toutes les observations ;
$\Phi_{eg} = \Big[\underline{\varphi}_{p_e}(t)\Big]_{t=1}^{t=N}$ est la matrice de régression augmentée globale ;
$W_i = \mathrm{Diag}\Big[\omega_i\Big(\underline{\xi}(t)\Big)\Big]_{t=1}^{t=N}$ est la matrice de pondération diagonale.

Dans le cas où les modèles locaux sont non linéaires par rapport aux paramètres, l'estimation paramétrique se fait suivant une optimisation non linéaire (par exemple la méthode de Levenberg-Marquardt présentée au paragraphe §1.6.1.2).

3.4.1.2 Optimisation des paramètres des fonctions d'activation

Le critère à optimiser est quadratique par rapport aux paramètres des modèles locaux, mais ne l'est pas par rapport aux paramètres des degrés d'activation. Une optimisation à deux niveaux peut alors être utilisée pour déterminer l'ensemble des paramètres des modèles locaux et des degrés d'activation. Cette technique d'optimisation est décrite par l'algorithme 3.5.

Algorithme 3.5 : Algorithme d'estimation paramétrique d'un multimodèle non-récurrent avec optimisation des paramètres des fonctions d'activation

Initialisation :

$k = 0$; fixer une tolérance δ correspondant à une variation minimale du critère entre deux itérations ;

fixer les paramètres des degrés d'activation $\underline{\beta}(0)$ (selon le mode de partitionnement choisi).

répéter

Calculer les paramètres des modèles locaux $\underline{\Theta}(k)$ par la méthode des moindres carrés ;

Calculer la valeur du critère $J(k)$;

Calculer les paramètres $\underline{\beta}(k)$ des degrés d'activation par une optimisation non linéaire (par exemple Levenberg-Marquardt) ;

Incrémenter k ;

jusqu'à $\|J(k) - J(k-1)\| < \delta$

3.4.2 Estimation paramétrique d'un multimodèle récurrent

Nous n'envisageons ici que le cas où les paramètres des degrés d'activation sont fixés. Les données d'apprentissage n'étant pas entièrement disponibles, les critères J_L et J_G sont évalués à chaque instant k par les expressions :

$$J_L(k) = \frac{1}{2} \sum_{t=1}^{k} \sum_{i=1}^{M} \omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big) \left[y_s(t) - f_i\Big(\underline{\varphi}(t), \underline{\theta}_i\Big) \right]^2 \tag{3.33}$$

$$J_G(k) = \frac{1}{2} \sum_{t=1}^{k} \sum_{i=1}^{M} \left[y_s(t) - \omega_i\Big(\underline{\xi}(t), \underline{\beta}_i\Big) f_i\Big(\underline{\varphi}(t), \underline{\theta}_i\Big) \right]^2 \tag{3.34}$$

Ces critères étant quadratiques par rapport aux paramètres des modèles locaux, ils peuvent être optimisés à chaque instant par la méthode des moindres carrés récursifs (décrite au paragraphe §1.6.2). L'optimisation est décrite par l'algorithme d'adaptation paramétrique 3.6.

Algorithme 3.6 : Algorithme d'estimation paramétrique d'un multimodèle récurrent

Initialisation :

$k = 0$; fixer une tolérance δ correspondant à une variation minimale du critère entre deux itérations ;

calculer les paramètres des modèles locaux $\underline{\Theta}(0)$ par la méthode des moindres carrés ordinaires en considérant que le modèle est non-récurrent ; calculer la valeur initiale du critère $J(0)$.

répéter

pour $k = 1$ ***à*** $N - 1$ **faire**

Utiliser les relations (1.57) et (1.58) pour estimer $\underline{\Theta}$ à l'itération l :

$$\underline{\Theta}_{k+1}^{(l)} = \underline{\Theta}_{k}^{(l)} + A_{k+1}^{(l)}\, \underline{\Phi}(k)\, \tilde{\varepsilon}(k+1)$$

$$A_{k+1}^{(l)} = A_{k}^{(l)} - \frac{A_{k}^{(l)}\, \underline{\Phi}(k)\, \underline{\Phi}(k)^T\, A_{k}^{(l)}}{1 + \underline{\Phi}(k)^T\, A_{k}^{(l)}\, \underline{\Phi}(k)}$$

où :

$\underline{\Phi}(k)$ est le vecteur de régression global pondéré (voir §3.3.1.2) ;

$\tilde{\varepsilon}(k+1) = y_s(k+1) - \underline{\Phi}(k)^T\, \underline{\Theta}_{k}^{(l)}$ est l'erreur de prédiction *a priori* (erreur à l'instant $k+1$ évaluée avec les paramètres estimés à l'instant k).

Calculer les variables rebouclées et mettre à jour $\underline{\Phi}(k+1)$ avec ces variables ;

fin

Calculer $J(l)$;

jusqu'à $\|J(l) - J(l-1)\| > \delta$,

3.4.3 Estimation paramétrique et identification structurelle des zones de validité

La principale difficulté pour la mise en place d'un multimodèle est la connaissance du nombre de modèles locaux nécessaires. Nous avons montré que le mode de partitionnement flou par « *subtractive clustering* » (paragraphe §3.3.2.3) permet de déterminer le nombre de groupes. Cependant, il est nécessaire de fixer la valeur de certains paramètres qui influent sur le nombre final de groupes .

Le mode de partitionnement hiérarchique orthogonal aux axes permet également de déterminer les zones de validité en même temps que l'estimation paramétrique. Le

principal inconvénient est la détermination des points de découpage des partitions des variables d'indexation. Pour tous les autres modes de partitionnement, il est nécessaire de fixer à l'avance le nombre de groupes avant que ces derniers ne soient déterminés.

Le problème consiste donc à élaborer plusieurs multimodèles (avec différents nombres de modèles locaux) et de sélectionner celui qui présente les meilleures performances. Nous proposons ici une démarche constructive qui consiste à partir d'un multimodèle avec un seul modèle local, et de subdiviser l'espace de fonctionnement du système de façon à augmenter progressivement le nombre de modèles locaux du multimodèle. A chaque étape, les paramètres du multimodèle généré sont calculés ainsi qu'un critère de sélection du multimodèle (critère AIC par exemple). Tant que ce critère s'améliore, la procédure de subdivision de l'espace de fonctionnement du système se poursuit. Elle est arrêtée si le critère de sélection se dégrade. Cette démarche est illustrée dans les organigrammes des figures 3.9 et 3.10 pour un multimodèle non-récurrent et un multimodèle récurrent respectivement. Dans le premier cas l'estimation paramétrique se fait par la méthode des moindres carrés ordinaires (MCO) si les modèles locaux sont linéaires par rapport aux paramètres, ou par programmation non linéaire (PNL) si les modèles locaux sont non linéaires par rapport aux paramètres (paragraphe §3.4.1). Dans le cas du multimodèle récurrent, un algorithme d'adaptation paramétrique (AAP) est utilisé comme indiqué au paragraphe §3.4.2.

3.4.4 Exemple d'identification d'un système dynamique non linéaire par un multimodèle

Nous considérons le système étudié au paragraphe §2.5 dont le modèle est donné par l'équation (2.24) rappelé ici :

$$y_s(t+1) = \frac{y_s(t)y_s(t-1)y_s(t-2)u(t-1)\left[y_s(t-2)-1\right]+u(t)}{1+y_s(t-1)^2+y_s(t-2)^2}$$

L'entrée $u(t)$ est constituée de créneaux d'amplitudes et de largeurs variables. Les données sont générées pour t variant de 1 à 1800 en considérant plusieurs cas de figures : présence de bruit de sortie, présence de bruit d'état et présence de bruit de sortie et de bruit d'état. Pour chaque cas de figure, la première moitié de la base de données générée est utilisée pour l'identification et l'autre moitié pour la validation.

Le critère AIC est utilisé pour la sélection des modèles dans les algorithmes d'identification présentés sur les figures 3.9 et 3.10.

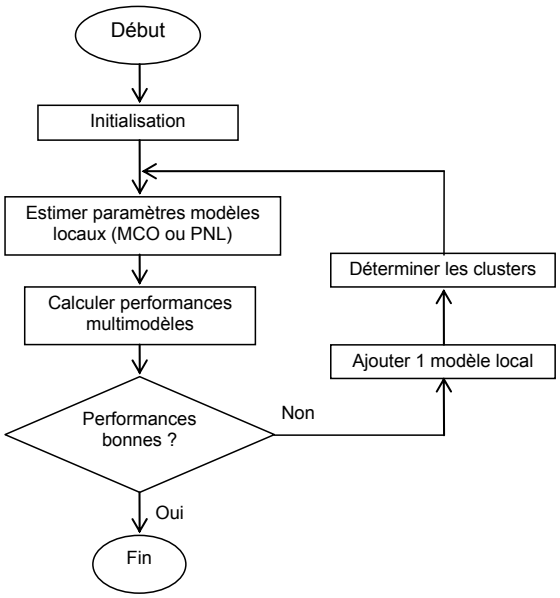

FIG. 3.9 – Combinaison de l'estimation paramétrique d'un multimodèle non-récurrent à la détermination des zones de validité des modèles locaux.

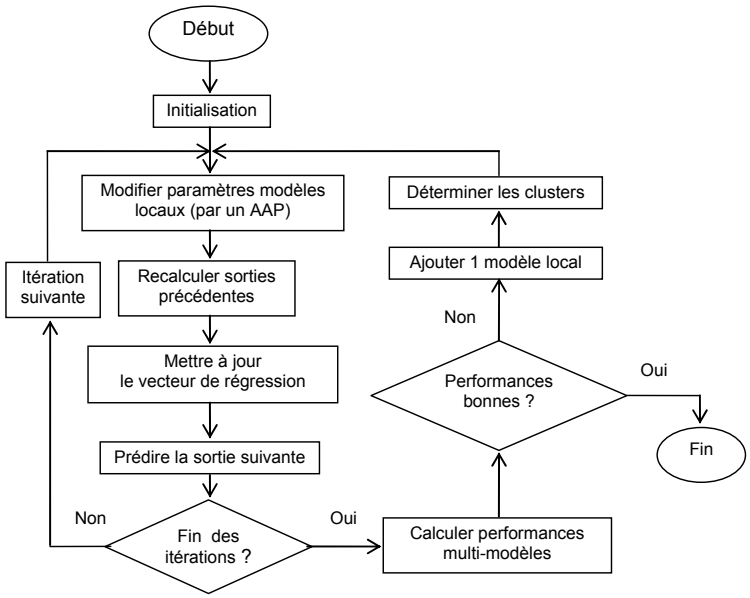

FIG. 3.10 – Combinaison de l'estimation paramétrique d'un multimodèle récurrent à la détermination des zones de validité des modèles locaux.

L'algorithme des « *fuzzy-c-means* » est utilisé pour la détermination des zones de validité des modèles locaux. Les modèles locaux sont choisis de structure affine.

Les vecteurs de régression utilisés pour les multimodèles NARX, NOE et NARMAX sont respectivement :

$$\begin{aligned}
\underline{\varphi}_{NARX}(t) &= \Big[u(t)\ u(t-1)\ y_s(t)\ y_s(t-1)\ y_s(t-2)\Big]^T \\
\underline{\varphi}_{NOE}(t) &= \Big[u(t)\ u(t-1)\ \hat{y}(t)\ \hat{y}(t-1)\ \hat{y}(t-2)\Big]^T \\
\underline{\varphi}_{NARMAX}(t) &= \Big[u(t)\ u(t-1)\ y_s(t)\ y_s(t-1)\ y_s(t-2)\ \varepsilon(t)\Big]^T
\end{aligned}$$

où $\hat{y}$ est la sortie prédite par le multimodèle et $\varepsilon(t) = y_s(t) - \hat{y}(t)$ est l'erreur de prédiction.

En présence de bruit de sortie

En présence de bruit de sortie, les signaux d'apprentissage sont obtenus par :

$$y_{s_{ba}}(t) = y_s(t) + e_1(t);$$

où le bruit $e_1(t)$ est une réalisation d'une variable aléatoire qui suit une distribution normale centrée de variance 0.2, $\mathcal{N}(0, 0.2)$. Le rapport signal/bruit est de 7.5 dB.

La procédure d'identification décrite sur les figures 3.9 et 3.10 a permis d'obtenir les résultats présentés au tableau 3.1.

Les sorties du système et des multimodèles NOE, NARX et NARMAX en validation sont tracées sur la figure 3.11. La figure 3.12 représente les résidus obtenus par les multimodèles en validation et la figure 2.16, les fonctions d'autocorrélation des résidus.

Classe de modèle	Nb. modèles locaux	AIC	*RMSE* app.	*RMSE* valid.
NOE	2	-2647	0.186	0.175
NARX	2	-2649	0.186	0.176
NARMAX	2	-2212	0.244	0.277

TAB. 3.1 – Résultats obtenus pour les multimodèles NOE, NARX et NARMAX en présence de bruit de sortie.

.

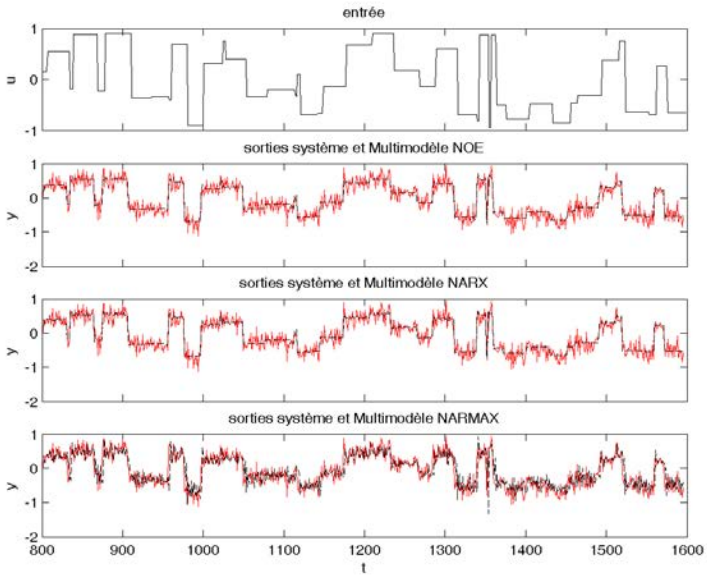

FIG. 3.11 – Identification en présence de bruit de sortie : entrée, sortie bruitée du système (trait plein) et sortie des multimodèles (trait discontinu) NOE, NARX et NARMAX en validation.

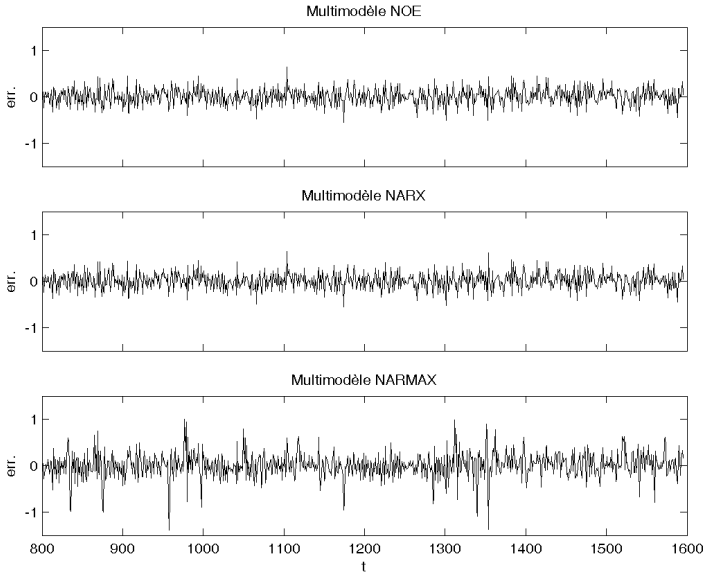

FIG. 3.12 – Influence d'un bruit de sortie : résidus des multimodèles en validation.

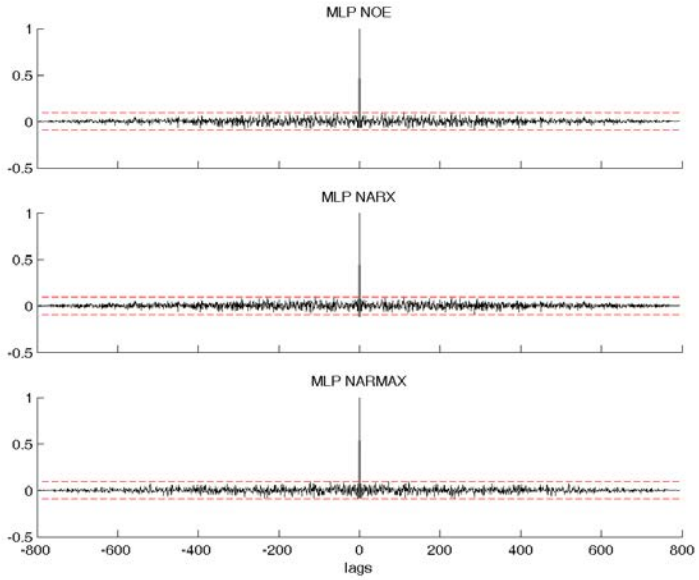

FIG. 3.13 – Fonctions d'autocorrélation des résidus des multimodèles en validation en présence de bruit de sortie.

En présence de bruit d'état

En présence de bruit d'état, les signaux d'apprentissage sont obtenus par :

$$\begin{cases} y_s(t+1) = & \dfrac{y_s(t)y_s(t-1)y_s(t-2)u(t-1)\left[y_s(t-2)-1\right]+u(t)}{1+y_s(t-1)^2+s(t-2)^3} + e_2(t+1) \\ y_{s_{be}}(t+1) = & y_s(t+1) \end{cases}$$

où le bruit $e_2(t)$ est une réalisation d'une variable aléatoire qui suit une distribution normale $\mathcal{N}(0, 0.17)$. Le rapport signal/bruit est de 7.3 dB.

Les résultats obtenus de la procédure d'identification décrite sur les figures 3.9 et 3.10 sont présentés au tableau 3.2.

La figure 3.14 représente les sorties du système et des multimodèles NARX, NOE et NARMAX en validation. La figure 3.15 représente les résidus obtenus par les multimodèles en validation. La figure 3.16 représente les fonctions d'autocorrélation des résidus.

En présence de bruit d'état et de bruit de sortie

En présence de bruit d'état et de bruit de sortie, les signaux d'apprentissage sont obtenus par :

Classe de modèle	Nb. modèles locaux	AIC	*RMSE* app.	*RMSE* valid.
NARX	4	-2659	0.181	0.183
NOE	2	-2645	0.186	0.184
NARMAX	2	-2251	0.238	0.252

TAB. 3.2 – Résultats obtenus pour les multimodèles NARX, NOE et NARMAX en présence de bruit d'état.

.

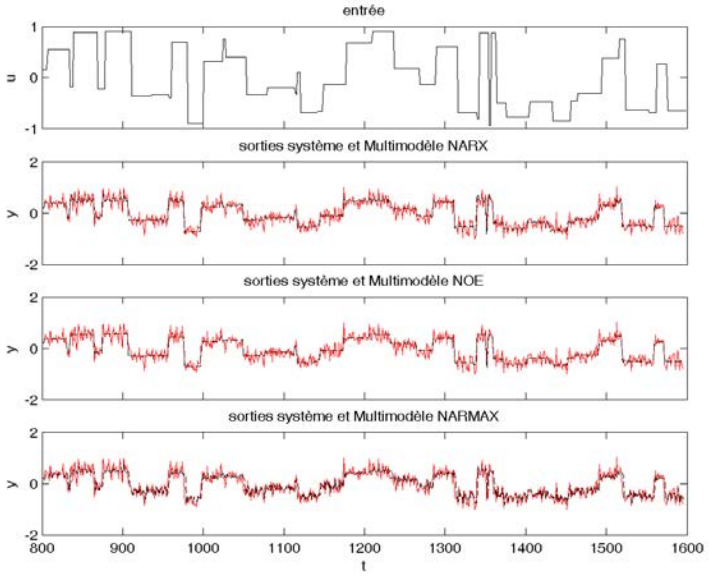

FIG. 3.14 – Identification en présence de bruit d'état : entrée, sortie bruitée du système (trait plein) et sortie des multimodèles (trait discontinu) NARX, NOE et NARMAX en validation

$$\begin{cases} y_s(t+1) = & \dfrac{y_s(t)y_s(t-1)y_s(t-2)u(t-1)\left[y_s(t-2)-1\right]+u(t)}{1+y_s(t-1)^2+s(t-2)^3} + e_2(t+1) \\ y_{s_{bse}}(t+1) = & y_s(t+1) + e_3(t+1) \end{cases}$$

$e_3(t)$ est une réalisation d'une variable aléatoire qui suit une loi de distribution normale $\mathcal{N}(0, 0.2)$. Le rapport signal/bruit est de 4.3 dB.

Les résultats de la procédure d'identification sont présentés au tableau 3.3. La figure 3.17 représente les sorties du système et des multimodèles NARX, NOE et NARMAX

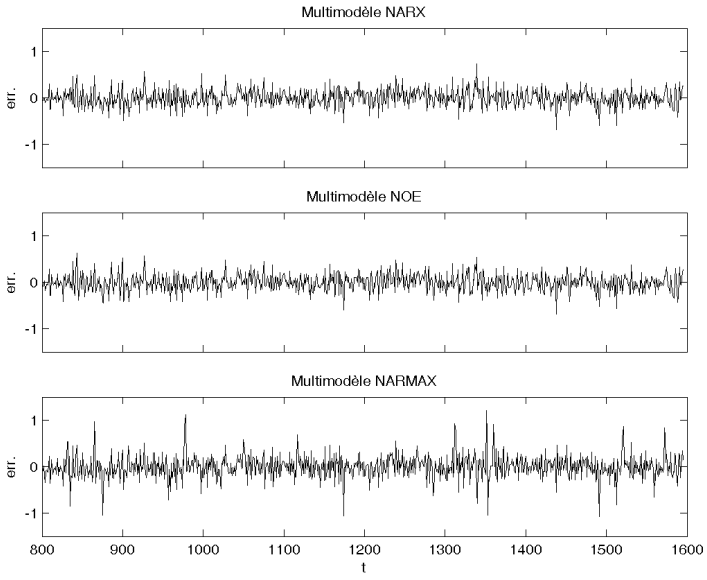

FIG. 3.15 – Influence d'un bruit d'état : résidus des multimodèles en validation.

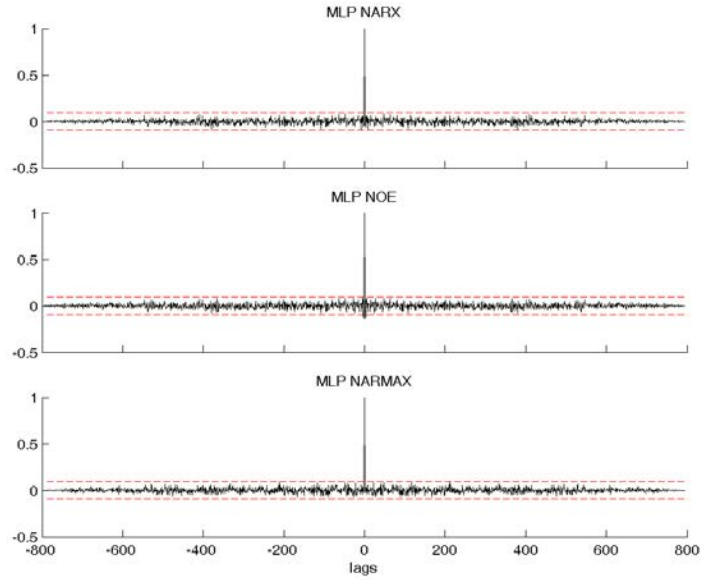

FIG. 3.16 – Fonctions d'autocorrélation des résidus des multimodèles en validation en présence de bruit d'état.

en validation. La figure 3.18 représente les résidus obtenus par les multimodèles en validation et la figure 3.19 les fonctions d'autocorrélation des résidus.

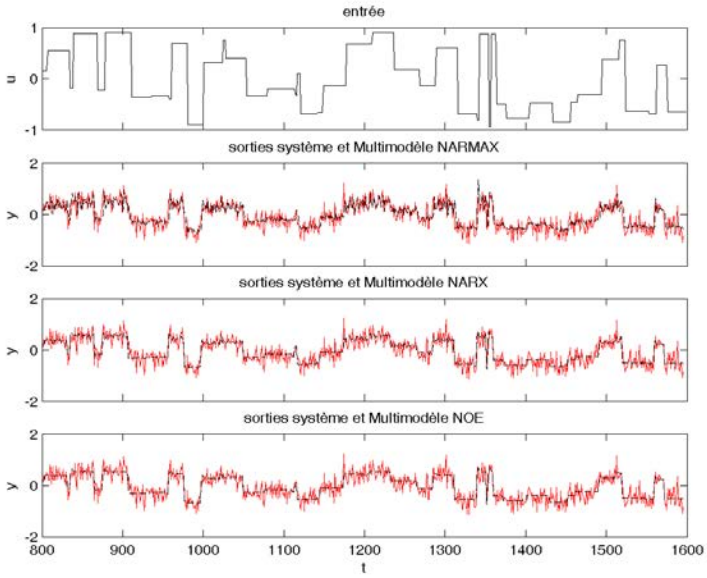

FIG. 3.17 – Identification en présence de bruit de sortie et de bruit d'état : entrée, sortie bruitée du système (trait plein) et sortie des multimodèles (trait discontinu) NARMAX, NARX et NOE en validation.

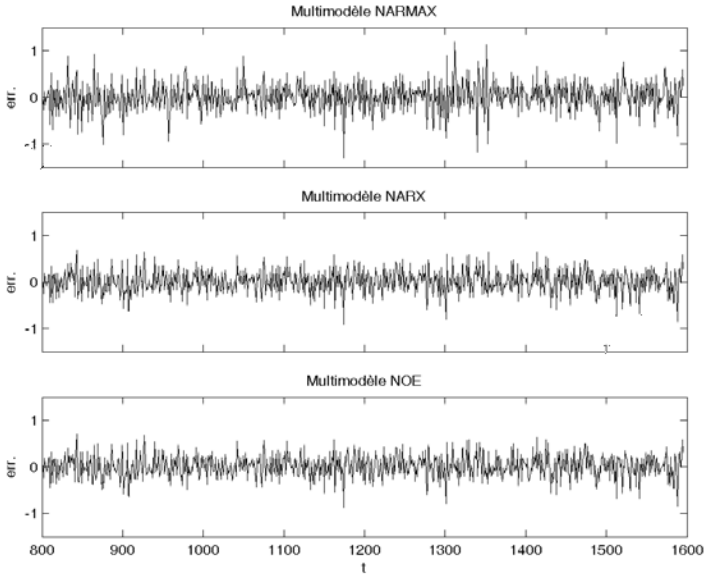

FIG. 3.18 – Influence d'un bruit de sortie et d'un brut d'état : résidus des multimodèles en validation.

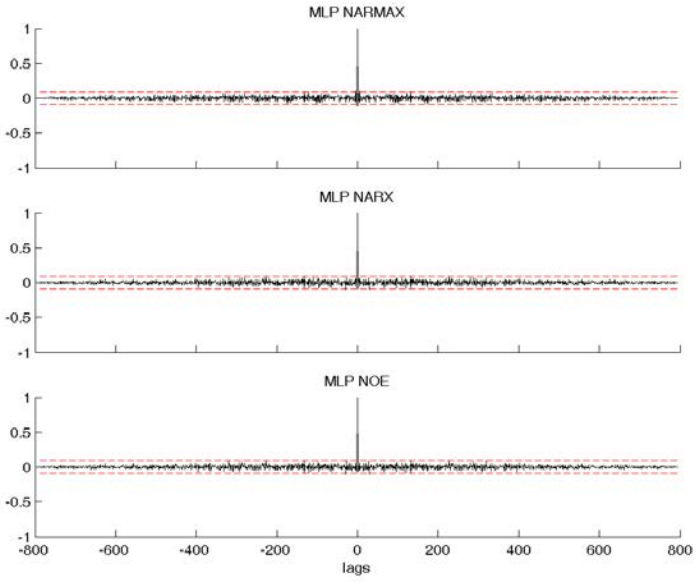

FIG. 3.19 – Fonctions d'autocorrélation des résidus des multimodèles en validation en présence de bruit de sortie et de bruit d'état.

Classe de modèle	Nb. modèles locaux	AIC	*RMSE* app.	*RMSE* valid.
NARMAX	2	-1860	0.304	0.301
NARX	2	-2110	0.261	0.248
NOE	2	-2098	0.263	0.246

TAB. 3.3 – Résultats obtenus pour les multimodèles NOE, NARX et NARMAX en présence de bruit de sortie et de bruit d'état.

3.4.5 Comparaison des modes de partitionnement sur un exemple simulé

La détermination du nombre de modèles locaux et de leur zone de fonctionnement constitue une tâche délicate en identification par multimodèle. L'algorithme présenté au paragraphe §3.4.3 permet de combiner l'estimation paramétrique et l'identification structurelle des zones de validité. Cependant, cette démarche nécessite un choix préalable d'un mode de partitionnement. Nous nous proposons de comparer les performances de multimodèles pour l'identification du système dynamique non linéaire dont la sortie y_s est donnée par la relation :

$$y_s(t+1) = 0.18y_s(t) + 0.3y_s(t-1) + 0.6u(t)^3 + 0.18u(t)^2 - 0.2u(t)$$

où u est l'entrée du système.

Les signaux d'apprentissage sont constitués de créneaux d'amplitudes comprises entre -1 et 1 et de largeur variable.

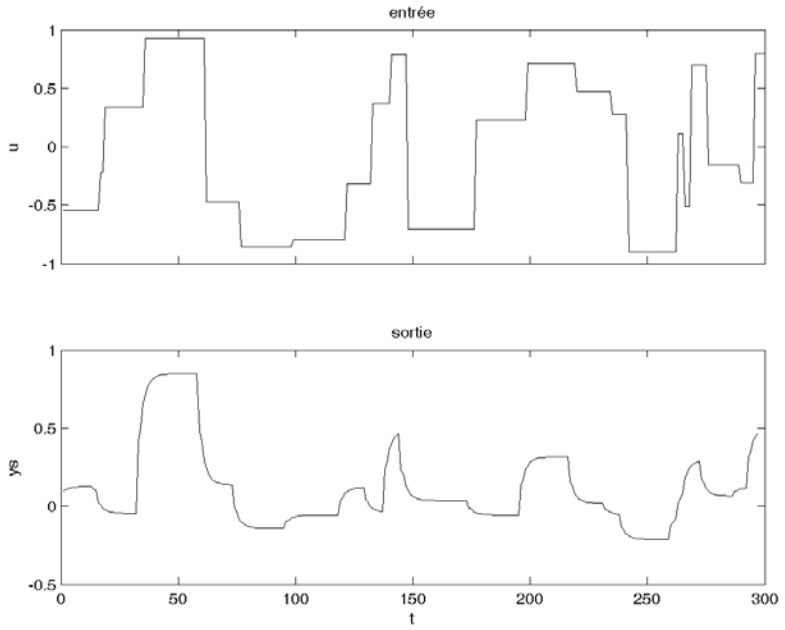

FIG. 3.20 – Signaux d'apprentissage du système dynamique non linéaire simulé.

Le signal de validation est généré à partir de :

$$u(t) = 0.7sin(2\pi t/300)$$

avec $t = 0,\ 1, \ldots, 300$.

Les figures 3.20 et 3.21 représentent l'entrée et la sortie du système utilisées pour l'apprentissage et la validation respectivement.

Nous considérons ici une structure multimodèles NARX avec des modèles locaux affines et un vecteur de régression défini par :

$$\underline{\varphi}(t) = \Big[u(t)\ y_s(t)\ y_s(t-1)\Big]^T$$

Le modèle recherché est donc de la forme :

$$y_s(t+1) = F\big(u(t),\ y_s(t),\ y_s(t-1)\big)$$

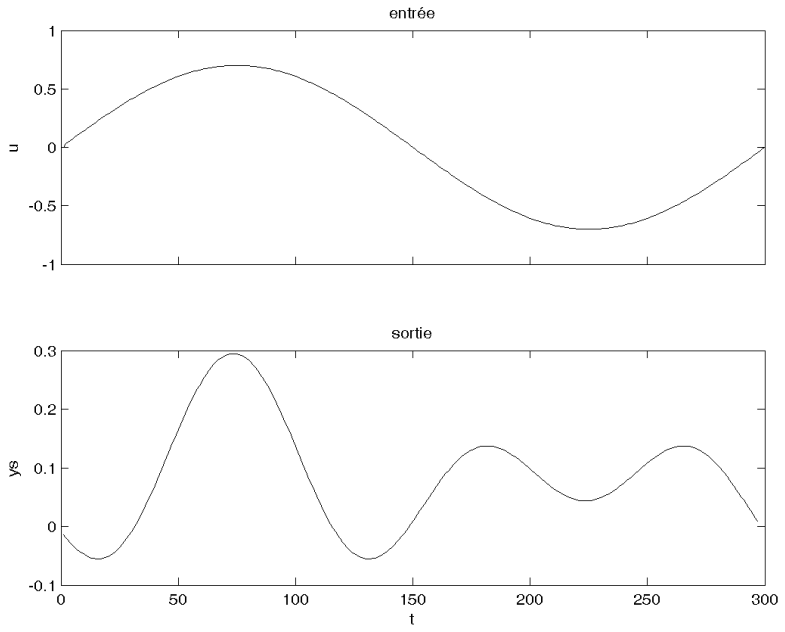

FIG. 3.21 – Signaux de validation du système dynamique non linéaire simulé.

Les multimodèles sont construits en utilisant différents modes de partitionnement : partitionnement hiérarchique orthogonal aux axes (HOA), partitionnement flou par l'algorithme des « *fuzzy-c-means* » (FCM), partitionnement flou par l'algorithme de « *Gustafson & Kessel* » (GK) et partitionnement flou par l'algorithme du « *Subtractive Clustering* » (SC). La variable d'indexation est $\underline{\xi}(t) = u(t)$.

Les résultats obtenus sont présentés au tableau 3.4. Le mode de partitionnement hiérarchique orthogonal aux axes conduit à 2 modèles locaux avec un critère AIC en apprentissage et une $RMSE$ en validation plus faible par rapport aux autres méthodes. Les méthodes de partitionnement par classification floue conduisent à 4 modèles locaux, avec un temps de calcul plus élevé pour l'algorithme de « *Gustafson & Kessel* ».

La figure 3.22 présente les sorties des multimodèles obtenus avec les différents modes de partitionnement. Les résidus obtenus par les multimodèles sont représentés sur la figure 3.23 et les fonctions d'autocorrélation des résidus sur la figure 3.24. Ces deux figures montrent que le résidu n'est pas blanc et qu'une modélisation plus fine est possible.

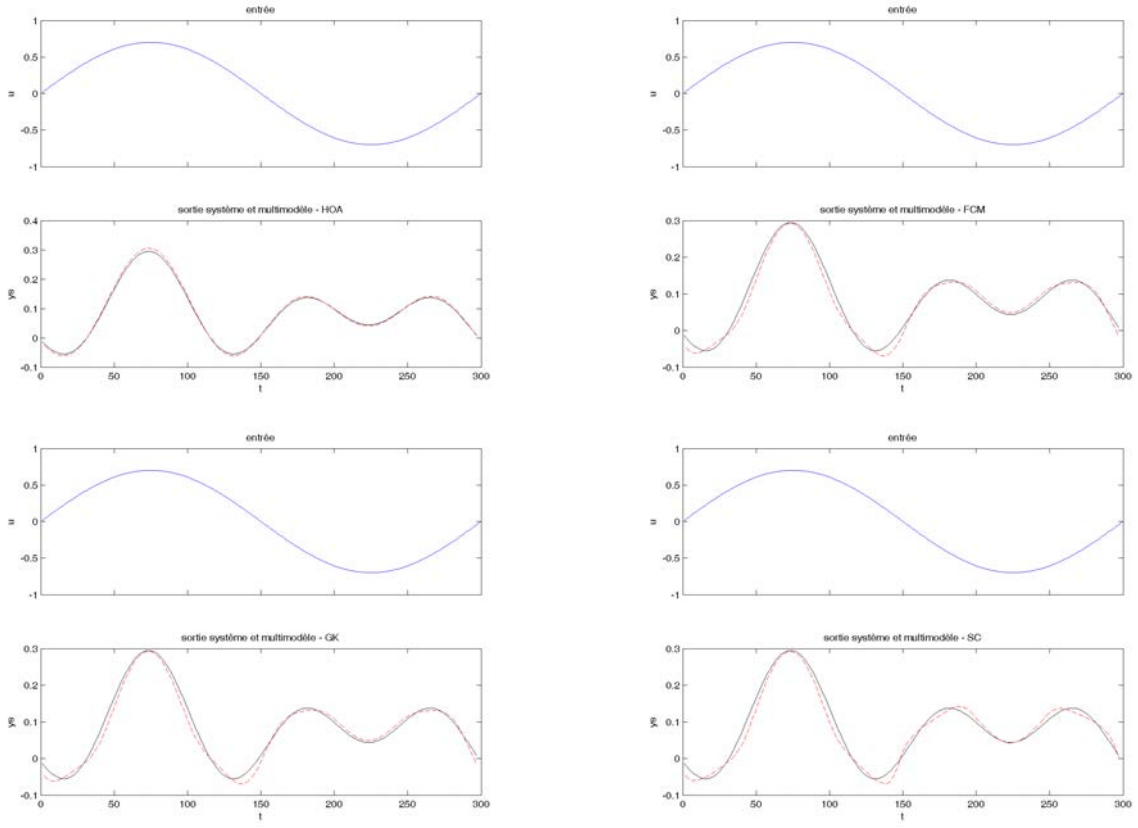

FIG. 3.22 – Sorties des multimodèles obtenus avec différents modes de partitionnement.

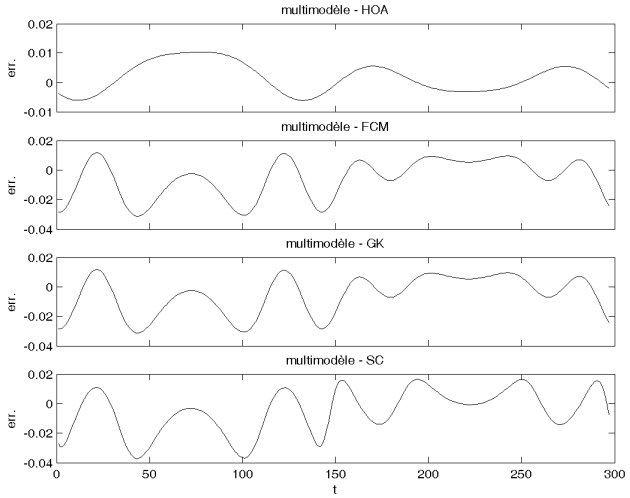

FIG. 3.23 – Résidus obtenus par les multimodèles.

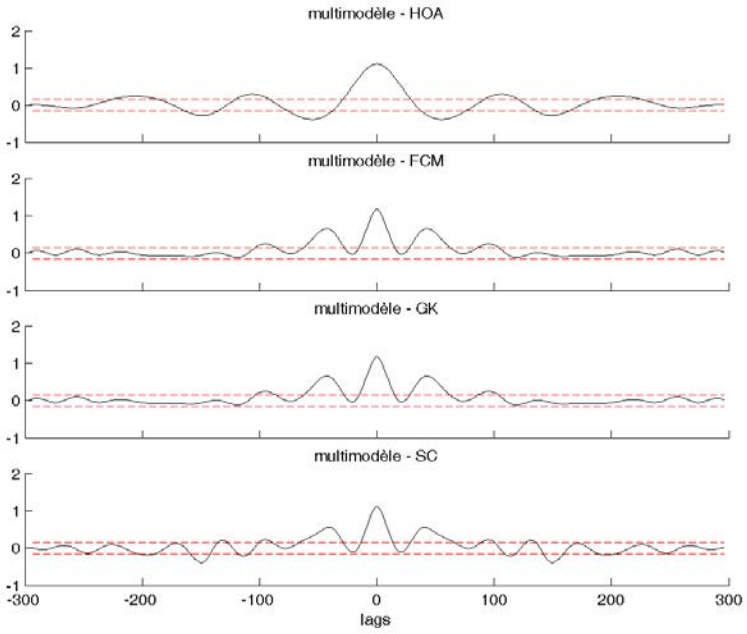

FIG. 3.24 – Fonctions d'autocorrélation des résidus.

Partitionnement	Nb. zones	AIC	*RMSE* app.	*RMSE* valid.	Durée ident. (s)
HOA	2	-3072	0.0054	0.0053	0.03
FCM	4	-2880	0.0073	0.0139	0.11
GK	4	-2880	0.0073	0.0139	0.20
SC	4	-2795	0.0084	0.0156	0.08

TAB. 3.4 – Performances des multimodèles NARX obtenus avec différents modes de partitionnement.

.

3.5 Multimodèles à modèles locaux hétérogènes

Quand le partitionnement de l'espace de fonctionnement du système est fait de façon à regrouper les données qui présentent une certaine « similarité », il est tout à fait naturel que les modèles locaux représentant le système dans les différents sous-espaces soient de structure différentes. Ainsi chaque modèle local, dans sa zone de validité, doit avoir une structure qui s'adapte à la complexité du système dans cette zone. Par exemple, on pourrait avoir des modèles locaux linéaires dans certains groupes de données et des

modèles locaux non linéaires dans d'autres groupes. On peut alors avoir une structure multimodèle avec des modèles locaux polynômiaux de différents degrés, ou une structure avec des modèles locaux polynômiaux et des modèles locaux neuronaux de type MLP à une couche cachée, ou une structure avec des MLP possédant un nombre différent de neurones cachés. Ces structures de multimodèle sont très proches des modèles multi-experts.

3.5.1 Modèles multi-experts

Les modèles multi-experts (« *mixture of experts - ME* ») font partie des modèles basés sur la stratégie de « diviser pour régner », qui consiste à décomposer un problème complexe en plusieurs sous problèmes de complexité moindre. Les applications des modèles multi-experts sont diverses : médecine ([65, 89]), génie logiciel ([90]), etc.

Une architecture multi-experts est constituée de plusieurs réseaux d'experts (« *expert networks* ») et d'un réseau de déclenchement (« *gating network* »). Chaque expert est spécialisé dans une tâche bien définie. Le réseau de déclenchement effectue une combinaison linéaire des sorties des réseaux experts pour produire la sortie du modèle. La figure 3.25 montre l'architecture d'un réseau multi-experts.

La modélisation par modèles multi-experts est basée sur une approche probabiliste. En effet, pour un vecteur des entrées $\underline{u}$ dont la sortie désirée est y_s, la sortie du réseau multi-experts est la probabilité conditionnelle $p(y_s|\underline{u})$ définie par :

$$p(y_s|\underline{u}) = \sum_{i=1}^{M} p(i|\underline{u})\, p(y_s|\underline{u}, i) = \sum_{i=1}^{M} g_i(\underline{u})\, p(y_s|\underline{u}, i) \tag{3.35}$$

où :

i est l'indice de l'expert ;

$g_i(\underline{u}) = p(i|\underline{u})$ est la sortie du réseau de déclenchement associée à l'expert i et correspond à la probabilité de sélectionner la sortie de l'expert i comme sortie du modèle. Les g_i doivent vérifier la condition :

$$\sum_{i=1}^{M} g_i(\underline{u}) = 1$$

La sortie y_i d'un expert i peut être mise sous la forme :

$$y_i(\underline{u}) = p(y_s|\underline{u}, i) = f(\underline{u}, \underline{W}_i) \tag{3.36}$$

où $f(\cdot)$ est une fonction non linéaire, $\underline{W}_i$ est le vecteur des paramètres de l'expert i.

La sortie g_i du réseau de déclenchement peut s'écrire :

$$g_i(\underline{u}) = g(\underline{u}, \underline{V}_i) \tag{3.37}$$

où $g(\cdot)$ est une fonction qui détermine le réseau de déclenchement et $\underline{V}_i$, un vecteur de paramètres du réseau de déclenchement relatif à l'expert i. Le réseau de déclenchement dispose d'autant de sorties qu'il y a d'experts. Chacune de ces sorties joue le rôle de coefficient de pondération associé à la sortie produite par un expert.

La sortie du réseau multi-experts est donc la somme pondérée des sorties de chaque expert et s'écrit :

$$y(\underline{u}, \underline{\theta}) = p(y_s|\underline{u}) = \sum_{i=1}^{M} g(\underline{u}, \underline{V}_i)\, f(\underline{u}, \underline{W}_i) \tag{3.38}$$

où $\underline{\theta} = [\underline{V}_1^T\ \underline{W}_1^T \cdots \underline{V}_i^T\ \underline{W}_i^T \cdots \underline{V}_M^T\ \underline{W}_M^T]^T$ est un vecteur contenant les paramètres des réseaux experts et du réseau de déclenchement.

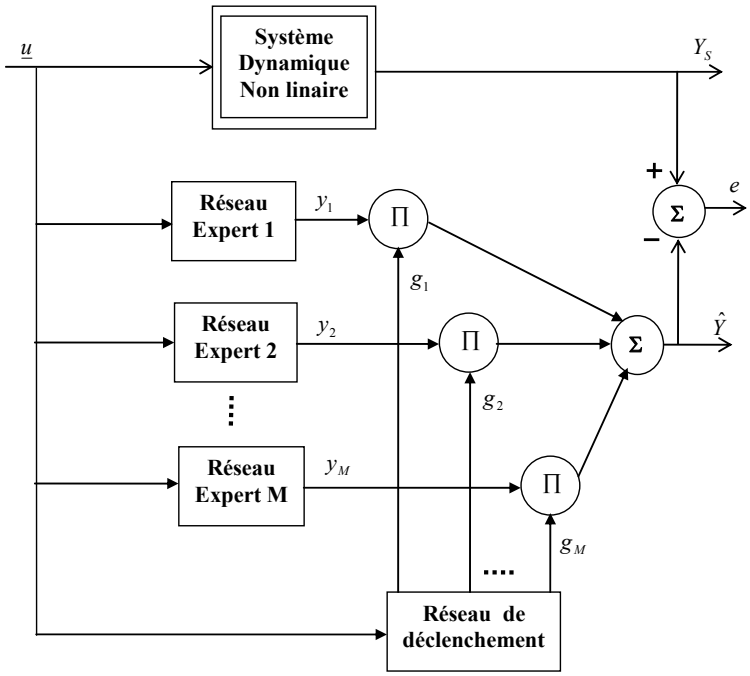

FIG. 3.25 – Architecture d'un réseau multi-experts.

Les experts et le réseau de déclenchement peuvent être des modèles linéaires généralisés ou des modèles non linéaires. Jacobs et al. [91] ont proposé une structure modulaire de réseaux de neurones.

La détermination des paramètres des réseaux experts et du réseau de déclenchement consiste à maximiser la fonction de vraisemblance définie par :

$$L = \prod_{t=1}^{N} p\Big(y_s(t)|\underline{u}(t)\Big) \tag{3.39}$$

où N est la taille de la base de données. Cela revient à minimiser la fonction :

$$\begin{aligned} C = -\ln L &= -\ln \prod_{t=1}^{N} p\left(y_s(t)|\underline{u}(t)\right) \\ &= \sum_{t=1}^{N} -\ln \sum_{i=1}^{M} g_i\left(\underline{u}(t)\right)\, p\left(y_s(t)|\underline{u}(t), i\right) \end{aligned} \tag{3.40}$$

L'estimation des paramètres du modèle multi-experts par la méthode du maximum de vraisemblance (minimisation de la fonction définie par (3.40) s'effectue par l'algorithme itératif « *Expectation-Maximization* » (EM) (voir [92, 93, 94, 95]) qui se résume en deux étapes : l'étape « *Expectation* » (Étape E) et l'étape« *Maximization* » (Étape M). A l'itération s, chacune de ces étapes consiste à :

Étape E : calculer les probabilités

$$h_i(t) = p\left(i|\underline{u}(t), y_s(t)\right) \quad i = 1, \ldots, M$$

qui correspondent aux probabilités que la sortie du réseau pour le couple entrées-sortie désirée $(\underline{u}(t), y_s(t))$ soit produite par l'expert i :

$$h_i(t) = \frac{g\left(\underline{u}(t), \underline{V}_i^{(s)}\right)\, p\left(y_s(t)|\underline{u}(t), \underline{W}_i^{(s)}\right)}{\sum_{k=1}^{M} g\left(\underline{u}(t), \underline{V}_k^{(s)}\right)\, p\left(y_s(t)|\underline{u}(t), \underline{W}_k^{(s)}\right)} \tag{3.41}$$

Étape M : résoudre les problèmes de maximisation suivants :

$$\underline{W}_i^{(s+1)} = \arg\max_{\underline{W}_i} \sum_{t=1}^{N} h_i(t) \ln p\left(y_s(t)|\underline{u}(t), \underline{W}_i\right) \tag{3.42}$$

$$\underline{V}_i^{(s+1)} = \arg\max_{\underline{V}} \sum_{t=1}^{N} \sum_{k=1}^{M} h_k(t) \ln g\left(\underline{u}(t), \underline{V}_k\right) \tag{3.43}$$

où $\underline{V}$ est l'ensemble des paramètres du réseau de déclenchement.

L'algorithme 3.7 représente la méthode « *Expectation-Maximization* ».

Algorithme 3.7 : Algorithme *EM*

Initialiser $\underline{\theta}^{(1)} = [\underline{V}_1^{(1)T}\ \underline{W}_1^{(1)T} \cdots \underline{V}_i^{(1)T}\ \underline{W}_i^{(1)T} \cdots \underline{V}_M^{(1)T}\ \underline{W}_M^{(1)T}]^T$ (vecteur de paramètres des réseaux experts et du réseau de déclenchement) ;

Choisir un critère d'arrêt.

s=1 //compteur du nombre d'itérations

tant que *Critère d'arrêt non satisfait* **faire**

Étape E :

pour $t = 1$ ***à*** N **faire**

pour $i = 1$ ***à*** M **faire**

Calculer

$$h_i(t) = \frac{g\left(\underline{u}(t), \underline{V}_i^{(s)}\right)\ p\left(y_s(t)|\underline{u}(t), \underline{W}_i^{(s)}\right)}{\sum_{k=1}^{M} g\left(\underline{u}(t), \underline{V}_k^{(s)}\right)\ p\left(y_s(t)|\underline{u}(t), \underline{W}_k^{(s)}\right)}$$

fin

fin

Étape M :

pour $i = 1$ ***à*** M **faire**

Résoudre

$$\underline{W}_i^{(s+1)} = \arg\max_{\underline{W}_i} \sum_{t=1}^{N} h_i(t) \ln p\left(y_s(t)|\underline{u}(t), \underline{W}_i\right)$$

$$\underline{V}_i^{(s+1)} = \arg\max_{\underline{V}} \sum_{t=1}^{N} \sum_{k=1}^{M} h_k(t) \ln g\left(\underline{u}(t), \underline{V}_k\right)$$

fin

$s = s + 1$

fin

Cette brève présentation des modèles multi-experts permet de voir leur lien avec les multimodèles. Les réseaux experts correspondent aux modèles locaux avec comme degrés d'activation les sorties du réseau de déclenchement. La structure est donc la même : d'un côté on a une approche déterministe et de l'autre côté une approche probabiliste. Nous proposons au paragraphe suivant une architecture de multimodèle qui est une généralisation des modèles multi-expert où les experts peuvent être des modèles polynômiaux ou/et

neuronaux.

3.5.2 Multimodèles à modèles locaux polynômiaux de degrés différents

La décomposition de l'espace de fonctionnement d'un système en plusieurs sous-espaces, en utilisant par exemple les algorithmes de classification floue présentés au paragraphe §3.3.2.3, permet de déterminer pour chaque groupe de données un modèle local dont la structure est la plus convenable pour y représenter le système. En choisissant une structure polynômiale des modèles locaux, le multimodèle ainsi obtenu est similaire à un modèle multi-experts. Cependant l'apprentissage d'un tel multimodèle est plus facile à implanter que celui du modèle multi-experts présenté au paragraphe §3.5.1. L'utilisation d'un mode d'apprentissage local (voir §3.4.1) permet la spécialisation de chaque modèle local qui joue ainsi le rôle d'expert.

Nous présentons ici une procédure d'identification d'un multimodèle non-récurrent dont les modèles locaux sont des polynômes de degrés différents. La détermination de la structure de chaque modèle local (le degré du polynôme) se fait suivant une procédure itérative dont les principales étapes sont décrites par l'algorithme 3.8 qui peut être combiné aux procédures de détermination du nombre de modèles locaux présentées sur les figures 3.9 et 3.10 pour un multimodèle non-récurrent et un multimodèle récurrent respectivement.

3.5.3 Multimodèles à modèles locaux neuronaux

De même que pour les multimodèles à modèles locaux polynômiaux, la décomposition de l'espace de fonctionnement d'un système non linéaire en plusieurs sous-espaces peut conduire à la nécessité d'implanter des modèles locaux neuronaux qui permettent dans certains cas de mieux appréhender les non linéarités locales. Chaque modèle local est alors un réseau MLP dont l'architecture est déterminée par la complexité du système dans la zone de validité du modèle local [10]. On obtient donc des réseaux MLP avec différents neurones cachés. Afin de permettre la spécialisation des modèles locaux, l'apprentissage est basé sur un critère local où l'erreur entre la sortie du système et celle de chaque modèle local est minimisée. Dans chaque zone de l'espace de fonctionnement du système, un réseau MLP représente presque entièrement le système, la contribution des

Algorithme 3.8 : Algorithme d'identification structurelle et paramétrique d'un multimodèle hétérogène à modèles locaux polynômiaux

Fixer le nombre M de modèles locaux (tous les modèles locaux sont affines) ;
Fixer la valeur maximale $E_{rrL_{max}}$ de l'erreur quadratique moyenne entre la sortie de chaque modèle local et celle du système dans les zones de validité des modèles locaux.
Initialiser les paramètres $\underline{\hat{\theta}}_i$ en utilisant (3.32) et calculer les erreurs locales E_{rrL_i}.
Calculer la valeur initiale du critère de sélection du modèle $AIC(0)$.

pour $i = 1$ ***à*** M **faire**
  $s = 1$ //initialisation compteur d'itérations
  tant que $E_{rrL_i} > E_{rrL_{max}}$ *et* $AIC(s) < AIC(s-1)$ **faire**
    Incrémenter le degré du polynôme du modèle local i.
    Former la matrice de régression Φ_{eg} en tenant compte du degré du polynôme du modèle local i (voir relation (3.5)).
    Calculer le vecteur de paramètre en utilisant (3.32) :
    $\underline{\hat{\theta}}_i = (\Phi_{eg}^T W_i \Phi_{eg})^{-1}(\Phi_{eg}^T W_i Y_s)$
    Calculer la sortie $\underline{\hat{y}}_i$ du modèle local i.
    Calculer l'erreur quadratique moyenne du modèle local i :

$$E_{rrL_i} = \frac{1}{2}\sum_{t=1}^{N} \omega_i \left(y_s(t) - \hat{y}_i(t)\right)^2$$

    Calculer $AIC(s)$.
    $s = s + 1$ //incrémentation compteur d'itérations
  fin
fin

autres réseaux étant très faible voire nulle.

L'apprentissage d'une telle architecture multimodèle peut être effectué suivant l'une ou l'autre des deux configurations suivantes :

– S'il n'y a pas de recouvrement entre les zones de validité des modèles locaux, la sortie du modèle à un instant donné est celle du modèle local dont la zone de validité contient l'observation courante. La sortie des autres modèles locaux est alors nulle. Ainsi la sortie du multimodèle est une concaténation des sorties des modèles locaux [10]. Le principe de l'apprentissage pour une telle architec-

ture multimodèle est représenté sur la figure 3.26. La base d'apprentissage est décomposée en plusieurs parties, chacune servant à l'apprentissage d'un réseau.

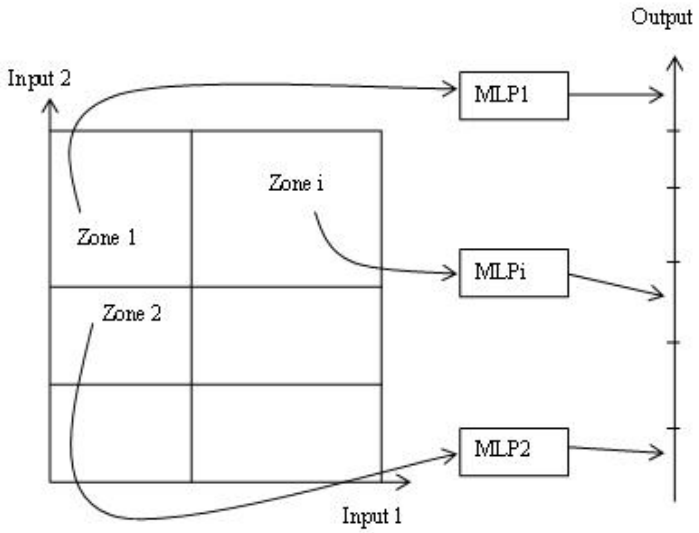

FIG. 3.26 – Principe d'apprentissage d'un multimodèle avec des modèles locaux neuronaux en absence de recouvrement entre les zones.

– S'il y a recouvrement entre les zones de validité des modèles locaux, la base d'apprentissage de chaque réseau MLP est constituée en fonction des degrés d'appartenance des données à la zone de validité du MLP. De la base de donnée globale, ne seront retenues que les données telles que les degrés d'appartenance ω sont supérieurs à un seuil ω_{seuil} (par exemple $\omega_{seuil} = 0.05$). Cette précaution permet de limiter la taille des réseaux MLP pour qu'ils n'apprennent pas des données pour lesquelles leurs sorties sont quasi nulles. Ce principe est illustré sur la figure 3.27 pour un système comportant trois groupes de données. La sortie du multimodèle est la somme pondérée des sorties des réseaux MLP. L'algorithme 3.9 représente la procédure d'identification structurelle d'une telle architecture multimodèle.

3.5.4 Multimodèles à modèles locaux polynômiaux et neuronaux

Il s'agit ici de mettre en place une structure multimodèles analogue à la structure d'un modèle multi-experts mais dont les réseaux experts peuvent être des polynômes ou des MLP à une couche cachée. La difficulté est de déterminer le nombre de modèles locaux, le degré de chaque polynôme ou le nombre de neurones cachés pour les MLP.

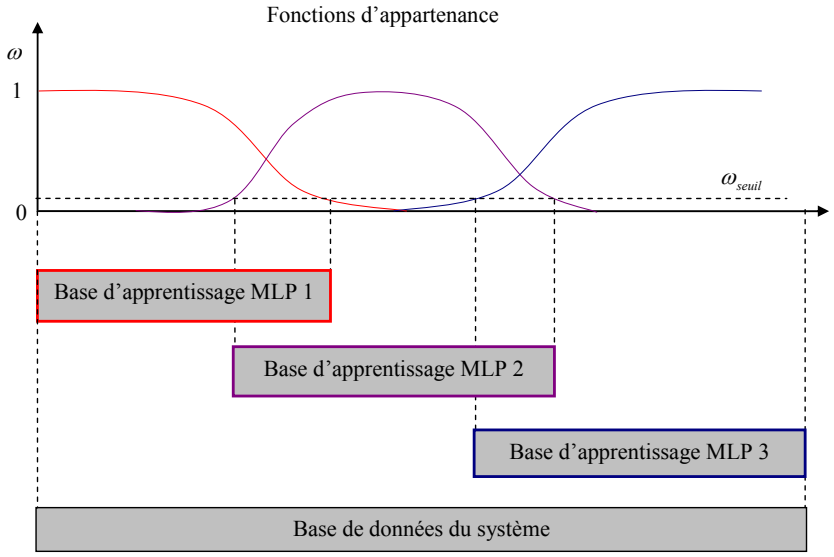

FIG. 3.27 – Principe d'apprentissage d'un multimodèle avec des modèles locaux neuronaux en présence de recouvrement entre les zones.

La procédure d'identification structurelle et paramétrique d'une telle architecture multimodèle est présentée sur l'algorithme 3.10. Le principe est de rechercher un multimodèle hétérogène à modèles locaux polynômiaux suivant l'algorithme 3.8 et de remplacer les modèles locaux dont les performances ne sont pas satisfaisantes par des MLP.

Ici aussi, il est possible de combiner l'algorithme 3.10 à la procédure de détermination du nombre de modèles locaux présentée sur la figure 3.9 pour un multimodèle non-récurrent.

3.5.5 Exemple d'identification d'un système dynamique non linéaire par un multimodèle à modèles locaux hétérogènes

Soit le système non linéaire statique défini par [96] :

$$y(x) = \begin{cases} (x+2)^2 & x \in [-2,\ 0] \\ \sin(x) + \sin(2x) + \sin(6x) + 4 & x \in [0,\ 3] \end{cases} \tag{3.44}$$

Nous nous proposons d'identifier un modèle capable d'approcher au mieux la sortie y du système connaissant l'entrée x. Le modèle recherché est de la forme :

$$y(x) = F(x)$$

Algorithme 3.9 : Algorithme d'identification structurelle et paramétrique d'un multimodèle hétérogène à modèles locaux neuronaux de type MLP à une couche cachée

Initialisation : fixer le nombre M de modèles locaux MLP, le nombre de neurones cachés pour chaque MLP est initialement fixé à 1 ; fixer la valeur maximale $E_{rrL_{max}}$ de l'erreur quadratique moyenne entre la sortie de chaque MLP et celle du système dans les zones de validité des modèles locaux ; déterminer les paramètres des réseaux MLP et calculer les erreurs locales E_{rrL_i}.
Calculer la valeur initiale du critère de sélection du modèle $AIC(0)$.

pour $i = 1$ ***à*** M **faire**
  si $E_{rrL_i} > E_{rrL_{max}}$ **alors**
    $s = 1$ //initialisation compteur d'itérations
    tant que $E_{rrL_i} > E_{rrL_{max}}$ *et* $AIC(s) < AIC(s-1)$ **faire**
      Incrémenter le nombre de neurones cachés du MLP.
      Déterminer les paramètres du MLP.
      Calculer la sortie $\underline{\hat{y}}_i$ du MLP.
      Calculer l'erreur $E_{rrL_i} = \frac{1}{2}\sum_{t=1}^{N} \omega_i \left(y_s(t) - \hat{y}_i(t)\right)^2$
      Calculer $AIC(s)$.
      $s = s + 1$
    fin
  fin
fin

Les performances de multimodèles à modèles locaux affines seront comparées à celles de multimodèles à modèles locaux hétérogènes .

La décomposition de l'espace de fonctionnement est faite avec le mode de partitionnement flou par l'algorithme du « *subtractive clustering* ». En utilisant les paramètres par défaut définis par (3.24), l'algorithme détecte 3 groupes de données avec comme centres :

$$\underline{c}_1 = \begin{pmatrix} -1.2 \\ 0.64 \end{pmatrix} \quad \underline{c}_2 = \begin{pmatrix} 0.5 \\ 5.46 \end{pmatrix} \quad \underline{c}_3 = \begin{pmatrix} 2.2 \\ 4.45 \end{pmatrix}$$

La sortie du système ainsi que les groupes de données obtenus sont représentés sur la figure 3.28.

Pour cette configuration de l'espace de fonctionnement du système, nous identifions les multimodèles à modèles locaux affines et à modèles locaux hétérogènes. Les algorithmes présentés dans les paragraphes §3.5.2, et §3.5.4 sont utilisés pour l'identification des architectures des multimodèles hétérogènes. Le tableau 3.5 résume les résultats obtenus.

Algorithme 3.10 : Algorithme d'identification structurelle et paramétrique d'un multimodèle hétérogène à modèles locaux polynômiaux et neuronaux

Initialisation : fixer le nombre M de modèles locaux, leur structure étant initialement affine ; fixer la valeur maximale $E_{rrL_{max}}$ de l'erreur quadratique moyenne entre la sortie de chaque modèle local et celle du système dans les zones de validité des modèles locaux ; initialiser les paramètres $\underline{\hat{\theta}}_i$ en utilisant (3.32).

Calculer les erreurs locales E_{rrL_i} et le critère $AIC(0)$.

Étape 1 : Multimodèle hétérogène à modèles locaux polynômiaux

pour $i = 1$ ***à*** M **faire**

  $s = 1$ //initialisation compteur d'itérations

  tant que $E_{rrL_i} > E_{rrL_{max}}$ *et* $AIC(s) < AIC(s-1)$ **faire**

    Incrémenter le degré du polynôme du modèle local i.

    Former la matrice de régression Φ_{eg} en tenant compte du degré du polynôme du modèle local i (voir (3.5)).

    Calculer le vecteur de paramètres (3.32) : $\underline{\hat{\theta}}_i = (\Phi_{eg}^T W_i \Phi_{eg})^{-1}(\Phi_{eg}^T W_i Y_s)$

    Calculer la sortie $\underline{\hat{y}}_i$ du modèle local i.

    Calculer : $E_{rrL_i} = \frac{1}{2}\sum_{t=1}^{N} \omega_i \left(y_s(t) - \hat{y}_i(t)\right)^2$ et $AIC(s)$

    $s = s + 1$

  fin

fin

Étape 2 : Multimodèle hétérogène à modèles locaux polynômiaux et neuronaux

pour $i = 1$ ***à*** M **faire**

  si $E_{rrL_i} > E_{rrL_{max}}$ **alors**

    Remplacer la structure polynômiale du modèle locale i par un MLP à 1 neurone caché.

    Déterminer les paramètres du MLP (par rétro-propagation du gradient) et calculer l'erreur locale E_{rrL_i} et $AIC(0)$.

    $s = 1$ //initialisation compteur d'itérations

    tant que $E_{rrL_i} > E_{rrL_{max}}$ *et* $AIC(s) < AIC(s-1)$ **faire**

      Incrémenter le nombre de neurones cachés du MLP et estimer les paramètres.

      Calculer la sortie $\underline{\hat{y}}_i$ du MLP.

      Calculer : $E_{rrL_i} = \frac{1}{2}\sum_{t=1}^{N} \omega_i \left(y_s(t) - \hat{y}_i(t)\right)^2$ et $AIC(s)$

      $s = s + 1$

    fin

  fin

fin

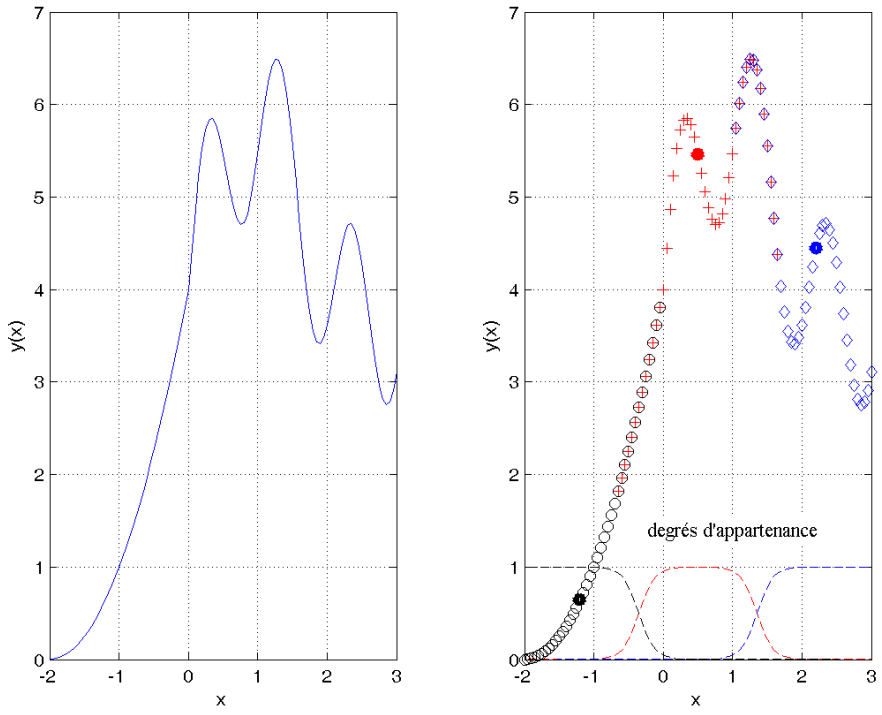

FIG. 3.28 – Sortie du système non linéaire statique et groupes de données obtenus avec l'algorithme du « *subtractive clustering* »

.

Les sorties obtenues par les différents multimodèles sont présentées sur la figure 3.29. La figure 3.30 représente les résidus obtenus des différents modèles.

Structure des modèles locaux	n_θ	AIC	*RMSE*
Affine (3)	10	-113	0.52
Hétérogène (3 poly.)	18	-203	0.31
Hétérogène (3 MLPs)	26	-727	0.02
Hétérogène (1 poly.+ 2 MLPs)	25	-656	0.03

TAB. 3.5 – Performances des multimodèles à modèles locaux affines et hétérogènes pour 3 zones identifiées. n_θ représente le nombre de paramètres du modèle (paramètres des modèles locaux et de leurs zones de validité).

.

Le multimodèle à modèles locaux hétérogènes polynômiaux obtenu est constitué de :

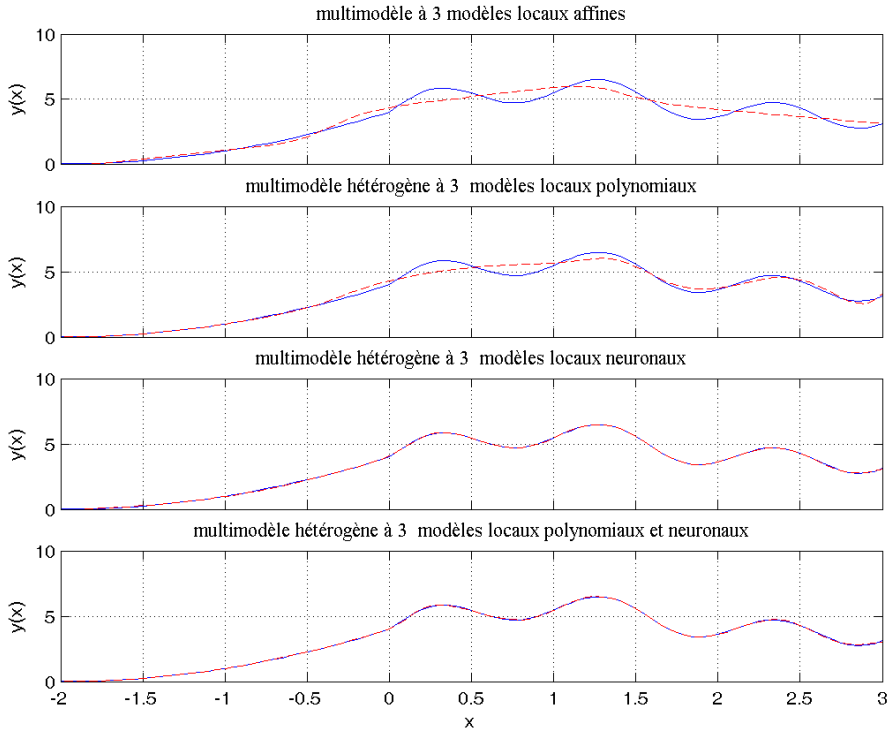

FIG. 3.29 – Sortie du système (trait plein), du multimodèle à modèles locaux affines et des multimodèles à modèles locaux hétérogènes (trait discontinu).

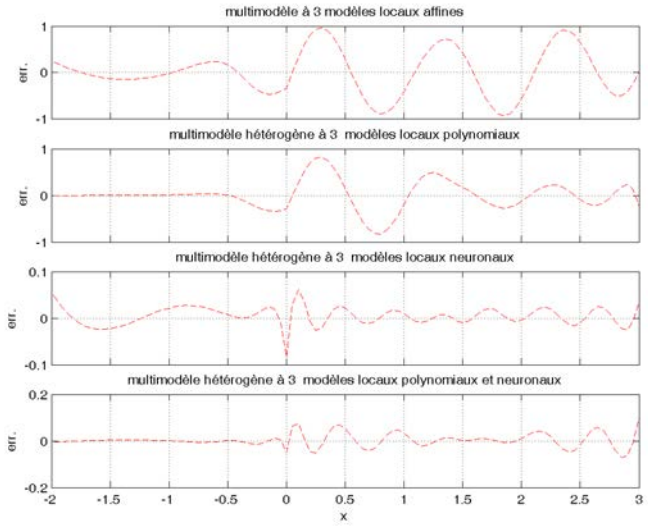

FIG. 3.30 – Résidus des modèles identifiés.

1 polynôme de degré 2 pour le premier groupe de données, 1 polynôme de degré 3 pour le deuxième groupe de données et 1 polynôme de degré 6 pour le troisième groupe de données.

Le multimodèle à modèles locaux hétérogènes de type MLP à une couche cachée est constitué de : 1 MLP à 1 neurone caché pour le premier groupe de données, 1 MLP à 4 neurones cachés pour le deuxième groupe de données et 1 MLP à 3 neurones cachés pour le troisième groupe de données.

Le multimodèle à modèles locaux hétérogènes polynômiaux et neuronaux est constitué de : 1 polynôme de degré 2 pour le premier groupe de données, 1 MLP à 4 neurones cachés pour le deuxième groupe de données et 1 MLP à 3 neurones cachés pour le troisième groupe de données.

Les résultats montrent que le degré du polynôme dans le premier groupe de données (voir (3.44)) est détecté par les architectures multimodèles comportant des modèles locaux polynômiaux et que les non linéarités dans les deuxième et troisième groupes de données sont mieux appréhendées par des MLP à 4 et 3 neurones cachés respectivement.

Le mode de partitionnement flou par « *subtractive clustering* » permet de déterminer de façon naturelle des groupes de données. On s'aperçoit à travers cet exemple que le multimodèle à modèles locaux affines ne permet pas d'expliquer le système dans les zones identifiées, alors que cela est possible avec une structure hétérogène des modèles locaux. Les meilleurs résultats sont obtenus avec le multimodèle à modèles locaux neuronaux et avec le multimodèle combinant des modèles locaux polynômiaux et des modèles locaux neuronaux.

3.6 Conclusion

Dans ce chapitre, l'approche multimodèles a été présentée en détail. L'objectif est de pouvoir représenter un système complexe par un ensemble de modèles simples à validité limité dans des zones bien définies. L'identification de systèmes par multimodèles est constituée de deux étapes : l'identification structurelle pour la détermination du nombre de modèles locaux et de leur structure, et l'estimation paramétrique des modèles locaux et de leur zones de validité. Des modèles locaux de structure polynômiale de degré $p \geq 1$ ont été proposés. L'estimation paramétrique peut alors se faire suivant une optimisation linéaire qui est plus économique en temps de calcul que l'estimation paramétrique utilisée

dans les réseaux de neurones. L'usage des modèles locaux de structure polynômiale permet de mieux appréhender les nonlinéarités locales, réduisant ainsi le nombre de sous-modèles. Cette structure de multimodèle constitue l'un des apports de ce travail de thèse.

Différents modes de partitionnement de l'espace de fonctionnement de systèmes ont également été présentés. Le partitionnement en grille et le partitionnement hiérarchique orthogonal aux axes sont des méthodes déterministes pour lesquelles les frontières entre les différentes zones de validité des modèles locaux sont définies. La principale difficulté réside dans la détermination des points de découpage des partitions. Le mode de partitionnement flou par « fuzzy-c-means », « Gustafson et Kessel » ou « subtractive clustering » permet de contourner le problème des points de découpage des partitions. Ces algorithmes permettent de regrouper les données qui présentent une certaine « similarité ». Un modèle local est alors déterminé pout chaque groupe de données.

Ce chapitre présente également l'implantation des multimodèles récurrents, une autre contribution apportée dans ce travail de thèse et qui a fait l'objet d'une publication dans une revue internationale [8]. Cette architecture de multimodèle permet de représenter des modèles récurrents tels que les modèles NOE ou les modèles NARMAX par exemple. Des algorithmes d'apprentissage de ces multimodèles ont été présentés, de même qu'un algorithme d'identification structurelle et paramétrique des multimodèles hétérogènes constitués de modèles locaux de structures différentes (polynômiaux de degrés différents, neuronaux ou polynômiaux et neuronaux). Cette architecture correspond à celle des modèles multi-experts. L'utilisation d'une structure hétérogène de modèles locaux permet de mieux appréhender les non linéarités locales et donc de réduire le nombre de modèles locaux du multimodèle.

La chapitre suivant est consacré à une étude comparative entre une structure multimodèle et un réseau de neurones de type MLP, pour l'identification de systèmes dynamiques non linéaires.

Chapitre 4

Etude comparative des architectures multimodèle et MLP

Sommaire

4.1 Introduction

Ce chapitre traite d'une étude comparative entre les architectures MLP et multimodèle. En raison de la dualité entre l'identification de systèmes et la prédiction [68], les différents modèles de représentation de systèmes dynamiques non linéaires étudiés dans les chapitres précédents sont utilisées ici dans des applications de prédiction. Plusieurs auteurs ont en effet utilisés les réseaux de neurones pour la prédiction de comportement de processus (dérive, évolution, etc.) : prédiction de l'indice boursier [35], prédiction de la consommation d'énergie électrique [36], prédiction hydrologique [97], prédiction à court terme de l'intensité d'orages [61], etc. A travers différents exemples, les performances des multimodèles sont étudiées, notamment dans des applications de prédiction. Les méthodologies développées dans les chapitres précédents sont implémentées au sein d'un outil logiciel d'identification de systèmes dynamiques non linéaires utilisé ici pour l'étude des systèmes.

Dans la première partie, après une description sommaire de l'outil logiciel développé, un exemple académique est utilisé pour la comparaison des architectures MLP et multimodèle. Différentes structures de modèles non linéaires sont étudiées en tenant compte de l'influence du bruit sur le système (bruit d'état, bruit de sortie et bruit d'état et de sortie). Dans la deuxième partie du chapitre, les architectures MLP et multimodèle sont utilisées pour l'identification du système de Box et Jenkins en vue de prédire la concentration de CO_2 résultant de la combustion dans un four d'un mélange de méthane et d'air. La troisième et dernière partie du chapitre est consacrée au problème de modélisation de débit sur le fleuve Sénégal. Les architectures MLP et multimodèle sont utilisées pour la prédiction de débit du fleuve sur un horizon de 1 à plusieurs jours.

4.2 Description de l'outil logiciel d'identification de systèmes dynamiques non linéaires

Les méthodologies d'identification structurelle et paramétrique étudiées dans les chapitres précédents ont été implémentées dans une boîte à outils sous Matlab 6.5 en utilisant une programmation orientée objet. La structure générale de la boîte à outils est présentée sur la figure 4.1 et décrite plus en détails en annexe B.

Une interface utilisateur facilitant l'utilisation du logiciel est également proposée (fi-

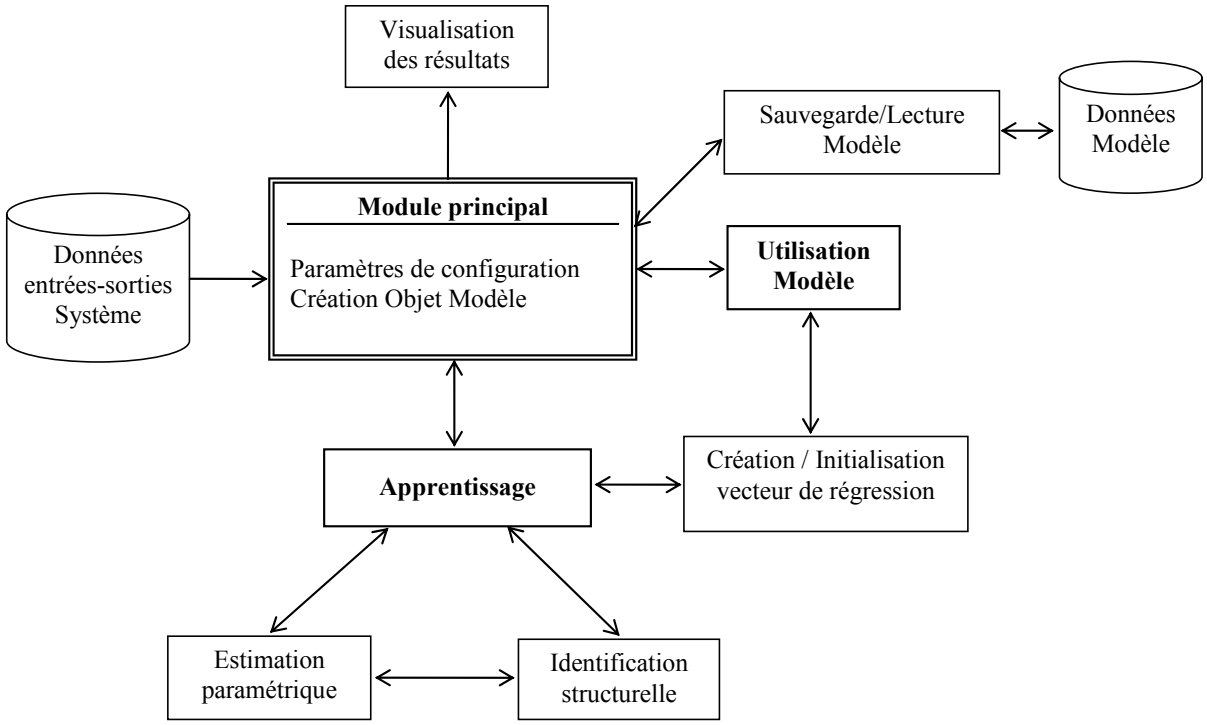

FIG. 4.1 – Structure de l'outil logiciel d'identification de systèmes dynamiques non linéaires.

gure 4.2). Les principales informations requises pour réaliser l'identification d'un système sont :

- la spécification du fichier contenant les données entrées-sortie du système ;
- la spécification de la classe de modèle non linéaire (NFIR, NARX, NOE, NARMAX) ;
- la spécification des ordres du modèle ;
- la structure des modèles locaux et le mode de décomposition des zones de fonctionnement (pour un multimodèle)
- les performances à atteindre.

Les méthodes d'estimation paramétrique sont choisies en fonction de la classe de modèle non linéaire indiquée. Si la structure du modèle n'est pas précisée, l'estimation paramétrique est faite conjointement à l'identification structurelle du modèle, en utilisant un critère de sélection de modèle basé sur une estimation de la parcimonie. Comme pour les données du système, les modèles identifiés peuvent être stockés pour une future utilisation.

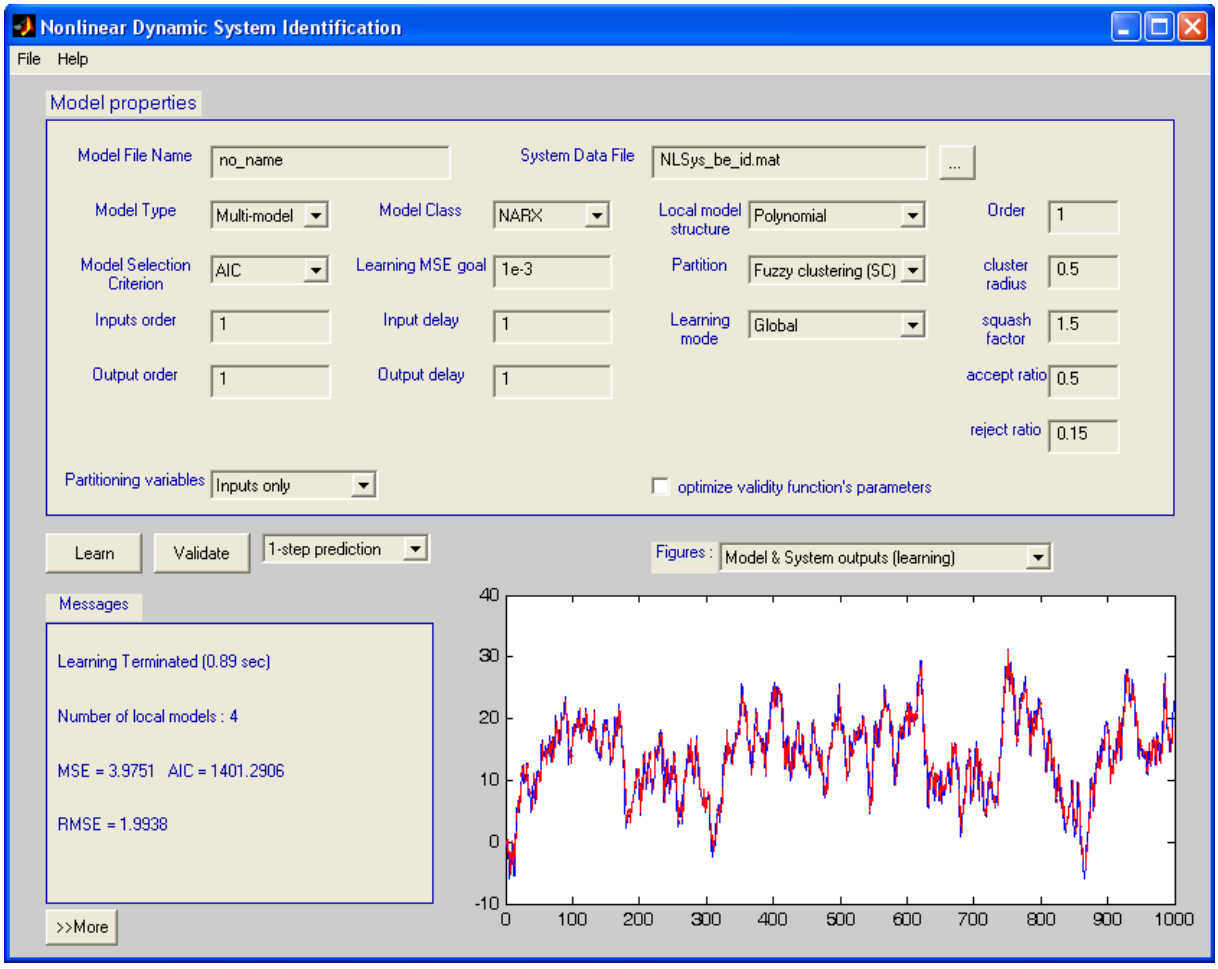

FIG. 4.2 – Interface utilisateur du logiciel d'identification de systèmes dynamiques non linéaires.

4.3 Comparaison d'un multimodèle et d'un MLP sur un exemple simulé

Dans cette partie, la performance d'un multimodèle est comparée à celle d'un réseau de neurones MLP à travers un problème d'identification d'un système dynamique non linéaire proposé dans [98] :

$$\begin{aligned} x(k+1) &= A\,x(k) + \sin\Big(\gamma\,u(k)\Big)\Big(\beta - u(k)\Big) \\ y_s(k) &= x(k) \end{aligned} \tag{4.1}$$

où :

$u(\cdot)$ est l'entrée du système, constituée de créneaux d'amplitudes variables (entre

0 et 1) et de largeurs variables ;

$A = 0.95$, $\gamma = 0.8\,\pi$ et $\beta = 1.5$.

Trois cas de figures sont étudiés ici : présence de bruit d'état, présence de bruit de sortie, et présence de bruit d'état et de sortie. Les systèmes bruités ainsi obtenus sont identifiés avec les modèles les mieux adaptés (voir §1.4) : un modèle NARX lorsque le bruit d'état est présent, un modèle NOE lorsque le bruit de sortie est présent, et un modèle NARMAX lorsque le bruit d'état et le bruit de sortie sont présents. Ces différents modèles sont implantés avec une structure multimodèle et avec une structure neuronale. Dans cet exemple, un multimodèle avec des modèle locaux affines (MM1), un multimodèle avec des modèle locaux polynômiaux de dégré 2 (MM2) et un MLP à une couche cachée sont utilisés. Les signaux d'identification sont générés pour t variant de 0 à 2000. La première moitié est utilisée pour l'apprentissage et la deuxième pour la validation.

Les performance des modèles sont comparées à partir du critère de sélection AIC décrit au paragraphe §1.5.2, par la capacité de généralisation mesurée par l'erreur quadratique moyenne de généralisation (RMSE) et par l'efficacité de l'algorithme d'apprentissage mesurée par la durée de la procédure d'identification. L'architecture des multimodèles est déterminée par les organigrammes des figures 3.9 ou 3.10 suivant que le modèle soit récurrent ou non, et celle des MLP par l'organigramme de la figure 2.8. Cette architecture est caractérisée par un nombre entier (désigné dans les tableaux par un paramètre c) correspondant au nombre de neurones cachés dans le cas d'une structure MLP et au nombre de modèles locaux dans le cas d'une structure multimodèle. Le nombre de paramètres des modèles est désignés par n_θ. Pour les multimodèles, ce nombre correspond au nombre de paramètres des modèles locaux additionné au nombre de paramètres relatifs à leurs zones de validité (qui dépend de la méthode de décomposition de l'espace caractéristique du système). Pour les MLP, n_θ correspond au nombre de poids (coefficients synaptiques) et de biais dans le réseau. La durée de la procédure d'identification d'un réseau MLP dépend naturellement du nombre d'époques[1] spécifiés dans l'algorithme d'apprentissage. Cependant, à partir d'une certaine valeur du nombre d'époques, les performances du réseau ne s'améliorent plus. Ce nombre est alors choisi pour tester l'efficacité de l'algorithme d'apprentissage du réseau.

Les modèles recherchés sont de la forme :

$$\hat{y}(t+1) = F\Big(\underline{\varphi}(t)\Big) \tag{4.2}$$

[1]Nombre d'itérations pour le calcul des paramètres du réseau. A chaque itération, l'ensemble de la base d'apprentissage est présenté au réseau.

où $\underline{\varphi}(t)$ est le vecteur de régression définie selon la classe du modèle utilisé.

4.3.1 Présence de bruit de sortie

En présence de bruit de sortie, l'équation du système (4.1) se met sous la forme :

$$\begin{aligned} x(t+1) &= A\,x(t) + \sin\Big(\gamma\,u(t)\Big)\Big(\beta - u(t)\Big) \\ y_s(t) &= x(t) + e(t) \end{aligned} \tag{4.3}$$

où $e(\cdot)$ est un bruit blanc généré à partir d'une distribution normale de moyenne $\mu = 0$ et de dispersion $\sigma = 1.5$. Le rapport signal/bruit est de 18.3 dB. La figure 4.3 représente les signaux d'entrée et de sortie utilisés pour l'apprentissage en présence de bruit de sortie.

Le système (4.3) est identifié par un modèle NOE avec des structures MLP et multimodèles. Le vecteur de régression utilisé pour chaque structure est :

– pour la structure MLP NOE :

$$\underline{\varphi}_{MLP}(t) = [u(t) \quad \hat{y}(t)]^T$$

– pour la structure MM1 NOE :

$$\underline{\varphi}_{MM1}(t) = [u(t) \quad \hat{y}(t)]^T$$

– pour la structure MM2 NOE :

$$\underline{\varphi}_{MM2}(t) = [u(t) \quad \hat{y}(t) \quad u(t)\hat{y}(t) \quad u^2(t) \quad \hat{y}^2(t)]^T$$

Les résultats obtenus sont présentés au tableaux 4.1 et montrent que les multimodèles MM1 NOE et MM2 NOE ont la même capacité de généralisation que la structure MLP NOE. La durée de l'identification est plus courte pour les multimodèles. Les sorties du système et des modèles NOE en validation sont représentées sur la figure 4.4. Les résidus obtenus sont présentés sur la figure 4.5 et les fonctions d'autocorrélation des résidus sur la figure 4.6 avec des intervalles de confiance de 99%. La figure 4.6 présage que le bruit n'est pas tout à fait blanc et qu'une modélisation plus fine aurait été possible.

Les résultats obtenus pour chaque structure de modèle sont détaillés ici.

Pour le multimodèle NOE à modèles locaux affines, l'algorithme d'identification structurelle présenté sur la figure 3.10, combiné à un mode de décomposition de l'espace caractéristique du système par l'algorithme des « *fuzzy-c-means* », a permis d'obtenir 5

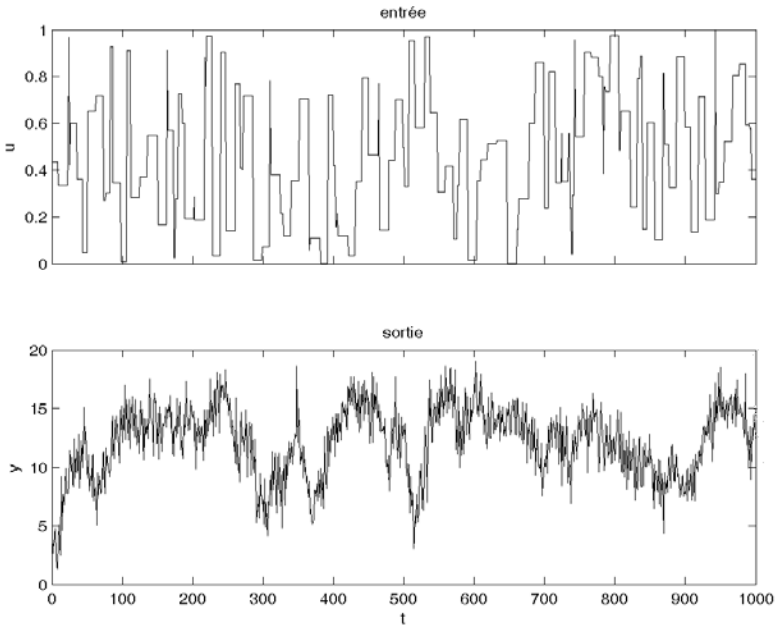

FIG. 4.3 – Signaux d'identification du système en présence de bruit de sortie.

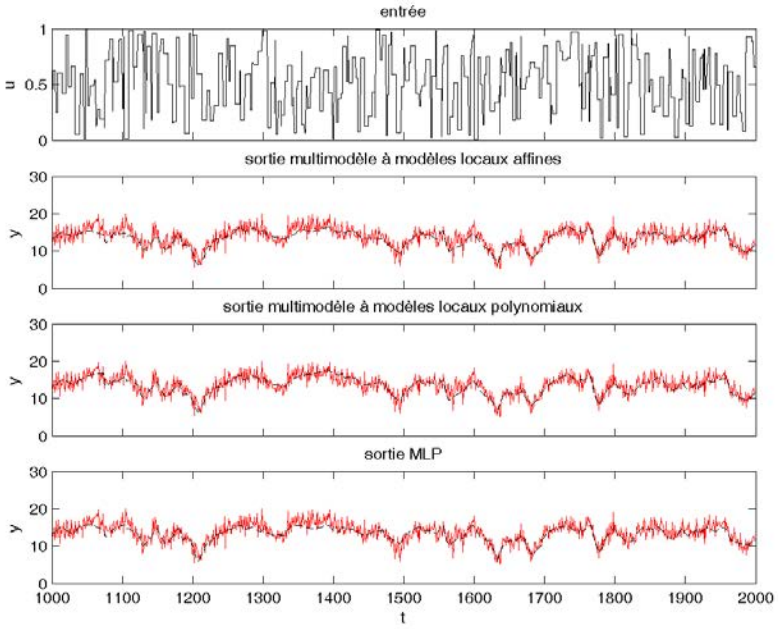

FIG. 4.4 – Entrée, sortie du système (trait plein) et sorties des modèles NOE (trait discontinu) en validation.

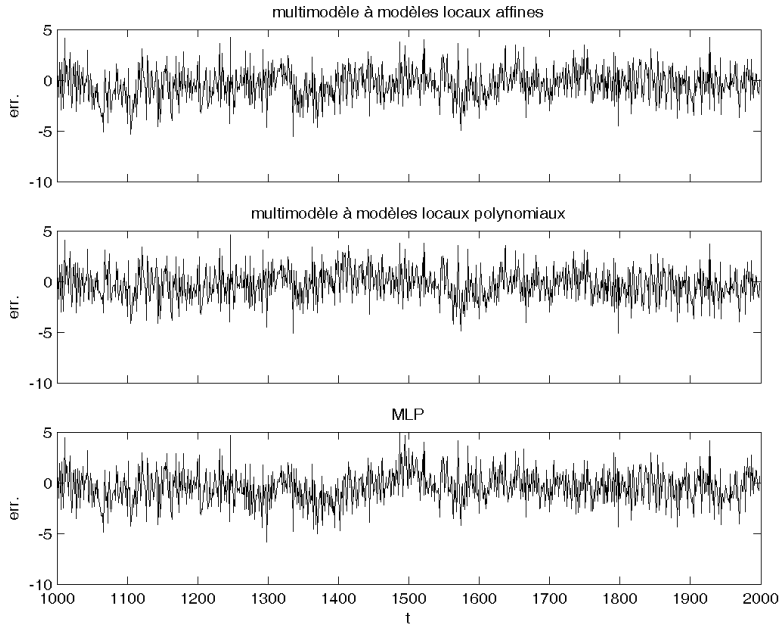

FIG. 4.5 – Résidus obtenus par les différents modèles NOE en validation.

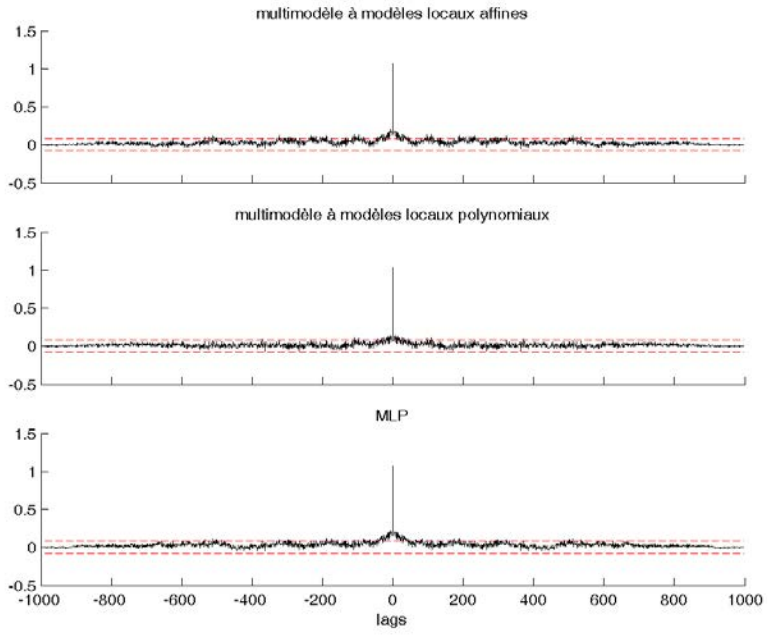

FIG. 4.6 – Fonctions d'autocorrélation des résidus obtenus par les différents modèles NOE en validation.

Structure	c	n_θ	AIC	*RMSE* app.	Durée id. (sec)	*RMSE* gen.
MM1 NOE	5	21	903	1.53	182	1.66
MM2 NOE	3	22	895	1.53	112	1.59
MLP NOE	3	13	951	1.60	650	1.67

TAB. 4.1 – Résultats obtenus par les modèles NOE en présence de bruit de sortie.

groupes de données dont les centres sont (l'entrée u est choisi comme variable d'indexation) :

$$c_1 = 0.944 \quad c_2 = 0.290 \quad c_3 = 0.086 \quad c_4 = 0.750 \quad c_5 = 0.477$$

Les degrés d'activation des modèles locaux sont représentés sur la figure 4.7.

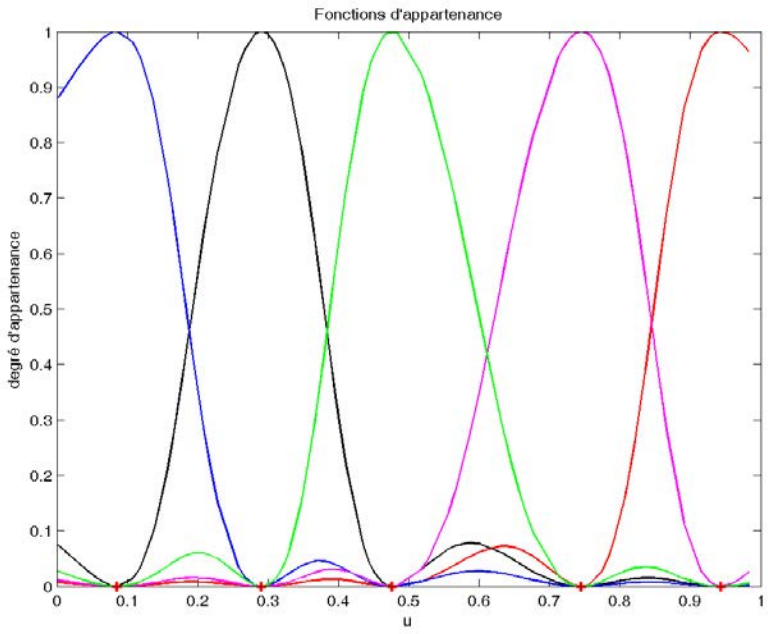

FIG. 4.7 – Identification du système par un multimodèle NOE à modèle locaux affines : degrés d'activation des modèles locaux obtenus avec l'algorithme des « *fuzzy-c-means* ».

Pour le groupe de données de centre c_1, le modèle local[2] est :

$$\hat{y}_1^*(t+1) = -0.051u^*(t) + 0.958\hat{y_1}^*(t) + .0374$$

[2]Tous les modèles sont établies pour des grandeurs normalisées entre -1 et 1. La valeur normalisée d'une variable x est : $x^* = 2 \times (x - x_{min})/(x_{max} - x_{min}) - 1$.

Pour le groupe de données de centre c_2, le modèle local est :

$$\hat{y}_2^*(t+1) = 0.115u^*(t) + 0.970\hat{y_2}^*(t) + 0.076$$

Pour le groupe de données de centre c_3, le modèle local est :

$$\hat{y}_3^*(t+1) = 0.253u^*(t) + 0.920\hat{y_3}^*(t) + 0.191$$

Pour le groupe de données de centre c_4, le modèle local est :

$$\hat{y}_4^*(t+1) = -0.008u^*(t) + 0.956 + \hat{y_4}^*(t) + 0.020$$

Pour le groupe de données de centre c_5, le modèle local est :

$$\hat{y}_5^*(t+1) = 0.047u^*(t) + 0.918\hat{y_5}^*(t) + 0.049$$

Pour le multimodèle à modèles locaux polynômiaux de degrés 2, l'identification structurelle a conduit à 3 modèles locaux. Les groupes de données correspondants ont pour centres :

$$c_1 = 0.381 \quad c_2 = 0.113 \quad c_3 = 0.835$$

Les degrés d'activation des modèles locaux sont représentés sur la figure 4.8.

Les modèles locaux sont :

pour le groupe de données de centre c_1 :

$$\hat{y}_1^*(t+1) = -0.002u^*(t) + 0.907\hat{y_1}^*(t) - 0.087u^*(t) \times \hat{y_1}^*(t) - 0.065u^{*2}(t) + 0.033\hat{y_1}^{*2}(t) + 0.05$$

pour le groupe de données de centre c_2 :

$$\hat{y}_2^*(t+1) = -0.001u^*(t) + 1.068\hat{y_2}^*(t) - 0.128u^*(t) \times \hat{y_2}^*(t) + 0.165u^{*2}(t) - 0.084\hat{y_2}^{*2}(t) + 0.0725$$

pour le groupe de données de centre c_3 :

$$\hat{y}_3^*(t+1) = 0.0247u^*(t) + 0.947\hat{y_3}^*(t) - 0.0630u^*(t) \times \hat{y_3}^*(t) - 0.010u^{*2}(t) + 0.043\hat{y_3}^{*2}(t) + 0.024$$

L'identification structurelle et paramétrique du réseau de neurones MLP NOE a conduit à 3 neurones cachés. L'architecture du réseau obtenu et les paramètres correspondants sont représentés sur la figure 4.9.

Le potentiel d'un neurone caché j est déterminé par :

$$v_j(t) = \omega_{j1}u^*(t) + \omega_{j2}\hat{y}^*(t) - b_j$$

où

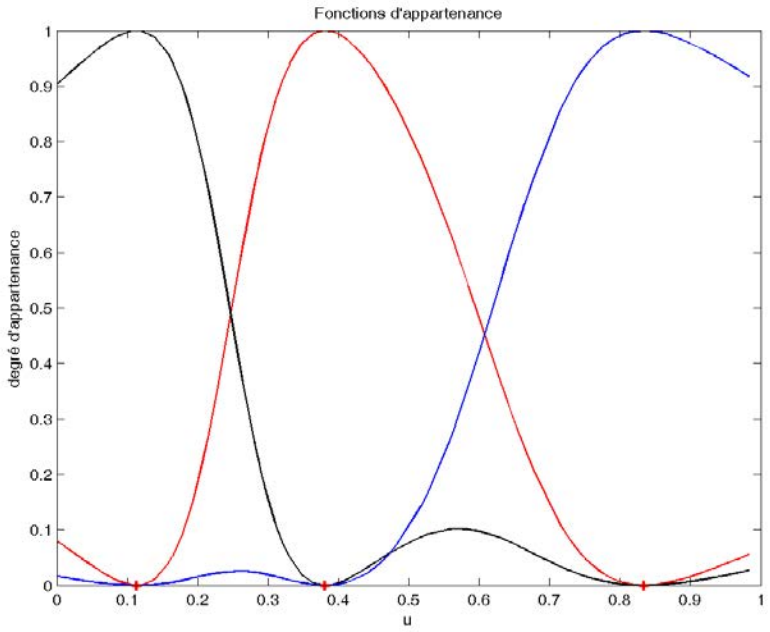

FIG. 4.8 – Identification du système par un multimodèle NOE à modèle locaux polynômiaux : degrés d'activation des modèles locaux obtenus avec l'algorithme des « *fuzzy-c-means* ».

$j = 1,\ 2,\ 3$;
ω_{j1} et ω_{j2} les poids entre le neurone caché j et les variables u et $\hat{y}$ respectivement ;
b_j est le biais.

La sortie du neurone est obtenue par :

$$O_j(t) = \frac{1}{1 + \exp(-v_j(t))}$$

La sortie du réseau est :

$$\hat{y}^*(t+1) = -0.67O_1(t) + 0.68O_2(t) + 2.54O_3(t) - 0.99$$

4.3.2 Présence de bruit d'état

En présence de bruit d'état, l'équation du système (4.1) se met sous la forme :

$$\begin{aligned} x(t+1) &= A\,x(t) + \sin\Big(\gamma\,u(t)\Big)\Big(\beta - u(t)\Big) + e(t+1) \\ y_s(t) &= x(t) \end{aligned} \tag{4.4}$$

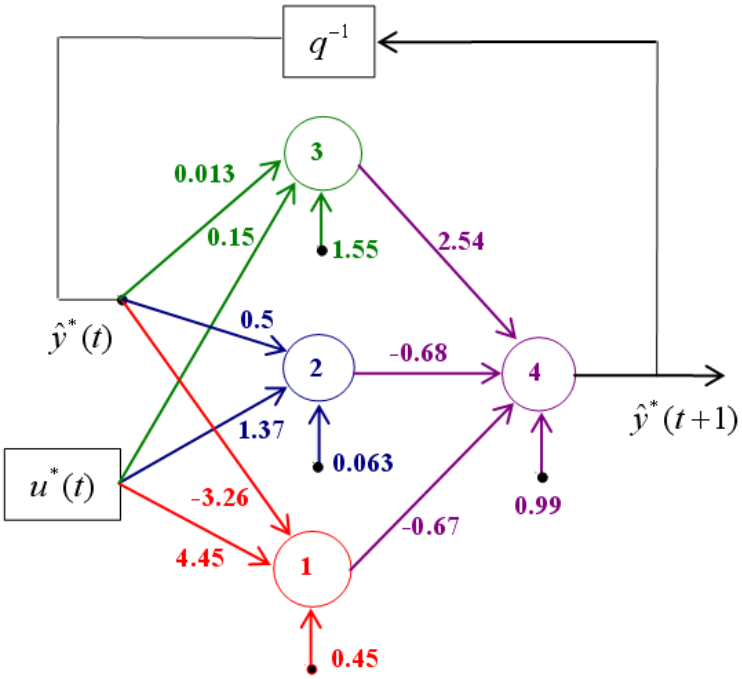

FIG. 4.9 – Architecture du réseau MLP NOE identifié.

où $e(\cdot)$ est un bruit blanc généré à partir d'une distribution normale de moyenne nulle et de dispersion $\sigma = 2.0$. Le rapport signal/bruit est de 6.1 dB. Les signaux d'identification du système sont présentés sur la figure 4.10.

Le système (4.4) est identifié par un modèle NARX à l'aide de structures MLP et multimodèles. Les vecteurs de régression utilisés pour les structures sont :

– pour la structure MLP NARX :

$$\underline{\varphi}_{MLP}(t) = [u(t) \quad y_s(t)]^T$$

– pour la structure MM1 NARX :

$$\underline{\varphi}_{MM1}(t) = [u(t) \quad y_s(t)]^T$$

– pour la structure MM2 NARX :

$$\underline{\varphi}_{MM2}(t) = [u(t) \quad y_s(t) \quad u(t) \times y_s(t) \quad u^2(t) \quad y_s^2(t)]^T$$

Pour les structures multimodèles, l'algorithme des « *fuzzy-c-means* » est utilisé pour la détermination des zones de validité des modèles locaux.

Les résultats obtenus sont résumés au tableaux 4.2. Le réseau MLP a un critère *AIC* légèrement plus faible mais présente la même capacité de généralisation que les

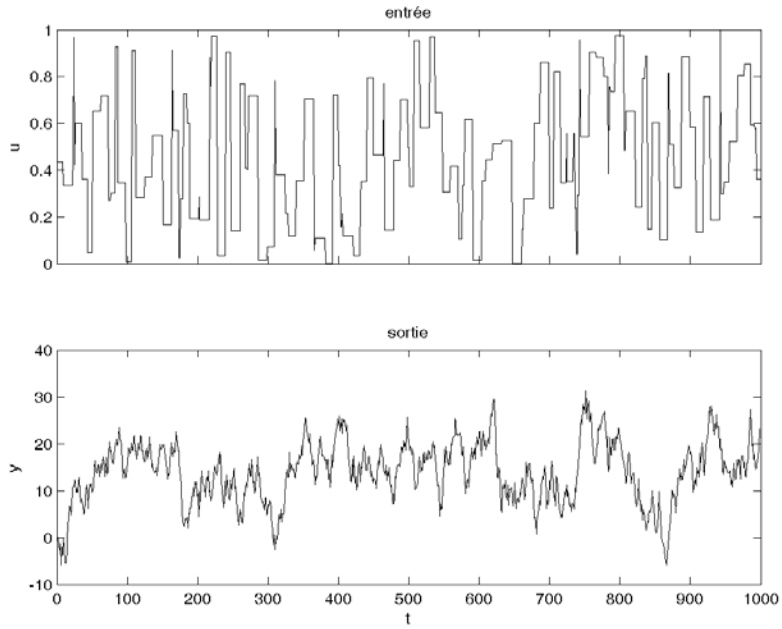

FIG. 4.10 – Signaux d'identification du système en présence de bruit d'état.

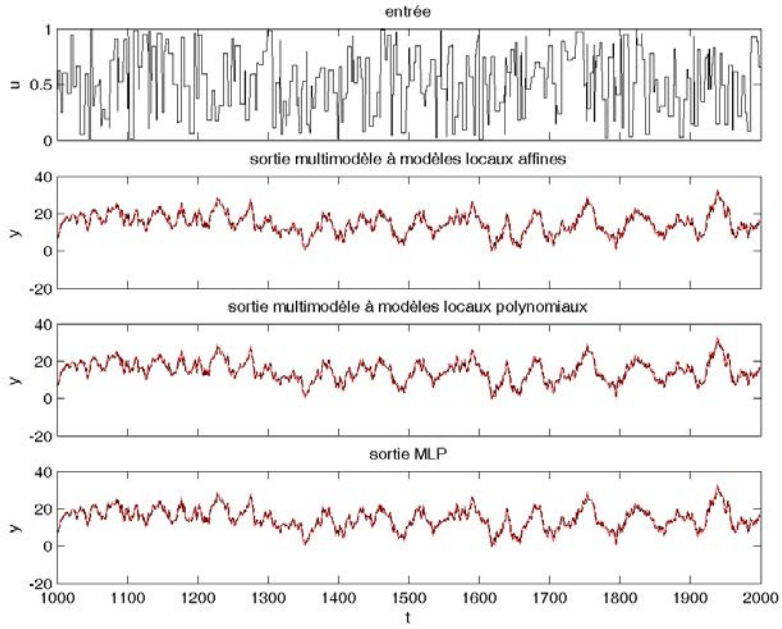

FIG. 4.11 – Entrée, sortie du système (trait plein) et sorties des modèles NARX (trait discontinu) en validation.

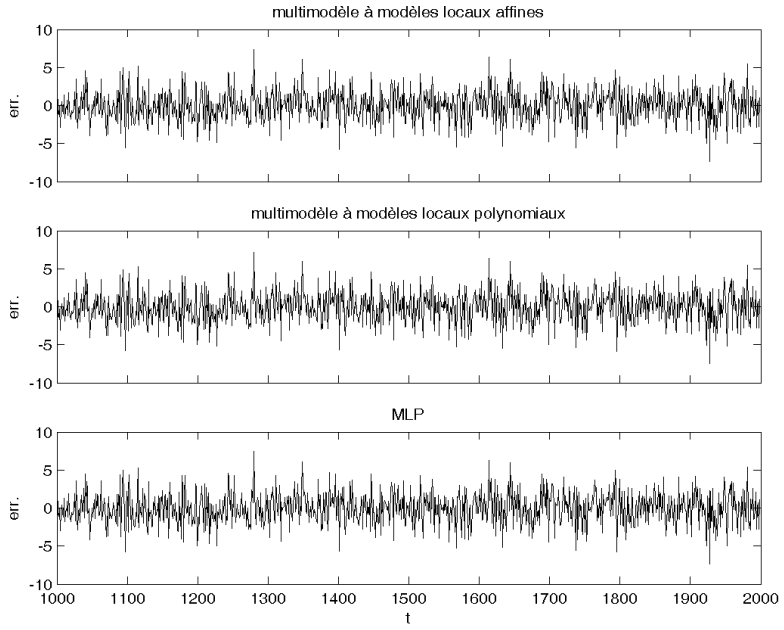

FIG. 4.12 – Résidus obtenus par les différents modèles NARX en validation.

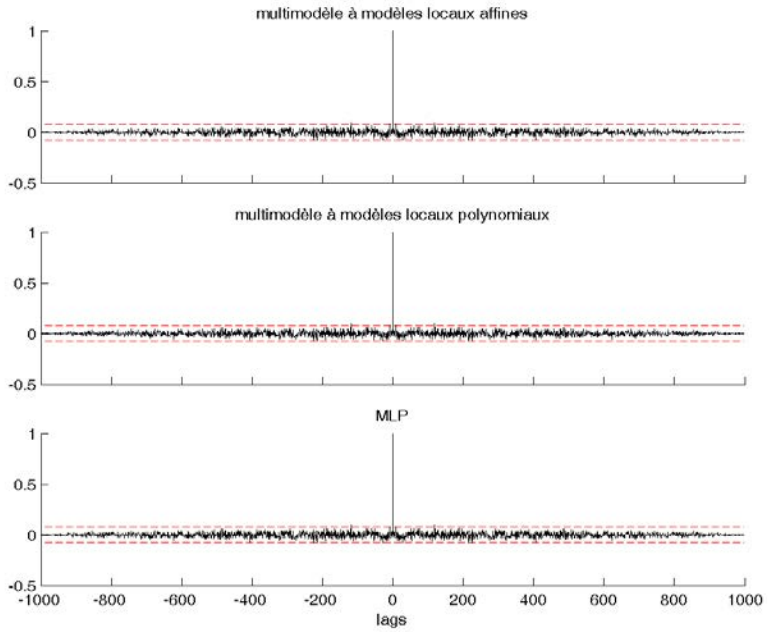

FIG. 4.13 – Fonctions d'autocorrélation des résidus obtenus par les différents modèles NOE en validation.

multimodèles. De plus, le temps nécessaire pour l'identification du système est beaucoup plus court avec un multimodèle qu'avec un MLP. Cet avantage de la structure multimodèle est dû a la simplicité de l'estimation paramétrique et sera beaucoup plus remarquable si le nombre d'échantillons est important. La figure 4.11 montrent la sortie du système et celles des modèles NARX en validation. Les résidus obtenus sont présentés sur la figure 4.12 et les fonctions d'autocorrélation des résidus sur la figure 4.13. Les marges représentées sur la figure correspondent à un intervalle de confiance à 99%.

Structure	c	n_θ	AIC	*RMSE* app.	Durée id. (sec)	*RMSE* gen.
MM1 NARX	2	9	1403	2.0	0.3	2.0
MM2 NARX	2	15	1410	2.0	0.4	2.0
MLP NARX	3	12	1399	2.0	86.0	2.0

TAB. 4.2 – Comparaison des structures multimodèle et MLP en présence de bruit d'état.

4.3.3 Présence de bruit d'état et de bruit de sortie

En présence de bruit d'état et de bruit de sortie, l'équation du système (4.1) se met sous la forme :

$$\begin{aligned} x(t+1) &= A\,x(t) + \sin\Big(\gamma\,u(t)\Big)\,\Big(\beta - u(t)\Big) + e_1(t+1) \\ y_s(t) &= x(t) + e_2(t) \end{aligned} \tag{4.5}$$

où $e_1(\cdot)$ est généré à partir d'une distribution normale de moyenne $\mu = 0$ et de dispersion $\sigma = 2.0$ et $e_2(\cdot)$ est généré à partir d'une distribution normale de moyenne $\mu = 0$ et de dispersion $\sigma = 1.5$. Le rapport signal/bruit est de 5.9 dB. La figure 4.14 représente les signaux d'identification utilisés.

Le système (4.5) est identifié par un modèle NARMAX avec des structures MLP et multimodèles en utilisant les vecteurs de régression suivants :

– pour la structure MLP NARMAX :

$$\underline{\varphi}_{MLP}(t) = [u(t) \quad y_s(t) \quad \varepsilon(t)]^T$$

– pour la structure MM1 NARMAX :

$$\underline{\varphi}_{MM1}(t) = [u(t) \quad y_s(t) \quad \varepsilon(t)]^T$$

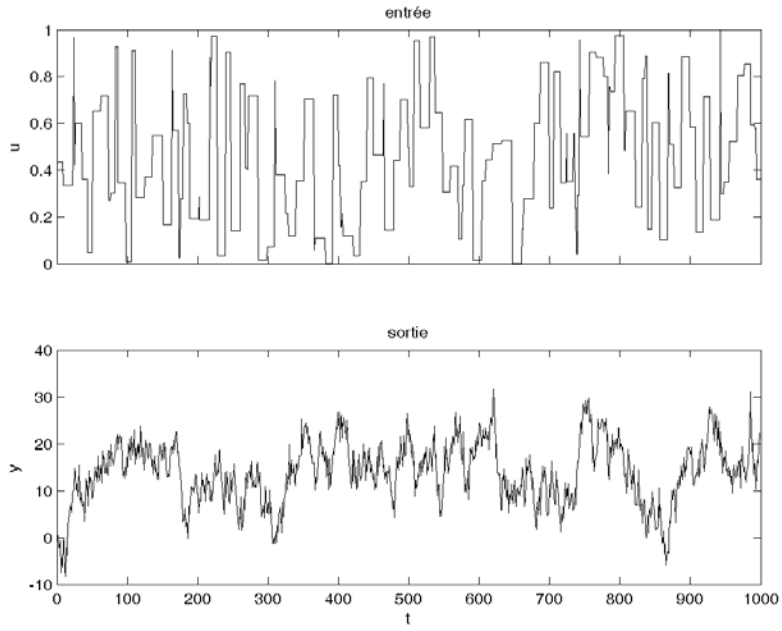

FIG. 4.14 – Signaux d'identification du système en présence de bruit de sortie et de bruit d'état.

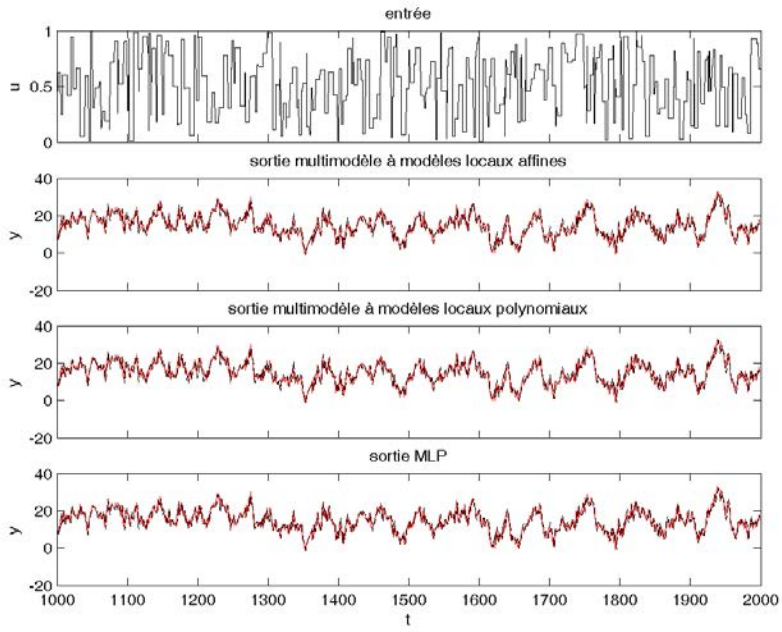

FIG. 4.15 – Entrée, sortie du système (trait plein) et sorties des modèles NARMAX (trait discontinu) en validation.

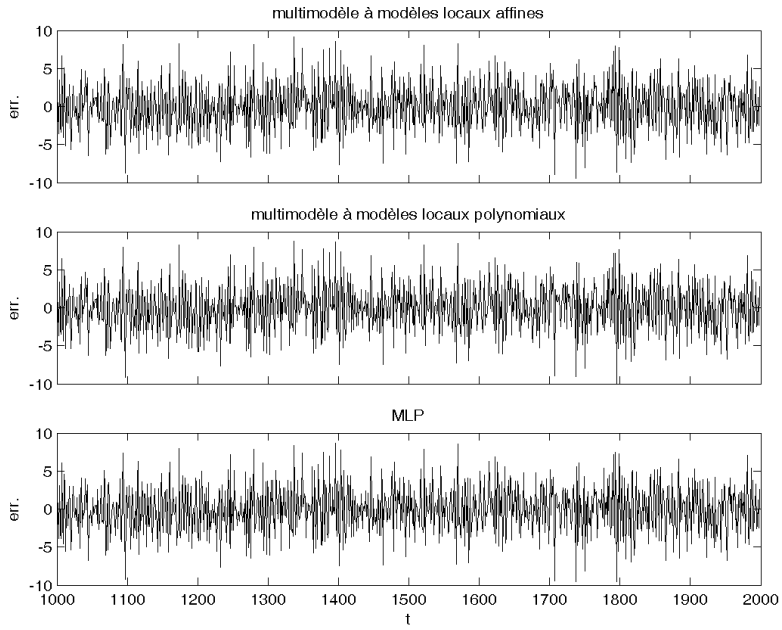

FIG. 4.16 – Résidus obtenus par les différents modèles NARMAX en validation .

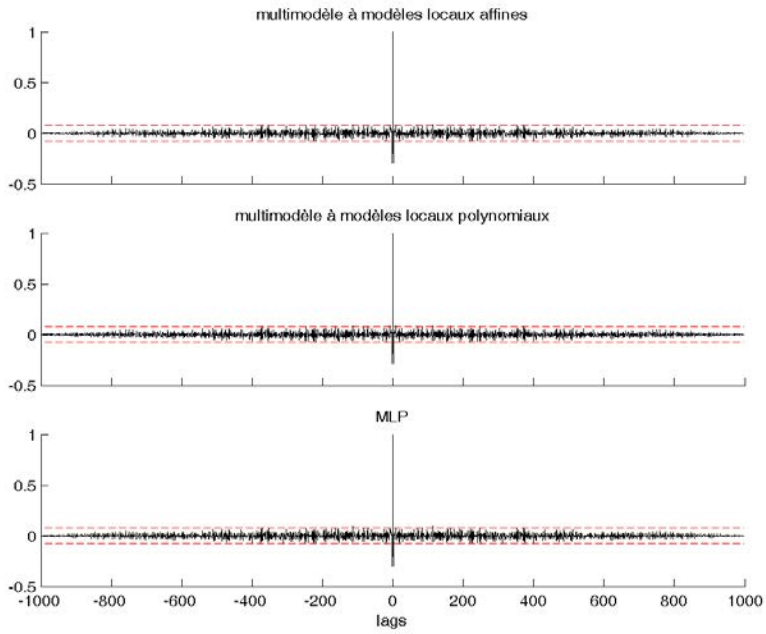

FIG. 4.17 – Fonctions d'autocorrélation des résidus des modèles NARMAX en validation.

– pour la structure MM2 NARMAX :

$$\underline{\varphi}_{MM2}(t) = [u(t) \;\; y_s(t) \;\; \varepsilon(t) \;\; u^2(t) \;\; u(t)y_s(t) \;\; u(t)\varepsilon(t) \;\; y_s^2(t) \;\; y_s(t)\varepsilon(t) \;\; \varepsilon^2(t)]^T$$

Le multimodèle à modèles locaux affines se présente comme le meilleur modèle avec le plus faible critère AIC. Tous les modèles possèdent la même capacité de généralisation. La durée d'identification des structures multimodèles est ici aussi plus courte que la structure MLP.

Structure	c	n_θ	AIC	*RMSE* app.	Durée id. (sec)	*RMSE* gen.
MM1 NARMAX	2	11	2058	2.8	14	3.01
MM2 NARMAX	2	23	2073	2.8	26	3.02
MLP NARMAX	3	16	2065	2.8	480	2.99

TAB. 4.3 – Résultats obtenus par les modèles NARMAX en présence de bruit d'état et de bruit de sortie.

La figure 4.15 montrent la sortie du système et celles des modèles NARMAX en validation sur des signaux non bruités. Les résidus obtenus sont présentés sur la figure 4.16 et les fonctions d'autocorrélation des résidus sur la figure 4.17 avec des intervalles de confiance de 99%. L'hypothèse selon laquelle le résidu est blanc est donc validée pour les trois modèles.

4.4 Comparaison des architectures multimodèle et MLP pour l'identification du système de Box et Jenkins

Dans le système de Box-Jenkins [24], les données sont obtenues de la combustion d'un mélange de méthane et d'air. L'entrée du système correspond à la variation de débit du méthane injecté dans le four et la sortie du système correspond à la concentration de CO_2 dans le gaz obtenu de la combustion du mélange. La base comporte 290 données. Les 2/3 sont utilisées pour l'identification et le tiers restant pour la validation. La figure 4.18 montre l'entrée et la sortie du système.

Des modèles NARX avec des implantations neuronale et multimodèle à modèles

locaux affines sont utilisés. Le vecteur de régression est :

$$\underline{\varphi}(t) = [u(t) \quad u(t-1) \quad y_s(t) \quad y_s(t-1) \quad y_s(t-2)]$$

Différentes méthodes de décomposition de l'espace de fonctionnement du système sont testées : partitionnement hiérarchique orthogonal aux axes (HOA), partitionnement flou par « *fuzzy-c-means* » (FCM) et par « *subtractive clustering* » (SC). La variable d'indexation utilisée pour la décomposition est l'entrée u ($\xi = u$).

Les résultats obtenus montrent que pour les multimodèles, les modes de partitionnement flou conduisent tous à trois modèles locaux alors que le mode partitionnement hiérarchique orthogonal aux axes donne deux modèles locaux. Les performances des modèles sont similaires aussi bien en apprentissage qu'en validation. Cependant la durée de la procédure d'identification du MLP est beaucoup plus importante. Le réseau MLP, bien qu'ayant le meilleur critère AIC, ne généralise pas mieux que les multimodèles.

Les résultats sont résumés au tableau 4.4. La figure 4.19 montre la sortie du système et celle des modèles en validation. Les résidus obtenus par les différents modèles sont représentés sur la figure 4.20. La figure 4.21 représente les fonctions d'autocorrélation des résidus.

Structure	c	n_θ	AIC	*RMSE* app.	Durée id. (sec)	*RMSE* gen.
MM1-FCM	3	22	-651	0.16	0.09	0.40
MM1-SC	3	22	-648	0.16	0.05	0.40
MM1-HOA	2	16	-625	0.17	0.2	0.40
MLP	2	15	-655	0.16	10	0.43

TAB. 4.4 – Résultats obtenus pour les modèles NARX du système de Box et Jenkins implantés avec des structures multimodèles et MLP.

Les résultats obtenus pour le multimodèle résultant de la décomposition de l'espace caractéristique du système par l'algorithme des « fuzzy-c-means » sont détaillés ici. Les trois groupes de données obtenus ont pour centres :

$$c_1 = 0.045 \quad c_2 = 1.29 \quad c_3 = -1.37$$

La figure 4.22 représente les fonctions d'appartenance aux groupes de données obtenus.

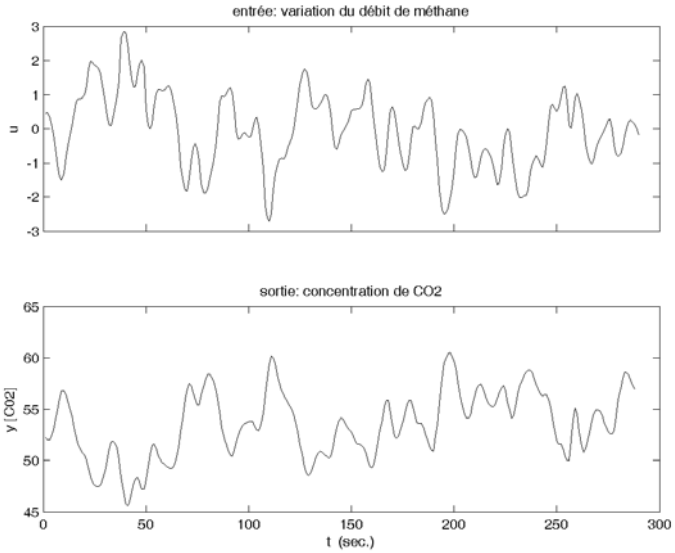

FIG. 4.18 – Entrée et sortie du système de Box-Jenkins.

Pour le groupe de données de centre c_1, le modèle local est :

$$y_1^*(t+1) = 0.159u^*(t) - 0.319u^*(t-1) + 1.436y_s^*(t) - 0.643y_s^*(t-1) + 0.0529y_s^*(t-2) + 0.002$$

Pour le groupe de données de centre c_2, le modèle local est :

$$y_2^*(t+1) = 0.069u^*(t) - 0.274u^*(t-1) + 1.334y_s^*(t) - 0.571y_s^*(t-1) + 0.080y_s^*(t-2) + 0.010$$

Pour le groupe de données de centre c_3, le modèle local est :

$$y_3^*(t+1) = 0.011u^*(t) - 0.297u^*(t-1) + 0.919y_s^*(t) + 0.164y_s^*(t-1) - 0.266y_s^*(t-2) - 0.031$$

Les degrés d'activation de ces modèles locaux (fonctions d'appartenance des données aux groupes, voir figure 4.22) pour une observation,

$$\underline{\varphi}_j = [u(j) \quad u(j-1) \quad y_s(j) \quad y_s(j-1) \quad y_s(j-2)]^T,$$

indexé par $\xi_j = u(j)$ sont déterminés conformément à la description du paragraphe §3.3.2.3) :

$$\omega_i = \frac{1}{\sum_{k=1}^{3} \left(D_{ij}/D_{kj} \right)^{2/(m-1)}} \qquad i = 1,\, 2,\, 3.$$

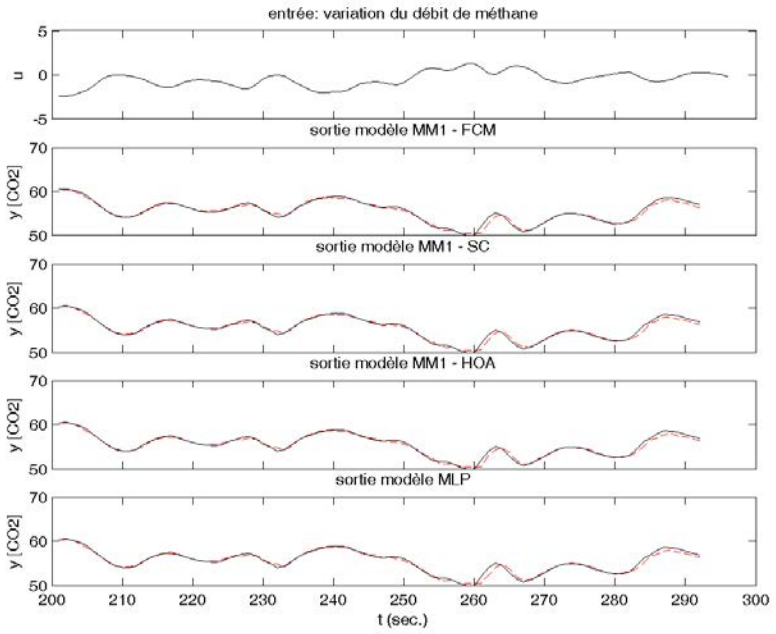

FIG. 4.19 – Entrée, sortie du système de Box-Jenkins (trait plein) et sortie des modèles (trait discontinu) en validation.

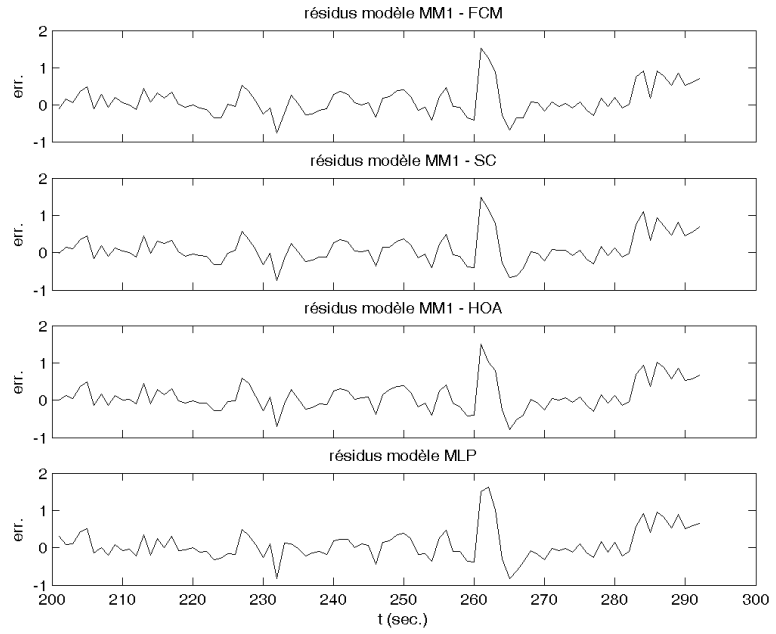

FIG. 4.20 – Résidus des modèles identifiés.

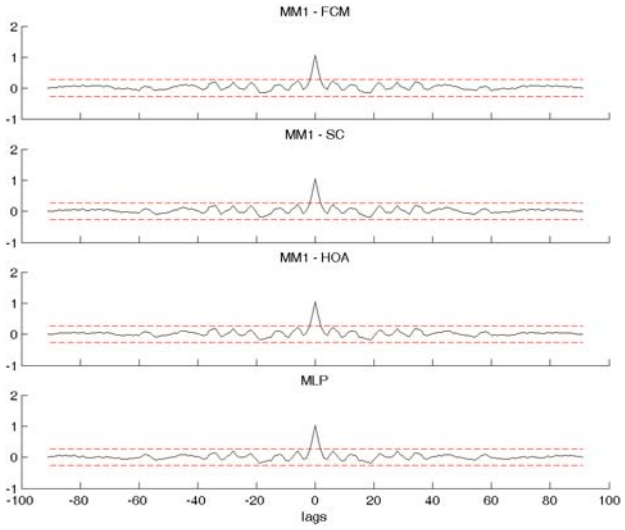

FIG. 4.21 – Fonctions d'autocorrélation des résidus obtenus en validation et marges correspondant à un intervalle de confiance à 99%.

où D_{ij} est la distance euclidienne entre le point ξ_j^* et le centre c_i^* :

$$D_{ij}^2 = \|\xi_j^* - c_i^*\|^2 = (\xi_j^* - c_i^*)^2$$

La sortie du multimodèle est alors : $y^* = \omega_1 y_1^* + \omega_2 y_2^* + \omega_3 y_3^*$

Avec le mode de décomposition par l'algorithme du « *subtractive clustering* », les centres des 3 groupes de données obtenus sont :

$$c_1 = 0.54 \quad c_2 = -0.94 \quad c_3 = 1.68$$

Les modèles locaux sont :

$$y_1^*(t+1) = 0.107u^*(t) - 0.302u^*(t-1) + 1.534y_s^*(t) - 0.817y_s^*(t-1) + 0.142y_s^*(t-2) + 0.007$$

$$y_2^*(t+1) = 0.0541u^*(t) - 0.305u^*(t-1) + 1.038y_s^*(t) - 0.038y_s^*(t-1) - 0.181y_s^*(t-2) - 0.0173$$

$$y_3^*(t+1) = 0.0460u^*(t) + 0.0460u^*(t-1) + 1.157y_s^*(t) - 0.307y_s^*(t-1) - 0.0259y_s^*(t-2) + 0.026$$

Le mode de décomposition hiérarchique orthogonal aux axes a donné 2 zones, le découpage étant effectué au milieu de la partition de la variable d'indexation (0). Les fonctions d'appartenance obtenues sont présentées sur la figure 4.24. Les modèles locaux sont :

$$y_1^*(t+1) = -0.299u^*(t) - 0.367u^*(t-1) + 0.529y_s^*(t) + 0.825y_s^*(t-1) - 0.586y_s^*(t-2) - 0.800$$

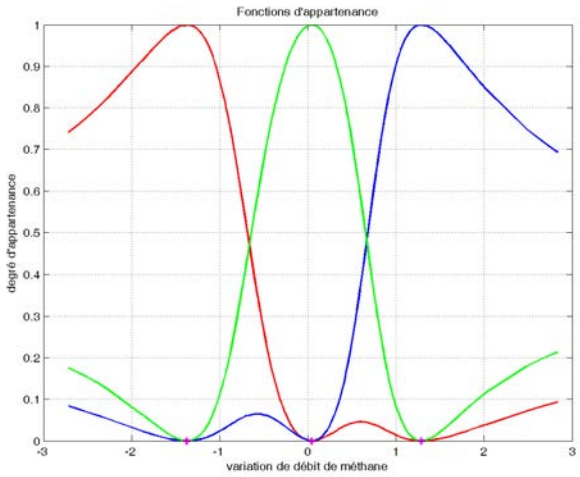

FIG. 4.22 – Fonctions d'appartenance aux groupes de données obtenus par l'algorithme des « *fuzzy-c-means* ».

$$y_2^*(t+1) = -0.271u^*(t) - 0.217u^*(t-1) + 2.097y_s^*(t) - 1.797y_s^*(t-1) + 0.616y_s^*(t-2) + 0.800$$

Pour le réseau MLP NARX, l'architecture obtenue est présentée sur la figure 4.25.

Les fonctions d'activation des 2 neurones cachés sont définies par :

$$f(v) = \frac{2}{1 + \exp(-2v)} - 1$$

où v est le potentiel du neurone. Pour un neurone d'indice j ($j = 1, 2$), le potentiel est :

$$v_j = \sum_{i=1}^{5} \omega_{ji} x_i - b_j$$

où x_i est l'entrée i du réseau (u ou y_s) et b_j le biais.

Le neurone de sortie a une fonction d'activation linéaire :

$$f(v_3) = v_3$$

où v_3 est le potentiel du neurone déterminé par :

$$v_3 = \omega_{31} O_1 + \omega_{32} O_2 - b_3$$

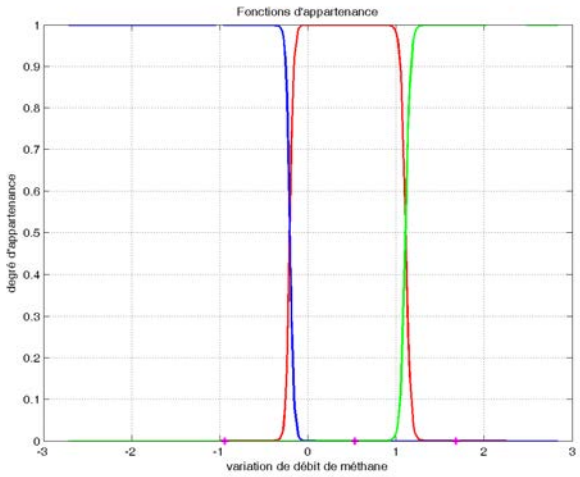

FIG. 4.23 – Fonctions d'appartenance aux groupes de données obtenus par l'algorithme du « *subtractive clustering* ».

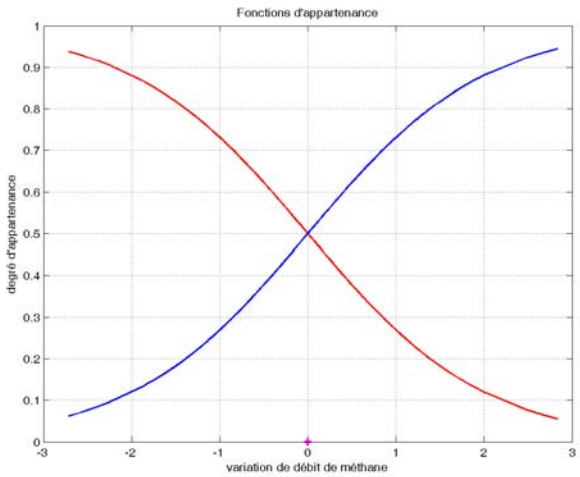

FIG. 4.24 – Fonctions d'appartenance aux groupes de données obtenus par le partitionnement hiérarchique orthogonal aux axes.

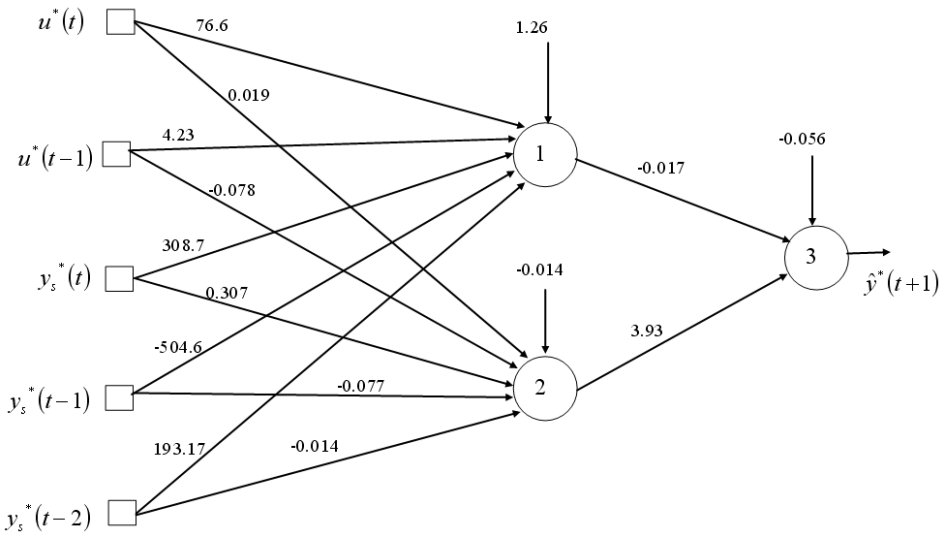

FIG. 4.25 – Architecture du réseau MLP obtenu pour l'identification du système de Box et Jenkins.

4.5 Comparaison des architectures multimodèle et MLP pour la prédiction de débit sur le fleuve Sénégal

La prédiction de débits des fleuves joue un rôle fondamental dans la prévention des catastrophes naturelles telles que par exemple les crues. Elle permet aussi de faire la planification du stockage d'eau dans les barrages, de gérer les projets d'irrigation et sert d'outil d'aide à la navigation. Plusieurs travaux ont démontré le potentiel des MLP non bouclés pour la prévision hydrologique [6, 99, 100].

Nous nous proposons de comparer une structure neuronale et une structure multimodèle pour la prédiction de débit sur le fleuve Sénégal (voir [101, 102] pour plus de détails) à la station de Bakel, à partir des mesures de débit en amont au niveau des stations de Fadougou et Oualia situées respectivement sur la rivière Falémé et sur la rivière Bakoye (voir figures 4.26 et 4.27). L'absence de données d'autres stations dont les débits influent sur celui de Bakel nous conduit à utiliser ces deux stations et l'historique des débits à Bakel comme entrées du modèle. Nous utilisons donc une structure NARX pour tenter de prédire les débits du fleuve à Bakel pour des horizons de prédiction variant de 1 jour à 12 jours.

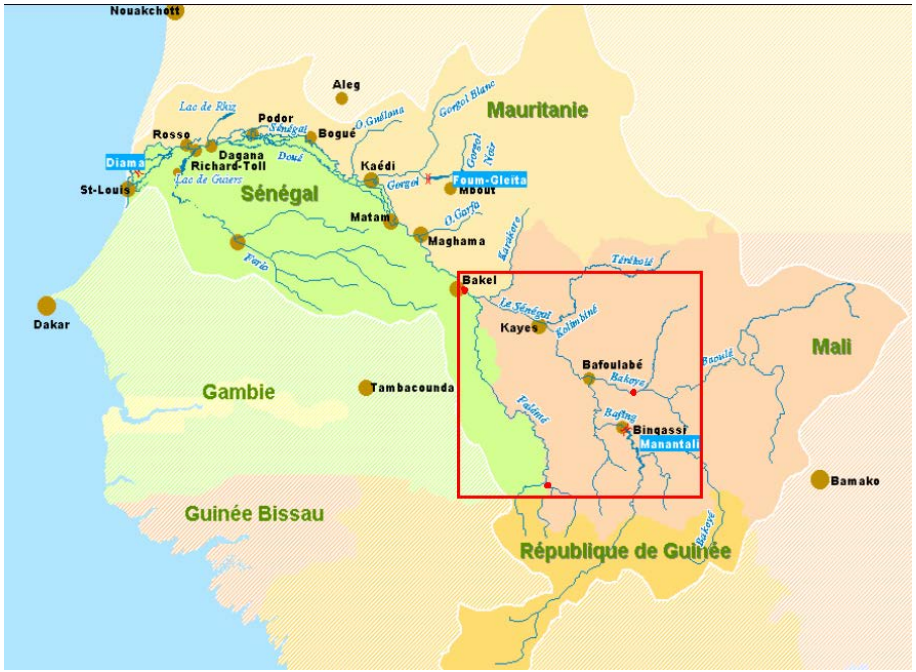

FIG. 4.26 – Le bassin du fleuve Sénégal.

La base de données comporte les mesures de 1985 à 1994. La base d'apprentissage est constituée des 8 premières années et la base de validation des 2 dernières années. Les courbes de débit des 3 stations sont représentées sur les figures 4.28 et 4.29.

Le modèle recherché est de la forme :

$$Q_B(t+h) = F\left(\underline{\varphi}(t)\right)$$

où :
$Q_B(t+h)$ est le débit à Bakel au jour $t+h$, h étant l'horizon de prédiction (en jour) ;
$\underline{\varphi}(t) = [Q_O(t)\ Q_F(t)\ Q_B(t)]^T$ est le vecteur de régression avec $Q_O(t)$, $Q_F(t)$ et $Q_B(t)$ les débits à Oualia, Fadougou et Bakel respectivement.

Les critères suivants, souvent utilisés en hydrologie [103], sont évalués sur les données de validation :

– *Root mean square error* ($RMSE$)

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left[Q_B(t) - \hat{Q}_B(t)\right]^2}$$

où $\hat{Q}_B(t)$ est le débit estimé.

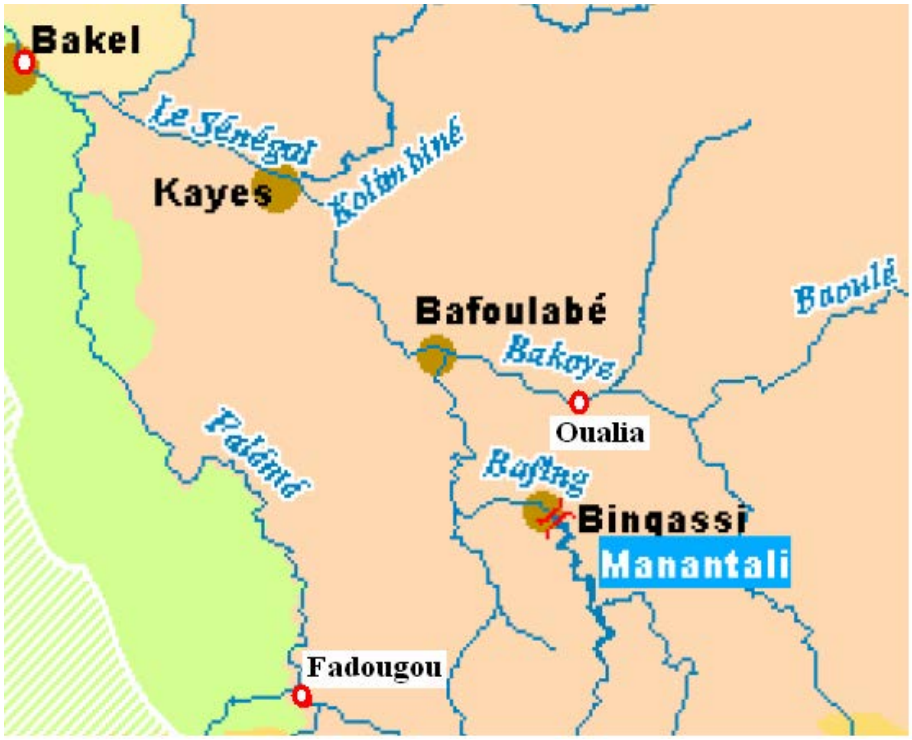

FIG. 4.27 – Station de Bakel, Fadougou et Oualia.

– Critère de Nash (NS) :

$$NS = 1 - \frac{\sum_{t=1}^{N} \left[Q_B(t) - \hat{Q}_B(t) \right]^2}{\sum_{t=1}^{N} \left[Q_B(t) - \overline{Q}_B \right]^2}$$

où $\overline{Q}_B$ est la valeur moyenne des débits observés. Le critère de Nash mesure la part de la variance expliquée grâce au modèle. NS peut varier de $-\infty$ à 1. En général, une valeur supérieure à $0,7$ est considérée comme satisfaisante, la valeur optimale étant 1. Le critère est utilisé en particulier pour estimer la qualité des débits simulés.

Les résultats obtenus sont présentés au tableau 4.5. La durée de la procédure d'identification (Durée id) de la structure multimodèle est ici aussi plus courte que celle du réseau MLP (de 40 à 60 fois plus faible). L'identification structurelle du MLP conduit à un réseau à 4 neurones cachés. Le mode de partitionnement flou par l'algorithme des « *fuzzy-c-means* » est utilisé pour le multimodèle. Les variables $Q_O(t)$ et $Q_F(t)$ sont utilisées comme variables d'indexation :

$$\underline{\xi}(t) = [Q_O(t)\ Q_F(t)]^T$$

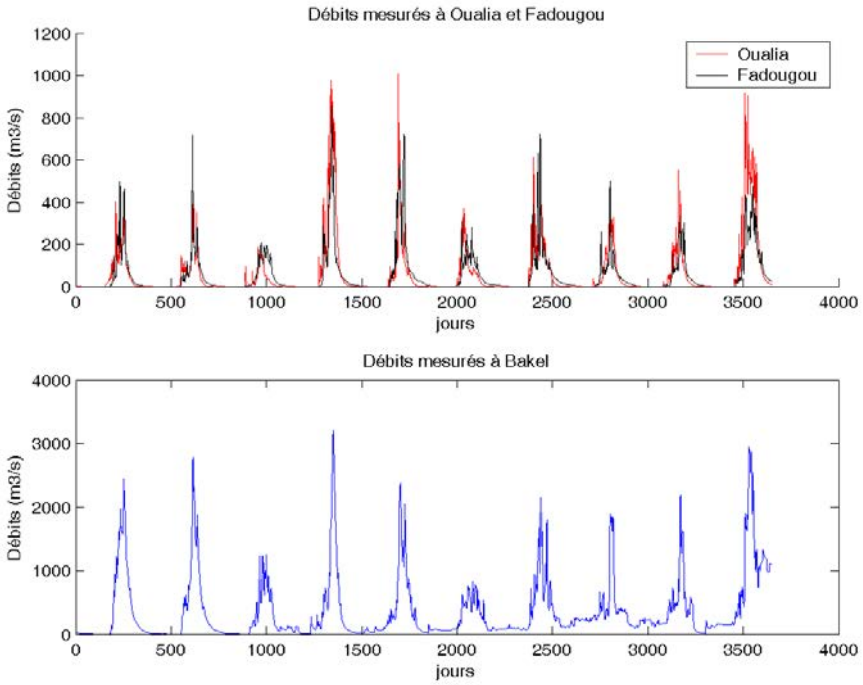

FIG. 4.28 – Courbes de débits (moyennes journalières en m^3/s) des stations de Bakel, Fadougou et Oualia de 1985 à 1994.

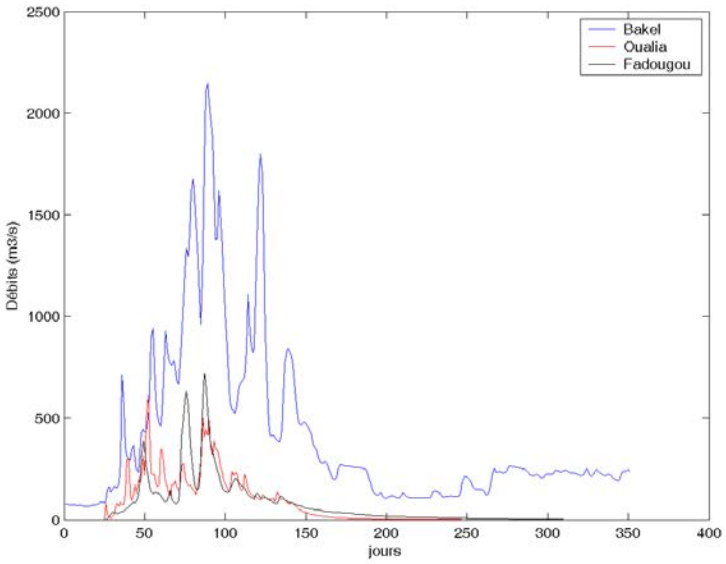

FIG. 4.29 – Débits mesurés aux stations de Bakel, Fadougou et Oualia pour l'année 1991.

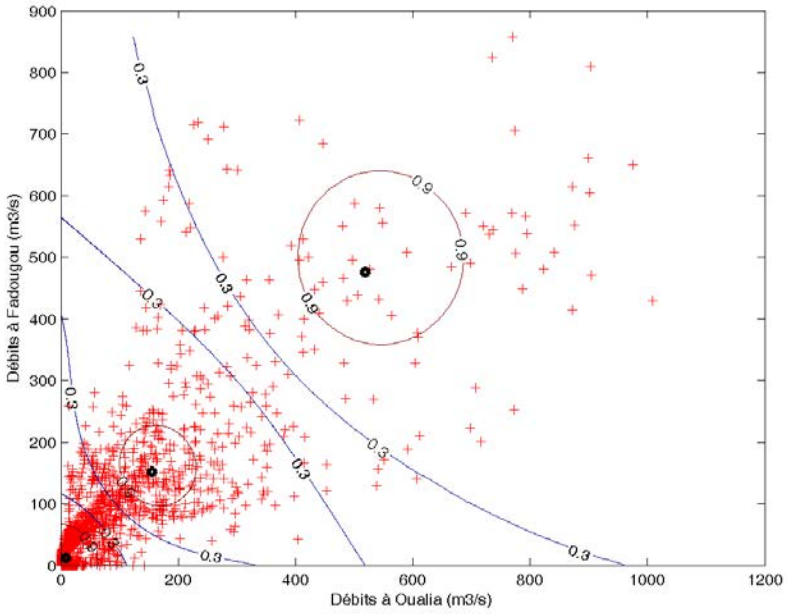

FIG. 4.30 – Groupes de données obtenus par la décomposition floue par « *fuzzy-c-means* ».

Trois groupes de données sont ainsi identifiés avec comme centres :

$$\underline{c}_1 = \begin{pmatrix} 8 \\ 12 \end{pmatrix} \quad \underline{c}_2 = \begin{pmatrix} 155 \\ 152 \end{pmatrix} \quad \underline{c}_3 = \begin{pmatrix} 519 \\ 476 \end{pmatrix}$$

Ces groupes de données sont représentés sur la figure 4.30. Pour chaque groupe de données, un modèle local affine est établi. La sortie du modèle local pour une observation donnée est pondérée par le degré d'appartenance de l'observation au groupe de données pour lequel est défini le modèle local. Les résultats montrent que le réseau MLP et le multimodèle ont pratiquement les mêmes performances en apprentissage. Cependant, la capacité de généralisation du multimodèle est souvent légèrement meilleure. La figure 4.31 présente l'évolution de la $(RMSE)$ en validation du MLP et du multimodèle en fonction de l'horizon de prédiction.

La figure 4.32 représente les courbes de débits mesurés et prédits par le multimodèle pour les horizons de prédiction $h = 1$, $h = 3$, $h = 5$ et $h = 7$ jours pour les 2 années de validation.

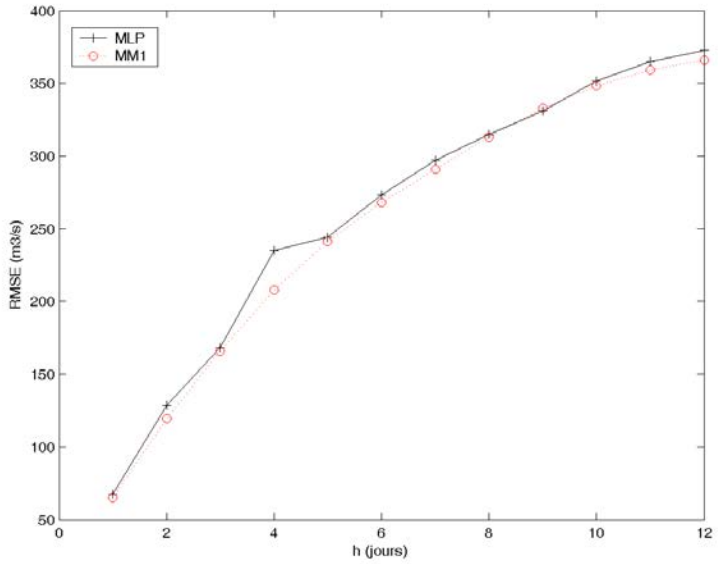

FIG. 4.31 – Evolution de la $RMSE$ de validation du MLP et du multimodèle en fonction de l'horizon de prédiction.

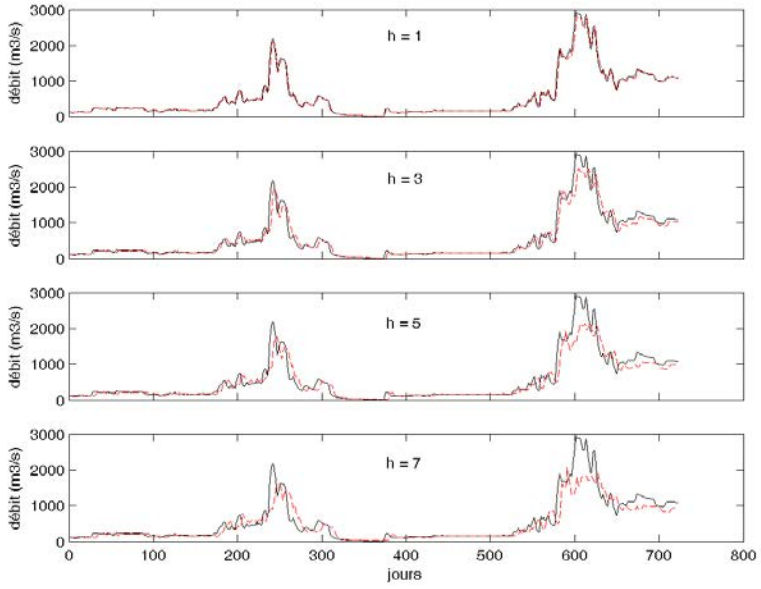

FIG. 4.32 – Débits mesurés à la station de Bakel (courbes en trait plein) et prédictions correspondantes (courbes en trait discontinu) pour divers horizons de prédiction h pour la période de validation.

horizon	MLP			MM1		
h	Durée id. (s)	$RMSE_v$	NS	Durée id. (s)	$RMSE_v$	NS
1	40	67	0.99	1	65	0.99
2	41	129	0.96	1	119	0.96
3	43	168	0.92	1	166	0.93
4	44	235	0.85	1	208	0.87
5	44	244	0.84	1	241	0.85
6	45	273	0.80	1	268	0.81
7	51	297	0.77	1	291	0.78
8	52	315	0.74	1	313	0.74
9	55	331	0.71	1	333	0.71
10	59	352	0.68	1	348	0.68
11	59	365	0.65	1	359	0.66
12	59	373	0.64	1	366	0.65

TAB. 4.5 – Evolution des performances du multimodèle et du MLP pour la prévision de débits en fonction de l'horizon de prédiction.

4.6 Conclusion

Dans ce chapitre, les structures multimodèle et MLP ont été étudiées et comparées à travers plusieurs exemples. Ces structures sont d'abord utilisées pour représenter un système dynamique non linéaire simulé avec l'influence de différents types de bruits : bruit d'état, bruit de sortie et bruit d'état et de sortie. Le système de Box et Jenkins a également été modélisé par des structures multimodèle et MLP, de même que le débit du fleuve Sénégal à la station de Bakel.

Pour chacun de ces exemples, il a été montré qu'il existe une structure multimodèle présentant des capacités d'identification identiques et souvent meilleures que la structure MLP. La durée de la procédure d'identification structurelle et paramétrique des multimodèles est beaucoup plus courte que celle des MLP. Cet avantage des multimodèles est principalement dû à la structure simple des modèles locaux, qui peut être linéaire en les paramètres, permettant ainsi l'utilisation de méthodes d'optimisation linéaire pour l'estimation paramétrique.

Conclusion générale et perspectives

La modélisation mathématique est un outil indispensable à la représentation de systèmes pour maîtriser ou optimiser leur fonctionnement, ou comprendre leur évolution. Cette technique de modélisation est constituée de la modélisation théorique et de la modélisation expérimentale. La première nécessite la connaissance des lois régissant le fonctionnement du système, alors que la deuxième ne nécessite que des mesures sur le système à modéliser. Elle est donc plus facile à mettre en œuvre et a fait l'objet de ce travail de thèse.

Différentes structures de modèles non linéaires ont été présentées : modèles NFIR, NARX, NOE et NARMAX. Les structures NFIR et NARX sont non bouclées (non récurrentes), leur représentation ne se faisant qu'avec les mesures courantes ou passées faites sur le système. Les structures NOE et NARMAX sont bouclées (récurrentes) en ce sens qu'elles utilisent des mesures courantes ou passées du système ainsi que l'estimation de la sortie et de l'erreur.

La représentation neuronale de ces structures est sujette à quelques difficultés liées à l'estimation paramétrique (problèmes inhérents à l'optimisation non linéaire) et à l'identification structurelle (détermination du nombre de neurones cachés). L'estimation paramétrique est faite avec la méthode de rétro-propagation du gradient pour les structures non-bouclées ou avec la méthode de rétro-propagation du gradient à travers le temps pour les structures bouclées. Une méthodologie combinant l'identification structurelle et l'estimation paramétrique a été présentée. C'est une démarche constructive qui consiste à augmenter progressivement le nombre de neurones cachés, à estimer les paramètres du réseau, et à évaluer un critère de sélection de modèle permettant d'identifier la meilleure topologie du réseau. Des exemples illustratifs permettant également de voir l'influence du

bruit sur le choix de la structure des modèles non linéaires ont été présentés.

Une architecture multimodèle capable de surmonter certaines difficultés de l'architecture neuronale de type MLP a été étudiée. L'approche multimodèle consiste à représenter un système complexe par un ensemble de modèles de structures simples à validité limitée dans des zones bien définies. Des algorithmes d'identification structurelle et paramétrique ont été proposés. Des modèles locaux de structure polynômiale de degré supérieur à 1 ont été étudiés et implantés. L'estimation paramétrique d'une telle architecture multimodèle peut se faire suivant une optimisation linéaire, moins coûteuse en temps de calcul que l'estimation paramétrique utilisée dans les réseaux de neurones. L'étude a montré que l'usage des modèles locaux de structure polynômiale permet de mieux appréhender les non linéarités locales, réduisant ainsi le nombre de modèles locaux. Cette structure de multimodèle constitue l'un des apports de ce travail de thèse.

La détermination du nombre de modèles locaux dans une architecture multimodèle nécessite la décomposition (le partitionnement) de l'espace de fonctionnement du système en plusieurs sous-espaces où sont définis les modèles locaux. Différents modes de partitionnement existent. Le partitionnement grille et le partitionnement hiérarchique orthogonal aux axes sont des méthodes déterministes pour lesquelles les frontières entre les différentes zones de validité des modèles locaux sont définies. La principale difficulté réside dans la détermination de ces frontières (points de découpage des partitions). Des modes de partitionnement flou (basé sur les algorithmes de « *fuzzy-c-means* », de « *Gustafson et Kessel* » et du « *subtractive clustering* »), permettent de regrouper des données qui présentent une certaine « similarité », contournant ainsi le problème des points de découpage des partitions. L'utilisation de telles méthodes permet l'implantation d'une architecture multimodèle hétérogène où les modèles locaux peuvent être de structures différentes : polynômiales de degrés différents, neuronale, polynômiale et neuronale. La structure de chaque modèle local est alors déterminée par les données qui constituent sa zone de validité. Des algorithmes d'identification de ces structures ont été proposés. L'architecture multimodèle ainsi obtenue correspond à celle des modèles multi-experts.

Une autre contribution apportée dans ce travail est l'implantation des multimodèles récurrents, avec un algorithme d'estimation paramétrique plus souple que l'algorithme « *Back Propagation Through Time* » et l'algorithme « *Real Time Recurrent Learning* » généralement utilisés pour l'apprentissage des réseaux de neurones récurrents. Cette architecture multimodèle permet de représenter des modèles non linéaires bouclés tels que les modèles NARMAX et NOE.

Une étude comparative entre les architectures MLP et multimodèle a été menée. Un exemple académique faisant intervenir différents modes de bruits (bruit de sortie, bruit d'état et bruit de sortie et d'état) est utilisé pour la comparaison des architectures multimodèle et MLP. Cette étude montre que les deux architectures ont pratiquement les mêmes performances en apprentissage et en validation, mais la durée de la procédure d'identification des multimodèles est beaucoup plus courte que celle des réseaux MLP, aussi bien pour les modèles récurrents que non récurrents. Ce même résultat est obtenu pour l'identification du système de Box et Jenkins où la sortie, qui est la concentration de CO_2 résultant de la combustion d'un mélange de méthane et d'air, est expliquée en fonction de la variation du débit de méthane injecté dans le four.

Les architectures multimodèles et neuronales ont également été comparées pour la modélisation du débit du fleuve Sénégal à la station de Bakel, en fonction des débits mesurés aux stations de Fadougou sur la rivière Falémé et Oualia sur la rivière Bakoye, les débits de ces deux rivières influant sur celui du fleuve Sénégal. Les résultats obtenus montrent aussi des performances similaires pour les deux architectures, mais des temps de calcul plus court pour l'architecture multimodèle. Des prédictions acceptables sur un horizons de 1 à 5 jours ont été obtenues.

Le principal avantage de l'architecture multimodèle par rapport à l'architecture neuronale de type MLP est la possibilité d'avoir des sous-modèles de structure simple dont l'estimation paramétrique peut souvent se faire par des algorithmes d'optimisation linéaire. Par ailleurs, l'utilisation dans une architecture multimodèle d'un mode de partitionnement basé sur la classification floue permet de déterminer facilement le nombre de modèles locaux, alors que la détermination du nombre de neurones cachés pour une architecture MLP est une tâche plus difficile.

A l'issue de ce travail, quelques perspectives se dégagent.

L'architecture multimodèle hétérogène combinant la classification des données à leur représentation semble être un bon outil d'aide au diagnostic. En effet, les méthodes de décomposition de l'espace caractéristique du système permettraient de classer les types de défauts, tandis que les modèles locaux permettraient de les modéliser afin de prédire leur apparition, voire leur gravité.

L'utilisation d'un réseau de Kohonen pour la décomposition de l'espace de fonctionnement d'un multimodèle pourrait améliorer l'identification structurelle des modèles locaux, notamment quand le système comporte plusieurs variables. En effet, les cartes

auto-organisatrices de Kohonen constituent un outil très performant en classification, notamment dans des cas où la base d'apprentissage est très importante et comporte beaucoup de variables. La difficulté de l'utilisation de cette méthode réside dans la détermination des degrés d'appartenance pour une telle architecture. En fait, la méthode permet de détecter des groupes de données, mais pas la détermination d'un centre à partir duquel il est possible de calculer une distance permettant d'évaluer le degré d'appartenance d'une donnée au groupe identifié.

Annexes

Annexe A

Rappels sur les réseaux de neurones artificiels

Le cerveau humain est constitué d'un très grand nombre de neurones, environs 10^{11}, interconnectés entre eux de manière très dense et très complexe. Le nombre de connexions entre un neurone et ses voisins est évalué entre 10^3 et 10^4. Le neurone biologique, schématisé sur la figure A.1, est une cellule nerveuse spécialisée dans le traitement de l'information (signaux électriques). Il est composé d'un corps cellulaire (appelé aussi soma) et de deux sortes de filaments, les dendrites et l'axone pouvant véhiculer des messages d'un neurone vers un autre. Les dendrites représentent l'entrée du neurone et l'axone sa sortie. Les ramifications issues de l'axone d'un neurone sont reliées aux dendrites des neurones auxquels il est connecté par l'intermédiaire des synapses. Il se produit au niveau d'un neurone une intégration (sommation) des signaux reçus et si cette somme dépasse un certain seuil, le neurone émet à son tour un signal électrique vers d'autres neurones. Ce signal peut renforcer ou diminuer l'activité des neurones qui le reçoivent, selon que les synapses soient excitatrices ou inhibitrices.

La particularité du système neuronal humain est une organisation massivement parallèle, un mode de calcul et une mémoire distribuée, une capacité d'apprentissage, de généralisation et d'adaptation. Les RNA tentent d'intégrer l'ensemble de ces caractéristiques.

A.1 Historique

La tentative de modélisation du neurone biologique par McCulloch et Pitts [104] en 1943 a permis d'obtenir un modèle simpliste : le neurone formel. C'est un automate à seuil constitué d'une unité de traitement logique reliée aux entrées par l'intermédiaire

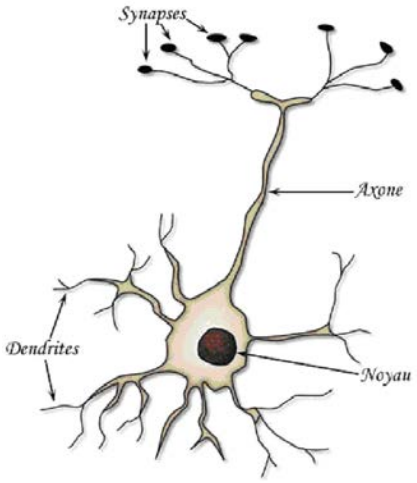

FIG. A.1 – Neurone biologique.

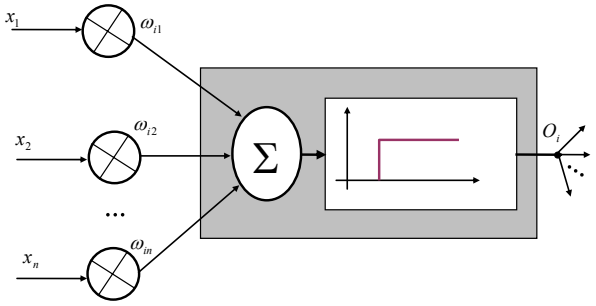

FIG. A.2 – Neurone formel.

de coefficients appelés poids synaptiques par analogie aux liaisons synaptiques dans le neurone biologique. Les signaux reçus par un neurone sont pondérés par les poids synaptiques et additionnés. Si la somme obtenue dépasse un certain seuil, la sortie du neurone est égale à 1, sinon elle vaut 0. La figure A.2 représente un neurone formel. La liaison synaptique entre le neurone formel d'indice i et l'entrée j est représentée par le coefficient ω_{ij}. Plusieurs neurones peuvent être associés pour former un réseau réalisant une tâche complexe. Les poids synaptiques doivent alors prendre des valeurs appropriées. L'apprentissage, processus de modification des valeurs des poids synaptiques, permet au réseau de réaliser la tâche qui lui est spécifiée.

En 1949, dans son livre intitulé « *The Organization of Behavior* » [105], D. Hebb présente une règle d'apprentissage postulant le renforcement de la connexion entre deux neurones simultanément actifs. Selon cette règle, « *quand une cellule $C1$ contribue fréquemment à exciter une cellule $C2$ à laquelle elle est connectée, un changement intervient au niveau de $C1$ ou de $C2$ pour que l'action de $C1$ sur $C2$ soit désormais plus efficace* ». Cette règle est utilisée encore de nos jours par de nombreux modèles de RNA.

Ce fût ensuite l'avènement du Perceptron, un réseau de neurones inspiré du système visuel et développé par F. Rosenblatt [106] en 1958. Ce réseau, premier système artificiel capable d'apprendre par expérience, comporte deux couches de neurones, la première étant la couche de perception et la deuxième la couche de prise de décision.

En 1960 B. Widrow a conçu le modèle de l'Adaline (*adaptive linear element*). C'est un réseau à un neurone qui effectue une combinaison linéaire de ses entrées. Son avantage

par rapport au perceptron est que le réseau n'est plus astreint à fournir une réponse binaire mais un éventail de valeurs possibles. L'extension multi-couche de l'Adaline par Widrow et Hoff a donné naissance au Madaline (*many adaline*). Un réseau Madaline est un réseau constitué de plusieurs niveaux de cellules Adaline. Le Madaline est le précurseur des réseaux multi-couches. Par la suite, des limites du perceptron notamment l'impossibilité de traiter des problèmes non linéaires et de connexité ont été mis en évidence par M. Minsky et S. Papert en 1969, ce qui a eu pour conséquence de ralentir la recherche dans le domaine des RNA.

Les travaux de T. Kohonen sur les mémoires associatives en 1972 avec des applications à la reconnaissance de forme, ont permis de relancer la recherche dans le domaine des RNA. Fukushima proposa le Cognitron en 1975. Les travaux d'Anderson, Ritz et Jones permirent le développement des mémoires associatives.

En 1982, les réseaux de Hopfield [107] (du nom de leur concepteur J. Hopfield) qui sont des réseaux complètement rebouclés (récurrents) ont vu le jour et ont donné une nouvelle impulsion à la recherche dans le domaine des RNA.

Avec l'apparition du perceptron multi-couches (*Multi-Layer Perceptron - MLP*) en 1986, capable de traiter les problèmes non linéaires, les RNA ont connu un grand succès qui ne cesse de croître jusqu'à nos jours.

D'autres types de réseaux ont également vu le jour mais généralement chaque réseaux se spécialise dans le traitement d'une tâche bien définie (e.g. classification, approximation de fonction, reconnaissance de forme, commande de processus, etc.).

A.2 Principe général des RNA

D'une manière générale, un RNA peut être défini comme un réseau complexe composé d'unités de calcul élémentaires (les neurones formels) interconnectées. Les neurones sont organisés en groupes ou couches et peuvent être connectés de différentes façons. C'est cette topologie de connexion entre les neurones qui définit l'architecture du réseau, liée à la tâche qu'il doit accomplir. Cette tâche est définie par le concepteur du réseau et est indiquée sous forme d'exemples comportant un ensemble de valeurs d'entrée et un ensemble de valeurs de sortie désirées. Ces exemples constituent la base d'apprentissage du réseau qui doit alors « apprendre » ces exemples et être capable de fournir des réponses correctes pour d'autres entrées inconnues. L'apprentissage est la procédure d'estimation

des paramètres du réseau lui permettant de satisfaire un critère de performance donné. Il est effectué suivant un algorithme propre à l'architecture du réseau.

Dans un RNA, les neurones du réseaux qui produisent la sortie sont appelés neurones de sortie et les autres neurones, les neurones cachés. La figure A.3 schématise un exemple de réseau de neurones ayant deux entrées, trois neurones cachés et deux neurones de sortie.

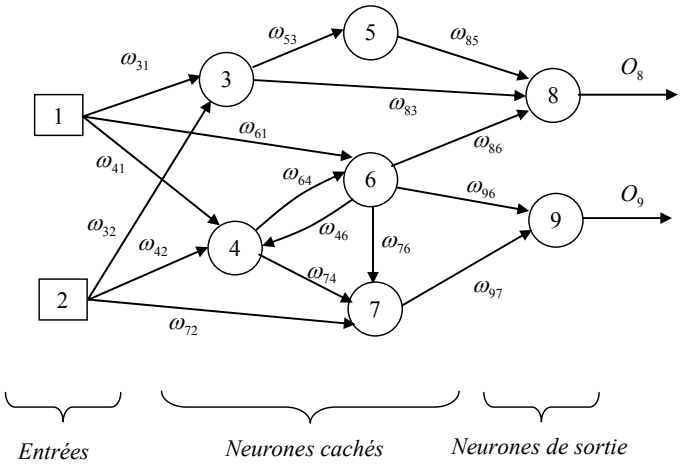

FIG. A.3 – Exemple d'un réseau de neurones artificiels ; ω_{ij} désigne le poids synaptique de la connexion entre le neurone i et l'unité j (qui peut être un neurone ou une entrée).

Un neurone se caractérise par sa manière d'évaluer son potentiel et son activité. Le potentiel du neurone est défini par sa fonction de combinaison qui évalue l'action des autres neurones qui lui transmettent une information. L'activité du neurone est déterminée par sa fonction d'activation (également appelée fonction de transfert ou de seuillage) qui reçoit comme argument le potentiel du neurone et qui permet d'introduire une non linéarité dans le fonctionnement du neurone. Les neurones diffèrent de par leur type de fonction de combinaison et d'activation. Par exemple dans le perceptron de Rosenblatt, la fonction de combinaison du neurone renvoie le produit scalaire entre le vecteur des entrées du neurone et le vecteur de ses poids synaptiques alors que la fonction d'activation est la fonction de Heaviside.

Les RNA sont subdivisés en deux grandes classes selon la topologie des connexions entre les neurones : les réseaux non récurrents (ou non bouclés) et les réseaux récurrents (ou bouclés). Chacune de ces classes comporte également différentes architectures.

Dans les réseaux non récurrents, appelés aussi réseaux « *feed forward* », l'information

circule des entrées vers la sortie sans possibilité de retour en arrière. Les réseaux non récurrents sont constitués du perceptron monocouche, du perceptron multicouche et des réseaux à fonctions radiales. Les réseaux à fonction radiale (ou *Radial Basis Function - RBF*) ont été proposés par Powell en 1985 [32]. Ils ont une architecture semblable à celle des MLP et sont surtout utilisés dans des problèmes de classification ou d'approximation de fonctions mais également en reconnaissance de forme (voir [108, 109, 110]).

Les réseaux récurrents (ou encore réseaux « *feed back* ») disposent d'une mémoire leur permettant de modéliser des systèmes dont l'évolution dépend des états dans lesquels ils se trouvaient aux instants antérieurs. Ils sont donc bien adaptés à la modélisation de systèmes dynamiques non linéaires. Ils peuvent être utilisés sous deux aspects différents :

comme mémoire associative : Pendant la phase d'apprentissage, le réseau atteint un certain état d'équilibre lui permettant de mémoriser les informations qu'on désire stocker et qui peuvent être restituées pendant la phase d'utilisation du réseau. Sous ce même aspect, le réseau récurrent peut aussi être utilisé pour résoudre des problèmes d'optimisation. A partir d'un état initial quelconque, le réseau évolue vers un état d'équilibre (stable) correspondant à l'optimum recherché.

comme système récursif : La sortie du réseau à un instant donné dépend des sorties et/ou des entrées aux instants antérieurs. Cette architecture est très utilisée pour la modélisation de série temporelles notamment pour la prédiction [58, 63, 111, 112].

L'apprentissage du réseau dépend naturellement de l'aspect sous lequel il est considéré. Dans le cas ou le réseau a pour rôle de trouver le ou les états d'équilibre du système (mémoire associative, optimisation), l'apprentissage sera non supervisé. Avec ce mode d'apprentissage, le réseau doit évoluer de lui même à partir de conditions initiales et se stabiliser à l'état d'équilibre au bout d'un certain temps. Dans le cas de la modélisation de systèmes récursifs, l'apprentissage sera supervisé comme dans un réseau non récurrent.

Parmi les réseaux récurrents, on distingue :

- les réseaux de Hopfield [107] utilisés pour résoudre des problèmes d'optimisation combinatoire ou pour mémoriser des états ;
- les cartes auto-organisatrices de Kohonen (Self Organizing Maps - SOM) très utilisées dans les problèmes de classification ;
- les réseaux ART (Adaptive Resonance Theory) [66].

Dans le cadre de la modélisation de systèmes dynamiques non linéaires, d'autres réseaux dynamiques ont vu le jour. Parmi ces réseaux, on distingue :

- les réseaux retardés dans le temps « *Time Delayed Neural Networks - (TDNN)* »

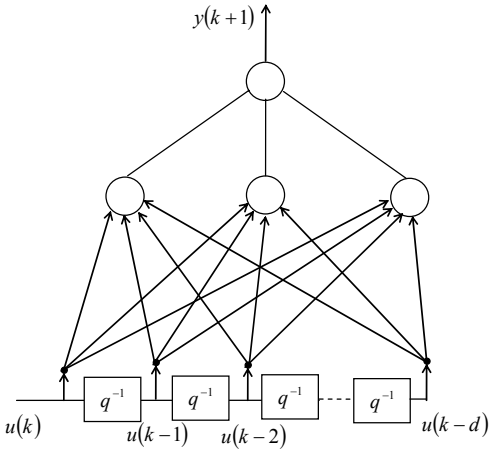

FIG. A.4 – Exemple de réseau TDNN.

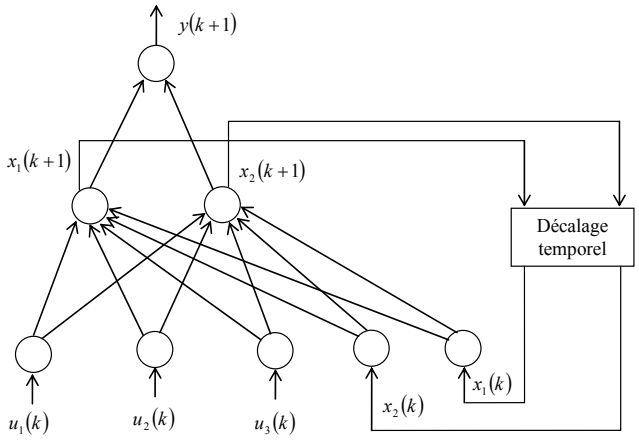

FIG. A.5 – Exemple de réseau d'Elman.

qui reçoivent les entrées du système à l'instant courant et aux instants antérieurs (figure A.4).

– les réseaux de neurones récurrents qui comportent une boucle interne avec décalages temporels : réseaux d'Elman (figure A.5), réseaux de Jordan (figure A.6) et réseaux complètement connectés ou réseaux de Williams-Zipser (figure A.7).

Les réseaux de neurones récurrents sont utilisés dans beaucoup d'applications (voir par exemple [62, 64, 113]), notamment celles nécessitant un réseau adaptatif où l'apprentissage du réseau se poursuit pendant la phase d'utilisation. Les poids du réseau sont alors constamment mis à jour à chaque fois que de nouvelles informations sont disponibles. Pour ces types de réseaux la base d'apprentissage est infinie (voir par exemple [18]).

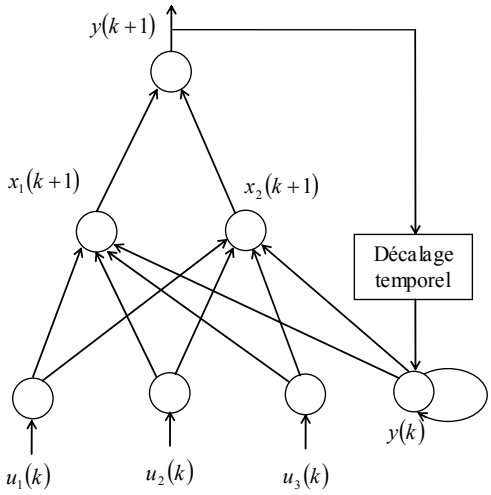

FIG. A.6 – Exemple de réseau de Jordan.

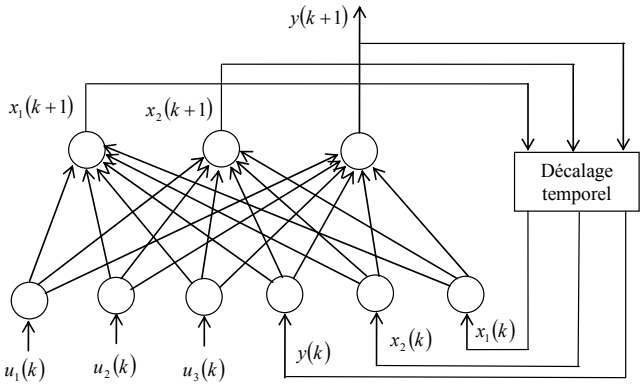

FIG. A.7 – Exemple de réseau de Williams-Zipser.

Annexe B

Mise en oeuvre d'une toolbox multimodèle

Les méthodologies présentées dans ce travail ont conduit au développement d'un outil logiciel pour la modélisation expérimentale de systèmes. Le développement est fait sous Matlab à travers une approche orientée objet. Après une brève présentation de l'approche orientée objet, l'outil logiciel développé dans ce travail est présenté.

B.1 Principe de la programmation orientée objet

Comme son nom l'indique, la programmation orientée objet (POO) utilise la notion d'objet pour la représentation et la manipulation de données. Un objet est une entité concrète (ou instance) d'une abstraction qu'on appelle classe. La classe formalise l'ensemble des objets ayant une structure commune et un comportement de même nature. Elle contient les membres données (représentation des attributs des objets de la classe) et les membres fonctions ou méthodes (qui sont des procédures permettant de manipuler les données). Un objet est caractérisé par son état (déterminé par la valeur de ses attributs) et un ensemble d'opérations - les méthodes - qui permettent de manipuler cet état. La valeur des attributs varie d'un objet de la classe à un autre et détermine le comportement de l'objet.

Le modèle à objet repose sur quatre principes : l'abstraction, l'encapsulation, la modularité et la hiérarchisation. l'abstraction permet de regrouper des entités de même nature au sein d'une structure générale, abstraite - la classe. L'encapsulation permet d'occulter la composition interne d'un objet. L'objet est vu comme une « boîte noire » et en fonction des données qu'on lui attribue, il a un comportement bien défini. Un utilisateur

de l'objet ignore sa composition et ne s'intéresse qu'à ses fonctionnalités. La modularité (ou décomposition en modules) assure la facilité d'extension des fonctionnalités, de réutilisabilité pour de nouvelles applications et de compatibilité. La hiérarchisation permet de définir les liens entre les abstractions. Une abstraction peut dériver d'une autre en héritant toutes ou certaines de ces spécificités. Cette technique simplifie la création de nouvelles classes en permettant de réutiliser des classes existantes et d'ajouter de nouveaux éléments ou redéfinir certains comportements. La possibilité de redéfinir un comportement est appelé polymorphisme.

Nous avons utilisé certains aspects de la programmation orientée objet décrits ici pour le développement de l'outil logiciel d'identification de système par multimodèle et réseaux de neurones MLP.

B.2 Présentation de l'outil logiciel

B.2.1 Principe de la mise en eouvre

Dans ce travail, nous avons montré qu'une structure multimodèle permet de représenter tout système dynamique non linéaire. Pour identifier un système donné, pour lequel il n'existe pas de connaissances *a priori* sur la structure du modèle à utiliser, chacune des structures non linéaires NFIR, NARX, NOE et NARMAX est une solution potentielle. Ces structures peuvent donc être considérées comme des entités concrètes d'une abstraction (classe) de modèle non linéaire (multimodèle ou MLP). Les caractéristiques de ces structures ainsi que les méthodes d'apprentissage et d'évaluation des performances permettent de formaliser la classe.

Nous avons défini une classe **multimodel** qui formalise un modèle dynamique de type « boîte noire » (linéaire ou non linéaire) par une approche multimodèle. Les principales données membres de la classe **multimodel** sont :

Les paramètres caractéristiques d'un multimodèle : Ce sont principalement le nombre de modèles locaux que comporte la structure multimodèle, le nombre de variables caractéristiques (des non linéarités du systèmes), les paramètres des modèles locaux et les paramètres caractéristiques des zones de validité des modèles locaux.

Les paramètres de configuration : Ces paramètres indiquent la classe du modèle à identifier (NFIR, NARX, NOE ou NARMAX), la structure des modèles locaux

(linéaire, polynômiale, neuronale, hybride, etc) les ordres et retard des variables du modèle pour la constitution du vecteur de régression, les paramètres des méthodes de partitionnement de l'espace de fonctionnement du système, le critère de sélection de modèle à utiliser pour la détermination de la meilleure topologie du modèle, les paramètres d'apprentissage.

Les paramètres de performances : Ce sont les variables contenant les valeurs des critères de sélection de modèles, des erreurs d'apprentissage, des durées d'apprentissage.

Les principales méthodes de la classe **multimodel** sont :

Le constructeur de la classe : C'est une fonction qui permet de créer un objet de la classe multimodèle et de l'initialiser.

Les méthodes d'accès : Ce sont des fonctions qui permettent d'accéder aux informations relatives à l'état d'un objet multimodèle, notamment aux valeurs de certains de ses champs.

Les méthodes d'altération : Ce sont des fonctions qui permettent l'accès à un objet multimodèle en vue de modifier son état (les valeurs de certains de ses champs).

Fonction d'ajout d'un modèle local : C'est une fonction qui permet d'ajouter une nouveau modèle local, après avoir reconfiguré l'espace de fonctionnement en ajoutant une nouvelle zone.

Fonction de suppression d'un modèle local : C'est une fonction qui supprime un modèle local et la zone de fonctionnement qui lui est associé.

Fonction d'apprentissage d'un système : Cette fonction réalise l'estimation paramétrique du multimodèle.

Fonctions de détermination des zones de validité : Ces fonctions permettent la décomposition de l'espace de fonctionnement du système en plusieurs sous-espaces pour lesquels sont élaborés des modèles locaux. Ce sont notamment les fonctions de décomposition en grille, de décomposition hiérarchique orthogonale aux axes et de décomposition par les algorithmes des « *fuzzy-c-means* », « *Gustafson - Kessel* » et « *subtractive clustering* ».

Fonction de calcule des degrés d'activation des modèles locaux : Permet de déterminer les degrés d'activation des modèles locaux en fonction de leur zone de validité.

Fonction de constitution du vecteur de régression : Cette fonction construit le vecteur de régression en fonction de la structure du modèle à identifier et des variables caractéristiques des non linéarités du système.

Fonction de simulation du multimodèle : Cette fonction permet de simuler le modèle identifié.

Fonction d'évaluation des performances du multimodèle : C'est une fonction qui permet d'évaluer les performances du modèle.

B.2.2 Interface graphique

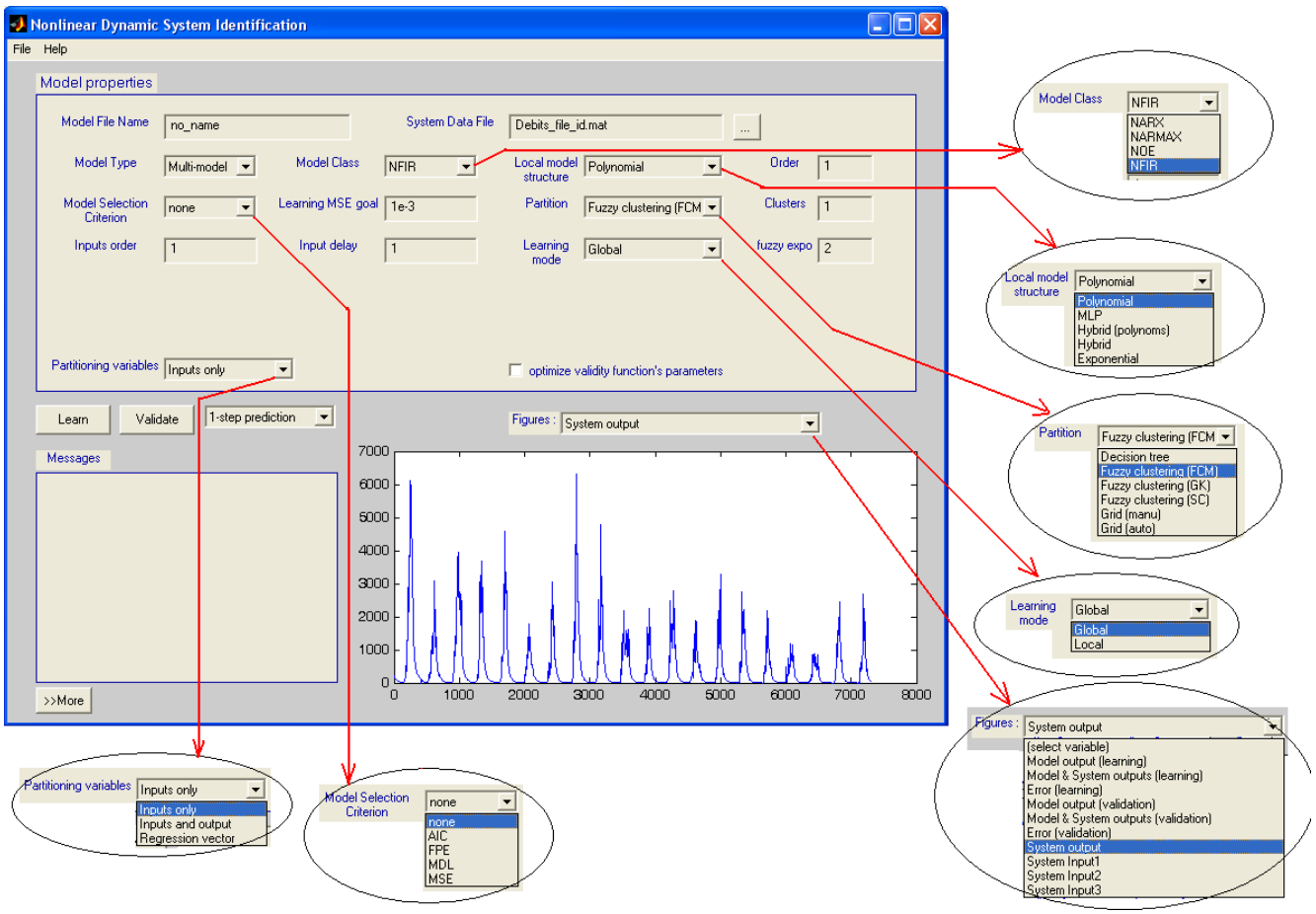

FIG. B.1 – Interface graphique pour l'identification de systèmes.

L'interface graphique de l'outil développé pour l'identification de systèmes est présenté sur la figure B.1. L'interface permet le choix de l'architecture multimodèle ou neuronale de type MLP, de charger la base de données du système à identifier, de faire la saisie

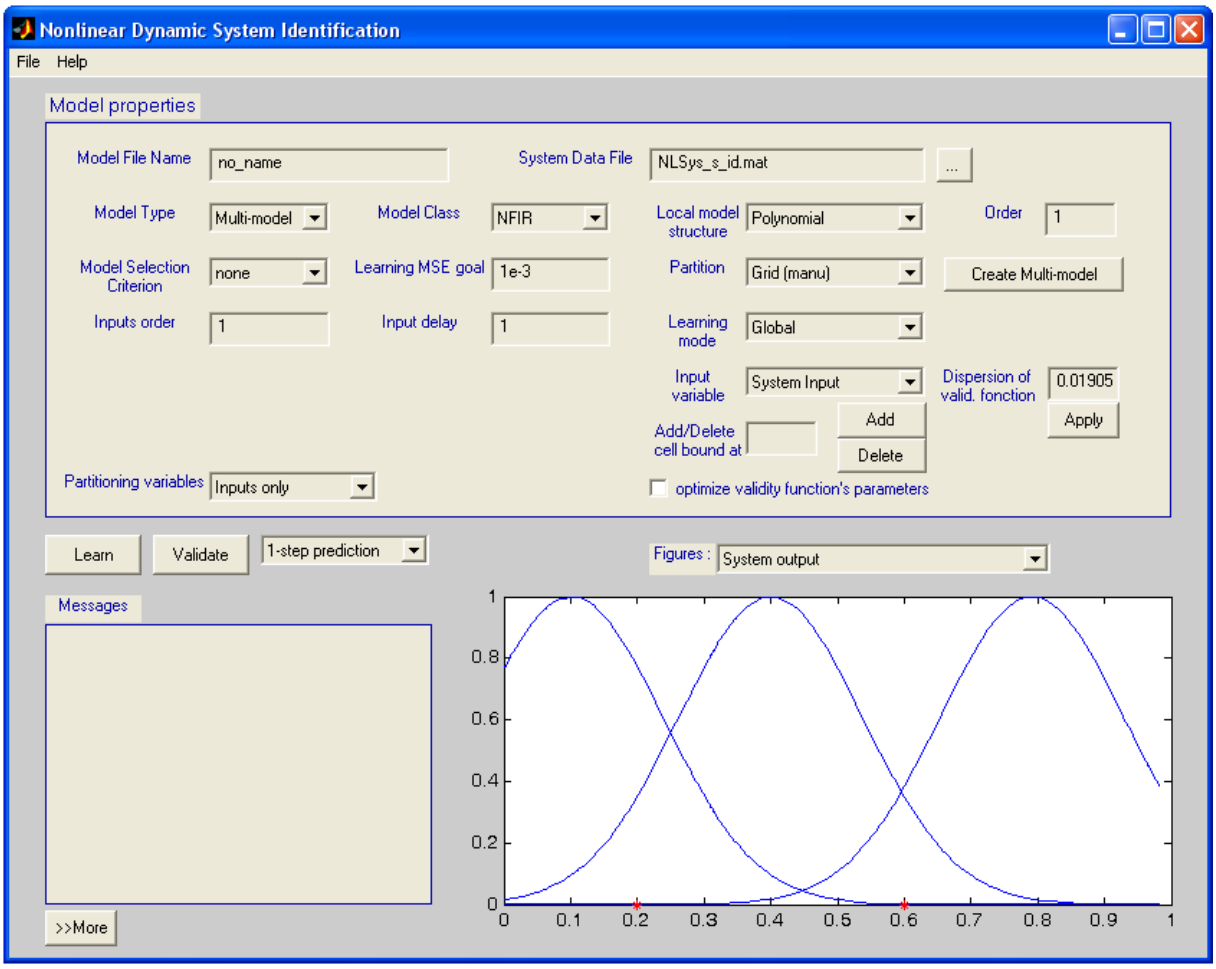

FIG. B.2 – Exemple d'utilisation du mode de partitionnement en grille avec possibilité d'ajouter des partitions, de les déplacer ou de les supprimer à l'aide de la souris.

des différents paramètres liés aux architectures, de visualiser les différentes variables (mesurées ou estimées). La figure B.2 illustre un exemple d'identification par multimodèle avec des modèles locaux affines (polynômes de degré 1) et un mode de partitionnement grille, permettant de créer les partitions manuellement, de les modifier, d'en rajouter ou supprimer. La figure B.3 montre un exemple d'identification d'un système par un multimodèle avec des modèles locaux affines (polynômes de degré 1) et un mode de partitionnement par « *subtractive clustering* ».

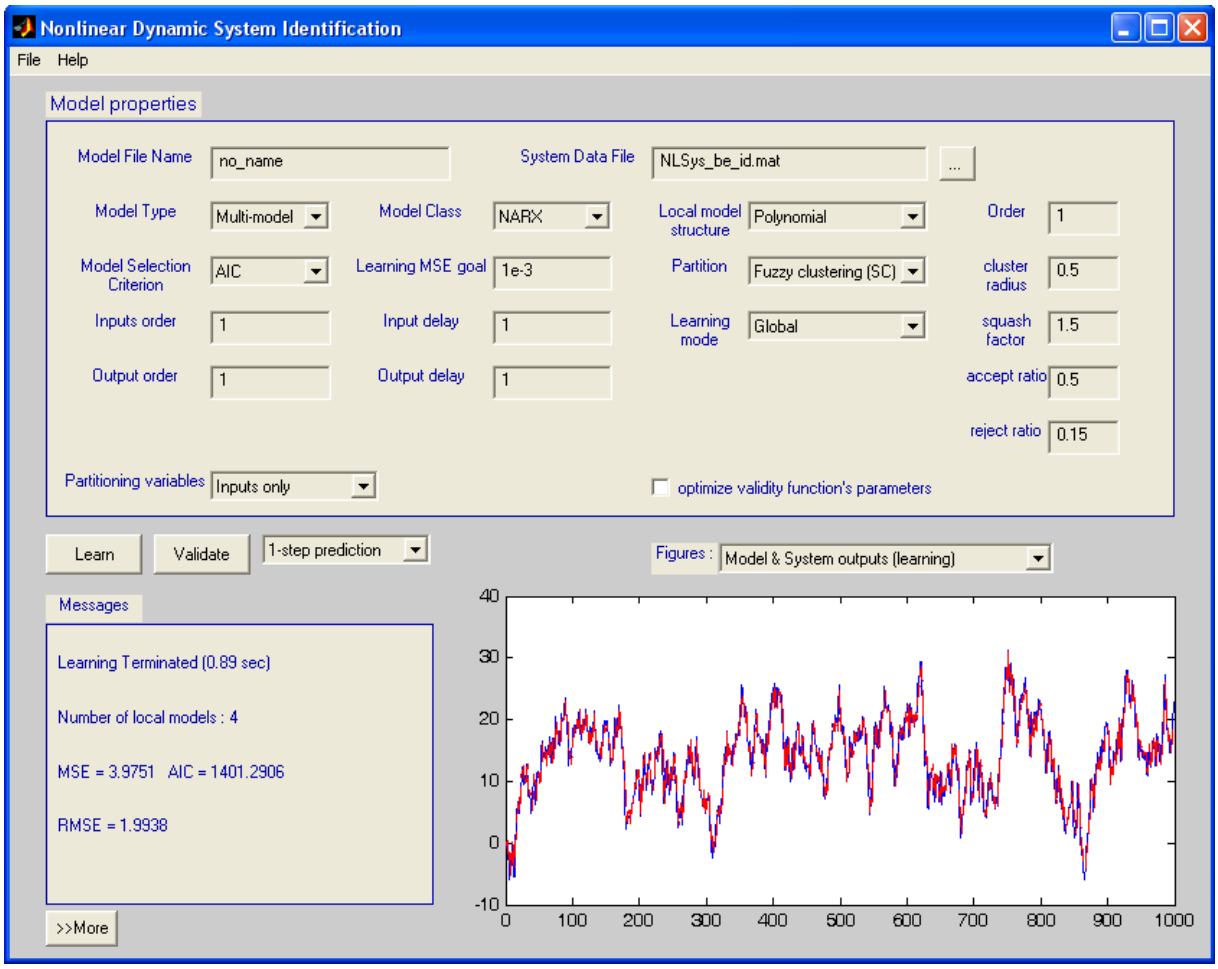

FIG. B.3 – Exemple d'identification d'un système par un multimodèle avec des modèles locaux affines (polynômes de degré 1) et un mode de partitionnement par « *subtractive clustering* ».

Bibliographie

[1] M. Schetzen. *The Voltera and Wiener Theories of Nonlinear Systems*. John Wiley, New York, 1980.

[2] N. Wiener. Non linear problems in random theory. *The Technology Press MIT, and John Wiley*, New York, 1958.

[3] C.H. Lu and C.C Tsai. Generalized predictive control using recurrent fuzzy neural networks for industrial processes. *Journal of process control*, 17 :83–92, 2007.

[4] S. Srivastava, M. Sing, M. Hanmandlu, and A.N. Jha. New fuzzy wavelet neural networks for system identification and control. *Applied Soft Computing*, 6 :1–17, 2005.

[5] Krzysztof Patan and Thomas Parisini. Identification of neural dynamic models for fault detection and isolation : the case of a real sugar evaporation process. *Journal of Process Control*, 15 :67–79, 2005.

[6] H.R. Maier and G.C. Dandy. The use of artificial neural networks for the prediction of water quality parameters. *Water Ressources Research*, 32(4) :1013–1022, 1996.

[7] K. Gasso. *Identification de systèmes dynamiques non linéaires : approche multi - modèle*. PhD thesis, INPL Nancy, 2000.

[8] K. Madani and L. Thiaw. Self-organizing multi-modeling : A different way to design intelligent predictors. *Neurocomputing*, 70 :2836–2852, 2007.

[9] H. Vernieuwe, O. Georgieva, B. D. Baets, V. R. N. Pauwels, and N. E. C. Verhoestand F. P. D. Troch. Comparison of data-driven takagi-sugeno models of rainfall-discharge dynamics. *Journal of Hydrology*, XX :1–14, 2004.

[10] L. Thiaw, M. Rybnik, R. Malti, A. Chebira, and K. Madani. A comparative study between a multi-models based approach and an artificial neural network based tech-

nique for nonlinear systems identification. *International Scientific Journal of Computing*, 3(1) :66–74, 2004.

[11] G. Dreyfus. *Réseaux de Neurones - Méthodologie et applications*. Eyrolles, 2002.

[12] B. Irie and S. Miyake. Capabilities of tree-layered perceptrons. *Proceedings of the IEEE Second International Conference on Neural Networks (San Diego)*, 1 :641–647, 1988.

[13] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematical Control Signals Systems*, 2 :303–314, 1989.

[14] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5) :359–366, 1989.

[15] H. Akaike. Fitting autoregressive models for prediction. *Annal of the Institute of Statistical Mathematics*, 21 :243 – 247, 1969.

[16] T.A. Johansen and B A. Foss. Nonlinear local model representation for adaptive systems. In *IEEE Int. Conf. on Intelligent Control and Instrumentation.*, volume 2, pages 677 – 682, Singapore, 1992.

[17] P.J. Werbos. Backpropagation through time : What it does and how to do it. *Proceedings of the IEEE*, 78(10) :1550–1560, 1990.

[18] R.J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1 :270–280, 1989.

[19] K. Madani and L. Thiaw. Multi-model based identification : Application to nonlinear dynamic behavior prediction. In *International Multi-Conference on Advanced Computer System and Computer Information Systems and Industrial Management (ACS-CISIM)*, Elk, Poland, June 2005.

[20] H. Vernieuve, B. De Baets, and N.E.C. Verhoest. Comparison of clustering algorithms in the identification of takagi-sugeno models : A hydrological case study. *Fuzzy Sets and Systems*, 157 :2876–2896, 2006.

[21] R. Babuska. *Fuzzy modelling for control*. Kluwer Academic Publishers Boston/Dordrecht/London, 1998.

[22] L. Thiaw, K. Madani, R. Malti, and G. Sow. Implementation of recurrent multimodels for system identification. In *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Angers, France, May 09-12 2007.

[23] L. Thiaw, K. Madani, R. Malti, and G. Sow. NARMAX multi-model based dynamic nonlinear systems behaviour prediction. In *Workshop on Advanced Control and Diagnosis (IAR-ACD)*, Mulhouse, France, November 17-18, 2005.

[24] G. Box and G. Jenkins. Time series analysis, forecasting and control. pages 532–533, San Francisco, Holden Day, 1970.

[25] L. Ljuing. *System Identification : Theory for the User*. Englewood Cliffs, NJ, 1987.

[26] D. Urbani. Méthodes statistiques de sélection d'architectures neuronales : application à la conception de modèles de processus dynamiques. Master's thesis, Université Paris 6, 1995.

[27] T. Cibas, F.F. Soulié, P. Gallinari, and S. Raudys. Variable selection with neural networks. *NEUROCOMPUTING*, 12 :223–248, 1996.

[28] S. Wu and M.J. Er. Dynamic fuzzy neural networks - a novel approach to function approximation. *IEEE Transactions on systems, man, and cybernetics - part B*, 30(2) :358–364, 2000.

[29] N. Li, S. Y. Li, and Y. G. Xi. Multi-model predictive control based on the takagi-sugeno fuzzy models : a case study. *Information Sciences*, 165 :247–263, 2004.

[30] L.X. Wang. *A course in fuzzy systems and control*. 1997.

[31] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *IEEE Trans. on Systems Man and Cybernetic*, 15 :116–132, 1985.

[32] M.J.D. Powell. Radial basis functions for multivariable interpolation : A review. *In IMA Conference on Algorithm for the approximation of Functions and Data, RMCS, Shrivenham, UK*, pages 143–167, 1985.

[33] N.B. Karayiannis and S. Behnkec. *New radial basis neural networks and their application in a large-scale handwritten digit recognition problem*, chapter 2. CRC Press, 2000.

[34] J. Park and I.W. Sandberg. Approximation and radial-basis-function networks. *Neural Computation*, 5 :305–316, 1993.

[35] S. Dablemont, G. Simon, A. Lendasse, A. Ruttiens, F. Blayo, and M. Verleysen. Time series forecasting with SOM and local-non-linear models - Application to the DAX30 index prediction. In *Workshop on Self-Organizing Maps*, pages 340–345, Hibikino, Japan, september 2003.

[36] A. Lendasse, M. Cottrell, V. Wertz, and M. Verleysen. Prediction of electric load using kohonen maps - application to the Polish electricity consumption. In *American Control Conference*, Anchorage, AK, May 2002.

[37] A. Bezerianos, S. Papadimitriou, and D. Alexopoulos. Radial basis function neural networks for the characterization of heart rate variability dynamics. *Artificial Intelligence in Medicine*, 15 :215–234, 1999.

[38] V.N. Vapnik. *The nature of statistical learning theory*. Springer, 1995.

[39] J. Shena, Y.R. Syaub, and E.S. Lee. Support vector fuzzy adaptive network in regression analysis. *Computers and Mathematics with Applications*, 54 :1353–1366, 2007.

[40] A. B. Gandhia, J. B. Joshia, V. K. Jayaramanb, and B.D. Kulkarni. Development of support vector regression (SVR)-based correlation for prediction of overall gas hold-up in bubble column reactors for various gas-liquid systems. *Chemical Engineering Science*, 62 :7078–7089, 2007.

[41] B.R. Chang and H.F. Tsai. Forecast approach using neural network adaptation to support vector regression grey model and generalized auto-regressive conditional heteroscedasticity. *Expert Systems with Applications*, 34 :925–934, 2008.

[42] Y. Gao and M.J. Er. Narmax time series model prediction : feedforward and recurrent fuzzy neural network approaches. *Fuzzy Sets and Sytems*, 150 :331–350, 2005.

[43] Wei Z. Yang, Yam L.H., and Cheng L. Narmax model representation and its application to damage detection for multi-laer composites. *Composite Structures*, 68 :109–117, 2005.

[44] T. A. Johansen and B. A. Foss. Constructing narmax models using armax models. *Int. J. Control*, 58 :1125 – 1153, 1993.

[45] H.K. Fung Eric, Y.K.Wong, H.F. Ho, and M.P. Mignolet. Modelling and prediction of machining errors using armax and narmax structures. *Applied Mathematical Modelling*, 27 :611–627, 2003.

[46] M. Stone. An asymptotic equivalence of choice of model by cross-validation and akaike's criterion. *J. R. Stat. Soc.*, 38 :44–47, 1977.

[47] L. Breiman. Heuristics of instability and stabilization in model selection. *Annals of Statistics*, 24 :2350–2383, 1996.

[48] B. Efron and R.J. Tibshirani. Improvements on cross-validation : The .632+ bootstrap method. *J. of the American Statistical Association*, 92 :548–560, 1997.

[49] H. Akaike. A new look at the statistical model identification. *IEEE Trans on Automatic control*, 9 :716 – 723, 1974.

[50] J. Rissanen. Modeling by shortest data description. *Automatica*, 14 :465 – 471, 1978.

[51] Maïza Bekara. *Optimisation de critères de choix de modèles pour un faible nombre de données*. PhD thesis, Université de Paris XI, Novembre 2004.

[52] J.C. Carmona. *Contribution à l'identification et la commande robuste de systèmes dynamiques complexes : de la théorie à l'application en mécanique*. HDR, Université de la Méditerranée (Aix-Marseille II), Septembre 2004.

[53] T. A. Johansen. Robust identification of takagi-sugeno-kang fuzzy models using regularization. In *IEEE Conf. Fuzzy Systems, New Orleans*, 1996.

[54] T. A. Johansen. On tikhonov regularization, bias and variance in nonlinear system identification. *Automatica*, 33 :441–446, 1997.

[55] J.C. Nash. *Compact numerical Methods for Computers, linear algebra and function minimisation*. 1990.

[56] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 7 :155–162, 1964.

[57] M.N. Eskander. Neural network controller for a permanent magnet generator applied in a wind energy conversion system. *Renewable Energy*, 26 :463–477, 2002.

[58] A.A. Vartak, M. Georgiopoulos, and G.C. Anagnostopoulos. On-line gauss-newton-based learning for fully recurrent neural networks. *Nonlinear Analysis*, 63 :867–876, 2005.

[59] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1 :4–27, 1990.

[60] M. Hüsken and P. Stagge. Recurrent neural networks for time series classification. *Neurocomputing*, 50 :223–235, 2003.

[61] Tsung-Lin Lee. Back-propagation neural network for the prediction of the short-term storm surge in taichung harbor, taiwan. *Engineering Applications of Artificial Intelligence*, 21 :63–72, 2008.

[62] T.G. Barbounis and J.B. Theocharis. Locally recurrent neural networks for long-term wind speed and power prediction. *Neurocomputing*, 69 :466–496, 2006.

[63] U. Konur and A. Okatan. Time series prediction using recurrent neural network architectures and time delay neural networks. *ENFORMATIKA*, pages 1305–1313, 2004.

[64] Q.P. Hu, M. Xie, S.H. Ng, and G. Levitin. Robust recurrent neural network modeling for software fault detection and correction prediction. *Reliabilituy engineering & System Safety*, 92 :332–340, 2007.

[65] I. Güler and E.D. Übeyli. A mixture of experts network structure for modeling doppler ultrasound blood flow signals. *Computers in Biology and Medecine*, 35 :565–582, 2005.

[66] G. Carpenter and S. Grossberg. *Neural dynamics of category learning and recognition : vigilance, memory consolidation and amnesia.* (AAAS Symposium series), 1986.

[67] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature (London)*, 323 :533 – 536, 1986.

[68] P.D. Mandic and J.A Chambers. *Recurrent neural networks for prediction - Learning algorithms, architectures and stability.* JOHN WILEY & SONS, LTD, 2001.

[69] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, Denver 1989, 1990. Morgan Kaufmann, San Mateo.

[70] M. Frean. The upstart algorithm : A method for constructing and training feedforward neural networks. *Neural Computation*, 2 :198–209, 1990.

[71] A. S. Weigend and D. E. Rumelhart. The effective dimension of the space of hidden units. In *International Joint Conference on Neural Networks, Singapore*, pages 2069–2074, 1991.

[72] Y. L. Cun, J. Denker, and S. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pages 598–605. Morgan Kaufman, 1990.

[73] L. Ozkan and M.V. Kothare. Stability analysis of a multi-model predictive control algorithm with application to control of chemical reactors. *Journal of Process Control*, 16 :81–90, 2006.

[74] F. Di Palma and L.Magni. A multimodel structure for model predictive control. *Annual Reviews in Control*, 28 :47–52, 2004.

[75] R. Murray-Smith and T.A. Johansen. *Multiple model approach to modelling and control.* Taylor and Francis, 1997.

[76] T.A. Johansen and A.B. Foss. Identification of non-linear system structure and parameters using regime decomposition. *Automatica*, 31(2) :321–326, 1995.

[77] M. Sugeno and G.T. Kang. Structure identification of fuzzy model. *Fuzzy sets and systems*, 28 :15–33, 1988.

[78] C.T. Sun. Rule-base structure identification in an adaptive-network-based fuzzy inference system. *IEEE Trans. on Fuzzy Systems*, 2(1) :64–73, 1994.

[79] R. Murray-Smith. *A local model network approach to nonlinear modelling*. PhD thesis, University of Strathclyde, Computer Science Departement, 1994.

[80] S. Ernst. Hinging hyperplane trees for approximation and identification. In *Proc. Of the* 37^th *IEEE Conf. on Decision and Control*, Tampa, Florida, USA, 1998.

[81] R. Babuska and H. B. Verbruggen. *Fuzzy sets methods for local modelling and control*, chapter 2. Taylor and Francis, 1997.

[82] R. Babuska and H. B. Verbruggen. Identification of composite linear models via fuzzy clustering. In *Proc. of ECC, Italy*, pages 1207–1212, 1995.

[83] R. Shorten and R. Murray-Smith. *Multiple model approches to modelling and control*, chapter Side-effects of normalizing basis functions in local model networks. Taylor and Francis, 1997.

[84] R. Babuska, P.J. van der Veen, and U. Kaymak. Improved covariance estimation for gustafson-kessel clustering. In *Proceedings of FUZZ-IEEE, Honolulu, Hawai, USA*, pages 1081–1085, 2002.

[85] J. Bezdek. Pattern recognition with fuzzy objective function algorithms. In *Plenum Press, USA*, 1981.

[86] D. Gustafson and W. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *Proceedings of IEEE CDC, San Diego, CA, USA*, pages 761–766, 1979.

[87] S.L. Chiu. Fuzzy model identification based on cluster estimation. *Intell. Fuzzy Sets*, 2 :267–278, 1994.

[88] K. Demirli, S. Cheng, and P. Muthukumaran. Subtractive clustering based modeling of job sequencing with parametric search. *Fuzzy Sets and Systems*, 137(2) :235–270, 2003.

[89] A. Subasi. Eeg signal classification using wavelet feature exrtraction and a mixture of expert model. *Expert Systems with Application*, 32 :1084–1093, 2007.

[90] S. Li, Q. Yin, P. Guo, and M.R. Lyu. A hierarchical mixture model for software reliability prediction. *Applied mathematics and computation*, 2006.

[91] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E Hinton. Adaptive mixtures of local experts. *Neural. Comput*, 3(1) :79–87, 1991.

[92] M.I. Jordan and R.A Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 6(2) :181–214, 1994.

[93] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of theRoyal Statistical Society : Series B,*, 39(1) :1–38, 1977.

[94] M. Welling. The expectation maximization (EM) algorithm. California Institute of Technology 136-93 Pasadena,CA 91125 welling@vision.caltech.edu.

[95] S. Borman. The expectation maximization algorithm a short tutorial. em-tut@seanborman.com, july 2004.

[96] M. Farinas and C. E. Pedreira. Mixture of experts and local-global neural networks. In *Poceedings - European Symposium on Artificial Neural Networks (ESANN)*, pages 331–336, Bruges, Belgium, 2003.

[97] M. Zounemat-Kermani and M. Teshnehlab. Using adaptive neuro-fuzzy inference system for hydrological time series prediction. *Applied Soft Computing*, 8 :928–936, 2008.

[98] R. Orjuela, D. Maquin, and J. Ragot. Identification des systèmes non linéaires par une approche multi-modèle à états découplés. *Journées Identification et Modélisation Expérimentale JIME'2006 - 16 et 17 novembre - Poitiers*, 2006.

[99] T.A. Clair and J.M. Ehrman. Using neural networks to assess the influence of changing seasonal climates in modifying discharge, dissolved organic carbon, and nitrogen export in eastern canadian rivers. *Water Resources Research*, 34(3) :447–455, 1998.

[100] S. Sakakima, T. Kojiri, and K. Itoh. Real-time reservoir operation with neural nets concept. In *Applications of artificial intelligence in engineering VII. Computational Mechanics Publicationns*, pages 502–514, Southampton, Angleterre, 1992.

[101] WHYCOS. Renforcement des capacités nationales et régionales d'observation, transmission et traitement de données pour contribuer au développement durable du bassin du fleuve sénégal. [en ligne]. [ref du 19 décembre 2007]. `http://www.whycos.org/IMG/pdf/Senegal_HYCOS_september_2007.pdf`.

[102] OMVS. Caractéristiques physiques du fleuve sénégal. [en ligne]. [ref du 19 décembre 2007]. `http://www.omvs-soe.org/physique.htm`.

[103] N. Flipo, S. Even, M. Poulin, and S. Théry. Modélisation des transfert d'azote sur le bassin du Grand Morin avec la plate-forme de modélisation cawaqs. Technical report, Centre d'Information Géologique, ENSMP, 33 rue Saint-Honoré, 77305 Fontainebleau.

[104] W. S McCulloch. and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, (5) :115–133, 1943.

[105] D. 0. Hebb. *The Organization of Behavior : A neuropsychological Theory.* Wiley, New York, 1949.

[106] F. Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65 :386–408, 1958.

[107] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of National Academy of Sciences*, 79 :2554–2558, 1982.

[108] D.J. Burr. Experiments on neural net recognition of spoken and written text. *IEEE Trans. Acoust., Speech, Signal Processing*, 36(7) :1162–1168, july 1988.

[109] R.P. Gorman and T.J. Sejnowski. Learned classification of sonar targets using a massively parallel network. *IEEE Trans. Acoust., Speech, Signal Processing*, 36(7) :1135–1140, july 1988.

[110] T.J. Sejnowski and C.R. Rosenberg. Parallel networks that learn to pronounce english text. *Complex Syst.*, 1 :145–168, 1987.

[111] M. Bielikova. Recurrent neural network training with the extended kalman filter. *IIT. SRC*, pages 57–64, April 27 2005.

[112] Y. Cheng, T.W. Karjala, and D.M. Himmelblau. Closed loop nonlinear process identification using internal recurrent nets. *Neural Networks*, 10(3) :573–586, 1997.

[113] A. Savran. An adaptive recurrent fuzzy system for nonlinear identification. *Applied Soft Computing*, 2006.

Bibliographie de l'auteur

Chapitre de livre

[1] K. Madani, L. Thiaw, R. Malti et G. Sow : Multi-modeling : a different way to design intelligent predictors. *In Lecture Notes in Computer Science (LNCS 3512) : Computational Intelligence and Bio-inspired Systems*, pages 976 – 984. J. Cabestany, A. Prieto, and D.F. Sandoval, Springer Verlag Berlin Heidelberg, june 2005. ISBN 3-540-26208-3.

[2] K. Madani et L. Thiaw : Multi-model based identification : Application to nonlinear dynamic behavior prediction. *In Image Analysis, Computer Graphics, Security Systems and Artificial intelligence Applications*, pages 365–375, 2005. ISBN 83-87256-86-2.

Publication dans des revues internationales

[1] K. Madani et L. Thiaw : Self-organizing multi-modeling : A different way to design intelligent predictors. *Neurocomputing*, 70:2836–2852, 2007.

[2] L. Thiaw et K. Madani : Self-organizing multi-model based identification : Application to nonlinear dynamic systems' behavior prediction. *Image Processing & Communication Journal*, 10:63–74, 2006.

[3] L. Thiaw, M. Rybnik, R. Malti, A. Chebira et K. Madani : A comparative study between a multi-models based approach and an artificial neural network based technique for nonlinear systems identification. *International Scientific Journal of Computing*, 3(1):66–74, 2004.

Communication dans des congrès internationaux avec actes et comité de lecture

[1] L. Thiaw, K. Madani, R. Malti et G. Sow : Implementation of recurrent multi-models for system identification. *In International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Angers, France, May 09-12 2007.

[2] L. Thiaw, K. Madani, R. Malti et G. Sow : NARMAX multi-model based dynamic nonlinear systems behaviour prediction. *In Workshop on Advanced Control and Diagnosis (IAR-ACD)*, Mulhouse, France, November 17-18, 2005.

[3] K. Madani et L. Thiaw : Multi-model based identification : Application to non-linear dynamic behavior prediction. *In International Multi-Conference on Advanced Computer System and Computer Information Systems and Industrial Management (ACS-CISIM)*, Elk, Poland, June 2005.

[4] L. Thiaw, R. Malti et K. Madani : A multiple models approach for nonlinear systems identification : Comparison between ANN based and conventional implementation. *In Proceeding Book of International Conference on Neural Networks and Artificial Intelligence - ICNNAI*, pages 210–214, Minsk, Byelorussia, 2003.

[5] L. Thiaw, R. Malti et K. Madani : First comparative study between linear and neural based local models in multi-modelling. *IAR-ICD Workshop on Advances Control and Diagnosis (ACD), Duisburg, Germany*, November 26-27 2003.

Résumé

Cette étude traite de l'identification de système dynamique non-linéaire. Une architecture multimodèle capable de surmonter certaines difficultés de l'architecture neuronale de type MLP a été étudiée. L'approche multimodèle consiste à représenter un système complexe par un ensemble de modèles de structures simples à validité limitée dans des zones bien définies. A la place de la structure affine des modèles locaux généralement utilisée, cette étude propose une structure polynômiale plus générale, capable de mieux appréhender les non-linéarités locales, réduisant ainsi le nombre de modèles locaux. L'estimation paramétrique d'une telle architecture multimodèle peut se faire suivant une optimisation linéaire, moins coûteuse en temps de calcul que l'estimation paramétrique utilisée dans une architecture neuronale. L'implantation des multimodèles récurrents, avec un algorithme d'estimation paramétrique plus souple que l'algorithme de rétro-propagation du gradient à travers le temps utilisé pour le MLP récurrent a également été effectuée. Cette architecture multimodèle permet de représenter plus facilement des modèles non-linéaires bouclés tels que les modèles NARMAX et NOE. La détermination du nombre de modèles locaux dans une architecture multimodèle nécessite la décomposition (le partitionnement) de l'espace de fonctionnement du système en plusieurs sous-espaces où sont définies les modèles locaux. Des modes de partitionnement flou (basé sur les algorithmes de« fuzzy-c-means », de « Gustafson et Kessel » et du « subtractive clustering ») ont été présentés. L'utilisation de telles méthodes nécessite l'implantation d'une architecture multimodèle où les modèles locaux peuvent être de structures différentes : polynômiales de degrés différents, neuronale ou polynômiale et neuronale. Une architecture multimodèle hétérogène répondant à ses exigences a été proposée, des algorithmes d'identification structurelles et paramétriques ont été présentés. Une étude comparative entre les architectures MLP et multimodèle a été menée. Le principal atout de l'architecture multimodèle par rapport à l'architecture neuronale de type MLP est la simplicité de l'estimation paramétrique. Par ailleurs, l'utilisation dans une architecture multimodèle d'un mode de partitionnement basé sur la classification floue permet de déterminer facilement le nombre de modèles locaux, alors que la détermination du nombre de neurones cachés pour une architecture MLP reste une tâche difficile.

Mots-clés : Identification, Systèmes dynamiques non-linéaires, multimodèles, réseaux de neurones MLP, modèles récurrents, classification floue

---

Identification of non linear dynamical system by neural networks and multiple models

Abstract

This work deals with non linear dynamical system identification. A multiple model architecture which overcomes certain insufficiencies of MLP neural networks is studied. Multiple model approach consists of modeling complex systems by mean of a set of simple local models whose validity are limited in well defined zones. Instead of using conventional affine models, a more general polynomial structure is proposed in this study, enabling to better apprehend local non linearities, reducing thus the number of local models. Models parameters of such a structure are estimated by linear optimization, which reduces computation time with respect to parameter estimation of a neural network architecture. The implementation of recurrent multiple models, with a more convenient learning algorithm than the back propagation through time, used in recurrent MLP models, is also studied. Such implementations facilitate representation of recurrent models like NARMAX and NOE. The determination of the number of local models in a multiple model architecture requires decomposition of system's feature space into several sub-systems in which local models are defined. Fuzzy partitioning methods (based of « fuzzy-c-means », « Gustafson and Kessel » and « subtractive clustering »algorithms) are presented. The use of such methods requires the implementation of a multiple model architecture where local models can have different structures : polynomial with different degrees, neural or polynomial and neural. A multiple model with a heterogeneous architecture satisfying these requirements is proposed and structural and parametrical identification algorithms are presented. A comparative study between multiple model and MLP architectures is done. The main advantage of the multiple model architecture is the parameter estimation simplicity. In addition, the use of fuzzy partitioning methods in multiple model architecture enables to find easily the number of local models while the determination of hidden neurons in an MLP architecture remains a hard task.

Key-words : Identification, non linear dynamical systems, multiple models, MLP neural networks, recurrents models, fuzzy classification