THÈSE de DOCTORAT de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité :

MICROÉLECTRONIQUE ET MICROINFORMATIQUE

présentée

par **Dominique URBANI**

pour obtenir le titre de **DOCTEUR DE L'UNIVERSITÉ PARIS 6**

Sujet de la thèse :

Méthodes statistiques de sélection d'architectures neuronales : application à la conception de modèles de processus dynamiques

Soutenue le 16 Novembre 1995

devant le jury composé de :

Mme	D. FOURNIER	
Mlle	S. MARCOS	Rapporteur
Mme	S. THIRIA	Rapporteur
M.	M. WEINFELD	
M.	L. PERSONNAZ	
M.	G. DREYFUS	

TABLE DES MATIÈRES

Introduction

Au cours des dernières années, l'une des évolutions les plus marquantes des réseaux de neurones formels a été, pour les ingénieurs, l'abandon de la métaphore biologique au profit de fondements théoriques solides dans le domaine des statistiques : on sait à présent que la propriété fondamentale des réseaux de neurones est l'approximation universelle [Hornik 89] parcimonieuse [Hornik 94]. De plus, le développement d'algorithmes performants pour l'apprentissage de ces réseaux, bouclés ou non bouclés, leur a ouvert de nouvelles perspectives d'utilisation. En particulier, les réseaux de neurones se sont avérés particulièrement adaptés, dans le domaine de l'Automatique, comme éléments de systèmes de commande de processus dynamiques non linéaires. Des travaux récents [Nerrand 92], [Rivals 95] ont permis de replacer la mise en œuvre des réseaux de neurones comme modèles de processus ou correcteurs de systèmes dans le cadre plus général de l'Automatique classique.

Cependant, peu de travaux abordent le problème important de la sélection de modèles : dans la pratique, les variables qui ont une action importante sur le processus à modéliser - qu'il soit statique ou dynamique - sont souvent inconnues, et il est nécessaire de les déterminer, puisqu'elles constituent les entrées du modèle neuronal. De plus, la qualité des résultats fournis par un modèle dépend en grande partie de la parcimonie de ce dernier : la sélection de la structure d'un réseau (nombre de neurones, donc nombre de paramètres ajustables) est donc de la plus grande importance.

D'autre part, l'intérêt croissant pour les réseaux de neurones formels a conduit à la réalisation de circuits dédiés aux applications neuronales (coprocesseurs spécifiques ou circuits intégrés). La plupart de ces réalisations sont tournées vers la recherche, leurs raisons d'être étant, d'une part, d'étudier la faisabilité et les performances de nouvelles architectures ou technologies, et, d'autre part, de satisfaire les besoins de simulations rapides. Néanmoins, on observe un nombre croissant d'applications des réseaux de neurones à des problèmes dépassant le cadre de la recherche, et, à moyen terme, on peut envisager l'utilisation de circuits "neuronaux" spécifiques pour la résolution de problèmes industriels présentant des contraintes de

vitesse importantes. Naturellement, la réalisation d'un réseau neuronal spécifique est d'autant plus simple que la taille du réseau est plus petite.

Ainsi, la qualité des résultats et la facilité d'implantation matérielle orientent les études dans la même direction : l'optimisation des réseaux, en termes de nombre d'entrées comme de nombre de neurones, pour obtenir des modèles aussi parcimonieux que possible.

C'est donc dans cette double optique que nous nous sommes intéressés au problème de la sélection de modèle. Dans ce travail, nous avons étudié les méthodes de sélection de modèles neuronaux dans le cadre de la modélisation de processus dynamiques non linéaire. Néanmoins, la plupart des méthodes et des résultats théoriques sur lesquels nous nous sommes appuyés peuvent, sans grandes difficultés, être transposés à des problèmes voisins, dans lesquels les réseaux de neurones ont prouvé leur efficacité (sélection des régresseurs de modèles de systèmes statiques non linéaires, ou détermination des caractéristiques pertinentes pour un problème de classification).

Diverses méthodes heuristiques ont été proposées dans le passé ; la plupart d'entre elles consistent à éliminer des paramètres du modèle, soit en les comparant à un seuil après apprentissage, et en supprimant ceux qui sont inférieurs à ce seuil, soit en incluant, dans la fonction de coût, un terme qui pénalise les paramètres dont les valeurs sont grandes, de sorte que les paramètres peu importants restent très voisins de zéro. L'inconvénient majeur de ces méthodes est que, si elles permettent d'optimiser le nombre de paramètres, le nombre d'entrées et le nombre de neurones du réseau restent généralement les mêmes, et l'implantation matérielle du réseau n'est pas simplifiée.

Dans ce travail, nous avons préféré utiliser des méthodes de tests statistiques, qui reposent sur des bases théoriques solides, et dont les performances en linéaire sont bien établies. Dans cette optique, les deux objectifs de notre travail ont été :

- de replacer le problème de la détermination de la structure de modèles neuronaux de processus non linéaires dynamiques dans le cadre plus général de la sélection de modèles à l'aide de méthodes statistiques;
- de proposer une procédure de sélection de modèles neuronaux qui s'inscrive dans le cadre théorique ainsi défini, tout en tenant compte des

difficultés rencontrées lors de la mise en œuvre sur des cas pratiques. De plus, la sélection concerne les entrées (régresseurs) du modèle, ainsi que le nombre de ses neurones, et non plus seulement ses paramètres.

Les trois premiers chapitres de ce mémoire répondent à la première de ces préoccupations.

Dans le chapitre I, nous rappelons quelques concepts fondamentaux pour la modélisation de processus. Nous montrons le lien qui existe entre, d'une part, les hypothèses *a priori* formulées sur le processus, qui conduisent à la définition d'un *modèle hypothèse* et de sa *forme prédicteur théorique* associée, et, d'autre part, le *système d'apprentissage* qu'il est nécessaire de mettre en œuvre pour l'identification d'un tel système. En particulier, nous mettons en évidence l'influence de la modélisation des perturbations dans le modèle hypothèse sur le choix du prédicteur du système d'apprentissage.
Dans le chapitre II, nous présentons les différentes méthodes pour l'identification des modèles linéaires ou non linéaires que nous utiliserons.
Le chapitre III est consacré à la présentation du problème de la sélection de modèles ; nous montrons le lien entre la sélection de modèles, au sens classique, et la détermination de la structure des réseaux de neurones. Dans le cas de modèles hypothèses NARMAX, l'approche EP (erreur de prédiction), qui permet de construire des estimateurs des paramètres du modèle qui sont asymptotiquement équivalents aux estimateurs du maximum de vraisemblance, est présentée. Différentes méthodes de sélection de modèles sont alors introduites, qui s'appliquent parfaitement à la sélection de modèles NARMAX, et des méthodes heuristiques permettant la réduction du nombre d'apprentissages nécessaires pour sélectionner un modèle sont présentées.

Une procédure originale de sélection de modèles NARX est présentée, puis mise en œuvre dans les chapitres IV et V. Cette procédure peut être partiellement utilisée pour sélectionner un modèle NARMAX.

Dans le chapitre IV, la procédure de sélection de modèles NARX, qui comporte trois phases, est exposée. Dans la première phase, le

fonctionnement du processus est étudié dans des domaines de fonctionnement restreints, où le processus peut être localement approché par des modèles linéaires ou polynomiaux. La sélection des régresseurs de ces modèles locaux se fait à l'aide d'une méthode heuristique, qui utilise la linéarité par rapport aux paramètres du modèle. Dans la deuxième phase, les entrées sélectionnées lors de la première phase sont utilisées pour construire un modèle neuronal global du processus. Une sélection des entrées de ce modèle non linéaire est effectuée à partir d'un ensemble d'apprentissage correspondant à un fonctionnement du processus sur l'ensemble du domaine étudié. L'objectif de la dernière phase est la détermination du nombre de neurones optimal du réseau.

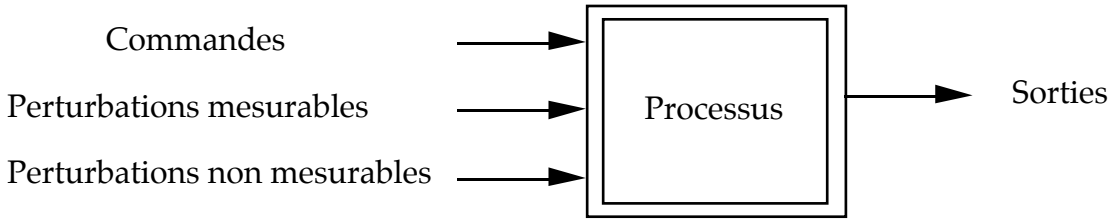
Cette procédure est mise en œuvre dans le chapitre V, sur des processus simulés. Ceci nous permet de mettre en évidence des difficultés rencontrées lors de l'apprentissage, qui sont, pour l'instant, incontournables, et de mettre en place des règles pragmatiques permettant d'obtenir des résultats aussi satisfaisants que possible dans l'état actuel des algorithmes d'estimation des poids.

Chapitre I. La modélisation de processus dynamiques

I.1. Processus et modèles

Un processus est un objet soumis à des actions externes, à l'intérieur duquel des grandeurs interagissent, et sur lequel on peut faire des mesures. Il existe de nombreux types de processus, de natures très différentes : artificiels, naturels (écologiques, biologiques, …), financiers ou sociaux. Les processus que nous considérons dans ce travail sont des objets sur lesquels on peut agir par des actionneurs dans un but déterminé de pilotage de systèmes divers, de production ou de transformation de matière, etc.

Les grandeurs d'intérêt mesurées (variable pilotée, débit et qualité d'un produit, …) sont appelées les *sorties* du processus. Les variables externes qui lui sont imposées par un opérateur (par exemple un régulateur ou un opérateur humain) sont appelées *commandes*; les autres variables externes qui agissent sur les sorties sont des *perturbations*, et l'on distinguera les *perturbations mesurables* des *perturbations non mesurables*. Les commandes et les perturbations mesurables sont appelées les *entrées* du processus.



Le but de toute modélisation de processus est de construire un *modèle,* c'est-à-dire une représentation mathématique de son fonctionnement.

Les processus dynamiques, auxquels nous nous intéressons dans ce travail, sont au cœur de nombreux secteurs d'activité (industrie, économie, recherche, …), et il est souvent nécessaire d'en construire un modèle, par exemple pour comprendre les phénomènes physiques mis en jeu, ou afin de concevoir un système automatique pour leur commande.

Nous distinguerons deux classes de modèles :

Les *modèles de simulation* ou *simulateurs* : un simulateur est un système qui possède un comportement dynamique analogue à celui du processus et qui est destiné à fonctionner indépendamment de celui-ci. Les simulateurs sont utiles dans de nombreuses applications :

- pour valider des hypothèses sur le processus que l'on étudie, pour extrapoler son comportement dans des domaines de fonctionnement où l'on ne dispose pas de résultats d'expériences;
- pour concevoir un système nouveau et appréhender à l'avance son comportement ou ses caractéristiques (c'est notamment le cas en micro-électronique, mais également en construction automobile, en aéronautique, …); on parle parfois de "modèle de conception";
- pour tester de nouveaux dispositifs de commande ou de régulation qu'il serait trop coûteux, ou dangereux, de tester sur le processus lui-même (par exemple si le processus est une partie d'une centrale nucléaire, un avion, …), ou comme outil de formation de personnel lorsque la manipulation du processus par des personnes inexpérimentées est irréalisable (pilote de chasse, personnel de surveillance d'une raffinerie de pétrole, …). Ces simulateurs devront être aussi "fidèles" que possible au processus, quel que soit le prix à payer en termes de complexité du modèle et de temps de calcul;
- pour la synthèse du correcteur d'un système de commande du processus.

Les *modèles de prédiction* ou *prédicteurs* : un prédicteur fonctionne en parallèle avec le processus modélisé, et il prédit la valeur de sortie du processus à l'instant t+d, à partir des valeurs des entrées et des sorties du processus disponibles à l'instant t.

Dans le domaine de la conception de systèmes de commande et de régulation de processus, les modèles de prédiction sont utilisés aussi bien dans les phases de conception d'un modèle du processus et d'apprentissage d'un correcteur que pendant la phase d'utilisation. De tels systèmes tiennent une place très importante dans l'industrie.

La distinction entre "modèles prédictifs" et "simulateurs" est essentiellement liée à l'utilisation que l'on fait du modèle, et un même modèle peut, dans certains cas, être utilisé soit comme prédicteur, soit comme simulateur.

Lors de la modélisation d'un processus, deux démarches sont envisageables :

La première consiste à construire un *modèle de connaissance*. La conception des modèles de connaissance découle d'une analyse physique des phénomènes mis en jeu dans le processus; lorsque cela est nécessaire, on décompose le processus étudié en éléments plus simples, pour lesquels on dispose déjà d'un modèle de connaissance éprouvé. Des données expérimentales sont ensuite utilisées, d'abord pour estimer numériquement les valeurs des paramètres du modèle, ensuite pour valider le modèle obtenu.

En particulier, la recherche scientifique a pour objet essentiel la construction de modèles de ce type, qui permettent non seulement de comprendre, mais aussi d'extrapoler le comportement d'un processus (réaction chimique, comportement d'un avion dans l'atmosphère, impact écologique d'un insecticide nouveau, …).

La seconde démarche est de construire un *modèle de type "boîte noire"*. Plus précisement, on cherche une expression mathématique qui traduise de manière la plus fidèle possible le comportement "entrée-sortie" du processus dans un domaine de fonctionnement défini par l'utilisation ultérieure. Les paramètres n'ont généralement pas de signification physique. L'estimation numérique de ces paramètres repose essentiellement sur un ensemble d'observations expérimentales dont on dispose sur le processus; dans le domaine des réseaux de neurones, cet ensemble d'observations est appelé *ensemble d'apprentissage*.

Les modèles "boîte noire" sont en général économiques en temps de calcul. Leur validité est limitée à un domaine de fonctionnement déterminé par l'ensemble d'apprentissage, tandis que celle des modèles de connaissance est déterminée par l'exactitude des hypothèses et la pertinence des approximations faites lors de l'analyse physique des phénomènes et de leur mise en équation.

Dans le cadre de la conception de modèles "boîte noire" de processus non linéaires, les modèles neuronaux sont d'excellents candidats qui permettent généralement d'approcher le comportement dynamique du processus de façon satisfaisante : en effet, les réseaux de neurones sont des approximateurs universels de fonctions non linéaires qui peuvent être employés aussi bien dans un cadre déterministe que dans un cadre probabiliste.

Dans ce travail, nous nous intéressons à **l'élaboration de modèles du type "boîte noire" de processus dynamiques non linéaires**. Nous ne considérerons que le cas de processus stationnaires, c'est-à-dire tels que les lois qui régissent leur comportement n'évoluent pas au cours du temps. Les modèles que nous considérons sont des modèles à temps discret.

Les modèles "boîtes noires" sont bien adaptés à la conception de modèles pour la commande de processus non linéaires, où l'on a souvent besoin de systèmes de structure relativement simple, afin de pouvoir effectuer de nombreux calculs, et ajuster si nécessaire les paramètres du modèle à des ensembles de données expérimentales nouvelles. Les modèles de connaissance, bien que fondés sur une analyse approfondie du processus et ayant des domaines de validité généralement moins limités, sont souvent trop complexes pour effectuer des calculs de façon rapide. De plus, leur conception est fortement liée au processus particulier que l'on cherche à modéliser, et aux connaissances dont on dispose sur la physique de celui-ci.

I.2. Modèle hypothèse et forme prédicteur

I.2.1 Modèle hypothèse

La conception d'un modèle prédictif de comportement repose sur l'hypothèse selon laquelle, pour le processus que l'on considère, il existe une description analytique de la relation entre les entrées et les sorties du processus. Cette description est généralement inconnue, et on l'exprime de façon formelle. Cette représentation formelle, appelée *modèle hypothèse*, prend en considération les connaissances *a priori* et les hypothèses concernant le comportement du processus. Elle constitue la base de départ de toute procédure de modélisation.

L'élaboration d'un modèle hypothèse consiste à effectuer des hypothèses concernant la nature et les caractéristiques du processus : caractère statique ou dynamique du processus; présence de perturbations (leur nature, leur mode d'action); caractère linéaire ou non linéaire du processus. Il faut en général fixer ou estimer un certain nombre de caractéristiques numériques du modèle.

La forme la plus générale d'un modèle hypothèse entrée-sortie non linéaire, déterministe, stationnaire, sans retard, à temps discret, est donnée par l'expression :

$$y_p(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u)) \qquad (I.1)$$

où $y_p(t)$ et $u(t)$ sont les sorties et les entrées (la commande et les perturbations mesurées) du processus à l'instant t.

Pour un modèle dynamique, il faut fixer ou estimer l'ordre n_y du modèle, ainsi que la valeur de la "mémoire" n_u sur la commande. Pour un modèle statique, la fonction $\varphi(.)$ ne dépend que de la variable de commande u aux instants t–1, t–2, …

Si l'on dispose de connaissances particulières, ou si l'on fait des hypothèses sur le comportement du processus, elles doivent être exprimées dans la formulation du modèle hypothèse : par exemple, si le comportement du processus est linéaire, la fonction $\varphi(.)$ est alors une somme pondérée des arguments $y_p(t–1), \ldots, y_p(t–n_y)$, $u(t–1), \ldots, u(t–n_u)$.

Supposons que des perturbations non mesurées agissent sur le processus. Ces perturbations peuvent être de deux types : les perturbations déterministes, et les perturbations de type "bruit".

Les *perturbations déterministes* peuvent être modélisées par une entrée non commandable (sinusoïde, constante, etc.) qui survient ou se modifie à des instants aléatoires. Par exemple, l'encrassement d'un appareil au cours de son utilisation peut se modéliser par une rampe. Dans le cadre de la conception d'un

système de commande, la présence de telles perturbations conduit à concevoir un système adaptatif.

Les *perturbations de type "bruit"* sont des perturbations qui peuvent être modélisées par une séquence de variables aléatoires. Un *bruit de mesure* est souvent modélisé par un bruit additif sur la sortie du processus; un *bruit d'état* est un bruit additif sur les variables d'état (variables bouclées) du système. Nous montrerons par la suite que le choix du mode d'action du bruit, qui entre dans l'élaboration d'un modèle hypothèse, détermine le choix de l'algorithme d'estimation des paramètres du modèle.

Lorsque les connaissances *a priori* sont insuffisantes pour choisir certaines caractéristiques du modèle hypothèse, il faut alors faire des hypothèses qui pourront être remises en cause au cours de la procédure de modélisation. Par exemple, on choisit un modèle hypothèse linéaire, on modélise les perturbations par un bruit d'état, on fixe les valeurs de n_y, n_u souvent de manière arbitraire.

Dans ce travail, nous nous sommes intéressés plus particulièrement aux modèles entrée-sortie. En effet, Leontaritis et Billings ont montré qu'il était possible de représenter une large classe de systèmes non linéaires bruités à temps discret par le modèle NARMAX (Non linéaire Auto-Régressif à Moyenne Ajustée avec entrée eXogène, [Leontaritis 85]). Son expression la plus générale est :

$$y_p(t) = \varphi(y_p(t-1), \ldots, y_p(t-n_y), u(t-1), \ldots, u(t-n_u), w(t-1), \ldots, w(t-n_w)) + w(t) \qquad (I.2)$$

où n_w est la "mémoire" sur le bruit w.

Les différentes variables intervenant dans la forme NARMAX sont :

- la sortie mesurée y_p du processus, qui peut être vectorielle;
- les perturbations non mesurables, modélisées à partir d'une séquence de variables aléatoires indépendantes d'espérance mathématique nulle {w(t)} (par abus de langage, on dira que w(t) est un bruit pseudo-blanc);
- le vecteur des entrées u : ses composantes sont les commandes, grandeurs sur lesquelles on peut agir pour influer sur le comportement du processus, et éventuellement des perturbations mesurables;
- les variables d'état du modèle qui sont, dans la représentation NARMAX, des valeurs passées de la sortie.

Dans la suite de ce mémoire, nous supposerons que les variables y_p, u et w sont scalaires, mais la généralisation au cas multivariable s'effectue sans difficulté majeure.

I.2.2 Forme prédicteur théorique et système d'apprentissage

Une fois le modèle hypothèse choisi, il faut déterminer ou estimer, à partir de séquences d'entrées {u(t)} et de sortie $\{y_p(t)\}$ du processus, l'ensemble de ses caractéristiques inconnues. Pour cela, on définit, d'une part, une fonction de coût théorique, et, d'autre part, une forme prédicteur théorique associée au modèle hypothèse. La *fonction de coût théorique* est une mesure de l'erreur de prédiction faite par un système de prédiction de la sortie du processus. On choisit généralement comme fonction de coût théorique la variance de l'erreur de prédiction. La *forme prédicteur théorique* est un système qui permet de calculer, à l'instant t, une prédiction y(t+d) de la sortie $y_p(t+d)$ du processus telle que, si l'on suppose le processus parfaitement décrit par le modèle hypothèse, la fonction de coût théorique soit minimale. Comme le modèle hypothèse, la forme prédicteur théorique est un système formel, qui s'exprime à l'aide des mêmes caractéristiques que le modèle hypothèse (c'est-à-dire $\varphi(.)$, n_y, n_u et n_w pour un modèle NARMAX).

Une fois le modèle hypothèse postulé, la démarche conduisant à la forme prédicteur théorique est la suivante :

– on suppose que le modèle hypothèse est une description parfaite du processus;

– on définit la sortie du processus comme une variable aléatoire $Y_p(t)$, dont une réalisation est notée $y_p(t)$ (de façon générale, nous noterons, dans ce mémoire, les variables aléatoires par des majuscules, et les réalisations de ces variables aléatoires par les minuscules correspondants). On peut alors exprimer la sortie du processus à l'instant t+d sous la forme :

$$Y_p(t+d) = E[Y_p(t+d)| t] + \nu(t+d)$$

où :

. $E[Y_p(t+d)| t]$ est l'espérance mathématique conditionnelle de $Y_p(t+d)$, lorsque l'on dispose de toutes les informations disponibles à l'instant t.

. $\nu(t+d)$ est la partie non prédictible de $Y_p(t+d)$ à l'instant t.

En supposant que le modèle hypothèse est exact, la meilleure prédiction que l'on puisse faire de $Y_p(t+d)$ à l'instant t est $E[Y_p(t+d)| t]$. La sortie de la forme prédicteur théorique, que l'on exprime à l'aide de n_y, n_u, n_w, et $\varphi(.)$, est une expression analytique égale à chaque instant à $E[Y_p(t+d)| t]$.

Lorsque la structure du modèle et de sa forme prédicteur théorique (entrées et ordre du modèle, nombre de neurones et architecture d'un modèle neuronal, etc.) sont définies, il faut déterminer la fonction $\varphi(.)$, ou en trouver la meilleure approximation possible. Pour cela, on met en œuvre un *système d'apprentissage*,

constitué d'un *prédicteur* et d'un *algorithme d'apprentissage*. Ce prédicteur est un système dont la structure est identique à celle de la forme prédicteur théorique, et qui réalise une fonction paramétrée $\phi(.;\theta)$. Les arguments de $\varphi(.)$ qui sont des variables aléatoires sont remplacés dans l'expression de $\phi(.;\theta)$ par leurs réalisations lorsque ces réalisations sont mesurables, et par des estimations lorsqu'elles ne sont pas mesurables. On définit alors une *fonction de coût empirique* à partir des écarts entre les sorties mesurées du processus et les valeurs calculées par le prédicteur, qui est une estimation de la fonction de coût théorique. A l'aide de l'algorithme d'apprentissage, on calcule la valeur de θ qui minimise cette fonction de coût empirique.

Si la structure du modèle n'est pas parfaitement définie, on considère alors un ensemble de modèles hypothèses, qui sont des cas particuliers du modèle hypothèse dont la structure est fixée, et l'on met en œuvre, pour chacun d'entre eux, un système d'apprentissage. Une *procédure de sélection* est alors utilisée pour choisir le "meilleur" de ces modèles, au sens d'un critère que l'on doit définir.

Nous reviendrons sur la mise en œuvre des systèmes d'apprentissage et la sélection de modèles dans les chapitres II et III. Nous allons auparavent montrer, sur quelques exemples, comment déterminer la forme prédicteur théorique associée à un modèle hypothèse particulier. Notons que l'appellation de "forme prédicteur" (ou "prédicteur") n'implique pas que le modèle sera ensuite utilisé pour faire de la prédiction, mais que, la procédure d'identification reposant sur la minimisation d'une fonction de coût construite à partir de l'erreur de prédiction, la mise en œuvre du système d'apprentissage nécessite l'utilisation d'un système de type "prédicteur". Après la procédure de modélisation, le modèle obtenu pourra être utilisé indifféremment pour construire un modèle de prédiction ou un modèle de simulation.

I.2.3 Forme prédicteur théorique associée à un modèle hypothèse

Nous allons montrer, sur quelques exemples, le lien entre le modèle hypothèse et la forme prédicteur qui lui est associée. Nous ne considérons ici que des prédicteurs à 1 pas (d=1). Dans le but d'illustrer l'influence du mode d'action du bruit sur le choix de la forme prédicteur, nous présentons les formes prédicteurs théoriques associées à un modèle hypothèse déterministe, à un modèle NARMAX, et à un modèle NBSX (Non linéaire à Bruit additif sur la Sortie et entrée eXogène).

I.2.3.1. Le modèle hypothèse est déterministe

Le modèle hypothèse est de la forme :

$$y_p(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u)) \qquad (I.1)$$

Le prédicteur calcule, à chaque instant t–1, la prédiction y(t) de la sortie $y_p(t)$ du processus. Le modèle hypothèse étant déterministe, la connaissance de φ(.) et des valeurs passées de l'entrée et de la sortie permet de calculer exactement la valeur de $y_p(t+1)$. La forme prédicteur théorique doit donc effectivement calculer à chaque instant $y(t) = y_p(t)$. Le prédicteur suivant :

$$y(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots., u(t–n_u))$$

est tel que la prédiction y(t) calculée à chaque instant est bien égale à la sortie $y_p(t)$ du processus, et l'erreur de prédiction est nulle. On a ainsi défini la forme prédicteur théorique associé au modèle déterministe (I.1). On remarque que cette expression correspond à un prédicteur non bouclé. Cependant, ce prédicteur est tel que, pour tout t, $y(t) = y_p(t)$; on peut aussi le mettre sous la forme :

$$y(t) = \varphi(y(t–1), \ldots, y(t–n_y), u(t–1), \ldots., u(t–n_u))$$

Ce dernier prédicteur (qui est bouclé) réalise alors également une prédiction parfaite s'il est correctement initialisé. Il existe donc, dans le cas d'un modèle déterministe, plusieurs représentations de la forme prédicteur théorique; par conséquent plusieurs structures de prédicteur peuvent être utilisées pour construire le prédicteur du système d'apprentissage. Le choix de l'une ou l'autre de ces représentations est dicté, d'une part, par la complexité du système d'apprentissage à mettre en œuvre (l'apprentissage d'un prédicteur bouclé nécessite des algorithmes d'apprentissage plus complexes que celui d'un prédicteur non bouclé), d'autre part, par l'utilisation qui sera faite du modèle obtenu à la fin de la procédure de modélisation (si le modèle doit être utilisé bouclé, il peut être préférable de faire son apprentissage avec un prédicteur bouclé).

I.2.3.2. Le modèle hypothèse est NARMAX.

Le modèle hypothèse est de la forme (I.2) :

$$y_p(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u), w(t–1), \ldots, w(t–n_w)) + w(t)$$

La séquence {w(t)}, réalisation de la séquence {W(t)} de variables aléatoires indépendantes d'espérance mathématique nulle et de variance σ_W^2 (bruit pseudo-blanc), est, par définition, imprédictible. La forme prédicteur théorique du modèle NARMAX (I.2) est donc le prédicteur tel que $y_p(t) – y(t) = w(t)$, soit :

$$y(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u), w(t–1), \ldots, w(t–n_w))$$

Cependant, un tel prédicteur est irréalisable, puisque les valeurs [w(t–1), .., $w(t–n_w)$] ne sont pas mesurables. Il faut donc les estimer, et pour cela, on utilise les valeurs de l'erreur de prédiction [e(t–1), e(t–2), …, $e(t–n_w)$] comme estimations des valeurs passées du bruit. L'expression du prédicteur devient alors :

$$y(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), u(t–n_u), e(t–1), \ldots, e(t–n_w))$$

Ce prédicteur est la forme prédicteur théorique associée au modèle NARMAX que l'on considère. En effet, si l'on suppose que les erreurs passées estiment parfaitement les valeurs passées du bruit, c'est-à-dire que l'on a, à l'instant t–1, la relation [e(t–1), e(t–2), …, $e(t–n_w)$] = [w(t–1), w(t–2), …, $w(t–n_w)$], l'erreur de prédiction e(t) à l'instant t est bien égale à w(t), qui est la meilleure prédiction que l'on puisse faire. De plus, on retrouve, à l'instant t, [e(t), e(t–1), …, $e(t–n_w+1)$] = [w(t), w(t–1), …, $w(t–n_w+1)$].

Cas particulier NARX

Le modèle hypothèse et la forme prédicteur théorique s'écrivent respectivement :

$$y_p(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u)) + w(t)$$

$$y(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u))$$

Les modèles NARX conduisent donc à des prédicteurs non bouclés.

I.2.3.3. Le modèle hypothèse est NBSX

Le modèle hypothèse est :

$$x_p(t) = \varphi(x_p(t–1), \ldots, x_p(t–n_y), u(t–1), \ldots, u(t–n_u)) \qquad \text{(I.3)}$$
$$y_p(t) = x_p(t) + w(t)$$

Le prédicteur associé s'écrit simplement :

$$y(t) = \varphi(y(t–1), \ldots, y(t–n_y), u(t–1), \ldots, u(t–n_u))$$

En effet, si à l'instant t–1, la relation [y(t–1), …, $y(t–n_y)$] = [$x_p(t–1)$, …, $x_p(t–n_y)$] est vérifiée, la prédiction à l'instant t est alors égale à $x_p(t)$, et $e(t) = y_p(t) – y(t) = w(t)$. On retrouve à l'instant t la relation [y(t), …, $y(t–n_y+1)$] = [$x_p(t)$, …, $x_p(t–n_y+1)$]. A chaque instant, l'erreur de prédiction est donc bien égale à la valeur de la perturbation w(t).

On constate, en comparant les formes prédicteurs obtenues pour les modèles NARMAX, NARX et NBSX, l'influence de la modélisation des perturbations sur la détermination de la forme prédicteur. Si le modèle hypothèse est exact, la forme prédicteur théorique fournit une variance de l'erreur de prédiction minimale et égale à la variance du bruit, et la variance de l'erreur de prédiction

obtenue avec un prédicteur différent est supérieure à la variance du bruit [Ljung 87]. Ceci est illustré par Nerrand sur des exemples NARX et NBSX dans le cas ou les prédicteurs sont des prédicteurs neuronaux [Nerrand 92a].

I.3. Conception de modèles NARMAX

La procédure de modélisation peut donc se décomposer en quatre étapes :

– conception d'un ensemble de modèles hypothèses candidats;
– définition des formes prédicteurs théoriques associées aux modèles hypothèses;
– pour chaque modèle hypothèse candidat : définition et apprentissage du modèle prédictif, défini par la forme prédicteur théorique, à l'aide de séquences d'entrées-sorties du processus (séquences d'apprentissage);
– sélection du meilleur candidat.

Dans le cas de modèles NARMAX, nous venons de montrer que l'on peut facilement déduire la structure du prédicteur théorique de l'expression du modèle hypothèse. Cependant, les valeurs de n_y, n_u et n_w, ainsi que l'expression de $\varphi(.)$, sont généralement inconnues. Lors de l'identification du processus, on doit donc trouver de "bonnes" valeurs de ces caractéristiques, et déterminer, dans une famille de fonctions $\phi(.;\theta)$ que l'on se donne (un réseau de neurones par exemple), la fonction qui approche au mieux $\varphi(.)$.

Nous pouvons à présent définir précisément le cadre dans lequel se place notre travail :

– nous nous intéressons à la conception de modèles "boîte noire" de processus dynamiques non linéaires. Nos efforts porteront sur la conception du prédicteur et l'estimation de ses paramètres.

– les connaissances *a priori* nous conduisent à faire l'hypothèse que le processus peut être décrit par un modèle NARMAX. L'expression et les arguments de la fonction $\varphi(.)$ définissant ce modèle doivent être déterminées. Les processus que nous étudierons étant non linéaires, les modèles neuronaux sont des candidats naturels [Nerrand 92a];

– nous disposons de données expérimentales, et l'on supposera que l'on peut effectuer autant d'expériences que nécessaire pour mener à bien la modélisation. Cette condition n'est évidemment pas toujours réalisable dans la réalité industrielle.

Chapitre II : Estimation des paramètres d'un modèle

Dans ce chapitre, nous considérons un modèle hypothèse de type NARMAX. L'ensemble des arguments de $\varphi(.)$ est supposé connu, mais la fonction $\varphi(.)$ est inconnue. On dispose d'une famille F de fonctions paramétrées $\phi(.;\theta)$, possédant les mêmes arguments que la fonction $\varphi(.)$ du modèle hypothèse, et à laquelle correspond une famille M de modèles paramétrés. L'identification consiste à trouver le modèle $M(\theta^*)$ du processus tel que la fonction $\phi(.;\theta^*)$ approche au mieux $\varphi(.)$ dans le domaine de fonctionnement défini par l'ensemble d'apprentissage. On note P le modèle hypothétique décrivant parfaitement le processus, qui n'est pas nécessairement un élément de M.

Si l'on considère le vecteur $\mathbf{y}_p$ des données expérimentales comme la réalisation d'un vecteur aléatoire $\mathbf{Y}_p$, toute estimation $\hat{\theta}$ calculée à partir de $\mathbf{y}_p$ est une réalisation d'un estimateur $\Theta(\mathbf{Y}_p)$ de θ^*. On construit donc un estimateur $\Theta(\mathbf{Y}_p)$ de θ^*, puis l'on calcule une estimation $\hat{\theta}$ à l'aide de l'ensemble d'apprentissage.

II.1 Position du problème

Pour calculer cette estimation, nous disposons, d'une part, de l'ensemble d'apprentissage constitué des séquences de données $\{u(t)\}$ et $\{y_p(t)\}$, et, d'autre part, d'une famille paramétrée de modèles ayant une structure commune $M(\theta)$. Pour obtenir la meilleure approximation de $\varphi(.)$, il faut choisir un critère pour comparer les différents modèles. Nous devons ici distinguer deux notions différentes : le critère de performance et la fonction de coût, que nous avons mentionnée dans le chapitre précédent.

Le *critère de performance*, qui peut être quantitatif ou qualitatif, permet de juger de la qualité d'un modèle vis à vis d'un cahier des charges. L'expression du critère de performance n'intervient pas nécessairement de manière explicite dans la procédure d'identification : il permet de juger *a posteriori* la performance d'un modèle, sur un ensemble de données généralement différent de l'ensemble d'apprentissage.

La *fonction de coût* est une fonction scalaire, que l'on utilise pour l'estimation des paramètres. Elle dépend des paramètres θ du modèle et des données d'apprentissage, et elle est choisie de telle manière qu'une faible valeur corresponde à un "bon" modèle. On la note $J(\theta)$.

Généralement, $J(\theta)$ s'exprime à partir de la séquence des erreurs de prédiction $\{e(t;\theta)\}$, où $e(t;\theta) = y_p(t) - y(t;\theta)$. Il faut donc définir un estimateur du vecteur des

paramètres du modèle qui minimise la fonction de coût. Nous allons tout d'abord présenter l'estimateur des moindres-carrés ordinaires, utilisé pour les modèles linéaires par rapport aux paramètres.

II.1.1 L'estimateur des moindres-carrés ordinaires

Considérons le problème suivant : on cherche à effectuer la modélisation d'un processus, et l'on dispose d'un ensemble de données expérimentales constitué de deux séquences de taille N, la séquence des vecteurs d'entrées $\{\mathbf{x}(n)\}$ et la séquence des sorties mesurées du processus $\{y_p(n)\}$. Les vecteurs d'entrées $\mathbf{x}(n)$ sont des vecteurs certains; si le processus est dynamique, leurs composantes sont par exemple des valeurs présentes et passées de la commande u, et des valeurs passées des sorties; si le problème est statique, la valeur $y_p(n)$ est la sortie obtenue avec les entrées $\mathbf{x}(n)$. Les étapes précédentes de la modélisation nous ont conduit à considérer le modèle hypothèse linéaire par rapport aux paramètres θ défini par :

$$Y_p(n) = \sum_{i=1}^{M} \theta_i \, x_i(n) + W(n) = \mathbf{x}^T(n)\,\theta + W(n) \qquad \text{(II.1)}$$

Le vecteur $\mathbf{x}(n) = [x_1(n), \ldots, x_M(n)]^T$ est un vecteur certain, et $\{W(n)\}$ est une séquence pseudo-blanche (séquence de variables aléatoires indépendantes de moyenne nulle, de variance σ^2). La matrice $\mathbf{x}$ dont les colonnes sont les *vecteurs de régression* $\mathbf{x}_i = [x_i(1), \ldots, x_i(N)]^T$, (i=1, …, M), correspondant aux régresseurs $x_1(n), \ldots, x_M(n)$, est une matrice certaine, appelée *matrice de régression*.
L'équation linéaire ci-dessus peut alors s'écrire :

$$\mathbf{Y}_p = \mathbf{x}\,\theta + \mathbf{W} \qquad \text{(II.2)}$$

soit encore :

$$\begin{bmatrix} Y_p(1) \\ \vdots \\ Y_p(N) \end{bmatrix} = \begin{bmatrix} \mathbf{x}(1)^T \\ \vdots \\ \mathbf{x}(N)^T \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_M \end{bmatrix} + \begin{bmatrix} W(1) \\ \vdots \\ W(N) \end{bmatrix} = \begin{bmatrix} x_1(1) & \cdots & x_M(1) \\ \vdots & \ddots & \vdots \\ x_1(N) & \cdots & x_M(N) \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_M \end{bmatrix} + \begin{bmatrix} W(1) \\ \vdots \\ W(N) \end{bmatrix}$$

W(n) étant imprédictible, le meilleur modèle que l'on puisse construire est :

$$y(n) = \mathbf{x}^T(n)\theta = E[Y_p(n) | \mathbf{x}(n)] \qquad \text{(II.3)}$$

On recherche la valeur θ^* qui minimise le vecteur des erreurs $\mathbf{Y}_p - \mathbf{x}\,\theta$. La mesure que l'on choisit pour effectuer cette minimisation est proportionnelle au carré de sa norme euclidienne :

$$J(\theta) = \frac{1}{N} (\mathbf{Y}_p - \mathbf{x}\theta)^T (\mathbf{Y}_p - \mathbf{x}\theta) \qquad \text{(II.4)}$$

Soit $\mathbf{y}_p$ une réalisation de $\mathbf{Y}_p$; on appelle *estimation des moindres-carrés* de θ^* la valeur $\hat{\theta}$ de θ qui minimise $J(\theta)$. Si f(.) est la fonction qui exprime $\hat{\theta}$ en fonction du vecteur des observations $\mathbf{y}_p$, *l'estimateur des moindres-carrés ordinaires* est le vecteur aléatoire $\Theta = f(\mathbf{Y}_p)$, que l'on exprime aussi sous la forme suivante :

$$\Theta = \arg\min J(\theta) = \arg\min \left[\frac{1}{N}(\mathbf{Y}_p - \mathbf{x}\theta)^T(\mathbf{Y}_p - \mathbf{x}\theta)\right] \qquad \text{(II. 5)}$$

Un raisonnement simple permet de résoudre (II.5) : pour une réalisation $\mathbf{y}_p$ de Y_p, on cherche la valeur de θ vérifiant :

$$\mathbf{grad}_\theta J(\theta) = \mathbf{grad}_\theta \left[\frac{1}{N}(\mathbf{y}_p - \mathbf{x}\theta)^T(\mathbf{y}_p - \mathbf{x}\theta)\right] = 0 \qquad \text{(II.6)}$$

soit :

$$\frac{1}{N}(-2\mathbf{x}^T\mathbf{y}_p + 2\mathbf{x}^T\mathbf{x}\theta) = 0$$

On résout donc l'équation suivante, appelée "équation normale" :

$$[\mathbf{x}^T\mathbf{y}_p] = [\mathbf{x}^T\mathbf{x}]\,\theta \qquad \text{(II.7)}$$

Si la matrice $[\mathbf{x}^T\mathbf{x}]$ est inversible (ce qui est généralement le cas pour N>>M), (II.7) conduit à l'estimation suivante :

$$\hat{\theta} = [\mathbf{x}^T\mathbf{x}]^{-1}\,[\mathbf{x}^T\mathbf{y}_p] \qquad \text{(II.8)}$$

Si le modèle est exact (c'est-à-dire si $\mathbf{x}\,\theta = E[\mathbf{Y}p]$), l'estimateur des moindres-carrés (II.8) est non biaisé. Si le bruit W est gaussien, la matrice de covariance de Θ est de norme minimale.

Pour un modèle dynamique ARX, le vecteur des entrées $X(t) = [Y_p(t-1), \ldots, Y_p(t-n_y), u(t-1), \ldots, u(t-n_u)]$ n'est plus un vecteur certain, mais l'estimateur des moindres-carrés garde les mêmes propriétés [Goodwin 77].

Dans le cas de modèles hypothèses linéaires ARMAX, les propriétés de l'estimateur des moindres-carrés ne sont plus démontrées. On utilise alors des méthodes plus générales, qui s'appliquent aussi bien aux modèles linéaires ARMAX qu'aux modèles non linéaires NARMAX. Ces méthodes, que nous présentons dans le paragraphe suivant, conduisent, dans le cas de modèles hypothèses ARX, à la même estimation des paramètres que la méthode des moindres-carrés présentée ci-dessus, et donc à un estimateur non biaisé de matrice de covariance minimale.

II.1.2. Les Méthodes fondées sur l'Erreur de Prédiction (Méthodes EP)

La séquence des erreurs de prédiction {e(t;θ)} peut être interprétée comme un vecteur $\mathbf{e}(\theta) = [e(1;\theta), \ldots, e(N;\theta)]^T$ de dimension N. Toute norme de R^N est une mesure de ce vecteur, et peut être utilisée pour construire une fonction de coût. La fonction de coût que nous considérons dans ce travail est la norme quadratique de R^N suivante :

$$J(\theta) = \frac{1}{N} \sum_{t=1}^{N} e^2(t;\theta) \qquad \text{(II.9)}$$

L'estimation de θ* est définie comme la valeur $\hat{\theta}$ qui minimise la fonction de coût J(θ) sur le domaine D_θ des valeurs de θ [Ljung 74]. On peut exprimer l'estimateur EP sous la forme :

$$\Theta = \arg_{[\theta \in D_\theta]} \min \{ J(\theta) \} \qquad \text{(II.10)}$$

On se ramène à un problème d'optimisation, qui peut être résolu à l'aide d'un grand nombre d'algorithmes. On peut remarquer qu'aucune hypothèse sur le caractère linéaire ou non linéaire du modèle n'intervient dans la formulation de l'approche EP.

Les algorithmes d'estimation correspondant à la norme (II.9), que nous avons choisie, sont appelés *algorithmes de moindres-carrés*. La programmation de ces algorithmes d'estimation et l'analyse de leur comportement sont simples. Une estimation de la variance de l'erreur de prédiction d'un modèle dont les paramètres sont θ est fournie par J(θ).

Dans le cas de modèles linéaires par rapport aux paramètres, l'estimateur défini par (II.10) n'est autre que l'estimateur des moindres-carrés ordinaires que nous avons présenté dans le paragraphe précédent. Nous verrons au chapitre III que les méthodes fondées l'erreur de prédiction s'étendent à des modèles plus généraux.

II.1.3. Les méthodes de corrélation

Nous avons jusqu'ici présenté le problème de la modélisation sous la forme suivante : le modèle que l'on recherche est le modèle pour lequel les erreurs de prédiction sur un ensemble d'apprentissage sont les plus faibles. Il existe une autre formulation du problème qui, quoiqu'assez proche, débouche sur des méthodes d'estimation différentes.

Elle suppose que l'on ait préalablement défini un ensemble de régresseurs du modèle (les arguments de φ), et consiste à extraire toutes les informations contenues dans l'ensemble d'apprentissage, pouvant être expliquées à l'aide de ces

régresseurs. Une fois ces informations extraites, la séquence des erreurs de prédiction doit être décorrélée de tous les régresseurs considérés.

Considérons un modèle linéaire ARX dont les entrées sont $\{y_p(t-1), .., y_p(t-n_y), u(t-1), \ldots, u(t-n_u)\}$, et un ensemble d'apprentissage de dimension finie N. On cherche la valeur $\hat{\theta}$ de θ telle que :

$$\sum_{t=1}^{N} \mathbf{x}(t)\, e(t;\hat{\theta}) = \mathbf{0} \qquad \text{(II.11)}$$

Si cette condition est vérifiée, l'estimation de la corrélation de l'erreur avec chacun des régresseurs sera donc nulle, et l'on peut considérer que l'on a extrait toute la connaissance pouvant être expliquée à l'aide des régresseurs du modèle. Reprenons l'équation normale (II.7); on peut la mettre sous la forme :

$$\mathbf{x}^T(\mathbf{y}_p - \mathbf{x}\hat{\theta}) = 0$$

soit :

$$\sum_{i=1}^{N} \mathbf{x}(t)\, e(t;\hat{\theta}) = 0 \qquad \text{(II.12)}$$

On retrouve bien l'équation (II.11).

Lorsque le modèle est non linéaire par rapport aux paramètres, les méthodes de corrélation sont plus complexes à mettre en œuvre. En effet, l'erreur peut être non corrélée à une entrée, mais corrélée à un régresseur qui est une fonction non linéaire de cette entrée. La mise en œuvre des méthodes de corrélation nécessite la résolution de toutes les équations de la forme :

$$\sum_{i=1}^{N} f(\mathbf{x}(t))\, e(t) = 0 \qquad \text{(II.13)}$$

où f(.) est une transformation non linéaire quelconque des entrées. Il est donc nécessaire d'annuler la corrélation de l'erreur avec toutes les fonctions non linéaires des entrées, ce qui est impossible dans la pratique.

La résolution de toutes les équations de la forme (II.13) étant irréalisable, les méthodes de corrélation semblent mal adaptées à l'estimation de paramètres de modèles non linéaires.

Nous avons donc choisi les méthodes EP, plus faciles à mettre en œuvre avec des modèles non linéaires et dont les algorithmes classiques d'apprentissage des réseaux de neurones sont des exemples [Nerrand 92].

II.2. Estimation des paramètres d'un modèle

Rappelons que nous disposons, pour mener à bien la modélisation du processus, d'un ensemble de données {u(t), y(t); t=1…N}, l'ensemble d'apprentissage, d'un modèle hypothèse obtenu à partir des connaissances *a priori* disponibles sur le processus, et d'une forme prédicteur, associée à ce modèle hypothèse. Ce prédicteur est une structure paramétrée dont on cherche à estimer les paramètres θ à partir de l'ensemble d'apprentissage en utilisant un estimateur de la forme (II.10) :

$$\Theta = \arg_{[\theta \in D_\theta]} \min \{ J(\theta) \}$$

Pour calculer l'estimation $\hat{\theta}$, nous avons besoin d'un algorithme d'optimisation. Revenons sur le problème simple de l'estimation des paramètres d'un modèle linéaire par rapport aux paramètres à l'aide de la méthode des moindres-carrés. L'estimateur utilisé est :

$$\Theta = \arg_{[\theta \in D_\theta]} \min \{ \frac{1}{N} (\mathbf{Y}_p - \mathbf{x}\theta)^T (\mathbf{Y}_p - \mathbf{x}\theta) \} \qquad \text{(II.14)}$$

La sortie du modèle étant linéaire par rapport aux paramètres, nous obtenons une estimation de θ :

- soit par la résolution de l'équation normale $\mathbf{x}^T\mathbf{y}_p = \mathbf{x}^T\mathbf{x}\ \theta$, qui mène à l'estimation $\hat{\theta} = [\mathbf{x}^T\mathbf{x}]^{-1}[\mathbf{x}^T\mathbf{y}_p]$, à l'aide d'une méthode de décomposition de la matrice $[\mathbf{x}^T\mathbf{x}]$ (décomposition "LU", Cholesky, …) [Press 92],
- soit à l'aide d'une méthode fondée sur la décomposition orthogonale de la matrice de régression **x** (Householder, Gram-Schmidt, …) dont nous décrivons le principe dans le paragraphe suivant [Press 92].

Si le modèle hypothèse est non linéaire, ces méthodes ne peuvent plus être utilisées; on met alors en œuvre des méthodes itératives de type "gradient".

De façon générale, le choix d'un estimateur étant effectué, il reste encore à choisir l'algorithme d'optimisation qui permet le calcul de l'estimation.

II.3. Algorithmes d'optimisation

II.3.1 Les méthodes linéaires de résolution

Il existe plusieurs familles de méthodes permettant de calculer l'estimation des moindres-carrés ordinaires dans le cas d'un modèle linéaire. Nous présentons ici deux d'entre elles :

– l'estimation des moindres-carrés est obtenue par la résolution de l'équation normale :

$$\mathbf{x}^T\mathbf{x}\,\theta = \mathbf{x}^T\,\mathbf{y}_p$$

par élimination de Gauss, ou en formant la décomposition de Cholesky de la matrice $[\mathbf{x}^T\mathbf{x}]$. La matrice $[\mathbf{x}^T\mathbf{x}]$ étant symétrique définie positive, la méthode de Cholesky consiste à la décomposer sous la forme $[\mathbf{x}^T\mathbf{x}] = L^TL$, où L est une matrice triangulaire inférieure. L'équation normale mise sous cette forme se résout alors ligne à ligne.

– on forme une décomposition orthogonale de la matrice de régression **x**. En effet, considérons le modèle hypothèse suivant :

$$\mathbf{Y}_p = \mathbf{x}\,\theta + W = [\mathbf{x}_1, \ldots, \mathbf{x}_M]\,\theta + \mathbf{W} \qquad \text{(II.15)}$$

Si le modèle est exact, les vecteurs colonnes $\mathbf{x}_1, \ldots, \mathbf{x}_M$ de la matrice de régression **x** engendrent un sous-espace orthogonal au vecteur W. Il est possible de construire une base orthogonale $\mathbf{b}=[\mathbf{b}_1, \ldots, \mathbf{b}_M]$ engendrant le même sous-espace que la base des vecteurs de régression, vérifiant la relation $\mathbf{x}=\mathbf{bA}$, où **A** est une matrice triangulaire. Le vecteur $\mathbf{Y}_p$ peut s'exprimer comme la somme d'un vecteur $\delta = (\beta, 0)$ où $\beta = (\mathbf{b}^T\mathbf{b})^{-1}(\mathbf{b}^T\mathbf{Y}_p)$ est le vecteur des coefficients de projection de $\mathbf{Y}_p$ sur la base **b**, et d'un vecteur γ orthogonal à ce sous-espace. Multiplions l'équation (II.15) par $(\mathbf{b}^T\mathbf{b})^{-1}\mathbf{b}^T$, on obtient :

$$(\mathbf{b}^T\mathbf{b})^{-1}\,\mathbf{b}^T\mathbf{Y}_p = (\mathbf{b}^T\mathbf{b})^{-1}\,\mathbf{b}^T\mathbf{x}\,\theta + (\mathbf{b}^T\mathbf{b})^{-1}\,\mathbf{b}^T\mathbf{W} = (\mathbf{b}^T\mathbf{b})^{-1}\,\mathbf{b}^T\mathbf{x}\,\theta$$

soit :

$$\beta = \mathbf{A}\theta \qquad \text{(II.16)}$$

La matrice A étant triangulaire, on résout (II.16) ligne à ligne.

Les méthodes d'orthogonalisation sont numériquement plus lourdes que les méthodes de résolution de l'équation normale (elles nécessitent entre deux et quatre fois plus de calculs que la méthode de Cholesky). En contrepartie, elles sont dans certains cas plus robustes aux erreurs numériques que les méthodes de résolution de l'équation normale, en particulier lorsque la matrice de régression est mal conditionnée (cela survient notamment lorsque le nombre de régresseurs du modèle est exagéré).

Dans le cadre de la procédure de sélection de modèles que nous proposons dans le chapitre IV, pour laquelle on est amené à estimer les paramètres de modèles parfois sur-dimensionnés, nous utilisons une méthode fondée sur la méthode d'orthogonalisation de Gram-Schmidt.

II.3.2. Modèles non linéaires : les méthodes de gradient

Si le modèle prédictif est non linéaire par rapport aux paramètres, les méthodes de résolution précédentes ne sont plus utilisables. On a alors recours à des méthodes itératives de type "gradient" pour effectuer l'estimation des paramètres des modèles non linéaires. Ces méthodes d'optimisation sont coûteuses en temps de calcul, mais restent simples à mettre en œuvre, et s'appliquent à toutes les fonctions $\phi(.;\theta)$ dérivables par rapport à θ. Ceci est en particulier le cas lorsque $\phi(.;\theta)$ est réalisé par un réseau de neurones.

II.3.2.1. Principe

La solution de :

$$\hat{\theta} = \arg_{[\theta \in D_\theta]} \min \left[\frac{1}{N}(\mathbf{y}_p - \mathbf{y})^T(\mathbf{y}_p - \mathbf{y})\right] = \arg_{[\theta \in D_\theta]} \min \{J(\theta)\} \qquad \text{(II. 17)}$$

est un minimum de $J(\theta)$, et vérifie donc également :

$$\hat{\theta} = \arg_{[\theta \in D_\theta]} \left(\text{grad } J(\theta) = 0\right) \qquad \text{(II. 18)}$$

Le Hessien de $J(\theta)$ en $\hat{\theta}$ est défini positif.

Les méthodes résolution non linéaire consistent à rechercher, de façon itérative, une solution numérique de (II.18). Il faut noter que lorsque $J(\theta)$ possède plusieurs minima, rien ne garantit que le minimum obtenu soit un minimum global. On procède de la manière suivante :

A l'itération 0 :

θ est initialisé à une valeur quelconque θ^0, avec laquelle on calcule le gradient **grad** $J(\theta^0)$, et, éventuellement, la fonction de coût $J(\theta^0)$.

A l'itération k :

i. $\| \textbf{grad } J(\theta^{k-1}) \| \leq \varepsilon$

la procédure s'arrête, la solution retenue de (II.18) est θ^{k-1}.

ii. $\| \textbf{grad } J(\theta^{k-1}) \| > \varepsilon$

on calcule θ^k à partir de θ^{k-1}, $J(\theta^{k-1})$ et **grad** $J(\theta^{k-1})$:

$$\theta^k = \theta^{k-1} + \mu_k \mathbf{d}_k,$$

où :

μ_k est le **pas** du gradient (scalaire positif), dont la valeur peut être constante ou optimisée à chaque itération,

$\mathbf{d}_k$ est une **direction de descente**, c'est-à-dire un vecteur tel que :

$$[\mathbf{grad}\ J(\theta^{k-1})]^T\ \mathbf{d}_k < 0$$

Les méthodes de résolution non linéaire se différencient par le choix de la direction de descente $\mathbf{d}_k$ et du pas μ_k. Dans les méthodes de type "gradient", la direction de descente $\mathbf{d}_k$ s'exprime toujours à partir de **grad** $J(\theta)$. Dans le cas de modèles neuronaux, le calcul du gradient utilise l'algorithme de rétropropagation [Rumelhart 86]. Pour plus de détails sur les algorithmes d'apprentissage des réseaux de neurones bouclés ou non bouclés, on se référera aux travaux de Nerrand [Nerrand 92a, b].

II.3.2.2. La méthode du gradient simple

La méthode la plus simple à mettre en œuvre est la méthode du gradient simple. Le pas μ_k est une constante μ, et la direction de descente est simplement l'opposé de celle du gradient. A l'itération k, la modification de θ est :

$$\theta^k = \theta^{k-1} - \mu\ \mathbf{grad}\ J(\theta^{k-1}) \qquad \text{(II.19)}$$

Cette méthode est très utilisée. Elle a pour avantages une grande facilité de mise en œuvre et une grande robustesse. Le choix de μ n'est pas critique pour la convergence. La méthode est efficace loin d'un minimum, mais la vitesse de convergence diminue lorsque l'on s'approche du minimum (la modification de θ est proportionnelle à **grad** $J(\theta^{k-1})$, qui tend vers **0**).

II.3.2.3. La méthode de Newton

Algorithme :

$$\theta^k = \theta^{k-1} - [H(\theta^{k-1})]^{-1}\ \mathbf{grad}\ J(\theta^{k-1}) \qquad \text{(II.20)}$$

où $H(\theta)$ est la matrice des dérivées secondes de $J(\theta)$ par rapport à θ (Hessien). Le pas est constant (μ =1), et la direction de déplacement est :

$$\mathbf{d}_k = -\ [H(\theta^{k-1})]^{-1}\ \mathbf{grad}\ J(\theta^{k-1}) \qquad \text{(II.21)}$$

Si $J(\theta)$ est quadratique, l'algorithme converge en une itération. La méthode est donc efficace si θ est proche d'un minimum autour duquel $J(\theta)$ est presque quadratique. Pour que la méthode converge vers le minimum, la matrice $H(\theta)$ doit être définie positive.

La méthode de Newton est peu employée, car elle nécessite le calcul et l'inversion du Hessien à chaque itération, et la définie-positivité du Hessien doit

être satisfaite à chaque itération. On lui préfère les méthodes économiques dites de "Quasi-Newton".

II.3.2.4. Les méthodes Quasi-Newtoniennes

Dans les méthodes quasi-newtoniennes, l'inverse du Hessien $[\Delta J(\theta^{k-1})]^{-1}$ est approximé par une matrice M_k définie positive, modifiée à chaque itération. La suite des matrices $\{M_k\}$ est construite de manière à converger vers l'inverse du Hessien lorsque la fonction $J(\theta)$ est quadratique, approximation qui peut être légitimement faite lorsque l'on s'approche du minimum. La modification des paramètres à chaque itération est :

$$\theta^k = \theta^{k-1} - \mu_{k-1} M_{k-1} \mathbf{grad}\, J(\theta^{k-1}) \qquad \text{(II.22)}$$

où μ_{k-1} est le pas de déplacement qui minimise la fonction :

$$g(\mu) = J(\theta^{k-1} + \mu\, \mathbf{d}_{k-1}).$$

La direction de déplacement est $\mathbf{d}_k = - M_{k-1} \mathbf{grad}\, J(\theta^{k-1})$.

En pratique, on a intérêt à commencer par des itérations de l'algorithme du gradient simple, qui est efficace loin du minimum. M_0 est ensuite initialisée à la matrice identité, puis l'on commute sur une méthode quasi-newtonnienne. Parmi les méthodes quasi-newtoniennes proposées dans la littérature [Minoux 83], nous avons choisi d'utiliser la méthode BFGS, développée indépendamment par Broyden [Broyden 70], Fletcher [Fletcher 70], Goldfarb [Goldfrab 70] et Shanno [Shanno 69], dont la vitesse de convergence est beaucoup plus grande que celle de la méthode du gradient.

II.3.2.5. Optimisation du pas

La vitesse de convergence de ces méthodes peut être améliorée en asservissant le pas μ. Les méthodes les plus efficaces sont les méthodes de dichotomie (la méthode de Fibonacci, par exemple), mais elles nécessitent généralement beaucoup de calculs. Des méthodes plus économiques ont été proposées, en particulier la méthode de Nash [Nash 90], et la méthode de Wolfe et Powell [Wolfe 69], [Powell 76]. Ces méthodes unidimensionnelles permettent d'obtenir un pas convenable avec un nombre limité d'évaluations de la fonction de coût et du gradient de $J(\theta)$.

Chapitre III. La sélection de modèles

III.1. Introduction

Lorsqu'un modèle hypothèse du processus a été choisi, et que la forme prédicteur associée est déterminée, la modélisation consiste à choisir, au sein d'une famille de fonctions paramétrées (un réseau de neurones par exemple), une fonction $\phi(.;\theta)$ réalisant une bonne approximation de la fonction $\varphi(.)$ de la forme prédicteur, dans un domaine de fonctionnement borné défini par l'ensemble d'apprentissage. Lorsque, comme nous l'avons supposé dans le chapitre précédent, les caractéristiques du modèle (les entrées, c'est-à-dire les arguments de $\phi(.;\theta)$, et la famille de fonctions) ont été préalablement fixées, le choix de la fonction $\phi(.;\theta)$ se réduit à l'estimation des paramètres θ la définissant.

En réalité, les connaissances *a priori* sont souvent incomplètes et conduisent à définir un ensemble de modèles concurrents. Chacun de ces modèles correspond à des hypothèses particulières faites sur le processus, et la modélisation consiste alors à choisir, à partir de mesures faites sur le processus, un modèle appartenant à cet ensemble. On peut classer les différentes caractéristiques définissant le modèle, que l'on doit fixer ou estimer lors de la procédure de modélisation, en trois niveaux :

– le *type* du modèle, c'est-à-dire les caractéristiques très générales, qui dépendent du processus auquel on s'intéresse (modèle linéaire ou non linéaire, statique ou dynamique, …), ainsi que l'approche choisie pour effectuer la modélisation (représentations d'état ou "entrée-sortie", …);

– la *structure* du modèle, définie par l'architecture de la famille M de modèles, et par l'ensemble des variables (entrées, variables d'état, sorties, …) nécessaires pour exprimer la famille F de fonctions paramétrées $\phi(.;\theta)$ correspondante;

– les valeurs des *paramètres* θ du modèle.

Dans la pratique, le choix du type du modèle découle généralement d'une étude préalable du processus qui permet, par exemple, d'identifier la nature, linéaire ou non linéaire, statique ou dynamique, des phénomènes mis en jeu dans le processus : informations provenant de modélisations antérieures de processus de nature proche, étude rapide de la structure physique du processus, analyse des réponses du processus à des commandes particulières, telles que des sinus ou des échelons, … Le choix du type de modèle est donc spécifique de chaque processus que l'on cherche à modéliser.

Notre travail suppose qu'une étude préalable a conduit à choisir un modèle entrée-sortie "boîte noire", et nous considérons les modèles hypothèses

NARMAX. Dans cette hypothèse, la sélection d'une structure consiste donc à déterminer les arguments de $\varphi(.)$ et l'architecture de la famille de modèles correspondant à la famille de fonctions $\phi(.;\theta)$.

Lors d'une telle démarche, la détermination de la structure du modèle et l'estimation de ses paramètres se font conjointement. En effet, la sélection entre plusieurs structures consiste à comparer les modèles obtenus après estimation des paramètres de chacune des structures. La procédure d'estimation se fait donc pendant la sélection, pour chacune des structures de modèle candidates.

Dans le cadre de l'identification de processus à l'aide de modèles linéaires, il existe des méthodes statistiques conduisant à la sélection du meilleur modèle d'un ensemble donné, c'est-à-dire à la détermination des paramètres n_y, n_u et n_w, qui définissent alors complètement la structure du modèle, puisque l'architecture est linéaire.

Dans le cas de modèles dynamiques non linéaires, les approches existantes sont moins nombreuses et moins performantes. Elles peuvent être classées en trois ensembles :

– les méthodes fondées sur l'utilisation de tests statistiques, qui sont le plus souvent des transpositions de méthodes linéaires au cas non linéaire,

– les méthodes fondées sur une approche bayesienne de l'estimation des paramètres d'un modèle, qui conduisent aux méthodes de "weight decay" qui peuvent être interprétées comme des méthodes de sélections de modèles [MacKay 1992a,b], [Williams 95].

– les méthodes heuristiques, qui dérivent souvent des méthodes classiques, et qui, si elles ne reposent pas toujours sur des bases théoriques solides, peuvent néanmoins s'avérer performantes en pratique [Le Cun 90], [Reed 93], [Moody 94].

Nous nous sommes intéressés plus particulièrement aux méthodes statistiques, et nous allons présenter dans ce chapitre les principales méthodes de sélection de modèles linéaires et non linéaires. On distingue deux classes :

– les méthodes utilisant des tests d'hypothèses, qui permettent de comparer les structures de modèles deux à deux à l'aide d'un test statistique.

– les méthodes de sélections multiples, qui consistent à sélectionner, parmi un ensemble quelconque de structures, celle qui satisfait le mieux un critère.

Nous présentons tout d'abord l'Estimateur du Maximum de Vraisemblance (EMV), et le Test du Rapport de Vraisemblance (TRV). En effet, les méthodes de sélection fondées sur les tests d'hypothèse utilisent les propriétés asymptotiques de l'estimateur du maximum de vraisemblance.

III.2. L'estimateur du maximum de vraisemblance (EMV)

III.2.1. L'estimateur du maximum de vraisemblance

La méthode du maximum de vraisemblance est une méthode de conception d'estimateurs dont le principe a été énoncé par Fisher [Fisher 1912, 1921]. On dispose d'un ensemble de N données, définissant le vecteur $\mathbf{y}_p$, qui sont des réalisations d'autant de variables aléatoires. Considérons le vecteur aléatoire $\mathbf{Y}_p$, dont la réalisation est $\mathbf{y}_p$, et dont la fonction densité de probabilité est un membre $p_{\mathbf{Y}_p}(\mathbf{x}; \theta_o)$ d'une famille paramétrée de fonctions densité de probabilité $p_{\mathbf{Y}_p}(\mathbf{x}; \theta)$.

On appelle *fonction de vraisemblance* la fonction $L(\theta ; \mathbf{y}_p) = p_{\mathbf{Y}_p}(\mathbf{x}=\mathbf{y}_p; \theta)$. $L(\theta; \mathbf{Y}_p)$ est alors une variable aléatoire. Pour une réalisation $\mathbf{y}_p$ de $\mathbf{Y}_p$, l'estimation du maximum de vraisemblance de la valeur inconnue θ_o est la valeur $\hat{\theta}$ qui maximise $p_{\mathbf{Y}_p}(\mathbf{x}=\mathbf{y}_p; \theta)$. Soit $f_{MV}(\mathbf{y}_p)$ la fonction qui exprime $\hat{\theta}$ en fonction de $\mathbf{y}_p$; l'*estimateur du maximum de vraisemblance* de θ_o est la variable aléatoire :

$$\Theta = f_{MV}(\mathbf{Y}_p). \qquad \text{(III.1)}$$

III.2.2. Propriétés de l'EMV dans le cas de processus linéaires

L'estimateur du maximum de vraisemblance a été appliqué à la modélisation de processus dynamiques linéaires par des modèles ARMAX par Åström et Bohlin [Åström 65]. Leurs travaux reposent en particulier sur l'hypothèse qu'il existe, dans la famille paramétrée de modèles M, un modèle qui décrit exactement le processus. Si l'on note P le modèle hypothétique décrivant le processus, on a donc $P \in M$. On note θ_o la valeur inconnue correspondant à ce modèle exact (soit $P=M(\theta_o)$, ou $\varphi(.) = \phi(.;\theta_o)$). Åström et Bohlin ont montré que, dans ce cas, l'estimateur du maximum de vraisemblance est *consistant* (c'est à dire que, lorsque N tend vers $+\infty$, il converge presque sûrement vers la valeur "exacte" θ_o); il est de plus *asymptotiquement gaussien* (i.e. la distribution de l'estimateur converge vers une distribution gaussienne), et *asymptotiquement efficace* (i.e. la matrice de covariance de l'estimateur converge vers la borne de Cramer-Rao, qui est la borne inférieure de la matrice de covariance d'un estimateur non biaisé). D'autre structures dynamiques linéaires particulières ont été étudiées, pour lesquelles les propriétés de l'estimateur du maximum de vraisemblance ont été établies [Balakrishnan 68], [Caines 74].

Ces résultats, bien que très importants, ne sont relatifs qu'à des systèmes et des modèles linéaires. D'autre part, les démonstrations des propriétés asymptotiques reposent sur l'hypothèse que le processus est exactement décrit par l'un des modèles. Nous allons maintenant présenter un formalisme s'inscrivant dans un cadre plus général permettant de construire des estimateurs dont les propriétés

asymptotiques sont, dans de nombreux cas, asymptotiquements équivalents à l'estimateur du maximum de vraisemblance. Les modèles neuronaux, qui sont des approximateurs universels, sont bien adaptés à cette approche.

III.2.3. Formulation à l'aide de l'approche EP

L'estimateur du maximum de vraisemblance est un concept puissant. Il n'est cependant pas toujours facile à mettre en œuvre, car il utilise la fonction densité de probabilité des mesures (c'est une approche paramétrique), que l'on connaît rarement. De plus, de nombreuses propriétés de cet estimateur ont été démontrées dans le cas où l'on considère un ensemble de variables aléatoires indépendantes, dans le cas d'un processus linéaire statique [Cramer 46], [Wald 49], ou pour des processus dynamiques pouvant être décrits parfaitement par un modèle analytique [Åström 65].

L'approche des Méthodes fondées sur l'Erreur de Prédiction (Prediction Error Identification Methods) [Ljung 74], [Ljung 76a, b] conduit à des estimateurs simples à mettre en œuvre pour une large gamme de modèles, et ne nécessitent pas la connaissance de la fonction densité de probabilité des mesures (c'est une approche non paramétrique).

On montre que, dans le cas de processus linéaires gaussiens, statiques ou dynamiques, ces estimateurs sont strictement équivalents à l'estimateur du maximum de vraisemblance, et possèdent par conséquent les mêmes propriétés asymptotiques que celui-ci (consistant, asymptotiquement gaussien, asymptotiquement efficace) [Goodwin 77].

Supposons que le comportement dynamique du processus est exactement décrit par le modèle hypothèse NARX suivant :

$$y_p(t) = \varphi(y_p(t-1), \ldots, y_p(t-n_y), u(t-1), \ldots, u(t-n_u)) + w(t) \qquad \text{(III.2)}$$

où {w(t)} est une réalisation d'une séquence de variables aléatoires indépendantes identiquement distribuées (v.a.i.i.d) {W(t)}, d'espérance mathématique nulle et de densité de probabilité $p_{W(t)}(x)$. Supposons que les variables {$y_p(t–1)$, …, $y_p(t–n_y)$, $u(t–1)$, …, $u(t–n_u)$}, qui sont éventuellement des réalisations de variables aléatoires, sont connues à l'instant t–1. La variable aléatoire $Y_p(t) \mid y_p(t–1)$, …, $y_p(1)$, que nous noterons ici $Y_p(t)\mid t–1$, est une fonction de W(t); la relation liant ces deux variables aléatoire est :

$$Y_p(t)\mid t–1 = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u)) + W(t) \qquad \text{(III.3)}$$

et l'on a :

$$E[Y_p(t)\mid t–1] = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u)) \qquad \text{(III.4)}$$

Nous pouvons exprimer la fonction densité de probabilité de la variable aléatoire $Y_p(t)\mid t–1$ à partir de celle de W(t) :

$$p_{Y_p(t)\mid t-1}(x) = p_{W(t)}(x – E[Y_p(t)\mid t–1]) \qquad \text{(III.5)}$$

Rappelons la définition de la densité de probabilité conjointe de deux variables aléatoires :

$$P_{Y_1, Y_2}(x_1, x_2) = P_{Y_1\mid Y_2}(x_1)\ P_{Y_2}(x_2)$$

On peut donc exprimer la densité de probabilité conjointe de l'échantillon $\mathbf{Y}_p$ sous la forme :

$$p_{\mathbf{Y}_p}(\mathbf{x}^N) = p_{Y_p(N)\mid \mathbf{y}_p{}^{N-1}}(x)\ \ p_{\mathbf{Y}_p{}^{N-1}}(\mathbf{x}^{N-1}) = p_{Y_p(N)\mid N-1}(x)\ \ p_{\mathbf{Y}_p{}^{N-1}}(\mathbf{x}^{N-1}) \qquad \text{(III.6)}$$

où $\mathbf{Y}_p{}^t = [Y_p(t), \ldots, Y_p(1)]$, $\mathbf{y}_p{}^t = [y_p(t), \ldots, y_p(1)]$, et $\mathbf{x}^t$ est un vecteur de dimension t. Remarquons que $\mathbf{Y}_p = \mathbf{Y}_p{}^N$. En décomposant de la même façon la densité de probalilité conjointe de $\mathbf{Y}_p{}^{N–1}$, puis de $\mathbf{Y}_p{}^{N–2}$, …, on peut finalement exprimer (III.6) sous la forme :

$$p_{\mathbf{Y}_p}(\mathbf{x}^N) = p_{\mathbf{Y}_p{}^N}(\mathbf{x}^N) = \prod_{t=1}^{N} p_{Y_p(t)\mid t-1}(x_t) \qquad \text{(III.7)}$$

Lorsque la fonction densité de probabilité de W(t) est une loi gaussienne, donnée par :

$$p_{W(t)}(x) = \frac{1}{\sqrt{2\pi}\,\sigma}\ \exp\left(-\frac{x^2}{2\sigma^2}\right), \qquad \text{(III.8)}$$

les fonctions densité de probabilité de $Y_p(t)\mid t–1$ et $\mathbf{Y}_p$ ont alors pour expressions :

$$p_{Y_p(t)\mid t-1}(x) = \frac{1}{\sqrt{2\pi}\,\sigma}\ \exp\left(-\frac{1}{2}\,\frac{(x – E[Y_p(t)\mid t–1])^2}{\sigma^2}\right) \qquad \text{(III.9.a)}$$

$$p_{\mathbf{Y}_p}(\mathbf{x}^N) = \prod_{t=1}^{N} \frac{1}{\sqrt{2\pi}\,\sigma}\ \exp\left(-\frac{1}{2}\,\frac{(x_t – E[Y_p(t)\mid t–1])^2}{\sigma^2}\right) \qquad \text{(III.9.b)}$$

Lors de l'énoncé du principe du maximum de vraisemblance, nous avons fait l'hypothèse que la densité de probabilité était un membre d'une famille paramétrée de fonctions densités. Nous allons donc supposer qu'il existe une famille de fonctions paramétrées $\phi(.;\theta)$ et une valeur θ_o telles que :

$$E[Y_p(t) \mid t-1] = \phi(y_p(t-1), \ldots, y_p(t-n_y), u(t-1), \ldots, u(t-n_u); \theta_o)$$

La densité de probabilité de la variable aléatoire $Y_p(t)|t-1$ s'écrit maintenant :

$$p_{Y_p(t)|t-1}(x;\theta_o, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{1}{2}\frac{(x - y(t;\theta_o))^2}{\sigma^2}\right) \qquad \text{(III.10)}$$

avec :

$$y(t;\theta) = \phi(y_p(t-1), \ldots, y_p(t-n_y), u(t-1), \ldots, u(t-n_u); \theta)$$

Notons $e(t;\theta) = y_p(t) - y(t;\theta)$, l'erreur de prédiction obtenue avec une valeur θ quelconque. Considérons d'autre part σ comme un paramètre inconnu. La fonction de vraisemblance relative aux paramètres (θ, σ) est :

$$L(\theta, \sigma; \mathbf{y}_p) = p_{\mathbf{Y}_p}(\mathbf{x}=\mathbf{y}_p; \theta, \sigma) = \prod_{t=1}^{N} \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{1}{2}\frac{e(t;\theta)^2}{\sigma^2}\right) \qquad \text{(III.11)}$$

On considère alors la quantité :

$$L^*(\theta, \sigma; \mathbf{y}_p) = \ln L(\theta, \sigma; \mathbf{y}_p) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2}\sum_{t=1}^{N}\frac{e(t;\theta)^2}{\sigma^2} \qquad \text{(III.12)}$$

Pour une valeur fixée θ des paramètres, l'estimation du maximum de vraisemblance $\hat{\sigma}^2$ de σ^2 est donné par :

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{t=1}^{N} e(t;\theta)^2 \qquad \text{(III.13)}$$

En remplaçant σ^2 par son estimation (III.13), la relation (III.12) s'écrit au point $(\theta, \hat{\sigma}^2)$:

$$L^*(\theta, \hat{\sigma}^2; \mathbf{y}_p) \cong -\frac{N}{2}(\ln(2\pi)+1) - \frac{N}{2}\ln\left(\frac{1}{N}\sum_{t=1}^{N} e(t;\theta)^2\right) \qquad \text{(III.14)}$$

La maximisation de $L^*(\theta; \mathbf{y}_p)$ par rapport à θ est donc équivalente à la minimisation de $\frac{N}{2}\ln(J(\theta))$, où $J(\theta)$ est la fonction de coût quadratique classique :

$$J(\theta) = \frac{1}{N}\sum_{t=1}^{N} e(t;\theta)^2 \qquad \text{(III.15)}$$

On retrouve l'expression du coût minimisée avec la méthode des moindres-carrés ordinaires, qui est un estimateur EP particulier. Lorsque le processus est

parfaitement décrit par l'un des modèles de la famille $M(\theta)$ ($P=M(\theta_o)$), cet estimateur est consistant, asymptotiquement gaussien et asymptotiquement efficace. Ljung montre que ces résultats restent valables pour tous les modèles pouvant être décrits par une relation du type :

$$Y_p(t) = f(\mathbf{y}_p^{t-1}, \mathbf{u}^{t-1} ; \theta) + V(t) \qquad \text{(III.16)}$$

où $\mathbf{y}_p^{t-1} = \{y_p(t-1), \ldots, y_p(-\infty)\}$, $\mathbf{u}^{t-1} = \{u(t-1), \ldots, u(-\infty)\}$, et $\{V(t)\}$ est une séquence de v.a.i.i.d. dont la distribution n'est pas nécessairement gaussienne. Il semble donc tout à fait légitime d'utiliser l'approche EP pour effectuer l'estimation des modèles NARMAX.

Lorsque $P \notin M$, on recherche la meilleure approximation du processus dans M, c'est-à-dire telle que la valeur de $E[J(\theta; \mathbf{Y}_p)]$ soit minimale. On cherche donc la valeur θ_N^* de θ telle que :

$$E[J(\theta_N^* ; \mathbf{Y}_p)] = \min \{ E[J(\theta; \mathbf{Y}_p)] \} \qquad \text{(III.17)}$$

La fonction $\phi(.;\theta_N^*)$ est la meilleure approximation de $\varphi(.)$ (ou de $f(.)$, si l'on choisit la formulation (III.16)) que l'on puisse construire à partir d'un ensemble d'apprentissage de taille N, au sens du coût théorique $E[J(\theta;Y_p)]$. On note θ^* la valeur limite de θ_N^* lorsque N tend vers l'infini. Il a été montré, sous des conditions générales portant sur l'ensemble d'apprentissage, la famille de modèles considérées, et la fonction de coût choisie [Ljung 78], [Ljung 79], qu'un estimateur EP de θ^* est consistant et asymptotiquement gaussien. De plus, s'il existe une valeur θ_o telle que la séquence $\{e(t;\theta_o)\}$ des erreurs de prédiction obtenue avec $M(\theta_o)$ soit une séquence de variables indépendantes, alors $\theta^* = \theta_o$.

Dans le cas d'ensembles d'apprentissage de grande taille, pour des modèles NARX, avec la fonction de coût classique des moindres carrés (III.15), l'utilisation d'une approche EP est parfaitement justifiée. De la même manière, les estimateurs EP seront utilisés pour des modèles NARMAX.

Lorsque l'on dispose d'un ensemble d'apprentissage de taille réduite, et qu'aucune information particulière sur la distribution des données n'est disponible, nous utilisons encore la fonction de coût $J(\theta)$ et l'estimateur EP correspondant, bien qu'aucune propriété de l'estimateur ne soit démontrée dans ce cas.

III.3. Les tests d'hypothèses statistiques

Les tests d'hypothèses peuvent s'appliquer pour résoudre de nombreux de problèmes dans lesquels il est nécessaire de prendre une décision. Nous les présentons ici dans le cadre particulier de la sélection d'un modèle parmi plusieurs. Pour faciliter la lecture, nous appellerons dans ce chapitre "modèle" aussi bien une structure qui définit une famille de modèles paramétrés qu'un modèle correspondant à une valeur particulière de θ.

III.3.1. Principe des tests d'hypothèses

On désire modéliser un processus, et l'on dispose pour cela d'un ensemble d'apprentissage constitué d'une séquence de taille N d'entrée-sortie $\{u(t), y_p(t)\}$ de ce processus.

On dispose d'un modèle M_A du processus, paramétré par le vecteur de paramètres θ_A, tel que $\theta_A = [\ \mathbf{a}^T\ \mathbf{b}^T\ ]^T$, et l'on fait l'hypothèse, non remise en cause par la suite, qu'il existe un vecteur $\theta_{Ao} = [\ \mathbf{a}_o^T\ \mathbf{b}_o^T\ ]^T$ tel que $P=M_A(\theta_{Ao})$. M_A est appelé le *modèle complet*, et l'on note d_A la dimension de θ_A, d_a la dimension de $\mathbf{a}$, et $s = (d_A - d_a)$ la dimension de $\mathbf{b}$.

Considérons maintenant un deuxième modèle noté M_0, paramétré par le vecteur de paramètre $\theta_0 = [\ \mathbf{a}^T\ \mathbf{b}^{*T}\ ]^T$, où $\mathbf{b}^*$ est un vecteur imposé, par exemple $\mathbf{b}^*=0$ (le modèle $M_0 = M(\theta_0)$ est alors un sous-modèle du modèle M_A). On désire savoir si le modèle restreint M_0 est suffisant pour représenter correctement le processus, c'est-à-dire si $\mathbf{b}_o = \mathbf{b}^*$.

On définit donc l'hypothèse suivante, que l'on appelle *hypothèse nulle* :

$$[\ \mathbf{H_0} : \quad \mathbf{b}_o = \mathbf{b}^*\ ]$$

On définit également *l'hypothèse alternative* :

$$[\ \mathbf{H_A} : \quad \mathbf{b}_o \neq \mathbf{b}^*\ ]$$

Afin de savoir si le modèle M_0 est suffisant relativement à l'ensemble d'apprentissage, on teste l'hypothèse nulle : pour cela, on construit une variable aléatoire qui suit une loi de distribution connue si l'hypothèse nulle est exacte; on divise l'ensemble des valeurs possibles que peut prendre la variable aléatoire en deux sous-ensembles, de telle sorte que, si l'hypothèse nulle est exacte, la probabilité qu'une réalisation de cette variable aléatoire appartienne au premier sous-ensemble soit beaucoup plus grande que la probabilité qu'elle appartienne au second; on calcule la valeur de sa réalisation obtenue avec l'ensemble d'apprentissage dont on dispose; si cette réalisation n'appartient pas au sous ensemble correspondant à la probabilité la plus grande, on rejette l'hypothèse nulle, sinon, on l'accepte.

Par exemple, on se donne un couple de valeurs (α_1, α_2), et l'on rejette l'hypothèse H_0 si la valeur de la variable aléatoire n'appartient pas à [α_1, α_2]. Deux types d'erreurs peuvent survenir :

– [1] on rejette l'hypothèse nulle alors qu'elle est vraie

– [2] on accepte l'hypothèse nulle alors qu'elle est fausse

On appelle *risque de première espèce* la probabilité $r=p_{(i)}$ pour que la réalisation d'une variable aléatoire suivant la loi de probabilité que l'on considère n'appartienne pas à [α_1, α_2] (erreur de type [1]), et le *niveau de confiance* la probabilité complémentaire $(1-p_{(i)})$. Le choix du couple de valeurs (α_1, α_2) fixe donc le risque r, et lorsque l'on désire effectuer un test, on choisit donc deux valeurs (α_1, α_2) telles que l'on obtienne un risque r.

III.3.2. Le Test du Rapport de Vraisemblance (TRV)

On dispose de deux modèles M_A et M_0, et l'on veut construire un test permettant de décider si le modèle restreint M_0 est suffisant pour décrire le processus, étant donné l'ensemble d'apprentissage dont on dispose. Pour construire le *test du rapport de vraisemblance*, on considère le rapport :

$$\lambda(\mathbf{y}_p) = \frac{p_{Y_p}(\mathbf{x}=\mathbf{y}_p;\theta_A)}{p_{Y_p}(\mathbf{x}=\mathbf{y}_p;\theta_0)} = \frac{L(\theta_A;\mathbf{y}_p)}{L(\theta_0;\mathbf{y}_p)} \qquad \text{(III.18)}$$

appelé *rapport de vraisemblance*, où $\theta_0 = \theta_0(\mathbf{y}_p)$ est l'estimation du maximum de vraisemblance des paramètres du modèle restreint (hypothèse nulle), et $\theta_A = \theta_A(\mathbf{y}_p)$ l'estimation du maximum de vraisemblance des paramètres du modèle complet (hypothèse alternative). Lorsque l'hypothèse nulle est vraie, ce rapport est proche de 1; dans le cas contraire, la valeur de $\lambda(y_p)$ est généralement grande. Si l'on remplace la valeur $\mathbf{y}_p$ par la variable aléatoire $\mathbf{Y}_p$, la fonction aléatoire $\lambda(\mathbf{Y}_p)$ est une statistique telle que, sous l'hypothèse nulle, la variable aléatoire :

$$d(\mathbf{Y}_p) = 2 \ln \lambda(\mathbf{Y}_p) = 2 \ln \left(\frac{L(\Theta_A;\mathbf{Y}_p)}{L(\Theta_0;\mathbf{Y}_p)}\right) = 2L^*(\Theta_A;\mathbf{Y}_p) - 2L^*(\Theta_0;\mathbf{Y}_p) \qquad \text{(III.19)}$$

converge en loi vers une variable aléatoire de Pearson (χ^2) à s degrés de liberté [Goodwin 77] lorsque N tend vers l'infini. Cette propriété permet de construire le test du rapport de vraisemblance.

III.3.3. Test du Rapport de Vraisemblance et estimateurs EP : le test LDRT

Comme pour l'estimateur du maximum de vraisemblance, la mise en œuvre du test du rapport de vraisemblance nécessite la connaissance de la fonction de vraisemblance, dont on ne dispose généralement pas. Cependant, nous avons vu que dans le cas d'un ensemble d'apprentissage de grande taille, l'estimation du maximum de vraisemblance pouvait être obtenue, sans connaître l'expression de la fonction de vraisemblance, en utilisant un estimateur EP. En effet, lorsque N tend vers $+\infty$, la maximisation de $\ln(L(\theta;\mathbf{Y}_p))$ est asymptotiquement équivalente à la minimisation de la fonction de coût $\frac{N}{2} \ln J(\theta)$, où $J(\theta)$ est l'erreur quadratique moyenne.

Les expressions de $\lambda(\mathbf{Y}_p)$ et de $d(\mathbf{Y}_p)$ peuvent être réécrites en tenant compte de ces équivalences, et la relation (III.19) devient alors :

$$d(\mathbf{Y}_p) = 2 \ln \lambda(\mathbf{Y}_p) = N \ln J(\Theta_0) - N \ln J(\Theta_A) = N \ln \frac{J(\Theta_0)}{J(\Theta_A)} \qquad \text{(III.20)}$$

où Θ_0 et Θ_A sont les estimateurs correspondants respectivement à l'hypothèse nulle et à l'hypothèse alternative. La variable aléatoire $d(\mathbf{Y}_p)$ converge en loi vers une variable aléatoire de Pearson (χ^2) à s degrés de liberté. Le test ainsi défini est connu, dans le cas plus général où la sortie $y_p(t)$ est vectorielle, sous le nom de "Logarithm Determinant Ratio Test", ou LDRT [Leontaritis 87].

III.3.4. Le test de Fisher

Le test de Fisher est un test classique dans le cas gaussien pour les modèles linéaires par rapport aux paramètres, fondé sur le résultat suivant :

Soient $\mathbf{Y}_p = [Y_p(1), \ldots, Y_p(N)]^T$ un vecteur aléatoire gaussien de R^N, d'espérance mathématique $\mu = [\mu(1), \ldots, \mu(N)]^T$ et de variance $\sigma^2 I$, $\mathbf{Y}_A$ sa projection orthogonale sur H_A (sous-espace vectoriel de R^N de dimension d_A), et $\mathbf{Y}_0$ sa projection sur H_0 (sous-espace de H_A de dimension $d_0 < d_A$), on a les propriétés suivantes :

– les vecteurs $\mathbf{Y}_A$ et $\mathbf{Y}_p - \mathbf{Y}_A$ sont indépendants,

– si μ est un vecteur de H_A, la variable aléatoire :

$$X = \frac{\|\mathbf{Y}_p - \mathbf{Y}_A\|^2}{\sigma^2}$$

suit une loi de Pearson (χ^2) à $(N - d_A)$ degrés de liberté

– si μ est un vecteur de H_0, la variable aléatoire :

$$T = \frac{\|\mathbf{Y}_A - \mathbf{Y}_0\|^2/(d_A - d_0)}{\|\mathbf{Y}_p - \mathbf{Y}_A\|^2/(N - d_A)} = \frac{\|\mathbf{Y}_A - \mathbf{Y}_0\|^2}{\|\mathbf{Y}_p - \mathbf{Y}_A\|^2} \left(\frac{N - d_A}{d_A - d_0}\right)$$

suit une loi de Fisher à $(d_A - d_0 , N - d_A)$ degré de liberté

Dans le contexte de la sélection de modèle à l'aide de tests d'hypothèses, $\mathbf{Y}_p$ peut être interprété comme le vecteur des données d'apprentissage, $\mathbf{Y}_A$ le vecteur des prédictions obtenues avec un modèle linéaire M_A, et $\mathbf{Y}_0$ le vecteur des prédictions obtenues avec un sous modèle M_0 de M_A.

Si l'hypothèse nulle est vraie, la variable T suit une loi de Fisher à (d_A–d_o, N – d_A) degrés de liberté. Le test de Fisher est le test d'hypothèse le plus utilisé dans le cas de modèles linéaires par rapport aux paramètres. On peut montrer que les tests TRV, LDRT et le test de Fisher sont asymptotiquement équivalents [Söderström 77].

III.3.5. Sélection d'un modèle dans un ensemble

On considère un ensemble de modèles E_m, et l'on désire sélectionner un modèle de cet ensemble, à l'aide de tests d'hypothèses. On suppose que l'on sait construire un modèle M_c dont tous les modèles de notre ensemble sont des sous-modèles, et qui est suffisamment complexe pour approcher le processus de façon satisfaisante. Le modèle M_c est appelé *modèle complet*. Tous les modèles peuvent ainsi être comparés au modèle complet à l'aide d'un test. Il faut alors définir une procédure qui permet, à partir du résultat de tous les tests, de sélectionner l'un de ces modèles.

En pratique, on procède généralement de la manière suivante : à partir des connaissances *a priori* sur le processus, on construit le modèle complet M_c. L'ensemble des modèles sur lequel va s'effectuer la sélection est alors l'ensemble (ou un sous ensemble) de tous les sous-modèles de M_c.

Toutes les comparaisons "modèle complet/sous-modèle" sont effectuées. Si aucun sous-modèle n'est accepté, le modèle sélectionné est donc M_c. Lorsque plusieurs sous-modèles sont acceptés, qui ne peuvent être comparés entre eux (aucunes relations d'inclusion ne peuvent être établies entre les différents modèles), on choisit le moins complexe. Lorsque plusieurs modèles de même taille sont acceptés, on ne peut pas les comparer à l'aide de tests. On utilise généralement pour choisir l'un d'entre eux un critère particulier, par exemple la valeur de la fonction de coût calculée sur un ensemble de données particulières (ensemble d'apprentissage ou ensemble de données nouvelles).

Cette méthode est simple à mettre en œuvre, mais elle est coûteuse en nombre d'estimations de paramètres et de tests à effectuer lorsque l'ensemble des modèles est de grande taille. Nous présenterons dans le paragraphe III.5. des moyens de réduire ce nombre de façon significative.

D'autre part, il faut noter que l'utilisation de tests d'hypothèses pour la sélection peut théoriquement conduire à des résultats contradictoires: soient M_1,

M_2 et M_3 trois modèles tels que M_3 est un sous-modèle de M_2, qui est un sous-modèle de M_1. Il n'existe aucune garantie que les résultats des tests (M_2 comparé à M_1) et (M_3 comparé à M_2) soient cohérents avec le résultat du test (M_3 comparé à M_1). Nous reviendrons sur ce point au paragraphe III.4.2.2.

III.4. Les méthodes de sélections multiples

Une approche différente de la sélection de modèle a été développée, en particulier par Akaike, à partir du principe suivant : on définit une fonction de coût qui tient compte à la fois de la performance d'un modèle sur les données d'apprentissage, et de la complexité de la structure du modèle. Le modèle sélectionné est celui qui minimise cette fonction de coût. Cette fonction pouvant être calculée pour chacun des modèles indépendamment des autres, une sélection peut être effectuée sur un ensemble quelconque de modèles.

Nous nous intéressons plus particulièrement aux méthodes développées par Akaike, dont nous présentons succinctement le principe [Akaike 69], [Akaike 74a,b].

III.4.1. Principe des méthodes d'Akaike

Dans les chapitres précédents, nous avons présenté l'estimation des paramètres d'un modèle comme un problème d'optimisation. Pour cela, nous avons défini la fonction de coût théorique, et une estimation de cette dernière, la fonction de coût empirique.

Le problème de la sélection d'un modèle peut être formulé de façon similaire. Considérons un ensemble E_m de modèles, dont chaque élément est caractérisé par une structure $M^k=M(\theta^k)$ (l'ensemble des régresseurs, son architecture, …), et par le vecteur θ^k des paramètres. Comme pour l'estimation des paramètres, on définit une fonction de coût $I(M^k, \theta^k)$ qui mesure la performance de chacun des modèles, mais prend également en considération leur structure en pénalisant les modèles les plus complexes. La sélection d'un modèle consiste donc à déterminer le couple $(M^k, \hat{\theta}^k)$ tel que :

$$(M^k, \hat{\theta}^k) = \arg_{[E_m]} \arg_{[\theta^k]} \min \{ I(M^k, \theta^k) \} \qquad \text{(III.22)}$$

Soient $\Theta_N^k(Y_p)$ l'estimateur des paramètres θ^k du modèle M^k, et $J_N(\theta^k)$ la fonction de coût utilisée pour construire l'estimateur des paramètres $\Theta_N^k(Y_p)$. On choisit comme "mesure" du modèle M^k la quantité :

$$I(M_k, \theta^k) \stackrel{\Delta}{=} \bar{J}(\hat{\theta}^k_*) \qquad \text{(III.23)}$$

avec :

$$\bar{J}(\theta^k) = \lim_{N->\infty} E[\, J_N(\theta^k)], \qquad \hat{\theta}^k_* = \lim_{N->\infty} E[\Theta_N^k(\mathbf{Y}_p)]$$

Cette valeur théorique ne peut être calculée, mais une bonne estimation est fournie par [Ljung 87] :

$$\hat{I}(M_k, \theta^k) \cong J_N(\hat{\theta}^k_N) + \frac{1}{N} \text{ trace}[\, \bar{J}\,''(\hat{\theta}^k_*)\, P^k_\theta] \qquad \text{(III.24)}$$

où :

- $\bar{J}\,''(\hat{\theta}^k_*)$ est la matrice des dérivées secondes de $\bar{J}(\theta^k)$ au point $\theta = \hat{\theta}^k_*$
- P^k_θ est la matrice de covariance asymptotique de $\sqrt{N(\Theta^k_N - \Theta^k_*)}$.

III.4.2. Le critère d'information d'Akaike (AIC)

III.4.2.1. Définition

Choisissons comme fonction de coût :

$$J_N(\theta^k) = -\frac{1}{N} \ln(\, L(\theta^k; \mathbf{Y}_p)\,) = -\frac{1}{N}\, L^*(\theta^k; \mathbf{Y}_p) \qquad \text{(III.25)}$$

où $L(\theta^k;\mathbf{Y}_p)$ est la fonction de vraisemblance des paramètres θ^k pour un ensemble d'apprentissage $\mathbf{Y}_p$ de taille N. Si l'on suppose, d'une part, que le processus correspond à la valeur $\theta_o = \hat{\theta}^k_*$, d'autre part que $\bar{J}\,''(\theta_o)$ est inversible, on montre [Ljung 87] que :

$$P^k_\theta = \bar{J}\,''(\hat{\theta}^k_*)^{-1} = \bar{J}\,''(\theta_o)^{-1}$$

En utilisant (III.24), on obtient alors le critère d'information d'Akaike ("Akaike Information Criterion" ou "AIC") :

$$\mathbf{AIC}(M^k, \theta^k) = J_N(\hat{\theta}^k_N) + \frac{\dim(\theta^k)}{N} = -\frac{1}{N}\, L^*(\hat{\theta}^k_N) + \frac{\dim(\theta^k)}{N}$$

(III.26)

Nous avons montré dans le paragraphe III.2.3. que, pour un modèle NARX, et lorsque le bruit {W(t)} est gaussien de variance inconnue σ^2, la fonction $L^*(\theta^k, \sigma; Y_p)$ s'écrit :

$$L^*(\theta^k, \sigma) = -\frac{1}{2} \sum_{t=1}^{N} \frac{e(t;\theta^k)^2}{\sigma^2} - \frac{N}{2} \ln(\sigma^2) - \frac{N}{2} \ln(2\pi)$$

En remplaçant σ^2 par son estimation donnée par (III.13), et en supprimant dans l'expression obtenue les termes indépendants de θ^k (puisqu'ils ont la même valeur pour tous les modèles), on obtient :

$$\mathbf{AIC}(M^k, \theta^k) = N \ln [\sum_{t=1}^{N} e(t;\hat{\theta}_N^k)^2] + 2 \dim(\theta^k) \qquad \text{(III.27)}$$

III.4.2.2. Lien avec la sélection à l'aide de tests d'hypothèse

Nous allons ici montrer que la sélection à l'aide du critère d'information d'Akaike peut être interprétée comme une méthode particulière de sélection à l'aide de tests d'hypothèse.

Soient $\mathbf{Y}_p$ un ensemble de données d'apprentissage, M_0, M_1, et M_2 trois modèles de dimensions respectives d_o, $d_1 = d_0 - 1$ et $d_2 = d_0 - 2$. M_0 est un modèle complet, M_1 un modèle restreint de M_0, et M_2 un modèle restreint de M_1. On veut sélectionner l'un de ces trois modèles à l'aide du test défini à partir de (III.19).

On considère les modèles M_0 et M_1. Pour une réalisation $\mathbf{y}_p$ de $\mathbf{Y}_p$, on compare la réalisation de (III.19) :

$$2 \ln(\lambda(\mathbf{y}_p)) = 2 \ln L(\hat{\theta}_0; \mathbf{y}_p) - 2 \ln(L(\hat{\theta}_1; \mathbf{y}_p)) = 2L^*(\hat{\theta}_0) - 2L^*(\hat{\theta}_1) \quad \text{(III.28)}$$

à une valeur critique k(1), correspondant au risque r(1) que l'on s'est fixé. Lorsque :

$$2[L^*(\hat{\theta}_0) - L^*(\hat{\theta}_1)] < k(1), \qquad \text{(III.29)}$$

on ne peut rejeter l'hypothèse nulle à partir des données d'apprentissage dont on dispose (par abus de langage, on dira qu'on accepte le modèle M_1). On compare maintenant de la même manière M_2 à M_1, et M_2 est également accepté ($2[L^*(\hat{\theta}_1) - L^*(\hat{\theta}_2)] < k(1)$).

Enfin, on compare M_2 directement à M_0, et l'on note k(2) la valeur critique pour un risque r(2) et s=2 degrés de liberté. Si les deux premiers tests ont conduit à ne pas rejeter, d'une part, M_1 par rapport à M_0, d'autre part, M_2 par rapport à M_1, on veut que, lorsque l'on compare M_2 directement à M_0, le test ne rejette pas M_2. Pour cela, on choisit k(2) tel que $k(2) \geq k(1)+k(1)$, d'où :

$$2 (L^*(\hat{\theta}_0) - L^*(\hat{\theta}_2)) = 2 [L^*(\hat{\theta}_0) - L^*(\hat{\theta}_1) + L^*(\hat{\theta}_1) - L^*(\hat{\theta}_2)] < k(1) + k(1) \leq k(2)$$

De façon analogue, si les deux premiers conduisent à rejeter d'une part M_1 par rapport à M_0, d'autre part M_2 par rapport à M_1, on veut que le test de M_2 par rapport à M_0 rejette M_2. On choisit alors k(2) tel que $k(2) \leq k(1)+k(1)$. Le seul choix

satisfaisant ces deux conditions est k(2) = 2 k(1), et de façon plus générale, k(s) = s k(1).

Pour s quelconque, on peut écrire :

$$2[L^*(\hat{\theta}_0) - L^*(\hat{\theta}_1)] < s\,k(1) = (d_0 - d_1)\,k(1)$$

soit :

$$-2L^*(\hat{\theta}_1) + d_1\,k(1) < -2L^*(\hat{\theta}_0) + d_0\,k(1)$$

Pour effectuer la sélection d'un modèle dans un ensemble, on calcule pour chaque modèle $M_i(\theta_i)$ de dimension d_i la valeur :

$$C_i = -2L^*(\hat{\theta}_i) + d_i\,k(1) = -2 \ln L(\hat{\theta}_i; \mathbf{y}_p) + d_i\,k(1) \qquad \text{(III.30)}$$

et l'on sélectionne le modèle qui minimise cette expression. Pour k(1) = 2, on retrouve, au facteur 1/2N près, l'expression du critère Akaike (III.26).

L'approche Akaike peut donc être interprétée comme un cas particulier de la méthode de sélection à l'aide du test TRV, dans laquelle le niveau de confiance (ou le risque) avec lequel on effectue un test dépend de la différence entre les dimensions de chaque modèle. Si l'on choisit pour s=1 la valeur critique k(1)=2, les valeurs critiques pour s=2, 3, … sont alors données par k(2)=2k(1), k(3)=3k(1), …

Shibata [Shibata 76] a montré que le critère d'information d'Akaike (III.26) a généralement tendance à sur-dimensionner le vecteur des paramètres. Différentes valeurs de k(1) ont été étudiées [Bhansali 77]. Stone [Stone 77] a étudié le lien entre sélection à l'aide du critère AIC et validation croisée : ce critère reflète la variance des erreurs d'un modèle que l'on obtient avec un modèle sur un ensemble de données différent de l'ensemble de l'ensemble d'apprentissage.

III.4.3 Critère Final d'Akaike fondée sur l'erreur de prédiction (FPE)

Choisissons maintenant la fonction de coût suivante :

$$J_N(\theta) = \frac{1}{N} \sum_{t=1}^{N} e(t;\theta)^2 \qquad \text{(III.31)}$$

Si l'on suppose, une fois encore, que le processus correspond à $\theta_o = \hat{\theta}^k_*$, et que $\bar{J}''(\theta_o)$ est inversible, on obtient l'expression du critère FPE d'Akaike (Akaike's Final Prediction-Error Criterion) [Akaike 1969] :

$$\mathbf{FPE}(M^k, \theta^k) = \frac{1}{2} \frac{N+\dim(\theta^k)}{N-\dim(\theta^k)} J_N(\hat{\theta}^k_N) \qquad \text{(III.32)}$$

Notons que si l'on rapproche, d'une part, l'expression du critère d'information d'Akaike (III.27) dans le cas d'une séquence de bruit {W(t)} gaussienne, et, d'autre part, le critère FPE (III.32), on peut montrer que ces deux critères sont

asymptotiquement équivalents. Toutes les propriétés asymptotiques du critère AIC sont donc valables pour le critère FPE. Comme pour le critère AIC, il existe

une formulation plus générale du critère qui fait apparaître la valeur de k(1) :

$$\mathbf{FPE}(Mk, \theta^{(k)}, k(1)) = \frac{1}{2} \frac{2N+k(1)\ \dim(\theta^k)}{2N-k(1)\ \dim(\theta^k)}\ J_N(\hat{\theta}_N^k) \qquad \text{(III.33)}$$

III.5. Méthodes de sélection "partielles"

Les méthodes de sélection que nous venons de présenter, aussi bien celles fondées sur les tests d'hypothèses que les méthodes de type "Akaike", nécessitent l'apprentissage de tous les modèles candidats. Ceci est souvent irréalisable en pratique : pour un ensemble E_m constitué de tous les sous-modèles d'un modèle complet de dimension n_θ, le nombre de modèles en compétition est 2^{n_θ}. Pour un modèle de taille "raisonnable", le nombre de sous-modèles devient très vite prohibitif : si l'on considère par exemple un réseau de neurones complètement connecté, possédant deux entrées (la commande, scalaire, et une entrée constante), et 3 neurones, le nombre de paramètres est 9 et il existe 512 sous-modèles de ce réseau. Si le modèle possède 3 entrées et 4 neurones (soit 18 paramètres), le nombre de sous-modèles est 262.144 !

Deux méthodes sous-optimales, que nous appellerons méthode "destructive" et méthode "constructive", permettent de réduire considérablement le nombre d'apprentissages nécessaire pour sélectionner un modèle à l'aide d'une méthode d'Akaike.

III.5.1 Méthode "destructive"

Cette méthode, que l'on rencontre parfois dans la littérature sous le nom de procédure SBE (Stepwise Backward Elimination) [Draper 81], consiste à comparer le modèle complet à tous les sous-modèles ayant un paramètre de moins que le modèle complet. Soit $C(\theta)$ un critère de type AIC ou FPE, le modèle qui correspond à la valeur de C la plus faible est sélectionné. Si ce n'est pas le modèle complet, la sélection continue, et s'étend alors à tous les sous-modèles du modèle sélectionné qui correspondent à s=2. La procédure continue ainsi jusqu'à ce qu'un modèle correspondant à $s=s_i$ minimise C, et qu'aucun de ses sous-modèles ne fournisse une meilleure valeur de C.

Le nombre maximum de modèles (et d'apprentissages) impliqués par une telle procédure est :

$$1 + \frac{n_\theta(n_\theta+1)}{2}$$

c'est-à-dire 46 modèles pour un modèle complet possédant 9 paramètres, et 172 modèles pour un modèle complet à 18 paramètres.

Cette méthode peut facilement être adaptée à une sélection à l'aide de tests d'hypothèses : parmi tous les sous-modèles du modèle complet possédant un paramètre imposé à une valeur nulle (s=1), on détermine celui qui minimise la fonction de coût qui intervient dans l'expression du test d'hypothèse, et on le compare au modèle complet à l'aide d'un test. S'il n'est pas rejeté, on considère l'ensemble de ses sous-modèles possédant un paramètre de moins que lui, c'est-à-dire deux de moins que le modèle complet (s=2). On détermine à nouveau parmi ses sous-modèles celui qui minimise la fonction de coût, et on le compare également au modèle complet. On continue jusqu'à obtenir un modèle qui n'est pas rejeté par le test, et dont tous les sous-modèles possédant un paramètre de moins sont rejetés. On appellera cette méthode la méthode de sélection avec "Modèle Complet Unique" (MCU).

Une autre méthode est envisageable : lorsque qu'un sous-modèle du modèle complet n'est pas rejeté par le test, on supposera que l'on peut légitimement l'utiliser comme modèle complet pour la suite. Chaque sous-modèle n'est comparé qu'à son "sur-modèle" immédiat. Nous parlerons alors de méthode avec "Modèles Complets Multiples" (MCM).

III.5.2 Méthode "constructive"

Cette méthode, appelée aussi procédure SFI (Stepwise Forward Inclusion), est très proche dans l'esprit de la méthode précédente : on considère tout d'abord le modèle correspondant à $s=n_\theta$ (c'est-à-dire que le modèle est constitué d'un paramètre constant), ainsi que tous les modèles correspondant à $s=n_\theta-1$ (modèles à 1 paramètre). Si l'un d'entre eux est meilleur que le "modèle constant", au sens d'un critère C de type AIC, on le sélectionne, puis l'on considère tous ses "sur-modèles" possédant deux paramètres libres ($s=n_\theta-2$), et l'on recommence la sélection. Lorsque le critère d'un modèle est meilleur que celui de tous ses "sur-modèles", ce modèle est sélectionné. Comme pour la méthode destructive, le nombre maximal de modèles que l'on peut avoir à considérer est :

$$1 + \frac{n_\theta(n_\theta+1)}{2}$$

Ces méthodes ne mènent pas forcément au modèle "optimal" que l'on obtient lorsque les 2^{n_θ} modèles possibles sont pris en considération. Cependant, lorsque les deux méthodes conduisent au même modèle, on peut raisonnablement penser que celui-ci est le meilleur modèle de notre ensemble.

III.6. Extension des méthodes de sélection de modèles

Dans ce mémoire, nous nous sommes intéressés aux méthodes statistiques de sélection de modèles dans le cadre de la modélisation de processus à l'aide de modèles NARMAX. Cependant, ces méthodes peuvent être appliquées à des problèmes de nature différente. C'est le cas en particulier lorsque le modèle hypothèse du système étudié peut se mettre sous la forme générale suivante (extension de (III.16) au cas d'entrées multivariables) :

$$Y_p(t) = f(\mathbf{y}_p^{t-1}, \mathbf{u}^{t-1} ; \theta) + V(t) \qquad \text{(III.34)}$$

où $\mathbf{y}_p^{t-1} = \{y_p(t-1), \ldots, y_p(-\infty)\}$, $\mathbf{u}^{t-1} = \{\mathbf{u}(t-1), \ldots, \mathbf{u}(-\infty)\}$, $\mathbf{u}(t_i)$ est le vecteur des entrées du processus à l'instant t_i, et $\{V(t)\}$ est une séquence de v.a.i.i.d. dont la distribution n'est pas nécessairement gaussienne.

Les processus statiques non linéaires multi-entrées sont des cas particuliers de (III.34); en effet, les modèles hypothèses candidats sont de la forme :

$$Y_p(n) = \varphi(u_1(n), \ldots, u_{n_x}(n)) + V(n)$$

où les composantes du vecteur $\mathbf{u}(n) = [u_1(n), \ldots, u_{n_x}(n)]^T$ sont les entrées du processus, qui peuvent être des variables que l'expérimentateur fixe, ou qu'il peut simplement mesurer. La séquence des mesures $\{Y_p(n)\}$ est alors constituée de variables aléatoires indépendantes. Par exemple, $y_p(t)$ est la valeur d'une caractéristique physico-chimique d'une molécule (par exemple, le coefficient de partage eau-octanol logP), et les entrées du modèle sont des descripteurs de la molécule (nombre d'atomes de carbone ou d'hydrogène, présence d'atomes de fluor, charges sur les atomes électronégatifs, …). Les méthodes statistiques présentées dans ce chapitre peuvent alors être utilisées pour éliminer les descripteurs peu pertinents pour le calcul de cette caractéristique, et pour déterminer le nombre de neurones d'un modèle qui fournit une bonne estimation de la valeur de cette caractéristique.

Chapitre IV : Procédure de sélection de modèles NARX

IV.1. Introduction

Dans ce chapitre, nous proposons une procédure originale de sélection de modèles. Nous supposons que le processus possède une caractéristique statique dans tout le domaine de fonctionnement considéré, et que son comportement dynamique peut être décrit par un modèle NARX stable. La procédure de sélection est fondée sur les méthodes et algorithmes présentés dans les chapitres précédents. Elle se décompose en trois phases qui conduisent à la détermination des arguments de la fonction φ(.) (première et deuxième phases) et de l'architecture du modèle neuronal (troisième phase).

IV.2. Principe de la procédure de sélection de modèles NARX

On considère un processus stable dont le comportement peut être représenté par le modèle hypothèse NARX suivant :

$$y_p(t) = \varphi(y_p(t–1), \dots , y_p(t–n_y^*), u(t–1), \dots , u(t–n_u^*)) + w(t), \quad \text{(IV.1)}$$

où φ(.) est une fonction continue par morceaux, et n_y^* et n_u^* sont des caractéristiques inconnues.
La forme prédicteur théorique associée à ce modèle est :

$$y(t) = \varphi[y_p(t–1), \dots , y_p(t–n_y^*), u(t–1), \dots , u(t–n_u^*)] \quad \text{(IV.2)}$$

Les familles de modèles que nous avons choisies dans ce travail sont donc des réseaux de neurones non bouclés, complètement connectés (Figure IV.1), dont les neurones cachés ont comme fonction d'activation la fonction sigmoïde f(x) = tanh(x), et le neurone de sortie est linéaire. Le problème consiste à choisir un modèle *performant* (c'est-à-dire qui permet de construire la meilleure approximation de φ(.) dans le domaine défini par l'ensemble d'apprentissage) tout en étant le plus *parcimonieux* possible (c'est-à-dire moins complexe que tout autre modèle de cette famille dont les performances sont équivalentes). On cherche pour cela à déterminer les arguments de la fonction φ(.) (les régresseurs de $y_p(t)$), et l'architecture de ce modèle neuronal. Notons que la détermination de l'architecture d'un modèle complètement connecté consiste seulement à déterminer le nombre de ses neurones cachés.

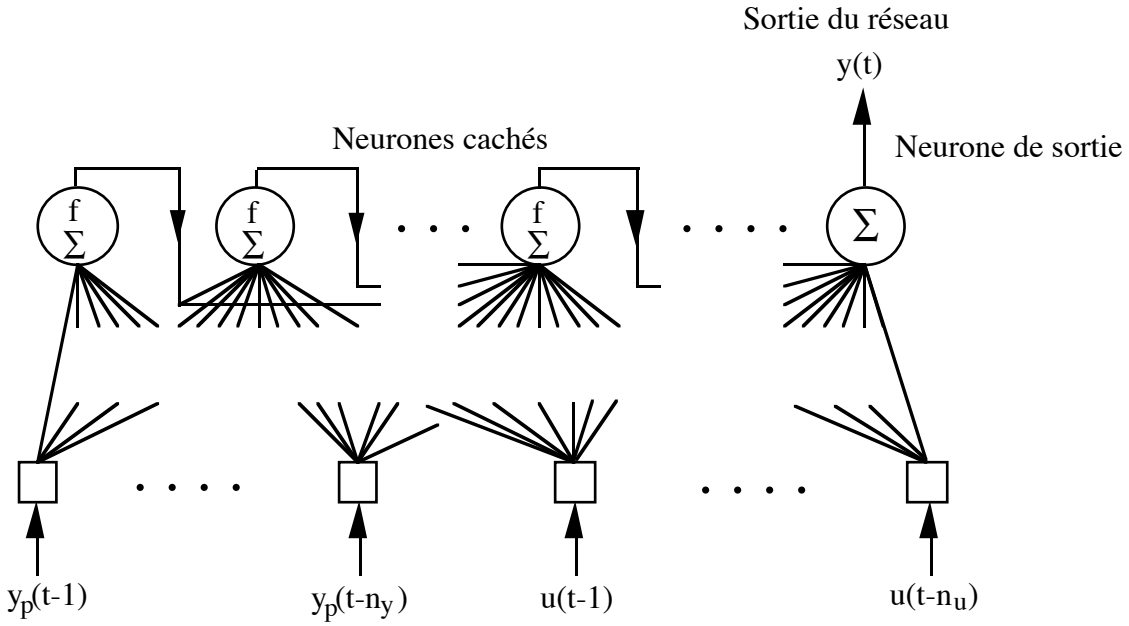


Figure IV.1 Modèle neuronal prédictif non bouclé complètement connecté

En pratique, on définit un modèle de la forme IV.2, que l'on appelle le modèle complet, et l'on sélectionne l'un de ses sous-modèles. Si l'on ne possède pas de connaissances *a priori* sur l'ordre de grandeur de n_y^* et n_u^*, on est confronté à une alternative :

– choisir un modèle complet correspondant à de grandes valeurs de n_y et n_u, mais l'ensemble de ses sous-modèles est alors généralement trop grand que l'on puisse effectivement mettre en œuvre une procédure de sélection;

– se restreindre à un modèle complet de petite taille, en choisissant de faibles valeurs de n_u et n_y; on prend alors le risque de choisir un modèle trop petit.

Pour limiter ces problèmes, nous avons choisi de décomposer la sélection de modèles en plusieurs parties. Dans une première phase, on considère le comportement du processus dans des domaines de fonctionnement restreints, autour de points de fonctionnement, et l'on modélise ces comportements à l'aide de modèles "locaux" linéaires ou polynomiaux. En effet, toute fonction non linéaire, bornée, continue par morceaux, peut être approximée, dans le voisinage d'un point pour lequel la fonction est continue, par une fonction linéaire ou polynomiale dont les arguments sont les arguments de la fonction non linéaire, ou des produits de ces arguments.

Ces modèles sont linéaires par rapport à leurs paramètres, et leur structure est entièrement définie par leurs régresseurs. On peut alors utiliser, pour l'estimation de leurs paramètres et la sélection de leurs régresseurs, des

méthodes de résolution rapides qui utilisent les propriétés de linéarité des modèles. Ceci permet de considérer des modèles ayant un grand nombre d'entrées.

Dans des domaines restreints de fonctionnement du processus, certains régresseurs ont une action importante sur le comportement du processus, alors que d'autres peuvent être négligés. Dans une zone locale particulière, il est donc possible que tous les régresseurs de $y_p(t)$ ne soient pas sélectionnés. Cependant, la procédure repose sur l'hypothèse que, si les domaines de fonctionnement locaux que l'on a choisi sont représentatifs du domaine de fonctionnement global du processus, tout régresseur significatif aura une action significative dans au moins l'un des domaines de fonctionnement local, et sera alors sélectionné. On considère donc que les régresseurs significatifs du modèle non linéaire du processus sont présents dans l'union de tous les régresseurs sélectionnés localement, et l'on utilise cette union de régresseurs pour le modèle global du processus.

Notons qu'il est nécessaire de disposer d'une séquence d'apprentissage pour chacun des domaines locaux choisis. De telles expériences, ne correspondant pas nécessairement à un fonctionnement habituel du processus, ne peuvent pas toujours être effectuées, en particulier lorsqu'il s'agit de processus industriels. Lorsque l'on ne peut pas construire ces séquences d'apprentissage, et que l'on ne dispose donc que d'observations correspondant à un fonctionnement global du processus, il est toujours possible d'effectuer la première phase de la procédure :

– en utilisant quand même un modèle linéaire, ou polynomial simple, dont on sélectionnera les régresseurs. Cependant, si le processus est fortement non linéaire, l'erreur de modélisation sera importante, et les résultats de la sélection pourront être assez médiocres;

– en utilisant un modèle polynomial de degré plus élevé. Un tel modèle permet de modéliser de façon plus exacte le comportement d'un processus non linéaire, mais le nombre de ses régresseurs devient vite prohibitif lorsque le degré considéré augmente.

Dans la suite de ce chapitre, nous définirons les domaines de fonctionnement locaux comme des voisinages autour de points de fonctionnement, et nous supposerons que la fonction $\varphi(.)$ est continue dans chacun de ces domaines.

Lorsque la première phase est effectuée, on construit un modèle non linéaire global du processus. La première phase conduisant généralement à la sélection de régresseurs non significatifs, on "valide" ceux sélectionnés

lors de la première phase en construisant un modèle neuronal, le modèle complet, dont le comportement sur l'ensemble du domaine de fonctionnement est satisfaisant. On détermine alors, dans l'ensemble de ses sous-modèles, le modèle le plus parcimonieux qui donne toujours une bonne approximation de φ(.). Nous séparerons la recherche des régresseurs de ce modèle, qui constitue l'objectif de la deuxième phase, de la détermination de son architecture, qui est effectuée lors de la troisième phase.

En pratique, la définition du modèle complet n'est pas toujours évidente : en effet, si les régresseurs sont déterminés lors de la première phase, il nous faut encore choisir le nombre de neurones.

Dans les chapitres précédents, lors de la présentation des méthodes statistiques de sélection, nous avons désigné par "modèle complet" un modèle fournissant une très bonne approximation de la partie déterministe du processus $E[y_P(t)|t-1]$. L'erreur quadratique moyenne (EQM) obtenue avec ce modèle tend vers la variance du bruit lorsque la taille de l'ensemble d'apprentissage tend vers l'infini. Si l'ensemble d'apprentissage est de très grande taille (le modèle généralise bien), l'EQM est peu différente de la variance du bruit w, qu'elle soit calculée sur l'ensemble d'apprentissage (EQMA), ou sur un ensemble de données nouvelles (EQMV), que nous appellerons *ensemble de validation*.

Par ailleurs, tout modèle contenant le modèle complet étant par définition suffisamment complexe pour construire la même approximation, il est également un modèle complet. Lorsque l'on doit choisir un modèle complet comme point de départ d'une procédure de sélection, on peut choisir le modèle le plus complexe possible.

Cependant, dans la pratique, cette démarche n'est absolument pas pertinente :

– d'une part, plus le modèle complet est complexe, plus la procédure de sélection est longue (la taille de l'ensemble des sous-modèles d'un modèle croît de façon exponentielle avec le nombre de paramètres);

– d'autre part, la taille de l'ensemble d'apprentissage est finie, et peut être assez limitée. Or, l'estimation des paramètres d'un modèle consiste à ajuster ces paramètres pour minimiser l'erreur quadratique moyenne entre les sorties observées et les prédictions calculées avec le modèle, ceci pour l'ensemble d'observations particulier qu'est l'ensemble d'apprentissage. Les prédictions de la sortie du processus pour des observations qui

n'appartiennent pas à l'ensemble d'apprentissage sont interpolés par le modèle.

Lorsque la complexité du modèle est trop importante par rapport à la taille de l'ensemble d'apprentissage, les estimations des sorties du processus calculées avec le modèle sont très proches des valeurs exactes de ses sorties pour les données de l'ensemble d'apprentissage, mais l'interpolation de la fonction φ(.) en dehors de ces points particuliers est généralement mauvaise, et un modèle plus simple suffirait à construire une meilleure approximation de φ(.). On dit alors qu'il y a *surajustement*. Dans un tel cas, l'EQM obtenue sur l'ensemble d'apprentissage (EQMA) est inférieure à la variance du bruit, mais l'EQM obtenue sur l'ensemble de validation (EQMV) est significativement plus élevée.

Un exemple est présenté sur les figures IV.2. et IV.3. : le processus est simulé par l'équation $y_p(t) = \sin(x(t))/x(t) + w(t)$, où w(t) est un bruit pseudo-blanc; on effectue l'apprentissage de deux réseaux de neurones à couches, le premier possédant 4 neurones cachés (soit 13 paramètres), le second 8 neurones cachés (25 paramètres); l'ensemble d'apprentissage est constitué de 20 points. Sur chacune des deux figures sont représentés : [i] un ensemble de points constitué de 1000 réalisations de $y_p(t)$, où x(t) prend ses valeurs dans [0, 15] (courbe gris clair), [ii] les 20 points de l'ensemble d'apprentissage (points noirs), [iii] la fonction réalisée par le modèle neuronal après apprentissage (courbe noire). La figure IV.2. correspond au modèle à 4 neurones cachés, la figure IV.3. au modèle à 8 neurones cachés. Pour le modèle à 8 neurones cachés, qui est trop complexe, on observe un phénomène de surajustement très important, alors que le modèle à 4 neurones cachés fournie une approximation de φ(.) très satisfaisante.

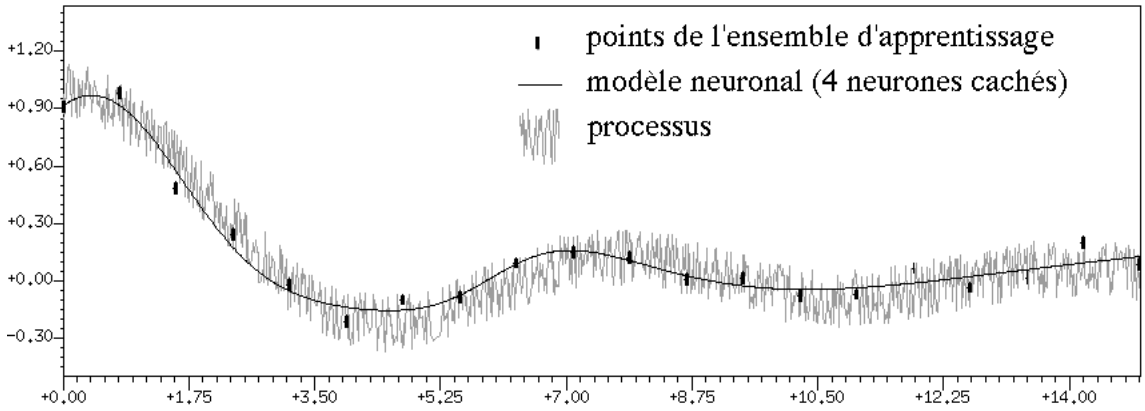


Figure IV.2.

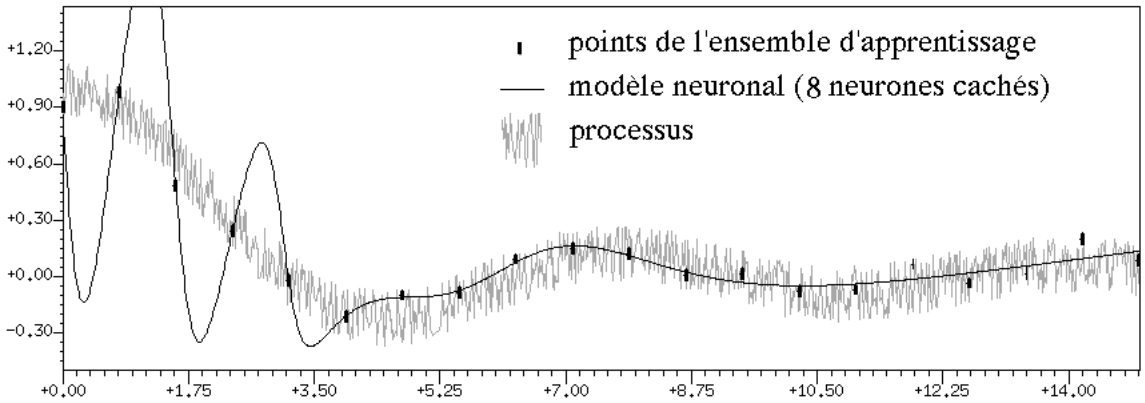


Figure IV.3.

Une démarche pragmatique pour choisir le modèle complet consiste donc à chercher le plus petit modèle avec lequel on observe du surajustement. Pour cela, on considère un modèle simple dont les régresseurs sont ceux sélectionnés lors de la première phase, et l'on compare l'EQMA et l'EQMV obtenues avec ce modèle. Si l'EQMV est beaucoup plus importante que l'EQMA, il est inutile de construire un modèle plus complexe. Si ces valeurs sont comparables, on considère un modèle plus complexe, et l'on compare à nouveau l'EQMA et l'EQMV. On procède de la sorte jusqu'à obtenir un modèle avec lequel on observe un surajustement, que l'on choisit comme modèle complet.

La deuxième phase consiste alors à sélectionner, à l'aide d'une méthode de sélection statistique, l'un de ses sous-modèles parmi ceux qui possèdent le même nombre de neurones, et dont les régresseurs sont constituées d'un sous-ensemble de l'ensemble des régresseurs du modèle complet.

Remarquons que l'on suppose implicitement que tous les régresseurs significatifs ont été sélectionnés lors de la première phase, c'est-à-dire qu'il n'existe pas de régresseurs dont l'action n'est significative que pour des signaux de grande amplitude, et négligeable dans les domaines de fonctionnements locaux que nous avons choisis. D'autre part, le modèle neuronal doit représenter le comportement non linéaire du processus dans un domaine de fonctionnement défini par l'utilisation future. Il est donc bien sûr nécessaire de disposer de séquences d'apprentissage explorant complètement ce domaine.

Le nombre de neurones du modèle est sélectionné lors de la troisième phase. Si un ou plusieurs régresseurs ont été éliminés lors de la deuxième phase, le nombre de paramètres du modèle peut maintenant être insuffisant. Comme au début de la deuxième phase, on augmente donc, si nécessaire, le nombre de neurones jusqu'à obtenir un modèle suffisamment complexe, qui

est choisi comme modèle complet. On effectue alors une sélection sur l'ensemble de ses sous-modèles possédant les mêmes régresseurs et un nombre de neurones inférieur. On utilise généralement lors de cette troisième phase les mêmes séquences d'apprentissage et de validation que lors de la deuxième.

Nous décrivons en détail les trois phases de la procédure dans les paragraphes IV.3, IV.4 et IV.5. Le modèle sélectionné par la procédure est le modèle obtenu à la fin de la troisième phase. Nous allons maintenant détailler cette procédure, et présenter les algorithmes et les méthodes qui permettent de mener à bien la sélection d'un modèle NARX.

IV.3. Première phase : sélection de modèles linéaires locaux.

IV.3.1. Linéarisation d'un modèle NARX

IV.3.1.1. Linéarisation d'un modèle déterministe

On cherche à modéliser un processus non linéaire, stable, que l'on suppose pour l'instant déterministe, dont le comportement dynamique est décrit par le modèle hypothèse suivant :

$$y_p(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u)) = \varphi(\mathbf{x}(t)) \qquad \text{(IV.3)}$$

avec :

$$\mathbf{x}(t) = [y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u)]^T$$

Pour une commande de valeur constante $u(t) = u_o$, le processus converge vers un régime stationnaire, correspondant à une sortie y_o.

$$y_p(t) = y_o = \varphi(y_o, \ldots, y_o, u_o, \ldots, u_o) = \varphi(\mathbf{x}_o) \qquad \text{(IV.4)}$$

Écrivons le développement limité de $\varphi(.)$ au voisinage de $\mathbf{x}_o$:

$$y_p(t) = y_o + (\mathbf{x}–\mathbf{x}_o)^T\varphi'(\mathbf{x}_o) + O(\mathbf{x}–\mathbf{x}_o)^2 \qquad \text{(IV.5)}$$

Pour des valeurs de $\mathbf{x}$ proches de $\mathbf{x}_o$, on peut faire l'approximation :

$$y_p(t) \approx y_o + (\mathbf{x}–\mathbf{x}_o)\varphi'(\mathbf{x}_o) = \mathbf{x}\,\varphi'(\mathbf{x}_o) + [y_o – \mathbf{x}_o\,\varphi'(\mathbf{x}_o)] \qquad \text{(IV.6)}$$

Dans un voisinage restreint de $\mathbf{x}_o$, correspondant à des valeurs moyennes u_o de la commande et y_o de la sortie, le modèle hypothèse peut donc être approché par un modèle affine. Autour du point (x_o, y_o), on obtient le modèle linéaire suivant :

$$\overline{y}_p(t) = \sum_{i=1}^{n_y} c_i\,\overline{y}_p(t–i) + \sum_{j=1}^{n_u} c_{n_y+j}\,\overline{u}(t–j) \qquad \text{(IV.7)}$$

où :

- $\bar{y}_p(t) = y_p(t) - y_o$, $\bar{u}(t) = u(t) - u_o$,
- c_i, i={1, …, n_y} est la dérivée partielle de φ(.) par rapport à $y_p(t–i)$ en $\mathbf{x}_o$,
- c_{n_y+j}, j={1, …, n_u} est la dérivée partielle de φ(.) par rapport à u(t–j) en $\mathbf{x}_o$.

IV.3.1.2. Linéarisation d'un modèle NARX

Considérons maintenant le modèle hypothèse NARX suivant :

$$y_p(t) = \varphi(y_p(t–1), \ldots, y_p(t–n_y), u(t–1), \ldots, u(t–n_u)) + w(t) \qquad \text{(IV.8)}$$

Supposons que ce modèle hypothèse est stable. On applique une commande constante u_o : l'espérance mathématique de $y_p(t)$ converge alors vers une valeur y_o, et le vecteur des régresseurs du modèle est :

$$\mathbf{x}(t) = [y_p(t–1), \ldots, y_p(t–n_y), u_o, \ldots, u_o].$$

Lorsque le bruit est d'amplitude suffisamment faible, le modèle hypothèse NARX avec commande constante peut être approximé autour de $\mathbf{x}_o$ par un modèle AR :

$$\bar{y}_p(t) = \sum_{i=1}^{n_y} c_i \bar{y}_p(t–i) + w(t) \qquad \text{(IV.9)}$$

Si l'on superpose à la commande constante u_o une séquence Δu(t) d'amplitude suffisamment faible (on a alors $\bar{u}(t) = \Delta u(t)$), le modèle NARX peut alors être "localement" approximé par un modèle ARX de la forme :

$$\bar{y}_p(t) = \sum_{i=1}^{n_y} c_i \bar{y}_p(t–i) + \sum_{j=1}^{n_u} c_{n_y+j} \Delta u(t–j) + w(t) \qquad \text{(IV.10)}$$

En pratique, il est parfois impossible de se placer dans les conditions où le processus peut être approché par un modèle ARX : lorsque les amplitudes du bruit w(t) ou de la séquence Δu(t) superposée à la commande sont trop importantes, le comportement du processus devient très non linéaire, et l'hypothèse de linéarité n'est plus valable. Or, si l'on peut toujours choisir les valeurs de Δu(t), on ne peut influer sur w(t). Cependant, on peut toujours utiliser un modèle non linéaire polynomial : la procédure de sélection que nous présentons dans le paragraphe qui suit concerne des modèles linéaires par rapport aux paramètres, nous pourrons donc l'appliquer aussi bien à des modèles linéaires qu'à des modèles polynomiaux par rapport aux *entrées primaires* (c'est-à-dire les régresseurs de type $y_p(t–i)$ ou u(t–j)).

IV.3.2. Procédure de sélection des régresseurs d'un modèle linéaire par rapport à ses paramètres

IV.3.2.1. Principe de la procédure

Nous nous plaçons ici dans le cadre de la modélisation d'un processus à l'aide d'un modèle linéaire par rapport à ses paramètres. Nous disposons d'un modèle, possédant n_θ régresseurs $\{x_1(t), \ldots, x_{n_\theta}(t)\}$, et nous voulons effectuer une sélection sur l'ensemble de ses sous-modèles. Dans ce paragraphe, nous appellerons, par abus de langage, le modèle à n_θ paramètres "modèle complet".

La sélection nécessite que, pour chaque modèle considéré, on effectue une estimation des coefficients (avec la méthode des moindres carrés par exemple), ainsi que le calcul de l'erreur quadratique moyenne obtenue sur l'ensemble d'apprentissage. Nous l'avons déjà dit, ceci devient irréalisable dès que le nombre de régresseurs du modèle complet est élevé, même pour des modèles linéaires. Nous avons présenté au chapitre précédent plusieurs méthodes permettant de réduire de façon très sensible le nombre d'apprentissages et de tests nécessaires pour sélectionner un modèle, mais ces méthodes ne tiennent pas compte du caractère linéaire des modèles que nous considérons.

La méthode que nous présentons met à profit les propriétés des modèles linéaires et utilise des méthodes spécifiques de ceux-ci. C'est une méthode de type "destructive", inspirée de travaux de Chen [Chen 89a], qui permet de réduire le nombre maximum de tests que l'on doit effectuer à $(n_\theta - 1)$.

On effectue dans un premier temps un classement des régresseurs du modèle complet. Pour cela, on procède de la manière suivante :

– à l'itération 1, on considère les n_θ modèles possédant un seul des n_θ régresseurs $\{x_1(t), \ldots, x_{n_\theta}(t)\}$ du modèle complet :

$$M^{1,j} \quad : \qquad y^{1,j}(t) = \theta^{1,j}\, x_j(t), \qquad\qquad j=\{1, \ldots, n_\theta\}.$$

On estime les valeurs des paramètres $\theta^{1,j}$ à l'aide de la méthode des moindres-carrés, puis l'on détermine le modèle M^{1,k_1} qui réalise la meilleure prédiction $y^{1,k_1}(t)$ de $y_p(t)$ sur l'ensemble d'apprentissage (c'est-à-dire celui avec lequel l'EQMA est la plus faible), et l'on classe en première position le régresseur unique $x_{k_1}(t)$ de ce modèle.

– à l'itération s, on a déjà classé (s–1) régresseurs $\{x_{k_1}(t), \ldots, x_{k_{s-1}}(t)\}$; on considère alors les n_θ–(s–1) modèles dont les régresseurs sont constitués des (s–1) régresseurs déjà choisis et de l'un des n_θ–(s–1) restants :

$$M^{s,j} \quad : \qquad y^{s,j}(t) = \sum_{k=1}^{s-1} \left[\theta_k{}^{s,j}\, x_k(t)\right] + \theta_s{}^{s,j}\, x_j(t), \qquad j=\{1, \ldots, n_\theta\},\ \ j \neq \{k_1, \ldots, k_{s-1}\}$$

on détermine le modèle $M^{s,\,k_s}$ qui réalise la meilleure prédiction $y^{s,\,k_s}(t)$ de $y_p(t)$ sur l'ensemble d'apprentissage, et l'on classe en $s^{ième}$ position le régresseur $x_{k_s}(t)$. La procédure utilisée lors de cette étape est décrite dans le paragraphe IV.3.2.2.

Une fois le classement des n_θ régresseurs effectué, on construit un ensemble ordonné $\{M_1, \ldots, M_{n_\theta}\}$ de n_θ modèles : le modèle M_s est le modèle de taille s, sous-modèle du modèle complet, ayant comme régresseurs les s premiers régresseurs classés. Le modèle complet est donc M_{n_θ}.

La sélection s'effectue sur cet ensemble restreint et ordonné de modèles. Pour cela, on compare, à l'aide d'un test d'hypothèse, le modèle $M_{n_\theta-1}$ à M_{n_θ} :

– si le test conduit au rejet du modèle $M_{n_\theta-1}$, la procédure s'arrête et l'on conserve donc tous les régresseurs du modèle complet. En effet, si le test rejette l'hypothèse selon laquelle le dernier régresseur (c'est-à-dire celui dont la contribution à l'EQMA est la plus faible) peut être supprimé, cela signifie qu'il est nécessaire pour la description du processus, ainsi que tous les régresseurs qui le précèdent dans la liste;

– si le test ne conduit pas à rejeter cette hypothèse, on élimine le dernier régresseur de la liste, puis l'on compare le modèle $M_{n_\theta-2}$ à M_{n_θ}, et ainsi de suite (méthode MCU) (Figure IV.4.a).

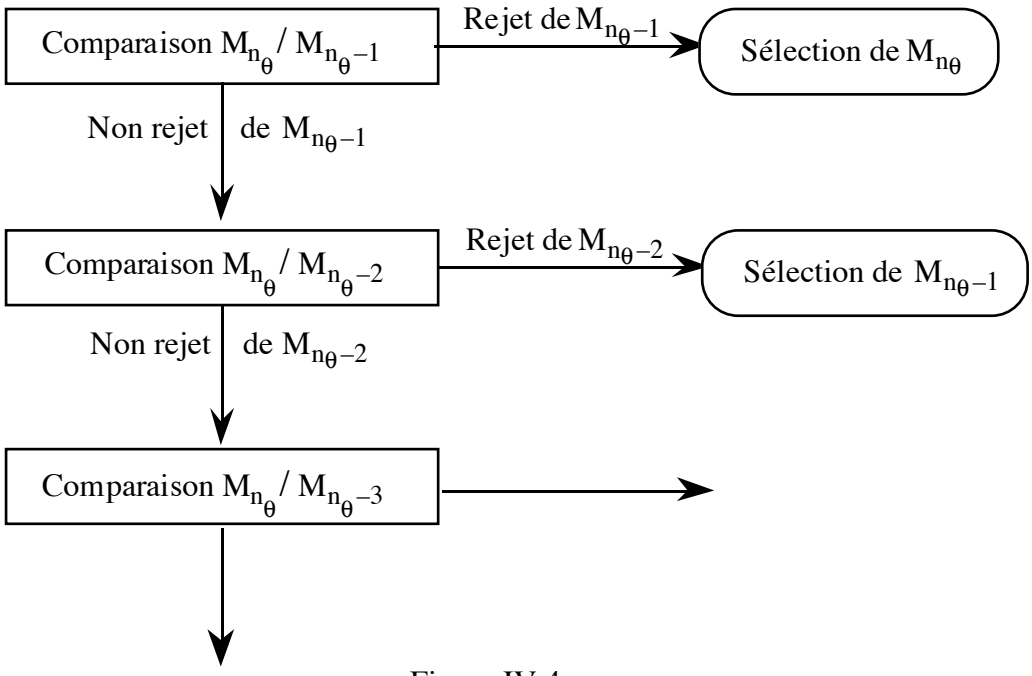


Figure IV.4.a

Nous utiliserons également la méthode MCM, qui consiste à choisir le modèle M_k comme modèle complet dès lors qu'il n'est pas rejeté par le test, et donc de comparer le modèle suivant, M_{k-1}, non pas avec M_{n_θ}, mais avec M_k. On suppose

alors que, lorsqu'un modèle n'est pas rejeté par le test, il peut être choisi comme modèle complet pour la suite (Figure IV.4.b).

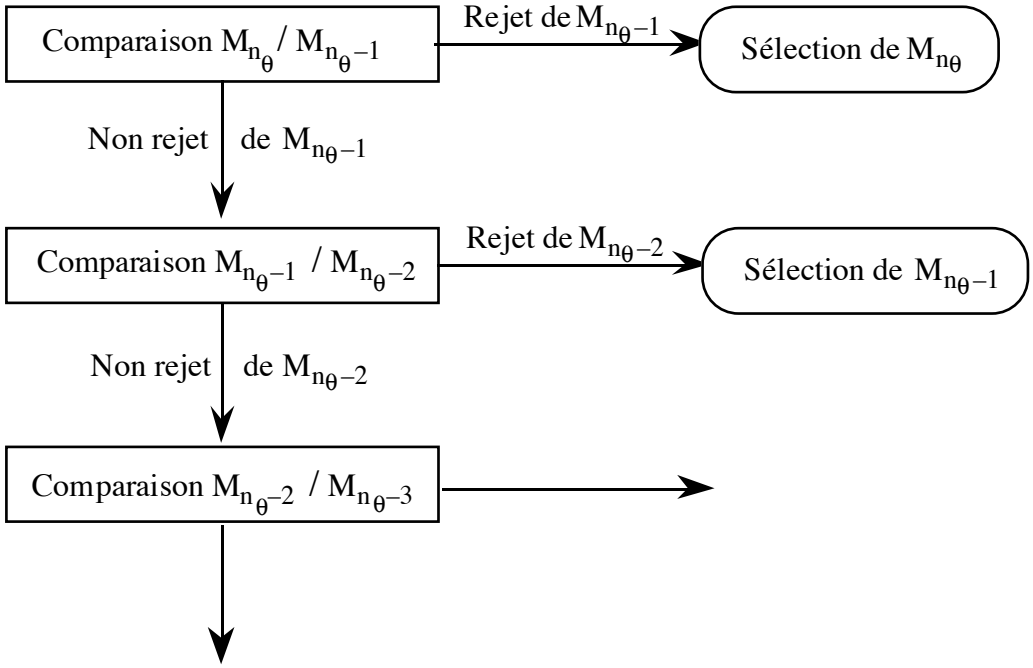


Figure IV.4.b

IV.3.2.2. Classement des régresseurs à l'aide d'une méthode d'orthogonalisation

L'algorithme de classement que nous présentons a été proposé dans différents articles, en particulier par Korenberg [Korenberg 85] et Billings [Billings 88], [Billings 89] sous le nom d'OFR (Orthogonal Forward Regression), et repose sur l'utilisation de la méthode d'orthogonalisation de Gram-Schimdt. Pour mieux comprendre la procédure de classement des régresseurs, nous allons donner une interprétation géométrique du problème : l'ensemble d'apprentissage que l'on considère est de taille N finie, les vecteurs $\{\mathbf{x}_1, \ldots, \mathbf{x}_{n_\theta}\}$ associés aux régresseurs $\{x_1(t), \ldots, x_{n_\theta}(t)\}$, ainsi que le vecteur $\mathbf{y}_p$ des sorties, sont donc des vecteurs d'un espace E^N de dimension N. Dans cet espace, ces n_θ vecteurs engendrent un sous espace vectoriel $E^{n\ \theta}$ de dimension n_θ (s'ils sont linéairement indépendants, ce qui est quasiment toujours vrai lorsque $N>>n_\theta$). Le modèle complet est supposé suffisant pour décrire les données : cela signifie ici que le vecteur des prédictions calculé avec le modèle complet (et correspondant à la projection de $\mathbf{y}_p$ sur $E^{n\ \theta}$) est une prédiction satisfaisante de $\mathbf{y}_p$. La sélection d'un sous modèle du modèle complet consiste alors à rechercher le plus petit sous espace E^k de $E^{n\ \theta}$ tel que la projection de $\mathbf{y}_p$ sur E^k soit une prédiction satisfaisante de $\mathbf{y}_p$.

Soient EQM_1 et EQM_2 les EQM obtenues respectivement avec un modèle M_1, dont $x_k(t)$ n'est pas un régresseur, et un modèle M_2 dont les régresseurs sont les régresseurs de M_1 et $x_k(t)$; nous appellerons dans ce paragraphe *contribution* du régresseur $x_k(t)$ à la prédiction de $y_p(t)$, pour un modèle M_1 donné, la valeur positive $EQM_1 - EQM_2$. Si l'on note $\mathbf{y}$ la prédiction de $\mathbf{y}_p$ obtenue avec M_1, la contribution de $x_k(t)$ s'exprime à partir du carré du cosinus de l'angle défini par $\mathbf{y}$ et $\mathbf{x}_k$.

A la première itération de la procédure de classement, aucun régresseur n'a été choisi, la seule prédiction de $\mathbf{y}_p$ dont on dispose est donc $\mathbf{y}=0$. Le premier vecteur $\mathbf{x}_{k_1}$ choisi est celui dont la contribution est maximale, c'est-à-dire tel que :

$$\cos^2(\mathbf{x}_{k_1}, \mathbf{y}_p) = \max_k [\cos^2(\mathbf{x}_k, \mathbf{y}_p)], \qquad k=1, \ldots, n_\theta$$

On note $\mathbf{p}_1 = \mathbf{x}_{k_1}$ ce vecteur, qui engendre le sous espace $E^{(1)}$ de dimension 1. Le premier régresseur est donc $x_{k_1}(t)$. $\mathbf{y}_p$ est alors exprimé comme la somme de deux vecteurs orthogonaux : la projection de $\mathbf{y}_p$ sur $E^{(1)}$ (notée $\mathbf{y}_k$ sur la figure IV.5), et $\mathbf{y}_p^2 = \mathbf{y}_p^1 - \mathbf{y}_k$. Le vecteur $\mathbf{y}_k$ est la partie de $\mathbf{y}_p$ "expliquée" par $\mathbf{x}_{k_1}$. On décompose de la même manière les $n_\theta - 1$ vecteurs $\{\mathbf{x}_1, \ldots, \mathbf{x}_{k_1-1}, \mathbf{x}_{k_1+1}, \ldots, \mathbf{x}_{n_\theta}\}$.

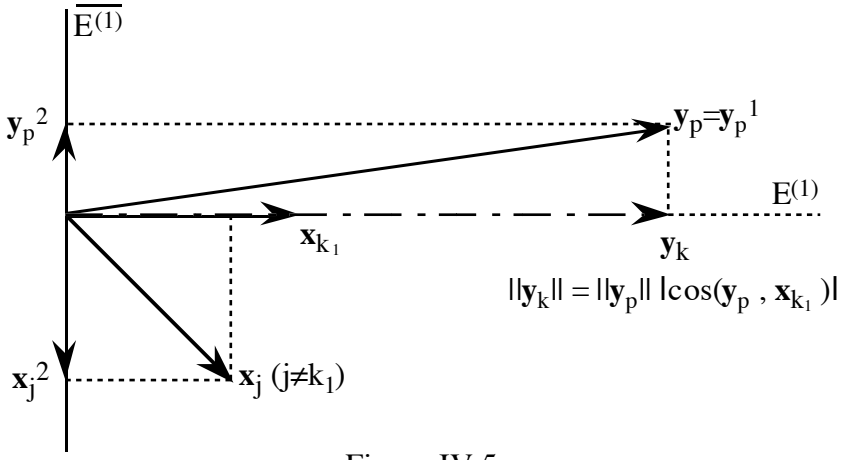


Figure IV.5.

Notons donc $\mathbf{y}_p = \mathbf{y}_p^1$, $\mathbf{y}_p^2$ la projection de $\mathbf{y}_p$ sur le sous espace de E^{n_θ} orthogonal à $E^{(1)}$, et $\{\mathbf{x}_1^2, \ldots, \mathbf{x}_{k_1-1}^2, \mathbf{x}_{k_1+1}^2, \ldots, \mathbf{x}_{n_\theta}^2\}$ les projections des vecteurs $\{\mathbf{x}_1, \ldots, \mathbf{x}_{k_1-1}, \mathbf{x}_{k_1+1}, \ldots, \mathbf{x}_{n_\theta}\}$ sur ce même sous-espace. On détermine, à nouveau, parmi ces vecteurs, le vecteur $\mathbf{p}_2 = \mathbf{x}_{k_2}^2$ qui maximise $\cos^2(\mathbf{x}_k, \mathbf{y}_p)$, et le régresseur correspondant $x_{k_2}(t)$. On procède une fois encore à l'orthogonalisation des vecteurs restants par rapport au sous espace vectoriel $E^2 = E^{(1)} \times E^{(2)}$ engendré par $\{\mathbf{p}_1, \mathbf{p}_2\}$, et l'on continue ainsi jusqu'à ce que tous les vecteurs soient classés.

A la fin du classement, on obtient alors une nouvelle base $[\mathbf{p}_1, \ldots, \mathbf{p}_{n_\theta}]$, orthogonale, et engendrant le même sous espace E^{n_θ} que $[\mathbf{x}_1, \ldots, \mathbf{x}_{n_\theta}]$.

Si les n_θ vecteurs sont liés, ils engendrent un espace E^M de dimension $M<n_\theta$. Les M premiers vecteurs de la liste sont obtenus de la même manière que précédemment, et forment une base de E^M. A l'itération M+1, les $(n_\theta - M)$ vecteurs restants, appartenant à E^M, sont projetés orthogonalement à E^M; leurs projections étant nulles, on peut alors supprimer les régresseurs correspondants.

IV.3.2.3. Description de l'algorithme

Itération k=1

Le régresseur classé en première position correspond au vecteur qui forme l'angle le plus petit avec $\mathbf{y}_p$. La contribution d'un vecteur $\mathbf{x}_i$ à "l'explication" du vecteur $\mathbf{y}_p$ est mesurée par :

$$\zeta_i = \frac{\|\mathbf{x}_i^T \mathbf{y}_p\|^2}{\|\mathbf{x}_i\|^2} = \left(\frac{\mathbf{x}_i^T \mathbf{y}_p}{\mathbf{x}_i^T \mathbf{x}_i}\right)^2 \mathbf{x}_i^T \mathbf{x}_i = \cos^2(\mathbf{x}_i, \mathbf{y}_p)\|\mathbf{y}_p\|^2 \qquad \text{(IV.11)}$$

où $\left(\frac{\mathbf{x}_i^T \mathbf{y}_p}{\mathbf{x}_i^T \mathbf{x}_i}\right)$ est le coefficient de projection de $\mathbf{y}_p$ sur le sous espace $E^{(i)}$, engendré par $\mathbf{x}_i$. Pour faciliter la présentation de la procédure, on note :

$$\mathbf{y}_p = \mathbf{y}_p^1 , \qquad [\mathbf{x}_1, \ldots, \mathbf{x}_{n_\theta}] = [\mathbf{x}_1^1, \ldots, \mathbf{x}_{n_\theta}^1]$$

On calcule donc les coefficients $\beta_1^1, \ldots, \beta_{n_\theta}^1$ de projection de $\mathbf{y}_p$ sur les n_θ vecteurs $\mathbf{x}_1^1, \ldots, \mathbf{x}_{n_\theta}^1$:

$$\beta_1^1 = \frac{(\mathbf{x}_1^1)^T \mathbf{y}_p^1}{(\mathbf{x}_1^1)^T(\mathbf{x}_1^1)} , \ \ldots, \quad \beta_{n_\theta}^1 = \frac{(\mathbf{x}_{n_\theta}^1)^T \mathbf{y}_p^1}{(\mathbf{x}_{n_\theta}^1)^T(\mathbf{x}_{n_\theta}^1)} \qquad \text{(IV.12)}$$

et l'on en déduit les contributions respectives de chacun de ces vecteurs :

$$\zeta_1^1 = (\beta_1^1)^2 (\mathbf{x}_1^1)^T(\mathbf{x}_1^1) , \ \ldots, \quad \zeta_{n_\theta}^1 = (\beta_{n_\theta}^1)^2 (\mathbf{x}_{n_\theta}^1)^T(\mathbf{x}_{n_\theta}^1) \qquad \text{(IV.13)}$$

Le vecteur $\mathbf{p}_1 = \mathbf{x}_{k_1}^1$ classé en premier est tel que :

$$\zeta_{k_1}^1 = \sup \{ \zeta_i^1, i=1, \ldots, n_\theta \} \qquad \text{(IV.14)}$$

Le premier régresseur de la liste est alors $x_{k_1}(t)$, correspondant au vecteur $\mathbf{p}_1 = \mathbf{x}_{k_1}^1$, et l'on note $\beta_1 = \beta_{k_1}^1$, et $\zeta_1 = \zeta_{k_1}^1$. On orthogonalise alors le vecteur des sorties $\mathbf{y}_p^1 = \mathbf{y}_p$, ainsi que tous les vecteurs non encore classés, par rapport à $\mathbf{p}_1$. L'orthogonalisation s'effectue de la façon suivante :

$$\mathbf{y}_p^2 = \mathbf{y}_p^1 - \beta^1 \mathbf{p}_1 , \qquad \text{(IV.15.a)}$$

et

$$\mathbf{x}_i^2 = \mathbf{x}_i^1 - \alpha_i^1 \mathbf{p}_1 , \qquad \text{avec : } \alpha_i^1 = \frac{(\mathbf{p}_1)^T \mathbf{x}_i^1}{(\mathbf{p}_1)^T(\mathbf{p}_1)} \qquad \text{(IV.15.b)}$$

Les α_i^1 (i=1, …, n_θ et i≠k_1) sont les coefficients de projection des vecteurs $\mathbf{x}_1$, .., $\mathbf{x}_{n_\theta}$ sur $\mathbf{p}_1$. Ces coefficients définissent une matrice triangulaire supérieure **A** telle que $[\mathbf{x}_{k1}, \ldots, \mathbf{x}_{kn\theta}] = [\mathbf{p}_1, \ldots, \mathbf{p}_{kn\theta}]\mathbf{A}$. L'ordonnancement de la nouvelle base n'étant pas encore connu (on ne connaît pour l'instant que le premier vecteur $\mathbf{p}_1$), les valeurs des coefficients α_i^1 sont stockées, et seront utilisées pour construire **A** au cours des itérations suivantes.

Itération s

Les s–1 vecteurs $\{\mathbf{p}_1, \ldots, \mathbf{p}_{ks-1}\} = \{\mathbf{x}_{k1}{}^1, \ldots, \mathbf{x}_{ks-1}{}^{s-1}\}$, ont été classés, ainsi que les s–1 régresseurs correspondants $\{x_{k1}(t), \ldots, x_{ks-1}(t)\}$. Les vecteurs restants sont les vecteurs $\{\mathbf{x}_i{}^k ; i \neq \{k_1, \ldots, k_{s-1}\}\}$, orthogonaux aux vecteurs $\{\mathbf{p}_1, \ldots, \mathbf{p}_{S-1}\}$, et $\mathbf{y}_p{}^s$ est la partie de $\mathbf{y}_p$ non encore "expliquée".

On calcule les coefficients $(\beta_i^s \mid i=1, \ldots, n_\theta, i \notin \{k_1, \ldots, k_{s-1}\})$ de projection de $\mathbf{y}_p$ sur les vecteurs $\mathbf{x}_i^s$ correspondants :

$$\beta_1{}^s = \frac{(\mathbf{x}_1{}^s)^T \mathbf{y}_p{}^s}{(\mathbf{x}_1{}^s)^T(\mathbf{x}_1{}^s)} , \quad \ldots, \quad \beta_{n_\theta}{}^s = \frac{(\mathbf{x}_{n_\theta}{}^s)^T \mathbf{y}_p{}^s}{(\mathbf{x}_{n_\theta}{}^s)^T(\mathbf{x}_{n_\theta}{}^s)} \qquad \text{(IV.16)}$$

et l'on en déduit les contributions respectives de chacun de ces vecteurs :

$$\zeta_1{}^s = (\beta_1{}^s)^2 (\mathbf{x}_1{}^s)^T(\mathbf{x}_1{}^s) , \quad \ldots, \quad \zeta_{n_\theta}{}^s = (\beta_{n_\theta}{}^s)^2 (\mathbf{x}_{n_\theta}{}^s)^T(\mathbf{x}_{n_\theta}{}^s) \qquad \text{(IV.17)}$$

Le vecteur $\mathbf{p}_s = \mathbf{x}_{ks}{}^s$ classé en position s est tel que :

$$\zeta_{ks}{}^s = \sup \{ \zeta_i{}^s, i=1, \ldots, n_\theta, i \notin \{k_1, \ldots, k_{s-1}\}\} \qquad \text{(IV.18)}$$

Le régresseur correspondant est $x_{ks}(t)$. On note $\beta_s = \beta_{ks}{}^s$, et $\zeta_s = \zeta_{ks}{}^s$. Les valeurs des coefficients $\alpha_{1,s}$, …, $\alpha_{s-1,s}$ de la matrice A sont égales aux valeurs $\alpha_{ks}{}^1$, …, $\alpha_{ks}{}^{s-1}$ calculées au cours des itérations précédentes.

On orthogonalise alors les vecteurs $\mathbf{y}_p{}^s$ et $\mathbf{x}_1{}^s$, …, $\mathbf{x}_{n_\theta}{}^s$ par rapport à $\mathbf{p}_s$, et l'on poursuit la procédure jusqu'à ce que tous les régresseurs soient classés. On obtient finalement les relations :

$$\mathbf{y}_p = \sum_{i=1}^{n_\theta} \beta_i \, \mathbf{p}_i + \mathbf{e}, \qquad \text{(IV.19)}$$

$$\|\mathbf{y}_p\|^2 = \sum_{i=1}^{n_\theta} \beta_i^2 \ \|\mathbf{p}_i\|^2 + \|\mathbf{e}\|^2 = \sum_{i=1}^{n_\theta} \zeta_i + \|\mathbf{e}\|^2 \qquad \text{(IV.20)}$$

où $\mathbf{e}$ et $\frac{\|\mathbf{e}\|^2}{N}$ sont respectivement le vecteur des erreurs de prédiction et l'EQMA obtenus avec le modèle à n_θ paramètres.

IV.3.2.4. Calcul de l'erreur quadratique moyenne

Cette procédure permet de calculer facilement la solution des moindres carrés ordinaires pour chacun des modèles de la liste que l'on a construite, cela au fur et à mesure de cette construction. Considérons le modèle hypothèse linéaire complet :

$$\mathbf{y}_p = \mathbf{x}\theta + \mathbf{w} \qquad \text{(IV.21)}$$

On peut maintenant également l'exprimer sous la forme :

$$\mathbf{y}_p = \mathbf{p}\beta + \mathbf{w} \qquad \text{(IV.22)}$$

D'où la relation :

$$\mathbf{x}\,\theta = \mathbf{p}\ A\theta = \mathbf{p}\,\beta \quad \Rightarrow \quad A\theta = \beta \qquad \text{(IV.23)}$$

soit :

$$\begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n_\theta} \end{bmatrix} = \begin{bmatrix} 1 & \alpha_{1,2} & \alpha_{1,3} & \dots & \alpha_{1,\,n_\theta} \\ 0 & 1 & \alpha_{2,3} & \dots & \alpha_{2,\,n_\theta} \\ & & 1 & & \\ & & & 1 & \alpha_{n_\theta-1,\,n_\theta} \\ 0 & & & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_\theta} \end{bmatrix}$$

A étant triangulaire supérieure, la résolution de cette équation en θ est immédiate, et fournit la solution des moindres carrés ordinaires pour le modèle complet. A l'itération s, on considère le modèle linéaire de dimension s ayant comme régresseurs $[x_{k1}(t), \dots, x_{ks}(t)]$. On dispose :

- des bases $\mathbf{p}_s = [\mathbf{p}_{k1}, \dots, \mathbf{p}_{ks}]$ et $\mathbf{x}_s = [\mathbf{x}_{k1}, \dots, \mathbf{x}_{ks}]$ engendrant le sous-espace E^s,
- de la projection $\beta_s = [\beta_1, \dots, \beta_s]$ de $\mathbf{y}_p$ sur E^s,
- de la matrice $\mathbf{A}_s$, correspondant à la sous-matrice de dimension (s,s) de A,

qui vérifient les relations :

$$\mathbf{y}_p = \mathbf{x}_s\,\theta_s + \mathbf{e}_s = \mathbf{p}_s\,\beta_s + \mathbf{e}_s \text{ et } \qquad \mathbf{x}_s = \mathbf{A}_s\,\mathbf{p}_s \qquad \text{(IV.24.a, b)}$$

L'estimation des moindres carrés des paramètres θ_s, ainsi que la valeur de l'EQMA, est alors obtenue de façon simple par la résolution des équations :

$$\beta_s = A_s\,\theta_s \qquad \text{et} \qquad \|\mathbf{y}_p\|^2 = \sum_{i=1}^{s} \beta_i^2\,\|\mathbf{p}_i\|^2 + \|\mathbf{e}_s\|^2 = \sum_{i=1}^{s} \zeta_i + \|\mathbf{e}_s\|^2 \qquad \text{(IV.25.a, b)}$$

IV.3.2.5. Sélection d'un modèle linéaire local

Pour sélectionner un modèle, nous utilisons le test LDRT présenté au chapitre III (il est aussi possible d'utiliser une méthodes de type Akaike). Cependant, nous allons introduire une modification à la procédure présentée dans le paragraphe IV.3.2.1. En effet, nous avons alors supposé que la sélection s'effectuait en choisissant comme modèle complet le modèle à n_θ paramètres. Or, lorsqu'elle est de très grande taille, la matrice $\mathbf{x}$ est souvent mal conditionnée. Nous proposons donc une heuristique pour éliminer, avant la sélection proprement dite, un grand nombre de régresseurs candidats, et "présélectionner" un modèle de petite taille. D'un point de vue géométrique, le sous-espace correspondant à ce modèle est de faible dimension, et la base des vecteurs qui l'engendre n'est composée que des vecteurs de régresseurs significatifs. La matrice est alors généralement bien conditionnée.

Nous utilisons l'heuristique suivante : le premier modèle (un seul régresseur) est comparé au deuxième (avec deux régresseurs), dont il est un sous-modèle; on calcule pour cela la valeur du rapport correspondant à un test LDRT (III.20). Si cette valeur est inférieure à un seuil fixé, correspondant à une probabilité élevée (par exemple 5 ou 10%), on réitère l'opération en comparant le modèle à deux régresseurs avec le modèle à trois régresseurs, et ainsi de suite. Les comparaison effectuées ne sont pas à proprement parler un test, puisque l'on ne peut vraiment définir de modèle complet. Cependant, tant que les modèles considérés sont trop simples, la procédure met en jeu de nouveaux modèles de plus en plus complexes. Lorsque la valeur du rapport est supérieure au seuil, le dernier modèle pris en considération est choisi comme modèle complet, et l'on effectue alors la véritable sélection de ses régresseurs, à l'aide d'une méthode MCU (ou MCM), comme nous l'avons décrit dans le paragraphe IV.3.2.1.

La valeur importante de la probabilité utilisée pour l'heuristique conduit généralement à un modèle légèrement trop grand, et la sélection proprement dite, effectuée avec un risque plus petit (≤1%), permet d'éliminer des régresseurs peu pertinents.

Dans la procédure OFR présentée par [Billings 89], la sélection des régresseurs ne se fait pas à l'aide de tests d'hypothèse : la procédure est arrêtée lorsque les contributions des vecteurs non encore classés deviennent négligeables devant la norme de $\mathbf{y}_p$. Plus précisément, si s est le nombre de vecteurs déjà classés et orthogonalisés, on arrête la procédure si la condition suivante est vérifiée :

$$\frac{\|\mathbf{y}_p\|^2 - \sum_{k=1}^{s} \zeta_{ks}}{\|\mathbf{y}_p\|^2} < \rho \qquad \text{(IV.26)}$$

La valeur du seuil ρ permet de moduler le nombre de régresseurs sélectionnés dans le modèle final. Les auteurs ont également utilisé d'autres tests d'arrêt, par exemple :

$$\frac{\zeta_{s+1}}{\|\mathbf{y}_p\|^2 - \sum_{i=1}^{s} \zeta_i} < \rho \qquad \text{(IV.27)}$$

Notons que les critères (IV.26) et (IV.27) tiennent compte de la valeur de l'EQMA obtenue avec le modèle, mais pas de sa complexité. D'autre part, l'utilisation du critère (IV.27) suppose implicitement que les valeurs successives $\zeta_1, \zeta_2, \ldots$ forment une suite décroissante. Bien qu'en pratique cette hypothèse soit le plus généralement vérifiée, elle n'est pas toujours vraie. Enfin, il faut surtout noter que le seuil ρ, choisi de façon arbitraire, n'est pas toujours facile à choisir *a priori*.

IV.3.3 Fin de la première phase : compilation des résultats

Pour chacun des domaines de fonctionnement locaux, on obtient un modèle local, défini par l'ensemble de ses régresseurs et de ses paramètres. On effectue alors la réunion de tous ces ensembles de régresseurs. Lorsque des modèles polynomiaux ont été utilisés, on ne conserve pas, pour la deuxième phase, les monômes de degré supérieur à 1, mais uniquement les termes linéaires à partir desquels ces monômes sont construits (par exemple, si le monôme $y_p(t–1)u(t–3)$ à été sélectionné, on ne garde que les entrées primaires $y_p(t–1)$ et $u(t–3)$).

IV.4. Deuxième phase : sélection des régresseurs d'un modèle neuronal global du processus

A partir des régresseurs obtenus à la fin de la première phase, nous allons maintenant construire un modèle neuronal du processus valable sur tout le domaine de fonctionnement auquel on s'intéresse. On doit donc disposer d'un nouvel ensemble de mesures dans ce domaine, que l'on divise en un ensemble d'apprentissage et un ensemble de validation.

Comme nous l'avons expliqué dans le paragraphe IV.2., on construit alors une suite de modèles de complexité croissante, dont les régresseurs sont les régresseurs sélectionnés lors de la première phase. Le premier modèle est simple (il possède très peu de neurones cachés). Si l'EQMV obtenue avec ce modèle est significativement supérieure à l'EQMA, il y a surajustement, provenant d'une complexité trop importante du réseau par rapport aux informations contenues dans l'ensemble d'apprentissage. Il est donc inutile de considérer des modèles plus complexes, et ce modèle est alors choisi comme modèle complet. Si l'EQMV est comparable à l'EQMA, on considère un nouveau modèle, avec comme entrées les mêmes régresseurs, et un ou plusieurs neurones supplémentaires. On construit ainsi une suite de modèles de complexité croissante, jusqu'à ce que l'on commence à observer du surajustement. Le modèle correspondant est alors choisi comme modèle complet.

On considère l'ensemble de ses sous-modèles qui possèdent le même nombre de neurones, mais dont un ou plusieurs régresseurs ont été supprimés. La seconde phase consiste à sélectionner l'un de ces modèles. Lorsque le nombre n_e de régresseurs du modèle complet est très petit, il est envisageable d'effectuer l'apprentissage de tous ces modèles, puis de sélectionner l'un d'entre eux, à l'aide de tests d'hypothèses ou d'une méthode de type Akaike.

Lorsque n_e est grand, nous utilisons une procédure de sélection partielle, plus économe en nombre d'apprentissages : on effectue l'apprentissage du modèle complet, puis de tous les sous-modèles possédant n_e-1 entrées. On détermine alors celui, noté M_{n_e-1} qui fournit la meilleure prédiction sur l'ensemble d'apprentissage. On le compare alors, soit à l'aide d'un test d'hypothèse, soit à partir d'un critère de type Akaike, au modèle complet M_{n_e}. Si M_{n_e-1} est rejeté, la procédure s'arrête, et M_{n_e} est sélectionné. Si M_{n_e-1} n'est pas rejeté, la sélection continue sur l'ensemble des sous modèles de M_{n_e-1} qui possèdent n_e-2 entrées. On sélectionne à nouveau le meilleur de ces modèles, M_{n_e-2}. Deux démarches sont alors possibles : on compare M_{n_e-2} au modèle complet initial (méthode MCU), ou l'on compare M_{n_e-2} à M_{n_e-1} (méthode MCM). Quelle que soit la méthode choisie, si le modèle M_{n_e-2} est rejeté, la procédure s'arrête, et M_{n_e-1} est

sélectionné. Dans le cas contraire, la procédure se poursuit jusqu'à ce qu'un modèle soit finalement sélectionné.

IV.5. Troisième phase : sélection du nombre de neurones du modèle

Les régresseurs du modèle du processus ont été sélectionnées au cours des deux premières phase de la procédure, et ne sont plus remis en cause. On cherche maintenant à optimiser l'architecture du modèle neuronal, c'est-à-dire à déterminer le nombre de neurones tel que le modèle soit le plus petit modèle suffisamment complexe pour modéliser les données d'apprentissage sans surajustement. Le modèle est alors à la fois performant (l'EQMA et l'EQMV ont des valeurs très proches, et l'EQMV est inférieure ou comparable aux EQMV obtenues avec des modèles plus complexes) et parcimonieux (les EQMA et EQMV obtenues avec tous les modèles comportant moins de neurones sont supérieures à l'EQMA et à l'EQMV obtenues avec ce modèle "optimal").

Le nombre d'entrées du modèle ayant été modifié lors de la deuxième phase, il faut à nouveau évaluer, même grossièrement, le nombre de neurones du modèle. Cette estimation ne constitue en aucun cas une optimisation efficace du nombre optimal de neurones du modèle, et ne doit pas se substituer à la procédure de sélection. Une fois le nombre n_c du nouveau modèle complet choisi, on considère l'ensemble des réseaux complètements connectés ayant comme régresseurs ceux sélectionnées lors des phases précédentes et un nombre de neurones variant de 1 à n_c. Toutes les architectures ainsi définies sont incluses les unes dans les autres. L'architecture incluant toutes les autres est celle du modèle complet, et celle inclue dans toutes les autres est l'architecture linéaire;

On compare donc au modèle complet possédant n_c neurones le modèle possédant n_c–1 neurones, puis, si ce dernier est accepté, on recommence avec le modèle possédant n_c–2 neurones, et ainsi de suite. La procédure s'arrête lorsqu'un modèle est rejeté. Le modèle sélectionné est le modèle précédent, non rejeté.

IV.6. Limitations et extension de la procédure

IV.6.1. Construction et caractérisation de comportements locaux du processus

Lors de la première phase, il faut choisir une décomposition particulière du domaine de fonctionnement du processus, puis, à l'aide d'un choix judicieux de la séquence de commandes, obtenir, pour chacun des domaines locaux, un ensemble d'observations caractéristique du comportement local du processus. Une fois ces ensembles de données collectés, il faut choisir, pour chaque domaine local, un modèle du comportement du processus linéaire par rapport aux paramètres.

La mise en œuvre de cette procédure n'est pas toujours simple. Tout d'abord, il est souvent malaisé de juger du caractère linéaire (ou polynomial) du comportement d'un processus, et d'apprécier la pertinence de l'hypothèse de linéarité. Comment, dans ce cas, choisir le modèle (linéaire ou polynomial ? quel degré pour un modèle polynomial ?), mais, surtout, comment juger de la pertinence du choix des séquences de commande, et donc justifier la partition du domaine de fonctionnement en zones locales que l'on a faite ?

En général, on dispose sur le processus de connaissances *a priori*, provenant par exemple de modélisations antérieures, ou d'une "expérience" du processus, qui permettent d'apporter des réponses satisfaisantes à ces problèmes. D'autre part, on ne cherche pas, lors de cette première phase, à construire des représentations très fidèles du processus, mais simplement à détecter les variables qui ont une action significative sur son comportement, et nous verrons dans le chapitre V que des modèles grossiers sont souvent suffisants. Soulignons enfin que le problème de la détection du caractère linéaire d'un processus a déjà été étudié, et plusieurs tests sont proposés, notamment dans [Haber 85] et [Billings 85].

Par ailleurs, dans cette première partie de la procédure de sélection, on souhaite recueillir sur le processus des observations correspondant à un fonctionnement non "ordinaire", à l'aide de séquences de commandes particulières. Or, il n'est pas toujours facile ou possible de faire fonctionner le processus dans des conditions de fonctionnement en général assez différentes de son mode de fonctionnement réel, et qui, de plus, peuvent être coûteuses à réaliser. Si, pour l'une des ces raisons, la première phase ne peut pas être effectuée, on passe directement à la deuxième phase.

IV.6.2. Sélection de modèles NARMAX

Nous n'avons considéré dans cette procédure que des modèles NARX. En pratique, de nombreux processus dynamiques réels sont mal représentés par des modèles NARX. En revanche, une large classe de processus dynamiques non linéaires peuvent être modélisés à l'aide de modèles hypothèses NARMAX. Dans des domaines restreints de fonctionnement, ces modèles peuvent généralement être approchés par des modèles ARMAX, ou PARMAX (Polynomial ARMAX).

La procédure de classement des régresseurs de modèles ARX (ou PARX), proposée dans le paragraphe IV.3, ne peut être appliquée telle quelle pour des modèles ARMAX ou PARMAX. En effet, dans la première phase, il n'est plus possible d'utiliser la procédure de classement par orthogonalisation, pas plus que tout autre méthode linéaire de type moindres-carrés, puisque les prédicteurs associés aux modèles hypothèses ARMAX ou NARMAX sont bouclés.

Cependant, il est possible d'effectuer la sélection des régresseurs de ces modèles à l'aide d'une des méthodes de sélection partielle que nous avons décrites dans le chapitre III. L'estimation des paramètres de ces modèles bouclés se fait à l'aide d'algorithmes d'estimation itératifs [Rivals 95], et l'on utilise le test LDRT ou un critère de type Akaike pour effectuer la sélection de modèles.

De façon générale, les méthodes de type EP (méthodes fondées sur l'erreur de prédiction, dont nous avons présenté le principe dans les chapitres précédents), peuvent être utilisées sans modification, pour les phases deux et trois, avec des modèles NARMAX. On veillera cependant à ne pas choisir un modèle complet trop complexe, afin de limiter le nombre d'apprentissages.

Chapitre V : Application de la procédure de sélection

Dans ce chapitre, nous mettons en œuvre la procédure de sélection de modèles NARX, présentée dans le chapitre IV. Plusieurs processus simulés sont utilisés pour illustrer les méthodes proposées, mettre en évidence les problèmes, notamment numériques, que l'on est susceptible de rencontrer, et présenter les solutions proposées. Nous nous sommes tout particulièrement intéressés au processus que nous présentons dans le paragraphe V.1, et sur lequel nous avons testé l'ensemble de la procédure.

V.1. Processus de référence P_1

Le processus est simulé par l'équation à temps discret (V.1) suivante :

$$y_p(t) = 50 \tanh\left[2.10^{-3}\left[\frac{24+y_p(t-1)}{3}\ y_p(t-1) - 8\ \frac{u(t-1)^2}{1+u(t-1)^2}\ y_p(t-2)\right]\right] + 0{,}5\ u(t-1) + w(t)$$

où w(t) est un bruit pseudo-blanc gaussien. Son comportement dynamique dépend beaucoup de l'amplitude de la commande :

– pour des commandes d'amplitudes voisines de 0 ($|u|\leq 0{,}1$), le processus se comporte comme un filtre passe-bas linéaire du premier ordre. Dans ce domaine, c'est la partie linéaire de la fonction tanh(.) qui intervient, et l'on peut faire les approximations $1+u(t-1)^2 \approx 1$, $y(t-1) << 24$, et $y_p(t-1) >> u(t-1)^2\ y_p(t-2)$. Cela conduit au modèle linéaire du premier ordre d'équation :

$$y_p(t) \cong 0{,}8\ y_p(t-1) + 0{,}5u(t-1) + w(t) \qquad (V.2)$$

Ce modèle est stable, avec un gain statique de l'ordre de 2,5. Si l'on considère que P_1 représente la discrétisation d'un processus à temps continu avec une période d'échantillonnage T, la constante de temps du processus vaut environ 3,6T (Figure V.1.a).

– pour des amplitudes de |u| variant de 0,1 à 0,6, le comportement du processus peut être représenté par un modèle du premier ordre non linéaire stable (Figure V.1.a)

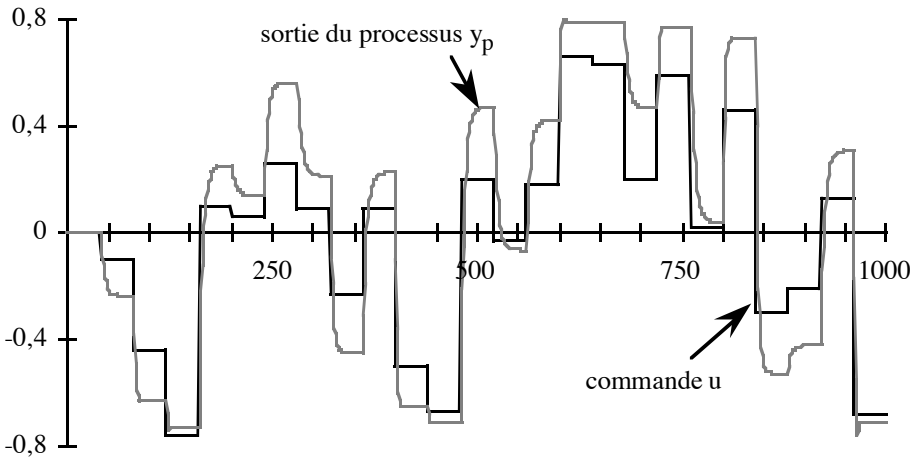


Figure V.1.a

Comportement du processus pour des commandes variant entre 0 et 0,8 (var(w)=0)

– lorsque l'amplitude de la commande varie entre 0,6 et 10, le comportement est non linéaire, avec un comportement oscillatoire pour des amplitudes supérieures à 1 (Figure V.1.b). Le processus devient non symétrique par rapport à u=0 pour des amplitudes de u supérieures à 5.

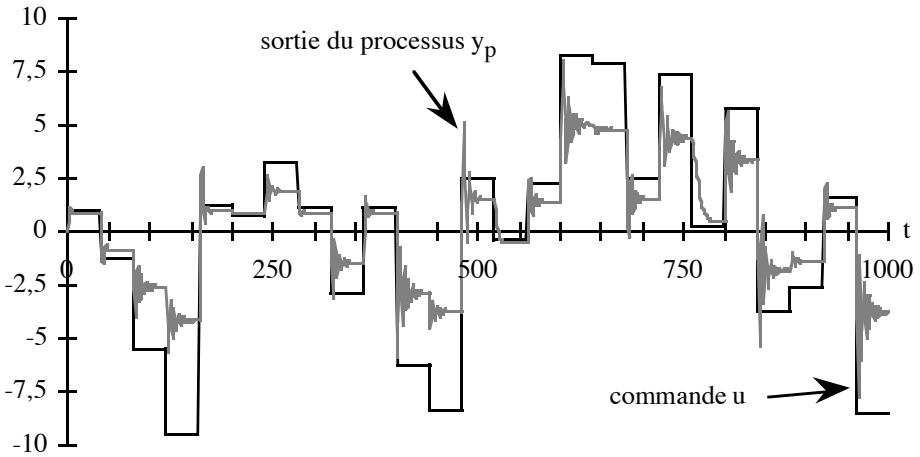


Figure V.1.b

Comportement du processus pour des commandes variant entre 0 et 10

Notons que le processus peut présenter, dans certains cas, un comportement très différent de ceux décrits ci-dessus : lorsque l'amplitude de la commande u appliquée au processus est très grande et à valeur positive (u>10), ou lorsque le bruit perturbant le processus est relativement important (|w|>5), le processus change de domaine de fonctionnement, et l'amplitude de $y_p(t)$ est supérieure à 50. On observe ce phénomène sur la figure V.1.c.

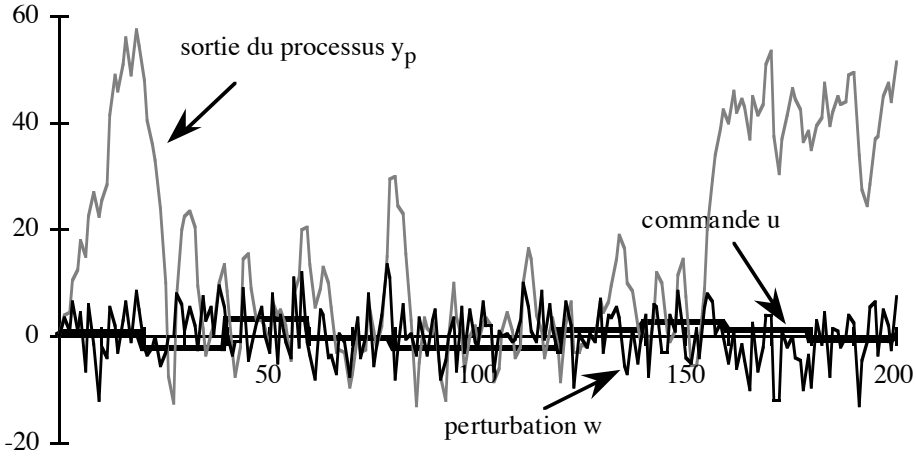


Figure V.1.c

Dans notre étude, nous avons évité ce domaine de fonctionnement en nous limitant à une séquence de commande comprise dans le domaine [-10, 10].

V.2. Première phase : sélection des entrées de modèles linéaires

V.2.1. Étude préliminaire : problème du surajustement

Nous avons évoqué, dans le chapitre précédent, le problème du surajustement : lorsque la taille de l'ensemble d'apprentissage est insuffisant, l'apprentissage conduit à une modélisation d'informations non représentatives du comportement général du processus, mais spécifiques de cet ensemble d'apprentissage particulier. Lorsque le processus est bruité, l'EQMA obtenue avec un modèle trop complexe peut être inférieure à la variance du bruit. Or, au cours de la première phase de la procédure de sélection, il faut choisir des modèles locaux comportant le plus grand nombre possible de régresseurs, afin d'être assuré que les régresseurs significatifs sont pris en considération. De plus, lorsque l'on utilise des modèles polynomiaux, le nombre de paramètres devient encore plus important.

Pour limiter le surajustement, il est donc nécessaire d'étudier l'influence du rapport (taille de l'ensemble d'apprentissage/taille du modèle) sur le surajustement. Si l'on peut considérer autant de données qu'on le désire, on choisira un ensemble d'apprentissage de taille suffisamment élevée. S'il n'est pas possible de recueillir suffisamment de données, il faut alors limiter la taille des modèles, au risque de ne pas obtenir tous les régresseurs significatifs. Afin de mieux appréhender ce problème, nous avons étudié quelques exemples, que nous présentons maintenant.

V.2.1.1. Premier exemple : processus ARX.

Le processus est simulé par un filtre passe-bas du second ordre ARX :

$$y_p(t) = 1{,}88\ y_p(t–1) – 0{,}882\ y_p(t–2) + 0{,}0166\ u(t–1) + 0{,}00715\ u(t–2) + w(t) \quad (V.3)$$

Le processus simulé est étudié autour du point de fonctionnement correspondant à une commande constante U_o=0. On explore le voisinage de ce point en superposant à U_o une perturbation aléatoire de distribution uniforme d'amplitude Δu=0,1. Le modèle complet choisi possède n_θ=100 entrées {$y_p(t–1)$, .., $y_p(t–50)$, $u(t–1)$, .., $u(t–50)$}. Nous avons effectué plusieurs simulations, correspondant à des variances de {w(t)} variant de 10^{-4} à 1, et nous avons utilisé à chaque fois deux ensembles d'apprentissage, le premier de taille N=100, le second de taille N=1000.

Jusqu'à présent, nous avons utilisé l'erreur quadratique moyenne (EQM) comme estimation de la variance de l'erreur de prédiction. Cependant, lorsque le nombre d'exemple et le nombre de paramètres du modèle sont de valeurs comparables, cette estimation est biaisée, et il est préférable d'utiliser l'estimation non biaisée EQM/(N–n_θ).

Chaque fois qu'un nouveau régresseur est choisi et classé, nous avons donc calculé les estimations non biaisées de la variance de l'erreur sur l'ensemble d'apprentissage et de la variance de l'erreur sur l'ensemble de validation, respectivement EQMA'=EQMA/(N–p) et EQMV'=EQMV/(N–p), où p est le nombre de régresseurs du modèle considéré. Des résultats, caractéristiques de ce que nous avons observé, sont présentés dans les figures V.2.a et V.2.b. Ils correspondent à une variance de {w(t)} égale à 1.

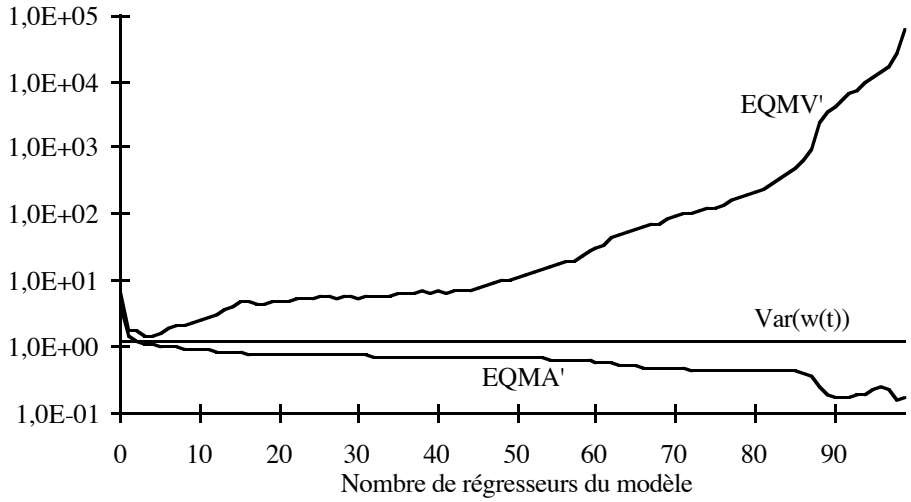


Figure V.2.a : var($y_p(t)$)≈8, var(w(t))≈1, N=100

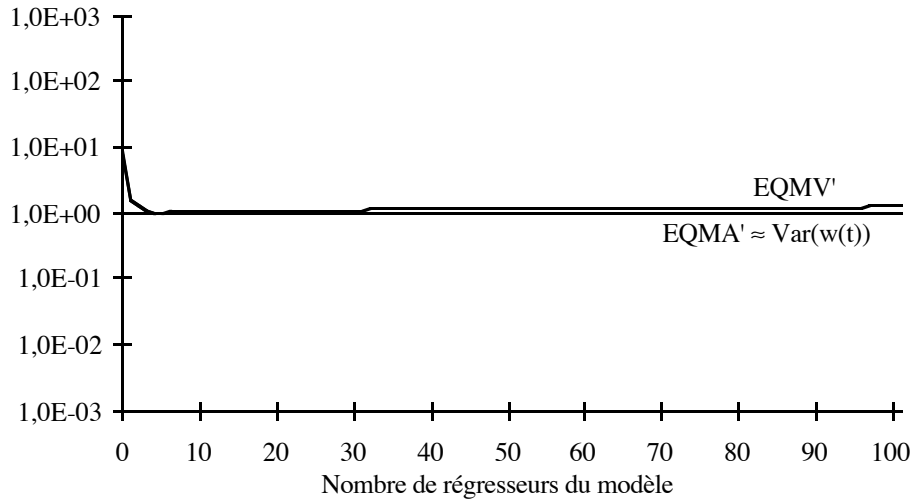


Figure V.2.b : var($y_p(t)$)≈8, var(w(t))≈1, N=1000

On constate que lorsque le nombre de paramètres est proche de la taille de l'ensemble d'apprentissage, l'EQMA diminue fortement, alors que l'EQMV' augmente de façon très importante (Figure V.2.a). Lorsque la taille de l'ensemble d'apprentissage est grande devant le nombre de paramètres du modèle, l'EQMA' et l'EQMV' restent très proches (Figure V.2.b). De plus, leurs valeurs peuvent être remplacés par celle de l'EQMA et de l'EQMV, ce que nous ferons dans la suite de ce chapitre.

V.2.1.2. Deuxième exemple : le processus NARX P_1

Nous avons répété la même expérience avec le processus NARX P_1, pour plusieurs points de fonctionnement. En chacun de ces points, nous avons superposé à la commande constante une composante aléatoire de densité uniforme d'amplitude maximale $\Delta u=0,1$ (soit $var(u(t))=3,33.10^{-3}$); le modèle complet possède $n_\theta=100$ entrées $\{y_p(t-1), \ldots, y_p(t-50), u(t-1), \ldots, u(t-50)\}$. Comme dans l'exemple précédent, nous avons effectué plusieurs simulations, correspondant à des variances de w(t) variant de 10^{-4} à 1. Les ensembles d'apprentissage sont de tailles N=100, 200 ou 1000.

Nous présentons ci-dessous les résultats obtenus pour une variance de w de l'ordre de 1, pour une valeur constante de la commande $U_o=0,1$ (Figures V.3.a, V.3.b et V.3.c).

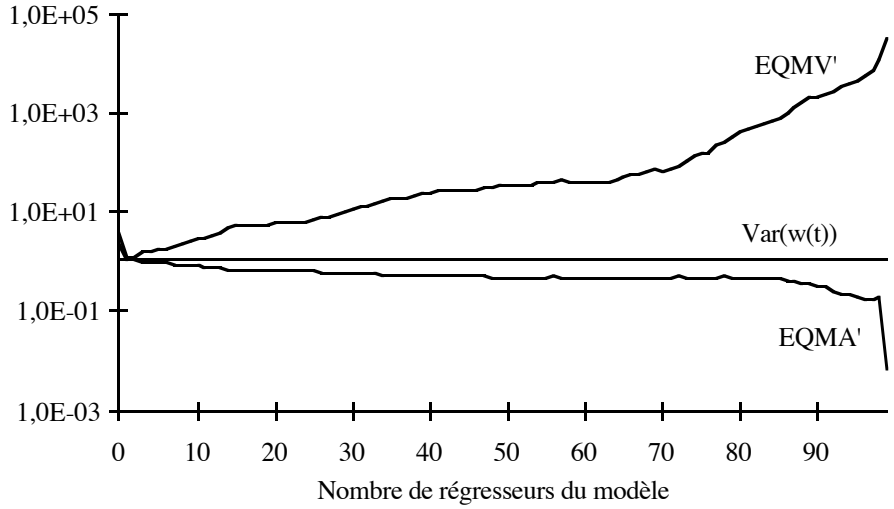


Figure V.3.a : var($y_p(t)$)≈2,1, var(w(t))≈1, N=100

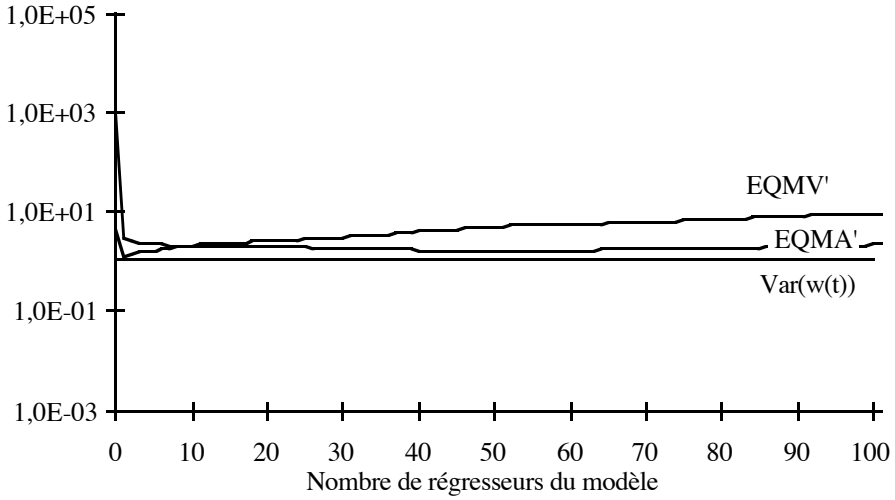


Figure V.3.b : var($y_p(t)$)≈2,1, var(w(t))≈1, N=200

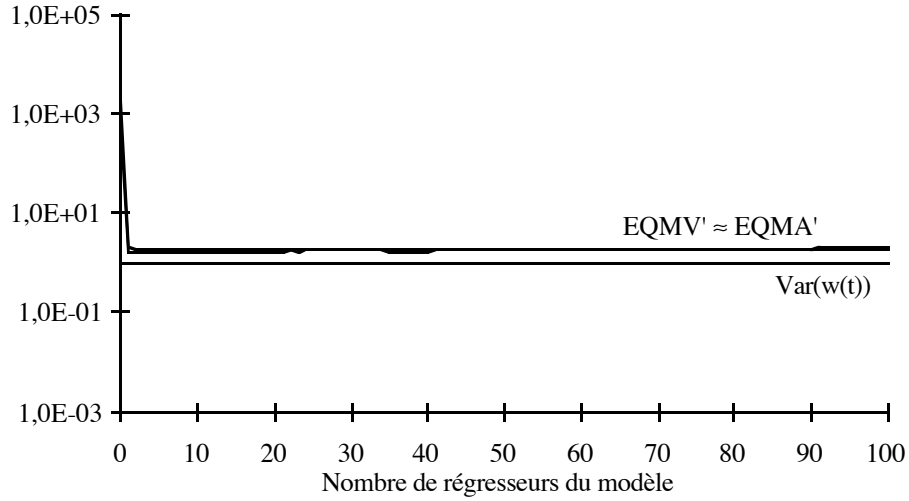


Figure V.3.c : var($y_p(t)$)≈2,1, var(w(t))≈1, N=1000

Pour N=100, le surajustement est important. Pour N=200, les courbes des EQMA' et des EQMV' sont proches, le surajustement observé est acceptable. Pour N=1000, le phénomène est négligeable, même pour les plus grands modèles ($n_\theta \approx 100$). Ces résultats sont représentatifs des résultats observés pour l'ensemble des essais que nous avons effectués.

V.2.2. Choix de l'amplitude de la perturbation superposée à la commande

Nous cherchons à construire, dans la première phase, une "collection" de modèles du processus, chacun d'entre eux correspondant à un domaine de fonctionnement restreint. Pour agir sur le processus dans un domaine de fonctionnement particulier, le seul moyen dont on dispose est la commande. Nous définirons ici un domaine local de fonctionnement du processus comme un voisinage d'un point de fonctionnement stable. Chacune des séquences de commande nécessaires est obtenue en superposant à une commande constante U_o, qui détermine le point de fonctionnement, une perturbation aléatoire $\Delta u(t)$ de distribution uniforme dans $[-\Delta u_o, \Delta u_o]$. Cependant, en pratique, la mise en œuvre d'une telle démarche n'est pas toujours possible, et il faut prendre en considération les remarques suivantes :

– dans la réalité, les processus subissent des perturbations aléatoires sur lesquelles on ne peut pas agir, et que l'on ne peut mesurer : il est donc important de savoir si une séquence de commande a une action significative sur le processus, ou, au contraire, si son influence est négligeable devant l'influence du bruit. Lors de la constitution des séquences d'apprentissage locales, on cherchera donc à détecter si la variance de $y_p(t)$ varie en fonction de l'amplitude Δu_o;

– d'autre part, supposons que l'on choisisse un modèle linéaire pour modéliser le processus dans un domaine local particulier; si la valeur de Δu_o est trop importante, le modèle linéaire n'est plus valable. Il est donc utile de détecter si le processus présente effectivement un comportement proche d'un comportement linéaire pour une valeur donnée Δu_o. Le problème se pose également lorsque l'on choisit un modèle polynomial de degré fixé;

– enfin, il faut s'efforcer de construire un ensemble de modèles recouvrant tout le domaine de fonctionnement étudié. Pour cela, il faut choisir un nombre suffisant de points de fonctionnement, puis déterminer pour chacun d'eux un domaine de fonctionnement local, de telle sorte que la réunion de ces domaines locaux englobe tout le domaine de fonctionnement. Remarquons que nous ne cherchons pas dans cette première phase à construire un modèle global à partir de modèles locaux, mais simplement à sélectionner les régresseurs ayant une action

significative sur le comportement du processus. Il n'est donc pas indispensable que les modèles locaux recouvrent parfaitement le domaine de fonctionnement que l'on étudie.

Notre problème est de trouver une "bonne" valeur de Δu_o. On commence par estimer la valeur minimale de Δu_o pour que les variations de la commande influent de façon significative sur le comportement du processus. Pour cela, on peut, par exemple, estimer la variance de $y_p(t)$ pour une commande constante U_o, puis l'on superpose à cette commande une séquence de perturbations d'amplitude maximale Δu_o faible, et l'on estime à nouveau la variance de $y_p(t)$. Si l'ordre de grandeur de celle-ci reste identique, on augmente la valeur de Δu_o (par exemple d'un facteur 10), et l'on estime à nouveau la variance de $y_p(t)$. On procède ainsi jusqu'à trouver une valeur Δu qui nous satisfait.

Le comportement du processus est-il encore linéaire pour cette valeur de Δu ? Nous proposons ici deux tests simples à mettre en œuvre qui peuvent apporter des informations sur le comportement d'un processus dans une zone particulière :

Étude de la caractéristique statique : on applique une commande de valeur U_o, et l'on note Y_o la valeur moyenne de $y_p(t)$ en régime stationnaire. Puis l'on applique des commandes $U_o+\Delta U_1$, $U_o+\Delta U_2$, …, et l'on mesure les valeurs moyennes de $y_p(t)$ correspondantes $Y_o+\Delta Y_1$, $Y_o+\Delta Y_2$, … Si la caractéristique statique du processus n'est pas linéaire autour du point (U_o, Y_o), (le rapport $\Delta Y_i/\Delta U_i$ n'est pas constant), le comportement du processus est non linéaire. Si le rapport ($\Delta Y_i/\Delta U_i$) est constant, cela n'implique pas nécessairement la validité d'un modèle linéaire.

Étude du comportement dynamique : nous cherchons à mettre en évidence un comportement dynamique correspondant à un processus linéaire; on cherche donc des caractéristiques du processus pendant un régime transitoire. Pour cela, on effectue n "expériences" : en partant d'un état initial (U_o, Y_o), on applique différents échelons de commande d'amplitude ΔU_1, … ΔU_n. On mesure alors, pour chaque expérience, des valeurs particulières Y_i (maximum, mi-hauteur entre valeur initiale et valeur finale, …) de l'écart entre les sorties $y_p(t)$ du processus et Y_o. Si le processus est linéaire, les points définis par les couples (ΔU_i, Y_i) appartiennent à une même droite passant par (0, 0). Cette méthode est cependant très sensible au bruit w(t), et la présence d'un niveau de bruit important peut fortement modifier l'allure de la courbe pour de faibles valeurs de ΔU_i. Pour remédier à cet inconvénient, on peut prendre en considération plusieurs points autour du point particulier choisi, afin de moyenner l'effet du bruit.

Les méthodes que nous venons de présenter sont simples, et peuvent s'avérer utiles pour choisir une valeur Δu_o. Cependant, pour être relativement performantes, elles nécessitent un nombre important d'expériences qu'il n'est pas toujours possible d'effectuer. Toutefois, le choix de Δu_o n'est pas critique dans la procédure que nous proposons : il n'est pas nécessaire que le modèle soit une très bonne approximation du processus dans le domaine local que l'on étudie, et une évaluation grossière de l'ordre de grandeur de Δu_o est généralement suffisante. De plus, lorsque l'on modélise un processus réel, on dispose souvent de connaissances suffisantes pour choisir, dans chaque zone locale étudiée, un modèle linéaire ou polynomial du processus qui soit une approximation acceptable du processus. On évite ainsi des expériences préliminaires nombreuses et coûteuses, ou impossibles à effectuer.

V.2.3. Résultats obtenus avec le processus NARX P_1

Le processus étudié est le processus P_1, défini par (V.1). Plusieurs études ont été effectuées, correspondant à des processus perturbés par des bruits de variances différentes : $var\{w(t)\} = \{10^{-4}; 10^{-2}; 10^{-1}; 1; 10\}$. Le domaine de fonctionnement global auquel nous nous intéressons correspond à des amplitudes de la commande u variant dans le domaine [-10, +10].

Pour plusieurs valeurs de U_o, une étude rapide est effectuée pour déterminer une valeur satisfaisante de Δu_o. Pour $U_o=0$, on se limite à une valeur maximale de $\Delta u=0,1$. En dehors de ce point particulier, un bon compromis est $\Delta u=1,0$. Ce choix permet de se limiter à 13 études locales correspondant aux valeurs de U_o suivantes : $U_o \in \{-10; -8; -6; -4; -2; -1; 0; 1; 2; 4; 6; 8; 10\}$.

En chacun de ces points, nous choisissons un modèle affine par rapport aux régresseurs $\{y_p(t–1), \ldots, y_p(t–n_y), u(t–1), u(t–n_u)\}$, avec 201 régresseurs ($n_y=n_u=100$, une entrée constante). L'entrée constante est classée, et éventuellement sélectionnée, au même titre que les autres. On simule le processus sur une séquence d'entrée-sortie de 1100 échantillons. Les 100 premières observations de $y_p(t)$ et u(t) ne sont pas prises en considération dans l'ensemble d'apprentissage, mais sont utilisées pour initialiser le modèle. Ainsi, la taille de l'ensemble d'apprentissage est N=1000.

V.2.3.1. Comparaison des procédures MCU et MCM

Nous avons comparé deux procédures de sélection, la procédure MCU (le même modèle complet est utilisé pour tous les tests), et la procédure MCM (lorsqu'un sous-modèle du modèle complet est accepté, il est choisi comme nouveau modèle complet). Dans ces deux procédures, (cf. paragraphe IV.3.2.5), nous procédons à une présélection des 201 régresseurs du modèle de départ, à

l'aide d'un "pseudo-test" LDRT à 5%. On obtient une première liste restreinte de régresseurs qui définit le modèle complet. On applique alors l'une ou l'autre des deux méthodes de sélection.

U_o	**Procédure MCM** (1)	**Procédure MCU** (1)+ (2)
–10,0	(1) = $\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{u_1}$, y_{10}	(2) = u_{15}, u_{72}, u_{62}, u_{24}, u_{100}, y_{14}, u_{22}
–8,0	$\mathbf{y_2}$, $\mathbf{y_1}$, $\mathbf{u_1}$, u_{73}, u_{61}, u_{77}	y_{52}, y_{79}
–6,0	y_3, $\mathbf{y_2}$, $\mathbf{y_1}$, $\mathbf{u_1}$, u_{84}	y_{93}, y_{100}, u_{11}, u_6
–4,0	y_3, $\mathbf{y_2}$, $\mathbf{y_1}$, $\mathbf{u_1}$	y_{42}, u_{24}, y_{79}, u_{11}, u_{47}, u_{26}, y_{12}
–2,0	y_3, $\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{u_1}$, u_{97}, u_{26}	u_{16}, y_{90}, u_{43}, y_{59}, u_{100}, u_{22}
–1,0	$\mathbf{y_1}$, $\mathbf{y_2}$, y_{48}, u_{65}	u_{45}, y_{79}, $\mathbf{u_1}$, u_{99}, u_{42}
0,0	$\mathbf{y_1}$, $\mathbf{u_1}$	u_{79}, u_5
1,0	$\mathbf{y_1}$, $\mathbf{y_2}$, u_{60}	$\mathbf{u_1}$, y_{93}, y_9, y_{26}, u_{82}, u_{29}, y_{51}, y_{65}, u_{65}
2,0	$\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{u_1}$, u_{78}, u_{27}	y_{62}, y_{70}, u_{96}, u_{60}, u_{86}, y_3, u_{69}
4,0	y_3, $\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{u_1}$, u_{30}	y_{92}, y_{37}, u_{13}, u_{48}, u_{87}
6,0	y_3, $\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{u_1}$	u_{35}, y_{69}, u_{44}
8,0	$\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{u_1}$, y_5	u_8, u_{35}, u_{98}, y_{99}, u_{24}, u_3, y_{77}
10,0	$\mathbf{y_1}$, $\mathbf{y_2}$, $\mathbf{u_1}$, u_{100}	u_7, u_{30}, u_{82}, u_{39}, u_{23}, u_{59}
Total	$\mathbf{y_p(t–1)}$, $\mathbf{y_p(t–2)}$, $\mathbf{u(t–1)}$ + 16 régresseurs	+ 50 régresseurs

Tableau V.1

Nous présentons les résultats obtenus pour un processus perturbé par un bruit de variance var(w)≈10^{-2} dans le tableau V.1. Pour faciliter sa lecture du tableau, nous avons noté y_i le régresseur $y_p(t–i)$, et u_j le régresseur $u(t–j)$. La valeur de U_o est indiquée dans la première colonne, les régresseurs sélectionnés avec la procédure MCM apparaissant dans la seconde colonne, ceux sélectionnés avec la procédure à MCU sont les régresseurs de la deuxième colonne auxquelles s'ajoutent ceux de la troisième colonne. Les arguments de la fonction φ(.) du processus simulé (V.1) sont indiqués en gras.

On constate que les modèles sélectionnés à l'aide de la procédure MCU sont beaucoup plus complexes que ceux sélectionnés avec la procédure MCM : outre les trois régresseurs qui sont des arguments de φ(.), 16 régresseurs sont conservés avec la procédure à modèles complets multiples, et 70 environ avec la procédure

à modèle complet unique, ce qui est beaucoup trop pour envisager de passer à la seconde phase de la sélection.

Notons que ces résultats illustrent le point souligné dans le chapitre III (III.3.5. et III.4.2.2.) : lorsque l'on sélectionne un modèle dans un ensemble, la façon dont on "explore" cet ensemble influe sur le choix du modèle sélectionné. Ces deux procédures de sélection de modèles, fondées sur le même test d'hypothèse, avec le même risque, conduisent à la sélection de modèles très différents.

D'autre part, nous utilisons pour modéliser le comportement du processus des modèles complets linéaires. Or, le processus n'étant pas parfaitement linéaire dans chacun des domaines étudiés, l'EQMA obtenue avec les différents modèles complets ne dépend pas seulement de la variance du bruit, mais également de l'*erreur déterministe* résultant de la modélisation imparfaite du processus, ce qui perturbe les résultats des tests d'hypothèses.

Ces résultats sont assez caractéristiques de ce que l'on observe pour d'autres niveaux de bruit, et sur d'autres processus simulés que le processus P_1. Nous n'utiliserons plus pour la première phase que la procédure MCM.

V.2.3.2. Modification de la procédure de sélection.

A chaque sélection, pour une valeur particulière de U_o, des régresseurs non pertinents sont sélectionnés (cf. Tableau V.1). Si l'on garde tous les régresseurs sélectionnés dans chaque domaine local, le nombre des régresseurs conservés à la fin de la première phase est trop important, et l'utilité de cette première partie de la procédure de sélection devient alors minime. Cependant, si l'on effectue plusieurs sélections avec des ensembles d'apprentissage différents, les régresseurs sélectionnés d'une expérience à l'autre sont généralement très différents. Nous allons donc distinguer les régresseurs dont la sélection est reproductible sur plusieurs expériences de ceux qui sont sélectionnés de façon aléatoire, en fonction de réalisations particulières des variables aléatoires W et Y_p. Nous proposons d'utiliser, pour chaque point de fonctionnement, plusieurs ensembles d'apprentissage, et d'effectuer, pour chacun d'eux, une sélection des régresseurs. Les régresseurs sélectionnés dans deux expériences, ou plus, sont conservés, les autres sont rejetés.

V.2.3.3. Résultats

Pour chacun des treize domaines locaux de fonctionnement choisis, on procède à la sélection des régresseurs du modèle en utilisant trois ensembles d'apprentissage différents, et l'on conserve ceux qui sont sélectionnés au moins deux fois sur trois. Pour chaque sélection, on effectue une présélection des régresseurs à l'aide d'un test LDRT à 5%. La sélection des régresseurs se fait

ensuite avec une procédure MCM, les modèles étant comparés à l'aide d'un test LDRT. Deux valeurs du risque ont été utilisées, 1% et 1‰. Les simulations ont été effectuées avec des niveaux de bruit dont la variance varie de 10^{-4} à 10. Le modèle complet possède 200 entrées, plus une entrée constante (n_y=100, n_u=100). Les résultats obtenus sont présentés dans les tableaux V.2 et V.3. Les variables marquées d'un astérisque correspondent aux régresseurs sélectionnées deux fois sur trois seulement.

Var(w)=10^{-2}	**Tests à 1‰** (1)	**Tests à 1%** (1)+ (2)
U_o = –10	(1) = **y_1, y_2, u_1**	(2) = {ø}
–8	**y_1, y_2, u_1**	{ø}
–6	**y_1, y_2,** y_3, **u_1**	{ø}
–4	**y_1, y_2,** y_3, **u_1,** u_2	{ø}
–2	**y_1, y_2,** y_3, **u_1,** u_2	{ø}
–1	**y_1, y_2, u_1**	{ø}
0	**y_1, y_2, u_1**	{ø}
1	**y_1, y_2, u_1**	{ø}
2	**y_1, y_2, u_1**	{ø}
4	**y_1, y_2,** y_3, **u_1,** u_2	{ø}
6	**y_1, y_2,** y_3, **u_1,** u_2	{ø}
8	**y_1, y_2, u_1**	{ø}
10	**y_1, y_2, u_1**	{ø}
Total	**y_p(t–1), y_p(t–2),** y_p(t–3) **u(t–1),** u(t–2)	{ø}

Tableau V.2

Sélections des régresseurs de modèles locaux, pour var(w)=0,01

Nous pouvons faire les remarques suivantes :

– deux régresseurs, y_p(t–3) et u(t–2), qui n'apparaissent pas dans l'expression du processus P_1, sont sélectionnés de façon presque systématique, et pour plusieurs points de fonctionnements (Tableau V.2). Or, si l'on revient à une interprétation géométrique, on peut remarquer que le vecteur $\mathbf{y}_3$ est presque colinéaire aux vecteurs $\mathbf{y}_1$, $\mathbf{y}_2$, tandis que $\mathbf{u}_2$ est très proche de $\mathbf{u}_1$. La sélection de ces deux régresseurs est donc peu surprenante, et permet de "compenser" en partie l'insuffisance du modèle linéaire;

– en revanche, les régresseurs y_p(t–52), y_p(t–49), y_p(t–6) ne sont sélectionnées que pour une seule valeur de U_o, et dans deux essais sur trois seulement (ces

détails des résultats n'apparaissent pas dans le tableau V.3). L'utilisation d'autres ensembles d'apprentissage conduit à la sélection de régresseurs différents. La sélection répétée de ces régresseurs est donc une simple coïncidence. Néanmoins, ce phénomène reste assez limité, et l'on peut espérer que ces régresseurs seront éliminés lors de la seconde phase;

– les résultats obtenus avec des risques de 1‰ et 1% sont très proches.

Var(w)	**Tests à 1‰** (1)	**Tests à 1%** (1)+ (2)
10^{-4}	**$y_p(t-1)$, $y_p(t-2)$**, $y_p(t-3)$ **$u(t-1)$**, $u(t-2)$	$y_p(t-52)$*
10^{-2}	**$y_p(t-1)$, $y_p(t-2)$**, $y_p(t-3)$ **$u(t-1)$**, $u(t-2)$	{ø}
10^{-1}	**$y_p(t-1)$, $y_p(t-2)$**, $y_p(t-3)$ **$u(t-1)$**, $u(t-2)$*	$y_p(t-49)$*
1,0	**$y_p(t-1)$, $y_p(t-2)$**, $y_p(t-3)$ **$u(t-1)$**	{ø}
10,0	**$y_p(t-1)$, $y_p(t-2)$**, $y_p(t-3)$ $y_p(t-6)$*, **$u(t-1)$**	{ø}

Tableau V.3

Régresseurs sélectionnés (total) pour des simulations avec différents niveaux de bruit

L'amélioration des résultats est très satisfaisante. Les régresseurs conservés en chaque point sont beaucoup moins nombreux et les arguments de φ(.), {$y_p(t-1)$, $y_p(t-2)$, $u(t-1)$}, sont sélectionnés pour toutes les simulations.

V.2.4. Résultats obtenus avec d'autres processus

Nous avons testé cette procédure sur trois autres processus. Nous présentons rapidement ici les processus simulés et les résultats obtenus.

V.2.4.1. Présentation des processus

**Processus P_2*

Le deuxième exemple choisi est un processus non linéaire du premier ordre, proposé par O. Nerrand [Nerrand 92a], et simulé à l'aide de l'équation suivante :

$$y_p(t) = [1 - \frac{T}{a + b\, y_p(t-1)}]\, y_p(t-1) + [\, T \frac{c + d\, y_p(t-1)}{a + b\, y_p(t-1)}\,]\, u(t-1) + w(t) \qquad (V.4)$$

avec :

$a = -0{,}139, \quad b = 1{,}20, \quad c = 5{,}633, \quad d = -0{,}326$

**Processus P_3*

Ce processus et le suivant ont été proposés par Narendra et Parthasarathy [Narendra 90]. L'équation de simulation du processus P_3 est :

$$y_p(t) = \frac{y_p(t-1)\, y_p(t-2)\, y_p(t-3)\, u(t-2)\, (y_p(t-3) - 1) + u(t-1)}{(1 + y_p(t-2)^2 + y_p(t-3)^2)} + w(t) \quad (V.5)$$

Il est stable pour des commandes variant entre –1,12 et +1,12. Nous nous limiterons donc à des valeurs de la commande appartenant à [–1, +1].

**Processus P_4*

L'expression de l'équation de simulation du processus non linéaire P_4 est :

$$y_p(t) = \frac{5\, y_p(t-1)\, y_p(t-2)}{1 + y_p(t-1)^2 + y_p(t-2)^2 + y_p(t-3)^2} + u(t-1) + 0.8\, u(t-2) + w(t) \quad (V.6)$$

Ce processus est étudié pour des valeurs de la commande variant dans le domaine [–10, 10].

V.2.4.2. Résultats

**Processus P_2 (Tableau V.4)*

Comme dans le cas du processus P_1, trois ensembles d'apprentissage sont utilisés pour chaque domaine local. Nous avons choisi les points de fonctionnement {0; 1; 2; 4; 6; 8; 10}. Le modèle de départ possède 201 entrées (n_u=n_y=100), la taille de chaque ensemble d'apprentissage est N=1000. Les résultats obtenus sont tout à fait satisfaisants.

Var(w)	**Tests à 1‰** (1)	**Tests à 1%** (1)+ (2)
10^{-4} (Δu=0,1)	**y_p(t–1)**, **u(t–1)**, y_p(t–42)*	y_p(t–14)*, y_p(t–46)*
10^{-2} (Δu=1)	**y_p(t–1)**, **u(t–1)**	{ø}
10^{-1} (Δu=10)	**y_p(t–1)**, **u(t–1)**	{ø}
1 (Δu=10)	**y_p(t–1)**, **u(t–1)**	{ø}

Tableau V.4

**Processus P_3 (Tableau V.5)*

Comme nous l'avons déjà dit, ce processus devient instable pour |u|≤1,12, ou pour un bruit w trop important : nous avons donc choisi comme ensemble de valeurs de U_o {–0,9; –0,8; –0,6; –0,4; –0,2; 0,0; 0,2; 0,4; 0,6; 0,8; 0,9}, et Δu=0,1. Nous avons effectué des simulations avec deux niveaux de bruit : var(w) = 10^{-4} et 10^{-2}. Les résultats sont présentés dans le tableau V.5.

Var(w)	**Tests à 1‰** (1)	**Tests à 1%** (1)+ (2)
10^{-4} (Δu=1)	**y_p(t–1), y_p(t–2), y_p(t–3) u(t–1), u(t–2)**	{ø}
10^{-2} (Δu=1)	**y_p(t–1), y_p(t–2), y_p(t–3) u(t–1), u(t–2)**	{ø}

Tableau V.5

**Processus P_4 (Tableau V.6)*

Nous avons choisi d'étudier le processus pour des commandes variant dans le domaine [–10, +10]. L'ensemble des valeurs de U_o est {–10; –8; –6; –4; –2; 0; 2; 4; 6; 8; 10}.

Var(w)	**Tests à 1‰** (1)	**Tests à 1%** (1)+ (2)
10^{-4} (Δu=1)	**y_p(t–1), y_p(t–2), y_p(t–3),** y_p(t–4), y_p(t–5)*, y_p(t–6) **u(t–1), u(t–2)**, u(t–3), u(t–4), u(t–5)	y_p(t–5)
10^{-2} (Δu=1)	**y_p(t–1), y_p(t–2), y_p(t–3),** y_p(t–4), y_p(t–5), y_p(t–83)* **u(t–1), u(t–2)**, u(t–3), u(t–4), u(t–5)	y_p(t–43)*
10^{-1} (Δu=1)	**y_p(t–1), y_p(t–2), y_p(t–3),** y_p(t–4), **u(t–1), u(t–2)**, u(t–3), u(t–4)	{ø}
1 (Δu=1)	**y_p(t–1), y_p(t–2), y_p(t–3), u(t–1), u(t–2)**	y_p(t–4)*
10 (Δu=1)	**y_p(t–1), y_p(t–2), y_p(t–3)*, u(t–1), u(t–2)**	{ø}

Tableau V.6

** Conclusions*

Deux conclusions apparaissent à l'analyse des résultats obtenus sur les quatre processus étudiés :

– lorsque le niveau de bruit est élevé et perturbe fortement le processus, le modèle sélectionné est généralement plus simple que lorsque la variance de la perturbation w est faible ou nulle;

– dans tous les exemples que nous avons présentés, les arguments des fonctions simulant les processus sont toujours sélectionnés.

Dans tous les cas, nous avons considéré 201 régresseurs "candidats", une dizaine d'entre eux au maximum est finalement conservé. Il est maintenant tout à fait envisageable de passer à la deuxième phase, et d'utiliser, comme modèle global du processus, un modèle neuronal, mieux adapté que des modèles linéaires ou polynomiaux à la modélisation de processus non linéaires.

V.3. Deuxième phase : sélection des entrées d'un modèle non linéaire global

Dans la première phase, la liste des régresseurs susceptibles d'être utilisés comme entrées d'un modèle neuronal du processus a été réduite de façon importante. Cependant, tous les régresseurs sélectionnés ne sont pas toujours utiles. Nous allons donc construire un modèle non linéaire global, dont les entrées sont les régresseurs sélectionnés lors de la première phase, puis effectuer une nouvelle sélection des régresseurs, comme nous l'avons décrit dans le chapitre IV.

V.3.1. Résultats obtenus avec le processus de référence P_1

Nous présentons ici les résultats obtenus avec un processus perturbé par un bruit de variance var(w(t))=10^{-2}. Les régresseurs sélectionnés lors de la première phase sont {$\mathbf{y_p(t-1)}$, $\mathbf{y_p(t-2)}$, $y_p(t-3)$, $\mathbf{u(t-1)}$, $u(t-2)$}.

Nous utilisons un ensemble d'apprentissage de N=1000 points, correspondant à la réponse du processus à une séquence de créneaux d'amplitude variant entre -10 et 10 (Figure V.4), et qui fait bien fait apparaître le comportement non linéaire global du processus.

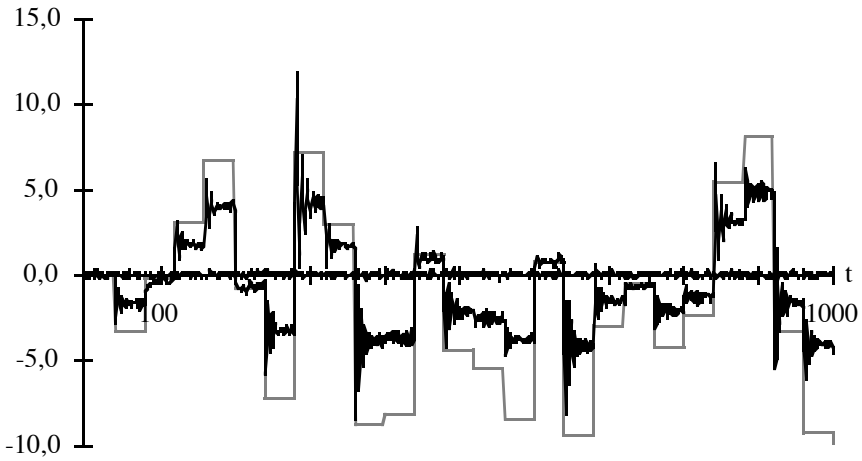


Figure V.4

Séquence d'apprentissage (var(w(t))≈0,1)

La figure V.5 représente la séquence de validation, obtenue pour des conditions de fonctionnement du processus similaires à celles utilisées pour construire l'ensemble d'apprentissage.

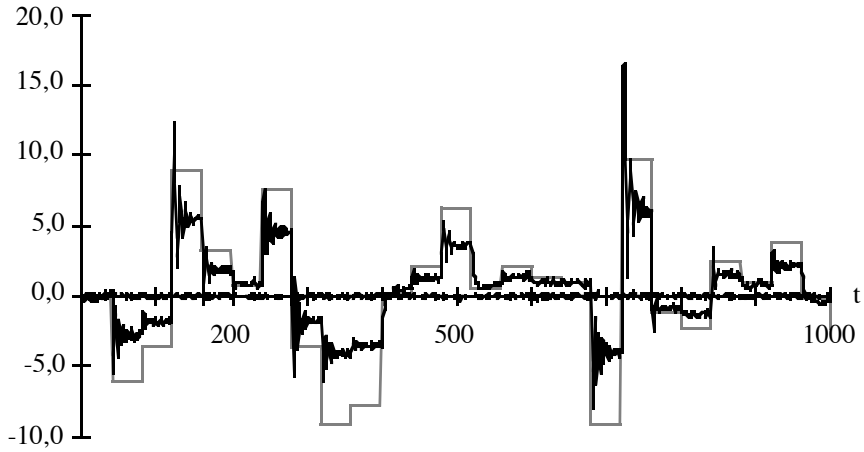


Figure V.5

Séquence de test (var(w(t))≈0,1)

V.3.1.1. Critère d'arrêt de l'apprentissage

Pour choisir un critère d'arrêt, nous avons étudié la sensibilité de l'apprentissage par rapport aux conditions initiales et aux paramètres de l'algorithme d'apprentissage. Dans tous les cas, on effectue d'abord quelques centaines d'itérations du gradient simple, puis on utilise une méthode de quasi-newton (la méthode BFGS, avec optimisation du pas par la méthode de Wolfe et Powell [Minoux 83]).

Il apparaît que l'apprentissage est très sensible aux conditions initiales : pour deux initialisations légèrement différentes des coefficients d'un modèle, l'algorithme peut converger vers des minima locaux tels que le rapport des

EQMA soit dans un rapport supérieur à 10; dans le même temps, pour la plupart des apprentissages, la valeur de l'EQMA après quelques centaines d'itérations est très proche de la valeur de l'EQMA à la convergence (les variations sont souvent inférieures à 1%).

Nous avons donc choisi de limiter le nombre d'itérations à 2000, le gradient simple étant utilisé pendant les 300 premières itérations. L'estimation obtenue est parfois un peu différente de la valeur obtenue à la convergence, mais ce choix permet de multiplier les apprentissages, et donc les chances de trouver un "bon" minimum. De plus, si on le juge nécessaire, l'apprentissage d'un modèle particulièrement intéressant peut être poursuivi jusqu'à son terme.

V.3.1.2. Choix du modèle complet

Les entrées du modèle complet sont les régresseurs sélectionnés lors de la première phase. L'étude des EQMA et EQMV obtenues avec des modèles de complexités différentes (2, 3, 5, 8 et 10 neurones cachés) nous a conduit à choisir un réseau de neurones complètement connecté possèdant 11 neurones (10 neurones cachés à sigmoïde, et un neurone de sortie linéaire). Ce modèle comporte donc 121 coefficients.

Après apprentissage, l'EQMA obtenue avec ce modèle est $9{,}32.10^{-3}$, et l'EQMV est $8{,}12.10^{-2}$. Le rapport entre EQMV et EQMA est de l'ordre de 10, on peut donc supposer qu'il y a surajustement lors de l'apprentissage du modèle, et que celui-ci est plus complexe qu'il n'est nécessaire. Néanmoins, cela est peu gênant tant que ce surajustement est peu important, et nous conserverons ce modèle comme modèle complet.

V.3.1.3. Sélection des régresseurs du modèle

Plusieurs méthodes de sélection ont été utilisées dans cette deuxième phase :

– une procédure de sélection à "modèle complet unique" (MCU) utilisant un test LDRT à 1%,

– une procédure de sélection à "modèles complets multiples" (MCM) utilisant un test LDRT à 1%,

– une méthode de sélection à l'aide d'un critère AIC modifié, pour lequel la valeur k(1) est choisie égale à 4 :

$$AIC_4 = 2\,N \ln(J(\hat{\theta})) + k(1)\,\dim(\theta) = 2\,N \ln(J(\hat{\theta})) + 4\,\dim(\theta)$$

Avec l'ensemble d'apprentissage que nous avons choisi (Figure V.4), ces trois méthodes conduisent à la sélection du modèle dont les régresseur sont $\{y_p(t-1), y_p(t-2), u(t-1)\}$, c'est-à-dire les arguments qui interviennent dans la définition du processus P_1 (Figure V.6).

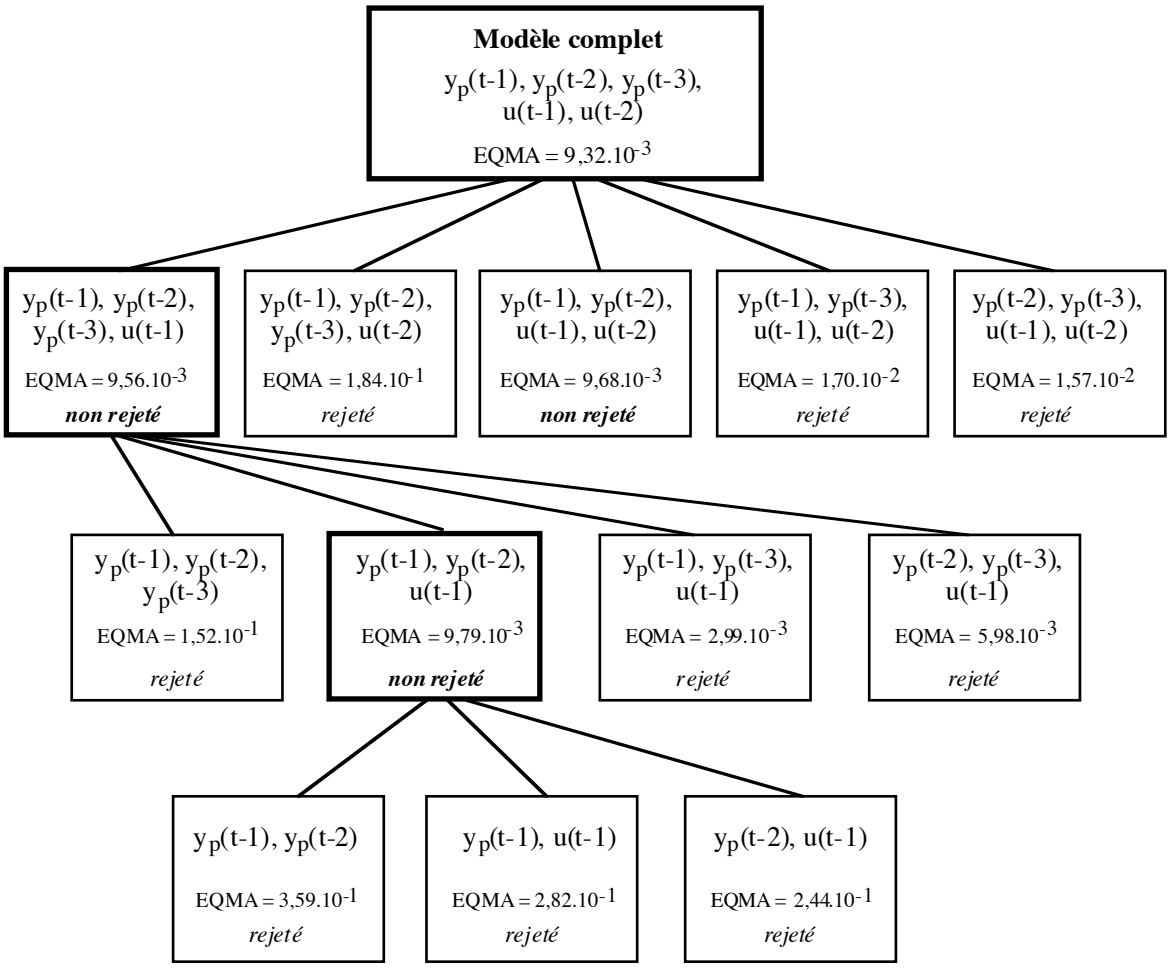


Figure V.6

Sélection des régresseurs d'un modèle neuronal du processus P_1 à l'aide d'un test LDRT

On a donc ainsi éliminé les régresseurs non pertinents, $y_p(t–3)$ et $u(t–2)$, et obtenu un modèle moins complexe que le modèle complet initial (il ne comporte que 99 paramètres). Les performances de ces deux modèles sont tout à fait comparables sur l'ensemble d'apprentissage :

modèle à 5 régresseurs : $EQMA(5) = 9,32.10^{-3}$,

modèle à 3 régresseurs : $EQMA(3) = 9,79.10^{-3}$,

$\hat{var}(w(t)) = 10,8.10^{-3}$.

où $\hat{var}(w(t))$ est l'estimation de la variance du bruit sur l'ensemble d'apprentissage. Sur l'ensemble de validation, l'EQMV obtenue avec le modèle à 3 régresseurs est légèrement meilleure celle obtenue avec le modèle complet. Elle reste cependant assez supérieure à l'EQMA :

$EQMV(5) = 8,12.10^{-2}$,

$EQMV(3) = 6,42.10^{-2}$,

$\hat{var}(w(t)) = 9,84.10^{-3}$.

Neuf autres ensembles d'apprentissage, de même taille que le premier ensemble ont été utilisés. Ils correspondent à un fonctionnement du processus analogue : les séquences de commande sont constituées d'autres suites de créneaux d'amplitude variant entre –10 et +10, et la variance du bruit est de 10^{-2}. Les résultats sont présentés dans le tableau V.7.

	y_1, y_2, u_1	y_1, y_2, y_3, u_1	y_1, y_2, y_3, u_1, u_2
MCU (1%)	7	1	2
MCM (1%)	6	2	2
AIC_4	8	2	0

Tableau V.7

On constate que, dans la majorité des cas, les trois méthodes conduisent à la sélection d'un modèle dont les régresseurs sont les arguments $\{y_p(t–1), y_p(t–2), u(t–1)\}$. Dans tous les cas, ces arguments sont présents dans les modèles sélectionnés. Cependant, certaines sélections conduisent à des modèles plus complexes qu'il n'est nécessaire : suivant les méthodes, 2 à 4 des 10 sélections conduisent à un modèle sur-dimensionné.

V.3.1.4. Modification de la procédure

Il existe, dans toute méthode statistique de sélection de modèle, un risque de choisir un modèle incorrect qui est inhérent à la méthode. Ce risque est fixé à 1% pour les tests d'hypothèse, et la méthode AIC_4 est asymptotiquement équivalente à une sélection à l'aide de tests d'hypothèse avec un risque inférieur à 2%. D'autre part, les propriétés des tests d'hypothèse et des méthodes de sélection de type Akaike, que nous avons présentées dans les chapitres précédents, sont des résultats asymptotiques, qui reposent donc sur l'hypothèse que l'ensemble d'apprentissage est de taille infinie, alors que les ensembles d'apprentissage qui sont utilisés dans toute procédure de sélection de modèles sont, évidemment, de taille finie. Cependant, ces considérations ne peuvent expliquer à elles seules les échecs que nous observons, et il est nécessaire de prendre en considération, d'une part, le choix d'arrêt de l'algorithme d'apprentissage, d'autre part, l'existence de minima locaux et les problèmes de convergence des algorithmes itératifs.

Ainsi, on peut obtenir, suivant les aléas des apprentissages, un modèle pour lequel l'EQMA est supérieure à l'EQMA obtenue avec l'un de ses sous-modèles : l'utilisation de méthodes statistiques pour choisir l'un de ces deux modèles est

alors inutile; inversement, l'apprentissage d'un modèle est parfois particulièrement bon, la valeur de l'EQMA que l'on obtient est alors très inférieure à celle obtenue avec un sous-modèle qui possède théoriquement une structure suffisamment complexe, mais dont l'apprentissage a convergé trop lentement, ou vers un minimum local. Le sous-modèle est alors rejeté à tort.

Afin d'améliorer les performances de la procédure de sélection, nous proposons plusieurs modifications motivées par les constations suivantes :

* Pour tous les processus étudiés, les tests rejettent toujours les modèles, auxquels manque un régresseur important, et les seuls échecs de la procédure concernent la sélection de modèles sur-dimensionnés. Nous proposons donc d'utiliser des méthodes correspondant à des risques faibles, qui favorisent la sélection de modèles peu complexes. Pour cela, nous avons utilisé une méthode de type Akaike, avec une valeur de k(1)=8. On obtient alors une nette amélioration des résultats, puisque toutes les sélections conduisent au choix du modèle ayant comme régresseurs $\{y_p(t–1), y_p(t–2), u(t–1)\}$ (Tableau V.8).

	y_1, y_2, u_1	y_1, y_2, y_3, u_1
AIC_4	8	2
AIC_8	10	0

Tableau V.8

Le choix entre les méthodes utilisant des tests d'hypothèses et les méthodes de type Akaike n'est pas déterminant, puisque dans les deux cas, il est possible de "moduler" le comportement des méthodes, ceci à l'aide d'un seul paramètre (le risque pour les tests statistiques, la valeur de k(1) pour les méthodes d'Akaike). Nous pourrions utiliser un test LDRT avec un risque plus faible pour obtenir les mêmes résultats. Les méthodes de type AIC sont cependant particulièrement simples à "régler", car le choix de la valeur k(1) est simple; on choisit habituellement une valeur comprise entre 4 et 8, les grandes valeurs favorisant la sélection de modèles de faible complexité.

* Les apprentissages sont très sensibles aux conditions initiales, et il est donc souhaitable d'effectuer plusieurs apprentissages avec des conditions initiales différentes pour chaque structure de modèles. On obtient ainsi plusieurs modèles de structures identiques, mais dont les coefficients, et les performances, diffèrent. Nous avons testé deux procédures différentes à partir de cette idée :

– on garde, pour chaque structure, le jeu de paramètres du modèle avec lequel l'EQMA est minimale. La sélection s'effectue alors sur cet ensemble des

"meilleurs" modèles. Le Tableau V.9 présente les résultats, peu satisfaisants, obtenus lorsque l'on effectue trois apprentissages de chaque structure.

	y_1, y_2, u_1	y_1, y_2, u_1, u_2	y_1, y_2, y_3, u_1	y_1, y_2, y_3, u_1, u_2
MCU (1%)	8	1	0	1
MCM (1%)	5	1	3	1
AIC_4	9	0	1	0
AIC_8	10	0	0	0

Tableau V.9

– on effectue plusieurs sélections en changeant les conditions initiales. On sélectionne ainsi plusieurs modèles, et l'on conserve celui dont la complexité est la plus faible. Si plusieurs modèles de complexité équivalente, mais dont les régresseurs sont différents, sont sélectionnés, on choisit celui avec lequel l'EQMA est la plus faible. Nous avons utilisé cette méthode en effectuant trois séries d'apprentissage. On peut constater que cette modification de la procédure conduit à une amélioration sensible des résultats (Tableau V.10).

	y_1, y_2, u_1	y_1, y_2, y_3, u_1
MCU (1%)	9	1
MCM (1%)	9	1
AIC_4	10	0
AIC_8	10	0

Tableau V.10

V.3.2. Conclusion

Les méthodes statistiques de sélection que nous avons mises en œuvre s'avèrent performantes pour sélectionner les régresseurs d'un modèle neuronal d'un processus dynamique non linéaire, même si elles ne conduisent pas systématiquement à la sélection du "meilleur" modèle. De plus, les résultats obtenus peuvent être améliorés par quelques modifications simples de la procédure proposée dans le chapitre IV, qui prennent en considération les problèmes classiques d'apprentissage des modèles neuronaux. Néanmoins, elles ne permettent en aucun cas de s'affranchir complètement de ces problèmes : le choix d'un ensemble d'apprentissage, et, d'autre part, la conception d'algorithmes d'apprentissage qui permettent de converger, de façon systématique et en un temps limité, vers le minimum global, sont des problèmes à part entière.

Il faut par ailleurs souligner que pour un processus réel, qui n'est jamais parfaitement décrit par un modèle mathématique, il n'existe pas d'ensemble de régresseurs "exacts", mais des variables qui, dans le domaine de fonctionnement auquel on s'intéresse, ont une action importante et permettent de construire une représentation du processus satisfaisante. Ce sont ces variables “importantes” que l’on cherche à déterminer.

V.4. Troisième phase : sélection du nombre de neurones du modèle

La liste des régresseurs du modèle étant déterminée lors de la seconde phase, la troisième phase consiste à chercher le nombre minimal de neurones permettant d’obtenir une approximation satisfaisante de la fonction non linéaire φ. Remarquons qu’il n’existe pas de “solution exacte” en ce qui concerne le nombre de neurones que doit comporter le modèle neuronal, puisque le modèle neuronal ne réalise qu’une approximation de φ, qui ne peut généralement pas s’exprimer de façon exacte à l’aide d’un réseau de neurones à sigmoïdes.

V.4.1. Résultats obtenus avec le processus de référence P_1

Nous supposerons que, lors de la seconde phase, on a sélectionné les régresseurs $\{y_p(t–1), y_p(t–2), u(t–1)\}$. On utilise les mêmes ensembles d’apprentissage que précédemment. Malgré la suppression de deux régresseurs (soit 22 paramètres), la comparaison de l’EQMA et l’EQMV du modèle obtenu à la fin de la phase deux (3 régresseurs et une entrée constante, 10 neurones cachés), montre que celui-ci est suffisamment complexe pour être conservé comme modèle complet.

L’ensemble des modèles sur lequel s’effectue la sélection lors de cette troisième phase est assez restreint, puisque le nombre de modèles qu’il contient est égal au nombre de neurones du modèle complet (soit n_n=11 le nombre de neurones du modèle complet, on considère les modèles possédant les mêmes régresseurs et un nombre de neurones variant entre n_n–1 et 1). Il est alors raisonnable de prendre en considération, dès le début de la procédure de sélection, l’ensemble de ces modèles.

Quatre méthodes ont été utilisée pour effectuer la sélection d’un modèle : MCU (1%), MCM (1%), AIC_4 et AIC_8. Pour juger de leur efficacité, nous avons, construit un ensemble de données de très grande taille (N=100000, var(w)) = 10^{-2}). Cet ensemble nous permet d’estimer de façon assez précise la variance de l’erreur de prédiction obtenue avec un modèle particulier, et ainsi de juger ses performances. Il ne doit pas être confondu avec l’ensemble de validation, et n’est

pas utilisé dans la procédure de sélection. On note EQME ("EQM Estimée") l'estimation de la variance de l'erreur calculée avec cet ensemble.

	MCU$_{(1\%)}$	EQME	**MCM**$_{(1\%)}$	EQME	Meilleur	EQME
E_1	**7**	$2{,}61\ 10^{-2}$	**7**	$2{,}61\ 10^{-2}$	**5**	$1{,}24\ 10^{-2}$
E_2	**10**	$6{,}69\ 10^{-2}$	**10**	$6{,}69\ 10^{-2}$	**5**	$1{,}12\ 10^{-2}$
E_3	**10**	$3{,}95\ 10^{-2}$	**10**	$3{,}95\ 10^{-2}$	**9**	$1{,}60\ 10^{-2}$
E_4	**8**	$4{,}26\ 10^{-2}$	**7**	$2{,}22\ 10^{-2}$	**5**	$1{,}74\ 10^{-2}$
E_5	**9**	$4{,}23\ 10^{-2}$	**9**	$4{,}23\ 10^{-2}$	**3**	$1{,}18\ 10^{-2}$
E_6	**10**	$3{,}35\ 10^{-2}$	**10**	$3{,}35\ 10^{-2}$	**3**	$1{,}32\ 10^{-2}$
E_7	**8**	$1{,}51\ 10^{-2}$	**8**	$1{,}51\ 10^{-2}$	**4**	$1{,}27\ 10^{-2}$
E_8	**10**	$6{,}36\ 10^{-2}$	**10**	$6{,}36\ 10^{-2}$	**4**	$1{,}29\ 10^{-2}$
E_9	**8**	$6{,}76\ 10^{-2}$	**8**	$6{,}76\ 10^{-2}$	**3**	$1{,}48\ 10^{-2}$
E_{10}	**8**	$2{,}31\ 10^{-2}$	**8**	$2{,}31\ 10^{-2}$	**3**	$1{,}14\ 10^{-2}$

Tableau V.11

Les résultats des sélections de modèles à l'aide des méthodes MCU et MCM, (avec un test LDRT à 1%), sont présentés dans le tableau V.11 pour 10 ensembles d'apprentissage différents. La première colonne indique quel ensemble d'apprentissage est utilisé; la deuxième et la troisième colonnes, le nombre de neurones cachés et l'EQME du modèle sélectionné par les méthodes MCU et MCM; la dernière colonne indique le nombre de neurones du modèle pour lequel l'EQME est la plus faible, et la valeur de cette EQME.

Les résultats obtenus avec les deux méthodes sont quasiment identiques, et assez peu satisfaisants : pour chaque ensemble d'apprentissage, l'EQME obtenue avec les modèles sélectionnés est sensiblement supérieure à la variance du bruit, et l'on constate surtout qu'il existe des modèles de complexité moindre qui permettent d'obtenir une EQME bien plus proche de la variance du bruit. Les résultats obtenus avec un test LDRT à 1‰ sont peu différents de ceux présentés dans le tableau V.11. On peut évidemment optimiser la valeur du risque pour obtenir des résultats plus satisfaisants, mais cette démarche ne peut être faite dans le cadre de l'étude d'un processus réel, où l'on ne connaît pas la variance du bruit.

	AIC_4 EQMV / EQME	**AIC_8** EQMV / EQME	Meilleur EQME
E_1	**5** $2{,}85\ 10^{-2}$ / $1{,}24\ 10^{-2}$	**5** $2{,}85\ 10^{-2}$ / $1{,}24\ 10^{-2}$	**5**
E_2	**5** $1{,}22\ 10^{-2}$ / $1{,}12\ 10^{-2}$	**5** $1{,}22\ 10^{-2}$ / $1{,}12\ 10^{-2}$	**5**
E_3	**4** $9{,}07\ 10^{-1}$ / $1{,}92$	**4** $9{,}07\ 10^{-1}$ / $1{,}92$	**9** $1{,}60\ 10^{-2}$
E_4	**5** $5{,}34\ 10^{-2}$ / $1{,}76\ 10^{-2}$	**5** $5{,}34\ 10^{-2}$ / $1{,}76\ 10^{-2}$	**5**
E_5	**3** $1{,}41\ 10^{-2}$ / $1{,}18\ 10^{-2}$	**3** $1{,}41\ 10^{-2}$ / $1{,}18\ 10^{-2}$	**3**
E_6	**3** $1{,}58\ 10^{-2}$ / $1{,}32\ 10^{-2}$	**3** $1{,}58\ 10^{-2}$ / $1{,}32\ 10^{-2}$	**3**
E_7	**4** $1{,}15\ 10^{-2}$ / $1{,}27\ 10^{-2}$	**4** $1{,}15\ 10^{-2}$ / $1{,}27\ 10^{-2}$	**4**
E_8	**4** $1{,}43\ 10^{-2}$ / $1{,}29\ 10^{-2}$	**4** $1{,}43\ 10^{-2}$ / $1{,}29\ 10^{-2}$	**4**
E_9	**4** $1{,}53\ 10^{-2}$ / $1{,}48\ 10^{-2}$	**3** $1{,}06\ 10^{-2}$ / $1{,}05\ 10^{-2}$	**3**
E_{10}	**3** $1{,}44\ 10^{-2}$ / $1{,}14\ 10^{-2}$	**3** $1{,}44\ 10^{-2}$ / $1{,}14\ 10^{-2}$	**3**

Tableau V.12

Le tableau V.12 présente les résultats obtenus lorsque l'on utilise les critères AIC_4 et AIC_8. Aux informations présentes dans les colonnes 2 et 3 du tableau V.11 (nombre de neurones cachés et EQME du modèle sélectionné), nous avons ajouté la valeur de l'EQMV (l'ensemble de validation est le même que dans la deuxième phase, figure V.7). Ces résultats sont très satisfaisants : à l'exception de la sélection correspondant à l'ensemble d'apprentissage E_3, les modèles sélectionnés avec AIC_8 sont ceux avec lesquels l'EQME est minimale, et le plus souvent très proche de la variance du bruit. Les résultats obtenus avec le critère AIC_4 sont identiques pour 9 des 10 ensembles d'apprentissage (pour l'ensemble E_9 cependant, le critère AIC_4 conduit à la sélection du modèle à 4 neurones cachés, le critère AIC_8 à la sélection du modèle possédant 3 neurones cachés, qui est le meilleur des deux). Les valeurs des EQMV sont également très proches de la variance du bruit (qui varie, suivant les ensembles d'apprentissages, de $9{,}6.10^{-3}$ à $1{,}02.10^{-2}$).

En ce qui concerne le modèle sélectionné pour l'ensemble d'apprentissage E_3, l'EQMV et l'EQME sont environ 100 à 200 fois supérieures à l'EQMA : l'apprentissage a visiblement convergé vers un minimum local, et il a un surajustement très important. Dans un tel cas, le modèle sélectionné ne peut être accepté, et un nouvel apprentissage doit être effectué. De plus, lorsque l'on boucle ce modèle et qu'on le commande avec la séquence utilisée pour construire l'ensemble de validation, on constate qu'il est instable. Lorsqu'on l'on boucle les

modèles sélectionnés pour les autres ensembles d'apprentissage, l'EQM sur l'ensemble de données de grande taille varie de 1.10^{-2} à 5.10^{-2} environ, ce qui est très satisfaisant si l'on considère que l'apprentissage est fait avec les modèles non bouclés.

Les méthodes de sélection fondées sur les critères AIC_4 et AIC_8 sont des outils performants pour la recherche, dans un ensemble de modèles, du modèle qui réalise le meilleur compromis entre performance et parcimonie. Cependant, l'objectif de cette troisième phase est de trouver la structure la mieux adaptée pour modéliser le processus que l'on étudie, dans un domaine de fonctionnement donné. Or, on constate dans le tableau V.12 que les structures choisies sont de complexité différentes (de 3 à 5 neurones cachés, soit respectivement 22 et 39 paramètres) suivant l'ensemble d'apprentissage que l'on considère. Deux facteurs entrent en jeu :

- suivant la séquence de commande que l'on applique au processus et la réalisation particulière du bruit, tous les ensembles d'apprentissage ne sont pas identiquement représentatifs du processus : certaines zones de fonctionnement peuvent être explorées plus que d'autres, et la structure "optimale" n'est pas de complexité identique suivant l'ensemble d'apprentissage qui est utilisé. Notons que la complexité d'un modèle varie également en fonction de la variance du bruit qui perturbe le processus : plus elle est importante, plus le modèle sélectionné est simple. Dans le cas qui nous concerne, les variations de la variance du bruit d'un ensemble d'apprentissage à l'autre sont peu importantes, et ne semblent pas influer sur la complexité du modèle choisi;
- l'apprentissage converge quelquefois vers un minimum local, ou est stoppé prématurément à cause du critère d'arrêt que l'on utilise.

Comme lors de la seconde phase, plusieurs apprentissages de chaque structure concurrente sont effectués pour chaque ensemble d'apprentissage. Lorsqu'un modèle est instable, il est éliminé, et un nouvel apprentissage est effectué. Les mêmes méthodes que dans la phase précédente sont utilisées pour choisir un modèle :

- on construit un ensemble de modèles en choisissant, pour chaque structure que l'on considère, le modèle pour lequel l'EQMA est minimale; on sélectionne alors un modèle dans cet ensemble;
- on effectue plusieurs sélections, et l'on conserve, parmi les modèles sélectionnés, celui dont la structure est la moins complexe.

	AIC_4	EQME	AIC_8	EQME	Meilleur	EQME
E_1	**4**	$1{,}20\ 10^{-2}$	**4**	$1{,}20\ 10^{-2}$	**4**	
E_2	**3**	$1{,}09\ 10^{-2}$	**3**	$1{,}09\ 10^{-2}$	**3**	
E_3	**3**	$1{,}16\ 10^{-2}$	**3**	$1{,}16\ 10^{-2}$	**3**	
E_4	**4**	$1{,}74\ 10^{-2}$	**4**	$1{,}74\ 10^{-2}$	**4**	
E_5	**3**	$1{,}18\ 10^{-2}$	**3**	$1{,}18\ 10^{-2}$	**3**	
E_6	**3**	$1{,}32\ 10^{-2}$	**3**	$1{,}32\ 10^{-2}$	**3**	
E_7	**4**	$1{,}27\ 10^{-2}$	**4**	$1{,}27\ 10^{-2}$	**5**	$1{,}18\ 10^{-2}$
E_8	**3**	$1{,}13\ 10^{-2}$	**3**	$1{,}13\ 10^{-2}$	**4**	$1{,}07\ 10^{-2}$
E_9	**4**	$1{,}48\ 10^{-2}$	**3**	$1{,}05\ 10^{-2}$	**3**	
E_{10}	**3**	$1{,}14\ 10^{-2}$	**3**	$1{,}14\ 10^{-2}$	**3**	

Tableau V.13

Pour la série d'apprentissages que nous avons effectués, les résultats sont identiques avec les deux méthodes (Tableau V.13). Dans la majorité des cas, les modèles sélectionnés sont ceux pour lequel l'EQME est minimale, et dans tous les cas, l'EQME est très proche de l'EQMA, qui est par ailleurs une excellente estimation de la variance du bruit sur l'ensemble d'apprentissage. De plus, il n'apparaît plus que des modèles contenant 3 ou 4 neurones cachés : en multipliant les apprentissages, on minimise l'effet des minima locaux et des problèmes de convergence, et l'on sélectionne de moins en moins souvent des modèles sur-dimensionnés (pour un grand nombre d'apprentissages, on ne sélectionne plus que des modèles à 3 neurones cachés).

Lorsque la structure du modèle est ainsi déterminée, on peut alors effectuer de nombreux apprentissages, sans restriction sur le nombre d'itérations et sans critère d'arrêt particulier, ceci afin de converger vers le minimum global, et de déterminer le meilleur jeu de coefficients. De plus, on peut encore chercher à optimiser la structure du modèle en supprimant les coefficients qui ne sont pas utiles, en utilisant par exemple la méthode OBD [Le Cun 90].

V.5. Conclusion

Dans ce chapitre, nous avons mis en œuvre la procédure de sélection de modèles proposée dans le chapitre IV sur différents processus simulés analytiquement. Nous nous sommes heurtés à certaines difficultés lors de cette mise en œuvre, qui tiennent essentiellement à deux facteurs :

– l'utilisation d'ensembles d'apprentissage de tailles limitées, qui ne sont pas toujours suffisamment représentatifs du comportement du processus dans le domaine de fonctionnement que nous avons choisi d'étudier,

– les problèmes de convergence vers le minimum global, causés notamment par l'existence de minima locaux, et par les limitations intrinsèques (ou imposées par l'utilisateur) des méthodes d'apprentissage des réseaux de neurones.

Nous avons proposé des solutions pragmatiques et simples qui permettent d'améliorer les performances de la procédure de sélection, sans toutefois s'affranchir complètement de ces problèmes. Ceci est illustré à l'aide d'exemples de simulations, qui ont permis de mettre en évidence l'intérêt des tests d'hypothèses statistiques pour la sélection de modèles.

Cependant, la procédure que nous avons proposée peut être mise en œuvre de façon différente suivante les possibilités d'action dont on dispose et les contraintes auxquelles on est soumis. Considérons deux configurations "extrêmes" :

– dans le premier cas, l'expérimentateur a la possibilité de faire de nombreuses expériences sur le processus qu'il étudie, et peut donc construire autant d'ensembles d'apprentissage et de validation qu'il lui semble nécessaire. Il peut alors effectuer la première phase en construisant des ensembles d'apprentissage correspondant à des comportements locaux du processus, et en sélectionnant les régresseurs de modèles locaux, linéaires par rapport aux paramètres, à l'aide de la procédure proposée dans le chapitre IV. Les modèles complets choisis lors de cette première phase pourront être de très grande taille, linéaires ou polynomiaux. S'ils sont polynomiaux, il est alors nécessaire de définir une procédure qui limite le nombre de monômes intervenant dans la sélection, et permette de sélectionner de façon itérative des monômes de degré de plus en plus élevé.
Pour les deuxième et troisième phases, les méthodes de sélection statistiques sont d'autant plus performantes que la taille des ensembles d'apprentissage est grande.
De plus, des méthodes de validation croisée peuvent être utilisées conjointement aux méthodes statistiques de sélection, pour confirmer la

sélection d'un modèle ou, au contraire, l'infirmer. La procédure de sélection ne repose alors plus uniquement sur les résultats des méthodes statistiques.

– l'expérimentateur dispose de peu de données, et ne peut en aucun cas procéder à des expériences correspondant à des fonctionnements locaux, non représentatifs du comportement classique du processus. On peut supposer, par exemple, qu'il dispose de suffisamment de données pour construire deux ensembles de tailles limitées, un ensemble d'apprentissage et un ensemble de validation.

Deux démarches sont alors envisageables. La première consiste à utiliser un modèle polynomial comme modèle global du processus dans la première phase, mais le nombre de paramètres du modèle doit rester peu élevé devant la taille de l'ensemble d'apprentissage. Il faut alors limiter le nombre des entrées du modèle et le degré maximum des monômes. La seconde possibilité est de restreindre plus ou moins arbitrairement le nombre des entrées du modèle, et, sans effectuer la première phase, de sélectionner l'architecture d'un modèle neuronal. Les données d'apprentissage étant peu nombreuses, les méthodes de sélection statistiques sont alors des outils performants pour la sélection d'un modèle qui soit une très bonne approximation du processus tout en ayant une structure aussi parcimonieuse que possible.

Conclusion

Le problème de la sélection de modèles neuronaux est important à plusieurs titres : d'une part, il est essentiel, pour faciliter l'implantation matérielle de ces modèles, que ceux-ci soient aussi compacts que possible, en termes de nombres d'entrées comme de neurones cachés ; d'autre part, l'objectif d'une modélisation est de trouver le modèle qui allie *parcimonie* et *performance* (celle-ci reflétant la capacité du modèle à reproduire le comportement du processus, aussi bien sur l'ensemble d'apprentissage que sur des données non apprises). On se trouve donc dans une situation où les exigences de performance comme les exigences de facilité d'implantation dirigent le concepteur vers le même objectif : l'obtention de modèles parcimonieux.

Le problème de la sélection de modèles *linéaires* constitue un chapitre important des statistiques ; pour les modèles *non linéaires*, et notamment pour les réseaux de neurones, ce problème était encore ouvert : il convenait donc de chercher des méthodes applicables dans ce cadre. Dans ce travail, nous nous sommes attachés à la modélisation de processus dynamiques, mais une grande partie des résultats que nous présentons peuvent être appliqués à des processus statiques.

Nous avons consacré les trois premiers chapitres de ce mémoire à la présentation du problème de la sélection de modèles de processus dynamiques. Le problème de la modélisation de processus dynamiques est présenté dans le chapitre I, où nous montrons le lien qui existe entre les hypothèses sur le comportement du processus, synthétisées par le modèle, et le système d'apprentissage qu'il est nécessaire de mettre en œuvre. Dans le chapitre II, les méthodes classiques d'estimation des paramètres des modèles linéaires et non linéaires ont été présentées. La sélection de modèles a été abordée dans le chapitre III. Nous avons en particulier présenté plusieurs méthodes de sélection classiques, et justifié leur utilisation en montrant leur lien avec la méthode statistique du maximum de vraisemblance.
Dans le chapitre IV, une procédure de sélection d'une classe particulière de modèles dynamiques non linéaires, les modèles NARX, a été proposée. Elle a été mise en œuvre dans le chapitre V sur des processus simulés. Certaines limitations ont été mises en évidence, et nous avons proposé des

modifications pragmatiques qui nous permettent de surmonter les difficultés rencontrées.

Le présent travail entre dans le cadre général d'un effort, mené au laboratoire depuis plusieurs années, pour mettre les méthodes dites "neuronales" dans la perspective des méthodes classiques, et pour progresser vers une utilisation optimale des réseaux de neurones dans le cadre de la modélisation de processus dynamiques non linéaires. En amont du présent travail se trouvent :

- la constitution de l'ensemble d'apprentissage (lorsque celui-ci n'est pas imposé par les données disponibles) ;
- le choix du type de modélisation : modélisation "boîte noire" ou modélisation "boîte grise" (modèle neuronal de connaissances) ; dans le présent travail, nous nous sommes placés dans le cadre d'une modélisation "boîte noire" ;
- le choix de la structure du prédicteur : prédicteur entrée-sortie (NARX, NARMAX, Output Error, ...) ou prédicteur d'état; nous avons traité le cas de modèles NARX ;
- la mise au point d'algorithmes d'optimisation performants.

En aval de cette étude se trouve l'évaluation des performances du modèle dans le cadre de l'application pour laquelle il est conçu, car les exigences de performances peuvent être très différentes selon que le modèle doit être utilisé comme simulateur, comme modèle interne dans un système de commande, etc.

Dans cette chaîne de choix, on voit que de nombreux problèmes restent ouverts à l'heure actuelle :

- l'optimisation de l'ensemble d'apprentissage,
- la systématisation de la modélisation "boîte grise", c'est-à-dire de l'introduction dans le modèle des connaissances *a priori,* exprimées sous forme mathématique,
- l'extension des techniques présentées dans ce travail à d'autres modèles que les modèles NARX.

Le présent travail représente donc un pas important vers la conception d'un générateur automatique de modèles non linéaires - ou simplement vers un outil d'aide à la conception de modèles non linéaires - bien que de nombreux problèmes restent encore ouverts.

Bibliographie

[Akaike 69]
H. AKAIKE
"Fitting autoregressive models for prediction"
Ann. Inst. Stat. Math., vol. 21, pp. 243-347, 1969

[Akaike 74a]
H. AKAIKE
"A new look at the statistical model identification"
IEEE Transactions on Automatic Control, vol. 19, pp. 716-723, 1974

[Akaike 74b]
H. AKAIKE
"Stochastic theory of minimal realization"
IEEE Transactions on Automatic Control, vol. 19, pp. 667-674, 1974

[Åström 65]
K.J. ÅSTRÖM, T. BOHLIN
"Numerical identification of linear dynamic systems from normal operating records"
IFAC Symposium on Self-adaptive Systems, Teddington, Engalnd, 1965

[Balakrishnan 68]
A.V. BALAKRISHNAN
"Stochastic system identification techniques"
dans M.F. KARREMAN (Ed.) *"Stochastic optimization and control"* John Wiley, New York, 1968

[Bhansali 77]
R.J. BHANSALI, D.Y. DOWNHAM
Biometrica, vol. 64, p. 547, 1977

[Billings 85]
S.A.BILLINGS, M.B. FADZILL
"The practical identification of systems with nonlinearities"
IFAC Identification and System Parameter Estimation 1985, York, UK, 1985.

[Billings 88]
S.A. BILLINGS, M.J. KORENBERG, S. CHEN
"Identification of non-linear output-affine systems using orthogonal least-squares algorithm"
International Journal of Systems Science, vol. 19, 1559-1568, 1988

[Billings 89]
S.A. BILLINGS, S. CHEN, M.J. KORENBERG
"Identification of MIMO nonlinear systems using a forward-regression orthogonal estimator"
International Journal of Control, vol. 49, 2157-2189, 1989

[Bohlin 78]
T. BOHLIN
"Maximum-power validation without higher order fitting"
Automatica, vol. 17, pp. 137-146, 1978

[Broyden 70]
C.G. BROYDEN
" The convergence of a class of double-rank minization algorithms 2: the new algorithm"
Journal Institute of Mathematics and its Applications 6, pp. 222-231, 1970.

[Caines 74]
P.E. CAINES, J. RISSANEN
"Maximum likelihood estimation in multivariable gaussian stochastic processes"
IEEE Transactions on automatic control, vol. 21, pp. 500-505, 1974

[Chen 89a]
S. CHEN, S.A. BILLINGS, W. LUO
"Orthogonal least squares methods and thier application to non-linear system identification"
International Journal of Control, 1989, vol. 50, no. 5, 1873-1896, 1989

[Cramer 46]
H. CRAMER
"Mathematical method of statistics"
Princeton University Press, Princeton, N.J., 1946

[Draper 81]
N.R. DRAPER, H. SMITH
"Applied Regression Analysis"
Wiley, New York, 1981.

[Fisher 1912]
R.A. FISHER
"On an absolute criterion for fitting frequency curves"
Mess. of Math., n° 41, p. 155, 1912.

[Fisher 1921]
R.A. FISHER
"On the mathematics foundations of theorical statistics"
Phil. Trans., A-222, 309, 1921

[Fletcher 70]
R. FLETCHER
"A new approach to variable metric algorithms"
The Computer Journal, vol. 13, n°3, pp. 317-322, 1970.

[Goldfarb 70]
D. GOLDFARB
"A family of variable metric methods derived by variational means"
Mathematics of Computation 24, pp. 23-26, 1970

[Goodwin 77]
G.C. GOODWIN, R.L.PAYNE
"Dynamic System Identification : Experiment Design and Data Analysis"
Mathematics in Science and Engineering, Volume 136, Academic Press, 1977

[Haber 85]
R. HABER
"Nonlinearity tests for dynamic processes"
IFAC Identification and System Parameter Estimation 1985, York, UK, 1985.

[Hornik 89]
K. HORNIK, M. STINCHCOMBE, H. WHITE
"Multilayer feedforward network are universal approximates"
Neural Networks, vol. 2, pp. 359-366, 1989

[Hornik 94]
K. HORNIK, M. STINCHCOMBE, H. WHITE, P. AUER
"Degree of Approximation Results for Feedforward Networks Approximating Unknow Mappings and Their Derivates"
Neural Computation, vol. 6, p. 1262, 1994

[Korenberg 85]
M.J. KORENBERG
"Orthogonal identification of nonlinear difference equation models"
Mid. West. Symposium on Circuits and Systems, Louisville, 1985

[Korenberg 85]
M.J. KORENBERG, S.A. BILLINGS, Y.P. LIU, P.J. McILOY
"Orthogonal parameter estimation for non-linear stochastic systems"
International Journal of Control, vol. 48, 193-210, 1988

[Le Cun 90]
Y. LE CUN, J.S. DENKER, S.A. SOLLA
"Optimal Brain Damage"
IEEE Advances on Neural Information Processing (Denver 89), D.S. TOURETZKY (Ed.), pp. 598-605, 1990

[Leontaritis 85]
I.J. LEONTARITIS, S.A. BILLINGS
"Input-output parametric models for non-linear systems–Part 1: Deterministic non-linear systems; Part 2 : Stochastic non-linear systems"
International Journal of Control, 1985, vol. 41, 311-341, 1985

[Leontaritis 87]
I.J. LEONTARITIS, S.A. BILLINGS
"Model selection and validation methods for non-linear systems"
Int. Journal of Control, vol. 45, n°1, pp. 311-341, 1987

[Ljung 74]
L. LJUNG
"On consistency for prediction error identification methods"
Report 7405, Div. Auto. Control, Lund Institute of Technology, 1974

[Ljung 76a]
L. LJUNG
"On the consistency of a prediction error identification methods"
dans, R.H. MERHA, D.G. LAINIOTIS (Eds.), *"Systems identification advances and Case study"*, Academic Press, New York, 1976

[Ljung 76b]
L. LJUNG
"On consistency and identifiability"
Mathematical programming study, n°5, pp. 169-190, North-Holland, 1976

[Ljung 78]
L. LJUNG
"Convergence analysis of parametric identification methods"
IEEE Transactions on Automatic Control, vol. 23, pp. 770-783, 1978

[Ljung 79]
L. LJUNG, P.E. CAINES
"Asymptotic normality of prediction error estimators for approximate systems models"
Stochastics, n° 3, pp. 29-46, 1979

[Ljung 87]
L. LJUNG
"System identification : theory for the user"
Prentice Hall, Englewood Cliffs, New Jersey, 1987

[MacKay 92a]
D.J.C. MACKAY
"Bayesian interpolation"
Neural Computation, 4(3), pp. 415-447, 1992

[MacKay 92b]
D.J.C. MACKAY
"A practical Bayesian framework for backdrop networks"
Neural Computation, 4(3), pp. 448-472, 1992

[Minoux 83]
M. MINOUX
"Programmation Mathématique, Théorie et Algorithmes"
Tome 1, Ed. Dunod, 1983

[Moody 94]
J. MOODY
"Prediction Risk and Architecture Selection for Neural Networks"
dans *"From Statistics to Neural Networks : Theory and Pattern Recognition Applications"*, Eds. V. Cherkassky, J.H. Friedmann, H. Wechsler, NATO ASI Series F, Springer-verlag, 1994.

[Narendra 90]
K.S. NARENDRA, K. PARTHASARATHY
"Identification and Control of Dynamicals Systems Using Neural Networks"
IEEE Transactions on Neural Networks, vol. 1, 4-27, 1990

[Nash 90]
J.C. NASH
"Compact Numerical Methods for Computers: linear algebra and function minimization"
Ed. Adam Hilger, 1990

[Nerrand 92a]
O. NERRAND
"Réseaux de neurones pour le filtrage adaptatif, l'identification et la commande de processus"
Thèse de doctorat de l'Université Pierre et Marie Curie-Paris VI, 1992

[Nerrand 92b]
O. NERRAND, P. ROUSSEL-RAGOT, L. PERSONNAZ, G. DREYFUS, S. MARCOS
"Neural Network and non-linear adaptive filtering : unifying concepts and new algorithms"
Neural Computation, vol 5, no. 2, 1992

[Powell 76]
M.J.D. POWELL
"Some global convergence properties of a variable metric algorithm for minimization without exact line searches"
dans, *Nonlinear Programming*, SIAM-AMS Proceedings 9, R. W. Cottel & C.E. Lemke, Eds., Providence R.I., 1976

[Press 92]
W.H. PRESS, S.A. TEUKOLSKY, W.T. VETTERLING, B.P. FLANNERY
"Numerical Recipies in C : The Art of Computing"
Second Edition, Cambridge University Press, 1992

[Reed 93]
R. REED
"Pruning algorithm - a survey"
IEEE Transactions on Neural Networks, vol. 4, n° 5, september 1993

[Rivals 81]
I. RIVALS
"Modélisation et commande de processus par réseaux de neurones : application au pilotage d'un véhicule autonome"
Thèse de doctorat de l'Université Pierre et Marie Curie-Paris VI, 1995

[Rumelhart 86]
D. RUMELHART, G. HINTON, R. WILLIAMS
"Learning Internal Representations by Error Propagation"

Parallel Distributed Procesing, MIT Press, 1986
[Shanno 69]
D.F. SHANNO
"Conditioning of quasi-newton methods for function minimization"
Mathematics of Computation 24, pp. 641-656, 1969

[Shibata 76]
R. SHIBATA
"Selection of an autoregressive model by Akaike's Information Criteria"
Biometrica, vol. 63, pp. 117-126, 1976

[Söderström 77]
T. SÖDERSTRÖM
"On model structure testing in system identification"
International Journal of Control, vol. 26, pp. 1-18, 1977

[Stone 77a]
M. STONE
"An asymptotic equivalence of choice of model by cross-validation and Akaike Criterion"
Journal of Royal Statistics Society, ser. B, vol. 39, pp. 44-47, 1977

[Stone 77b]
M. STONE
"Asymptotics for and against cross-validation"
Biometrika, vol. 64, pp. 29-35, 1977

[Wald 49]
A. WALD
"Note on the consistency of the maximum likelihood estimate"
Ann. Math. Statis., vol. 20, pp. 595-601, 1949

[Williams 95]
P.M. WILLIAMS
"Bayesian Regularization and Pruning Using a Laplace Prior"
Neural Computation, 7, pp. 117-145, 1995.

[Wolfe 69]
P. WOLFE
"Convergence conditions for ascent methods"
S.I.A.M. Review 11, pp. 226-235, 1969

ADAPTIVE TRAINING OF FEEDBACK NEURAL NETWORKS FOR NON-LINEAR FILTERING

G. Dreyfus*, O. Macchi**, S. Marcos**, O. Nerrand*, L. Personnaz*,
P. Roussel-Ragot*, D. Urbani*, C. Vignat**

*Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris
10, rue Vauquelin
75005 PARIS - FRANCE

**Laboratoire des Signaux et Systèmes
Ecole Supérieure d'Electricité, Plateau de Moulon
91192 GIF SUR YVETTE - FRANCE

Abstract. The paper proposes a general framework which encompasses the training of neural networks and the adaptation of filters. It is shown that neural networks can be considered as general non-linear filters which can be trained adaptively, i.e. which can undergo continual training. A unified view of gradient-based training algorithms for feedback networks is proposed, which gives rise to new algorithms. The use of some of these algorithms is illustrated by examples of non-linear adaptive filtering and process identification.

INTRODUCTION

In recent papers [1, 2], a general framework, which encompasses algorithms used for the training of neural networks and algorithms used for the adaptation of filters, has been proposed. Specifically, it was shown that neural networks can be used *adaptively*, i.e. can undergo *continual* training with a possibly *infinite* number of *time-ordered* examples - in contradistinction to the traditional training of neural networks with a *finite number* of examples presented in an *arbitrary order;* therefore, neural networks can be regarded as a class of non-linear adaptive filters, either transversal or recursive, which are quite general because of the ability of neural nets to approximate non-linear functions. It was further shown that algorithms which can be used for the adaptive training of feedback neural networks fall into three broad classes; these classes include, as special instances, the methods which have been proposed in the recent past for training neural networks adaptively, as well as algorithms which have been in current use in linear adaptive filtering and control. This framework will be summarized briefly in the first part of the paper.

In addition, this general approach leads to new algorithms. The second part of the paper shows illustrative examples of the application of the latter to problems in adaptive filtering and identification.

ADAPTIVE TRAINING OF FEEDBACK NEURAL NETS FOR NON-LINEAR FILTERING

Network

A neural network architecture of the type shown on Figure 1, featuring M external inputs, N feedback inputs and one output, can implement a fairly large class of non-linear functions; the most general form for the feedforward part is a *fully-connected net*. The basic building block of the network is a "neuron", which performs a weighted sum of its inputs and computes an "activation function" f - usually non linear - of the weighted sum:

$z_i = f_i[v_i]$ with $v_i = \sum_j C_{ij}\, x_j$

where z_i denotes the output of neuron i, and x_j denotes the j-th input of neuron i; x_j may be an external input, a state input, or the output of another neuron.

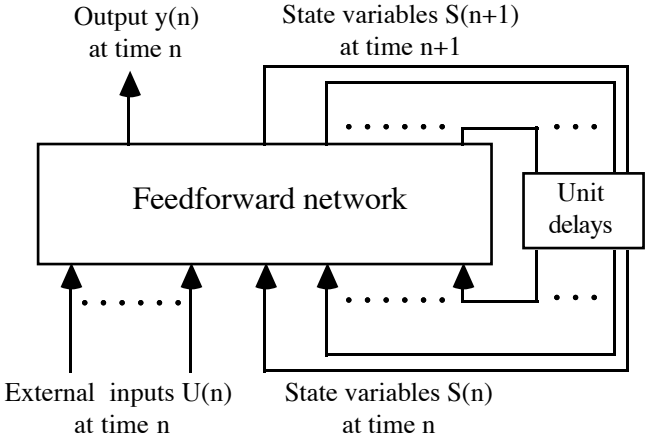


Figure 1.

If the external inputs consist of the values U(n)={u(n),u(n-1),....,u(n-M+1)} of a signal u at successive instants of time, the network may be viewed as a general non-linear recursive filter.

The task of the network is determined by a (possibly infinite) set of inputs and corresponding desired outputs. At each sampling time n, an error e(n) is defined as the difference between the desired output d(n) and the actual output of the network y(n): e(n)=d(n)-y(n). The network adaptation algorithms aim at

finding the synaptic coefficients which minimize a given satisfaction criterion involving, usually, the squared error $e(n)^2$ [3].
Thus, it is clear that adaptive filters and neural networks are formally equivalent, and that neural networks, which are potentially capable of realizing *non-linear* input-output relations, are simple generalizations of linear filters. In the next section, we put into perspective the training algorithms developed for discrete-time feedback neural networks and the algorithms used classically in adaptive filtering.

General presentation of the algorithms

The present paper focusses on gradient-based methods using a sliding window of length N_c, whereby the updating of the synaptic coefficients is given, at time n, by

$$\Delta c_{ij}(n) = - \mu \left[\frac{\partial}{\partial c_{ij}} \left(\frac{1}{2} \sum_{k=n-N_c+1}^{n} e(k)^2 \right) \right]_{C(n-1)} \qquad (1)$$

where μ is the gradient step.
The choice of N_c depends on several factors, including the typical time scale of the non-stationarity of the signals.
For the computation of the gradient to be meaningful, the coefficients must be considered as being constant on a window of length $N_t \geq N_c$. Thus, *for the updating at time n,* the N_c errors $\{e(k)\}$ and their partial derivatives, appearing in relation (1), must be computed from N_t *computational blocks,* corresponding to the last N_t sampling times; the values of the coefficients used for all N_t blocks are the coefficients C(n-1) which were updated at time n-1. We denote by $S_{in}^{m}(n)$ the value of the state input of block m at time n and by $S_{out}^{m}(n)$ the state output.
The choice of the state inputs and of their partial derivatives, as inputs of each block, gives rise to a variety of algorithms. These algorithms fall into three categories depending on the choice of the state inputs:
(i) **directed algorithms**, in which the state inputs are taken equal to their desired values, for all blocks;
(ii) **semi-directed algorithms**, in which the state inputs of the first block at time n are taken equal to their desired values, and in which the state inputs of the other blocks are taken equal to the state outputs of the previous block,
(iii) **undirected algorithms**, in which the state inputs of the first block at time n are taken equal to the corresponding states computed at time n-1, and in which the state inputs of the other blocks are taken equal to the state outputs of the previous block.
Directed and semi-directed algorithms can be used only if all state variables have desired values, as is the case for NARMAX models [4]. If some, but not all, state inputs do not have desired values, **hybrid** versions of the above algorithms can be used: those state inputs for which no desired values are available are taken equal to the corresponding computed state variables (as in an undirected algorithm), whereas the other state inputs may be taken equal to their desired values (as in a directed or semi-directed algorithm).

In each category, three algorithms are defined, depending on the choice of the partial derivatives of the state inputs. This is summarized in Table 1.

Algorithm	$S^1_{in}(n)$	$S^m_{in}(n)$	$\frac{\partial S^1_{in}}{\partial c_{ij}}(n)$	$\frac{\partial S^m_{in}}{\partial c_{ij}}(n)$
Directed (D)	Des. val.	Des. val.	zero	zero
D-SD	Des. val.	Des. val.	zero	$\frac{\partial S^{m-1}_{out}}{\partial c_{ij}}(n)$
D-UD	Des. val.	Des. val.	$\frac{\partial S^1_{out}}{\partial c_{ij}}(n-1)$	$\frac{\partial S^{m-1}_{out}}{\partial c_{ij}}(n)$
Semi-Directed (SD)	Des. val.	$S^{m-1}_{out}(n)$	zero	$\frac{\partial S^{m-1}_{out}}{\partial c_{ij}}(n)$
SD-D	Des. val.	$S^{m-1}_{out}(n)$	zero	zero
SD-UD	Des. val.	$S^{m-1}_{out}(n)$	$\frac{\partial S^1_{out}}{\partial c_{ij}}(n-1)$	$\frac{\partial S^{m-1}_{out}}{\partial c_{ij}}(n)$
Undirected (UD)	$S^1_{out}(n-1)$	$S^{m-1}_{out}(n)$	$\frac{\partial S^1_{out}}{\partial c_{ij}}(n-1)$	$\frac{\partial S^{m-1}_{out}}{\partial c_{ij}}(n)$
UD-D	$S^1_{out}(n-1)$	$S^{m-1}_{out}(n)$	zero	zero
UD-SD	$S^1_{out}(n-1)$	$S^{m-1}_{out}(n)$	zero	$\frac{\partial S^{m-1}_{out}}{\partial c_{ij}}(n)$

Table 1.
Summary of algorithms.
Des. val. = desired value

Relations with known algorithms for neural nets and for adaptive filtering

Some of the above algorithms have been proposed independently in the field of neural nets and in the field of signal processing, under different names.
Two approaches have been used in order to adapt linear recursive filters: the *equation-error* formulation and the *output-error* formulation. In the equation-error formulation (also termed series-parallel in the control literature), the recursive nature of the filter is not taken into account: thus, directed algorithms generalize the equation-error approach; they generate stable adaptation behaviours. The "*Teacher Forcing*" algorithm [5] is based on the same idea. On the other hand, the output-error formulation takes into account the recursive form of the filter during adaptation: thus, undirected algorithms generalize the output-error approach. The stability of these algorithms is not easy to predict.
For instance, the "*Recursive Prediction Error (RPE)*" algorithm [6], used in linear adaptive filtering, is a UD algorithm with $N_t=N_c=1$. The "*Real-Time*

Recurrent Learning Algorithm" [7] is the generalization of RPE to non-linear filters. The "*Truncated Backpropagation Through Time*" algorithm [8] is a UD-SD algorithm with N_c=1 and N_t>1. The *extended-LMS* algorithm [9] is identical to the UD-D algorithm with N_t=N_c=1 and is used in linear adaptive filtering for its autostabilization property. The "*A Posteriori Error Algorithm*" is a UD-D algorithm with N_t=2, N_c=1 [10]. The choice of N_t=N_c=1 is economical in terms of computation time; it is justified if the coefficients change slowly, i.e. if the gradient step μ is small enough.

APPLICATION TO AN ADAPTIVE FILTERING PROBLEM

The use of the new algorithms introduced above is illustrated in the case of the Adaptive Differential Pulse Code Modulation (ADPCM) system for bit rate reduction in speech transmission [11] (Figure 2). We show the influence of the training algorithm on the behaviour of the system, in the simple case of a predictor with a single adaptive coefficient b, and a two-level quantizer implemented as a neuron with transfer function f(x) = a tanh (px/a). The input signal is constant.

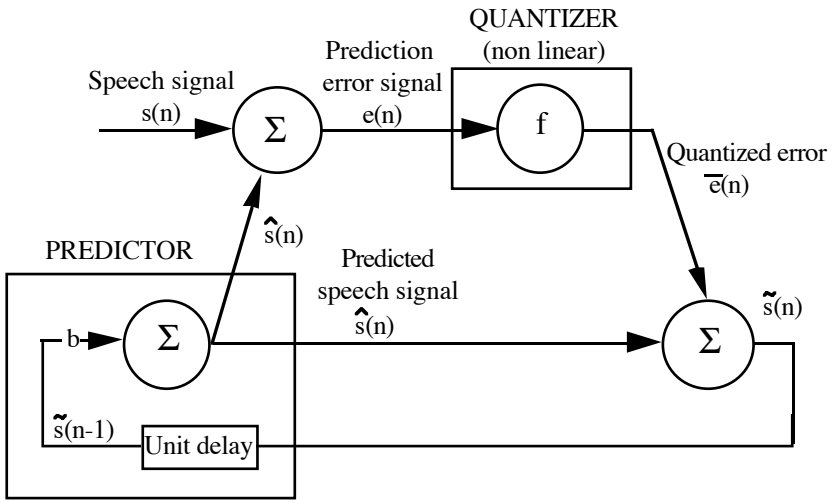


FIGURE 2

We first analyze the behaviour of the fixed, i.e. non adaptive, encoder. Fig.3 shows the prediction error e(n) versus b: for b<0.55, the error is a fixed point whose value decreases with b. For higher values of b, successive bifurcations generate limit cycles of lengths 2, 4 and 8.

The dynamical behaviour of the adaptive system depends on the choice of the adaptation algorithm, as illustrated on Figure 4. For example, the cycle P1 (of length 2) which is an attractive cycle for the non-adaptive system, remains attractive when the system is adapted with the UD or UD-SD algorithms. However, this cycle becomes a repeller when the system is adapted with the

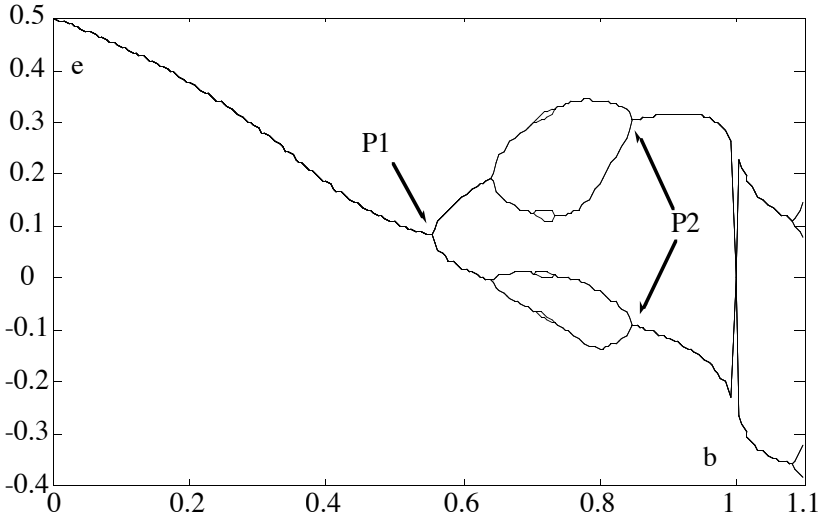


FIGURE 3

UD-D algorithm. Conversely, point P2 on Fig. 2 was found to be a repeller for the UD and UD-SD algorithms, while it is an attractor when the network is trained with a UD-D algorithm. The reported results were obtained with N_c=1 and N_t=5. The parameter N_t was found to have no influence on the results in this case; this is a specific feature of the system under consideration [12].

The mean square error for point P1 is smaller than for point P2.

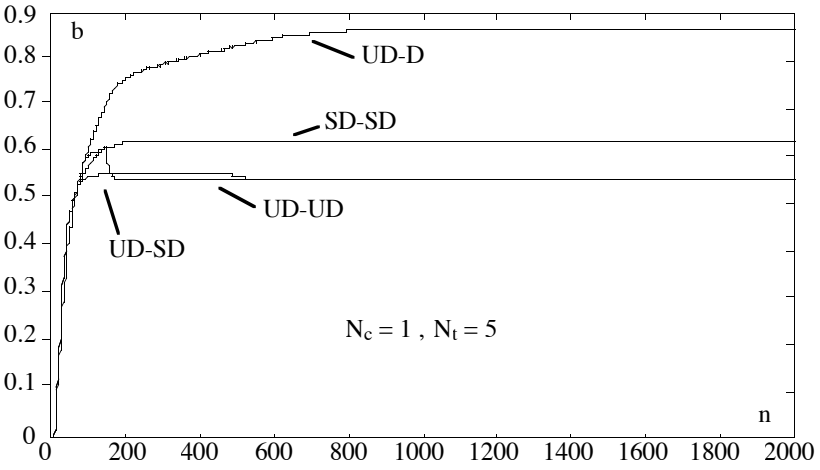


FIGURE 4

We focus now on the SD or D algorithms. Since the system to be adapted is trained by these algorithms as if it were a feedforward net consisting of N_t identical blocks and initialized with the desired outputs, the curves of Fig. 3 are irrelevant. Fig. 4 shows that the system adapted with the SD algorithm

converges towards b≈0.63, which corresponds to a very small mean square error: this indicates that the feedforward structure consisting of 5 blocks, adapted with the SD algorithm, is appropriate for the problem under consideration. Conversely, when the feedback nature of the system must be preserved, SD- or D-type algorithms are inappropriate.

APPLICATION TO IDENTIFICATION PROBLEMS

We show on the following example that semi-directed algorithms bridge the gap between the *output-error* approach (UD algorithms) and the *equation-error* approach (D algorithms).
We first consider the process identification example described in [13], which illustrates the fact that, in the presence of additive noise, the *equation-error* formulation may lead to biased estimates of the coefficients, in contrast to the *output-error* formulation.
The process to be modelled is simulated by the linear recursive equation
$y^*(n) = \alpha\, y^*(n-1) + \beta\, x(n)$
and $d(n) = y^*(n) + v(n)$ with $\alpha=\beta=0.5$,
where x is the input, d the measured output and v an additive noise.
The model used in the adaptive filter is described by $y(n) = a\, y(n-1) + b\, x(n)$, and the desired value is d(n).
If the input x(n) and the noise v(n) are uncorrelated, white sequences with zero mean value and a signal-to-noise ratio $S = \sigma_x^2/\sigma_v^2$, the equation-error (D algorithm) estimate *a* of the coefficient α is biased:
$(a-\alpha)/\alpha = (\alpha^2-1) / (1-\alpha^2+\beta^2 S)$.
The equation-error estimate of *b* is unbiased ($b = \beta$).
Conversely, both output-error estimates (UD algorithms) are unbiased.
We computed analytically the expectation value of the squared error ($N_c=1$) in the case of a semi-directed algorithm, and determined the values of *a* and *b* which minimize it. Figure 5 shows the biases with respect to N_t for S = 10; the bias of *a* decreases from the above value (for $N_t=1$) to zero ($N_t \to \infty$), which is consistent with the fact that a SD algorithm with $N_t=1$ is a D algorithm, and that it is a UD algorithm if $N_t \to \infty$. Furthermore, it is shown that the bias of *b* is zero in the two limiting cases (D and UD), and that it is small, but non-zero for $N_t>1$.
To summarize, the use of SD algorithms provides, in this example, a tradeoff between the stability of D algorithms and the unbiased estimates which result from the use of a UD algorithm.

Similarly, we consider the second process identification example described in [13] which illustrates the fact that, if the order of the model is smaller than the order of the process, the *output-error* formulation generates an error surface (MSOE) which may have local minima, whereas the *equation-error* formulation generates an error surface (MSEE) which has only a global minimum.
The process to be modelled is simulated by:

$d(n) = \alpha_1 d(n-1) + \alpha_2 d(n-2) + \beta_0 x(n) + \beta_1 x(n-1)$,
where x is the input and d the output of the process.
The model used in the adaptive filter is described by:
$y(n) = a\, y(n-1) + b\, x(n)$, and the desired value is d(n).
The MSE surface is a paraboloid when using the *equation-error* formulation (MSEE). In the case of the *output-error* formulation, the MSE surface (MSOE) exhibits one local minimum (which corresponds to a damped oscillatory behaviour, $-1<a<0$), and one global minimum ($0<a<1$) . We have computed analytically the MSE in the case of a semi-directed algorithm with $N_c=1$. As for the first example, the MSE surface changes from the MSEE surface to the MSOE surface when N_t increases from 1 to ∞: for $N_t=2$, a second minimum appears, with $-1<a<0$, which shifts to the location of the local minimum of the MSOE surface when N_t grows; meanwhile, the other minimum shifts from the location of the minimum of the MSEE surface to the location of the global minimum of the MSOE surface.

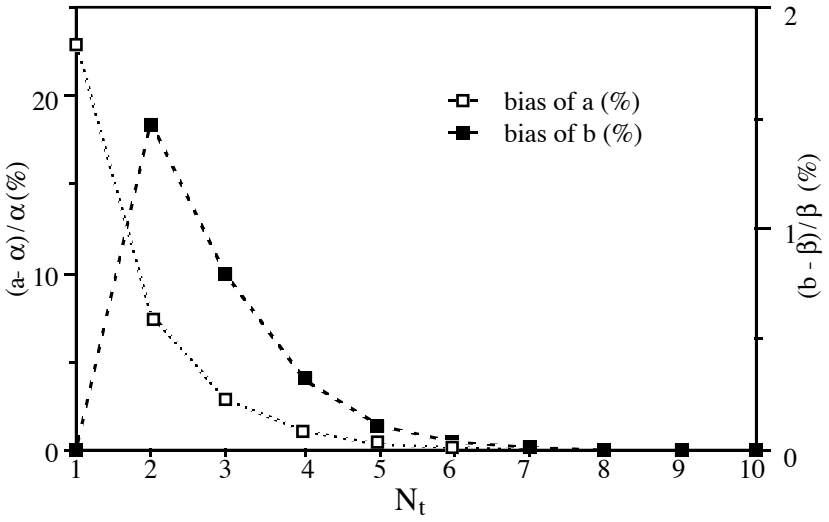


FIGURE 5

CONCLUSION

We have shown that a large variety of algorithms are available for training recurrent neural networks to perform adaptive filtering, and that the algorithms used thus far are but a small fraction of the available possibilities. We have illustrated some features of the new algorithms on three examples. Neural networks, viewed as adaptive non-linear filters, have a considerable potential which needs be explored, and basic issues, such as the stability of the algorithms, are still open.

Acknowledgements

The authors wish to thank L. CAPELY and D. MARSAN for computer simulations.

References

[1] O. Nerrand , P. Roussel-Ragot, L. Personnaz, G. Dreyfus, S. Marcos, "Neural Networks and Non-linear Adaptive Filtering: Unifying Concepts and New Algorithms", Neural Computation, to be published.

[2] S. Marcos, P. Roussel-Ragot, L. Personnaz, O. Nerrand, G. Dreyfus, C. Vignat, "Réseaux de Neurones pour le Filtrage Non Linéaire Adaptatif", Traitement du Signal, in press (1992).

[3] B. Widrow, S.D. Stearns, Adaptive Signal Processing (Prentice-Hall, 1985).

[4] S. Chen, S.A. Billings, "Representations of Non-Linear Systems: the NARMAX Model", Int. J. Control, vol. 49, pp. 1013-1032, 1989.

[5] M.I. Jordan, "Attractor Dynamics and Parallelism in a Connectionist Sequential Machine", in Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986, pp. 531-546.

[6] L. Ljung, T. Söderström, Theory and Practice of Recursive Identification, M.I.T. Press, 1983.

[7] R.J. Williams, D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", Neural Computation, vol. 1, pp. 270-280, 1989.

[8] R.J. Williams, J. Peng, "An Efficient Gradient-based Algorithm for On-Line Training of Recurrent Network Trajectories", Neural Computation, vol. 2, pp. 490-501, 1990.

[9] P.L. Feintuch, "An Adaptive Recursive LMS Filter", Proc. IEEE, pp. 1622-1624, 1976

[10] C.R. Johnson, I.D. Landau, "On Adaptive IIR Filters and Parallel Adaptive Identifiers with Adaptive Error Filtering", Proc. ICASSP, pp. 5387, 1981.

[11] N.S. Jayant, P. Noll, Digital Coding of Waveforms. Principles and Applications to Speech and Video, Signal Processing Series, A. Oppenheim, ed., Prentice-Hall, 1984.

[12] C. Vignat, C. Uhl, S. Marcos, "Analysis of gradient-based adaptation algorithms for linear and nonlinear recursive filters", Proceedings of ICASSP-92, Vol. IV, pp. IV 189-IV 192, March 23-26, 1992, San Francisco.

[13] J. J. Shynk, "Adaptive IIR Filtering", IEEE ASSP Magazine, pp. 4-21, 1989.

Training Recurrent Neural Networks: Why and How ? An Illustration in Dynamical Process Modeling.

O. NERRAND, P. ROUSSEL-RAGOT,
D. URBANI, L. PERSONNAZ, G. DREYFUS, Senior Member, IEEE
Ecole Supérieure de Physique et de Chimie Industrielles
de la Ville de Paris,
Laboratoire d'Electronique
10, rue Vauquelin
75005 PARIS, FRANCE

ABSTRACT

The paper first summarizes a general approach to the training of recurrent neural networks by gradient-based algorithms, which leads to the introduction of four families of training algorithms. Because of the variety of possibilities thus available to the "neural network designer", the choice of the appropriate algorithm to solve a given problem becomes critical. We show that, in the case of process modeling, this choice depends on how noise interferes with the process to be modeled; this is evidenced by three examples of modeling of dynamical processes, where the detrimental effect of inappropriate training algorithms on the prediction error made by the network is clearly demonstrated.

1 INTRODUCTION

During the past few years, there has been a growing interest in the training of recurrent neural networks, either for associative memory tasks, or for tasks related to grammatical inference, time series prediction, process modeling and process control. A general framework for the training of recurrent networks by gradient descent methods, which has been proposed recently [1, 2], is summarized in section 2; it encompasses algorithms which have been used classically in linear filtering, identification and control, and algorithms which have been established in the framework of neural network research; in addition, this general approach leads to original algorithms. The variety of algorithms thus available raises the question of the choice of an appropriate one in a given situation. In section 3, we show, in the framework of non-linear process identification (i.e., of the estimation

of the parameters of a model of a non-linear process), that the choice of an appropriate algorithm depends of how *noise* interferes with the process. The striking effect of using either an appropriate algorithm or an inappropriate one for modeling a non-linear process undergoing non-measurable, random perturbations, is shown on examples.

2 A GENERAL FRAMEWORK FOR THE TRAINING OF RECURRENT NETWORKS BY GRADIENT-BASED DESCENT ALGORITHMS

In this section, we summarize a general approach described in more detail in [1]. We first define the terms which will be used in the paper. Some of this terminology is borrowed directly from the fields of filtering and automatic control; since many familiar concepts in the neural network area have been in use in other disciplines, we deem it unnecessary, and in most cases confusing, to coin new words for old concepts; conversely, we shall introduce a few new terms whenever required for clarity. In the second part of this section, we recall the ingredients of the algorithms whose use is illustrated in section 3.

2.1 Some definitions

Because the terminologies used in adaptive filtering, in automatic control, and in the literature on neural networks, are sometimes conflicting, we first define the terms that we use in the paper.

Adaptive vs. non-adaptive training
The training of a network makes use of two sequences, the sequence of inputs and the sequence of corresponding desired outputs. If the network is first trained (with a training sequence of finite length), and subsequently used (with the fixed weights obtained from training), we shall refer to this mode of operation as "non-adaptive". Conversely, we term "adaptive" the mode of operation whereby the network is trained permanently while it is used (with a training sequence of infinite length).

Performance criterion, cost function and training function
The computation of the coefficients during training aims at finding a system whose operation is optimal with respect to some performance criterion which may be either quantitative, e.g., maximizing the signal to noise ratio for spatial filtering, or qualitative, e.g. the (subjective) quality of speech reconstruction. In the

following, we assume that we can define a positive *training function* which is such that a decrease of this function through modifications of the coefficients of the network leads to an improvement of the performance of the system.
In the case of non-adaptive training, the training function is defined as a function of all the data of the training set (in such a case, it is usually termed *cost function*); the minimum of the function corresponds to the optimal performance of the system. Training is an optimization procedure, using gradient-based methods.
In the case of adaptive training, it is impossible, in most instances, to define a time-independent cost function whose minimization leads to a system which is optimal with respect to the performance criterion. Therefore, the training function is time-dependent. The modification of the coefficients is computed continually from the gradient of the training function. The latter involves the data pertaining to a time window of finite length, which shifts in time (sliding window), and the coefficients are updated at each sampling time for instance.

Recursive vs. non-recursive algorithms, iterative vs. non-iterative algorithms
A *non-recursive* algorithm makes use of a cost function (i.e. a training function defined on a fixed window). A *recursive* algorithm makes use of a training function defined on a sliding window [3]. Therefore, an adaptive system must be trained by a recursive algorithm, whereas a non-adaptive system may be trained either by a non-recursive or by a recursive algorithm.
An *iterative* algorithm performs coefficient modifications *several times* from a set of data pertaining to a given time window; a *non-iterative* algorithm does this *only once*. The popular LMS (Least Mean Squares) algorithm is thus a recursive, non-iterative algorithm operating on a sliding window of length 1.

In the following, we focus on the computation of the coefficients by gradient-based descent; in the recursive, non-iterative case, the modification of the coefficients at time n can be written as $\Delta C(n) = \mu(n)\, D(n)$ where $\{\mu(n)\}$ is a sequence of positive real numbers and D(n) is a linear transformation of the gradient of the training function; in the simple gradient method, D(n) is just the opposite of the gradient and $\mu(n)$ is constant.

2.2 Training algorithms for recurrent networks

Canonical form
All the computational details on the material presented in this section can be found in reference [1].

It has been shown in [1] that any feedback network can be cast into a canonical form which consists of a feedforward (static) network

- whose outputs are the outputs of the neurons which have desired values, and the values of the state variables,
- whose inputs are the inputs of the network and the values of the state variables, the latter being delayed by one time unit (Figure 1a).

The canonical form is thus expressed as

$S(k) = \varphi_1[S(k-1), I(k-1)]$; $z(k-1) = \varphi_2[S(k-1), I(k-1)]$,

where S(k) is the state vector, whose dimension N_r is the order of the network, where z(k-1) is the output, and where I(k) is the vector of non-feedback inputs.

The transformation of a non-canonical form to a canonical form is described in [1]. Note that this concept can be used with any type of discrete-time neuron, including for instance the high-order units used for grammatical inference [4]

Training function

The main difficulty in the *recursive* training of recurrent networks arises from the fact that the output of the network and its partial derivatives with respect to the coefficients depend on the values of the inputs since the beginning of the training process, and on the initial state of the network. Therefore, a rigorous computation of the gradient of the training function would imply taking into account all the past inputs, and related desired outputs. This is not practical for two reasons: first, it would require ever increasing computation times; second, in the case of the modeling or control of a non-stationary process, taking the whole past into account would not make sense, since a large part of the past might be irrelevant. Therefore, the estimation of the gradient of the training function is performed by truncating the computations to a fixed number of sampling periods N_t into the past. Thus, at time n, this estimation will involve N_t identical copies of the feedforward part of the canonical form of the network, with coefficients computed at time n-1 (Figure 1b).

The training function at time n is defined on a sliding window of length N_c as a sum of N_c quadratic errors:

$$J(C, n) = \frac{1}{2} \sum_{m=N_t-N_c+1}^{N_t} [e^m(n)]^2 \text{ with } e^m(n) = d(n-N_t+m) - y^m(n) \text{ and } 1 \le N_c \le N_t \ ,$$

where $y^m(n)$ is the output of copy m ($1 \le m \le N_t$). $y^m(n)$ is the value that the output of the network would have taken on, at time $n-N_t+m$, had the vector of coefficients at that time been equal to C(n-1).

In the case of *non-recursive* training, the training (or cost) function is defined on a fixed window of length N_c ; at iteration i:

$$J(C, i) = \frac{1}{2} \sum_{m=N_t-N_c+1}^{N_t} \left[e^m(i)\right]^2 \text{ with } e^m(i) = d(m) - y^m(i) \text{ and } 1 \le N_c \le N_t \ ,$$

where $y^m(i)$ is the output of copy m ($1 \le m \le N_t$), computed with the weights C(i-1) obtained at iteration i-1.

Algorithms

The computation of the above training function requires the computation of the outputs $y^m(n)$ (or $y^m(i)$), which in turns require the computation of the state $S_{in}^m(.)$ of the network (Figure 1b); various algorithms arise from different choices of the values of the state inputs. In [1], four families of algorithms were introduced: undirected, semi-directed, directed, and hybrid. In the following, we restrict our discussion to the case where the desired values of the state variables are available; thus, the first three families only will be considered in the present paper.

3 APPLICATION: NON-LINEAR PROCESS IDENTIFICATION BY NEURAL NETWORKS

3.1 The problem

Assume that a set of measurements can be carried out on a non-linear process. From this data, a predictor model must be derived, whose dynamical behaviour should be as close as possible to that of the process. The identification of the process is the estimation of the parameters of the predictor, based on the available data; if the predictor is implemented as a neural network, the identification is the training of the network. When identifying a non-linear, dynamical process, a recurrent network is a logical candidate. We show in the following how the choice of the appropriate training algorithm results from assumptions made on the role on random noise in the process. We use non-linear generalizations of three popular models corresponding to three different assumptions on the noise; we describe the predictor associated to each model, i.e. the predictor which is such that *the prediction error is the unpredictable part of the process output*. We show, in each case, which of the above algorithms is the most appropriate, if the predictor is implemented as a neural network.

3.2 Three black-box models

Three approaches with black-box models will be considered, depending on the assumptions made on the process [3]: (i) the *output error* model, (ii) the *NARMAX* model, and (iii) the *NARX* (or *equation error*) model .

In the *output error* approach, it is assumed that the output $y_p(k)$ of the process (Figure 2a) obeys the following equations:

$x(k) = \Phi\left[X(k-1), U(k-1)\right]$,

$y_p(k) = x(k) + w(k)$,

with $X(k-1) = \{x(k-1), x(k-2), ...x(k-N)\}$, and $U(k-1) = \{u(k-1), u(k-2), ...u(k-M\}$.

$\{w(k)\}$ is a white noise sequence.

The output y(k) of the associated predictor (Figure 2b), such that $y_p(k)-y(k)=w(k)$, is given by:

$y(k) = \Phi\left[y(k-1), ... , y(k-N), U(k-1)\right]$.

Therefore, the associated predictor of the output error process is *recurrent* of order N. If there exists a neural network which can approximate function Φ, this network can implement the predictor, *and it must be trained by an undirected algorithm* [1], since it is essential that the predictor be recurrent.

A NARMAX (Non-linear Auto-Regressive Moving Average with eXogeneous inputs) model [5] (Figure 3a) obeys the following equation:

$x_p(k) = \Phi\left[X_p(k-1), U(k-1), W(k-1)\right] + w(k)$,

$y_p(k) = x_p(k)$.

where $X_p(k-1) = \{x_p(k-1), x_p(k-2), ..., x_p(k-N)\}$ and

$W(k-1) = \{w(k-1), w(k-2), ..., w(k-P)\}$.

The output y(k) of the associated predictor (Figure 3b) is defined by:

$y(k) = \Phi\left[Y_p(k-1), U(k-1), e(k-1), ..., e(k-P)\right]$ where $e(k) = y_p(k)-y(k)$

and $Y_p(k-1) = \{y_p(k-1), y_p(k-2), ..., y_p(k-N)\}$.

Therefore, the predictor of the NARMAX process is *recurrent* of order P, and, if it is implemented as a neural network, *it must be trained by an undirected algorithm* [1].

In the *equation error* approach (Non-linear Auto-Regressive with eXogeneous inputs, or NARX, model, Figure 4a), it is assumed that the process obeys the following equations:

$x_p(k) = \Phi\left[X_p(k-1), U(k-1)\right] + w(k)$,

$y_p(k) = x_p(k)$.

The output y(k) of the associated predictor (Figure 4b) is given by

$y(k) = \Phi\left[Y_p(k-1), U(k-1)\right]$ with $Y_p(k-1) = \left\{y_p(k-1), y_p(k-2), ... , y_p(k-N)\right\}$.

Therefore, the predictor of the equation error process is actually a *non-recursive* predictor, whose inputs are the external inputs of the process and the (measured) outputs of the process. If there exists a neural network which can approximate

function Φ, this network can implement the predictor, *and a directed algorithm* [1] is the only suitable choice, since the predictor is not recursive.

To summarize, the algorithms derived for the training of discrete-time recurrent neural networks can readily be applied to the identification of dynamical non-linear processes. Directed algorithms are best suited to the training of neural networks intended to predict the output of processes satisfying the perturbation-free hypothesis or the equation error hypothesis, whereas undirected algorithms are best suited to the NARMAX and output error hypotheses. It is intuitive, and it can be shown analytically in simple cases [6], that semi-directed algorithms bridge the gap between these approaches.

3.3 Illustration: identification of a first-order non-linear process

In this section, we propose several illustrations of the above algorithms. We first train a neural network, both adaptively and non-adaptively, to model a deterministic, noise-free simulated process. In section 3.3.2, we add output noise to the same deterministic equation as above, and we train a network, non-adaptively, to model the resulting process. We pretend that we do not know how noise interferes with the process; we first make the assumption that the process is appropriately described by an output error model, and we train the network accordingly with an undirected algorithm; we subsequently make the assumption that the process is appropriately described by an equation error model, and we train the neural network accordingly with a directed algorithm; we compare the results obtained in these two cases. Finally, in section 3.3.3, we add state noise to the same deterministic equation as above, and we train a network, non-adaptively, to model the resulting process; we make the same two assumptions as above. The detrimental effect of using the wrong algorithm, i.e. of making the wrong assumption on the influence of the noise on the process, is shown clearly on all these examples.

All results presented here were obtained by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [7], with step adaptation by the method of Wolfe and Powell [8].

3.3.1 - Example 1: perturbation-free process

3.3.1.1 - Simulation equation

A continuous-time process is simulated by the following discrete-time equation

$$y_p(k) \equiv \Psi\left[y_p(k\text{-}1),\, u(k\text{-}1)\right] = \left[1 - \frac{T}{a+by_p(k\text{-}1)}\right] y_p(k\text{-}1) + \left[T\,\frac{c+dy_p(k\text{-}1)}{a+by_p(k\text{-}1)}\right] u(k\text{-}1) \quad ,$$

where $y_p(k)$ is the output of the process at time k, and u(k) is the external input at time k. In the following, the values of the parameters are:
a=-0.139, b=1.2, c=5.633, d=-0.326, sampling period T= 0.1 sec.

3.3.1.2 - Adaptive identification of the noise-free process
We first identify the process adaptively, making the assumption that it can be adequately described in the vicinity of an operating point by a linear first-order model; this approach is useful if the model is to be used, with small input and ouput signals, within an adaptive control system as an alternative to gain scheduling. It can be implemented by a "neural network" made of a single, linear neuron; in the absence of perturbations, the appropriate training algorithm is a directed algorithm. The behaviour of the adaptive predictor, and the prediction error, are illustrated on Figure 5.

3.3.1.3 - Non-adaptive identification of the noise-free process
The process can also be identified non-adaptively. The predictor must then be valid in a suitable region of state space; it can be used in the case of large input and output signals. This can be achieved, in the present case, by a feedforward neural network with one hidden layer of five neurons. The training set is a sequence of 100 steps of random amplitude. Training has been performed by a non-recursive, iterative, directed algorithm with $N_c(=N_t)=2000$. Figure 6 illustrates the behaviour of the non-adaptive predictor, and the prediction error.

3.3.2 - Example 2: process with additive output noise
The simulated process that we consider now is described by the same equation as in the previous section, with additive noise on the output:
$x_p(k) = \Psi\left[x_p(k-1), u(k-1)\right]$
$y_p(k) = x_p(k) + w(k)$.
w(k) is white noise with maximum amplitude 0.5.
The goal of identification is to find a (neural network) predictor that implements a function as close as possible to Ψ in a bounded domain of state space; therefore, the prediction error should be as close as possible to the noise w(k) once training is completed.
We first make the (correct) assumption that an output error model is appropriate. Thus, we use a recurrent predictor of the type shown on Figure 2b; the feedforward part of the neural network has the same architecture as in the perturbation-free case (since we know from the previous section that such a network can approximate function Ψ with satisfactory accuracy), and we train it with an

undirected algorithm (undirected, non recursive, iterative with $N_t=N_c=2000$). Figure 7a show the response of the network at the end of training, and Figure 7b shows the prediction error. As expected, the latter is just white noise of amplitude 0.5, which shows that (i) the feedforward part of the predictor network is appropriate for the approximation of function Ψ, that (ii) the undirected training algorithm is the appropriate algorithm for training the predictor, or, in other words, that the assumption that the process can be described by an output error model is correct, and that (iii) the quasi-Newton gradient method (of constant use in recursive identification [3]) allows a very efficient optimization of the cost function; this may seem to be a side issue, but it is worth pointing out that the results presented in this paper would not have been obtained in any reasonable time otherwise.

We now make the (wrong) assumption that the process can be described by a NARX model. Accordingly, we choose a neural network predictor of the type shown on Figure 4, which we train with a *directed* algorithm on the same data as before. Figure 8 shows the prediction error after training (directed, non recursive, iterative algorithm with $N_c=2000$), with the same inputs as shown on Figure 7a: the vairance of the prediction error is much larger than in the previous case, thereby showing that the training algorithm is inappropriate for extracting the model in the presence of the additive output noise.

3.3.3 - Example 3: process with additive state noise

In this section, we consider again the same simulation equation as in section 3.3.1.1, but we add white noise, with amplitude 0.5, to its state:

$x_p(k) = \Psi\left[x_p(k-1), u(k-1)\right] + w(k)$,

$y_p(k) = x_p(k)$.

We first make the (correct) assumption that a NARX model is appropriate. Thus, we use a predictor as shown on Figure 4, with five hidden neurons, trained by a *directed* algorithm (non recursive, iterative with $N_c=2000$). The result after training is exactly as shown on Figure 7b: the prediction error is just white noise, which shows that the identification of the process by the neural network has been perfectly successful.

If we now make the (wrong) assumption that the process can be represented by an output error model, we take a predictor as shown on Figure 2, we use five hidden neurons in the feedforward part of the network, and we train the model with an undirected algorithm; the resulting prediction error is as shown on Figure 9: the error is clearly not white noise, thereby showing that the use of an *undirected* algorithm with additive state noise prevents the network from correctly extracting

the model, although we know from the previous examples that the network has the appropriate number of hidden units for approximating function Ψ.

4 CONCLUSION

We have shown the importance of choosing an appropriate training algorithm for the modeling of a dynamical system in the presence of noise. Directed (teacher forcing) algorithms are appropriate for the modeling of noiseless dynamical systems, or for systems in which random perturbations can be considered as white noise added to the state variables of the black-box model, whereas undirected algorithms are appropriate for predicting the output of systems in which random perturbations can be considered as white noise added to the output of the black-box model. Although the architecture of the feedforward part of the neural predictors is the same in all the above examples, and is known to be appropriate for describing the non-linearity of the process, very different results can be obtained, depending on the algorithm used. Within the appropriate family, other choices (recursive or non-recursive algorithm, iterative or non-iterative algorithm, values of N_c and N_t, ...) are important but less critical; they will be made on the basis of the stationarity time of the process, of the computer time available, etc...

In the above examples, semi-directed algorithms have not been used because no stability problem was encountered with undirected algorithms: semi-directed algorithms are useful when an output error model describes the process appropriately, but when the corresponding predictor is unstable. Detailed stability analyses of undirected algorithms have been performed in simple cases [9].

REFERENCES

[1] O. Nerrand, P. Roussel-Ragot, L. Personnaz, G. Dreyfus, S. Marcos, "Neural Networks and Non-linear Adaptive Filtering: Unifying Concepts and New Algorithms", *Neural Computation,* vol. 5, pp. 165-197 (1993).

[2] S. Marcos, P. Roussel-Ragot, L. Personnaz, O. Nerrand, G. Dreyfus, C. Vignat, "Réseaux de Neurones pour le Filtrage Non-linéaire Adaptatif", *Traitement du Signal*, vol. 8, pp. 409-422 (1993).

[3] L. Ljung, T. Söderström, *Theory and Practice of Recursive Identification,* MIT Press (1983).

[4] C.L. Giles, G.Z. Sun, H.H. Chen, Y.C.Lee, D. Chen, "Higher Order Recurrent Networks and Grammatical Inference", *Advances in Neural Information Processing Systems 2*, D.S. Touretzky, ed., pp. 380-387 (1990).

[5] S. Chen, S.A. Billings, "Representations of Non-Linear Systems: the NARMAX Model". *Int. J. Control,* vol 49, pp. 1013-1032 (1989).

[6] G. Dreyfus, O. Macchi, S. Marcos, O. Nerrand, L. Personnaz, P. Roussel-Ragot, D. Urbani, C. Vignat, "Adaptive Training of Feedback Neural Networks for Non-linear Filtering", *Neural Networks for Signal Processing II,* S.Y. Kung, F. Fallside, J. Aa. Sorenson, C.A.Kamm, eds (1992).

[7] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press (1986).

[8] P. Wolfe, *Convergence Conditions for Ascent Methods,* S.I.A.M. Review vol. 11, pp. 226-235 (1969).

[9] C. Vignat, "Convergence des Approches Filtrage Adaptatif et Réseaux de Neurones Formels. Cas des Systèmes Non-Linéaires Bouclés". Thèse de l'Université de Paris-Sud, Orsay (1993).

FIGURE CAPTIONS

Figure 1:
a. The canonical form of a discrete-time recurrent network.
b. Copy m at time n of the feedforward part of the canonical form.

Figure 2:
a. Structure of the model of the process under the *output error* hypothesis.
b. Associated neural predictor.

Figure 3:
a. Structure of the model of the process under the *NARMAX* hypothesis.
b. Associated neural predictor.

Figure 4:
a. Structure of the model of the process under the *equation error* hypothesis.
b. Associated neural predictor.

Figure 5:
Example 1: adaptive identification of the perturbation-free process. Training with a directed recursive algorithm (N_c=20).
a. Input and output of the adaptive predictor.
b. Prediction error.

Figure 6:
Example 1: non-adaptive identification of the perturbation-free process. Training with a directed iterative algorithm (N_c=2000).
a. Input and output of the non-adaptive predictor after training.
b. Prediction error.

Figure 7:
Example 2: non-adaptive identification of the process with additive output noise . Training with an undirected iterative algorithm (N_t=N_c=2000), corresponding to the correct hypothesis (output error model).
a. Input and outputs of the simulated process and of the predictor.
b. Prediction error.

Figure 8:
Example 2: non-adaptive identification of the process with additive output noise. Prediction error after training with a directed iterative algorithm (N_c=2000), corresponding to a wrong hypothesis (equation error model).

Figure 9:
Example 3: non-adaptive identification of the process with additive state noise. Prediction error after training with an undirected iterative algorithm (N_t=N_c=2000), corresponding to a wrong hypothesis (output error model).

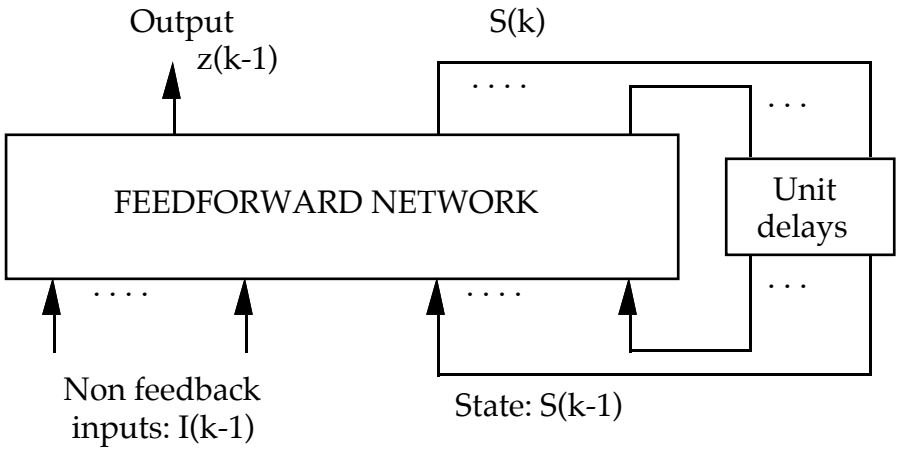


FIGURE 1a

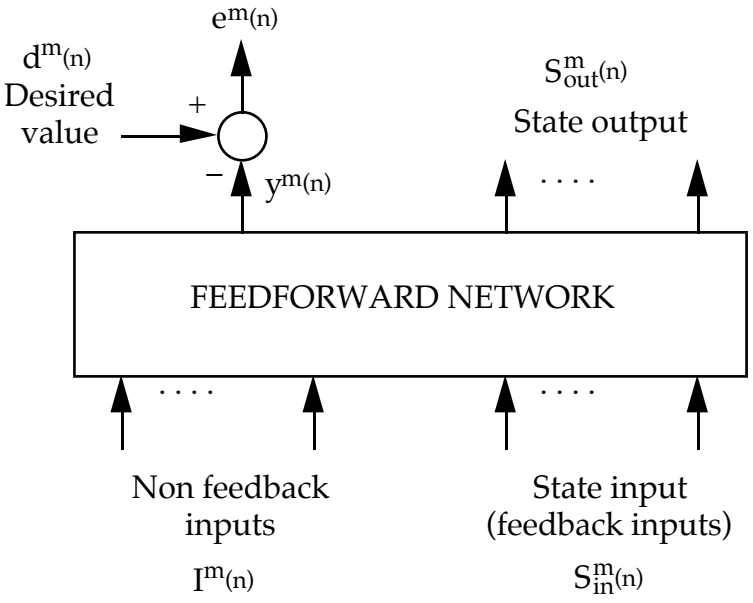


FIGURE 1b

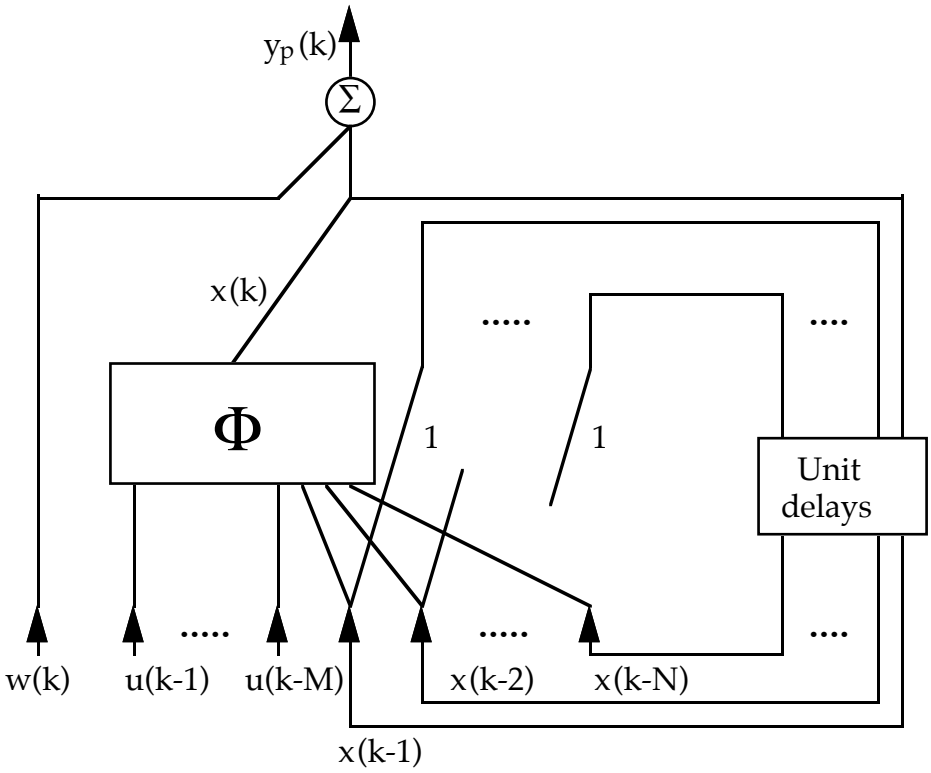


FIGURE 2a

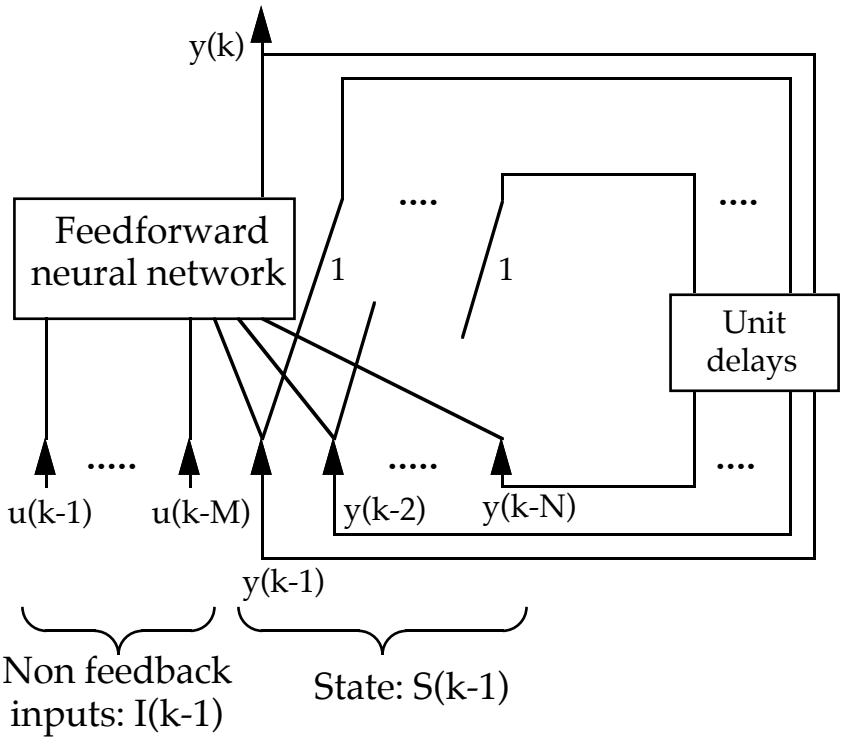


FIGURE 2b

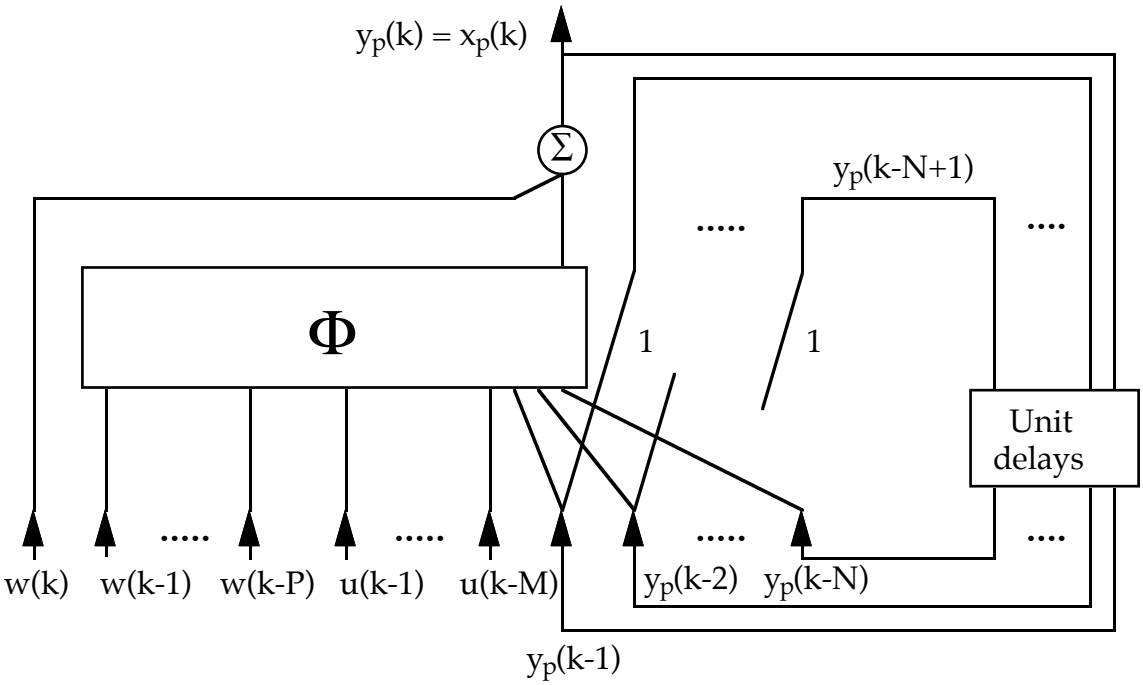


FIGURE 3a

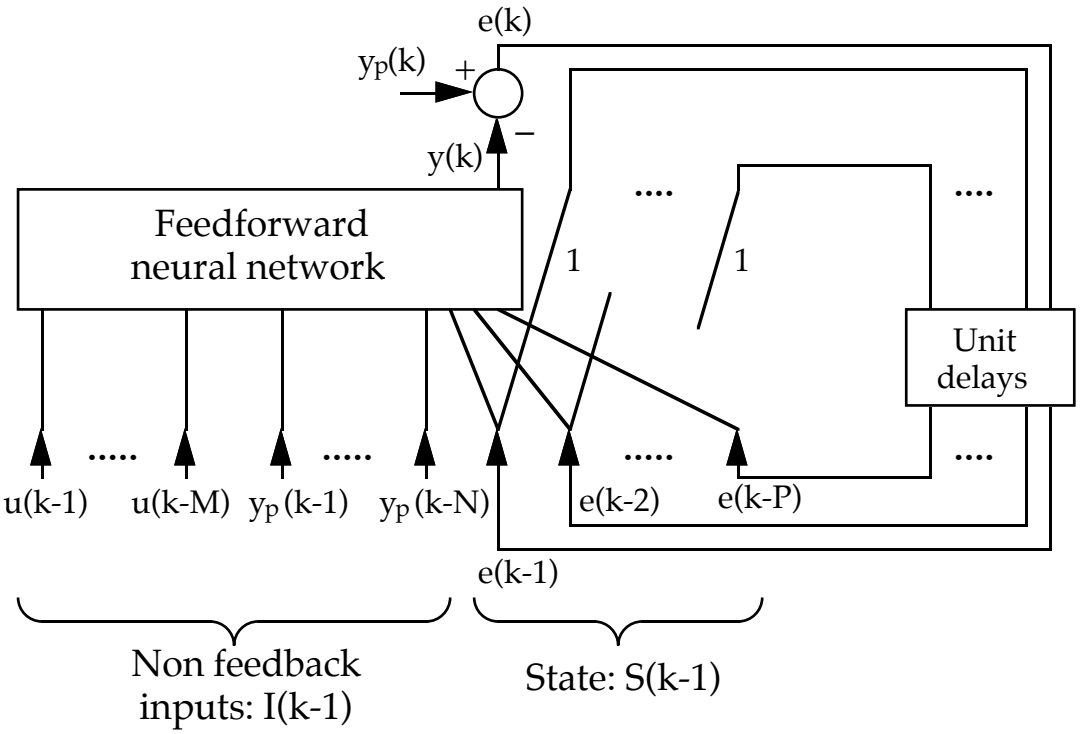


FIGURE 3b

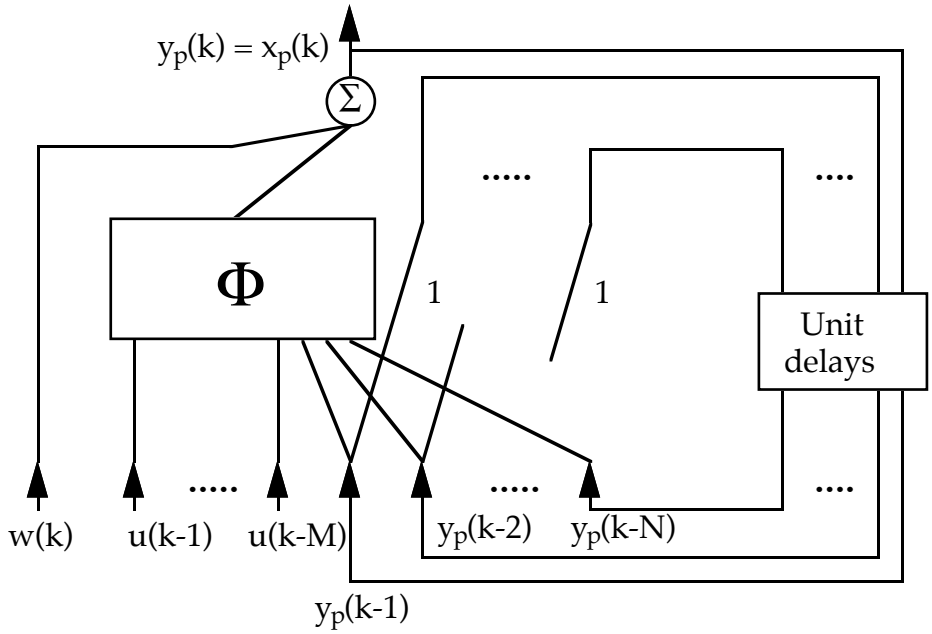


FIGURE 4a

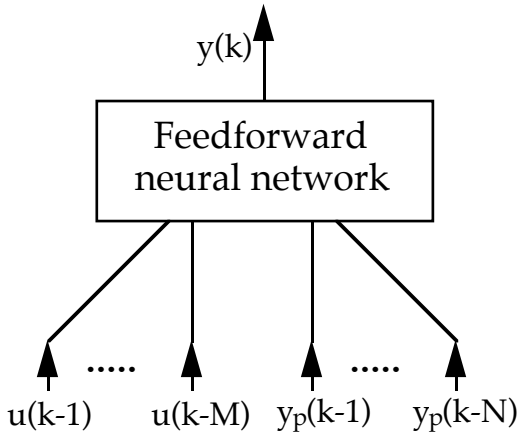


FIGURE 4b

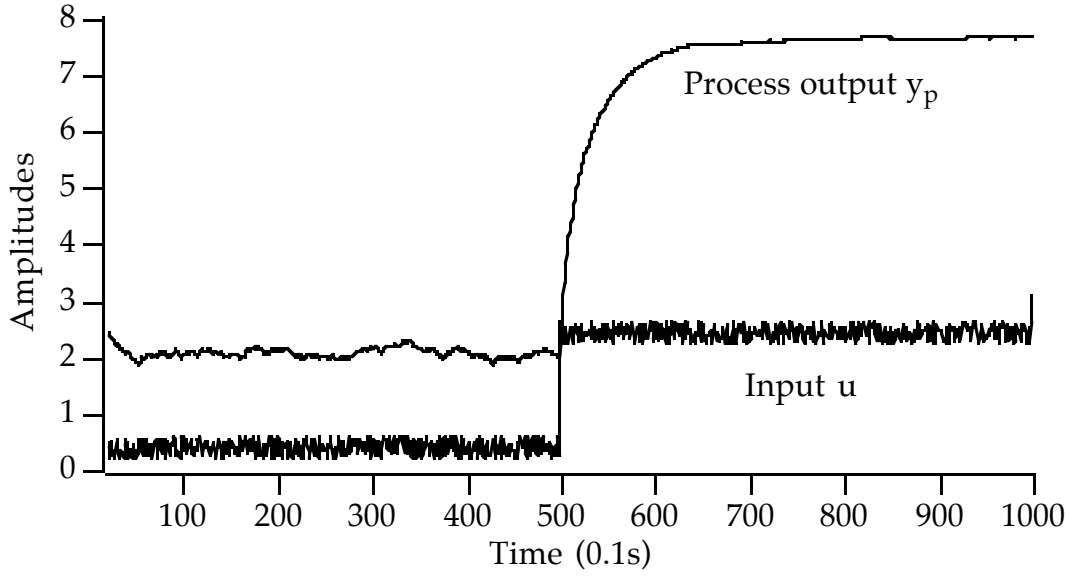


FIGURE 5a

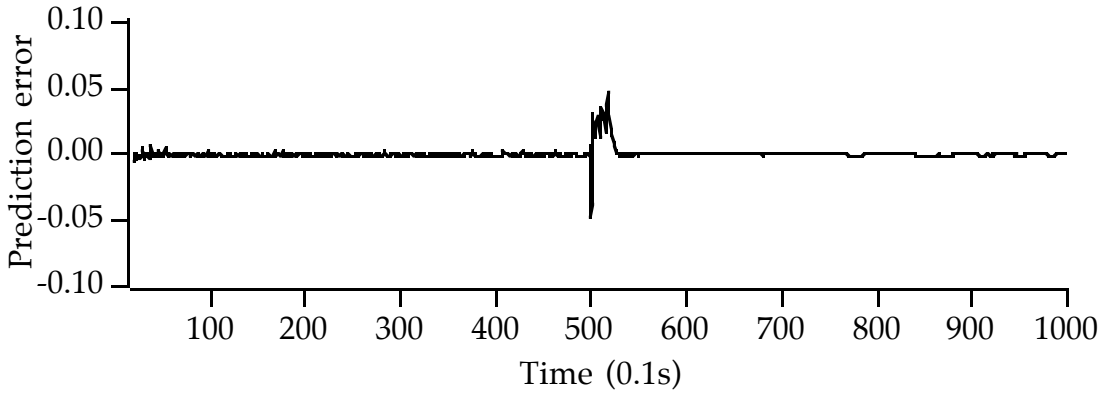


FIGURE 5b

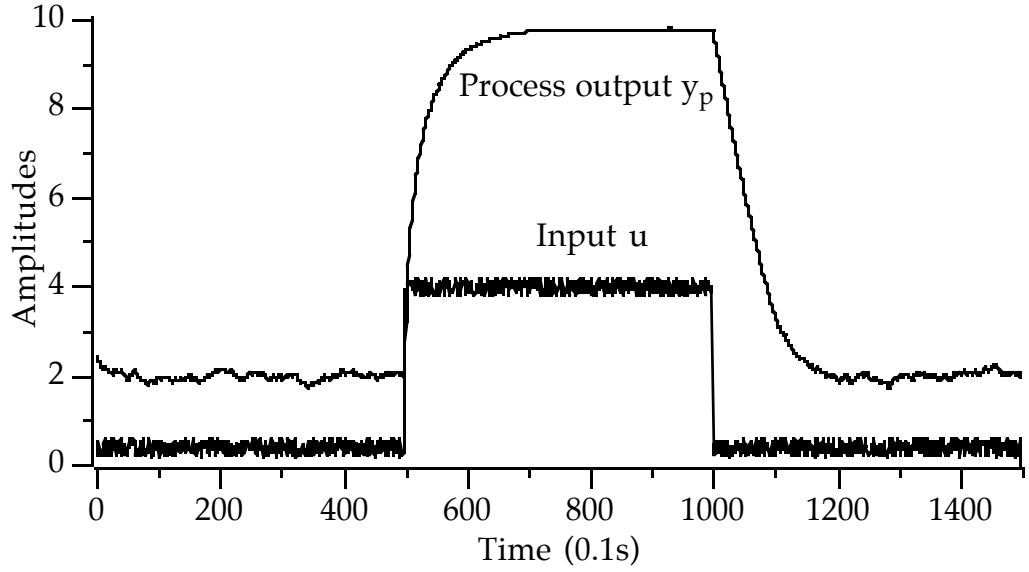


FIGURE 6a

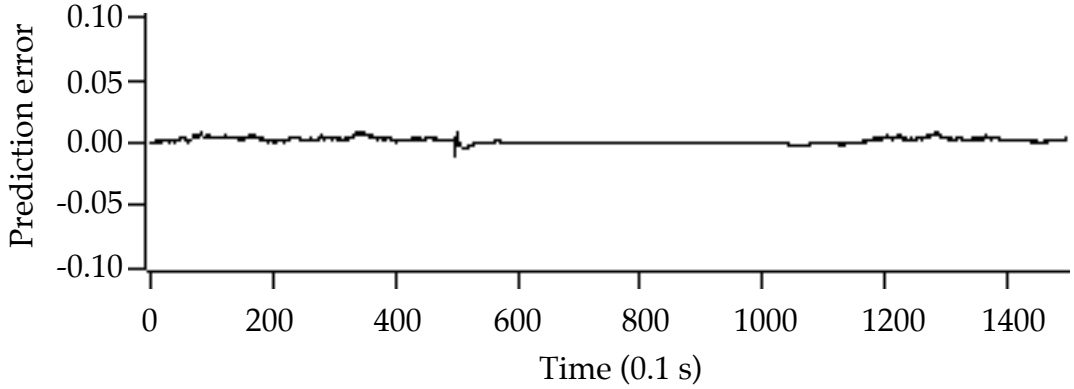


FIGURE 6b

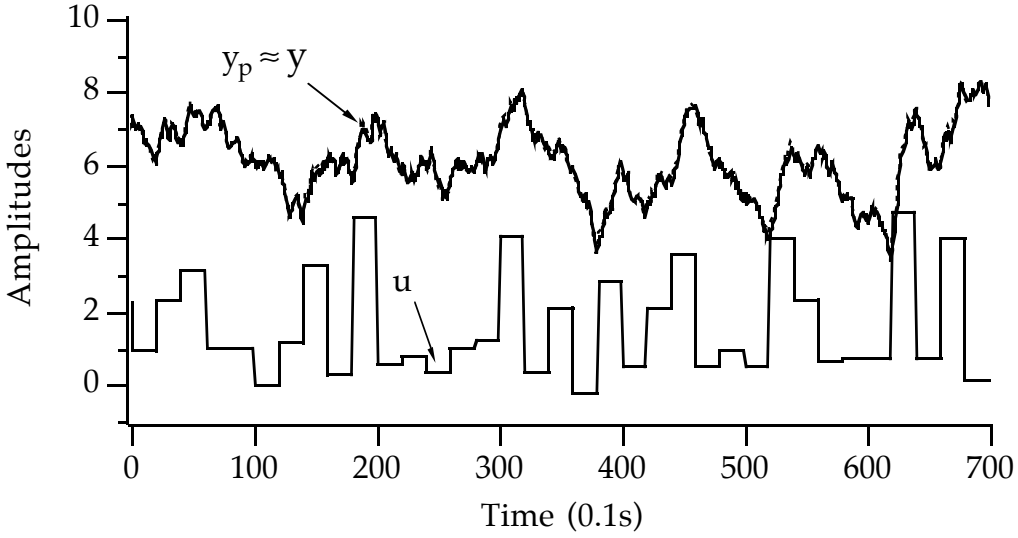


FIGURE 7a

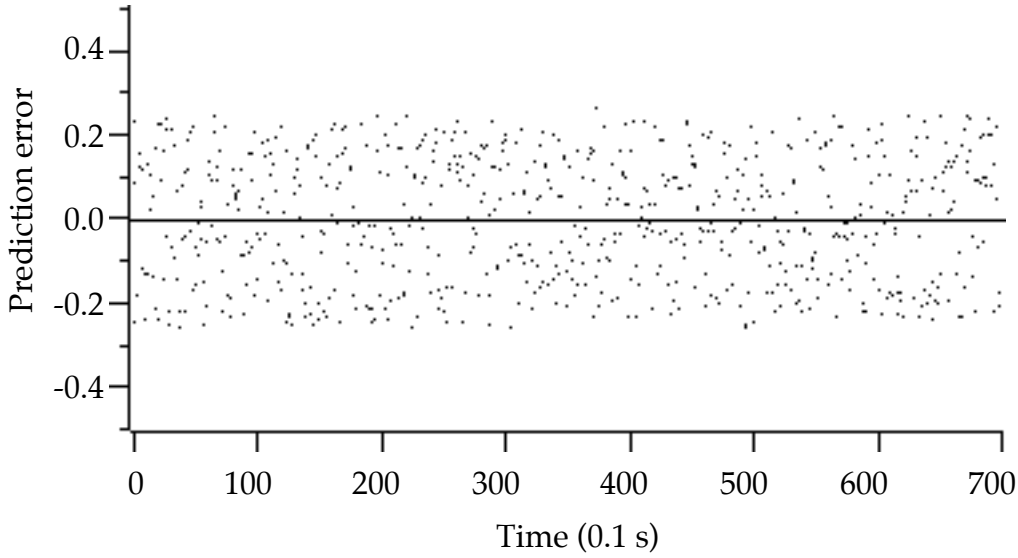


FIGURE 7b

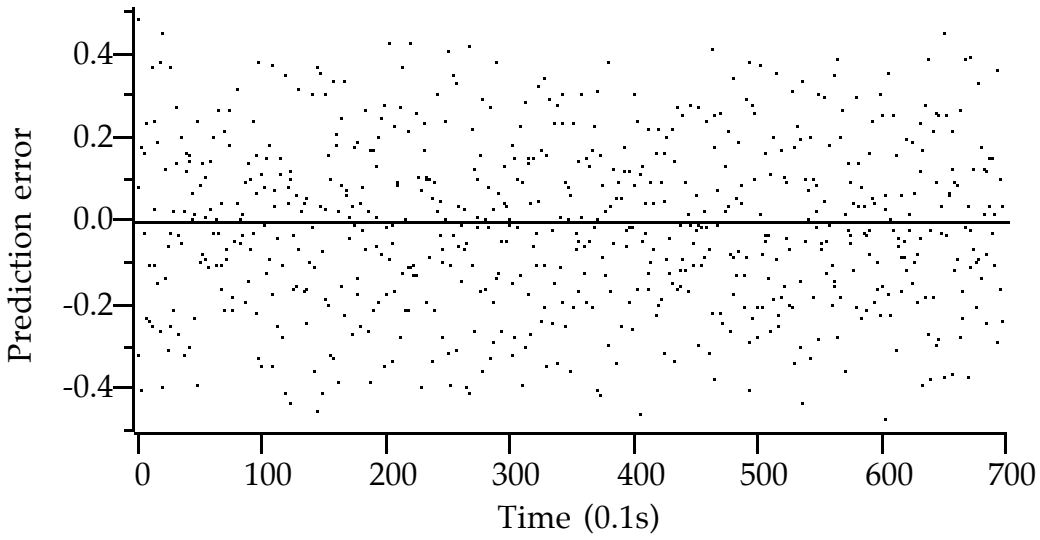

Prediction error
0.4
0.2
0.0
-0.2
-0.4
0
100
200
300
400
500
600
700
Time (0.1s)

FIGURE 8

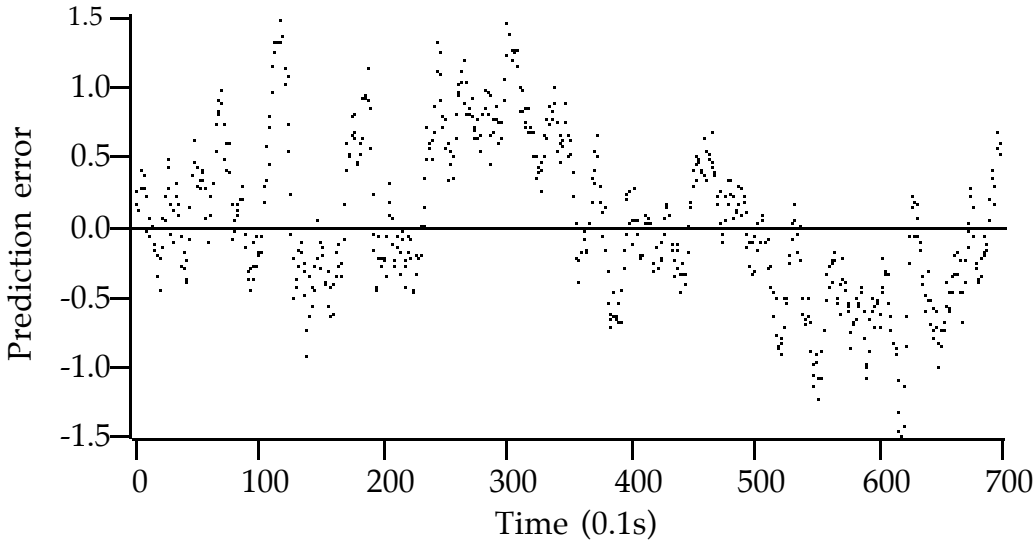

1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
Prediction error
0
100
200
300
400
500
600
700
Time (0.1s)

FIGURE 9

Neural Networks for Signal Processing IV, J. Vlontzos, J. Hwang, E. Wilson, eds, pp. 229-237 (IEEE , 1994).

THE SELECTION OF NEURAL MODELS OF NON-LINEAR DYNAMICAL SYSTEMS BY STATISTICAL TESTS

D. URBANI, P. ROUSSEL-RAGOT,
L. PERSONNAZ, G. DREYFUS
Ecole Supérieure de Physique et de Chimie Industrielles de la Ville de Paris
Laboratoire d'Electronique
10, rue Vauquelin
F - 75005 PARIS - FRANCE
Phone: 33 1 40 79 45 41 ; Fax: 33 1 40 79 44 25
e-mail: dreyfus@neurones.espci.fr

Abstract - A procedure for the selection of neural models of dynamical processes is presented. It uses statistical tests at various levels of model reduction, in order to provide optimal tradeoffs between accuracy and parsimony. The efficiency of the method is illustrated by the modeling of a highly non-linear NARX process.

INTRODUCTION

The representation of the behaviour of dynamical processes is a conceptually straightforward application of neural networks, whether feedforward or recurrent, as non-linear regressors. In practice, however, the modeling of a process requires solving several problems:
(i) the choice of the nature of the model (static model *vs* dynamic model, input-output representation *vs* state representation, ...) requires an analysis of the future use of the model (for instance, whether it will be used for predicting the future evolution of the process, or whether it will be used within a control system), and an analysis of the *a priori* knowledge on the phenomena involved in the process;
(ii) the choice of the structure of the model, defined by the number of its inputs, by the number of its outputs, by the type of input-output relationship (linear, polynomial, radial-basis function, multi-layer neural network, etc.), and by its structural parameters (degree of the polynomial approximation, number of radial basis functions, number of neurons, etc.);
(iii) the estimation of the optimal set of adjustable coefficients (synaptic weights in the case of neural net models) of the chosen structure ("identification" in automatic control, "training" in neural network parlance);

The first problem is fully application-dependent: no general statement can be made. The third problem has been investigated in great depth in the case of

linear models [1]; in the case of neural network models, a variety of training algorithms is available [2], and it has been shown that the choice of a training algorithm, in the context of dynamical process modeling, is based on the nature of the noise present in the process to be modeled [3].

In the present paper, we investigate the second problem, namely, that of model selection, which is a key factor for a model to be successful [4]. We suggest a pragmatic model selection procedure for dynamical input-output non-linear models, which features three steps in succession: first, the inputs (external inputs and feedback inputs) of *linear* models of the process around operating points are selected; in a second step, the relevant inputs of the *non-linear* model are selected, thereby determining the order of the model; finally, the structural parameter of the model is determined. An optimized model of a dynamical process is thus derived.

We describe the selection procedure in the case of stable (within the range of operation for which a model is needed), single-input-single-output processes. We assume that the process is NARX:
$y_p(t) = \Phi[y_p(t-1), ..., y_p(t-\nu), u(t-1), ..., u(t-\mu)] + w(t)$
where $\{w(t)\}$ is a gaussian sequence of zero mean independent random variables, ν is the order of the assumed model, and μ is the memory span of the control sequence $\{u(t)\}$.
The following predictor is used:
$y(t) = \Psi[y_p(t-1), ..., y_p(t-n), u(t-1), ..., u(t-m)]$;
We know from [3] that such a predictor (trained with a directed, or teacher-forcing, algorithm) is optimal as a predictor for a NARX process.
If $n = \nu$, if $m = \mu$, and if $\Psi(.)$ is an accurate approximation of $\Phi(.)$, then the predictor is optimal for the process.
In the following, we describe the three steps of the procedure, in the case of a neural network model.

THE PROCEDURE

First step

In the stability domain of the process, operating points (u_i, y_i) are chosen. The process is subjected to time-dependent control sequences of length N in the ranges $[u_i + \Delta u_i, u_i - \Delta u_i]$, such that a *linear* model of the process can be considered valid in each of these ranges. For each operating point, we select, as described below, a linear model which is a satisfactory tradeoff between accuracy and parsimony. At the end of the first step, the set of all inputs which were selected is available for use in the second step of model selection.

For each operating point, we make the assumption that the process can be described as an ARX model :

$$y_p(t) = \sum_{i=1}^{\nu} \alpha_i \, y_p(t-i) + \sum_{i=1}^{\mu} \alpha_{\nu+1} \, u(t-i) + w(t) \ .$$

where ν et μ are unknow parameters.

We consider a training set of size N, and a family of predictors of the form:

$$y(t) = \sum_{i=1}^{n} \theta_i \, y_p(t-i) + \sum_{i=1}^{m} \theta_{n+i} \, u(t-i) \ .$$

The aim of the procedure is to find a predictor such that $n = \nu$, $m = \mu$.

We denote by $\mathbf{y}_p$, $\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$, $\mathbf{x}_{n+1}$, …, $\mathbf{x}_{n+m}$, $\mathbf{w}$, $\mathbf{y}$ the N-vectors, corresponding to the values $y_p(t)$, $y_p(t-1)$, ..., $y_p(t-m)$, u(t-1), ..., u(t-n), w(t), y(t), for t=1 to N; thus:

$\mathbf{y} = [\mathbf{x}_1, \ ... \ \mathbf{x}_M] \, \theta$, where M = m + n.

We have to find M regressors, corresponding to M independent vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ such that the subspace spanned by these vectors is the subspace of smallest dimension containing $E[\mathbf{y}_p]$. In order to find this subspace, we start with a complete model, whose parameters n' and m' are chosen to be larger than can be expected from the *a priori* knowledge available on the process. We thus make the assumption that the subspace H spanned by the M'=n'+m' vectors contains $E[\mathbf{y}_p]$, and we expect to extract the satisfactory subset of significant regressors from the initial set. This could be achieved by computing and comparing all possible regressions; however, this method becomes too expensive for large M'.

In order to decrease the amount of computation, we build from the initial set $\{\mathbf{x}_1, \ldots, \mathbf{x}_{M'}\}$ an ordered set of orthonormal vectors $\{\mathbf{p}_1, \ldots, \mathbf{p}_{M'}\}$ such that the model defined by $\{\mathbf{p}_1, \ldots, \mathbf{p}_k\}$, for all $1 \le k \le M'$, gives a sum of squares of errors (SSE) which is smaller than the SSE given by all other models with k regressors [5].

We first choose, among the M' vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_{M'}\}$, the vector $\mathbf{x}_j$ giving the largest square regression $|\mathbf{p}_1^T \mathbf{y}_p|^2$, with $\mathbf{p}_1 = \mathbf{x}_j / \|\mathbf{x}_j\|$. The (M'-1) remaining $\{\mathbf{x}_i\}$ vectors are orthonormalized with respect to $\mathbf{p}_1$.

Consider the k^{th} step of the ordering procedure, where $\mathbf{p}_1, \ldots, \mathbf{p}_{k-1}$ have been selected. We denote by SSE(k) the SSE obtained with the selected model having k regressors, thus :

$SSE(k-1) - SSE(k) = |\mathbf{p}_k^T \mathbf{y}_p|^2$,

with :

$SSE(0) = \|\mathbf{y}_p\|^2$.

This contribution decreases as k increases. This procedure is iterated M'-1 times for $\mathbf{p}_2$, $\mathbf{p}_3$, ... until completion of the list. Thus :

$$\|y_p\|^2 = \sum_{k=1}^{M'} |\mathbf{p}_k^T \mathbf{y}_p|^2 + SSE(M')$$

where SSE(M') is the sum of squares of errors for the complete model.

Subsequently, the above list is scanned in the inverse order of its construction, and each model is compared with the complete model, using the Log Determinant Ratio Test (LDRT). The number of models we have to take into account is at most equal to M'. Note that the comparison between these models by LDRT is easy (see Appendix for further details about this test), since the variable used to compare the k-regressor model and the complete model is :

$$X_{LDRT} = N \frac{\log\left[SSE(k)\right]}{\log\left[SSE(M')\right]}.$$

We select the smallest predictor model accepted by the test.

In order to further decrease the number of tests, we introduce a simple stopping criterion during the formation of the subset $\{\mathbf{p}_1, \ldots, \mathbf{p}_{M'}\}$: at the k^{th} step, the procedure is terminated if $|\mathbf{p}_k^T \mathbf{y}_p|^2 < \rho \|\mathbf{y}_p\|^2$. The choice of ρ is not critical provided it is small (typically $\rho<10^{-8}$).

In the present work, we use LDRT, but Fisher-Snedecor test, Akaike's Information Criterion (AIC) test are also available (for a review see [4]) and lead to similar results.

Thus, for each chosen operating point, a linear model is available, which achieves a satisfactory tradeoff between accuracy and parsimony. Note that the techniques which are used in the linear context of this step are not computationally expensive, so that a large number of external inputs n and feedback inputs m can be used as a starting model for selection.

At the end of the first step, each regressor which was selected for at least one operating point is available for consideration in the second step of model selection.

Second step

In this step, the process is subjected to large-amplitude control signals corresponding to the conditions of operation which the model is expected to account for. A non-linear model is defined (e.g. a neural network), whose inputs are the set of inputs which were determined during the previous step, and whose structural parameters are deemed to be appropriate for the non-linear input-output function to be accurately approximated (e.g. a neural network with an appropriate, possibly too large, number of neurons, trained by an algorithm which allows an efficient minimization of the SSE). Such methods tend to be computationally expensive, so that the chosen number of neurons should not be excessively large. The best subset of inputs is selected by statistical tests (LDRT or AIC criterion (see appendix)) : we compare the complete non-linear model with all these sub-models with one input less. If all the models are rejected, this step of the procedure is terminated. Otherwise, the best submodel is chosen, and compared with all these sub-models having one input less, and so on.

At the end of this step, a non-linear model M_1 is available, whose inputs have been selected.

Third step

The final step aims at determining the structural parameter of the model: in the case of a neural network model, this parameter is the number of hidden neurons. Here, the accuracy/parsimony tradeoff is expressed by the fact that too large a number of hidden neurons leads to overtraining (small SSE on the training set, large SSE on the test set), whereas too small a number of neurons leads to poor approximation (large SSE on the training set itself). The model M_1 resulting from the previous two steps is considered as the complete model, and models with a smaller number of hidden neurons than M_1 are considered for selection. As in the previous steps, statistical tests are used in order to find a satisfactory tradeoff. Note that most model reduction algorithms used for neural networks aim at eliminating connections [6], whereas this final step aims at eliminating neurons.

EXAMPLE

The efficiency of the above procedure is illustrated by the modeling of a second-order, highly non-linear NARX process, which is simulated by the following equation:

$$y_p(t) = 50 \tanh\left(2.\ 10^{-3}\left[\frac{24 + y_p(t-1)}{3}\, y_p(t-1) - 8\,\frac{u(t-1)^2}{1 + u(t-1)^2}\, y_p(t-2)\right]\right) + 0.5\, u(t-1) + w(t)\ ,$$

where w(t) is white noise with variance $(\sigma_w)^2$. The behaviour of this process is essentially that (i) of a linear first-order low-pass filter for amplitudes smaller than or on the order of 0.1, and (ii) of a second-order, oscillatory, linear ($0.1 < |u| < 0.5$), or non-linear ($0.5 < |u| < 5$) system for larger amplitudes; it becomes almost static for positive signals of very large amplitude; in addition, it is not symmetrical with respect to zero. Figure 1 shows the response of the process to steps of random amplitude in the region of interest, with $(\sigma_w)^2 = 10^{-2}$.

First step

The operating points were $u_i = \{-10, -8, -5, -2, -1, -0.5, 0.1, 1, 2, 5, 8, 10\}$. At each of these points, a uniformly distributed random sequence was added to the control input, with maximum amplitude $\Delta u_i = 0.1$ ($\sigma_u^2 = 3.10^{-3}$). The initial model was chosen to have n' = m' = 100. The training sequence was of length N = 1000. The orthonormalization procedure retained 15 inputs, and the subsequent LDRT tests (with 1% risk) led to the selection of n+m = 2 to 5 inputs, depending on the operating points.

Second step

The training set was a sequence of large-amplitude steps, such as shown on Figure 1. M_1 was a fully connected neural network, with the 5 inputs (n = 3, m = 2) selected in the first step, and with 10 hidden neurons. After training, the variance of the prediction error (as estimated by SSE/N) was on the same order of magnitude as σ_W , which shows that the network was sufficiently large, and had been trained efficiently. Subsequently, the networks obtained by suppressing 1 input, then 2 inputs, etc., were trained and submitted to the LDRT procedure, as illustrated on Table 1: the full model M_1 is compared to $M_2, M_3, ..., M_6$. The test selected only M_2 and M_4 (the deletion of one input leads to the deletion of 11 connections; the corresponding value of the χ^2 variable for a 1% risk is 24.7). Since the SSE of M_4 was smallest, it was selected for comparison with all models smaller than M_4[1]; M_7 is the only three-input model which was selected. All models smaller than M_7 were rejected. Therefore, M_7 was finally accepted. The success of the procedure is shown by the fact that M_7 is indeed the only model which has the same inputs as the simulated process. A similar result is obtained if the AIC test is used.

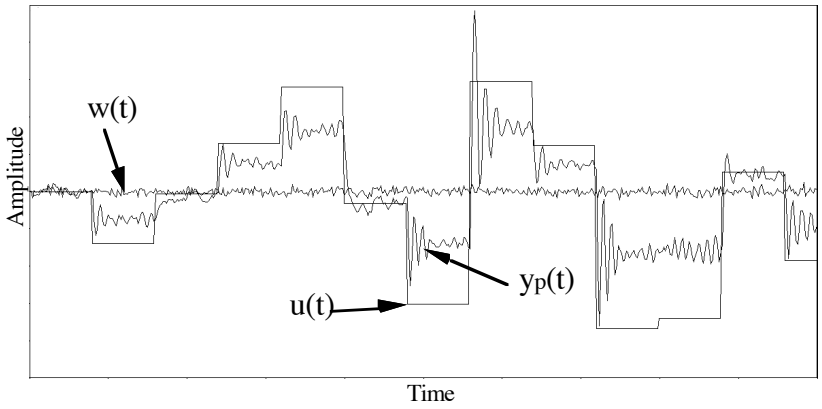


FIGURE 1
Sequence of control input and process output.

Third step

Model selection is performed on neural nets having the inputs of M_7, and 0 to 10 hidden neurons, with the same training set for all nets. The result of the selection depends on σ_W. With $\sigma_W = 10^{-2}$, a model with 9 neurons is selected. With $\sigma_W = 10^{-1}$, the same inputs are selected by the first two steps and the third step leads to a neural network with 4 neurons. As should be

[1] Actually, the SSE's of M_2 and M_4 are very close; if M_2 is selected instead of M_4, the same result is obtained, since M_7 is a sub-model of both M_2 and M_4.

Model	$y_p(t-1)$	$y_p(t-2)$	$y_p(t-3)$	$u(t-1)$	$u(t-2)$	$SSEx10^2$	X_{LDRT}
1	X	X	X	X	X	19.1	
2	X	X	X	X	–	19.6	11
3	X	X	X	–	X	13.0	832
4	X	X	–	X	X	19.5	10
5	X	–	X	X	X	31.6	218
6	–	X	X	X	X	31.8	221
7	X	X		X	–	19.6	1.2
8	X	X		–	X	97.7	697
9	X	–		X	X	11.8	980
10	–	X		X	X	39.4	1303
11	X	X		–		25.4	1114
12	X	–		X		18.7	978
13	–	X		X		18.2	968

TABLE1
Models labelled by boldface figures are those
whose inputs include the inputs of the process.

expected, the procedure selects a smaller number of neurons if the noise level is high than if it is low.

CONCLUSION

A pragmatic three-step procedure for non-linear dynamical model selection has been proposed, which uses statistical tests at various levels of model reduction. It relies on the fact that efficient training procedures are available. It allows the selection of the delayed external inputs, of the feedback inputs (hence the determination of the order of the model) and of the structural parameters such as the number of hidden neurons. Its main shortcoming seems to be the fact that its application is subject to the availability of two types of data from the process, namely, small-signal responses around chosen operating points, and large-signal responses in "normal" operation. Its efficiency is shown on an illustrative example: the neural modeling of a highly non-linear NARX process.

APPENDIX

The Logarithm Determinant Ratio Test (LDRT) [4]

The problem of the selection of one model out of two can be formulated as a statistical testing problem. We suppose that an accurate model M_1, described by the vector of paramters θ, is available to explain a set of N experimental data. The null hypothesis states that a part θ_2 of the vector parameter θ is

equal to zero; if this assumption is true, $\theta=[\theta_1, \theta_2]$ can be reduced to θ_1. If the alternative hypothesis is true, then θ_2 cannot be taken equal to a zero vector. A very efficient test to solve such a problem is the Likelihood Ratio Test (LRT), but this test requires the expression of the likelihood function. In our case, with very large N, it reduces to the Log Determinant Ratio Test (LDRT) : under the null hypothesis $\theta_2=0$, with a scalar output, the distribution of the statistics :

$$X_{LDRT} = N \log \frac{SSE(\theta_1)}{SSE(\theta)}$$

converges to a chi-square distribution with $\dim(\theta_2)$ degrees of freedom.

The Akaike's Information Criterion Tests (AIC)

The AIC is an alternative way of selecting a model from a set of models, using statistical tests. For each model of the set, we compute the AIC value :

$AIC = 2 N \log(SSE/N) + 2M$

where N is the number of data and M is the number of parameters of the model.

The model corresponding to the smallest AIC value is thus selected as the best model of the set, with respect to this criterion. This procedure requires no assumptions on the models. There exist more efficient variants of the classical AIC [4], such as the AIC*, used in this work :

$AIC^*= 2 N \log(SSE/N) + 4 M$

.

REFERENCES

[1] See for instance:
L. Ljung, System Identification: Theory for the User: Prentice Hall, 1987.
G.C. Goodwin, R.L. Payne, Dynamic System Identification: Experiment Design and Data Analysis: Academic Press,1977.

[2] O. Nerrand, P. Roussel-Ragot, L. Personnaz, G. Dreyfus, "Neural Networks and Non-linear Adaptive Filtering: Unifying Concepts and New Algorithms", Neural Computation, vol. 5, pp.165-197, 1993..

[3] O. Nerrand, P. Roussel-Ragot, D. Urbani, L. Personnaz, G. Dreyfus, "Training Recurrent Neural Networks: Why and How ? An Illustration in Dynamical Process Modeling", IEEE Transactions on Neural Networks, vol. 5, pp. 178-184, 1994.

[4] I.J. Leontaritis, S.A. Billings, "Model Selection and Validation for Non-Linear Systems", International Journal of Control, vol. 1, pp. 311-341, 1987.

[5] S. Chen, S.A. Billings, W. Luo, "Orthogonal Least Squares Methods and their Application to Non-Linear System Identification" International Journal of Control, vol. 50, pp. 1873-1896, 1989.

[6] R. Reed, "Pruning Algorithms - A Survey", IEEE Transactions on Neural Networks, vol. 4, pp. 740-747, 1993.