Identification et commande des systèmes non linéaires : Utilisation des modèles de type NARMA

Brahim Tlili

UNIVERSITE DE TUNIS EL MANAR
ECOLE NATIONALE D'INGENIEURS DE TUNIS

THESE

Présentée pour obtenir le

Diplôme de Doctorat

SPECIALITE GENIE ELECTRIQUE

Par

Tlili Brahim

IDENTIFICATION ET COMMANDE DES SYSTEMES NON LINEAIRES : UTILISATION DES MODELES DE TYPE NARMA

Soutenue le 29 juillet 2008 devant le jury composé de :

Mme.	**MOUFIDA KSOURI**	**Présidente**
MM.	**RIDHA BEN ABDENNOUR**	**Rapporteur**
	FARHAT FNAIECH	**Rapporteur**
	MEKKI KSOURI	**Examinateur, directeur de thèse**
	FAOUZI BOUANI	**Examinateur**

TABLE DES MATIERES

ABREVIATIONS

MCR	Méthode des Moindres Carrées Récursifs
Méthode AGB	Méthode d'identification basée sur l'Algorithme Génétique Binaire
Méthode K	Méthode d'identification basée sur l'algorithme de Kortmann
Méthode RN	Méthode d'identification basée sur les Réseaux de Neurones Artificiels
Méthode NM	Méthode d'optimisation de Nelder-Mead
Méthode ROS	Méthode d'optimisation de Rosenbrock
AIC	Akaike Information Criterion
BIC	Bayes's Information Criterion
CFON	Constrained First Objective Next
FPE	Final Prediction Error
GPC	Generalized Predictive Control
MBC	Model Based Control
MBPC	Model Based Predictive Control
NARMA	Nonlinear AutoRegressive Moving Average
NMPC	Nonlinear Model Predictive Control
OVF	Overal F_test

NOTATIONS

u	Commande appliquée au système
y	Sortie du système
yc	Consigne
$\hat{y}$	Sortie prédite à l'aide du modèle
q	Degré de non linéarité du modèle
Hc	Horizon de commande
Hp	Horizon de prédiction sur la sortie
λ	Coefficient de pondération
ΔU	Vecteur des incréments de commande
J, J_1, J_2, J_3, J_4	Critères de performances
n_1	Nombre de sorties du système multivariable
m_1	Nombre d'entrées du système multivariable
U	Vecteur des entrées
Y	Vecteur des sorties
Φ	Vecteur d'observation
$\hat{\Theta}^T$	Vecteur des paramètres transposé
$\bar{y}_i$	Valeur moyenne de la sortie i
λj	Coefficient de pondération sur la commande (pour l'entrée j)
Hc_j	Horizon de commande sur l'entrée j
Hp_j	Horizon de prédiction sur la sortie j
R^2_{ajust}	Coefficient de corrélation ajusté
R^2_{mult}	Coefficient de corrélation multiple
R^2_{tot}	Coefficient de corrélation total
PF_i	Critères d'information partiels
NM-V_0 et ROS-V_0	Versions originales de Nelder-Mead et de Rosenbrock
NM-V_1, NM-V_2 et NM-V_3	Versions de Nelder-Mead avec améliorations
ROS-V_1, ROS-V_2 et ROS-V_3	Versions de Rosenbrock avec améliorations

LISTE DES FIGURES

LISTE DES TABLEAUX

Introduction générale

Dans ce travail, on s'intéresse à la commande des systèmes non linéaires. L'approche utilisée est basée sur la commande prédictive à base de modèle MBPC (Model Based Predictive Control). En effet, ce type de commande présente plusieurs avantages. Comme illustré dans les travaux de Clarke [Clarke, 1988], la commande prédictive s'avère être une structure suffisamment complète proposée pour résoudre un problème de commande très général. Elle peut également conduire à un système asservi stable pour des paramètres de réglage donnés. De plus, cette stratégie a montré son efficacité, sa flexibilité et son succès dans des applications industrielles, même pour des systèmes à faible période d'échantillonnage.

Cependant, la mise en œuvre de la commande prédictive à base de modèle nécessite :

- La définition d'un modèle numérique qui peut être défini comme un ensemble d'équations mathématiques, permettant de représenter au mieux le comportement du système réel afin de prédire le comportement futur de celui-ci. Cette particularité permet de classer la commande prédictive dans la grande famille des commandes à base de modèles MBC (Model Based Control).

- Le calcul d'une séquence de commandes futures qui est obtenue par la minimisation d'un critère de performance quadratique, sur un horizon fini, comportant un terme lié aux erreurs de prédiction futures (écarts entre la sortie prédite du système et la consigne) et un terme lié à la commande.

Dans la pratique plusieurs types de modèles sont utilisés pour la synthèse de la loi de commande, parmi lesquels les modèles de type entrée-sortie. Dans la famille des modèles de type entrée-sortie, on distingue plusieurs types tels que le modèle de Volterra [Boyd, 1985], [Hazem, 2005], [Hernando, 1988], le modèle de Wiener [Westwick, 1996] [Wigren, 1994], [Zhu, 1999], le modèle de Hammerstein [Bai, 2008], [Bai, 2003], [Bai, 1990], [Greblicki, 2000], [Narendre, 1966], les modèles cascades Wiener-Hammerstein [Bai, 2002], [Bauer, 2002], [Hunter, 1986], [Zhu, 2002] et le modèle NARMA (Nonlinear AutoRegressive Moving Average) [Billings, 1981], [Leontaritis, 1985a]. L'utilisation d'un modèle de type

NARMA permet de réduire d'une manière considérable le nombre de termes, et par conséquent le nombre de paramètres du modèle. Ceci, conduit à la réduction du temps de calcul de la loi de commande comparé aux régulateurs prédictifs basés sur les autres types de modèles. Ces avantages nous ont poussés à l'exploitation de ce modèle dans le calcul de la loi de commande. Dans ce travail, nous présentons d'abord la méthode de Kortmann [Kortmann, 1988], puis nous proposons deux autres méthodes basées sur les approches non conventionnelles pour l'identification du modèle NARMA.

La loi de commande est obtenue par minimisation d'un critère de performance non convexe. La minimisation du critère de performance est généralement effectuée par une approche classique tel que la méthode du gradient, qui nécessite le calcul de la dérivée de ce dernier [Filali, 2002], [Kenneth, 1999], [Yonghong, 1996]. L'emploi de cette méthode a souvent été couronné de succès, puisqu'elle dispose de nombreuses qualités (preuve de convergence, faible nombre d'évaluations nécessaires, faible dépendance vis-à-vis de la dimension du problème, prise en compte des contraintes). Néanmoins, plusieurs difficultés limitent son usage. Pour certains problèmes, la fonction de coût est non différentiable par nature ou présente des discontinuités (par exemple les problèmes de type Min Max), il s'en suit que le calcul du gradient est délicat (différences finies, coûteux, entaché d'erreur, mise en oeuvre délicate pour des problèmes complexes ou présence de minima locaux). Dès lors, cette approche ne peut pas être employée sans aménagement spécifique, l'utilisation des méthodes sans gradient se révèle alors plus adéquate.

Ce mémoire comporte quatre chapitres, il est organisé comme suit :

Le premier chapitre est un chapitre d'introduction qui présente la problématique générale de la commande prédictive à base de modèle. Nous présentons dans un premier temps les modèles de type entrée-sortie les plus connus, en particulier le modèle NARMA. Ensuite, nous nous intéressons à la stratégie et au principe de la commande prédictive non linéaire.

Le deuxième chapitre sera consacré à l'identification paramétrique et structurelle du modèle NARMA. Trois approches sont utilisées pour l'identification du modèle de type NARMA, la première exploite l'algorithme de régression par pas échelonné, proposé par Kortmann et Unbehauen [Kortmann, 1988]. La deuxième méthode est basée sur l'algorithme génétique avec sa représentation binaire, et la troisième approche constitue une combinaison

des réseaux de neurones artificiels à fonction d'activation polynomiale avec l'algorithme génétique sous sa représentation réelle.

Dans le troisième chapitre, nous nous intéresserons à la synthèse d'un régulateur prédictif sous contraintes qui est basé sur le modèle NARMA. La minimisation du critère de performance est effectuée avec deux méthodes qui n'utilisent pas la dérivée. La première méthode est basée sur l'algorithme de Nelder-Mead. La deuxième approche utilise la méthode de Rosenbrock qui est également une méthode d'optimisation numérique. Une fonction de pénalité a été employée pour le traitement des contraintes.

Le quatrième chapitre traite les systèmes multivariables non linéaires. Dans ce cadre, nous proposons l'extension de la troisième approche utilisée dans le chapitre 2, pour l'identification du modèle NARMA dans le cas multivariable. Cette approche a été validée par une application pratique sur un procédé réel, qui présente un système à trois réservoirs d'eau interconnectés. Ensuite, nous proposons l'extension des méthodes d'optimisation présentées dans le chapitre 3, pour traiter la commande prédictive non linéaire des systèmes multivariables.

Une conclusion générale ainsi que les perspectives de ces travaux clôturent ce rapport de synthèse.

Chapitre 1

Contexte et état de l'art

Chapitre 1 Contexte et état de l'art

Chapitre 1

Contexte et état de l'art

1.1. Introduction

L'objectif de ce chapitre est de situer ce travail dans son contexte scientifique qu'est la commande des systèmes non linéaires. Dans ce cadre, la commande prédictive non linéaire, en utilisant les modèles de type NARMA, est développée.

La stratégie de commande à base de modèle (MBC : Model Based Control) nécessite un modèle qui décrit le comportement dynamique du système. L'utilisation d'un modèle linéaire ou linéarisé autour d'un point de fonctionnement est souhaitable lorsqu'il s'agit de l'identification des systèmes simples, puisque ce type de modèle possède peu de paramètres tout en restant facile à déterminer. Malheureusement la complexité des systèmes industriels limite l'utilisation des modèles linéaires et plusieurs approches de modélisation des systèmes non linéaires ont été développées. Dans ce travail, on s'intéressera à la commande prédictive non linéaires sous contraintes en utilisant les modèles de type NARMA.

1.2. Sur l'identification et la modélisation

Dans la pratique, deux démarches sont possibles pour l'obtention du modèle d'un système. La première approche est analytique et la deuxième est expérimentale [Ben Abdennour, 2001]. La modélisation analytique utilise les équations physico-chimiques qui régissent le comportement dynamique du système. La modélisation expérimentale traite le système comme une boîte noire et utilise les mesures expérimentales pour extraire le modèle. Bien que la modélisation analytique offre une description précise facilitant la compréhension du comportement des procédés, elle est souvent complexe et peu utilisée pour la commande. Les automaticiens utilisent surtout la modélisation expérimentale car elle présente une grande souplesse lors de la synthèse des régulateurs. Cette dernière modélisation comporte les modèles de type entrée-sortie, appelés aussi modèles de fichiers [Ben Abdennour, 2001], [Landau, 1988]. Dans [Giannakis, 2001] les auteurs ont donné une liste bibliographique, regroupant plus de 1400 références, qui porte sur l'identification des systèmes non linéaires. Cette liste présente la majorité des travaux publiés dans la période 1979-2000.

Les modèles non linéaires de types entrée/sortie existent sous plusieurs formes, parmi lesquels on distingue le modèle de Volterra [Hazem, 2004], [Hazem, 2005], [Hernando, 1988], le modèle de Hammerstein, le modèle de Wiener [Haber, 1990] et le modèle NARMA [Billings, 1981], [Leontaritis, 1985a].

a- Modèle de Volterra :

Le modèle de Volterra appartient à la classe des modèles polynomiaux non récursifs. La relation d'entrée-sortie qui lui est associée est :

$$y(k) = \sum_{i=1}^{\infty} \sum_{j_1=0}^{\infty} \sum_{j_2=0}^{\infty} \dots \sum_{j_N=0}^{\infty} h_i(j_1,\dots,j_N) \prod_{l=1}^{N} u(k-j_l) \qquad (1\text{-}1)$$

où u, y et h_i sont respectivement l'entrée, la sortie et le noyau de Volterra d'ordre N. Les j_l, $l = 1,\dots,N$ sont des entiers. Un noyau d'ordre N peut être vu comme une généralisation à l'ordre N de la notion de réponse impulsionnelle bien connue dans le cas des systèmes linéaires. En appelant M la mémoire du système, on peut considérer le modèle tronqué, à temps discret, à l'ordre N suivant :

$$y(k) = \sum_{i=1}^{N} \sum_{j_1=0}^{M-1} \sum_{j_2=0}^{M-1} \dots \sum_{j_N=0}^{M-1} h_i(j_1,\dots,j_N) \prod_{l=1}^{N} u(k-j_l) \qquad (1\text{-}2)$$

b- Modèles de Wiener et de Hammerstein

Ces modèles sont obtenus en mettant en cascade un modèle dynamique linéaire et un bloc non linéaire statique. Lorsque la non-linéarité statique suit le bloc dynamique linéaire, le modèle est dit modèle de Wiener (figure 1-1); dans le cas contraire, on parle de modèle de Hammerstein (figure 1-2).

Modèle de Wiener

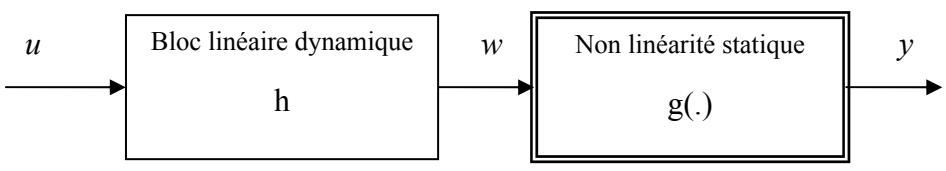


Figure 1-1 : Modèle de Wiener

Si la non-linéarité statique est de forme polynomiale d'ordre N et le bloc linéaire dynamique représente une réponse impulsionelle finie FIR (Finite Impulse Response), la relation d'entrée-sortie de ce modèle sera comme suit :

$$w(k) = \sum_{j_1=0}^{M-1} h(j_1)\, u(k - j_1) \tag{1-3}$$

$$\text{et } y(k) = g\left[w(k)\right] = \sum_{i=0}^{N} c_i w^i(k) \tag{1-4}$$

Modèle de Hammerstein

Le modèle de Hammerstein peut être vu comme étant le dual du modèle de Wiener composé des mêmes blocs mais placés dans un ordre inversé. On a :

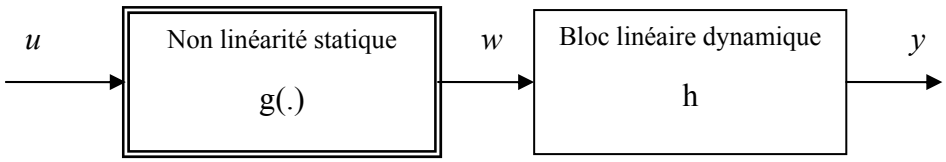


Figure 1-2 : Modèle de Hammerstein

Comme dans le cas du modèle de Wiener, si g(.) est un polynôme et h est une FIR, alors la relation de la sortie du modèle sera comme suit :

$$w(k) = g\left[u(k)\right] = \sum_{i=0}^{N} c_i u^i(k) \tag{1-5}$$

$$\text{et } y(k) = \sum_{j_1=0}^{M-1} h(j_1)\, w(k - j_1) \tag{1-6}$$

Modèles cascades

A partir des combinaisons des modèles de Wiener et de Hammerstein, on distingue d'autres types de modèles. Le modèle Wiener-Hammerstein (LNL, figure 1-3) est défini par un bloc dynamique linéaire en cascade avec un bloc non linéaire statique suivi par un autre bloc dynamique linéaire et le modèle Hammerstein-Wiener (NLN, figure 1-4) est défini par un bloc non linéaire statique en cascade avec un bloc dynamique linéaire suivi par un autre bloc non linéaire statique.

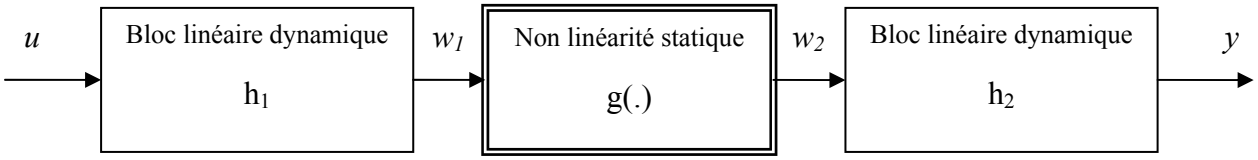


Figure 1-3 : Modèle de Wiener-Hammerstein (LNL)

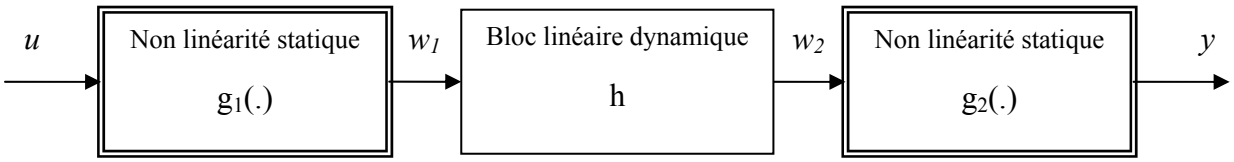


Figure 1-4 : Modèle de Hammerstein-Wiener (NLN)

c- Modèle NARMA

La structure dynamique d'un modèle NARMA opérant dans un environnement déterministe, est régie par l'équation mathématique discrète suivante [Billings, 1981], [Kortmann, 1988] :

$$\begin{aligned}\hat{y}(k) = \bar{y} &+ \sum_{i=1}^{m} b_i u(k-i) + \sum_{i=1}^{m}\sum_{j=i}^{m} b_{ij} u(k-i)u(k-j) \\ &+ \ldots + \sum_{i=1}^{m}\ldots\sum_{v=p}^{m}\sum_{l=v}^{m} b_{i\ldots l} u(k-i)\ldots u(k-l) \\ &+ \sum_{i=1}^{n} a_i y(k-i) + \sum_{i=1}^{n}\sum_{j=i}^{n} a_{ij} y(k-i)y(k-j) \\ &+ \ldots + \sum_{i=1}^{n}\ldots\sum_{v=p}^{n}\sum_{l=v}^{n} a_{i\ldots l} y(k-i)\ldots y(k-l) + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} u(k-i)y(k-j) \\ &+ \ldots + \sum_{i=1}^{m}\ldots\sum_{v=1}^{m}\sum_{l=1}^{n} c_{i\ldots l} u(k-i)\ldots u(k-v)y(k-l) \\ &+ \ldots + \sum_{i=1}^{m}\sum_{j=1}^{n}\ldots\sum_{l=1}^{n} c_{ij\ldots l} u(k-i)y(k-j)\ldots y(k-l).\end{aligned} \tag{1-7}$$

$\bar{y}$ représente la valeur moyenne.

Les paramètres *m* et *n* représentent, respectivement, l'ordre de la régression sur l'entrée *u(k)* et l'ordre de la régression sur la sortie *y(k).*

Comme on peut le constater, avec la relation ci-dessus, le modèle NARMA est un modèle non linéaire qui présente l'avantage d'être linéaire par rapport à ses paramètres, la non linéarité s'exprimant par rapport à l'entrée et à la sortie du système. L'implication pratique de cette propriété est que plusieurs des résultats obtenus pour des problèmes d'estimation paramétrique linéaire peuvent être repris pour ce type de modèles, moyennant quelques aménagements appropriés.

Dans [Hernando, 1988], [Hazem, 2004] et [Hazem, 2005], les auteurs ont proposé des méthodes pour l'identification des modèles de Volterra. Cependant, ce type de modèle nécessite un nombre élevé des paramètres et par conséquent le temps de calcul de la loi de commande sera élevé. Le modèle NARMA fournit une représentation unifiée pour une large classe des systèmes non linéaires [Leontaritis, 1985a]. Billings et Leontaritis [Billings, 1981]

ont prouvé que plusieurs modèles bien connus tels que le modèle Hammerstein, de Wiener et les modèles bilinéaires sont des cas spéciaux du modèle NARMA. Malgré le nombre important de termes existants dans la forme générale du modèle NARMA, ce dernier nécessite un nombre limité parmi les anciennes mesures et les couples croisés (entrée-entrée, entrée-sortie et sortie-sortie) dans l'expression finale de la sortie [Liu, 2003].

Plusieurs approches ont été développées pour la détermination des modèles de type NARMA. L'algorithme de régression par pas échelonné, qui est basé sur des tests statistiques, assure l'estimation des paramètres ainsi que la sélection des termes qui forment le modèle NARMA [Kortmann, 1988]. L'exploitation des réseaux de neurones artificiels à fonction d'activation polynomiale a été également développée dans les travaux de [Ki, 1997]. Cependant, dans ce papier aucune approche n'a été proposée pour la détermination des coefficients de la fonction d'activation des neurones.

Dans [Sheng, 2001] et [Sheng, 2003], les auteurs ont utilisé la géométrie affine et la minimisation de la distance «hyper surface» pour l'estimation des paramètres du modèle NARMA. Cette approche qui est basée sur la combinaison de l'algorithme OPS (Optimal Parameter Search) avec la méthode TLS (Total Least Squares), permet d'estimer les paramètres du modèle NARMA même en présence d'un bruit significatif. Cependant, cette approche nécessite un temps de calcul élevé et la solution analytique ne peut pas être calculée pour les systèmes de dimension supérieure à trois [Sheng, 2003].

Dans [Ruano, 2003], l'optimisation multi objectif basée sur les algorithmes génétiques MOGA (Multi Objective Genetic Algorithm) a été exploitée pour la détermination du modèle NARMA. Dans ce cas, l'algorithme MOGA a été utilisé pour réaliser un compromis entre plusieurs critères de performances. Ces critères sont exprimés en fonction de l'ordre du modèle, de l'erreur de modélisation et du nombre de termes du modèle. Dans le cas linéaire, l'ordre des régressions sur l'entrée et sur la sortie du modèle ARIMA a été également déterminé à l'aide des algorithmes génétiques à codage binaire [Ong, 2005].

Les modèles NARMA qui sont non linéaires par rapport aux anciennes valeurs de la sortie mais linéaires par rapport à la valeur actuelle de l'entrée ont été également développés dans [Narendra, 1997] et [Adetona, 2004]. Ces modèles, obtenus en utilisant un développement en série de Taylor des modèles neuronaux, sont utiles pour la synthèse des régulateurs. En effet, la loi de commande peut être obtenue par minimisation d'un critère de performance convexe.

1.3. Sur la commande prédictive à base de modèle

1.3.1. Un aperçu historique de la commande prédictive

La commande prédictive (MPC) est née d'un besoin réel dans le monde industriel. Un besoin de système de régulation capable de performances plus élevées que les contrôleurs classiques, tel que le régulateur de type PID, tout en respectant des contraintes de fonctionnement et de production toujours plus sévères. Historiquement, cette approche a commencé à donner ses premiers résultats théoriques et pratiques à la fin des années 1970. Sa mise en œuvre pratique dans l'industrie a vu le jour dans le domaine pétrolier et pétrochimique, notamment avec les travaux pionnier de Richalet en 1976. Les publications de Richalet et al. [Richalet, 1976] [Richalet, 1978] présentaient la commande prédictive dite heuristique MPHC (Model Predictive Heuristic Control), qui fut connue plus tard sous le nom de commande algorithmique MAC (Model Algorithmic Control) et qui utilise la réponse impulsionnelle du système, tronquée aux N premiers échantillons [Camacho, 1999]. Cette approche c'est vite étendu à d'autres industries grâce à ses succès incontestables dans l'industrie pétrolière sérieusement éprouvée par des raisons économiques. En effet, le but économique recherché par des commandes évoluées, et par MPC en particulier, et de réduire les coûts de production en ramenant le point de régulation (setpoint), à des niveaux proches des contraintes de fonctionnement et/ou de production. Ceci est obtenue en réduisant la variance donnée par un contrôleur classique, tel que PID, en utilisant une commande évoluée, e.g. MPC, le point de régulation est ramené vers un niveau plus bas engendrant un coup de production moindre.

En 1980 apparaît la commande matricielle dynamique DMC (Dynamic Matrix Control) présentée par Cutler et Ramaker [Cutler, 1980], qui utilise explicitement le modèle de la réponse indicielle du système pour prédire l'effet sur la sortie des commandes futures. Celles-ci étaient calculées par la minimisation de l'erreur prédite qui était répétée à chaque période d'échantillonnage avec les dernières mesures fournies par le processus. Ces formulations étaient heuristiques et algorithmiques et tiraient parti du potentiel croissant des ordinateurs de l'époque.

La commande GPC (Generalized Predictive Control) développée par Clarke et al. en 1987 [Clarke, 1987], qui applique des idées de la commande GMV (Generalized Minimum Variance) [Clarke, 1979] est sans doute la plus populaire actuellement. Dans les travaux de Morari [Morari, 1994], la commande MPC a également été formulée dans l'espace d'état.

Ceci permet non seulement d'utiliser des théorèmes bien connus de la théorie de la représentation d'état, mais facilite aussi la généralisation de la commande à des cas plus complexes tels que les systèmes avec des perturbations stochastiques et du bruit dans les variables mesurées. Le principe de l'horizon fuyant, l'une des idées centrales de la commande MPC, fut quant à lui proposé en 1963 par Propoi [Propoi, 1963] dans le cadre du « retour optimal en boucle ouverte », et a été largement repris ensuite dans les années soixante-dix.

1.3.2. Principe de la commande prédictive

La commande prédictive MPC (Model Predictive Control) est une méthode relativement récente, elle n'a connu un réel essor dans l'industrie que depuis le milieu des années 80. Actuellement, cette approche représente l'une des techniques de commandes les plus avancées en automatique. En effet, sa formulation intègre des concepts tirés de la commande optimale, de la commande stochastique, de la commande de systèmes à temps morts, de la commande avec modèle interne, de la commande multivariable et prend en compte les références futures lorsqu'elles sont connues d'avance. En outre, des contraintes sur la commande et sur la sortie peuvent être considérées en pratique avec ce type de commande.

Cette méthode se repose sur les idées suivantes :

- utilisation d'un modèle pour prédire les sorties du procédé à des instants futurs [Borne, 1997] [Clarke, 1989],
- calcul de la séquence des commandes à appliquer au système de façon à minimiser un critère à horizon fini portant sur l'écart entre la sortie prédite et la sortie future désirée,
- à chaque instant d'échantillonnage, l'horizon de prédiction est déplacé vers le futur, et seule la première valeur des commandes calculées est effectivement appliquée au système (notion d'horizon fuyant).

La stratégie de commande prédictive MPC, appelée aussi commande à horizon glissant ou fuyant, est similaire à la stratégie utilisée pour le pilotage d'un avion. Le pilote connaît la trajectoire de référence désirée sur un horizon de commande fini (son champ visuel), et en prenant en compte les caractéristiques de son avion, il décide quelles actions (accélérer, décélérer, monter ou abaisser en altitude) doit prendre afin de suivre la trajectoire désirée.

Seule la première action est exécutée à chaque instant, et la procédure est répétée à nouveau pour les prochaines actions. Le principe général de MPC, illustré par la figure 1-5 dans le domaine temporel, se caractérise par les éléments suivants :

- à l'aide du modèle du processus, on prédit à chaque instant *k* les sorties futures, sur un horizon déterminé de taille *Hp* (horizon de prédiction),
- calcul de la suite de commandes à appliquer au système, de façon à minimiser un critère quadratique sur un horizon fini. Le critère quadratique dépend des erreurs de prédiction futures qui représentent l'écart entre la sortie prédite du système et la trajectoire de référence future et de l'énergie de la commande. Cette optimisation a pour objectif de garder le processus aussi proche que possible de la trajectoire de référence. Ainsi, une séquence de commandes futures sera générée par le calculateur. Des hypothèses peuvent être faites sur la structure de la loi de commande future, cette dernière sera constante à partir d'un instant donné *Hc* (horizon de commande),
- parmi la séquence de commande calculée juste le premier élément *u(k)* sera appliqué à l'entrée du système, tous les autres éléments de la séquence sont omis,
- répétition de la procédure complète de calcul à chaque période d'échantillonnage, selon le principe de l'horizon fuyant.

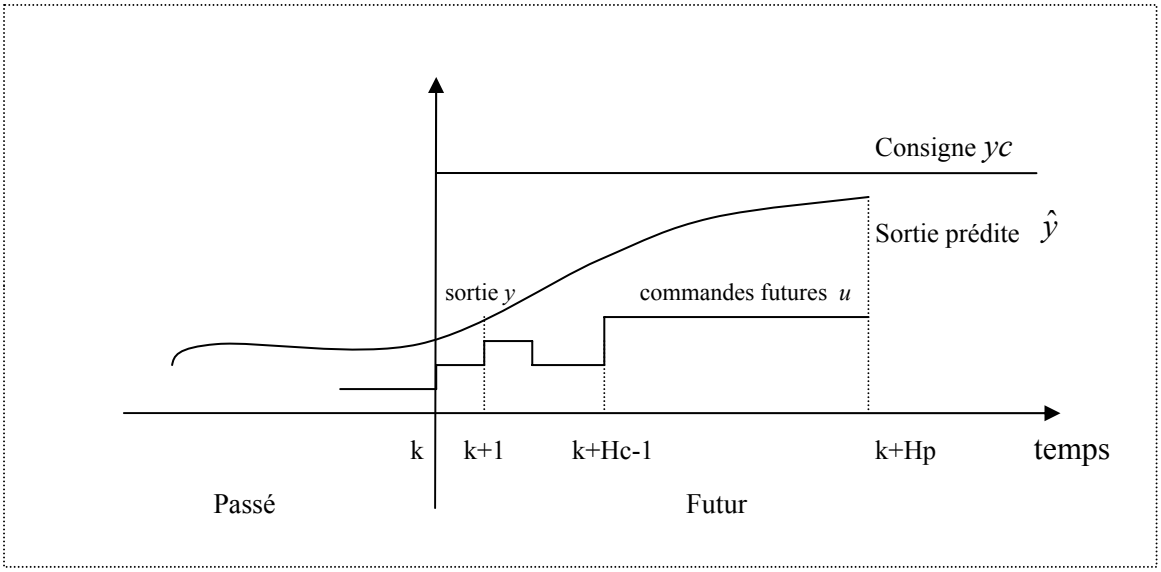


Figure 1-5 : Représentation temporelle du principe de la commande prédictive

La figure 1-6 représente un schéma bloc simplifié qui caractérise la commande prédictive à base de modèle. Le pointillé correspond aux calculs effectués par le calculateur.

Avec $yc(k)$ consigne.

$y(k)$ sortie du processus.

$\hat{y}(k)$ sortie prédite à l'aide du modèle.

$u(k)$ commande appliquée au système.

CNA : Convertisseur Numérique Analogique.

CAN : Convertisseur Analogique Numérique.

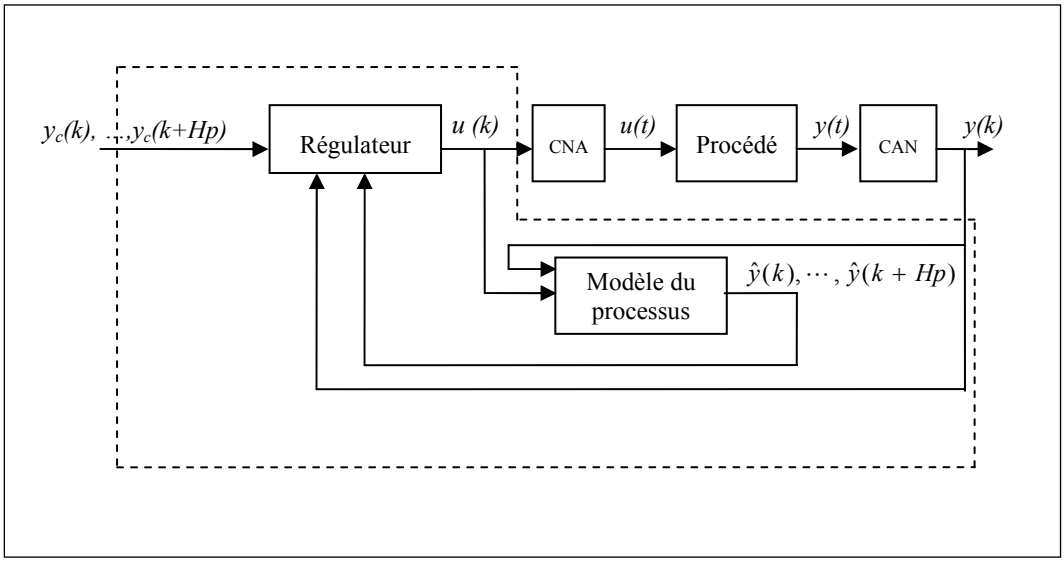


Figure 1-6 : Schéma de principe d'une commande prédictive à base de modèle

Aujourd'hui, la commande prédictive représente l'une des techniques modernes de commande parmi les plus utilisées dans la pratique industrielle [Boucher, 1996]. Cette technique, présente un certain nombre d'avantages par rapport aux autres méthodes de commande. Parmi lesquels, on distingue :

- son principe très intuitif et le réglage relativement facile de ses paramètres qui la rendent accessible même aux personnes ayant des connaissances limitées en automatique,
- elle peut être utilisée pour commander une grande variété de processus, ceux avec des dynamiques simples ou complexes, par exemple les systèmes à retard, à phases non minimales ou instables,
- elle traite, également, les systèmes multivariables,

- elle est capable intrinsèquement de compenser les retards et les temps morts,
- le traitement de contraintes sur le système à commander peut être inclus systématiquement dans la définition du correcteur,
- elle est très utile lorsque les consignes ou trajectoires à suivre sont connues à l'avance (ce qui est le cas dans certains processus industriels).

La plupart des algorithmes de commande prédictive reposent sur des modèles internes linéaires. La raison de cette approche linéaire se fonde dans la maîtrise complète de la théorie des systèmes linéaires. En effet, un modèle linéaire utilisé avec une approche prédictive donne un résultat analytique, c'est-à-dire, une loi de commande linéaire sous forme d'une équation mathématique. Un tel résultat permet le calcul de la commande future en des temps très courts, ceci avait une importance non négligeable quand les calculateurs n'offraient pas la rapidité de calcul désirée.

Toutefois, les procédés à contrôler se révéleront de plus en plus non linéaires. Les méthodes linéaires arrivent donc à leurs limites et l'utilisation des modèles non linéaires, et par conséquent la commande prédictive non linéaire NMPC (Nonlinear Model Predictive Control) commence à s'imposer.

Grâce à ses concepts intuitifs et aux bons résultats obtenus, la commande prédictive a été implantée dans un grand nombre d'applications industrielles. Ces applications industrielles ont toutes un dénominateur commun : la connaissance de la trajectoire à suivre par le système dans le futur, au moins sur un certain horizon. En 1997, Qin et Badgwell [Qin, 1997] recenseront plus de 2000 applications industrielles commandées avec MPC, dont 600 procédés commandés avec NMPC.

1.4. Conclusion

L'objet de ce chapitre a été de situer la problématique de la mise en place d'un correcteur prédictif à base de modèle décrivant le type de modèle utilisé, à savoir le modèle NARMA.

Nous avons ainsi présenté les différents modèles (Volterra, Wiener, Hammerstein) et nous avons donné le principe de la commande prédictive.

L'objet du chapitre suivant est la présentation de la structure du modèle NARMA ainsi que la présentation des méthodes heuristiques proposées pour l'identification de ce type de modèle, sur lequel se base notre commande prédictive.

Chapitre 2

Identification des systèmes non linéaires monovariables

Chapitre 2 Identification des systèmes non linéaires monovariables

Chapitre 2

Identification des systèmes non linéaires monovariables

2.1. Introduction

L'utilisation d'un modèle de type NARMA permet non seulement de réduire le nombre de termes, et par conséquent le nombre de paramètres du modèle mais il permet aussi la modélisation des systèmes avec un ordre relativement faible [Liu, 2003].

Dans ce chapitre, on s'intéressera à l'identification paramétrique et structurelle des modèles de type NARMA des systèmes monovariables. D'abord, on présentera l'algorithme de régression par pas échelonné [Kortmann, 1988]. Ensuite, on proposera deux nouvelles méthodes pour la détermination du modèle NARMA. La première méthode utilise l'algorithme génétique avec sa représentation binaire. Dans ce cas, les individus de la population forment les termes du modèle alors que les paramètres sont estimés avec la méthode des moindres carrées récursifs. Des critères sont utilisés afin d'assurer la convergence de la population d'une génération à une autre vers le vecteur d'observation souhaitable. La seconde approche est basée sur la combinaison du réseau de neurones artificiels à fonction d'activation polynomiale avec l'algorithme génétique sous sa représentation réelle. Dans ce cas, les individus de la population de l'algorithme génétique représentent les coefficients du polynôme de la fonction d'activation. Les paramètres du modèle NARMA sont alors estimés à partir des pondérations des connexions du réseau neuronal après la fin de la phase d'apprentissage.

Le chapitre est organisé comme suit : Le deuxième paragraphe sera consacré pour présenter la structure du modèle NARMA. Le troisième paragraphe est réservé au développement des trois approches utilisées pour l'estimation structurelle et paramétrique du modèle NARMA. Dans le quatrième paragraphe, on présente les résultats de simulation. La conclusion est donnée dans le dernier paragraphe.

2.2. Le modèle NARMA

On considère la classe des systèmes non linéaires monovariables qui peuvent être modélisés par la relation suivante :

$$y(k) = g(X(k)) + e(k) \tag{2-1}$$

avec : $X(k) = \left[y(k-1),...,y(k-n),u(k-1),..u(k-m)\right]$

et *g* une fonction non linéaire supposée inconnue. Les paramètres *m* et *n* représentent, respectivement, l'ordre de la régression sur l'entrée *u(k)* et l'ordre de la régression sur la sortie *y(k)*, *e(k)* est un bruit blanc de moyenne nulle et de variance finie.

Dans ce travail, l'objectif de la modélisation consiste à caractériser le comportement du système par un modèle général de type NARMA. En effet, la sortie du modèle général $\hat{y}(k)$ (équation 1-7) peut s'écrire sous la forme vectorielle suivante [Kortmann, 1988] :

$$\hat{y}(k) = \hat{\Theta}^T \Phi(k) \tag{2-2}$$

avec $\hat{\Theta}^T$ et $\Phi(k)$ désignent, respectivement, le vecteur des paramètres et le vecteur d'observation.

$$\hat{\Theta} = [\bar{y} \quad \theta_1 \quad \theta_2 \quad ... \quad \theta_r]^T \tag{2-3}$$

$$\Phi(k) = [1 \quad v_1 \quad v_2 \quad ... \quad v_r]^T \tag{2-4}$$

où

$$\begin{aligned} &v_1(k) = u(k\text{-}1), ..., v_m(k) = u(k-m), \\ &v_{m+1}(k) = y(k\text{-}1), \ldots, v_{n+m}(k) = y(k\text{-}n), \\ &v_{n+m+1}(k) = u^2(k\text{-}1), \; v_{n+m+2}(k) = u(k\text{-}1)\cdot u(k\text{-}2), \ldots, \\ &v_r(k) = y^q(k\text{-}n) \end{aligned} \tag{2-5}$$

Le paramètre *q* représente le degré de non linéarité du modèle. Les paramètres *m*, *n* et *q* caractérisent le modèle NARMA. Puisque la sortie du modèle est linéaire par rapport aux paramètres, on peut alors utiliser la méthode des Moindres Carrées Récursifs (MCR) pour

estimer ces paramètres [Landau, 1988]. Le nombre de termes intervenant dans l'expression de la sortie du modèle $\hat{y}(k)$ est donné par la relation suivante [Kortmann, 1988] :

$$r = \frac{(n+m+q)!}{q!\,(n+m)!} - 1 \tag{2-6}$$

Par conséquent, le nombre de combinaisons possibles qu'on peut avoir est :

$$w = 2^r - 1 \tag{2-7}$$

Pour *m, n* et *q* donnés, il existe un nombre élevé de combinaisons possibles. Par exemple pour un cas simple (*m*=*n*=*q*=2), on aura 14 termes et par conséquent 16383 combinaisons. Il est donc nécessaire d'utiliser une procédure permettant la sélection d'un modèle parmi l'ensemble des modèles possibles qui représente le comportement du système réel.

Si on désigne par $v_i(k)$ le $i^{ème}$ terme du modèle considéré pour $(n = m = q = 2)$, alors les 14 termes possibles seront comme suit :

$$\begin{aligned}
v_1(k) &= u(k-1) & v_8(k) &= u(k\text{-}1)\,y(k\text{-}1)\\
v_2(k) &= \mathrm{u(k-2)} & v_9(k) &= u(k\text{-}1)\,y(k-2)\\
v_3(k) &= \mathrm{y(k-1)} & v_{10}(k) &= u(k\text{-}2)\,y(k\text{-}1)\\
v_4(k) &= \mathrm{y(k-2)} & v_{11}(k) &= \mathrm{u(k-2)\cdot y(k-2)}\\
v_5(k) &= \mathrm{u^2(k-1)} & v_{12}(k) &= \mathrm{y^2(k-1)}\\
v_6(k) &= \mathrm{u(k-1)\cdot u(k-2)} & v_{13}(k) &= \mathrm{y(k-1)\cdot y(k-2)}\\
v_7(k) &= \mathrm{u^2(k-2)} & v_{14}(k) &= \mathrm{y^2(k-2)}
\end{aligned} \tag{2-8}$$

2.3. Identification du modèle NARMA

La détermination du modèle d'un système revient à déterminer les termes qui forment ce modèle ainsi que les paramètres correspondants. On présente par la suite trois méthodes différentes pour l'identification des modèles de type NARMA. Ces méthodes exploitent hors ligne un fichier formé par les mesures de l'entrée et la sortie du système : {*y(k)*, *u(k)*, *k*=1, …, *N*}, *N* est le nombre total des mesures.

2.3.1. Méthode basée sur l'algorithme de Kortmann

Cette méthode exploite l'algorithme de régression par pas échelonné modifié, proposé initialement par Kortmann et Unbehauen [Kortmann, 1988]. Cet algorithme, basé sur les corrélations des mesures entrée-sortie, permet à la fois de déterminer les termes et d'estimer les paramètres correspondants d'un modèle. Il comporte plusieurs critères d'informations ainsi que des tests statistiques, afin de donner des informations sur la qualité du modèle identifié [Akaike, 1974], [Labrousse, 1977] et [Pukkila, 1988]. Cet algorithme comporte huit étapes, ces étapes sont résumées comme suit :

Etape 1 : Initialisation de l'algorithme

Etape 2 : Calcul des coefficients de corrélation normalisés de tous les termes du vecteur d'observation avec la sortie du système.

Etape 3 : Sélection d'un terme à ajouter à l'expression de la sortie du modèle.

Etape 4 : Estimation des paramètres à l'aide de la méthode de (MCR) ; et calcul des critères de performances (*OVF*, R^2, *FPE*, *AIC*, *LILC*, *BIC*).

Etape 5 : Réalisation d'un test qui permet d'accepter provisoirement le dernier terme sélectionné, de le rejeter ou de quitter la procédure.

Etape 6 : Calcul des critères d'informations partiels (PF_i, PE_i, AIC_i, $LILC_i$, BIC_i).

Etape 7 : Réalisation d'un test qui permet la sélection des termes à garder dans l'expression du modèle actuel et des termes qui seront rejetés.

Etape 8 : Calcul des coefficients de corrélation partiels normalisés et retour à l'étape 3.

L'algorithme de régression par pas échelonné modifié suppose que les paramètres m, n et q du modèle NARMA sont connus. Ceci permet de calculer, dans l'étape 2, les coefficients de corrélation normalisés de tous les termes du vecteur d'observation. Ensuite, l'analyse d'un certain nombre de critères de performances permet de juger la qualité du modèle trouvé et par conséquent, l'arrêt de la procédure de recherche du modèle.

Pour expliquer le fonctionnement de cet algorithme, on présente par la suite les tests statistiques et les critères d'informations utilisés dans les étapes 4 et 6.

2.3.1.1. Tests statistiques et critères d'informations [Akaike, 1974], [Haber, 1990]

A. Tests statistiques

a. Critère de l'erreur de prédiction finale

Ce critère noté FPE (Final Prediction Error), est défini par :

$$FPE = V(\hat{\Theta}) \frac{N + n}{N - n} \tag{2-9}$$

avec N est le nombre d'observations, $\hat{\Theta}$ est le vecteur de paramètres de dimension n et $V(\hat{\Theta})$ est l'estimation de la variance du résidu.

$$V(\hat{\Theta}) = \frac{1}{N}\sum_{k=1}^{N}\varepsilon^2(k) \tag{2-10}$$

avec ε(k) est l'erreur de prédiction ($\varepsilon(k) = y(k) - \hat{y}(k)$). Dans notre cas, $n = r+1$.

b. Le coefficient de corrélation total.

Il est défini par : $$R_{tot}^2 = 1 - \frac{\sum_{k=1}^{N}\varepsilon^2(k)}{\sum_{k=1}^{N} y^2(k)} \tag{2-11}$$

avec N est le nombre de mesures.

La valeur de R_{tot}^2 peut être utilisée pour juger la qualité du modèle obtenu. En effet, lorsque la valeur de R_{tot}^2 est très proche de l'unité, alors le modèle estimé caractérise convenablement le comportement du système considéré.

c. Le coefficient de corrélation multiple.

Ce critère est défini par : $$R^2_{mult} = \frac{\sum_{k=1}^{N} [\hat{y}(k) - \bar{y}]^2}{\sum_{k=1}^{N} [y(k) - \bar{y}]^2} \; 100 \; \% \qquad (2\text{-}12)$$

R^2_{mult} mesure la proportion de la variation totale de la valeur moyenne $\bar{y}$ par rapport à la sortie du modèle.

d. Le coefficient de corrélation ajusté.

Ce critère est défini par : $$R^2_{ajust} = 1 - \left(1 - R^2_{mult}\right) \frac{N-1}{N-v} \qquad (2\text{-}13)$$

v : désigne le nombre de termes du modèle. R^2_{ajust}, prend en considération le critère R^2_{mult} tout en pénalisant les modèles qui possèdent un nombre élevé de termes.

B. Critères d'information [Landau, 1988], [Sheng, 2003].

La plupart des critères d'information utilisés pour déterminer le nombre de paramètres d'un modèle sont exprimés par :

$$\Gamma(v) = N \, Ln(\sigma^2) + v \, g(N) \qquad (2\text{-}14)$$

avec :

- σ^2 : l'estimé du maximum de vraisemblance ou son approximation par la variance du résidu qui est obtenu en utilisant l'estimateur des moindres carrés récursifs.

σ^2 est défini par :

$$\sigma^2 = \frac{\sum_{k=1}^{N} \varepsilon^2(k)}{N} = \frac{\sum_{k=1}^{N} (y(k) - \hat{y}(k))^2}{N} \qquad (2\text{-}15)$$

- N est le nombre d'observations

- v est le nombre de paramètres estimés.

- $g(N)$: est un terme de pénalité.

$g\ (N)$ augmente quand le nombre de paramètres dans le modèle augmente, par contre $N\,Ln\,(\sigma^2)$ a tendance à décroître quand le nombre de paramètres croît.

Le modèle optimal est choisi pour la valeur de v minimisant le critère $\Gamma\,(v)$.

a. Le critère d'information d'Akaike (AIC) [Akaike, 1974]

Le critère d'information d'Akaike (Akaike Information Criterion) *AIC* est défini en prenant $g\ (N) = 2$ dans la relation (2-14) :

$$AIC\ (v) = N\,Ln\,(\sigma^2) + 2\,v \tag{2-16}$$

***b. Le critère d'information de Bayes (BIC)* :**

Le critère d'information de Bayes (Bayes's Information Criterion) *BIC* est défini lorsque $g(N) = Ln\,(N)$ dans la relation (2-14) :

$$BIC\ (v) = N\ Ln\,(\sigma^2) + v\ Ln(N) \tag{2-17}$$

c. La Loi du Critère Logarithmique Itératif (test LILC)

La Loi du Critère Logarithmique Itératif (Khinchin's Law of Iterated Logarithm Criterion) LILC est définie en fixant $g(N) = cLn[Ln(N)]$ dans la relation (2-14) :

$$LILC\ (v) = N\ Ln\,(\sigma^2) + v\,c\ Ln[\ Ln(N)] \tag{2-18}$$

où c est une constante positive $(c \geq 2)$.

2.3.1.2. L'algorithme de régression par pas échelonné

On présente par la suite les différentes étapes de l'algorithme de régression par pas échelonné modifié. L'algorithme, décrit par l'organigramme de la figure 2-1, comporte huit étapes [Kortmann, 1988].

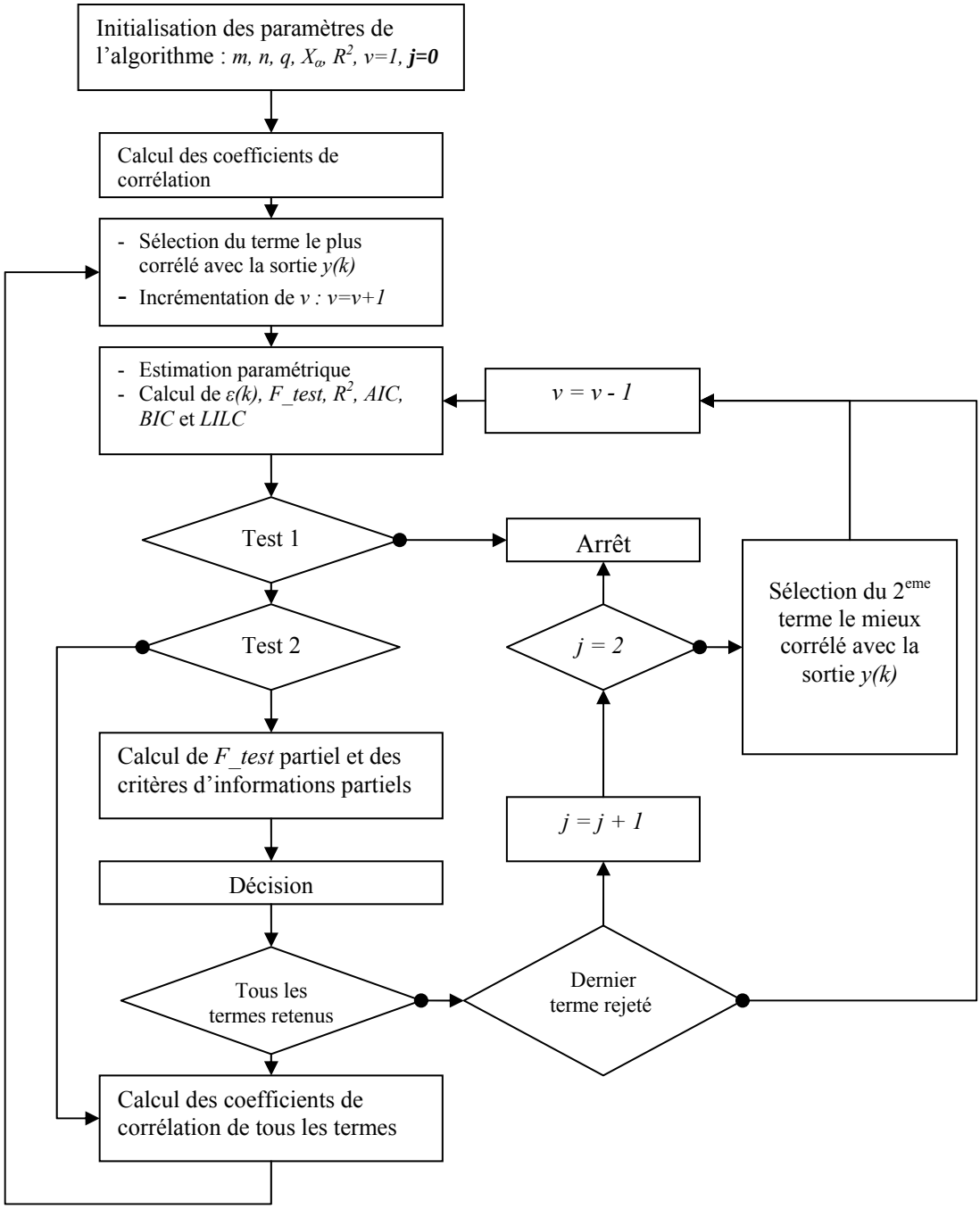


Figure 2-1 : Organigramme de l'algorithme de régression par pas échelonné modifié

<u>Etape 1</u> (Initialisation):

Au cours de cette étapes, on spécifie les valeurs maximales de l'ordre de la régression sur l'entrée (m) et l'ordre de la régression sur la sortie (n) ainsi que le degré de non linéarité (q). Ensuite, on calcule le nombre de termes possibles r donné par l'équation (2-6) et on forme le

vecteur d'observation $\Phi(k)$. On initialise également le nombre de paramètres : $v = 1$ (valeur moyenne $\bar{y}$) et on spécifie le seuil de signification α et on donne la valeur X_α correspondante (en utilisant les abaques de Fisher donnés dans l'annexe1).

Etape 2

On calcule les coefficients de corrélation normalisés de tous les termes du vecteur d'observation avec la sortie $y(k)$

$$\rho_i = \frac{N \sum_{k=1}^{N} v_i(k) \, . \, y(k) \; - \; \sum_{k=1}^{N} v_i(k) \; \sum_{k=1}^{N} y(k)}{\sqrt{\left[\left[N \sum_{k=1}^{N} v_i{}^2(k) \; - \left[\sum_{k=1}^{N} v_i(k) \right]^2 \right] \left[N \sum_{k=1}^{N} y^2(k) \; - \left[\sum_{k=1}^{N} y(k) \right]^2 \right] \right]}} \qquad i = [1,...,r\,] \quad (2\text{-}19)$$

avec *N* le nombre d'observation et $v_i\ (k)$ les termes du modèle général donnés par l'équation (2-5).

Etape 3

Au niveau de cette étape, on détermine le plus grand coefficient de corrélation (ρ_{opt}) permettant de sélectionner le terme optimal à ajouter au modèle.

$$\rho_{opt} \; = \; \max(\rho_i), \quad v_{opt} \; = v_i$$

Ensuite, on incrémente le nombre des paramètres de 1, ($v = v + 1$).

Exemple :

On considère un modèle NARMA avec $n = m = q = 2$, on a 14 termes possibles $\{v_1(\mathrm{k}),...,v_{14}(\mathrm{k})\}$ donnés par la relation (2-8).

Si ρ_{12} est le coefficient de corrélation maximal. Le terme sélectionné par l'algorithme est

$$v_{opt}(k) \; = \; v_{12}(k) \; = \; y^2(k\text{-}1) \qquad (2\text{-}20)$$

Le modèle actuel est alors :

$$\hat{y}(k) = \; \hat{\Theta}^T \; \Phi(k) \qquad (2\text{-}21)$$

avec : $\Phi(k) = [\ 1\ y^2(k\text{-}1)\ ]^T$ (2-22)

et $\Theta^T = [\ \bar{y}\ \ \theta_{12}\ ]$ (2-23)

Etape 4

Estimer les paramètres (valeur moyenne incluse) en utilisant l'algorithme des moindres carrés récursifs et calculer les résidus :

$$\varepsilon(k) = y(k) - \hat{y}(k)\ ; \qquad k = 1, \ldots, N \tag{2-24}$$

Déterminer le *F_Test* global (Overal F_test) *OVF* [Bariani, 1988] :

$$OVF = \frac{\left[\sum_{k=1}^{N} \left[\hat{y}(k) - \bar{y}\right]^2\right] \Big/ (v - 1)}{\left[\sum_{k=1}^{N} \left[\hat{y}(k) - y(k)\right]^2\right] \Big/ (N - v)} \tag{2-25}$$

Calculer le coefficient de corrélation multiple.

Calculer les critères d'informations définis comme suit :

a) L'erreur de prédiction finale (Final Prediction Error) *FPE*

$$FPE(v) = N\,Ln\left[\frac{1}{N}\sum_{k=1}^{N}\varepsilon^2(k)\right] + N\,Ln\left[\frac{N + v}{N - v}\right] \tag{2-26}$$

b) Le critère d'information d'Akaike *AIC*.

c) Le critère d'information de Bayes *BIC*.

d) La loi du critère logarithmique itératif *LILC* (en prenant *c=2* dans l'équation (2-18))

$$LILC(v) = N\,Ln\left[\frac{1}{N}\sum_{k=1}^{N}\varepsilon^2(k)\right] + 2\,v\,Ln\left[Ln(N)\right] \tag{2-27}$$

Etape 5

Cette étape contient deux tests [Boujmil, 1991] :

Test 1 :

Si la valeur du coefficient de corrélation R^2_{mult} est supérieure ou égale à une valeur fixée a priori, on arrête la procédure.

Test 2 :

Tester si le *F_Test* est significatif en comparant sa valeur à une valeur fixée a priori ($X\alpha$).

Comparer R^2_{mult}, *FPE(v)*, *AIC(v)*, *BIC(v)*, *LILC(v)* à leurs valeurs précédentes (valeurs à *v–1*) :

Si $R^2_{mult}(v) < R^2_{mult}(v-1)$ la dernière variable sélectionnée est rejetée.

ou si *FPE(v)* (ou respectivement *AIC(v)*, *BIC(v)*, *LILC(v)*) est supérieur à *FPE(v-1)* (respectivement *AIC(v-1)*, *BIC(v-1)*, *LILC(v-1)*), la dernière variable sélectionnée est rejetée.

La procédure passe à l'étape 8 si la valeur du *F_Test* est non significative ou si la dernière variable sélectionnée est rejetée.

Remarque : Dans la version originale de Kortmann [Kortmann, 1988], la procédure s'arrête si le *F_test* est non significatif ou si les critères d'informations sont supérieurs à ceux du modèle précédent. Dans [Boujmil, 1991], l'auteur a proposé d'effectuer deux tests assurant plus de chance à l'algorithme pour trouver un meilleur modèle.

Etape 6 :

Calculer pour chaque terme du modèle actuel (sauf la valeur moyenne) le *F_Test* partiel :

$$PF_i = \frac{\left[\sum_{k=1}^{N}[\hat{y}(k)-\bar{y}]^2\right]_{v \text{ variables}} - \left[\sum_{k=1}^{N}[\hat{y}(k)-\bar{y}]^2\right]_{v-1 \text{ variables}}}{\left(\left[\sum_{k=1}^{N}[\hat{y}(k) - y(k)]^2\right]_{v \text{ variables}}\right) \Big/ (N-v)}, (i = 1, \ldots, v-1) \quad v>2 \qquad (2\text{-}28)$$

et les critères d'informations partielles : AIC_i, BIC_i, FPE_i et $LILC_i$, $i = 1, \ldots, v-1$, $v > 2$, en utilisant les équations (2-16), (2-17), (2-26) et (2-27).

Dans l'équation (2-28), la première somme du numérateur ainsi que la somme des résidus du dénominateur sont calculée à l'étape 4. La deuxième somme du numérateur de l'équation (2-28) et les critères d'informations partielles se calculent comme suit :

Rejeter la première variable du modèle, estimer les paramètres restant du modèle et calculer :

$$\sum_{k=1}^{N} [\hat{y}(k) - \bar{y}]^2 \Big|_{v-1 \text{ variables}} \tag{2-29}$$

et la somme des résidus :

$$\sum_{k=1}^{N} \varepsilon(k)^2 \Big|_{v-1 \text{ variables}} \tag{2-30}$$

avec les valeurs estimées, calculer le PF_i et les critères d'informations partielles.

Introduire le premier paramètre et rejeter le second. Pour ce nouveau modèle estimer les paramètres et calculer le PF_i et les critères d'information partiels. Refaire cette opération jusqu'au dernier paramètre du modèle.

Ces valeurs caractérisent la mesure de la contribution du terme considéré dans le modèle encore inconnu.

Exemple

On considère l'exemple de l'étape 3. Soit $y^2(k\text{-}1)$ est le premier terme sélectionné. Si $u(k\text{-}2)$ est le dernier terme sélectionné, le modèle actuel sera alors :

$$\hat{y}(k) = \theta_{12}\ y^2(k\text{-}1) + \theta_2\ u(k\text{-}2) + \varepsilon(k). \tag{2-31}$$

Pour calculer le PF_1 et les critères d'informations partiels, on procède comme suit : on rejette le terme $y^2(k\text{-}1)$ d'où le nouveau modèle :

$$\hat{y}(k) = \theta_2^{'}\ u(k\text{-}2) + \varepsilon_1(k) \tag{2-32}$$

- on estime $\theta_2^{'}$ et on calcule $\hat{y}_1(k) = \theta_2^{'}$ u(k - 2) et ε_1(k). puis on calcule le *F_Test* partiel PF_1 donné par l'équation (2-28) et les différents critères d'informations partiels AIC_1, BIC_1, FPE_1 et $LILC_1$, donnés par les équations (2-16), (2-17), (2-26) et (2-27),

- on introduit le terme $y^2(k\text{-}1)$ et on rejette $u(k\text{-}2)$, ainsi le nouveau modèle sera :

$$\hat{y}(k) = \theta_{12}^{'} \, y^2(k\text{-}1) + \varepsilon_2(k). \tag{2-33}$$

- On estime le paramètre $\theta_{12}^{'}$ et on calcule $\hat{y}_2(\mathrm{k}) = \theta_{12}^{'} \, y^2(k\text{-}1)$ et $\varepsilon_2(\mathrm{k})$ puis on calcule le *F_Test* partiel correspondant PF_2 et les différents critères d'informations partiels correspondants AIC_2, BIC_2, FPE_2 et $LILC_2$.

Etape 7 :

Comparer la plus petite valeur de AIC_i (respectivement BIC_i, FPE_i et $LILC_i$) à la valeur de AIC_v (respectivement, BIC_v, FPE_v et $LILC_v$) calculée à l'étape 4.

$$\text{si} \quad \min_i \left(AIC_i\right) < AIC_v \quad \text{ou} \quad \min_i \left(BIC_i\right) < BIC_v$$

$$\text{ou} \quad \min_i \left(FPE_i\right) < FPE_v \quad \text{ou} \quad \min_i \left(LILC_i\right) < LILC_v$$

alors la variable correspondante est rejetée.

- De plus, tester si la valeur du *F_Test* partiel est significative ou non en comparant sa valeur à la valeur $X\alpha$ fixée a priori (valeur de la distribution de Fisher donnée à l'étape 1).

Si $PFi < X\alpha$, la variable correspondante est rejetée. La variable est retenue dans le cas contraire.

Si toutes les variables sont retenues, la procédure passe à l'étape 8. Dans le cas où une variable est rejetée, on décrémente le nombre des paramètres v ($v=v\text{-}1$) et on passe à l'étape 4 pour refaire l'estimation paramétrique et l'évaluation du modèle. Si la dernière variable est rejetée, on choisit alors le deuxième terme de plus grand coefficient de corrélation partiel et on passe à l'étape 4. Par ailleurs, si la dernière variable sélectionnée est rejetée deux fois, la procédure s'arrête.

Etape 8 :

Calculer les coefficients de corrélation partiels normalisés de tous les termes restants :

$$\rho i = \frac{N\sum_{k=1}^{N}\varepsilon_i'(k)\,\varepsilon(k) \;-\; \sum_{k=1}^{N}\varepsilon_i'(k)\;\sum_{k=1}^{N}\varepsilon(k)}{\sqrt{\left[N\sum_{k=1}^{N}(\varepsilon_i'(k))^2 \;-\; [\sum_{k=1}^{N}\varepsilon_i'(k)\,]^2\right]\left[N\sum_{k=1}^{N}\varepsilon^2(k) \;-\; [\sum_{k=1}^{N}\varepsilon(k)\,]^2\right]}} \qquad (2\text{-}34)$$

où $\varepsilon(k),\ \ k = 1, \ldots, N$ est calculé déjà à l'étape 4 et $\varepsilon'(k)$ est obtenu comme le montre l'exemple suivant :

Exemple

On reprend le modèle présenté dans l'exemple de l'étape 6. Le modèle actuel est donné par :

$$\hat{y}(k) \;=\; \theta_{12}\; y^2(k\text{-}1) \;+\; \theta_2\; u(k\text{-}2) + \varepsilon(k)\,. \qquad (2\text{-}35)$$

Les termes restant dont on veut calculer leurs coefficients de corrélation partiels sont :

$$\begin{aligned}
v_1(k) &= \mathrm{u(k-1)} & v_8(k) &= \mathrm{u(k-1)\cdot y(k-1)}\\
v_3(k) &= \mathrm{y(k-1)} & v_9(k) &= \mathrm{u(k-1)\cdot y(k-2)}\\
v_4(k) &= \mathrm{y(k-2)} & v_{10}(k) &= \mathrm{u(k-2)\ y(k-1)}\\
v_5(k) &= \mathrm{u^2(k-1)} & v_{11}(k) &= \mathrm{u(k-2)\ y(k-2)}\\
v_6(k) &= \mathrm{u(k-1)\cdot u(k-2)} & v_{13}(k) &= \mathrm{y(k-1)\ y(k-2)}\\
v_7(k) &= \mathrm{u^2(k-2)} & v_{14}(k) &= \mathrm{y^2(k-2)}
\end{aligned} \qquad (2\text{-}36)$$

Soit $v_1(k)$ le premier terme que l'on veut calculer son coefficient de corrélation. On considère alors le nouveau modèle ayant pour sortie $v_1(k)$ et pour entrée le modèle actuel :

$$v_1(k) = u(k-1) \;=\; \theta'_{12}\; \mathrm{y^2(k-1)} \;+\theta'_2\; \mathrm{u(k-2)} + \varepsilon'_1\,(\mathrm{k}) \qquad (2\text{-}37)$$

On estime les paramètres et on calcule $\varepsilon'_1\,(\mathrm{k})$ par simulation d'où :

$$\varepsilon_1'(\mathrm{k}) = \mathrm{u(k-1)} - \left[\theta_{12}'\cdot\ \mathrm{y^2(k-1)} \;+\theta_2'\cdot \mathrm{u(k-2)}\right] \qquad (2\text{-}38)$$

On répète le même travail pour les autres termes : $v_i\,(\mathrm{k}),\quad i = 3, \cdots, 14 \quad$ et $\quad i \neq 12.$

Commentaires :

La stratégie de cet algorithme consiste à construire le modèle en commençant par le terme le mieux corrélé avec la sortie. Puis, on ajoute par la suite des nouveaux termes jusqu'à l'obtention du modèle final.

Cet algorithme utilise plusieurs tests statistiques, des critères de performances (*OVF,* R^2_{mult} *, FPE, AIC, BIC, LILC,* …), des critères d'information partiels (FPE_i, AIC_i, BIC_i, $LILC_i$) ainsi que des coefficients de corrélation et des coefficients de corrélation partiels normalisés. Bien que l'adoption de tous ces critères et de ces tests statistiques offre une robustesse à cet algorithme, la difficulté lors de leurs programmations reste contraignante. L'implémentation de l'algorithme nécessite beaucoup de précaution pour ne pas faire des confusions ou des erreurs, surtout dans les étapes 6 et 8 lors du calcul des critères d'information partiels et des coefficients de corrélation partiels. De même cette difficulté se présente dans l'étape 7 pour décider quels sont les termes à rejeter de la structure actuelle. C'est pour ces raisons qu'on a choisi des exemples simples, expliquant les différentes étapes de cet algorithme.

La complexité de cette méthode et son limite aux cas des systèmes monovariables nous ont incités à concevoir d'autres méthodes qui utilisent d'autres approches, moins complexes, et qui peuvent êtres extensibles pour traiter les systèmes non linéaires multivariables.

Nous proposons dans ce qui suit deux nouvelles approches pour l'identification des modèles de type NARMA.

2.3.2. Méthode basée sur les Algorithmes Génétiques Binaires

2.3.2.1. Présentation de l'algorithme génétique

L'algorithme génétique est appliqué à de nombreux problèmes d'optimisation [Goldberg, 1989]. Il a été mis au point dans les années 70 par l'équipe de John Holland à l'Université du Michigan. Après être demeuré mal connu pendant des années, l'intérêt pour cet algorithme s'est accru récemment. L'avantage de cette méthode d'optimisation est qu'elle trouve une solution qui s'approche de la solution optimale, et ce même pour des problèmes fortement non linéaires [Goldberg, 1989]. Cet algorithme donne de bonnes performances même lorsque le nombre de variables est très élevé. De nombreuses variantes quant aux opérateurs et à l'encodage utilisés existent dans la littérature [Goldberg, 1989], [Man, 1996]. L'algorithme

génétique peut fonctionner également, soit avec une représentation réelle soit avec une représentation binaire des variables [Goldberg, 1989].

Les algorithmes génétiques sont inspirés du phénomène de la sélection naturelle qui permet à la nature de créer avec succès des organismes adaptés à leur environnement [Goldberg, 1989]. Les organismes qui sont le moins adaptés disparaissent peu à peu tandis que ceux qui survivent et qui sont mieux adaptés peuvent se reproduire. Les descendants sont semblables à leurs parents, mais pas identiques. Ils retiennent une partie de leurs parents. Lorsque l'environnement change, les espèces s'adaptent graduellement. Les mécanismes qui gèrent la recherche d'un optimum dans un algorithme génétique sont déduits de ce phénomène.

Des opérateurs génétiques servent à faire évoluer artificiellement un ensemble de vecteurs. Par analogie, l'ensemble des vecteurs forme une population et chaque vecteur constitue un individu. Dans les algorithmes génétiques simples, trois opérateurs principaux sont utilisés : la reproduction, la mutation et le croisement. Les opérateurs modifient de façon successive la population, ce qui permet d'explorer l'espace de recherche afin de trouver un optimum. Une fonction d'évaluation sert à évaluer l'adaptation des individus. Cette fonction évalue l'adéquation de l'individu à la solution recherchée. Plus la valeur de la fonction évaluée est élevée, plus cet individu est adapté. L'application des trois opérateurs sur une population d'individus représente une itération ou une génération. On présente ci-dessous les principaux opérateurs de traitement de l'algorithme génétique.

• Reproduction : La reproduction est une opération selon laquelle les individus d'une population sont copiés dans la population de la génération suivante selon la fonction d'évaluation. Plus la valeur obtenue par la fonction d'évaluation est grande, plus la probabilité d'être copiée dans la prochaine population est élevée. La sélection étant aléatoire, les meilleurs ont tendance à être copiés plusieurs fois dans la prochaine population proportionnellement à l'importance de la valeur de la fonction d'évaluation des parents dans la population. Ceux qui sont moins adaptés sont probablement abandonnés et non présents dans la population suivante. Cette phase a pour but d'accroître l'importance des bonnes solutions. Elle tend à augmenter l'uniformité de la population.

• Croisement : C'est un opérateur qui permet de mélanger les caractéristiques de deux individus dans la recherche de nouvelles solutions. Le principe consiste à utiliser l'information contenue dans deux solutions afin d'en produire deux nouvelles. Dans le cas du

croisement uniforme les couples sont créés en sélectionnant deux individus de façon aléatoire. Ensuite, les caractéristiques des individus sont mélangées de façon aléatoire.

• Mutation : La mutation consiste à modifier aléatoirement la valeur d'un individu. La probabilité que cette modification ait lieu est en réalité très faible. Cet opérateur permet de changer la direction de l'exploration et ainsi de ne pas garder un optimum local.

• Fonction d'évaluation : Cette fonction évalue l'adéquation de l'individu à la solution recherchée. Dans un problème de maximisation, plus la valeur de la fonction est élevée, plus cet individu est adapté.

L'initialisation des individus de la population initiale se fait de façon aléatoire. L'algorithme s'arrête lorsqu'il y a convergence ou après un nombre d'itérations (ou générations) prédéfini. Les paramètres à ajuster pour l'algorithme génétique sont :

- Le nombre maximal de générations : qui spécifie le nombre d'itérations.
- Le nombre d'individus formant une population : il représente en quelque sorte la mémoire de l'algorithme, plus ce nombre est grand, plus l'algorithme a tendance à garder de bonnes parties de solutions.
- La probabilité de croisement : elle spécifie le taux avec lequel les deux parents (individus) sont mélangés lors du croisement.
- La probabilité de mutation : cette probabilité contrôle la variation aléatoire des individus et dicte alors l'exploration de l'espace de recherche.
- Le nombre d'évaluations requises (*Ne*) pour une recherche par algorithme génétique correspond au produit du nombre de générations effectué (*Ng*) avec le nombre d'individus (*Ni*).

$$Ne = Ng\ Ni \tag{2-39}$$

Dans le paragraphe suivant, on présente l'identification des paramètres du modèle NARMA en utilisant un algorithme génétique binaire.

2.3.2.2. Méthode basée sur les algorithmes génétiques binaires

Dans un algorithme génétique binaire simple, chaque individu est représenté par un vecteur binaire qui désigne la variable de la fonction à optimiser. Ce vecteur binaire sera codé

pour donner une valeur réelle dans l'intervalle espace de solutions [Goldberg, 1989]. La souplesse que présente ce type de codage permet de résoudre plusieurs types de problème [Fnaiech, 2000]. Cet algorithme a été utilisé depuis le début de la dernière décennie, pour l'identification des systèmes de type entrée-sortie [Li, 1993], [Li, 1994], [Luh, 1998], [yang, 2000] et [Madár, 2005]. Les méthodes présentées exploitent différents types de codages. Dans [Li, 1993], les auteurs ont proposé un codage binaire où chaque bit représente un terme du modèle à déterminer. Avec cette méthode, le nombre de termes dans un individu peut dépasser le nombre maximal de termes possibles, ce qui engendre l'annulation de la solution trouvée et la création d'une nouvelle génération. On note aussi que la condition d'arrêt dépend de l'écart entre la valeur du meilleur *fitness* actuel et celle de l'ancien meilleur *fitness*. Si cet écart est inférieur à une certaine précision, la procédure sera arrêtée et le modèle sera donné après avoir éliminé les termes dont les coefficients sont proches de zéro. L'application de cette condition d'arrêt et la possibilité d'avoir des solutions non adaptées au cours du fonctionnement de l'algorithme, engendrent un temps très important lors de l'identification.

Le codage proposé dans [Luh, 1998], consiste à diminuer le nombre de termes possibles dans le modèle en divisant chaque individu en plusieurs sous vecteurs binaires et en conservant les termes significatifs inchangés. Cette approche est basée sur les résultats de [Leontaritis, 1985b] qui montrent que 10 termes sont généralement suffisants pour la modélisation d'un système, avec une légère détérioration du modèle. Puis, dans [yang, 2000] les auteurs ont utilisé une nouvelle technique de codage dite hiérarchique. Ce codage permet de classer les chromosomes sous formes de couches. Dans chaque couche on trouve les termes qui représentent le même degré de non linéarité, ainsi que leur nombre. La couche d'ordre 0 contient la valeur constante (valeur moyenne), la couche d'ordre 1 représente les termes simples (les régressions linéaires sur l'entrée et sur la sortie). Ensuite, on trouve les différentes combinaisons (les doublés puis les triplés, etc…) jusqu'à l'atteinte de l'ordre du polynôme.

La méthode présentée par [Madár, 2005], consiste à mettre les individus de chaque population de l'algorithme génétique sous forme arborescente. Les branches de chaque individu présentent les termes possibles du modèle à définir. Ainsi, l'algorithme OLS (Orthogonal Least Squares) consiste à donner la contribution de chaque branche et donc permet d'éliminer les termes qui ne sont pas significatifs dans le modèle.

On note que dans les méthodes [Li, 1993] et [yang, 2000], la valeur moyenne est considérée dans le codage binaire, d'où la possibilité d'absence de ce terme de la structure du

modèle. Dès lors, si on veut garder ce terme constant dans l'équation du modèle, le fonctionnement de l'algorithme génétique sera forcé à chaque itération. Cependant, dans le travail de [Madár, 2005], ce terme constant ne figure pas dans la structure arborescente.

Notre méthode, contrairement aux autres méthodes, permet de considérer le terme constant dans l'évaluation du *fitness*, au cours de l'estimation des paramètres, sans l'avoir intégré dans le codage de l'algorithme génétique. En plus, cette approche permet d'avoir plusieurs structures différentes présentant des performances proches. Cet avantage permet le choix d'une structure optimale en faisant un compromis entre la complexité du modèle et sa précision.

Dans ce travail, le vecteur binaire est l'image du vecteur d'observation $\Phi(k)$. Une ligne de la population de l'algorithme génétique, notée *Ipop*, ainsi que le vecteur d'observation sont donnés par :

$$Ipop = [1 \quad 0 \quad 1 \quad ... \quad 1] \tag{2-40}$$

$$\Phi(k) = \begin{bmatrix} 1 & v_1 & v_2 & ... & v_r \end{bmatrix} \tag{2-41}$$

L'analyse de *Ipop* permet de sélectionner les termes intervenants dans l'expression de la sortie du modèle. Seuls les termes dont la valeur du bit correspondant est à ''1'' logique sont sélectionnés, les autres ne seront pas considérés dans l'expression de la sortie du modèle. Chaque individu comporte autant de valeurs booléennes que de termes possibles du modèle NARMA. L'exemple suivant explique le codage du vecteur binaire. Dans le cas où (*m=n=q=2*), on a 14 termes possibles. On considère alors une représentation binaire de l'algorithme génétique où chaque individu comporte 14 bits, chaque bit représente un terme du vecteur d'observation. Un individu peut être présenté par le vecteur suivant :

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2-42}$$

Dans ce vecteur, les bits 1, 2, 4, 5 et 12 sont à ''1'', les autres bits sont à ''0''. Ceci correspond au modèle défini par la structure suivante :

$$\hat{y}(k) = \bar{y} + b_1\, u(k-1) + b_2\, u(k-2) + a_2\, y(k-2) + b_{11}\, u^2(k-1) + a_{11}\, y^2(k-1) \tag{2-43}$$

Ensuite, la procédure de MCR est utilisée pour estimer les paramètres du modèle considéré (y compris la valeur moyenne). Pour chaque individu, on calcule la fonction fitness afin de l'utiliser dans la phase de sélection. La fonction d'évaluation (*fitness*) dépend de

l'erreur entre la sortie du modèle obtenu et la sortie du système. Cette fonction est définie par la relation suivante :

$$fitness = \frac{1}{1+J_1} \quad \text{avec} \quad J_1 = \sum_{k=1}^{N} \left(y(k) - \hat{y}(k)\right)^2 \tag{2-44}$$

Avec cet algorithme, on utilise les opérateurs génétiques traditionnels (sélection, croisement et mutation) comme le montre la figure 2-2. Les caractéristiques de l'algorithme génétique considéré sont les suivantes :

- La population initiale : la population initiale est choisie d'une façon aléatoire. La taille de chaque individu est égale à la taille du vecteur d'observation (sans considérer la valeur moyenne).
- La sélection : la sélection consiste à choisir les individus qui sont caractérisés par les meilleurs fitness. La probabilité de sélection des individus est proportionnelle à leurs fitness et elle est donnée par :

$$P_i = \frac{fitness_i}{\sum_{j=1}^{\max\_pop} fitness_j} \tag{2-45}$$

avec $fitness_i$ est le fitness de l'individu i et max_pop est le nombre total d'individus. Dans ce travail, on a utilisé la sélection basée sur la roue de loterie [Goldberg, 1989].

- Le croisement : l'objectif du croisement est l'échange d'information entre deux parents pour former deux nouveaux individus. Dans le présent travail, nous avons opté pour le croisement à un cite avec une probabilité Pc.
- La mutation : la mutation est un processus aléatoire qui consiste à modifier l'état d'un bit de 0 à 1 ou vice versa. La probabilité de mutation Pm est généralement choisie de valeur très faible.
- Condition d'arrêt : l'algorithme génétique évolue d'une génération à une autre jusqu'à une condition d'arrêt est atteinte. Dans la littérature, il existe plusieurs critères qui peuvent être utilisés comme condition d'arrêt de l'algorithme génétique. On cite par exemple l'utilisation d'un nombre maximal de génération, le test d'uniformité des individus de la population après un certain nombre de génération

ou bien la non amélioration du *fitness*. Dans notre travail, on a considéré le nombre maximal de génération de l'algorithme génétique comme condition d'arrêt.

La solution obtenue, avec cette approche, détermine le modèle identifié du système. Le meilleur modèle est celui qui donne l'erreur de modélisation la plus faible. Cependant, avec cette approche, on peut avoir plusieurs solutions donc plusieurs modèles puisque l'algorithme génétique est une méthode non déterministe.

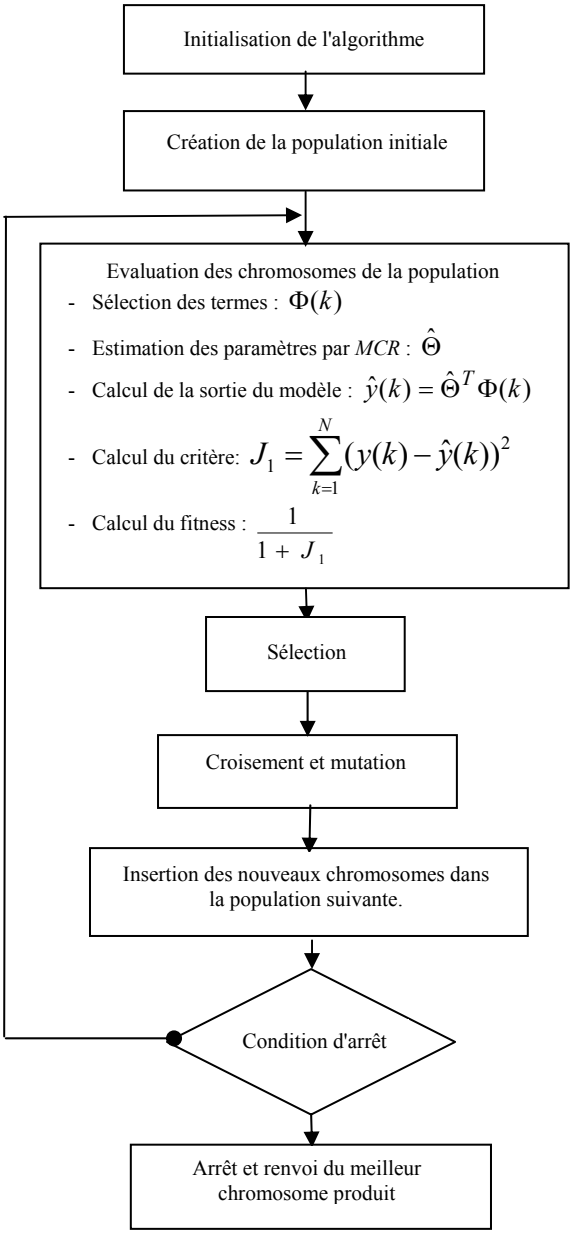


Figure 2-2 : Organigramme de l'identification du modèle NARMA avec l'algorithme génétique binaire

2.3.3. Méthode basée sur les Réseaux de Neurones Artificiels

2.3.3.1. Réseau de neurones artificiels à rétro-propagation de l'erreur

Les réseaux de neurones sont des fonctions mathématiques complexes et non linéaires. Ces réseaux sont composés de couches, qui sont elles mêmes composées d'éléments appelés neurones. Chaque neurone a pour entrée les sorties de chacun des neurones de la couche précédente. Ces entrées sont pondérées par les poids, pour ensuite être combinées et traitées par une fonction d'activation. L'objectif de l'apprentissage est d'obtenir ces poids. L'ajustement des poids se fait de façon à réduire l'erreur quadratique entre les valeurs obtenues en sortie et les valeurs désirées. Le gradient de l'erreur est utilisé à cette fin. Le processus se poursuit jusqu'à la convergence, c'est-à-dire tant que l'erreur totale n'atteint pas une valeur inférieure à un seuil.

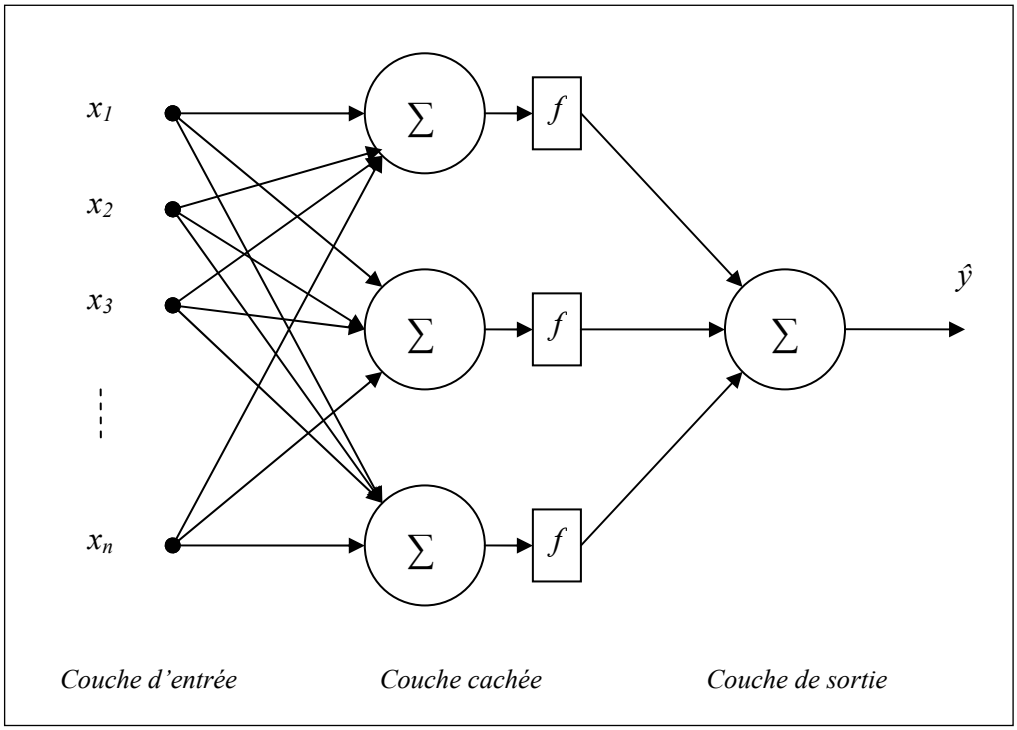


Figure 2-3 : Schéma descriptif d'un Réseau de neurone non bouclé à une seule couche cachée

f étant la fonction d'activation des neurones. Il existe plusieurs fonctions d'activation utilisées en pratique tel que la fonction tangente hyperbolique (*tanh*) et les fonctions (*signe*, *sigmoïde*, *polynomiale*, …). La fonction polynomiale est utilisée dans notre cas.

2.3.3.2. Méthode d'identification utilisant les réseaux de neurones artificiels et l'algorithme génétique

Les réseaux de neurones à trois couches illustrés à la figure 2-3, sont utilisés dans ce travail. Soit le vecteur d'entrée :

$$X(k) = [\, y(k\text{-}1) \cdots y(k\text{-}n) \;\; u(k-1) \;\; \cdots \;\; u(k-m) \,] \tag{2-46}$$

m et *n* désignes respectivement l'ordre de la régression sur l'entrée et l'ordre de la régression sur la sortie. Soient W^1 et W^2 respectivement la matrice des pondérations entre la couche d'entrée et la couche cachée et le vecteur des pondérations entre la couche cachée et la couche de sortie.

Les paramètres du modèle NARMA peuvent être estimés en utilisant un réseau de neurones à fonction d'activation polynomiale [Ki, 1997]. La structure du réseau de neurones est de type «feed-forward» avec une seule couche cachée. La topologie de ce réseau est représentée dans la figure 2-4. Le réseau neuronal reçoit en entrée les éléments du vecteur $X(k)$ et délivre à sa sortie une estimation de la sortie actuelle.

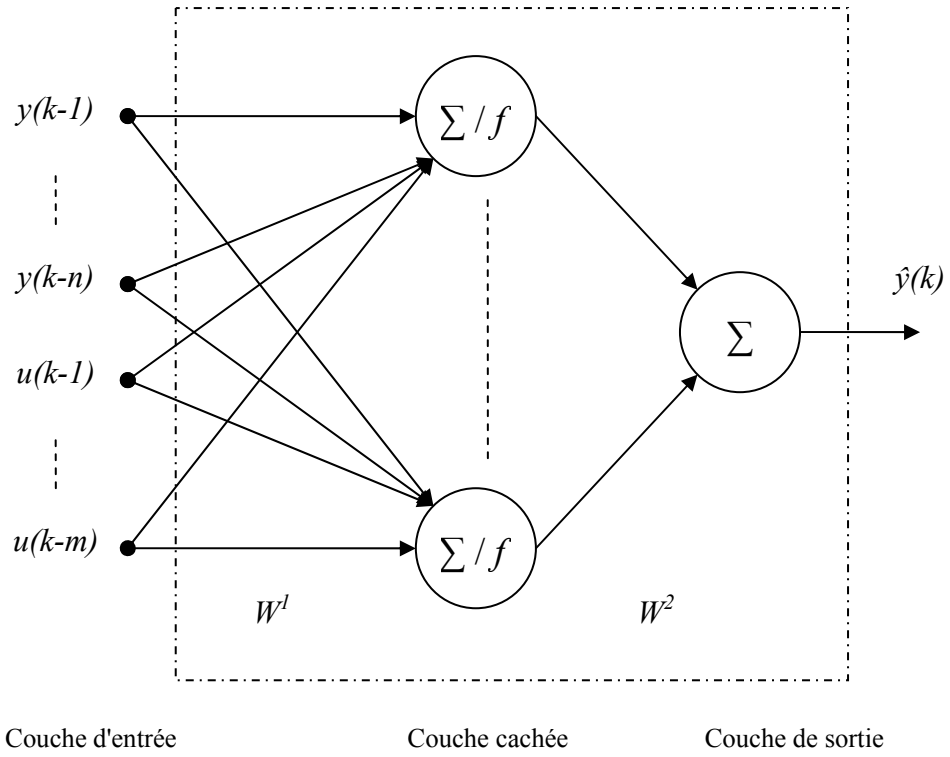


Figure 2-4 : Réseau de neurones à une seule couche cachée

Les sorties de la première couche sont données par :

$$S(k) = f(W^1 X^T(k)) = \left[s_1(k) \ldots s_M(k)\right]^T , \tag{2-47}$$

f étant une fonction d'activation polynomiale. La relation suivante présente une fonction d'activation polynomiale d'ordre q :

$$f(x_i) = f_0 + f_1 x_i + f_2 x_i^2 + \ldots + f_q x_i^q \tag{2-48}$$

En tenant compte des poids des connexions entre la couche d'entrée et la couche cachée, la quantité x_i sera exprimée par :

$$x_i = \sum_{j=1}^{m} w_{ji}^1 u(k-j) + \sum_{j=m+1}^{m+n} w_{ji}^1 y(k-j+m),\ i \in \left[1, M\right] \tag{2-49}$$

avec M le nombre de neurones de la couche cachée. Le choix de M dépend de la complexité du système. En effet, un nombre faible de neurones cachés peut conduire à un modèle non précis alors que le choix d'un nombre élevé de neurones cachés peut rendre le modèle très compliqué et le temps de calcul sera plus important. En général, le nombre de neurones cachés est choisi de façon à réaliser un compromis entre la précision et la complexité du modèle estimé.

L'équation de la sortie du réseau sera alors:

$$\hat{y}(k) = \sum_{i=1}^{M} w_i^2 f(x_i) = W^2 S(k) \tag{2-50}$$

Les dimensions de W^1 et W^2 sont fonction de m, n et du nombre de neurones dans la couche cachée. Pour un nombre de neurones dans la couche cachée égal à (M) les dimensions de W^1 et W^2 seront comme suit :

$$\dim(W^1) = (M,\ m+n) \ \text{ et } \ \dim(W^2) = (M,\ 1). \tag{2-51}$$

En remplaçant la quantité $f(x_i)$ de l'équation (2-48) par sa valeur dans l'équation (2-50), on obtient :

$$\begin{aligned}\hat{y}(k) = & w_1^2\left(f_0 + f_1 x_1 + \ldots + f_q x_1^q\right) \\ & + w_2^2\left(f_0 + f_1 x_2 + \ldots + f_q x_2^q\right) \\ & + \ldots + w_M^2\left(f_0 + f_1 x_M + \ldots + f_q x_M^q\right).\end{aligned} \tag{2-52}$$

En remplaçant (x_i) par son expression donnée par l'équation (2-49), l'expression de la sortie du réseau pour un degré de non linéarité $q = 2$ sera comme suit :

$$\begin{aligned}\hat{y}(k) = & f_0 \sum_{i=1}^{M} w_i^2 + f_1 \sum_{j=1}^{m}\left(w_1^2 w_{j1}^1 + \ldots + w_M^2 w_{jM}^1\right)u(k-j) \\ & + f_1 \sum_{j=m+1}^{m+n}\left(w_1^2 w_{j1}^1 + \ldots + w_M^2 w_{jM}^1\right)y(k-j+m) \\ & + f_2 \sum_{j=1}^{m}\sum_{l=1}^{m}\left(w_1^2 w_{j1}^1 w_{l1}^1 + \ldots + w_M^2 w_{jM}^1 w_{lM}^1\right)u(k-j)u(k-l) \\ & + f_2 \sum_{j=m+1}^{m+n}\sum_{l=m+1}^{m+n}\left(w_1^2 w_{j1}^1 w_{l1}^1 + \ldots + w_M^2 w_{jM}^1 w_{lM}^1\right)y(k-j+m)y(k-l+m) \\ & + f_2 \sum_{j=1}^{m}\sum_{l=m+1}^{m+n}\left(w_1^2 w_{j1}^1 w_{l1}^1 + \ldots + w_M^2 w_{jM}^1 w_{lM}^1\right)u(k-j)y(k-l+m)\end{aligned} \tag{2-53}$$

Les pondérations (W^1 et W^2) des connexions du réseau de neurones sont déterminées avec l'algorithme de rétro propagation du gradient. La déduction des paramètres du modèle NARMA se fait à partir des pondérations (W^1 et W^2) et des paramètres du polynôme considéré ($f_0, f_1, \ldots, f_q$). En effet, par identification de l'équation (2-53) avec l'équation (1-7), on trouve les coefficients des termes du modèle NARMA [Ki, 1997].

Exemple : pour $q = 2$ et (m, n) quelconques, on aura le polynôme d'ordre 2 suivant :

$$f(x_i) = f_0 + f_1 x_i + f_2 x_i^2 \tag{2-54}$$

Les paramètres du modèle NARMA seront alors obtenues comme suit :

$$\bar{y} = \sum_{j=1}^{M} f_0 w_j^2$$

$$a_i = \sum_{j=1}^{M} w_j^2 f_1 \; w_{ij}^1 \qquad i = 1,...,m$$

$$b_i = \sum_{j=1}^{M} w_j^2 f_1 \; w_{ij}^1 \qquad i = m+1,...,m+n$$

$$a_{ij} = \sum_{j=1}^{M} w_j^2 f_2 \; w_{ij}^1 w_{jl}^1 \qquad i = j, \quad i = 1,...,m$$

$$a_{ij} = 2\sum_{j=1}^{M} w_j^2 f_2 \; w_{ij}^1 w_{jl}^1 \qquad i \neq j, \quad i = 1,...,m \; ; j = i+1,...,m$$

$$b_{ij} = \sum_{j=1}^{M} w_j^2 f_2 \; w_{ij}^1 w_{jl}^1 \qquad i = j, \quad i = m+1,...,m+n$$

$$b_{ij} = 2\sum_{j=1}^{M} w_j^2 f_2 \; w_{ij}^1 w_{jl}^1 \qquad i \neq j, \quad i = m+1,...,m+n \; ; j = i+1,...,n$$

$$c_{ij} = 2\sum_{j=1}^{M} w_j^2 f_2 \; w_{ij}^1 w_{jl}^1 \qquad i = 1,...,m \; ; j = m+1,...,m+n \qquad (2\text{-}55)$$

La qualité du modèle obtenu, avec cette méthode, dépend du choix adéquat des coefficients du polynôme ($f_0, f_1, ..., f_q$). Dans [Ki, 1997], on ne retrouve aucune indication concernant le choix de ces coefficients. Dans le but de réduire le nombre d'essais, on propose l'incorporation d'une procédure basée sur l'algorithme génétique réel avec l'algorithme de rétro propagation du gradient (figure 2-5).

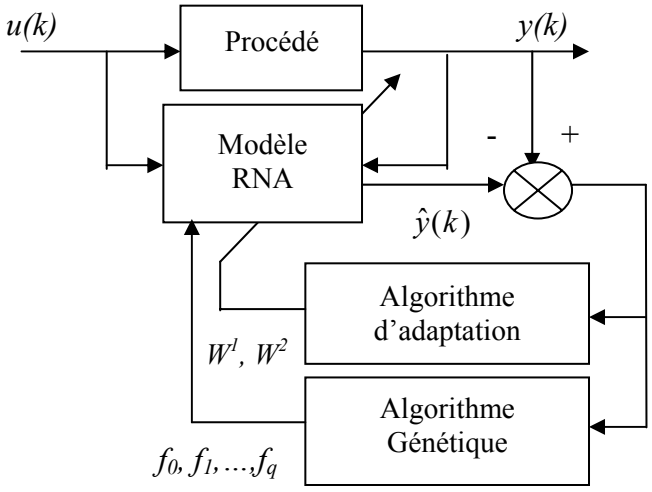


Figure 2-5 : Structure du superviseur basé sur l'algorithme génétique

Chaque individu de la population de l'algorithme génétique caractérise les coefficients du polynôme de la fonction d'activation des neurones. La population initiale de l'algorithme

génétique est choisie d'une manière aléatoire à partir d'un intervalle fixé. Chaque génération de l'algorithme génétique consiste à prendre les coefficients donnés par l'individu de la population et à réaliser l'apprentissage du réseau neuronal avec l'algorithme de rétro propagation. Pour chaque individu, on calcule une fonction d'évaluation (i.e. : la somme des erreurs quadratiques entre la sortie du système et celle du réseau de neurones). Ensuite, les opérateurs génétiques (sélection, croisement et mutation) sont utilisés pour former les individus de la nouvelle population. Dans ce travail, on a considéré la sélection basée sur la roue de loterie, le croisement arithmétique et la mutation uniforme. Les nouveaux individus (e_1 et e_2) sont obtenus à l'aide des parents (p_1 et p_2) en utilisant les deux relations suivantes :

$$e_1 = \alpha_c \; p_1 + (1 - \alpha_c) \; p_2 \tag{2-56}$$

$$e_2 = (1 - \alpha_c) \; p_1 + \alpha_c \; p_2, \qquad \alpha_c \in [0, 1] \tag{2-57}$$

La mutation consiste à modifier aléatoirement la valeur de l'individu en respectant son intervalle de variation.

Les étapes de l'algorithme génétique seront activées jusqu'à une condition d'arrêt est atteinte. Dans notre travail, on a considéré le nombre maximal de génération de l'algorithme génétique comme condition d'arrêt. La population initiale de l'algorithme génétique est choisie d'une manière aléatoire. De même, les valeurs des matrices de pondération initiales du réseau de neurones sont choisies aléatoirement dans l'intervalle [-1,1].

Le paragraphe suivant comporte les résultats de simulation pour les trois méthodes présentées précédemment.

2.4. Résultats de simulation

Le choix des paramètres (*m, n* et *q*) a été fixé après le calcul du coefficient de corrélation total R^2_{tot}, qui est exprimé par l'équation (2-11). Le coefficient de corrélation total est calculé en utilisant la méthode de Kortmann. Les paramètres (*m, n* et *q*) qui donnent la plus grande valeur de R^2_{tot} sont alors utilisés pour caractériser le modèle NARMA. Trois critères d'informations sont utilisés (les coefficients de corrélations : total R^2_{tot}, multiple R^2_{mult} et ajustable R^2_{ajust}) pour décider si le modèle obtenu est acceptable ou non [Haber, 1990]. En

effet, si l'un de ces critères est proche de ''1'' alors le modèle est acceptable [Haber, 1990]. Les paramètres de réglage utilisés dans les simulations sont résumés dans le tableau 2-1.

	Algorithme génétique binaire	Algorithme génétique réel	Réseau de neurones artificiels
- nombre de génération	40	40	-
- nombre d'individus	50	100	-
- probabilité de croisement	0.4	0.8	-
- probabilité de mutation	0.05	0.1	-
- nombre de passages	-	-	3000
- coef. d'apprentissage	-	-	5.10^{-3}

Tableau 2-1 : Paramètres des algorithmes

Au cours de la phase d'identification, le signal d'entrée utilisé est un bruit blanc de moyenne nulle et de variance égale à un. On a considéré la notation suivante :

Méthode K : méthode basée sur l'algorithme de Kortmann.
Méthode AGB : méthode basée sur l'Algorithme Génétique Binaire.
Méthode RN : méthode basée sur l'utilisation des Réseaux de Neurones Artificielles.

On a considéré quatre systèmes non linéaires et monovariables.

Système 1 : le premier système simulé est caractérisé par l'équation aux différences suivante :

$$y(k) = 0.3528\ u(k-1) + 0.9844\ y(k-1) + 0.1892\ u^2(k-1) + 0.6235\ u(k-1)y(k-1) - 0.283\ y^2(k-1). \quad (2\text{-}58)$$

Dans ce cas, le modèle NARMA estimé est caractérisé par 14 termes ($n=m=q=2$). Le tableau 2-2 comporte les termes sélectionnés ainsi que les paramètres estimés avec les trois méthodes. Les coefficients du polynôme de la fonction d'activation identifiés par l'algorithme génétique sont les suivants : f_0 = -4.6758; f_1=14.5596; f_2=15.

D'après les résultats du tableau 2-2, on constate que les modèles identifiés sont pratiquement identiques. En effet, les termes des modèles ainsi que les paramètres correspondants sont identifiés avec succès par les trois méthodes. Les valeurs des coefficients de corrélation (R^2_{tot}, R^2_{mult} et R^2_{ajust}) sont toutes proches de un. Ceci signifie que les modèles

identifiés avec les différentes méthodes caractérisent convenablement le comportement du système, par conséquent, ils sont tous acceptables.

Pour valider les modèles trouvés, on a comparé les réponses des modèles avec la réponse du système considéré pour une séquence du signal d'entrée différente de la séquence utilisée en identification. La figure 2-6, présente la séquence d'entrée et les sorties des modèles identifiés. Cette figure montre que la sortie du système et les sorties des modèles trouvés sont presque confondues.

N°	Termes possibles	Modèle Simulé (termes et paramètres)		Les modèles trouvés avec les différentes méthodes		
				K	**AGB**	**RN**
1	***u(k-1)***	***x***	*0.3528*	*0.3528*	*0.3527*	*0.3528*
2	***u(k-2)***					*0*
3	***y(k-1)***	***x***	*0.9844*	*0.9843*	*0.9842*	*0.9845*
4	***y(k-2)***					*0*
5	***(u(k-1))²***	***x***	*0.1892*	*0.1889*	*0.1879*	*0.1892*
6	***u(k-1)u(k-2)***					*0*
7	***(u(k-2))²***					*0*
8	***u(k-1)y(k-1)***	***x***	*0.6235*	*0.6235*	*0.6236*	*0.6234*
9	***u(k-1)y(k-2)***					*0.0001*
10	***u(k-2)y(k-1)***					*-0.0001*
11	***u(k-2)y(k-2)***					*0*
12	***(y(k-1))²***	***x***	*-0.283*	*-0.2831*	*-0.2830*	*-0.283*
13	***y(k-1)y(k-2)***					*0*
14	***(y(k-2))²***					*0*
15	$\bar{y}$		*0*	*0.0005*	*0.0021*	*0*
	R^2_{tot}			*0.9997*	*0.9918*	*0.9996*
	R^2_{mult}			*1.0075*	*1.0418*	*0.9973*
	R^2_{ajust}			*1.0076*	*1.0422*	*0.9972*

Tableau 2-2 : Structures de l'équation simulée et des modèles identifiés (système 1)

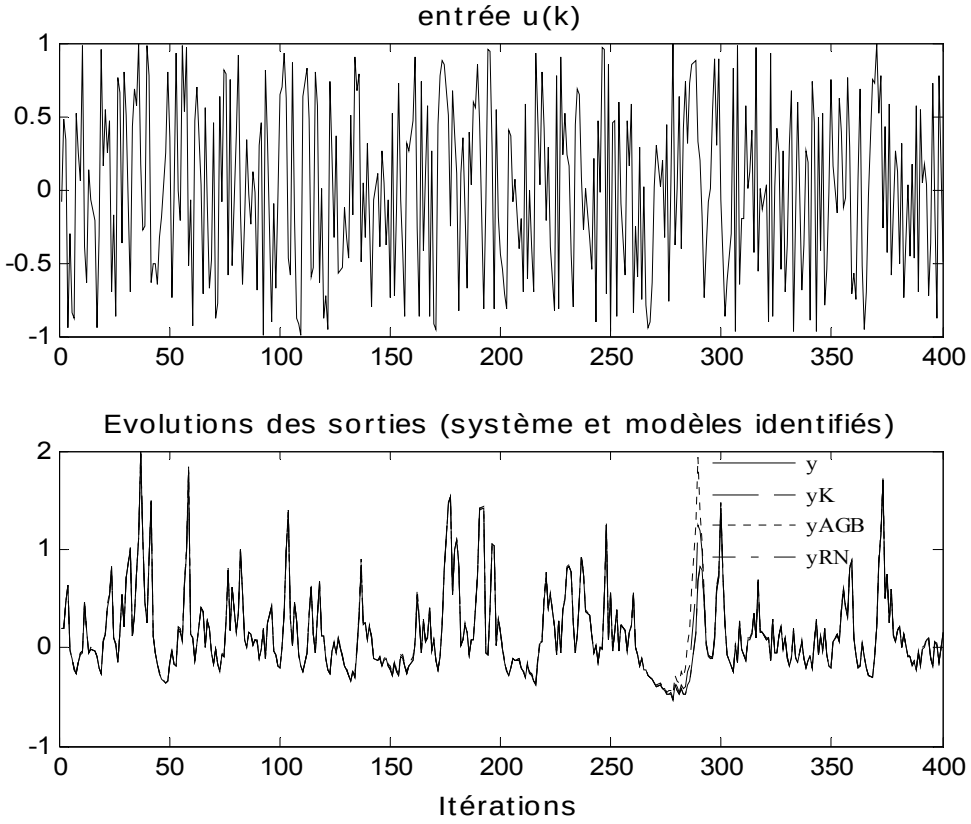


Figure 2-6 : Les évolutions de la séquence d'entrée, de la sortie du système et des sorties des modèles considérés (système 1)

Afin d'étudier l'influence de bruit sur la qualité de l'estimation, on a ajouté à la sortie du système un bruit blanc de moyenne nulle et de variance finie. Le tableau 2-3 comporte les paramètres estimés par les trois méthodes dans le cas où la variance de bruit est égale à 7.1 10^{-3}. Le tableau 2-4 présente les valeurs du coefficient de corrélation multiple R^2_{mult} pour différentes valeurs de la variance de bruit. L'analyse des résultats donnés sur les tableaux 2-3 et 2-4 permet de conclure que la présence de bruit de mesure entraîne une légère dégradation de la qualité des modèles trouvés. En effet, les paramètres estimés sont biaisés et les modèles sont moins précis puisque les valeurs de R^2_{mult} diminuent au fur et à mesure que la variance de bruit augmente.

Numéro du paramètre	Les modèles trouvés avec les différentes méthodes		
	K	**AGB**	**RN**
1	*0.3579*	*0.3552*	*0.3597*
2			*0.0300*
3	*0.9286*	*0.9961*	*0.9357*
4			*0.0165*
5	*0.1806*	*0.1876*	*0.1932*
6			*0.0108*
7			*-0.0002*
8	*0.5951*	*0.6374*	*0.6130*
9		*-0.0208*	*0.0125*
10		*-0.0082*	*0.0414*
11			*0.0012*
12	*-0.2928*	*-0.2807*	*-0.3074*
13		*-0.0176*	*0.0449*
14			*-0.0011*
15	*0.012*	*0.0021*	*-0.0366*

Tableau 2-3 : Paramètres du modèle NARMA pour une variance du bruit égale à 7.1 10^{-3} (système 1)

Variance du bruit	R^2_{mult}		
	K	**AGB**	**RN**
3.1 10^{-3}	*0.9837*	*0.9895*	*0.9854*
7.1 10^{-3}	*0.9709*	*0.9814*	*0.9737*
13 10^{-3}	*0.9516*	*0.9626*	*0.9545*

Tableau 2-4 : Les valeurs de R^2_{mult} pour différentes valeurs de la variance du bruit (système 1)

Système 2 : le fonctionnement du deuxième système est décrit par la relation suivante [Narendra, 1997] :

$$y(k) = \frac{y(k-1)}{1 + y^2(k-1)} + u^2(k-1) \tag{2-59}$$

Le choix ($n=m=q=2$) permet d'avoir la plus grande valeur du coefficient de corrélation total R^2_{tot}. Les différents termes possibles du modèle NARMA sont présentés dans le tableau 2-5. Les coefficients du polynôme de la fonction d'activation identifiés par l'algorithme génétique réel sont les suivants : f_0=0.4388; f_1=5.8 et f_2=7.41.

D'après les résultats obtenus on constate que :

- le modèle identifié par la méthode de Kortmann, comporte seulement cinq termes,
- la méthode utilisant l'algorithme génétique binaire permet d'avoir trois représentations possibles, donc trois modèles possibles malgré leurs ressemblances.

La première et la deuxième représentation comportent 9 termes et la troisième comporte 10 termes. Si on écarte les termes dont la valeur absolue du paramètre correspondant est inférieure à (0.01) on aura alors un modèle qui comporte seulement sept termes,

- la troisième méthode suppose l'existence de tous les termes possibles du modèle NARMA. La sélection des termes du modèle choisi est basée sur les valeurs des paramètres correspondants. Ces paramètres, contrairement aux autres méthodes ne sont pas estimés par la méthode de MCR. Dans cet exemple, si on écarte les termes dont les paramètres sont inférieurs à (0.001) en valeur absolue, on aura alors un modèle qui comporte seulement sept termes,
- les valeurs des coefficients de corrélation (R^2_{tot}, R^2_{mult} et R^2_{ajust}) sont toutes proches de un. Ceci signifie que les modèles identifiés avec les différentes méthodes caractérisent convenablement le comportement du système, par conséquent, ils sont tous acceptables.

Termes	Les modèles identifiés avec les différentes méthodes				
	K	**AGB**			**RN**
	Termes et paramètres estimés				
u(k-1)		***-0.0121***	***-0.0118***	***-0.0119***	***0.0034***
u(k-2)		***-0.0165***	***-0.0148***	***-0.0172***	***-0.0004***
y(k-1)	***0.5223***	***0.3559***	***0.3490***	***0.3495***	***-0.0146***
y(k-2)		***0***	***-0.0032***	***-0.0034***	***-0.0045***
(u(k-1))²	***0.9988***	***0.9991***	***0.9994***	***0.9993***	***1.0034***
u(k-1)u(k-2)					***-0.0027***
(u(k-2))²	***-0.3923***	***-0.2316***	***-0.2246***	***-0.2254***	***0***
u(k-1)y(k-1)		***0.0136***	***0.0133***	***0.0133***	***-0.003***
u(k-1)y(k-2)					***-0.0023***
u(k-2)y(k-1)		***0.0082***	***0.0080***	***0.0081***	***0***
u(k-2)y(k-2)		***0.0016***	***0***	***0.0024***	***0***
(y(k-1))²	***-0.0392***	***-0.0388***	***-0.0389***	***-0.0388***	***0***
y(k-1)y(k-2)					***0***
(y(k-2))²					***0***
$\bar{y}$	***0.1897***	***0.2632***	***0.2690***	***0.2690***	***0.4418***
R^2_{tot}	*0.9999*	*0.9965*	*0.9965*	*0.9965*	*0.9972*
R^2_{mult}	*0.9864*	*0.9844*	*0.9844*	*0.9843*	*0.9922*
R^2_{ajust}	*0.9863*	*0.9841*	*0.9840*	*0.9840*	*0.9917*

Tableau 2-5 : Structures des modèles identifiés (système 2)

Pour valider les modèles trouvés, on a comparé la réponse du système avec les réponses des modèles. Dans ce cas, la séquence d'entrée est un bruit blanc de moyenne nulle et de variance finie. La figure 2-7 présente la séquence d'entrée, la sortie du système et les sorties des modèles trouvés. La figure 2-8 présente les différentes sorties entre les itérations 19 et 41. On remarque que les sorties des modèles identifiés sont pratiquement confondues avec la sortie du système.

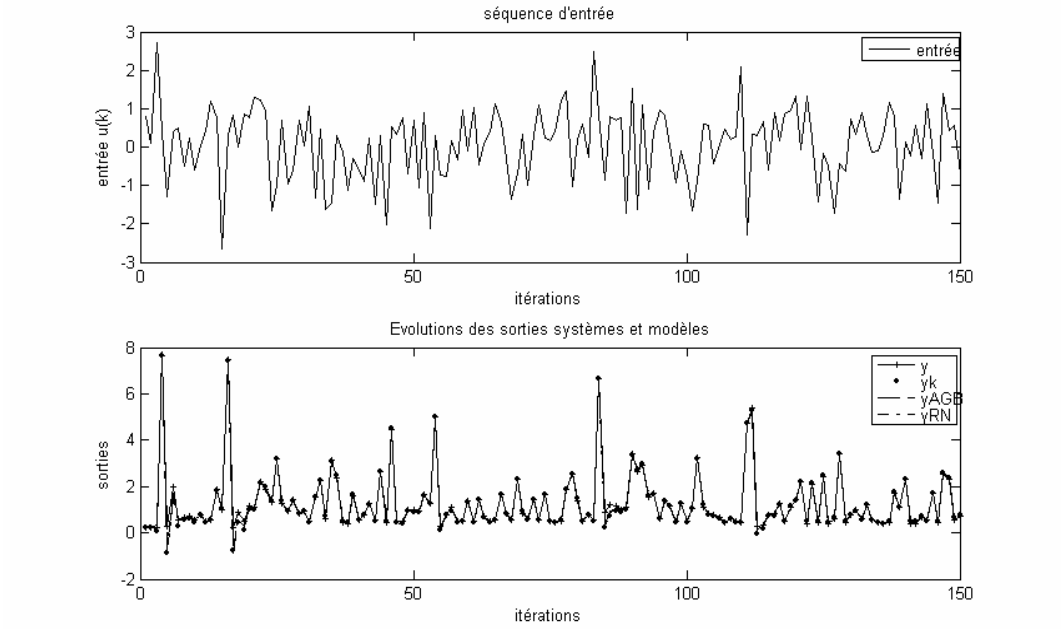


Figure 2-7 : Les évolutions de la séquence d'entrée, de la sortie du système et des sorties des modèles considérés (système 2)

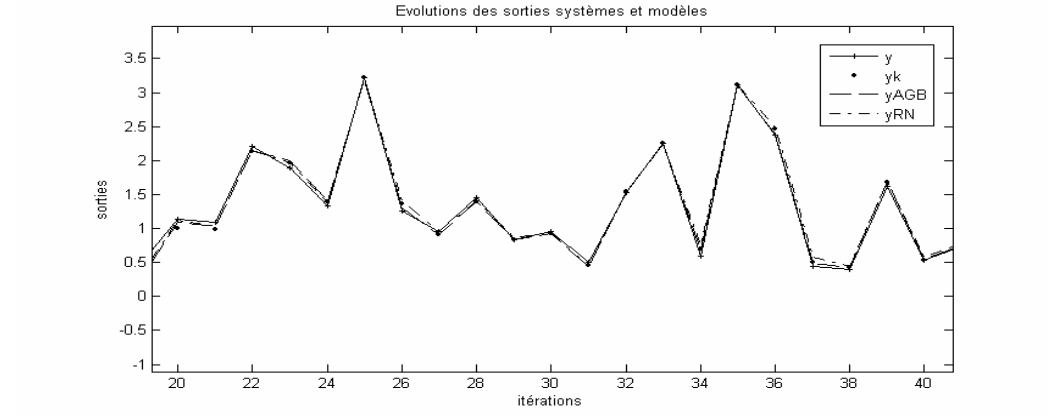


Figure 2-8 : Les évolutions des sorties (système et modèles).

Pour vérifier la robustesse des différents modèles identifiés, on a ajouté à la sortie du système un bruit de variance finie. Le rapport entre la variance du bruit et l'amplitude du

signal d'entrée est noté R_v. Le signal d'entrée utilisé dans ce cas est décrit par la relation suivante :

$$u(k)=\begin{cases}1.6 & pour\ \ k=1:100\\ 1.6\ \exp((k-100)/5) & pour\ \ k=101:200\\ 0.013\,k-2.5373 & pour\ \ k=201:300\\ 1.6\ \sin(\frac{\pi}{2}+\frac{k}{5\pi}) & pour\ \ k=301:400\end{cases} \qquad (2\text{-}60)$$

La figure 2-9(a) représente l'évolution des sorties du système et des modèles considérés ainsi que les erreurs de modélisation pour un rapport R_v=2%. La figure 2-9(b) représente les mêmes résultats pour R_v=10%. Le tableau 2-6 présente les variances de l'erreur de validation pour chaque méthode présentée.

	Var. Err. (**K**)	Var. Err. (**AGB**)	Var. Err. (**RN**)
R_v=2%	0.02	0.0205	0.0193
R_v=10%	0.1628	0.1650	0.1621

Tableau 2-6 : Variance des erreurs de validation

On remarque, d'après les figures 2-9(a) et 2-9(b), que même avec un rapport R_v qui atteint 10%, la sortie du modèle ne diverge pas par rapport à la sortie du système. Le tableau 2-6 montre que les variances des erreurs, pour les trois méthodes, sont faibles. Ce qui valide la robustesse des modèles identifiés. Néanmoins, la variance de l'erreur obtenue par la troisième méthode (RN) est moins faible que les variances obtenues par les deux autres méthodes.

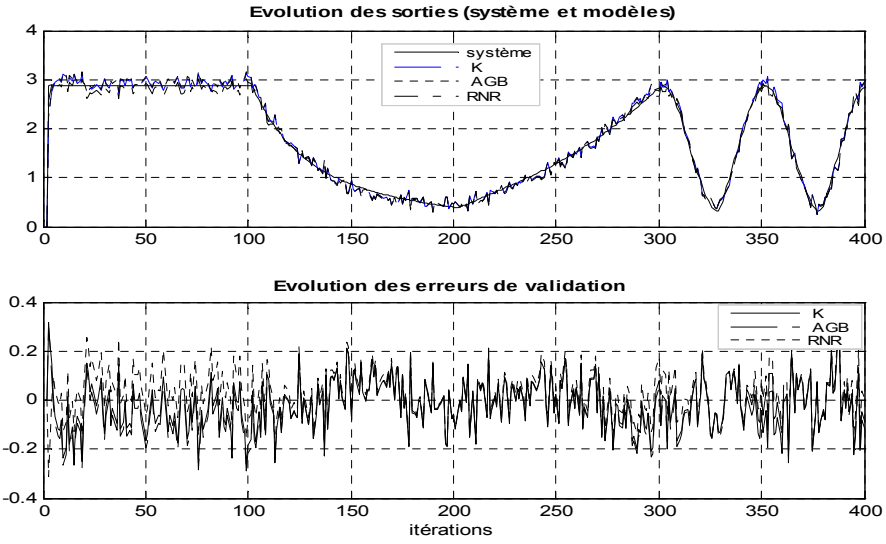


Figure 2-9(a) : Evolutions des sorties (système et modèles)
et des erreurs de validation (système 2, R_v =2%)

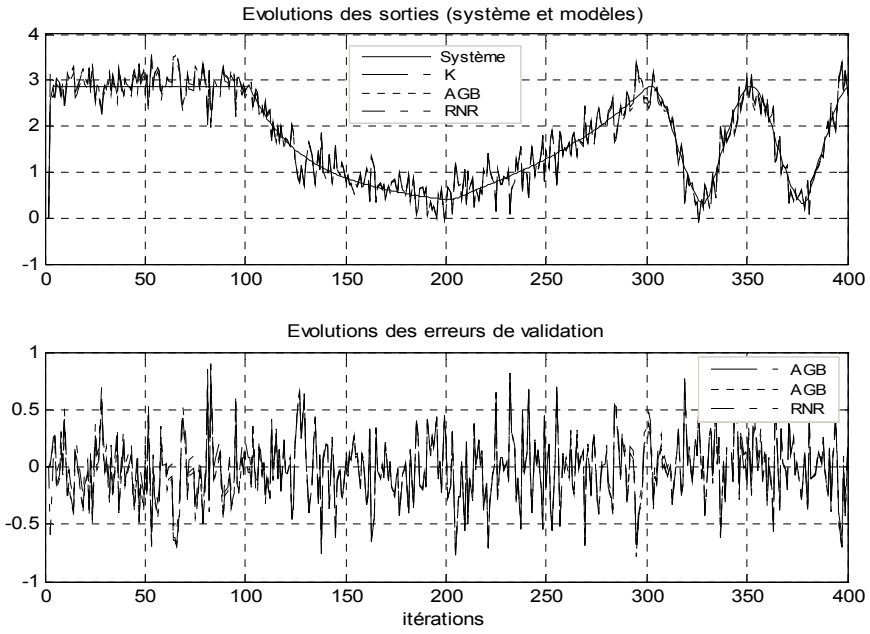


Figure 2-9(b) : Evolutions des sorties (système et modèles) et des erreurs de validation (système 2, R_v =10%)

Système 3 : le troisième système est décrit par [Chen, 1999] :

$$\begin{aligned} x_1(k+1) &= 0.1x_1(k) + 2\frac{u(k)+x_2(k)}{1+\left(u(k)+x_2(k)\right)^2} \\ x_2(k+1) &= 0.1x_2(k) + u(k)\left(1+\frac{u^2(k)}{1+\left(x_1^2(k)+x_2^2(k)\right)}\right) \\ y(k) &= x_1(k)+x_2(k). \end{aligned} \tag{2-61}$$

Le tableau 2-7 comporte les valeurs de R^2_{mult} en fonction des paramètres m, n et q. On constate que la plus grande valeur de R^2_{mult} est obtenue lorsque n=2 et m=q=3. Par conséquent, le modèle NARMA considéré est caractérisé par 55 termes.

m n q	R^2_{mult}		m n q	R^2_{mult}
2 2 2	87.5018		2 2 3	96.0043
2 3 2	87.5018		**3 2 3**	**96.5732**
3 3 2	87.5018		3 4 3	95.4492
4 3 2	87.5018		4 5 3	96.4712
1 2 3	91.1783		5 4 3	94.9868
2 1 3	96.1299		5 5 3	95.4942

Tableau 2-7 : Evolution de R^2_{mult} en fonction des paramètres m, n et q (système 3)

Le tableau 2-8 comporte les termes sélectionnés, les paramètres estimés, les valeurs de R^2_{tot} ainsi que le temps demandé par chaque méthode. Les coefficients du polynôme de la fonction d'activation identifiés par l'algorithme génétique réel sont les suivants : f_0=3.35; f_1=3.43; f_2=-4.83 et f_3=-2.05. D'après le tableau 2-8, on constate que la méthode de Kortmann a conduit à un modèle NARMA caractérisé par un nombre réduit des paramètres. De plus, cette méthode nécessite un temps très faible par rapport aux deux autres approches. Le nombre de termes du modèle obtenu par la méthode (RN) peut être réduit en supprimant les termes ayant des paramètres de valeurs très faibles.

N°	Termes possibles	***K***	***AGB***	***RN***	N°	Termes possibles	***K***	***AGB***	***RN***
0	$\bar{y}$	*-0.0444*	*-0.0600*	*0.1295*	*28*	$u^2(k-1)\ y(k-1)$		*0.0222*	*-0.0103*
1	*u(k-1)*	*2.1089*	*2.2439*	*1.0432*	*29*	$u^2(k-1)\ y(k-2)$	*-0.071*	*-0.0685*	*-0.0341*
2	*u(k-2)*	*1.2887*	*1.0852*	*-0.0046*	*30*	$u(k-1)\ u(k-2)\ y(k-1)$	*-0.715*		*0.1493*
3	*u(k-3)*	*0.2361*	*0.1517*	*0.0166*	*31*	$u(k-1)\ u(k-2)\ y(k-2)$		*-0.0413*	*-0.0289*
4	*y(k-1)*		*0.0357*	*0.2333*	*32*	$u(k-1)\ u(k-3)\ y(k-1)$			*-0.0488*
5	*y(k-2)*			*0.2299*	*33*	$u(k-1)\ u(k-3)\ y(k-2)$			*-0.0441*
6	$u^2(k-1)$	*0.1337*	*0.1078*	*0.0368*	*34*	$u^2(k-2)\ y(k-1)$			*0.4282*
7	*u(k-1) u(k-2)*		*-0.0938*	*-0.0172*	*35*	$u^2(k-2)\ y(k-2)$		*-0.0041*	*-0.0261*
8	*u(k-1) u(k-3)*		*0.0407*	*-0.0012*	*36*	$u(k-2)\ u(k-3)\ y(k-1)$		*0.0032*	*-0.2515*
9	$u^2(k-2)$		*0.3288*	*-0.3361*	*37*	$u(k-2)\ u(k-3)\ y(k-2)$		*0.0134*	*-0.0144*
10	*u(k-2) u(k-3)*		*-0.3308*	*0.1543*	*38*	$u^2(k-3)\ y(k-1)$		*-0.0067*	*0.0169*
11	$u^2(k-3)$		*-0.6170*	*0.0568*	*39*	$u^2(k-3)\ y(k-2)$		*-0.0089*	*-0.0415*
12	$u^3(k-1)$	*0.3172*		*-0.0344*	*40*	$u(k-1)\ y^2(k-1)$	*0.1465*		*-0.0865*
13	$u^2(k-1)\ u(k-2)$			*-0.0056*	*41*	$u(k-2)\ y^2(k-1)$			*-0.2774*
14	$u^2(k-1)\ u(k-3)$		*-0.0364*	*0.0176*	*42*	$u(k-3)\ y^2(k-1)$			*0.1519*
15	$u(k-1)\ u^2(k-2)$		*0.0267*	*-0.0704*	*43*	$u(k-1)\ y(k-1)\ y(k-2)$			*-0.0001*
16	*u(k-1) u(k-2) u(k-3)*			*0.0488*	*44*	$u(k-2)\ y(k-1)\ y(k-2)$			*0.1989*
17	$u(k-1)\ u^2(k-3)$			*0.0005*	*45*	$u(k-3)\ y(k-1)\ y(k-2)$			*-0.0371*
18	$u^3(k-1)$	*-0.269*		*-0.1228*	*46*	$u(k-1)\ y^2(k-2)$		*0.0092*	*0.0524*
19	$u^2(k-2)\ u(k-3)$			*0.1222*	*47*	$u(k-2)\ y^2(k-2)$		*0.0230*	*0.0738*
20	$u(k-2)\ u^2(k-3)$		*-0.0904*	*-0.0119*	*48*	$u(k-3)\ y^2(k-2)$		*-0.0043*	*0.0049*
21	$u^3(k-3)$			*-0.0055*	*49*	$y^2(k-1)$		*-0.0011*	*-0.1593*
22	*u(k-1) y(k-1)*		*-0.2168*	*0.1097*	*50*	*y(k-1) y(k-2)*			*0.7010*
23	*u(k-1) y(k-2)*		*-0.0346*	*0.0032*	*51*	$y^2(k-2)$		*0.0032*	*-0.0340*
24	*u(k-2) y(k-1)*		*0.0018*	*0.3310*	*52*	$y^3(k-1)$			*-0.0536*
25	*u(k-2) y(k-2)*			*-0.0852*	*53*	$y^2(k-1)\ y(k-2)$			*-0.0094*
26	*u(k-3) y(k-1)*		*0.0192*	*-0.1431*	*54*	$y(k-1)\ y^2(k-2)$			*-0.0769*
27	*u(k-3) y(k-2)*			*-0.0565*	*55*	$y^3(k-2)$			*-0.0148*

R^2_{mult}	*0.9657*	*0.9751*	*0.9740*
Temps (sec)	*9.5505*	*61.501*	*2665.09*

Tableau 2-8 : Les valeurs des paramètres estimés, de R^2_{mult} et du temps de calcul (système 3)

Système 4 : le quatrième système est décrit par [Wen, 2004]:

$$y(k) = \frac{y(k-1)\,y(k-2)\left(y(k-1)+2.5\right)}{1+y^2(k-1)+y^2(k-2)} + u(k-1) \tag{2-62}$$

Dans ce cas, le modèle NARMA estimé est caractérisé par 219 termes (n=5, m=4 et q=3). En effet, ce choix des paramètres m, n et q a conduit à la plus grande valeur de R^2_{mult} (tableau 2-9). Les modèles trouvés par les méthodes (K) et (AGB) sont donnés par les équations (2-63) et (2-64). Les paramètres du modèle NARMA estimés par la méthode (RN) sont donnés dans l'annexe 2. Les coefficients du polynôme de la fonction d'activation identifiés par l'algorithme génétique réel sont les suivants : f_0=0.41; f_1=-3.00; f_2=-1.59 et f_3=-0.47.

m n q	R^2_{mult}		m n q	R^2_{mult}
2 2 2	88.4192		3 2 3	93.9618
3 2 2	88.6387		2 3 3	94.5875
2 3 2	90.0294		3 3 3	95.9938
3 3 2	90.3396		3 4 3	95.0294
4 3 2	90.3396		4 3 3	96.0186
4 4 2	88.8969		4 4 3	96.0007
1 1 3	69.7119		**4 5 3**	**96.5307**
1 2 3	89.8282			
2 1 3	79.6418		5 4 3	95.9317
2 2 3	92.9585		5 5 3	96.1293

Tableau 2-9 : Evolution de R^2_{mult} en fonction des paramètres m, n et q (système 4)

- Méthode de Kortmann :

$$\begin{aligned}\hat{y}(k) &= 0.0173 + 0.99u(k-1) - 0.1045u(k-3)\\ &+ 0.1433y(k-1) + 01715y(k-2) - 0.0464u^2(k-2)\\ &+ 0.0744u(k-3)y(k-2) + 0.3391u(k-2)y(k-1)y(k-2)\\ &- 0.2011u(k-2)y^2(k-1) + 0.1418u(k-3)y^2(k-1)\\ &+ 0.1663u^2(k-2)y(k-1) - 0.1141u(k-3)y(k-1)y(k-2)\\ &- 0.0263u(k-4)y(k-3)y(k-4) + 0.6115y(k-1)y(k-2)\\ &+ 0.1598y(k-1)y(k-3) - 0.2141y^2(k-1)y(k-2)\end{aligned} \tag{2-63}$$

- Méthode AGB :

$$\begin{aligned}\hat{y}(k) = {} & 0.015 + 0.9928\,u(k-1) - 0.606\,u(k-2) + 0.0078\,u(k-3) \\ & + 0.158\,u(k-4) + 0.59\,y(k-1) + 0.26\,y(k-2) - 0.0574\,y(k-4) \\ & + 0.0331\,y(k-5) + 0.1371\,u(k-2)u(k-3) + 0.0677\,u(k-2)u(k-4) \\ & - 0.0713\,u^2(k-3) - 0.0062\,u^3(k-1) - 0.0592\,u^2(k-1)u(k-3) \\ & - 0.0434\,u(k-1)u(k-2)u(k-4) - 0.021\,u(k-1)u^2(k-3) \\ & - 0.022\,u(k-1)u(k-3)u(k-4) + 0.0354\,u(k-1)u^2(k-4) \\ & + 0.1418\,u^2(k-2)u(k-3) - 0.0624\,u(k-2)u^2(k-3) - 0.0438\,u^3(k-3) \\ & + 0.0294\,u(k-2)u(k-3)u(k-4) - 0.1723\,u(k-3)y(k-3) \\ & + 0.0254\,u(k-1)y(k-1) + 0.3764\,u(k-2)y(k-2) + 0.1217\,u(k-3)y(k-2)\end{aligned}$$

(2-64)

Les valeurs de R^2_{mult} ainsi que le temps demandé par chaque méthode sont donnés sur le tableau 2-10. D'après ce tableau, on constate que la plus grande valeur de R^2_{mult} est assurée par la méthode basée sur les réseaux de neurones artificiels. Cependant, cette méthode exige un temps très élevé pour l'estimation des paramètres du modèle NARMA.

	K	**AGB**	**RN**
R^2_{mult}	*0.9653*	*0.9116*	*0.9707*
Temps (sec)	*36.589*	*2198.2*	*2401.03*

Tableau 2-10 : Les valeurs de R^2_{mult} et du temps de calcul (système 4)

2.5. Conclusion

Dans ce chapitre, nous nous sommes intéressé à l'identification paramétrique et structurelle des modèles de type NARMA. Nous avons présenté en premier lieu l'algorithme de régression par pas échelonné modifié, qui est basé sur des critères statistiques. Ensuite, nous avons proposé deux méthodes heuristiques pour l'identification des modèles de type NARMA. La première méthode est basée sur les algorithmes génétiques binaires et la deuxième méthode constitue une combinaison entre le réseau de neurones artificiels à fonction d'activation polynomiale et l'algorithme génétique sous sa représentation réelle. Cette dernière permet de chercher les coefficients de la fonction polynomiale simultanément avec l'apprentissage du réseau de neurones. Ces algorithmes permettent la sélection des

termes ainsi que l'estimation des paramètres du modèle NARMA. Les résultats de simulation ont confirmé l'efficacité des algorithmes proposés. En effet, des modèles NARMA caractérisés par une précision acceptable et par une complexité raisonnable ont été déterminés. L'utilisation de telles méthodes est très recommandée quand nous n'avons aucune connaissance a priori de la structure du système. Cet avantage permet à cette approche de traiter les systèmes complexes.

D'autre part, l'expression de la sortie du modèle NARMA en fonction des paramètres du réseau neuronal (méthode RN) peut être calculée par un programme approprié. Par conséquent, on évite la difficulté d'implémentation de l'algorithme de régression par pas échelonné modifié surtout dans le cas des systèmes multivariables.

Dans le chapitre qui suit, nous allons exploiter les modèles identifiés pour la synthèse d'un correcteur prédictif sous contraintes qui ne nécessite pas le calcul de la dérivée.

Chapitre 3

La commande prédictive non linéaire

Chapitre 3 La commande prédictive non linéaire

Chapitre 3

La commande prédictive non linéaire

3.1. Introduction

Dans ce chapitre, on s'intéresse à la commande prédictive basée sur un modèle de type ARMA non linéaire. L'utilisation d'un modèle non linéaire conduit à un problème de commande non convexe. La minimisation du critère de performance est généralement effectuée par une approche classique tel que la méthode du gradient, qui nécessite le calcul de la dérivée de ce dernier. Cette méthode d'optimisation donne une loi de commande sous optimale à cause de la présence des minimums locaux [Mrabet, 2002], [Filali, 2002], [Kenneth, 1999], [Yonghong, 1996].

Dans notre travail, la loi de commande est calculée par deux méthodes différentes qui ne nécessitent pas le calcul de la dérivée. La première méthode utilise une version améliorée de l'algorithme de Nelder-Mead, appelé encore méthode du simplexe, et la deuxième exploite l'algorithme de Rosenbrock (méthode de recherche multidirectionnelle). Pour traiter correctement les contraintes et pour avoir plus de probabilité de convergence vers des optimums globaux, nous avons introduit des améliorations aux méthodes d'optimisation utilisées.

Tous les développements dans ce chapitre se basent sur une approche discrète de la commande prédictive. On se focalisera essentiellement sur l'application à la Commande Prédictive à base de Modèle MPC.

Le chapitre est organisé comme suit : Dans le deuxième paragraphe on présente le calcul du prédicteur. Le troisième paragraphe concerne le critère de performance. Le paragraphe quatre sera consacré au développement des méthodes d'optimisations utilisées ainsi que les améliorations introduites. Le calcul de la loi de commande est présenté dans le cinquième paragraphe. Les résultats de simulation seront donnés dans le paragraphe six et la conclusion dans le dernier paragraphe.

3.2. Calcul du prédicteur à j - pas :

Le calcul de la loi de commande requiert la définition d'un prédicteur qui permet d'anticiper le comportement futur du processus sur un horizon fini. Le calcul du prédicteur $\hat{y}(k + Hp)$, c'est-à-dire la prédiction effectuée à l'instant *k* de la valeur de la sortie *y* à *Hp* pas d'échantillonnage en avance, nécessite l'exploitation de l'équation du modèle obtenu.

La figure 3-1 présente le schéma de principe du prédicteur avec un horizon de prédiction égal à *Hp*, en utilisant un modèle de type NARMA.

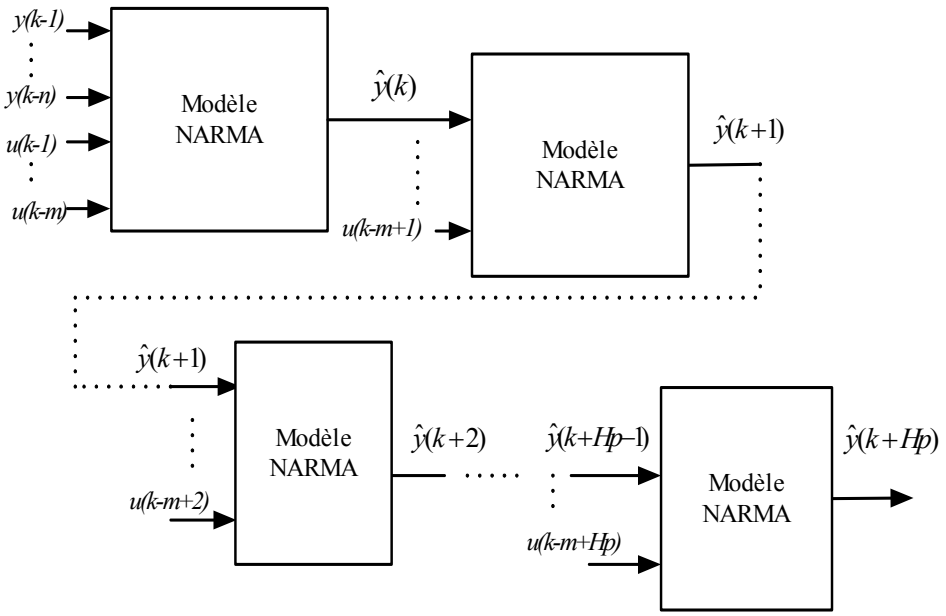


Figure 3-1 : Schéma de principe du prédicteur

L'équation du prédicteur sera déduite à partir de l'équation du modèle obtenu. La relation (3-1) donne l'expression du prédicteur, déduite à partir de l'équation (1-7) du modèle général de type NARMA.

$$
\begin{aligned}
\hat{y}(k+l) = \bar{y} &+ \sum_{i=1}^{m} b_i u(k-i+l) + \sum_{i=1}^{m}\sum_{j=i}^{m} b_{ij} u(k-i+l)u(k-j+l) \\
&+ \ldots + \sum_{i=1}^{m}\ldots\sum_{v=p}^{m}\sum_{h=v}^{m} b_{i\ldots l} u(k-i+l)\ldots u(k-h+l) \\
&+ \sum_{i=1}^{n} a_i \hat{y}(k-i+l) + \sum_{i=1}^{n}\sum_{j=i}^{n} a_{ij} \hat{y}(k-i+l)\hat{y}(k-j+l) \\
&+ \ldots + \sum_{i=1}^{n}\ldots\sum_{v=p}^{n}\sum_{h=v}^{n} a_{i\ldots l} \hat{y}(k-i+l)\ldots\hat{y}(k-h+l) \\
&+ \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} u(k-i+l)\hat{y}(k-j+l) \\
&+ \ldots + \sum_{i=1}^{m}\ldots\sum_{v=1}^{m}\sum_{l=1}^{n} c_{i\ldots l} u(k-i+l)\ldots u(k-v+l)\hat{y}(k-l) \\
&+ \ldots + \sum_{i=1}^{m}\sum_{j=1}^{n}\ldots\sum_{l=1}^{n} c_{ij\ldots l} u(k-i+l)\hat{y}(k-j+l)\ldots\hat{y}(k-l).
\end{aligned}
\tag{3-1}
$$

3.3. Le critère de performance

La détermination de la loi de commande prédictive est associée à la minimisation d'un critère quadratique, à horizon fini, qui fait intervenir deux termes quadratiques, le premier est lié aux erreurs futures entre la sortie prédite et la consigne et le deuxième correspond aux incréments futurs de commande.

Le critère utilisé pour calculer la loi de commande est donné par la relation suivante :

$$
J_2(k) = \frac{1}{2}\left[\sum_{i=1}^{Hp} (\hat{y}(k+i) - y_c(k+i))^2 + \lambda \sum_{i=1}^{Hc} (\Delta u(k+i-1))^2\right] \tag{3- 2}
$$

avec :

Hp: horizon de prédiction sur la sortie,

Hc: horizon de commande,

λ: coefficient de pondération sur la commande qui permet de donner plus ou moins de poids à la minimisation de l'énergie (commande) par rapport à l'erreur de poursuite.

On note par ΔU le vecteur des incréments de commande, qui est donné par la relation suivante :

$$\Delta U = [\Delta u(k) \quad ... \quad \Delta u(k + Hc - 1)]^T \tag{3-3}$$

Puisque le modèle est non linéaire, le critère J_2 est non convexe vis-à-vis du vecteur ΔU. Par conséquent, l'utilisation d'une méthode classique d'optimisation, telle que la méthode de descente du gradient, peut conduire à une solution locale.

Dans ce qui suit, nous présentons deux méthodes d'optimisation utilisées pour l'obtention de la loi de commande, ces deux méthodes sont d'ordre zéro, i.e. ne nécessitent pas le calcul de la dérivée.

3.4. Optimisation du critère

3.4.1. Généralités sur les méthodes d'optimisation

Un facteur très important qui n'est pas traité par la théorie de la commande prédictive mais qui a une importance pratique est celui de l'influence des temps de calculs. Si le temps de calcul de la loi de commande est supérieur au temps d'application (la période d'échantillonnage), on risque d'engendrer une instabilité dans le schéma de commande prédictive. On a intérêt, donc, à réduire le temps de calcul par rapport à la constante de temps du système à commander. Malgré l'arrivée de calculateurs qui sont de plus en plus puissants, cette contrainte reste importante dans le cas des systèmes rapides. Ceci explique le grand succès de la commande prédictive dans le domaine du génie des procédés par exemple et son succès plus que modeste dans d'autres domaines tel que l'aéronautique. En effet, le temps de calcul de la loi de commande dépend aussi bien de la complexité du système que de la méthode utilisée pour la résolution du problème.

La formule (3-4) présente la formulation mathématique d'un problème d'optimisation de dimension *n* :

$$\begin{cases} Min\ f(x) & \in \Re^{n} \\ g_{i}(x) & \leq 0 \quad i = 1,...,p \\ h_{j}(x) & = 0 \quad j = 1,...,q \\ x_{l\,\min} & \leq x_{l} \leq x_{l\,\max} \quad l = 1,...,n \end{cases} \tag{3-4}$$

où

$f(x)$ est le critère à minimiser appelé aussi fonction objectif

x est un vecteur à n variables qui représentent les paramètres du problème à optimiser

$g_i(x)$ et $h_j(x)$ représentent respectivement les contraintes d'inégalité et d'égalité,

$x_l\ min$ et $x_l\ max$ désignent les contraintes de domaine.

$\Re^n$ est l'espace de recherche borné par les contraintes de domaine.

La solution d'un problème d'optimisation est alors donnée par un ensemble de paramètres x^* pour lesquels la fonction objectif présente une valeur minimale, en respectant les contraintes d'égalité, d'inégalité et de domaine.

Selon les caractéristiques du problème d'optimisation, nous pouvons appliquer différentes méthodes de résolution pour identifier sa solution. Ces méthodes sont classées en deux grands groupes: les méthodes déterministes et les méthodes stochastiques.

Dans les méthodes d'optimisation déterministes l'évolution vers la solution du problème est toujours la même pour un même contexte (point de départ). Ce sont en général des méthodes efficaces, peu coûteuses, mais qui nécessitent une configuration initiale (point de départ) pour résoudre le problème. Ce sont souvent des méthodes locales, c'est-à-dire qu'elles convergent vers l'optimum le plus proche du point de départ.

Selon la dimension de la fonction objectif à optimiser, les méthodes déterministes peuvent être classifiées en unidimensionnelles ou multidimensionnelles.

Parmi les méthodes déterministes unidimensionnelles, aussi appelées méthodes de Recherche Linéaire (Line Search Methods), on distingue la méthode de Dichotomie [Culioli, 1994], la méthode de la Section Dorée [Culioli, 1994] [Press, 1992] et la méthode de Brent [Brent, 1973] [Press, 1992]. La plupart de ces méthodes ne supposent pas que la fonction à minimiser soit différentiable, ni même continue, mais seulement uni modale [Culioli, 1994].

Les méthodes déterministes multidimensionnelles, quant à elles, sont consacrées à l'optimisation de fonctions à un paramètre ou plus. Elles peuvent être classées selon l'utilisation de l'information des dérivées de la fonction objectif par rapport aux paramètres x_l. Elles sont dites directes ou d'ordre 0 si elles n'utilisent que l'information de la valeur de la fonction elle-même. Dans le cas où elles nécessitent aussi le calcul du gradient de la fonction, elles sont dites indirectes ou d'ordre 1.

Les méthodes d'ordre 0 offrent l'avantage de se passer du calcul du gradient, ce qui peut être intéressant lorsque la fonction n'est pas différentiable ou lorsque le calcul de son gradient représente un coût important.

Les méthodes multidimensionnelles, peuvent être groupées en deux groupes: les méthodes analytiques ou de descente et les méthodes heuristiques ou géométriques. Les méthodes analytiques se basent sur la connaissance d'une direction de recherche souvent donnée par le gradient de la fonction. Les exemples les plus significatifs de méthodes analytiques sont la méthode de la Plus Grande Pente [Culioli, 1994], le Gradient Conjugué [Culioli, 1994] [Fletcher, 1987] [Press, 1992], la méthode de Powell [Powell, 1965] et les méthodes Quasi-Newton [Culioli, 1994] [Fletcher, 1987] [Press, 1992].

Les méthodes heuristiques explorent l'espace par essais successifs en recherchant les directions les plus favorables. À l'opposé des méthodes analytiques, la plupart de ces méthodes sont d'ordre 0. Les implémentations de méthodes géométriques les plus souvent utilisées sont celles de la méthode du Simplexe [Nelder, 1965], la méthode de Rosenbrock [Rao, 1996] et la méthode de variations locales de Hooke et Jeeves [Cherruault, 1999].

Le deuxième groupe qui englobe les méthodes d'optimisation stochastiques s'appuie sur des mécanismes de transition probabilistes et aléatoires. Cette caractéristique indique que plusieurs exécutions successives de ces méthodes peuvent conduire à des résultats différents pour une même configuration initiale d'un problème d'optimisation.

Malgré l'aspect du hasard, ces méthodes ont une grande capacité de trouver l'optimum global du problème, contrairement à la plupart des méthodes déterministes. Cependant, elles demandent un nombre important d'évaluations avant d'arriver à la solution du problème.

Parmi les méthodes stochastiques les plus employées, nous distinguons le Recuit Simulé [Kirkpatrick, 1983], la Recherche Tabou [Glover, 1989] [Glover, 1990] [Hu, 1992] et les Méthodes Évolutionnistes tel que les Algorithmes Génétiques [Holland, 1975], [Goldberg, 1989].

La figure 3-2 présente un organigramme de classification des méthodes d'optimisation les plus utilisées (stochastiques et déterministes). Cette classification n'est pas exhaustive, mais regroupe les principales stratégies d'optimisations employées à l'heure actuelle. Dans ce travail, on propose le calcul de la loi de commande par deux méthodes qui n'utilisent pas la dérivée. La première est basée sur l'algorithme du Simplexe (Nelder-Mead), et la deuxième exploite l'algorithme de Rosenbrock. Ces méthodes sont destinées à des problèmes d'optimisation sans contraintes. Dans nos travaux, nous avons associés à ces méthodes des éléments afin de traiter des fonctions non linéaires, non convexes et en présence des contraintes.

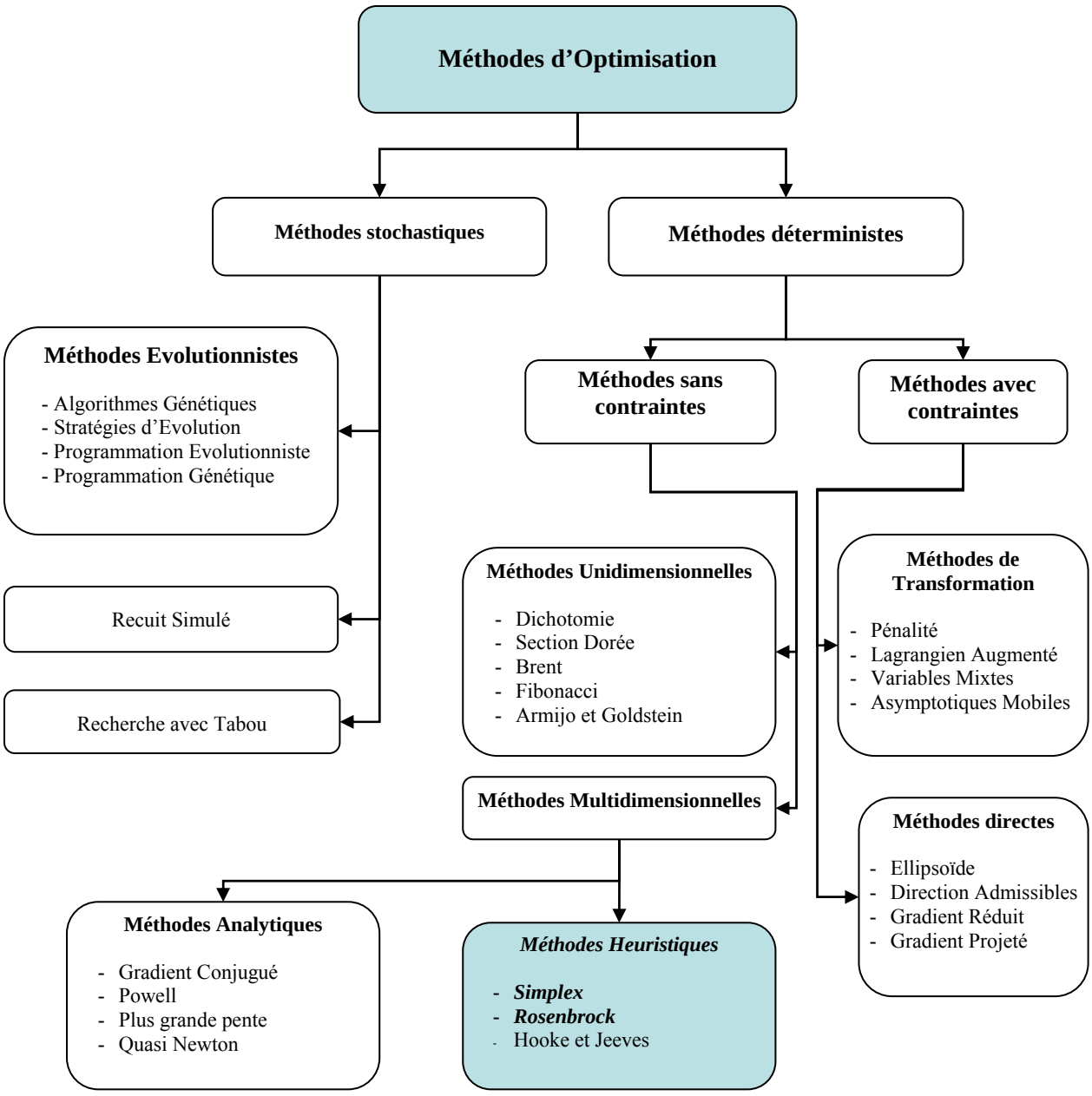


Figure 3-2 : Les méthodes d'optimisation les plus importantes

3.4.2. La méthode de Nelder-Mead

L'algorithme de Nelder-Mead, appelé encore algorithme du simplexe, permet de déterminer le minimum d'une fonction sans calcul de la dérivée. Dans notre cas, on a exploité cette propriété pour minimiser le critère de performance $J_2(k)$ donné par la relation (3-2).

Avant de présenter le principe de l'algorithme de Nelder-Mead, il est nécessaire de donner quelques définitions.

3.4.2.1. Définitions

- Un polytope, un polyèdre : est une figure géométrique de $(n+1)$ points dans un espace à n dimensions.
- Un simplexe : on appelle simplexe de $\Re^n$ tout ensemble de points $(x_1, x_2, ..., x_{n+1})$ de $\Re$ tel que $f(x_1) \leq f(x_i) \quad \forall\, i \in [2, \cdots, n+1]$.
- Une réflexion : consiste à échanger le point de plus grande valeur avec son symétrique par rapport au barycentre des n points restants.
- Une contraction : consiste à échanger le point de plus grande valeur avec son image par l'homothétie de centre (le barycentre) des n points restants et d'un facteur $\gamma < 1$. On distingue deux cas :
  - soit une contraction externe, dans ce cas $\gamma = 0.5$.
  - soit une contraction interne, $\gamma = -0.5$.
- Une expansion : consiste à échanger le point de plus grande valeur avec son image par l'homothétie de centre (le barycentre) des n points restants et d'un rapport χ supérieur à 1 (généralement $\chi = 2$).
- Le rétrécissement ou « multi-contraction » : consiste à échanger tous les points $x_i \;\; (i = 2, .., n+1)$ du polytope par (n) nouveaux points en utilisant la relation suivante :

$$x_i = \frac{(x_i + x_1)}{2} \quad , \quad (\mathrm{i} = 2, .., \mathrm{n}+1)$$

3.4.2.2. Méthode du polytope de Nelder-Mead

C'est une méthode d'optimisation locale [Nelder, 1965] [Wright, 1996] qui est fréquemment utilisée. Cette méthode déterministe est dite "directe" : elle tente de résoudre le problème en utilisant directement la valeur de la fonction objectif, sans faire appel à ses dérivées. Cette méthode est surtout appréciée pour sa robustesse, sa simplicité de programmation, sa faible consommation de mémoire (peu de variables) et son faible temps de calcul. Comparé à d'autres méthodes de recherche directes, la méthode NM est relativement rapide. Le temps de calcul peut être amélioré suivant l'initialisation du simplexe. Une fois le nombre de points du simplexe est fixé, leurs valeurs influent la vitesse de l'algorithme. Dans [Yang, 2005] les auteurs ont exploité les Algorithmes Génétiques et la méthode NM pour l'optimisation globale, la population initiale de l'Algorithme Génétique devrait éviter le risque d'avoir un grand nombre d'individus dans la même région. D'autre part, puisque la méthode NM n'est pas une méthode d'optimisation globale, l'utilisation des points de départs multiples (multi initialisation) est avantageux et aide à vérifier qu'une valeur fortement optimale sera trouvée [Bortolot, 2005]. Deux autres propriétés de l'algorithme NM sont développées par M. A. Luersen et R. le Riche [Luersen, 2004]. Premièrement, l'algorithme NM peut converger à un optimum local, ce qui arrive en particulier quand le simplexe s'effondre dans un sous-espace. Deuxièmement, la méthode peut échapper à une région qui serait un bassin d'attraction si le simplexe est assez large. D'autre part, si la taille du simplexe est petite, l'algorithme devient local.

Cependant, cet algorithme est robuste car il est très tolérant aux bruits dans les valeurs de la fonction objectif. En conséquence, la fonction n'a pas besoin d'être calculée exactement et il est possible d'avoir recours à une approximation de la valeur de la fonction. Contrairement aux autres méthodes qui démarrent à partir d'un point initial, la méthode de Nelder-Mead utilise un "polytope" de départ.

3.4.2.3. Principe de la méthode de Nelder-Mead

L'algorithme de Nelder-Mead est basé sur l'évaluation de la fonction (le critère J_2) aux sommets du simplexe, sans calcul de dérivées [Kelley, 1999] [Wright, 1996]. Il est utilisé, dans notre cas, pour estimer itérativement le minimum du critère de performance donné par l'équation (3-2).

Pour cela on initialise un polyèdre sur une zone aléatoire, ce polyèdre de départ sera donc obtenu par le tirage aléatoire d'un point x_1 dans l'espace solution. Les autres points $x_i \ \ (i = 2, .., n+1)$ sont choisis de manière à former une base, qui est généralement orthogonale.

Dans chaque itération de l'algorithme de Nelder-Mead, un simplexe formé de (n+1) points est utilisé pour déterminer un pas d'essai. Chaque point du simplexe correspond à une valeur de la fonction. Les points $x_i \ \ (i = 1, \cdots, n+1)$ sont ordonnés, au début de l'itération, de manière à avoir:

$$f(x_1) \leq f(x_2) \ \leq \ \ldots \ \leq f(x_{n+1}). \tag{3-5}$$

Après avoir calculer un ou plusieurs points de test et évaluer la valeur de la fonction, de ces points, l'itération actuelle dégénère un ensemble de n+1 sommets qui définis un simplexe différent pour la prochaine itération.

Les points de test sont obtenus moyennant quatre opérations possibles, à savoir : réflexion, expansion, contraction et rétrécissement. Chaque opération est associée à un paramètre. Les coefficients associés aux opérations de réflexion, expansion, contraction et rétrécissement sont respectivement notés par ρ, χ, γ et σ. Ces coefficients doivent remplir les conditions suivantes [Kelley, 1999] :

$$\rho > 0, \ \chi > 1, \ 0 < \gamma < 1, \text{ et } \ 0 < \sigma < 1. \tag{3-6}$$

D'après C.T. Kelley [Kelley, 1999] et M.H. Wright [Wright, 1996], La séquence standard associée aux coefficients $\{\rho, \chi, \gamma, \sigma\}$ est $\left\{1, \ 2, \ \frac{1}{2}, \ \frac{1}{2}\right\}$ (3-7)

Dans chaque itération de l'algorithme de Nelder-Mead deux transformations sont possibles :

un seul point du simplexe actuel sera modifié. Le nouveau point de test accepté par l'algorithme remplace le plus mauvais point du simplexe actuel (x_{n+1}), et prépare le nouveau simplexe pour l'itération suivante,

ou bien un ensemble de *n* nouveaux points sera calculé et remplace les *n* actuels points $x_i \quad (i = 2,\cdots, n+1)$ afin de former un nouveau simplexe pour l'itération suivante. Il s'agit de l'opération de rétrécissement.

L'algorithme s'arrête lorsque la différence $(f(x_{n+1}) - f(x_1))$, qui présente la distance entre les valeurs de la fonction du meilleur et du plus mauvais point, soit inférieure à un certain seuil de précision ε_0.

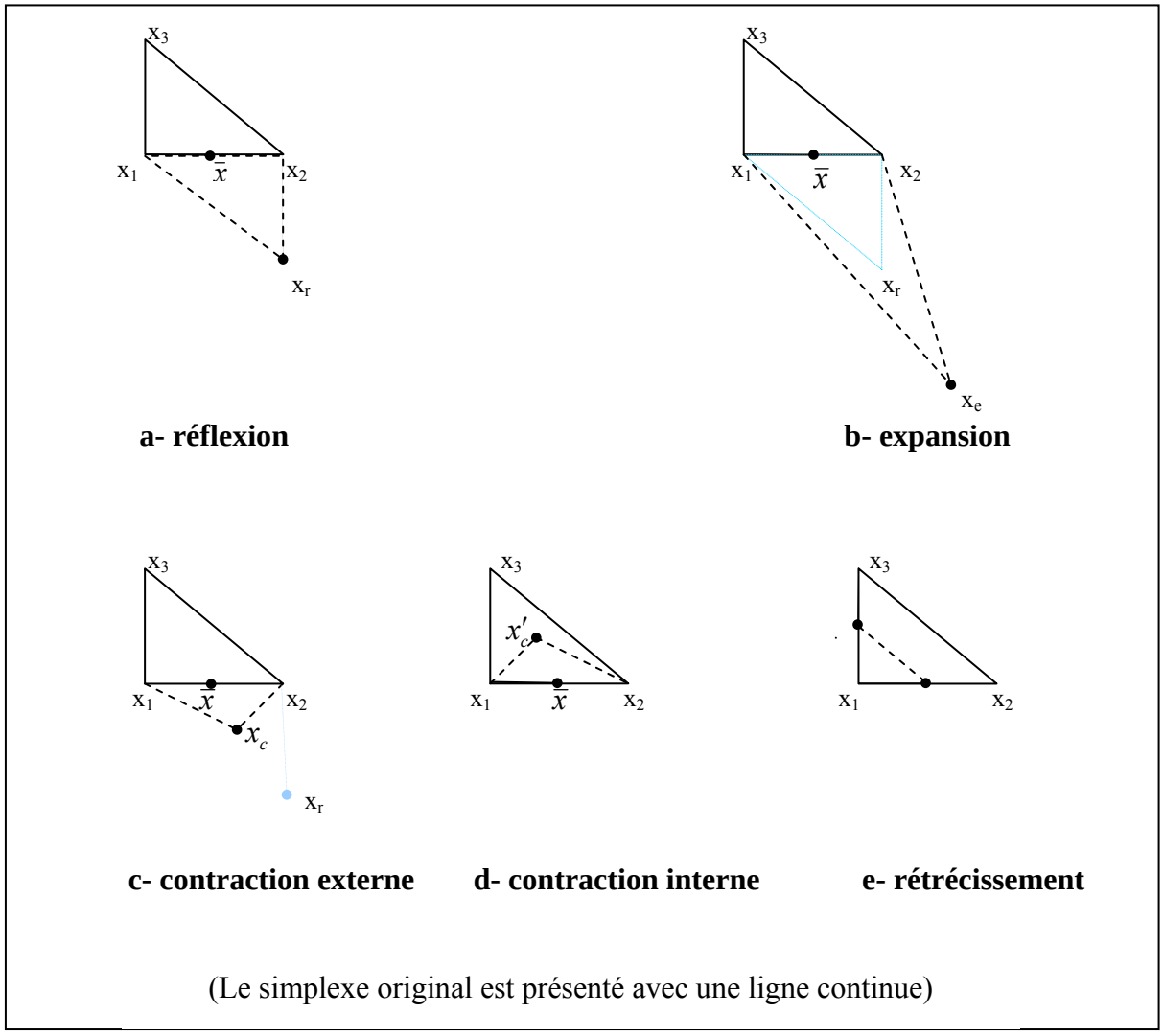


Figure 3-3 : Les mouvements possibles dans la méthode du polytope de Nelder-Mead dans $\Re^2$

La figure 3-3 montre l'effet des opérations de réflexion, de l'expansion, de la contraction (interne et externe) et du rétrécissement pour un simplexe à deux dimensions, soit un triangle (x_1, x_2, x_3) en utilisant la séquence standard des coefficients $\left\{\rho = 1,\ \chi = 2,\ \gamma = \frac{1}{2},\ \text{et}\ \sigma = \frac{1}{2}\right\}$.

3.4.2.4. L'algorithme de Nelder-Mead

L'algorithme de Nelder-Mead, comme décrit par C. T. Kelley [Kelley, 1999] et M. H. Wright [Wright, 1998], [Wright, 1996], est résumé par les étapes suivantes :

Etape 1 :

Prendre les sommets $(x_1, x_2, ..., x_{n+1})$ d'un polyèdre de départ vérifiant

$$f(x_1) \le f(x_2) \le \ldots \le f(x_{n+1}).$$

Etape 2 :

Tant que la différence entre $f(x_{n+1})$ et $f(x_1)$ est supérieure à un certain seuil de précision ε_0.

2.1 calculer $\bar{x}$ et $f_r = f(x(\rho))$

avec $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, est le centroïde des n meilleurs sommets.

et $x(\rho) = \bar{x} + \rho \cdot (\bar{x} - x_{n+1})$

2.2 *réflexion :*

si $f(x_1) \le f_r < f(x_n)$.

Alors $x_{n+1} = x(\rho)$, aller à 2.7.

2.3 *expansion :*

si $f_r < f(x_1)$ alors calculer $f_e = f(x(\chi))$, avec $x(\chi) = \bar{x} + \chi \cdot (x_r - \bar{x})$

si $f_e < f_r$ alors $x_{n+1} = x(\chi)$ et aller à 2.7

sinon $x_{n+1} = x(\rho)$ et aller à 2.7.

2.4 *contraction externe :*

si $f(x_n) \le f_r < f(x_{n+1})$ alors calculer $f_o = f(x(\gamma))$,avec $x(\gamma) = \bar{x} + \gamma \cdot (x_r - \bar{x})$

si $f_o \le f_r$ alors $x_{n+1} = x(\gamma)$ puis aller à 2.7

sinon aller à 2.6

2.5 *contraction interne*

si $f_r \ge f(x_{n+1})$ alors calculer $f_i = f(x(\gamma))$, avec $x(\gamma) = \bar{x} - \gamma \cdot (\bar{x} - x_{n+1})$

si $f_i < f(x_{n+1})$ alors $x_{n+1} = x(\gamma)$ puis aller à 2.7

sinon aller à 2.6.

2.6 *rétrécissement*

pour $i = 2, \ldots, n+1$, calculer $x_i = \frac{(x_i + x_1)}{2}$, puis calculer $f(x_i)$

2.7 *donner les sommets* $(x_1, x_2, ..., x_{n+1})$ *tel que* $f(x_1) \le f(x_2) \le \ldots \le f(x_{n+1})$.

retour à l'étape 2.

3.4.3. La méthode de Rosenbrock

Cette méthode d'optimisation a été publiée par Rosenbrock en 1960 [Rosenbrock, 1960], l'approche utilisée met en œuvre plusieurs stratégies d'optimisation différentes. Elle a été capable de trouver de bons compromis même avec des objectifs d'optimisation difficiles à atteindre.

3.4.3.1. Présentation de l'algorithme de Rosenbrock

Dans le passé cet algorithme a déjà été utilisé avec succès par Rosenbrock pour l'optimisation de fonctions cibles jusqu'à 50 paramètres et plus (pour des filtres digitaux et analogiques ou la conception de circuit électrique). La stratégie est basée sur un algorithme de recherche très stable d'ordre 0 qui n'exige pas de dériver la fonction cible, bien qu'il s'approche de la méthode du gradient. Donc il combine les avantages des stratégies d'ordre 0 et 1.

Il s'est avéré que cette approche simple est plus stable que beaucoup d'algorithmes sophistiqués et exige beaucoup moins de calculs de la fonction cible que des stratégies plus élaborées. À cause de cette stabilité inhérente et avec de bonnes heuristiques pour le calcul des longueurs de pas, cet algorithme est même approprié et a déjà prouvé sa valeur pour des problèmes d'optimisation impliquant des fonctions objectif fortement non linéaires et non monotones.

La méthode de Rosenbrock (dite aussi ‘’rotation de coordonnées’’) permet de rendre l'exploration indépendante des directions initiales. C'est une méthode itérative qui se décompose en deux étapes répétitives. Une étape d'exploration qui réalise des améliorations successives via des directions privilégiées, et une étape globale qui permet de construire une nouvelle base dans la direction de deux sorties consécutives. La vitesse de convergence de la méthode dépend du choix de la base de départ. De plus, elle nécessite la reconstruction d'une nouvelle base à chaque itération ce qui rend la méthode coûteuse lorsque la dimension du vecteur des variables de décision est grande.

Cependant, la simplicité de cette méthode offre une chance réelle à un utilisateur qui n'est pas un expert en optimisation de le comprendre et de fixer et ajuster ses paramètres correctement. Des procédures de contrôle heuristiques ont été mises en œuvre pour ajuster les

paramètres d'optimisation en cours d'exécution dès que l'optimisation ralentit ou se bloque. Cela permet de réussir des optimisations sans aucune intervention d'utilisateur.

3.4.3.2. Principe de fonctionnement

La première itération de cet algorithme est une simple recherche d'ordre 0 dans les directions des vecteurs de base d'un système de coordonnées à *n* dimensions. En cas de succès, un essai fournissant une nouvelle valeur minimale de la fonction cible, la longueur de pas est augmentée, tandis que dans le cas d'un échec elle est diminuée et on essaye la direction opposée.

Une fois qu'un succès a été trouvé et exploité dans chaque direction de la base, le système de coordonnées est réorienté pour que le premier vecteur de base soit dans la direction du gradient. Toutes les longueurs de pas sont réinitialisées et le processus est répété en utilisant le nouveau système de coordonnée. En initialisant les longueurs de pas à des valeurs plutôt grandes, on permet à la stratégie de laisser de côté des optimums locaux et de continuer la recherche de minimums plus globaux.

Dans chaque itération de l'algorithme, une exploration itérative dans les *n* directions de recherche est effectuée. *n* étant le nombre de variables et les directions de recherche sont orthogonales et linéairement indépendantes. Quand une nouvelle solution est localisée à la fin de l'itération, un nouvel ensemble de directions de recherche $d_1,\cdots,d_n$ sera construit. La procédure est répétée jusqu'à ce que la valeur de la fonction objectif ne change que de moins d'une valeur donnée (précision).

Construction des directions de recherche :

Soient $d_1,...,d_n$, des vecteurs linéairement indépendants et dont leurs normes sont égaux à 1. En outre, supposant que les vecteurs sont mutuellement orthogonaux, c. à d., $d_i^t\ d_j = 0$, pour $i \neq j$. En commençant par le vecteur actuel x_k, la fonction objectif est minimisée itérativement dans toutes les directions de recherche. La solution obtenue sera notée x_{k+1}. En particulier, $x_{k+1} - x_k = \sum_{j=1}^{n} \lambda_j\ d_j$, ou λ_j est la distance déplacée le long de d_j.

La nouvelle direction de recherche $d_1,...,d_n$ est ainsi formée par la procédure de Gram-Schmidt (Gram-Schmidt orthogonalization), comme suit :

$$\begin{cases} a_j = \begin{cases} d_j & \text{si } \lambda_j = 0 \\ \sum_{i=j}^{n} \lambda_i d_i & \text{si } \lambda_j \neq 0 \end{cases} \\ b_j = \begin{cases} a_j & \text{si } j = 1 \\ a_j - \sum_{i=1}^{j-1} \left(a_j^t \bar{d}_i\right)\bar{d}_i & \text{pour } j \geq 2 \end{cases} \\ \bar{d}_j = \dfrac{b_j}{\|b_j\|} \end{cases} \tag{3-8}$$

$$d_j = \bar{d}_j ;\ j = 1,..., n$$

Une particularité intéressante de cette méthode est qu'un minimum local est toujours encadré dès qu'un succès est suivi d'un échec. Quand cela arrive, le point du milieu des 3 derniers points est toujours plus bas que les 2 autres, ainsi on est dans une position favorable pour essayer une interpolation quadratique. C'est peut être la méthode la plus efficace à utiliser dans le cas d'une fonction unidimensionnelle générale.

3.4.3.3. L'algorithme de Rosenbrock

L'algorithme de Rosenbrock est résumé dans les étapes suivantes [Bazaraa, 1993] :

Étape d'initialisation

Soient $\varepsilon > 0$, un scalaire positif non nul (critère d'arrêt), $d_1,...,d_n$ les directions de recherche et *n* le nombre de variables.

Choisir un point initial x_1, $y_1 = x_1$, Prendre $k=j=1$; $x_1 \in \Re^n$ et passer à l'étape principale.

Étape principale

1. On cherche λ_j la solution du problème suivant :

   min ($f(y_j + \lambda d_j)$)

   $y_{j+1} = y_j + \lambda_j d_j$

   si $j < n$, $j = j+1$, retour à l'étape 1.

   Sinon aller à l'étape 2.

2. $x_{k+1} = y_{n+1}$,

   si $\|x_{k+1} - x_k\| < \varepsilon$ arrêter ;

   sinon $y_1 = x_{k+1}$, $k=k+1$, $j=1$, aller à l'étape 3 ;

3. Construire une nouvelle direction de recherche, $(d_1,...,d_n)$ et retour à l'étape 1.

Dans l'étape d'initialisation, les vecteurs directions de recherche $(d_1,...,d_n)$ sont orthogonaux, linéairement indépendants et de norme égale à 1. Dans le cas où le nombre de variables n est supérieur ou égale à 2, les vecteurs sont initialisés de manière qu'ils forment une matrice identité de dimension n.

$$\left[d_1\ d_2\ ...\ d_n\right]^T = I_n \tag{3-9}$$

Dans le cas monovariable $(n=1)$, on aura un seul vecteur, donc une seule direction de recherche ; soit $d_1 = \begin{bmatrix}1 & 0\end{bmatrix}$. Dans l'étape 3, les directions de recherche sont calculées suivant l'équation (3-8).

3.4.4. Améliorations des performances des méthodes d'optimisation

Bien que les méthodes d'optimisation déterministes présentées précédemment possèdent des avantages, tel que la faible consommation du temps de calcul, la robustesse et elles ne nécessitent pas le calcul du gradient. Ces méthodes ne sont pas directement adaptables à notre problème. En effet, puisque ces méthodes sont locales (la solution n'est pas globale), elles dépendent de l'initialisation et ne tiennent pas compte des contraintes. D'autre part, les systèmes réels peuvent avoir des contraintes d'ordre pratique ou des limitations physiques qu'on doit prendre en considération.

Pour exploiter ces méthodes dans le calcul de la loi de commande, on propose l'introduction de quatre améliorations aux algorithmes d'optimisation de Nelder-Mead et de Rosenbrock. Ces améliorations sont détaillées comme suit :

1) Application de la méthode de pénalité pour que la méthode d'optimisation traite les contraintes.
2) Application de l'approche CFON (Constrained First Objective Next) [Kurpati, 2002] dans l'initialisation du (des) point(s) de départ(s), en considérant un intervalle qui respecte les contraintes. Ce qui permet de réduire le temps de calcul.
3) Application de la notion de multi initialisation, pour avoir plus de probabilité de convergence vers un minimum global.
4) Application de la notion d'élitisme qui consiste à exploiter la dernière solution trouvée.

Application de la méthode de pénalité

Les méthodes d'optimisation proposées (Simplexe et Rosenbrock) sont capables de traiter des problèmes non linéaires. Cependant, ces méthodes ne sont pas conçues pour résoudre des problèmes avec des contraintes. L'emploi d'une fonction de pénalité exacte, avec les méthodes d'optimisation proposées, nous permet d'atteindre cet objectif en transformant un problème non contraint en un sous contraintes [Gesing, 1979] [Wright, 1998].

Dans ce cas, les contraintes sont placées dans la fonction objectif par l'intermédiaire d'un paramètre de pénalité de manière qu'il pénalise chaque violation des contraintes. En général, une fonction de pénalité appropriée doit avoir une pénalité positive pour les points infaisables et aucune pénalité pour les points faisables [Bazaraa, 1993]. Ainsi, une fonction de pénalité est généralement traitée comme suit :

Si on considère le problème d'optimisation de l'équation (3-4) dans $\Re$, alors la fonction de pénalité sera de la forme suivante :

$$\alpha(x) = \sum_{i=1}^{p} \phi\left[g_i(x)\right] + \sum_{i=1}^{q} \psi\left[h_i(x)\right] \tag{3-10}$$

avec ϕ et ψ sont des fonctions continues qui vérifient les conditions suivantes :

$$\begin{aligned} &\phi(y) = 0 \quad si\ y \le 0 \quad et \quad \phi(y) > 0 \quad si\ y > 0 \\ &\psi(y) = 0 \quad si\ y = 0 \quad et \quad \psi(y) > 0 \quad si\ y \ne 0 \end{aligned} \tag{3-11}$$

ϕ et ψ peuvent s'écrire comme suit :

$$\phi(y) = \left[Max\{0, y\}\right]^n \quad \text{et} \quad \psi(y) = |y|^n \tag{3-12}$$

ou *n* est un entier positif (généralement *n*=2). La forme de la fonction de pénalité sera alors :

$$\alpha\ (x)\ =\ \sum_{i=1}^{p}\ \left[Max\quad \{0\ ,\ g_{\,i}\ (x)\ \}\right]^n\ +\ \sum_{i=1}^{q}\ |h_{\,i}\ (x)\ |^n \tag{3-13}$$

Le problème avec contraintes sera remplacé par un problème sans contraintes, en considérant la fonction auxiliaire $f\,(x) + \mu\ \alpha\,(x)\ ,\quad (\mu\ >\ 0)$.

Dans notre cas, la minimisation du critère J_2 est faite en respectant les contraintes sur le signal de commande et sur ses incréments. Ces contraintes peuvent être écrites comme suit :

$$u_{\,\min}\ \le\ u\,(k\ +\ j)\ \le\ u_{\,\max}\ ,\qquad j\ =\ 0\ ,\ \ldots\ ,\ Hc\ -\ 1 \tag{3-14}$$

$$\Delta\,u_{\,\min}\ \le\ \Delta\,u\,(k\ +\ j)\ \le\ \Delta\,u_{\,\max}\ ,\qquad j\ =\ 0\ ,\ \ldots\ ,\ Hc\ -\ 1 \tag{3-15}$$

Le nombre de contraintes est égal à 4 *Hc*.
Ces contraintes peuvent être réécrites de la manière suivante :

$$\Omega\ =\ \left\{\Delta U\ /\ f_{\,i}\,(\Delta\,U\,)\ \le\ 0,\quad i\ =\ 1,\ldots,4\ Hc\ \right\} \tag{3-16}$$

Le but est de concevoir, à chaque période d'échantillonnage, une loi de commande qui minimise le critère de performance J_2 (équation 3-2) en tenant compte des contraintes sur la commande et sur ses incréments (équation 3-16).

Donc la fonction de pénalité sera comme suit :

$$\alpha\,(x)\ =\ \sum_{i=1}^{4\,Hc}\ \left[Max\ \{0,\ f_{\,i}\,(x)\}\right]^2 \tag{3-17}$$

Algorithme de la méthode de pénalité

Un sommaire de l'algorithme de la méthode de pénalité est donné comme suit [Bazaraa, 1993] :

Étape d'initialisation
Soit $\varepsilon > 0$, un scalaire positif non nul (critère d'arrêt). Choisir un point initial x_1, un paramètre de pénalité $\nu_1 > 0$ et une grandeur scalaire $\beta > 0$. Prendre $k=1$ et passer à l'étape principale.

Étape principale

1. On cherche x_k, solution du problème suivant :

   min ($f(x) + \nu_k \alpha(x)$) tel que : $x \in X$.

   Prendre x_{k+1} une solution optimale et passer à l'étape 2.
2. Si $\nu_k \alpha(x) < \varepsilon$, arrêter ;
   sinon, prendre $\nu_{k+1} = \beta \nu_k$, remplacer k par $k+1$ et retour à l'étape 1.

Application de l'approche CFON

L'approche CFON consiste à prendre le(s) point(s) d'initialisation de manière à satisfaire les contraintes dès le départ. Cette approche est employée dans [Kurpati, 2002], pour améliorer la manipulation des contraintes en utilisant les algorithmes génétiques multi objectif. Elle permet de réduire le temps de calcul, puisque tous les points d'initialisation respectent toutes les contraintes dès le début. Ainsi, l'emploi d'une fonction de pénalité avec cette approche évite l'apparition des solutions non faisables dans les boucles intérieures de l'algorithme d'optimisation.

Multi-initialisation des algorithmes d'optimisation

Cette étape, intégrée dans les algorithmes d'optimisation, permet d'effectuer l'optimisation en partant de différents points d'initialisation. Tous les points de départ respectent l'approche CFON. Après un certain nombre de répétitivité de l'optimisation, on garde le meilleur optimum. Cette étape permet d'augmenter la probabilité de convergence vers le minimum globale.

Pour vérifier l'influence de cette approche, on a appliqué trois versions d'initialisation sur des fonctions benchmarks avec des minimums globaux connus. Les fonctions benchmarks (F1, F2, F3 et F4) sont données dans l'annexe 3 [Chelouah, 2000]. La première version proposée consiste à prendre une seule initialisation dans l'intervalle de recherche *S* (figure 3-4), la deuxième consiste à prendre plusieurs initialisations aléatoires dans le même intervalle *S*

(figure 3-4) et la troisième consiste à diviser l'intervalle *S* (espace de recherche) en plusieurs sous intervalles égaux (figure 3-5). La résolution du problème d'optimisation s'effectue chaque fois avec une initialisation appartenant à un sous intervalle, de façon à balayer tout l'intervalle de recherche *S*. A la fin, on garde la meilleure solution parmi les solutions partielles.

Dans les figures 3-4 et 3-5, on considère un problème d'optimisation simple, avec deux contraintes c_1 et c_2. Dans ce cas l'intervalle de recherche *S* sera donc compris entre c_1 et c_2.

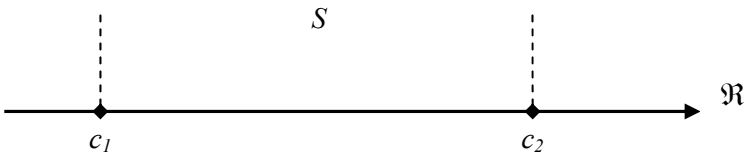


Figure 3-4 : Initialisation des algorithmes d'optimisation versions (NM-V_1 et NM-V_2 ou ROS-V_1 et ROS-V_2)

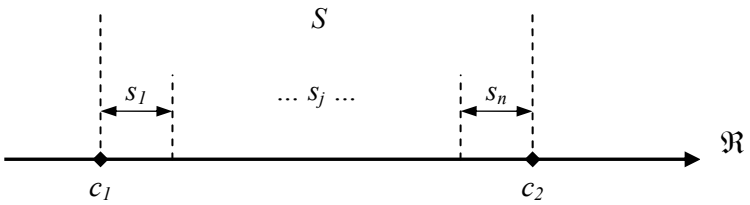


Figure 3-5 : Initialisation des algorithmes d'optimisation versions (NM-V_3 ou ROS-V_3)

Afin d'évaluer l'efficacité de ces méthodes, nous avons considéré des critères récapitulant les résultats de 100 minimisations par fonction. Les critères maintenus sont : le taux de minimisations réussies (%), la moyenne du temps de calcul dans chaque méthode d'optimisation et la moyenne de l'erreur quadratique (variance) entre la solution trouvée par l'optimisation et l'optimum global connu. L'optimisation est considérée réussie si l'erreur quadratique entre la solution obtenue et le minimum global est moins de $5\ 10^{-4}$. Les tableaux 3-1, 3-2 et 3-3 présentent respectivement les résultats obtenus avec les versions originales des algorithmes de Nelder-Mead et de Rosenbrock (NM-V_0 et ROS-V_0), les trois versions améliorées (NM-V_1, NM-V_2 et NM-V_3) en exploitant l'algorithme de Nelder Mead ainsi que les trois versions (ROS-V_1, ROS-V_2 et ROS-V_3) qui exploitent l'algorithme de Rosenbrock.

Fonctions Benchmarks (tests)	Méthodes	taux de minimisations réussies (%)	temps de calcul moyen (s)	erreur quadratique moyenne
	NM-V_0	0	0.0395	136.4803
F_1	ROS-V_0	38	0.0039	68.6517
	NM-V_0	0	0.2819	5.8859
F_2	ROS-V_0	50	0.0044	1.7130
	NM-V_0	0	0.0402	0.3723
F_3	ROS-V_0	88	0.0052	0.2985
	NM-V_0	0	0.0683	0.0076
F_4	ROS-V_0	100	0.0250	$4.23\ 10^{-9}$

Tableau 3-1 : Résultats de l'optimisation avec versions originales (méthode NM et ROS)

Fonctions Benchmarks (tests)	Méthodes	taux de minimisations réussies (%)	temps de calcul moyen (s)	erreur quadratique moyenne
	NM-V_1	25	0.0547	0.0173
F_1	NM-V_2	82	2.7533	$6.252\ 10^{-4}$
	NM-V_3	100	4.7341	$1.479\ 10^{-5}$
	NM-V_1	8	0.0421	0.161
F_2	NM-V_2	94	3.0414	$1.596\ 10^{-5}$
	NM-V_3	100	3.8663	5.04110^{-7}
	NM-V_1	1	0.0020	0.148
F_3	NM-V_2	27	0.0681	0.007
	NM-V_3	87	0.0742	$2.68\ 10^{-5}$
	NM-V_1	100	0.0023	$4.388\ 10^{-4}$
F_4	NM-V_2	100	0.0253	$4.388\ 10^{-5}$
	NM-V_3	100	0.0489	$1.799\ 10^{-5}$

Tableau 3-2 : Résultats de l'optimisation avec réinitialisation (méthode NM)

Fonctions benchmarks	Méthodes	taux de minimisations réussies (%)	temps de calcul moyen (s)	erreur quadratique moyenne
F_1	ROS -V_1	44	0.0048	48.6253
	ROS -V_2	99	0.0025	1.3635
	ROS -V_3	100	0.0022	$1.0396\ 10^{-8}$
F_2	ROS -V_1	56	0.0042	1.5258
	ROS -V_2	100	0.0028	$3.0969\ 10^{-9}$
	ROS -V_3	100	0.0016	$3.4586\ 10^{-9}$
F_3	ROS -V_1	74	0.0391	0.3563
	ROS -V_2	88	0.0358	0.2335
	ROS -V_3	100	0.0363	0
F_4	ROS -V_1	100	0.0207	$4.9297\ 10^{-9}$
	ROS -V_2	100	0.0222	$4.6731\ 10^{-9}$
	ROS -V_3	100	0.0259	$4.0415\ 10^{-9}$

Tableau 3-3 : Résultats de l'optimisation avec réinitialisation (méthode ROS)

A partir des tableaux (3-1, 3-2 et 3-3), nous notons que la troisième version des algorithmes NM et ROS (NM-V_3 et ROS-V_3) présente les meilleurs taux de minimisation réussie, avec une moyenne de l'erreur quadratique la plus basse. On remarque aussi que le temps de calcul est relativement faible. Cependant, la méthode ROS est plus rapide que la méthode NM, ceci est dû au fait que la méthode ROS utilise un seul point de recherche, par contre la méthode NM utilise un ensemble de points qui forment un simplexe.

Contrairement à la méthode ROS, les améliorations effectuées sont nettement visibles sur la méthode NM. D'après le tracé de la fonction F3 (annexe 3), cette fonction présente le cas le plus complexe avec presque 50 minimums locaux. Les résultats trouvés dans les tableaux 3-2 et 3-3 prouvent que les deux méthodes NM et ROS combinées avec la version V_3 sont capables de trouver des bonnes performances, puisque l'erreur est très faible dans les deux cas.

3.5. La loi de commande :

L'algorithme qui permet la détermination de la loi de commande, en utilisant la méthode de Nelder-Mead, est résumé par les étapes suivantes :

1. Calcul de la sortie du système *y(k)*.
2. Calcul de la sortie $\hat{y}(k)$ à l'aide de l'équation du modèle.
3. Calcul de la valeur u_{opt}, qui permet la minimisation du critère de performance *J(k)*, en utilisant l'algorithme de Nelder-Mead ou bien la méthode de Rosenbrock.
4. Poser $u\ (k+i) = u_{opt}$, pour $i = 0,\ldots, Hp$.
5. Incrémenter *k*, retour à l'étape 1.

L'initialisation des algorithmes d'optimisation en exploitant l'approche CFON, impose que les valeurs initiales de la commande doivent respectées les contraintes données par les relations (3-14) et (3-15). La figure 3-6, décrit le principe général d'exploitation de cette approche pour le calcul de la loi de commande.

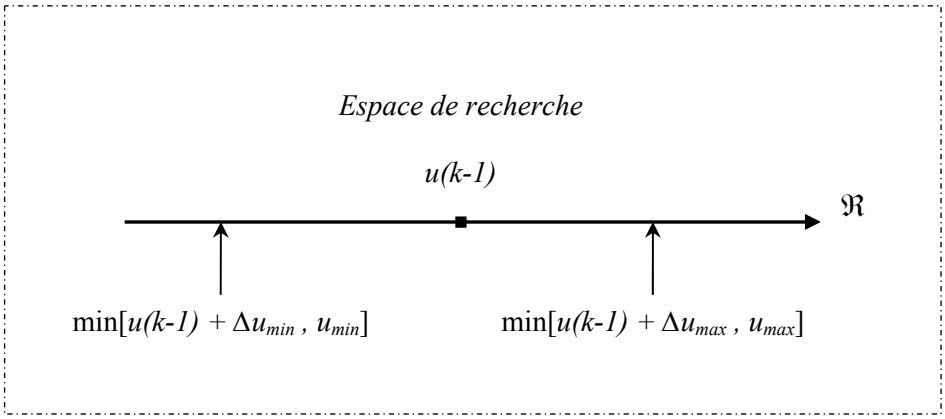


Figure 3-6 : Initialisation des algorithmes d'optimisation (approche CFON) dans ℜ

Dans le cas des versions NM-V_1 et ROS-V_1, on considère une seule initialisation des variables dans le domaine de recherche *S*. Dans le cas des versions NM-V_2 et ROS-V_2, on prend plusieurs initialisations différentes de domaine de recherche *S*. Cependant, dans le cas des algorithmes NM-V_3 et ROS-V_3, on divise l'espace de recherche *S* en plusieurs sous

intervalles puis on considère une initialisation par sous intervalle. Les figures 3-7 et 3-8 présentent, respectivement, l'espace de recherche *S* considéré avec les versions V_1, V_2 et V_3.

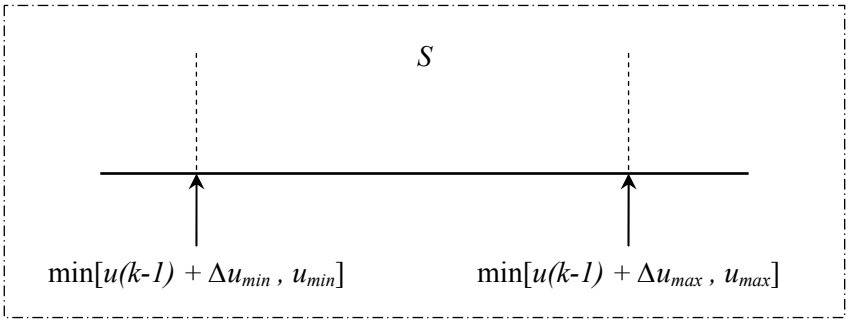


Figure 3-7 : Initialisation des algorithmes d'optimisation versions (NM-V_1 et NM-V_2 ou ROS-V_1 et ROS-V_2)

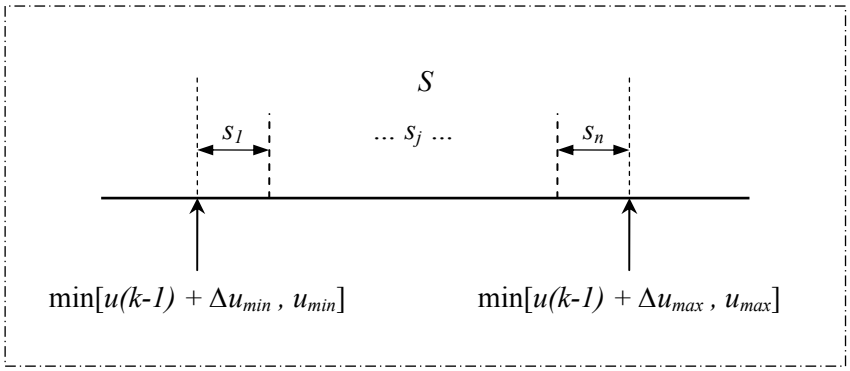


Figure 3-8 : Initialisation des algorithmes d'optimisation versions (NM-V_3 ou ROS-V_3)

3.6. Résultats de simulation :

Système 1:

Nous considérons le système non-linéaire décrit par l'équation suivante [Narendra, 1990]:

$$y(k)=\frac{y(k-1)}{1+y^2(k-1)}+u^2(k-1)+d(k) \tag{3-18}$$

Le signal de perturbation *d(k)* est défini comme suit :

$$d(k)=\begin{cases}1 & k\in[120,\ldots,150]\\ 0 & ailleurs\end{cases} \tag{3-19}$$

Pour (*m=n=q=2*), le modèle identifié en exploitant l'algorithme génétique binaire est comme suit :

$$\begin{aligned} ym(k) &= 0.2632 - 0.0121u(k-1) - 0.0165u(k-2) + 0.3559y(k-1) \\ &+ 0.9991u(k-1)^2 - 0.2316u(k-2)^2 + 0.0136u(k-1)\,y(k-1) - 0.0388y(k-1)^2 \end{aligned} \quad (3\text{-}20)$$

Les paramètres du correcteur prédictif considéré dans la simulation sont, $Hc = 1$, $Hp = 4$ et $\lambda = 0.2$. Les contraintes sur la commande et ses incréments sont comme suit :

$$0 \le u(k) \le 3 \quad (3\text{-}21)$$

$$-0.5 \le \Delta u(k) \le 0.5 \quad (3\text{-}22)$$

Dans ces simulations, l'intervalle *S* de l'espace de recherches est donné par :

$$S = \left[u(k-1) + \Delta u_{\min},\ u(k-1) + \Delta u_{\max} \right] \quad (3\text{-}23)$$

L'approche CFON est considérée. Si on considère la méthode NM (respectivement ROS), alors le simplexe (point) initial est ainsi choisi de manière à satisfaire les contraintes sur l'entrée et sur ses incréments. Nous avons considéré les trois versions (V_1, V_2 et V_3) d'optimisation basées sur l'algorithme de NM (ROS) et la fonction de pénalité. Dans la première méthode, un seul simplexe (point) initial est choisi dans l'intervalle de recherches S (NM-V_1, ROS-V_1). Dans la deuxième version (NM-V_2, ROS-V_2), une initialisation multiple et aléatoire de simplexes (points) est employée. Dans ce cas chaque simplexe (point) se trouve sur l'intervalle *S* du domaine de recherches. Dans la troisième version (NM-V_3, ROS-V_3), l'intervalle *S* est divisé en 10 sous intervalles égaux. La résolution du problème d'optimisation est effectué en employant l'algorithme NM (ROS) avec la fonction de pénalité dans chaque sous intervalle. Quand tous les sous intervalles sont considérés, on aura un ensemble de solutions partielles. Nous récupérons, ainsi, la meilleure solution parmi les solutions partielles.

Avant de donner les résultats obtenus par les trois versions des algorithmes NM et ROS, nous présentons les résultats de simulations obtenues avec la version originale de chacun des algorithmes NM et ROS (NM-V_0 et ROS-V_0). La figure 3-9 présente les résultats trouvés avec la méthode NM et la figure 3-10 présente les résultats trouvés avec la méthode ROS.

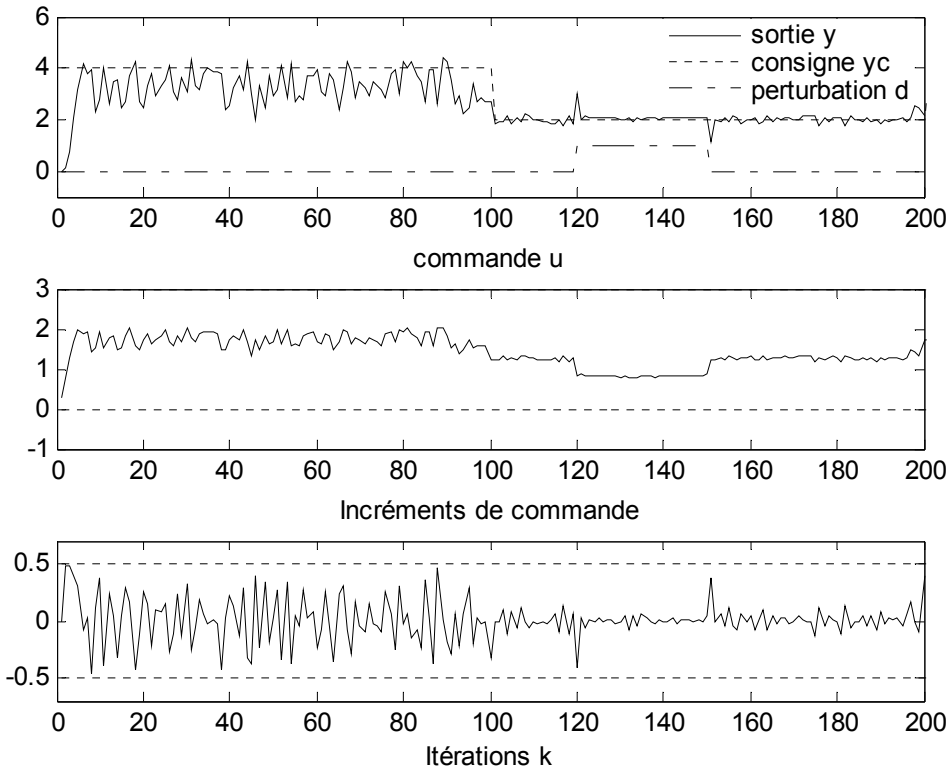


Figure 3-9 : Sortie du système, commande et incréments de la commande (NM-V_0)

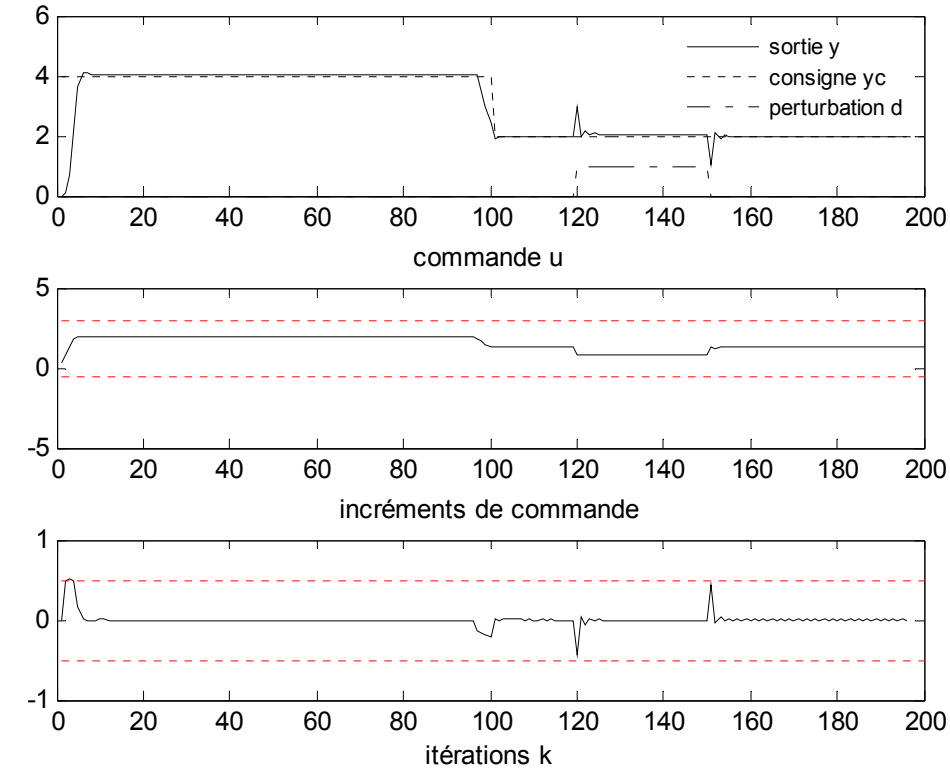


Figure 3-10 : Sortie du système, commande et incréments de la commande (ROS-V_0)

Les évolutions de la consigne *yc*, de la sortie *y*, de la commande u et de ses incréments Δu, obtenues par les trois versions des méthodes NM et ROS sont données dans les figures (3-11,…, 3-16).

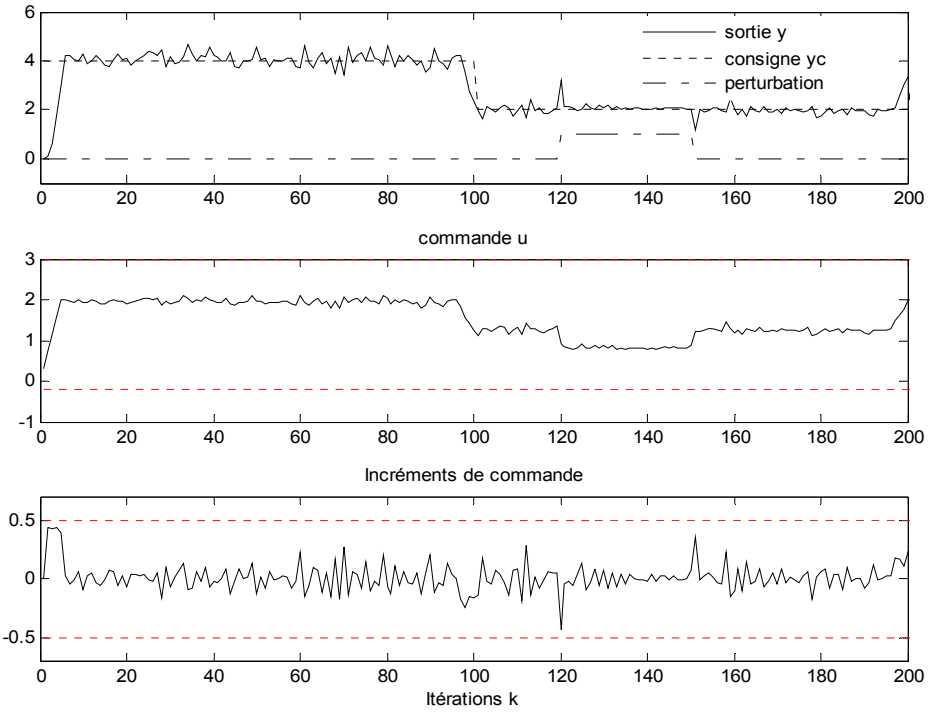


Figure 3-11 : Sortie du système, commande et incréments de la commande (NM-V_1)

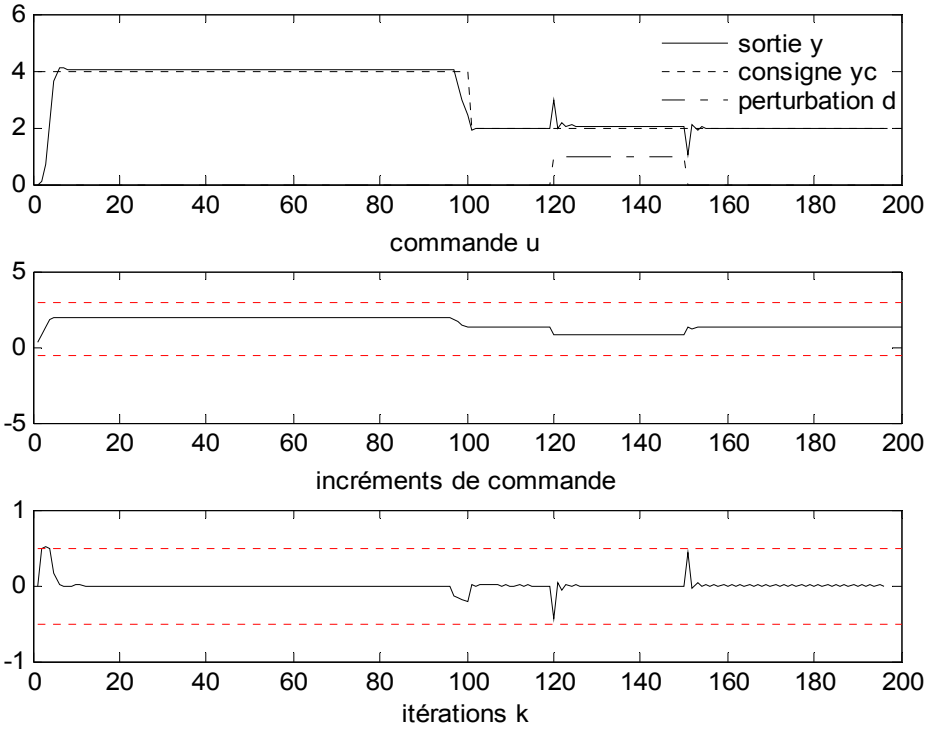


Figure 3-12 : Sortie du système, commande et incréments de la commande (ROS-V_1)

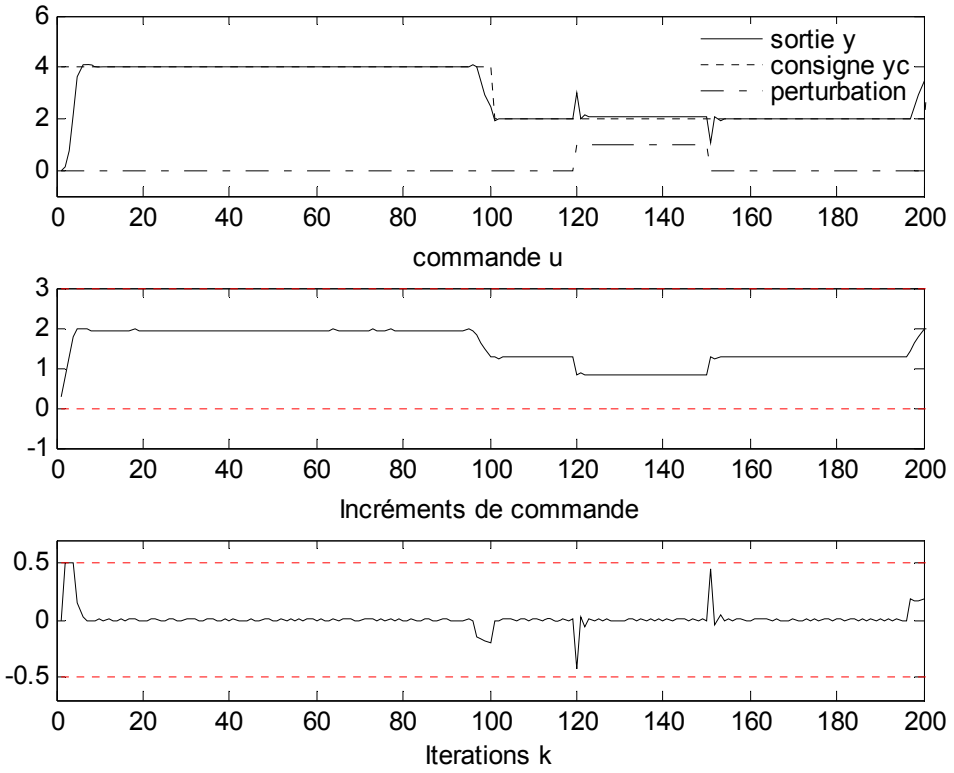


Figure 3-13 : Sortie du système, commande et incréments de la commande (NM-V_2)

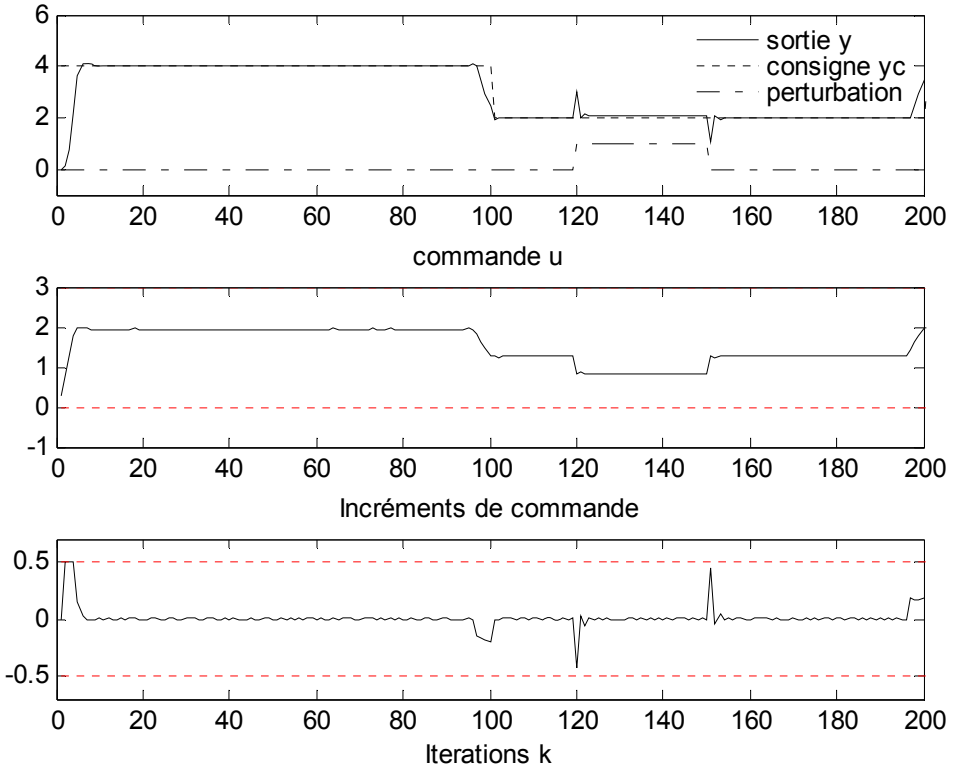


Figure 3-14 : Sortie du système, commande et incréments de la commande (ROS-V_2)

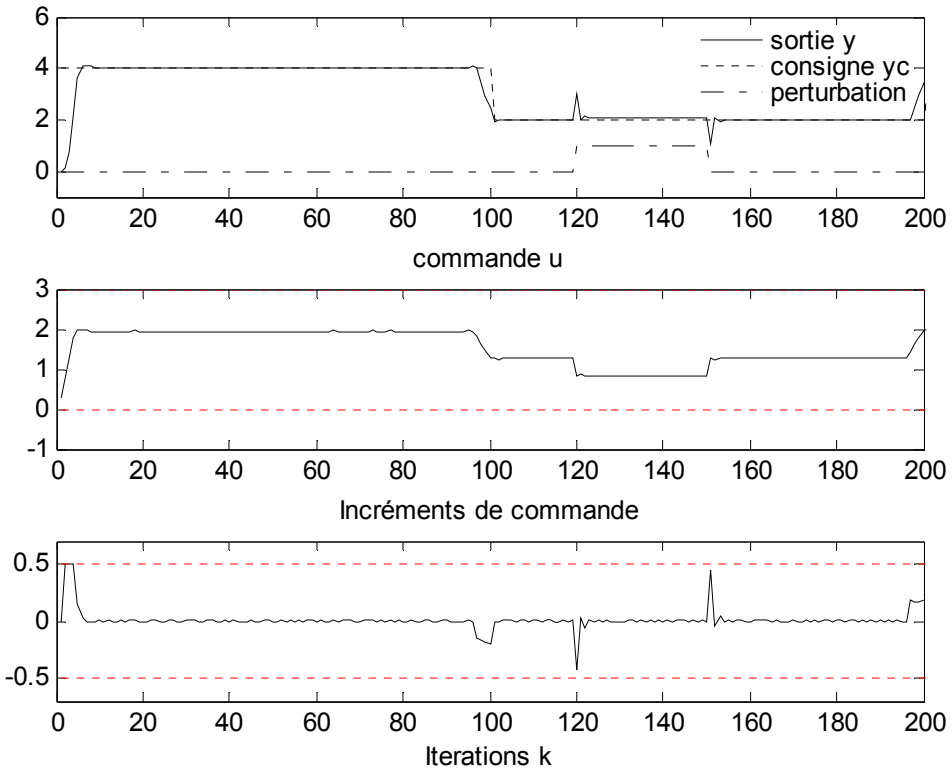


Figure 3-15 : Sortie du système, commande et incréments de la commande (NM-V_3)

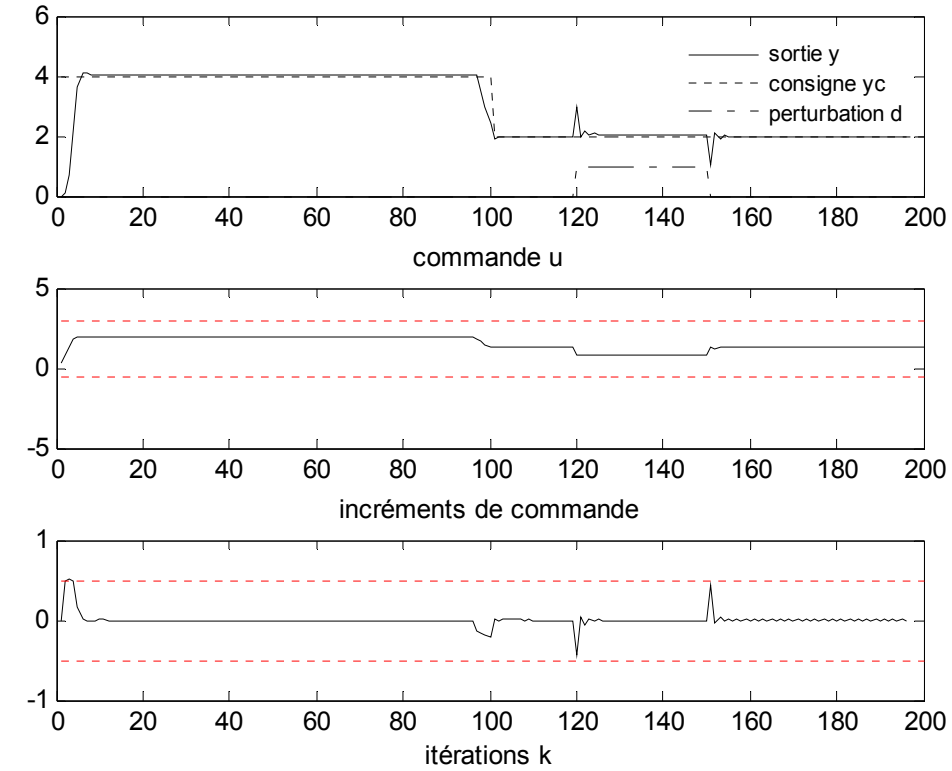


Figure 3-16 : Sortie du système, commande et incréments de la commande (ROS-V_3)

Les figures 3-17 et 3-18 représentent respectivement les résultats obtenus avec la méthode NM-V_3 et la méthode ROS-V_3. Dans ce cas, les incréments de commande sont limités entre -0.1 et 0.1.

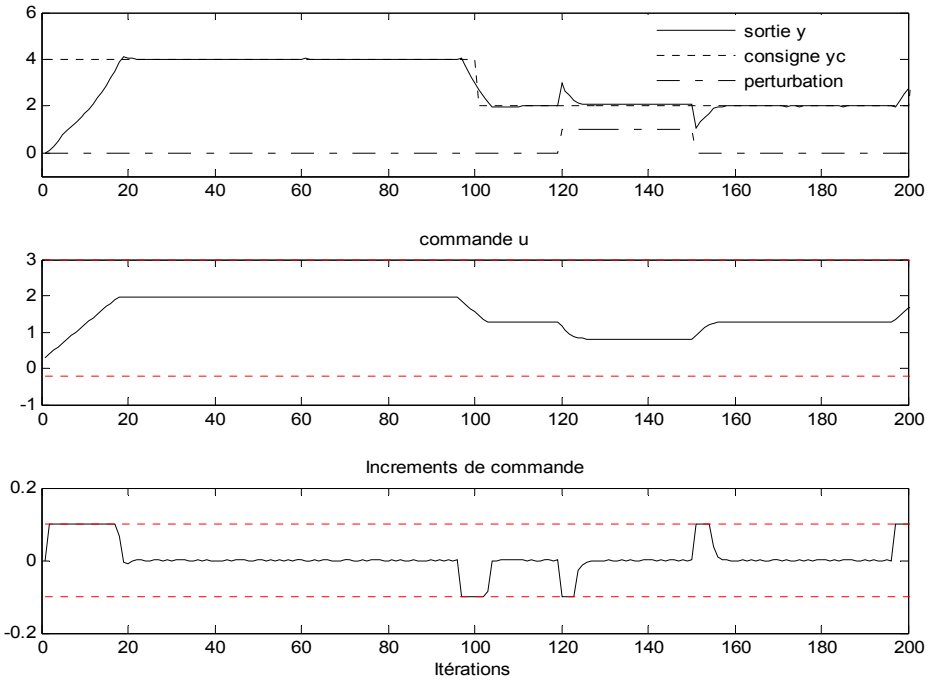


Figure 3-17 : Sortie système, commande et incréments de commande avec $|\Delta u| \leq 0.1$ (NM-V3)

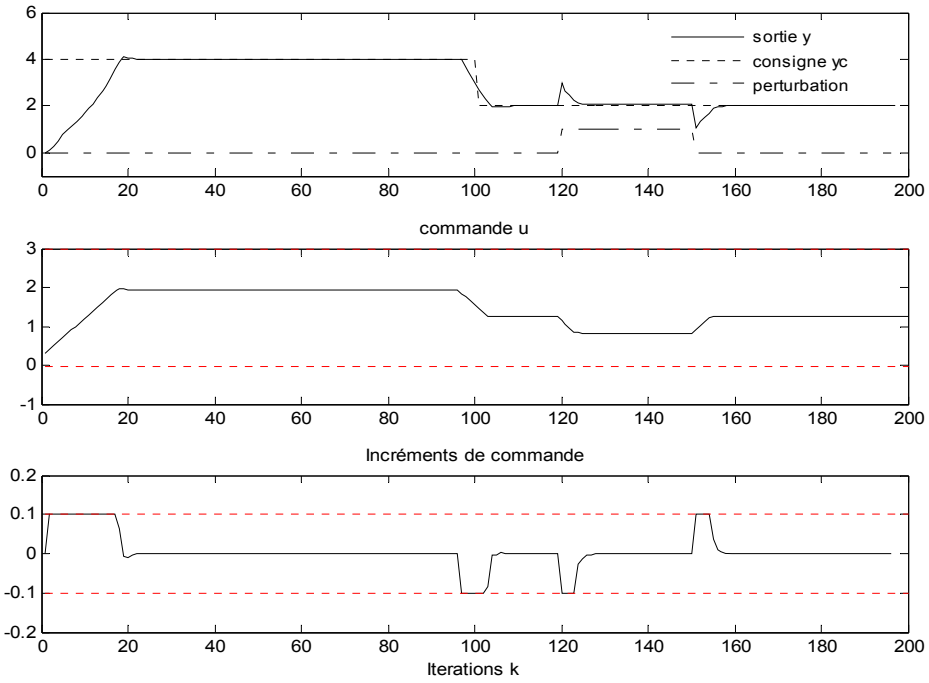


Figure 3-18 : Sortie système, commande et incréments de commande avec $|\Delta u| \leq 0.1$ (ROS-V3)

Les résultats trouvés, en simulation, confirment les résultats obtenus avec les fonctions benchmark. En effet, avec la méthode NM : la version NM-V_3 (figure 3-15) offre les meilleurs résultats trouvés. La version NM-V_2 (figure 3-13) donne des résultats très proches de ceux trouvés avec NM-V_3, offrant un signal de commande stable. L'effet des améliorations introduites est nettement visible sur la loi de commande et par conséquent sur la qualité du signal de sortie obtenue. La version originale NM-V_0 (figure 3-9) est incapable de donner une loi de commande stable permettant de ramener le signal de sortie *y(k)* à la consigne désirée *yc(k)*. La version NM-V_1 (figure 3-11) permet de réaliser cet objectif, cependant, le signal de commande présente beaucoup de fluctuations ce qui entraîne une variance importante du signal de commande.

La méthode ROS offre des résultats très intéressants. En effet, d'après les figures 3-10, 3-12, 3-14, 3-16 et 3-18, toutes les versions (ROS-V_0, ROS-V_1, ROS-V_2 et ROS-V_3) ont aboutis à des résultats comparables avec ceux obtenus avec la version NM-V_3. Cependant, dans le cas des systèmes plus complexes, les deux dernières versions offrent plus de succès. Le traitement de la fonction benchmark (F3) de l'annexe 3, qui présente plusieurs minimums, montre que les versions ROS-V_2 et ROS-V_3 sont beaucoup plus performantes que la première version.

Le tableau 3-4 donne les valeurs des critères de performances considérés (I_1, I_2 et I_3 , Eq. 3-24 au 3-26) obtenues par les deux méthodes d'optimisation, avec les trois versions considérées.

$$I_1 = \frac{1}{N}\sum_{i=1}^{N}(y_c(k+i) - y(k+i))^2 \tag{3-24}$$

$$I_2 = \frac{1}{N}\sum_{i=1}^{N}u(i)^2 \tag{3-25}$$

$$I_3 = \frac{1}{N}\sum_{i=1}^{N}\Delta u(i)^2 \tag{3-26}$$

Méthode		I_1	I_2	I_3
Nelder-Mead	NM-V_1	0.3173	2.4795	0.0092
	NM-V_2	0.2751	2.4979	0.0070
	NM-V_3	0.2745	2.4980	0.0070
Rosenbrock	ROS-V_1	0.2625	2.4893	0.0066
	ROS-V_2	0.2625	2.4893	0.0066
	ROS-V_3	0.2625	2.4893	0.0066

Tableau 3-4 : Critères de performances

A partir du tableau 3-4, on remarque que les trois versions ROS-V_1, ROS-V_2 et ROS-V_3 de la méthode de Rosenbrock convergent vers la même valeur de la commande u. Cependant, les résultats des versions NM-V_3 et NM-V_2 sont comparables et leurs performances sont meilleures que la version NM-V_1. Les résultats obtenues par la méthode de Rosenbrock sont légèrement meilleurs que ceux obtenus par la méthode de Nelder-Mead (NM-V_2 et NM-V_3).

Dans le tableau 3-5, on présente le temps requis par chaque méthode pour calculer la loi de commande. Ces résultats sont obtenus en effectuant 20 simulations aléatoires avec un PC Pentium 4-2.4GHz.

	Temps (s)		
Méthode	t_{min}	t_{moy}	t_{max}
NM-V_1	*0.0457*	*0.0471*	*0.0488*
ROS-V_1	*0.0784*	*0.08*	*0.084*
NM-V_2	*0.4482*	*0.4546*	*0.4582*
ROS-V_2	*0.0859*	*0.1261*	*0.1284*
NM-V_3	*0.4456*	*0.4528*	*0.4583*
ROS-V_3	*0.1282*	*0.1307*	*0.1311*

Tableau 3-5 : Temps de simulation selon la méthode d'initialisation

Le tableau 3-5, indique, respectivement, le temps minimum, le temps maximum et le temps moyen (t_{min}, t_{max} et t_{moy}) mis dans les 20 simulations aléatoires. On constate que la méthode de Rosenbrock est plus rapide que celle de Nelder-Mead. Ceci s'explique par le fait que la première méthode utilise un seul point de recherche, contrairement à la seconde qui exploite un ensemble de point (un simplexe). La taille du simplexe dans la deuxième méthode

a une influence sur le temps de calcul, puisque dans chaque itération il y a évaluation de tous les points du simplexe. Nous notons, aussi, que la première version V_1 de chacune des deux méthodes est plus rapide que les deux autres, mais les résultats fournis en boucle fermée ne sont pas satisfaisants. La deuxième et la troisième version nécessitent presque le même temps de calcul. Cependant, la troisième version fournit toujours des performances plus satisfaisantes en boucle fermée.

Système 2 : le deuxième système est décrit par [Chen, 1999] :

$$\begin{aligned} x_1(k+1) &= 0.1x_1(k) + 2\frac{u(k)+x_2(k)}{1+\left(u(k)+x_2(k)\right)^2} \\ x_2(k+1) &= 0.1x_2(k) + u(k)\left(1+\frac{u^2(k)}{1+x_1{}^2(k)+x_2{}^2(k)}\right) \\ y(k) &= x_1(k)+x_2(k). \end{aligned} \tag{3-27}$$

Le modèle identifié est celui donné par l'algorithme génétique binaire (méthode AGB), puisqu'il présente la plus grande valeur de R^2_{tot}. Le modèle NARMA (*m=3, n=2 et q=3*) est donné par l'équation suivante :

$$\begin{aligned} y(k) &= -0.0444 + 2.1089\,u(k-1) + 1.2887\,u(k-2) + 0.2361\,u(k-3) \\ &+ 0.1337\,u^2(k-1) + 0.3172\,u^3(k-1) - 0.269\,u^3(k-1) - 0.071\,u^2(k-1)\,y(k-2) \\ &- 0.715\,u(k-1)u(k-2)\,y(k-1) + 0.1465\,u(k-1)\,y^2(k-1) \end{aligned} \tag{3-28}$$

Les paramètres du correcteur prédictif considéré dans la simulation sont, $Hc = 1$, $Hp = 4$ et $\lambda = 0.2$. Les contraintes sur la commande et ses incréments sont comme suit :

$$-0.1 \le u(k) \le 0.5 \tag{3-29}$$

$$-0.05 \le \Delta u(k) \le 0.05 \tag{3-30}$$

Les figures 3-21 et 3-22 comportent, respectivement, les évolutions de la sortie, du signal de commande ainsi que les incréments de commande obtenues par la méthode de Nelder-Mead et la méthode de Rosenbrock. Dans cette simulation, nous avons considéré la version 3 des méthodes d'optimisation.

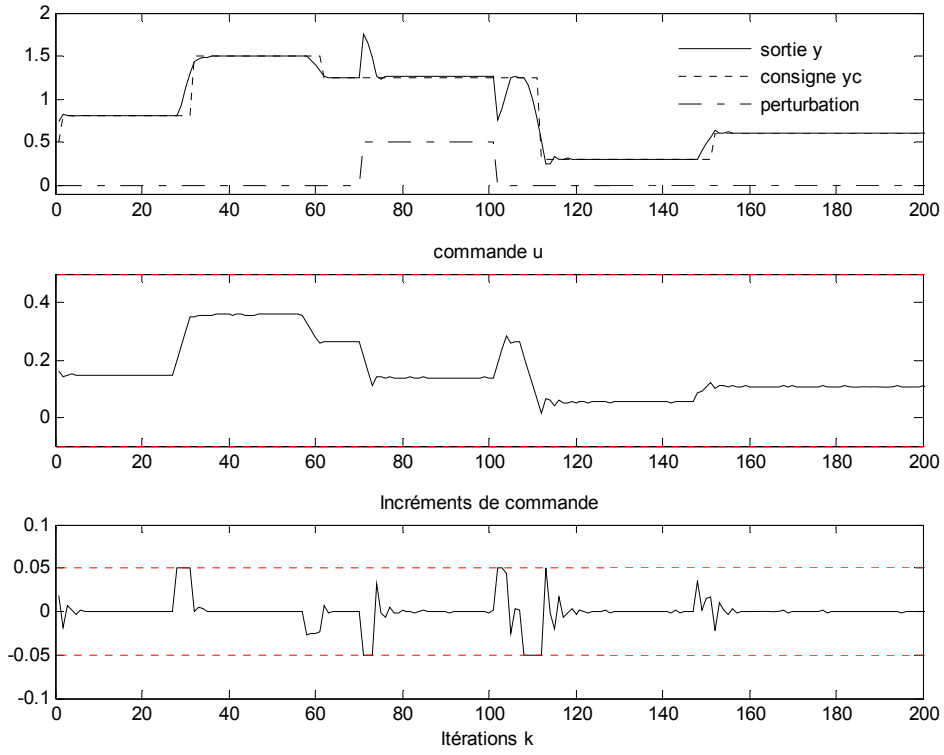


Figure 3-19 : Sortie système (S2), commande et incréments de commande (méthode NM-V_3)

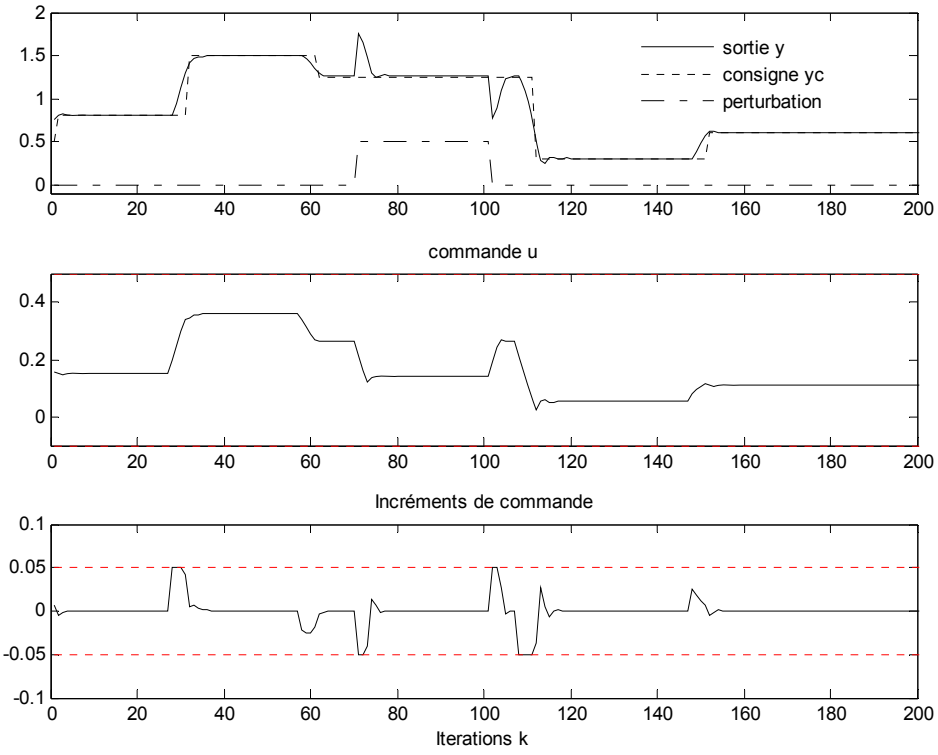


Figure 3-20 : Sortie système (S2), commande et incréments de commande (méthode ROS-V_3)

A partir des figures 3-19 et 3-20, on remarque que les deux méthodes NM-V_3 et ROS-V_3 sont capables de fournir une loi de commande stable permettant de ramener la sortie du système à la consigne désirée. Cependant, la différence entre les deux méthodes se manifeste lors des changements de la valeur de la consigne ou bien en présence de perturbations. Ceci est remarquable dans la figure 3-19. En effet, entre les itérations 100 à 120 et 140 à 160, le signal des incréments de commande présente des fluctuations contrairement à la méthode ROS-V_3 (figure 3-20) qui présente un signal plus stable.

Le tableau 3-6 présente les critères de performances (I_1, I_2 et I_3) ainsi que le temps de calcul moyen par itération des algorithmes NM-V_3 et ROS-V_3.

Méthode	Critères de performances			
	I_1	I_2	I_3	t_{moy}
NM-V_3	0.0129	0.0351	$3.019\ 10^{-4}$	0.0161
ROS-V_3	0.0115	0.0354	$2.516\ 10^{-4}$	0.0175

Tableau 3-6 : Critères de performances (système S2)

Le tableau 3-6 montre que les résultats obtenus avec les méthodes NM-V_3 et ROS-V_3 sont comparables de points de vue stabilité, énergie de commande et temps de calcul. La méthode ROS-V_3 donne une valeur plus faible du critère de performance I_3. Ceci s'explique par la présence des fluctuations au niveau du signal des incréments de commande avec la méthode NM-V_3.

Les figures 3-21 et 3-22 présentent le comportement du système en présence de bruit. Le bruit qui a été ajouté à la sortie, est un signal aléatoire d'amplitude égale à 0.3.

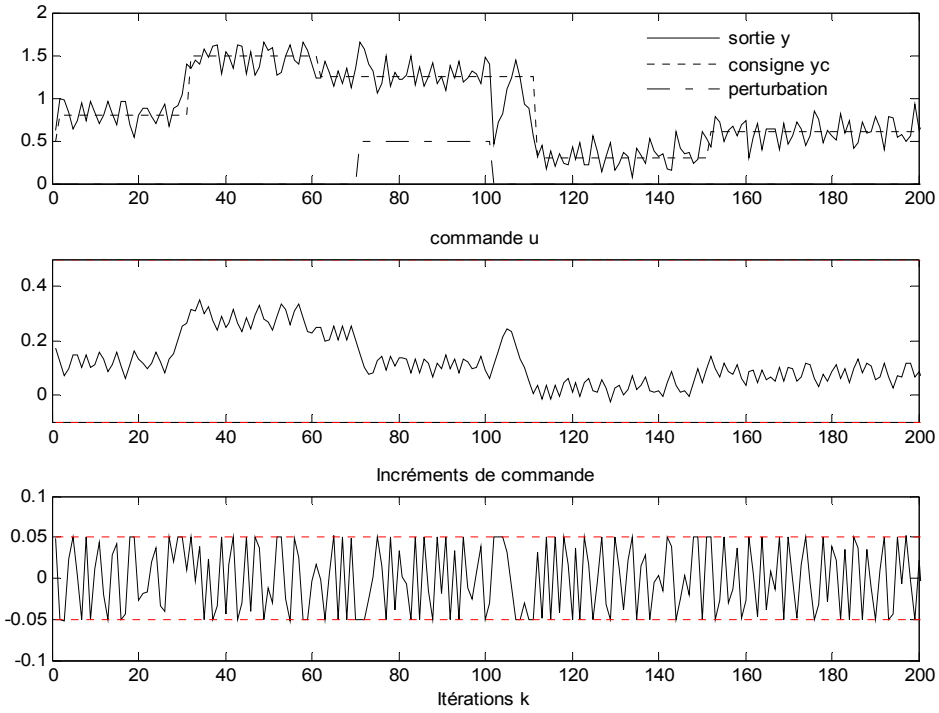


Figure 3-21 : Sortie système (S2), commande et incréments de commande (méthode NM-V_3) avec bruit

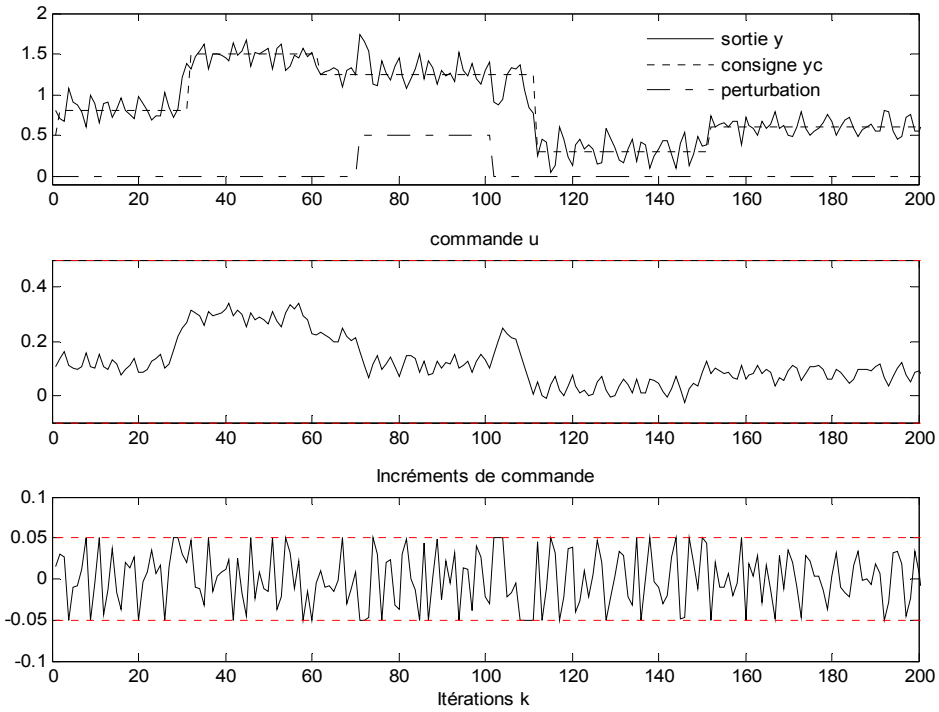


Figure 3-22 : Sortie système (S2), commande et incréments de commande (méthode ROS-V_3) avec bruit

D'après les figures 3-21 et 3-22, on constate que la commande calculée avec les deux méthodes NM-V_3 et ROS-V_3 est capable de rejeter le signal de perturbation *d(k)*, dans un temps raisonnable. Cette commande permet de ramener la sortie à la consigne désirée, même en présence de bruit non négligeable.

3.7. Conclusion

Dans ce chapitre, un contrôleur prédictif des systèmes non-linéaires sous contraintes a été présenté. Le modèle de type NARMA est employé pour caractériser le comportement du système. La loi de commande est obtenue en minimisant un critère quadratique non convexe. Le problème d'optimisation est résolu par deux méthodes utilisant respectivement l'algorithme de Nelder-Mead et l'algorithme de Rosenbrock. Ces méthodes, qui sont déterministes, locales et sans contraintes ont étés améliorées par l'introduction d'une fonction de pénalité, de l'approche CFON ainsi que l'utilisation de la notion de multi initialisation, ce qui favorise la convergence vers le minimum global. La qualité du signal de commande obtenu avec ces méthodes est très bonne, comparée aux résultats obtenus avec les versions originales. Le temps de calcul de la loi de commande qui est relativement faible ainsi que la bonne précision qu'offre la méthode de Rosenbrock améliorée nous encourage à traiter des systèmes plus complexes et même à dynamiques rapides.

Comparée aux méthodes d'optimisation qui utilisent la dérivée, les méthodes que nous avons proposées ont l'avantage d'êtres exploitées facilement dans le cas des systèmes multivariables. Dans le chapitre suivant nous nous intéresserons à l'identification et à la commande des systèmes multivariables en exploitant les modèles de type NARMA.

Chapitre 4

Identification et commande prédictive des systèmes non linéaires multivariables

Chapitre 4 Identification et commande prédictive des systèmes non linéaires multivariables

Chapitre 4

Identification et commande prédictive des systèmes non linéaires multivariables

4.1. Introduction

Dans ce chapitre, on s'intéressera à l'identification et la commande prédictive des systèmes multivariables en utilisant les modèles de type NARMA. Dans le deuxième chapitre, trois approches ont été développées pour la détermination des modèles de type NARMA. La première approche est basée sur l'algorithme de régression par pas échelonné [Kortmann, 1988], les deux autres méthodes exploitent respectivement les algorithmes AGB et RN.

L'analyse des trois approches, présentées dans le deuxième chapitre, permet de conclure que la dernière méthode (RN) peut être développée pour traiter les systèmes multivariables. La méthode proposée dans ce chapitre constitue, donc, une combinaison du réseau de neurones artificiels à fonction d'activation polynomiale avec l'algorithme génétique sous sa représentation réelle. Dans ce cas, les individus de la population de l'algorithme génétique représentent les coefficients du polynôme de la fonction d'activation. Les paramètres du modèle NARMA sont alors estimés à partir des pondérations des connexions du réseau neuronal à la fin de la phase d'apprentissage. Le réseau de neurones utilisé dans le cas des systèmes monovariables nécessite uniquement la modification du nombre d'entrées et du nombre de neurones de la couche de sortie pour être exploité dans le cas des systèmes multivariables. L'algorithme génétique codé réel utilisé pour l'estimation des coefficients de la fonction d'activation polynomiale des neurones de la couche cachée reste pratiquement le même. Le seul changement dans cet algorithme concerne le critère de performance qui intervient dans la fonction d'évaluation (*fitness*). Pour le calcul de la loi de commande prédictive non linéaire sous contraintes, nous proposons l'extension des deux méthodes utilisées dans le chapitre 3 (NM et ROS).

Ce chapitre est organisé comme suit : Dans le deuxième paragraphe, on présente la structure du modèle NARMA dans le cas multivariable. Le troisième paragraphe est réservé au développement de l'approche utilisée pour l'estimation structurelle et paramétrique du modèle. Le quatrième paragraphe traite la commande prédictive multivariable, en exploitant

les méthodes utilisées dans le chapitre 3, et le cinquième paragraphe présente les résultats de simulation. La conclusion est donnée dans le dernier paragraphe.

4.2. Le modèle NARMA (cas multivariable)

On considère la classe des systèmes non linéaires qui peuvent être modélisés par la relation suivante :

$$Y(k) = G(\phi(k)) + E(k) \text{ avec } \phi(k) = \left[Y(k-1)\dots Y(k-n)\,U(k-1)\dots U(k-m)\right] \tag{4-1}$$

G est une fonction non linéaire supposée inconnue. Les paramètres *m* et *n* représentent, respectivement, l'ordre de la régression sur les entrées et l'ordre de la régression sur les sorties. *E(k)* est le vecteur de bruit. $U(k) \in \Re^{(m_1,1)}$ et $Y(k) \in \Re^{(n_1,1)}$ désignent, respectivement, le vecteur des entrées et le vecteur des sorties.

Pour un système multivariable ayant m_1 entrées et n_1 sorties, chaque sortie peut être exprimée en fonction du vecteur d'observation par la relation suivante :

$$y_i(k) = g_i(\phi(k)), \qquad i = 1,\dots,n_1 \tag{4-2}$$

avec g_i est une fonction non linéaire et $\phi(k)$ est formé par les anciennes mesures de toutes les sorties et de toutes les entrées :

$$\begin{aligned}\phi(k) = \big[&u_1(k-1)\dots u_1(k-m)\dots u_{m1}(k-1)\dots u_{m1}(k-m)\\ &y_1(k-1)\dots y_1(k-n)\dots y_{n1}(k-1)\dots y_{n1}(k-n)\big]\end{aligned} \tag{4-3}$$

La structure dynamique de la sortie du modèle général de type NARMA, opérant dans un environnement déterministe, peut être régie par une équation mathématique discrète. Pour des raisons de simplification nous proposons une présentation matricielle, donnée par l'équation (4-4). Dans cette présentation nous avons intégré la valeur moyenne de chaque sortie du système dans la matrice des paramètres, puisque leurs valeurs seront identifiées avec l'ensemble des paramètres du modèle.

$$\hat{Y}(k) = \left[\hat{y}_1(k) \quad \hat{y}_2(k) \quad \ldots \quad \hat{y}_{n_1}(k)\right]^T \tag{4-4}$$

$$= \begin{bmatrix} \bar{y}_1 & \bar{y}_2 & \ldots & \bar{y}_{n_1} \\ b_{111} & b_{211} & \ldots & b_{n_1 11} \\ \vdots & & & \vdots \\ b_{11m} & b_{21m} & \ldots & b_{n_1 1m} \\ b_{121} & b_{221} & \ldots & b_{n_1 21} \\ \vdots & & & \vdots \\ b_{12m} & b_{22m} & \ldots & b_{n_1 2m} \\ \vdots & & & \vdots \\ b_{1m_1 m} & b_{2m_1 m} & \ldots & b_{n_1 m_1 m} \\ a_{111} & a_{211} & \ldots & a_{n_1 11} \\ \vdots & & & \vdots \\ a_{1n_1 n} & a_{2n_1 n} & \ldots & a_{n_1 n_1 n} \\ \vdots & & & \vdots \\ c_{1r} & c_{2r} & \ldots & c_{n_1 r} \end{bmatrix}^T \cdot \begin{bmatrix} 1 \\ u_1(k-1) \\ \vdots \\ u_1(k-m) \\ u_2(k-1) \\ \vdots \\ u_2(k-m) \\ \vdots \\ u_{m_1}(k-m) \\ y_1(k-1) \\ \vdots \\ y_{n_1}(k-n) \\ \vdots \\ y_{n_1}(k-n) \ldots y_{n_1}(k-n) \end{bmatrix}$$

$\bar{y}_i$ représente la valeur moyenne de la sortie *i*.

Chaque sortie $\hat{y}_i(k) \quad (i = 1,\ldots,n_1)$ du modèle peut être écrite sous la forme suivante:

$$\hat{y}_i(k) = \hat{\Theta}_i{}^T \Phi(k) \tag{4-5}$$

avec $\hat{\Theta}_i{}^T$ et $\Phi(k)$ désignent, respectivement, le vecteur de paramètres et le vecteur d'observation.

$$\hat{\Theta}_i = \left[\bar{y}_i \quad \theta_{i1} \quad \theta_{i2} \quad \ldots \quad \theta_{ir}\right]^T \tag{4-6}$$

$$\Phi(k) = [1 \quad v_1 \quad v_2 \quad \ldots \quad v_r]^T \tag{4-7}$$

où

$$\begin{aligned} &v_1(k) = u_1(k-1), \ldots, v_m(k) = u_1(k-m), \\ &v_{m+1}(k) = u_2(k-1), \ldots, v_{2m}(k) = u_2(k-m), \\ &v_{2m+1}(k) = u_3(k-1), \ldots, v_{m_1 m}(k) = u_{m_1}(k-m), \\ &v_{m_1 m+1}(k) = y_1(k-1), \ldots, v_{n_1 n+m_1 m}(k) = y_{n_1}(k-n),\ldots, \\ &\vdots \\ &v_{n_1 n+m_1 m+1}(k) = u_1(k-1)^2, \ldots, v_r(k) = y_{n_1}(k-n)^q \end{aligned} \tag{4-8}$$

Le nombre de termes possibles qui peuvent intervenir dans l'expression de chaque sortie du modèle $\hat{y}_i(k)$ est donné par la relation suivante :

$$r = \frac{(n_1 n + m_1 m + q)!}{q!(n_1 n + m_1 m)!} - 1 \tag{4-9}$$

où le paramètre q représente le degré de non linéarité du modèle. Par conséquent, le nombre de modèles possibles qu'on peut avoir pour un système qui comporte n_1 sorties est :

$$w = (2^r - 1)^{n_1} \tag{4-10}$$

Pour m, n, q, n_1 et m_1 donnés, il existe un nombre très élevé de combinaisons possibles. Par exemple, si le système comporte deux entrées ($m_1=2$) et deux sorties ($n_1=2$), pour $m=n=q=2$, on aura $3.0949\ 10^{26}$ combinaisons différentes. Il est donc nécessaire d'utiliser une procédure permettant la sélection d'un modèle parmi l'ensemble des modèles possibles qui représente le comportement du système réel.

On présente par la suite la méthode utilisée pour l'identification du modèle multivariable de type NARMA.

4.3. Identification du modèle NARMA

Notre méthode exploite hors ligne un fichier formé par les mesures de l'entrée et de la sortie du système : ($Y(k), U(k),\ k=1,\ldots,N$). En effet, les paramètres du modèle NARMA peuvent être estimés en utilisant un réseau de neurones à fonction d'activation polynomiale, comme décrit dans le premier chapitre. La structure du réseau de neurones est de type «feed-forward» avec une seule couche cachée. La typologie de ce réseau dans le cas multivariable est représentée dans la figure 4-1. Le réseau neuronal reçoit en entrée les éléments du vecteur d'observation $\phi(k)$ et délivre à sa sortie une estimation des sorties actuelles.

Les sorties de la première couche sont données par :

$$S(k) = f(W^1 \Phi^T(k)) = \left[s_1(k) \ldots s_M(k)\right]^T \tag{4-11}$$

W^1 est la matrice des pondérations entre la couche d'entrée et la couche cachée. f étant une fonction d'activation polynomiale et M désigne le nombre de neurones de la couche cachée. La relation (4-12) présente une fonction d'activation polynomiale d'ordre q :

$$f(x_i) = f_0 + f_1 x_i + f_2 x_i^2 + ... + f_q x_i^q \tag{4-12}$$

En tenant compte des poids des connexions entre la couche d'entrée et la couche cachée, la quantité x_i sera exprimée par :

$$\begin{aligned} x_i &= \sum_{j=1}^{m} w_{ji}^1 u_1(k-j) + \sum_{j=m+1}^{2m} w_{ji}^1 u_2(k-j+m) \\ &+ ... + \sum_{j=m(m_1-1)+1}^{m_1 m} w_{ji}^1 u_{m_1}(k-j+m(m_1-1)) + \\ &\sum_{j=m_1 m+1}^{m_1 m+n} w_{ji}^1 y_1(k-j+m_1 m) + \sum_{j=m_1 m+n+1}^{m_1 m+2n} w_{ji}^1 y_2(k-j+m_1 m+n) \\ &+ ... + \sum_{j=m_1 m+n(n_1-1)+1}^{m_1 m+n_1 n} w_{ji}^1 y_{n_1}(k-j+m_1 m+n(n_1-1)). \quad i \in [1, M] \end{aligned} \tag{4-13}$$

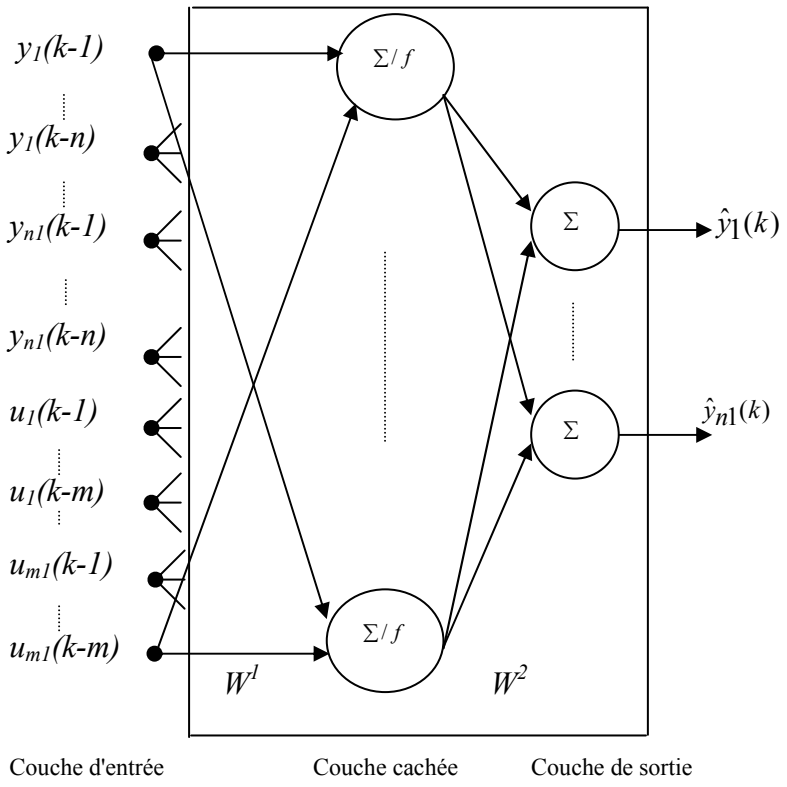


Figure 4-1 : Réseau de neurones à une seule couche cachée (cas d'un système multivariable)

Comme dans le cas des systèmes monovariables, le choix de M dépend aussi de la complexité du système. En effet, un nombre faible de neurones cachés peut conduire à un modèle non précis alors que le choix d'un nombre élevé de neurones cachés peut rendre le modèle beaucoup plus compliqué, puisqu'il s'agit des combinaisons non linéaires de toutes les entrées et de toutes les sorties.

Ainsi, le nombre de neurones cachés est choisi de façon à réaliser un compromis entre la précision et la complexité du modèle estimé.

L'équation de la sortie du réseau sera alors:

$$\hat{Y}(k) = \left[\hat{y}_1(k) \ldots \hat{y}_{n1}(k)\right]^T = W^2 S(k) \tag{4-14}$$

où $\quad \hat{y}_j(k) = \sum_{i=1}^{M} w_{ij}^2 f(x_i). \qquad j = \left[1,.., n_1\right]$

W^2 désigne la matrice des pondérations entre la couche cachée et la couche de sortie.

En remplaçant la quantité $f(x_i)$ de l'équation (4-12) par sa valeur dans (4-14), on obtient :

$$\begin{aligned} \hat{y}_j(k) = {} & w_{1j}^2 \left(f_0 + f_1 x_1 + \ldots + f_q x_1^q\right) \\ & + w_{2j}^2 \left(f_0 + f_1 x_2 + \ldots + f_q x_2^q\right) \\ & + \ldots + w_{Mj}^2 \left(f_0 + f_1 x_M + \ldots + f_q x_M^q\right) \end{aligned} \tag{4-15}$$

En remplaçant (x_i) par son expression donnée par (4-13), l'expression de la sortie du réseau sera comme suit :

$$\hat{y}_j(k) = \Theta_F^T \ \Phi(k) \tag{4-16}$$

où $\quad \Theta_F = h\left(f_0, f_1 \ldots, f_q, W^1, W^2\right),$

h: fonction non linéaire, $\dim \Theta_F = \left[r+1, 1\right].$

Les pondérations (W^1 et W^2) des connexions du réseau de neurones sont déterminées avec l'algorithme de rétro propagation du gradient. La déduction des paramètres du modèle NARMA se fait alors à partir des pondérations (W^1 et W^2) et des paramètres du polynôme considéré ($f_0, f_1, \ldots, f_q$). En effet, par identification de l'équation (4-16) avec l'équation (4-4), on trouve les coefficients des termes du modèle NARMA.

Puisque la qualité du modèle obtenu, avec cette méthode, dépend du choix adéquat des coefficients ($f_0, f_1, ..., f_q$) du polynôme. Nous proposons l'incorporation de la même procédure utilisée dans le chapitre 2, c'est à dire la combinaison de l'algorithme génétique réel avec l'algorithme de rétro propagation du gradient pour déterminer ces coefficients. La figure 4-2 présente la structure du superviseur utilisé pour la détermination du modèle NARMA.

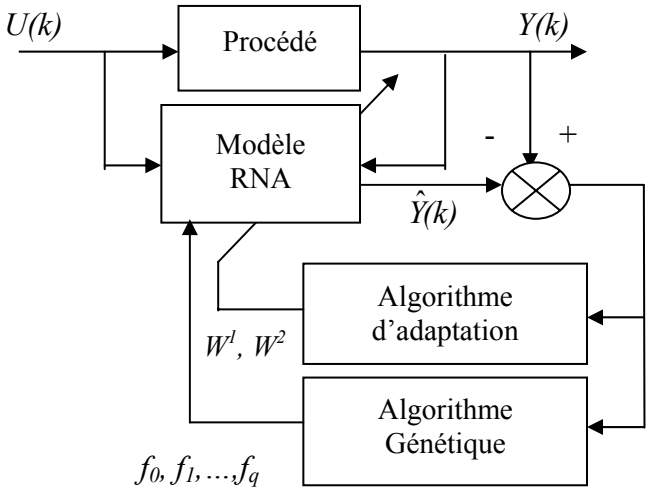


Figure 4-2 : Structure du superviseur basé sur l'algorithme génétique

Le critère J_3 qui intervient dans la fonction d'évaluation (*fitness*) de l'algorithme génétique codé réel s'exprime comme suit :

$$J_3 = \sum_{k=1}^{N} \left(\left(y_1(k) - \hat{y}_1(k) \right)^2 + ... + \left(y_{n1}(k) - \hat{y}_{n1}(k) \right)^2 \right) \tag{4-17}$$

Avec cette méthode, le calcul des paramètres du modèle NARMA se fait de la même façon que dans le cas monovariable, il faut tenir compte seulement du nombre d'entrées et de sorties. Par ailleurs, le rôle de l'algorithme génétique est de trouver les valeurs des coefficients de la fonction d'activation des neurones alors que l'algorithme de propagation du gradient s'occupe de l'ajustement des poids des connexions.

4.4. Commande prédictive multivariable

Le principe de la commande prédictive présenté dans la cas monovariable reste le même dans le cas multivariable. Cependant, pour traiter les systèmes multivariables, on doit tenir compte de certaines spécificités tel que :

- le modèle utilisé,
- le calcul du prédicteur,
- le critère de performance
- et la méthode d'optimisation adoptée.

4.4.1. Calcul du prédicteur

Le calcul du prédicteur multivariable $\hat{y}(k+j) \quad j=1,\ldots,Hp$, nécessite l'exploitation de l'équation du modèle. A partir de l'équation (4-4), qui décrit le modèle général de type NARMA, l'équation du prédicteur sera donnée comme suit :

$$\hat{Y}(k+l)=\left[\hat{y}_1(k+l) \quad \hat{y}_2(k+l) \quad \ldots \quad \hat{y}_{n_1}(k+l)\right]^T ;\ l=1,\ldots,Hp \tag{4-18}$$

$$=\begin{bmatrix} \bar{y}_1 & \bar{y}_2 & \ldots & \bar{y}_{n_1} \\ b_{111} & b_{211} & \ldots & b_{n_1 11} \\ \vdots & & & \vdots \\ b_{11m} & b_{21m} & \ldots & b_{n_1 1m} \\ b_{121} & b_{221} & \ldots & b_{n_1 21} \\ \vdots & & & \vdots \\ b_{12m} & b_{22m} & \ldots & b_{n_1 2m} \\ \vdots & & & \vdots \\ b_{1m_1m} & b_{2m_1m} & \ldots & b_{n_1m_1m} \\ a_{111} & a_{211} & \ldots & a_{n_1 11} \\ \vdots & & & \vdots \\ a_{1n_1n} & a_{2n_1n} & \ldots & a_{n_1n_1n} \\ \vdots & & & \vdots \\ c_{1r} & c_{2r} & \ldots & c_{n_1r} \end{bmatrix}^T \cdot \begin{bmatrix} 1 \\ u_1(k-1+l) \\ \vdots \\ u_1(k-m+l) \\ u_2(k-1+l) \\ \vdots \\ u_2(k-m+l) \\ \vdots \\ u_{m_1}(k-m+l) \\ y_1(k-1+l) \\ \vdots \\ y_{n_1}(k-n+l) \\ \vdots \\ y_{n_1}(k-n+l)\ldots y_{n_1}(k-n+l) \end{bmatrix}$$

4.4.2. Le critère de performance

Le critère de performance utilisé contient (n_1+m_1) termes quadratiques. Les (n_1) premiers termes sont en relation avec les erreurs futures entre les sorties prédites et les

consignes désirées, tandis que, les m_1 autres termes correspondent aux incréments futurs de commandes liées aux m_1 entrées du système.

Le critère utilisé est ainsi donné par la relation suivante :

$$J_4(k) = \frac{1}{2}\left[\sum_{j=1}^{n_1}\sum_{i=1}^{Hp_j} (\hat{y}_j(k+i) - y_{j_c}(k+i))^2 + \sum_{j=1}^{m_1}\sum_{i=1}^{Hc_j}\lambda_j(\Delta u_j(k+i\text{-}1))^2\right] \qquad (4\text{-}19)$$

avec :

m_1 : nombre des signaux de commande du système,

n_1 : nombre de sorties du système,

Hp_j: horizon de prédiction sur la sortie j,

Hc_j : horizon de commande sur le signal de commande j,

λj: coefficient de pondération sur la commande (pour l'entrée j).

Dans le cas où les valeurs de l'horizon de prédiction pour toutes les sorties sont identiques, les valeurs de l'horizon de commande pour toutes les entrées sont égales et les coefficients de pondération sur la commande sont identiques, l'équation du critère sera alors comme suit :

$$J(k) = \frac{1}{2}\left[\sum_{j=1}^{n_1}\sum_{i=1}^{Hp} (\hat{y}_j(k+i) - y_{j_c}(k+i))^2 + \lambda\sum_{j=1}^{m_1}\sum_{i=1}^{Hc} (\Delta u_j(k+i\text{-}1))^2\right] \qquad (4\text{-}20)$$

4.4.3. Optimisation du critère

Pour le calcul de la loi de commande, nous proposons l'extension des méthodes d'optimisation présentées dans le chapitre 3, à savoir la méthode du simplexe et celle de Rosenbrock, pour le cas multivariable. Afin de traiter les contraintes et pour avoir des bonnes performances, nous avons intégré les mêmes améliorations introduites dans le cas monovariable, c'est-à-dire, la méthode de pénalité, l'approche CFON, la notion de multi initialisation ainsi que la notion d'élitisme.

Au niveau de l'initialisation des algorithmes d'optimisations on considère une matrice de dimension (m, $n+1$) dans le cas de la méthode de Nelder-Mead et m directions de recherche

pour la méthode de Rosenbrock. m désigne le nombre de variables à optimiser et $n+1$ représente la taille de chaque simplexe.

La fonction de pénalité est définie de manière à respecter, pour chaque entrée du système, les contraintes sur le signal de commande et sur ses incréments.

Les contraintes peuvent être écrites comme suit :

$$u_{i_{\min}} \le u_i(k+j) \le u_{i_{\max}}, \quad j = 0,\dots,Hc-1,\ i = 1,\dots,m \tag{4-21}$$

$$\Delta u_{i_{\min}} \le \Delta u_i(k+j) \le \Delta u_{i_{\max}}, \quad j = 0,\dots,Hc-1,\ i = 1,\dots,m \tag{4-22}$$

Hc : Horizon de commande.

En partant l'équation (3-17) du chapitre 3, nous pouvons écrire la fonction de pénalité dans le cas multivariable comme suit :

$$\alpha(x) = \sum_{i=1}^{m} \sum_{j=1}^{4\,Hc} \left[Max \ \{0, f_{ij}(x)\} \right]^2 \tag{4-23}$$

L'application de l'approche CFON et de la notion multi initialisation, en utilisant les versions d'optimisation NM-V_3 et ROS-V_3, nécessite l'intégration d'une procédure qui comporte deux étapes :

Etape 1 : Fixer les intervalles espaces de recherche pour chaque variable (S_j, $j=1,\dots,m$) en respectant globalement toutes les contraintes.

Etape 2 : Diviser chaque espace de recherche S_j en n sous intervalles égaux s_i ($i = 1,\dots, n$) et refaire à chaque fois l'initialisation des différentes variables de manière à balayer toutes les combinaisons possibles des sous intervalles s_i.

Le choix du nombre de divisions de l'espace de recherche n dépend de la complexité du problème. En effet, pour un problème d'optimisation simple on prend n faible et pour un cas complexe on choisi un nombre plus important.

Dans le chapitre 3 nous avons présenté l'algorithme de Rosenbrock dans le cas général (mono et multivariable).

L'algorithme de Nelder-Mead (cas multivariable)

L'algorithme de Nelder-Mead, appliqué aux problèmes d'optimisation multivariables, est résumé par les étapes suivantes :

Etape 1 :

Prendre m polyèdres de départ (m : nombre de variables)

Prendre les sommets $(x_1, x_2, ..., x_{n+1})$ de chaque polyèdre vérifiant

$$f(x_1) \le f(x_2) \le \ldots \le f(x_{n+1}).$$

Etape 2 :

Tant que la différence entre $f(x_{n+1})$ et $f(x_1)$ est supérieure à un certain seuil de précision ε_0.

2.1 calculer $\bar{x}$ et $f_r = f(x(\rho))$ pour chaque polyèdre

avec $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, est le centroïde des n meilleurs sommets, de chaque polyèdre, et $x(\rho) = \bar{x} + \rho \cdot (\bar{x} - x_{n+1})$

2.2 réflexion :

si $f(x_1) \le f_r < f(x_n)$.

Alors $x_{n+1} = x(\rho)$ pour chaque polyèdre, aller à 2.7.

2.3 expansion :

si $f_r < f(x_1)$ alors calculer $f_e = f(x(\chi))$, avec $x(\chi) = \bar{x} + \chi \cdot (x_r - \bar{x})$

si $f_e < f_r$ alors, pour chaque polyèdre prendre $x_{n+1} = x(\chi)$ et aller à 2.7

sinon $x_{n+1} = x(\rho)$ et aller à 2.7.

2.4 contraction externe :

si $f(x_n) \le f_r < f(x_{n+1})$ alors calculer $f_o = f(x(\gamma))$,avec $x(\gamma) = \bar{x} + \gamma \cdot (x_r - \bar{x})$

si $f_o \le f_r$ alors pour chaque polyèdre prendre $x_{n+1} = x(\gamma)$ puis aller à 2.7

sinon aller à 2.6

2.5 contraction interne

si $f_r \ge f(x_{n+1})$ alors calculer $f_i = f(x(\gamma))$, avec $x(\gamma) = \bar{x} - \gamma \cdot (\bar{x} - x_{n+1})$

si $f_i < f(x_{n+1})$, pour chaque polyèdre prendre $x_{n+1} = x(\gamma)$ puis aller à 2.7

sinon aller à 2.6.

2.6 rétrécissement

pour $i = 2, \ldots, n+1$, calculer pour chaque polyèdre prendre $x_i = \frac{(x_i + x_1)}{2}$,

puis calculer $f(x_i)$

2.7 donner les sommets $(x_1, x_2, \ldots, x_{n+1})$ *de chaque polyèdre prendre tel que*

$f(x_1) \leq f(x_2) \leq \ldots \leq f(x_{n+1})$.

retour à l'étape 2.

4.5. Résultats de simulation

A. Identification des systèmes multivariables

Au cours de la phase d'identification, le signal d'entrée utilisé est un bruit blanc de moyenne nulle et de variance égale à un. Les coefficients de corrélation totaux $R^2_{tot_i}$ ($i=1, \ldots, n_1$) sont calculés afin de décider si le modèle obtenu est acceptable ou non. En effet, le modèle est acceptable si ces critères sont proches de 1 [Haber, 1990].

$$R^2_{tot_i} = 1 - \frac{\sum_{k=1}^{N} (y_i(k) - \hat{y}_i(k))^2}{\sum_{k=1}^{N} (y_i(k))^2}, \quad i=1, \ldots, n_1 \tag{4-24}$$

Les paramètres de réglage utilisés dans les simulations sont résumés dans le tableau 4-1.

	Algorithme génétique réel	Réseau de neurones artificiels
Nombre de générations	40	-
Nombre d'individus	100	-
Probabilité de croisement	0.8	-
Probabilité de mutation	0.1	-
Nombre d'époques	-	5000
Coef. d'apprentissage	-	0.005

Tableau 4-1 : Paramètres de réglage des algorithmes

Trois systèmes multivariables non linéaires, avec deux entrées et deux sorties, sont considérés. Le premier est présenté par une structure de type NARMA. Le deuxième exemple est un benchmark qui présente un système plus compliqué. Cependant, le troisième exemple

concerne une application pratique, il s'agit d'un procédé de régulation de niveau d'eau caractérisé par trois réservoirs interconnectés.

Le choix ($m_1=n_1= m=n=q=2$), permet d'avoir 44 termes possibles dans l'expression générale de chacune des sorties du modèle NARMA.

***Système 1* :**

Le premier système est décrit par une équation de type NARMA donnée comme suit :

$$\begin{cases} y_1(k) = 0.8\,y_1(k-1) + u_1(k-2) + 0.4(u_1(k-2))^2 \\ \qquad\quad - 1.2u_1(k-1)u_2(k-2) - 0.1y_2(k-1). \\ y_2(k) = 0.5\,y_2(k-1) + u_2(k-2) + (u_1(k-1))^2 \\ \qquad\quad + 0.5\,y_2(k-2)u_2(k-1). \end{cases} \tag{4-25}$$

Système 2 :

Le deuxième système multivariable est caractérisé par l'équation 4- 26. Il présente un cas plus complexe [Song, 2006], [Petlenkov, 2007].

$$\begin{cases} y_1(k) = \dfrac{a_1\,y_1(k-1)\,y_1(k-2)}{1+a_2\,y_1^2(k-1)+a_3\,y_2^2(k-2)} + a_4\,u_1(k-2) + a_5\,u_1(k-1) + a_6\,u_2(k-2) \\ \\ y_2(k) = \dfrac{b_1\,y_2(k-1)\sin(y_2(k-2))}{1+b_2\,y_2^2(k-1)+b_3\,y_1^2(k-2)} + b_4\,u_2(k-2) + b_5\,u_2(k-1) + b_6\,u_1(k-2) \end{cases} \tag{4-26}$$

Les paramètres du système sont : $a_1=0.7$; $a_2=1$; $a_3=1$; $a_4=0.3$; $a_5=1$ et $a_6=0.2$; $b_1=0.5$; $b_2=1$; $b_3=1$; $b_4=0.5$; $b_5=1$ et $b_6=0.2$.

Système 3 :

Le troisième système multivariable est schématisé par la figure 4-3(b). Il présente un procédé de régulation de niveau d'eau. Ce procédé de laboratoire est constitué de trois réservoirs cylindriques connectés entre eux par des valves et qui possèdent une connexion avec un bassin de niveau plus bas (figure 4-3(a)). Le niveau d'eau dans chaque réservoir

dépend de la quantité d'eau qui entre et qui sort du réservoir. Cette quantité dépend du débit donné par les vannes V1 et V2. Le paramètre *hi (i=1,2,3)* désigne le niveau de l'eau dans le réservoir *i*, q_1 et q_2 sont les débit dans les réservoirs 1 et 2, S_j représente la section de la colonne *j*, q_{ij} représente le débit de l'eau qui passe du réservoir *i* au réservoir *j* et q_{i0} représente le débit sortant du réservoir *i*.

Dans notre cas, les sorties du modèle NARMA sont les niveaux dans les réservoirs 1 et 2 ($y_1=h_1$, $y_2=h_2$) et les entrées sont les débits d'entrées ($u_1=q_1$, $u_2=q_2$). Un fichier de 1600 mesures (entrées/sorties) est considéré pour la modélisation du système, les 1100 premières mesures sont utilisées pour l'identification du modèle et les 500 dernières mesures sont utilisées pour la validation de ce dernier.

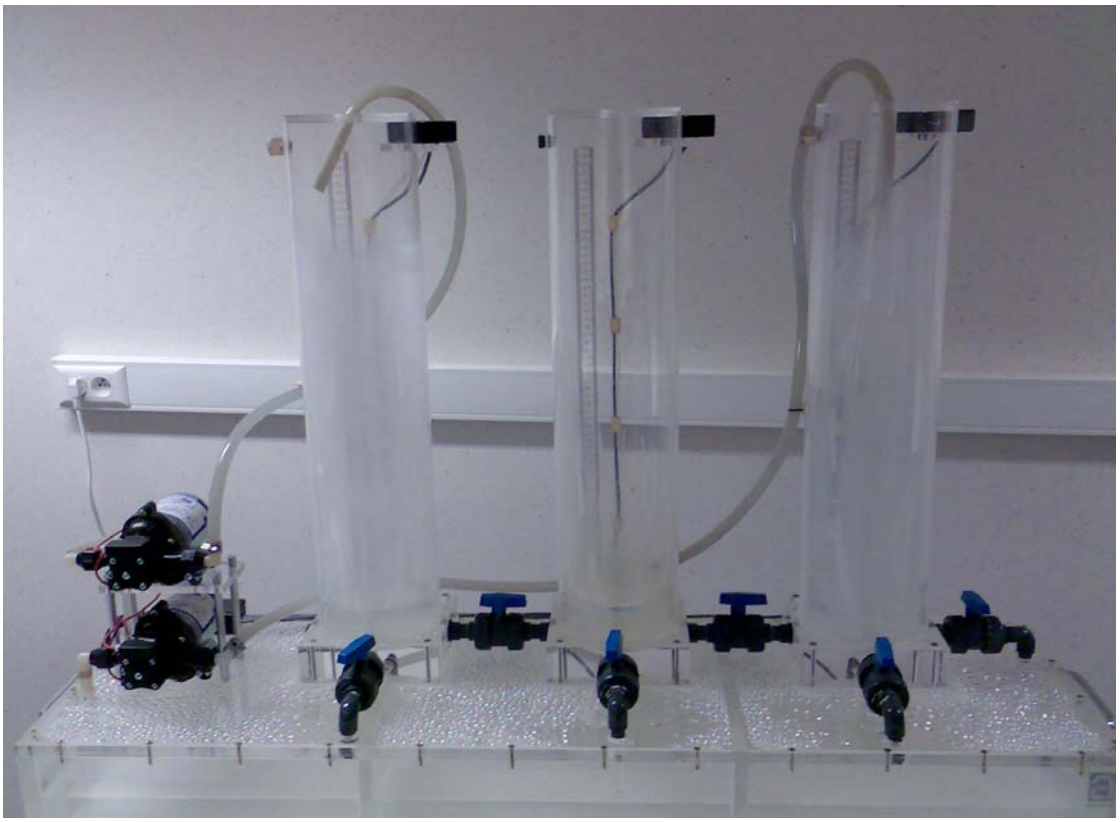


Figure 4-3(a) : Procédé de régulation du niveau d'eau à trois réservoirs interconnectés

Le procédé de la figure 4-3(a) peut être schématisé par la figure 4-3(b), comme suit :

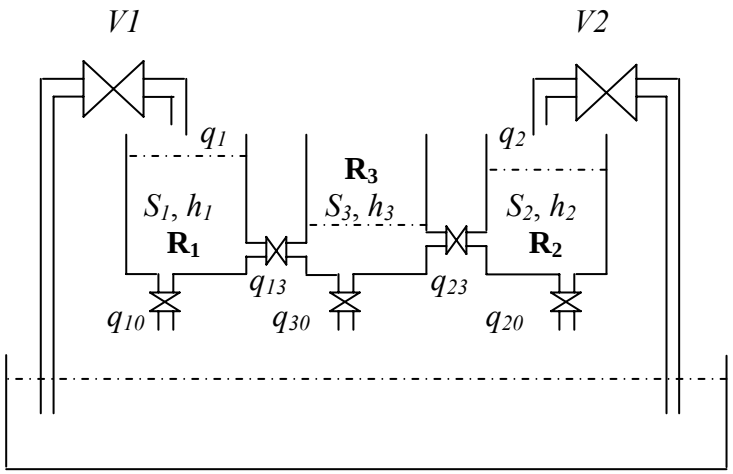


Figure 4-3(b) : Schéma du système à trois réservoirs (système multivariable 3)

Les coefficients de la fonction d'activation trouvés par l'algorithme génétique réel, pour les trois systèmes sont donnés dans le tableau 4-2.

	Coefficients de la fonction d'activation polynomiale		
	f_0	f_1	f_2
Système 1	15	-4.16	13.5
Système 2	2.5	14.2	25
Système 3	5	26.3	35

Tableau 4-2 : Coefficients de la fonction d'activation polynomiale pour les trois systèmes identifiés

Le tableau 4-3 présente les termes sélectionnés ainsi que les paramètres estimés des modèles NARMA correspondants aux trois systèmes identifiés.

N	Termes Possibles	Système 1		Système 2		Système 3		Système 3 (rectifié)	
		$y_1(k)$	$y_2(k)$	$y_1(k)$	$y_2(k)$	$y_1(k)$	$y_2(k)$	$y_1(k)$	$y_2(k)$
0	$\bar{y}$	0	0	0.0465	-0.0142	0.0113	0.0005	0.0113	0
1	$u_1(k-1)$	0	0	0.9918	0.0001	0.0037	-0.0028	0.0037	-0.0028
2	$u_1(k-2)$	1.0001	0	0.4076	0.1826	0.0027	0.0034	0.0027	0.0034
3	$u_2(k-1)$	0	0	0.0012	0.9988	-0.0023	0.0032	-0.0023	0.0032
4	$u_2(k-2)$	0.0001	1	0.1880	0.5282	0.0040	-0.0074	0.0040	-0.0074
5	$y_1(k-1)$	0.7999	0	-0.0749	0.0104	0.4862	0.1909	0.4862	0.1909
6	$y_1(k-2)$	0.0001	0	0.0610	0.0015	0.3090	-0.0553	0.3090	-0.0553
7	$y_2(k-1)$	-0.100	0.5	0.0174	-0.0318	0.0342	0.4367	0.0342	0.4367
8	$y_2(k-2)$	0	0	0.0024	0.0040	0.0500	0.4573	0.0500	0.4573
9	$u_1^2(k-1)$	0	1	0.0148	-0.0120	0.0000	0.0000	0	0
10	$u_1(k-1)\,u_1(k-2)$	0	0	0.0297	-0.0240	0.0000	0.0001	0	0
11	$u_1(k-1)\,u_2(k-1)$	0	0	0.0028	-0.0017	-0.0003	0.0006	0	0
12	$u_1(k-1)\,u_2(k-2)$	-1.200	0	0.0103	0.0104	-0.0000	0.0010	0	0
13	$u_1^2(k-2)$	0.3998	0	0.0043	-0.0128	0.0000	0.0000	0	0
14	$u_1(k-2)\,u_2(k-1)$	0	0	-0.0258	-0.0235	0.0002	-0.0002	0	0
15	$u_1(k-2)\,u_2(k-2)$	0	0	-0.0062	-0.0179	-0.0000	-0.0002	0	0
16	$u_2^2(k-1)$	0	0	-0.0178	0.0065	0.0012	-0.0005	0	0
17	$u_2(k-1)\,u_2(k-2)$	0	0	-0.0035	0.0246	-0.0006	0.0033	0	0.0033
18	$u_2^2(k-2)$	0	0	0.0090	0.0409	-0.0006	0.0090	0	0.0090
19	$u_1(k-1)\,y_1(k-1)$	0	0	-0.0533	0.0357	-0.0020	-0.0013	0	0
20	$u_1(k-1)\,y_1(k-2)$	0	0	0.0525	-0.0186	-0.0010	-0.0002	0	0
21	$u_1(k-1)\,y_2(k-1)$	0	0	0.0126	-0.0157	-0.0005	-0.0019	0	0
22	$u_1(k-1)\,y_2(k-2)$	0	0	0.0193	0.0224	-0.0004	-0.0018	0	0
23	$u_1(k-2)\,y_1(k-1)$	0	0	-0.0181	0.0192	0.0028	0.0006	0.0028	0
24	$u_1(k-2)\,y_1(k-2)$	0	0	0.0196	-0.0079	0.0018	-0.0002	0	0
25	$u_1(k-2)\,y_2(k-1)$	0	0	0.0026	-0.0245	0.0005	0.0018	0	0
26	$u_1(k-2)\,y_2(k-2)$	0	0	0.0087	-0.0058	0.0002	0.0019	0	0
27	$u_2(k-1)\,y_1(k-1)$	0	0	0.0291	0.0257	0.0016	-0.0012	0	0
28	$u_2(k-1)\,y_1(k-2)$	0	0	-0.0148	0.0089	0.0006	0.0005	0	0
29	$u_2(k-1)\,y_2(k-1)$	0	0	-0.0112	-0.0489	0.0017	-0.0039	0	-0.0039
30	$u_2(k-1)\,y_2(k-2)$	0	0.5	0.0213	0.0240	-0.0006	-0.0043	0	-0.0043
31	$u_2(k-2)\,y_1(k-1)$	0	0	0.0083	0.0212	0.0003	-0.0044	0	-0.0044
32	$u_2(k-2)\,y_1(k-2)$	0	0	0.0006	0.0031	0.0025	-0.0045	0.0025	-0.0045
33	$u_2(k-2)\,y_2(k-1)$	0	0	-0.0065	-0.0147	-0.0053	0.0019	-0.0053	0
34	$u_2(k-2)\,y_2(k-2)$	0	0	0.0149	0.0455	-0.0048	-0.0043	-0.0048	-0.0043
35	$y_1^2(k-1)$	0	0	0.0042	-0.0036	0.0732	-0.0336	0.0732	-0.0336
36	$y_1(k-1)\,y_1(k-2)$	0	0	-0.0042	0.0032	0.0476	0.0019	0.0476	0
37	$y_1(k-1)\,y_2(k-1)$	0	0	0.0050	0.0017	0.0488	-0.0389	0.0488	-0.0389
38	$y_1(k-1)\,y_2(k-2)$	0	0	0.0070	0.0024	0.0175	-0.0516	0.0175	-0.0516
39	$y_1^2(k-2)$	0	0	0.0042	-0.0017	0.0215	0.0180	0.0215	0.0180
40	$y_1(k-2)\,y_2(k-1)$	0	0	-0.0029	-0.0001	0.0529	-0.0440	0.0529	-0.0440
41	$y_1(k-2)\,y_2(k-2)$	0	0	-0.0035	0.0036	0.0336	-0.0410	0.0336	-0.0410
42	$y_2^2(k-1)$	0.0002	0	-0.0100	0.0190	-0.0438	0.0829	-0.0438	0.0829
43	$y_2(k-1)\,y_2(k-2)$	0	0	-0.0088	0.0305	-0.0466	0.0588	-0.0466	0.0588
44	$y_2^2(k-2)$	0.0001	0	-0.0028	0.0611	-0.0489	0.0345	-0.0489	0.0345

Tableau 4-3 : Termes et paramètres estimés pour les trois systèmes multivariables

Dans le tableau 4-3, la colonne notée (système 3 (rectifié)), comporte seulement les termes dont les paramètres sont inférieurs ou égales à $2\ 10^{-3}$. Le tableau 4-4 présente la somme des erreurs quadratiques normalisées des deux sorties du modèle du système 3 et du modèle rectifié correspondant.

Erreurs quadratiques normalisées (modèle et modèle rectifié) du système 3	
Erreur par rapport à la sortie y_1	Erreur par rapport à la sortie y_2
$1.5446\ 10^{-5}$	$1.3019\ 10^{-5}$

Tableau 4-4 : Valeurs de la somme des erreurs quadratiques normalisées entre le modèle trouvé et le modèle rectifié (système 3)

Le tableau 4-5 comporte les valeurs des coefficients de corrélation totaux (R^2_{tot}) et du temps de calcul des simulations effectuées avec un PC Pentium 4, 2 GHz.

	Système 1		Système 2		Système 3	
	$y_1(k)$	$y_2(k)$	$y_1(k)$	$y_2(k)$	$y_1(k)$	$y_2(k)$
R^2_{tot}	0.9953	0.9998	0.9926	0.9911	0.9936	0.9931
Temps (sec)	2295.120		2331.56		2531.56	

Tableau 4-5 : Les valeurs de R^2_{tot} et du temps de calcul (système multivariable)

Le nombre de neurones de la couche caché est : $M = 5$.

Commentaires :

On remarque, d'après le tableau 4-3, que les termes et les paramètres du système 1 sont identifiés avec succès. Le bon choix des coefficients de la fonction d'activation a permis d'avoir une bonne précision.

On constate, d'après le tableau 4-5 que les valeurs de R^2_{tot} sont très proche de 1 ce qui prouve que les modèles trouvés sont précis. Les performances des méthodes d'identification peuvent être améliorées en augmentant le nombre d'itérations, puisque l'identification se fait hors ligne.

Afin de valider le modèle correspondant au système 3, on a comparé les évolutions des sorties du système avec celles du modèle identifié. Les séquences d'entrées sont choisies de type échelon de différents niveaux. La figure 4-4 présente respectivement les séquences d'entrées (u_1, u_2), les sorties système/modèles et l'erreur de modélisation pour la première

sortie y_1 et les sorties système/modèles et l'erreur de modélisation pour la deuxième sortie y_2. On remarque que les erreurs de modélisation sont très faibles, ce qui permet de conclure que le modèle obtenu caractérise convenablement la dynamique du système considéré.

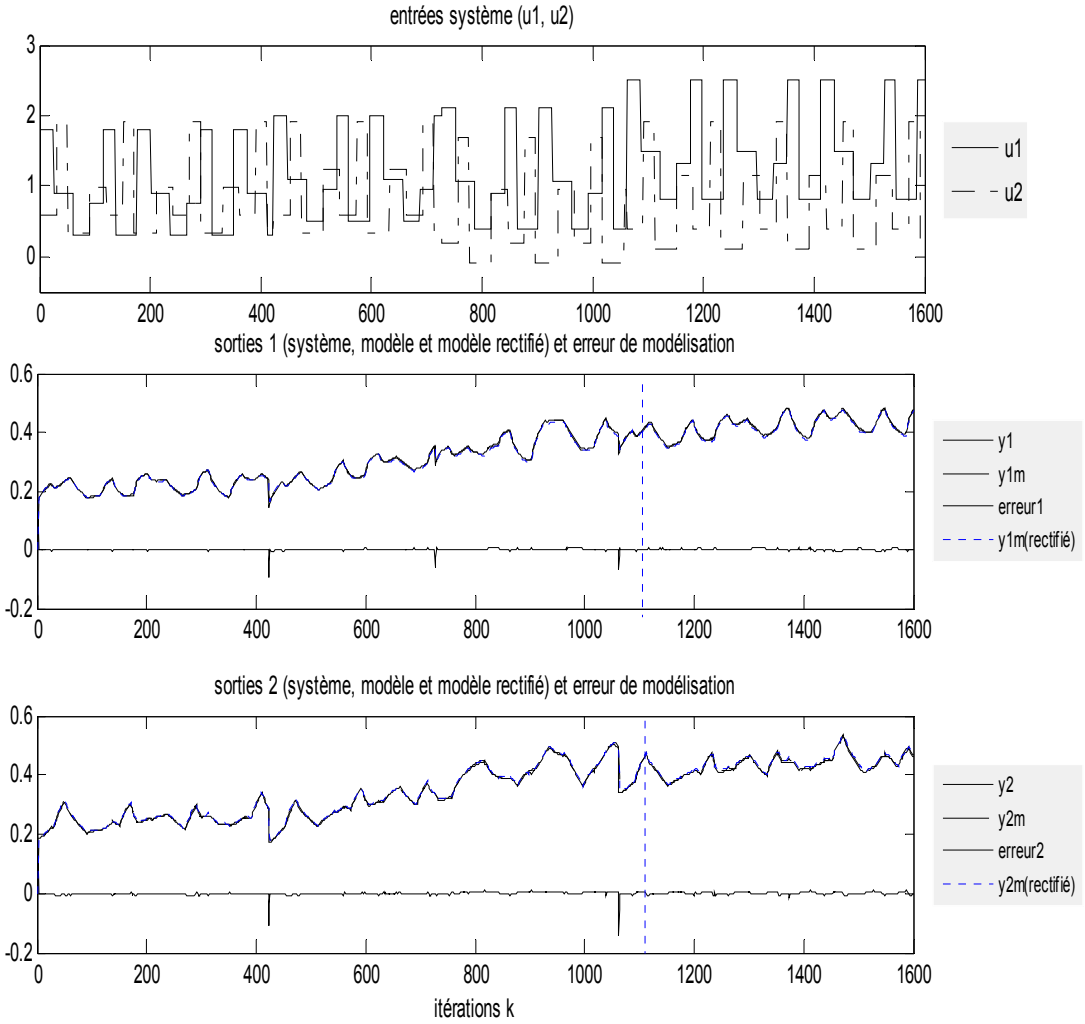


Figure 4-4 : Evolutions des séquences d'entrées (u_1, u_2), des sorties (système et modèle) et des erreurs de modélisation

Les sorties du système, du modèle et du modèle rectifié sont pratiquement confondues. Le tableau 4-4 montre aussi que l'erreur entre le modèle et le modèle rectifié est très faible. Ceci nous permet de considérer le modèle rectifié, qui présente moins de termes avec une précision comparable au modèle obtenu initialement.

B. Commande des systèmes multivariables

Pour appliquer nos algorithmes de commande, nous avons considéré le système multivariable non linéaire de l'équation (4-26). Dans une première étape, nous avons introduit des perturbations (d_1 et d_2) respectivement aux deux sorties (y_1 et y_2) du système considéré. Ensuite, nous avons supposé des variations au niveau des paramètres du système de l'ordre de ± 10 % et ± 20 %. La loi de commande a été calculée en utilisant le modèle nominal.

Ainsi, le système objet de l'étude, avec incertitudes sur les paramètres et en présence de perturbations, est donné par l'équation 4-27.

$$\begin{cases} y_1(k) = \dfrac{a_1\, y_1(k-1)\, y_1(k-2)}{1 + a_2\, y_1^2(k-1) + a_3\, y_2^2(k-2)} + a_4\, u_1(k-2) + a_5\, u_1(k-1) + a_6\, u_2(k-2) + d_1(k) \\ y_2(k) = \dfrac{b_1\, y_2(k-1)\sin(y_2(k-2))}{1 + b_2\, y_2^2(k-1) + b_3\, y_1^2(k-2)} + b_4\, u_2(k-2) + b_5\, u_2(k-1) + b_6\, u_1(k-2) + d_2(k) \end{cases} \tag{4-27}$$

Les caractéristiques des signaux de perturbation $d_1(k)$ et $d_2(k)$, ajoutés respectivement aux sorties du système $y_1(k)$ et $y_2(k)$, sont définies par l'équation (4-28).

$$\begin{aligned} d_1(k) &= 0.2 \qquad k \in [120, \ldots, 170] \quad ; \quad 0 \ \textit{ailleurs} \\ d_2(k) &= 0.2 \qquad k \in [80, \ldots, 120] \quad ; \quad 0 \ \textit{ailleurs} \end{aligned} \tag{4-28}$$

Dans les algorithmes de commande, nous avons considéré les versions d'optimisation (NM-V_3 et ROS-V_3) associés avec l'approche CFON ainsi que la fonction de pénalité.

Les paramètres du correcteur prédictif considéré dans la simulation sont, $Hc = 1$, $Hp = 4$ et $\lambda = 0.2$. Les contraintes sur les entrées et leurs incréments sont comme suit :

$$\begin{aligned} -0.1 \le u_1(k) \le 1; \quad -0.1 \le u_2(k) \le 1 \\ -0.1 \le \Delta u_1(k) \le 0.1; \quad -0.1 \le \Delta u_2(k) \le 0.1 \end{aligned} \tag{4-29}$$

Le tableau 4-6 présente les paramètres a_i et b_i (i=1,…,6) du système nominal et du système incertain.

	Paramètres du système		
	Système nominal	*Système incertain (± 10 %)*	*Système incertain (± 20 %)*
a_1	*0.7*	0.77	0.56
a_2	*1*	0.9	1.2
a_3	*1*	1.1	0.8
a_4	*0.3*	0.33	0.36
a_5	*1*	0.9	1.2
a_6	*0.2*	0.22	0.16
b_1	*0.5*	0.55	0.4
b_2	*1*	1.1	1.2
b_3	*1*	0.9	1.2
b_4	*0.5*	0.55	0.4
b_5	*1*	1.1	0.8
b_6	*0.2*	0.18	0.24

Tableau 4-6 : Paramètres des systèmes nominal et incertains

Dans ces simulations, on a considéré les paramètres du système nominal pour les itérations 1 jusqu'à 200. Entre les itérations 201 et 400, nous avons utilisé les mêmes séquences des signaux des consignes et des perturbations pour le système avec incertitude de (+/- 10 %) sur les paramètres et entre les itérations 401 et 600 une incertitude de (+/- 20 %) sur les paramètres. Les figures 4-5 et 4-6 présentent les résultats trouvés avec la méthode *NM* et les figures 4-7 et 4-8 présentent les résultats trouvés avec la méthode *ROS*.

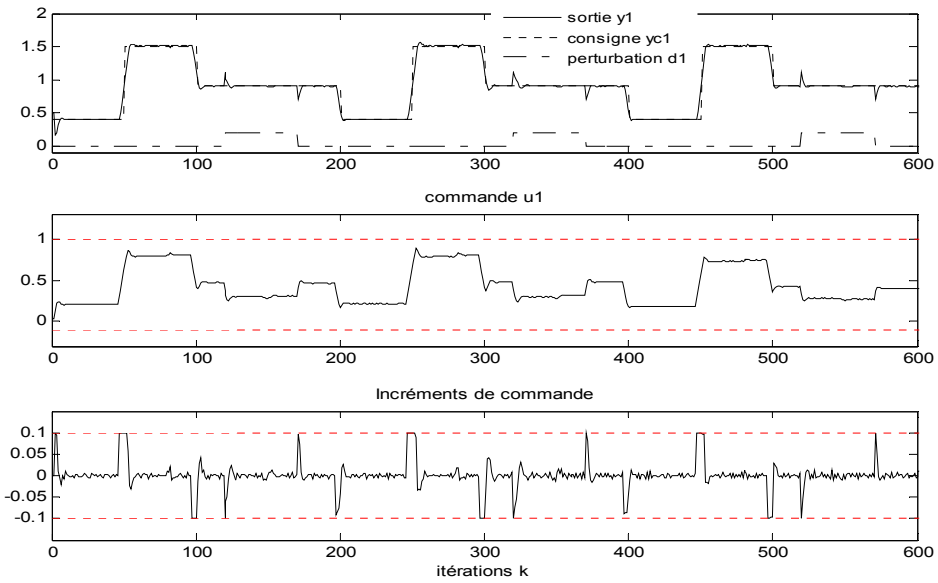


Figure 4-5 : Sortie y_1, commande et incréments de commande (NM-V_3)

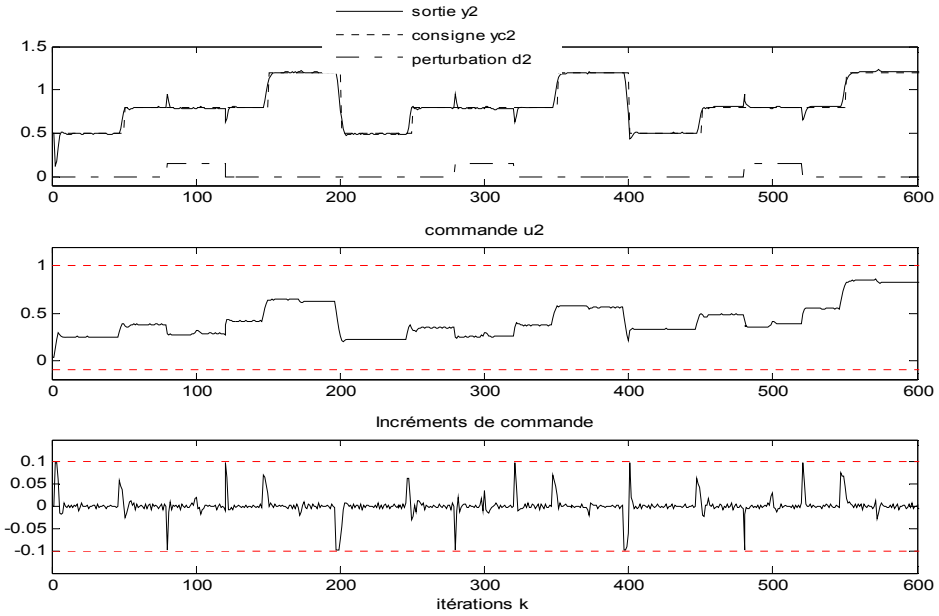


Figure 4-6 : Sortie y_2, commande et incréments de commande (NM-V_3)

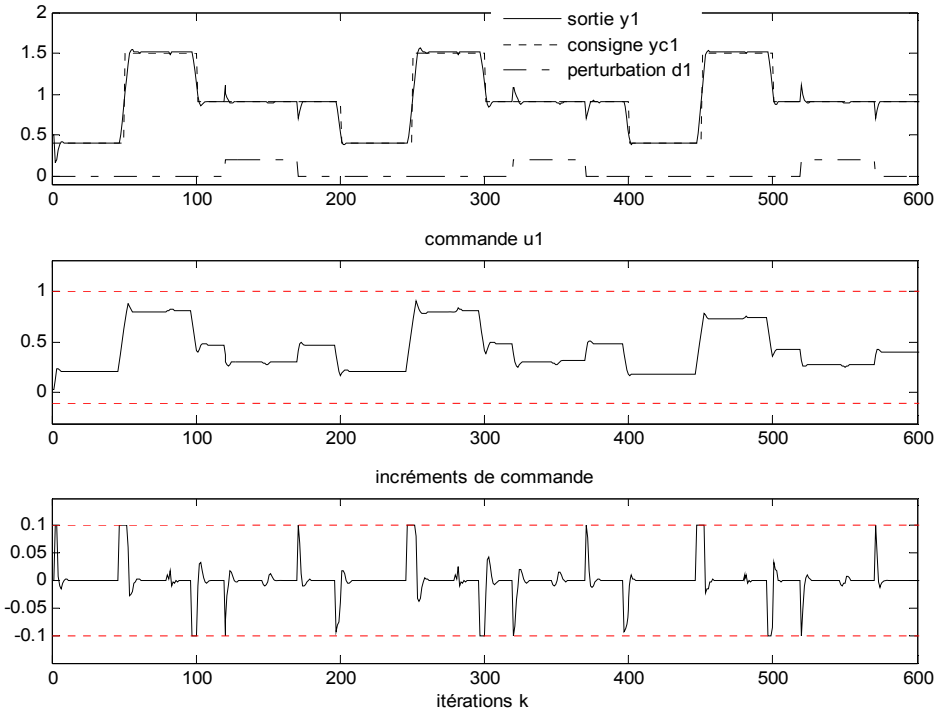


Figure 4-7 : Sorties y_1, commande et incréments de commande (ROS-V_3)

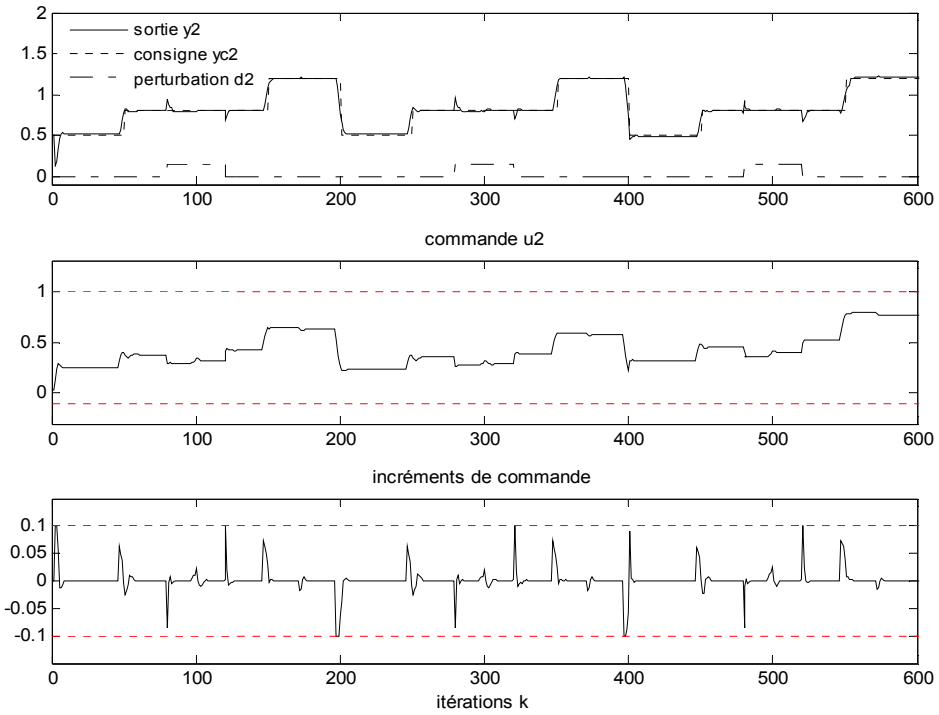


Figure 4-8 : Sortie y_2, commande et incréments de commande (ROS-V_3)

Le tableau 4-7 présente les valeurs des critères de performances I_1, I_2 et I_3, donnés par les équations (3-24 - 3-26) ainsi que le temps moyen par itération, obtenues par les deux méthodes d'optimisation avec le système nominal.

Méthode		I_1	I_2	I_3	t_{moy} *(sec)*
Nelder-Mead	Sortie y1	0.0041	0.1803	$7.0547\ 10^{-4}$	1.3128
	Sortie y2	0.0049	0.2285	$5.4768\ 10^{-4}$	
Rosenbrock	Sortie y1	0.0042	0.1802	$7.0238\ 10^{-4}$	0.2767
	Sortie y2	0.0049	0.2286	$5.4668\ 10^{-4}$	

Tableau 4-7 : Valeurs des critères de performances et temps de calcul moyen (système multivariable)

On remarque, d'après le tableau 4-7, que les performances des deux méthodes d'optimisation sont très proches, assurant un signal de commande de faible variance et une erreur de poursuite quadratique minimale. Cependant, la méthode de Rosenbrock est pratiquement cinq fois plus rapide que celle de Nelder-Mead.

4.6. Conclusion

Le travail présenté, dans ce chapitre, concerne la modélisation et la commande prédictive des systèmes non linéaires multivariables en exploitant les modèles NARMA. Nous avons proposé l'extension de la méthode qui combine le réseau de neurones artificiels à fonction d'activation polynomiale et l'algorithme génétique réel, présentée dans le chapitre 2, pour l'identification des modèles multivariables de type NARMA. Cette méthode permet la sélection des termes ainsi que l'estimation des paramètres du modèle NARMA. Le modèle trouvé sera par la suite exploité pour la synthèse de la commande prédictive. Trois exemples ont été présentés pour valider notre méthode d'identification. Le premier exemple est un cas simple, le deuxième représente un cas plus compliqué et le troisième est une application pratique qui présente un procédé physique de laboratoire. Les résultats trouvés ont confirmé l'efficacité de la méthode proposée. En effet, les modèles NARMA déterminés caractérisent avec une précision acceptable et avec une complexité raisonnable le comportement des systèmes étudiés.

Dans une deuxième étape, nous avons appliqué les algorithmes de commande (NM-V_3 et ROS-V_3) pour commander un système qui possède deux entrées et deux sorties. Les performances trouvées montrent la robustesse des algorithmes de commande proposés. En effet, les signaux de commande respectent d'une part les contraintes imposées et permettent d'autre part aux sorties du système d'atteindre les consignes désirées. De plus, le correcteur prédictif permet d'éliminer les perturbations et traite convenablement les incertitudes sur les paramètres du système.

Conclusion générale

Dans ce mémoire, nous nous sommes intéressé à l'identification et la commande prédictive des systèmes non linéaires mono et multivariables, en exploitant les modèles NARMA. Pour l'identification des modèles de type NARMA, nous avons présenté en premier lieu l'algorithme de régression par pas échelonné modifié. Dans l'implémentation de cet algorithme nous avons généralisé son fonctionnement de manière à considérer des ordres élevés, à savoir les ordres de régressions sur l'entrée et la sortie ainsi que le degré de non linéarité du modèle. Parmi les difficultés rencontrées dans la programmation de cet algorithme, on distingue le calcul des coefficients de corrélation partiels, des tests statistiques partiels et des critères d'informations partiels, puisqu'il s'agit de la permutation des termes afin de tester l'influence de chacun de ceux qui forment la structure du modèle courant. Ensuite, nous avons proposé deux autres méthodes heuristiques. La première méthode est basée sur les algorithmes génétiques binaires et la deuxième méthode constitue une combinaison entre le réseau de neurones artificiels à fonction d'activation polynomiale et l'algorithme génétique sous sa représentation réelle. Cette dernière permet de chercher les coefficients de la fonction polynomiale simultanément avec l'apprentissage du réseau de neurones. Les méthodes que nous avons proposées sont moins compliquées que la méthode de Kortmann et offrent des résultats similaires. D'autre part ces méthodes, contrairement à la méthode de Kortmann, sont capables de traiter les systèmes multivariables. Cet avantage nous a permis de faire l'extension de la deuxième méthode proposée pour la modélisation des systèmes multivariables. Les méthodes proposées sont également efficaces quand nous n'avons aucune connaissance a priori de la structure du système. Cet avantage permet à ces approches de traiter les systèmes complexes.

Pour valider ces méthodes, nous avons considéré quatre exemples monovariables et trois multivariables, dont une application pratique. Les résultats trouvés, pratiques ou de simulations, ont confirmé l'efficacité et la robustesse des méthodes proposées. En effet, les modèles NARMA déterminés caractérisent avec une précision acceptable et avec une complexité raisonnable le comportement des systèmes étudiés.

Par la suite nous avons proposé un contrôleur prédictif des systèmes non-linéaires sous contraintes, qui exploite les modèles de type NARMA. La loi de commande est obtenue en minimisant un critère quadratique non convexe. Le problème d'optimisation est résolu par deux méthodes utilisant l'algorithme de Nelder-Mead et l'algorithme de Rosenbrock qui

n'utilisent pas la dérivée. Ces méthodes, qui sont à l'origine déterministes, locales et sans contraintes, combinées avec la fonction de pénalité, l'approche CFON ainsi que l'utilisation de la notion de multi initialisation permettent une meilleure convergence vers le minimum global. Les méthodes d'optimisation proposées présentent deux avantages essentiels, d'une part elles ne nécessitent pas le calcul de la dérivée et d'autre part elles sont facilement exploitables dans le cas des systèmes multivariables. Dans ce cadre nous avons appliqué les algorithmes de commande avec les versions (NM-V_3 et ROS-V_3) pour commander un système qui possède deux entrées et deux sorties. Les performances trouvées montrent la robustesse des algorithmes de commande proposés. En effet, la commande calculée respecte les contraintes imposées et permet de ramener les sorties du système aux consignes désirées même en présence de perturbation ou voir même dans le cas d'un système incertain.

D'après les résultats de simulations, on constate que la méthode ROS requière peu de temps de calcul, par rapport à celle de NM et offre des bonnes performances. Ceci nous permet de recommander cette approche pour traiter les systèmes rapides. Dans la méthode d'identification basée sur la combinaison du réseau de neurones artificiels à fonction d'activation polynomiale et l'algorithme génétique sous sa représentation réelle, l'expression de la sortie du modèle NARMA peut être calculée par un programme informatique approprié qui prend en considération l'ensemble des paramètres du réseau de neurones. Par conséquent, on peut généraliser le fonctionnement de cet algorithme pour des degrés de non linéarités plus importants.

Suite à ces travaux, plusieurs perspectives peuvent êtres proposées. Parmi lesquelles on peut citer:

- Simplification, d'avantage, la complexité du modèle NARMA, en éliminant les termes ayant une faible contribution dans la sortie du modèle.
- Optimisation de la structure et des paramètres du réseau de neurones permettant l'identification du modèle NARMA.
- Amélioration des performances de l'algorithme NM par optimisation de ses paramètres, surtout pour la définition de la taille du simplexe.
- Mise en œuvre pratique des approches proposées sur des procédés réels.

Bibliographie

[Adetona, 2004] O. Adetona, S. Sathananthan, and L. H. Keel, 2004. *Approximations of the NARMA model of Non-affine Plants*, Proceeding of the 2004 American Control Conference, Boston, Massachusetts, June 30-July 2, 2004.

[Akaike, 1974] H. Akaike, 1974. *A new look at the statistical model – validation*, IEEE. Transactions on Automatic Control, Vol. AC 19, N° 6, pp. 716-723.

[Bai, 1990] E.W. Bai and D. Li, 1990. *Convergence of the Iterative Hammerstein System Identification Algorithm*, Automatica, Vol. 26, N° 4, pp. 651-677.

[Bai, 2002] E. W. Bai, 2002. *A blind approach to the Hammerstein-Wiener model identification*, Automatica, Vol. 38, N° 6, pp. 967-979.

[Bai, 2003] E. W. Bai, 2003. *Frequency domain identification of Hammerstein models*, IEEE Transactions on Automatic Control, Vol. 48, pp. 530-542.

[Bai, 2008] E. W. Bai, 2008. *Identification of an additive nonlinear system and its applications in generalized Hammerstein model*, Automatica, Vol. 44, pp. 430-436.

[Bariani, 1988] J. P. Bariani, 1988, *Conception et réalisation d'un logiciel de CAO en automatique : identification structurelle et commande PID*, Thèse de 3eme cycle, université de NICE.

[Bauer, 2002] D. Bauer and B. Ninness, 2002. *Asymptotic properties of least-squares estimates of Hammerstein-Wiener models*, International Journal of Control, Vol. 75, N° 1, pp. 34-51.

[Bazaraa, 1993] M. S. Bazaraa, H. D. Sherali and C. M. Shetty, 1993. *Nonlinear Programming theory and algorithms* – second edition, John Wiley and Sons, Inc.

[Ben Abdennour, 2001] R. Ben Abdennour, P. Borne, M. Ksouri et F. M'Sahli, 2001, *Identification et commande numérique des procédés industriels*, Edition Technip, Paris.

[Billings, 1981] S. A. Billings and I. J. Leontaritis, 1981. *Identification of nonlinear systems using parameter estimation techniques*, in Proc. IEE Conference on Control Application, Warwick, U.K, pp. 183-187.

[Borne, 1997] P. Borne, 1997. *Analyse et régulation des processus industriels*, Eyrolles.

[Bortolot, 2005] Z. J. Bortolot and R. H. Wynne, 2005. *Estimating forest biomass using small foot print LiDAR data: An individual tree-based approach that incorporates training data*, ISPRS Journal of Photogrammetry and Remote Sensing, N° 59, pp. 342-360.

[Boucher, 1996] P. Boucher et D. Dumur, 1996. *La commande prédictive*, Edition Technip.

[Boujmil, 1991] M. H. Boujmil, 1991, *Identification paramétrique et structurelle des systèmes non linéaires discrets, déterministes, stochastiques et monovariables*, DEA, ENSET de Tunis.

[Boyd, 1985] S. Boyd, L. Chua, 1985. *Fading Memory and the Problem of Approximating Nonlinear Operators with Volterra Series*, IEEE Transactions on Circuits and Systems, Vol. CAS-32, N° 11.

[Brent, 1973] R. P. Brent, 1973. *Algorithms for Minimization without Derivatives*, Prentice-Hall.

[Camacho, 1999] E. F. Camacho et C. Bordons, 1999. *Model Predictive Control*, Springer-Verlag, London, 1999. ISBN 3-540-76241-8.

[Chelouah, 2000] R. Chelouah and P. Siarry, 2000. *A continuous genetic algorithm designed for the global optimization of multimodal functions*, Journal of Heuristics, N° 6, pp. 191-213.

[Chen, 1999] L. Chen and K. S. Narendra, 1999. *On a new Approach to the design of tracking controllers for nonlinear dynamical systems*, Proceedings of the American Control Conference Sans Diego, California, pp. 3534-3538.

[Cherruault, 1999] Y. Cherruault, 1999. *Optimisation: Méthodes Locales et Globales*", Presses Universitaires de France, ISBN 2-130-49910-4.

[Clarke, 1979] D. W. Clarke, P. J. Gawthop. 1979. *Self-tuning control*, Proceedings IEEE, Vol. 126, pp. 633-640.

[Clarke, 1987] D. W. Clarke, C. Mohtadi et P.S. Tuffs. 1987. *Generalized predictive control- part I et II. Automatica*, Vol. 23, N° 2, pp. 137-160.

[Clarke, 1988] D. W. Clarke, 1988. *Application of Generalized Predictive Control to Industrial Processes*, IEEE control systems Magazine, Vol. 8, pp. 49-55.

[Clarke, 1989] D. W. Clarke and T. Mohtadi, 1989. *Properties of Generalized Predictive Control*, Automatica, Vol. 25, N° 6, pp. 859-875.

[Culioli, 1994] J. C. Culioli, 1994. *Introduction à l'Optimisation*, Ellipses, ISBN 2-729-89428-4.

[Cutler, 1980] C. R. Cutler, B. C. Ramaker, 1980. *Dynamic matrix control - a computer control algorithm*, Automatic Control Conference. San Fransisco, Californie.

[Filali, 2002] S. Filali, K. Kemih and A. Kias, 2002. *Constrained predictive control using gradient method*, Advances in Modelling Analysis -C- AMSE Journal, Vol. 57, N° 3, pp. 35-46.

[Fletcher, 1987] R. Fletcher, 1987. *Practical Methods of Optimization*, John Wiley & Sons, ISBN 0-471-49463-1.

[Fnaich, 2000] Fnaich F., 2000. *Algorithmes Génétiques Application à l'Automatique*, CA2I'2000, 19,20 et 21 Février, Hammamet, Tunisie.

[Gesing, 1979] W. Gesing and E. J. Davison, 1979. *An Exact Penalty Function Algorithm for Solving General Constrained Parameter Optimization Problems*. *Automation,* Vol. 15, pp. 175-188. Pergamon Press Ltd.

[Giannakis, 2001] G. B. Giannakis and E. Serpedin, 2001. *A bibliography on nonlinear system identification*, Signal Processing, Vol. 81, pp. 533-580.

[Glover, 1989] F. Glover, 1989. *Tabu Search - Part I*, ORSA Journal on Computing, Vol. 1, N° 3, pp. 190-206.

[Glover, 1990] F. Glover, 1990. *Tabu Search - Part II*, ORSA Journal on Computing, Vol. 2, N° 1, pp. 4-32.

[Goldberg, 1989] D. E. Goldberg, 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, ISBN 0-201-15767-5.

[Greblicki, 2000] W. Greblicki, 2000. *Continuous time Hammerstein system identification*, IEEE Transactions on Automatic Control, Vol. 45, pp. 1232-1236.

[Haber, 1990] R. Haber, and H. Unbehauen, 1990. *Structure identification of Non linear systems–A survey on Input/output Approaches*, Automatica, Vol. 26, N° 4, pp. 651-677.

[Hazem, 2004] M. A. Hazem and M. M. Bayoumi, 2004. *Volterra system identification using adaptive genetic algorithms*, Applied Soft Computing, N° 5, pp. 75-86.

[Hazem, 2005] M. A. Hazem and M. M. Bayoumi, 2005. *An adaptive evolutionary algorithm for Volterra system identification*, Pattern Recognition Letters N° 26, pp. 109-119.

[Hernando, 1988] D. Hernando and A. A. Desrochers, 1988. *Modelling of Nonlinear Discrete-time Systems from Input-Output Data*, Automatica, Vol. 24, N° 5, pp. 629-641.

[Holland, 1975] J. H. Holland, 1975. *Adaptation in Natural and Artificial System*, The University of Michigan Press, ISBN 0-472-08460-7.

[Hu, 1992] N. Hu, 1992. *Tabu Search Method with random moves for globally optimal design*, International Journal for Numerical Methods in Engineering, Vol. 35, N° 5, pp. 1055-1070.

[Hunter, 1986] I. W. Hunter and M. J. Korenberg, 1986. *The identification of nonlinear biological systems: Wiener and Hammerstein cascade models*, Biolog. Cybernet., Vol. 55, pp. 135-144.

[Kelley, 1999] C. T. Kelley, 1999. *Detection and Remediation of Stagnation in the Nelder-Mead. Algorithm using a sufficient decrease condition*, SIAM journal on optimization. Vol. 1, pp 43-55, http : // epubs.siam.org/sam-bin/dbq/article/31520.

[Kenneth, 1999] E. Kenneth, J. Jer-Nan and S. Richard, 1999. *Direct adaptive predictive control using gradient descent*, The Journal of the Acoustical Society of America, Vol. 105, Issue 2, February 1999, p.1300.

[Ki, 1997] H. Ki and J. Cohen, 1997. *Linear and Nonlinear ARMA Model Parameter Estimation Using an Artificial Neural Network*, IEEE Trans. on Biomedical Engineering, Vol. 44, N° 3, pp. 168-174.

[Kirkpatrick, 1983] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, 1983. *Optimization by Simulated Annealing*, Science Magazine, Vol. 220, N° 4598, pp. 671-680.

[Kortmann, 1988] M. Kortmann and H. Unbehauen, 1988. *Structure detection in the identification of non linear systems*, APII, N° 22, pp. 5-25.

[Kurpati, 2002] A. Kurpati, S. Azarm and J. Wu, 2002. *Constraint handling improvements for multiobjective genetic algorithms*, Structural and Multidisciplinary Optimization Revue, Springer-Verlag, Vol. 23, pp. 204-213.

[Labrousse, 1977] C. Labrousse, 1977. *Statistique,Tome 3*, Edition Dunod, Paris.

[Landau, 1988] I. D. Landau, 1988. *Identification et commande des systèmes*, édition Hermès.

[Leontaritis, 1985a] I. J. Leontaritis and S. A. Billings, 1985. *Input–output parametric models for nonlinear systems—Part I: Deterministic nonlinear systems*, International Journal of Control, Vol. 41, N° 2, pp. 303-328.

[Leontaritis, 1985b] I. J. Leontaritis and S. A. Billings, 1985. *Input-Output Parametric Models for Nonlinear Systems—Part II: Stochastic Nonlinear Systems*, International Journal of Control, Vol. 41, N° 2, pp. 329-344.

[Li, 1993] C. J. Li and Y. C. Jeon, 1993. *Genetic Algorithm in Identifying Nonlinear Autoregressive with Exogenous Input Models for Nonlinear Systems*, Proceeding of American Control Conference, San Francisco, CA, June 2-4, IEEE, Piscataway, NJ, pp. 2305-2309.

[Li, 1994] C. J. Li and Y. C. Jeon, 1994. *A Learning Controller Based on Non-Linear ARX Inverse Model Identified by a Genetic Algorithm*, Proceeding of Symposium on Intelligent Process Control in ASME International Mechanical Engineering Congress and Exposition, Chicago, IL, Nov. 6-11, Vol. 55, N° 1, pp 447-458.

[Liu, 2003] W. Liu, L. Zhengpei, L. Fu and W. Yaqi, 2003. *A systematic method to identify nonlinear dynamics of BWR by using the reactor noise*, Progress in Nuclear Energy, Vol. 43, N° 1-4, pp. 209-216.

[Luersen, 2004] M. A. Luersen and R. Le Riche, 2004. *Globalized Nelder-Mead method for engineering optimization*, Computers and Structures, N° 82, pp. 2251-2260.

[Luh, 1998] G. C. Luh and G. Rizzoni, 1998. *Nonlinear System Identification Using Genetic Algorithms with Application to Feedforward Control Design*, Proceedings of the American Control Conference, Philadelphia, Pennsylvania, June1998, pp. 2371-2375.

[Madár, 2005] J. Madár, J. Abonyi and F. Szeifert, 2005. *Genetic programming for the identification of nonlinear input-output models*, Industrial and Engineering Chemistry Research, Vol. 44, N° 9, pp. 3178-3186.

[Man, 1996] K. F. Man , K.S. Tang and S. Kwong, 1996. *Genetic algorithms: Concepts and Applications in engineering design*, IEEE Transactions on Industrial Electronics, Vol. 43, N° 5, pp. 519-533.

[Mrabet, 2002] M. Mrabet, F. Fnaiech, A. Chaari and K. Al-Haddad, 2002. *Nonlinear Predictive Control Based on NARX Models with Structure Identification,* IECON 02, 28th Annual Conference of the Industrial Electronics Society, IEEE 2002. Vol. 3, pp. 1757-1762.

[Morari, 1994] M. Morari, 1994. *Model Predictive Control*: *Multivariable Control Technique of Choice in the 1990s?*, Advances in Model-Based Predictive Control. Oxford University Press.

[Narendre, 1966] K. S. Narenda and P. Gallman, 1966. *An iterative method for the identification of nonlinear systems using a Hammerstein model*, IEEE Transactions on Automatic Control, Vol. AC-11, pp. 546-550.

[Narendra, 1990] K. S. Narendra and K. Parthasarathy, 1990. *Identification and control of dynamical systems using neural networks*, IEEE Transactions on Neural Networks, Vol. 1, N° 1, pp. 4-27.

[Narendra, 1997] K. S. Narendra, and S. M. Mukhopadhyay, 1997. *Adaptive Control Using Neural Networks and Approximate Models*, IEEE Transactions on Neural Networks, Vol. 8, N° 3, pp. 475-485.

[Nelder, 1965] J. A. Nelder, R. Mead, 1965. *A Simplex Method for Function Minimization*, Computer Journal, Vol. 7, pp. 308-312.

[Ong, 2005] C. S.Ong, J. J. Huang and G. H. Tzeng, 2005. *Model identification of ARIMA family using genetic algorithms*, Applied Mathematics and Computation, Vol. 164, Issue 3, pp. 885-912.

[Petlenkov, 2007] E. Petlenkov, 2007. *NN-ANARX Structure Based Dynamic Output Feedback Linearization for Control of Nonlinear MIMO Systems*. Proceedings of the 15th Mediterranean Conference on Control and Automation, Athens-Greece.

[Powell, 1965] M. J. D. Powell, 1965. *An efficient method for finding the minimum of a function of several variables without calculating derivations*, Computer Journal, Vol. 7, pp. 155-162.

[Press, 1992] W. H. Press, 1992. *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, ISBN 0-521-43108-5.

[Propoi, 1963] A. I. Propoi., 1963. *Use of linear programming methods for synthesizing sampled-data automatic systems*. Automation and Remote Control, Vol. 24, pp. 837-844.

[Pukkila, 1988] T. M. Pukkila and P. R. Krishnaiah, 1988. *On the use of autoregressive order determination criteria in Univariate white noise*

tests, IEEE Transactions on Acoustic Speech and Signal Processing, Vol. ASSP-36, pp. 764-774.

[Qin, 1997] S. J. Qin, and T.J. Badgwell, 1997. *An overview of industrial model predictive control technology*, Chemical Process Control-V, edited by Jeffrey C. Kantor, Carlos E. Garcia and Brice Carnahan, pp.232-256, Tahoe, California.

[Rao, 1996] S. S. Rao, 1996. *Engineering Optimization: Theory and Practice*, John Wiley & Sons, ISBN 0-471-55034-5.

[Richalet, 1976] J. Richalet, A. Rault, J.L. Testud and J. Papon, 1976. *Algorithmic control of industrial processes*, 4th IFAC Symposium on Identification and System Parameter Estimation. Tribilsi, URSS.

[Richalet, 1978] J. Richalet, A. Rault, J.L. Testud and J. Papon, 1978. *Model Predictive Heuristic Control : Application to Industrial processes*, Automatica, Vol. 14, N° 2, pp. 413-428.

[Rosenbrock, 1960] H. H. Rosenbrock, 1960. *An automatic method for finding the greatest or least value of a function*, Computer Journal, N° 3, pp. 175-184.

[Ruano, 2003] A. E.Ruano, P. J. Fleming, C. Teixeira, K. Rodriguez-Vazquez and C. M. Fonseca, 2003. *Nonlinear identification of aircraft gas-turbine dynamics*, Neurocomputing Journal, Vol. 55, pp. 551-579.

[Sheng, 2001] L. Sheng, K. H. Ju and K. H. Chon, 2001. *A new Algorithm for linear and Nonlinear ARMA Model parameter estimation using Affine Geometry*, IEEE Transactions on Biomedical Engineering, Vol. 48, N° 10, pp. 1116-1124.

[Sheng, 2003] L. Sheng and K. H. Chon, 2003. *Nonlinear Autoregressive and Nonlinear Autoregressive Moving Average Model parameter estimation by Minimising Hyper surface Distance*, IEEE Transactions on Signal processing, Vol. 51, N° 12, pp. 3020-3026.

[Song, 2006] F. Song and P. Li, 2006. *MIMO Decoupling Control Based on Support Vector Machines αth-order Inversion,* Proceeding of the 6th World Congress on Intelligent Control and Automation. Dalian, China, pp. 1002-1006.

[Wen, 2004] Y. Wen and L. Xiaoou, 2004. *Fuzzy identification Using Fuzzy Neural Networks with stable learning algorithms.* IEEE Transactions on Fuzzy Systems, Vol. 12, N° 3, pp. 411-420.

[Westwick, 1996] D. Westwick, M. Verhaegen, 1996. *Identifying MIMO Wiener systems using subspace model identification method*, Signal Processing Journal, Vol. 52, pp. 235-258.

[Wigren,1994] T. Wigren, 1994. *Convergence analysis of recursive identification algorithm based on the nonlinear Wiener model*, IEEE Transactions on Automatic Control, Vol. 39, pp. 2191-2206.

[Wright, 1996] M. H. Wright, 1996. *Direct Search Methods : Once Scorned, Now Respectable". Numerical Analysis 1995,* proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis, Edition D.F. Griffiths and G.A. Watson, Vol. 344, pp. 191-208.

[Wright, 1998] M. H. Wright, 1998. *Optimization methods for base station placement in wireless systems*, Proceedings of the IEEE VTC'98 Conference, pp. 387-391.

[Yang, 2005] X. H. Yang, Z. F. Yang, G. H. Lu and J. Q. Li, 2005. *A grey-encoded, hybrid-accelerated, genetic algorithm for global optimizations in dynamical systems*, Communications in Nonlinear science and Numerical Simulation, N° 10, pp. 355-363.

[Yang, 2000] Z. J. Yang, T. Fujimoto and K. Kumamaru, 2000. *A Genetic Algorithm Approach to Identification of Nonlinear Polynomial Models*, IFAC System Identification, Santa Barbara, California, USA, 2000.

[Yonghong, 1996] T. Yonghong, A. R. Van Cauwenberghe, 1996. *Optimization techniques for the design of a neural predictive controller*, Journal Neurocomputing, Vol. 10, Issue 1, pp. 83-96.

[Zhu, 1999] Y. Zhu, 1999. *Distillation column identification for control using Wiener model*, Proceeding of American Control Conference 1999, pp. 3462-3466.

[Zhu, 2002] Y. Zhu, 2002. *Estimation of an N-L-N Hammerstein-Wiener model*, Automatica, Vol. 38, N° 9, pp. 1607-1614.

Liste des publications

Conférences internationales **:**

- *Brahim Tlili, Faouzi Bouani*, ‘’***Commande prédictive non linéaire’’***, *Conférence Internationale Francophone d'Automatique (CIFA'2004)*, Novembre 2004, DOUZ, Tunisie, Art. N°285.
- *Brahim Tlili, Faouzi Bouani, Mekki Ksouri*, ‘’***Estimation des paramètres du modèle NARMA à l'aide des réseaux de neurones et des algorithmes génétiques’’***, *Actes de la 6° conférence francophone de MOdélisation et SIMulation.* Avril 2006, *Rabat, Maroc*. Lavoisier, Doc et Tech. Vol.1, pp.47-53, Mars 2006. http://www.isima.fr/mosim06/actes/articles/1-Metaheuristiques/89.pdf.
- *Brahim Tlili, Faouzi Bouani, Mekki Ksouri*,***‘’A derivative-free constrained predictive controller’’***, *Proceedings of the 10th WSEAS International Conference on Systems*, Vouliagmeni, Athens, Greece, July 10-12, 2006 (pp:358-363).
- *Brahim Tlili, Faouzi Bouani, Mekki Ksouri*, ‘***’ Identification of multivariable NARMA models using Artificial Neural Networks’’,*** *Industrial Simulation Conference ISC 2008,* Lyon, France, June 9-11, 2008. pp. 40-44.

Revues scientifiques **:**

- *Brahim Tlili, Faouzi Bouani, Mekki Ksouri*, ***‘’Constrained model predictive control using a derivative-free optimization method’’***, *WSEAS Transactions on Systems*, Issue 10, Vol. 5, October 2006 (pp:2307-2313). http://www.worldses.org/journals/systems/systems-october2006.doc
- *Brahim Tlili, Faouzi Bouani, Mekki Ksouri*, ‘***’ Identification des systèmes non linéaires par des modèles de type NARMA’’,*** *à paraître au Journal Européen des Systèmes Automatisés* JESA. Vol. 9, Novembre 2008.

ANNEXES

ANNEXE 1

1) Le F_Test ou le test de Fisher :

Le F_Test est utilisé pour comparer deux variances observées afin de savoir si elles diffèrent entre elles de manière significative ou non.

La loi de Fisher-Snedecor à n_1 et n_2 degrés de liberté [Labrousse, 1997]

a) Définition

Soient S^2_1 et S^2_2 deux variances de deux variables aléatoires indépendantes X et Y, avec $X = (x_1, x_2, \ldots, x_i, \ldots, x_{n1})$ *et* $Y = (y_1, y_2, \ldots, y_i, \ldots, y_{n2})$

La quantité :

$$F = \frac{S^2_1 \ / \mathrm{n}_1}{S^2_2 \ / \mathrm{n}_2} \qquad \text{(A1.1)}$$

est appelée variable aléatoire de Fisher à n_1 et n_2 degrés de liberté.

b) Tables de la Loi de Fisher

Les tables permettent de déterminer par simple inspection, pour α donné, et pour n_1 et n_2 degrés de liberté la valeur X_α telle que la probabilité de F supérieure à X_α est égale à α (Pr ($F > X_\alpha$) = α), avec α un seuil de signification. En général, on choisit α = 0.05. Ces tables sont données dans le paragraphe 2 de cet annexe.

c) Principe du test

Il s'agit d'un test statistique obtenu en utilisant un raisonnement par l'absurde :

- On suppose que S^2_1 et S^2_2 ne diffèrent pas entre elles de manière significative.
- Sous cette hypothèse (appelée hypothèse nulle H_0) on peut déterminer, au seuil α, un intervalle **I** de confiance pour F.
- Si l'hypothèse était juste, la valeur de F a une probabilité α d'être à l'extérieur de l'intervalle.
- Si l'on constate que $F \in \mathrm{I}$ cela est compatible avec l'hypothèse donc on l'accepte.
- Si par contre l'on constate que $F \notin \mathrm{I}$ alors l'hypothèse faite n'a que α chance sur 100 d'être vraie. Si α est suffisamment petit, on peut, avec un risque faible, rejeter l'hypothèse nulle (H_0).

d) Différentes étapes du F_Test

On va appliquer les différentes étapes du *F_Test* sur un critère utilisé pour estimer l'ordre exact n d'un système. Ce critère est défini comme suit [Bariani, 1988], [Boujmil, 1991]:

Soient $V(n)$: la variance du résidu pour l'ordre n,
$V(\hat{n}_1)$: la variance du résidu pour l'ordre $\hat{n}_1$,
$V(\hat{n}_2)$: la variance du résidu pour l'ordre $\hat{n}_2$.

Le critère est basé sur l'indépendance statistique des quantités $V(\hat{n}_1)$ *et* $[V(\hat{n}_1) - V(\hat{n}_2)]$.
Pour $\hat{n}_2 > \hat{n}_1 \geq n$, la quantité :

$$F = \frac{V(\hat{n}_1) - V(\hat{n}_2)}{V(\hat{n}_2)} \frac{N - \hat{n}_2}{\hat{n}_2 - \hat{n}_1} \qquad \text{(A1.2)}$$

a une distribution de Fisher à ($\hat{n}_2 - \hat{n}_1$) et (N-$\hat{n}_2$) degrés de liberté avec N est le nombre de mesures et n_1 est le nombre de paramètres du modèle.

Nous pouvons donc établir six étapes du *F_Test* utilisées pour estimer l'ordre n :

Etape 1 *Définir les hypothèses*

H_0 est l'hypothèse nulle : n_1 étant l'ordre exact du modèle (c'est-à-dire il n'y a pas de différence significative entre les deux variances).

Etape 2 *Définir un paramètre qui sous l'hypothèse nulle obéit à une loi connue*.

$$F = \frac{V^N(\hat{n}_1) - V^N(\hat{n}_2)}{V^N(\hat{n}_2)} \cdot \frac{N - \hat{n}_2}{\hat{n}_2 - \hat{n}_1} \qquad \text{(A1.3)}$$

Etape 3 *Définir un seuil α que l'on appellera seuil de signification, soit α = 0.05*.

Etape 4 *Définir une région critique associée à ce seuil* :
Par exemple pour α = 0.05 on lit sur les abaques de Fisher à ($n_1 = 1$) et ($n_2 > 120$) degrés de liberté une valeur $X\alpha = 3.84$.

Etape 5 *Calculer la valeur du paramètre en fonction des données du problème*, soit F cette valeur.

Etape 6 *Décider*
Si F ≤ 3.84 on accepte H_0 avec un risque d'erreur inférieur ou égal à 5% (c'est-à-dire que $n_1 = n$, l'ordre exact recherché). Les écarts entre les deux variances sont dus au hasard.

2) Table de Fisher-Snedecor

TABLE DU F DE BEHRENS-FISHER-SNEDECOR

La table donne, en fonction des degrés de liberté ν_1 et ν_2, la valeur de F_α ayant la probabilité α d'être dépassée :

$$\alpha = Pr\{F \geq F_\alpha\}, \quad \text{avec} \quad F = \frac{s_1^2}{s_2^2}.$$

La variance s_1^2 est *nécessairement* plus grande que la variance s_2^2.

ν_2 \ ν_1		1		2		3		4		5	
	α=	0,05	0,01	0,05	0,01	0,05	0,01	0,05	0,01	0,05	0,01
1		161,4	4 052	199,5	4 999	215,7	5 403	224,8	5 625	230,2	5 764
2		18,51	98,49	19,00	99,00	19,16	99,17	19,25	99,25	19,30	99,30
3		10,13	34,12	9,55	30,81	9,28	29,46	9,12	28,71	9,01	28,24
4		7,71	21,20	6,94	18,00	6,59	16,69	6,39	15,98	6,26	15,52
5		6,61	16,26	5,79	13,27	5,41	12,06	5,19	11,39	5,05	10,97
6		5,99	13,74	5,14	10,91	4,76	9,78	4,53	9,15	4,39	8,75
7		5,59	12,25	4,74	9,55	4,35	8,45	4,12	7,85	3,97	7,45
8		5,32	11,26	4,46	8,65	4,07	7,59	3,84	7,01	3,69	6,63
9		5,12	10,56	4,26	8,02	3,86	6,99	3,63	6,42	3,48	6,06
10		4,96	10,04	4,10	7,56	3,71	6,55	3,48	5,99	3,33	5,64
11		4,84	9,65	3,98	7,20	3,59	6,22	3,36	5,67	3,20	5,32
12		4,75	9,33	3,88	6,93	3,49	5,95	3,26	5,41	3,11	5,06
13		4,67	9,07	3,80	6,70	3,41	5,74	3,18	5,20	3,02	4,86
14		4,60	8,86	3,74	6,51	3,34	5,56	3,11	5,03	2,96	4,69
15		4,54	8,68	3,68	6,36	3,29	5,42	3,06	4,89	2,90	4,56
16		4,49	8,53	3,63	6,23	3,24	5,29	3,01	4,77	2,85	4,44
17		4,45	8,40	3,59	6,11	3,20	5,18	2,96	4,67	2,81	4,34
18		4,41	8,28	3,55	6,01	3,16	5,09	2,93	4,58	2,77	4,25
19		4,38	8,18	3,52	5,93	3,13	5,01	2,90	4,50	2,74	4,17
20		4,35	8,10	3,49	5,85	3,10	4,94	2,87	4,43	2,71	4,10
21		4,32	8,02	3,47	5,78	3,07	4,87	2,84	4,37	2,68	4,04
22		4,30	7,94	3,44	5,72	3,05	4,82	2,82	4,31	2,66	3,99
23		4,28	7,88	3,42	5,66	3,03	4,76	2,80	4,26	2,64	3,94
24		4,26	7,82	3,40	5,61	3,01	4,72	2,78	4,22	2,62	3,90
25		4,24	7,77	3,38	5,57	2,99	4,68	2,76	4,18	2,60	3,86
26		4,22	7,72	3,37	5,53	2,98	4,64	2,74	4,14	2,59	3,82
27		4,21	7,68	3,35	5,49	2,96	4,60	2,73	4,11	2,57	3,78
28		4,20	7,64	3,34	5,45	2,95	4,57	2,71	4,07	2,56	3,75
29		4,18	7,60	3,33	5,42	2,93	4,54	2,70	4,04	2,54	3,73
30		4,17	7,56	3,32	5,39	2,92	4,51	2,69	4,02	2,53	3,70
40		4,08	7,31	3,23	5,18	2,84	4,31	2,61	3,83	2,45	3,51
60		4,00	7,08	3,15	4,98	2,76	4,13	2,52	3,65	2,37	3,34
120		3,92	6,85	3,07	4,79	2,68	3,95	2,45	3,48	2,29	3,17
∞		3,84	6,64	2,99	4,60	2,60	3,78	2,37	3,32	2,21	3,02

TABLE DU F DE BEHRENS-FISHER-SNEDECOR

ν_2 \ ν_1		6		8		12		24		∞	
	α=	0,05	0,01	0,05	0,01	0,05	0,01	0,05	0,01	0,05	0,01
1		234,0	5 859	238,9	5 981	243,9	6 106	249,0	6 234	254,3	6 366
2		19,33	99,33	19,37	99,36	19,41	99,42	19,45	99,46	19,50	99,50
3		8,94	27,91	8,84	27,49	8,74	27,05	8,64	26,60	8,53	26,12
4		6,16	15,21	6,04	14,80	5,91	14,37	5,77	13,93	5,63	13,46
5		4,95	10,67	4,82	10,27	4,68	9,89	4,53	9,47	4,36	9,02
6		4,28	8,47	4,15	8,10	4,00	7,72	3,84	7,31	3,67	6,88
7		3,87	7,19	3,73	6,84	3,57	6,47	3,41	6,07	3,23	5,65
8		3,58	6,37	3,44	6,03	3,28	5,67	3,12	5,28	2,93	4,86
9		3,37	5,80	3,23	5,47	3,07	5,11	2,90	4,73	2,71	4,31
10		3,22	5,39	3,07	5,06	2,91	4,71	2,74	4,33	2,54	3,91
11		3,09	5,07	2,95	4,74	2,79	4,40	2,61	4,02	2,40	3,60
12		3,00	4,82	2,85	4,50	2,69	4,16	2,50	3,78	2,30	3,36
13		2,92	4,62	2,77	4,30	2,60	3,96	2,42	3,59	2,21	3,16
14		2,85	4,46	2,70	4,14	2,53	3,80	2,35	3,43	2,13	3,00
15		2,79	4,32	2,64	4,00	2,48	3,67	2,29	3,29	2,07	2,87
16		2,74	4,20	2,59	3,89	2,42	3,55	2,24	3,18	2,01	2,75
17		2,70	4,10	2,55	3,79	2,38	3,45	2,19	3,08	1,96	2,65
18		2,66	4,01	2,51	3,71	2,34	3,37	2,15	3,00	1,92	2,57
19		2,63	3,94	2,48	3,63	2,31	3,30	2,11	2,92	1,88	2,49
20		2,60	3,87	2,45	3,56	2,28	3,23	2,08	2,86	1,84	2,42
21		2,57	3,81	2,42	3,51	2,25	3,17	2,05	2,80	1,81	2,36
22		2,55	3,76	2,40	3,45	2,23	3,12	2,03	2,75	1,78	2,31
23		2,53	3,71	2,38	3,41	2,20	3,07	2,00	2,70	1,76	2,26
24		2,51	3,67	2,36	3,36	2,18	3,03	1,98	2,66	1,73	2,21
25		2,49	3,63	2,34	3,32	2,16	2,99	1,96	2,62	1,71	2,17
26		2,47	3,59	2,32	3,29	2,15	2,96	1,95	2,58	1,69	2,13
27		2,46	3,56	2,30	3,26	2,13	2,93	1,93	2,55	1,67	2,10
28		2,44	3,53	2,29	3,23	2,12	2,90	1,91	2,52	1,65	2,06
29		2,43	3,50	2,28	3,20	2,10	2,87	1,90	2,49	1,64	2,03
30		2,42	3,47	2,27	3,17	2,09	2,84	1,89	2,47	1,62	2,01
40		2,34	3,29	2,18	2,99	2,00	2,66	1,79	2,29	1,51	1,80
60		2,25	3,12	2,10	2,82	1,92	2,50	1,70	2,12	1,39	1,60
120		2,17	2,96	2,01	2,66	1,83	2,34	1,61	1,95	1,25	1,38
∞		2,09	2,80	1,94	2,51	1,75	2,18	1,52	1,79	1,00	1,00

ANNEXE 2

0	*0.1008*	**44**	*0.0443*	**88**	*0.0129*	**132**	*-0.0694*	**176**	*-0.2917*
1	*0.8103*	**45**	*0.0911*	**89**	*0.0129*	**133**	*-0.0524*	**177**	*-0.0061*
2	*-0.3830*	**46**	*0.1078*	**90**	*0.0675*	**134**	*-0.0444*	**178**	*-0.0540*
3	*-0.0246*	**47**	*-0.1242*	**91**	*-0.0002*	**135**	*-0.0094*	**179**	*0.0820*
4	*-0.1490*	**48**	*-0.0498*	**92**	*-0.0411*	**136**	*-0.1216*	**180**	*-0.1573*
5	*0.5467*	**49**	*-0.0085*	**93**	*-0.0088*	**137**	*0.0021*	**181**	*0.0199*
6	*0.1016*	**50**	*-0.1170*	**94**	*-0.0013*	**138**	*0*	**182**	*0.0079*
7	*0.1382*	**51**	*0.1985*	**95**	*-0.0343*	**139**	*0.0043*	**183**	*-0.0125*
8	*-0.0765*	**52**	*0.0948*	**96**	*0.0426*	**140**	*-0.0080*	**184**	*-0.0161*
9	*-0.0194*	**53**	*-0.0352*	**97**	*0.0206*	**141**	*-0.0303*	**185**	*-0.1048*
10	*0.0094*	**54**	*0.0376*	**98**	*-0.0015*	**142**	*-0.0055*	**186**	*0.1340*
11	*0.0755*	**55**	*-0.0791*	**99**	*0.0030*	**143**	*-0.0260*	**187**	*0.0593*
12	*0.0926*	**56**	*0.0995*	**100**	*-0.0509*	**144**	*-0.0076*	**188**	*0.0979*
13	*0.0058*	**57**	*-0.1126*	**101**	*0.0722*	**145**	*-0.0039*	**189**	*0.0056*
14	*-0.1685*	**58**	*0*	**102**	*-0.0354*	**146**	*0.0257*	**190**	*-0.1437*
15	*0.1899*	**59**	*-0.0132*	**103**	*0.0150*	**147**	*0.0243*	**191**	*-0.2177*
16	*0.0631*	**60**	*0.0206*	**104**	*-0.0020*	**148**	*0.0231*	**192**	*-0.0714*
17	*-0.0204*	**61**	*-0.0043*	**105**	*0.0115*	**149**	*-0.0097*	**193**	*0.0318*
18	*-0.0221*	**62**	*-0.0078*	**106**	*-0.0040*	**150**	*-0.0299*	**194**	*0.0149*
19	*-0.0147*	**63**	*0.0028*	**107**	*-0.0012*	**151**	*0.0276*	**195**	*-0.0377*
20	*0.0130*	**64**	*-0.0034*	**108**	*0.0003*	**152**	*-0.0399*	**196**	*0.0449*
21	*-0.0117*	**65**	*-0.2038*	**109**	*0.0031*	**153**	*0.0096*	**197**	*0.0019*
22	*0.0009*	**66**	*0.0521*	**110**	*0.1239*	**154**	*0.0132*	**198**	*-0.0358*
23	*0.0193*	**67**	*-0.0282*	**111**	*-0.1606*	**155**	*-0.0096*	**199**	*0.0077*
24	*0.1151*	**68**	*0.0118*	**112**	*-0.0945*	**156**	*0.0121*	**200**	*0.0436*
25	*0.0052*	**69**	*0.0355*	**113**	*-0.0096*	**157**	*-0.0211*	**201**	*0.0893*
26	*0.0627*	**70**	*0.0162*	**114**	*-0.0137*	**158**	*0.0070*	**202**	*0.0190*
27	*0.0090*	**71**	*-0.0031*	**115**	*0.1104*	**159**	*-0.0004*	**203**	*-0.0023*
28	*-0.0119*	**72**	*0.0107*	**116**	*0.1694*	**160**	*0*	**204**	*-0.0018*
29	*0.0317*	**73**	*-0.0087*	**117**	*0.1526*	**161**	*-0.0058*	**205**	*0.0680*
30	*-0.1598*	**74**	*0.0019*	**118**	*0.0387*	**162**	*-0.0035*	**206**	*-0.0199*
31	*0.0738*	**75**	*-0.0550*	**119**	*-0.0424*	**163**	*0.0137*	**207**	*-0.0035*
32	*-0.0305*	**76**	*0.0386*	**120**	*0.1047*	**164**	*-0.0075*	**208**	*0.0164*
33	*0.0187*	**77**	*-0.0493*	**121**	*-0.0103*	**165**	*0.0189*	**209**	*-0.0094*
34	*0.0654*	**78**	*0.0203*	**122**	*-0.0135*	**166**	*0.0086*	**210**	*0.0086*
35	*0.0178*	**79**	*-0.0030*	**123**	*-0.0444*	**167**	*-0.0122*	**211**	*-0.0171*
36	*-0.0035*	**80**	*0.4138*	**124**	*0.0346*	**168**	*0*	**212**	*0.0142*
37	*-0.0085*	**81**	*-0.0321*	**125**	*0.0033*	**169**	*-0.0100*	**213**	*0.0066*
38	*0.0317*	**82**	*-0.0524*	**126**	*-0.0500*	**170**	*-0.0508*	**214**	*-0.0222*
39	*-0.0000*	**83**	*-0.0316*	**127**	*0.0680*	**171**	*0.3694*	**215**	*0.0085*
40	*-0.0542*	**84**	*-0.0845*	**128**	*-0.0047*	**172**	*0.2845*	**216**	*-0.0019*
41	*-0.1153*	**85**	*-0.0427*	**129**	*-0.0262*	**173**	*0.0607*	**217**	*0.0033*
42	*-0.0047*	**86**	*0.0109*	**130**	*0.0130*	**174**	*-0.0308*	**218**	*-0.0011*
43	*0.0296*	**87**	*-0.0589*	**131**	*0.0016*	**175**	*-0.1426*	**219**	*0.0016*

Tableau A2 : Les paramètres du modèle NARMA estimés par la méthode (RN), (système 4. Chapitre I).

ANNEXE 3

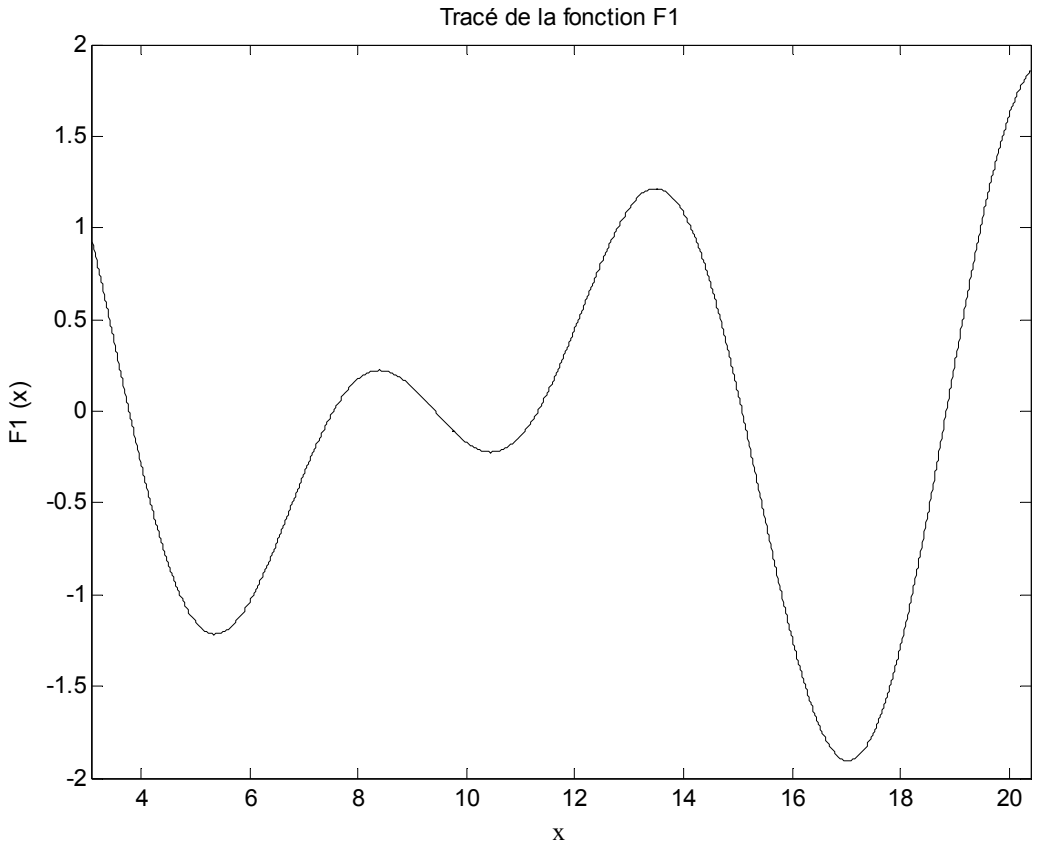
Liste des fonctions benchmarks

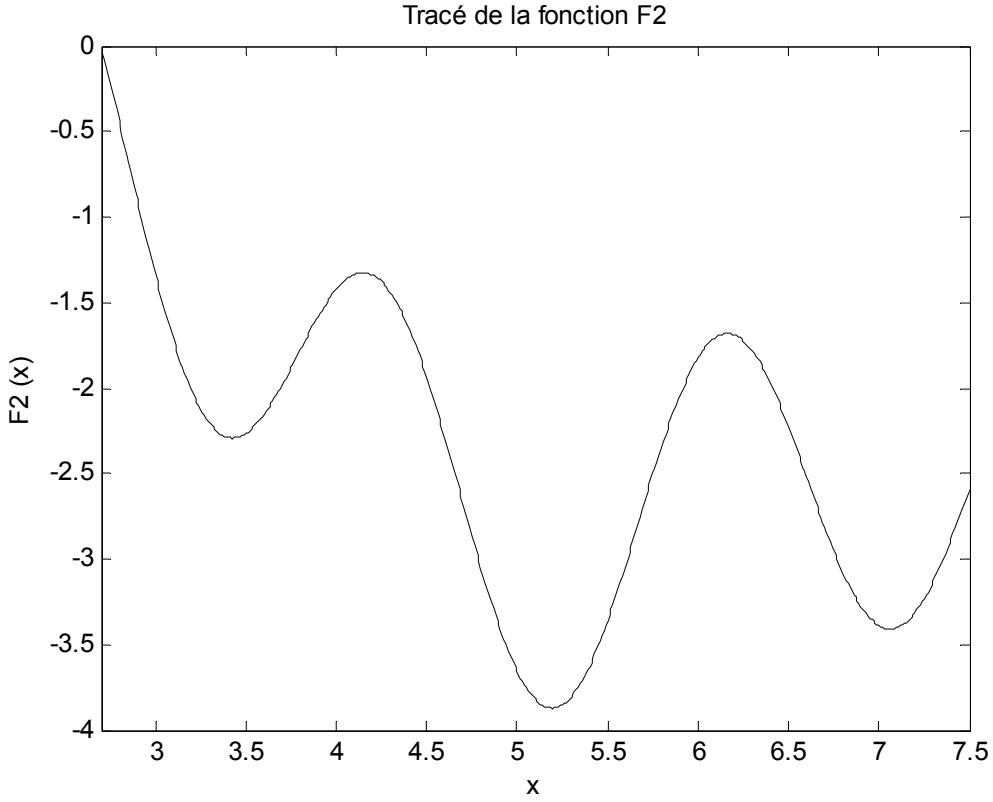
F1 (1 variable):
F1(x) = sin(x) + sin(2x/3)
Domaine de recherche: 3.1<x<20.4.
3 minima locaux
minimum global: x = 17.0393;*
F1(x) = -1.90596.*

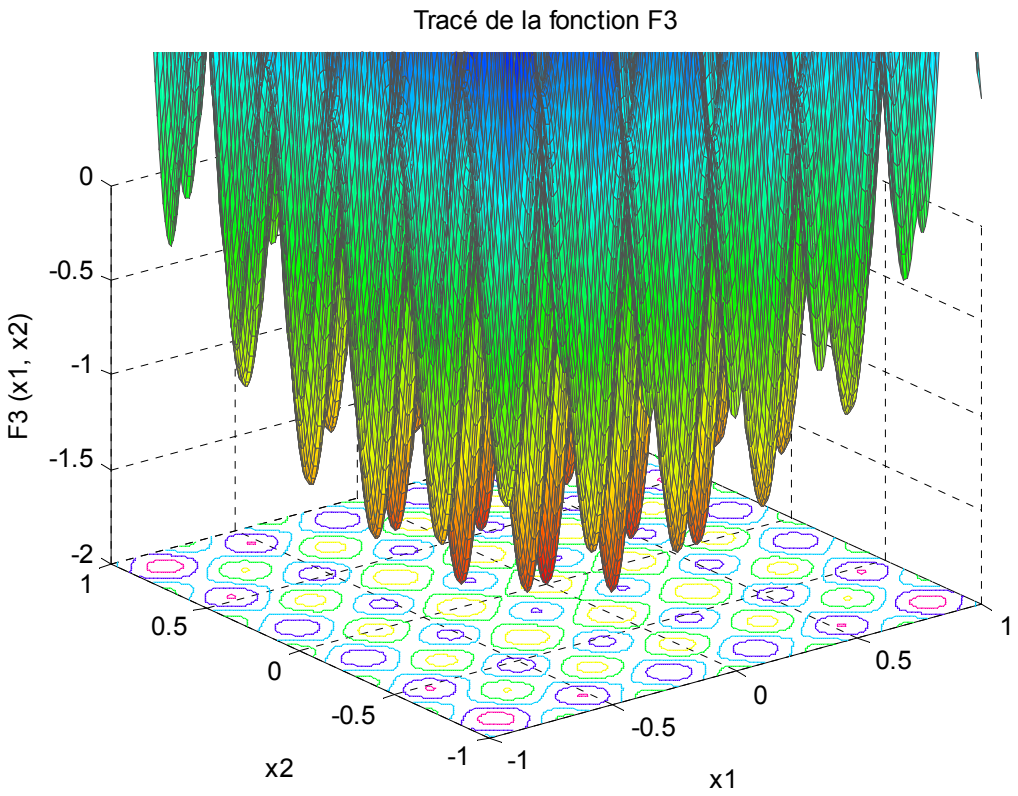
F2 (1 variable):
F2(x)=sin(x) + sin(10x/3) + Ln(x) – 0.84x
Domaine de recherche: 2.7<x<7.5.
3 minima locaux
1 minimum global: x = 5.19 ;*
F2(x) = -3.8717.*

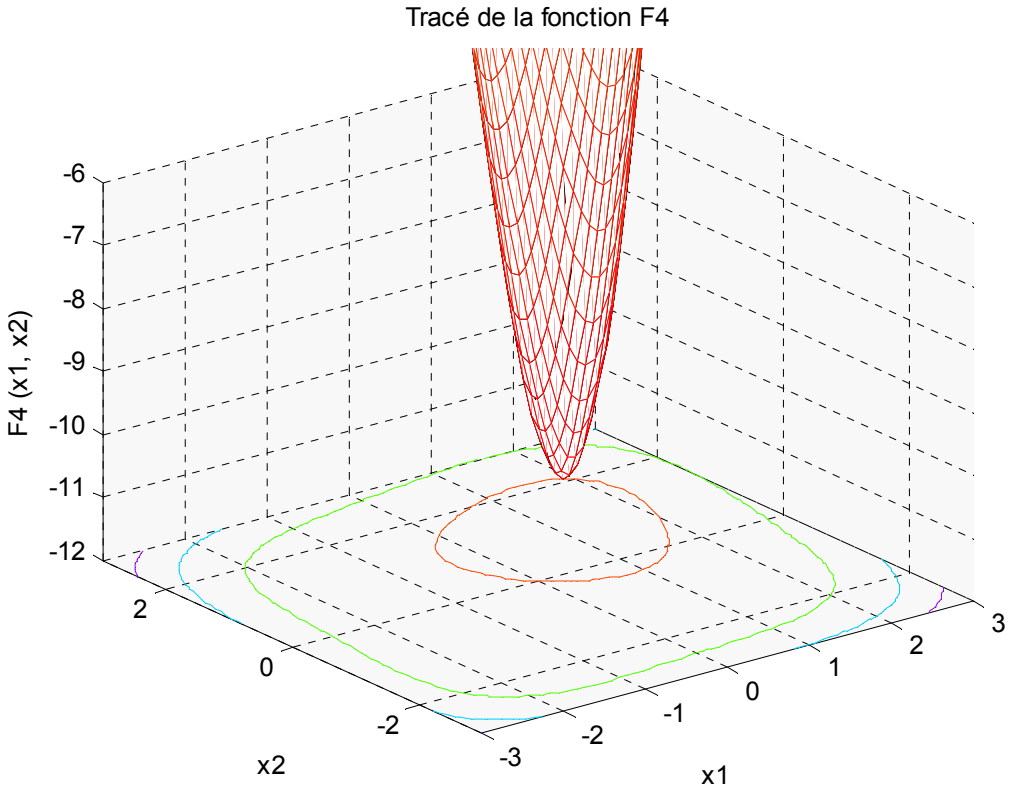
F3 (2 variables):
$F3(x_1,x_2) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$.
Domaine de recherche: $(-1,-1) < (x_1,x_2) < (1,1)$
A peut près 50 minima locaux
1 minimum global: $(x_1^*,x_2^*) = (0,0)$;
$F3(x_1^*,x_2^*) = -2$.

F4 (2 variables):
$F4(x_1,x_2) = (x_1^2 +1)^2 + (x_2^2+1)^2 - 2 (x_1 +x_2+1)^2$.
Domaine de recherche: $(-3,-3) < (x_1,x_2) < (3,3)$
1 minimum global: $(x_1^*,x_2^*) = (1.3247,1.3247)$;
$F4(x_1^*,x_2^*) = -11.45851$.


Tracé de la fonction F1
F1 (x)
x
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
4
6
8
10
12
14
16
18
20


Tracé de la fonction F2
F2 (x)
x
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5


Tracé de la fonction F3
F3 (x1, x2)
0
-0.5
-1
-1.5
-2
1
0.5
0
-0.5
-1
x2
-1
-0.5
0
0.5
1
x1


Tracé de la fonction F4
F4 (x1, x2)
-6
-7
-8
-9
-10
-11
-12
2
0
-2
-3
x2
-2
-1
0
1
2
3
x1