

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mentouri, Constantine
Faculté des sciences de l'ingénieur
Département d'Electronique

THESE

Présentée par

LAMAMRA KHEIREDDINE

Pour l'obtention du Diplôme de

DOCTORAT ES SCIENCES

Spécialité Electronique

Option Contrôle

THEME

**Optimisation multi-objectifs par les algorithmes
génétiques et application à la commande des
systèmes**

Date de Soutenance : 01/03/2012

Devant le Jury composé de :

Mr FILALI Salim	Professeur	Université de Constantine	Président
Mr BELARBI Khaled	Professeur	Université de Constantine	Rapporteur
Mr Dib Abderrahmane	Maitre de Conférences A	Université Oum El Bouaghi	Examineur
Mr MEZACHE Amar	Maitre de Conférences A	Université de Msila	Examineur
Mr BOUTAMINA Brahim	Maitre de Conférences A	Université de Constantine	Examineur



Dédicaces

A mes très chers parents

A ma femme

A ma très chère fille qui verra le jour

Incha-Allah dans quelques mois

A mes frères et mes sœurs

A mes neuneus et mes nièces

A ma grande famille

A mes Amis





Remerciements

*J'adresse mes vifs remerciements à **Monsieur Khaled Belarbi**, professeur à l'université Mentouri de Constantine à qui je témoigne ma gratitude pour son soutien et enthousiasme qu'il m'a manifestés pour la réalisation de cette thèse et aussi pour sa collaboration ainsi que ses conseils qu'il n'a jamais manqué de m'apporter.*

*Mes remerciements à **Monsieur Salim Filali**, professeur à l'université de Constantine d'avoir honoré par sa présidence du jury ainsi que tous les membres du jury d'avoir accepté d'examiner ce travail : **Monsieur Abderrahmane Dib**, maitre de conférences A, à l'université de Oum El Bouaghi, **Monsieur Amar Mezache**, maitre de conférences A, à l'université de Msila, **Monsieur Brahim Boutamina**, maitre de conférence A, à l'université de Constantine.*

Je remercie monsieur Ahmed Elhajjaji et monsieur Jérôme Bosche du laboratoire MIS (ex CREA) à l'université de Picardie -France, pour leurs aides et assistance durant ma période de stage.

Sans oublier les professeurs David Edward Goldberg et Carlos Artemio Coello Coello pour leurs aides et conseils.

Je tien à remercier aussi monsieur Nacereddine Meghezzi, pour son soutien et ses encouragements.

En fin je tiens à remercier tous ceux qui m'ont aidé de près et de loin pour l'accomplissement de ce travail.

Kheireddine Lamamra




Table des Matières

Dédicaces	i
Remerciements	ii
Résumé en Arabe, Français et Anglais	iii
Table des Matières	iv

Introduction générale

1. Introduction	1
2. Position du problème	2
3. Etat de l'art	2
3.1. Apprentissage des RN	2
3.2. Structure du RN	4
4. Description du travail	4
5. Organisation de la thèse	5

Chapitre 1

L'optimisation Multi-Objectifs par les Algorithmes Génétiques

1. Introduction	6
2. L'optimisation multicritères	6
3. Techniques d'optimisation multi-objectif par les algorithmes	9

évolutionnaires	
3.1. Techniques non Pareto	9
3.1.A. L'algorithme VEGA (Vector Evaluated Genetic Algorithm)	10
3.1.B. Techniques basées sur la transformation du problème multi-objectif en un problème mono-objectif	10
3.2. Techniques utilisant le concept de Pareto	11
3.2.1. Techniques non élitistes	11
3.2.2. Techniques élitistes	12
4. Programmation de l'algorithme génétique multi-objectif NSGA2	13
4.1. Principe de l'algorithme NSGA2	13
4.2. Organigramme de l'algorithme NSGA2	15
5. Conclusion	17

Chapitre 2

Les réseaux de neurones

1. Introduction	18
2. Le neurone artificiel	19
2.1. Le neurone biologique	19
2.2 Modèle d'un neurone artificiel	20
2.3 Fonctionnement d'un neurone artificiel	21
2.4 Fonction d'activation	22

3. Réseaux de neurones et architectures	23
3.1. Le choix de l'architecture du réseau de neurones	23
3.1.1. Les réseaux statiques	24
3.1.2. Les réseaux dynamiques	25
4. Les propriétés d'un réseau de neurones	26
4.1. La propriété d'approximation universelle	26
4.2. La propriété de parcimonie	26
4.3. L'apprentissage des réseaux de neurones	27
4.3.1. Types d'apprentissage	27
4.3.1.1. Le mode supervisé	27
4.3.1.2. Le renforcement	28
4.3.1.3. Le mode non supervisé (ou auto organisationnel)	28
4.3.1.4. Le mode hybride	28
5. Description des types de réseaux à entraîner	29
5.1. Le réseaux à perceptrons multicouches	29
5.2. Les réseaux de neurones à fonction de base radiale RBF	30
5.2.1. Architecture générale d'un réseau RBF	31
5.2.2. Apprentissage et placement des centres dans un réseau RBF	32
6. La commande des processus par les RN	32

Chapitre 3

Optimisation des réseaux de neurones avec et sans sélection des entrées par les algorithmes génétiques multi-objectifs

1. Introduction	34
2. Optimisation multi-objectifs des RN MLP	34
2.1. Optimisation à deux objectifs: structure du chromosome	35
2.2. Optimisation à trois objectifs: structure du chromosome	36
3. Optimisation multi-objectifs des RN RBF	38
3.1. Optimisation à deux objectifs: structure du chromosome	38
3.2. Optimisation à trois objectifs	40
4. Résultats d'application	40
4.1. Première partie : Application des réseaux MLP à la modélisation et la commande des systèmes	41
4.1.1. Application 1 : Modélisation du système de Box et Jenkins	41
4.1.1.a. Première approche : Optimisation de deux fonctions objectifs	41
4.1.1.b. Deuxième approche : Optimisation de trois fonctions objectifs	42
4.1.2. Application 2 : Modélisation d'un système chaotique (Application à la prédiction des séries temporelles du problème chaotique non linéaire de Mackey-Glass)	46
4.1.2.a. Première approche : Optimisation de deux fonctions objectifs	47
4.1.2.b. Deuxième approche : Optimisation de trois fonctions objectifs	47
4.1.3. Application 3 : Commande d'un processus à comportement variable simulé par un modèle d'état	51

4.1.3.1. Première approche : Optimisation de deux fonctions objectifs	51
4.1.3.2. Deuxième approche : Optimisation de trois fonctions objectifs	53
4.1.3.3. Résultats d'entraînement	54
4.1.3.4. Résultats de validation	56
4.1.3.5. Test de la robustesse	58
4.2. Deuxième partie : Application des réseaux RBF à la modélisation et la commande des systèmes	59
4.2.1. Application 1 : Modélisation du système de Box et Jenkins	59
4.2.1.a. Première approche : Optimisation de deux fonctions objectifs	59
4.2.1.b. Deuxième approche : optimisation de trois fonctions objectifs	60
4.2.2. Application 2 : Modélisation d'un système chaotique (Application à la prédiction des séries temporelles du problème chaotique non linéaire de Mackey-Glass)	62
4.2.2.1. Système M-G : Optimisation de deux fonctions objectifs	63
4.2.2.2. Deuxième approche : Optimisation de trois fonctions objectifs	63
4.2.3. Application 3 : Commande d'un processus à comportement variable simulé par un modèle d'état	66
4.2.3.1. Première approche : Optimisation de deux fonctions objectifs	66
4.2.3.2. Deuxième approche : Optimisation de trois fonctions objectifs	67
4.2.3.3. Résultats d'entraînement	67
4.2.3.4. Résultats de validation	70

4.2.3.5. Test de la robustesse	71
5. Comparaison des résultats (RN RBF et RN MLP)	71
6. Conclusion	73

Chapitre 4

Commande du système de freinage

Anti Blocage des Roues -ABS

1. Introduction	74
2. Présentation du système ABS	74
3. Le système ABS de laboratoire utilisé	75
3.1 Description	75
3.2 Modèle de simulation	76
4. Commande du système de freinage ABS	78
4.1. Conception de contrôleur neuronal	79
4.2. Résultats de simulation	80
4.3. Test de la poursuite	83
4.4. Test de robustesse	84
5. Conclusion	85
Conclusion Générale	86
Bibliographie	88
Annexe	94



INTRODUCTION

Générale

Introduction Générale

1. Introduction :

Les méthodes de commande avancée telles que la commande optimale, la commande prédictive ou la commande robuste, se basent sur une connaissance plus ou moins précise du modèle mathématique du système à contrôler. Ainsi la modélisation des processus constitue une étape très importante dans l'automatique, elle consiste à représenter le comportement dynamique d'un système à l'aide d'un modèle mathématique paramétré. Ce modèle sera utilisé pour la conception d'un contrôleur au sein d'un système de commande, ou encore comme simulateur du processus. Mais un problème majeur se pose lorsque le système à modéliser ou à contrôler est fortement non linéaire, imprécis ou très complexe, les méthodes de modélisation et de contrôle classiques utilisées pour l'analyse et la synthèse des systèmes linéaires, s'avèrent inutiles. Pour cela les automaticiens font appel à des techniques moins conventionnelles au sens mathématique.

Les réseaux de neurones, les algorithmes génétiques et la logique floue, qu'on regroupe parfois sous le terme de systèmes intelligents en anglais *soft computing* ou *intelligent computation* constituent une voie encourageante pour aborder ce problème. Ces méthodes font intervenir des mécanismes qui ont été observés et étudiés dans des domaines de recherche initialement très éloignés de l'informatique : physiologie humaine et animale pour les réseaux de neurones, linguistique et raisonnement humain pour la logique floue, ou encore génétique et sélection naturelle pour les algorithmes évolutionnistes.

Durant ces deux dernières décennies, il y a eu une grande activité dans le domaine de l'application des réseaux de neurones aux différents problèmes de modélisation et de commande des systèmes complexes. Ceci est dû à leurs simplicités et leurs propriétés d'approximation universelle et la capacité du traitement parallèle de l'information. Ces propriétés font que ces réseaux sont de plus en plus utilisés pour modéliser et commander les systèmes dynamiques non linéaires ou linéaires, là où les méthodes classiques échouent.

Les réseaux de neurones, sont des ensembles d'opérateurs non linéaires interconnectés, forment une famille de fonctions non linéaires, qui permet de construire, par apprentissage, une très large classe de modèles et de contrôleurs.

2. Position du problème :

A partir du comportement du cerveau humain et d'un modèle neuronal biologique simple, les chercheurs ont arrivé à construire des modèles neuronaux artificiels. Les réseaux de neurones présentent une grande diversité. En effet un type de réseau neuronal est défini par sa topologie, sa structure interne et, son algorithme d'apprentissage.

Jusqu'à présent, le problème qui reste le plus difficile à résoudre est l'obtention de l'architecture adéquate du réseau. Cette difficulté est mise en évidence par des questions, telles que le nombre de couches cachées qu'il faut utiliser dans un réseau multicouche, le nombre optimal de neurones dans chaque couche, les valeurs initiales des poids de connexions du réseau pendant la phase d'apprentissage...etc. Un mauvais choix peut conduire à de mauvaises performances du réseau correspondant [1].

3. Etat de l'art :

3.1. Apprentissage des RN :

L'apprentissage est une procédure adaptative par laquelle les connexions des neurones d'un réseau sont ajustées face à une source d'information. Il existe plus

d'une vingtaine de types de règles d'apprentissage qui peuvent être regroupées en trois catégories : les règles d'apprentissage supervisé, non supervisé, et renforcé. Dans tous les cas, la procédure adaptative de mise à jour des poids a pour objectif la recherche du minimum de la fonction de coût (ou fonction d'erreur) dans un espace multidimensionnel. Il existe deux techniques de recherche du minimum de la fonction d'erreur: la première est une technique de recherche du minimum local basée sur la méthode du gradient, la seconde est fondée sur des approches heuristiques telles que les algorithmes génétiques [1, 2].

Les algorithmes d'apprentissage peuvent être classés suivant la procédure d'apprentissage et la technique d'optimisation utilisée. L'algorithme de rétro-propagation (ARP) ou de propagation arrière «Back-propagation» est l'exemple d'apprentissage supervisé le plus populaire à cause de la réussite de certaines applications telles que la démonstration de Sejnowski et Rosenberg dans laquelle l'ARP est utilisé dans un système qui apprend à lire un texte, la prédiction des cours du marché boursier, la détection de la fraude dans les opérations par cartes de crédit ...etc. L'historique des publications montre que l'ARP a été découvert indépendamment par différents auteurs mais sous différentes appellations (Grossberg 1998). Il est important de noter que l'ARP souffre des limitations inhérentes à la technique du gradient à cause du risque d'être piégé dans un minimum local. L'ARP standard est également trop lent et très sensible aux variations du taux d'apprentissage, aux conditions initiales (poids et biais initiaux) et à la taille de l'ensemble d'apprentissage (Rumelhart et al. 1986; Hinton 1989) [3].

Plusieurs approches ont été proposées pour améliorer la méthode de la rétro-propagation et réduire ces faiblesses, tel que la modification du pas d'apprentissage, l'utilisation d'algorithmes de type quasi-Newton, la décentralisation du pas d'apprentissage, utilisation des algorithmes génétiques ...etc. [4, 5, 6, 7, 8, 9, 10].

3.2. Structure du RN :

Les premières tentatives de résolution du problème de détermination de l'architecture ont consisté à tester plusieurs réseaux ayant des architectures différentes jusqu'à atteindre la performance désirée. Récemment, de nombreux travaux ont été consacrés au développement de méthodes d'optimisation de l'architecture des réseaux de neurones. Les principaux algorithmes qui ont été proposés peuvent être classés en trois familles :

- Les algorithmes d'élagage (Pruning) : détectent et éliminent les poids ou les unités qui contribuent peu à la performance du réseau.
- Les algorithmes constructifs ou ascendants : partent d'une solution approchée au problème avec un réseau simple, puis ajoutent si nécessaire des unités ou des couches cachées pour améliorer les performances du réseau.
- Les algorithmes directs : définissent une architecture convenable puis réalisent l'apprentissage ou effectuent les deux opérations en même temps [11].

4. Description du travail :

Dans ce travail nous présentons une technique permettant de résoudre les problèmes cités ci-dessus. Cette technique consiste à traiter ces problèmes comme étant un problème d'optimisation multicritères. Il s'agit de trouver la meilleure structure du réseau de neurone utilisé, assurer son apprentissage, et optimiser le nombre et le type de variables à l'entrée du réseau.

Dans ce travail nous avons utilisé les Algorithmes Génétique Multi-Objectifs *AGMO* de type *NSGA2 (Non-dominated Sorting Genetic Algorithm2)* afin d'optimiser les Réseaux de Neurones multi couches *MLP* et à fonction radiale *RBF* avec et sans sélection des variables d'entrées du réseau. Ainsi l'*AGMO* doit fournir la meilleure structure du RN en assurant aussi son apprentissage.

Dans ce travail, nous avons considéré l'optimisation de deux types de réseaux, les réseaux de neurones *MLP* et les réseaux de neurones *RBF*. Pour chaque type nous

avons adopté deux approches. Dans la première approche nous optimisons deux fonctions objectifs qui sont le nombre de neurones de la couche cachée et l'erreur entre la sortie désirée et la sortie du réseau, dans la deuxième approche une troisième fonction objectif est ajoutée, le nombre de régresseurs à l'entrée du réseau. Cette technique est utilisée dans les domaines de modélisation et de contrôle, en particulier le contrôle du système de freinage Anti Blocage des Roues (*Anti-lock Braking System*) connu par l'abréviation *ABS*.

5. Organisation de la thèse :

Cette thèse est organisée comme suit : Dans le prochain chapitre nous présentons le principe de fonctionnement des algorithmes génétiques multi-objectifs et leur mise en œuvre, dans le chapitre 2, nous présentons les réseaux de neurones avec leurs différentes structures et méthodes d'apprentissage, ainsi que leurs caractéristiques et propriétés. Dans le troisième chapitre nous décrivons en détail le travail réalisé, et nous présentons les résultats de son application sur deux problèmes de modélisation et un problème de contrôle. Dans le quatrième chapitre nous exposons les résultats de l'application de cette technique à la commande du système de freinage Anti Blocage des Roues *ABS*.



CHAPITRE 1

L'optimisation Multi-Objectifs

par les Algorithmes Génétiques

Chapitre 1

L'optimisation Multi-Objectifs par les Algorithmes Génétiques

1. Introduction

Les problèmes d'optimisation rencontrés en pratique sont rarement à un seul objectif. La plupart d'entre eux nécessitent l'optimisation simultanée de plusieurs objectifs souvent contradictoires. L'optimisation multi-objectif s'intéresse à la résolution de ce type de problèmes. Elle est apparue au 19^{ème} siècle avec les travaux en économie. Elle a été appliquée initialement en économie et dans les sciences de la gestion avant d'être graduellement utilisée dans d'autres domaines comme l'informatique ou l'ingénierie.

L'optimisation multicritère est actuellement un domaine de recherche en plein essor. En effet, il s'avère de plus en plus que beaucoup de problème monde réel sont de nature multicritère [12]. Dans ce chapitre, nous intéressons à l'optimisation multi objectifs par les algorithmes génétiques que nous allons utiliser pour résoudre le problème de conception des réseaux de neurones. Après avoir introduit les notions de base de l'optimisation multi-objectifs ou multi critères, nous passons brièvement les méthodes basées sur les algorithmes génétiques pour résoudre ce problème.

2. L'optimisation multicritères

De manière formelle, l'optimisation multicritère peut être définie comme suit:

Trouver le vecteur $x = [x_1, x_2, \dots, x_n]^T$ qui satisfait les m contraintes inégalités et les p contraintes égalités suivantes :

$$g_i(x) \geq 0 \quad i = 1, 2, \dots, m \quad \dots\dots\dots (1.1)$$

$$h_i(x) = 0 \quad i = 1, 2, \dots, p \quad \dots\dots\dots (1.2)$$

Tout en optimisant le vecteur de fonctions :

$$f(x) = [f_1(x), f_2(x), \dots, f_k(x)]^T \quad \dots\dots\dots (1.3)$$

A la fin du 19^{ème} siècle, l'économiste *Vilfredo Pareto* formule le concept d'optimum de *Pareto*, qui constitue les origines de la recherche sur l'optimisation multicritère :

Soit F l'espace de solutions faisables délimité par les contraintes définies en (1.1) et (1.2).

On considère qu'un point $x^* \in F$ est *Pareto optimal* si pour chaque $x \in F$, Ou bien

$$\forall i \in I, (f_i(x^*) = f_i(x)) \text{ avec } I = (1, 2, \dots, k) \quad \dots\dots\dots (1.4)$$

$f_i(x^*) > f_i(x)$ même s'il existe un indice $J \in I$ tel que $f_j(x^*) < f_j(x)$

En d'autres termes, cette définition dit que x est *Pareto optimal* s'il n'existe aucun vecteur faisable x qui fasse diminuer un critère sans augmenter en le même temps au moins un autre critère.

Cependant, dans la plupart des cas, l'optimum de *Pareto* n'est pas constitué d'une seule solution mais d'un ensemble de solutions appelées *solutions non-dominées* au sens de *Pareto*.

Les deux techniques les plus utilisées permettant de ramener le problème multicritère à un problème monocritère sont la méthodologie de la *somme pondérée* ou la méthode ε -*contraintes*. En modifiant leurs paramètres, ces deux méthodes permettent de retrouver l'ensemble de solutions optimales au sens de *Pareto*. Elles sont très souvent utilisées car elles permettent l'utilisation de techniques d'optimisation monocritère pour aborder des problèmes multi objectifs. Nous rappelons leur principe à titre illustratif.

- **Somme pondérée**

La méthodologie de somme pondérée la plus utilisée, consiste en fait à ramener un problème multicritère à un problème monocritère. Le critère à optimiser est défini comme la somme pondérée de l'ensemble des critères.

Le problème revient alors à trouver :

$$\min \sum_{i=1}^k w_i f_i(x) \text{ Avec } w_i \geq 0 \quad \dots\dots\dots (1.5)$$

Où w_i représente l'importance relative de chaque critère dans le problème.

Afin que chaque fonction soit de même ordre de grandeur, on exprime en général (1.5) sous la forme suivante :

$$\min \sum_{i=1}^k w_i f_i(x) c_i \text{ avec } c_i = \frac{1}{f_i^0} \quad \dots\dots\dots (1.6)$$

Où f_i^0 est l'optimum global de la fonction f_i .

L'inconvénient de ce type de méthode réside bien évidemment dans le choix des pondérations. Il est impératif de très bien connaître le problème pour évaluer l'importance relative de chacun des critères.

- **Méthode ϵ -contraintes**

Cette approche effectue une optimisation monocritère sur l'un des critères (le plus important selon le décideur), les autres critères étant assimilés à des contraintes du problème. La formulation est la suivante : Trouver le minimum de la $r^{\text{ème}}$ fonction, i.e.

$$\text{trouver } x \in F \text{ tel que : } f_r(x) = \min(f_r(x)) \quad \dots\dots\dots (1.7)$$

Avec les contraintes additionnelles :

$$f_r(x) \leq \epsilon_i \text{ pour } i=1,2,\dots,k \text{ et } i \neq r \quad \dots\dots\dots (1.8)$$

On répète la procédure autant de fois que nécessaire en changeant les valeurs des scalaires ϵ_i jusqu'à atteindre une solution satisfaisante pour le décideur.

Pour déterminer des valeurs adéquates des ϵ_i , il peut être nécessaire d'effectuer une optimisation monocritère de chaque fonction.

3. Techniques d'optimisation multi-objectif par les algorithmes évolutionnaires

Les Algorithmes Génétiques AG sont des algorithmes d'optimisation stochastiques fondés sur le principe Darwinien et des techniques dérivées de la génétique et des mécanismes de la sélection naturelle. Ils sont basés sur le processus biologique naturel de l'évolution des espèces vivants à savoir: la sélection naturelle et la reproduction [13].

Un Algorithme Génétique Simple, AGS, est un algorithme itératif de recherche d'optimum. Il traite un problème d'optimisation à un seul critère. Ce critère (représenté par la fonction à optimiser) a été transformé sous forme d'une fonction d'adaptation, cette approche fonctionne bien pour beaucoup de problèmes, mais il arrive que plusieurs critères soient utilisés simultanément, il n'est plus possible, ni souhaitable de les combiner en une seule fonction. Dans cette situation, on se trouve confronté à un problème à objectifs multiples [12].

L'optimisation multicritère par les AG a été abordée par de nombreux auteurs dans la littérature. C'était en 1985 que *Schaffer* avait proposé le premier algorithme génétique multi-objectif VEGA (Vector Evaluated Genetic Algorithm), plus tard plusieurs implémentations différentes des AG ont été proposées et appliquées avec succès [14]. Les techniques d'optimisation multi-objectives basées sur des algorithmes génétiques sont largement utilisées et dans divers domaines, parmi les travaux récents on peut citer à titre d'exemples [15, 16].

Ces techniques d'optimisation multi-objectives basées sur des algorithmes génétiques, peuvent être classifiées comme suit : les techniques qui n'utilisent pas le concept de Pareto et les techniques utilisant ce concept.

3.1. Techniques non Pareto

Ces méthodes utilisent généralement soit un processus de recherche qui traite les objectifs séparément sans utiliser le concept de *Pareto*, soit elles transforment le problème multi objectif en un problème à un seul objectif. Parmi ces méthodes on

peut citer le premier algorithme multi-objectif utilisant les AG, qui est le VEGA (Vector Evaluated Genetic Algorithm) de *Schaffer*.

3.1.A. L'algorithme VEGA (Vector Evaluated Genetic Algorithm)

En 1985 *Schaffer* avait présenté la méthode de VEGA [14], il s'agit de l'une des premières méthodes utilisant des algorithmes évolutifs. Avant les procédures de croisement et mutation, les meilleurs individus sont sélectionnés pour chaque critère séparément. La sélection se fait par un poids proportionnel à chaque critère, c'est-à-dire la population est divisée en sous-populations selon le nombre de critères. Chacune des fractions de la population subit la procédure de sélection, chacune d'entre elles étant sélectionnée suivant un critère différent. Une fois ces sélections étant réalisées, les fractions de la population sont mélangées et les procédures de croisement et mutation sont appliquées classiquement.

En général, on obtient une convergence vers des optima propres à chacun des critères, c'est pour cela que l'inconvénient principal de cette méthode est qu'à la fin de la procédure, les solutions se répartissent sur les extrêmes de la frontière *Pareto* au détriment des solutions de «compromis et d'arrangement» qui ont une performance généralement acceptable et qui ne possèdent aucun objectif fort au détriment des autres.

3.1.B. Techniques basées sur la transformation du problème multi-objectif en un problème mono-objectif

Ce sont des techniques basées sur des approches classiques pour générer des compromis sur l'ensemble problème, certaines approches utilisent la technique de la moyenne pondérée, où chaque individu utilise une combinaison particulière de poids qui est soit choisie aléatoirement, soit encodée dans l'individu, tous les membres de la population sont évalués par différentes fonctions objectifs. En effet, l'optimisation est effectuée dans plusieurs directions simultanément. Cependant, les inconvénients majeurs de ces techniques classiques, tel que pour certaines, la sensibilité à la forme de la frontière *Pareto*, peuvent limiter l'efficacité de ces techniques. Parmi ces

approches on peut citer : le WBGA Weight-Based Genetic Algorithm [17], et le MOGLS Multi Objective Genetic Local Search [18].

3.2. Techniques utilisant le concept de Pareto

Pour résoudre les inconvénients présentés par les anciennes techniques telle que celle de *Schaffer* (VEGA, 1985), là où les solutions finales convergent vers les extrêmes des fonctions à optimiser, *Goldberg* avait proposé en 1989, l'utilisation du concept de dominance de *Pareto* [13]. Ainsi l'utilisation d'une sélection basée sur la notion de dominance de *Pareto* va faire converger la population vers un ensemble de solutions efficaces, contenant plusieurs choix au décideur en offrant différentes solutions aux extrêmes et des solutions de compromis (au milieu du front de *Pareto*).

Ces techniques peuvent être classées en deux catégories :

3.2.1. Techniques non élitistes

Ces techniques n'utilisent pas le principe d'archivage pour conserver les solutions *Pareto* optimales durant le processus d'optimisation.

Elles mettent en œuvre le concept d'optimalité de *Pareto* pour classer les solutions en différents groupes de solutions non dominées, en vue d'aider à la convergence des algorithmes. Pour assurer la diversité de la population dans la région de *Pareto*, ces algorithmes utilisent la technique de la fonction de partage. Ces algorithmes comprennent le MOGA (Multi-Objective Genetic Algorithm) proposé par *Fonseca* et *Fleming* [19], le NPGA (Niche-Pareto Genetic Algorithm) proposé par *Horn* et *al* [20], et le NSGA1 (Non-dominated Sorting Genetic Algorithm) proposé par *Srinivas* et *Deb* [21]. Les inconvénients majeurs de ces techniques sont leur convergence lente et la difficulté de maintenir une bonne diversité dans la population.

Par exemple le NSGA1 utilise un mode de sélection par classement avec une méthode de nichage. Cette méthode permet d'obtenir de très bons individus et de maintenir une certaine diversité à l'intérieur de la zone optimale. Les individus non-dominés sont d'abord triés et constituent le premier front dans la population courante. Il leur est affecté un grand poids pour leur évaluation, ceci étant le même pour chacun des

individus du même front. Pour maintenir une certaine diversité dans la population, le poids de ces individus est d'autant plus dégradée qu'ils ont des individus voisins. Un nouveau poids est alors déterminé si l'individu n'est pas isolé, en divisant l'ancien par la valeur de la fonction de partage. Le deuxième front de domination est ensuite extrait. Un poids plus faible que celui du dernier individu du premier front leur est attribué puis est dégradé suivant le même mécanisme. Ce système de front est utilisé jusqu'à ce qu'un poids soit attribué à tous les individus de la population. L'évaluation étant opérée de cette façon, une sélection aléatoire proportionnelle est appliquée. La reproduction par croisement et mutation est ensuite réalisée de façon classique jusqu'à un nombre maximum de générations.

3.2.2. Techniques élitistes

C'est une génération d'algorithmes évolutifs multi-objectifs qui intègrent le concept d'archivage dans leur fonctionnement. Il s'agit entre autres de l'algorithme SPEA1 [22] et SPEA2 [23] (Strength Pareto Evolutionary Algorithm 1,2), NSGA2 [24] (Non Dominated Sorting Genetic Algorithm).

Pour remédier à toutes les critiques formulées sur NSGA1, à savoir, complexité, non élitisme et utilisation du partage «sharing» et pour diminuer la complexité de calcul de NSGA1, *Deb* propose une modification de la procédure de tri de la population en plusieurs frontières [24]. L'autre critique sur NSGA1 est l'utilisation de la fonction de partage «sharing», méthode qui exige le réglage d'un ou plusieurs paramètres et qui augmente considérablement les calculs. Dans NSGA2, il remplace la fonction de «*sharing*» par une fonction de remplacement «*Crowding*» en attribuant deux caractéristiques à chaque individu: le rang de non domination et la *distance de Crowding*. Pour répondre à la critique de non élitisme, il utilise une sélection par tournoi, modifie la procédure de passage entre deux générations et définit une notion de préférence entre deux solutions en fonction des deux caractéristiques des individus.

En 2001 *Deb* et *Goel*, ont amélioré le NSGA2 et une nouvelle version a été proposée, il s'agit du NSGA2 avec contrôle de l'élitisme [25], qui est une technique qui vise à assurer une diversité de la population dans tout l'espace faisable.

4. Programmation de l'algorithme génétique multi-objectif NSGA2

L'Algorithme Génétique Multi Objectif que nous avons utilisé dans notre travail est le NSGA2 (Non-dominated Sorting Genetic Algorithm 2) présenté et amélioré par *Deb* et *al.* [24, 25]. C'est l'un des algorithmes les plus utilisés et les plus cités dans la littérature. Il est très utilisé par plusieurs auteurs non seulement dans le cadre de l'optimisation multi-objectifs mais aussi pour comparaison avec d'autres algorithmes, il est considéré comme une référence par beaucoup de chercheurs [26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39].

4.1. Principe de l'algorithme NSGA2

L'algorithme NSGA2 est un algorithme évolutif multi-objectif, établissant les rapports de dominance entre les individus et offrant une méthode de tri particulièrement rapide des chromosomes. Cet algorithme utilise la mesure du surpeuplement autour des individus pour assurer la diversité dans la population. Le principe de cet algorithme est illustré à la figure 1.1.

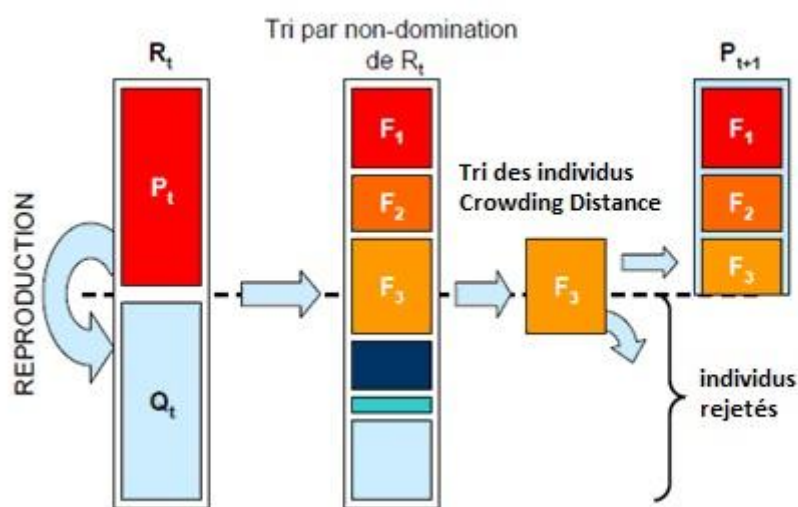


Figure 1-1 – Principe de fonctionnement de l'algorithme NSGA2

Au début, une population initiale est générée aléatoirement, puis elle subit un tri en utilisant le concept de la non-dominance. Chaque solution se voit affecter une force, ou rang, égal à son niveau de non-dominance (1 pour le meilleur niveau, 2 pour le niveau suivant, etc...). L'étape de reproduction consiste en un tournoi pour la sélection des parents. Deux individus de la population sont choisis aléatoirement dans la population, le tournoi est basé sur une comparaison de la domination sous contraintes des deux individus.

Pour une génération t donnée, on crée $R_t = P_t \cup Q_t$, Q_t étant la population enfants de la population précédente P_t (générés à partir des parents à travers les opérateurs de croisement et de mutation), R_t inclut les individus de P_t , ce qui assure le caractère élitiste de l'algorithme NSGA2. La population R_t contient $2N$ individus (elle est composée de N parents et N enfants). R_t subit ensuite un tri en utilisant le concept de la non dominance de Pareto.

Les individus sont regroupés dans des fronts de non-dominance tels que F_1 représente les individus de rang 1, F_2 les individus de rang 2, ... etc. L'objectif suivant est de réduire le nombre d'individus de $2N$ dans la population R_t pour obtenir une population P_{t+1} de taille N . Si la taille de F_1 est inférieure à N , alors tous les individus de F_1 sont conservés. Il en est de même pour les autres fronts tant que le nombre d'individus conservés ne dépasse pas la taille N . Si l'on prend l'exemple de la figure 1.1, les fronts F_1 et F_2 sont intégralement conservés mais la conservation du front F_3 va entraîner un dépassement de la taille N de la population P_{t+1} . Il faut alors procéder à une sélection des individus de F_3 à conserver.

Dans ce cas l'algorithme NSGA2 fait intervenir un mécanisme de préservation de la diversité de la population basé sur l'évaluation de la densité des individus autour de chaque solution, à travers une procédure de calcul de «Distance de proximité». Une faible valeur de la Distance de proximité pour un individu correspond à un individu «bien entouré». On procède alors à un tri décroissant selon cette distance de proximité pour retenir des individus du front F_3 et éliminer ainsi les individus des

zones les plus denses. On complète de cette façon la population P_{t+1} . Les individus ayant des valeurs extrêmes pour les critères sont également préservés par ce mécanisme, permettant ainsi de conserver les bornes extérieures du front de Pareto.

À la fin de cette phase, la population P_{t+1} est créée. Puis une nouvelle population Q_{t+1} est générée par reproduction à partir de P_{t+1} . On poursuit itérativement la procédure décrite ci-dessus jusqu'à la satisfaction de critère d'arrêt fixé par l'utilisateur.

De point de vue général, le NSGA2 permet de maintenir l'élitisme et la diversité sans ajouter de paramètres supplémentaires, tout en utilisant un algorithme séduisant par sa simplicité avec un minimum de paramètres de réglage.

4.2. Organigramme de l'algorithme NSGA2

La figure 1.2 représente l'organigramme de l'algorithme NSGA2.

5. Conclusion

Dans ce chapitre nous avons présenté le problème multi-objectifs et l'état de l'art des différentes techniques d'optimisation multi-objectifs avec les algorithmes génétiques. Ainsi nous avons présenté le principe de base et le fonctionnement de l'algorithme NSGA2, ce dernier est utilisé dans notre travail afin d'optimiser les réseaux de neurones. Ces derniers sont présentés dans le chapitre suivant. La procédure d'optimisation des réseaux de neurones, la mise en œuvre de l'algorithme NSGA2 et ainsi que leurs applications font l'objet des prochains chapitres.



CHAPITRE 2

LES RÉSEAUX DE NEURONES

Chapitre 2

Les réseaux de neurones

1. Introduction

L'idée des réseaux de neurones vient originellement de la modélisation biophysique du cerveau, cette modélisation tente d'expliquer la bio-physiologie du cerveau et ses fonctionnalités. Historiquement, les réseaux de neurones artificiels (ou *Artificial Neural Network* en anglais) sont apparus avec la description d'un modèle du neurone, cellule du cerveau, par *McCulloch* et *Pitts* en 1943 et d'une règle d'apprentissage par *Hebb* en 1950.

La recherche dans le domaine des réseaux de neurones est centrée sur les architectures selon lesquelles les neurones sont combinés et les méthodologies par lesquelles les poids des interconnexions sont calculés ou ajustés. De nos jours les chercheurs sont divisés en deux groupes, le premier est constitué de biologistes, des physiciens et des psychologues, ce groupe tente de développer un modèle neuronal capable d'imiter, avec une précision donnée, le comportement du cerveau, le second groupe se compose d'ingénieurs qui sont concernés par la façon dont les neurones artificiels sont interconnectés pour former des réseaux possédant des capacités de calcul puissantes. Actuellement, les études des réseaux de neurones sont en pleine expansion et leur utilisation est en accroissement rapide.

Un Réseau de Neurones (RN) est un modèle de calcul dont la conception est très schématiquement inspiré du fonctionnement de vrais neurones. L'un des avantages d'un RNA est la facilité de recharger ses paramètres (poids, nombre de neurones, nombre de couches, ...etc.), lors d'une modification possible de son environnement.

Cependant, le RNA est une boîte noire qui fournit une réponse quand on lui présente une donnée, mais ne fournit pas de justification facile à interpréter [40, 41].

2. Le neurone artificiel

2.1. Le neurone biologique

Le neurone biologique est une cellule nerveuse qui se compose d'un corps cellulaire appelé «soma» qui contient le noyau où se déroulent les activités cellulaires vitales, de prolongements appelés «nérites», ces dernières sont de deux types, les dendrites qui servent de canaux d'entrées et l'axone, unique qui est le canal de sortie. La figure 2.1 représente un neurone biologique.

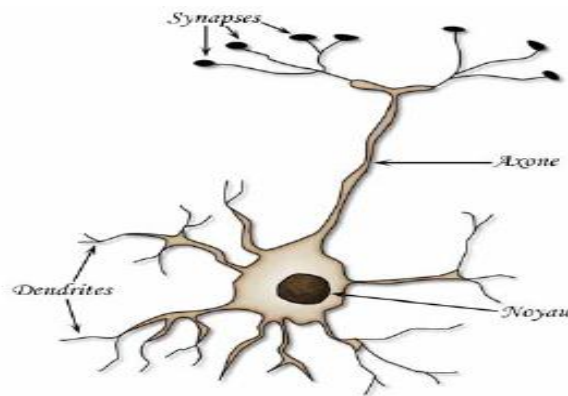


Figure 2.1 neurone biologique

Au point de vue fonctionnel, le neurone est considéré comme une entité polarisée, c'est-à-dire que l'information ne se transmet que dans un seul sens: des dendrites vers l'axone. Le neurone va donc recevoir des informations, venant d'autres neurones, grâce à ses dendrites. Il va ensuite y avoir sommation, au niveau du corps cellulaire, de toutes ces informations et via un potentiel d'action (un signal électrique) le résultat de l'analyse va transiter le long de l'axone jusqu'aux terminaisons synaptiques. A cet endroit, lors de l'arrivée du signal, des vésicules synaptiques vont venir fusionner avec la membrane cellulaire, ce qui va permettre la libération des neurotransmetteurs

(médiateurs chimiques) dans la fente synaptique. Le signal électrique ne pouvant pas passer la synapse (dans le cas d'une synapse chimique), les neurotransmetteurs permettent donc le passage des informations, d'un neurone à un autre.

Les synapses possèdent une sorte de «mémoire» qui leur permet d'ajuster leur fonctionnement. En fonction de leur «histoire», c'est-à-dire de leur activation répétée ou non entre deux neurones, les connexions synaptiques vont donc se modifier. Ainsi, la synapse va faciliter ou non le passage des influx nerveux. Cette plasticité est à l'origine des mécanismes d'apprentissage.

2.2 Modèle d'un neurone artificiel

Le neurone formel (figure 2.2) est une modélisation mathématique qui reprend les principes du fonctionnement du neurone biologique, en particulier la sommation des entrées. En général, un neurone formel est un élément de traitement possédant n entrées $x_1, x_2, \dots, x_i, \dots, x_n$ (qui sont les entrées externes ou les sorties des autres neurones) et une ou plusieurs sorties. Son traitement consiste à effectuer à sa sortie y_j le résultat d'une fonction de seuillage f (dite aussi la fonction d'activation) de la somme pondérée.

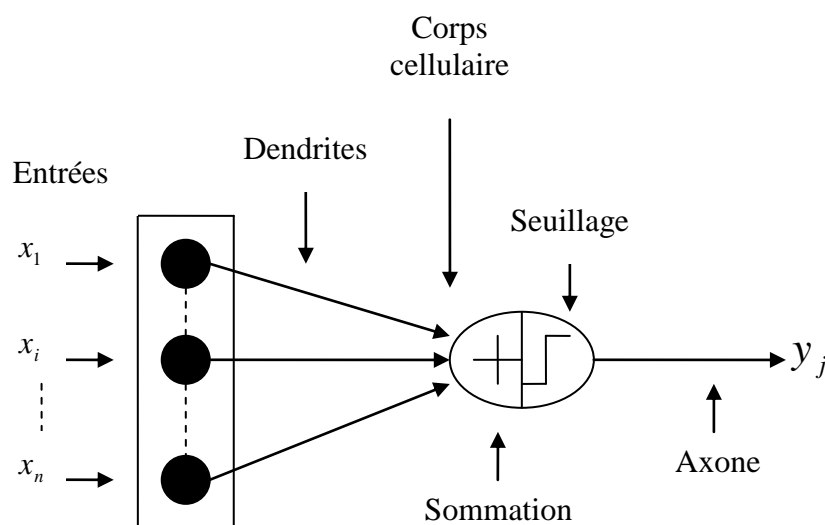


Figure 2.2 : Le neurone artificiel

La modélisation consiste à rassembler les connaissances que l'on a du comportement dynamique du processus, par une analyse physique des phénomènes mis en jeu, et une analyse des données expérimentales. Ces analyses conduisent à la définition des grandeurs caractérisant le processus, c'est-à-dire ses entrées, ses variables d'état et ses sorties.

La figure 2.3 présente l'analogie entre le neurone biologique et le neurone artificiel.

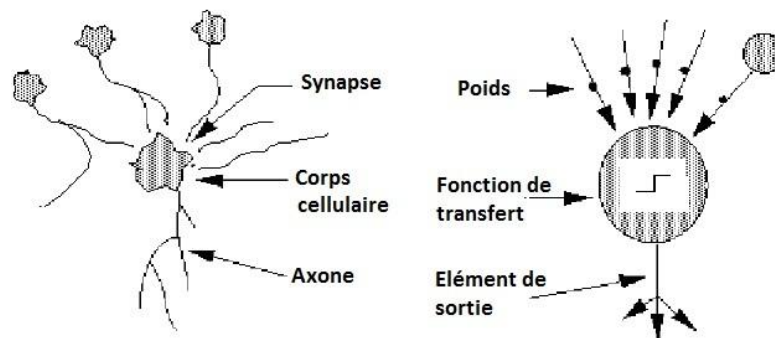


Figure 2.3 : Analogie entre le neurone biologique et le neurone artificiel

2.3 Fonctionnement d'un neurone artificiel

Le neurone calcule la somme de ses entrées qui passe à travers la fonction d'activation pour produire sa sortie. Le calcul de la valeur de la sortie se fait selon les deux formules 2.1 et 2.2 suivantes :

Une combinaison linéaire des entrées

$$v = w_0 + \sum_{i=1}^n w_i x_i \quad \dots\dots\dots (2.1)$$

w_i : sont appelés poids synaptiques, w_0 est appelé biais, il peut être considéré comme la pondération de l'entrée 0 fixée à 1.

x_i : les entrées du neurone.

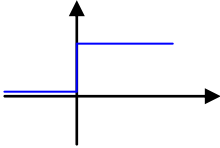
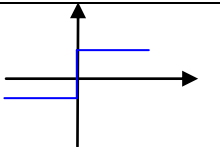
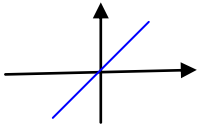
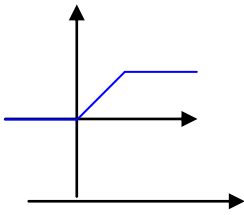
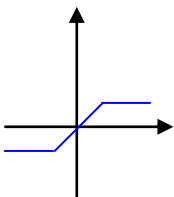
$$\text{La sortie du neurone est : } y = f(v) = f\left(\sum_{i=0}^n w_i x_i\right) \quad \dots\dots\dots (2.2)$$

f : est la fonction d'activation du neurone.

2.4 Fonction d'activation

La fonction d'activation ou de seuillage sert à introduire une non linéarité dans le fonctionnement du neurone.

Dans sa première version, le neurone formel était implémenté avec une fonction à seuil, mais en réalité de nombreuses versions existent. Ainsi le neurone de *McCulloch* et *Pitts* a été généralisé de différentes manières, en choisissant d'autres fonctions d'activations, comme les fonctions de neurones énumérées au tableau 2.1. Les trois fonctions les plus utilisées sont les fonctions « seuil », « linéaires » et « sigmoïdes » [42, 43].

Nom de la fonction	Type	Equation	Allure
Seuil	Binaire (fonction de Heaviside)	$f(x) = 1$ si $x > 0$ $f(x) = 0$ si $x \leq 0$	
	Signe	$f(x) = 1$ si $x > 0$ $f(x) = -1$ si $x \leq 0$	
Linéaire	Identité	$f(x) = x$	
	Saturé positif	$f(x, k) = 0$ si $x < 0$ $f(x, k) = 1$ si $x \geq 1/k$ $f(x, k) = kx$ sinon	
	Saturé symétrique	$f(x, k) = -1$ si $x < -1/k$ $f(x, k) = 1$ si $x \geq 1/k$ $f(x, k) = kx$ sinon	

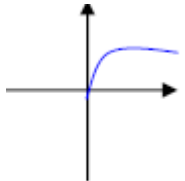
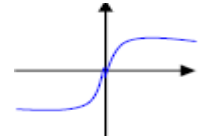
Sigmoïde	Positive (type logistique)	$f(x, k) = \frac{1}{1 + e^{-kx}}$	
	Symétrique (type tanh)	$f(x, k) = \frac{2}{1 + e^{-kx}} - 1$	

Tableau. 2. 1 : Différentes fonctions de seuillage des neurones.

Toutes les fonctions d'activation utilisées doivent être différentiables, car l'architecture du réseau de neurones l'impose pour que l'apprentissage soit possible.

3. Réseaux de neurones et architectures

Un RNA est en général composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. Chaque couche (i) est composée de n_i neurones, prenant leurs entrées sur les n_{i-1} neurones de la couche précédente. À chaque synapse est associé un poids synaptique, de sorte que les n_{i-1} sont multipliés par ce poids, puis additionnés par les neurones de niveau i . C'est grâce à ces poids, que le réseau acquiert l'aptitude de résolution de problèmes complexes.

On distingue deux types d'architectures : les réseaux statiques (non récurrents) et les réseaux dynamiques (récurrents).

3.1. Le choix de l'architecture du réseau de neurones

Le choix de l'architecture du réseau de neurones artificiel pour un problème donné a longtemps été un problème. Des développements, montrent qu'il est souvent possible de rechercher (en utilisant les algorithmes génétiques par

exemple) une architecture du RNA qui améliore fortement les résultats obtenus avec les méthodes classiques. Le réseau multicouche par ses qualités de simplicités et potentialités est le modèle le plus utilisé actuellement au niveau des applications [44].

3.1.1. Les réseaux statiques

Un réseau statique dépend uniquement des sorties des couches précédentes des neurones. Chaque neurone est connecté seulement aux neurones de la couche suivante, l'information circule dans un seul sens, de l'entrée vers la sortie. Il réalise une ou plusieurs fonctions algébriques de ses entrées, par composition des fonctions réalisées par chacun des neurones.

Le réseau de neurones le plus simple comporte deux couches de neurone formel (figure 2.4), les neurones d'entrée reçoivent les informations à identifier et d'autres neurones effectuent les calculs et fournissant la réponse du réseau.

Il existe également des réseaux multicouche, dans lesquels les neurones effectuant les calculs ne sont pas directement reliés à l'extérieur; les neurones représentés au centre en figure (b) sont des neurones cachés, il n'existe aucune connexion entre les neurones d'une même couche.

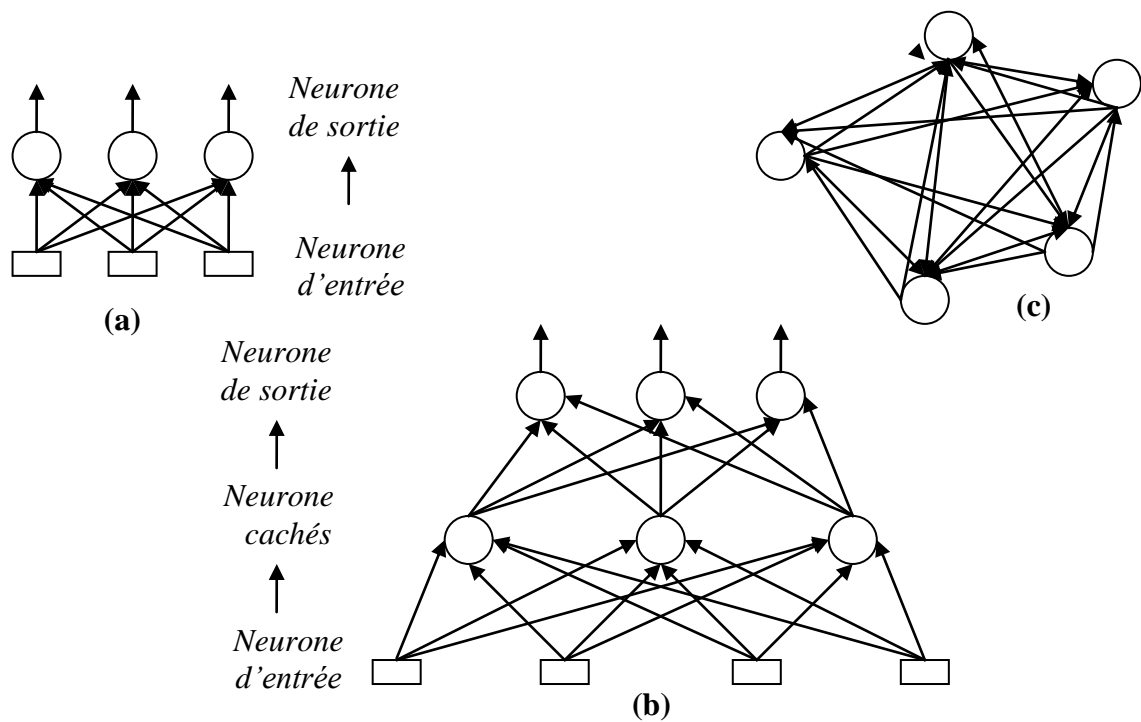


Figure 2.4 (a): Architecture simple, (b): architecture d'un RN statique multicouches, (c): architecture d'un RN dynamique de Hop Field.

3.1.2. Les réseaux dynamiques

Contrairement aux réseaux statiques, les réseaux dynamiques peuvent avoir une topologie de connexions quelconque. Les réseaux dynamiques sont organisés aussi en couches, la circulation de l'information est bidirectionnelle, telle que l'entrée d'un neurone dépend de sa sortie ou des neurones des couches suivantes. Chaque neurone peut recevoir comme entrées, les valeurs passées des autres neurones (que ce soit de la couche qui le précède ou bien même à partir des neurones de sa propre couche). Cette particularité d'interconnexion des neurones fait en sorte que ce type de réseaux puisse être utilisé dans la résolution de problèmes d'identification et de contrôle des systèmes régis par des équations dynamiques.

4. Les propriétés d'un réseau de neurones

4.1. La propriété d'approximation universelle

La propriété d'approximation universelle a été démontrée par *Cybenko* et *Funahashi*, et peut s'énoncer de la façon suivante:

Théorème

Toute fonction bornée suffisamment régulière peut être approchée uniformément, avec une précision arbitraire, dans un domaine fini de l'espace de ses variables, par un réseau de neurones comportant une couche de neurones cachée en nombre fini, possédant tous la même fonction d'activation, et un neurone de sortie linéaire.

Comme le montre ce théorème, le nombre de neurones cachés doit être choisi convenablement pour obtenir la précision voulue.

4.2. La propriété de parcimonie

Lorsqu'on cherche à modéliser un processus à partir des données, on s'efforce toujours d'obtenir les résultats les plus satisfaisants possibles avec un nombre minimum de paramètres ajustables, c'est le concept de parcimonie. Dans cette optique, *Hornik et al.* 1994 ont montré que si le résultat de l'approximation (c'est-à-dire la sortie du réseau de neurones) est une fonction non linéaire des paramètres ajustables, elle est plus parcimonieuse que si elle est une fonction linéaire de ces paramètres. De plus, pour des réseaux de neurones à fonction d'activation sigmoïdale, l'erreur commise dans l'approximation varie comme l'inverse du nombre de neurones cachés, et elle est indépendante du nombre de variables de la fonction à approcher. Par conséquent, pour une précision donnée, donc pour un nombre de neurones cachés donné, le nombre de paramètres du réseau est proportionnel au nombre de variables de la fonction à approcher.

Ce résultat s'applique aux réseaux de neurones à fonction d'activation sigmoïdale puisque la sortie de ces neurones n'est pas linéaire par rapports aux poids synaptiques. Cette propriété montre l'intérêt des réseaux de neurones par rapport à d'autres approximateurs comme les polynômes dont la sortie est une fonction linéaire des paramètres ajustables: pour un même nombre d'entrées, le nombre de paramètres ajustables à déterminer est plus faible pour un réseau de neurones que pour un polynôme.

4.3 L'apprentissage des réseaux de neurones

On associe généralement aux RNA un algorithme d'apprentissage permettant de modifier de manière plus ou moins automatique le traitement effectué afin de réaliser une tâche donnée. Pour les RNA, l'apprentissage peut être considéré comme le problème de la mise à jour des poids des connexions au sein des réseaux, afin de réussir la tâche qui lui est demandée.

4.3.1. Types d'apprentissage

L'apprentissage est la caractéristique principale des RNA et il peut se faire de différentes manières et selon différentes règles.

4.3.1.1. Le mode supervisé

Dans ce type d'apprentissage, le réseau s'adapte par comparaison entre le résultat qu'il a calculé, en fonction des entrées fournies, et la réponse attendue en sortie. Ainsi, le réseau va se modifier jusqu'à ce qu'il trouve la bonne sortie, c'est-à-dire celle attendue, correspondant à une entrée donnée.

Cette procédure est souvent réalisée par les algorithmes de la rétro propagation "Back Propagation Algorithm", basée sur la minimisation de l'erreur d'apprentissage et la règle de chaînage. Cet algorithme basé sur la méthode de la descente du gradient a montré plusieurs inconvénients tel que la lenteur de la convergence, la sensibilité aux minimums locaux et la difficulté de régler les

paramètres d'apprentissage (le nombre de neurones dans les couches cachées, le pas d'apprentissage...etc.) [41].

4.3.1.2. Le renforcement

Le renforcement est en fait une sorte d'apprentissage supervisé et certains auteurs le classe d'ailleurs, dans la catégorie des modes supervisés. Dans cette approche le réseau doit apprendre la corrélation entrée/sortie via une estimation de son erreur, c'est-à-dire du rapport échec/succès. Le réseau va donc tendre à maximiser un index de performance qui lui est fourni, appelé signal de renforcement. Le système étant capable ici, de savoir si la réponse qu'il fournit est correcte ou non, mais il ne connaît pas la bonne réponse.

4.3.1.3. Le mode non supervisé (ou auto organisationnel)

Dans ce cas, l'apprentissage est basé sur des probabilités. Le réseau va se modifier en fonction des régularités statistiques de l'entrée et établir des catégories, en attribuant et optimisant une valeur de qualité, aux catégories reconnues.

4.3.1.4. Le mode hybride

Le mode hybride reprend en fait les deux approches supervisée et non supervisée, puisque une partie des poids va être déterminée par apprentissage supervisé et l'autre partie par apprentissage non supervisé.

Le tableau 2.2 représente les différents algorithmes d'apprentissage les plus utilisés.

Type d'algorithme d'apprentissage	Recherche du minimum local	Recherche du minimum global
Apprentissage supervisé	Règle du perceptron (Rosenblatt 1958) $W(k+1) = W(k) + eX^T$	Technique du recuit simulé $W(X \rightarrow X') = \begin{cases} 1 & \text{si } \Delta E < 0 \\ e^{-\Delta E/KT} & \text{autre} \end{cases}$
	Règle de Widrow-Hoff (1960) $W(k+1) = W(k) + \alpha(t-y)_k \frac{X(k)}{\ X(k)\ ^2}$	Règle de Geman (1984) $T(t) = \frac{T_0}{1 + \ln t}$
	L'Algorithme de retropropagation (Werbos 1974) $W^m(k+1) = W^m(k) - \rho s^m (y^{m-1})^T$ $\frac{\partial \hat{F}}{\partial W^m} = s^m y^{m-1}$ $s^m = \frac{\partial \hat{F}}{\partial n^m} = \left(\frac{\partial n^{m+1}}{\partial n^m} \right) \frac{\partial \hat{F}}{\partial n^{m+1}} = \left(\frac{\partial n^{m+1}}{\partial n^m} \right) s^{m+1}$	Algorithmes génétiques (Holland 1975; Goldberg 1989)
	Améliorations de l'algorithme de retro-propagation	Amélioration des AG et utilisation des AG multi-objectifs
Apprentissage non supervisé	Règle d'Hebb (Changeux et Danchin 1976) $W(k) = W(k-1) + \rho y(k)X^T(k)$	-
	Règle de Hinton (1989) $W(k) = (1 - \gamma)W(k-1) + \rho y(k)X^T(k)$	
	Règle d'Oja (1982) $W(k+1) = W(k) + \rho y(k)X(k) - \rho y(k)^2 W(k)$	
	Règle de Yuille et al. (1989) $W(k+1) = W(k) + \rho (y(k)X(k) - \ W(k)\ ^2 W(k))$	
	Règle de Hassoum (1995) $W(k+1) = \rho \left[y(k)X(k) - \lambda W(k) \left(1 - \frac{1}{\ W(k)\ } \right) \right]$	

Tableau 2.2 : Algorithmes d'apprentissage des RNA

5. Description des types de réseaux à entraîner

5.1. Les réseaux à perceptrons multicouches

Le réseau à perceptrons multicouches, Multi Layered Perceptrons, MLP, figure 2.5, est de la famille générale des réseaux statiques ou à «propagation vers

l'avant», *feedforward*, l'information se propage dans un sens unique, des entrées vers les sorties sans rétroaction. Son apprentissage est de type supervisé, le signal d'erreur est rétro-propagé vers les entrées pour mettre à jour les poids des neurones [43].

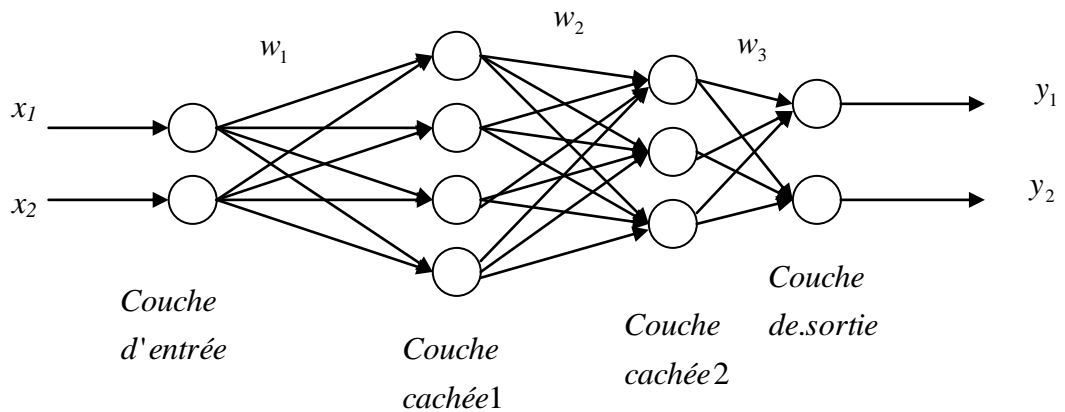


Figure 2.5: Architecture générale d'un réseau MLP

5.2. Les réseaux de neurones à fonction de base radiale RBF

Les réseaux de type RBF, *Radial Basis Functions* [41, 44], sont des réseaux à trois couches qui ont comme origine une technique d'interpolation. Cette technique s'avère être efficace, en particulier pour les applications de classification. Son principe est d'approximer un comportement désiré par une collection de fonctions, appelées fonctions de base radiales. Les fonctions de base de la méthode RBF sont locales, c'est-à-dire qu'elles ne donnent des réponses utiles que dans un domaine d'influence délimité par un *seuil de distance*. Ce seuil est défini autour d'un point, le centre. La distance Euclidienne est généralement utilisée pour la mesure de distance :

$$d(x) = \|x - c_i\| \dots\dots\dots (2.3)$$

Où $d(x)$ mesure la distance entre le vecteur x et le centre c_i de la fonction de la réponse, la plus utilisée est la gaussienne donnée par la formule 2.4 suivante:

$$f_i(x) = e^{\frac{-d(x)^2}{\sigma_i^2}} \dots\dots\dots (2.4)$$

Ainsi, chaque fonction noyau est décrite par deux paramètres: La position de son centre c_i et le seuil de distance σ_i .

Pour approximer un comportement donné, les fonctions de base associées à une catégorie sont assemblées de façon à couvrir avec leurs domaines d'influence l'ensemble des vecteurs de cette catégorie. Ces fonctions sont ensuite pondérées puis sommées pour produire une valeur de sortie.

5.2.1. Architecture générale d'un réseau RBF

La connexion entre la couche cachée et la couche de sortie est locale, c'est-à-dire que les centres associés à chaque catégorie ne sont connectés qu'à un neurone de sortie qui indique si le stimulus appartient à cette catégorie. Cette structure du réseau fournit les avantages de réduire la complexité de l'algorithme et l'espace mémoire nécessaire pour stocker la matrice de poids de la couche de sortie. Elle facilite aussi l'exploitation du parallélisme intrinsèque de l'algorithme en éliminant une grande partie de l'interdépendance des données. La figure 2.6 présente la méthode RBF sous la forme d'un réseau à trois couches : une couche d'entrée, une couche cachée composée des fonctions de base et une couche de sortie dont les neurones sont généralement animés par une fonction d'activation linéaire [41, 44].

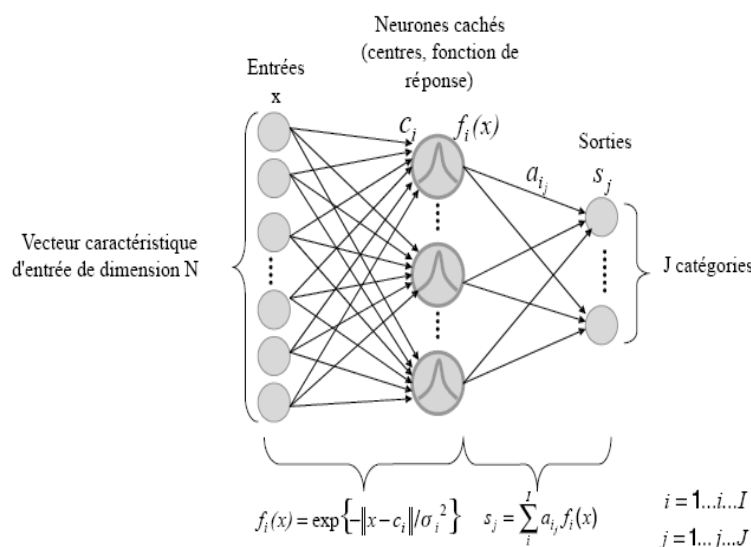


Figure 2.6: Architecture générale d'un réseau RBF

5.2.2. Apprentissage et placement des centres dans un réseau RBF

L'apprentissage des réseaux RBF commence par le paramétrage des fonctions de base [41, 44]. Nous devons d'abord acquérir une collection d'exemples pour chaque catégorie. Certains exemples d'une catégorie peuvent alors être assez ressemblants. Dans ce cas, nous cherchons à les fusionner en déterminant un unique vecteur d'apprentissage les représentant. A chaque vecteur d'apprentissage c_i est associé un seuil de distance σ_i délimitant le domaine d'influence.

La plupart des algorithmes d'apprentissage des réseaux RBF procèdent à une sélection aléatoire d'un certain nombre d'échantillons de l'espace d'entrée comme étant les centres des fonctions de base, ensuite une méthode d'optimisation linéaire est utilisée pour déterminer les poids des connexions. Cependant, une sélection aléatoire des centres n'est pas toujours satisfaisante. Pour cela, un algorithme d'apprentissage non supervisé est utilisé pour l'ajustement des centres des fonctions de base dans la couche cachée.

L'algorithme des *k-means* et la méthode des *moindres carrés récursifs* ou la *règle d'adaptation instantanée* sont souvent respectivement utilisées pour l'ajustement des centres et l'adaptation des poids des connexions respectivement

6. La commande des processus par les RN

Les capacités de traitement parallèle, d'apprentissage et d'approximation que possèdent les réseaux de neurones permettent de concevoir les contrôleurs neuronaux capables de résoudre les problèmes rencontrés dans les méthodes classiques [40, 41]. L'utilisation des réseaux de neurones pour la commande de processus non linéaires dépend naturellement des aptitudes de ces derniers à la modélisation. Il s'agit essentiellement d'une extension non linéaire de la commande optimale avec coût quadratique sur un horizon infini.

En pratique, l'utilisation de méthodes neuronales à base de réseaux multicouche pose certaines difficultés, dont les principales sont :

- Combien de couches cachées faut-il utiliser dans un réseau multicouche ?
- Quels neurones faut-il connecter ?
- Quelles sont les valeurs initiales des poids de connexion du réseau pendant la phase d'apprentissage ?

Pour remédier à ces difficultés, nous proposons dans la suite de cette thèse une approche basée sur les algorithmes génétiques multi-objectifs dans le but de modéliser et contrôler les systèmes.



CHAPITRE 3

*Optimisation des réseaux de neurones
avec et sans sélection des entrées
par les algorithmes génétiques
multi-objectifs*

Chapitre 3

Optimisation des réseaux de neurones avec et sans sélection des entrées par les algorithmes génétiques multi-objectifs

1. Introduction

Dans ce chapitre, nous considérons l'optimisation des réseaux de neurones par les algorithmes génétiques multi-objectifs, AGMO de type *NSGA2* (*Non-dominated Sorting Genetic Algorithm2*) proposé par *Deb* [24, 25] et qui est très utilisé dans plusieurs travaux récents [45, 46, 47]. Cette approche sera appliquée sur deux types de réseaux : les réseaux MLP et les réseaux RBF.

Nous développons dans un premier temps une optimisation à deux objectifs : les poids des connexions et le nombre de neurones dans la couche cachée ensuite à trois objectifs, les deux précédents plus les entrées du réseau. Plusieurs exemples de modélisation et de commande seront étudiés.

2. Optimisation multi-objectifs des RN MLP

Dans cette section, nous décrivons l'optimisation des réseaux de neurones de type MLP par les algorithmes génétiques multi objectifs. Le réseau MLP est caractérisé par le nombre de ses entrées et sorties, le nombre de couches cachées, le nombre de neurones par couche et les poids des connexions. Nous considérons, un réseau à une seule couche cachée (*monocouche, single hidden layer*). Deux problèmes seront traités, un problème à deux objectifs et un problème à trois objectifs.

2.1. Optimisation à deux objectifs: structure du chromosome

Les paramètres définissant un réseau monocouche sont le nombre de ses entrées et sorties, le nombre de neurones dans la couche cachée et les poids des connexions. Les entrées et la sortie du réseau étant fixées, il s'agit alors de trouver le réseau qui minimise les deux fonctions suivantes:

1. Le nombre de neurones de la couche cachée (noté N_{ncc})
2. L'erreur quadratique entre la sortie désirée et la sortie réelle du modèle neuronal dans le cas modélisation ou la sortie du processus commandé dans le cas de la commande.

Les paramètres de l'optimisation sont les poids des connexions. Pour l'application des algorithmes génétiques multi-objectifs de type NSGA 2, nous devons définir la structure du chromosome contenant les paramètres à optimiser. Dans notre cas, le chromosome est constitué des paramètres (allèles) suivants:

$$[N_{ncc} \ W_{11} \ W_{12} \ \dots \ W_{1N_{ncc}} \ W_{21} \ W_{22} \ \dots \ W_{2N_{ncc}} \ \dots \ W_{i1} \ W_{i2} \ \dots \ W_{iN_{ncc}} \ Z_1 \ Z_2 \ \dots \ Z_{N_{ncc}}]$$

Où:

N_{ncc} : est le Nombre de neurone de la couche cachée,

$W_{11} \ W_{12} \ \dots \ W_{1N_{ncc}}$: poids de connexion entre l'entrée 1 et les neurones de la couche cachée,

$W_{21} \ W_{22} \ \dots \ W_{2N_{ncc}}$: poids de connexion entre l'entrée 2 et les neurones de la couche cachée,

$W_{i1} \ W_{i2} \ \dots \ W_{iN_{ncc}}$: poids de connexion entre l'entrée i et les neurones de la couche cachée,

$Z_1 \ Z_2 \ \dots \ Z_{N_{ncc}}$: poids de connexion entre les neurones de la couche cachée et le neurone de la couche de sortie.

La longueur du chromosome « L_{ch} » dépend du nombre de neurones de la couche cachée « N_{ncc} », le nombre de neurones de la couche d'entrée « N_{nce} » et de la couche de sortie « N_{ncs} » :

$$L_{ch}=(N_{nce}+N_{ncs})*N_{ncc}+1 \quad \dots \dots \dots \quad (3.1)$$

Dans cette 1^{ère} approche, le nombre de neurones de la couche d'entrée « *Nnce* » et de la couche de sortie « *Nncs* » sont fixes. La taille de la population T_m (matrice population) est donnée:

$$T_m = N * ((Nnce + Nncs) * \max(Nncc) + 1) \dots\dots\dots (3.2)$$

N: est le nombre d'individus de la population.

Par exemple, pour un réseau de neurones à 2 entrées et une sortie et si le nombre de neurones de la couche cachée est 20, la longueur du chromosome est 61, alors si par exemple *Nncck*=6 le chromosome *k* est organisé comme suit: [*Nncck*(=6) $W_{11} \dots W_{16} W_{21} \dots W_{26} Z_1 \dots Z_6 0 0 \dots 0$] (*Lch*=61), *Nncck* étant le nombre de neurones de la couche cachée du *k^{ème}* individu.

Ainsi, trois 3 chromosomes d'une population de 80 individus où *Nnccmax*=20, avec 2 entrées (*Nnce*=2) et une seule sortie (*Nncs*=1) et *Nnc₁*=5; *Nnc₃₀*=20; *Nnc₈₀*=2 sont structurés comme suit :

<i>Nnc₁</i> (=5) $W_{11} \dots W_{15} W_{21} \dots W_{25} Z_1 \dots Z_5 0 0 \dots 0$
...
<i>Nnc₃₀</i> (=20) $W_{11} \dots W_{120} W_{21} \dots W_{220} Z_1 \dots Z_{20}$
...
<i>Nnc₈₀</i> (=2) $W_{11} W_{12} W_{21} W_{22} Z_1 Z_2 0 0 \dots 0$

2.2. Optimisation à trois objectifs: structure du chromosome

De la même façon que précédemment, nous considérons un réseau MLP à une seule couche cachée. Cependant dans ce cas, les variables d'entrée, et donc le nombre de neurones a l'entrée, ne sont plus fixées et seront donc incluses dans le processus d'optimisation. Nous aurons donc à minimiser les trois fonctions suivantes:

1. Le nombre de neurones de la couche cachée (*Nncc*)
2. L'erreur quadratique qui est la différence entre l'entrée référence (ou sortie désirée) et la sortie réelle du modèle neuronal (dans le cas modélisation) ou la sortie du processus commandé (dans le cas contrôle).
3. Le nombre de regresseurs à l'entrée du RN.

Les entrées du réseau seront des regressseurs des variables du système. Par exemple si le système a une entrées $u(t)$ et une sortie $y(t)$, les entrées du réseau seront choisies parmi les regressseurs $u(t)$, $u(t-1)$, $u(t-2)$, ..., $y(t-1)$, $y(t-2)$, Le nombre maximal des d'entrées est fixé à dix (10). Un regressseur est validé si sa position dans le chromosome est égale à 1, sinon si sa position=0, il n'est pas validé et il sera donc désactivé. Ainsi, par exemple pour un système à deux variables d'entrée et une sortie, la structure de chromosome est la suivante:

$$[reg_{11} \ reg_{12} \ \dots \ reg_{15} \ \mathbf{reg_{21}} \ \mathbf{reg_{22}} \ \dots \ \mathbf{reg_{25}} \ Nncc \ W_{111} \ W_{112} \dots W_{11Nncc} \ W_{121} \ W_{122} \ \dots W_{12Nncc} \ \dots \ W_{151} \ W_{152} \ \dots \ W_{15Nncc} \ \dots \ \mathbf{W_{211}} \ \mathbf{W_{212}} \dots \mathbf{W_{21Nncc}} \ \mathbf{W_{221}} \ \mathbf{W_{222}} \ \dots \ \mathbf{W_{22Nncc}} \ \dots \ \mathbf{W_{251}} \ \mathbf{W_{252}} \ \dots \ \mathbf{W_{25Nncc}} \ Z_1 \ Z_2 \ \dots \ Z_{Nncc}]$$

Où:

- $reg_{11}, \dots, reg_{14}$ sont les regressseurs d'une variables du système : $u(t-1), \dots, u(t-4)$
- reg_{15} correspond à l'erreur instantanée (eq).
- $reg_{21} \dots reg_{25}$ sont les regressseurs de l'autre variable du système : $y(t-1), \dots, y(t-5)$
- $Nncc$: est le Nombre de neurone de la couche cachée.
- $W_{111} \ W_{112} \ \dots W_{knp} \ \dots \ W_{25Nncc}$: poids de connexion entre le regressseur n de l'entrée k et le neurone p de la couche cachée.
- $Z_1 \ Z_2 \ \dots \ Z_{Nncc}$: poids de connexion entre les neurones de la couche cachée et le neurone de la couche de sortie.

Exemple : Le chromosome suivant

$$[1 \ 1 \ 0 \ 1 \ 0 \ \mathbf{0} \ \mathbf{1} \ \mathbf{0} \ \mathbf{0} \ \mathbf{1} \ 3(Nncc) \ W_{111} \dots W_{253} \ Z_1 \ Z_2 \ Z_3]$$

Active les entrées : $u(k)$, $u(k-1)$, $u(k-3)$, et $y(k-1)$, $y(k-4)$ et $Nncc=3$, La longueur du chromosome « Lch » est en général obtenue par:

$$Lch=(Nreg * max(Nncc)) + (Nnce + Nncc) * max(Nncc) + 1 \ \dots \dots \dots \ (3.3)$$

La longueur du chromosome (Lch) dépend du nombre de neurones de la couche cachée ($Nncc$), le nombre de neurones de la couche d'entrée ($Nnce$) et de la couche de sortie ($Nncc$).

La taille de la population T_m (matrice population) sera calculée alors par la formule 3.4 :

$$T_m = N * Lch \quad \dots\dots\dots (3.4)$$

N : est le nombre d'individus de la population.

3. Optimisation multi-objectifs des RN RBF

Dans cette section, nous décrivons l'optimisation des réseaux RBF par les algorithmes génétiques multi objectifs NSGA 2. De la même façon que ci-dessus, deux problèmes seront traités, un problème à deux objectifs et un problème à trois objectifs.

Un réseau RBF est défini par le nombre de ses entrées, le nombre des fonctions de base, les paramètres des fonctions de base et les poids des connexions. Comme pour les réseaux MLP, dans une première approche, on considère deux objectifs dans la seconde, trois objectifs. Dans ce travail, nous avons utilisé des fonctions de base radiales de forme gaussienne, leurs centres (C_i) et les largeurs σ (sig_i) de ces fonctions seront inclus dans l'optimisation.

3.1. Optimisation à deux objectifs: structure du chromosome

Dans cette approche, les entrées du réseau étant préalablement fixées, on recherche le réseau défini par les poids et les paramètres des fonctions de base qui minimise les deux objectifs suivants :

1. Le nombre de neurones de la couche cachée (N_{ncc}).
2. L'erreur quadratique cumulée entre sortie désirée et la sortie du RBF.

Le chromosome est constitué des paramètres suivants:

$$[N_{ncc} \ C_1 \ C_2 \ \dots \ C_{N_{ncc}} \ sig_1 \ sig_2 \ \dots \ sig_{N_{ncc}} \ Z_1 \ Z_2 \ \dots \ Z_{N_{ncc}}]$$

Où:

- N_{ncc} : est le Nombre de neurone de la couche cachée.
- $C_1 \ C_2 \ \dots \ C_{N_{ncc}}$: Les centres des fonctions gaussiennes des neurones de la couche cachée.
- $sig_1 \ sig_2 \ \dots \ sig_{N_{ncc}}$: Les largeurs σ des fonctions gaussiennes des neurones de la couche cachée.
- $Z_1 \ Z_2 \ \dots \ Z_{N_{ncc}}$: Les poids de connexion entre les neurones de la couche cachée et le neurone de la couche de sortie.

La longueur du chromosome (*Lch*) dépend uniquement du nombre de neurones de la couche cachée (*Nncc*) et le nombre de neurones de la couche de sortie (*Nnsc*). Son expression générale est donnée par:

$$Lch=(2+Nnsc)*max(Nncc)+1 \dots\dots\dots (3.5)$$

Par exemple pour un réseau de neurones à une sortie et le nombre de neurones de la couche cachée *Nncc* est égale à 2, la longueur du chromosome est donc *Lch=7* (le nombre de neurone *Nncc* (1 allèle); les centres des fonctions gaussiennes des neurones de la couche cachée *C₁ C₂* (2 allèles), les largeurs "sigma" des fonctions gaussiennes des neurones de la couche cachée *sig₁ sig₂* (2 allèles), les poids de connexion entre les neurones de la couche cachée et le neurone de la couche de sortie *Z₁ Z₂* (2 allèles)).

Et si *Nncc=20* la longueur du chromosome sera égale à 61.

La taille de la population *T_m* (matrice population) est donnée par la formule 3.6 :

$$T_m=N*((2+Nnsc)*max(Nncc)+1) \dots\dots\dots (3.6)$$

N: est le nombre d'individus de la population.

Par exemple si le nombre maximal des neurones de la couche cachée est égale à 20, alors la longueur des chromosomes de la population est égale à 61, dans ce cas si par exemple le nombre de neurones du chromosome *i* est égale à 6 (*Nncci=6*), il sera organisé comme suit: [*Nncci(=6) C₁ ... C₆ sig₁ ...sig₆Z₁ ...Z₆ 0 0 ...0*] (*Lch=61*)

Nncci: Nombre de neurones de la couche cachée du *i^{ème}* individu.

Exemple :

3 individus d'une population composée de 80 individus où *Nnccmax=20*,

Nncc₁=5; Nncc₃₀=20; Nncc₈₀=2

<i>Nncc₁(=5)</i>	<i>C₁</i>	<i>C₅ sig₁</i>	<i>sig₅ Z₁</i>	<i>Z₅ 0 0.....</i>	<i>0</i>
...
<i>Nncc₃₀(=20)</i>	<i>C₁</i>	<i>C₂₀ sig₁</i>	<i>sig₂₀ Z₁</i>	<i>Z₂₀</i>	<i>0</i>
...
<i>Nncc₈₀(=2)</i>	<i>C₁₁ C₁₂ sig₁ sig₂ Z₁ Z₂ 0 0</i>	<i>0</i>			

3.2. Optimisation à trois objectifs

Dans cette approche, les entrées du réseau ne sont plus fixées mais incluses dans le processus de minimisation. La structure du chromosome et le codage sont les mêmes que dans le cas du réseau MLP, soit :

$$[reg_{11} \ reg_{12} \ \dots \ reg_{15} \ \color{red}reg_{21} \ \color{red}reg_{22} \ \dots \ \color{red}reg_{25} \ Nncc \ C_1 \ C_2 \ \dots \ C_{Nncc} \ \color{red}sig_1 \ \color{red}sig_2 \ \dots \ \color{red}sig_{Nncc} \ Z_1 \ Z_2 \ \dots \ Z_{Nncc}]$$

La signification des paramètres a été donnée ci-dessus.

Dans ce cas aussi nous aurons à minimiser les trois fonctions suivantes :

1. Le nombre de neurones de la couche cachée ($Nncc$)
2. L'erreur quadratique qui est la différence entre l'entrée référence (ou sortie désirée) et la sortie réelle du modèle neuronal (dans le cas modélisation) ou la sortie du processus commandé (dans le cas contrôle).
3. Le nombre de regresseurs à l'entrée du RN.

4. Résultats d'application

Dans cette section, nous présentons l'application des approches décrites dans la section précédente à plusieurs exemples de références. Dans les exemples suivants nous présentons l'optimisation des réseaux de neurones MLP et RBF en utilisant les algorithmes génétiques multi-objectifs *NSGA2 (Non-dominated Sorting Genetic Algorithm2)* proposé par *Deb* [24, 25].

La normalisation des données est faite par la formule suivante :

$$x_{rn}(i) = 2 \frac{x(i) - x_{\min}}{x_{\max} - x_{\min}} - 1 \quad \dots \dots \dots (3.7)$$

Où $x_{rn}(i)$ est la $i^{\text{ème}}$ donnée à l'entrée du réseau de neurones.

x_{\min} et x_{\max} sont respectivement la valeur minimale et maximale du vecteur d'entrée x .

Pour toutes les applications, pour ne pas surcharger le texte, les résultats que nous présentons concernent la meilleure solution obtenue entre les deux approches à deux et trois objectifs.

4.1. Première partie : Application des réseaux MLP à la modélisation et la commande des systèmes

4.1.1. Application 1 : Modélisation du système de Box et Jenkins

Le processus de *Box et Jenkins* est une chaudière à gaz avec une entrée, le débit du gaz dans la chaudière et une sortie la concentration du CO₂ dégagé. Les données de *Box et Jenkins* consistent en 296 mesures d'entrées et 296 mesures de sorties du système [48].

Le RN optimisé a deux entrées, une sortie et une couche cachée dont le nombre de neurones (N_{ncc}) est déterminé par l'AGMO NSGA2. Les entrées du RN : X_e et e_q qui sont respectivement, le vecteur des données d'entrée du processus de Box et Jenkins et l'erreur instantanée qui est la différence entre la sortie désirée du processus et la sortie du modèle neuronal.

La sortie du RN : y_r doit suivre le vecteur y_d qui représente la sortie désirée du processus.

L'apprentissage du réseau de neurone se fait sur 100 premières données du modèle de *Box et Jenkins* et le reste des données seront utilisé pour la validation du modèle neuronal.

Les paramètres utilisés dans le processus de minimisation par NSGA II sont :

- Intervalles de recherche : $N_{ncc} \in [2, 20]$; $w \in [-1, 1]$
- La taille de la population (le nombre d'individu): $N=100$
- Nombre de générations : $gen=200$
- Probabilité de croisement : $P_c=0.9$
- Probabilité de mutation : $P_m=0.008$
- Type de codage : codage réel

4.1.1.a. Première approche : Optimisation de deux fonctions objectifs

L'algorithme NSGA2 optimise simultanément le nombre de neurones dans la couche cachée N_{ncc} et l'erreur quadratique cumulée E_c donnée par :

$$E_c = \sum_{i=1}^m (y_{ri} - y_{di})^2 \quad \dots\dots\dots (3.8)$$

Où :

Ec: L’erreur quadratique cumulée.

m =100, le nombre de donnés du vecteur d’entrée X (égale au nombre de données du vecteur de sorties désirées Yd) du modèle de Box et Jenkins

y_{ri} : La sortie du modèle neuronal qui corresponde à la i^{ème} donnée d’entrée.

y_{di} : La i^{ème} donnée de sortie désirée.

Les résultats obtenus qui représentent le front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.1.

N° d’individus	Nombre de neurones de la couche cachée N _{cc}	Erreur d’entraînement E _{en} (100 données)	Erreur de validation E _{val} (196 données)	Erreur globale Ec (296 données)
Individu 1	3	1,072371*10 ⁻¹	5,189829*10 ⁻¹	6.2622*10 ⁻¹
Individu 2	4	1,634823*10⁻²	7,001477*10⁻²	8.6363*10⁻²
Individu 3	5	1.413726*10 ⁻²	6.443274*10 ⁻²	7.857*10 ⁻²
Individu 4	6	1.191689*10 ⁻²	4.650611*10 ⁻²	5.8423*10 ⁻²
Individu 5	7	1.031713*10 ⁻²	3.532987*10 ⁻²	4.5647*10 ⁻²

Tableau 3.1 : le front de Pareto du RN MLP

(Modélisation du système de Box et Jenkins, 1^{ère} approche)

4.1.1.b. Deuxième approche : Optimisation de trois fonctions objectifs

Dans ce cas l’algorithme NSGA2 doit optimiser :

- Le nombre de neurones de la couche cachée (N_{cc}).
- L’erreur quadratique cumulée (Ec) qui est la différence entre la sortie désirée et la sortie réelle du RN (formule 3.8 paragraphe 5.1.1.a).
- Le nombre de regresseurs à l’entrée du RN (N_{reg}).

Les résultats obtenus qui représentent le front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.2.

N° d'individus	Nombre de regresseurs Nreg	Nombre de neurones de la couche cachée Nncc	Erreur d'entraînement E_en (100 données)	Erreur de validation E_val (196 données)	Erreur globale E(w) (296 données)
Individu 1	5	2	$0.7511*10^{-2}$	$2.5672*10^{-2}$	$3.3183*10^{-2}$
Individu 2	3	2	$1.7328*10^{-2}$	$7.2926*10^{-2}$	$9.0254*10^{-2}$
Individu 3	5	3	$0.3218*10^{-2}$	$1.5649*10^{-2}$	$1.8867*10^{-2}$
Individu 4	4	3	$0.6764*10^{-2}$	$2.5390*10^{-2}$	$3.2154*10^{-2}$
Individu 5	4	4	$0.4473*10^{-2}$	$1.4878*10^{-2}$	$1.9351*10^{-2}$
Individu 6	5	4	$0.3113*10^{-2}$	$1.3835*10^{-2}$	$1.6948*10^{-2}$
Individu 7	5	5	$1.9331*10^{-3}$	$7.5591*10^{-3}$	$9.4922*10^{-3}$

Tableau 3.2 : le front de Pareto du RN MLP

(Modélisation du système de Box et Jenkins, 2^{ème} approche)

Nous choisissons comme meilleure solution entre les deux approches d'optimisation, l'individu 2, du tableau 3.2. Les figures ci-dessous représentent respectivement, la concentration du CO2 dégagé à la sortie de la chaudière (sortie désirée y_d) et la sortie du modèle neuronal y_r , figure 3.1, l'erreur d'entraînement figure 3.2, l'erreur de validation figure 3.3, le débit du gaz à l'entrée de la chaudière (X_e) figure 3.4. Le front de Pareto donnant l'erreur globale cumulée en fonctions du nombre de regresseurs et du nombre de neurones est donné par la figure 3.5, pour un modèle neuronal à 3 regresseurs à l'entrées et 2 neurones à la couche cachée et une erreur globale de $9.0254*10^{-2}$ (Tableau 3.2, individus N°2). Nous avons privilégié le résultat dont l'architecture est plus petite puisque l'erreur globale des différents résultats est très proche. Les figures présentées pour cette deuxième approche sont presque similaires à celle obtenues avec la première approche.

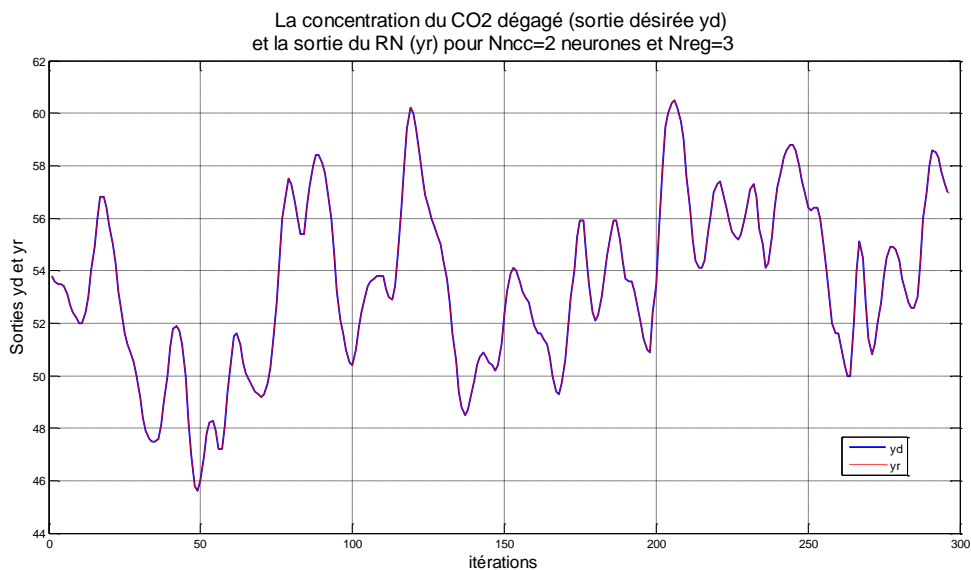


Figure 3.1 : La concentration du CO2 dégagé à la sortie de la chaudière (sortie désirée y_d) et la sortie du modèle neuronale (y_r) système de Box and Genkins

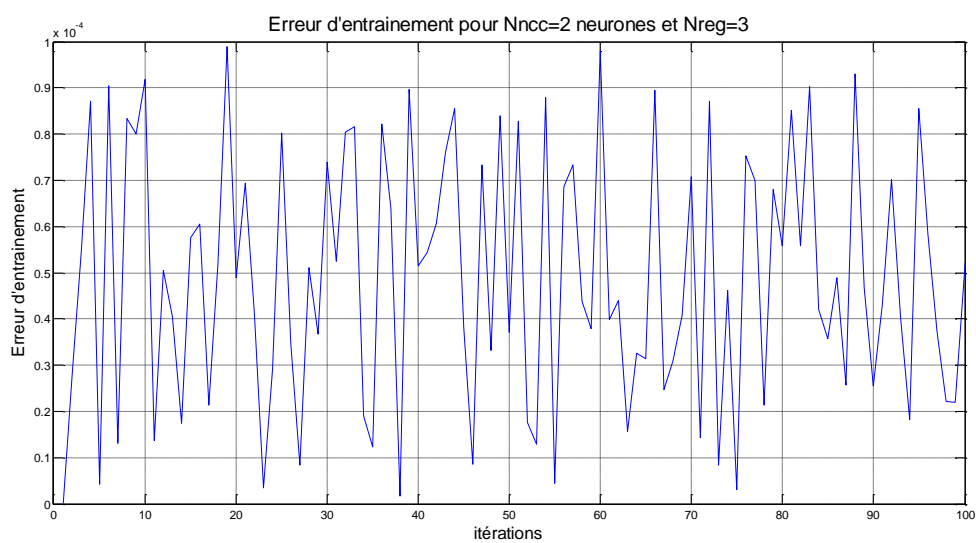


Figure 3.2 : L'erreur instantanée d'entraînement système de Box Genkins

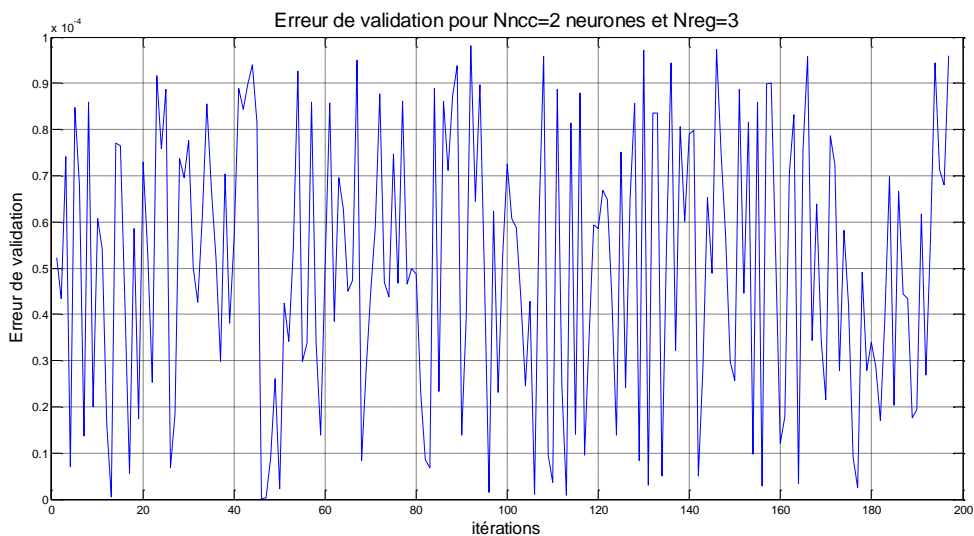


Figure 3.3 : L'erreur de validation de Box and Jenkins

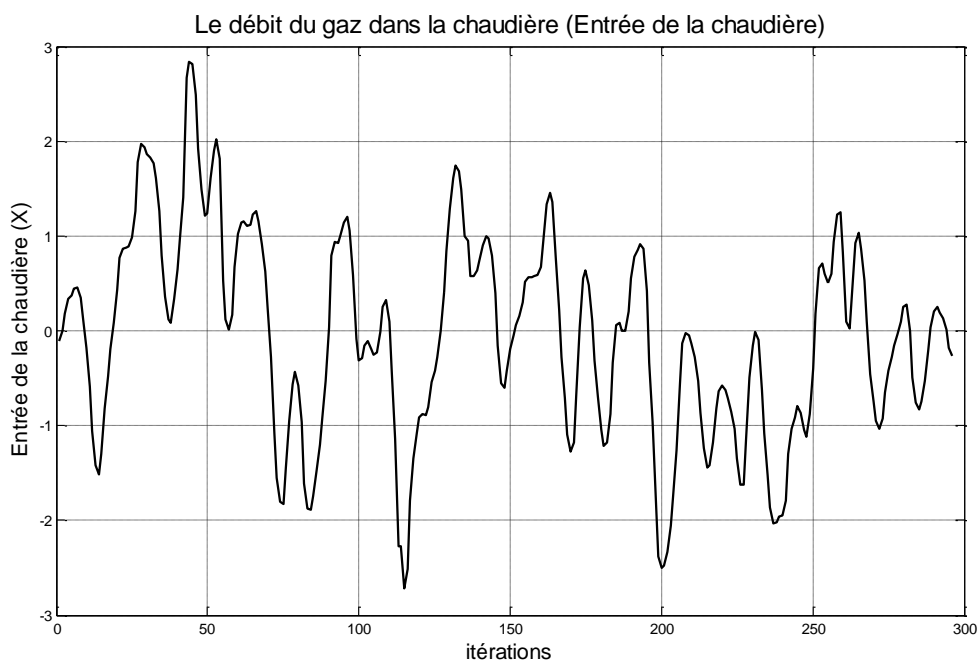


Figure 3.4 : Le débit du gaz à l'entrée de la chaudière (Xe) système de Box and Jenkins

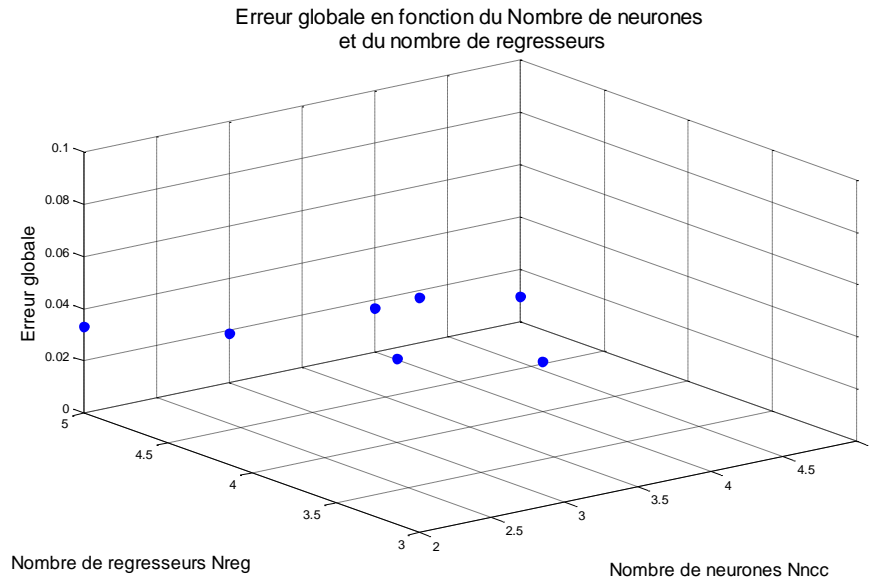


Figure 3.5: Front de Pareto, l'erreur globale en fonctions du nombre de regresseurs et du nombre de neurones, système de Box and Jenkins (Approche 2 : optimisation de trois objectifs)

4.1.2. Application 2 : Modélisation d'un système chaotique (Application à la prédiction des séries temporelles du problème Chaotique non linéaire de Mackey-Glass)

Il s'agit de modéliser le système chaotique non linéaire de *Mackey-Glass* à 1000 données [49]. La série temporelle du processus *Mackey-Glass* étudiée ici est générée par l'équation suivante:

$$\dot{x}(t) = \beta x(t) + \frac{\alpha x(t - \tau)}{1 + x^{10}(t - \tau)} \dots\dots\dots (3.9)$$

Où :

$\alpha=0.2, \beta=-0.1, \tau=17$, Tel que mentionné par *Martinetz et al.* [50].

$x(t)$ est quasi-périodiques et chaotiques avec une dimension fractale attractrice 2,1 pour les paramètres ci-dessus.

L'apprentissage du réseau de neurone se fait sur 200 données du model de *Mackey-Glass* et la validation sur le reste des données.

Les paramètres de l'algorithme génétique multi-objectifs utilisés ici sont les mêmes que ceux utilisés dans le cas du processus de Box et Jenkins (paragraphe 5.1.1).

4.1.2.a. Première approche : Optimisation de deux fonctions objectifs

L'algorithme NSGA2 optimise simultanément le nombre de neurones de la couche cachée N_{ncc} et l'erreur quadratique cumulée E_c donnée par la formule 3.8 (paragraphe 5.1.1.a).

Avec m : est le nombre de donnés du vecteur d'entrée x qui le vecteur de sorties désirées y_d du modèle de Mackey-Glass, dans cette application $m=1000$.

Les résultats obtenus qui représentent le front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.3.

N° d'individus	Nombre de Neurones N_{ncc}	Erreur d'entraînement E_{en} (200 données)	Erreur de validation E_{val} (800 données)	Erreur globale E_g (1000 données)
Individu 1	2	94,6988	402,8652	497,5640
Individu 2	3	15,7195	72,5293	88,2488
Individu 3	4	10,3344	34,9016	45,2360
Individu 4	5	4,6085	17,2876	21,8961
Individu 5	6	1,6512	7,7178	9,3690
Individu 6	7	1,3056	5,6454	6,9510
Individu 7	8	0,9011	3,4248	4,3259
Individu 8	9	0,6778	3,182	3,8598

Tableau 3.3 : le front de Pareto du RN MLP

(Modélisation du système chaotique de Mackey-Glass, 1^{ère} approche)

4.1.2.b. Deuxième approche : Optimisation de trois fonctions objectifs

L'algorithme NSGA2 doit optimiser les trois fonctions suivantes:

- Le nombre de neurones de la couche cachée (N_{ncc}).

- L'erreur quadratique cumulée (E_c) qui est la différence entre la sortie désirée et la sortie réelle du modèle neuronal (formule 3.8, paragraphe 5.1.1.a).
- Le nombre de régresseurs à l'entrée du réseau de neurones (N_{reg}).

Les résultats obtenus qui représentent le front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.4.

N° d'individus	Nombre de Régresseurs N_{reg}	Nombre de Neurones N_{ncc}	Erreur d'entraînement E_{en} (200 données)	Erreur de validation E_{val} (800 données)	Erreur globale E_g (1000 données)
Individu 1	1	2	162.7419	729.2257	891.9676
Individu 2	1	4	14.3541	58.7677	73.1218
Individu 3	2	2	85.0923	363.0909	448.1831
Individu 4	2	3	10.6110	31.3461	41.9572
Individu 5	3	2	6.5238	19.7742	26.2980
Individu 6	3	5	0.8865	2.6161	3.5026
Individu 7	4	2	6.1505	20.0912	26.2417
Individu 8	5	2	5.6965	18.4464	24.1429
Individu 9	5	3	5.2020	17.2150	22.4171

Tableau 3.4 : le front de Pareto du RN MLP

(Modélisation du système chaotique de Mackey-Glass, 2^{ème} approche)

L'individu 6 de la deuxième approche est choisi comme meilleure solution. Les figures ci-dessous représentent respectivement, la sortie désirée (y_d) et la sortie du modèle neuronal (y_r) figure 3.6, l'erreur d'entraînement figure 3.7, l'erreur de validation figure 3.8. Le front de Pareto de la deuxième approche avec l'erreur globale en fonctions du nombre de régresseurs et du nombre de neurones est donné par la figure 3.9 pour un modèle neuronal à 3 régresseurs à l'entrée et 5 neurones à la couche cachée et une erreur globale de 3.5026 (Tableau 3.4, individus N°6). Dans ce cas, nous avons privilégié le résultat dont l'erreur globale est la plus petite puisqu'il y a une différence importante entre cette valeur et celle des autres résultats. Les figures présentées pour cette approche sont presque similaires à celle obtenues avec la première approche pour le réseau à 9 neurones dans la couche cachée et une erreur globale de 3,8598.

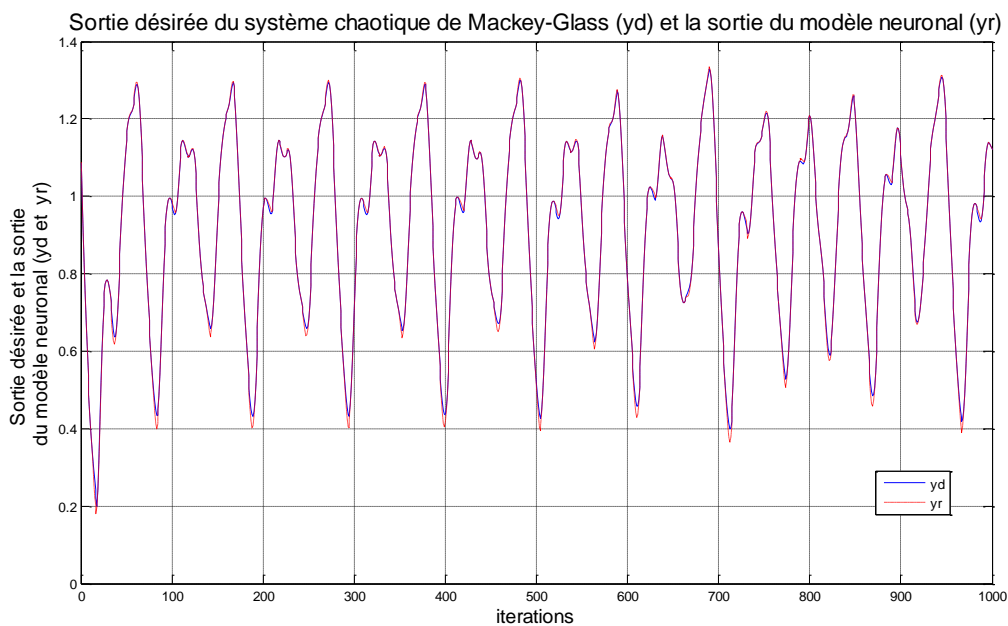


Figure 3.6 : Sortie désirée du système chaotique de Mackey-Glass (y_d) et la sortie du modèle neuronal (y_r)

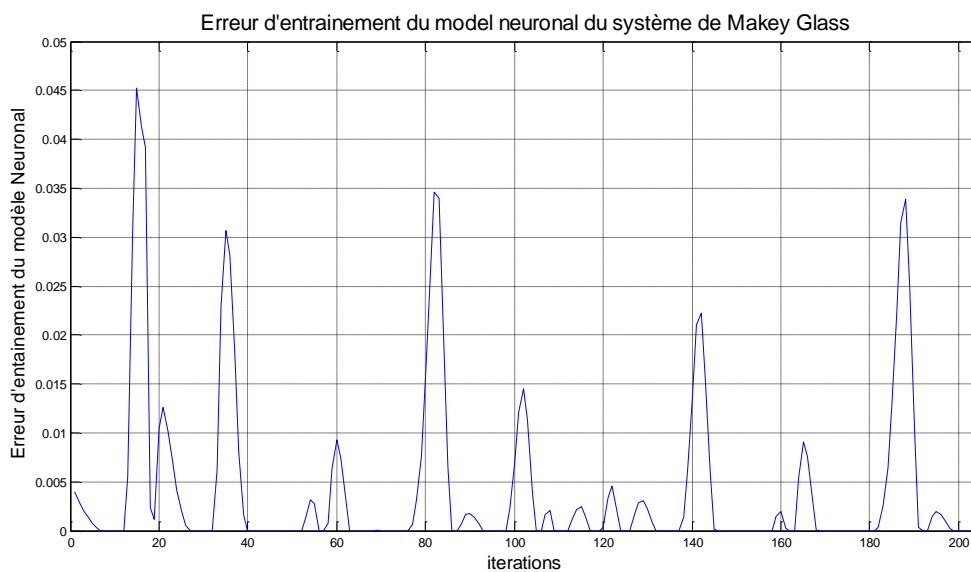


Figure 3.7 : Erreur instantanée d'entraînement du système chaotique de Mackey-Glass

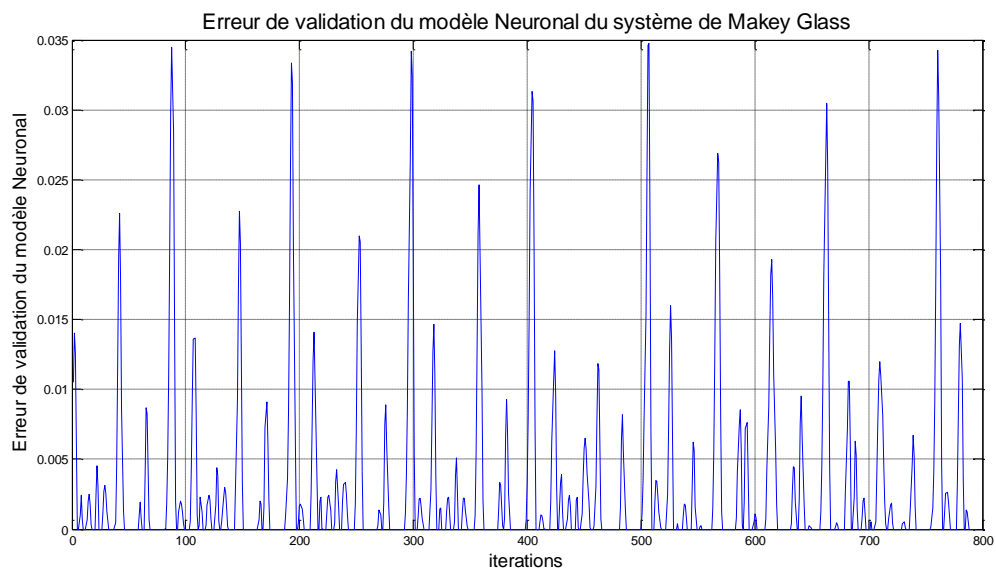


Figure 3.8 : L'erreur instantanée de validation du système chaotique de Mackey-Glass

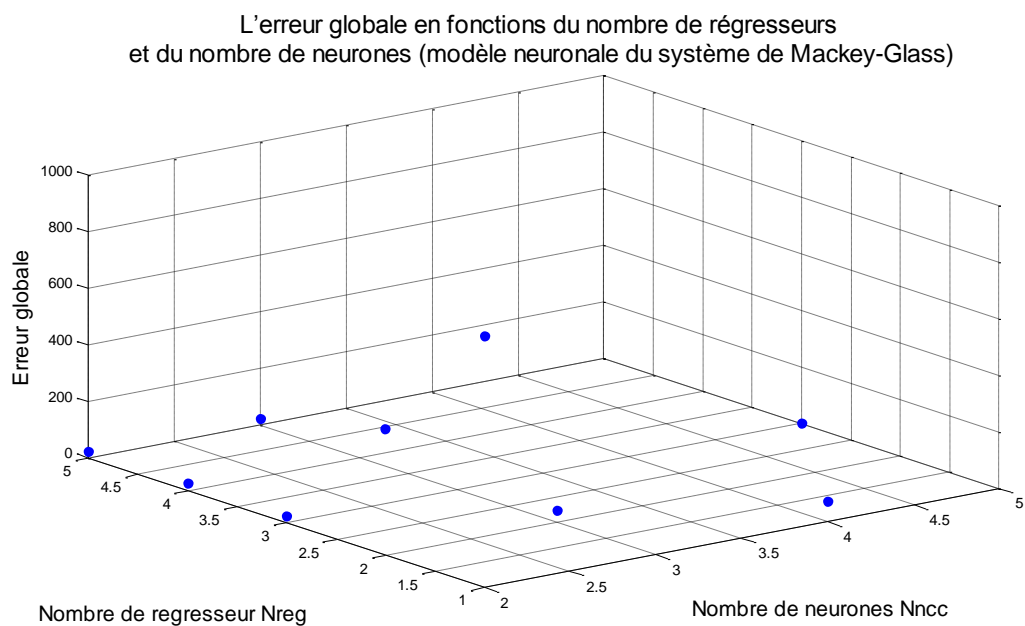


Figure 3.9 : Système chaotique : l'erreur globale en fonctions du Nombre de neurones et du nombre de regresseurs.

4.1.3. Application 3 : Commande d'un processus à comportement variable simulé par un modèle d'état

Dans ce paragraphe, nous considérons un problème de commande d'un système non linéaire, pris souvent comme système de référence. Le modèle du système est donné par les équations d'état suivantes [51]:

$$\begin{cases} x_1(k+1) = a_1 x_1(k) + a_2 x_2(k) + (b_1 + 2b_2)u(k) \\ x_2(k+1) = \frac{x_1(k)}{1 + 0.01x_2(k)^2} - \frac{b_2}{a_2}u(k) \\ y(k) = 4 \tanh\left(\frac{x_1(k)}{4}\right) \end{cases} \dots\dots\dots (3.10)$$

Avec : $a_1 = 1,145$; $a_2 = -0,549$; $b_1 = 0,222$; $b_2 = 0,181$.

$y(k)$ est la sortie du processus, $u(k)$ est le signal de commande.

Le comportement de ce processus est intéressant. Autour de zéro, le processus a une dynamique linéaire oscillante et un gain unité. Le dénominateur de la deuxième variable d'état x_2 lui donne un caractère de plus en plus amorti et plus lent pour des amplitudes supérieures à 1, il diminue aussi le gain, ce qui est encore accentué par la tangente hyperbolique de l'équation d'observation ($y(k)$) [51].

Le contrôleur à RN MLP est conçu en hors-ligne en utilisant les algorithmes génétiques multi-objectifs NSGA2.

Les paramètres de l'algorithme génétique multi-objectifs sont les mêmes utilisés dans les applications précédentes et mentionnés au paragraphe 4.1.1, mais avec un nombre de générations : gen=300.

4.1.3.1. Première approche : Optimisation de deux fonctions objectifs

L'algorithme NSGA2 optimise simultanément le nombre de neurones dans la couche cachée N_{ncc} et l'erreur quadratique cumulée E_c qui est dans ce cas la différence entre l'entrée référence (ou sortie désirée) et la sortie du processus commandé donnée par la formule 3.8 (paragraphe 4.1.1.a), avec y_r représente la sortie du processus commandé.

Dans cette étude, les entrées du contrôleur neuronal sont la sortie désirée qui est la référence (R_{ef}), l'erreur quadratique instantanées (eq) et sa sortie est la commande (u_c) appliquée au processus. Le schéma de la figure 3.10 illustre la stratégie d'optimisation hors ligne adoptée. L'entraînement est réalisé pour une référence variable donnée par la figure 3.11. Lorsque le processus d'optimisation est terminé, on extrait le contrôleur du front de Pareto de la dernière génération. Pour la validation, le contrôleur sera alors placé dans la boucle pour contrôler le système pour une autre référence variable. Un test de robustesse renforcera la validation.

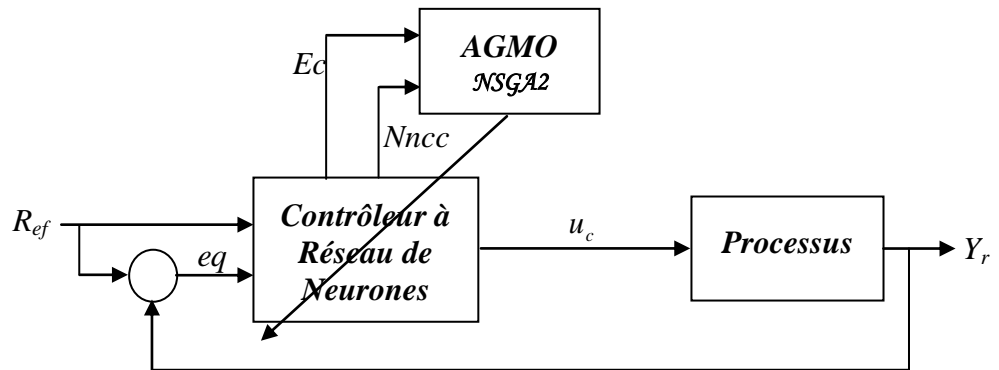


Figure 3.10 : stratégie de la commande neuronale optimisée par l'AGMO NSGA2

Les résultats donnés par le front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.5.

N° d'individu	Nombre de Neurones N _{ncc}	Erreur Cumulée E _c
Individu 1	2	19.2394
Individu 2	3	3.1341
Individu 3	4	0.1459
Individu 4	5	0.0886
Individu 5	6	0.0532

Tableau 3.5 : le front de Pareto du RN MLP

(Commande d'un système simulé par un modèle d'état, 1^{ère} approche)

4.1.3.2. Deuxième approche : Optimisation de trois fonctions objectifs

De la même façon que ci dessus, l'algorithme *NSGA2* doit optimiser simultanément les trois fonctions suivantes:

- Le nombre de neurones de la couche cachée N_{ncc} .
- L'erreur quadratique cumulée E_c qui est la différence entre la référence désirée et la sortie réelle du processus (formule 3.8 paragraphe 4.1.1.a).
- Le nombre de regresseurs à l'entrée du réseau de neurones N_{reg} .

Dans la deuxième approche de cette application de commande, à l'entraînement, les entrées possibles du contrôleur neuronal sont la commande appliquée au processus ($u_c(k)$) et ses trois états précédents ($u_c(k-1)$, $u_c(k-2)$, $u_c(k-3)$), l'erreur quadratique instantanées (eq), et la sortie réelle du processus contrôlé ($y_r(k)$) et ses états précédents ($y_r(k-1)$... $y_r(k-4)$). Comme indiquée plus haut la mise à un d'une de ces entrées dans le chromosome, entraînera son inclusion dans les entrées du réseau.

Les résultats donnés par le front de *Pareto* (individus non dominés) de la dernière génération sont donnés dans le tableau 3.6.

N° d'individus	Nombre de Regresseurs N_{reg}	Nombre de Neurones N_{ncc}	Erreur Cumulée E_c
Individu 1	1	2	26.7944
Individu 2	1	7	21.9363
Individu 3	2	3	20.0484
Individu 4	2	6	18.8423
Individu 5	3	3	8.8501
Individu 6	3	4	0.0294
Individu 7	3	6	0.0287
Individu 8	4	5	0.0292
Individu 9	5	4	4.6413

Tableau 3.6 : le front de Pareto du RN MLP

(Commande d'un système simulé par un modèle d'état, 2^{ème} approche)

4.1.3.3. Résultats d'entraînement

L'individu 6 du tableau 4.6 est choisi comme solution. Les figures ci-dessous donnent les résultats d'entraînement, elles représentent respectivement, la référence ou sortie désirée (R_{ef} ou y_d) et la sortie réelle du processus contrôlé (y_r) figure 3.11, l'erreur quadratique instantanée, figure 3.12, le signal de commande, figure 3.13. Le front de Pareto de la deuxième approche, avec l'erreur globale cumulée en fonctions du nombre de regresseurs et du nombre de neurones est donné par la figure 3.14, pour un contrôleur neuronal à 3 regresseurs à l'entrées et 4 neurones à la couche cachée et une erreur globale cumulée de 0.0294 (Tableau 3.6, individu N°6). Dans ce cas, nous avons privilégié le résultat dont l'architecture du réseau est la plus petite pour une erreur globale de l'ordre de 0.02, puisqu'il y a une différence importante entre cette valeur et celle des autres résultats, nous avons favorisé la plus petite structure des trois individus 6,7 et 8, où l'erreur est de l'ordre de 0.02.

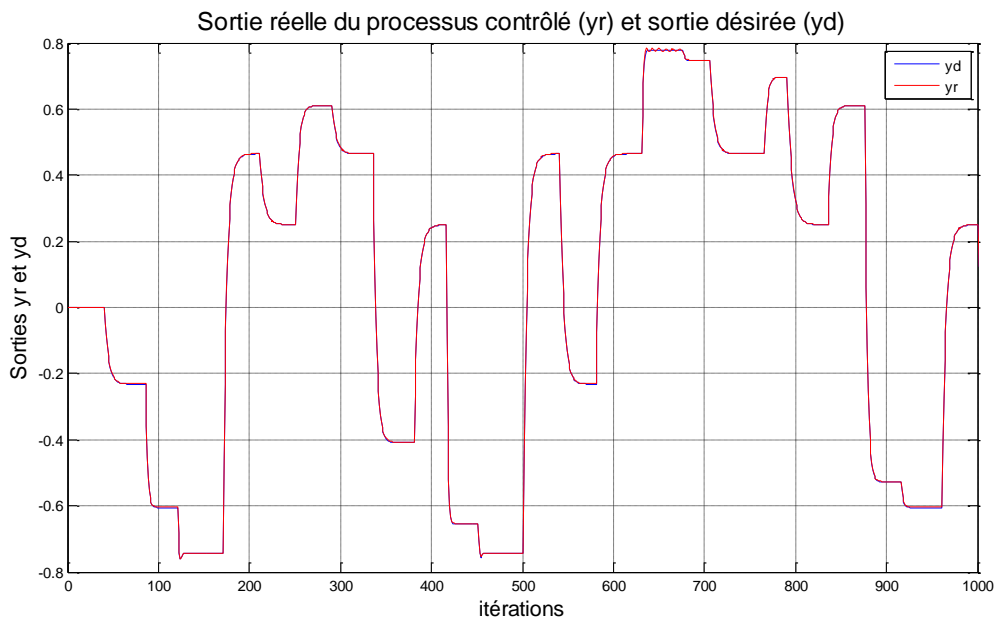


Figure 3.11 : La référence ou sortie désirée (R_{ef} ou y_d) et la sortie réelle du processus contrôlé (y_r)

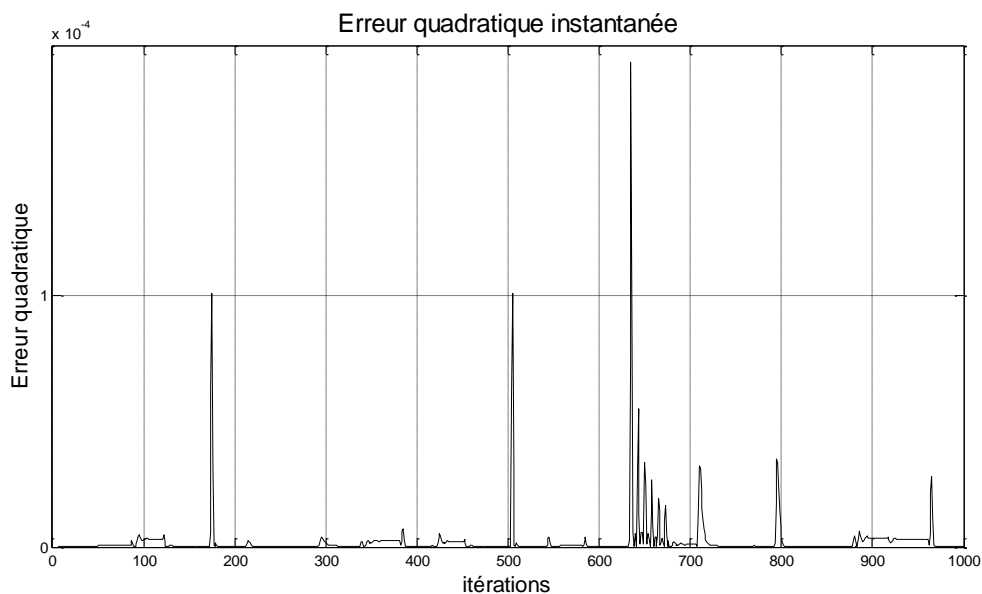


Figure 3.12 : L'erreur quadratique instantanée (eq)

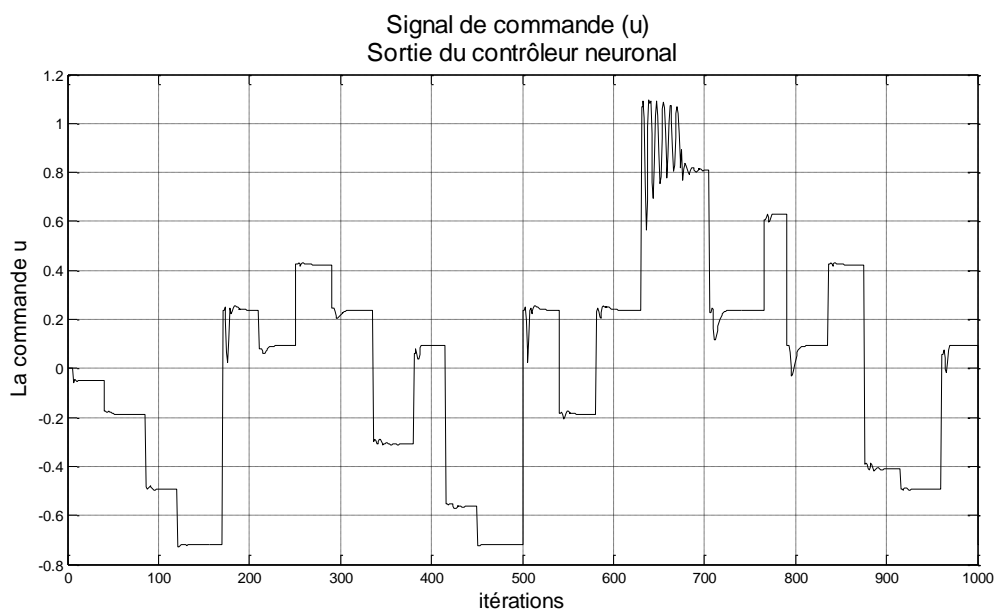


Figure 3.13 : le signal de commande (sortie du contrôleur neuronal)

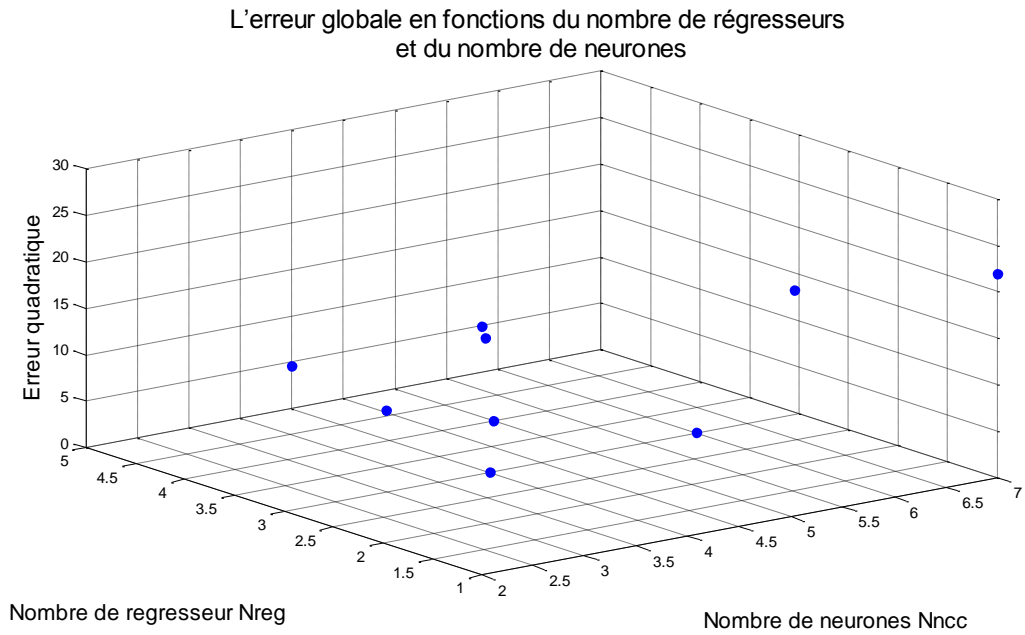


Figure 3.14 : l'erreur quadratique cumulée en fonctions du nombre de régresseurs et du nombre de neurones

4.1.3.4. Résultats de validation

Après avoir extrait le contrôleur neuronal choisi parmi ceux du front de *Pareto*, nous avons modifié la référence désirée afin de vérifier si notre contrôleur va s'adapter. Les figures ci-dessous représentent les résultats de validation obtenus. Elles représentent respectivement, la référence de validation ou sortie désirée (R_{efval}) et la sortie réelle du processus contrôlé (y_r) figure 3.15, l'erreur quadratique instantanée, figure 3.16.

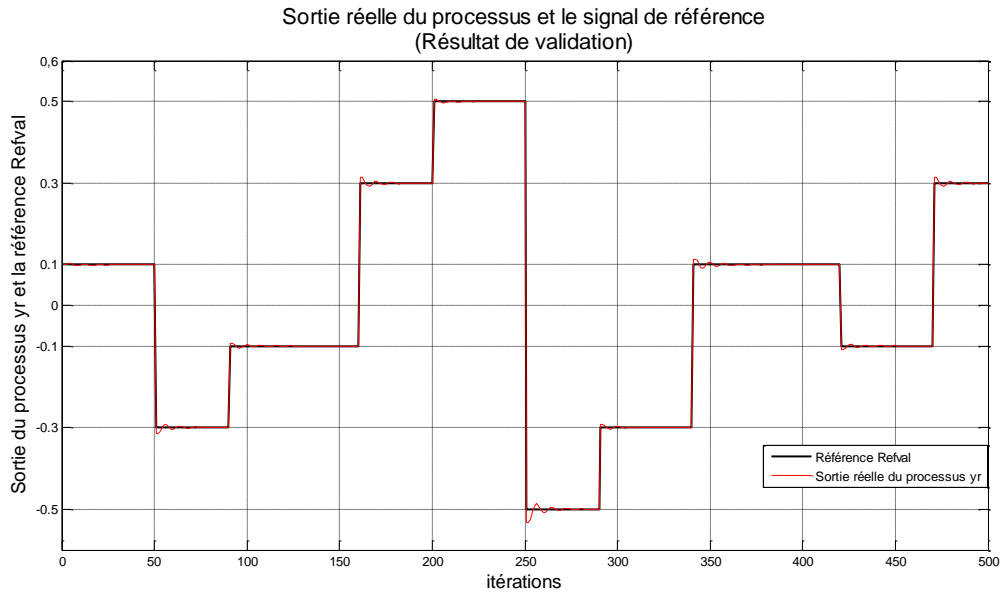


Figure 3.15 : La référence de validation ou sortie désirée (R_{efval}) et la sortie réelle du processus contrôlé (y_r)

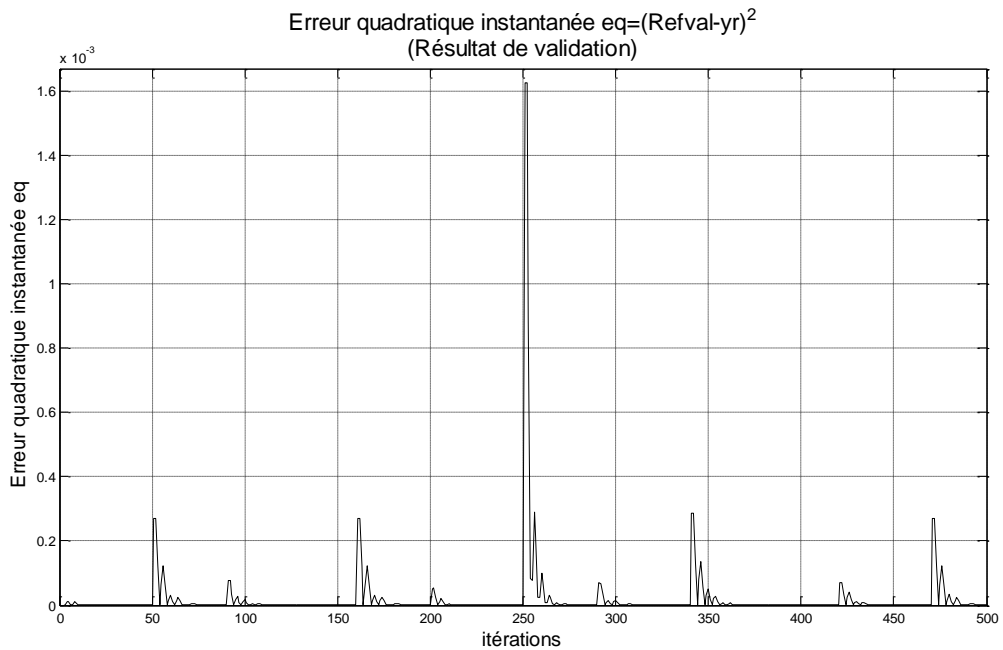


Figure 3.16 : L'erreur quadratique instantanée (e_q) du résultat de validation

D'après ces résultats, nous remarquons que le contrôleur suit la nouvelle référence, s'adapte bien aux nouvelles données avec une erreur quadratique cumulée de 0.0107, malgré les très faibles oscillations observées avec un dépassement maximal de 0.0403 voire une erreur quadratique instantanée de $1.62409 \cdot 10^{-3}$ aux itérations 251 et 252.

4.1.3.5. Test de la robustesse

Afin de tester la robustesse du contrôleur neuronal choisi, nous avons introduit une perturbation de $\pm 10\%$ et sur les coefficients (a_1 , a_2 , b_1 et b_2) du processus (équations d'état 3.10). Après essais, nous avons remarqué que les perturbations sur les paramètres b_1 et b_2 , ont un effet très faible sur le comportement du système, par contre les paramètres a_1 et a_2 ont un effet important sur le comportement du système. L'effet des perturbations sur les paramètres a_1 et a_2 est presque le même, la figure 3.17 représente les sorties réelles du processus contrôlé pour des perturbations de $\pm 10\%$ sur le paramètre a_1 (courbes en couleurs bleu et noire) et la sortie réelle du processus sans perturbations (courbe en couleur rouge).

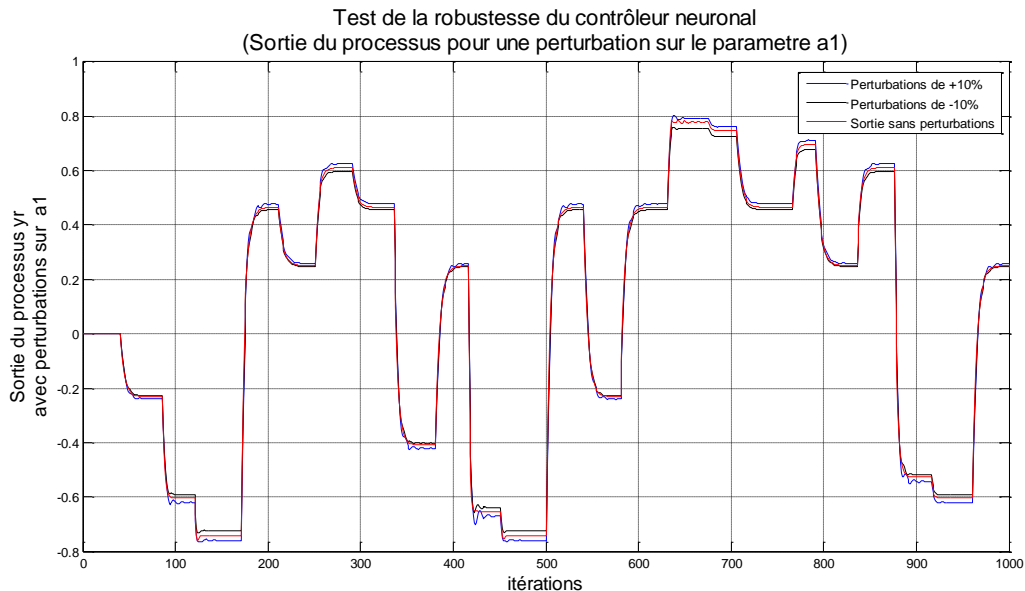


Figure 3.17 : Sorties réelles du processus contrôlé pour des perturbations le paramètre a_1

Dans la figure 3.17, nous pouvons remarquer que le contrôleur a pratiquement suivi la sortie réelle du processus sans perturbations malgré l'existence d'erreurs statiques qui atteignent une valeur maximale de 0.028 et le comportement du processus n'a pas été dégradé.

D'après ces résultats nous constatons que ce contrôleur neuronal optimisé par les algorithmes génétiques multi-objectifs *NSGA2* a montré son efficacité pour s'adapter aux nouvelles situations.

4.2. Deuxième partie : Application des réseaux RBF à la modélisation et la commande des systèmes

Dans cette section, nous considérons l'optimisation des RBF avec, pour pouvoir comparer les résultats, les mêmes exemples que pour les réseaux MLP. Nous noterons que le nombre de paramètres est plus important dans le cas des RBF. En effet, en plus des poids des connexions, nous avons les centres et les largeurs des fonctions de base.

4.2.1. Application 1 : Modélisation du système de Box et Jenkins

La sortie du RN RBF: y_r doit suivre le vecteur de la sortie désirée du processus y_d . L'apprentissage du réseau de neurone se fait sur 100 données du model de *Box* et *Jenkins* et la validation sur le reste des données. Les paramètres de l'algorithme utilisé sont les mêmes utilisés dans le RN MLP (paragraphe 3.1.1).

4.2.1.a. Première approche : Optimisation de deux fonctions objectifs

L'algorithme NSGA2 optimise simultanément le nombre de neurones dans la couche cachée N_{ncc} et l'erreur quadratique cumulée E_c .

Les résultats obtenus du front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.9.

N° d'individus	Nombre de neurones de la couche cachée N_{ncc}	Erreur d'entraînement E_{en} (100 données)	Erreur de validation E_{val} (196 données)	Erreur globale $E(w)$ (296 données)
Individu 1	2	6,8344	12,1503	18.9847
Individu 2	3	5,8027	9,8297	15.6324
Individu 3	4	3,6376	6,4649	10.1025
Individu 4	6	3,3507	5,6476	8.9983
Individu 5	7	1,4346	2,5505	3.9851
Individu 6	8	0,5285	0,8063	1.3348
Individu 7	9	0.2182	0.4015	0.6197

Tableau 3.9 : le front de Pareto du RN RBF

(Modélisation du système de Box et Jenkins, 1^{ère} approche)

4.2.1.b. Deuxième approche : optimisation de trois fonctions objectifs

Dans ce cas l'algorithme *NSGA2* doit optimiser les trois fonctions suivantes:

- Le nombre de neurones de la couche cachée (N_{ncc})
- L'erreur quadratique cumulée E_c qui est la différence entre la sortie désirée et la sortie réelle du modèle neuronal (formule 3.8 paragraphe 4.1.1.a).
- Le nombre de regresseurs à l'entrée du réseau de neurones N_{reg} .

Les résultats donnés par le front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.10.

N° d'individus	Nombre de regresseurs N_{reg}	Nombre de neurones de la couche cachée N_{ncc}	Erreur d'entraînement E_{en} (100 données)	Erreur de validation E_{val} (196 données)	Erreur globale E_c (296 données)
Individu 1	4	2	8,6137	14,9212	23,5349
Individu 2	5	2	7,9511	13,303	21,2541
Individu 3	3	3	4,0386	7,2899	11,3285
Individu 4	4	3	3,3029	5,7812	9,0841
Individu 5	5	4	2,0436	3,9111	5,9547
Individu 6	5	5	0,7083	1,4171	2,1254
Individu 7	5	7	0,5015	1,0448	1,5463
Individu 8	6	8	0,2679	0,4557	0.7236

Tableau 3.10 : le front de Pareto du RN RBF

(Modélisation du système de Box et Jenkins, 2^{ème} approche)

L'individu 7 du tableau 3.9 de l'optimisation à deux objectifs est choisi comme solution. Les figures ci-dessous représentent respectivement, la concentration du CO₂ dégagé à la sortie de la chaudière (sortie désirée y_d) et la sortie du modèle neuronale (y_r), figure 3.18, l'erreur d'entraînement, figure 3.19, l'erreur de validation, figure 3.20. Le front de Pareto de la première approche avec l'erreur globale cumulée en fonctions du nombre de neurones (résultats du tableau 3.9) est donné par la figure 3.21, pour un modèle neuronal à 9 neurones à la couche cachée et une erreur globale de 0.6197 (Tableau 3.9, individu N°7). Dans ce cas,

nous avons privilégié le résultat dont l'erreur globale est la plus petite, puisqu'il y a une différence importante entre cette valeur et celle des autres résultats.

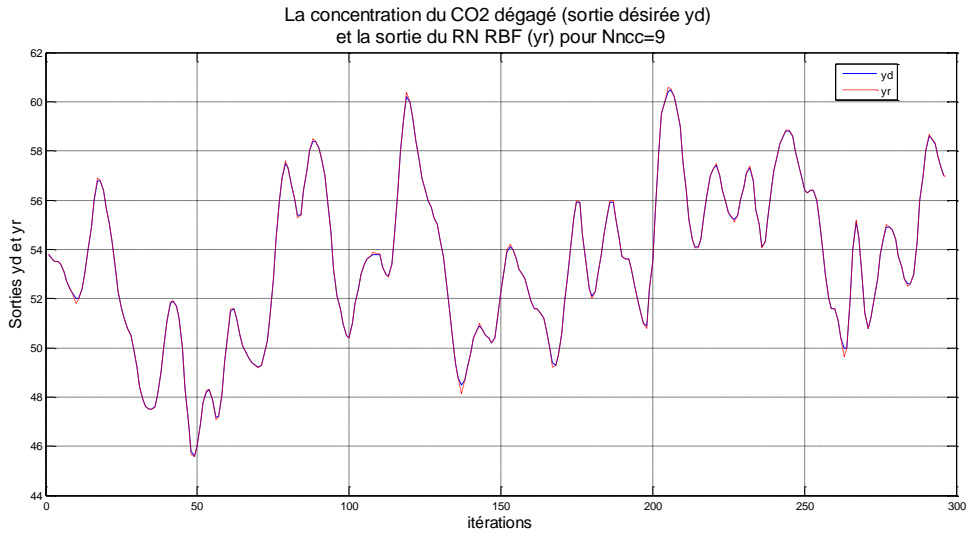


Figure 3.18 : La sortie désirée (y_d) (concentration du CO2 dégagé à la sortie de la chaudière) et la sortie du modèle neuronale (y_r)

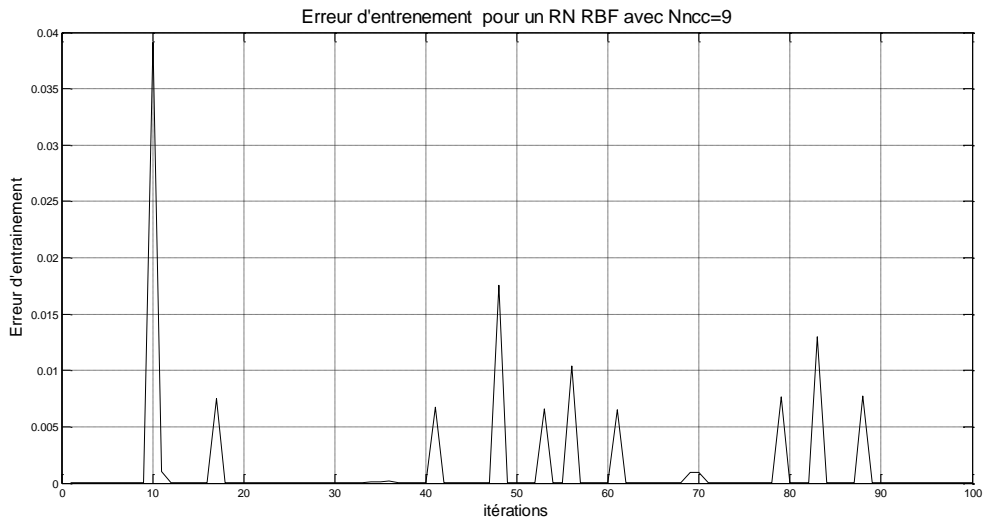


Figure 3.19 : L'erreur d'entraînement

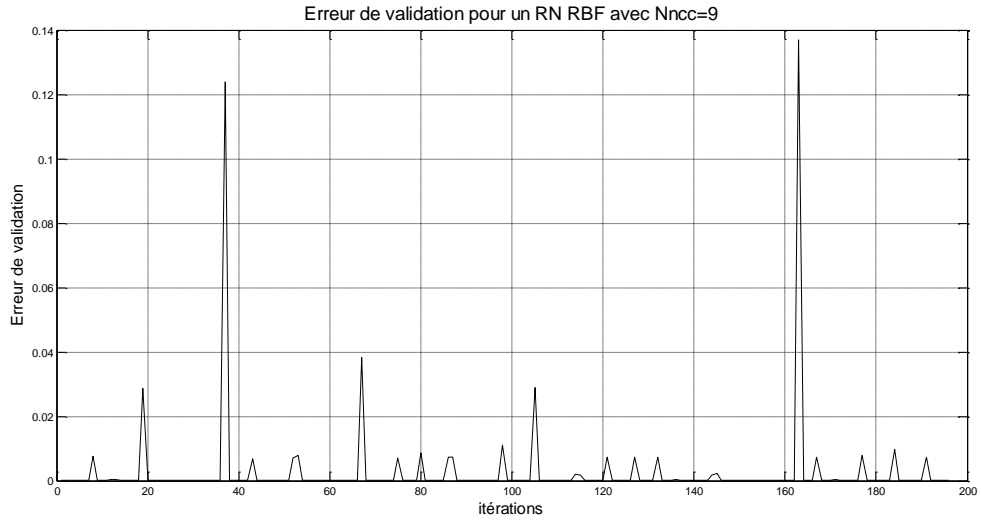


Figure 3.20 : L'erreur de validation

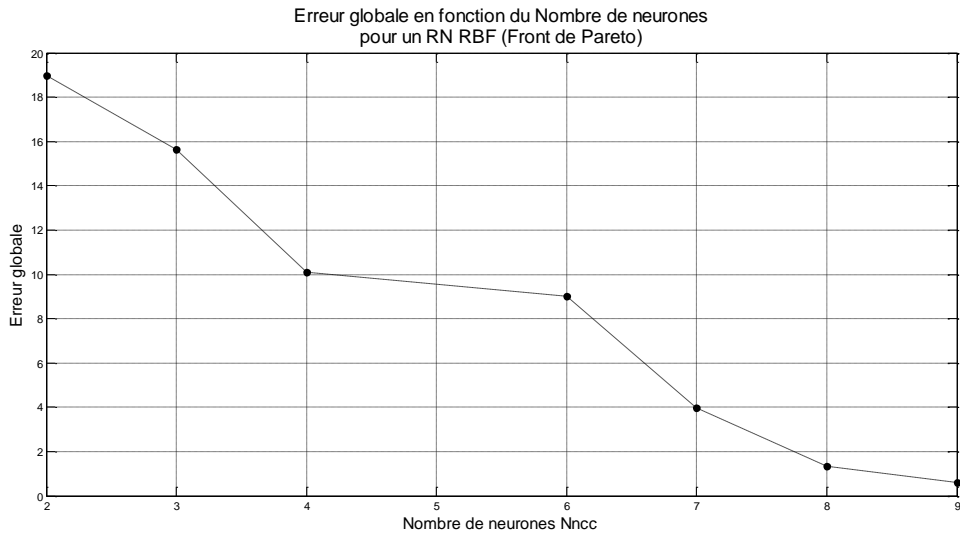


Figure 3.21 : le front de Pareto de la première approche
l'erreur en fonctions du nombre de neurones»

4.2.2. Application 2 : Modélisation d'un système chaotique (Application à la prédiction des séries temporelles du problème Chaotique non linéaire de Mackey-Glass)

De même que la première partie, il s'agit de modéliser et prédire le système chaotique non linéaire de *Mackey-Glass* (M-G) à 1000 données. La Série

temporelle du processus *Mackey-Glass* étudiée ici est générée par l'équation différentielle 3.9 mentionnée au paragraphe 4.1.2.

L'apprentissage du réseau de neurone se fait sur 200 données du model de *Mackey-Glass* et la validation sur le reste des données (800 données).

Les paramètres de l'algorithme génétique multi-objectifs utilisés ici sont les mêmes utilisés à ce système dans la première partie (application des RN MLP) et mentionnés au paragraphe 4.1.1.

4.2.2.1. Système M-G : Optimisation de deux fonctions objectifs

L'algorithme NSGA2 optimise simultanément le nombre de neurones de la couche cachée N_{ncc} et l'erreur quadratique cumulée E_c donnée par la formule 3.8 (paragraphe 4.1.1.a).

Les résultats du front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.11.

N° d'individus	Nombre de Neurones N_{ncc}	Erreur d'entraînement E_{en} (200 données)	Erreur de validation E_{val} (800 données)	Erreur globale E_g (1000 données)
Individu 1	2	27,1305	75,5649	102,6954
Individu 2	3	22,0483	71,6065	93,6548
Individu 3	4	14,4543	49,6692	64,1235
Individu 4	5	5,7469	20,9682	26,7151
Individu 5	6	3,0391	10,5223	13,5614
Individu 6	7	1,4566	4,8303	6.2869

Tableau 3.11 : le front de Pareto du RN RBF
(Modélisation du système chaotique de Mackey-Glass, 1^{ère} approche)

4.2.2.2 Deuxième approche : Optimisation de trois fonctions objectifs

L'algorithme NSGA2 doit optimiser les trois fonctions suivantes:

- Le nombre de neurones de la couche cachée (N_{ncc}).
- L'erreur quadratique cumulée E_c qui est la différence entre la sortie désirée et la sortie réelle du modèle neuronal (formule 3.8 paragraphe 5.1.1.a).
- Le nombre de regressseurs à l'entrée du réseau de neurones N_{reg} .

Les résultats obtenus qui représentent le premier front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.12.

N° d'individus	Nombre de Regresseurs Nreg	Nombre de Neurones Nncc	Erreur d'entraînement E_en (200 données)	Erreur de validation E_val (800 données)	Erreur globale E_g (1000 données)
Individu 1	2	3	27,1861	86,1694	113,3555
Individu 2	2	4	15,0282	54,0954	69,1236
Individu 3	2	5	5,5291	20,7998	26,3289
Individu 4	3	7	2,3049	7,5602	9,8651
Individu 5	4	6	1,8531	5,5592	7,4123
Individu 6	4	7	1,7135	5,3736	7,0871
Individu 7	5	9	1,4425	4,8899	6,3324

Tableau 3.12: le front de Pareto du RN RBF

(Modélisation du système chaotique de Mackey-Glass, 2^{ème} approche)

L'individu 6 du tableau 3.11 est choisi comme solution. Les figures ci-dessous représentent respectivement, la sortie désirée (y_d) et la sortie du modèle neuronal (y_r) figure 3.22, l'erreur d'entraînement figure 3.23, l'erreur de validation figure 3.24 et le front de Pareto, figure 3.25, pour un modèle neuronal à 7 neurones à la couche cachée et une erreur globale de 6.2869 (Tableau 3.11, individus N°6). Nous avons privilégié le résultat dont l'erreur globale est la plus petite puisqu'il y a une différence importante entre cette valeur et celle des autres résultats.

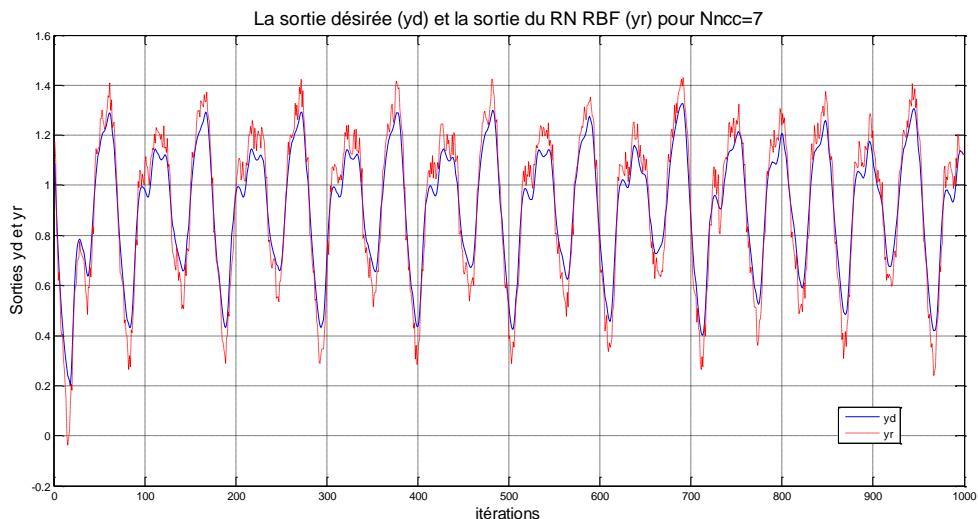


Figure 3.22 : Système M-G : la sortie désirée (y_d) et la sortie du modèle neuronal (y_r)

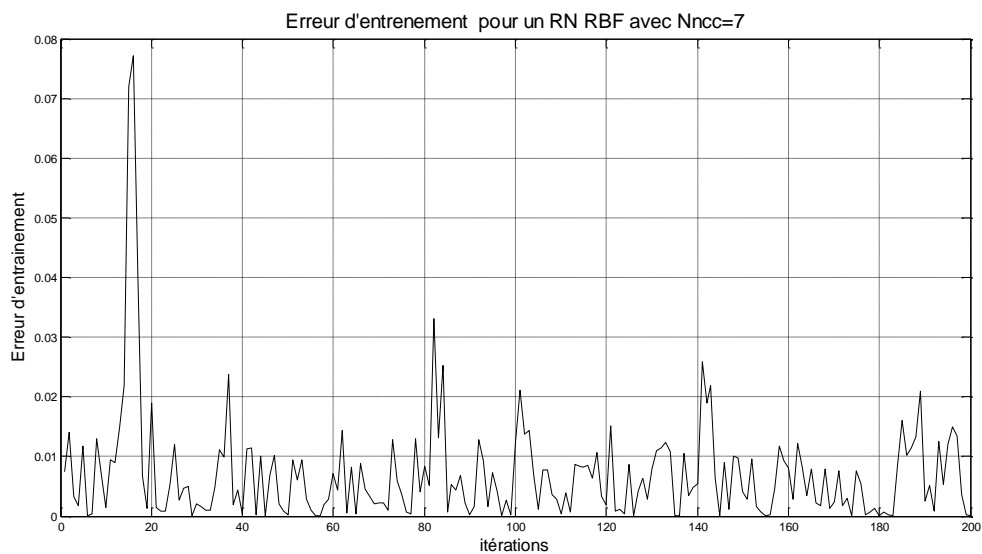


Figure 3.23 : L'erreur d'entraînement pour le système M-G

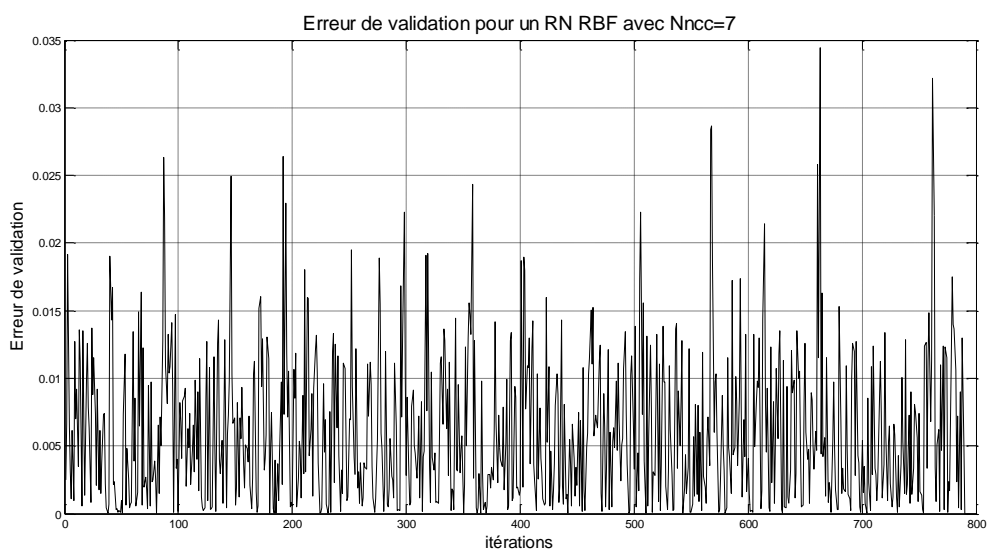


Figure 3.24 : L'erreur de validation pour le système M-G

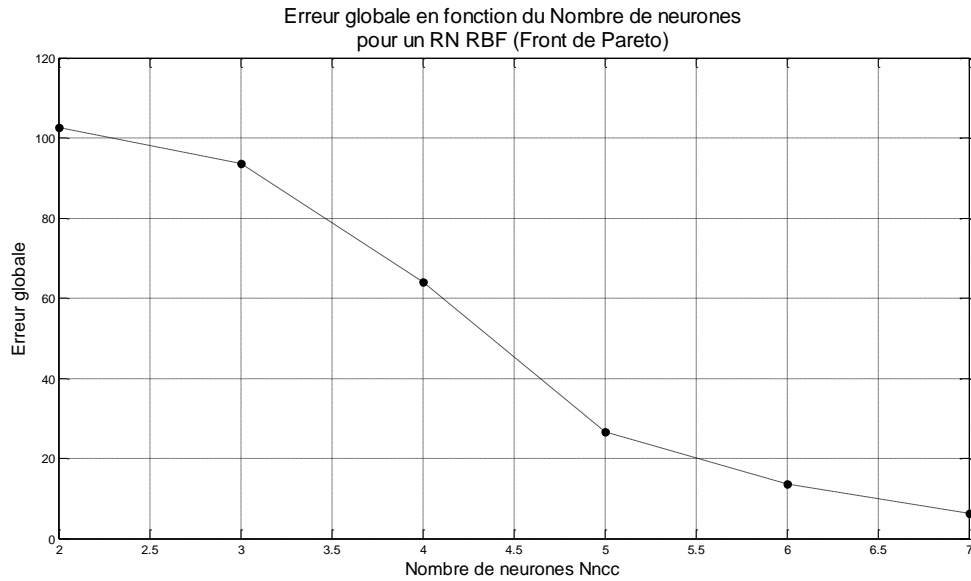


Figure 3.25 : Le front de Pareto pour le système M-G

4.2.3. Application 3 : Commande d'un processus à comportement variable simulé par un modèle d'état

De même que la première partie, il s'agit de commander le processus donné par les équations d'état 3.10, mentionnées au paragraphe 4.1.3.

Le contrôleur à RN RBF est conçu en hors-ligne en utilisant les algorithmes génétiques multi objectifs NSGA2. Les paramètres de l'algorithme génétique multi-objectifs sont les même utilisés pour ce processus avec le RN MLP.

4.2.3.1. Première approche : Optimisation de deux fonctions objectifs

L'algorithme NSGA2 optimise simultanément le nombre de neurones dans la couche cachée Nncc et l'erreur quadratique cumulée E_c qui est dans ce cas aussi, la différence entre l'entrée référence (ou sortie désirée) et la sortie du processus commandé donnée par la formule 3.8 (paragraphe 4.1.1.a), avec y_r représente la sortie du processus commandé.

Les résultats du front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.13.

N° d'individus	Nombre de Neurones N _{cc}	Erreur Cumulée E _c
Individu 1	2	19.8147
Individu 2	3	14.9058
Individu 3	4	11.1270
Individu 4	5	8.9134
Individu 5	6	6.6324
Individu 6	7	4.0975
Individu 7	9	3.4268

Tableau 3.13 : le front de Pareto du RN RBF

(Commande d'un système à comportement variable, 1^{ère} approche)

4.2.3.2. Deuxième approche : Optimisation de trois fonctions objectifs

Les résultats du front de Pareto (individus non dominés) de la dernière génération sont donnés dans le tableau 3.14

N° d'individus	Nombre de Regresseurs N _{reg}	Nombre de Neurones N _{cc}	Erreur Cumulée E _c
Individus 1	2	3	23.1576
Individus 2	2	7	17.9572
Individus 3	3	4	8.5469
Individus 4	3	6	5.9649
Individus 5	4	6	4.4854
Individus 6	4	9	3.7525

Tableau 3.14 : le front de Pareto du RN RBF

(Commande d'un système à comportement variable, 2^{ème} approche)

4.2.3.3. Résultats d'entraînement

L'individu 7 du tableau 3.13 est choisi comme solution. Les figures ci-dessous représentent les résultats d'entraînement obtenus, elles représentent respectivement, la référence ou sortie désirée (R_{ef} ou y_d) et la sortie réelle du processus contrôlé (y_r) figure 3.26, l'erreur quadratique cumulée (E_c), figure 3.27, le signal de commande, figure 3.28, le front de Pareto de la première approche, figure 3.29, pour un contrôleur neuronal à 9 neurones à la couche cachée et une erreur globale cumulée de 3.4268 (Tableau 3.13, individus N°7). Dans ce cas nous

avons privilégié le résultat dont l'erreur globale est plus petite puisqu'il y a une différence importante entre cette valeur et celle des autres résultats, même le résultat de la plus proche erreur (celui de l'individu N°6, tableau 3.13) donne une mauvaise réponse du processus.

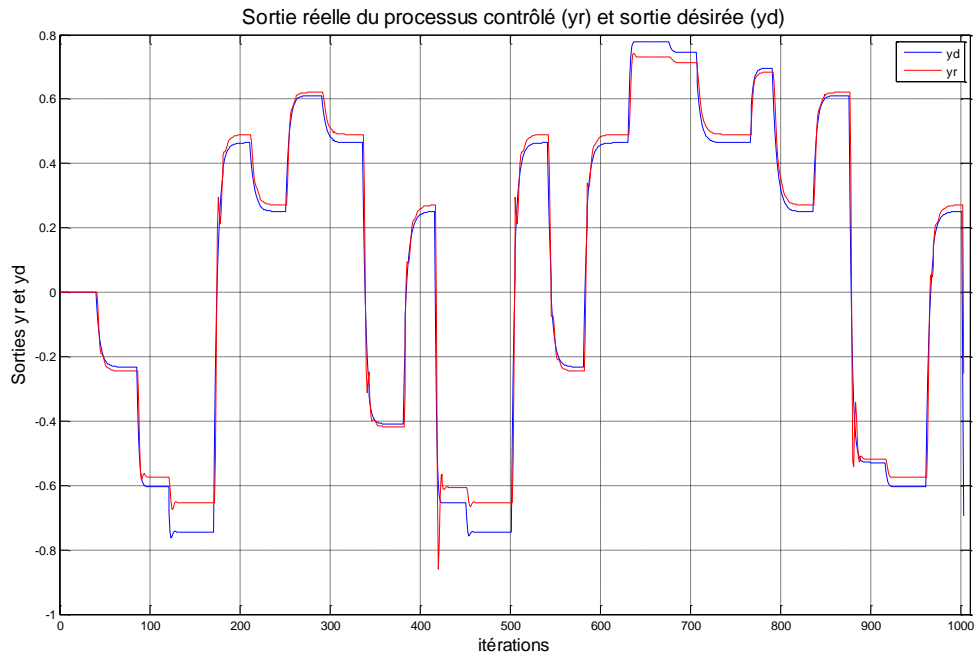


Figure 3.26: La référence ou sortie désirée (y_d) et la sortie réelle du processus contrôlé (y_r)

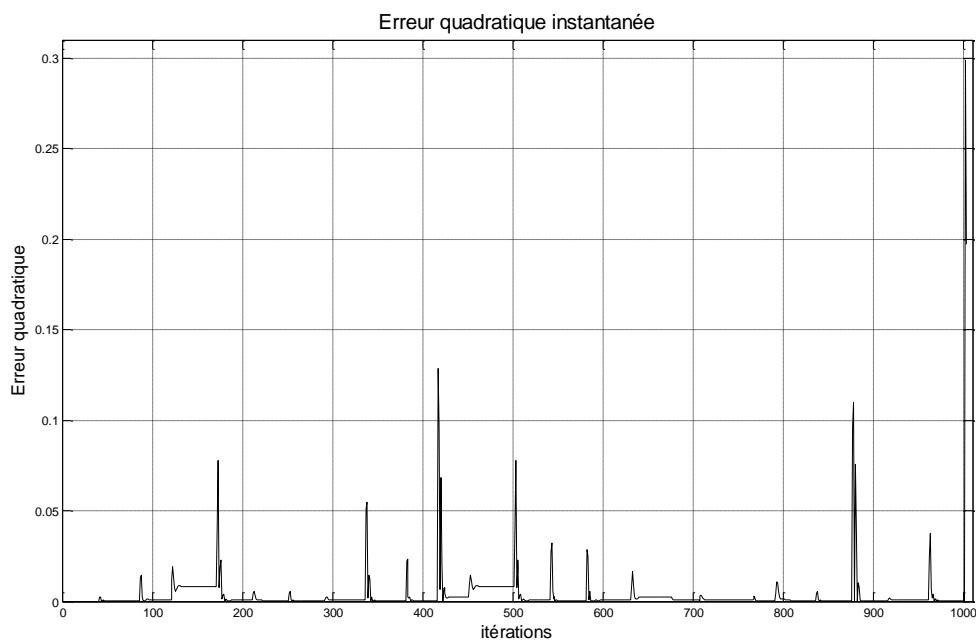


Figure 3.27 : L'erreur quadratique cumulée (E_c)

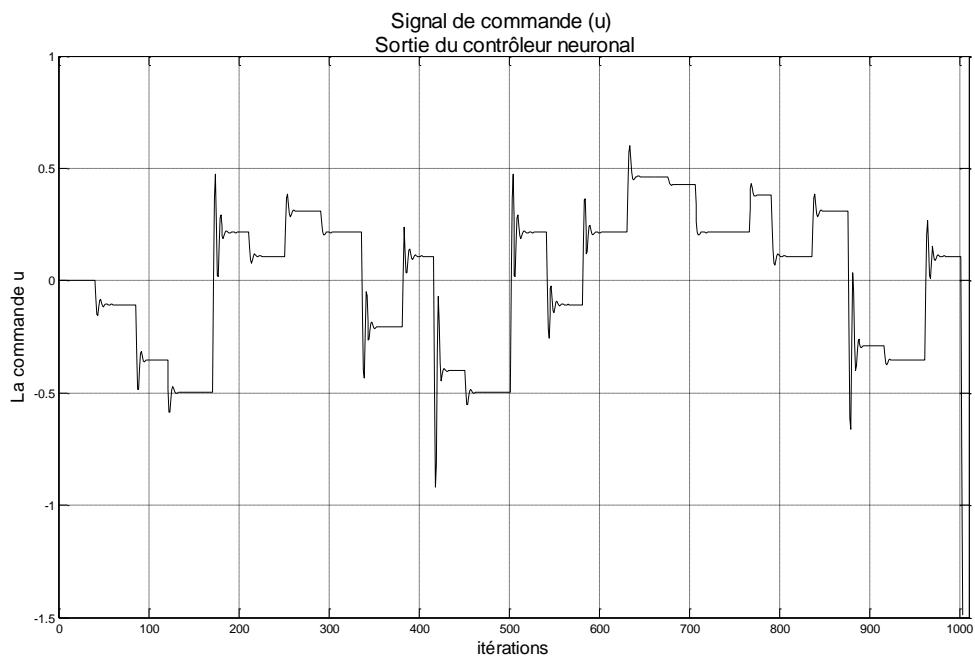


Figure 3.28 : le signal de commande ou sortie du contrôleur neuronal (u_c)

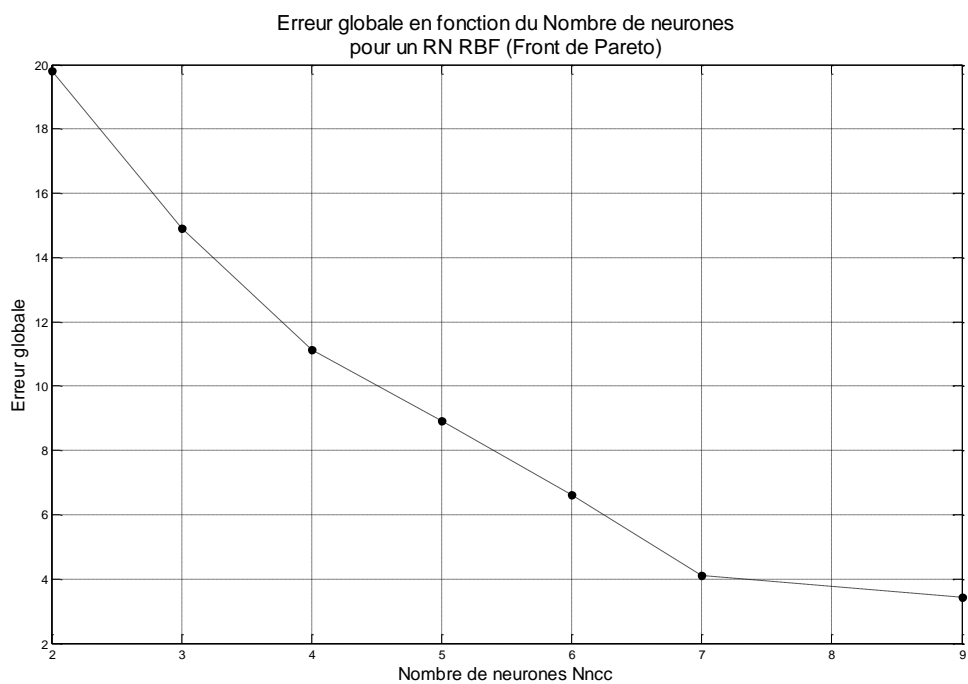


Figure 3.29 : le front de Pareto de la première approche « l'erreur globale cumulée en fonctions du nombre de neurones »

4.2.3.4. Résultats de validation

Après avoir extrait le contrôleur neuronal *RBF* choisi parmi ceux du front de *Pareto*, nous avons modifié la référence afin de vérifier l'adaptation du contrôleur. Les figures ci-dessous représentent les résultats de validation obtenus. Elles représentent respectivement, la référence de validation ou sortie désirée (R_{efval}) et la sortie réelle du processus contrôlé (y_r) figure 3.30, L'erreur quadratique instantanée (E_q) figure 3.31.

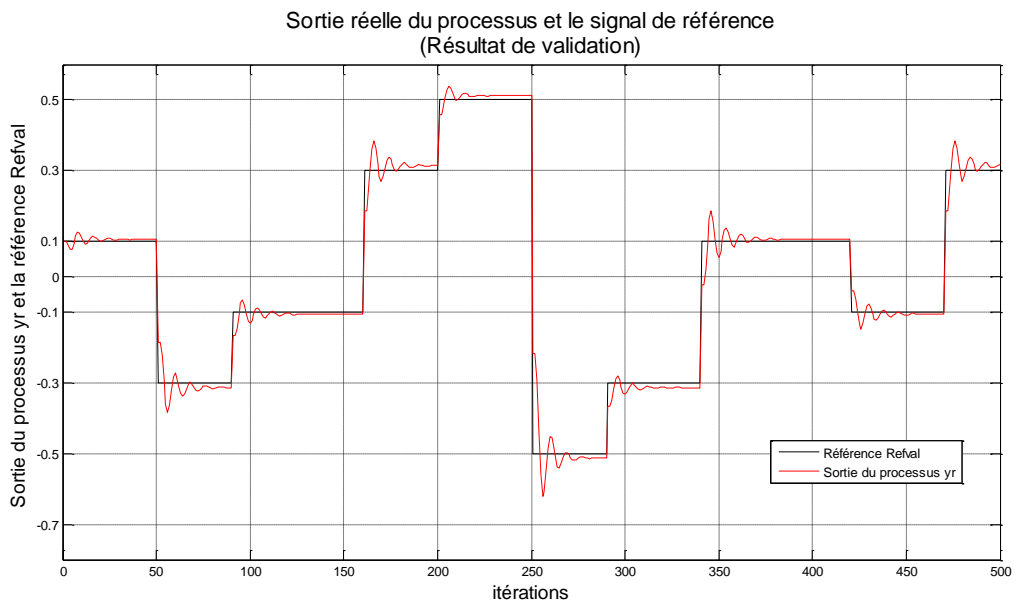


Figure 3.30 : La référence de validation ou sortie désirée (R_{efval}) et la sortie réelle du processus contrôlé (y_r)

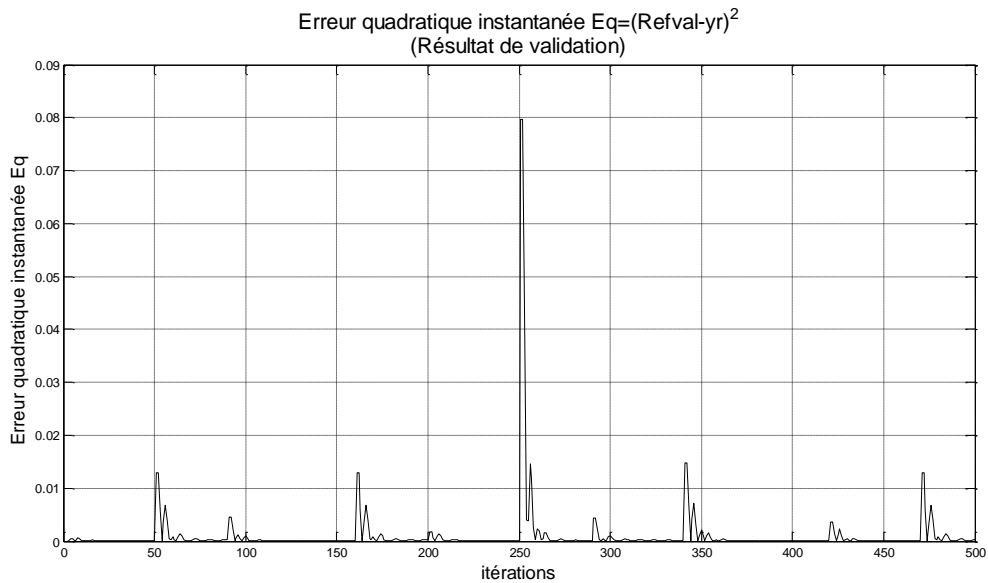


Figure 3.31 : L'erreur quadratique instantanée (E_q) du résultat de validation

D'après ces figures, nous remarquons que ce contrôleur arrive à commander le système mais les performances sont dégradées, en particulier une erreur statique apparaît avec des oscillations.

4.2.3.5. Test de la robustesse

Les tests de robustesse n'ont pas donné de bons résultats, nous n'avons pas jugé utile de les représenter.

5. Comparaison des résultats (RN RBF et RN MLP)

Les tableaux 3.17 et 3.18 ci-dessous représentent un récapitulatif des individus choisis des fronts de Pareto pour les deux types de RN (MLP et RBF) et pour les deux approches d'optimisation à 2 et à 3 objectifs en utilisant pour la même application les mêmes paramètres de l'algorithme génétique multi-objectif, *NSGA2* avec le même nombre d'individus de la population, le même nombre de générations, même probabilités de croisement et de mutation, même type de codage.

Application		Individus privilégiés des premiers fronts de Pareto			
		Approche 1 : optimisation de deux fonctions objectifs			
		Réseau RBF		Réseau MLP	
		Nncc	Erreur	Nncc	Erreur
Modélisation	Système de Box et Jenkins	9	0.6197	4	0.086363
	Système chaotique de Mackey-Glass	7	6.2869	9	3.8598
Contrôle	Système à comportement variable simulé par un modèle d'état	9	3.4268	6	0.0532

Tableau 3.17 : récapitulatif des individus choisis du 1^{er} front de Pareto en optimisant deux fonctions objectifs: le nombre de neurones de la couche cachée Nncc et l'erreur quadratique globale Ec

Application		Individus privilégiés des premiers fronts de Pareto					
		Approche 2 : optimisation de trois fonctions objectifs					
		Réseau RBF			Réseau MLP		
		Nreg	Nncc	Erreur	Nreg	Nncc	Erreur
Modélisation	Système de Box et Jenkins	6	8	0.7236	3	2	0.090254
	Système chaotique de Mackey-Glass	5	9	6.3324	3	5	3.5026
Contrôle	Système à comportement variable simulé par un modèle d'état	4	9	3.7525	3	4	0.0294

Tableau 3.18 : Récapitulatif des individus choisis du 1^{er} front de Pareto en optimisant trois fonctions objectif: le nombre des regresses à l'entrée du réseau Nreg, le nombre de neurones de la couche cachée Nncc et l'erreur quadratique globale Ec.

D'après ces résultats, nous remarquons que l'algorithme génétique multi-objectif a donnée des structures meilleures pour les contrôleurs et les modèles à réseau de neurones MLP que ceux de type RBF. L'algorithme NSGA II converge plus rapidement vers le front de Pareto pour les réseaux MLP. Ceci s'explique par le

nombre de paramètres du MLP nettement inférieur au RBF. Pour le MLP, l'optimisation à trois objectifs donne toujours de meilleurs résultats que l'optimisation à deux objectifs, ce qui n'est pas le cas pour le RBF où le problème à trois objectifs devient nettement plus difficile à résoudre.

Dans le cas de l'optimisation des MLP avec trois objectifs l'introduction des entrées du réseau comme objectif a été déterminante dans la qualité des solutions obtenues.

6. Conclusion

Dans ce chapitre nous avons présenté la technique de modélisation et de contrôle par les algorithmes génétiques multi-objectifs pour la modélisation et la commande des processus. Pour chaque type de RN nous avons opté pour deux approches, la première consiste à trouver le front de Pareto pour un problème à deux objectifs, le nombre de neurones de la couche cachée du RN et l'erreur quadratique cumulée, la deuxième consiste à trouver le front de Pareto pour un problème à trois objectifs, les deux précédents plus les des entrées du réseau.

Ces deux approches sont appliquées sur deux applications de modélisation et une application de contrôle. Les résultats obtenus montrent que le RN MLP a convergé plus rapidement que celui du RBF et a donné des résultats meilleurs. Aussi ces résultats montrent que l'introduction de la troisième fonction (optimisation des entrées) influe nettement sur la qualité des solutions.



CHAPITRE 4

Commande du système de freinage

Anti Blocage des Roues -ABS

Chapitre 4

Commande du système de freinage Anti Blocage des Roues -ABS

1. Introduction

Dans ce chapitre, nous présentons une application de l'optimisation des réseaux de neurones MLP par les algorithmes génétiques multi-objectifs NSGA2 pour la commande le système de freinage Anti Blocage des Roues (*Anti-lock Braking System*) connu par l'abréviation *ABS*, qui est un système fortement non linéaire. L'optimisation à trois objectifs ayant donné des résultats encourageants au chapitre précédent, nous l'appliquerons dans ce chapitre pour la conception du contrôleur MLP.

2. Présentation du système ABS

Le système de freinage anti-blocage des roues *ABS*, a été mis en œuvre à la fin des années soixante-dix. C'est un système d'assistance au freinage utilisé sur les véhicules roulants, empêchant les roues de se bloquer pendant les périodes de freinage intense. Il est utilisé dans les avions lors de l'atterrissage et dans les véhicules automobiles ou motocyclettes, où il fait de plus en plus partie de l'équipement standard.

L'ABS a été conçu à l'origine pour aider le conducteur à conserver la maîtrise de son véhicule, dans le cas d'un freinage dans des conditions d'adhérence difficile (pluie, neige, verglas, gravier...). Il permet de garder la directivité du véhicule afin d'effectuer une manœuvre d'évitement éventuelle, tout en diminuant la distance de freinage qui peut augmenter considérablement lorsque les roues se bloquent et que les pneumatiques glissent sur la chaussée. En effet, sans le system ABS, quand

le conducteur actionne trop fortement la pédale de frein de son véhicule à la suite d'un danger, les roues se bloquent ce qui provoque en plus de l'augmentation de la distance de freinage, la perte de la stabilité directionnelle et le dérapage du véhicule.

Le système ABS empêchera donc le blocage des roues au moment du freinage ce qui permet de conserver la directivité du véhicule. Lors d'un freinage d'urgence, l'ABS régule la pression dans le circuit de freinage en adaptant le niveau de pression hydraulique maximum pour chaque roue par le biais d'électrovannes: si une roue ralentit anormalement, il relâche instantanément la pression du circuit de freinage au niveau de cette roue et ce, tant que celle-ci ne ré-accélère pas suffisamment. La conception du système de contrôle est compliquée par le fait que le modèle du système est fortement non linéaire et il est donc difficile, voire impossible d'utiliser des méthodes de conception linéaire, pour cela que plusieurs chercheurs travaillent pour l'étude et l'amélioration de ces systèmes [52, 53].

3. Le système ABS de laboratoire utilisé

3.1 Description

Le système ABS utilisé ici est un système de laboratoire composé de deux roues et il est entraîné par un moteur à courant continu. Il comporte trois encodeurs identiques mesurant les angles de rotation des deux roues et l'angle de déviation du levier de l'équilibre de la roue de voiture. Les encodeurs mesurent les mouvements avec une haute résolution égale à 4096 impulsions par rotation [54].

La figure 4.1 représente la structure du modèle ABS utilisé.

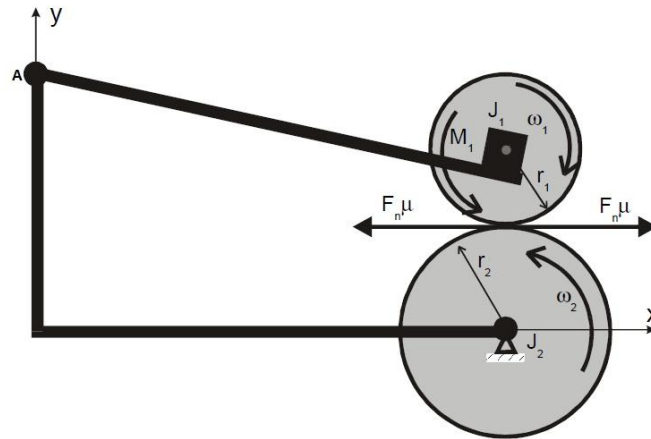


Figure 4.1 : Structure du modèle ABS de laboratoire utilisé

La roue inférieure simule le mouvement relatif de la route et la roue supérieure représente la roue du véhicule et elle est équipée d'un pneu, en contact permanent avec la roue inférieure. La surface de la roue inférieure est lisse et peut être recouvert de matériaux donnés pour simuler les différentes conditions et états de la route. L'objectif principal du système de contrôle est d'empêcher le blocage des roues lors du freinage. Ceci est réalisé grâce au contrôle du glissement des roues afin d'optimiser le coefficient de frottement entre le pneu de la roue et la route pour toute surface donnée de la route (tout état de la route: verglas, neige, boue, goudron mouillé, goudron sec, béton mouillé, béton sec etc.), la voiture devient ainsi contrôlable. Si la roue est immobile et bloquée et la vitesse de la voiture n'est pas égal à zéro cela signifie que la voiture est restée en mouvement (il y a donc un glissement). Mais si la roue est bloquée et la vitesse de la voiture est égale à zéro cela signifie que la voiture est totalement arrêtée. Dans le premier cas le système ABS tend à débloquer la roue afin d'éviter le glissement de la voiture. La roue commence à tourner et après une courte durée elle est de nouveau arrêtée. Ce processus est répété jusqu'à ce que la voiture soit arrêtée sans glissement.

3.2 Modèle de simulation

Afin de simuler le processus de freinage, nous utilisons un modèle du système de laboratoire ABS donnée dans [54].

Soit x_1 et x_2 respectivement la vitesse angulaire de la roue supérieure et la roue inférieure (qui représente la route), et λ le coefficient du glissement défini par la différence relative entre la vitesse des deux roues. Les équations du mouvement peuvent être mises sous la forme:

$$\dot{x}_1 = S(\lambda)(c_{11}x_1 + c_{12}) + c_{13}x_1 + c_{14} + (c_{15}S(\lambda) + c_{16})s_1M$$

$$\dot{x}_2 = S(\lambda)(c_{21}x_1 + c_{22}) + c_{23}x_2 + c_{24} + (c_{25}S(x_1, x_2))s_1M$$

Où:

$$S(\lambda) = \frac{s\mu(\lambda)}{L(\sin(\phi) - s\mu(\lambda)\cos(\phi))} \quad ; \quad \mu(\lambda) = \frac{w_4\lambda^p}{a + \lambda^p} + w_3\lambda^3 + w_2\lambda^2 + w_1\lambda$$

$$c_{11} = \frac{r_1 d_1}{J_1}, \quad c_{12} = -\frac{(s_1 M_{10} + M_g) r_1}{J_1}, \quad c_{13} = -\frac{d_1}{J_1}, \quad c_{14} = -\frac{s_1 M_{10}}{J_1}, \quad c_{15} = \frac{r_1}{J_1}, \quad c_{16} = -\frac{1}{J_1}$$

$$c_{21} = \frac{r_2 d_1}{J_2}, \quad c_{22} = -\frac{(s_1 M_{10} + M_g) r_2}{J_2}, \quad c_{23} = -\frac{d_2}{J_2}, \quad c_{24} = -\frac{s_2 M_{20}}{J_2}, \quad c_{25} = -\frac{r_2}{J_2}$$

$$s = \text{sign}(r_2 x_2 - r_1 x_1), \quad s_1 = \text{sign}(x_1), \quad s_2 = \text{sign}(x_2)$$

$$\dot{M} = c_{31}(b(u) - M)$$

$$b(u) = \begin{cases} b_1 u + b_2 & u > u_0 \\ 0 & u < u_0 \end{cases}$$

Le coefficient de glissement λ est calculé comme suit:

$$\lambda = \begin{cases} \frac{r_2 x_2 - r_1 x_1}{r_2 x_2}, & r_2 x_2 \geq r_1 x_1, x_1 \geq 0, x_2 \geq 0. \\ \frac{r_1 x_1 - r_2 x_2}{r_1 x_1}, & r_2 x_2 < r_1 x_1, x_1 \geq 0, x_2 \geq 0. \\ \frac{r_2 x_2 - r_1 x_1}{r_2 x_2}, & r_2 x_2 < r_1 x_1, x_1 < 0, x_2 < 0. \\ \frac{r_1 x_1 - r_2 x_2}{r_1 x_1}, & r_2 x_2 \geq r_1 x_1, x_1 < 0, x_2 < 0. \\ 1, & x_1 < 0, x_2 \geq 0. \\ 1, & x_1 \geq 0, x_2 < 0. \end{cases}$$

Où :

- u est le signal de commande.
- M est le couple de freinage,
- b_1, b_2 et c_{31} sont des constants.

- r_1 et r_2 sont respectivement les rayons de la roue supérieure et la roue inférieure.
- J_1 et J_2 sont respectivement les moments d'inertie de la roue supérieure et la roue inférieure.
- \varnothing est angle entre l'axe verticale du point de contact des deux roues et la droite entre ce point et le point A.
- L est la distance entre le point de contact des roues et l'axe de rotation du levier.
- M_{10} et M_{20} sont respectivement les frottements statiques de la roue supérieure et la roue inférieure.
- M_g est le couple absorbeur de choc gravitationnel et agissant sur le levier.
- $\mu(\lambda)$ est le coefficient de frottement entre les deux roues.

Valeurs numériques

$$c_{31}=20.37, w_1=-0.04240011450454, w_2=0.00000000029375,$$

$$w_3=0.03508217905067, w_4=0.40662691102315, a=0.00025724985785,$$

$$p=2.09945271667129.$$

$$r_1=0.0995 \text{ m}; r_2=0.099 \text{ m}; b_1 =15.24, b_2 =-6.21, J_1 =7.53 \cdot 10^{-3} \text{ kgm}^2; J_2= 25.60 \cdot 10^{-3} \text{ kgm}^2; \varnothing=65.61^\circ; L=0.37 \text{ m}.$$

Ces équations sont fortement non linéaires et compliqués par la fonction signe qui intervienne aussi dans les fonctions s , s_1 et s_2 . Il est difficile d'obtenir un modèle linéaire pour la conception d'un contrôleur linéaire ou même d'utiliser des techniques de conception non linéaires. Il serait intéressant d'essayer quelques approches alternatives qui ne se basent pas sur la structure du modèle tel que le contrôleur neuronal.

4. Commande du système de freinage ABS

L'objectif est de commander le dérapage ou bien le patinage des roues en essayant de ramener le coefficient de frottement entre le pneu de la roue et la

route à la valeur désirée permettant le contrôle du véhicule. Idéalement, cela doit être réalisé pour différents états de la route, ce qui implique que le contrôleur doit être particulièrement robuste.

4.1. Conception de contrôleur neuronal

La conception de ce contrôleur neuronal optimisé par l'algorithme génétique multi objectifs est réalisée en utilisant l'approche à trois objectifs. Les paramètres de l'algorithme génétique multi-objectifs utilisés sont :

- Type de l'algorithme utilisé : NSGA2
- Intervalles de recherche : $N_{cc} \in [2, 20]$; $w \in [-1, 1]$
- La taille de la population (le nombre d'individu): $N=100$
- Nombre de générations : $gen=200$
- Probabilité de croisement : $P_c=0.9$
- Probabilité de mutation : $P_m=0.08$
- Type de codage : codage réel

L'approche d'optimisation des réseaux MLP à trois objectifs est appliquée au modèle de simulation du système de freinage antiblocage des roues ABS de laboratoire. Le scénario de simulation consiste à accélérer les deux roues jusqu'à ce qu'elles atteignent 240 rad/s pour la roue supérieure et 220 rad/s pour la roue inférieure, ce qui correspond à une vitesse d'environ 86 km/h. Le freinage est alors commencé afin d'éviter le blocage des roues. La valeur de glissement de référence est $\lambda_d=0.2$ qui représente le taux de glissement désiré. Cette valeur de taux de glissement assure un meilleur frottement entre la roue et la route et elle donne le meilleur compromis entre la stabilité, la maniabilité directionnelle et la force de freinage, garantissant ainsi une meilleure contrôlabilité du véhicule.

Après l'exécution de l'algorithme NSGA2, un contrôleur neuronal est extrait à partir des meilleurs individus de la dernière génération, (front de Pareto). Le contrôleur est alors appliqué pour contrôler le modèle Simulink fourni par le vendeur du système ABS.

4.2. Résultats de simulation

Après optimisation, les individus du front de Pareto (individus non dominés) de la dernière génération sont extraits. Nous choisissons parmi ceux-ci, un individu présentant une erreur cumulée de 0.4328 et nombre de neurones de la couche cachée N_{ncc} égale à 6 et 2 entrées, l'erreur $e(t-1)$ et la commande $u(t-1)$.

Dans ce cas, nous avons privilégié le résultat dont l'erreur globale est la plus petite puisqu'il y a une différence importante entre cette valeur et celle des autres résultats obtenu dans le front de Pareto.

Le temps d'échantillonnage utilisé est de 0.01 secondes, le temps de calcul du signal de commande par le réseau MLP dans un PC compatible IBM avec une horloge à 1,4 GHz est inférieur à ce temps d'échantillonnage.

Les figures ci-dessous représentent respectivement, le taux de glissement $\lambda r(t)$ figure 4.3, l'erreur instantanée figure 4.4, les vitesses angulaires x_1 et x_2 des deux roues figure 4.5, le signal de commande ou la sortie du contrôleur neuronal, figure 4.6, le couple de freinage M figure 4.7.

Nous remarquons d'après ces résultats que l'arrêt des roues atteint sa référence en douceur et très rapidement (x_1 et x_2 atteignent la valeur zéro, figure 4.5). Le signal de commande est aussi lisse (figure 4.6). Le temps de freinage, (lorsque les deux roues sont arrêtés), est d'environ 0,3 secondes. Ces résultats sont très encourageants.

D'après ces résultats nous remarquons que le contrôleur neuronal optimisé par les algorithmes génétiques multi-objectif a montré son efficacité pour la stabilisation de ce système instable et non linéaire.

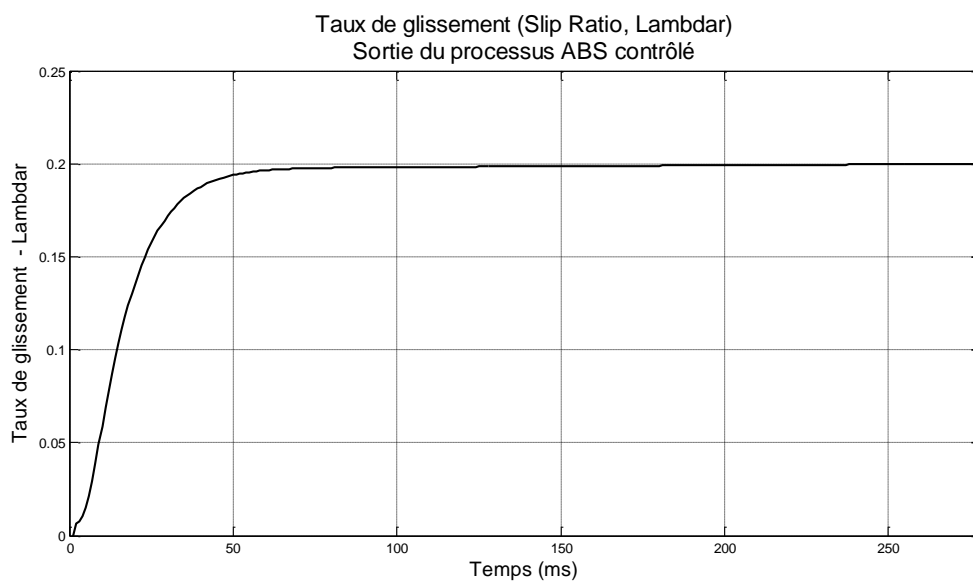


Figure 4.2: Taux de glissement (la sortie réelle du processus contrôlé λ_r)

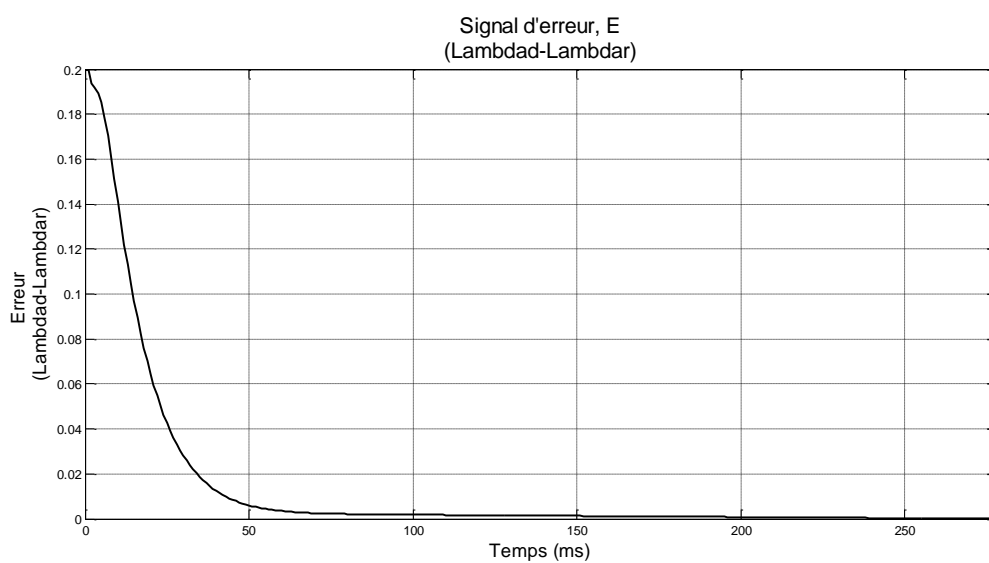


Figure 4.3 : Le signal d'erreur $e(t)$

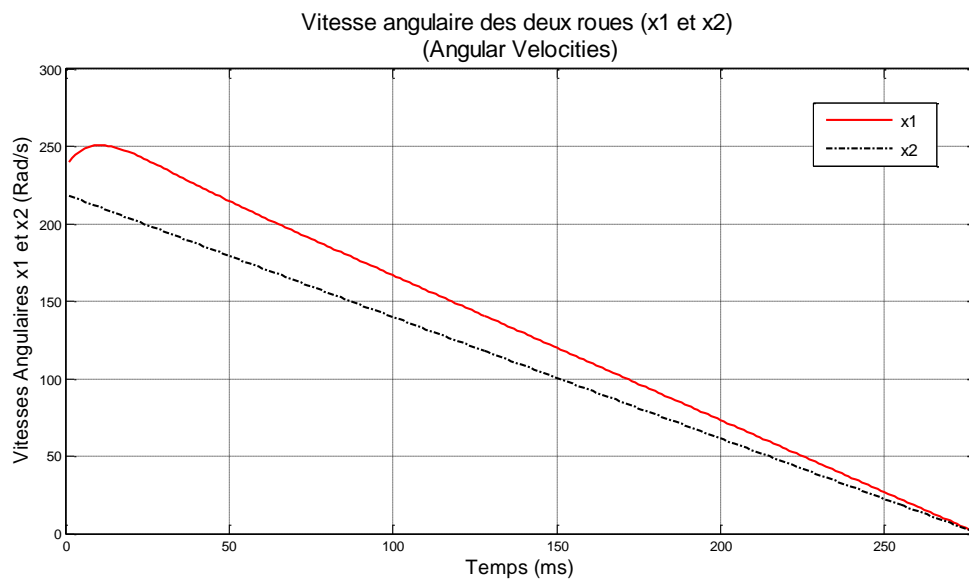


Figure 4.4: La vitesse angulaire des deux roues (x1 et x2)

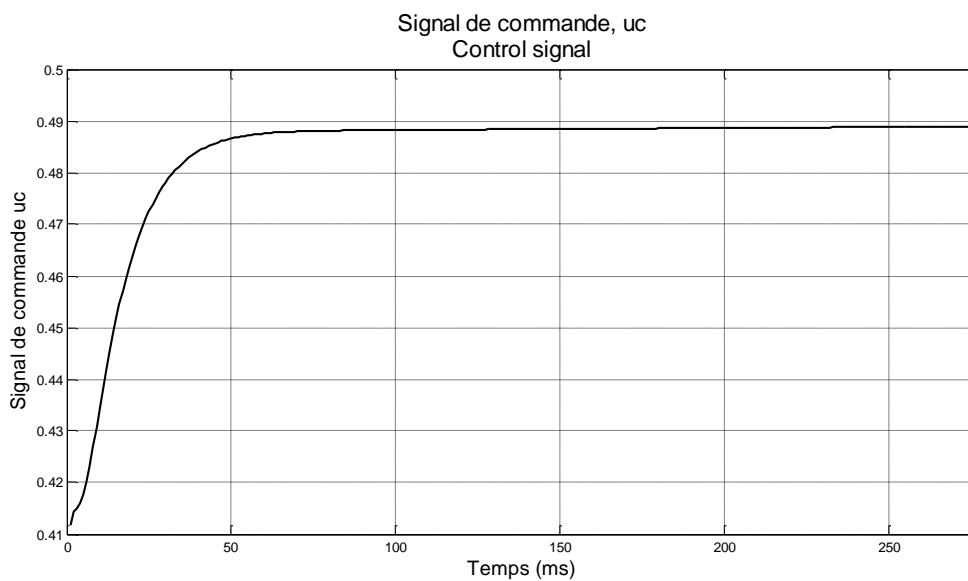


Figure 4.5 : le signal de commande

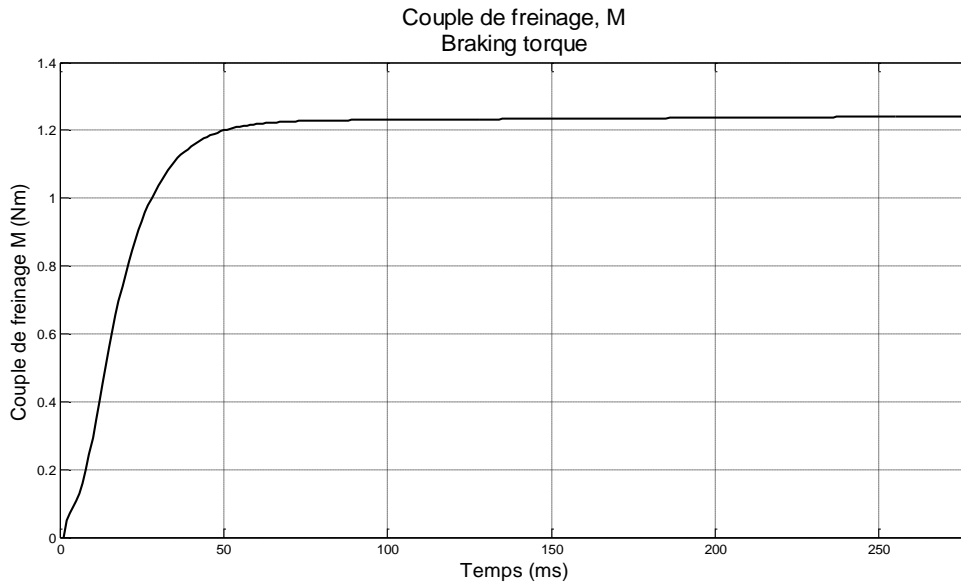


Figure 4.6: Le couple de freinage M

4.3. Test de poursuite

En pratique, le taux de glissement désiré peut varier suivant les applications, cependant sa valeur de référence reste proche de 0.2. Pour tester la capacité du contrôleur à suivre une référence autre que celle de l'entraînement, nous avons varié la référence dans une large gamme. Les résultats montrent que le contrôleur MLP perd ses performances au-delà d'une variation de référence de $\pm 10\%$. Les figures 4.8 et 4.9 représentent respectivement le taux de glissement $\lambda_r(t)$ pour une variation de la référence de $\pm 10\%$ soit une référence de 0.22 et 0.18.

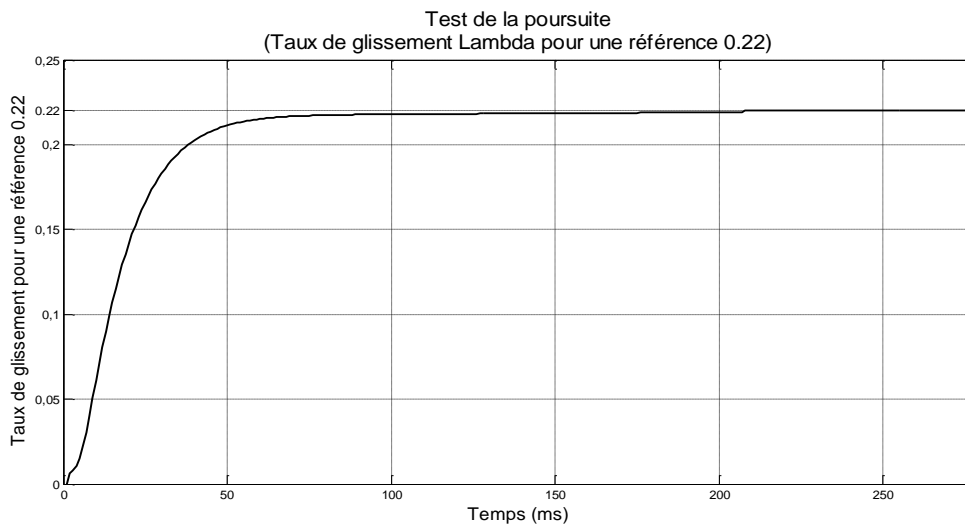


Figure 4.7 : le taux de glissement pour une référence de 0.22

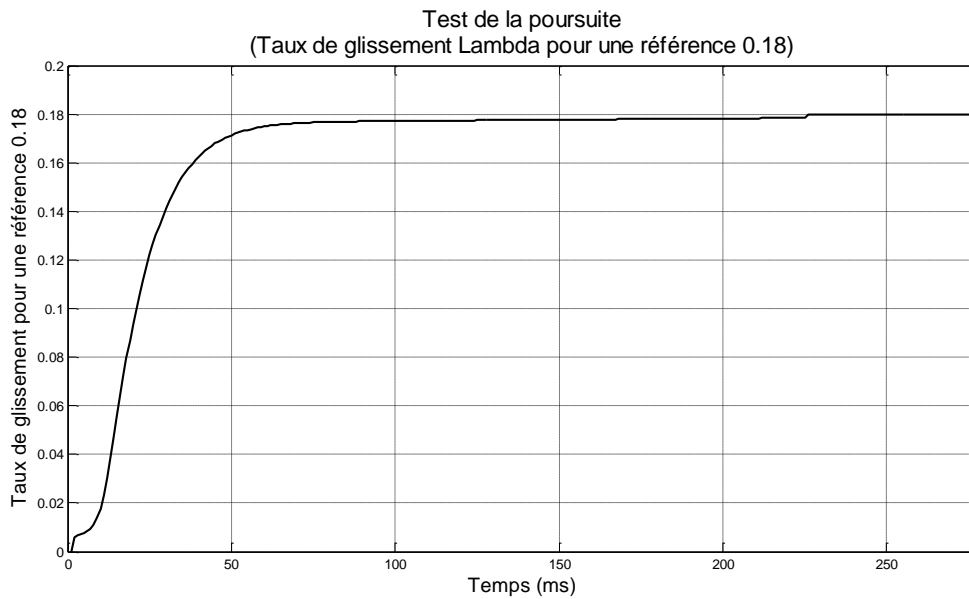


Figure 4.8 : le taux de glissement pour une référence de 0.18

4.4. Test de robustesse

La robustesse est facteur important dans les systèmes ABS. En effet, le système doit être capable d'agir pour une large variété de conditions routières. Pour simuler différentes conditions nous introduit des variations sur le coefficient de frottement entre la roue et la route μ donné par :

$$\mu(\lambda) = w_4 \frac{\lambda^p}{a + \lambda^p} + w_3 \lambda^3 + w_2 \lambda^2 + w_1 \lambda$$

$w_1 \dots w_4$ et p sont des constantes définissant l'état de la route.

Après les tests, nous avons remarqué que les perturbations sur les paramètres w_1 , w_2 , w_3 , n'influent pas sur le comportement du système, par contre le paramètre w_4 a un effet considérable sur le comportement du système.

La figure 4.10 représente les taux de glissement (sortie du processus contrôlé) pour des perturbations de $\pm 10\%$ (courbes en couleurs bleu et noire) et sans perturbations (courbe en couleur rouge).

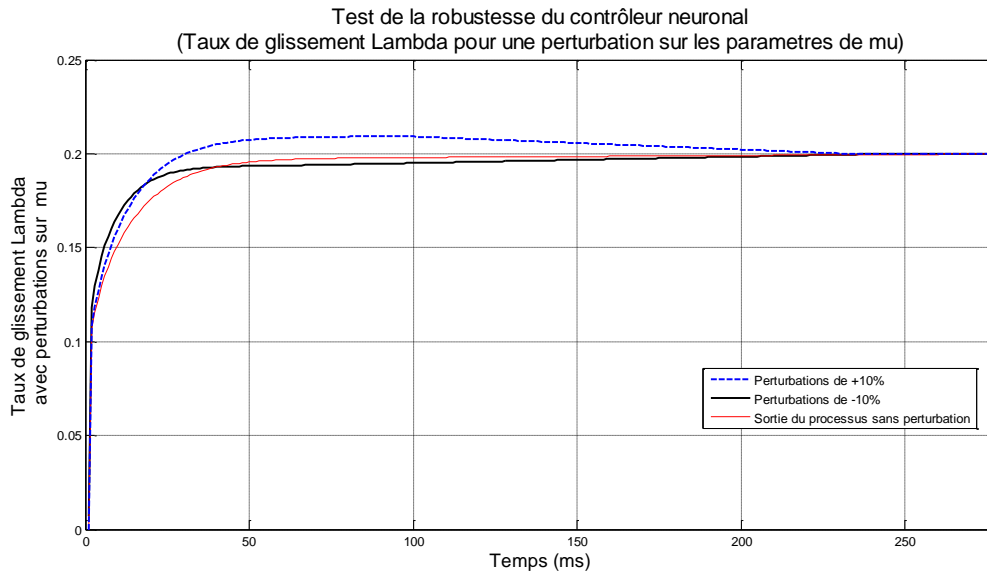


Figure 4.9 : Taux de glissement pour une perturbation sur les paramètres du coefficient de frottement

Dans cette figure nous pouvons constater que pour la variation maximale +10%, une erreur statique de 0.0094 est introduite. Pour les valeurs inférieures, le temps de réponse augmente légèrement. Au-delà d'une variation de +10%, la sortie commence à se dégrader.

5. Conclusion

Dans ce chapitre, nous avons présenté l'application de la méthode conception des contrôleurs neuronaux présentée au chapitre précédent à la commande du système de freinage anti blocage des roues, ABS. Après présentation du fonctionnement du système à contrôler et description de modèle utilisé nous avons conçu notre contrôleur neuronal en utilisant les algorithmes génétiques multi-objectifs et ceci en optimisant trois fonctions objectives qui sont le nombre de neurones dans la couche cachée du réseau, le nombre des regresseurs présentés à entrée du réseau et l'erreur. Les résultats obtenus sont très satisfaisants du fait que le contrôleur a permis d'atteindre la référence désirée. Les tests de la poursuite et de la robustesse ont montrés que le comportement du processus n'a pas été dégradé dans des limites de $\pm 10\%$.



CONCLUSION

Générale

Conclusion Générale

Le grand avantage des réseaux de neurones réside dans leur capacité de traitement parallèle, d'apprentissage et d'approximation qu'ils possèdent et qui permettent de concevoir des modèles et des contrôleurs neuronaux capables de résoudre les problèmes rencontrés dans les méthodes classiques, sans nécessité des règles complexes.

Les réseaux de neurones présentent une très grande diversité, en effet un type de réseau neuronal est défini par sa topologie, sa structure interne et son algorithme d'apprentissage.

L'obtention d'une architecture appropriée était toujours un problème pour la construction des réseaux. Cette difficulté réside dans la détermination des paramètres constituant la structure du réseau : le nombre de couches cachées dans un réseau multicouche, le nombre optimal de neurones dans chaque couche et les valeurs initiales des poids de connexions du réseau pendant la phase d'apprentissage.

Nous avons proposé dans ce travail de résoudre ces problèmes par une optimisation multi-objectifs. Il s'agit de déterminer la meilleure structure du réseau de neurone utilisé et assurer son apprentissage. Pour cela nous avons utilisé les Algorithmes Génétique Multi-Objectifs de type NSGA2 (Non-dominated Sorting Genetic Algorithm2) afin d'optimiser les réseaux de neurones multi couches, MLP, et à fonction radiale RBF.

Pour chaque réseau, nous avons adopté deux approches, dans la première nous avons optimisé deux fonctions objectives qui sont le nombre de neurones de la couche cachée et l'erreur d'apprentissage, dans la seconde, nous avons optimisé trois fonctions, en plus des deux fonctions objectifs précédentes une troisième fonction est ajoutée, le nombre et le type des entrées du réseau.

Cette technique d'optimisation est utilisée dans la modélisation et la commande. Deux exemples de modélisation et un exemple de commande sont traités. A partir de ces exemples, nous avons observé que l'introduction des entrées des réseaux dans le processus d'optimisation améliore nettement la qualité du réseau obtenu que ce soit pour la modélisation ou pour la commande. Cette approche a alors été appliquée pour commander en simulation un système de freinage antiblocage des roues ABS de laboratoire.

D'une façon générale, les résultats obtenus montrent que l'algorithme génétique multi-objectif a donné des structures meilleures pour les contrôleurs et les modèles à réseau de neurones MLP que ceux de type RBF. Ceci est dû au fait que pour les RBF, le nombre de paramètres à optimiser est nettement supérieur à celui des MLP, les risques de tomber sur un minimum local de mauvaise qualité augmente.

Les résultats obtenus sont très satisfaisantes et confirment que l'utilisation de cette technique d'optimisation des réseaux de neurones pour la modélisation et la commande des systèmes est un outil approprié et efficace et peut apporter de très bons résultats notamment en modélisant et contrôlant des systèmes non linéaire complexes, là où il est difficile d'utiliser les méthodes classiques.



BIBLIOGRAPHIE

Bibliographie

- [1] G. Dreyfus, J.-M. Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. Thiria, L. Hérault, "Réseaux de neurones, méthodologie et applications", Eyrolles, 2ème édition, 2004.
- [2] Wolfgang Maass and, Eduardo D. Sontag, "Neural Systems as Nonlinear Filters", Neural Computation, 2000, Vol. 12, No. 8 , Pages 1743-1772, Posted Online March 2006.
- [3] P. J. Werbos, "BackPropagation through time : What it does and how to do it?", Proc. IEEE, vol. 78, pp. 1550-1560, 1990.
- [4] J. M. Zurada, "Introduction to artificial neural systems", West Publishing, 1992.
- [5] N. Baba, "utilisation of stochastic automata and genetic algorithms for neural network learning", In parallel Problem Solving from Nature 2. Elsevier, 1992.
- [6] R.S. Sexton, R.E. Dorsey, J.D. Johnson, "Toward global optimization of neural networks : Acomparaison of the genetic algorithm and backpropagation", Decision Support Systems, Vol. 22, N°. 2, pp. 171-185, 1998.
- [7] Jin, Chengjun, Chang, Guiran; Cheng, Wei; Jiang, Huiyan, "Improved Particle Swarm Optimization for Fuzzy Neural Network Training", Fifth International Conference on Genetic and Evolutionary Computing (ICGEC), pp. 299–302, Aug 2011.
- [8] Min Zhu, Wei Dong Liu, Wen Song Hu, "Application of Fuzzy Neural Network in Relay Protection", Advanced Materials Research , AMST, Vol. 181-182, pp. 434-438, January 2011.
- [9] M. Koeppen, M. Teunis, B. Nickolay, "Neural network that uses evolutionary learning", Proceeding of the 1997 IEEE International Conference on Evolutionary Computation, ICEC'97, Piscataway, New Jersey, USA, pp. 635-639, 1997.

- [10] S. Majumdar, K. Mitra, "Modeling of a reaction network and its optimization by genetic algorithm", *Chemical Engineering Journal*, Vol. 100, pp. 109-118, 2004.
- [11] P.G. Korning, "Training neural network by means of genetic algorithms working on very long chromosomes", *International Journal of Neural Systems*, Vol.6, N°3, pp. 299-316, 1995.
- [12] Yann Collette, Patrick Siarry, "Optimisation Multi Objectif", Edition Eyrolles, 2002.
- [13] D.E.Goldberg, "Algorithmes Génétiques, exploration, optimisation et apprentissage automatique", Edition Addison-Wesley, Edition 1997.
- [14] James David Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms", In *genetic Algorithm and their Applications: Proceedings of the First International Conference on Genetic Algorithm*, Hillsdale, NJ, USA, p.93-100, 1985.
- [15] C. A. Coello Coello, Gary B. Lamont, "Applications of Multi-Objective Evolutionary Algorithms", Edition World Scientific, Singapore, 2004.
- [16] Rajeev Kumar, Nilanjan Banerjee, "Multiobjective network topology design", Elsevier B.V. *Applied Soft Computing*, Vol. 11, Issue 8, pp. 5120-5128, Available online 17 June 2011.
- [17] P. Hajela, C.-Y. Lin, "Genetic search strategies in multicriterion optimal design", *Structural Optimization* 4, pp.99-107, 1992.
- [18] H. Ishibuchi, T. Murata, "Multi-Objective Genetic Local Search Algorithm", *Proceedings of the International Conference on Evolutionary Computation*, Nagoya, Japan, pp. 119-124, 1996.
- [19] C. M. Fonseca, P. J. Fleming, "Genetic algorithms for multi-objective optimization: formulation, discussion and generalization", *Proceedings of the 5th International Conference on Genetic Algorithms*, San Mateo, California, pp. 416-423, 1993.

- [20] J. Horn, N. Nafpliotis, D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization", Proceedings of the 1st IEEE Conference on Evolutionary Computation, IEEE World Congress on Evolutionary Computation, vol. 1, pp. 82-87, Piscataway, NJ, IEEE Press, 1994.
- [21] N. Srinivas, K. Deb, "Multiobjective optimization using non-dominated sorting in genetic algorithms", Evolutionary Computation, Vol. 2, N°. 3, pp. 221-248, 1994.
- [22] E. Zitzler, L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach", IEEE Transactions on Evolutionary Computation, Vol.3, N°. 4, pp. 257-271, 1999.
- [23] E. Zitzler, M. Laumanns, L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm", Technical report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, 2001.
- [24] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan , "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II". Parallel Problem Solving from Nature (PPSN VI), Berlin, pp. 849-858, 2000.
- [25] K. Deb, T. Goel, "Controlled elitist non-dominated sorting genetic algorithms for better convergence", Proceedings of the first International Conference on Evolutionary Multi-criterion Optimization, Berlin, pp. 67-81, 2001.
- [26] Renata Furtuna, Silvia Curteanu, Florin Leon, "Multi-objective optimization of a stacked neural network using an evolutionary hyper-heuristic", Elsevier B.V. Applied Soft Computing, doi:10.1016, Available online 24 September 2011.
- [27] Jean Dipama, "Optimisation multi-objectif des systèmes énergétiques", thèse PHD, université de Montréal, avril 2010.
- [28] Yuhui Wang, Xiaohui Lei, Yunzhong Jiang, Hao Wang , "Performance comparison of three multi-objective optimization algorithms on calibration of hydrological

- model", Sixth International Conference on Natural Computation (ICNC), Vol. 6, pp. 2798 - 2803 , Aug 2010 .
- [29] Hui Li; Qingfu Zhang; "Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II", IEEE Transactions on Evolutionary Computation, Vol. 13, Issue. 2, pp 284 – 302, 2009.
- [30] S. Kannan, S.vBaskar, J. D. McCalley, P. Murugan, "Application of NSGA-II Algorithm to Generation Expansion Planning", IEEE Transactions on Power Systems, Vol. 24, Issue. 1, pp. 454-461, 2009.
- [31] P. Murugan, S. Kannan, S. Baskar, "Application of NSGA-II Algorithm to Single-Objective Transmission Constrained Generation Expansion Planning", IEEE Transactions on Power Systems, Vol. 24, Issue. 4, pp. 1790-1797, 2009.
- [32] M. M. Efen, C. A. Coello Coello, "A simple multimembered evolution strategy to solve constrained optimization problems", IEEE Trans. Evolutionary Computation, pp. 1-17, 2005.
- [33] J. Dipama, F. Aubé, A. Teyssedou, "A grid-based multi-objective evolutionary algorithm for the optimization of energy systems", Applied Thermal Engineering, Vol. 30, N° 8-9, pp. 807-816, 2009.
- [34] J. Dipama, F. Aubé, A. Teyssedou, "Optimisation multi-objectif d'un système de cogénération à l'aide de l'algorithme évolutif BEST", VIIIème Colloque Interuniversitaire Franco-Québécois sur la Thermique des Systèmes, Montréal, 2007.
- [35] Yang Xiawen, Shi Yu, "A real-coded quantum clone multi-objective evolutionary algorithm", International Conference on Consumer Electronics, Communications and Networks (CECNet), pp. 4683-4687, April 2011.
- [36] Jorge E. Rodríguez, Andrés L. Medaglia, C. A. Coello Coello, "Design of a motorcycle frame using neuroacceleration strategies in MOEAs", Journal of

- Heuristics, Special Issue on Heuristic Research: Advances and Applications , Vol. 15, N°. 2, April 2009.
- [37] Shaine Joseph, Hyung W. Kang, Uday K. Chakraborty, "Optical Design with Epsilon-Dominated Multi-objective Evolutionary Algorithm", ICANNGA 2007, Part I, LNCS 4431, pp. 77 – 84, 2007.
- [38] J. J. Durillo, A. J. Nebro, F. Luna, C. A. Coello Coello, E. Alba, "Convergence Speed in Multi-Objective Metaheuristics: Efficiency Criteria and Empirical Study", International Journal for Numerical Methods in Engineering, Vol. 84, No. 11, pp. 1344--1375, December 2010.
- [39] Qingfu Zhang, Hui Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition", IEEE Transactions on Evolutionary Computation, Vol 11, issue 6, pp. 712–731, 2007.
- [40] J.F.Jodouin, "Les réseaux de neurones. Principes et définitions", édition Hermis, Paris, 1994.
- [41] G. Dreyfus, J.-M. Martinez, M. Samuelides, M. B. Gordon, F. Badran, S. Thiria, L. Hérault, "Réseaux de neurones, Méthodologie et applications", Eyrolles, 2ème édition, 2004.
- [42] Xin Yao, "Evolving Artificial Neural Networks", Proceeding of IEEE, 87(9), pp 1423-1447, 1999.
- [43] A. CORNUÉJOLS, L. MICLET "Apprentissage artificiel, Concepts et algorithmes", édition Eyrolles, 2002.
- [44] H. Yu; T. Xie; S. Paszczyński; B.M. Wilamowski, "Advantages of Radial Basis Function Networks for Dynamic System Design", IEEE Transactions on Industrial Electronics, Vol 58, Issue 12 , pp 5438 – 5450, 2011.
- [45] R. Furtuna, S. Curteanu, F. Leon, "Multi-objective optimization of a stacked neural network using an evolutionary hyper-heuristic", Elsevier B.V. Applied Soft Computing, doi:10.1016, Available online 24 September 2011.

- [46] A. Boloori Arabani, M. Zandieh, S.M.T. Fatemi Ghomi, "Multi-objective genetic-based algorithms for a cross-docking scheduling problem" , Elsevier B.V. Applied Soft Computing, Volume 11, Issue 8,. Pages 4954-4970, Available online 17 June 2011.
- [47] Yuhui Wang Xiaohui Lei Yunzhong Jiang Hao Wang , "Performance comparison of three multi-objective optimization algorithms on calibration of hydrological model" Sixth International Conference on Natural Computation (ICNC), Volume: 6 pages 2798 - 2803 , aug 2010 .
- [48] G.E.P BOX and G.M JENKINS, "Time series analysis: forecasting and control", CA: Holden-Day, San Francisco, pp 575, 1976.
- [49] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems", Sci., vol. 197, p. 287, 1977.
- [50] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "Neuralgas' network for vector quantization and its application to time-series prediction", IEEE Trans. Neural Networks, vol. 4, pp. 558–569, 1993.
- [51] I. Rivals, "Modélisation et commande de processus par réseaux de neurones; application au pilotage d'un véhicule autonome", Thèse de Doctorat de l'Université Pierre et Marie Curie - Paris VI, 1995.
- [52] A. B. Will, S. Hui and S. Zak, "Sliding Mode Wheel Slip Controller for an Antilock Braking System", Int. J. of Vehicle Design, 19(4), p523-539, 1998.
- [53] Ming-chin Wu, Ming-chang Shih, "Simulated and experimental study of hydraulic anti-lock braking system using sliding-mode PWM control", Elsevier Science Ltd, Mechatronics 13, 331–351, 2003.
- [54] "The laboratory ABS system user manual", INTECO Ltd, 2006.



ANNEXE

Annexe

Les Algorithmes Génétiques Simples

1. Introduction

Les Algorithmes Génétiques (AG) sont des techniques d'optimisation stochastiques qui tentent d'imiter les processus d'évolution naturelle des espèces et de la génétique. Ils agissent sur une population d'individus assujettis à une sélection darwinienne: les individus les mieux adaptés à leur environnement survivent et peuvent se reproduire. Ils sont alors soumis à des mécanismes de recombinaisons analogues à ceux de la génétique.

2. Algorithmes génétiques

Les AG ont été initialement développées par John Holland et ses collaborateurs dans les années 1970 puis Goldberg, présentent des qualités intéressantes pour la résolution de problèmes combinatoires complexes. Ils ont été introduits comme algorithmes de recherche, ensuite ils ont été utilisés comme outils d'optimisation.

La génétique représente un individu par un code, c'est-à-dire un ensemble de données (appelées chromosomes), identifiant complètement l'individu qui se compose d'un ou plusieurs chromosomes, les chromosomes eux même constitués de gènes qui contiennent les caractères héréditaires de l'individu.

Le fonctionnement des AG est simple, on part avec une population de solutions possibles (chromosomes) initiales arbitrairement choisies. On évalue leur performance (fitness) relative. Sur la base de ces performances on crée une nouvelle population en utilisant des opérateurs évolutionnaires tels que: la sélection, le croisement et la mutation. On recommence ce cycle jusqu'à ce que l'on trouve une solution satisfaisante.

Les AG agissent sur une population d'individus qui évolue durant une succession d'itérations appelées générations jusqu'à ce qu'un critère d'arrêt, qui prend en compte a priori la qualité des solutions obtenues, soit vérifié, ce critère peut aussi être un nombre de générations défini à priori. Seuls les individus bien adaptés à leur environnement peuvent survivre et se reproduire.

Pour un problème d'optimisation donné, un individu représente un point de l'espace d'état. On lui associe la valeur du critère à optimiser. L'algorithme génère ensuite de façon itérative des populations d'individus sur les quelles on applique des processus de sélection, de croisement et de mutation, la sélection a pour but de favoriser les meilleurs éléments de la population, tandis que le croisement et la mutation assurent une exploration efficace de l'espace d'état.

Durant chaque génération, une succession d'opérateurs est appliquée aux individus d'une population pour engendrer la nouvelle population à la génération suivante. Lorsqu'un ou plusieurs individus sont utilisés par un opérateur, on convient de les désigner comme des *parents*. Les individus résultants de l'application de l'opérateur sont des *enfants*.

Une procédure d'évaluation est nécessaire à la détermination de la "force" de chacun des individus de la population. De génération en génération, la force des individus de la population augmente et après un certain nombre d'itérations, la population est constituée d'individus tous très forts, soit de solutions quasi optimales du problème posé.

2.1. Objectif des AG

Le but des Algorithmes Génétiques est de déterminer les extrêmes d'une fonction, $f: E \rightarrow \mathcal{R}$, où E est un ensemble quelconque appelé espace de recherche et f est appelée fonction d'*adaptation* ou fonction d'*évaluation* ou encore fonction de *fitness*. La fonction agit comme une boîte noire pour l'AG. Aussi des problèmes très complexes peuvent être approchés par programmation génétique sans avoir de compréhension particulière du problème.

2.2. Structure de l'Algorithme Génétique

Un AG est un algorithme itératif de recherche d'optimum, il manipule une population de taille constante. Cette population est formée de points appelés chromosomes, chaque chromosome représente le codage d'une solution possible au problème à résoudre, il est constitué d'un ensemble d'éléments appelés gènes.

A chaque itération, appelée génération, est créée une nouvelle population avec le même nombre de chromosomes. Cette génération consiste en des chromosomes mieux "adaptés" à leur environnement tel qu'il est représenté par la fonction d'évaluation (fonction coût ou fitness).

Au fur et à mesure des générations, les chromosomes vont tendre vers l'optimum de la fonction d'évaluation. La création d'une nouvelle population à partir de la précédente se fait par application des opérateurs génétiques qui sont : la *sélection*, le *croisement* et la *mutation*.

L'organigramme présenté dans la figure A.1, illustre la structure générale de l'AG. Nous détaillons dans la suite les diverses phases qui le constituent et présenterons les mécanismes associés à chacune d'entre, sous l'hypothèse implicite que le codage est binaire.

Un Algorithme Génétique générique à la forme suivante :

- 1) Initialiser la population initiale P.
- 2) Evaluer P.
- 3) Tant Que (Pas Convergence) faire :
 - a) P' = Sélection des Parents dans P
 - b) P' = Appliquer l'opérateur de Croisement sur P'
 - c) P' = Appliquer l'opérateur de Mutation sur P'
 - d) P = Remplacer les Anciens de P par leurs descendants de P'
 - e) Evaluer P

Fin Tant Que

Le critère de convergence peut être de nature diverse, par exemple: un taux minimum qu'on désire atteindre d'adaptation de la population au problème, un certain temps de calcul à ne pas dépasser, une combinaison de ces deux points.

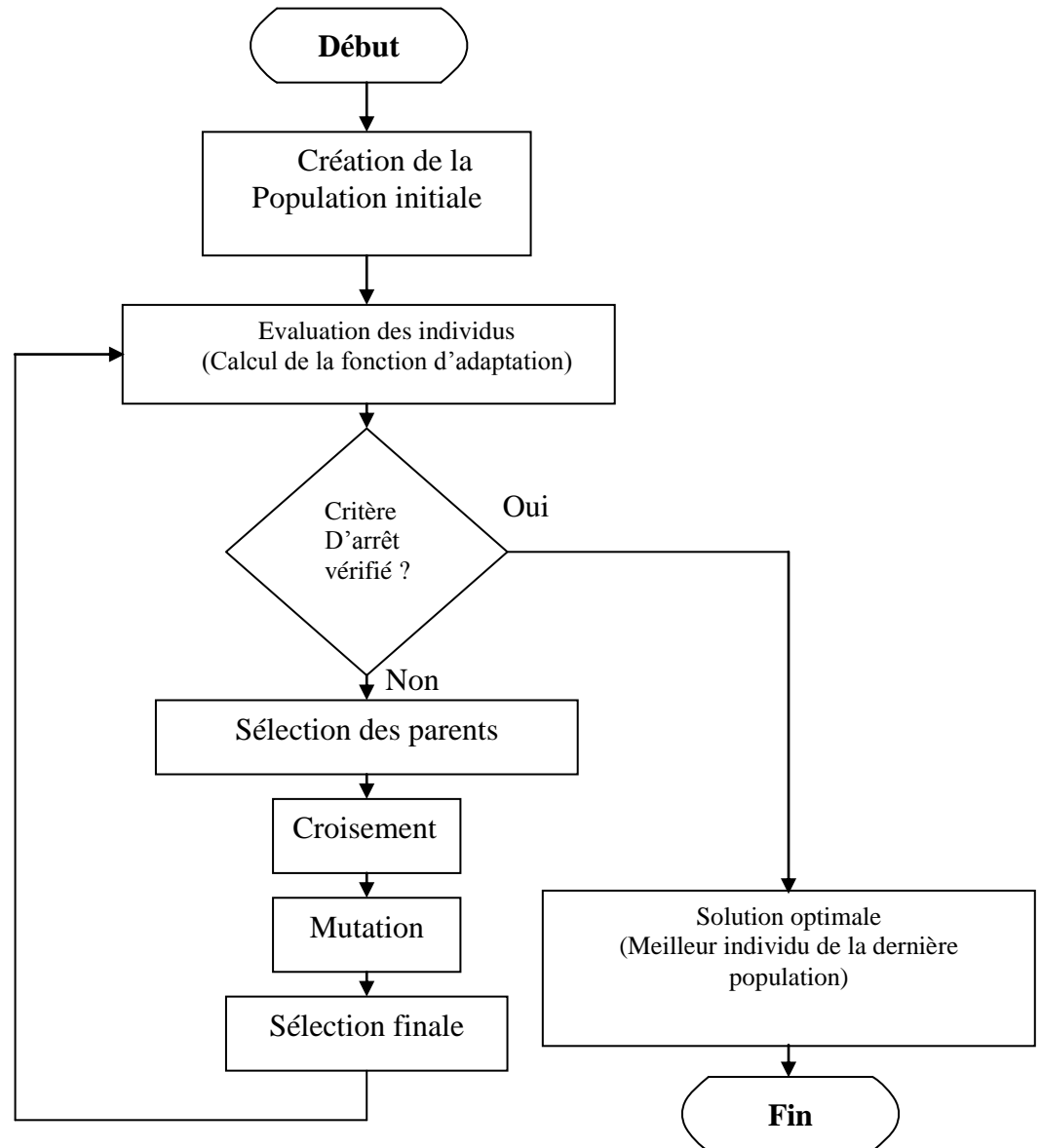


Figure A.1 : Organigramme général d'un Algorithme Génétique

2.4. Population Initiale

La première étape de l'algorithme est la formation de la population initiale, c'est-à-dire le choix des individus de départ que nous allons faire évoluer. Cette population est constituée de " N " individus, ceux-ci représentent un ensemble de solutions de départ. On pourrait prendre des individus régulièrement répartis dans l'espace. Néanmoins, une initialisation aléatoire est plus simple à

réaliser. Les individus sont tirés au hasard selon une distribution uniforme. Le choix de la population initiale peut conditionner fortement la rapidité de l'algorithme, il doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. La taille de la population doit être choisie de façon à réaliser un bon compromis entre temps de calcul et qualité du résultat.

2.5. Evaluation

Pour calculer le coût d'un point de l'espace de recherche, on utilise une fonction d'évaluation. L'évaluation d'un individu ne dépend pas de celle des autres individus, le résultat fourni par la fonction d'évaluation va permettre de sélectionner ou de refuser un individu pour ne garder que ceux ayant le meilleur coût de la population courante. Cette méthode permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population génération par génération. En général l'AG est utilisé pour *maximiser* une performance (population de plus en plus "forte"), mais cela n'a pas empêché les chercheurs de l'exploiter pour résoudre des problèmes de minimisation.

2.6. Les Opérateurs d'un AG

Les opérateurs génétiques sont appliqués à une population initiale de façon à produire, dans le temps, des populations successives de qualité. Les quatre opérateurs génétiques de base sont la reproduction, la sélection, le croisement et la mutation. Ces opérateurs agissent selon divers critères qui lui sont propres (valeur sélective des individus, probabilité d'activation de l'opérateur, etc.).

2.6.1. La reproduction

La reproduction est le processus selon lequel des individus de la population globale sont choisis suivant la valeur de leur fonction objectif f , qu'on désire maximiser ou minimiser. Dans le cas de maximisation, plus la valeur de la fonction objective d'un individu est élevée, plus ces individus ont des chances d'être choisis pour la reproduction. Si un individu n'a pas une valeur de fonction

objective élevée, il a peu de chance d'être choisi pour reproduction. Ainsi, il disparaîtra d'une certaine manière, puisque ses gènes ne se retrouveront pas dans la prochaine génération, et vis versa pour la minimisation.

2.6.2. Opérateur de Sélection

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. Cet opérateur est peut-être le plus important puisqu'il permet aux individus d'une population de survivre, de se reproduire ou de mourir. En règle générale, la probabilité de survie d'un individu sera directement reliée à son efficacité relative au sein de la population. Il existe plusieurs méthodes pour la sélection des individus à reproduire.

2.6.2.1. Les différentes méthodes de sélection

a. sélection par roulette biaisée (*Sélection proportionnelle*)

Cette méthode est la plus connue et la plus utilisée, "RWS" (Roulette Wheel Sélection), Avec cette méthode chaque individu a une chance d'être sélectionné proportionnelle à sa performance, donc plus les individus sont adaptés au problème, plus ils ont de chances d'être sélectionnés. Cette technique de sélection consiste à associer à chaque individu de la population un segment dans la roulette. La largeur de ce segment est proportionnelle à sa fitness ou, plus précisément, à une probabilité d'être sélectionné proportionnelle à sa fitness. La valeur de la fonction fitness d'un individu particulier x étant $f(x)$, la probabilité P_s avec laquelle il sera réintroduit dans la nouvelle population de

taille N est donnée par la formule A.1:
$$P_s(x) = \frac{f(x)}{\sum_{i=1}^N f(x_i)} \dots\dots\dots (A.1)$$

La sélection d'un individu revient à choisir aléatoirement un point du segment avec une distribution de probabilité uniforme. Avec ce système, les grands segments, c'est à dire les bons individus, seront plus souvent adressés que les petits. Une certaine diversité est cependant maintenue, car même les individus les moins performants conservent une chance d'être choisis. La place accordée

à chacun des individus étant en relation avec sa valeur d'adaptation. Ensuite, la bille est lancée et s'arrête sur un individu. Les meilleurs individus peuvent ainsi être tirés plusieurs fois et les plus mauvais ne jamais être sélectionnés. Cette roulette est représentée par la figure A.2

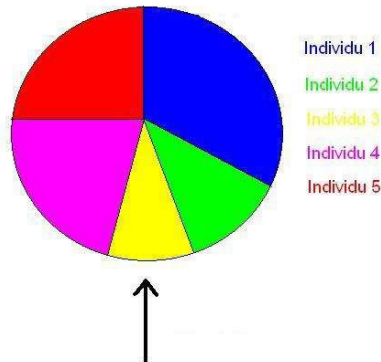


Figure A.2 : La méthode de sélection de la loterie biaisée

b. Sélection par rang

La sélection précédente rencontre des problèmes lorsque la valeur d'adaptation des individus varie énormément. Si la meilleure fonction d'évaluation d'un individu représente 90% de la roulette alors les autres chromosomes auront très peu de chance d'être sélectionnés et on arriverait à une stagnation de l'évolution. La sélection par rang trie d'abord la population par fitness. Ensuite, chaque individu se voit associé un rang en fonction de sa position. Ainsi le plus mauvais aura le rang 1, le suivant 2 et ainsi de suite jusqu'au meilleur individu qui aura le rang N (pour une population de N individus). La sélection par rang est la même que par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation.

c. sélection par tournois

La sélection par tournoi consiste à choisir aléatoirement un certain nombre d'individus, et à sélectionner pour la reproduction celui qui a la plus grande adaptation. Cette étape est répétée autant de fois qu'il y a d'individus à remplacer dans la génération. Donc le principe de cette méthode est le suivant : on effectue un tirage avec remise de deux individus de la même population P, et

on les compare. Celui qui a la fitness la plus élevée se sélectionne avec une probabilité p_s comprise entre 0.5 et 1. On répète ce processus N fois de manière à obtenir les N individus de P' qui serviront de parents.

d. Sélection à reste stochastique

Dans ce mode de sélection, le nombre de copies $n(x_i)$ d'un individu x_i est fixé par le rapport de sa performance avec la performance moyenne de la population.

$$n(x_i) = \text{partie entière} \left(\frac{f(x_i)}{f_{\text{moy}}} \right) \dots \dots \dots (A.2)$$

$$\text{Où } f_{\text{moy}} = \frac{1}{N} \sum_{i=1}^N f(x_i) \dots \dots \dots (A.3)$$

Dans un premier temps, on fixe $\alpha = \sum_{i=1}^N n(x_i)$ individus à partir de l'équation (A.2) puis on complète la population en associant à chaque individu une probabilité d'être sélectionné.

$$P_s(x_i) = \frac{1}{N - \alpha} \left(\frac{f(x_i)}{f_{\text{moy}}} \right) - n(x_i) \dots \dots \dots (A.4)$$

e. La sélection universelle stochastique

Cette méthode possède une variance faible, elle introduit peu de diversité, c'est la cause qui a fait que cette méthode est très peu utilisée, c'est pour cette raison que nous n'entrerons donc pas dans les détails, on se contentera d'exposer sa mise en œuvre : On prend l'image d'un segment découpé en autant de sous-segments qu'il y a d'individus. Les individus sélectionnés sont désignés par un ensemble de points équidistants.

f. Sélection élitiste

Cette méthode est utilisée en plus des méthodes précédentes et non pas séparément. A la création d'une nouvelle population, il y a de grandes chances que les meilleurs individus soient perdus après les opérations d'hybridation et de mutation. Pour éviter cela, on utilise la méthode d'élitisme. Cette méthode consiste à sélectionner les N individus dont on a besoin pour la nouvelle génération P' en prenant les N meilleurs individus de la population P après

l'avoir triée selon la fitness de ses individus. Cette méthode améliore considérablement les AG, car elle permet de ne pas perdre les meilleures solutions.

2.6.3. Opérateur de Croisement (hybridation ou crossover)

Le croisement utilisé par les AG est la transposition informatique du mécanisme qui permet, dans la nature, la production de chromosomes qui héritent partiellement des caractéristiques des parents. Son rôle fondamental est de permettre la recombinaison des informations présentes dans le patrimoine génétique de la population.

Cet opérateur est appliqué après avoir appliqué l'opérateur de sélection. Les chromosomes des parents sont copiés et recombinaison de façon à former deux descendants possédant des caractéristiques issues des deux parents.

Le croisement a pour but d'enrichir la diversité de la population en manipulant les composantes des individus. Généralement, les croisements sont envisagés avec deux parents et génèrent deux enfants.

La valeur de la probabilité de croisement est choisie généralement entre 0.5 et 0.9, plus elle est élevée plus la population subit de changements importants. Cette probabilité représente la fréquence à laquelle les hybridations sont appliquées.

L'opérateur de croisement est un mécanisme qui procède en trois étapes :

- La sélection aléatoire des deux parents.
- Le choix aléatoire du site de croisement.
- La génération des deux enfants en faisant permuter les génotypes des deux parents à partir du site du croisement.

2.6.3.1. Les méthodes de croisement

a. croisement à un site

Le croisement à un site est le croisement le plus simple. Ce type de croisement consiste à échanger les gènes de chacun des parents de longueur L (longueur du chromosome), on sélectionne aléatoirement un site de coupure K qui soit

compris entre 1 et $L-1$, puis on subdivise le chromosome de chacun des parents en deux parties de part et d'autre de ce point. On échange ensuite les deux sous chaînes terminales de chacun des deux chromosomes, Le changement se fait entre le site sélectionné et la position finale L des deux chaînes ce qui produit deux nouveaux chromosomes (enfants). Ce type de croisement est représenté par figure A.3

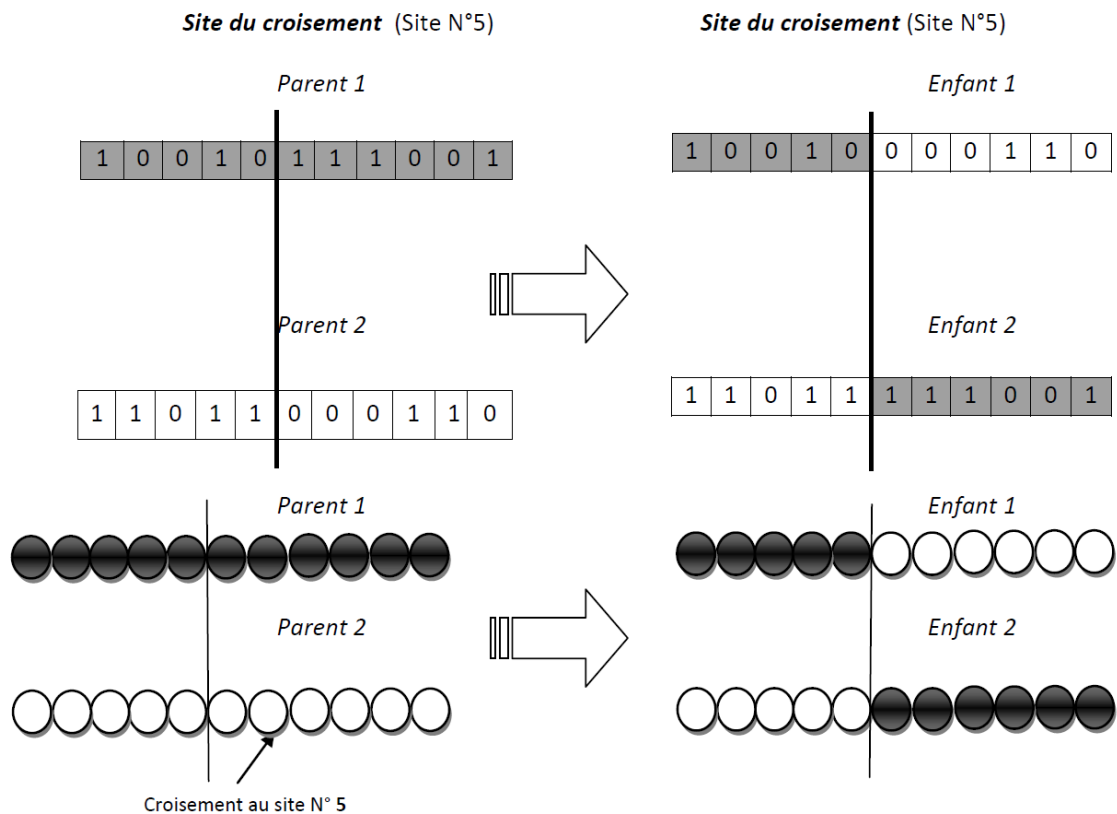


Figure A.3. : Représentation schématique du croisement à un site (Site N°5).

b. Croisement à plusieurs sites

Ce type de croisement peut être vu comme une généralisation du croisement à un site, en découpant le chromosome en K sous chaînes, puis on choisit à l'hasard plusieurs sites de croisement, et on échange les gènes des deux parents en créant ainsi deux enfants (figure A.4).

Croisement à plusieurs sites (1, 5...n)

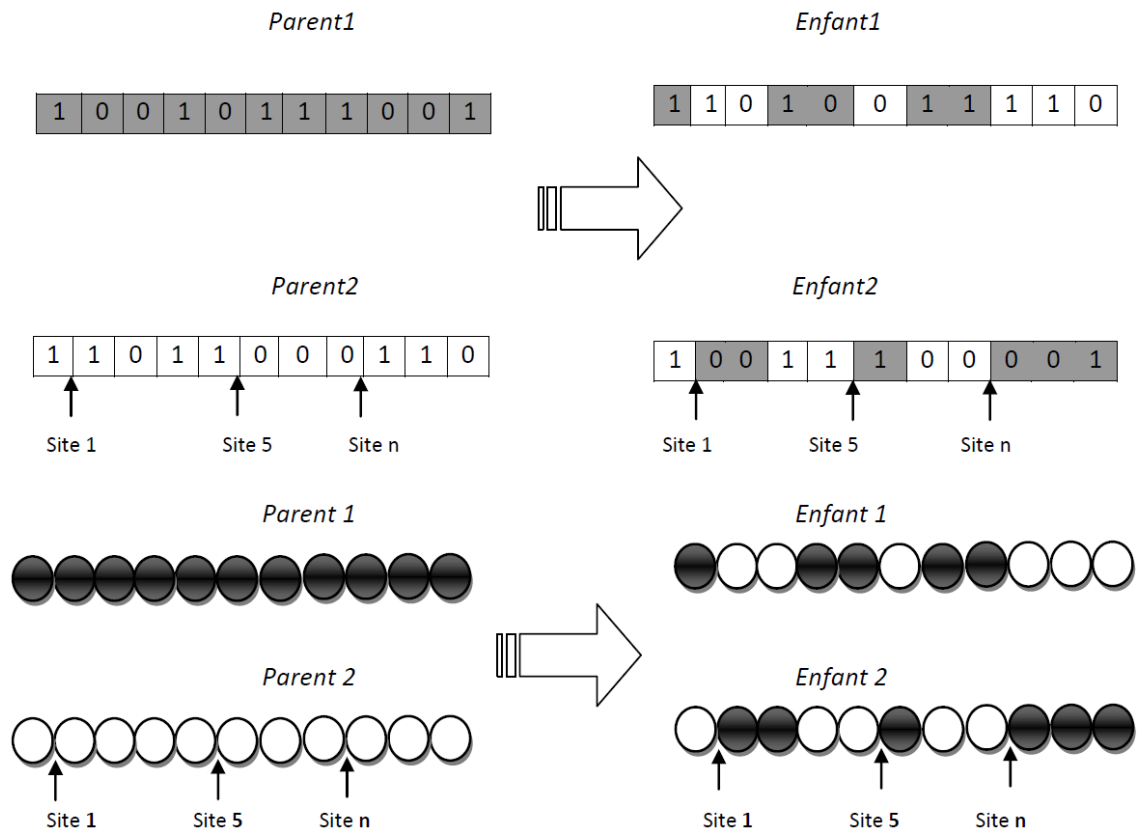


Figure A.4: Représentation schématique du croisement à plusieurs sites

2.6.4. L'opérateur de Mutation

La mutation est une modification aléatoire d'un chromosome, il s'agit de modifier aléatoirement la valeur d'un gène de chromosome. Cet opérateur de mutation apporte à l'AG l'aléa nécessaire à une exploration efficace de l'espace afin de créer un nouvel individu qui n'existait pas auparavant, elle a donc pour rôle le maintien d'une certaine diversité dans la population et protège les individus contre une perte des informations essentielles contenues dans leurs gènes.

Les individus de la population issue du croisement subissent au processus de mutation avec une probabilité P_m . Cet opérateur garantit que l'AG sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Le but est d'éviter à l'AG de

converger vers des extrema locaux de la fonction et de permettre de créer des éléments originaux. Si cette opération génère un individu plus faible il sera éliminé par la suite. La mutation n'intervient que sur une partie suffisamment petite pour ne pas détruire les caractéristiques qui ont été sélectionnées mais elle est importante pour apporter des gènes nouveaux à un individu (figure A.5). La probabilité de mutation représente la fréquence à laquelle les gènes d'un chromosome sont mutés. Si la mutation est appliquée, une partie du chromosome est changée, sinon le fils est inséré dans la nouvelle population sans changement.

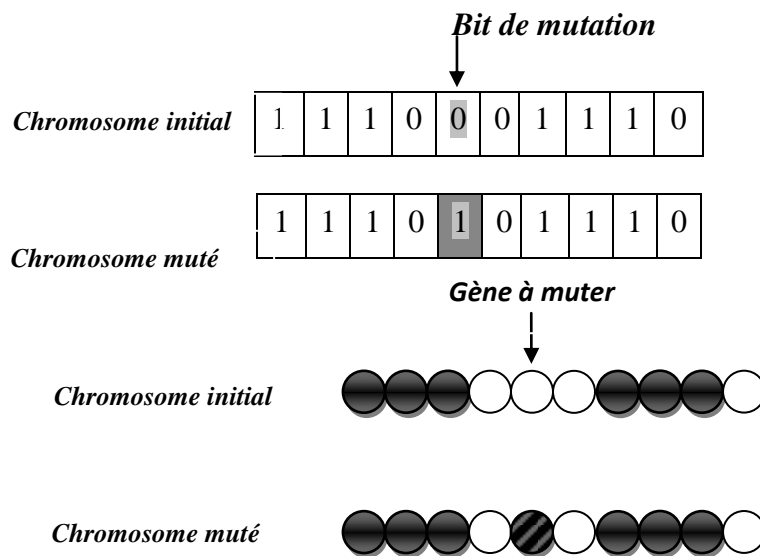


Figure A.5: Représentation schématique d'une opération de mutation.

2.6.5. La sélection des individus d'une nouvelle génération (sélection finale)

A la suite de l'application des opérateurs de recombinaison génétique précédents, la population comporte $2N$ individus (N enfants et N parents). Pour construire la nouvelle génération, il faut donc éliminer N individus, pour cela plusieurs méthodes de sélection sont proposées:

2.6.5.1. La sélection par Compétition

Une compétition aura lieu entre parents et enfants pour déterminer les individus de la nouvelle génération. Ainsi, les enfants peuvent être choisis dans

la nouvelle population si leur adaptation est supérieure à celle de leurs parents à rang équivalent.

2.6.5.2. Steady State Sélection

C'est une variante de la sélection précédente, qui consiste à effectuer une compétition après chaque recombinaison génétique, entre parents et enfants en retenant les deux meilleurs individus parmi les quatre.

2.6.5.3. La sélection par descendance

Dans ce type de sélection, il n'y a aucune compétition entre parents et enfants. La population de la nouvelle génération est obtenue par descendance, les enfants remplacent automatiquement leurs parents quelle que soit leur adaptation.

L'inconvénient de ce type de sélection est que l'on risque de voir disparaître les caractéristiques génétiques des parents les mieux adaptés si elles n'ont pas été totalement transmis lors de la recombinaison génétique.

2.7. Le codage

Chaque paramètre d'une solution est assimilé à un *gène*, toutes les valeurs qu'il peut prendre sont les *allèles* de ce *gène*, on doit trouver une manière de coder chaque *allèle* différent de façon unique.

Avec le codage binaire, un gène est composé de un ou plusieurs bits, mais dans le codage réel, le gène est représenté par une seule valeur réelle, un chromosome est une suite de gène, un individu est représenté par un ensemble de chromosomes, et une population est un ensemble d'individus (figure A.6). Il y a trois principaux types de codage utilisables.

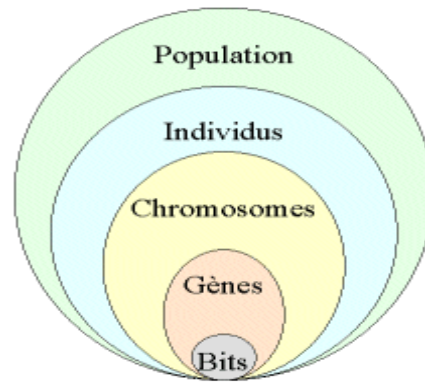


Figure A.6 : Niveaux d'organisation d'une population d'un AG

2.7.1. Le codage binaire

Dans ce type de codage, chaque gène dispose du même alphabet binaire {0, 1}. Un gène est alors représenté par un entier long 32 bits, les chromosomes et les individus qui sont des suites de gènes sont représentés par des vecteurs.

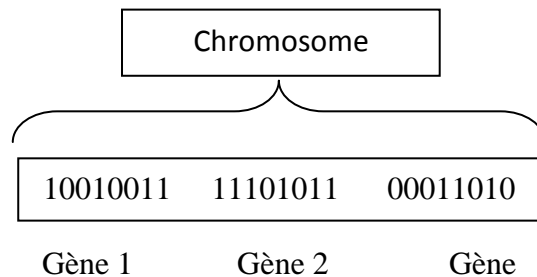


Figure A.7 : Structure d'un chromosome codé en binaire

2.7.2. Le codage réel

Cela peut-être utile notamment dans le cas où l'on recherche le maximum d'une fonction réelle.

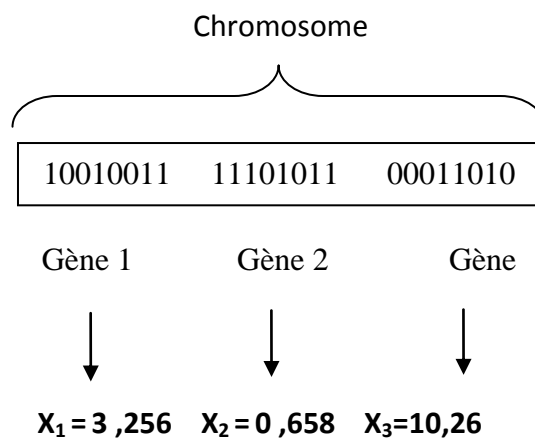


Figure A.8 : Illustration schématique du codage des variables réelles

2.7.3. Le codage de Gray

Dans le cas d'un codage binaire on utilise souvent la "*distance de Hamming*" comme mesure de différence entre deux éléments de population, cette mesure compte les différences de bits de même rang de ces deux séquences.

En effet, deux éléments voisins en termes de distance de *Hamming* ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant un "*codage de Gray*": le codage de Gray est un codage qui a comme propriété qu'entre un élément n et un élément $n+1$, donc voisin dans l'espace de recherche, un seul bit diffère.

2.8. Les AG à codage réel

Les AG à codage binaire (ou à codage classique) sont moins efficaces dans le cas où ils seraient appliqués à des problèmes multidimensionnels, à grande précision ou des problèmes continus; dans ce type de codage les variables (les gènes) dans le chromosome sont en binaires (une chaîne composée des 0 et des 1) ce qui nécessite à chaque fois de décoder ces chaînes pour calculer leurs valeurs en réel avant de calculer la fonction du coût, et ça se fait pour chaque individu et à chaque itération. Mais dans les AG à codage réel, les variables apparaissent directement dans le chromosome en réel et sont modifiées par des opérateurs génétiques spéciaux, chaque chromosome est en fait un vecteur dont les coordonnées sont les variables du processus d'optimisation. Lors de l'évolution de la population, les opérateurs génétiques assurant la modification des chromosomes doivent respecter cette contrainte imposée aux valeurs prises par les variables. Plusieurs opérateurs génétiques associés au codage par nombres réels ont été décrits dans la littérature.

2.8.1. Les opérateurs de la recombinaison génétique du codage réel

Dans ce type de codage les opérateurs de la recombinaison génétique agissent d'une façon différente de celle à codage binaire.

Dans ce qui suit :

$r \in [0,1]$: est un nombre aléatoire (à distribution uniforme).

$t = 0, 1, \dots, T$: Est le numéro de la génération.

S_v et S_w : sont les chromosomes sélectionnés par l'opérateur génétique.

$k \in \{1, 2, \dots, N\}$: Est la position d'un élément dans le chromosome.

V_k^{\min} et V_k^{\max} : sont respectivement les limites inférieures et supérieures de l'élément dont la position dans le chromosome est k .

2.8.1.1. L'opérateur du croisement

Concernant l'opérateur du croisement, les chromosomes sont sélectionnés par paires (S_v, S_w) . trois types de croisement à codage réel ont été proposés.

a. Le croisement arithmétique simple

S_v^t et S_w^t sont croisés au site k . Les enfants obtenus comme résultat de ce croisement sont : $S_v^{t+1} = (v_1, \dots, v_k, w_{k+1}, \dots, w_N)$ et $S_w^{t+1} = (w_1, \dots, w_k, v_{k+1}, \dots, v_N)$ où k est choisi aléatoirement de l'ensemble $\{2, \dots, N-1\}$

b. Le croisement arithmétique entier

Les enfants S_v^{t+1} et S_w^{t+1} obtenus sont le résultat d'une combinaison linéaire des deux parents S_v^t et S_w^t : $S_v^{t+1} = r.(S_v^t) + (1-r).(S_w^t)$ et $S_w^{t+1} = r.(S_w^t) + (1-r).(S_v^t)$

c. Le croisement heuristique

S_v^t et S_w^t sont combinés tel que : $S_v^{t+1} = S_v^t + r.(S_w^t - S_v^t)$ et $S_w^{t+1} = S_w^t + r.(S_v^t - S_w^t)$

2.8.1.2. L'opérateur de mutation

Dans l'opération de la mutation un seul chromosome est sélectionné. Trois types de mutation à codage réel ont été proposés.

a. La mutation uniforme

L'élément sélectionné aléatoirement $V_k, k = \{1, 2, \dots, N\}$ est remplacé par V_k' qui est une valeur aléatoire qui appartient à l'intervalle $[V_k^{\min}, V_k^{\max}]$. Le résultat est alors le chromosome : $S_v^{t+1} = (V_1, \dots, V_k', \dots, V_N)$

b. La mutation uniforme multiple

Le même principe que la méthode précédente, mais n variables du même chromosome sont sélectionnés aléatoirement, ou n est aléatoirement choisi de l'ensemble $\{1, 2, \dots, N\}$.

c. La mutation Gaussienne

Tout les éléments du chromosome sont mutés tel que :

$$S_v^{t+1} = (V_1', \dots, V_k', \dots, V_N'), \text{ ou : } V_k' = V_k + F_k, k = 1, 2, \dots, N.$$

F_k : Un nombre aléatoire tiré d'une distribution Gaussienne de moyenne nulle et une variance adaptative $\sigma_k = ((T - t)/T)(V_k^{\min} - V_k^{\max})$

2.9. Les caractéristiques d'un AG

Les AG, en tant qu'approche de résolution de problèmes, se caractérisent par un certain nombre d'aspect qui sont : le codage des paramètres du problème à traiter, l'espace de recherche de solution, la fonction d'évaluation servant à sélectionner les chromosomes parents et le rôle du hasard dans le choix des chromosomes.

2.9.1. Codage des paramètres

Selon Goldberg, l'utilisateur doit choisir le plus petit alphabet qui permette une expression naturelle des paramètres du problème (principe des alphabets minimaux).

2.9.2. L'espace de recherche

Selon les méthodes de recherche conventionnelles, le déplacement dans l'espace de recherche se fait d'un point à un autre en général par des règles de transition déterministes. On avance d'un pas à la fois : on évalue la solution et, si elle n'est pas satisfaisante, on passe à la prochaine valeur de paramètres. Une fois les variables d'optimisation choisies, il faut définir l'espace de recherche correspondant à l'aide de considérations physiques, technologiques et numériques.

2.9.3. La fonction d'évaluation

Les AG utilisent des fonctions d'évaluation dites objectives pour entreprendre une recherche efficace de structures plus performantes ; ils ne requièrent que des fonctions objectives associées aux individus. Les valeurs de la fonction objective serviront au processus de sélection des candidats aptes à la reproduction et au processus de survie de certaines espèces.

2.10. Les paramètres de base d'un AG

2.10.1. Population

La rapidité de l'algorithme est fortement dépendante du choix de la population initiale d'individus. Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel et plus simple de procréer aléatoirement des individus en faisant des tirages uniformes dans l'espace d'état en veillant à ce que les individus produits respectent les contraintes. Si par contre, des informations à priori sur le problème sont disponibles, les individus sont générés dans un sous domaine particulier afin d'accélérer la convergence. Les AG doivent travailler sur des populations importantes.

2.10.2. Taille de la population

Certaines approches utilisent une population à taille fixe, alors que d'autres proposent des tailles variables, qui évoluent au cours du temps. Au lieu d'utiliser une population de taille fixe ou la fonction d'adaptation sert à la sélection, elles proposent plutôt d'indexer l'espérance de vie d'un individu à sa fonction d'adaptation. Les meilleurs individus reçoivent une espérance de vie supérieure à celle des moins bons. Au fur et à mesure que la population évolue, les individus vieillissent et finissent par mourir lorsqu'ils arrivent au terme de leur espérance de vie.

2.10.3. Taux de croisement

Il détermine la proportion des individus qui sont croisés parmi ceux qui remplaceront l'ancienne génération. Si le taux de mutation est trop élevé, les structures performantes sont trop fortement détruites. Par contre, s'il est petit, la population se stagne

2.10.4. Taux de mutation

A chaque reproduction, le nouveau-né est soumis à un processus mutation. En cas de succès, un gène est modifié. Pour les populations faibles, nous constatons que le taux de mutation est très élevé.

2.10.5. Critère d'arrêt

Comme tous les algorithmes évolutionnaires, les AG sont dans l'absolu, sans fin, donc pour décider quand arrêter l'algorithme, la technique la plus courante et la plus simple est d'effectuer un nombre prédéfini d'itérations. On peut distinguer trois grandes familles de critères d'arrêt :

- Le temps ou le nombre d'itérations voulu est atteint.
- La fonction de coût est constante de puis quelque temps.
- La population est dominée par quelques individus.

2.10.6. Nombre maximum de générations

Ce nombre doit être suffisant pour permettre une exploration correcte de l'espace de recherche, mais pas trop grand pour ne pas devenir pénalisant sur le plan du temps de calcul.

Au terme de l'évolution de l'algorithme, une procédure de tri est effectuée sur tous les individus évalués pour trouver toutes les solutions optimales ayant la même valeur de la fonction *objective* mais correspondant à des individus différents.

ملخص

في هذا العمل نقدم أسلوباً لحل مشكلة الحصول على البنية الأمثل للشبكة العصبية، و يتمثل هذا الأسلوب في: العثور على أفضل هيكل للشبكة العصبية المستخدمة، و ضمان التعلم، و تحسين دالة التكلفة. لهذا الغرض استعملنا الخوارزميات الجينية متعددة الأهداف من نوع (الخوارزميات الجينية بفرز عدم الهيمنة) NSGA2 بهدف تحسين الشبكات العصبية متعددة الطبقات MLP و شبكات الدالة الشعاعية RBF مع اختيار و بدون اختيار مدخلات الشبكة و الانحدارات السابقة. و تستخدم هذه التقنية في مجالات النمذجة و التحكم، في هذا العمل استعملنا نهجين مختلفين لنمذجة ناظمين غير خطيين اثنين و لمراقبة نظام آخر. و بعد ذلك استخدمنا النهج الأفضل لمراقبة نظام مختبر غير الخطية للغاية، و هو نظام الكبح المضادة لقفلة فرملة العجلة ABS. و قد تم الحصول على نتائج مرضية للغاية.

Résumé

Dans ce travail nous proposons une technique permettant de résoudre le problème de l'obtention d'une architecture optimale de réseau de neurones, cette technique consiste à : trouver la meilleure structure du réseau de neurone utilisé, assurer son apprentissage, et optimiser la fonction coût. Pour cela nous avons utilisé les Algorithmes Génétique Multi-Objectifs de type NSGA2 (Algorithme génétique avec tri des non-dominés²) afin d'optimiser les réseaux de neurones multi couches MLP et à fonction de base radiale RBF avec et sans sélection des entrées du réseau et leurs régressions. Cette technique est utilisée dans les domaines de modélisation et du contrôle, dans ce travail nous avons utilisés deux approches différentes pour modéliser deux systèmes non linéaires et contrôler un autre. En suite nous avons exploité la meilleur approche afin de contrôler un système de laboratoire fortement non linéaire, qui est le système de freinage Anti Blocage des Roues (*Anti-lock Braking System*) ABS. Les résultats obtenus sont très satisfaisants.

Abstract

In this work we propose a technique to solve the problem of obtaining an optimal architecture of neural network, this technique consists of: find the best structure of the used neural network, ensure his learning, and optimize the cost function. For that, we used the multi-objective genetic algorithms type NSGA2 (*Non-dominated Sorting Genetic Algorithm²*) to optimize multi-layer neural networks MLP and radial basis function RBF with and without input and regressions selection. This technique is used in modeling and control, in this work we used two different approaches to model two nonlinear systems and control another. As a result we used the best approach to control a highly nonlinear laboratory system, which is the Anti-lock Braking System ABS. The results obtained are very satisfying.