

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique  
Université Mohamed Seddik Ben Yahia -  
JIJEL



N° d'ordre : .....  
Série : .....

*Faculté des Sciences et de Technologie  
Département d'Électronique*

*Spécialité : Électronique  
Option : Contrôle des Systèmes*

**THÈSE**  
EN VUE DE L'OBTENTION DU DIPLOME DE  
DOCTORAT 3<sup>ème</sup> Cycle

Présentée par :

**Mohammed Amin KHELIFA**

**Étude et application des algorithmes  
évolutionnaires aux systèmes  
chaotiques**

Soutenue le : 10/ 12/ 2016

Devant le Jury d'examen composé de :

<b>Mr.</b> K. KEMIH	<i>Prof.</i> Université MSB JIJEL	<b>Président</b>
<b>Mr.</b> A. BOUKABOU	<i>Prof.</i> Université MSB JIJEL	<b>Rapporteur</b>
<b>Mr.</b> T. BOUDEN	<i>Prof.</i> Université MSB JIJEL	<b>Examineur</b>
<b>Mr.</b> D. BOUDJEHEM	<i>MCA.</i> Université de Guelma	<b>Examineur</b>

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

## Résumé

La motivation du contrôle du chaos vient du fait, que le chaos permet à un système d'explorer chacune de ses dynamiques. Aujourd'hui, on commence à réaliser que la conception traditionnelle d'engineering qui essaye toujours de supprimer les comportements irréguliers et, par conséquent, d'éliminer le chaos afin d'éviter la complexité peut engendrer beaucoup d'incidents. Notre contribution à travers le travail développé dans cette thèse concerne la proposition et le développement de nouvelles approches et méthodes afin de pouvoir modéliser, stabiliser, contrôler et conduire la trajectoire des systèmes chaotiques dites multi-scroll vers des trajectoires bien spécifiques. Ces trajectoires sont les orbites périodiques instables du système chaotique.

**Mots clés :** chaos, multi-scroll, réseaux de neurones, prédiction, stabilisation, optimisation.

## Abstract

The motivation of controlling chaos comes from the fact that chaos allows a system to explore each of its dynamics. Today we are starting to realize that the traditional engineering design that always tries to remove irregular behavior and therefore eliminate chaos in order to avoid complexity can lead to many incidents. Our contribution through the work developed in this thesis concerns the proposal and the development of new approaches and methods in order to be able to model, stabilize, control and lead the trajectory of the so-called multi-scroll chaotic systems towards very specific trajectories. These trajectories are the unstable periodic orbits (UPO) of the chaotic system.

**Keywords :** chaos, multi-scroll, neural networks, prediction, stabilization, optimization.

## ملخص

الدافع للسيطرة (أو التحكم) على الفوضى يأتي من حقيقة أن الفوضى قد تسمح للنظام باستكشاف كل من ديناميكياته. اليوم بدأنا ندرك أنّ التصميم الهندسي التقليدي الذي يحاول دائماً إزالة السلوك غير النظامي، وبالتالي القضاء على الفوضى من أجل تجنب التعقيد يمكن أن يؤدي إلى العديد من الحوادث. تتعلق مساهمتنا من خلال العمل الذي تم تطويره في هذه الأطروحة باقتراح ووضع مقاربات وأساليب جديدة من أجل أن نكون قادرين على النمذجة، استقرار (أو تثبيت) ومراقبة مسار ما يسمى بنظم الفوضى نحو مسارات محددة جداً. هذه المسارات هي المدارات الدورية غير المستقرة للنظام الفوضوي.

**كلمات البحث:** الفوضى، متعددة التمرير، الشبكات العصبية، التنبؤ، تحقيق الاستقرار، التحسين.

# Remerciements

Avant tout, je tiens à remercier Dieu qui m'a donné la force et la volonté pour réaliser cette modeste thèse. J'ai eu le plaisir et la chance de travailler sous l'orientation de Mr **Abdelkrim. BOUKABOU**, Professeur à l'Université de MSB Jijel. Je tiens à lui exprimer toute ma reconnaissance de m'avoir proposé un sujet d'actualité et à diriger un tel travail. Il m'a fait profiter en toute sympathie et une grande compétence. Je lui suis également très reconnaissant de m'avoir partagé tous les outils et la documentation nécessaire ainsi que ses conseils précieux durant toute cette période.

Je tiens ensuite à remercier Mr **Karim. KEMIH**, Professeur à l'Université de MSB Jijel, pour l'honneur qu'il me fait en acceptant de présider mon jury de thèse. Je présente également, mes sincères remerciements à Mr **Toufik. BOUDEN**, Professeur à l'Université de MSB Jijel, Mr **Djalil. BOUDJEHEM**, Maître de conférences à l'Université de Guelma, et qui ont bien voulu me faire l'honneur d'examiner ce travail et d'être membres du jury.

Enfin je remercie toutes les personnes ayant participé de loin ou de près à ce travail, qu'ils trouvent ici l'expression de ma profonde gratitude.

# Dédicaces

## **A la mémoire de mon Père**

Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour vous.

Rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être.

Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation.

## **A ma chère mère**

Ta prière et ta bénédiction m'ont été d'un grand secours pour mener à bien mes études.

## **A tous les membres de ma famille**

Veillez trouver dans ce travail l'expression de mon affection

## **A tous mes amis**

## LISTE DES FIGURES

### Chapitre I

<b>Figure I.1</b> Exemple de trajectoire pour le système de Lorenz. ....	9
<b>Figure I.2</b> Étude du comportement dynamique pour la fonction logistique.....	12
<b>Figure I.3</b> Evolution dans le temps pour deux conditions initiales très proches. ....	15
<b>Figure I.4</b> La SCI illustrée par les trajectoires des boules de billard .....	16
<b>Figure I.5</b> Bifurcation de la fonction quadratique (de droite à gauche). ....	19
<b>Figure I.6</b> Divergence de deux trajectoires dans le plan de phase .....	20
<b>Figure I.7</b> Intersections de la trajectoire de l'attracteur de Rössler avec un plan $\varphi$ d'équation $y = 0, x \leq 4$ .....	22
<b>Figure I.8</b> Section de Poincaré des attracteurs à intervalle de temps régulier. ....	23
<b>Figure I.9</b> Aspects aléatoires des états du système de Lorenz.....	25
<b>Figure I.10</b> Attracteur étrange de Lorenz. ....	25
<b>Figure I.11</b> Exposants de Lyapunov du système de Lorenz .....	26
<b>Figure I.12</b> Evolution de l'équation logistique pour $r = 4$ . ....	27
<b>Figure I.13</b> Portrait de phase du système de Mackey-Glass avec $\tau = 8$ . ....	27

### Chapitre II

<b>Figure II.1</b> Modèle de base d'un neurone formel.....	31
<b>Figure II.2</b> Types de fonctions d'activation. ....	34
<b>Figure II.3</b> Structure d'un réseau de neurones statique PMC.....	36
<b>Figure II.4</b> Forme canonique d'un réseau de neurones dynamique .....	37
<b>Figure II.5</b> Optimum local et global d'une fonction.....	52
<b>Figure II.6</b> Classification des métaheuristiques.....	53
<b>Figure II.7</b> Squelette d'un algorithme évolutionnaire .....	55
<b>Figure II.8</b> Principe général d'un algorithme évolutionnaire. ....	56
<b>Figure II.9</b> Classes des algorithmes évolutionnaire (EA). ....	57
<b>Figure II.10</b> Diagramme illustratif de l'AG .....	62

### Chapitre III

<b>Figure III.1</b> Système d'apprentissage pour la modélisation.....	66
<b>Figure III.2</b> Système d'apprentissage pour la commande.....	67

<b>Figure III.3</b> Modélisation de la fonction logistique en utilisant 3 neurones dans la couche cachée. ....	68
<b>Figure III.4</b> Modélisation de la fonction logistique en utilisant 4 neurones dans la couche cachée. ....	69
<b>Figure III.5</b> Modélisation de la fonction logistique en utilisant 5 neurones dans la couche cachée. ....	69
<b>Figure III.6</b> Modélisation de la fonction logistique en utilisant 15 neurones dans la couche cachée. ....	69
<b>Figure III.7</b> Evolution temporelle des variables d'états du système et du modèle neuronal. ....	70
<b>Figure III.8</b> Représentation dans l'espace des phases du système de Lorenz et du modèle neuronal. ....	71
<b>Figure III.9</b> Performance du modèle neuronal. ....	72
<b>Figure III.10</b> Erreurs d'approximation du système de Lorenz par le PMC. ....	72
<b>Figure III.11</b> Indice de régression du modèle neuronal. ....	72
<b>Figure III.12</b> Evolution temporelle des variables d'états du système et du modèle neuronal. ....	73
<b>Figure III.13</b> Erreurs d'approximation du système de Chen par le PMC. ....	74
<b>Figure III.14</b> Représentation dans l'espace des phases du système et du modèle neuronal de Chen. ....	74
<b>Figure III.15</b> Evolution de l'apprentissage du PMC. ....	75
<b>Figure III.16</b> Indice de régression du modèle neuronal. ....	75
<b>Figure III.17</b> Modélisation du système de Henon. ....	88
<b>Figure III.18</b> Le modèle neuronal contrôlé du système de Henon. ....	89
<b>Figure III.19</b> Modélisation du système de Duffing. ....	90

## Chapitre IV

<b>Figure IV.1</b> Circuit de Chua a $n$ -scroll. ....	95
<b>Figure IV.2</b> Système de Chen a $n$ -scroll. ....	96
<b>Figure IV.3</b> Bloc diagramme du contrôle. ....	99
<b>Figure IV.4</b> Performance de différentes structures RN. ....	104
<b>Figure IV.5</b> Modélisation du circuit de Chua multi-scroll. ....	105
<b>Figure IV.6</b> Modélisation du système de Chen multi-scroll. ....	105
<b>Figure IV.7</b> Performance de différentes structures RN. ....	106
<b>Figure IV.8</b> Stabilisation du modèle RN du circuit de Chua sur $X_0$ . ....	107
<b>Figure IV.9</b> Stabilisation du modèle RN du circuit de Chua sur $X \pm 1$ . ....	108

<b>Figure IV.10</b>	Stabilisation du modèle RN du circuit de Chua sur $X \pm 2$ .....	108
<b>Figure IV.11</b>	Stabilisation du modèle RN du circuit de Chua sur $X \pm 3$ .....	109
<b>Figure IV.12</b>	Stabilisation du modèle RN du circuit de Chua sur $X \pm 4$ .....	109
<b>Figure IV.13</b>	Stabilisation du modèle RN du circuit de Chua sur $X \pm 5$ .....	110
<b>Figure IV.14</b>	Stabilisation du modèle RN du circuit de Chua sur $X \pm 6$ .....	110
<b>Figure IV.15</b>	Stabilisation du modèle RN du circuit de Chua sur $X \pm 7$ .....	111
<b>Figure IV.16</b>	Stabilisation du modèle RN du système de Chen sur $X_0$ .....	112
<b>Figure IV.17</b>	Stabilisation du modèle RN du système de Chen sur $X \pm 1$ .....	112
<b>Figure IV.18</b>	Stabilisation du modèle RN du système de Chen sur $X \pm 2$ .....	113
<b>Figure IV.19</b>	Stabilisation du modèle RN du système de Chen sur $X \pm 3$ .....	113
<b>Figure IV.20</b>	Stabilisation du modèle RN du système de Chen sur $X \pm 4$ .....	114
<b>Figure IV.21</b>	Stabilisation du modèle RN du système de Chen sur $X \pm 5$ .....	114



## Liste des Tableaux

### Chapitre I

<b>Tableau I.1</b> Classification des régimes permanents en fonction du spectre Lyapunov .....	20
--	----

### Chapitre II

<b>Tableau II.1</b> Définitions de symbolisme.....	32
--	----

<b>Tableau II.2</b> Fonctions d'activation $a = f(n)$ .....	33
---	----

### Chapitre IV

<b>Tableau IV.1</b> Performance de différentes structures des RN modélisant le circuit de Chua. .....	102
--	-----

<b>Tableau IV.2</b> Performance de différentes structures des RN modélisant le Système de Chen. .....	103
--	-----

## Sommaire

Liste des figures .....	i
Liste des tableaux .....	iv
Sommaire .....	v
Introduction générale .....	1

### Chapitre I : Introduction aux systèmes dynamiques et chaos

<b>1. Introduction</b> .....	<b>5</b>
<b>2. Les systèmes dynamiques</b> .....	<b>7</b>
2.1. Temps continu .....	7
2.2. Temps discret .....	9
2.3. Comportement des systèmes dynamiques .....	10
2.3.1. <i>Point d'équilibre</i> .....	11
2.3.2. <i>Régime périodique</i> .....	11
2.3.3. <i>Régime quasi-périodique</i> .....	13
2.3.4. <i>Régime chaotique</i> .....	13
2.4. Evaluation du comportement dynamique .....	13
<b>3. Concepts mathématiques et définitions</b> .....	<b>14</b>
<b>4. La bifurcation (Route vers le chaos)</b> .....	<b>17</b>
<b>5. Les exposants de Lyapunov</b> .....	<b>19</b>
<b>6. Section de Poincaré</b> .....	<b>21</b>
<b>7. Les dynamiques chaotiques</b> .....	<b>24</b>
7.1. Le chaos continu .....	24
7.2. Le chaos discret .....	26
7.3. Le chaos à retard .....	27
<b>8. Applications du chaos</b> .....	<b>28</b>
<b>9. Nouveaux attracteurs chaotiques</b> .....	<b>28</b>
<b>10. Conclusion</b> .....	<b>29</b>

## Chapitre II : Techniques de l'intelligence computationnelle

<b>1. Introduction .....</b>	<b>30</b>
<b>2. Réseaux de neurones artificiels .....</b>	<b>30</b>
2.1. Neurone formel .....	31
2.2. Types des réseaux de neurones .....	34
2.2.1. <i>Réseaux de neurones statiques (Feed-forward)</i> .....	35
2.2.2. <i>Réseaux de neurones dynamiques (Feed-back)</i> .....	37
2.3. Propriétés des réseaux de neurones artificiels .....	38
2.3.1. <i>Le parallélisme</i> .....	38
2.3.2. <i>La capacité d'adaptation</i> .....	38
2.3.3. <i>La facilité de construction</i> .....	38
2.3.4. <i>L'approximation universelles parcimonieuse</i> .....	39
<b>3. Réseaux de neurones et régression non linéaire .....</b>	<b>40</b>
<b>4. Apprentissage des réseaux de neurones .....</b>	<b>40</b>
<b>5. La conception d'un réseau de neurones .....</b>	<b>42</b>
5.1. Choix et préparation des échantillons .....	42
5.2. Élaboration de la structure du réseau de neurones .....	43
5.3. Apprentissage du réseau de neurones .....	43
5.4. Validation et Test .....	43
<b>6. Apprentissage des réseaux PMC (réseaux multicouches) .....</b>	<b>44</b>
6.1. Algorithme de retro-propagation du gradient (RP) .....	44
6.2. Variantes de l'algorithme de RP .....	46
<b>7. Problèmes pour la généralisation .....</b>	<b>47</b>
7.1. Compromis Biais-Variance .....	47
7.2. Choix de l'architecture .....	48
7.3. Stopper l'apprentissage avant convergence .....	49
7.4. Régularisation .....	49
7.5. Bruitage .....	50
<b>8. Optimisation .....</b>	<b>50</b>
8.1. Problème d'optimisation .....	51
8.2. Méthodes d'optimisation .....	52
8.2.1. <i>Méthodes exactes</i> .....	53

8.2.2 Méthodes approchées .....	53
a. Méthodes à base de voisinage .....	54
b. Méthodes à base de population .....	54
c. Méthodes hybrides .....	54
<b>9. Algorithmes évolutionnaires .....</b>	<b>55</b>
9.1 Importance de la diversité .....	57
9.2 Exploitation et exploration .....	57
9.3. Pourquoi le recours à des métaheuristiques (AE) .....	59
9.4. Algorithme génétique (AG ou GA) .....	60
9.4.1. Les étapes de l'algorithme génétique .....	60
<b>10. Conclusions .....</b>	<b>63</b>

**Chapitre III : Modélisation et commande des systèmes chaotiques :  
discrets et continus**

<b>1. Introduction .....</b>	<b>64</b>
<b>2. Apprentissage d'un RNA .....</b>	<b>65</b>
<b>3. Modélisation des systèmes chaotiques par les RNA .....</b>	<b>67</b>
3.1. Modélisation de l'équation logistique par les RNA .....	68
3.2. Modélisation du système de Lorenz par RNA .....	70
3.3. Modélisation du système de Chen par RNA .....	73
<b>4. Problèmes de contrôle des systèmes chaotiques .....</b>	<b>76</b>
4.1. Les problèmes de stabilisation .....	78
4.2 Les problèmes du contrôle d'excitation .....	79
4.3. Les problèmes de synchronisation contrôlable .....	79
<b>5. Motivations du contrôle des systèmes chaotiques.....</b>	<b>81</b>
<b>6. Applications .....</b>	<b>81</b>
6.1. Contrôle des UPO de systèmes chaotiques inconnus par les RN .....	81
6.1.1. Modélisation des systèmes chaotiques inconnus par les RN .....	83
6.1.2. Conception du contrôleur neuronal .....	85
6.1.3. Résultats de simulations.....	87
a. Le système de Henon.....	87
b. Le système Duffing.....	89

6.2. Transmission sécurisée des données du saint Coran par les dynamiques chaotiques contrôlées .....	90
<b>7. Conclusion .....</b>	<b>91</b>

### **Chapitre IV : Conception d'un contrôleur intelligent NN-PBC**

<b>1. Introduction .....</b>	<b>92</b>
<b>2. Description des systèmes .....</b>	<b>94</b>
2.1. Les systèmes chaotiques multi-scroll.....	94
2.1.1. <i>Le circuit de Chua à n-scroll</i> .....	94
2.1.2. <i>Le système de Chen multi-scroll</i> .....	95
2.2. La structure du réseau de neurones .....	96
<b>3. Amélioration de la méthode de contrôle basé sur la prédiction et formulation de problème .....</b>	<b>97</b>
3.1 Principes de contrôle prédictif.....	97
3.2. Formulation de problème .....	98
<b>4. La conception du NN-PbC .....</b>	<b>99</b>
<b>5. Résultats expérimentales .....</b>	<b>102</b>
5.1. Modélisation du circuit de Chua de n-scroll.....	103
5.2. Modélisation du système multi-scroll de Chen .....	105
<b>6. Résultats de la simulation pour le contrôle.....</b>	<b>106</b>
6.1. Stabilisation du modèle RN du circuit de Chua multi-scroll.....	107
6.2. Stabilisation du modèle RN du système de Chen multi-scroll .....	111
<b>7. Conclusion .....</b>	<b>115</b>
 <b>Conclusions générales et Perspectives .....</b>	 <b>116</b>
 <b>Références Bibliographiques.....</b>	 <b>118</b>

---

## INTRODUCTION GENERALE

---

## Introduction générale

Il existe un comportement entre la régularité rigide et l'aspect aléatoire. Ce comportement s'appelle : *chaos*. On a longtemps supposé que c'était de l'aléatoire et la vaste théorie de la probabilité et des statistiques appliquée. En fait, c'est un champ nouveau d'investigation qui s'ouvrait en même temps qu'une nouvelle manière d'appréhender des effets parfois méconnus depuis longtemps.

La théorie du chaos est devenue indispensable pour la science et l'ingénierie à tous les niveaux de la recherche d'aujourd'hui. Les recherches actives les plus récentes incluent le contrôle du chaos et la synchronisation du chaos, entre autres, avec une tendance visible vers des applications du monde réel [1,2].

Après une phase d'excitation où les scientifiques cherchaient et trouvaient du chaos presque partout ; on s'est rendu compte que les outils de la théorie du chaos peuvent être appliqués pour la compréhension, la manipulation et le contrôle d'une variété de systèmes avec des applications pratiques intenses.

La motivation du contrôle du chaos vient du fait, que le chaos permet à un système d'explorer chacune de ses dynamiques : lorsqu'il est sous contrôle, il fournit une variété de propriétés passionnantes ce qui permet de donner des opportunités dans diverses applications technologiques [3]. En effet, on commence ces dernières années à réaliser que la conception traditionnelle d'engineering qui essaye toujours de supprimer les comportements irréguliers et, par conséquent, d'éliminer le chaos afin d'éviter la complexité peut engendrer beaucoup d'incidents. Ceci s'avère être quelque chose d'impraticable, peu économique, et physiquement très difficile si ce n'est pas impossible. Dans la tentative de contrôler le chaos, les chercheurs essaient d'apporter de nouvelles idées et techniques qui utilisent la nature elle-même du chaos afin de le contrôler.

Aujourd'hui, même 26 ans après son apparition pour la première fois, le contrôle du chaos reste un sujet d'actualité et ne cesse de se développer et une multitude de méthodes de contrôle sont proposées, se développent et s'appliquent sur les systèmes biologiques, mécaniques, circuits électroniques et divers autres systèmes.

La majorité des systèmes dynamiques utilisés dans la réalité sont des systèmes non linéaires. Ce type de système est généralement sujet aux variations et incertitudes

structurelles et non structurelles causées par l'environnement d'où le recours aux techniques de l'intelligence computationnelle pour palier à ce problème. Ces techniques sont connues par leur adaptation aux changements environnementaux.

Les systèmes intelligents conviennent à un raisonnement incertain ou approximatif. Ils nous permettent de représenter des expressions descriptives ou qualitatives particulières pour les systèmes ayant un modèle mathématique difficile à définir. Les réseaux de neurones artificiels (RNA) peuvent être utilisés pour modéliser un système dynamique à comportement chaotique complexe. Ils peuvent mener à une meilleure compréhension d'un cerveau humain et apprennent eux-mêmes à construire des modèles. Cet apprentissage peut être appliqué à la modélisation, la prédiction, et le contrôle des systèmes chaotiques.

Dans le monde réel, nous pouvons trouver beaucoup de problèmes et systèmes qui sont trop complexes ou incertains pour les représenter par des modèles mathématiques complets et précis. Et pourtant, nous avons toujours la nécessité de concevoir, optimiser, ou contrôler le comportement de tels systèmes. Comme exemple, les humains peuvent apprendre l'information par leur processus cognitif, prennent des décisions, et s'adaptent même à l'environnement dynamique sans employer les modèles mathématiques. La motivation pour concevoir les systèmes intelligents vient des efforts d'imiter un tel processus sans exiger les modèles mathématiques précis. Il y a trois composants importants pour développer les systèmes intelligents : acquisition et représentation de connaissance, inférences, et processus décisionnels.

Nous pouvons considérer trois types possibles de représentation de la connaissance ou de l'information décrivant les relations d'entrée-sortie ou de cause-effet des systèmes non linéaires. Ils sont les modèles analytiques quantitatifs, les données, et les règles heuristiques. Si aucun de ces types d'information ne peut fournir la connaissance pertinente complète et suffisante au sujet du comportement de système par lui-même, il sera nécessaire de combiner chacun des trois domaines d'information hétérogènes. Par conséquent, nous devons considérer différentes stratégies de modélisation pour capturer différents types d'information.

Pour la commande des systèmes chaotiques, les méthodes conventionnelles de l'automatique ont montré leurs limites en termes de stabilisation et de performances. Avec le développement considérable des calculateurs numériques, les automaticiens ont adopté de plus en plus de nouvelles approches telles que la commande prédictive et la commande robuste [2].



Notre contribution à travers le travail développé dans cette thèse concerne la proposition et le développement de nouvelles approches et méthodes afin de pouvoir modéliser, stabiliser, contrôler et conduire la trajectoire des systèmes chaotiques vers des trajectoires bien spécifiques. Ces trajectoires sont les orbites périodiques instables (UPO) du système chaotique.

Nous avons ainsi proposé plusieurs solutions, liées à des formes différentes des systèmes chaotiques. Ces solutions correspondent aux points suivants :

- **Implémentation de l'algorithme de la rétro-propagation pour l'approximation des fonctions** : L'approximation d'une fonction monotone par un réseau de neurones multicouche est assurée toujours quelles soient les conditions initiales en variant le nombre de neurones de la couche cachée à condition que ça ne provoque pas le sur-apprentissage.
- **Contrôle du chaos en utilisant le modèle de réseau de neurones basé sur le contrôle prédictif** : Le feedback et les mécanismes ne dépendent que des mesures locales pour assurer le suivi asymptotique d'une trajectoire de référence. La performance du contrôleur proposé est démontrée en utilisant des systèmes chaotiques typiques.
- **Transmission de données Sécurisée du Coran en utilisant des dynamiques Chaotiques contrôlée**: Le signal d'information extrait du Coran est codé en l'appliquant comme une entrée à un système chaotique. Avec un message injecté dans la dynamique chaotique, la complexité du chaos est trouvée à augmenter, et aussi, la qualité de la régulation n'est pas influencée par le codage des messages.
- **Contrôle des orbites périodiques instables (UPO) des systèmes chaotiques inconnus par les RNA** : Dans la phase de modélisation, le RNA est entraîné sur des systèmes chaotiques inconnus en utilisant des données entrée-sortie obtenus à partir des systèmes chaotiques sous-jacents inconnus, et un algorithme de calcul spécifique est utilisé pour l'optimisation des paramètres. Dans la phase de contrôle, le critère de stabilité  $L_2$  est utilisé, et qui forme la base du principe de conception principale.
- **Modélisation et contrôle des systèmes chaotiques multi-scroll par les RNA** : Dans des cas réels, le système est souvent représenté par un ensemble de données d'entrées-sorties. Dans une telle situation, on a recours à des méthodes de modélisation. Les réseaux de neurones artificiels et la logique floue sont des candidats très intéressants.

Sur le plan théorique, nous présentons la modélisation et le contrôle des systèmes chaotiques par réseaux de neurones dans un cadre aussi général que possible, en les plaçant dans la perspective de l'Automatique moderne, non-linéaire et complexe en particulier. Du point de vue de la modélisation ; une méthodologie d'apprentissage spécifique fournit des prédicteurs neuronaux qui sont des réalisations des prédicteurs théoriques. Nous proposons ensuite une famille de systèmes de commande neuronaux, dont nous étudions les propriétés et les liens avec les systèmes de commande classique, linéaire ou non, en insistant notamment sur la robustesse.

Sur le plan pratique, nous illustrons notre démarche par des applications numériques ou bien des simulations sur MATLAB.

Le travail est présenté selon le plan suivant :

Au premier chapitre, nous rappelons les principales définitions mathématiques relatives aux systèmes chaotiques, notamment des notions préliminaires sur les dynamiques du comportement chaotique, nous présentons quelques exemples des systèmes chaotiques qui sont des systèmes repères pour la modélisation et la validation des techniques et des algorithmes proposés.

Le chapitre II expose les propriétés mathématiques des réseaux de neurones, et les principes généraux de leur apprentissage pour la modélisation et la commande. Il présente les différentes structures de modèles neuronaux que nous utilisons. Aussi, nous allons présenter des algorithmes évolutionnaires et leurs applications dans l'optimisation.

Le chapitre III montre la mise en œuvre des principes exposés pour la modélisation et la commande des systèmes chaotiques. Lorsque le système chaotique est représenté seulement par un ensemble de données d'entrées-sorties, dans ce cas, l'équation mathématique du système chaotique est inconnue et doit être modélisée pour activer le contrôle, cela nécessite le recours à des approches intelligentes telle que les réseaux de neurones. Ce qui fera l'objet de ce chapitre.

Le chapitre IV est consacré à la conception d'un contrôleur neuronal intelligent à base de prédiction pour les systèmes chaotiques multi-scroll. Aussi, la comparaison entre les systèmes d'apprentissages, et la favorisation du modèle hybride.

Ce travail sera finalisé par une conclusion et synthèse des différentes phases d'étude et de développement. Elle permettra de qualifier les modèles théoriques mis en œuvre et d'envisager d'autres perspectives plus intéressantes.

---

# Chapitre I

Introduction aux Systèmes Dynamiques et Chaos

---

---

## 1. Introduction

Depuis longtemps, le chaos était synonyme de désordre et de confusion. Il s'opposait à l'ordre et devait être évité. La science était caractérisée par le déterminisme, la prévisibilité et la réversibilité. *Poincaré* fut l'un des premiers à entrevoir la théorie du chaos [4]. Il découvrit la notion de sensibilité aux conditions initiales à travers le problème de l'interaction de trois corps célestes.

Selon le philosophe *Daniel Parrochia* [5], la théorie du chaos constitue une des trois grandes révolutions scientifiques du 20<sup>ème</sup> siècle et correspond à un changement de paradigme comparable à ceux qu'entraînèrent la théorie de la relativité et la mécanique quantique. Ce siècle a vu s'écrouler l'un après l'autre les murs de certitudes qui entouraient la forteresse de la physique newtonienne. *Einstein* avec sa théorie de la Relativité, a éliminé en 1905 l'illusion newtonienne d'un espace et d'un temps absolus. Dans les années 1920 à 1930, la mécanique quantique a détruit la certitude de tout pouvoir mesurer aussi précisément que possible. Le chaos, lui a éliminé l'*utopie Laplacienne* d'une prédictibilité déterministe.

Très succinctement, la théorie du chaos a pour objet l'étude des phénomènes non linéaires régis par des lois simples et déterministes dont le comportement sous certaines conditions, deviennent imprédictibles. En particulier, cette théorie se constitue, depuis les années 1970, à partir d'une triple confrontation :

- La théorie mathématique des systèmes dynamiques.
- L'étude du phénomène non linéaire, le désordre, la turbulence dans la nature et/ou dans le monde technologique.
- La technologie et le développement de l'ordinateur.

A la fin du 19<sup>ème</sup> siècle, *Henri Poincaré* réussit à mettre en évidence la possibilité de comportements irréguliers dans les systèmes déterministes. C'est *Edward Lorenz*, un météorologue américain qui fut le premier à comprendre et à déterminer un modèle mathématique du chaos, mais comme conclusion de l'histoire de naissance de la théorie du chaos, elle est le résultat d'une confrontation entre l'histoire de longue durée, qui trouve ses racines au dix-neuvième siècle dans les travaux d'*Henri Poincaré* [6], et une période de reconfiguration, constituée par les travaux séminaires d'*Edward Lorenz* [7], *Stephen Smale* [8], *David Ruelle* et *Floris Takens* [9,10].

Dans un système déterministe, des conditions initiales identiques conduisent à des évolutions identiques, pour un système chaotique qui est un système dynamique déterministe possédant un comportement imprévisible à long terme, cette imprévisibilité est due à la Sensibilité aux Conditions Initiales (SCI), particularité des systèmes chaotiques.

Le terme chaos, dans l'ancienne philosophie signifiait l'état de désordre dans la matière non formée supposée existée avant l'univers ordonné [11,12]. Comme pour beaucoup de limites en science, il n'y avait aucune définition standard du chaos. Néanmoins, les dispositifs typiques du chaos incluent :

- **La non-linéarité.** Si le système est linéaire, il ne peut pas être chaotique.
- **Le déterminisme.** Un système chaotique a des règles fondamentales déterministes plutôt que probabilistes.
- **La sensibilité aux conditions initiales (SCI).** De très petit changement sur l'état initial peuvent mener à un comportement radicalement différent dans son état final.
- **L'imprévisibilité.** En raison de la sensibilité aux conditions initiales qui peuvent être connues seulement à un degré fini de précision.
- **L'irrégularité.** Ordre caché comprenant un nombre infini de modèles périodiques instables.

La dynamique chaotique a été découverte dans de nombreux différents systèmes dynamiques et plusieurs applications du chaos dans l'engineering où les modes chaotiques peuvent paraître quelque fois comme nuisibles où l'on doit le contrôler ou le réduire et des classes entières de problèmes de contrôle qui sont d'importance pratique sont apparus, et la combinaison de la théorie et le contrôle du chaos ajoute un sens paradoxal et un intérêt immense au sujet.

Les problèmes du chaos et du contrôle du chaos ont fait l'objet des études intenses durant les deux dernières décennies. Le terme contrôle du chaos est principalement utilisé pour désigner le domaine d'étude :

- ❖ incluant la théorie du contrôle et la théorie des systèmes dynamiques,
- ❖ étudiant les méthodes de contrôle des systèmes déterministes présentant un comportement chaotique non régulier [13].

L'objectif de ce chapitre est de donner quelques notions élémentaires sur les systèmes dynamiques afin de mieux appréhender ce qu'est le chaos : ses apparitions dans un système et la manière de le quantifier. Ensuite, des exemples les plus trouvés dans la littérature, vont être évoqué. A la fin de ce chapitre, les attracteurs *multi-scroll* (multi-spirales) ont été brièvement présentés.

## 2. Les systèmes dynamiques

Un système dynamique est une structure qui évolue au cours du temps de façon à la fois :

- *Causale*, où son avenir ne dépend que de phénomènes du passé ou du présent
- *Déterministe*, c'est-à-dire qu'à partir d'une « condition initiale » donnée à l'instant « présent » va correspondre à chaque instant ultérieur un et un seul état « futur » possible.

L'évolution déterministe du système dynamique peut alors se modéliser de deux façons distinctes :

- *Une évolution continue dans le temps*, représentée par des équations différentielles.
- *Une évolution discrète dans le temps*, l'étude théorique de ces modèles discrets est fondamentale, car elle permet de mettre en évidence des résultats importants, qui se généralisent souvent aux évolutions dynamiques continues. Elle est représentée par le modèle général des équations aux différences finies.

Du point de vue mathématique la notion générale de système dynamique est défini à son tour à partir d'un ensemble de variables qui forment le vecteur d'état  $x = \{x_i \rightarrow \mathfrak{R}\}$ ,  $i = 1 \dots n$  où  $n$  représente la dimension du vecteur.

### 2.1. Temps continu

Un système dynamique en temps continu est décrit par un système d'équations différentielles, soit :

$$x(t) = F(x(t), t) \quad (\text{I.1})$$

où  $F: \mathfrak{R}^n \times \mathfrak{R}^+ \rightarrow \mathfrak{R}^n$  désigne la dynamique du système.

Si on associe à cette dynamique un état initial  $x_0 = x(t_0)$ , pour chaque couple choisi  $(x_0, t_0)$  on peut identifier une solution unique  $\phi(\cdot; x_0, t_0): \mathfrak{R}^+ \rightarrow \mathfrak{R}^n$  telle que :

$$\phi_F(t_0; x_0, t_0) = x_0 \text{ et } \dot{\phi}_F(t; x_0, t_0) = F(\phi_F(t; x_0, t_0), t) \quad (\text{I.2})$$

Cette solution unique déterminée à l'aide des équations (eq. I.2), et qui fournit l'ensemble d'états successifs occupés par le système à chaque instant  $t$ , s'appelle généralement la trajectoire du système dynamique [14].

Même que les oscillations complexes comme la relaxation, pouvait être décrite par un simple modèle non linéaire dépendant des conditions initiales "systèmes avec plusieurs cycles limites", les modèles des oscillations linéaires et non linéaires satisfaisaient énormément les besoins des chercheurs pour plusieurs décennies. Il a été admis que ces modèles non linéaires, pouvaient décrire tous types d'oscillations et cette conviction était supportée par des fondements mathématiques, telle que le théorème de *Poincaré-Bendixson* qui affirmait que les points d'équilibres et les cycles limites étaient les seules type possible des états limites stables dans les systèmes continus du second ordre, cependant au milieu du siècle dernier, quelques mathématiciens établissaient que ce n'était pas le cas pour les systèmes du troisième ordre, qui présentaient des comportements plus complexes comme des oscillations non périodiques limités [13].

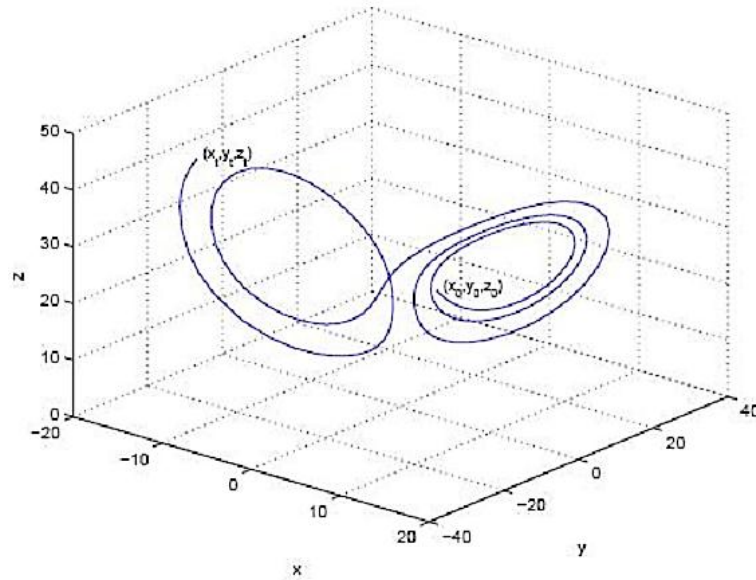
En 1963, *E. Lorenz* révolutionnait le monde et démontrait la nature qualitative de l'atmosphère turbulent, qui obéissait aux équations différentielles partielles (EDP) complexes de *Navier-Stokes* et le représentait par un simple modèle non linéaire du troisième ordre appelé plus tard les *équations de Lorenz* [7]:

$$\begin{cases} \dot{x} = \sigma (y - x) \\ \dot{y} = rx - y - xz \\ \dot{z} = -bz + xy \end{cases} \quad (\text{I.3})$$

avec  $x, y, z \in \mathfrak{R}$  sont les variables des états d'entrées et  $\sigma, r, b$  sont les paramètres de contrôle du système.

On considère l'exemple du célèbre système de Lorenz donné par (eq. I.3), pour des valeurs :  $\sigma = 10, r = 97, b = \frac{2}{3}$ , et avec la condition initiale  $(x_0, y_0, z_0 = 2, 5, 20)$ , les solutions du système de Lorenz semblaient à des oscillations non périodiques et les trajectoires dans l'espace de phase approchaient des ensembles limites appelés *attracteurs* caractérisés par une forme étrange (cf. Figure. I.1).

On observe, que la dynamique du système de Lorenz donnée par les équations (eq. I.3) est indépendante de l'instant  $t$  considéré. Généralement ce type de système est qualifié d'autonome. La dynamique, dans ce cas particulier, a la forme suivante :



**Figure. I.1** Exemple de trajectoire pour le système de Lorenz.

L'intention des physiciens et des mathématiciens, et plus tard des ingénieurs, a été attirée pour ce modèle par les travaux de *D. Ruelle* et *F. Takens* qui appelaient pour la première fois attracteurs "étrange" et aussi par les travaux de *Li* et *Yorke* qui introduisaient le terme chaos pour désigner le phénomène non régulier dans les systèmes déterministes, notant que les fondements mathématiques apparus pour étudier le phénomène chaotique sont mis dès 1960-1970. Durant ce temps, le comportement chaotique a été découvert dans plusieurs systèmes mécaniques, lasers, physiques, chimiques, biologiques et médicales, circuits électroniques et dans beaucoup d'autres.

## 2.2. Temps discret

Le système dynamique, dans ce cas, est représenté par des équations aux différences finies comme il a été déjà précisé, avec le modèle général suivant :

$$x(k+1) = G(x(k), k) \quad (\text{I.4})$$

où  $G: \mathbb{R}^n \times \mathbb{Z}^+ \rightarrow \mathbb{R}^n$  désigne la dynamique du système en temps discret.



De même qu'en temps continu, si on associe à cette dynamique un état initial  $x_0 = x(k_0)$ , pour chaque couple choisi  $(x_0, k_0)$  on peut identifier une solution unique  $\phi_G(\cdot; x_0, k_0) : Z^+ \rightarrow \mathfrak{R}^n$  telle que :

$$\phi_G(k_0; x_0, k_0) = x_0 \text{ et } \phi_G(k+1; x_0, k_0) = G(\phi_G(k; x_0, k_0), k) \quad (\text{I.5})$$

En temps discret, on définit aussi le système autonome comme une dynamique qui ne dépend pas de l'instant  $k$ .

$$x(k+1) = G(x(k)) \quad (\text{I.6})$$

### 2.3. Comportement des systèmes dynamiques

A partir d'un état initial  $x_0$  et après un régime transitoire, la trajectoire d'un système dynamique atteint une région limitée de l'espace des phases. Ce comportement asymptotique, obtenu quand  $t$  et  $k$  tendent vers l'infini, est une des caractéristiques les plus importantes à étudier pour tout système dynamique.

Dans le cas d'un système linéaire, la solution asymptotique est indépendante de la condition initiale et unique. En présence de non-linéarités, le cas de notre étude, il existe donc une plus grande variété de régimes permanents, parmi lesquelles on trouve, par ordre de complexité: points d'équilibre, solutions périodiques, solutions quasi-périodiques et chaos, respectivement. Il faut préciser cette fois que, le comportement développé par un système dynamique particulier est fortement dépendant de la condition initiale choisie [14].

Dans ce qui suit, on va étudier les comportements évoqués ci-dessus, en utilisant une dynamique typique très connue dans la théorie des systèmes non linéaires. Il s'agit de l'équation logistique "cette équation a joué un rôle important en modélisation des systèmes écologiques" définie comme suit :

$$x_{k+1} = G(x_k) = r x_k (1 - x_k) \quad (\text{I.7})$$

où  $k = 0, 1, \dots$  dénote le temps discret,  $x$  la variable dynamique et  $r$  un paramètre réel.

Sur la figure I.2 on montre la dynamique propre à l'équation logistique ainsi que certains modes asymptotiques particuliers. Le mécanisme de construction d'une séquence est tout d'abord montré sous la forme d'un diagramme en toile. Cette méthode permet la génération de la séquence choisie graphiquement en utilisant la projection des états successifs par rapport à la diagonale principale (cf. Figure. I.2 a) (pour  $r = 3.9$ ).

La figure (I.2 b) présente le diagramme de *bifurcation* qui montre la distribution des états limites pour différents choix du paramètre  $r$ . On appelle cette représentation *diagramme de bifurcation* car le comportement asymptotique subit pour des valeurs du paramètre  $r$  bien déterminées, une bifurcation de l'ensemble des états limites. Dans le cas continu la bifurcation se manifeste comme une multiplication des trajectoires possibles. Pour cette représentation on a choisi pour chaque valeur  $r \in [1,4]$  une séquence de 500 échantillons avec une période de transition de 50 échantillons.

Par la suite, pour chaque type de régime permanent on a quatre cas [14] :

### 2.3.1. Point d'équilibre

Dans ce cas, la solution asymptotique est représentée par un point, sa valeur étant déterminée en fonction de la condition initiale choisie. Ainsi, on peut retrouver plusieurs points d'équilibres pour des conditions initiales différentes. De même, ces points d'équilibres peuvent être stables ou instables suivant que les trajectoires voisines convergent ou divergent entre elles.

Dans le cas de la dynamique *logistique* (équ. I.7), on observe que pour toute valeur  $r \in [1,3]$ , le régime permanent est formé par un point limite stable, sa valeur étant dépendante du choix de paramètre  $r$ . La figure (I.2 c) nous donne un aperçu d'une telle trajectoire pour  $r = 2$ . Ainsi on observe qu'après une période de transition relativement courte, la séquence se stabilise autour du point fixe  $x_\infty = 0.5$ .

### 2.3.2. Régime périodique

Le régime permanent périodique correspond à une trajectoire dont les répliques d'une portion élémentaire sont espacées à des intervalles  $n \cdot T$ , avec  $n \in \mathbb{N}^+$  et  $T$  désigne la période.

Pour la fonction logistique, on a choisi deux exemples ;  $r = 3.2$  et  $r = 3.55$ . Pour le premier cas, ce choix nous garantit que l'ensemble des états limites est formé par deux points, et la période correspond à deux échantillons (Figure. I.2 d). La deuxième solution nous permet d'augmenter la dimension de l'ensemble des états limites et la période de répétition à 8 (cf. Figure. I.2 e).

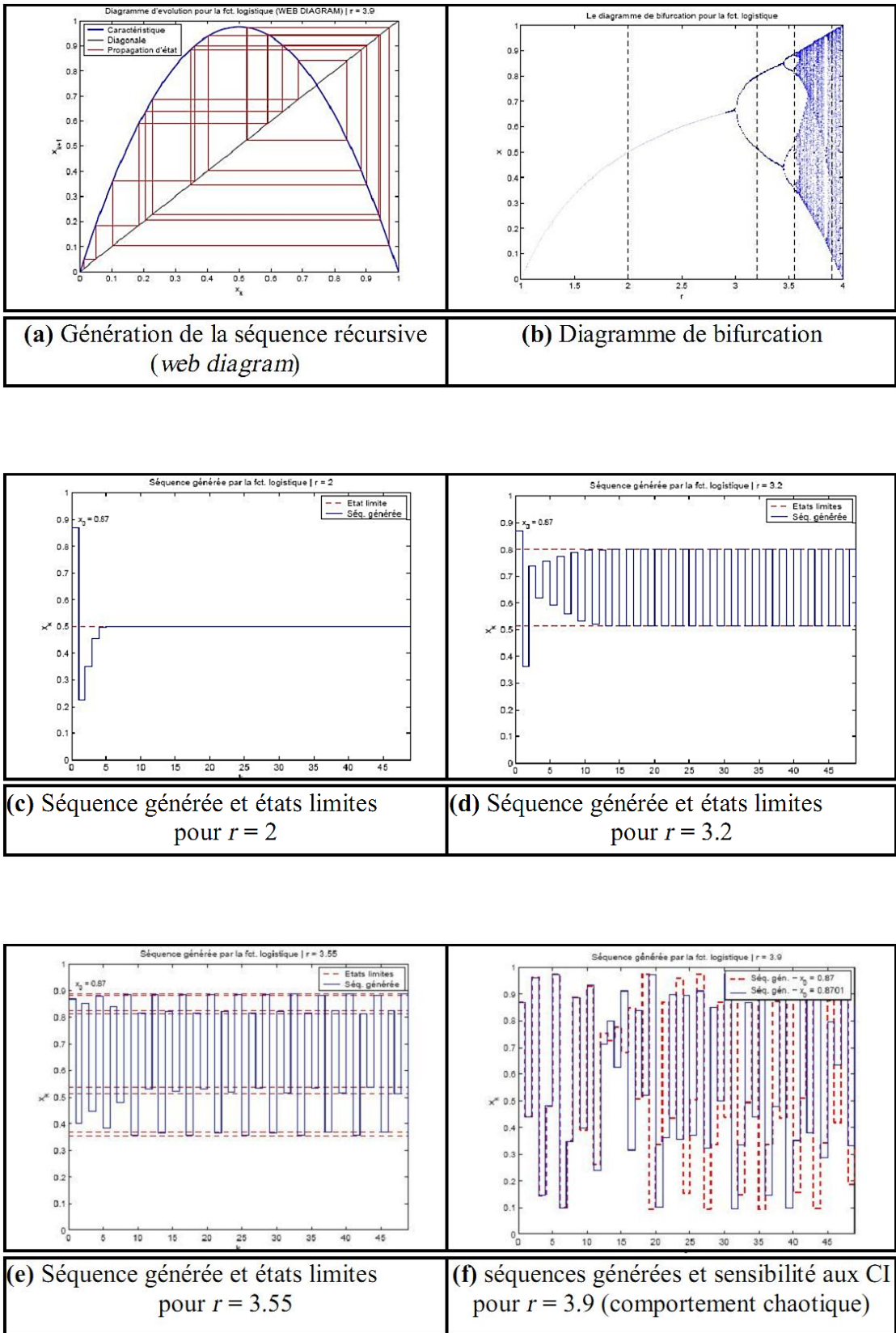


Figure. I.2 Étude du comportement dynamique pour la fonction logistique (eq. I.7).

### 2.3.3. Régime quasi-périodique

Il correspond à une somme de solutions périodiques, dont le rapport des périodes est un nombre irrationnel. Un régime quasi-périodique peut être représenté dans l'espace d'état par un tore.

### 2.3.4. Régime chaotique

Le régime chaotique est par définition, tout régime permanent qui n'appartient à aucune des classes présentées antérieurement. Une telle solution a une trajectoire asymptotique bornée avec une extrême sensibilité aux conditions initiales (SCI). Ainsi deux trajectoires générées à partir de deux conditions initiales (CI) très proches, vont diverger très vite l'une de l'autre. Cette sensibilité par rapport aux CI traduit aussi le comportement, en apparence stochastique, des générateurs chaotiques, de telle sorte qu'une prévision à long terme du comportement du système est impossible.

Dans la figure (I.2 f), pour deux CI espacées par une valeur de  $10^{-4}$ , on peut observer que juste après quelques itérations les deux trajectoires divergent et deviennent non corrélées.

Généralement, l'ensemble des solutions asymptotiques stables décrites ci-dessus est qualifié d'attracteur. En parallèle, avec la définition de l'attracteur apparaît la notion de *bassin d'attraction*, défini comme *la région de l'espace d'état formée par l'ensemble des CI à partir desquelles l'attracteur sera atteint.*

## 2.4. Evaluation du comportement dynamique

La présence d'un comportement chaotique pour un système dynamique quelconque, peut être déterminée par élimination de comportements introduits auparavant : s'il n'est pas un point fixe, périodique ou quasi-périodique on conclut qu'il est chaotique. Mais dans le cas où la dynamique employée pour générer la séquence observée n'est pas connue et si en plus un bruit affecte les observations une telle méthode n'est pas envisageable. Par conséquent, la communauté scientifique a proposé des solutions avec une approche statistique du problème comme le calcul de la *dimension de corrélation*, l'*entropie de Kolmogorov* ou les *exposants de Lyapunov*.

La dimension de corrélation est un outil qui offre la possibilité de déterminer la dimension de l'attracteur reconstruit à partir d'une série temporelle observée, tandis que

l'entropie ou les exposants de Lyapunov sont employés pour l'évaluation de l'instabilité propre au phénomène chaotique. Dans la pratique, ces exposants se sont imposés comme des outils performants, même dans le cas de séries temporelles courtes, avec un coût de calcul relativement réduit par rapport à la dimension de corrélation ou l'entropie de Kolmogorov [14].

Dans cette thèse, on se limite à la description des exposants de Lyapunov, la solution la plus pertinente dans le contexte des systèmes à dimension d'état réduite. Les exposants de Lyapunov se définissent comme une mesure invariante propre à un système dynamique qui caractérise la séparation exponentielle en temps de deux trajectoires proches. Cette propriété est aussi qualifiée de SCI, mais elle se réfère généralement à la divergence de trajectoires à n'importe quel instant temporel. Ainsi dans le cas d'un attracteur chaotique, deux trajectoires initialement voisines vont diverger à une vitesse exponentielle quantifiée par l'exposant de Lyapunov.

### 3. Concepts mathématiques et définitions

Les nouvelles méthodes analytiques et numériques développées pour les systèmes, démontraient que le chaos n'est qu'un type exceptionnel de comportement des systèmes non linéaires, grossièrement parler du comportement chaotique survenait toute fois :

- ❖ que les trajectoires des systèmes sont globalement bornées et localement instables, dans les systèmes chaotiques, une petite divergence initiale et arbitraire des trajectoires ne reste pas insignifiante mais croit exponentiellement.
- ❖ Le spectre de fréquence des trajectoires chaotiques est continu.

Il faut noter qu'en pratique, il est impossible de distinguer à l'œil le processus chaotique du processus périodique ou quasi-périodique.

Considérons le système dynamique continu dans le temps suivant [4]:

$$\begin{cases} \dot{x} = F(x) \\ y = h(x) \\ x(t_0) = x_0 \end{cases} \quad (I.8)$$

où :  $x = x(t) \in \mathbb{R}^n$  : est le vecteur d'état du système,  $0 \leq t \leq \infty$ , et  $\dot{x} = \frac{dx}{dt}$ .

**Définition I.1** L'ensemble fermé  $\Omega \in \mathbb{R}^n$  est appelé un **attracteur** pour le système  $\dot{x} = F(x, u)$ , où :  $x = x(t)$  est le vecteur des variables d'état de dimension  $n$ ,  $u = u(t)$  est le

vecteur des entrées (les commandes) de dimension  $m$  et  $F(x, u)$  est le vecteur fonction qui est supposé continu :

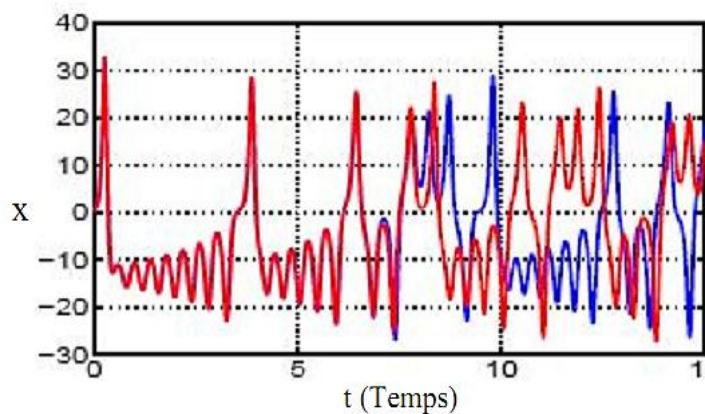
- a- S'il existe un sous ensemble ouvert,  $\Omega_0 \subset \Omega$  tel que toutes les trajectoires  $x(t)$  du système (éq. I.1), commençant dans  $\Omega_0$  sont définies pour tout  $t \geq 0$  et tendent vers  $\Omega$  quand  $t \rightarrow \infty$ , c.à.d : Si  $x_0 \in \Omega_0$  et  $\|x - y\|$  représente la distance entre le point  $x$  et l'ensemble limite  $\Omega$ , alors  $\|x - y\| \rightarrow 0$  quand  $t \rightarrow \infty$ .
- b- Aucun autre sous ensemble de  $\Omega$  a cette propriété.

Autrement dit : On appelle attracteur un ensemble de points vers lequel converge la trajectoire de l'espace des phases. Pratiquement, à partir de quelques itérations, on considère que l'ensemble des points de l'espace des phases décrit l'attracteur.

**Définition I.2** Un système dynamique est appelé chaotique s'il a au moins un attracteur chaotique.

**Définition I.3** L'application  $F: J \rightarrow J$  possède une sensibilité aux conditions initiales s'il existe  $\delta > 0$  tel que, pour un certain  $x \in J$  et un certain voisinage  $V \subset J$  de  $x$ , il existe  $y \in V$  tel que  $\|F^n(x) - F^n(y)\| > \delta$ .

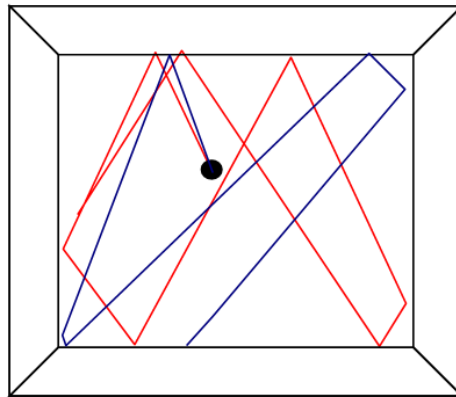
L'instabilité de Lyapunov caractérise la propriété principale des oscillations chaotiques appelée super sensibilité ou dépendance sensible aux conditions initiales, c.à.d. toutes deux trajectoires fermées arbitraires proches s'éloignent l'une de l'autre nécessairement à une distance finie. Dans un premier temps, les deux systèmes évoluent de la même manière, mais, très vite, leur comportement devient différent pour n'avoir plus grand chose à voir (cf. Figure. I.3).



**Figure. I.3** Evolution dans le temps pour deux conditions initiales très proches.

Pour mieux comprendre la sensibilité aux conditions initiales (SCI), prenons l'exemple des trajectoires de deux boules de billard qui commencent très proche l'une de l'autre. Les deux boules se séparent exponentiellement avec le temps et après un nombre de

rebondissements fini (et souvent très petit), leur séparation  $\delta x(t)$  atteint l'amplitude maximale  $L$  qui caractérise le système global (table de billard) (cf. Figure. I.4).



**Figure. I.4** La SCI illustrée par les trajectoires des boules de billard [15].

Cette sensibilité aux conditions initiales peut être quantifiée par :

$$|\delta x(t)| \approx e^{\lambda t} |\delta x(0)| \tag{I.9}$$

où  $\lambda$ , le taux moyen de séparation des trajectoires du système est appelé « exposant de Lyapunov ».

Dans ce cas  $\lambda$  est positive et par conséquent, pour n'importe quelle précision finie  $\delta x$  des données initiales, la dynamique du système n'est prédictible qu'à un temps fini appelé *temps de Lyapunov*  $T_{Lyap}$  [15].

$$T_{Lyap} \approx -\frac{1}{\lambda} \ln(\delta x/L) \tag{I.10}$$

**Définition I.4** *Un attracteur est appelé chaotique s'il est borné et toute trajectoire qui commence dedans est une trajectoire instable de Lyapunov.*

Il y a d'autres définitions de l'attracteur chaotique et du chaos. Par exemple, la définition de l'attracteur chaotique qui souvent inclut des exigences supplémentaires telles que l'existence des trajectoires ou une famille de trajectoires périodiques. La notion d'attracteur chaotique coïncide souvent avec celui de l'attracteur étrange introduite en 1971 par *Ruelle* et *Takens* comme un ensemble accessible et nommé plus tard un *ensemble fractal*.

**Définition I.5 (Point limite positif)** *Soit  $\phi_t(x_0)$  la solution de l'équation (I.8) qui passe par  $x_0$  quand  $t = t_0$ . Un point  $y \in \mathfrak{R}^n$  est dit appartenant au point limite positif d'un point  $x \in \mathfrak{R}^n$ , si pour tout voisinage  $U$  de  $y$  on a :  $\phi_t(x_0)$  tend vers  $U$  quand  $t$  tend vers l'infini.*

**Définition I.6** (*Ensemble limite positif*) L'ensemble  $L(x)$  de tous les points limites positifs de  $x$  est appelé : l'ensemble limite positif de  $x$ .

**Définition I.7** (*Ensemble limite attractif*) Un ensemble limite  $L$  est dit attractif, s'il existe un voisinage ouvert  $U$  de  $L$  tel que :  $L(x) = L$ , pour tous  $x \in U$ .

**Définition I.8** (*Bassin d'attraction*) On appelle bassin d'attraction d'un ensemble  $A \subset \mathbb{R}^n$ , l'ensemble défini par :  $B(A) = \{x \in \mathbb{R}^n / A \text{ tel que } : L(x) \subset A\}$ .

**Définition I.9** (*Ensemble limite positif non-stable*) On dit qu'un ensemble limite positif  $L$  est non-stable, s'il existe au moins une trajectoire qui n'appartient pas à  $L$ , qui est attiré par  $L$ , et il existe au moins une trajectoire dans le voisinage de  $L$  qui n'est pas attirée par  $L$ .

Cette série de définition nous permet de donner des définitions assez claires sur le phénomène du chaos.

En générale il n'y a pas de définition rigoureuse du chaos, car ce phénomène est plus une notion philosophique qu'une notion scientifique. On peut observer le phénomène du chaos dans plusieurs domaines, mais comment le formaliser ?. La réponse est négative car jusqu'à l'heure actuelle, il n'existe pas une théorie générale qui donne une explication ou une caractérisation finale de ce phénomène. Tout ce qu'il est possible de dire est qu'il existe plusieurs critères physiques qui permettent de confirmer qu'un système est chaotique.

Notons qu'il existe quelques définitions du chaos, mais elles restent restrictives, la plus efficace du point de vue pratique est celle donnée dans [16] : *Le chaos se produit quand le comportement du système, n'est pas un point d'équilibre, n'est pas périodique, n'est pas quasi-périodique.*

#### 4. La bifurcation (Route vers le chaos)

A ce jour, on a distingué au moins trois routes ou transitions dans lesquelles un système non linéaire peut devenir chaotique si un paramètre de contrôle externe est varié. Toutes ces routes peuvent être vérifiées expérimentalement et montrent un comportement universel fascinant.

La route vers le chaos la plus récente a été trouvée par *Grosmann et Thomae* (1977), *Feigenbaum* (1978) et *Coulet et Tresser* (1978). Ils ont considéré une simple équation de différence utilisée par les biologistes pour décrire la dynamique d'une population dont le nombre varie avec le temps. Ils ont trouvé que, en variant un paramètre externe, l'état du système oscille entre des valeurs stables (points fixes) dont le nombre augmente pour



certaines valeurs distinctes du paramètre externe. Ceci continue jusqu'au moment où le nombre de points fixes devient infini à une valeur finie et précise du paramètre externe, c'est à ce moment-là que le système devient chaotique.

*Feigenbaum* a montré dans un travail remarquable que ces résultats ne sont pas restreints à ce modèle spécial, mais sont universels, et peuvent être vérifiés pour une grande variété de systèmes biologiques, chimiques ou physiques.

Une deuxième transition vers le chaos, appelé la route d'intermittence, a été trouvée par *Manneville et Pomeau* (1979). Intermittence veut dire qu'un signal qui a un comportement régulier (laminaire) est interrompu par des périodes de comportement irrégulier statistiquement distribué (des éclats intermittents). Le nombre moyen de ces éclats augmentent en variant un paramètre de contrôle externe jusqu'à ce que le système devienne complètement chaotique.

La troisième possibilité a été trouvée par *Ruelle et Takens* (1971) et *Newhouse* (1978). Ils ont trouvé une transition vers la turbulence qui est différente de celle proposée par *Landau* (1944, 1959), qui a considéré les turbulences dans le temps comme la limite d'une séquence infinie d'instabilité (*Bifurcation de Hopf*), chacune créant une nouvelle fréquence de base. Cependant, *Ruelle, Takens, et Newhouse* ont montré qu'après deux instabilités seulement, la trajectoire rejoint au niveau de la troisième étape un attracteur chaotique et ainsi le système devient turbulent [17].

Comme exemple de la bifurcation, on prend l'équation de récurrence du premier ordre :

$$x_n = f_a(x_{n-1}) = x_{n-1}^2 + a \quad (\text{I.11})$$

Cette famille de fonctions est appelée les fonctions quadratiques. Le paramètre externe à varier est la constante  $a$ . L'équation admet un simple point d'équilibre à  $x = 0$  quand  $a = 0$ . Pour  $a > 0$  cette équation n'admet aucun point d'équilibre, car  $f_a(x) > 0$  quelque soit  $x$ , cependant pour  $a < 0$  cette équation admet deux états d'équilibres. Ainsi, une bifurcation a lieu quand le paramètre  $a$  varié en passant par la valeur 0. La figure I.5 montre clairement ce phénomène, cette figure a été réalisée sous Scilab où les 5000 itérations sur la trajectoire de la fonction  $f_a$  ont été calculées en partant du point initial  $x_0 = 0$ . Le même calcul est répété pour plusieurs valeurs de  $a$  dans l'intervalle  $[0,-2]$ , on remarque les différents points de bifurcation et finalement la transition vers le chaos [18].

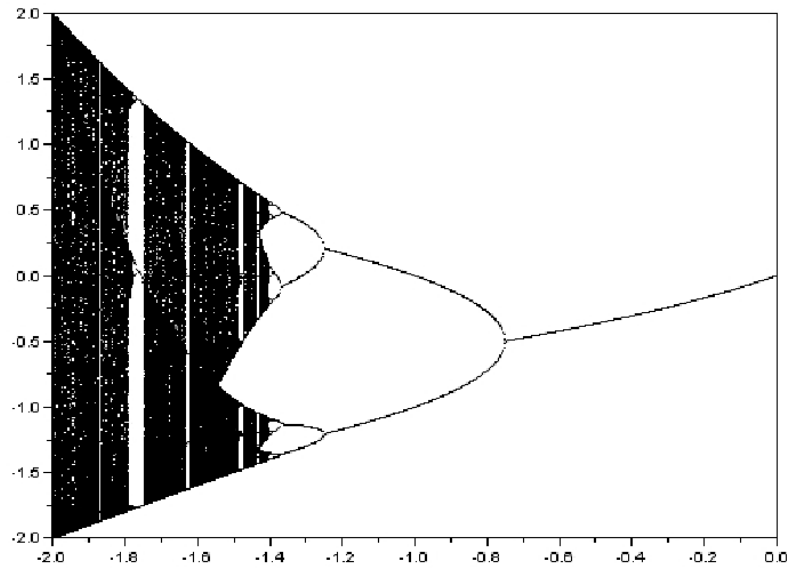


Figure. I.5 Bifurcation de la fonction quadratique (de droite à gauche) [18].

### 5. Les exposants de Lyapunov

L'évolution chaotique est difficile à appréhender car la divergence des trajectoires sur l'attracteur est rapide. Pour cette raison on essaie si c'est possible de mesurer sinon d'estimer la vitesse de divergence ou de convergence. Cette vitesse est donnée par l'exposant de Lyapunov qui caractérise le taux de séparation de deux trajectoires très proches [19,20] (cf. Figure. I.6).

Soit  $f : \mathfrak{R} \rightarrow \mathfrak{R}$  une fonction de classe  $C^1$ . Pour chaque point  $x_0$ , on définit un exposant de Lyapunov  $\lambda(x_0)$  comme suit :

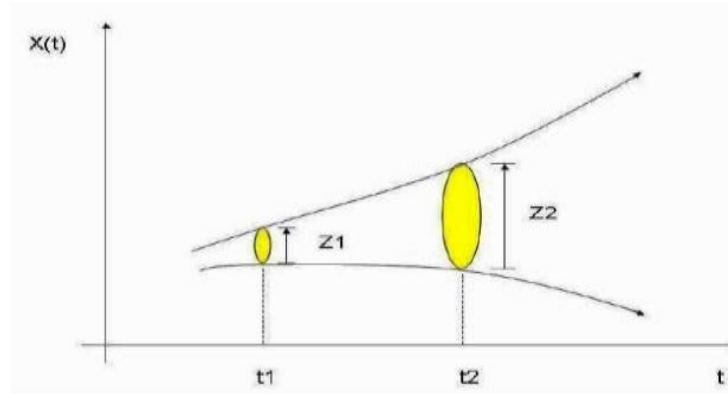
$$\lambda(x_0) = \lim_{n \rightarrow \infty} \sup \frac{1}{n} \log(|(f^n)'(x_0)|) = \lim_{n \rightarrow \infty} \sup \frac{1}{n} \sum_{j=0}^{n-1} \log(|f'(x_j)|) \quad (I.12)$$

avec  $x_j = f^j(x_0)$

Donc deux trajectoires dans le plan de phase initialement séparées par un taux  $Z_1$ , divergent après un temps  $\Delta t = t_2 - t_1$  vers  $Z_2$ , tel que :

$$|Z_2| \approx e^{\lambda \Delta t} |Z_1| \quad (I.13)$$

Où  $\lambda$  est l'exposant de Lyapunov



**Figure. I.6** Divergence de deux trajectoires dans le plan de phase [20].

Les exposants de Lyapunov sont une généralisation des valeurs propres pour le point fixe et des multipliers caractéristiques pour les solutions périodiques.

Pour un attracteur non chaotique, les exposants de Lyapunov sont tous inférieurs ou égaux à zéro et leur somme est négative. Un attracteur étrange possèdera toujours au moins trois exposants de Lyapunov, dont un au moins doit être positif (voir le tableau ci-dessous).

Le tableau suivant représente la classification des régimes permanents en fonction du spectre Lyapunov :

**Tableau I.1** : Classification des régimes permanents en fonction du spectre Lyapunov.

Régime permanent	Attracteur	Spectre	Exposants de Lyapunov
Point d'équilibre	Point	Composante continue	$0 > \lambda_1 \geq \dots \geq \lambda_n$
Périodique	Courbe fermée	Fréquence Fondamentale + harmoniques entières	$\lambda_1 = 0$ $0 > \lambda_2 \geq \dots \geq \lambda_n$
Quasi-périodique	tore	Composantes fréquentielles en rapport irrationnel	$\lambda_1 = \dots = \lambda_i = 0$ $0 > \lambda_{i+1} \geq \dots \geq \lambda_n$
Chaotique	fractale	Spectre large	$\lambda_1 > 0$ $0 > \lambda_2 \geq \dots \geq \lambda_n$

➤ *Espace de phase*

Une façon de contourner le problème de SCI des systèmes chaotiques, est d'éliminer le temps entre les équations du système. C'est bien le rôle de l'espace des phases (ou espace des états), il s'agit d'un espace de dimension 2 ou 3 dans lequel chaque coordonnée est une variable d'état du système considéré [21].

## 6. Section de Poincaré

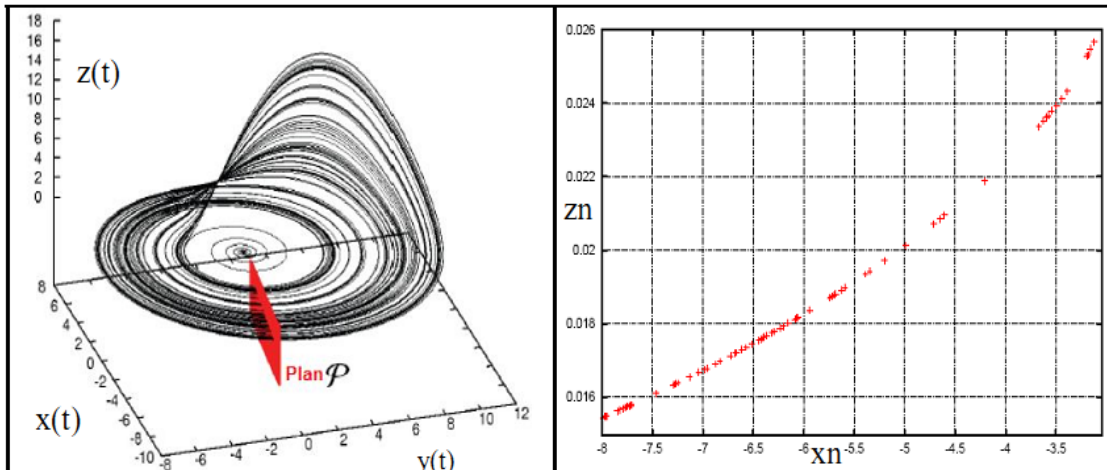
Cette section peut être considérée comme une transformation du système en temps continu en un système en temps discret.

*Henri Poincaré* a apporté une contribution très utile pour l'étude des systèmes chaotiques. Parmi ces contributions on trouve les sections de Poincaré. Faire une section de Poincaré revient à couper la trajectoire dans l'espace des phases, afin d'étudier les intersections de cette trajectoire avec, par exemple en dimension trois, un plan. On passe alors d'un système dynamique à temps continu à un système dynamique à temps discret. Les mathématiciens ont bien sûr démontré que les propriétés du système sont conservées après la réalisation d'une section de Poincaré judicieusement choisie, en particulier, un cycle limite simple du système continu est remplacé par un point fixe de l'application de Poincaré.

Une application de Poincaré peut être vue comme un système dynamique discret avec un espace d'état de dimension égale à celle du système dynamique continu original, moins une. Comme ce nouveau système dynamique conserve plusieurs propriétés des orbites périodiques et quasi périodiques du système original et comme le nouveau système comporte un espace d'états de dimension inférieure, il est souvent utile pour l'analyse du système original. Par contre, ceci n'est pas toujours possible en pratique puisqu'il n'existe aucune méthode générale de construction de l'application de Poincaré.

Citons quelques méthodes proposées pour tracer une section de Poincaré :

- ❖ La section de Poincaré la plus simple et naïve est de couper la trajectoire dans l'espace des phases par un plan (en 3D) ou par une droite (en 2D).



**Figure. I.7** Intersections de la trajectoire de l’attracteur de Rössler avec un plan  $\varphi$  d’équation  $y = 0, x \leq 4$  [21].

On voit clairement dans la Figure. I.7, que l’intersection de la trajectoire du système de Rössler avec le plan  $\varphi$  nous mène à l’étude des deux suites  $(x_n)_{n \in \mathbb{N}}$  et  $(z_n)_{n \in \mathbb{N}}$ .

- ❖ Une autre méthode consiste en des coupes faites par un plan passant par le point fixe et parallèle à un axe. Ce choix est arbitraire, et presque n’importe quelle surface ou plus généralement une variété de dimension  $d - 1$  permet de déterminer une section de Poincaré. La seule contrainte à respecter est que la trajectoire traverse la surface et n’y soit pas tangente.
- ❖ Une autre façon "toute aussi intéressante" de réaliser une section de Poincaré, consiste à regarder la suite des maximums de l’une des grandeurs du système (la surface d’équation  $\dot{x} = 0$ ).

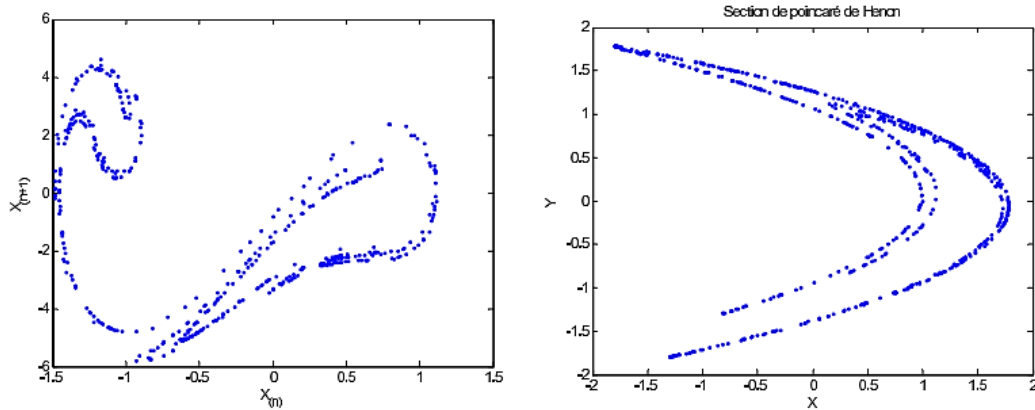
La première et la troisième technique que nous venons d’exposer au-dessus sont surtout valables pour des systèmes autonomes, c’est-à-dire des systèmes qui ne sont pas forcés.

*Remarque I.1 : En ce qui concerne l’étude du comportement chaotique, La section de Poincaré permet de visualiser s’il y a chaos ou non et les zones de stabilité. Aussi, d’après Poincaré, si toutes les trajectoires restaient dans un tore, la trajectoire est régulière et prédictible. Au cas où toute la section de Poincaré est percée de trajectoires, celle-ci sera chaotique et très instable dans le temps. Pratiquement, les zones stables sont liées aux orbites bien réguliers et aux phénomènes de résonance.*

*Remarque I.2 : Pour le contrôle des processus chaotiques, en considérant un point de la section de Poincaré pour une valeur du paramètre de contrôle, et en supposant que la condition initiale pour le système soit très proche de ce point, lors de son évolution, le système ne se stabilise jamais de lui-même autour du point. Ceci signifie qu’à chaque*

passage dans la section de Poincaré, le point courant est de plus en plus éloigné du point considéré. Pour contrôler le système, on se propose de lui imposer de rester autour du point, en modifiant légèrement la valeur du paramètre de contrôle. En introduisant des perturbations sur le paramètre, on modifie le comportement du système de façon à ramener les valeurs propres, régissant l'évolution dans la section de Poincaré, dans le cercle unité.

La figure suivante présente des sections de Poincaré pour deux fameux attracteurs :



(a) Section de Poincaré de l'attracteur de Moon

(b) Section de Poincaré de l'attracteur de Hénon

**Figure. I.8** Section de Poincaré des attracteurs à intervalle de temps régulier.

Pour résumer ce qui précède, les notions (équation non linéaire, représentation dans l'espace des phases, instabilité du système traduisant la SCI, notion d'attracteur, exposant de Lyapunov et la section de Poincaré) permettent de montrer si les phénomènes étudiés sont chaotiques ou non. Ces notions font donc partie du domaine théorique (elles reposent sur des équations). Expérimentalement, nous ne disposons le plus souvent que d'une seule variable parmi les différentes variables d'état qui caractérisent entièrement le système. En fin, la caractérisation du comportement dynamique du système se fait alors par le biais des divers outils vus précédemment : exposant de Lyapunov, diagramme de bifurcation (ou de *Feigenbaum*), et la section de Poincaré.

Ci-dessous, nous présentons quelques exemples des systèmes chaotiques très connus dans la littérature. Pour amples détails, le lecteur peut se référer aux références qui traitent les systèmes dynamiques, par exemple : L'attracteur de *Lorenz* [7], la transformation de *Hénon* [22], l'attracteur du *Double-scroll* [23] ; observé dans le circuit de *Chua*, etc... [16], [24-26].

## 7. Les dynamiques chaotiques

Depuis que l'attracteur de *Lorenz* est découvert en 1963, la recherche dans le domaine du chaos a attiré l'attention des chercheurs et experts, Plusieurs sortes de systèmes chaotiques et hyper-chaotiques ont été présentés par la suite ; les systèmes de *Chen*, *Chua*, *Rössler*, *Lü* ...etc. En termes de modèles mathématiques et de leurs propriétés, les dynamiques chaotiques peuvent être classées en chaos continu, chaos discret, chaos commuté, chaos retardé, hyper chaos ...etc. On s'intéresse beaucoup plus, dans ce manuscrit, sur les deux premières classes.

Dans ce qui suit, quelques propriétés de systèmes chaotiques continus et discrets seront exposées.

### 7.1. Le chaos continu

Plusieurs systèmes chaotiques continus ont été étudiés dans la littérature. Prenons comme exemple, le système de *Lorenz* (eq. I.3) [27,28] :

*Le système de Lorenz est un système d'équations non linéaires à cause des termes  $xy$  et  $xz$ . Ce système n'est pas intégrable dans le cas général. La détermination de ce système doit se faire à l'aide des méthodes d'approximation [28].*

Pour :  $\sigma = 10$ ,  $r = 28$  et  $b = \frac{8}{3}$ , et avec les conditions initiales  $x_0 = y_0 = z_0 = 0.01$ , le système (eq. I.3) est chaotique.

Dans ce qui suit, nous vérifions quelques propriétés (vue précédemment) du système chaotique de *Lorenz* (eq. I.3) :

#### ➤ Aspect aléatoire

La représentation de l'évolution temporelle des états du système de *Lorenz*, illustre l'aspect aléatoire de ces derniers (cf. Figure. I.9).

#### ➤ Attracteur étrange

Le système de *Lorenz* présente un superbe attracteur étrange, en forme *d'ailes de papillon* (cf. Figure. I.10). La trajectoire commençant par s'enrouler sur une aile, puis sautant pour commencer à s'enrouler sur l'autre aile, et ainsi de suite.

On constate que la dynamique du système de *Lorenz* est indépendante du temps  $t$ , par conséquent, ce type de système est qualifié d'être autonome [2].

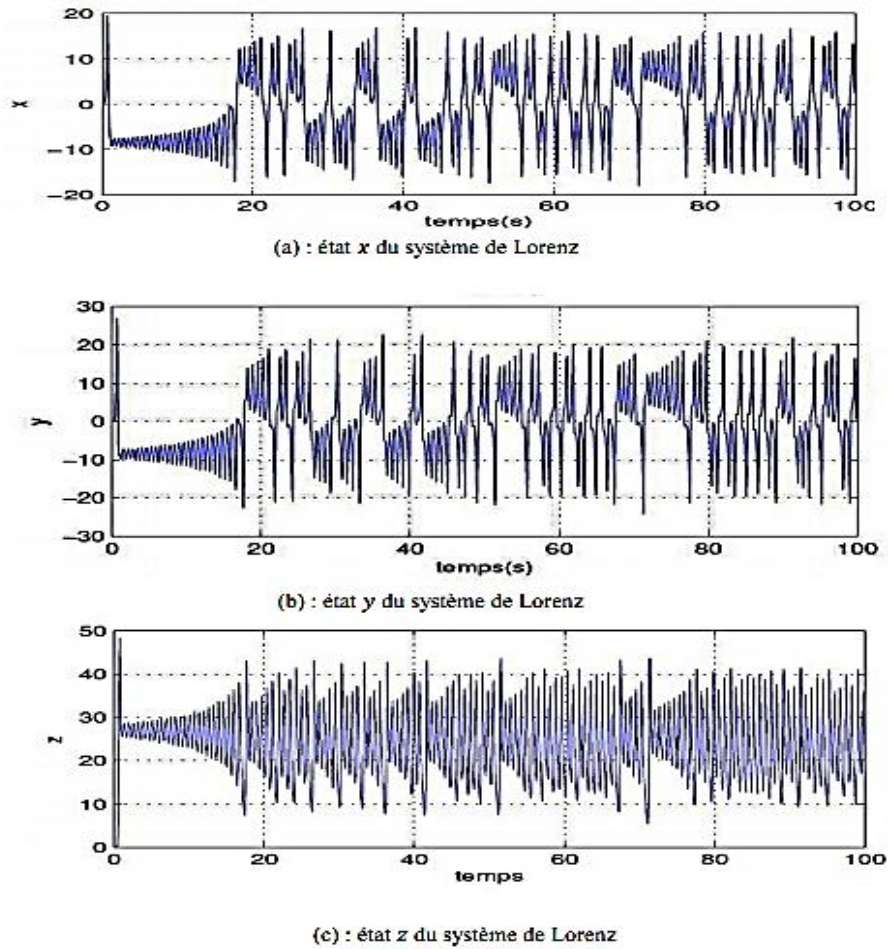


Figure. I.9 Aspects aléatoires des états du système de Lorenz.

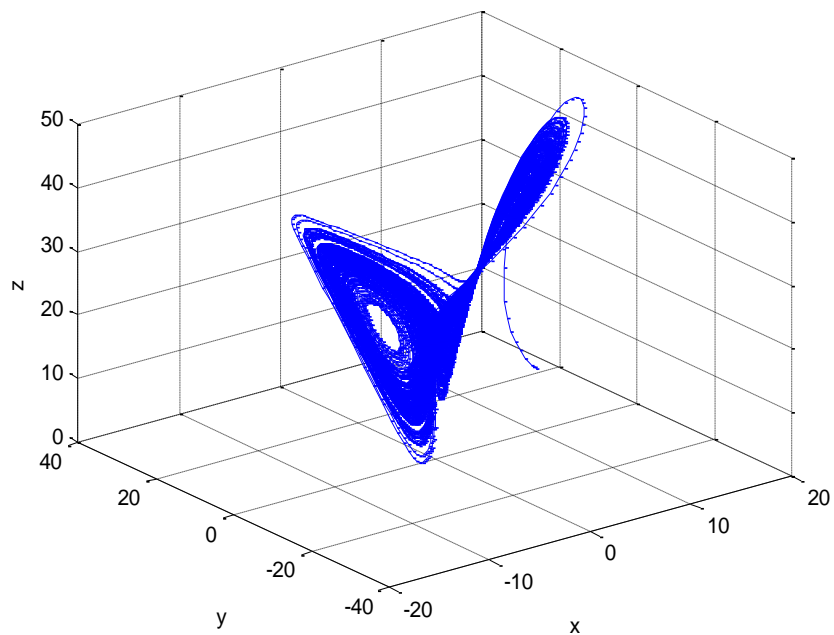


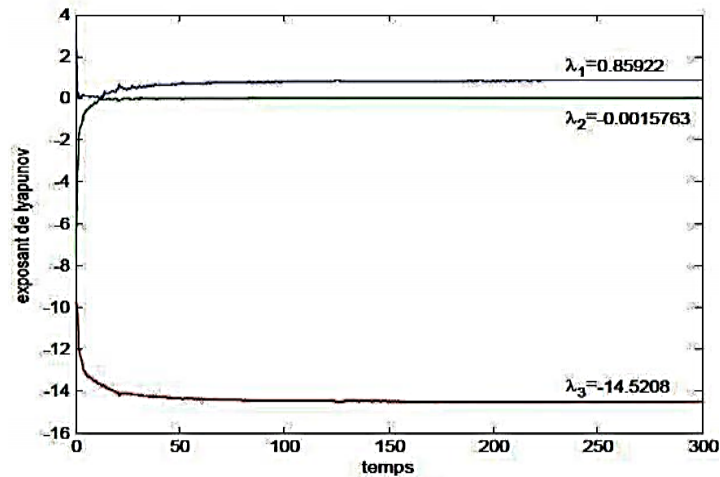
Figure. I.10 Attracteur étrange de Lorenz.



### ➤ Exposants de Lyapunov

Toujours avec le système de *Lorenz* (eq. I.3), le calcul de ses exposants de *Lyapunov*, nous donne les valeurs suivantes :  $\lambda_1 = 0.85922$ ,  $\lambda_2 = -0.0015763$  et  $\lambda_3 = -14.5208$ .

On voit clairement, qu'il y a un exposant de *Lyapunov* positif, ce qui signifie que le système est chaotique (cf. Figure. I.11).



**Figure. I.11** Exposants de Lyapunov du système de Lorenz [2].

## 7.2. Le chaos discret

Parmi les systèmes chaotiques discrets les plus connus, on fait appel à l'équation logistique (eq. I.7) qui est l'une des fonctions de *Chebyshev*, il existe toutefois d'autres systèmes chaotiques discrets comme le système de *Hénon* et la fonction *Gaussienne discrète* ...etc.

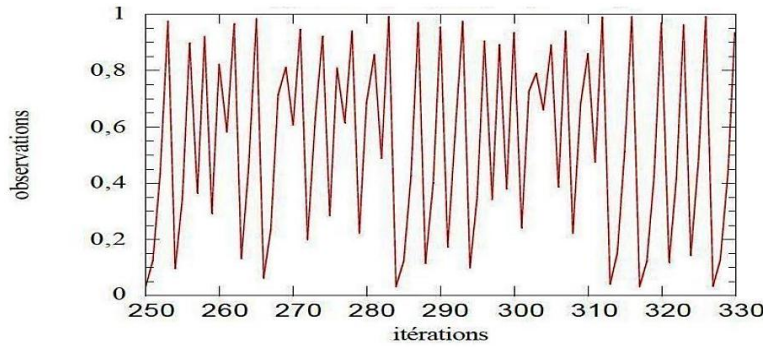
La fonction logistique (eq. I.7) très connue dans la théorie des systèmes non linéaires, est une application non bijective du domaine  $[0, 1]$  dans lui-même qui sert de récurrence à la suite [28].

La dynamique de cette application correspond à un comportement très différent ; ainsi selon la valeur du paramètre  $r$ , une plus grande variété de régimes permanents se présente (voir section 2.3 de ce chapitre et (Figure. I.2)).

De même que pour le cas continu, nous présentons dans ce qui suit quelques propriétés du système chaotique discret (eq. I.7).

➤ **Aspect aléatoire**

La figure au-dessous (Figure. I.12), illustre l'aspect aléatoire de l'équation logistique (eq. I.7) pour  $r = 4$ . Il est alors impossible de discerner à l'œil nu cette trajectoire de celle d'une variable aléatoire.



**Figure. I.12** Evolution de l'équation logistique pour  $r = 4$ .

➤ **Exposant de Lyapunov**

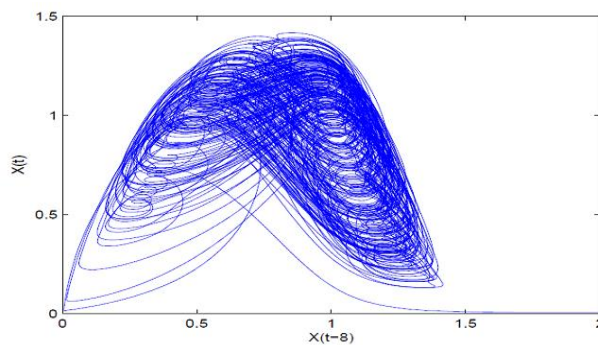
Comme il a été déjà mentionné l'équation logistique présente un comportement chaotique à partir d'une valeur spécifique du paramètre  $r$ . Pour  $r = 4$ , l'exposant de Lyapunov de l'équation logistique est positif,  $\lambda = \ln 2 > 0$ , d'où le comportement chaotique.

**7. 3. Le chaos à retard**

Le système de *Mackey- Glass* est le premier chaos à retard découvert en 1977 à partir d'un modèle physiologique. Il est donné par l'équation suivante [2] :

$$\dot{x}(t) = -x(t) + \frac{2x(t-\tau)}{1+x(t-\tau)^{10}} \tag{I.14}$$

Le portrait de phase de ce système est donné par (Figure. I.13).



**Figure. I.13** Portrait de phase du système de Mackey-Glass avec  $\tau = 8$ .

Un autre système chaotique à retard, est celui d'*Ikeda* obtenu comme modèle d'un système optique passif :

$$\dot{x}(t) = -x(t) + \mu \sin(x(t - \tau)) \quad (\text{I.15})$$

## 8. Applications du chaos

Pour nombre de scientifiques, la théorie du chaos représente le premier pas vers l'unification des sciences. En effet, il s'agit là d'une théorie dont les applications embrassent pratiquement toutes les sciences. Ainsi, elle a des applications en météorologie, sociologie, physique, informatique, ingénierie, économie, biologie et même en philosophie. On présente ci-dessous quelques applications du chaos [29] :

*La communication* : L'utilisation du chaos pour sécuriser les télécommunications est un sujet d'études depuis plusieurs années. L'intérêt des attracteurs *multi-plis* (*multi-scroll*) réside dans leur possibilité de permettre de générer des orbites plus courtes (par un chaos plus compliqué) et donc une transmission plus rapide des messages, ainsi qu'une meilleure sécurité dans les communications.

*En biologie* : La théorie du chaos permet, d'expliquer les variations des populations animales, les oscillations du cerveau. Aussi, les arythmies cardiaques typiques de nombreuses maladies du cœur se trouvent expliquées par la théorie du chaos.

*En économie* : Les mouvements commerciaux et les marchés financiers, ainsi que les cycles économiques, peuvent être expliqués en partie par la théorie du chaos, où les fractales ont un lien très étroit avec le hasard, et permettent donc de modéliser des expériences aléatoires complexes, d'où l'utilisation en finance, pour modéliser les variations des cours de la Bourse.

## 9. Nouveaux attracteurs chaotiques

L'intérêt grandissant des applications du chaos, qu'il apparaisse naturellement ou par chaotification, conduit la communauté scientifique à approfondir l'étude de modèles mathématiques simples, mais présentant un comportement chaotique.

Par exemple, ceci a conduit à la découverte de la notion de chaos *multi-spirales* (*multi-scroll*), développée à partir du circuit électrique de Chua [25,30].

Enfin, les attracteurs chaotiques découverts ouvrent de nouvelles voies de recherche et d'applications.

## 10. Conclusion

Dans le présent chapitre, quelques définitions et notions sur les systèmes dynamiques ont été présentées. A cet effet, nous avons rappelé les principales définitions mathématiques relatives aux systèmes dynamiques chaotiques, notamment des notions préliminaires sur les dynamiques du comportement chaotique. Et pour une meilleure compréhension du chaos déterministe, on a présenté par la suite, les propriétés des systèmes chaotiques, ainsi que la bifurcation et la section de Poincaré. Ensuite, les exemples les plus célèbres des systèmes chaotiques à temps continus et discrets ont été exposés. Et finalement on a fait une brève présentation des attracteurs multi-scroll (aussi ; multi-plis ou multi-spirales).

---

## Chapitre II

Techniques de l'Intelligence Computationnelle

---

---

## 1. Introduction

La majorité des systèmes dynamiques utilisés dans la réalité sont des systèmes non linéaires. Ce type de système est généralement sujet aux variations et incertitudes structurelles et non structurelles causées par l'environnement d'où le recours aux techniques de l'intelligence computationnelle pour palier à ce problème. Ces techniques sont connues par leur adaptation aux changements environnementaux.

Dans ce chapitre, nous allons présenter ces techniques (Réseaux de neurones et Algorithmes évolutionnaires) et leurs applications dans la modélisation et la commande des systèmes non linéaires (chaotiques).

## 2. Réseaux de neurones artificiels

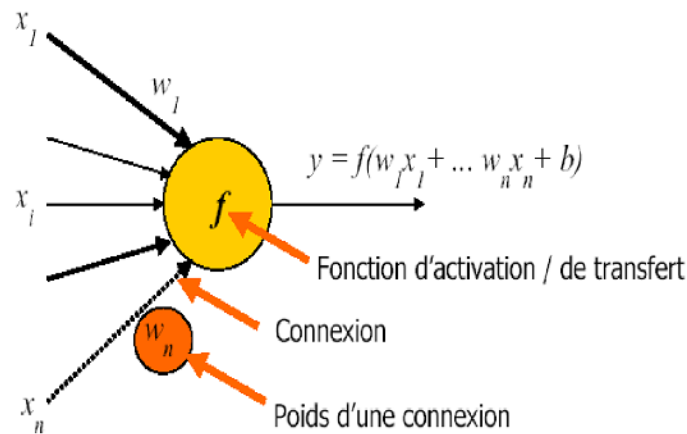
Les réseaux de neurones sont une des composantes importantes du domaine de l'intelligence artificielle [31]. Les réseaux de neurones *artificiels* (RNA), pour les différencier des réseaux de neurones biologiques, sont composés de neurones artificiels simples, petites fonctions mathématiques, qui permettent, montés en réseau de former des fonctions complexes très utiles. Par analogie aux neurones biologiques, les neurones artificiels ont pour but de reproduire des raisonnements *intelligents* d'une manière artificielle. Ces neurones peuvent adopter de certaines qualités habituellement propres au biologique, c'est-à-dire, la généralisation, l'évolutivité, et une certaine forme de déduction [32].

On doit le renouveau d'intérêt pour les RNA (ANN en anglais) en 1982, à un physicien reconnu *J. J. Hopfield*. A cela plusieurs raisons : Au travers d'un article court, clair et bien écrit, il présente une théorie du fonctionnement et des possibilités des réseaux de neurones [33]. Il faut remarquer la présentation anticonformiste de son article. Alors que les auteurs s'acharnent jusqu'alors à proposer une structure et une loi d'apprentissage, puis à étudier les propriétés émergentes ; *Hopfield* fixe préalablement le comportement à atteindre pour son modèle et construit à partir de là, la structure et la loi d'apprentissage correspondant au résultat escompté. Ce modèle est aujourd'hui encore très utilisé pour des problèmes d'optimisation. D'autre part, entre les mains de ce physicien distingué, la théorie des réseaux de neurones devient respectable. Elle n'est plus l'apanage d'un certain nombre de psychologues et neurobiologistes hors du coup.

Les RNA occupent de nos jours une place importante dans le domaine de la modélisation, de la commande et de la classification, ce qui justifie l'intérêt que le monde scientifique et industriel continue à porter au *connexionnisme*.

### 2.1. Neurone formel

Le neurone formel (artificiel) est un modèle mathématique simplifié du neurone biologique. Il présente un certain nombre d'entrées, un corps traitant ces entrées et un axone véhiculant la réponse du neurone. La première modélisation d'un neurone découle des résultats des travaux significatifs de *J. Mc Culloch et W. Pitts* [34]. Un neurone est donc avant tout un opérateur mathématique, dont on peut calculer la valeur numérique par quelques lignes de logiciel. On a pris l'habitude de représenter graphiquement un neurone comme indiqué sur la Figure II.1.



**Figure II.1** Modèle de base d'un neurone formel [29].

où le modèle de la Figure II.1 est composé :

- Des entrées (ou variables)  $x_i, i = 1, 2, \dots, n$ ; l'entrée spéciale  $b$  est appelée le biais ;
- Des paramètres ajustables (Poids)  $w_i$  ;
- De la fonction d'activation non linéaire  $f$  (généralement une tangente hyperbolique ou une sigmoïde) ;
- D'une sortie  $y$  donnée par la relation (eq. II.1) :

$$y = f(w_1x_1, w_2x_2, \dots, w_nx_n + b) = f(\sum_{i=1}^n w_ix_i + b) \quad (\text{II.1})$$

Chaque neurone artificiel est un processeur élémentaire, il reçoit un nombre de variables d'entrées en provenance des neurones en amont. A chaque entrée est associé un

poids qui représente la force de connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie pour alimenter un nombre variable de neurones en aval.

Dans cette thèse, des symboles et des variables sont utilisées pour représenter les composantes des réseaux de neurones. Le tableau ci-dessous donne les définitions de ce symbolisme (**Tableau II.1**).

**Tableau II.1:** Définitions de symbolisme.

<b>S</b>	Système que représente le réseau de neurone, de l'extérieur
<b>X</b>	Vecteur d'entrée du réseau, ou d'une manière locale, vecteur d'entrée d'une couche du réseau.
$x_i$	i-ème élément du vecteur d'entrée <b>X</b>
<b>Y</b>	Vecteur de sortie du réseau, ou d'une manière locale, vecteur de sortie produit par une couche du réseau.
$y_i$	i-ème élément du vecteur de sortie <b>Y</b>
$w_i$	Poids du i-ème entrée du neurone
$w_{i,j}$	Poids sur le lien entre le neurone i et le neurone j
<b>W<sub>i</sub></b>	Vecteur de poids associé au neurone i
<b>W</b>	Matrice de poids, associée à une couche.
$f()$	Fonction d'activation d'un neurone
<b>b</b>	Seuil d'activation d'un neurone

Les fonctions d'activation (ou de transfert) peuvent avoir l'une des formes définies dans le **Tableau II.2**. La fonction d'activation utilisée dans le modèle de *McCulloch & Pitts* est la fonction échelon (Figure II.2-a). Elle fait passer l'activation du neurone d'une valeur à une autre dès que l'entrée résultante dépasse un certain seuil. L'inconvénient de cette fonction est qu'elle n'est pas différentiable, ce qui pose un problème pour les algorithmes basés sur le gradient.

Pour remédier à cet inconvénient, on cherche à approximer cette fonction d'activation par une fonction *non linéaire différentiable*.

Deux fonctions de ce type sont particulièrement intéressantes et sont souvent utilisées : la fonction tangente hyperbolique (Figure II.2.b) définie par :

$$f(u) = \tanh(\beta u) = \frac{e^{\beta u} - e^{-\beta u}}{e^{\beta u} + e^{-\beta u}} \tag{II.2}$$

et la fonction logistique (Figure II.2.c) dont l'expression est la suivante :

$$f_{\beta}(u) = \frac{1}{1 + e^{-\beta u}} \tag{II.3}$$



La fonction « *tanh* » est bornée entre -1 et +1 alors que la fonction logistique est bornée entre 0 et 1. Ces deux fonctions, appelées *fonctions sigmoïdes*, sont liées par la relation :

$$\tanh(\beta u) = 2f_{\beta}(u) - 1 \tag{II.4}$$










où le paramètre  $\beta$  est appelé le gain. Plus le gain est important, plus la saturation du neurone est rapide (dans notre travail, le paramètre  $\beta$  est pris égal à 1).

La fonction logistique (II.3) est appelée aussi *sigmoïde unipolaire*. Elle admet une variante appelée *sigmoïde bipolaire* (Figure II.2.d) ayant pour expression :

$$f_{\beta}(u) = \frac{1 - e^{-\beta u}}{1 + e^{-\beta u}} \tag{II.5}$$

Les fonctions sigmoïdes ont la propriété d'être différentiables, ce qui est nécessaire pour les algorithmes basés sur le gradient. Une autre propriété intéressante est le fait que les fonctions dérivées peuvent s'exprimer facilement à l'aide des fonctions elles-mêmes, ce qui permet un gain significatif de temps de calcul [35].

**Tableau II.2:** Fonctions d'activation  $a = f(n)$  [14].

Nom de la fonction	Relation d'entrée/sortie	Icône	Nom Matlab
seuil	$a = 0$ si $n < 0$ $a = 1$ si $n \geq 0$		hardlim
seuil symétrique	$a = -1$ si $n < 0$ $a = 1$ si $n \geq 0$		hardlims
linéaire	$a = n$		purelin
linéaire saturée	$a = 0$ si $n < 0$ $a = n$ si $0 \leq n \leq 1$ $a = 1$ si $n > 1$		satlin
linéaire saturée symétrique	$a = -1$ si $n < -1$ $a = n$ si $-1 \leq n \leq 1$ $a = 1$ si $n > 1$		satlins
linéaire positive	$a = 0$ si $n < 0$ $a = n$ si $n \geq 0$		poslin
sigmoïde	$a = \frac{1}{1 + \exp^{-n}}$		logsig
tangente hyperbolique	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
compétitive	$a = 1$ si $n$ maximum $a = 0$ autrement		compet

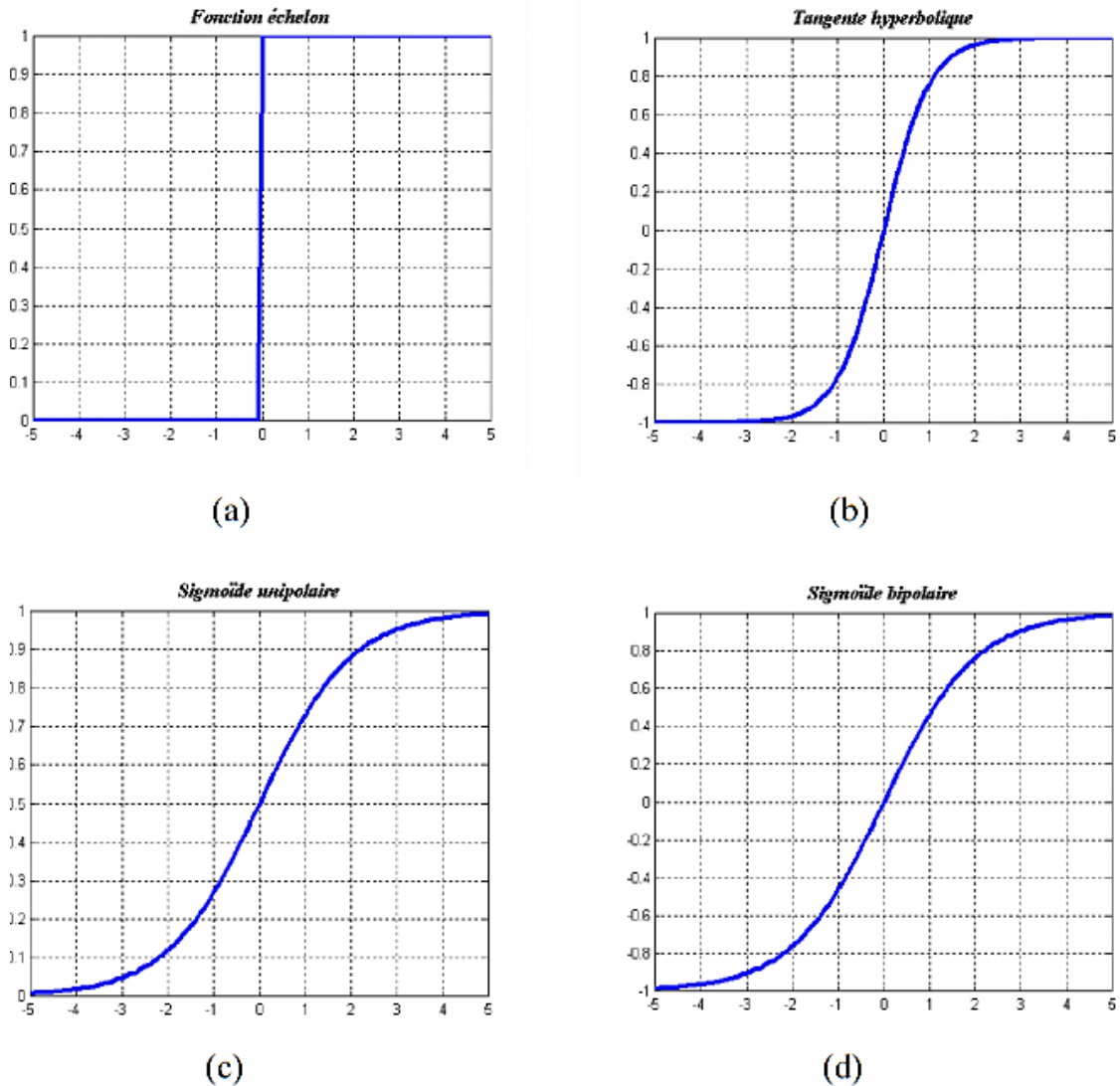


Figure II.2 : Types de fonctions d'activation.

## 2.2. Types des réseaux de neurones

Un réseau de neurones est constitué d'un nombre fini d'unités de traitement identiques (de neurones artificiels). Le réseau peut donc se représenter par un réseau ou un graphe orienté, dont les nœuds sont les neurones artificiels.

L'architecture générale des réseaux de neurones consiste en la représentation des neurones en couches successives, la première représentant la couche d'entrée, la dernière étant la couche de sortie, les couches intermédiaires étant les couches cachées du réseau. Ces couches sont dites cachées car de l'extérieur du réseau, on ne peut analyser clairement leur fonctionnement. On ne connaît vraiment que les signaux d'entrées-sorties du réseau. Les neurones de la couche d'entrée ne sont pas vraiment des neurones traitants (*computing neurons*), mais ont pour seule utilité de normaliser l'entrée des signaux ainsi que la

distribution des signaux d'entrées. Dans cette architecture normalisée, les couches de neurones sont totalement interconnectées, c'est-à-dire que les neurones d'une couche sont tous reliés à tous les neurones des couches adjacentes. Cette architecture normalisée peut sembler rigide, mais elle permet une représentation correcte de la plupart des réseaux de neurones, tout en permettant l'utilisation d'algorithmes d'entraînement plus généraux [31,32]. En effet, le type de connections, la nature de la fonction d'activation et d'autres paramètres détermineront le type du réseau de neurones (Perceptron multicouches (MLP), Réseaux à fonctions de base radiales (RBF), Réseau de Kohonen, ...etc) [34].

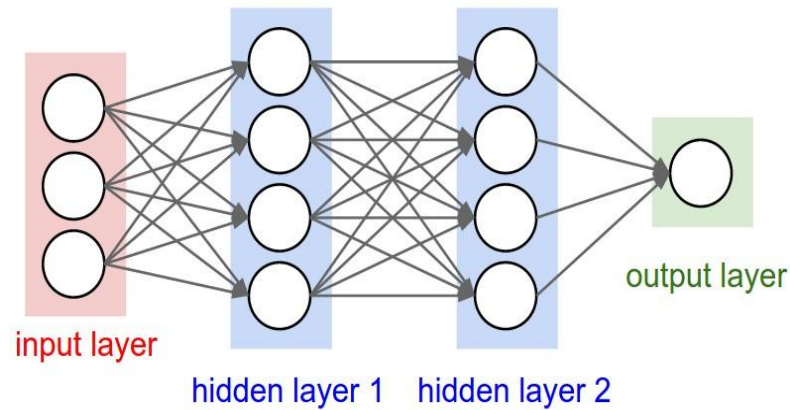
Les réseaux de neurones peuvent être soit *statiques*, soit *dynamiques*<sup>1</sup>. Les réseaux dynamiques ont la capacité de s'actualiser (modifier les poids des liaisons) en tout temps. Ces réseaux possèdent habituellement des algorithmes d'entraînements non-supervisés (Apprentissage non-supervisés), ce qui permet au réseau en question d'*évoluer* à chaque utilisation. Les réseaux statiques se séparent en deux catégories : La première concerne les réseaux entraînés, c'est-à-dire les réseaux qui ont vu leurs poids modifiés par un algorithme d'entraînement, sous la supervision d'un instructeur jusqu'à ce que ces réseaux donnent des résultats satisfaisants aux données d'entraînement. La seconde catégorie concerne les réseaux *non-évolutifs*, comme les réseaux de mémoire. Les poids de ces réseaux sont choisis (et non pas modifiés par un entraînement), pour donner l'effet escompté [36].

### 2.2.1. Réseaux de neurones statiques (Feed-forward)

Un réseau de neurones non bouclé réalise une ou plusieurs fonctions algébriques de ses entrées, par composition des fonctions réalisées par chacun de ses neurones. Il contient plusieurs couches ; une couche d'entrée, une couche de sortie, et au moins une couche cachée. Les signaux d'entrées vont être propagés vers la couche de sortie via les couches cachées, sans aucune connexion entre les neurones de la même couche et sans retour en arrière.

La Figure II.3 représente une structure d'un réseau de neurones statique avec deux couches cachées.

<sup>1</sup> Les réseaux sont des structures statiques, c'est-à-dire que, une fois construit, leur architecture ne change pas. Ici, on parle plutôt du caractère statique ou dynamique des poids du réseau lors de son utilisation, c'est-à-dire, si les valeurs des poids du réseau peuvent ou non se modifier lors de l'utilisation (à différencier de l'entraînement) du réseau.



**Figure II.3** Structure d'un réseau de neurones statique PMC [35].

En 1958, *F. Rosenblatt* proposa le *perceptron*, accompagné d'un algorithme d'entraînement [37]. Le perceptron peut être vu comme le type de réseau de neurones le plus simple. Ce type de réseau neuronal ne contient aucun cycle (en anglais feedforward neural network FFNN). De plus, comme *Minsky* et *Papert* l'ont montré dans leur livre *Perceptrons* en 1969, les perceptrons ne peuvent généraliser sur des exemples appris localement [38]. Malgré que théoriquement il soit possible de construire des réseaux avec un très grand nombre de couches cachées, les réseaux comprenant plus de couches cachées sont très rares, étant donné que chaque nouvelle couche augmente la quantité de calculs d'une manière exponentielle. La plupart des réseaux de neurones multicouches sont, dans la pratique, des perceptrons multicouches (PMC, en anglais MLP).

Par convention, les neurones d'entrée des PMC ont toujours une fonction d'activation *identité*, laissant passer l'information sans la modifier. Pour les neurones de sortie, on peut leur associer une fonction d'activation linéaire ou non, dérivable ou non, suivant la nature du problème à résoudre. En ce qui concerne la fonction d'activation associée aux neurones cachés, on utilise dans le cadre de cette thèse une fonction d'activation de la famille des sigmoïdes.

Les PMC sont des objets statiques : si les entrées sont indépendantes du temps, les sorties le sont également. Ils sont utilisés principalement pour effectuer des tâches d'approximation de fonction non linéaire, de classification ou de modélisation des systèmes non linéaires (chaotiques).

2.2.2. Réseaux de neurones dynamiques (Feed-back)

A l'opposé des réseaux à sens unique, on trouve les modèles dits récurrents (dynamiques), qui fut publiée pour la 1<sup>ère</sup> fois en 1982. Dans un modèle récurrent, l'architecture de connexion autorise des couplages réciproques entre les neurones. Le modèle pionnier dans ce domaine est celui de *Hopfield* [33], dont le mode de calcul met en œuvre la convergence de la dynamique sur un attracteur. Une telle structure est représentée par la Figure II.4.

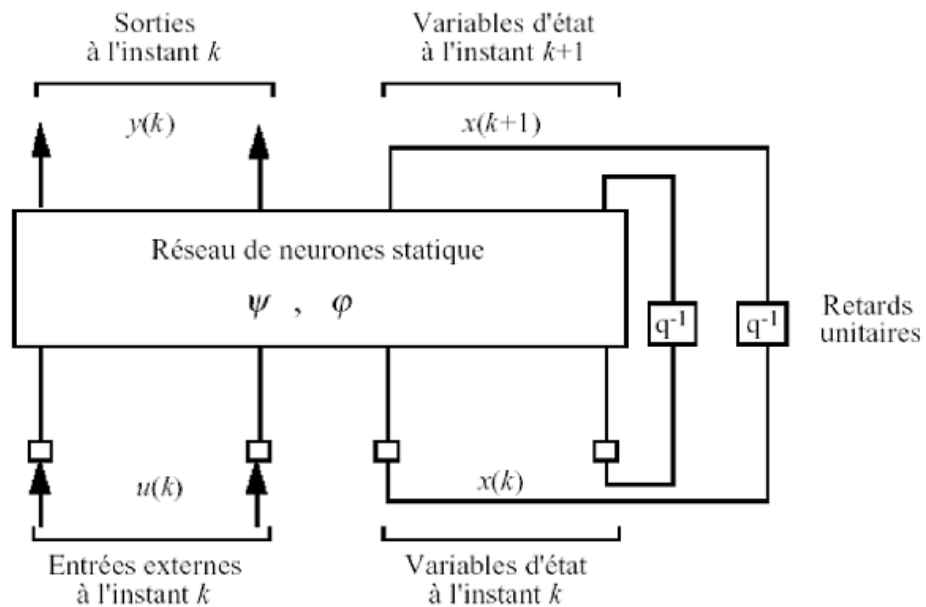


Figure II.4 Forme canonique d'un réseau de neurones dynamique [39].

Un réseau de neurones bouclé est donc un système dynamique, régi par des équations différentielles ; comme l'immense majorité des applications sont réalisées par des programmes d'ordinateurs, on se place dans le cadre des systèmes à temps discret, où les équations différentielles sont remplacées par des équations aux différences. Un réseau de neurones bouclé à temps discret est donc régi par une (ou plusieurs) équations aux différences non linéaires, résultant de la composition des fonctions réalisées par chacun des neurones et des retards associés à chacune des connexions.

La forme la plus générale des équations régissant un réseau de neurones bouclé est appelée forme canonique ;

$$\begin{aligned} x(k + 1) &= \varphi[x(k), u(k)] \\ y(k) &= \Psi[x(k), u(k)] \end{aligned} \tag{II.6}$$

où  $\varphi$  et  $\Psi$  sont des fonctions non linéaires réalisées par un réseau de neurones non bouclé (par exemple, mais pas obligatoirement, un PMC), et où  $k$  désigne le temps (discret). Il est à noter au passage, que tout réseau de neurones bouclé, aussi compliqué soit-il, peut être mis sous une forme canonique [40]. Cette forme canonique est représentée sur la Figure II.4.

*Remarque II.1 : Il existe une grande quantité d'architectures intermédiaires qui marient plusieurs des aspects présentés séparément. On peut citer en particulier l'architecture NARX [41], qui possède une architecture de type feed-forward, avec une partie des sorties qui rebouclent sur les entrées.*

L'architecture du réseau étant fixée, l'étape suivante consiste à estimer les coefficients représentant la force de connexion des neurones (poids) pour remplir au mieux la tâche à laquelle le réseau est destiné (modélisation, approximation, prédiction, contrôle...etc.). Les valeurs des coefficients sont en général déterminées par un processus d'apprentissage à partir des entrées externes.

### 2.3. Propriétés des réseaux de neurones artificiels

Les propriétés essentielles des réseaux de neurones sont :

#### 2.3.1. Le parallélisme

Cette notion se situe à la base de l'architecture des réseaux de neurones considérés comme ensemble d'entités élémentaires qui travaillent simultanément. Le parallélisme permet une rapidité de calcul supérieure mais exige de penser et de poser différemment les problèmes à résoudre.

#### 2.3.2. La capacité d'adaptation

Elle se manifeste par la capacité d'apprentissage qui permet de tenir compte des nouvelles contraintes ou de nouvelles données du monde extérieur. Certains réseaux se caractérisent aussi par leur capacité d'auto-organisation qui assure leur stabilité en tant que systèmes dynamiques capables de tenir compte de situations non encore connues.

#### 2.3.3. La facilité de construction

La simulation informatique de réseaux de neurones pour une petite application est simple et ne nécessite qu'un temps de développement assez court. Pour des applications plus complexes, l'utilisation de simulateur ou de carte accélératrice se révèle utile.

### 2.3.4. L'approximation universelles parcimonieuse (propriété fondamentale des RNA)

Les réseaux de neurones à couches ont la propriété générale d'être des approximateurs universels parcimonieux (Hagan et al, 1995) [42,43].

Les travaux de Cybenko [44] et *Funahashi* [45], en 1989, ont prouvé la possibilité d'approximer n'importe quelle fonction non linéaire avec la précision voulue par des réseaux de neurones à une seule couche cachée avec des fonctions d'activations non linéaires. Cependant, rien n'est dit sur le nombre de neurones nécessaires à cette approximation. Dans le cas du théorème de *Cybenko*, l'hypothèse sur la fonction d'activation est qu'elle a pour limite 0 en  $-\infty$  et 1 en  $+\infty$ , dans celui de *Funahashi*, qu'elle est croissante, non constante et bornée.

Les fonctions non continues, mais mesurables, peuvent aussi être approchées, mais dans un sens moins fort (Hornik et al., 1990) [46]. Il existe par ailleurs quelques résultats sur le nombre de neurones requis pour approcher une fonction avec une précision fixée, pour certaines classes de fonctions (Neelakantan et Guiver, 1998) [47].

Ces résultats affirment donc, pour toute fonction déterministe usuelle, l'existence d'une approximation par un réseau de neurones. Les réseaux complètement connectés ou à couches, et à neurones cachés sigmoïdaux, remplissent les conditions d'application des théorèmes [43]. Dans cette thèse, nous utilisons systématiquement ce type de réseaux (PMC), à l'exclusion par exemple des réseaux utilisant des fonctions à base radiale (RBF), pour la raison que, même si ces réseaux jouissent également de propriétés d'approximation intéressantes, et même si leur apprentissage peut être réalisé à l'aide de la méthode des moindres carrés ordinaires, leur utilisation est souvent beaucoup moins économique, ou parcimonieuse du point de vue du nombre de connexions, que celle des réseaux à sigmoïdes.

*Remarque II.2 : En toute rigueur, les réseaux à RBF peuvent être aussi parcimonieux que les réseaux à sigmoïdes, mais à condition d'ajuster la position des centres des RBF de manière non linéaire (Park et Sandberg, 1991) [48], ce qui supprime le principal intérêt des RBF : la simplicité de l'apprentissage par la méthode des moindres carrés ordinaires.*

La spécificité des réseaux de neurones réside dans le caractère *parcimonieux* de l'approximation ; où à précision égale, les réseaux de neurones nécessitent moins de paramètres ajustables que les autres approximateurs ; plus précisément, le nombre de poids

varie *linéairement* avec le nombre de variables de la fonction à approcher, alors qu'il varie *exponentiellement* pour la plupart des autres approximateurs (Hornik et al., 1994) [49].

C'est cette remarquable parcimonie qui justifie l'intérêt pratique des réseaux de neurones : dès qu'un problème fait intervenir plus de deux variables, les réseaux de neurones sont, en général, préférables aux autres méthodes [49]. Nous allons voir ultérieurement pourquoi cette propriété de parcimonie est précieuse dans les applications.

### 3. Réseaux de neurones et régression non linéaire

Dans la pratique, on n'utilise pas les réseaux de neurones pour réaliser des approximations de fonctions connues. Le plus souvent, le problème qui se pose à l'ingénieur est le suivant : il dispose d'un ensemble de mesures de variables d'un processus de nature quelconque (physique, chimique, économique, financier, ...), et du résultat de ce processus ; il suppose qu'il existe une relation déterministe entre ces variables et ce résultat, et il cherche une forme mathématique de cette relation, valable dans le domaine où les mesures ont été effectuées, sachant que les mesures sont en nombre fini, qu'elles sont certainement entachées de bruit, et que toutes les variables qui déterminent le résultat du processus ne sont pas forcément mesurées. En d'autres termes, l'ingénieur cherche un modèle du système qu'il étudie, à partir des mesures dont il dispose, et d'elles seules : on dit qu'il effectue une modélisation **boîte noire** [39, 50].

De manière générale, un réseau de neurones permet donc de faire un meilleur usage des mesures disponibles que les méthodes de régression non linéaires conventionnelles. Ce gain peut être considérable lorsque le système à modéliser dépend de plusieurs variables (le cas des systèmes chaotiques). Ainsi, à la lumière de cette propriété fondamentale, la technique des réseaux de neurones apparaît comme une puissante méthode de régression non linéaire : ce n'est donc rien d'autre qu'une extension des méthodes de régression linéaire ou multilinéaires proposées par tous les logiciels qui permettent de faire de la modélisation de données.

### 4. Apprentissage des réseaux de neurones

L'apprentissage et l'adaptation constituent deux caractéristiques essentielles des réseaux de neurones. Le rôle de l'apprentissage est de définir les poids de chaque connexion. De nombreuses règles existent pour modifier les poids des connexions pour arriver à un apprentissage correct. Lorsque la phase d'apprentissage est achevée, le réseau doit être



capable de faire les bonnes associations pour les vecteurs d'entrées qu'il n'aura pas appris (Moody et Darken, 1995) [51]. Le principe d'apprentissage est l'*optimisation* d'une *fonction de coût* calculée à partir des exemples de la base d'apprentissage et de la sortie du réseau de neurones.

En d'autres termes, l'apprentissage d'un réseau de neurones consiste à déterminer les valeurs des poids permettant à la sortie du réseau de neurones d'être aussi proche que possible de l'objectif fixé.

Il existe deux types principaux d'apprentissages : l'apprentissage supervisé et l'apprentissage non-supervisé.

#### ❖ Apprentissage supervisé

Pour les réseaux à apprentissage supervisé, on présente au réseau des entrées, et au même temps les sorties désirées pour cette entrée. Le réseau doit ajuster ses poids de façon à réduire l'écart entre la réponse désirée et la sortie du réseau. Cette procédure est répétée jusqu'à ce qu'un critère de performance soit satisfait. L'algorithme le plus utilisé est celui de la *rétro-propagation* de l'erreur. Le réseau a alors comme but d'approximer ces exemples aussi bien que possible et de développer à la fois la bonne représentation mathématique qui lui permet de généraliser ces exemples pour ensuite traiter des nouvelles situations [43].

#### ❖ Apprentissage non-supervisé

Dans le cas de l'apprentissage non-supervisé le réseau décide lui-même quelles sont les bonnes sorties. Cette décision est guidée par un but interne au réseau qui exprime une configuration idéale à atteindre par rapport aux exemples introduits ; on présente une entrée au réseau et on le laisse évoluer librement jusqu'à ce qu'il se stabilise. Les cartes auto organisatrices de *Kohonen* sont un exemple de ce type de réseau [52].

Outre la classification présentée ci-dessus, les méthodes d'apprentissage sont souvent différenciées par leur caractère hors ligne (off-line) ou en ligne (on-line) [53] :

#### ❖ Apprentissage hors ligne

Ce mode d'apprentissage consiste à accumuler les erreurs instantanées consécutives, et à n'effectuer l'adaptation des poids synaptiques que lorsque l'ensemble des données d'apprentissage ont été présentées au réseau. Cette dernière méthode permet de mieux

estimer le gradient réel de la fonction coût, puisqu'il est à présent calculé à partir d'un ensemble d'exemples, plutôt qu'à partir d'un seul.

### ❖ Apprentissage en ligne

Il consiste à modifier les valeurs des poids synaptiques immédiatement après la présentation d'un exemple. Dans ce cas, seul le gradient instantané de la fonction coût est utilisé pour l'adaptation des paramètres du système. Sous la condition que les exemples soient présentés au réseau de neurones de manière aléatoire, l'apprentissage en-ligne rend la recherche du minimum de la fonction coût stochastique en nature, ce qui rend moins probable, pour l'algorithme d'apprentissage, de tomber dans un minimum local.

L'efficacité relative des modes d'apprentissage en-ligne et hors-ligne dépend essentiellement du problème considéré. L'apprentissage en-ligne présente cependant l'avantage que, pour une seule présentation de l'ensemble de la base de données, il implique de multiples phases d'adaptations des poids synaptiques lorsque des données similaires se représentent, ce qui se produit fréquemment pour des bases de données très étendues [43].

*Remarque II.3 : On dit que le réseau est supervisé lorsque les modèles sont constitués de couples valeurs d'entrées-valeurs de sorties désirées, et il est non supervisé lorsque seules les valeurs d'entrées sont disponibles.*

## 5. La conception d'un réseau de neurones

Les réseaux de neurones sont utilisés pour résoudre plusieurs problèmes ; de modélisation, de commande, de régression, ...etc. Afin de concevoir un réseau de neurones, nous citons chronologiquement, les quatre grandes étapes qui doivent être suivies [43].

### 5.1. Choix et préparation des échantillons

Le processus d'élaboration d'un réseau de neurones commence toujours par le choix et la préparation des échantillons de données (*les exemples*). Comme dans les cas d'analyse de données, cette étape est cruciale et va aider le concepteur à déterminer le type de réseau le plus approprié pour résoudre son problème. La façon dont se présente l'échantillon conditionne : le type de réseau, le nombre de neurones d'entrée, le nombre de neurones de sortie et la façon dont il faudra mener l'apprentissage, les tests et la validation [51].

## 5.2. Élaboration de la structure du réseau de neurones

La structure du réseau dépend étroitement du type des échantillons. Il faut d'abord choisir le type de réseau : un perceptron standard, un réseau de Hopfield, un réseau de fonctions à bases radiales, un réseau de Kohonen ...etc. Dans le cas du perceptron par exemple, il faudra aussi choisir le nombre de neurones dans la couche cachée. Plusieurs méthodes existent et nous pouvons par exemple prendre une moyenne du nombre de neurones d'entrée et de sortie, mais rien ne vaut de tester toutes les possibilités et de choisir celle qui offre les meilleurs résultats [43].

## 5.3. Apprentissage du réseau de neurones

L'apprentissage consiste tout d'abord à calculer les poids optimaux des différentes liaisons, en utilisant un échantillon. La méthode la plus utilisée est la rétro-propagation : nous entrons des valeurs des cellules d'entrée (*couples valeurs d'entrées-valeurs de sorties désirées*) et en fonction de l'erreur obtenue en sortie, nous corrigeons les poids. Ce cycle est répété jusqu'à ce que la courbe d'erreur du réseau ne soit croissante (il faut bien prendre garde de ne pas surentraîner un réseau de neurones qui deviendra alors moins performant) [51].

## 5.4. Validation et Test

Alors que les tests concernent la vérification des performances d'un réseau de neurones hors apprentissage et sa capacité de généralisation, la validation est parfois utilisée lors de l'apprentissage (e.g : cas du early stopping). Une fois le réseau calculé, il faut toujours procéder à des tests afin de vérifier que le réseau réagit correctement. Il y a plusieurs méthodes pour effectuer une validation : la cross-validation, le bootstrapping, ...etc. mais pour les tests, dans le cas général, une partie de l'échantillon est simplement écartée de l'échantillon d'apprentissage et conservée pour les tests hors échantillons. Nous pouvons par exemple utiliser 70% de l'échantillon pour l'apprentissage, 15% pour la validation et 15% pour les tests. Dans les cas de petits échantillons, nous ne pouvons pas toujours utiliser une telle distinction, simplement parce qu'il n'est pas toujours possible d'avoir suffisamment de données dans chacun des groupes ainsi créés. Nous avons alors parfois recours à des procédures comme la cross-validation pour établir la structure optimale du réseau [43].

## 6. Apprentissage des réseaux PMC (réseaux multicouches)

La plupart des réseaux de neurones multicouches sont, dans la pratique, des perceptrons multicouches (PMC) ; car même s'il est théoriquement possible de construire des réseaux avec un très grand nombre de couches cachées, les réseaux comprenant plus de deux couches cachées sont très rares, étant donné que chaque nouvelle couche augmente la quantité de calculs d'une manière exponentielle. En présente au-dessous l'algorithme d'apprentissage adapté au PMC.

### 6.1. Algorithme de retro-propagation du gradient (RP)

La méthode de modification des poids est très simple avec l'algorithme de *Rosenblatt*, mais il implique quelques limitations d'apprentissage [37]. Dans le cas de PMC, comme on ne sait pas les sorties (outputs) désirés des couches cachées, mais seulement de la dernière couche (couche de sortie), il faut propager la responsabilité des erreurs de la dernière couche à la première, dans le sens contraire de l'exécution de réseau, d'où le nom *rétro-propagation*. L'algorithme de RP du gradient est certainement à la base des premiers succès des réseaux de neurones. Sa mise en application a permis au domaine du *connexionnisme* de sortir de la période de silence qui a régné après la sortie du livre « Perceptrons » de (Minsky et Papert, 1969) [38].

Selon la littérature, la RP a été proposée plusieurs fois et de manière indépendante par : (Bryson et Ho, 1969) [54], (Werbos, 1974) [55], (Parker, 1985) [56], et enfin *Rumelhart* et les membres du groupe PDP en 1986 [57]. Une approche similaire a également été proposée par *Le Cun* [58]. Voici les principales lignes de la méthode de rétro-propagation :

#### **Calcul de l'erreur :**

$e_k(p) = y_{d,k} - y_k$  : l'erreur du neurone  $k$  est la différence entre la valeur de sortie désiré et la valeur actuelle du neurone  $k$ )

#### **Correction de poids (par la loi delta):**

$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p)$  : la correction du poids du lien du neurone  $j$  au neurone  $k$ , où  $\delta_k(p)$  est le gradient d'erreur du neurone  $k$  à l'itération  $p$ .

#### **Gradient d'erreur (pour les neurones de la couche de sortie) :**

$\delta_k(p) = F'[X_k(p)] \times e_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p)$

La deuxième partie est obtenue par la dérivée de la fonction sigmoïde appliquée à l'intrant net du neurone  $k$  à l'itération  $p$

**Gradient d'erreur (pour les neurones d'une couche cachée) :**

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^l \delta_k(p) w_{j,k}(p)$$

où  $l$  est le nombre de neurones sur la couche suivante.

Algorithme de traitement

**1- Initialisation**

Mettre tous les poids et les seuils d'activation du réseau à des valeurs aléatoires uniformément distribuées dans un petit intervalle :

$$\left( -\frac{2,4}{F_i}, \frac{2,4}{F_i} \right) \text{ (selon [41]), où } F_i \text{ est le nombre total de signaux intrants du neurone } i \text{ dans le réseau.}$$

Cette initialisation est faite un neurone à la fois.

Mettre la valeur du taux d'apprentissage à une petite valeur positive.

**2- Activation**

Activer le réseau de neurones en appliquant l'échantillon d'apprentissage; couples d'entrées-sorties désirés

Calculer les signaux sortant des neurones des couches successives, de la première couche cachée à la couche de sortie.

**3- Entraînement des poids**

Mettre à jour les poids du réseau en propageant les erreurs dans le sens inverse :

Calculer les gradients d'erreurs pour les neurones des couches successives (ordre inverse), de la couche de sortie à la première couche cachée.

Calculer les corrections de poids pour chaque lien et mettre à jour les poids.

**4- Itération**

Incrémenter  $p$ , retourner à l'étape 2 et répéter le procédé jusqu'à ce que le critère d'erreur soit atteint.

La méthode de rétro-propagation peut mettre énormément de temps avant que la sommation des erreurs carrées ne soit réduite à zéro (Ceci peut être impossible selon les cas). C'est pourquoi on utilise plutôt un critère d'erreur, soit une certaine valeur critique, qui doit être atteinte par la sommation des erreurs carrées (habituellement entre 0,001 et 0,1). La loi delta, utilisée par la méthode de rétro-propagation, a pour objectif de réduire la sommation des erreurs carrées produites par le réseau. Si on représente la courbe de la fonction de la somme des erreurs carrées par rapport à un poids du réseau, on peut observer que la variation idéale dépend de la pente de la courbe de la fonction, d'où l'utilisation de la dérivée dans le calcul du gradient d'erreur (et par le fait même, de la variation de poids). Comme cette recherche, telle la méthode du *hillclimbing*, recherche les *extréma* par une vérification de la pente, elle peut être limitée par l'atteinte de *minima locaux* [36].

L'entraînement d'un PMC avec l'algorithme de rétro-propagation du gradient se fait d'une manière similaire à l'entraînement d'un perceptron à neurone unique, mais il faut procéder neurone par neurone et poids par poids, en commençant par la dernière couche. Il faut calculer le gradient d'erreur pour chaque neurone, puis calculer la variation de poids pour les liens rattachés à ce neurone. Ensuite on passe à la couche précédente. Cette méthode devient très rapidement lourde en calculs.

La grande majorité des PMC sont des réseaux avec apprentissage supervisé, mais il en existe quelques-uns qui peuvent apprendre sans professeur (non-supervisé) pour les guider. Une classe de ces réseaux non-supervisés est celle des réseaux auto-associatifs. Par défaut, un PMC est un réseau hétéro-associatif, ce qui veut dire que la sortie est différente de l'entrée. Dans le cas particulier où l'entrée doit être la même que la sortie, ces réseaux étant appelés auto-associatifs, on n'a pas besoin des valeurs de sortie que devraient produire le PMC, celles-ci étant les mêmes que l'entrée. Ces PMC auto-associatifs ont ceci d'intéressants que la première partie des réseaux constitue un codage des données, alors que la seconde partie est le décodage, pour retrouver l'information entrée.

## 6.2. Variantes de l'algorithme de RP

Depuis son introduction, l'algorithme de RP a été largement étudié et plusieurs modifications y ont été apportées. Quelques améliorations ont été proposées pour augmenter la vitesse d'apprentissage des PMC, car l'algorithme de base décrit ci-dessus converge très lentement pour les réseaux multicouches :

- a) L'utilisation de la fonction tangente hyperbolique  $Y^{\tan h} = \frac{2a}{1+e^{-bX}} - a$ , où  $a$  et  $b$  sont constants. Les valeurs de  $a = 1,716$  et  $b = 0,667$  sont efficaces [59]. L'avantage de cette fonction par rapport à la fonction sigmoïde de base a été testé empiriquement.
- b) Le choix de la fonction d'erreur utilisée pour l'apprentissage des PMC a une certaine influence sur la rapidité d'apprentissage et sur la qualité de généralisation du réseau. Cette question a été étudiée par plusieurs chercheurs (Gayner et Downs, 1994) [60]. Le critère d'erreur le plus utilisé est la fonction d'erreur *quadratique moyenne* ; Cette fonction a tendance à amplifier les erreurs les plus importantes. Par conséquent, au cours de l'apprentissage, la mise à jour des poids est largement

déterminée par la correction des grandes erreurs, ce qui est recherché en général (Hassibi et Stork, 1993) [61].

- c) L'utilisation d'un terme de moment (*momentum term*) dans l'équation de la loi delta :  $\Delta w_{jk}(p) = \beta \times \Delta w_{jk}(p-1) + \alpha \times y_j(p) \times \delta_k(p)$ , avec  $0 \leq \beta < 1$  (une bonne valeur de  $\beta$  est 0,95). Cette nouvelle équation porte le nom de loi delta généralisée. L'ajout du terme de moment permet de stabiliser l'apprentissage, tout en augmentant la descente de la courbe de la somme des erreurs carrées selon les itérations.
- d) L'utilisation d'un taux d'apprentissage variable plutôt que constant. Si le changement dans la somme des erreurs carrées porte le même signe algébrique, le taux d'apprentissage devrait être augmenté. Si le signe algébrique du changement de la somme des erreurs carrées alterne d'une itération à l'autre, le taux d'apprentissage devrait être décrétementé.

## 7. Problèmes pour la généralisation

Les réseaux de neurones devant généraliser sur les exemples de l'ensemble d'apprentissage, ils ne sont qu'une approximation des fonctions que l'on recherchait vraiment. Quelques problèmes de l'approximation de réseaux de neurones sont identifiés ici et des solutions y sont proposées [36].

### 7.1. Compromis Biais-Variance

Le paramètre à minimiser dans le cas de l'entraînement d'un réseau de neurone est l'erreur sur les résultats donnés par le réseau de neurone. Cependant, celle-ci ne tient compte que des valeurs de l'ensemble d'entraînement, alors qu'on devrait tenir compte de l'erreur sur toutes les données à traiter. Ceci étant impossible, on base plutôt l'optimisation sur un risque moyen. Le risque quadratique moyen est un bon paramètre à optimiser. Ce risque se décompose en trois termes : l'*erreur bayésienne*, le *biais* et la *variance*. L'erreur bayésienne tient à l'apprentissage, mais elle est indépendante de la procédure d'apprentissage. La difficulté de cette optimisation est de contrôler à la fois le biais et la variance. Il faut donc trouver un compromis où la somme du biais et de la variance est minimale. Cela revient à accepter un certain biais pour maintenir la variance relativement faible.

Dans le cas des réseaux de neurones, c'est le contrôle de la complexité du réseau de neurones qui permet de trouver l'estimateur réalisant le bon compromis biais-variance. Ainsi, des bornes théoriques de la taille du réseau en fonction de la taille de l'ensemble d'entraînement ont été proposées pour garantir une bonne qualité de généralisation (Baum et Haussler, 1989) [62]. Cependant, si ces résultats sont d'une grande importance sur le plan théorique, ils le sont nettement moins dans la pratique. En effet, ces résultats ne constituent que des preuves d'existence et ne permettent pas de déterminer l'architecture d'un réseau de neurones pour un problème donné (classification, approximation, prédiction,...etc).

## 7.2. Choix de l'architecture

Le choix de l'architecture d'un réseau détermine la classe des fonctions calculables par celui-ci, ou encore sa complexité potentielle. C'est évidemment le premier paramètre sur lequel les utilisateurs de réseaux ont joué pour contrôler les performances d'un système [63]. La démarche la plus évidente pour choisir la meilleure architecture est bien entendu de tester plusieurs modèles différents, changeants les types de neurones, le nombre de couches, le nombre de neurones cachés. Cependant, l'évaluation comparative des réseaux ainsi créé pose problème, de nombreuses méthodes existant mais étant beaucoup trop lourdes en calculs. Pour cette raison, la communauté de réseaux de neurones a adopté des procédures sous-optimales. La plus courante consiste en l'utilisation d'un ensemble de validation, le réseau offrant les erreurs les moindres sur cet ensemble étant considéré le meilleur. Cette méthode est également coûteuse en temps de calculs et soumise à de nombreux aléas. Les principaux algorithmes qui ont été proposés peuvent être classés en cinq familles [64]:

- i. Les méthodes de *pénalisation* (*weight decay*) : consistent à modifier la fonction coût de manière à pénaliser les poids ou les unités peu utiles pour le réseau, voire nuisibles au bon fonctionnement du réseau.
- ii. Les méthodes d'*injection de bruit* : en général, ces méthodes ajoutent une quantité de bruit aux vecteurs d'entrée avant de les présenter au réseau pour l'apprentissage (Grandvalet et al., 1997) [65].
- iii. Les algorithmes d'*élagage*<sup>2</sup> (*pruning*): détectent et éliminent les poids ou les unités qui contribuent peu à la performance du réseau.

<sup>2</sup> L'élagage est l'opération de suppression de connexions ou d'unités dont la présence est jugée inutile. Plusieurs méthodes existent pour l'élagage des poids dans un PMC, comme la méthode Optimal Brain Damage (OBD), proposée par Le Cun et al.[67], et la méthode Optimal Brain Surgeon (OBS) proposée par Hassibi et Stork [61], une extension de la méthode OBD est dans [68]. Elles permettent d'aboutir à une réalisation optimale du réseau de neurones.



- iv. 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 (Lengellé et Denoeux, 1996) [66].
- v. Enfin, les algorithmes *directs* définissent une architecture convenable puis réalisent l'apprentissage ou effectuent les deux opérations en même temps.

Les algorithmes de pénalisation et d'élagage sont appelés algorithmes *destructifs* ou *descendants*. Dans certains articles, les méthodes de pénalisation sont considérées comme des algorithmes d'élagage : La plus populaire des techniques, appelée OBD, consiste en deux phases, entraîner le réseau jusqu'à l'atteinte d'un minimum local, puis éliminer les poids d'utilité faible. L'utilité d'un poids  $w_i$ , dans l'OBD est définie comme étant la variation du coût résultant de sa suppression qui correspond à poser  $w_i = 0$ . Par quelques approximations, on obtient que l'utilité d'un poids  $w_i$  est donnée par :  $\Delta Q(w_i) = \frac{1}{2} w_i^2 \frac{\partial^2 Q}{\partial w_i^2}$ , où  $Q$  est la fonction de coût à minimiser. Comparée à un seuil, on décide si un certain poids est utile ou non. Cette procédure appliquée plusieurs fois peut conduire à une réduction importante du nombre de poids du système sans baisse de performances.

### 7.3. Stopper l'apprentissage avant convergence

Trop apprendre à partir d'un ensemble limité de données d'apprentissage nuit aux performances de généralisation du réseau. C'est le phénomène du sur-apprentissage : L'erreur calculée sur l'ensemble d'apprentissage décroît d'une manière continue et se stabilise ensuite, mais l'erreur de test passe par un minimum avant de croître. Il faut donc stopper l'apprentissage là où il produira les meilleures performances de généralisation. La sélection du meilleur moment pour stopper l'apprentissage pose quelques problèmes. L'utilisation d'un ensemble de validation reste une des techniques les plus utilisées. Il suffit de stopper l'apprentissage avant que l'erreur sur l'ensemble de validation augmente.

### 7.4. Régularisation

Les techniques d'apprentissage des réseaux de neurones sont principalement basées sur l'optimisation, à savoir, la recherche de l'erreur, du risque ou du coût minimal, selon les cas. Les méthodes de régularisation visent à modifier les fonctions de coûts naturelles en ajoutant des termes supplémentaires aux critères des fonctions de coûts. La nouvelle fonction de coût  $Q$ , est représentée comme suit :  $Q = Q_1 + \lambda Q_2$ , où  $Q_1$  est la fonction de coût naturelle modifiée,  $\lambda$  le paramètre de pondération de  $Q_2$  et  $Q_2$  le terme supplémentaire. Ici, on montre

le cas d'un seul terme ajouté à la fonction de coût de base, mais plusieurs autres termes peuvent être ajoutés. Par exemple, le choix de  $Q_2 = \sum_i w_i^2$  aurait pour effet de contraindre les poids à tendre vers 0. L'utilisation de ce terme correspond à la technique connue sous le nom de *pénalisation* (*weight decay*), est très facile à implanter à l'aide de l'algorithme de RP du gradient. Cette technique vise à mettre à zéro les poids inutiles au réseau. Une variation du terme précédent a été proposé par *Weigend* [69], qui permet de faire tendre vers zéro les poids faibles, alors que les autres ne seront pas soumis à cette contrainte :  $Q_2 = \sum_i \frac{w_i^2/K}{1 + w_i^2/K}$ , où  $K$  est une constante, permettant de différencier les poids faibles des poids forts. Ces deux termes sont plutôt rustiques, étant donné que les poids d'un réseau peuvent avoir des significations différentes, ce qui nécessite des contraintes différentes.

### 7.5. Bruitage

Le bruitage des données est une autre technique empirique qui permet d'augmenter la qualité de la généralisation. Elle consiste à ajouter un léger bruit à chaque vecteur d'entrée pendant la phase d'apprentissage, la sortie désirée demeurant inchangée. Cela permet d'assurer un lissage de la fonction apprise par le PMC autour des points d'apprentissage. Cette technique favorise les solutions qui sont peu sensibles à des variations de la taille du bruit appris sur les entrées, donc elle est inutile dans le cas des systèmes chaotique dû au SCI.

## 8. Optimisation

Les premières tentatives de minimisation de fonctions sont attribuées à *Cauchy*, en 1847, qui a utilisé une méthode dite de la plus grande pente. Cette méthode a été largement modifiée par la suite. *Fletcher*, en 1975 a défini l'optimisation comme étant *la science qui détermine la "meilleure" solution d'un problème mathématique* [70]. Dans la littérature, différentes méthodes d'optimisations sont proposées. On distingue deux familles principales des méthodes d'optimisation : Les méthodes d'optimisation qui se basent sur le calcul du gradient de la fonction à minimiser et les méthodes stochastiques (n'utilisent pas le calcul de gradient). Ces dernières constituent, actuellement, un grand champ d'applications dans le domaine de l'engineering.

Les méthodes du gradient sont classiques en optimisation, comprenant principalement les méthodes du *gradient à pas optimal*, du *gradient conjugué* et de *Gauss-Newton*. Les méthodes dites de *Newton* nécessitent la détermination de la matrice *Jacobienne* et la

matrice *Hessienne* de la fonction à minimiser. La matrice Hessienne n'est pas toujours facile à inverser, pour cette raison, ils existent essentiellement deux méthodes dites de *Quasi-Newton* : la première est celle de BFGS (*Broyden-Fletcher-Goldfarb-Shanno*) et la deuxième dite DFP (*Davidon-Fletcher-Powell*) [71]. Ces méthodes sont des méthodes d'ordre 1 qui tentent, par des approximations successives de la matrice Hessienne, de retrouver la méthode de Newton.

Les méthodes stochastiques ou méthode d'ordre zéro (d'exploration directe), utilisent l'espace des paramètres par l'évaluation des valeurs de la fonction objectif au cours des itérations. On peut citer la méthode d'exploration au hasard de Monte-Carlo, les méthodes génétiques et les RNA. Bien que cette dernière utilise l'une des méthodes d'optimisation à base de calcul de gradient pour l'ajustement des paramètres du modèle RNA. Toutes ces méthodes sont caractérisées par des schémas itératifs, exigeant une définition d'un point d'initialisation permettant à l'algorithme de progresser en engendrant une suite de points qui présentent des améliorations successives de la solution.

### 8.1. Problème d'optimisation

Un problème d'optimisation consiste à trouver parmi un ensemble de solutions potentielles celle qui répond le mieux à certains critères décrits sous forme d'une fonction objectif à maximiser ou minimiser.

Mathématiquement, un problème d'optimisation s'écrit sous la forme :

$$\begin{aligned}
 & \text{Minimiser } f_i(x), i = (1, \dots, M) \\
 & x \in \mathfrak{R}^n \text{ avec } x = (x_1, x_2, \dots, x_n) \\
 & \text{Sujet à} \\
 & Y_j(x) = 0, j = (1, \dots, J) \\
 & \Psi_k(x) \leq 0, k = (1, \dots, K)
 \end{aligned} \tag{II.7}$$

où  $x$  est un vecteur à éléments réels ou entiers de dimension  $n$ . Ses composantes  $x_i$  sont appelées variables de décision, elles peuvent être continues ou discrètes, dans ce dernier cas, nous parlerons alors d'optimisation combinatoire. Les valeurs possibles des  $x_i$  représentent l'espace de recherche noté  $S$  dans  $\mathfrak{R}^n$ . La fonction  $f_i(x)$  avec  $i = (1, 2, \dots, M)$  est appelée la fonction coût ou la fonction objectif. Elle attribue à chaque instance de l'espace de recherche une valeur. Les inégalités  $Y_j$  et  $\Psi_k$  sont appelées les contraintes du problème, qui peuvent être linéaires ou non linéaires.

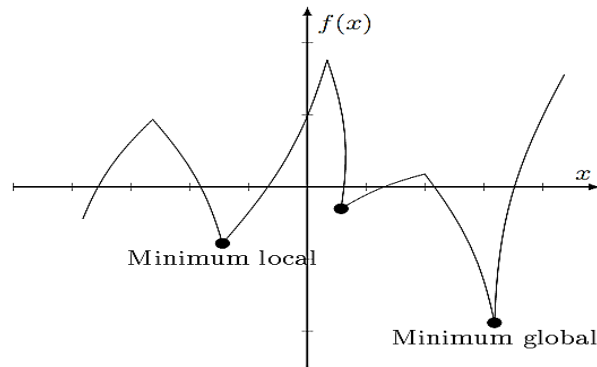
Pour un problème de minimisation, l'objectif est de rechercher les vecteurs qui minimisent la fonction  $f$  tout en respectant les contraintes représentées par les fonctions  $Y_j$  et  $\Psi_k$ . Si  $M = 1$ , le problème est un problème d'optimisation mono-objectif et si  $K = J = 0$ , le problème étudié serait un problème d'optimisation sans contraintes. Dans ce type de problèmes, l'optimisation peut s'effectuer en tout point de l'espace de recherche.

Un problème d'optimisation se définit alors comme la recherche de l'optimum d'une fonction donnée. Cette fonction a des optima locaux et globaux (cf. Figure. II.5).

- Un point  $x^*$  est appelé minimum global de la fonction  $f$  si :

$$\forall x, x \neq x^* \Rightarrow f(x^*) < f(x) \tag{II.8}$$

Il est à noter que pour les problèmes de maximisation, il suffit de multiplier la fonction coût par (-1).



**Figure II.5** Optimum local et global d'une fonction [72].

### 8.2. Méthodes d'optimisation

Généralement, pour résoudre un problème d'optimisation combinatoire, nous utilisons les méthodes exactes mais lorsque nous sommes confrontés à un problème difficile nous avons recours aux méthodes approchées, dans ce cas le choix est parfois possible entre une heuristique spécialisée, dédiée au problème considéré, et une méta-heuristique qui est une méthode générale [73].

Parmi les métaheuristiques, nous pouvons différencier les métaheuristiques à base de voisinage, et les métaheuristiques à base de population et enfin les méthodes hybrides qui associent souvent une métaheuristique et une méthode locale.

### 8.2.1. Méthodes exactes

Les méthodes exactes sont des méthodes qui garantissent la complétude de la résolution. Autrement dit ces méthodes donnent à tous les coups la solution optimale. Le temps de calcul nécessaire de telles méthodes augmente en général exponentiellement avec la taille du problème à résoudre. Il faut être conscient que ces méthodes peuvent prendre beaucoup de temps, surtout lorsque les problèmes sont de grande taille [72].

### 8.2.2 Méthodes approchées

Contrairement aux méthodes exactes, les méthodes approchées ne procurent pas forcément une solution optimale, mais seulement une bonne solution (de qualité raisonnable) en un temps de calcul aussi faible que possible. Une partie importante des méthodes approchées est désignée sous le terme de métaheuristiques qui sont des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé [73].

Plusieurs classifications des métaheuristiques ont été proposées ; la plupart distinguent globalement entre trois catégories : les méthodes à base de solution courante unique, qui travaillent sur un seul point de l'espace de recherche à un instant donné, appelées méthodes à base de voisinage comme la méthode de la descente, le recuit simulé et la recherche tabou, et les méthodes à base de population, qui travaillent sur un ensemble de points de l'espace de recherche, comme les algorithmes évolutionnaires et les colonies de fourmis. Enfin, les méthodes hybrides : leur principe consiste à combiner des algorithmes exacts et/ou des algorithmes approchés pour essayer de tirer profit des points forts de chaque approche et améliorer le comportement global de l'algorithme [73]. (cf. Figure. II.6).

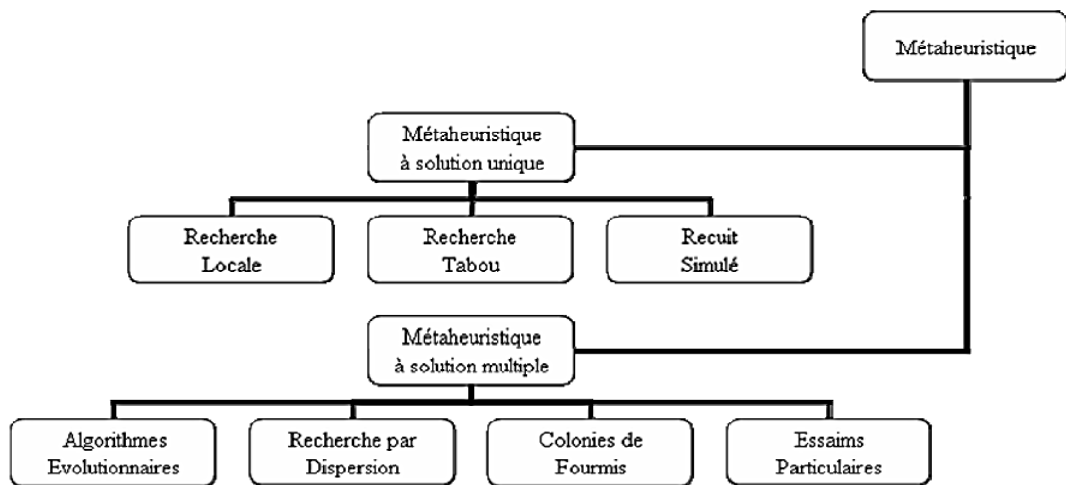


Figure II.6 Classification des métaheuristiques [72].

**a. Méthodes à base de voisinage**

Sur un espace de recherche donné, une petite perturbation sur un point de cet espace induit souvent une petite variation des valeurs de la fonction objectif en ce point. Sachant que les bonnes solutions sont souvent dans d'autres voisinages, l'idée est qu'une bonne stratégie consisterait à se déplacer à travers l'espace de recherche en effectuant de petits pas dans des directions qui améliorent la fonction objectif. Ces méthodes s'appuient toutes sur un même principe : elles résolvent le problème d'optimisation de manière itérative. Elles débutent avec une solution initiale (souvent aléatoire dans l'espace de recherche), et réalisent ensuite un processus itératif qui consiste à effectuer un mouvement choisi par le mécanisme d'exploration en tenant compte de la fonction coût. Ce processus s'arrête et retourne la meilleure solution trouvée quand la condition d'arrêt est réalisée. Cette condition peut porter sur le nombre d'essais effectués, sur une limite temporelle ou sur le degré de qualité de la meilleure configuration courante.

Les méthodes de voisinage diffèrent essentiellement entre elles par le voisinage utilisé et la stratégie de parcours de ce voisinage [73].

**b. Méthodes à base de population**

Dans le domaine de l'optimisation combinatoire, la complexité des phénomènes naturels a servi de modèle pour des algorithmes toujours plus sophistiqués constituant ainsi la base d'un nouveau champ de la programmation informatique en pleine effervescence.

Les méthodes à base de population comme leur nom l'indique, travaillent sur une population de solutions et non pas sur une solution unique comme dans les méthodes à base de voisinage. Nous pouvons distinguer deux grandes classes de ces techniques :

- Les algorithmes évolutionnaires inspirés des concepts issus de la théorie de l'évolution naturelle de Darwin, un exemple typique de ces algorithmes nous trouvons les algorithmes génétiques.
- Les algorithmes inspirés de l'éthologie (l'étude du comportement des diverses espèces animales) comme les colonies de fourmis [73].

**c. Méthodes hybrides**

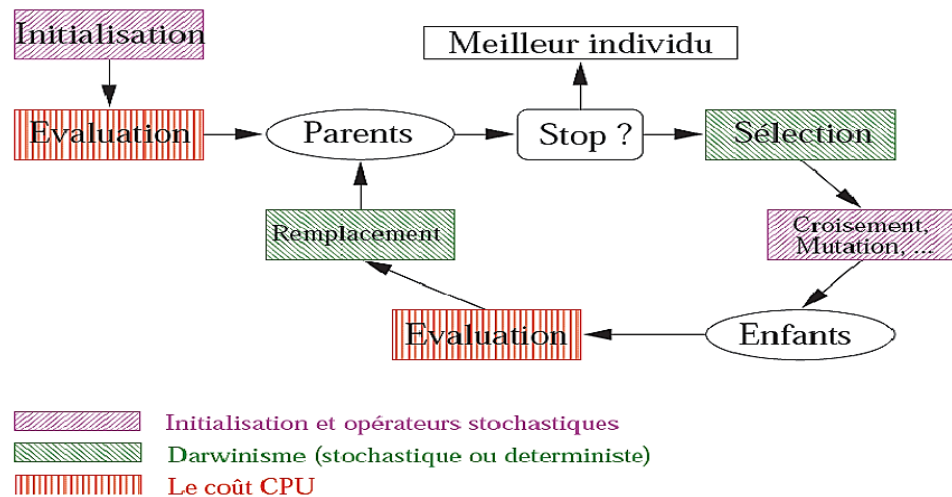
Le mode d'hybridation qui semble le plus fécond concerne la combinaison des deux méthodes précédentes. L'idée essentielle de cette hybridation consiste à exploiter pleinement la puissance de recherche des méthodes de voisinage et de recombinaison des algorithmes

évolutifs sur une population de solutions. Un tel algorithme utilise une ou plusieurs méthodes de voisinage sur les individus de la population pendant un certain nombre d'itérations ou jusqu'à la découverte d'un ensemble d'optima locaux et invoque ensuite un mécanisme de recombinaison pour créer de nouveaux individus [74].

### 9. Algorithmes évolutionnaires

Les algorithmes évolutionnaires AE (Evolutionary algorithms EA) sont des techniques d'optimisation stochastiques itératives, inspirées de la théorie de l'évolution naturelle de Darwin pour résoudre des problèmes divers, souvent d'optimisation. Ils simulent un processus d'évolution sur une population d'individus, dans le but de les faire évoluer vers les optima globaux du problème considéré [72].

Le vocabulaire utilisé dans les AE est directement inspiré de celui de la théorie de l'évolution et de la génétique. En effet, nous trouvons des mots d'individus (solutions potentielles), de population, de gènes (variables), de chromosomes, de parents, de croisement, de mutations, ...etc., et nous nous appuyons constamment sur des analogies avec les phénomènes biologiques. Il s'agit donc de simuler l'évolution d'une population d'individus diverse à laquelle on applique différents opérateurs d'évolution comme les recombinaisons, les mutations, la sélection, etc. (cf. Figure. II.7).



**Figure II.7** Squelette d'un algorithme évolutionnaire [72].

Le fonctionnement d'un algorithme évolutionnaire repose sur trois composants :

- 1- Une population constituée de plusieurs individus représentant des solutions candidates du problème d'optimisation.

2- Une fonction d'adaptation pour évaluer la performance d'un individu (fonction coût).

3- Un mécanisme d'évolution de la population composé de plusieurs opérateurs de modification et de sélection permettant d'éliminer certains individus et d'en créer de nouveaux.

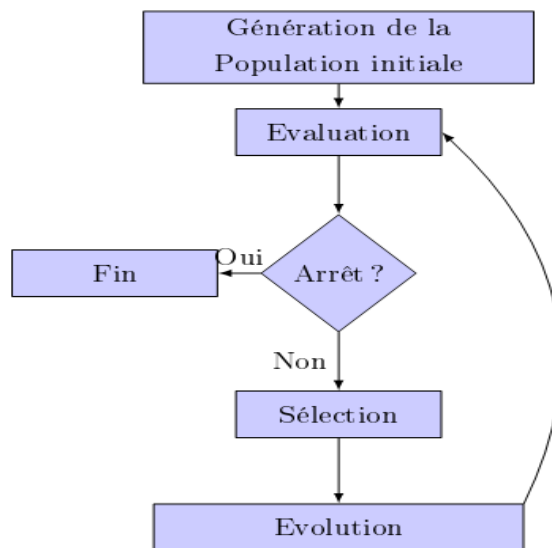
Le cycle d'évolution d'un algorithme évolutionnaire débute avec une population initiale générée de façon aléatoire, puis répète la procédure suivante (cf. Figure. II.8) :

1- Mesurer la qualité de chaque individu de la population par le mécanisme d'évaluation,

2- Sélectionner une partie des individus,

3- Produire de nouveaux individus par recombinaisons des individus sélectionnés (Mécanisme d'évolution).

Ce processus se termine quand la condition d'arrêt est vérifiée [73].



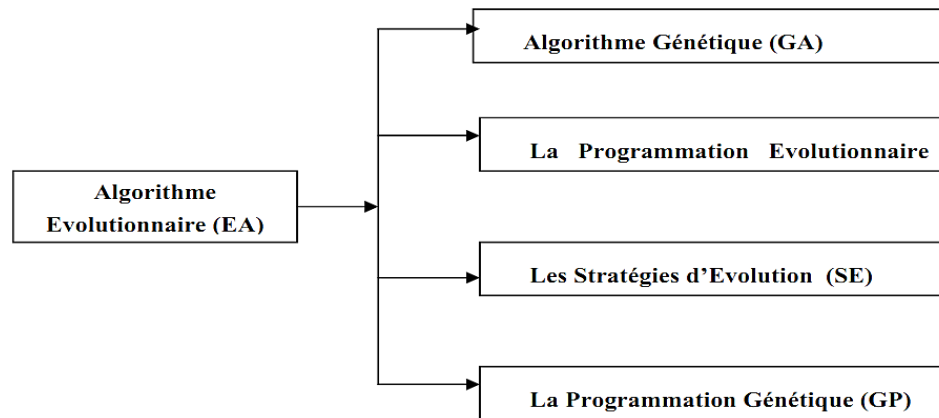
**Figure II.8** Principe général d'un algorithme évolutionnaire.

Un tel algorithme ne nécessite aucune connaissance du problème. En effet, on peut représenter celui-ci par une boîte noire comportant des entrées (les variables) et des sorties (les fonctions objectifs). L'algorithme ne fait que manipuler les entrées, lire les sorties, manipuler à nouveau les entrées de façon à améliorer les sorties, etc.

D'une manière générale, nous pouvons distinguer quatre grandes classes d'algorithmes évolutionnaires (cf. Figure. II.9): les algorithmes génétiques (GA), les stratégies d'évolution (ES), la programmation évolutive (EP), et la programmation génétiques



(GP). Ces méthodes se différencient par leur manière de représenter l'information et par leur façon de faire évoluer la population d'une génération à l'autre.



**Figure II.9** Classes des algorithmes évolutionnaire (EA).

### 9.1 Importance de la diversité

Dans le cas de certains problèmes difficiles, l'algorithme peut très bien ne jamais converger vers une solution satisfaisante. Soit parce que celle-ci est trop difficile à atteindre (temps de convergence infini), soit parce que l'algorithme converge prématurément vers un optimum local et ne peut plus sortir. Pour pallier à ce problème, plusieurs idées ont été implémentées. L'une d'elles est l'introduction d'un objectif de diversité soit en sélectionnant les solutions sur un critère de nouveauté seulement où en optimisant un critère de diversité en même temps que la fitness. Ceci est possible en utilisant un algorithme évolutionnaire multi-objectifs [73].

### 9.2 Exploitation et exploration

Les performances des algorithmes évolutionnaires sont conditionnées par les notions de l'exploitation et de l'exploration et comment trouver un bon compromis entre elles [75].

- L'exploitation (*Intensification*) insiste sur la capacité d'une méthode de voisinage d'examiner en profondeur des zones de recherche particulières.
- L'exploration (*Diversification*) met en avant la capacité de découvrir de nouvelles parties prometteuses de l'espace de recherche.

Les méthodes de voisinage reposent sur l'hypothèse que les zones les plus prometteuses sont situées à proximité des meilleures solutions (déjà visitées). Le principe d'exploitation consiste à examiner en priorité ces zones [75].

Dans les algorithmes évolutionnaires, la sélection a pour effet de concentrer la recherche autour des configurations de meilleure performance.

L'application du seul principe d'exploitation ne permet pas une recherche efficace. En effet, l'exploitation conduit à confiner la recherche dans une zone limitée qui finit par s'épuiser. Une illustration souvent évoquée est fournie par le problème de convergence prématurée des algorithmes évolutionnaires. Du fait de la sélection, la population finit par n'être constituée que d'individus tous similaires. L'une des préoccupations majeures dans les algorithmes évolutionnaires consiste d'ailleurs à préserver le plus longtemps possible une diversité suffisante dans la population. Face à ce type de difficulté, la solution consiste à diriger la poursuite de la recherche vers de nouvelles zones, i.e., recourir à l'exploration [75]. Pour cela, deux stratégies principales sont employées dans le but d'explorer un espace de recherche :

La première stratégie qui est aussi la plus simple consiste à introduire des perturbations aléatoires : c'est le cas pour les mutations aléatoires dans les algorithmes génétiques ainsi que pour la génération aléatoire d'un voisin.

La deuxième stratégie pour explorer consiste à mémoriser au cours de la recherche des caractéristiques des régions visitées et à introduire un mécanisme permettant de s'éloigner de ces zones. C'est ce que fait la méthode tabou avec la liste tabou.

La recombinaison constitue un autre principe général qui complète l'exploitation et l'exploration. Elle consiste à construire de nouvelles configurations en combinant la structure de deux ou plusieurs bonnes configurations déjà trouvées [75].

*Remarque II.4 : Pour les problèmes discrets, de nombreuses heuristiques ont été développées pour réduire les temps de calcul et approcher au mieux l'optimalité : une heuristique procède d'une méthode algorithmique d'approximation qui permet de fournir au plus en temps polynomial une solution pas nécessairement optimale au problème posé. De plus les heuristiques sont spécifiques à un type de problème particulier. Pour les problèmes à variables continues, des méthodes d'optimisation globale existent mais elles deviennent inefficaces dès que certaines propriétés de la fonction objectif ne sont plus vérifiées (par exemple si la fonction objectif n'est pas convexe [76].*

### 9.3. Pourquoi le recours à des métaheuristiques (AE)

Dans ces cas d'optimisation difficile, des approches nouvelles apparaissent avec les métaheuristiques à partir des années 80, qui peuvent s'adapter à l'un et l'autre type de problèmes. Elles partent de principes plus génériques que les heuristiques et sont susceptibles de s'appliquer à un cadre plus large de problèmes. On peut les caractériser par les points suivants [76] :

- Elles possèdent au moins une composante stochastique pour assurer l'exploration de l'espace de recherche,
- Elles ne nécessitent que très peu d'information sur la fonction objectif et procèdent dans les problèmes continus par calcul direct des valeurs de la fonction en un point de l'espace de recherche, sans recours au calcul du gradient qui nécessite la connaissance des valeurs de la fonction au moins sur plusieurs points (nécessaire dans toute méthode de type "descente de gradient"),
- Leur principe de fonctionnement s'inspire de divers domaines scientifiques : physique (ex : le recuit simulé), biologie (ex : les algorithmes évolutionnaires dont les algorithmes génétiques), éthologie (les algorithmes par colonie de fourmis, les essaims particuliers),
- Elles ont l'inconvénient d'un temps de calcul élevé pour obtenir des solutions de qualité,
- Bien que plus génériques que des approches classiques, elles demandent une étude approfondie du réglage de leurs paramètres.

Le principe important pour les métaheuristiques et qui fait échouer les approches classiques est d'éviter le piègeage dans des optimums locaux de la fonction objectif. Pour y parvenir, une stratégie principale est mise en œuvre : distribuer la recherche en travaillant sur une population de solutions candidates simultanément. Ce principe est appliqué dans les algorithmes génétiques, les algorithmes de colonies de fourmis ou d'essaims particuliers.

Pour finir, la difficulté réside sur la question du choix et du réglage d'une métaheuristique pour un problème donné. Il semble qu'il n'y ait pas de consensus sur ce point et certaines approches consistent à hybrider les méthodes entre elles de façon à tirer parti des avantages de chaque technique. L'objectif reste dans tous les cas d'obtenir la meilleure solution possible à coût computationnel réduit.

#### 9.4. Algorithme génétique (AG ou GA)

Le AG est une méthode de la recherche globale stochastique (métaheuristique) qui imite le processus de l'évolution naturelle. L'algorithme génétique commence sans connaissance a priori de la solution correcte et dépend tout à fait des réponses de son environnement et des opérateurs de l'évolution (c.-à-d. la reproduction, le croisement et la mutation) pour arriver à la meilleure solution. En commençant par plusieurs points aléatoires indépendants et une recherche en parallèle, l'algorithme évite des minimums locaux et converge pour obtenir une des solutions optimales si elles existent.

Utilisant les AGs pour rendre le contrôleur plus performant, l'AG est initialisé typiquement avec une population aléatoire qui consiste entre 20-100 individus. Cette population est représentée habituellement par un nombre estimé ou une chaîne binaire appelée un chromosome. La performance de chaque individu est mesurée par une fonction de coût. Cette fonction assigne à chaque individu un nombre correspondant appelé l'aptitude. L'aptitude de chaque chromosome est répartie et une stratégie de survie du plus haut coût est appliquée. La valeur de l'erreur est habituellement utilisée pour répartir le coût de chaque chromosome.

##### 9.4.1. Les étapes de l'algorithme génétique

Il y a trois étapes principales pour l'algorithme génétique, la reproduction, le croisement et la mutation :

**La reproduction** : Juste comme dans l'évolution naturelle, pendant la phase de la reproduction la valeur de coût de chaque chromosome est assignée, cette valeur est utilisée dans le processus de la sélection pour fournir les individus de plus haut coût, un chromosome de plus haut coût a une plus haute probabilité d'être sélectionné pour la reproduction. Un exemple d'une technique de la méthode de la sélection commune est la Roulette. Une section de la roue est allouée à chaque individu dans la population, la dimension de la section est proportionnelle au nombre de l'individu. Un compteur est mis en marche et l'individu à qui il est pointé est sélectionné. Cela continue jusqu'à ce que le critère de la sélection soit atteint. La probabilité qu'un individuel soit sélectionné est donc en rapport avec son aptitude, en s'assurant que les individus de plus grand coût laissent places aux membres de la nouvelle génération. Des copies multiples de la même chaînes peuvent être sélectionnées pour la reproduction et les chaînes de plus valeur de coût devraient commencer à dominer.

**Croisement** : Une fois le processus de la sélection est achevé, l'algorithme du croisement est débuté : Les échanges des opérations du croisement certaines parties des deux chaînes sélectionnées dans une offre de capturer les bonnes parties des vieux chromosomes et de créer mieux nouveaux. Les opérateurs génétiques manipulent directement les caractères d'un chromosome, en utilisant la supposition que le gène de certain individu codé produira des individus en meilleure santé.

La probabilité du croisement indique où le croisement est exécuté. La technique du croisement la plus simple est le croisement en un point seul. En utilisant la sélection et le croisement sur une population produira un grand nombre de chaînes différentes. Cependant il y a deux problèmes principaux selon la population initiale choisie

- Il ne peut pas y avoir assez de diversité dans les chaînes initiales pour assurer que le GA explore dans l'espace entier de recherche.
- Le GA peut converger vers une optimisation locale dû à un mauvais choix d'une population initiale.

Ces problèmes peuvent être vaincus par l'introduction d'un opérateur de la mutation dans le GA.

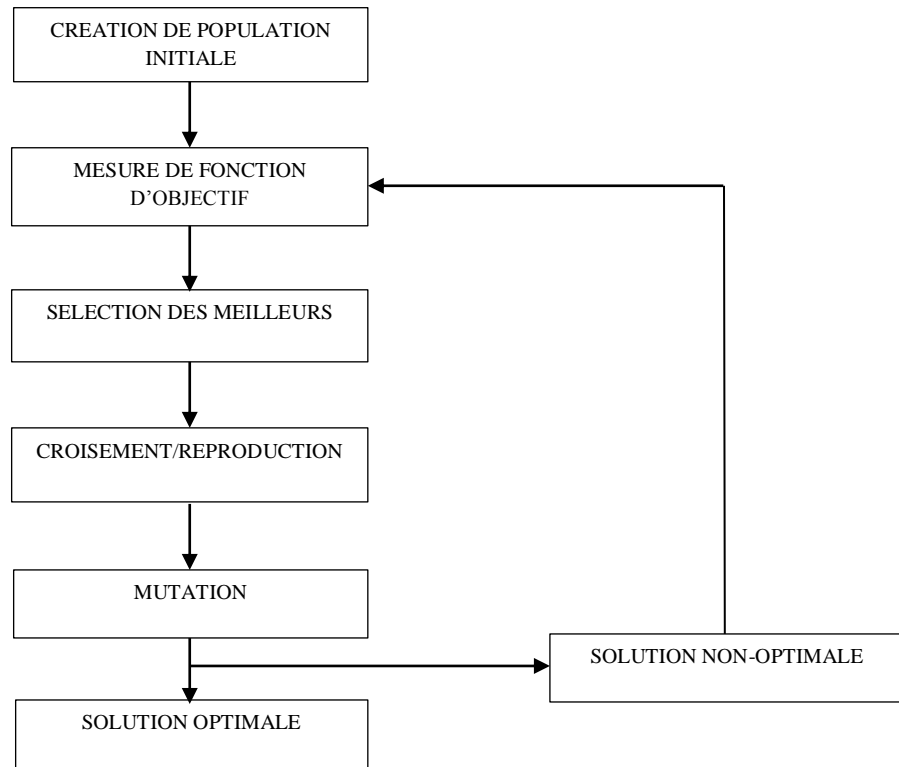
**La mutation** : Elle est la modification aléatoire occasionnelle d'une valeur d'une place de la chaîne. Il est considéré comme un opérateur original dans l'algorithme génétique, la probabilité de mutation est normalement basse parce qu'un haut taux de la mutation détruirait des chaînes de la population et dégénérerait l'algorithme génétique dans une recherche aléatoire. La probabilité de la mutation s'évalue autour de 0.1 ou 0.01. Une fois une chaîne est sélectionnée pour la mutation, un élément est choisi aléatoirement de la chaîne est changé ou a subi une mutation.

Le processus de l'algorithme génétique se résume dans le diagramme suivant :

Les étapes suivies pour la création et pour rendre efficace un algorithme génétique sont comme suit (cf. Figure. II.10) :

- Génération d'une population initiale aléatoire d'individus pour une dimension fixe.
- évaluation de la fonction d'objective pour chaque individu de la population.
- Sélection des membres dont la valeur de la fonction d'objective élevée de la population.
- La reproduction en utilisant une méthode probabiliste par exemple, la roulette.

- Efficacité de l'opération du croisement par un choix probabiliste du point de croisement.
- exécution de l'opération de la mutation avec une probabilité basse.
- répétition des étapes jusqu'à ce qu'un critère de la convergence prédéfini soit atteint.



**Figure II.10** Diagramme illustratif de l'AG

Le critère de la convergence d'un AG est une condition spécifiée par l'utilisateur, généralement le nombre maximal de générations ou quand la valeur de la fonction objectif dépasse un certain seuil.

Les AGs sont substantiellement différents des méthodes de recherche traditionnelle et des techniques d'optimisation en cinq principaux points qui sont :

- Ils cherchent une population de points parallèlement et non pas un point seul.
- Ils n'exigent pas d'information dérivée ou d'autre connaissances auxiliaires seulement une fonction de coût et un niveau d'évaluation pour l'influence de la direction de recherche.
- Ils utilisent des règles de transition probabilistes et pas déterministes.
- Ils travaillent sur un codage d'un ensemble de paramètres, mais pas l'ensemble de paramètres lui-même (excepté que les valeurs réelles des individus sont utilisées).

– Ils peuvent fournir plusieurs solutions potentielles à un problème donné et le choix d'une solution finale est laissé à utilisateur.

Mais les inconvénients majeurs sont :

- Espace mémoire plus grand d'où un temps de plus pour atteindre la meilleure solution.
- L'optimalité locale les algorithmes génétiques.

## 10. Conclusions

Dans ce chapitre, nous avons présenté un état de l'art sur les techniques de l'intelligence computationnelle en particulier les réseaux de neurones et les algorithmes évolutionnaires. Après un aperçu sur les réseaux de neurones, et le rappel de quelques théorèmes relatifs à leurs capacités d'approximation, nous avons présenté un algorithme d'apprentissage très utilisé dans la communauté connexionniste : l'algorithme de rétro-propagation (RP). Dans la pratique, l'objectif d'une modélisation statistique n'est pas d'ajuster finement un modèle sur un ensemble d'apprentissage, mais d'obtenir un bon compromis entre performances d'apprentissage et performances de généralisation. Lors de la construction d'un réseau de neurones, l'objectif donc est de déterminer la meilleure architecture possible et le meilleur jeu de poids au sens d'un critère donné (ex : la généralisation).

Sous cet angle, la construction du réseau de neurones peut être vue comme un problème de recherche d'une solution dans l'espace des architectures et des poids. Le prix à payer pour cette approche est le temps de calcul. En général, on se contente dans la pratique d'utiliser des méthodes qui ne garantissent pas l'optimalité de la solution, mais qui ont l'avantage d'être plus rapides.

---

## Chapitre III

Modélisation et commande des systèmes chaotiques :  
discrets et continus

---

---



## 1. Introduction

Beaucoup de processus réels peuvent être représentés en tant que systèmes dynamiques non linéaires. Un système dynamique non linéaire est décrit par l'évolution *non linéaire* d'un ou de plusieurs variables d'état en réponse à une ou plusieurs variables externes [77].

L'objet de ce chapitre est l'utilisation des RNA pour la modélisation et la commande des systèmes chaotiques, aussi une transmission sécurisée des données est abordée. Les tâches auxquelles ces réseaux sont destinés sont donc essentiellement celles de prédicteurs ou de modèles de simulation des systèmes à commander.

La famille des modèles paramétrés définie précédemment (cf. Chapitre II) présente les attraits suivants :

- Il existe théoriquement toujours un réseau de neurones capable d'approcher avec une précision fixée différentes types de fonctions.
- Il est possible d'estimer les coefficients d'un réseau à l'aide de séquences de couples {entrées-sorties désirées}, disponibles ou déterminés par le concepteur à partir de la tâche à effectuer, de manière à satisfaire un indice de performance mesurant la ressemblance entre les sorties effectives du réseau et les sorties désirées. L'un des intérêts des réseaux de neurones réside dans le fait que cette estimation des coefficients est le résultat d'une procédure algorithmique, l'apprentissage, dont la mise en œuvre obéit à des règles indépendantes de l'architecture et de la complexité du réseau.

Dans le cas où la tâche du RN est une tâche de modélisation, il semble raisonnable de supposer que les sorties mesurées sur le processus obéissent à des lois déterministes, et de chercher une expression mathématique des fonctions à modéliser. La propriété d'approximation universelle est donc une propriété nécessaire du modèle utilisé à cette fin, mais elle n'est pas suffisante. En effet : dans la pratique, les fonctions à déterminer sont définies par un ensemble fini de couples {*entrées-sorties mesurées*}, qui ne permet pas de déterminer ces fonctions de façon univoque ; le but de l'apprentissage est alors de trouver la solution la plus parcimonieuse, passant par tous les points d'apprentissage, qui, si l'ensemble d'apprentissage est bien choisi, tendra vers les fonctions supposées régir le fonctionnement du système.

La modélisation d'un processus consiste à représenter son comportement dynamique à l'aide d'un modèle. Ce modèle sera utilisé pour l'apprentissage d'un correcteur, ou au sein d'un système de commande, ou encore comme simulateur du processus...etc.

La première phase d'une modélisation consiste à rassembler les informations sur le comportement du processus, à partir d'expériences et/ou d'une analyse théorique des phénomènes physiques mis en jeu. Ces informations servent à définir un modèle capable de reproduire la relation non linéaire existante entre les variables de commande, d'état et de sortie.

La deuxième phase de la modélisation consiste à estimer les paramètres du modèle choisi. L'estimation des paramètres du modèle est en générale effectuée en minimisant un critère quadratique définie à partir de l'écart entre les sorties du processus et les sorties du modèle. La qualité de cette estimation dépend de la richesse des séquences d'apprentissage (ou échantillon d'apprentissage) et de l'efficacité de l'algorithme de minimisation du critère quadratique utilisé [50].

Les réseaux de neurones artificiels (RNA) sont utilisés, le plus souvent, dans le cas des processus non modélisés par des modèles mathématiques. En général, le problème qui se pose est le suivant : on dispose d'un ensemble de mesures de variables d'un processus de nature physique, chimique, économique, électronique, etc. et du résultat de ce processus, et on suppose qu'il existe une relation entre ces variables et ce résultat, on cherche alors une forme mathématique de cette relation.

Dans le cas où la tâche du réseau est de réaliser une loi de commande imposant une dynamique désirée à un système pour lequel on dispose d'un modèle, la démonstration de l'existence d'une telle loi de commande est en elle-même un problème. En effet, si la synthèse du contrôleur est effectuée à l'aide d'un modèle neuronal, dont il est difficile de déterminer les caractéristiques de façon analytique, cette existence peut être difficile à établir. Ces problèmes spécifiques à la commande sont abordés au chapitre IV.

## 2. Apprentissage d'un RNA

L'architecture du RN n'est souvent que partiellement imposée par la tâche à réaliser : les entrées, l'état, et les sorties du réseau peuvent être fixées en fonction de celle-ci par le concepteur, ainsi que le type et la connectivité des neurones (comme nous l'avons précisé

précédemment, nous utilisons dans ce travail des réseaux à neurones cachés à fonction d'activation tangente hyperbolique ou sigmoïde complètement connectés). Mais le nombre de neurones ne peut être fixé *a priori*, et il est en général déterminé selon une procédure itérative : test-erreur, suivant le succès de l'apprentissage (il existe des méthodes systématiques de sélection de modèles dynamiques [78]).

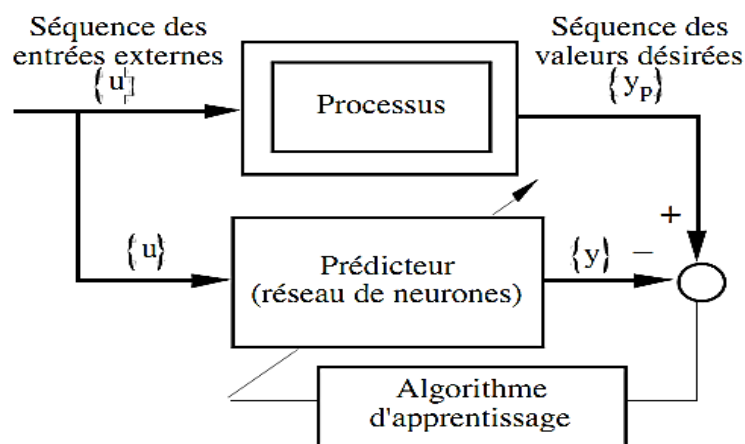
Comme mentionné plus haut, la tâche du réseau est définie par :

- Deux séquences de nombres, une séquence appliquée aux entrées externes, et une séquence de valeurs désirées correspondantes pour les sorties. Ces séquences constituent l'*échantillon d'apprentissage* (apprentissage supervisé).
- Une *fonction de coût* à minimiser : en effet, la tâche ne consiste pas nécessairement à rendre les sorties du réseau égales aux sorties désirées ou proches de celles-ci. Par exemple, pour un problème de régulation, on peut souhaiter minimiser également le coût énergétique de la commande. Le critère fera donc intervenir non seulement l'écart de la sortie à la valeur de consigne, mais également l'énergie dépensée.

L'apprentissage d'un RNA est ainsi défini comme un problème d'optimisation qui consiste à trouver les coefficients du réseau minimisant une fonction de coût :

a) Système d'apprentissage pour une modélisation :

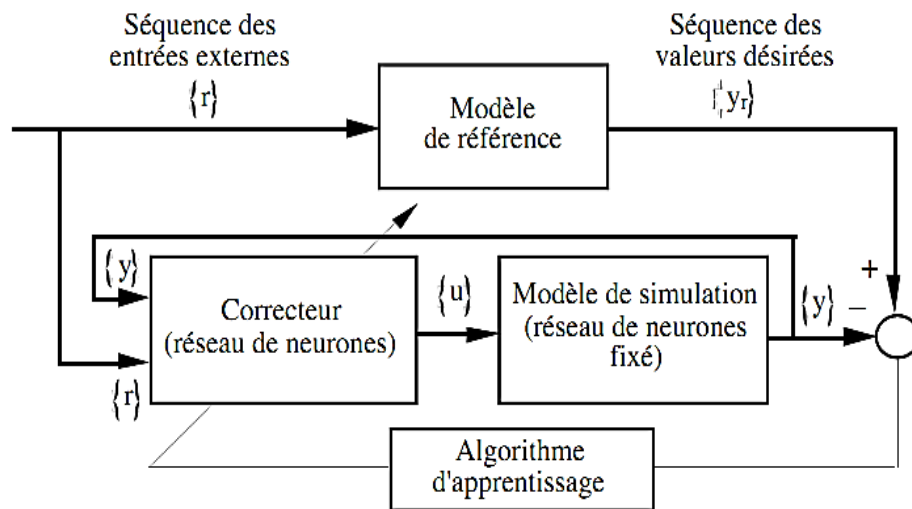
La figure III.1 représente le schéma-bloc d'un système d'apprentissage pour la modélisation d'un processus : le but est d'estimer les coefficients du réseau prédicteur (RN) de façon que ses sorties soient *aussi proches que possible* de celles du processus.



**Figure. III.1** Système d'apprentissage pour la modélisation [39].

b) Système d'apprentissage pour une commande (commande indirecte) :

On dispose d'un modèle neuronal du processus. La séquence des entrées externes est constituée des consignes s'il s'agit de régulation ; par exemple, le cas de stabilisation autour des points d'équilibre. La séquence des sorties désirées pour le système {correcteur + modèle neuronal} est la séquence des sorties d'un modèle de référence dont la dynamique traduit les exigences du cahier des charges sur le comportement en boucle fermée du système de commande, c'est-à-dire du processus réel avec son organe de commande. La figure III.2 représente le schéma-bloc du système d'apprentissage.



**Figure. III.2** Système d'apprentissage pour la commande [39].

Notons que dans le cadre de la commande indirecte, le "réseau" pour lequel la tâche est définie (entrées externes, sorties désirées, et fonction de coût à minimiser) est composé du réseau dont les coefficients sont à estimer, le correcteur, et d'un réseau dont les coefficients sont fixés.

### 3. Modélisation des systèmes chaotiques par les RNA

Puisque nous utilisons des réseaux de neurones, les sorties du système subissant un apprentissage sont en général des fonctions non linéaires des coefficients à estimer. La recherche du minimum de la fonction de coût ne peut s'effectuer à l'aide des moindres carrés ordinaires, et demande donc l'utilisation de méthodes de programmation non linéaire. Nous avons présenté dans le chapitre précédent un algorithme général de calcul du gradient de la fonction de coût dans le cas de l'utilisation de RNA, gradient qui est utilisé soit comme

direction de descente, soit pour calculer une direction de descente permettant une convergence plus rapide (méthode quasi newtonienne ; LM, par exemple).

L'algorithme de calcul du gradient repose sur le calcul des dérivées de la fonction de coût par rapport aux sorties de tous les neurones. Or, le calcul des dérivées par rapport aux sorties des neurones du réseau modèle fournit le *Jacobien* de celui-ci.

### 3.1. Modélisation de l'équation logistique par les RNA

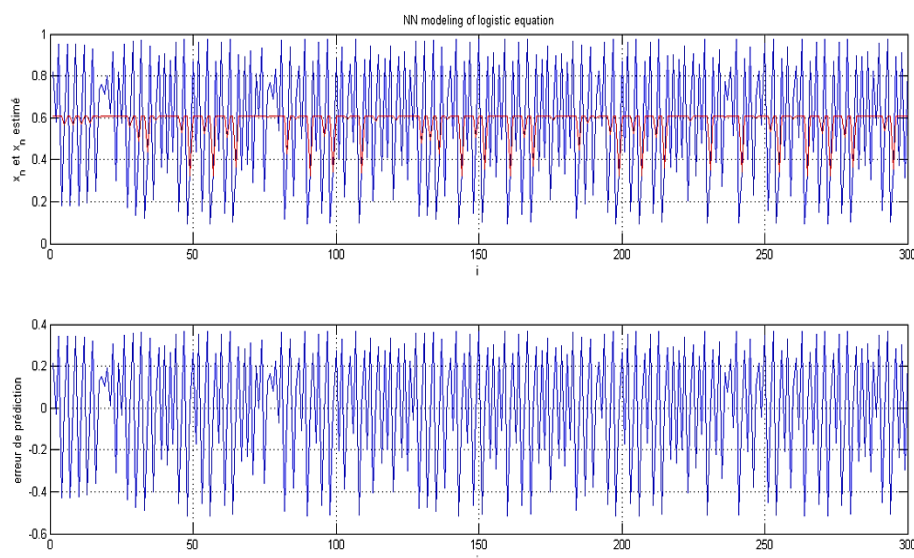
Pour tester l'efficacité et la faisabilité de la méthode de modélisation par les réseaux de neurones d'un système chaotique, on l'a appliquée pour modéliser des données provenant de la simulation de la fonction logistique définie par :

$$x_{i+1} = p x_i(1 - x_i) \quad p \in \mathfrak{R} \quad (\text{III.1})$$

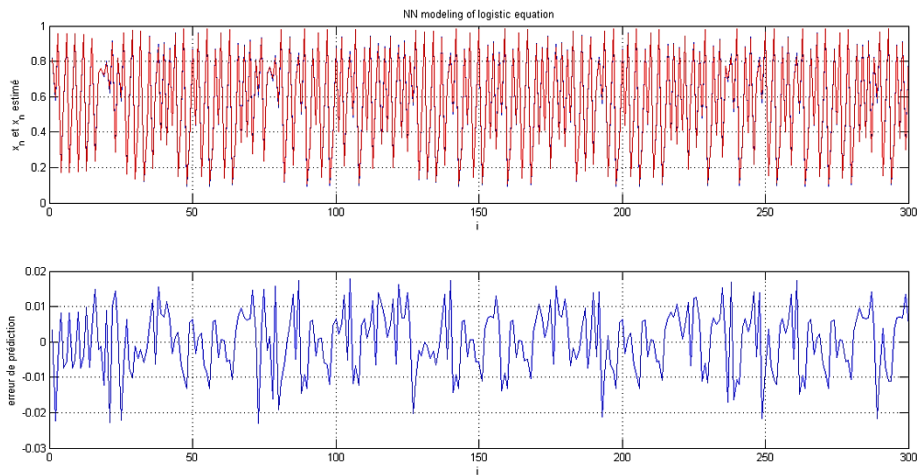
où  $p$  est un paramètre de contrôle.

On considère l'équation (III.1), avec  $x_0 = 0.3$  et  $p = 3.9$  pour ce qui suit.

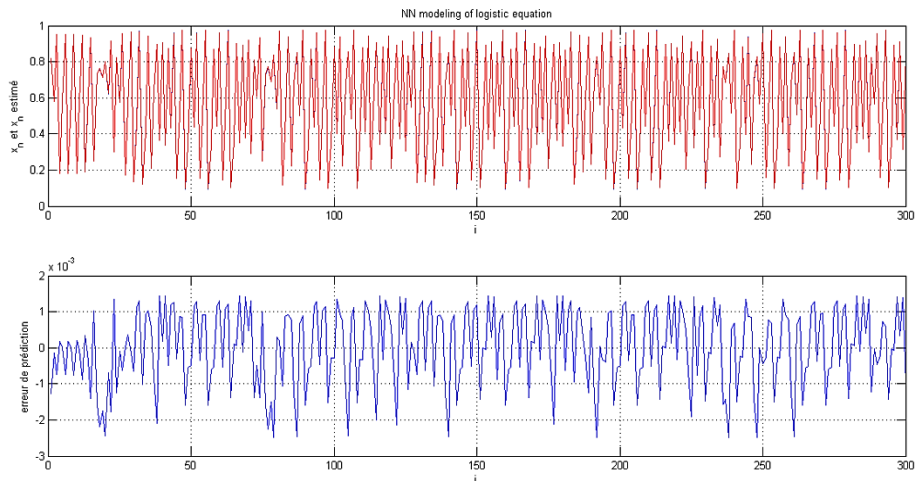
Le réseau de neurones utilisé est constitué d'un neurone d'entrée et un neurone de sortie. Il est entraîné par 300 exemples avec un pas d'apprentissage fixe  $\lambda = 0.01$ . L'entraînement du réseau se fait en changeant le nombre de neurones dans la couche cachée ; on obtient ainsi plusieurs modèles. Les différents résultats de la modélisation sont représentés par les figures III.3 à III.6.



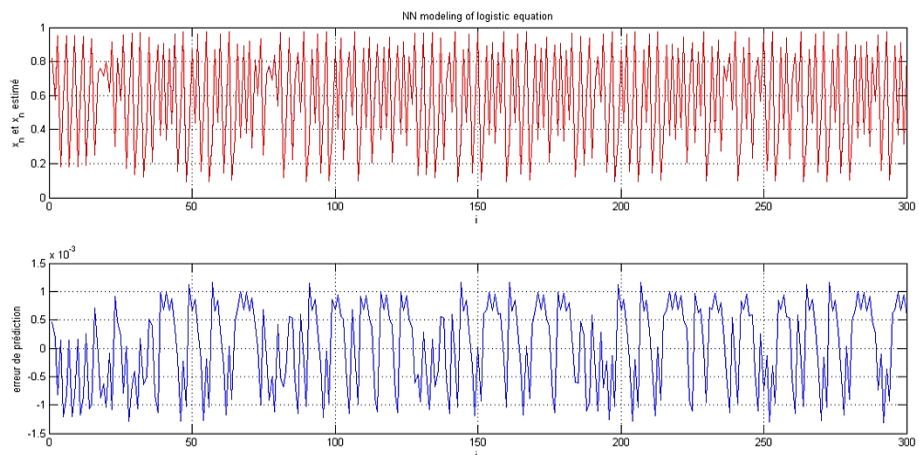
**Figure. III.3** Modélisation de la fonction logistique en utilisant 3 neurones dans la couche cachée.



**Figure. III.4** Modélisation de la fonction logistique en utilisant 4 neurones dans la couche cachée.



**Figure. III.5** Modélisation de la fonction logistique en utilisant 5 neurones dans la couche cachée.



**Figure. III.6** Modélisation de la fonction logistique en utilisant 15 neurones dans la couche cachée.

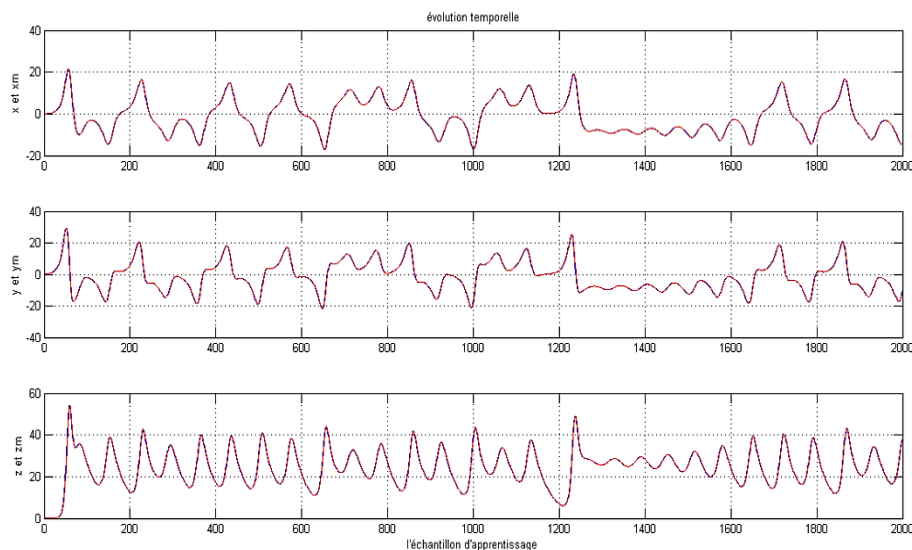
Après plusieurs essais, on remarque que pour l'algorithme de *Levenberg-Marquardt*, les paramètres initiaux du réseau n'ont pas une grande influence sur sa convergence. Par contre, l'augmentation du nombre de neurones dans la couche cachée entraîne la convergence rapide du réseau, mais à partir d'un certain nombre de neurones le réseau converge dans un même nombre d'itération.

### 3.2. Modélisation du système de Lorenz par RNA

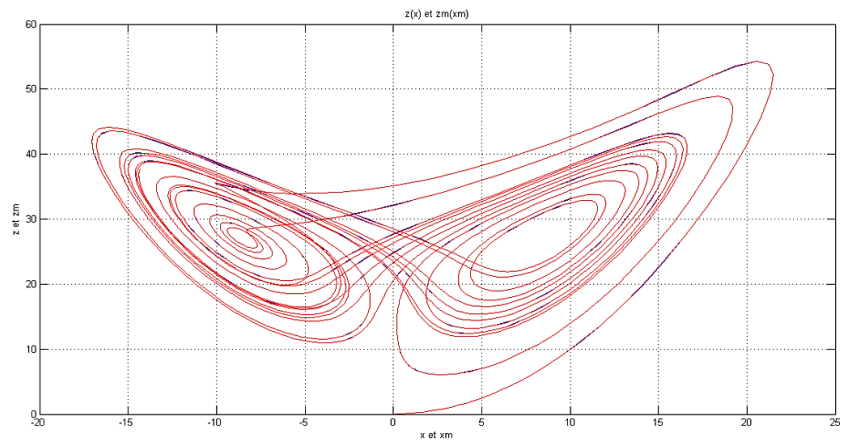
On considère l'exemple du célèbre système de *Lorenz* donné par (eq. I.3), pour des valeurs :  $\sigma = 10$ ,  $r = 28$ ,  $b = \frac{8}{3}$ , et avec la condition initiale  $(x_0, y_0, z_0 = 0.1, 0.1, 0.1)$ , les solutions du système de *Lorenz* semblaient à des oscillations non périodiques et les trajectoires dans l'espace de phase approchaient des ensembles limites (attracteurs) caractérisés par une forme étrange (cf. Figure. I.1).

Pour tester l'efficacité et la faisabilité de modélisation du système de *Lorenz* par les réseaux de neurones, on utilise des données provenant de la simulation des équations (eq. I.3) du système de *Lorenz* (un échantillon = des couples de valeurs d'entrées-sorties).

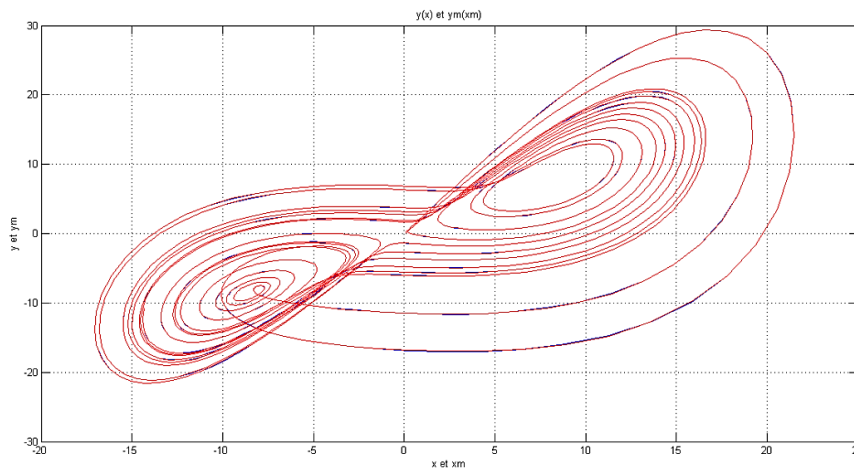
Le réseau de neurones utilisé est constitué de trois neurones d'entrée et trois neurones de sortie. Il est entraîné par 2000 exemples avec un pas d'apprentissage fixe. L'entraînement du réseau se fait en changeant le nombre de neurones dans la couche cachée ; on obtient ainsi plusieurs modèles. L'un des meilleurs résultats de la modélisation est représenté par les figures III.7 à III.11, pour 12 neurones dans la couche cachée.



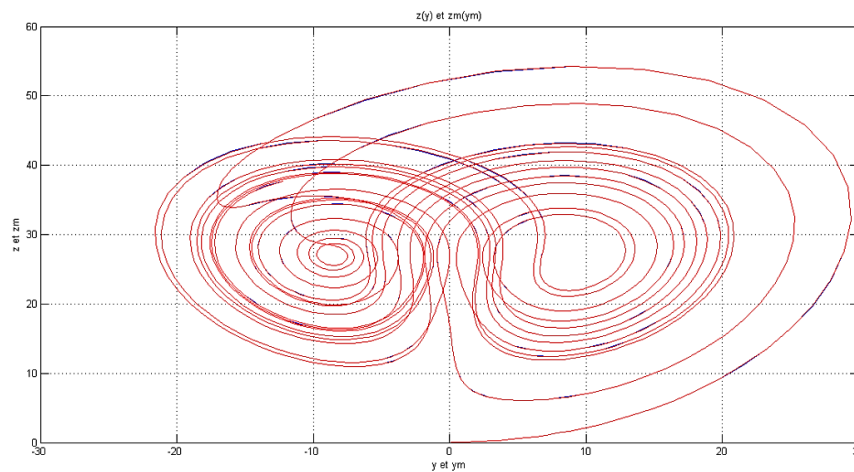
**Figure. III.7** Evolution temporelle des variables d'états du système et du modèle neuronal.



(a)



(b)



(c)

**Figure. III.8** Représentation dans l'espace des phases du système de Lorenz et du modèle neuronal. (a) :  $z(x)$ , (b) :  $y(x)$  et (c) :  $z(y)$ .



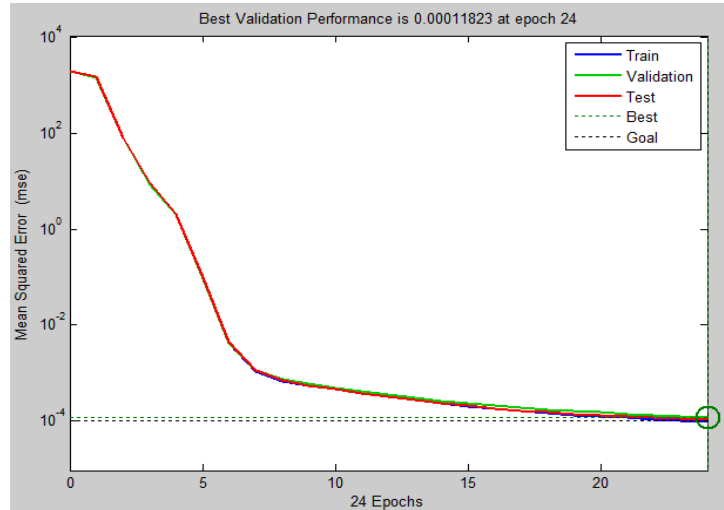


Figure. III.9 Performance du modèle neuronal.

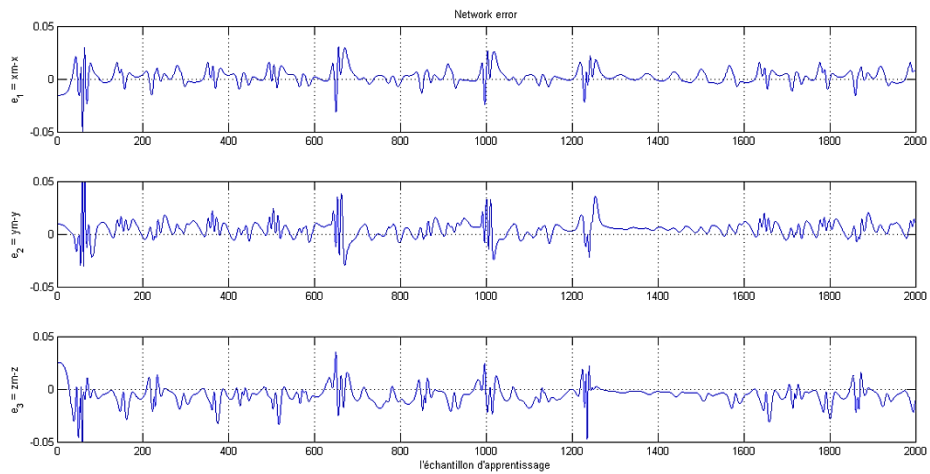


Figure. III.10 Erreurs d'approximation du système de Lorenz par le PMC.

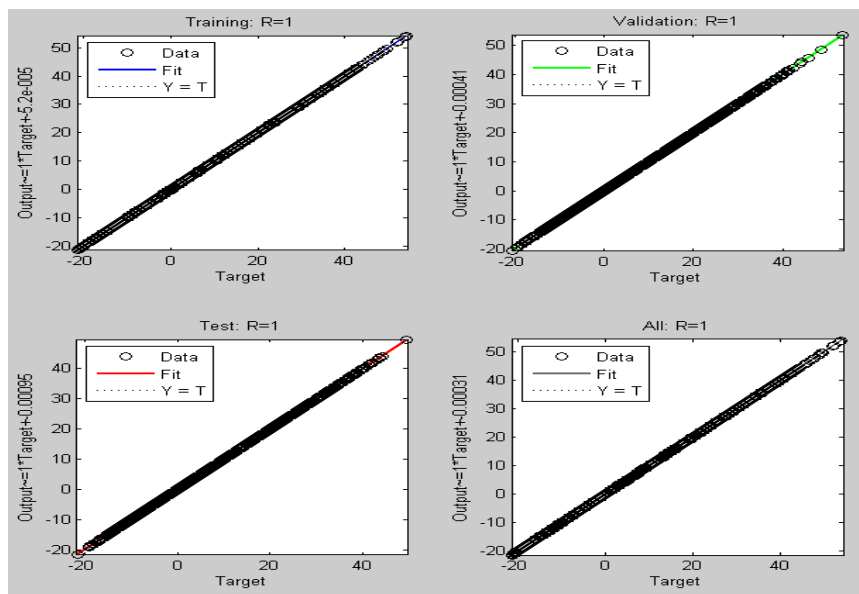


Figure. III.11 Indice de régression du modèle neuronal.

Après plusieurs essais, on remarque que pour l'algorithme de Levenberg-Marquardt, les paramètres initiaux du réseau n'ont pas une grande influence sur sa convergence. Par contre, l'augmentation du nombre de neurones dans la couche cachée entraîne la convergence rapide du réseau, mais à partir d'un certain nombre de neurones le réseau converge dans un même nombre d'itération.

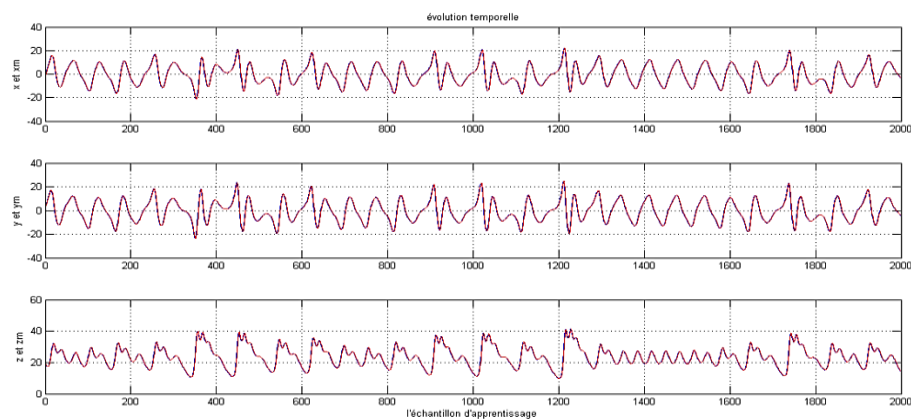
### 3.3. Modélisation du système de Chen par RNA

Le système de *Chen* est un système chaotique continu défini par les équations (III.2) :

$$\begin{cases} \dot{x} = a(y - x) \\ \dot{y} = (c - a - z)x + cy \\ \dot{z} = -bz + xy \end{cases} \quad (\text{III.2})$$

Pour des valeurs :  $a = 35, b = 3, c = 28$ , et avec la condition initiale  $(x_0, y_0, z_0 = -3, 2, 20)$ , les solutions du système de Chen semblaient à des oscillations non périodiques et les trajectoires dans l'espace de phase approchaient des ensembles limites caractérisés par une forme étrange (cf. Fig. III.14).

Pour tester l'efficacité et la faisabilité de modélisation du système de Chen par les réseaux de neurones, on applique la même démarche précédente (pour le système de Lorenz). Le réseau de neurones utilisé est constitué de trois neurones d'entrée et trois neurones de sortie. Il est entraîné par 2000 exemples avec un pas d'apprentissage fixe. L'entraînement du réseau se fait en changeant le nombre de neurones dans la couche cachée ; on obtient ainsi plusieurs modèles. Les résultats de la meilleure modélisation sont représentés par les figures III.12 à III.16 (Le nombre de neurones dans la couche cachée du PMC est 17).



**Figure. III.12** Evolution temporelle des variables d'états du système et du modèle neuronal.

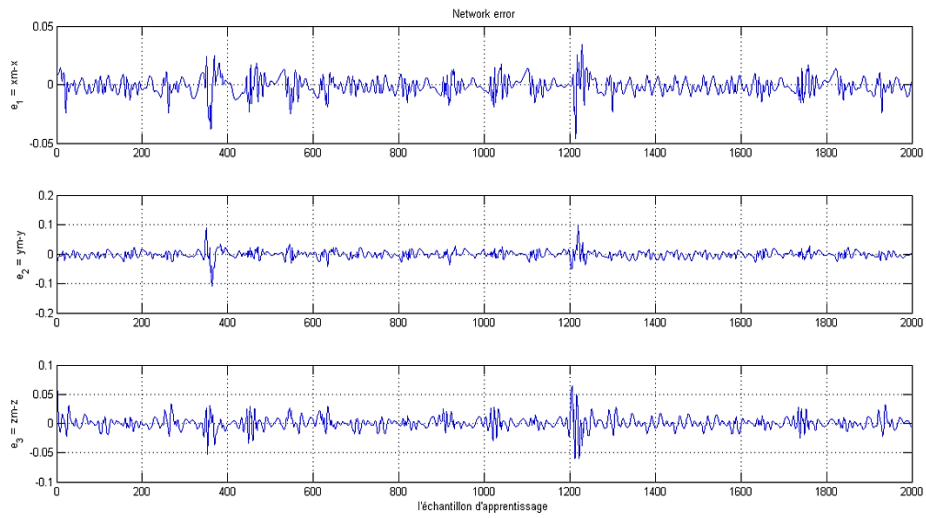
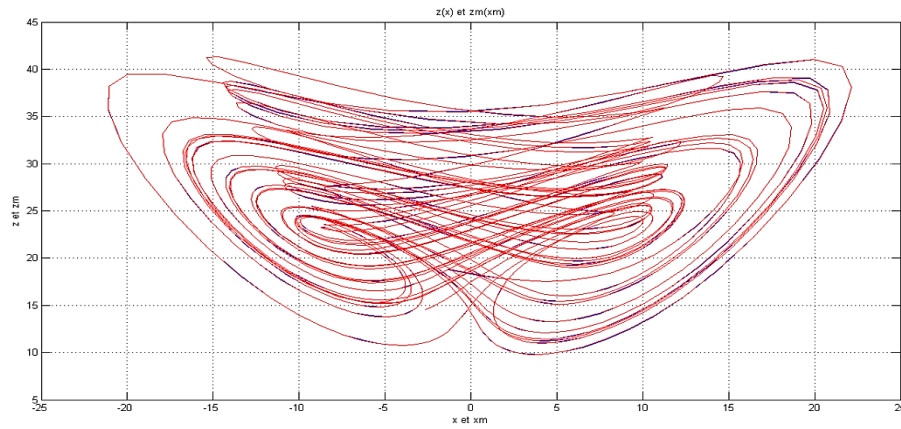
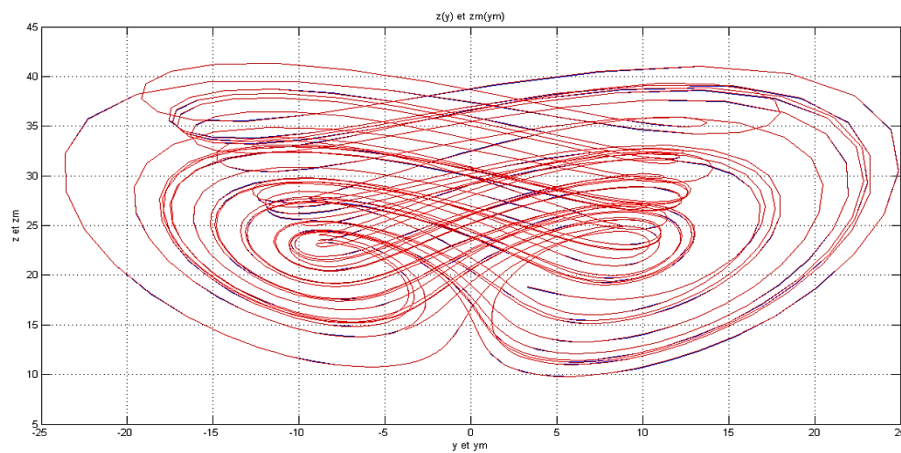


Figure. III.13 Erreurs d'approximation du système de Chen par le PMC.



(a)



(b)

Figure. III.14 Représentation dans l'espace des phases du système et du modèle neuronal de Chen. (a) :  $z(x)$  et (b) :  $z(y)$ .

Après plusieurs essais, on remarque aussi que pour l'algorithme de Levenberg-Marquardt, les paramètres initiaux du réseau n'ont pas une grande influence sur sa convergence. Par contre, l'augmentation du nombre de neurones dans la couche cachée entraîne la convergence rapide du réseau, mais à partir d'un certain nombre de neurones le réseau converge dans un même nombre d'itération.

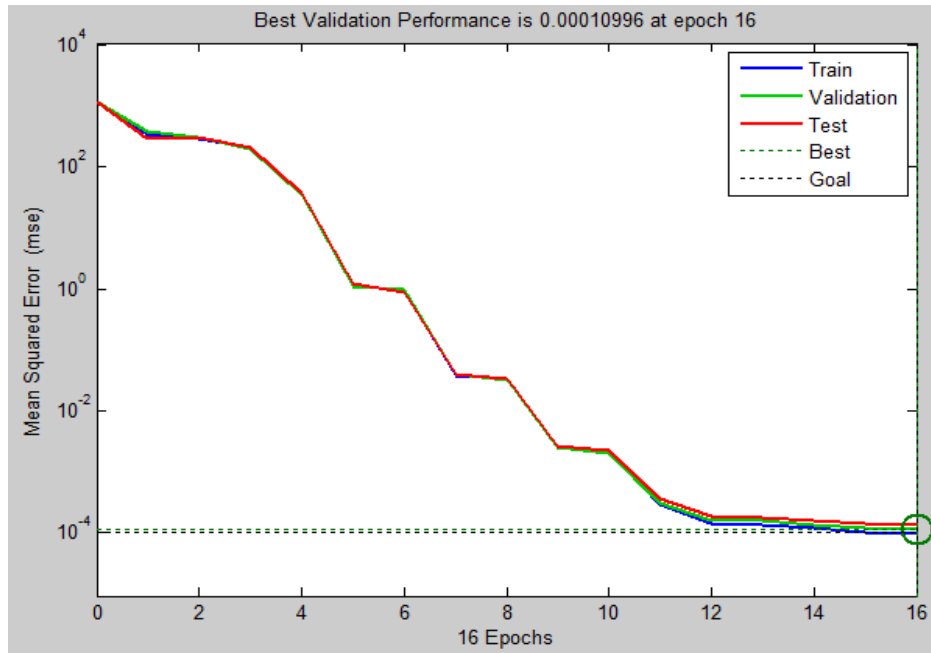


Figure. III.15 Evolution de l'apprentissage du PMC.

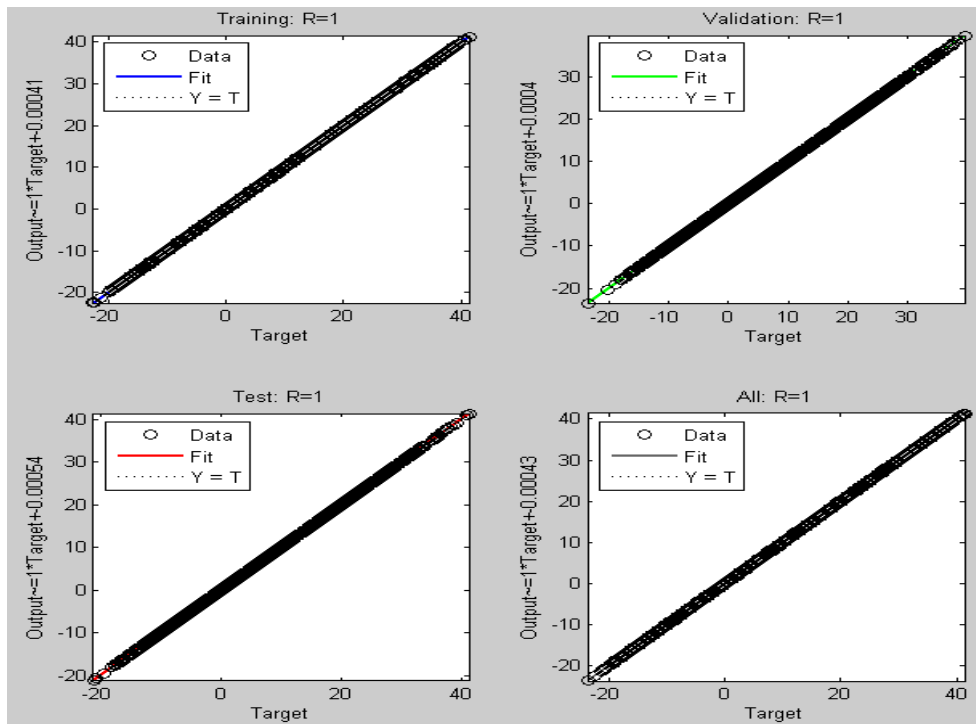


Figure. III.16 Indice de régression du modèle neuronal.

Un modèle du processus est nécessaire à la synthèse de l'organe de commande. La modélisation consiste à rassembler les connaissances que l'on a du comportement dynamique du processus. La structure de modèle dont on fait ainsi l'hypothèse qu'elle décrit correctement le processus est appelée *modèle-hypothèse*. En général, la démarche de modélisation conduit à plusieurs modèles-hypothèse concurrents.

Pour un modèle-hypothèse donné parmi ceux retenus, le but est de configurer et de sélectionner le meilleur modèle parmi les différents modèles de la structure « modèle-hypothèse », sur la base d'un critère de performances. Bien entendu, comme le véritable but de notre démarche est la conception d'un organe de commande à partir d'un modèle, le meilleur d'entre eux est celui qui conduit aux meilleures performances du système de commande. Il est évidemment plus économique, et donc préférable, de se fonder sur un critère qui ne nécessite pas la réalisation complète du système de commande pour sélectionner ce modèle : le meilleur modèle est défini comme celui dont l'erreur de prédiction est la plus faible. Pour obtenir ce modèle, il faut le chercher au sein d'une famille de modèles paramétrés. L'estimation des paramètres est donc effectuée de manière à *minimiser l'erreur de prédiction*, à partir de mesures effectuées sur le processus (ensemble d'apprentissage). Dans le cadre de ce travail, le réseau obtenu en fin d'identification est essentiellement utilisé comme *modèle de simulation* pour la synthèse hors-ligne d'un organe de commande. Une fois avoir le meilleur modèle hypothèse, nous passons à l'étape du contrôle. La section suivante aborde ce problème [79].

#### 4. Problèmes de contrôle des systèmes chaotiques

Les problèmes du contrôle du chaos attiraient l'attention de plusieurs chercheurs et ingénieurs depuis le début de l'année 1990, et plusieurs centaines de publications avaient apparues durant les deux dernières décennies, étant surpris par la découverte de *E. Ott* et ses collaborateurs [3] en 1990, concernant la possibilité de variation des caractéristiques du système dynamique pour une petite variation des paramètres du système, en utilisant le modèle discret de *M. Hénon*, ils démontraient qu'il suffisait une petite variation dans les paramètres du système pouvait transformer les trajectoires chaotiques en une périodique et inversement, ceci a été confirmé expérimentalement par d'autres publications [80] dans une

variété de domaine d'application tel que, les lasers, les systèmes de communications, systèmes chimiques, technologiques et médicales.

La conclusion paradoxale que le chaos est imprédictible mais contrôlable a fait l'objet d'un intérêt immense des chercheurs et un avalanche de publications en utilisant toujours des modèles mathématiques, confirmant la possibilité de variation substantielle des caractéristiques pour une variété des systèmes chaotiques naturels et artificiels par une petite variation relative externe dans ses paramètres [81-85].

La formulation mathématique des problèmes de contrôles des processus chaotiques les plus célèbres sont précédés par la présentation des modèles de base des systèmes chaotiques qui sont souvent utilisés. Les modèles mathématiques les plus connus rencontrés dans la littérature pour le contrôle de chaos sont représentés par des systèmes d'équations différentielles ordinaires ou les équations d'état :

$$\dot{x} = F(x, u) \quad (\text{III.3})$$

où  $x = x(t)$  est le vecteur des variables d'état de dimension  $n$ ,  $u = u(t)$  est le vecteur des entrées (les commandes) de dimension  $m$  et  $F(x, u)$  est le vecteur fonction qui est supposé continu.

Dans la présence de perturbations externes, le modèle non stationnaire est défini par :

$$\dot{x} = F(x, u, t) \quad (\text{III.4})$$

Par conséquent, il est clair que le comportement dynamique d'un système non linéaire peut être changé en variant certaines valeurs de ces paramètres, à condition que ces dernières soient accessibles pour l'ajustement. De ce fait, le contrôle du chaos implique l'extraction de mouvements périodiques désirés en dehors des zones chaotiques, par l'application de petites perturbations judicieusement choisies. La suppression de la dynamique chaotique dans un système dynamique est le seul but pour un problème de contrôle, Dans beaucoup de cas, un modèle de contrôle affine simple (III.5) peut être utilisé.

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ y(t) = h(x(t)) \end{cases} \quad (\text{III.5})$$

La sortie mesurée du système est notée par  $y(t)$ . Elle peut être définie comme une fonction de l'état courant du système. Maintenant, nous procédons à formuler les problèmes de contrôle de processus chaotiques.

#### 4.1. Les problèmes de stabilisation

Les problèmes de stabilisation de la solution périodique instable (orbite) a pour objectif la suppression de bruit ou élimination des harmoniques dans les systèmes de communication, appareils électroniques, etc. Ces problèmes sont dues au fait que le système contrôlé est fortement oscillatoire, c'est-à-dire, les valeurs propres de la matrice du système linéarisé sont proches de l'axe imaginaire, ainsi, ces vibrations peuvent être régulières ou quasi régulières ou même chaotique [83, 84], [86 - 89].

Les problèmes de suppression des oscillations chaotiques ou les réduire aux oscillations régulières ou les supprimer complètement, [90, 91], peuvent être formalisés comme suit :

Si  $x_*(t)$  est une trajectoire oscillatoire périodique du système (III.3) sous la condition initiale  $x_*(0)$  tel que,

$$x_*(t + T) = x_*(t) \quad (\text{III.6})$$

Pour stabiliser ce mouvement on doit ramener la solution  $x(t)$  du système (III.3) vers  $x_*(t)$  c.à.d. :

$$\lim_{t \rightarrow \infty} (x(t) - x_*(t)) = 0 \quad (\text{III.7})$$

où conduire la sortie du système  $y(t)$  vers une fonction donnée  $y_*(t)$  :

$$\lim_{t \rightarrow \infty} (y(t) - y_*(t)) = 0 \quad (\text{III.8})$$

Pour tout solution  $x(t)$  de système (III.3) sous l'état initial  $x_0 \in \Omega$ , où  $\Omega$  est un ensemble des conditions initiales donné, le problème se réduit à déterminer une fonction de contrôle soit comme :

- Une commande en boucle ouverte :  $u(t) = U(x_0, t)$ .
- Ou une commande contre réaction :  $u(t) = U(x(t))$ .

- Ou une commande de sortie en contre réaction :  $u(t) = U(y(t))$ .

qui satisfaisant l'objectif du contrôle.

Cette formulation du problème de stabilisation de solution périodique est similaire au problème de poursuite de la théorie de contrôle conventionnelle. Néanmoins, il existe une distinction fondamentale est que, pour contrôler les processus chaotiques, on a besoin d'atteindre l'objectif avec un niveau de contrôle minimale, la résolution de ce problème n'est pas évidente à cause de l'instabilité des trajectoires chaotique  $x_*(t)$  [3].

La stabilisation d'un point d'équilibre instable est un cas spécial définit par  $F(x_{*0}, 0) = 0$  pour  $u(t) = 0$ . Ainsi, le système (III.3) admet un point d'équilibre  $x_{*0}$  qui doit être stabilisé toujours en choisissant une loi de commande appropriée. Ce problème est caractérisé par une exigence supplémentaire sur le plus petit niveau de contrôle [92].

#### 4.2 Les problèmes du contrôle d'excitation ou de génération d'oscillations chaotiques

La deuxième classe inclut les problèmes du contrôle d'excitation ou de génération d'oscillations chaotiques. Ces problèmes sont aussi appelés la chaotisation ou anti contrôle. Ils surviennent où le mouvement chaotique est le comportement désiré du système.

La forme de l'objectif de contrôle pourrait être représentée comme (III.8), mais ici la trajectoire objective  $x_*(t)$  n'est plus périodique. De plus, il peut être exigé qu'au lieu du mouvement le long de la trajectoire donné, le processus de contrôle satisfait un certain critère de chaotisation.

Par exemple, étant donnée une fonction objective scalaire  $G(x)$ , et le but de contrôle peut être formulé comme :

$$\lim_{t \rightarrow \infty} G(x(t)) = G_* \quad \text{ou} \quad \lim_{t \rightarrow \infty} G(x(t)) \geq G_* \quad (\text{III.9})$$

Pour les problèmes du chaotisation, le plus grand exposant de *Lyapunov*, qui est un critère principal mesurant la dispersion des trajectoires initialement proches caractérisant l'instabilité locale, est habituellement pris comme une fonction d'objective. L'énergie totale mécanique ou électrique des oscillations est prise quelquefois aussi comme  $G(x)$ .

#### 4.3. Les problèmes de synchronisation contrôlable



La troisième classe importante des objectifs du contrôle correspond aux problèmes de synchronisation ou, plus précisément, synchronisation contrôlable. La synchronisation trouve des applications importantes dans la technologie des vibrations (synchronisation d'excitation vibrationnelle), dans la communication (la synchronisation entre le récepteur et l'émetteur des signaux), dans la biologie et la biotechnologie, et ainsi de suite. Nombreuses publications sur la synchronisation contrôlable des processus chaotiques et son application dans les systèmes de la transmission des données sont apparus [93- 95], [85].

Dans le cas général, on sous-entend par la synchronisation la variation des coordonnées des états de deux ou plusieurs systèmes, ou peut-être la variation de quelques caractéristiques telle que les fréquences des oscillations. Dans le cas où la synchronisation ne peut pas être atteinte dans un système sans contrôle ( $u = 0$ ), alors on peut poser le problème de déterminer une loi du contrôle sous laquelle le système de boucle fermée est synchronisé et par conséquent, la synchronisation peut être utilisée comme le but de contrôle de coïncidence totale ou partielle des vecteurs d'état telle que l'égalité :  $x_1 = x_2$ . Qui peut être une expression formelle du mouvement synchrone de deux sous-systèmes avec les vecteurs d'état :  $x_1 \in \mathfrak{R}^n$  et  $x_2 \in \mathfrak{R}^n$ .

L'exigence de synchronisation asymptotique des états  $x_1$  et  $x_2$  des deux systèmes peut être représentée aussi comme :

$$\lim_{t \rightarrow \infty} (x_1(t) - x_2(t)) = 0 \quad (\text{III.10})$$

Le fait que le comportement désiré n'est pas uniquement fixé et ses caractéristiques sont définies seulement partiellement est commun aux problèmes de contrôle d'excitation et de synchronisation des oscillations. Ces buts de contrôle, souvent peuvent être réarrangés en termes plus commode d'une fonction d'objective correspondante :

$$Q(x) = \|x_1 - x_2\|^2 \quad : \quad Q(x) = (x - x_*)^T \Gamma (x - x_*) \quad (\text{III.11})$$

où  $\Gamma$  est une matrice définie positive symétrique. On note ici que le choix d'une fonction d'objective appropriée est une étape importante dans la conception de l'algorithme du contrôle.

Un type important de problèmes de contrôle des processus chaotiques est représenté par la modification des attracteurs, par exemple, transformation des oscillations chaotiques

vers des oscillations périodiques ou vice versa. Le développement des approches aux problèmes de ce genre a été stimulé par des nouvelles applications dans les technologies des lasers et chimiques, dans les télécommunications, la biologie, et la médecine [96], [97-101].

## 5. Motivations du contrôle des systèmes chaotiques

Le contrôle du chaos fait référence à la manipulation du comportement dynamique du système chaotique, dont l'objectif est la suppression du chaos quand il est nuisible ou au contraire le créer et le préserver quand il est bénéfique, plusieurs méthodes et algorithmes ont été proposés et développés pour réaliser le contrôle du chaos dans divers systèmes dynamiques non linéaires. Nombreux articles, livres, et publications sont disponibles sur ce thème. Il est important de préciser que la littérature dans cette matière est très large et que les méthodologies décrites et utilisés dans ce travail ne sont pas les seules valides.

## 6. Applications

Il y a plusieurs approches utilisant les réseaux neuronaux pour le contrôle de processus chaotiques. Beaucoup d'études comptent sur la capacité universelle des RN pour le contrôle et la prédiction du comportement des systèmes non linéaire. Du moment que les systèmes chaotiques sont fondamentalement non linéaires, la potentialité de leur contrôle neuronal n'est pas surprenante [102].

D'autres études décrivent l'identification des systèmes contrôlés par les réseaux neuronaux combinés avec une méthode standard de contrôle de systèmes chaotiques tel que la méthode de OGY [89], contrôleur de la boucle de retour proportionnel [103], cette identification est souvent obtenu grâce à l'apprentissage adaptatif [104], ou par optimisation, l'algorithme génétique [105], etc.

### 6.1. Contrôle des UPO de systèmes chaotiques inconnus par les RN

Dans cette section, un modèle à base de réseaux de neurones (RN) est développé pour la modélisation et le contrôle d'orbites périodiques instables (UPO) des systèmes chaotiques. Durant la phase de modélisation, le RN est entraîné sur le système chaotique inconnu en utilisant les données entrée-sortie obtenues d'un système chaotique inconnu sous-jacent, où un algorithme de calcul spécifique est utilisé pour l'optimisation des paramètres. Durant la

phase de contrôle, le critère de stabilité- $L_2$ , qui est à la base du principe de conception du modèle, est utilisé. Afin de démontrer l'efficacité du modèle de contrôle et du réseau de neurones utilisés, quelques résultats de simulation sur les systèmes chaotiques *Henon* et *Duffing* sont illustrés, pour les deux phases : modélisation et contrôle.

Il existe plusieurs méthodes et techniques qui traitent le chaos d'une manière succincte ces dernières années, dont plusieurs sont l'amélioration ou la généralisation de la méthode OGY [3]. Quelques travaux donnent une généralisation et une explication systématique de la méthode OGY [106-109]. Ces dernières méthodes ont été établies dans le même état d'esprit. Dans un premier temps, une identification de la dynamique chaotique autour d'un point fixe instable ou d'une orbite périodique instable est réalisée. Dans une deuxième étape, en éliminant l'erreur dans la direction instable, la cible désignée (point fixe instable ou orbite périodique instable) est stabilisée.

Par contre, la prédiction du chaos est un autre sujet qui a été développé pendant plusieurs années. Dû à la non-linéarité des systèmes chaotiques, les méthodes d'approximations linéaires sont inadéquates pour la prédiction du chaos. Il y'a une compréhension générale que les approximations locales peuvent souvent donner des résultats de prédictions plus intéressantes qu'avec des approximations globales, d'où le recours aux RN. Donc, l'utilisation des RNA est une méthode prometteuse pour la prédiction du chaos, spécialement quand le modèle du système chaotique est inconnu ou incertain.

Plusieurs méthodes de prédiction du chaos basées sur différents réseaux de neurones ont été proposées, tel que les RN récurrents prédictifs [110], les ondelettes [111], les réseaux fonctionnels [112] et les RN dynamiques à temps de retard (Time delay dynamic neural networks) [113]. Mais ces méthodes ont les mêmes limitations. Parce que le chaos dynamique est identifié seulement autour du point fixe, le contrôle est actif seulement autour de ce point. *Shen* et al. ont proposé dans [114] un réseau de neurones qui peut être divisé en deux parties : une première partie qui effectue une classification de l'entrée, et une deuxième partie comprenant une série de sous-réseaux de neurones qui seront permutés suivant le résultat de classification. En utilisant ce principe, l'entrée de contrôle peut éliminer les erreurs pas seulement dans la direction instable mais aussi dans la direction stable. Le problème essentiel est comment construire la partie classification. *Lin* et al. proposent dans [115] une méthode efficace de contrôle pour les systèmes chaotiques incertains en

utilisant pour la modélisation les RN et la logique floue, et pour le contrôle l'approche adaptative de poursuite '*backstepping tracking control*'. La méthode proposée est très utile pour le contrôle des systèmes chaotiques incertains seulement autour des cibles désirées.

Inspiré par la méthode de contrôle prédictive, nous nous intéressons à la possibilité de développer un modèle neuronal simple qui peut être utilisé pour la prédiction des systèmes chaotiques inconnue ou incertaine et les contrôler pour converger vers leurs orbites périodiques instable en utilisant seulement des données d'entrée-sortie.

### 6.1.1. Modélisation des systèmes chaotiques inconnus par les RN

Pour un réseau de neurones avec des paires d'entrée-sortie  $(x(i), y(i))$ ,  $i = 1, 2, \dots, n$  le nombre d'itération et une seule couche cachée de  $j = 1, 2, \dots, m$  neurons. L'entrée du  $j^{\text{ème}}$  neurone de la couche cachée  $I_j$  est donnée par :

$$I_j = \sum_{i=1}^n w_{ji}x(i) + b_{ji} \quad (\text{III.12})$$

où  $w_{ji}$  est l'interconnexion (poids) entre le  $j^{\text{ème}}$  neurone et la  $i^{\text{ème}}$  entrée de la sortie et  $b_{ji}$  sont des biais. L'entrée  $I_j$  passe alors par une fonction d'activation pour produire une sortie  $O_j$ . Il existe plusieurs fonctions d'activation, la fonction sigmoïde est utilisée dans ce réseau :

$$O_j(x(i)) = \frac{1}{1 + \exp(-b_{ji} - w_{ji}x(i))} \quad (\text{III.13})$$

La sortie du réseau  $y^{NN}(i)$  est obtenue d'une manière similaire à une somme pondérée de sorties :

$$y^{NN}(i) = F(x(i)) = \sum_{j=1}^m w_0 O_j(x(i)) + b_0 \quad (\text{III.14})$$

Le réseau neuronal doit être entraîné de telle sorte qu'il puisse effectuer des tâches de prédiction et de contrôle. Dans un premier temps, les coefficients de pondération du réseau  $w_{ji}$  et  $b_{ji}$  de la couche cachée et  $w_0$  et  $b_0$  de la couche de sortie sont attribués de façon aléatoire. La sortie  $y^{NN}(i)$  est calculée pour chaque entrée associée  $x(i)$ .

Afin d'obtenir le modèle RNA, nous devons déterminer les poids et les biais du réseau  $w_{ji}$  et  $b_{ji}$  de la couche cachée et le poids  $w_0$  et bias  $b_0$  de la couche de sortie. Pour cela, et pour un système chaotique inconnu donné représenté par des paires de données d'entrée et de sortie mesurées, nous cherchons à minimiser une fonction quadratique  $e$  qui quantifie l'erreur entre les données de sortie désirée  $y$  et la sortie du modèle neuronal  $y^{NN}$  pour toutes les paires de données d'entrée-sortie :

$$e(i) = \frac{1}{2} \sum_{i=1}^n (y(i) - y^{NN}(i))^2 \quad (\text{III.15})$$

Ceci est fait en ajustant les différents paramètres du modèle RNA. Si nous désignons le *Jacobien* pour chacun des paramètres  $w_{ji}$ ,  $b_{ji}$ ,  $w_0$  et  $b_0$  par :

$$J_{w_{ji}} = \frac{\partial O_j(x(i))}{\partial w_{ji}}, J_{b_{ji}} = \frac{\partial O_j(x(i))}{\partial b_{ji}}, J_{w_0} = \frac{\partial F(x(i))}{\partial w_0}, J_{b_0} = \frac{\partial F(x(i))}{\partial b_0} \quad (\text{III.16})$$

Ensuite, les poids et les biais peuvent être entraînés en utilisant l'algorithme de Levenberg-Marquardt [116] comme suit :

$$w_{ji}(i+1) = w_{ji}(i) - \left( \lambda I + J_{w_{ji}}^T J_{w_{ji}} \right)^{-1} J_{w_{ji}}^T e(i) \quad (\text{III.17})$$

$$b_{ji}(i+1) = b_{ji}(i) - \left( \lambda I + J_{b_{ji}}^T J_{b_{ji}} \right)^{-1} J_{b_{ji}}^T e(i) \quad (\text{III.18})$$

$$w_0(i+1) = w_0(i) - \left( \lambda I + J_{w_0}^T J_{w_0} \right)^{-1} J_{w_0}^T e(i) \quad (\text{III.19})$$

$$b_0(i+1) = b_0(i) - \left( \lambda I + J_{b_0}^T J_{b_0} \right)^{-1} J_{b_0}^T e(i) \quad (\text{III.20})$$

où le paramètre  $\lambda$  aidera à garder les mises à jour se déplaçant dans la bonne direction et  $I$  représente la matrice d'identité.

Les poids sont ajustés pour minimiser la fonction d'erreur (III.15) en utilisant l'algorithme d'optimisation de Levenberg-Marquardt. Cet algorithme d'optimisation très efficace est capable de converger vers la valeur optimale d'une fonction non-linéaire quadratique, avec une bonne approximation. Les ajustements des poids devraient entraîner une diminution de l'erreur. L'algorithme d'optimisation de Levenberg-Marquardt s'exécute jusqu'à ce que la fonction quadratique d'erreur  $e$  soit minimisée, de sorte que l'algorithme a trouvé un minimum et il peut être terminé; cependant, les paramètres  $w_{ji}$ ,  $b_{ji}$ ,  $w_0$  et  $b_0$  sont réinitialisés.

L'algorithme proposé peut être exécuté à plusieurs reprises pour entraîner le système neuronal à apprendre d'une manière efficace la relation ou bien la fonction non linéaire des paires de données. L'algorithme s'exécute jusqu'à ce que tous les paramètres cessent de changer ou changent très peu au cours d'une série d'étapes de mise à jour. Cela indique que la valeur de l'erreur quadratique (III.15) est minimisée, donc l'algorithme a trouvé un minimum et il peut être s'arrêté.

### 6.1.2. Conception du contrôleur neuronal

L'objectif du contrôle est de stabiliser le chaos sur l'UPO, c'est-à-dire de conduire le système chaotique inconnu d'un régime chaotique à un mouvement régulier. Nous utilisons le modèle neuronal (III.14) pour effectuer le contrôle. Ainsi, le modèle neuronal contrôlé sera donné par :

$$y^{NN} = F(x(i)) + u(i) = \sum_{j=1}^m w_0 O_j(x(i)) + b_0 + u(i) \quad (\text{III.21})$$

Par conséquent, et sur la base du modèle neuronal, il est possible de prédire l'état futur du modèle RN non contrôlé comme suit :

$$F(x(i+1)) = \sum_{j=1}^m w_0 O_j(x(i+1)) + b_0 \quad (\text{III.22})$$

Nous formulons le contrôle d'orbite périodique instable UPO par RNA comme un problème de stabilité- $L_2$  d'entrée-sortie d'un système de retour d'état périodique et linéaire, ce qui permet d'utiliser le critère prédictif par le concept  $L_2$ -setting pour déterminer les limites de stabilité de l'orbite périodique.

En considérant une relation entrée-sortie  $(r_0, v_0) \in L_2 \times L_2$  d'un système dynamique  $\Sigma$ , nous rappelons maintenant les définitions de la stabilité- $L_2$  entrée-sortie [117] :

**Définition III.1 :** La solution  $v_0$  du système  $\Sigma$  est stable au sens  $L_2$ , seulement s'il existe une constante  $\gamma$  tel que :

$$\|v - v_0\|_2 < \gamma \|r - r_0\|_2 \quad (\text{III.23})$$

pour tout  $r, (r - r_0) \in L_2$ .

**Définition III.2 :** La solution  $v_0$  du système  $\Sigma$  est localement stable en  $L_2$ , seulement s'il existe deux constantes positives  $M_v$  et  $M_r$  tel que :

$$\|v - v_0\|_2 < M_v \quad (\text{III.24})$$

pour n'importe quel signal  $r$  qui satisfait :

$$\|r - r_0\|_2 < M_r \quad (\text{III.25})$$

Notre objectif est de concevoir un contrôleur qui stabilise la famille des solutions périodiques correspondant aux orbites périodiques instables UPO originales incorporées dans les systèmes chaotiques inconnus.

Considérons  $(x_d, y_d)$  les états de l'UPO qu'on souhaite stabiliser. Donc, à partir des définitions du critère de stabilité- $L_2$  d'entrée-sortie [117],  $u(i)$  stabilise l'UPO du système chaotique inconnu s'il existe une constante  $\gamma$  telle que :

$$\|x(i+1) - y_d\|_2 \leq \gamma \|x(i) - x_d\|_2 \quad (\text{III.26})$$

La condition (III.26) assure la stabilité- $L_2$  locale d'entrée-sortie du système (III.14) seulement si la trajectoire du système se trouve à proximité des états de l'orbite périodique instable  $(x_d, y_d)$ .

La commande  $u(i)$  est alors formée par les informations de l'orbite périodique instable (UPO) et du modèle neuronal équivalent (modèle hypothèse) au système chaotique inconnu. Dans ce cas, la loi de commande est déterminée par :

$$u(i) = -F(x(i+1)) + F(x_d) = -\sum_{j=1}^m w_0 O_j(x(i+1)) - b_0 + \sum_{j=1}^m w_0 O_j(x_d) + b_0 = \sum_{j=1}^m w_0 [O_j(x_d) - O_j(x(i+1))] \quad (\text{III.27})$$

qui sera appliquée à proximité de l'UPO. Le contrôleur basé sur RN devient ainsi :

pour toute petite valeur réelle positive  $\gamma$  ;

$$u(k) = \begin{cases} \sum_{j=1}^m w_0 [O_j(x_d) - O_j(x(i+1))] , & \text{si } \|x(i+1) - y_d\|_2 \leq \gamma \|x(i) - x_d\|_2 \\ 0, & \text{sinon} \end{cases} \quad (\text{III.28})$$

**Remarque** On note que la stabilisation des systèmes chaotiques sur l'UPO originale est une procédure standard [3], [105-106]. De même, de tels systèmes avec une rétroaction additive de contrôle peuvent également être stabilisés sur les autres périodes (4, 8, ...) ou même sur

des points fixes instables, où les paramètres du contrôleur à base de RN sont modifiés de sorte qu'ils garantissent l'exigence de stabilité.

### 6.1.3. Résultats de simulations

Dans cette section, des simulations numériques des deux systèmes de Henon et de Duffing sont utilisées pour vérifier l'efficacité de la méthode de prédiction et de contrôle du chaos proposée par les RNA. Les paires de données entrées-sorties ont été générées à l'aide des systèmes sous-jacents des deux systèmes, utilisées pour établir les modèles neuronaux des systèmes chaotiques inconnus.

Dans l'exemple un et deux, on considère que les systèmes chaotiques inconnus sont représentés respectivement par  $N = 200$  et  $N = 600$  paires de données d'entrée-sortie. Nous choisissons  $\lambda = 0,01$  pour le paramètre d'apprentissage de l'algorithme de Levenberg-Marquardt et  $\gamma = 0,05$  pour la condition de stabilité- $L_2$ . D'autre part, il est nécessaire de déterminer la taille maximale de la région attractive donnée en (III.22) de sorte que les modèles neuronaux contrôlés convergent vers les orbites périodiques instables souhaitées rapidement. A partir de différents tests, la stabilité est garantie pour  $|x(i) - x_d| \leq 0,01$  fournissant la stabilité- $L_2$  locale d'entrée-sortie des modèles commandés par RN.

#### a. Le système de Henon

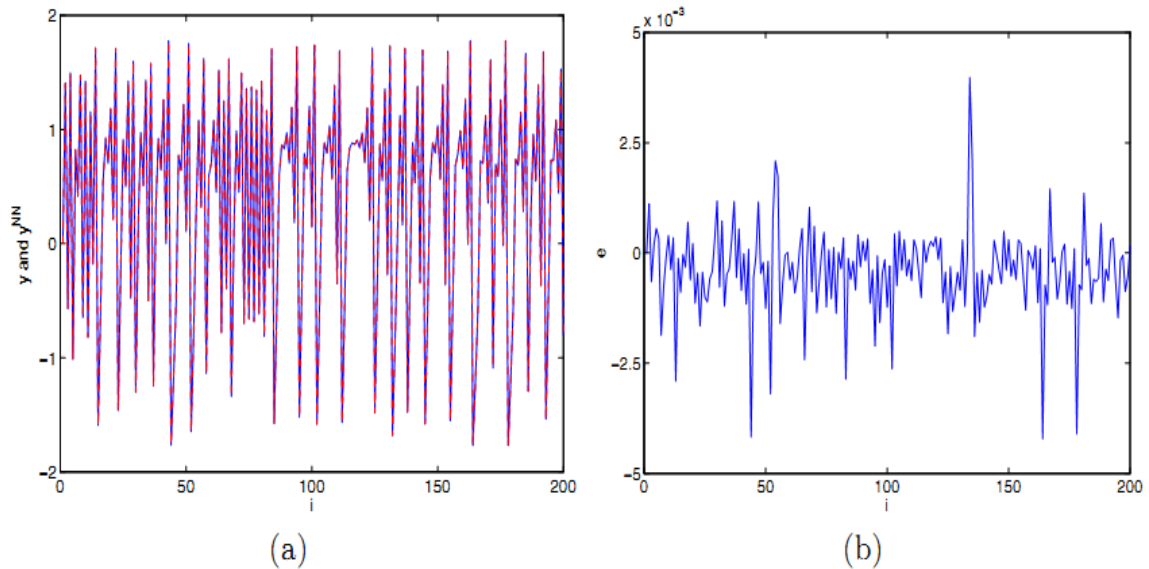
Dans cette simulation, les paires de données d'entrée-sortie sont générées à partir du système de Henon [84]:

$$y(i) = a - x^2(i) + bx(i - 1) + u(i) \quad (\text{III.29})$$

où  $a = 1,4$  et  $b = 0,3$  et pour  $u = 0$ , on obtient le système de Henon standard.

De plus, on suppose que le modèle mathématique du système de Henon est inconnu et qu'on utilise la méthode de prédiction neuronale proposée. Le résultat de simulation du modèle neuronal prédit est montré par la Figure. III.17.



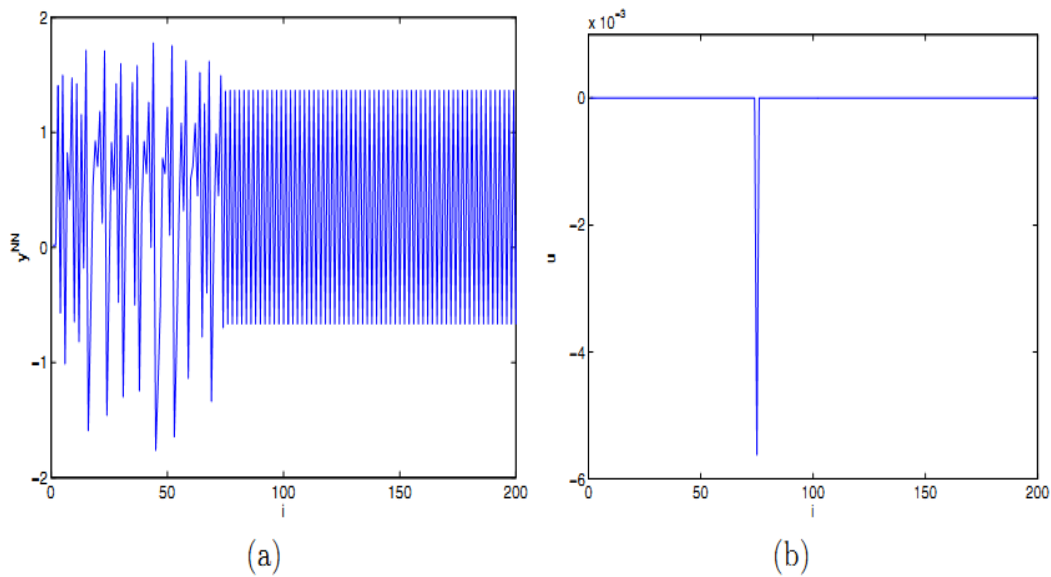


**Figure. III.17** Modélisation du système de Henon (a) : les sorties  $y$  et  $y^{NN}$ , (b) : l'erreur de prédiction du modèle neuronal.

Pour construire le domaine attractif qui correspond à l'orbite périodique instable particulière, il est nécessaire de déterminer la taille maximale de la région attractive donnée en (III.26). En variant la largeur  $x(i) - x_d$ , il est possible de changer la taille du domaine attractif par rapport à l'orbite périodique instable particulière.

A partir de différents tests, la stabilité est garantie pour  $|x(i) - x_d| \leq 0,01$  fournissant la stabilité- $L_2$  locale d'entrée-sortie du système commandé par RN. En revanche, l'orbite périodique instable (UPO) du système d'équation (III.29) est satisfaite, ce qui donne lieu à une orbite périodique instable originale. Une fois que les états du système atteignent le domaine attractif, le contrôleur RN pilotera le système Henon vers l'orbite périodique instable.

La Figure III.18 montre l'efficacité du contrôle à l'aide du concept proposé.



**Figure. III.18** Le modèle neuronal contrôlé du système de Henon (a) : La sortie  $y^{NN}$  sous contrôle, (b) : La commande  $u(i)$ .

### b. Le système Duffing

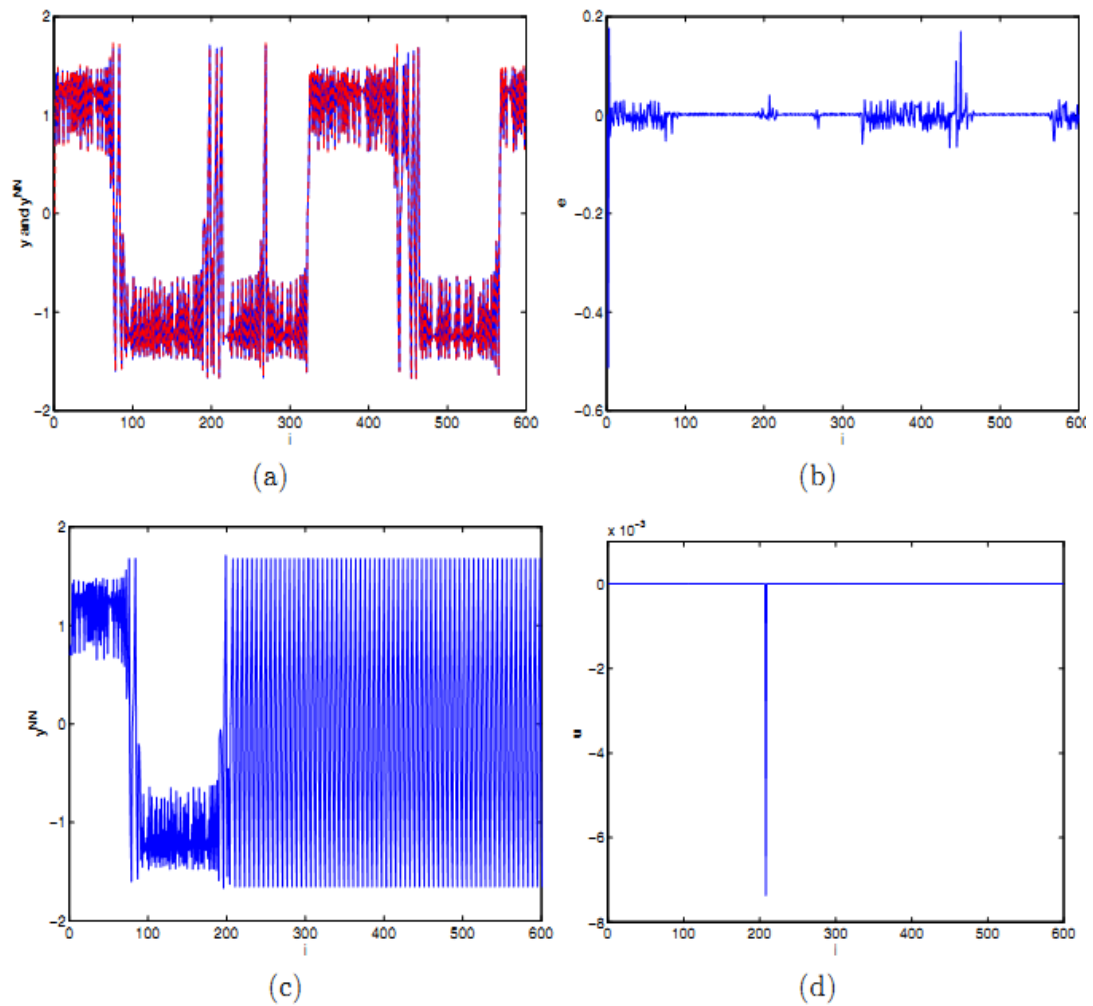
Nous considérons le système chaotique inconnu généré par le système chaotique de Duffing avec les équations modifiées suivantes [84]:

$$y(i) = ax(i) - bx^3(i) + cx(i-1) + u(i) \quad (\text{III.30})$$

où  $u$  est l'action du contrôle et pour  $u = 0$ , on obtient le système standard de Duffing.

Pour  $a = 2,75$ ,  $b = -1$  et  $c = -0,2$ , le système non contrôlé de Duffing montre un comportement chaotique.

Pour implémenter la méthode proposée basée sur la prédiction et le contrôle du chaos, il est supposé que le modèle mathématique du système de Duffing est inconnu et représenté uniquement par des paires de données d'entrée-sortie. La loi de commande a la forme de l'équation (III.28). La figure III.19 (a)-(d) illustre les résultats de modélisation et de contrôle. Une fois que les états du système atteignent le domaine d'attraction, le contrôleur neuronal pilotera le système vers l'orbite périodique instable.



**Figure. III.19** (a) Modélisation du système de Duffing, (b) L'erreur de prédiction, (c) Le modèle sous contrôle, (d) La commande

## 6.2. Transmission sécurisée des données du saint Coran par les dynamiques chaotiques contrôlées

Dans l'article [85], nous avons proposé un système de communication sécurisée pour la transmission du Saint Coran à travers les médias mondiaux tels que l'Internet, le Bluetooth, etc. Dans ce contexte, le signal d'information extrait du Saint Coran est encodé en l'appliquant comme une entrée à un système chaotique avec un message injecté dans la dynamique chaotique, la complexité du chaos ne fait qu'augmenter. Entre temps, la qualité de contrôle n'est pas influencée par le codage des messages. Un exemple illustratif est présenté pour démontrer l'efficacité du système de communication proposé. Pour d'amples détails (cf. [85]).

## 7. Conclusion

Dans ce chapitre on a proposé un algorithme de prédiction et de contrôle basé sur les RN, qui a pour objectif de stabiliser les orbites périodiques instables des systèmes chaotiques en utilisant seulement des paires de données entrée-sortie. Ainsi, nous avons utilisé dans un premier temps les réseaux de neurones artificiels pour modéliser des systèmes chaotiques, où un algorithme de la rétro-propagation avec deux approches d'optimisation ont été implémentées pour la modélisation de différents systèmes chaotiques. Les résultats obtenus avec les exemples présentés montrent que l'optimisation de l'erreur en utilisant la méthode de Levenberg-Marquardt est plus efficace en raison de la rapidité de convergence par rapport à la méthode du gradient. De plus, on a proposé une approche basée sur les critères de stabilité- $L_2$  en boucle fermée du modèle neuronal. Les résultats de simulations montrent la qualité de la méthode proposée et son efficacité. Elle peut être encore améliorée en employant d'autres méthodes d'apprentissage plus rapide, et peut être appliquée à d'autres systèmes dynamiques non linéaires complexes tels que les systèmes nerveux qui sont considérés comme des systèmes à haute dimension.

---

## Chapitre IV

Conception d'un Contrôleur Intelligent NN-PBC

---

---

## 1. Introduction

Les systèmes chaotiques qui possèdent un ensemble d'orbites périodiques instables UPO ont été largement étudiés et appliqués dans de nombreux systèmes du monde réel et dans des expériences de laboratoire, tels que les circuits électroniques, la protection des systèmes de puissance, la communication sécurisée, et les réseaux intelligents. En outre, la conception et la mise en œuvre des systèmes chaotiques multi-scroll a été un sujet d'intérêt croissant en raison de leurs applications potentielles dans diverses technologies à base de chaos et des systèmes d'information. En conséquence, un bon nombre de chercheurs et des praticiens dans le monde, ont donné une attention particulière au contrôle des systèmes chaotiques multi-scroll.

Des recherches récentes ont abordé le problème de contrôle de plusieurs systèmes chaotiques multi-scroll [83], [118- 121]. Toutefois, peu d'entre eux ont considéré la tâche de modélisation en utilisant le réseau neuronal (RN). En fait, les principaux défis du contrôle des systèmes chaotiques multi-scroll sont leurs sensibilités aux conditions initiales SCI, l'imprévisibilité et le fait que les modèles mathématiques pour ce type de systèmes chaotiques ne sont pas toujours disponibles. Les RN ont reçu une grande attention pour ce genre de problèmes car ils sont capables d'approximer les fonctions non-linéaires complexes [41, 122]. Un certain nombre d'études ont considérés les RN pour la modélisation et le contrôle des systèmes chaotiques [123, 124]. Actuellement, les méthodes habituelles de modélisation et de contrôle des systèmes chaotiques peuvent être classées en mode hors ligne et/ou mode en ligne, où le but est de permettre la prédiction et la prise de décision. Les méthodes hors ligne explorent les données stationnaires. Par contre, les méthodes en ligne dépendent principalement des algorithmes intelligents et des théories d'optimisation afin d'estimer les paramètres des réseaux de neurones et/ou du contrôleur. Comme exemples, l'algorithme de *Levenberg-Marquardt* (LM), le gradient descend, et autre algorithmes intelligents. *Poznyak* et al, a développé une technique de sliding mode pour optimiser les poids des RN dynamiques [125]. Un RN récurrent d'ordre élevé a été développé par *Lu* et al. à la fois pour identifier et contrôler les systèmes chaotiques inconnus, dans lequel la technique de linéarisation par retour d'états est utilisée de manière adaptative [126]. *Qin* et al. [124] a introduit un système de contrôle basé sur la rétro-propagation des RN. Cependant, l'utilisation des RN a surtout mis l'accent sur les systèmes chaotiques classiques et très peu de résultats ont été obtenus pour la

modélisation des systèmes chaotiques multi scroll [127, 128]. En fait, le développement des modèles pour les systèmes chaotiques multi-scroll est nécessaire pour prédire avec précision, contrôler et diagnostiquer un tel ou tel système où les modèles des systèmes sont soumis à des incertitudes des paramètres. Contrairement aux méthodes classiques de contrôle du chaos, les algorithmes intelligents modernes utilisant les RNA et la logique Floue sont capables de trouver rapidement des solutions aux problèmes de contrôle du chaos dans les applications du monde réelles. Les contributions récentes concernant le contrôle des systèmes chaotiques à base de prédiction neuronal peuvent être divisées en deux catégories principales :

- 1- L'amélioration du contrôle prédictif pour stabiliser les systèmes chaotiques à temps continu [129] ;
- 2- La combinaison du contrôle prédictif à base des RNA avec les algorithmes intelligents pour stabiliser un large éventail de systèmes chaotiques soumis à des incertitudes [84], [130- 131].

Même si de nombreuses techniques ont été consacrées au sujet du contrôle des systèmes chaotiques, d'après notre connaissance, il n'y a aucune contribution qui concerne la stabilisation des systèmes chaotiques multi-scroll soumis à des incertitudes et des dynamiques non modélisées. En conséquence, nous abordons dans ce chapitre la conception d'un contrôleur intelligent à base des RN pour modéliser et stabiliser des systèmes chaotiques multi-scroll qui incorporent une connaissance a priori de leurs équations différentielles.

**Objectif :** Une partie indispensable du contrôle précis des systèmes chaotiques multi-scroll : l'identification de modèle a reçu une attention croissante au cours de ces dernières années. À cause de l'incertitude des systèmes chaotiques et de la dynamique non modélisée, les méthodes de contrôle classique ne peuvent pas, souvent, garantir une performance suffisamment élevée pour stabiliser les systèmes chaotiques multi-scroll. Pour aborder le problème de mieux, nous proposons un contrôleur intelligent appelé adaptive Neural Network Prediction-Based Controller (NN-PBC). Le RN est entraîné avec les données (l'échantillon d'apprentissage) du modèle réel : les données sont divisées en deux ensembles, un ensemble est utilisé pour l'entraînement et l'autre ensemble pour les tests. En fait, un RN (ou Neural Network NN) généralisé donnera de bon résultats pour les deux ensembles de données (c'est à dire, il est capable

d'approximer le système réel même pour des données hors échantillon d'apprentissage), la loi de contrôle prédictif est ensuite appliquée sur le modèle neuronal (modèle RN) pour stabiliser le système sur ses multiples points d'équilibre. La stabilité des systèmes en boucle fermée est prouvée. De plus, des exemples de simulation sur deux systèmes chaotiques multi-scroll typiques sont présentés pour démontrer l'efficacité du contrôleur proposé.

## 2. Description des systèmes

### 2.1. Les systèmes chaotiques multi-scroll

Un système chaotique multi-scroll est un système chaotique modifié de sorte qu'il génère plusieurs scrolls (défilements ou plis). Vu que son modèle prototype a été introduit premièrement par *Suykens* et al. en 1993 [132], des investigations approfondies ont été menées pour développer des nouveaux modèles multidirectionnels des systèmes chaotiques multi-scroll.

Les systèmes chaotiques multi-scroll peuvent être conçus en utilisant des approches différentes, comme : les fonctions de modulation à largeur d'impulsions (Pulse Width Modulation PWM) [132], les fonctions de modulation non linéaire [30], step circuits [133], positive-type second generation current conveyors (CCII+), negative-type second generation current conveyors (CCII-) [134], et les circuits FPGA (Field-Programmable Gate Array) [135]. En particulier, la conception des systèmes chaotiques multi-scroll via des fonctions de modulation non linéaires a attiré beaucoup d'intérêt en raison de leurs implémentations simple en circuit. L'un des principaux résultats de cette approche de conception est l'utilisation de la fonction *Sinus*.

Considérons le système chaotique multi-scroll exprimé au-dessous :

$$\dot{X} = f(t, X(t)) \tag{IV.1}$$

où  $X = [x_1, x_2, x_3, \dots, x_n]^T \in \mathfrak{R}^n$  est le vecteur d'états, et  $f(t, X(t))$  est une fonction continue.

Deux exemples typiques de systèmes chaotiques multi-scroll qui sont basés sur des fonctions de modulation non linéaire sont présentés dans ce qui suit.



### 2.1.1. Le circuit de Chua à n-scroll

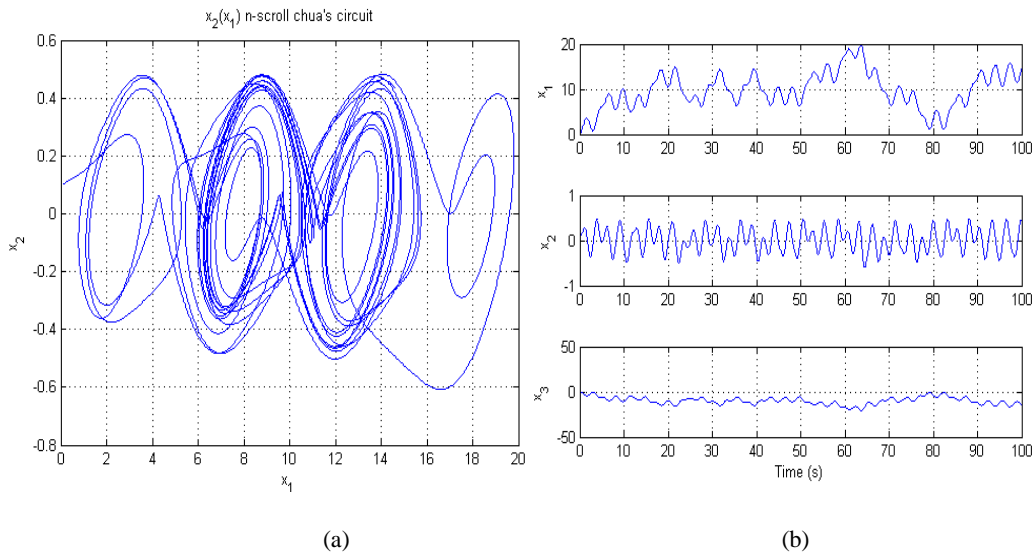
Considérons le circuit de Chua donné par le système d'équations suivant [23, 133]:

$$\begin{cases} \dot{x}_1 = \alpha(x_2 - h(x_1)) \\ \dot{x}_2 = x_1 - x_2 + x_3 \\ \dot{x}_3 = -\beta x_2 \end{cases} \quad (\text{IV.2})$$

où  $x_1, x_2, x_3$  sont les variables d'états,  $\alpha, \beta$  les paramètres du système et  $h(x_1)$  est une fonction non linéaire tel que :

$$h(x_1) = \begin{cases} \frac{b\pi}{2a}(x_1 - 2ac) & \text{if } x_1 \geq 2ac \\ -b \sin\left(\frac{\pi x_1}{2a} + d\right) & \text{if } -2ac < x_1 < 2ac \\ \frac{b\pi}{2a}(x_1 + 2ac) & \text{if } x_1 \leq -2ac \end{cases} \quad (\text{IV.3})$$

Lorsque  $\alpha = 10.814$ ,  $\beta = 14$ ,  $a = 1.3$ ,  $b = 0.11$ ,  $c = 7$ ,  $d = 0$ , un attracteur multi-scroll se produit (cf. Figure. IV.1).



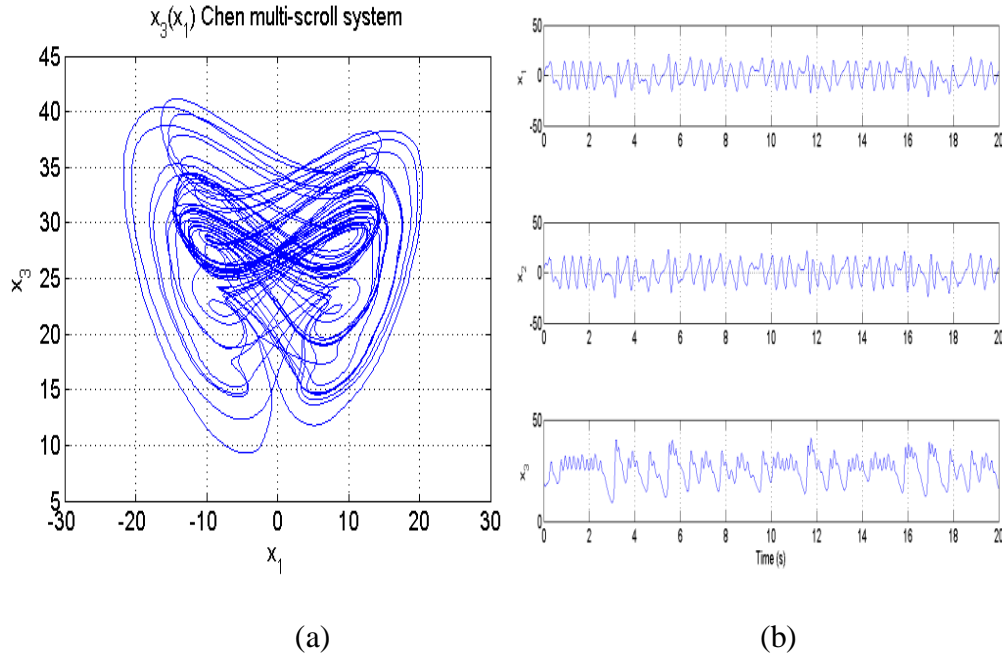
**Figure IV.1** Circuit de Chua à  $n$ -scroll. (a) plan de phase. (b) évolutions temporelle.

### 2.1.2. Le système de Chen multi-scroll

Un système multi-scroll de Chen, est décrit par le système d'équations suivant [83]:

$$\begin{cases} \dot{x}_1 = a(x_2 - x_1) \\ \dot{x}_2 = (c - a - x_3 + d \sin x_3)x_1 + cx_2 \\ \dot{x}_3 = x_1x_2 - bx_3 \end{cases} \quad (\text{IV.4})$$

où  $x_1, x_2, x_3$  sont les variables d'états,  $a, b, c, d$  sont des paramètres du système. Quand  $a = 35, b = 3, c = 28, d = 8$ , un attracteur multi-scroll se produit (cf. Figure. IV.2).



**Figure IV.2** système de Chen a  $n$ -scroll. (a) plan de phase. (b) évolutions temporelle.

## 2.2. La structure du réseau de neurones

Une variété de structures des RNA ont été proposées dans la littérature pour la prédiction, le contrôle et le diagnostic pour répondre aux différents types de besoins de modélisation [41, 122], Le MLP a rétro-propagation est l'une des structures les plus connus, capable d'approximer les fonctions non linéaire complexes. En commun avec d'autres méthodes basées sur les RNA, les couches cachées et le nombre de neurones sont sélectionnées par la méthode d'essai-erreur.

Basant sur la propriété d'approximation des MLP. Une fonction non linéaire  $f(t, X(t))$  peut-être approximer par [122] :

$$f(t, X(t)) = W^T \varphi(x) + \varepsilon(x) \quad (IV.5)$$

où  $W$  est le vecteur des poids,  $\varphi(x)$  est la fonction d'activation,  $\varepsilon(x)$  est l'erreur d'approximation des MLP, et  $X \in \mathbb{R}^q$  est le vecteur d'entrée, où  $q$  est le nombre des nœuds d'entrées. Une fonction Sigmoidé est choisie comme la fonction d'activation de la couche cachée pour aider le réseau à apprendre la dynamique non linéaire du

système chaotique multi-scroll, à savoir les rapports entre les couches d'entrée et de sortie. La fonction d'activation de la couche de sortie est une fonction linéaire (Purelin). De plus, les hypothèses suivantes sont considérées tout au long de ce chapitre.

A1 : le vecteur poids  $W$  est limité, i.e.,  $\|W\| \leq W_M \in \mathbb{R}^+$ .

A2 : l'erreur approximative  $\varepsilon(X)$  est limité, i.e.,  $\|\varepsilon(X)\| \leq \varepsilon_M \in \mathbb{R}^+$ .

Il est nécessaire de disposer des données d'apprentissage suffisantes et précises pour le bon développement d'un modèle neuronal, ainsi la modélisation est réalisée en deux phases :

- 1) Phase d'apprentissage : le système neuronal (NN) mémorise les relations de l'ensemble des données d'apprentissage. Il sélectionne les caractéristiques de neurones et de la topologie, minimise l'erreur, et arrête en fonction des critères d'arrêt.
- 2) Phase de test : le système neuronal prédit et test les ensembles de données. La performance du RN sur les données des tests représente sa capacité de généraliser.

L'ensemble des données est divisé en deux ensembles, un ensemble est utilisé pour l'apprentissage et l'autre pour les tests, en effet, un RN généralisé performera bien pour les deux ensembles.

### 3. Amélioration de la méthode de contrôle basé sur la prédiction et formulation de problème

La méthode de contrôle à base de prédiction peut être considérée comme une sorte de stratégie de contrôle adaptatif [136]. Elle a été initialement introduite pour les systèmes chaotiques par *Ushio* et *Yamamoto* en 1999 [137] et elle a réussi à contrôler les systèmes chaotiques à temps discret. De nombreuses études ont proposé des extensions pour traiter les différents types de systèmes chaotiques. Parmi eux *Boukabou* et al. [129] ont proposé une méthode de contrôle de chaos pour les systèmes continus. Il a été montré que le nombre d'étapes de prédiction a très peu d'effets sur les performances de poursuite, ce qui signifie que même une petite avance de contrôle prédictif est suffisamment efficace.

### 3.1 Principes de contrôle prédictif

Considérons un système chaotique avec une équation d'état sous contrôle :

$$\dot{X}(t) = f(t, X(t)) + u_p(t) \quad (\text{IV.6})$$

où  $X \in \mathfrak{R}^n$  est le vecteur d'état,  $f: \mathfrak{R}^n \rightarrow \mathfrak{R}^n$  est une fonction continue, et  $u_p \in \mathfrak{R}^n$  est l'action de contrôle. Nous supposons que  $f(\cdot)$  est différentiable et le système génère le chaos lorsque  $u_p = 0$ . La tâche est de trouver une commande  $u_p(t)$  de telle sorte que la trajectoire d'un système contrôlé suit la cible

$$\lim_{t \rightarrow \infty} \|X(t) - \bar{X}\| = 0 \quad (\text{IV.7})$$

où  $\bar{X}$  est le point d'équilibre instable à stabiliser.

La méthode de contrôle proposé est basée sur la prédiction d'un état futur de temps et la loi de commande est calculée à partir de la différence entre l'état actuel et l'état futur du système chaotique multi-scroll non contrôlé. Nous choisissons la commande par rétroaction comme suit

$$u_p(t) = K(A - I)X(t) \quad (\text{IV.8})$$

où  $A = \left. \frac{\partial f}{\partial X} \right|_{X=\bar{X}}$  représente la matrice Jacobien du système, évaluée au point d'équilibre instable choisi pour la stabilisation,  $I$  la matrice d'identité, et  $K$  la matrice des gains de retour.

Le système chaotiques multi-scroll linéarisé à un point d'équilibre instable désiré est sous contrôle comme suit

$$\dot{X}(t) = AX(t) + K(A - I)X(t) = -\bar{A}X(t) \quad (\text{IV.9})$$

**Remarque IV.1 :** La commande prédictive stabilise le système chaotique (eq. IV.6) pour le gain  $K$ , satisfait les conditions suffisantes  $\bar{A} > 0$  et  $\det(A - I) \neq 0$  [129].

### 3.2. Formulation de problème

En raison des incertitudes et des dynamiques non modélisés. Les méthodes de contrôle conventionnel, y compris le contrôle prédictif ne peuvent pas garantir une performance suffisamment élevée pour stabiliser les systèmes chaotique multi-scroll.

Pour résoudre ce problème nous proposons une conception d'un contrôleur intelligent appelé Neural Network Prediction-based Controller (NN-PbC).

Le système chaotique multi-scroll (IV.1) est sous l'entrée de commande  $U(t)$  comme suit :

$$\dot{X}(t) = f(t, X(t)) + U(t) \tag{IV.10}$$

Dans la section suivante, nous abordons le problème de la conception d'un RN pour construire des modèles qui incorporent une connaissance a priori sous forme d'équations différentielles pour les systèmes chaotiques. La commande prédictive est ensuite appliquée sur les modèles RN obtenus afin de forcer la trajectoire du système d'aller vers un point d'équilibre désiré.

### 4. La conception du NN-PbC

La figure IV.3 représente le block diagramme du système de contrôle NN-PbC. Il est clair que la mise en œuvre des RN ne dépend que de l'information de référence désirée.

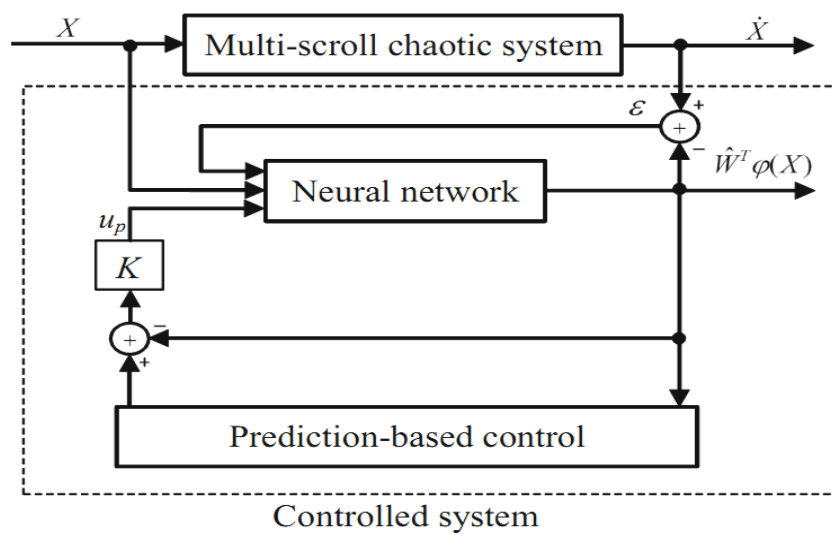


Figure IV.3 Bloc diagramme du contrôle.

Afin de stabiliser efficacement le système chaotique multi-scroll, nous proposons un contrôleur prédictif intelligent. On définit l'entrée de commande  $U(t)$  comme suit :

$$U(t) = -\hat{f}(t, X(t)) + u_p(t) \quad (IV.11)$$

où  $u_p(t)$  est la loi de commande basée sur la prédiction donnée dans (IV.8) et  $\hat{f}(t, X(t))$  est le système neuronal adaptatif tels que :

$$\hat{f}(t, X(t)) = \hat{W}^T \varphi(X) \quad (IV.12)$$

où  $\hat{W}$  est l'estimation du vecteur des poids  $W$ .

Substituant (IV.5) et (IV.11) dans (IV.10)

$$\begin{aligned} \dot{X}(t) &= -\bar{A}X(t) + \hat{W}^T \varphi(X) - \hat{W}^T \varphi(X) + \varepsilon(X) \\ &= -\bar{A}X(t) + \hat{W}^T \varphi(X) + \varepsilon(X) \end{aligned} \quad (IV.13)$$

où  $\tilde{W} = W - \hat{W}$  représente l'erreur d'approximation des poids.

**Théorème IV.1** *Laisser l'action de commande être fournie par le contrôleur basé sur la prédiction (IV.8). La loi d'adaptation des poids du RN est donnée par*

$$\dot{\hat{W}} = F (\varphi(X)X^T - \gamma \|X\| \hat{W}) \quad (IV.14)$$

où  $F$  est une matrice définie positive et  $\gamma$  est une constante positive. La stabilité du système en boucle fermée (IV.13) peut alors être garantie.

**Démonstration** : considérons la fonction du Lyapunov comme suit :

$$V(v) = \frac{1}{2} X^T(t)X(t) + \frac{1}{2} \text{tr}(\tilde{W}^T F^{-1} \tilde{W}) \quad (IV.15)$$

La dérive temporelle de  $V$  est donnée par

$$\begin{aligned} \dot{V} &= X^T(t)\dot{X}(t) + \text{tr}(\tilde{W}^T F^{-1} \dot{\tilde{W}}) \\ &= X^T(t)\dot{X}(t) - \text{tr}(\tilde{W}^T F^{-1} \dot{\hat{W}}) \end{aligned} \quad (IV.16)$$

En appliquant (IV.13) et (IV.14) à (IV.16), on aura

$$\dot{V} = -X^T \bar{A}X + X^T \hat{W}^T \varphi(X) + X^T \varepsilon(X) - \text{tr}(\tilde{W}^T \varphi(X)X^T - \gamma \|X\| \hat{W}) \quad (IV.17)$$

Du fait que  $\alpha^T \beta = \text{tr}(\beta \alpha^T)$ , donc

$$X^T \tilde{W}^T \varphi(X) = \text{tr}(\tilde{W}^T \varphi(X) X^T) \quad (\text{IV.18})$$

Qui, substitué en (IV.17) donne

$$\dot{V} = -X^T \bar{A} X + X^T \varepsilon(X) + \gamma \|X\| \text{tr}(\tilde{W}^T \hat{W}) \quad (\text{IV.19})$$

En utilisant la norme matricielle de *Frobenius*  $\|\cdot\|_F$ , on aura

$$\begin{aligned} \text{tr}(\tilde{W}^T \hat{W}) &= \text{tr}[\tilde{W}^T (W - \tilde{W})] = \langle W, \tilde{W} \rangle_F - \|\tilde{W}\|_F^2 \\ &= \|\tilde{W}\|_F \|W\|_F - \|\tilde{W}\|_F^2 \end{aligned} \quad (\text{IV.20})$$

Selon l'hypothèse A1, on a

$$\|\tilde{W}\|_F \|W\|_F - \|\tilde{W}\|_F^2 \leq \|\tilde{W}\|_F W_M - \|\tilde{W}\|_F^2 \quad (\text{IV.21})$$

En complétant le carré, on obtient

$$\|\tilde{W}\|_F W_M - \|\tilde{W}\|_F^2 = -\left(\|\tilde{W}\|_F - \frac{1}{2} W_M\right)^2 + \frac{1}{4} W_M^2 \quad (\text{IV.22})$$

En réécrivant (IV.19), on aura

$$\dot{V} \leq -\|X\| \left\{ \lambda_{\min}(\bar{A}) \|X\| + \alpha \left( \|\tilde{W}\|_F - \frac{1}{2} W_M \right)^2 - \varepsilon_M - \frac{1}{4} \alpha W_M^2 \right\} \quad (\text{IV.23})$$

Comme conséquence de l'hypothèse A2, où  $\lambda_{\min}(\bar{A})$  est la plus petite valeur propre de la matrice  $\bar{A}$ . L'inégalité (IV.23) est garantie d'être négative tant que (IV.24) ou (IV.25) est vérifiée :

$$\|X\| \geq \left( \varepsilon_M - \frac{1}{4} \alpha W_M^2 \right) / \lambda_{\min}(\bar{A}) = \Omega_X \quad (\text{IV.24})$$

$$\|\tilde{W}\|_F \geq \sqrt{\varepsilon_M + \frac{1}{4} \alpha W_M^2} + \frac{1}{2} W_M = \Omega_{\tilde{W}} \quad (\text{IV.25})$$

où  $\Omega_X$  et  $\Omega_{\tilde{W}}$  sont des régions de convergence. Selon la théorie de stabilité de Lyapunov [138], le système en boucle fermée est stable, ceci termine la démonstration.

**Remarque IV.2 :** Notez qu'il n'y a pas de méthode claire et précise pour décider/choisir les paramètres, structures, ou méthode d'entraînement. L'architecture

du RN est composée d'une couche d'entrée, une couche de sortie, et une ou deux couches cachées. L'ensemble des données est extrait à partir du système réel et ensuite divisé en trois ensembles : (60%) pour la l'entraînement (ces données sont présentées au réseau pendant l'apprentissage et le réseau est ajusté suivant son erreur). Un autre ensemble (20%) pour les tests (ces données ne sont pas utilisées pendant l'apprentissage et ainsi elle fournit une mesure indépendante de la performance du réseau pendant et après l'apprentissage). Et le dernier ensemble (20%) sert à la validation (ces données sont utilisées pour mesurer la généralisation du réseau et mettre un terme à l'apprentissage quand la généralisation cesse l'amélioration), le processus d'ajustement de poids et des biais du système RN est répété jusqu'à ce que l'une des conditions d'arrêt est remplie. Les critères d'arrêt de l'apprentissage sont : (i) le (MSE) va au-dessous d'une valeur spécifique  $\varepsilon_M$ , (ii) L'ampleur du gradient tombe en dessous d'une certaine valeur, ou (iii) Un nombre spécifié d'itérations a été effectué.

**Remarque IV.3 :** De plus, notez que l'erreur d'approximation  $\varepsilon_M$  a une certaine influence sur le processus d'apprentissage et les performances du contrôle différemment. Au cours du processus d'apprentissage,  $\varepsilon$  est une mesure de similarité entre le système réel et le modèle neuronal. En outre, lorsque  $\varepsilon_M$  est relativement grand, un modèle RN différent est généré à partir du système réel, et par conséquence, son contrôle est inutile parce que le but est de contrôler le système réel initialement modélisé. Barron [139] a constaté qu'il existe des limites inférieures d'ordre  $(1/N_A)^{2/n}$  sur l'erreur d'approximation  $\varepsilon_M$  si seulement les paramètres d'une combinaison linéaire de fonction de base sont ajustés. La démonstration de la stabilité de NN-Pbc montre que l'effet des bornes sur l'erreur d'approximation peut être atténué, par le choix judicieux de la matrice de gain  $K$ .

**Remarque IV.4 :** Notez que le processus de stabilisation des systèmes chaotiques multi-scroll sur des points fixes instables est une procédure standard. De même, les modèles RN avec contrôle de retour d'état additif peuvent également être stabilisés, où les paramètres du contrôleur RN sont modifiés de sorte à satisfaire le critère de stabilité.



### 5. Résultats expérimentales

Dans cette section, nous présentons les résultats des simulations. La performance du modèle neuronal est mesurée quantitativement par le temps moyen de CPU, nécessaire pour entraîner le RN à approximer chaque système chaotique multi-scroll, MSE (Erreur quadratique moyenne) et l'indice de régression linéaire [140].

Les résultats sont résumés dans les Tableaux IV.1 et IV.2. Toutes les expériences ont été exécutées 10 fois pour assurer la validité et la précision des mesures expérimentales.

**Tableau. IV.1** Performance de différentes structures des RN modélisant le circuit de Chua.

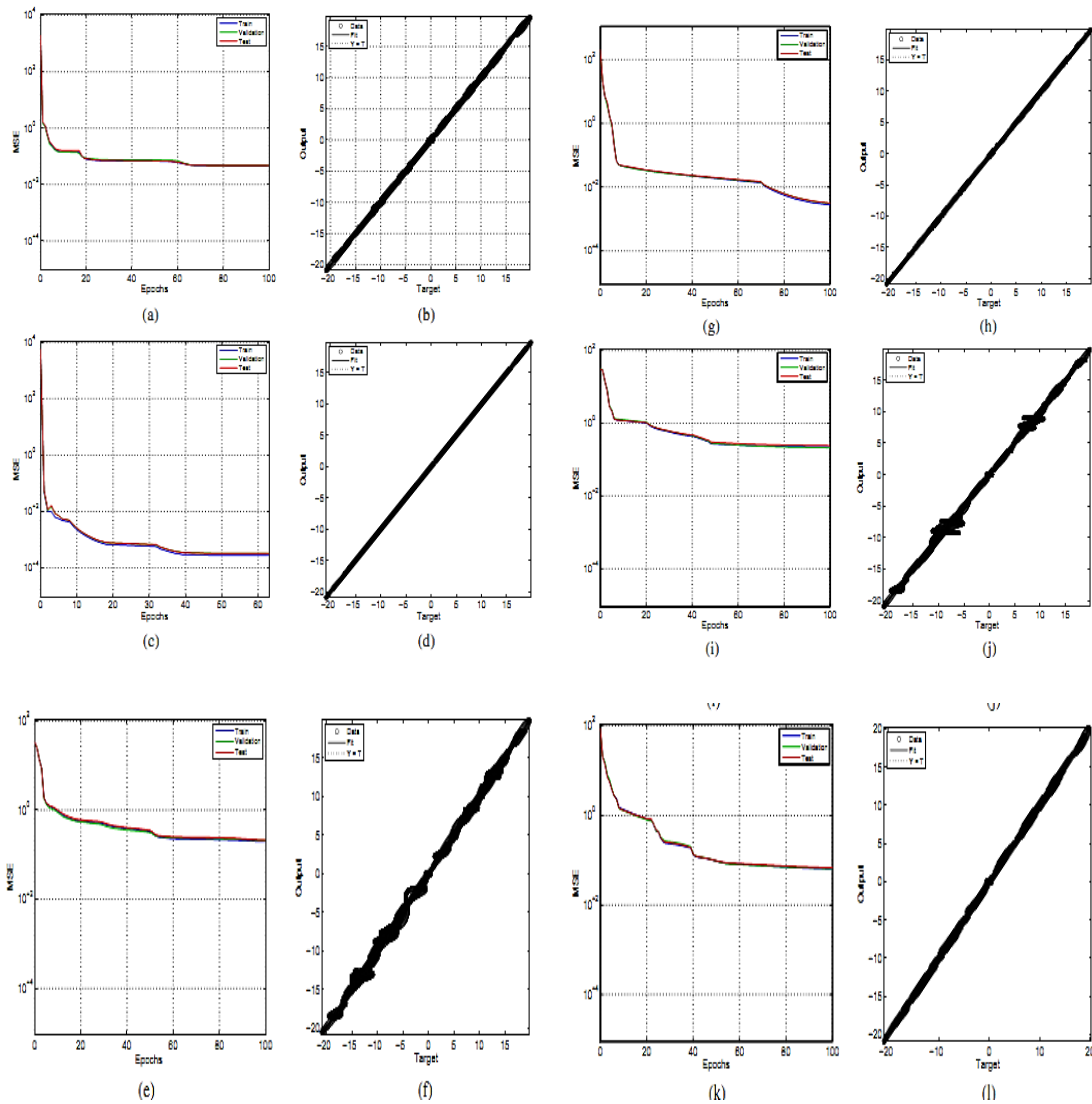
Résultats	1 couche cachée		2 couches cachées		3 couches cachées	
	60	160	[30 10]	[72 12]	[30 20 10]	[42 12 6]
Temps T (sec)	69	155	77	343	259	180
R.test	1	1	0.998	0.999	0.998	0.999
MSE	0.0461	0.0002	0.2010	0.0027	0.2210	0.0645

**Tableau. IV.2** Performance de différentes structures des RN modélisant le Système de Chen.

Résultats	1 couche cachée		2 couches cachées		3 couches cachées	
	160	280	[60 20]	[90 12]	[26 14 6]	[40 20 10]
Temps T (sec)	267	362	531	853	144	392
R.test	0.9970	0.999	0.980	0.995	0.902	0.985
MSE	1.050	0.004	4.880	1.970	38	5.79

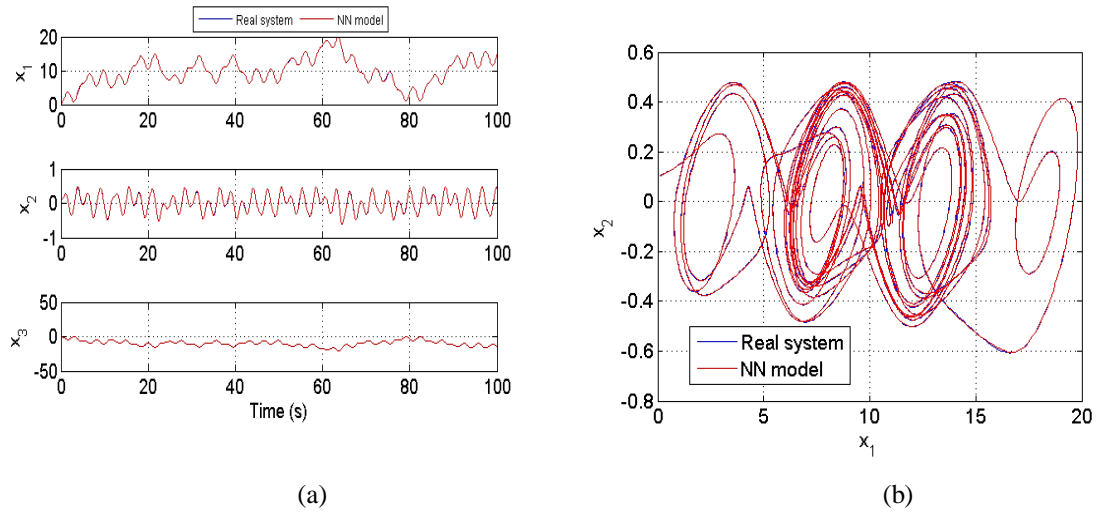
### 5.1. Modélisation du circuit de Chua de n-scroll

Afin de modéliser un circuit de Chua de n-scroll, l'ensemble de données utilisé pour l'apprentissage a été extrait du système réel (IV.2). La Figure IV.4 (a)-(l) représente les résultats d'entraînement et des tests du RN avec les paires de données du système réel. Nous pouvons voir que le modèle neuronal se rapproche du circuit de Chua de n-scroll avec une grande précision (cf. Figure IV.5).



**Figure. IV.4** Performance de différentes structures RN

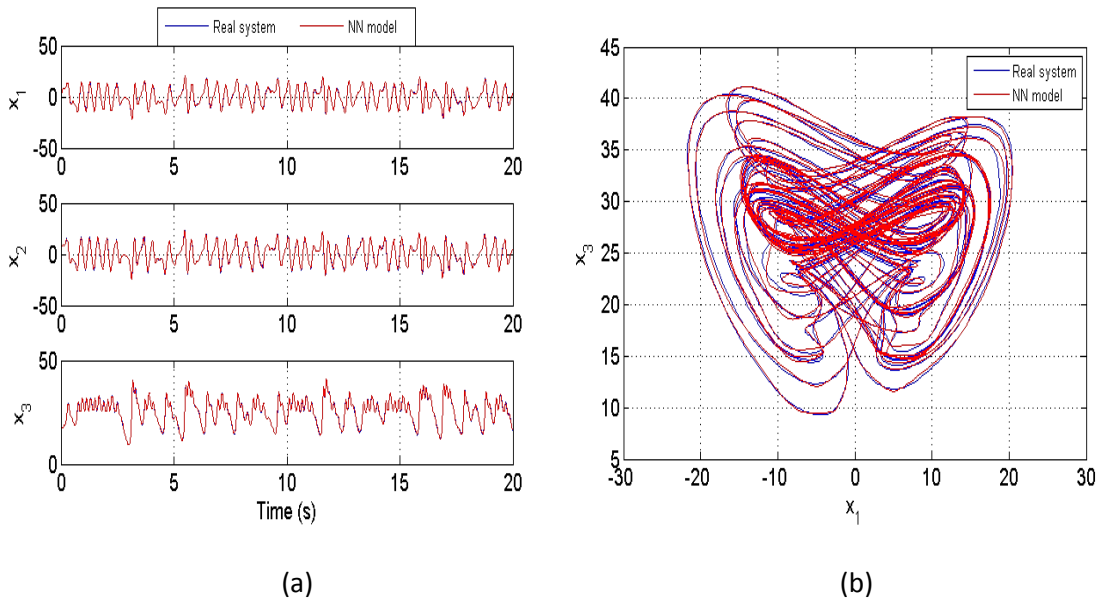
Le Tableau IV.1 et la figure précédente présentent les résultats de diverses architectures des RN pour la modélisation du système multi-scroll de Chua. À partir des résultats, nous pouvons clairement voir que le RN avec une seule couche cachée fonctionne mieux que pour les autres architectures.



**Figure. IV.5** Modélisation du circuit de Chua multi-scroll. (a) Evolutions temporelle. (b) Plan de phase.

## 5.2. Modélisation du système multi-scroll de Chen

La Figure IV.7 montre les résultats d'entraînement et des tests du RN sur les paires de données de l'échantillon d'apprentissage. Nous pouvons voir que le modèle du RN se rapproche du système multi-scroll de Chen avec précision (cf. Figure IV.6).



**Figure. IV.6** Modélisation du système de Chen multi-scroll. (a) Evolutions temporelle. (b) Plan de phase.

Le Tableau IV.2 et la Figure. IV.7 (a)-(l) présentent les résultats de diverses architectures des RN pour modéliser le système multi-scroll de Chen. A partir des résultats, on peut clairement observer que le RN avec une seule couche cachée se comporte mieux que pour les autres architectures.

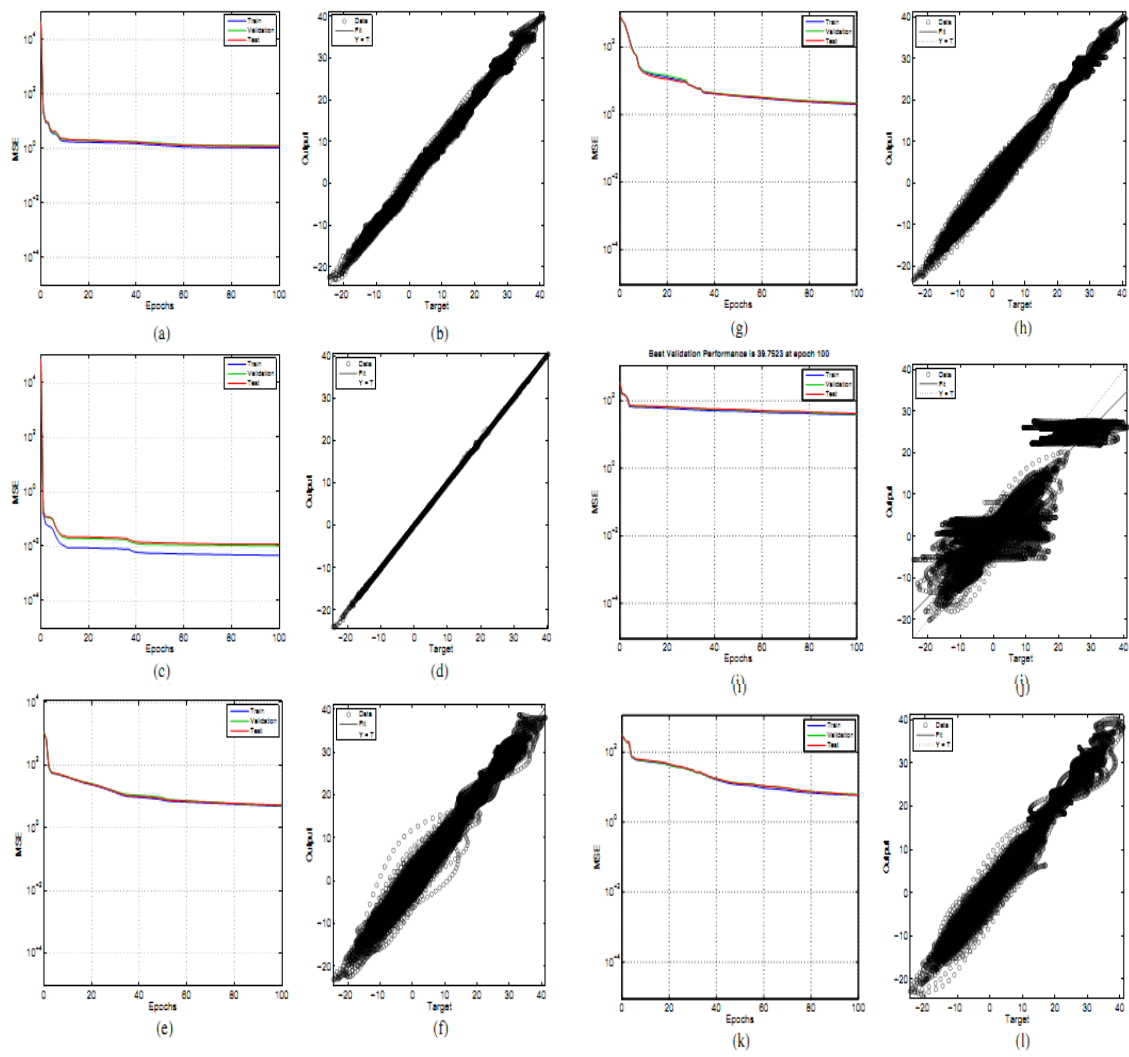


Figure. IV.7 Performance de différentes structures RN

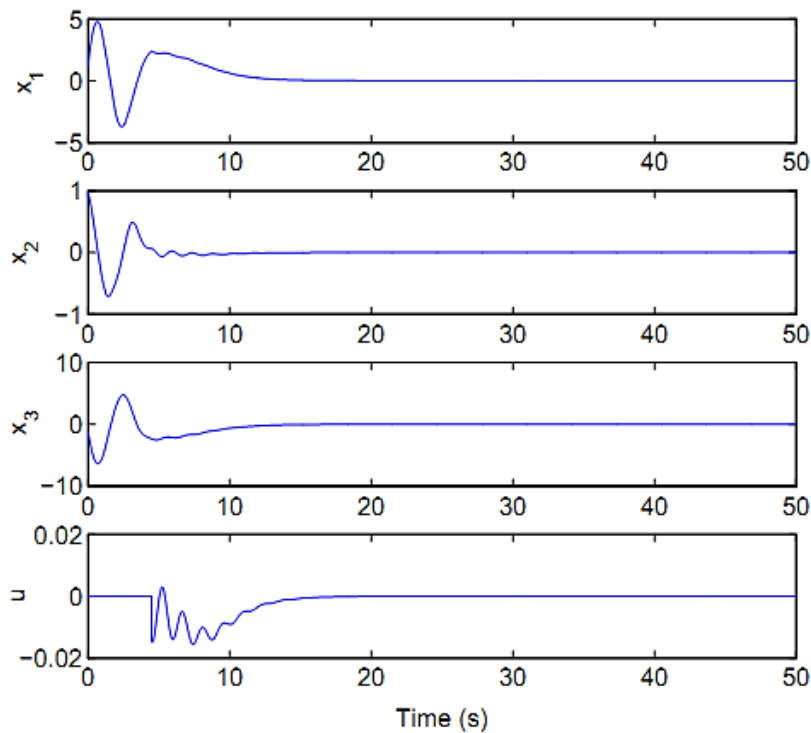
## 6. Résultats de la simulation pour le contrôle

Dans cette section, nous présentons les résultats de stabilisation des modèles neuronaux obtenus des deux systèmes chaotiques multi-scroll.

### 6.1. Stabilisation du modèle RN du circuit de Chua multi-scroll

La loi de commande basée sur la prédiction intelligente du RN a été construite sous forme de (IV.8) et (IV.14) avec des gains  $F = 10I_{3 \times 3}$  et  $\gamma = 2$ . Les états initiaux du système ont été définie par  $X(0) = (0.1, 0.1, 0.1)$ .

Pour  $K = \text{diag}(-1.5, 0, 0)$  et pour des valeurs du temps différentes, le modèle neuronal du circuit de Chua converge vers des points d'équilibre instables  $\bar{X}_{\pm 0}, \bar{X}_{\pm 2}, \bar{X}_{\pm 4}$  et  $\bar{X}_{\pm 6}$ . Pour  $K = \text{diag}(-0.8, 0, 0)$ , Les états du système contrôlé convergent vers les points d'équilibre  $\bar{X}_{\pm 1}, \bar{X}_{\pm 3}, \bar{X}_{\pm 5}$  et  $\bar{X}_{\pm 7}$ . Les résultats de simulation sont présentés dans les Figures. IV.8 à IV.15. Évidemment, une performance de poursuite satisfaisante peut être garantie, comme les états du système convergent vers le point d'équilibre souhaité avec une petite force appliquée.



**Figure. IV.8** Stabilisation du modèle RN du circuit de Chua sur  $\bar{X}_0$ .

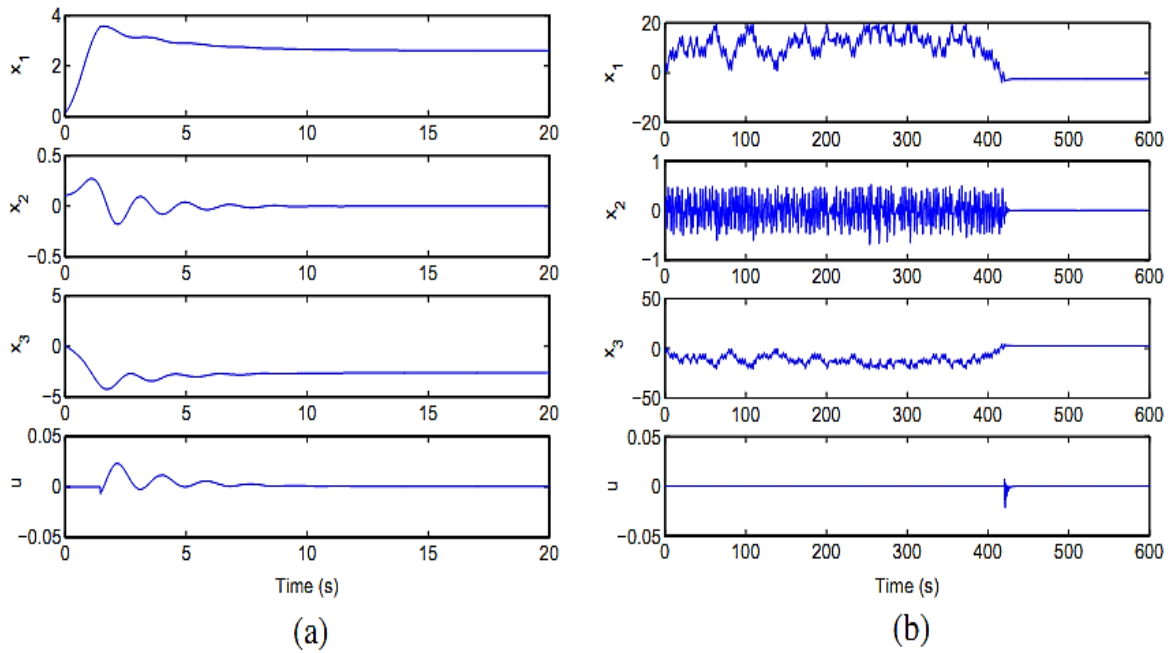


Figure. IV.9 Stabilisation du modèle RN du circuit de Chua sur  $\bar{X}_{\pm 1}$ .

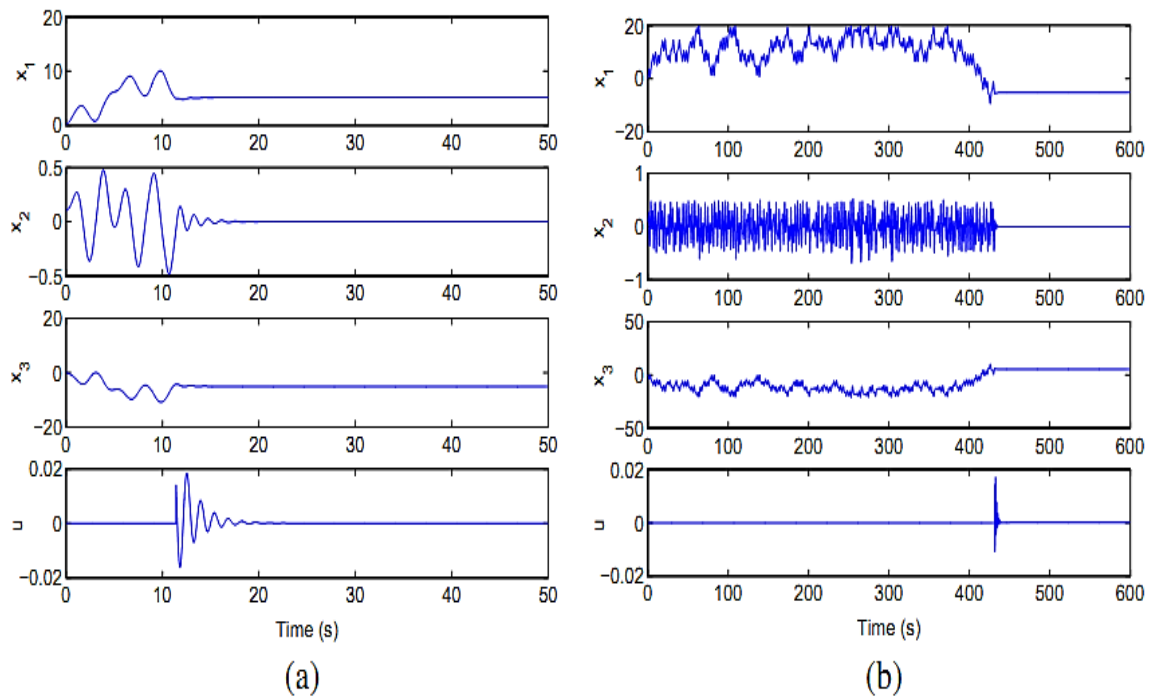


Figure. IV.10 Stabilisation du modèle RN du circuit de Chua sur  $\bar{X}_{\pm 2}$ .

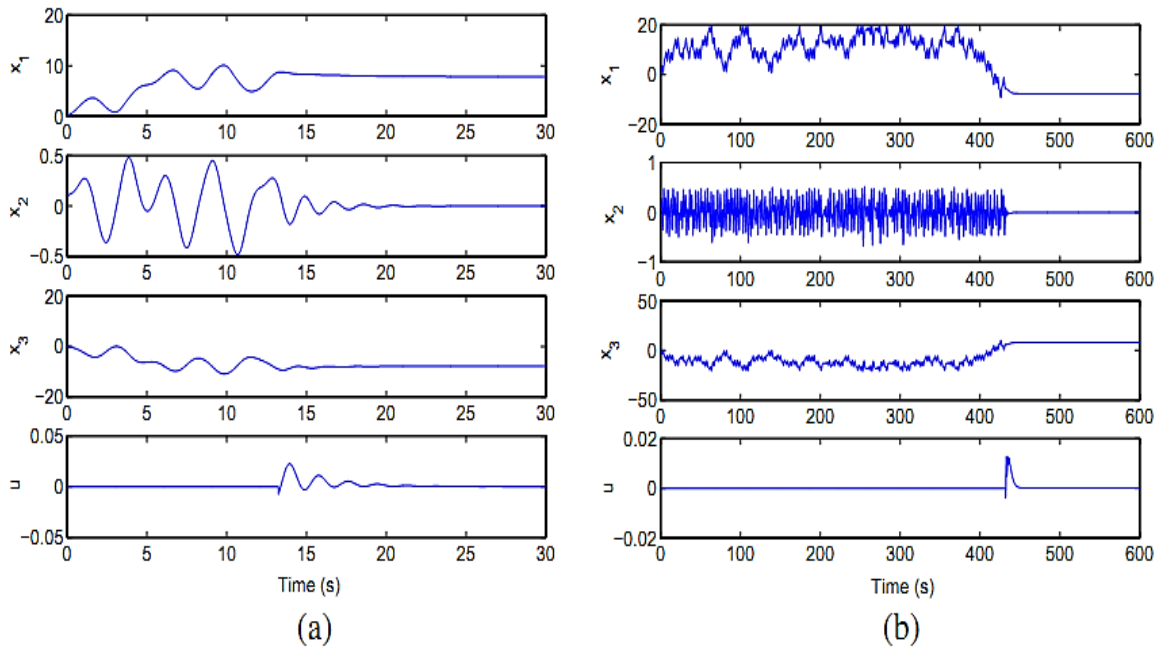


Figure. IV.11 Stabilisation du modèle RN du circuit de Chua sur  $\bar{X}_{\pm 3}$ .

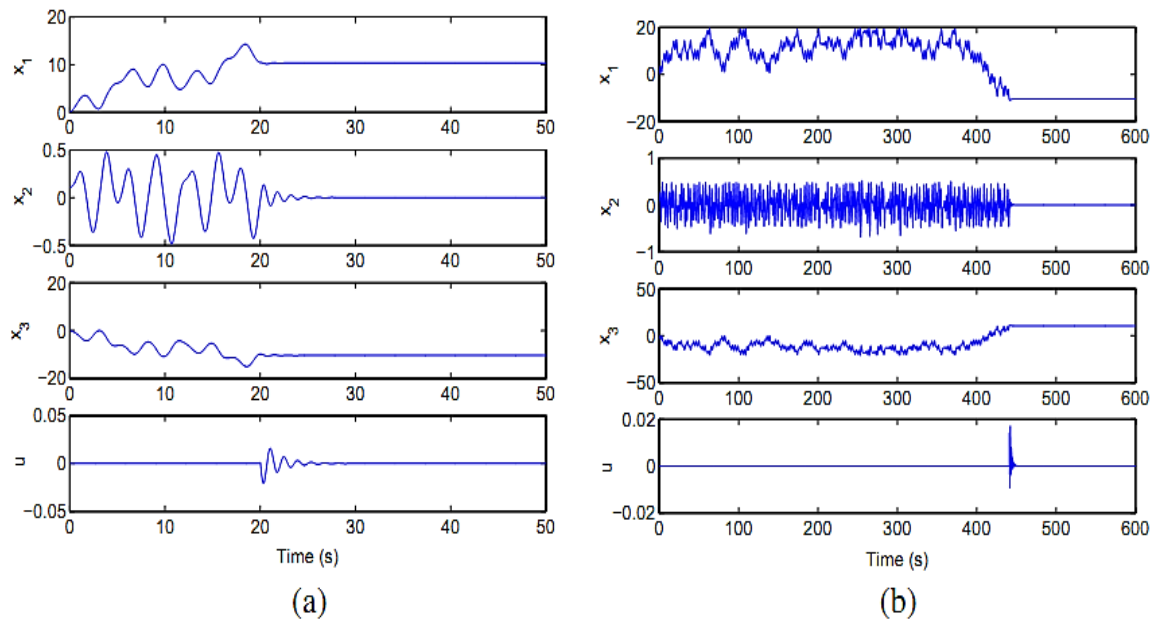


Figure. IV.12 Stabilisation du modèle RN du circuit de Chua sur  $\bar{X}_{\pm 4}$ .

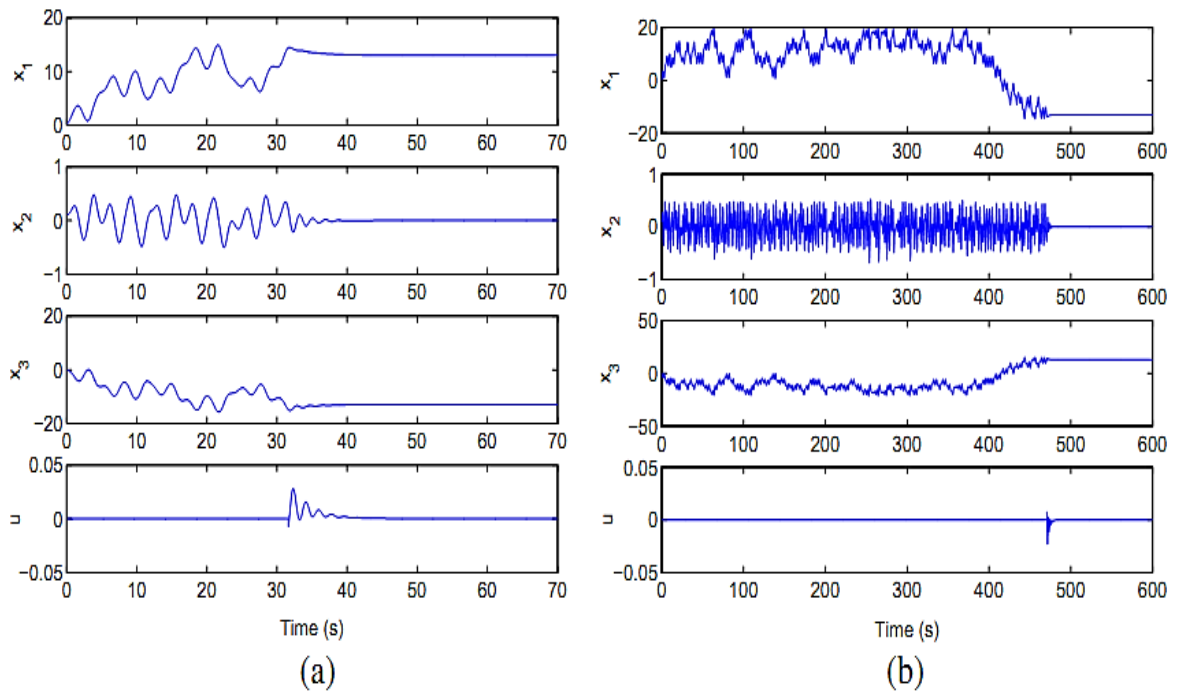


Figure. IV.13 Stabilisation du modèle RN du circuit de Chua sur  $\bar{X}_{\pm 5}$ .

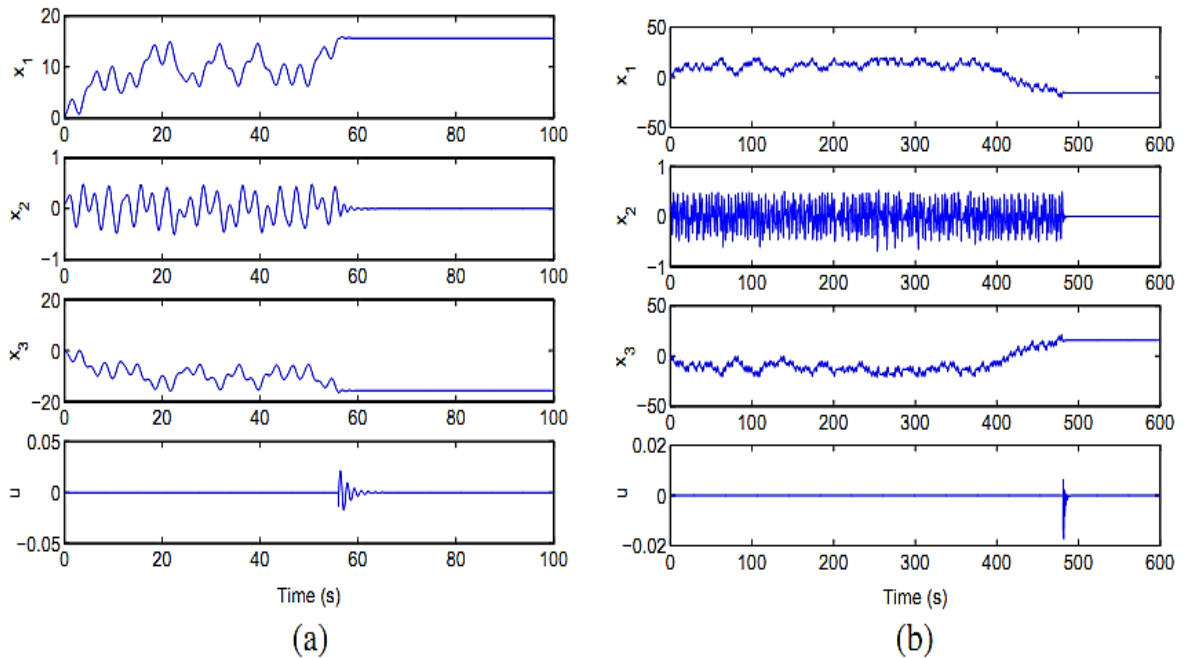
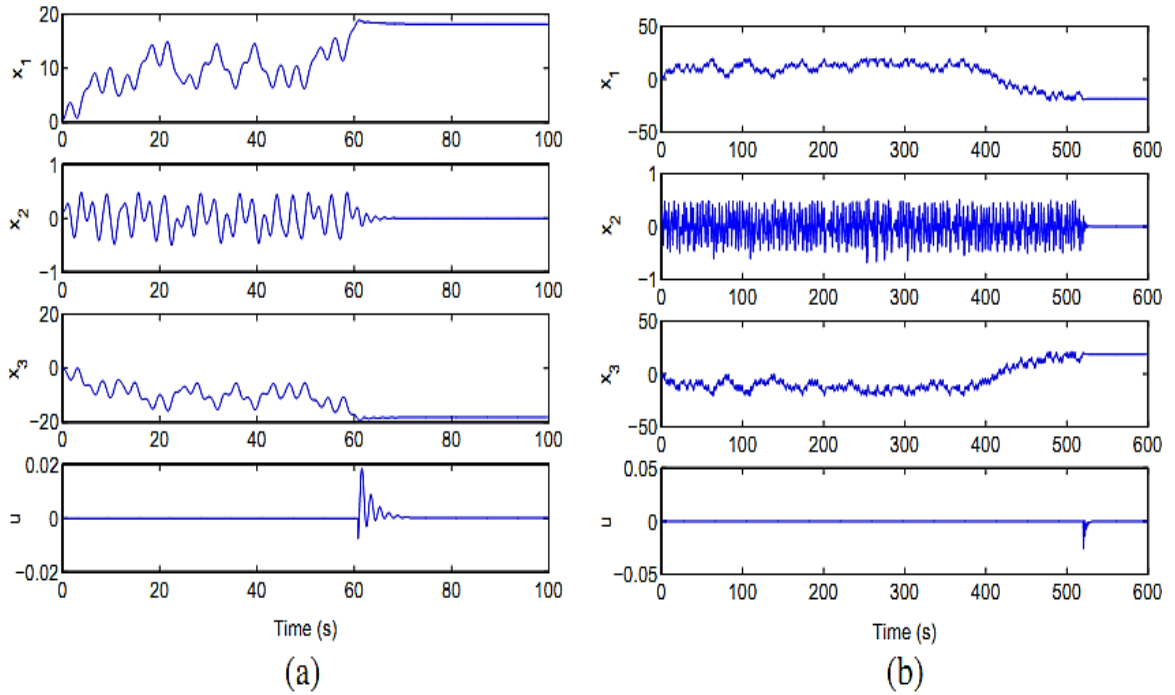


Figure. IV.14 Stabilisation du modèle RN du circuit de Chua sur  $\bar{X}_{\pm 6}$ .





**Figure. IV.15** Stabilisation du modèle RN du circuit de Chua sur  $\bar{X}_{\pm 7}$ .

### 6.2. Stabilisation du modèle RN du système de Chen multi-scroll

La loi de commande basée sur la prédiction intelligente du RN a été construite sous forme de (IV.8) et (IV.14) avec des gains  $F = 10I_{3 \times 3}$  et  $\gamma = 2$ . Les états initiaux du système ont été définie ;  $X(0) = (-3, 2, 20)$ .

Pour  $K = \text{diag}(0, -1.09, 0)$  et pour des valeurs du temps différentes, le modèle neuronal du système de Chen converge vers des points d'équilibre instables  $\bar{X}_{\pm 0}, \bar{X}_{\pm 2}$ , et  $\bar{X}_{\pm 4}$ . Pour  $K = \text{diag}(0, -0.97, 0)$ , Les états du système contrôlé convergent vers les points d'équilibre  $\bar{X}_{\pm 1}, \bar{X}_{\pm 3}$ , et  $\bar{X}_{\pm 5}$ .

Les résultats de simulation sont présentés dans les Figures. IV.16 à IV.21. Évidemment, une performance de poursuite satisfaisante peut être garantie, comme les états du système convergent vers le point d'équilibre souhaité avec une petite force appliquée. Par conséquent, les effets causés par la modélisation ont été réduits.

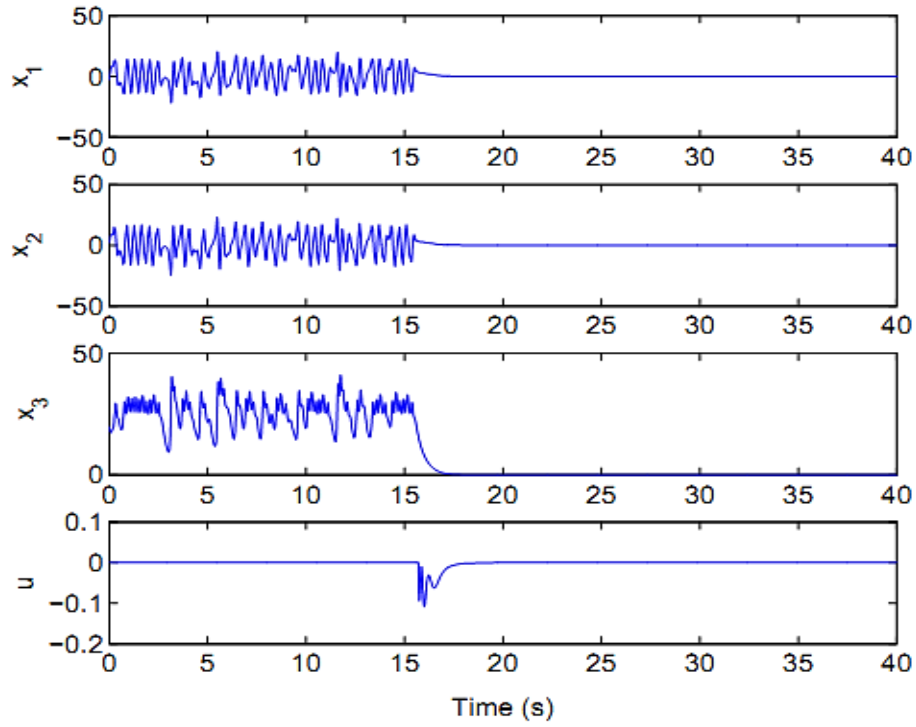


Figure. IV.16 Stabilisation du modèle RN du système de Chen sur  $\bar{X}_0$ .

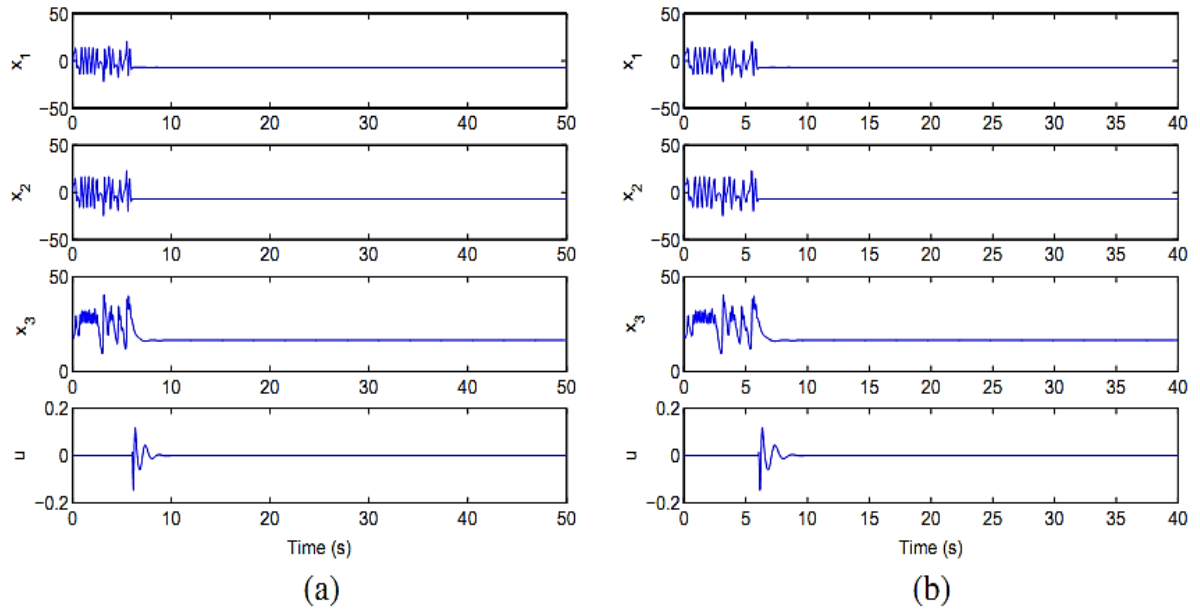
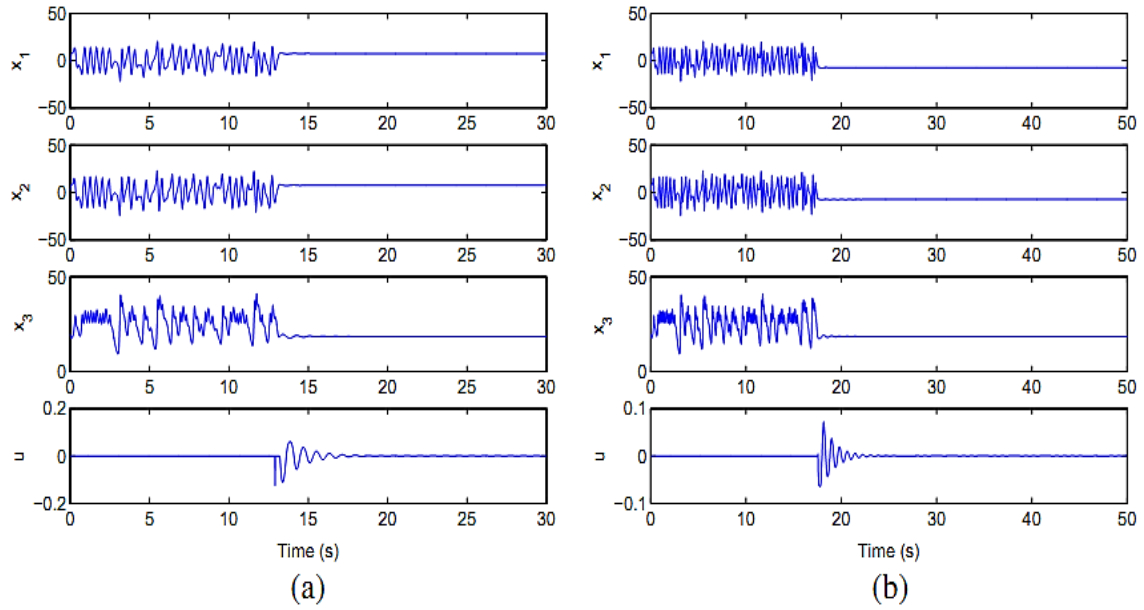
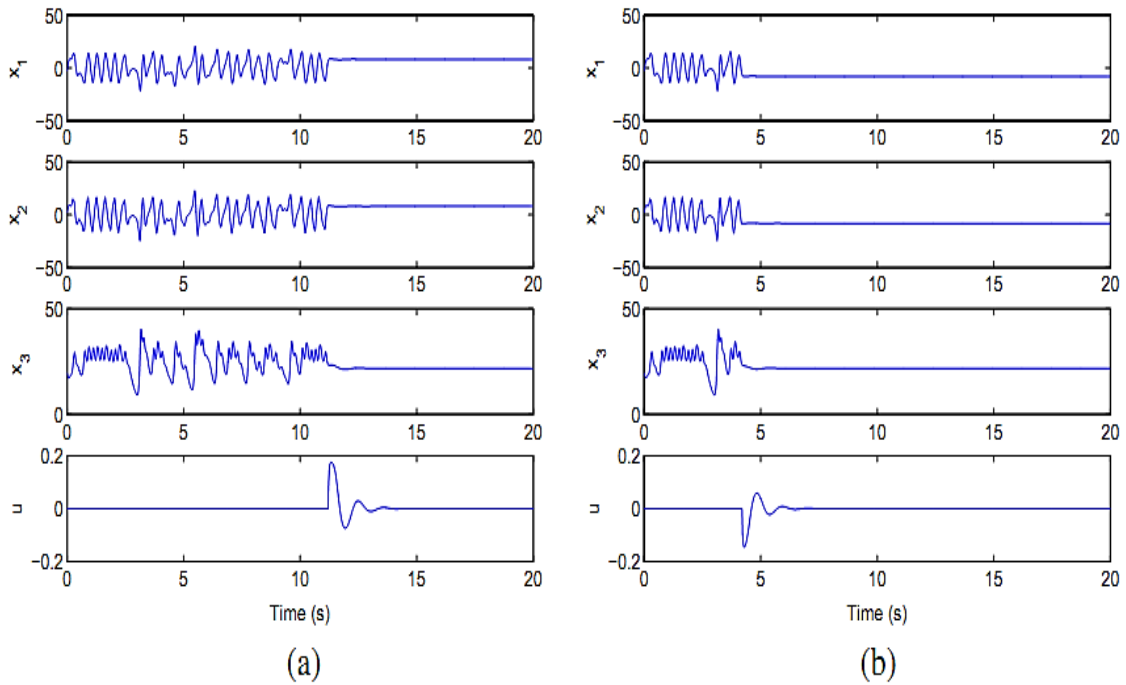


Figure. IV.17 Stabilisation du modèle RN du système de Chen sur  $\bar{X}_{\pm 1}$ .



**Figure. IV.18** Stabilisation du modèle RN du système de Chen sur  $\bar{X}_{\pm 2}$ .



**Figure. IV.19** Stabilisation du modèle RN du système de Chen sur  $\bar{X}_{\pm 3}$ .

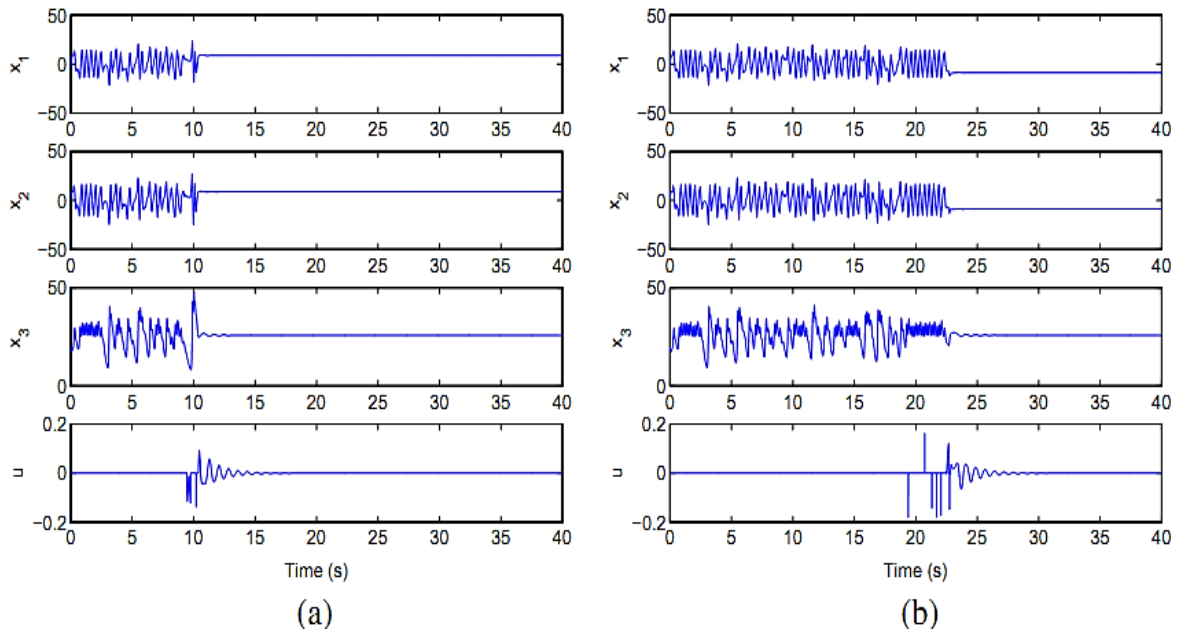


Figure. IV.20 Stabilisation du modèle RN du système de Chen sur  $\bar{X}_{\pm 4}$ .

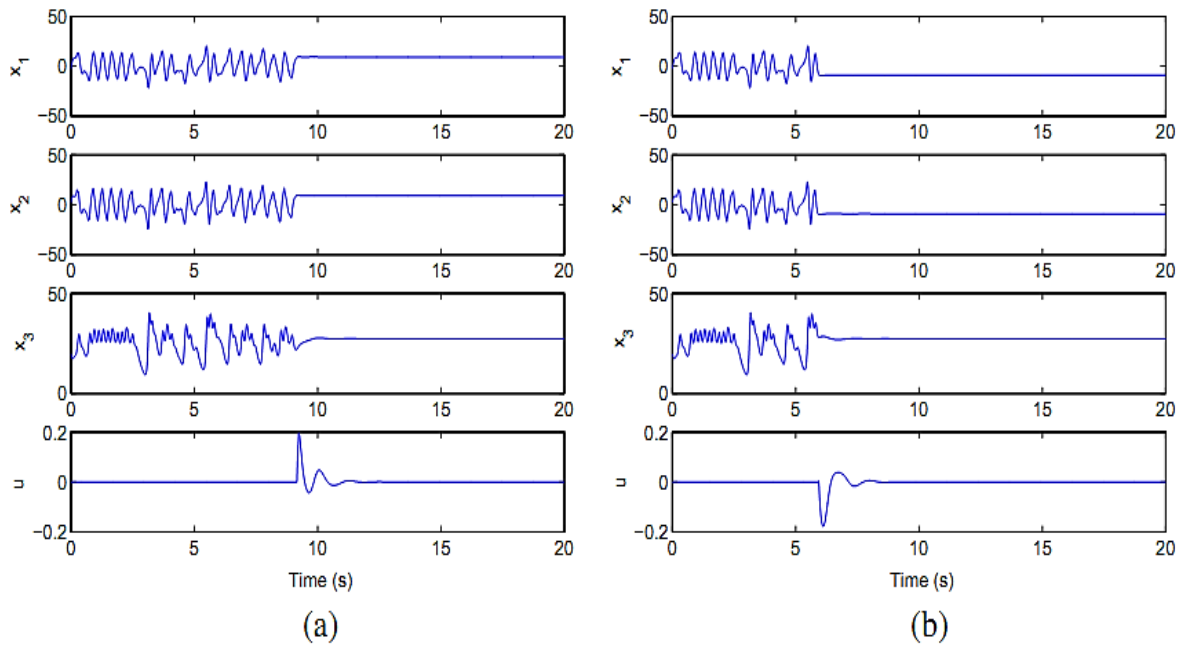


Figure. IV.21 Stabilisation du modèle RN du système de Chen sur  $\bar{X}_{\pm 5}$ .

## 7. Conclusion

Dans ce chapitre, une conception de contrôle et de poursuite, basée sur la prédiction, a été proposée et résolue pour une classe de systèmes chaotiques multi-scroll. Un modèle de RN a été utilisé pour approximer les comportements des dynamiques chaotiques multi-scroll qui incluent des incertitudes et des dynamiques non-modélisées. Par conséquent, un contrôleur prédictif NN-PbC a été développé de telle sorte que tous les états du système en boucle-fermée convergent vers le point d'équilibre souhaité tel que l'entrée de contrôle est aussi petite que possible. La méthode de contrôle proposée est une stratégie de contrôle optimale puisque les entrées de contrôle futures et les réponses futures du système sont prédites à l'aide d'un modèle de RN. La caractéristique essentielle de ce contrôleur est que l'objectif de contrôle est obtenu avec des non-linéarités dans le système chaotique multi-scroll, qui sont complètement inconnus. L'apprentissage adaptatif neuronal proposé montre à la fois la robustesse et l'adaptation à la dynamique variante du système. A cet effet, un signal de commande est incorporé dans le schéma d'apprentissage adaptatif de telle sorte que le système contrôlé obtenu converge vers le point d'équilibre instable désiré.

Par conséquent, pour des applications pratiques, le schéma de contrôle intelligent développé ici peut être utilisé pour traiter une classe plus large de systèmes multidirectionnels en présence d'incertitudes variant dans le temps en degré élevé. Enfin, à partir des résultats de simulation pour divers systèmes, on peut conclure que la conception proposée a atteint les résultats souhaités.

---

## Conclusions et Perspectives

---

---

## Conclusions générales

La majorité des systèmes dynamiques utilisés dans la réalité sont des systèmes non linéaires. Ce type de système est généralement sujet aux variations et incertitudes structurelles et non structurelles causées par l'environnement, d'où le recours aux techniques de l'intelligence computationnelle pour remédier à ce problème. Ces techniques sont connues par leur adaptation aux changements environnementaux.

La commande prédictive est une commande robuste largement utilisée dans l'industrie pour contrôler les systèmes non linéaires. Par contre, cette commande souffre de quelques limites que les chercheurs ont essayé de résoudre par les techniques intelligentes.

Les techniques de l'intelligence computationnelle constituent des solutions possibles à ces limites. Ces techniques comportent les réseaux de neurones, la logique floue parmi d'autres.

L'industrie moderne a besoin d'obtenir les meilleurs résultats pour la commande des systèmes travaillant toujours aux limites des contraintes. Plusieurs méthodes de commande ont été proposées comme éventuelle solution. Dans cette thèse, nous avons développé des techniques basées sur les réseaux de neurones, soit pour la conception du modèle et/ou soit pour la conception du contrôleur. Les RN présentent plusieurs avantages tels que l'approximation globale des solutions, qui peut fournir des solutions efficaces aux problèmes de contrôle des systèmes non linéaires chaotiques. La possibilité d'apprentissage peut réduire l'effort humain lors de la conception des contrôleurs, ainsi, permet de découvrir des structures de contrôle plus efficace que celles déjà connues.

L'utilisation des réseaux de neurones et plus particulièrement le perceptron multicouche PMC permet de modéliser n'importe quelle fonction non linéaire [36].

En commande prédictive non linéaire (CPNL), la prévision de sorties futures n'est pas obtenue par une linéarisation, mais à partir de l'utilisation récursive et successive d'un modèle non linéaire. Ce type de commande peut être appliqué à des procédés difficiles à commander. Ceci est possible grâce à un choix particulier des paramètres de synthèse (Horizon de prédiction, horizon de commande) permettant d'avoir commandes souple et de meilleures performances (l'erreur, temps de réponse, rapidité et respect des contraintes). De plus, cette commande présente aussi un avantage très intéressant qui réside dans l'élimination des effets de toute perturbation.

Les résultats des simulations, de modélisation, de stabilisation et/ou de contrôle, nous permettent de conclure que la technique NN-PBC est plus précise, plus stable, avec un temps de réponse relativement rapide par rapport à des techniques semblable existant dans la littérature.

De plus, nous avons présenté une amélioration de deux algorithmes évolutionnaires pour les utiliser ensuite pour l'apprentissage des réseaux de neurones MLP.

La présente thèse a été dédiée à la résolution des limites de la commande prédictive par l'application des réseaux de neurones multicouches (MLP) et les algorithmes d'optimisation.

En effet, les réseaux de neurones par leurs qualités d'adaptation et d'approximation sont un outil performant pour estimer les paramètres du modèle du système commandé. Ces performances dépendent essentiellement de la qualité de l'algorithme d'apprentissage utilisé.

En fin, nous avons essayé de développer un algorithme d'apprentissage hors ligne en utilisant les algorithmes évolutionnaires. L'espace de recherche des paramètres d'un réseau de neurones (poids, biais) est vaste et limité seulement par la capacité de calcul, ceci conduit forcément l'algorithme d'apprentissage à être piégé dans un ou plusieurs minima locaux. Ayant cela comme problématique secondaire, nous avons amélioré deux algorithmes d'optimisation afin d'accroître leurs effets de diversification : L'idée était d'hybrider avec les algorithmes d'apprentissage à base de gradient avec les algorithmes d'optimisation à base de populations.

En perspective, on se propose de réaliser une commande prédictive englobant tous les aspects évoqués dans ce manuscrit. On propose d'implémenter un système de contrôle réel basé sur un contrôleur RN flou prédictif. De plus, on projette de faire une commande des systèmes chaotiques multi-scroll multidirectionnelles ayant des incertitudes lors de l'évolution avec le temps.



### RÉFÉRENCES BIBLIOGRAPHIQUES

- [1] T. Kapitaniak, *Chaos for Engineers: Theory, Applications, and Control*, Springer Berlin Heidelberg (1998).
- [2] H. Zhang, *Chaos Synchronization and Its Application to Secure Communication*. Thèse de Doctorat, Université de Waterloo, Ontario, Canada, 2010.
- [3] E. Ott, C. Grebogi, J.A. Yorke, Controlling chaos. *Phys. Rev. Lett.* 64, 1196–1199 (1990).
- [4] E. Cherrier, *Estimation de l'état et des entrées inconnues pour une classe de systèmes non linéaires*, Thèse de doctorat, Centre de Recherche en Automatique de Nancy 2006.
- [5] D. Parrochia, *Les Grandes Révolutions scientifiques du XXe siècle*, Paris, Presses Universitaires de France, 1997.
- [6] H. Poincaré, *Science et méthode*, Edition Ernest Flammarion, Paris 1908, pp. 66.
- [7] E. Lorenz, Deterministic nonperiodic flow, *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130-141, 1963.
- [8] S. Smale, Differentiable Dynamical Systems, *Bulletin of the American Mathematical Society*, vol. 73, pp. 1-82, 1967.
- [9] D. Ruelle, F. Takens, On the nature of turbulence, *Communications in Mathematical Physics*, vol. 20, no. 2, pp. 167-192, 1971.
- [10] D. Ruelle, *Hasard et chaos*, Paris, Éditions du Seuil, Collection Points-Odile Jacob 1993.
- [11] R.L. Devaney, *An Introduction to Chaotic Dynamical Systems*, 2nd edn. Boulder, CO: Westview Press, 2003.
- [12] T.S. Parker, L.O. Chua, Chaos : A tutorial for engineers, proceeding of the IEEE vol.75, No.8, 1987, pp. 982-1000.
- [13] B.R. Andrievskii, A.L. Fradkov, Control of Chaos: Methods and Applications, *Automation and Remote Control* Vol. 64, No. 5, 2003, pp. 673-713.
- [14] M.B. Luca, *Apports du chaos et des estimateurs d'états pour la transmission sécurisée de l'information*. Thèse doctorat l'Université de Bretagne Occidentale 2006.
- [15] P. Cvitanov, R. Artuso, R. Mainieri, G. Tanner, G. Vattay, *Chaos: Classical and Quantum*, Niels Bohr Institute, (Feb 26 2002) Disponible sur <ChaosBook.org>.
- [16] T.S.Parker, L.O.Chua, *Practical numerical algorithms for chaotic system*, Springer-verlag, 1989
- [17] H. Schuster Georg. *Deterministic Chaos - an Introduction*, 4ed, Wiley, 2005.
- [18] H.M. Smale, R. Devaney, *Differential equations, dynamical systems and an introduction to chaos*, 2ed, Elsevier Academic Press, 2004.
- [19] A.J. Michaels, *Digital Chaotic Communications*. Thèse de Doctorat, Georgia Institute of Technology, 2009.
- [20] T. Yang, *Impulsive Control theory*, Springer Verlag, Lecture Notes in Control and Information sciences, 2001.
- [21] J. Oden, *Le chaos dans les systèmes dynamique*. Thèse de Doctorat, Université des Sciences et Technologies de Lille, 2007.
- [22] M. Hénon, A two dimensional mapping with a strange attractor, *Commun. Math. Phys.* 50, 69–77, 1976
- [23] T. Matsumoto, A chaotic attractor from Chua's circuit, *IEEE Transactions on Circuits & Systems*, vol.CAS-31, no.12, pp.1055-1058, 1984

## RÉFÉRENCES BIBLIOGRAPHIQUES

- [24] A.I. Khibnik, D. Roose, L.O. Chua, On periodic orbits and homoclinic bifurcations in Chua's circuit with a smooth nonlinearity, *International Journal of Bifurcation & Chaos in Applied Sciences & Engineering*, vol.3, no.2, pp.363-384, 1993.
- [25] L.O.Chua, G-N. Lin, Canonical realization of Chua's circuit family, *IEEE Transactions on Circuits & Systems*, vol. 37, no.7, pp.885-902, 1990.
- [26] J. Guckenheimer, P. Holmes, *Nonlinear Oscillations Dynamical Systems and Bifurcations of Vector Fields*, Springer-Verlag, New York, 1986.
- [27] E. Lorenz. *The Essence of Chaos*. University of Washington Press, 1993.
- [28] Y. Moussa, *Elaboration d'Algorithmes de Masquage pour Les Systèmes de Communication Chaotique*, Thèse de Doctorat, Université Mentouri Constantine, 2012.
- [29] E. Zeraouila, *Etude de quelques types de systèmes chaotiques : généralisation d'un modèle issu du modèle de Chen.* Thèse de Doctorat, Université Mentouri Constantine, 2006.
- [30] W.K.S. Tang, G.Q. Zhong, G. Chen, K.F. Man. Generation of N-scoll attractors via sine function. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Application*, 48(11):1369–1372, 2001.
- [31] G. Luger, F. George, *Artificial Intelligence : Structures & Strategies for complex problem solving*, 4ed, Addison-Wesley, 2002.
- [32] M. Negnevitsky, *Artificial Intelligence : A guide to Intelligent Systems*. Addison-Wesley, 2002.
- [33] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554-2558, 1982.
- [34] W. McCulloch, P. Walte, A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics*, vol. 5, pp.115-133, 1943.
- [35] S.A. Billings, H.B., Jamaluddin, S. Chen, Properties of Neural Networks With Applications to Modelling non-linear Dynamical Systems, *Int. Journal of Control*, vol. 55, No 1, 1992, pp.193-224.
- [36] M.O. LaBarre, [www.uqtr.quebec.ca/~biskri/Personnel/mol/RRN.doc](http://www.uqtr.quebec.ca/~biskri/Personnel/mol/RRN.doc), réseaux de neurones, Mai 2002.
- [37] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, 1958- Repris par J.A. Anderson & E. Rosenfeld, *Neurocomputing. Foundations of Research*, MIT Press, 1988.
- [38] M. Minsky, *An Introduction to Computational Geometry*, 2ed, The MIT Press, Cambridge MA, 1972.
- [39] I. Rivals, *Modélisation et commande par réseaux de neurones : application au pilotage d'un véhicule autonome*, Thèse de Doctorat de l'Université Pierre et Marie Curie, Paris, 1995.
- [40] G. Dreyfus, Y. Idan, The Canonical Form of Non-linear Discrete-Time Models, *Neural Computation* Vol. 10, No. 1, pp.133-164, 1998.
- [41] S. Haykin, *Neural Networks : A Comprehensive Foundation*. Upper Saddle River, N.J. Prentice Hall, 2ed, 1999.
- [42] M.T. Hagan, M.B. Menhaj, Training feedforward networks with the Marquardt algorithm, *IEEE Trans. Neural Networks*, vol. 5, pp. 989-993, 1994.
- [43] T. M. Hagan, H. Demuth, M. Beale, *Neural network design*. Ed PWS publishing company, 1995.
- [44] G. Cybenko, Approximation by superposition of a sigmoid function. *Math. of Control, Signals and Systems*, vol. 2(4), pp.303-314,1989.

## RÉFÉRENCES BIBLIOGRAPHIQUES

- [45] K. Funahashi, On the Approximate Realization of Continuous Mappings by Neural Networks, *Neural Networks*, vol. 2, pp.183-192, 1989.
- [46] K. Hornik, M. Stinchcombe, H. White, Universal approximation of an unknown mapping and its derivatives using multiplayer feedforward networks, *Neural Networks*, vol. 3, pp.551-560, 1990.
- [47] R. Neelakantan, J. Guiver, Applying Neural Networks, *Hydrocarbon Processing*, Gulf Publishing Company, Houston, September, pp.91-96, 1998.
- [48] J. Park, I.W. Sandberg, Universal Approximation using Radial-Basis-Function Networks, *Neural Computation* vol. 3, No. 2, 1991, pp.246-257.
- [49] K. Hornik, M. Stinchcombe, H. White, P. Auer, Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives, *Neural Computation*, vol. 6, pp.1262-1275, 1994.
- [50] I. Rivals. L. Personnaz, G. Dreyfus, J.L.Ploix, *Modélisation, classification et commande par réseaux de neurones: principes fondamentaux, méthodologie de conception et applications industrielles*, Lavoisier Tec et Doc, Paris, 1995.
- [51] J. Moody, C. Darken, Fast learning in network of locally-tuned processing units, *Neural Computation*, vol. 1, 606623, 1995.
- [52] N. Rizkalla, *Nanoparticules et réseaux de neurones artificiels : de la préparation à la modélisation*, Thèse de doctorat de l'université de Montréal Canada, 2005.
- [53] M. Salem, *Approches de l'intelligence artificielle pour la commande robuste des systèmes non linéaires*, Thèse de doctorat, Université d'Oran, 2014.
- [54] A.E. Bryson, C.H. Yu, *Applied Optimal Control. Optimization, Estimation and Control*, by Ginn and Company Waltham, Massachusetts, 1969.
- [55] P. Werbos, *Beyond regression: new tools for prediction and analysis in the behavioural sciences*, PhD thesis, Harvard University, Cambridge, MA, 1974.
- [56] D.B. Parker, *Learning-logic*, Technical Report TR-47, Center for Computational Research in Economics and Management Sci., MIT, April, 1985.
- [57] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing : Explorations in the Microstructures of Cognition*, vol. 1, Cambridge: MIT Press, pp.318-362, 1986.
- [58] Y. Le Cun, Une procédure d'apprentissage pour réseau à seuil asymétrique. In *Cognitiva 85: A la Frontière de l'Intelligence Artificielle des Sciences de la connaissance des Neurosciences*, Paris: CESTA, Paris, pp.599-604, 1985.
- [59] I. Guyon, Neural networks and applications tutorial. *Physics Reports*, 207(3-5), 215-259. 1991.
- [60] R.J. Gayner, T. Downs, On the properties of error functions that affect the speed of backpropagation learning, *Proceeding of the International Conference on Artificial Neural Networks ICANN'94; 26-29 May 1994, Sorrento, Italy*, pp.557-560, 1994.
- [61] B. Hassibi, D.G. Stork, Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, San Mateo, CA, pp. 164-171, 1993.
- [62] E.B. Baum, D. Haussler, What size net gives valid generalization, *Neural Computing*, vol. 1(1), pp.151-160, 1989.
- [63] D. Urbani, *Méthodes Statistiques de Sélection d'Architectures Neuronales: Application à la Conception de Modèles de Processus Dynamiques*, Thèse de Doctorat de l'Université Pierre et Marie Curie, Paris VI, 1995.
- [64] A. Moussaoui, *Contribution à la modélisation et la commande des processus industriels par réseaux de neurones*, Thèse de Doctorat, Université Badji Mokhtar Annaba, 2006.

## RÉFÉRENCES BIBLIOGRAPHIQUES

- [65] Y. Grandvalet, S. Canu, S. Boucheron, Noise injection : Theoretical prospects, *Neural Computation*, Vol. 9, No. 7, pp.1241-1256, 1997.
- [66] R. Lengellé, T. Denoeux, Training MLPs layer by layer using an objective function for internal representations. *Neural Networks*, 9, pp.83-97, 1996.
- [67] Y.Le Cun, B. Boser, J.S. Denker, D. Hendersen, R.E. Howard, W. Hubard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, *Neural Computation*, vol. 1, pp.541-551, 1989.
- [68] P. Thomas, G. Bloch, Robust Pruning for Multilayer Perceptrons. In *Proceedings of IMACS/IEEE Multiconference on Computational Engineering in Systems Applications CESA'98*, Vol. 4, Nabeul-Hammamet, Tunisia, April 1-4, pp.17-22, 1998.
- [69] A.S. Weigend, D. E. Rumelhart, The effective dimension of the space of hidden units, *IEEE International Joint Conference on Neural Networks*, 1991.
- [70] R. Fletcher, An Ideal Penalty Function for Constrained Optimization, *J. Inst. Maths Applics.*, vol.15, p.319-342, 1975.
- [71] A. F. Izmailov, M.V. Solodov, *Newton-Type Methods for Optimization and Variational Problems*. Springer Series in Operations Research and Financial Engineering. Springer, 2014.
- [72] V.T. Paschos, *Optimisation combinatoire : concepts fondamentaux*. Hermes science publication, 2005.
- [73] M. Mitchell, *An Introduction to Genetic Algorithms*. Prentice-Hall, 1998.
- [74] P. Jog, J. Suh, G. Gucht, Parallel genetic algorithms applied to the traveling salesman problem. *SIAM Journal on Optimization*, 1(4) :515-529, 1991.
- [75] M.Crepinsek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys*, 45 pp. 1-33, 2013.
- [76] G. Berthiau, P. Siarry, État de l'art des méthodes "d'optimisation globale". *RAIRO-Operations Research*, 35(3), 329-365, 2001.
- [77] C.A.L. Bailer-Jones, D.J.C. MacKay, A recurrent neural network for modelling dynamical systems, *network: computation in neural systems*, vol.9, pp.531-547, 1998.
- [78] D. Urbani, P. Roussel-Ragot, L. Personnaz, G. Dreyfus, The selection of neural models of non-linear dynamical systems by statistical tests, *Neural Networks for Signal Processing, Proceedings of the 1994 IEEE Workshop*, pp. 229-237, 1994.
- [79] L. Guessas, *Backstepping Backstepping adaptatif pour le contrôle la poursuite et la synchronisation des systèmes dynamiques non linéaires chaotiques*, Université Ferhat Abbas Sétif, 2012.
- [80] W.L. Ditto, S.N.Rauseo, M.L.Spano, Experimental Control of Chaos, *Phys. Rev. Lett.*, vol. 65, pp. 3211-3214, 1990.
- [81] A.L. Fradkov, R.J. Evans, Control of Chaos: Some Open Problems, *Proceeding. 40<sup>th</sup> IEEE Conference, Control, Orlando*, pp. 698-703, 2001.
- [82] .L. Fradkov, R.J. Evans, Control of Chaos: Survey, *Preprints of 15<sup>Th</sup> Triennial World Congress IFAC, Plenary Papers, Survey Papers, Milestones, Barcelona*, pp. 143-154, 2002.
- [83] M.A. Khelifa, A. Boukabou, Design of an intelligent prediction-based neural network controller for multi-scroll chaotic systems. *Applied Intelligence*, vol. 45, Issue 3, pp 793–807. doi:10.1007/s10489-016-0793-z, 2016
- [84] M.A. Khelifa, A. Boukabou, Control of UPOs of unknown chaotic systems via ANN. In *International Conference on Artificial Neural Networks*, pp. 627-634, Springer International Publishing, 2014.

## RÉFÉRENCES BIBLIOGRAPHIQUES

- [85] M.A. Khelifa, A. Boukabou, Secure Data Transmission of Holy Quran Using Controlled Chaotic Dynamics. In IEEE Conference on Advances in Information Technology for the Holy Quran and Its Sciences (32519), Taibah University International Conference on (pp. 47-51), 2013.
- [86] A.L. Fradkov, A.Yu. Pogromsky, Introduction to Control of Oscillations and Chaos, Singapore World Scientific, 1998.
- [87] W.J. Freeman, H.J. Chang, B.C. Burke, et al., Taming Chaos : Stabilization of Aperiodic Attractors by Noise, IEEE Transactions Circuit Systems I, vol. 44, pp. 989-996, 1997.
- [88] L. Fronzoni, M. Giocondo, M. Pettini, Experimental Evidence of Suppression of Chaos by Resonant Parametric Perturbations, Physics Review A, vol. 43, pp. 6483-6487, 1991.
- [89] M. Funke, M. Herrmann, R. Der, Controlling Low-dimensional Chaos : Determination and Stabilization of Unstable Periodic Orbits by Kohonen Neural Nets, International Journal of Adaptive Control Signal Processing, vol. 11, pp. 489-499, 1997.
- [90] A.Yu. Loskutov, Problems of Nonlinear Dynamics. II. Suppression of Chaos and Control of Dynamic Systems, Vestn. MGU, no. 2, pp. 3-21, 2001.
- [91] N.A. Magnitskii, S.V. Sidorov, Control of Chaos in Nonlinear Dynamic Systems, Diff. Uravn., no. 11, pp. 1501-1509, 1998.
- [92] N.A. Magnitskii, On Stabilization of Fixed Points of Chaotic Maps, Dokl. Ross. Akad. Nauk, vol. 351, no. 2, pp. 175-177, 1996.
- [93] L.M. Pecora, T.L. Carroll, Synchronization in chaotic systems, Physics Review Letter, vol. 64, no 8, pp 821-824, 1990.
- [94] L.M. Pecora, T.L. Carroll, Synchronizing chaotic circuits, IEEE Trans. Circuits Syst. vol. 38,453-456, 1991.
- [95] L.M. Pecora, T.L. Carroll, Synchronizing nonautonomous chaotic circuits, IEEE Trans Circuits Syst 2, Special Issue on Chaos in Nonlinear Electronic Circuits Part C: Applications, vol. 40, pp 646-650, 1993.
- [96] A. Garfinkel, M. Spano, W.L. Ditto, et al., Controlling Cardiac Chaos, Science, volume. 257, pp. 1230-1235, 1992.
- [97] G. H. Li, S. P. Zhou, K. Yang, Controlling chaos in Colpitts oscillator, The 2003 IEEE International Symposium on circuits and systems, Bangkok, Thailand. [ieeexplore.ieee.org/iel15/8570/27140/0125756](http://ieeexplore.ieee.org/iel15/8570/27140/0125756).
- [98] C. Grebogi, Y.C. Lai, S. Hayes, Control and Applications of Chaos, Int. J. Bifurcat. Chaos vol. 7, pp. 2175-2197, 1997.
- [99] M.C. Mackey, L. Glass, Oscillation and Chaos in Physiological Control Systems, Science vol. 197, pp. 287-280, 1977.
- [100] A.V. Naumenko, N.A. Loiko, S.I. Turovets, et al., Bias Current Impulsive Feedback Control of Nonlinear Dynamics in External Cavity Laser Diodes, Electron. Lett., vol. 34, pp. 181-182, 1998.
- [101] K. Matsumoto, I. Tsyda, Noise-induced Order, J. Stat. Phys. vol. 31, pp.87-106, 1983.
- [102] K. Hirasawa, J. Murata, J.L. Hu, et al., Chaos Control on Universal Learning Networks, IEEE Trans. Syst. Man Cybern. C-Appl. Rev., vol. 30, pp. 95-104, 2000.
- [103] I.Z. Kiss, V. Gaspar, Controlling Chaos with Artificial Neural Network : Numerical Studies and Experiments, J. Phys. Chem. A, vol. 104, pp. 8033-8037, 2000.
- [104] D.P.A. Greenwood, R.A. Carrasco, Neural Networks for the Adaptive Control of Disruptive Nonlinear Network Traffic, IEE Proc. Commun., vol. 147, pp. 285-291, 2000.
- [105] C.T. Lin, C.P. Jou, Controlling Chaos by GA-Based Reinforcement Learning Neural Network, IEEE Trans. Neural Netw, vol. 10, pp. 846-859, 1999.
- [106] K. Pyragas, Continuous control of chaos by self-controlling feedback. Phys. Lett. 1992.

## RÉFÉRENCES BIBLIOGRAPHIQUES

- [107] H. Richter, K.J Reinschke, Optimization of local control of chaos by an evolutionary algorithms, *Physica D* 144, 309–334, 2000.
- [108] A. Boukabou, B. Sayoud, H. Boumaiza, N. Mansouri, Control of n-scroll Chua's circuit. *Int. J. Bifurcation Chaos* 19, 3813–3822 (2009).
- [109] W.M. Bessa, A.S. de Paula, M.A. Savi, Chaos control using adaptive sliding mode controller with application to a nonlinear pendulum. *Chaos Solitons Fractals* 42, 784–791 2009.
- [110] M. Han, J. Xi, S. Xu, F. Yin, Prediction of chaotic time series based on the recurrent predictor neural network. *IEEE Trans. Signal Proc.* 52, 3409–3416, 2004.
- [111] L. Cao, Y. Hong, H. Fang, G. He, Predicting chaotic time series with wavelet networks. *Physica D* 85, 225–238, 1995.
- [112] H. Leung, T. Lo, S. Wang, Prediction of noisy chaotic time series using an optimal radial basis function neural network. *IEEE Trans. Neural Networks* 12, 1163–1172, 2001.
- [113] Y. Becerikli, Y. Oysal, Modeling and prediction with a class of time delay dynamic neural networks. *Appl. Soft Comput.* 7, 1164–1169, 2007.
- [114] L. Shen, M. Wang, W. Liu, G. Sun, Prediction based chaos control via a new neural network. *Phys. Lett. A* 372, 6916–6921, 2008.
- [115] D. Lin, X. Wang, F. Nian, Y. Zhang, Dynamic fuzzy neural networks modeling and adaptive backstepping tracking control of uncertain chaotic systems. *Neurocomputing* 73, 2873–2881, 2010.
- [116] J. Nocedal, S.J. Wright, *Numerical optimization*, 2nd Springer, NY, 2006.
- [117] S. Sastry, *Nonlinear systems: Analysis, stability, and control*. Springer, Berlin, 1999.
- [118] F. Xu, P. Yu, Chaos control and chaos synchronization for multi-scroll chaotic attractors generated using hyperbolic functions. *J Math Anal Appl* 362:252–274, 2010.
- [119] W.M. Ahmad, Generation and control of multi-scroll chaotic attractors in fractional order systems. *Chaos Solitons Fractals* 25:727–735, 2005.
- [120] A. Boukabou, B. Sayoud, H. Boumaiza, N. Mansouri, Control of n-scroll Chua's circuit. *Int. J. Bifurcation Chaos* 19, 3813–3822 (2009).
- [121] S. Hadeif, A. Boukabou, Control of multi-scroll Chen system. *J Franklin Inst* 351:2728–2741, 2014.
- [122] D. Pham, X. Liu, *Neural networks for identification, prediction and control*. Springer, London, 1995.
- [123] E.R. Weeks, J.M. Burgess, Evolving artificial neural networks to control chaotic systems. *Phys Rev E* 56:1531–1540, 1997.
- [124] W. Qin, Y. Yang, J. Zhang, Controlling the chaotic response to a prospective external signal using back-propagation neural networks. *Nonlinear Anal Real World Appl* 10:2985–2989, 2009.
- [125] A.S. Poznyak, W. Yu, E.N. Sanchez, Identification and control of unknown chaotic systems via dynamic neural networks. *IEEE Trans Circuits Syst. I* 46:1491–1495, 1999.
- [126] Z. Lu, L.S. Shieh, G. Chen, N.P. Coleman, Adaptive feedback linearization control of chaotic systems via recurrent high-order neural networks. *Inf. Sci.* 176:2337–2354, 2006.
- [127] M. Turk, H. Ogras, Recognition of multi-scroll chaotic attractors using wavelet-based neural network and performance comparison of wavelet families. *Expert Syst Appl* 37:8667–8672, 2010.
- [128] M. Turk, A. Gulden, Modelling and simulation of the multi-scroll chaotic attractors using bond graph technique. *Simul Model Pract Theory* 19 pp. 899–910, 2011.
- [129] A. Boukabou, A. Chebbah, N. Mansouri, Predictive control of continuous chaotic systems. *Int J Bifurcat Chaos* 18:587–592, 2008.

## RÉFÉRENCES BIBLIOGRAPHIQUES

- [130] A. Senouci, A. Boukabou, Predictive control and synchronization of chaotic and hyper-chaotic systems based on a T-S fuzzy model. *Math Comput Simul* 105:62–78, 2014.
- [131] Y. Zheng, Fuzzy prediction-based feedback control of fractional order chaotic systems. *Int. J Light Electron Optics* 126:5645–5649, 2015.
- [132] J.A. Suykens, J. Vandewalle, Generation of n-double scrolls ( $n = 1, 2, 3, 4, \dots$ ). *IEEE Trans Circuits Syst I* 40(11):861–867, 1993.
- [133] M.E. Yalcin, J.A.K. Suykens, J. Vandewalle, S. Ozoguz, Families of scroll grid attractors. *Int J Bifurcat Chaos* 12:23–41, 2002.
- [134] Y. Lin, C. Wang, H. He, A simple multi-scroll chaotic oscillator employing CCIs. *Int J Light Electron Opt* 126:824–827, 2015.
- [135] E. Tlelo-Cuautle, J.J. Rangel-Magdaleno, A.D. Pano-Azucena, P.J. Obeso-Rodelo, J.C. Nunez-Perez, FPGA realization of multi-scroll chaotic oscillators. *Commun Nonlinear Sci Numer Simul* 27:66–80, 2015.
- [136] L. Grune, J. Pannek, *Nonlinear model predictive control: theory and algorithms*. Springer, London, 2011.
- [137] T. Ushio, S. Yamamoto, Prediction-based control of chaos. *Phys Lett A* 264:30–35, 1999.
- [138] H. Khalil, *Nonlinear systems*. Prentice Hall, New Jersey, 2002.
- [139] A.R. Barron, Universal approximation bounds for superposition of a sigmoidal function. *IEEE Trans Inform Theory* 39:930–945, 1993.
- [140] N. Draper, H. Smith, *Applied linear regression*, 2ed. Wiley, New York, 1981.