REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ MOHAMED BOUDIAF - M'SILA

FACULTÉ : TECHNOLOGIE

DEPARTEMENT : ELECTRONIQU

DOMAINE: Sciences et Technologies

FILIÈRE : télécommunications

OPTION : Systèmes des télécommunications

Mémoire présenté pour l'obtention du diplôme de Master Académique

Par :

- OUALI Marwa
- BEN MESLI Amina

Intitulé

Modélisation et identification des systèmes dynamiques par les réseaux de neuronesartificielset les algorithmes méta-heuristiques

Soutenu publiquement le / / devant le jury composée de :

- KHENNOUF Salah	***President***
- OUALI Mohammed Assam	***Encadreur***
- LADJAL Mohamed	***Co-encadreur***
- DJERIOUI Mohamed	***Examinateur***

Promotion : 2019/2020

Remerciement :

Tout d'abord nous remercions les encadreurs Dr. LADLAL Mohamed et Dr. OUALI Mohammed Assam, qui ont acceptés de nous encadrer pour réaliser ce mémoire. Ils ont le mérite, après Dieu Tout-Puissant, pour la recherche et le chercheur puisque le sujet était un titre et une idée jusqu'à ce qu'il devienne un mémoire et une recherche, Nous leurs sommes reconnaissants de nous permettre de bénéficier de leur grande efficacité, de leurs précieux conseils et de notre assistance continue à tout moment et sans restriction.

Nous remercions également les membres du jury qui nous ont honorés en jugeant cette lettre et en venant annoncer sa valeur scientifique

Nous remercions tous ceux qui ont contribué directement ou indirectement à notre aide.

À toutes nos familles et amis ...

TABLE DES MATIERES

Chapitre III :Algorithmes d'optimisation

Chapitre IV : Modélisation neuronale des systèmes dynamiques

LISTE DES FIGURES

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Introduction générale

Un système dynamique est un système qui transforme dans le temps une fonction de façon causale et déterministe, il désigne généralement la branche de recherche active des mathématiques, a la frontière de la topologie, de l'analyse, de la géométrie, de la théorie de la mesure et des probabilités. Les systèmes dynamiques sont néanmoins apparus assez tôt dans l'histoire scientifique puisqu'on peut les reconnaitre dans les premiers travaux de la mécanique donnant lieu à des équations différentielles.

Les principales difficultés dans la théorie de la commande des systèmes dynamiques réels sont les non-linéarités et les incertitudes. Or la commande passe par l'élaboration d'un modèle mathématique du système en trouvant une relation entre les entrées et les sorties, ce qui suppose une bonne connaissance de la dynamique du système et ses propriétés [1].

La problématique qui se pose est l'incapacité de prévenir le fonctionnement des systèmes dynamiques après une certaine période et pour le résoudre on peut remplacez cette fonction par un réseau neuronal.

L'application des techniques neuronales pour l'identification et le contrôle des systèmes non- linéaires peut fournir de nouvelles solutions pour ce problème, qui est l'identification neuronale et le contrôle neuronal, alors le mot « identification » est un ensemble de techniques dont l'objet est la détermination de modèle d'attitude d'un procédé physique à partir de mesures caractéristiques de son fonctionnement dynamique.

En effet, parmi les modèles utilisés pour l'identification, on trouve les réseaux de neurones, que ce soit statiques ou dynamiques, et pour utiliser un réseau de neurones nous devons savoir la définition de sa structure. Il est souvent nécessaire de construire des modèles à partir de données réelles (entrées-sorties), capables d'apprendre la dynamique

du système. Mais le nombre de couches cachées et le nombre optimal de neurones dans chaque couche ne peuvent être fixés a priori.

De plus, si l'identification des systèmes linéaires est relativement bien maîtrisée, l'identification des systèmes non-linéaires reste un problème principal. Actuellement, plusieurs approches ont été proposées pour utiliser les réseaux de neurones dans le cadre de l'identification des systèmes dynamiques. Toutefois, ces méthodes sont principalement basées sur une modélisation de type "boîte noire", qui suppose que le processus à modéliser peut-être décrit par des modèles entrée-sortie ; ce qui signifie qu'ils ont une complexité de calcul et de mise en œuvre [2].

L'objectif de ce travail est de mettre en évidence en premier lieu la modélisation des systèmes dynamique et le notions de base relative à la modélisation des processus. Puis nous allons présenter une structure de modélisation des systèmes dynamiques à basedes réseaux de neurones artificiels et les algorithmes d'optimisation.

Le présent mémoire est organisé autour de quatre chapitres :

- **Le premier chapitre** donne un aperçu général sur les processus et leurs modélisations.
- **Le deuxième chapitre** détaille d'une façon simple les réseaux de neurones artificiels, en commençant par le domaine d'application, et en terminant par l'architecture et les applications des réseaux de neurones artificiels.
- **Le troisième chapitre** est consacré à présenter les notions de base relatives à l'optimisation ainsi que les différents algorithmes d'optimisation.
- **Le Chapitre quatre** sera consacré **à** présenter une structure de modélisation des systèmes dynamiques à base des réseaux de neurones artificiels et les algorithmes d'optimisation, ainsi que les résultats de simulation trouvés.

Nous terminons notre mémoire par une conclusion générale, où nous rappellerons les principaux résultats obtenus dans cette étude.

CHAPITRE I

Processus&Modélisation

La modélisation des systèmes dynamiques, elle a pour objectif de trouver une représentation mathématique, aussi simple et compacte que possible, qui permet de rendre compte les relations existantes entre leurs entrées et leurs sorties. Nous exposons dans ce chapitre les notions de base relatives à la modélisation des processus.

Sommaire

1. Introduction

Les systèmes réels sont difficiles a étudié, donc on est amené à les modéliser mathématiquement pour pouvoir les commander [1].La modélisation d'un processus consiste à trouver une description mathématique de son fonctionnement, qui permet de rendre compte les relations existantes entre ses entrées et ses sorties, et qui sont habituellement représentées par des équations. Si ces équations sont algébriques, le modèle est dit statique. Si ces équations sont des équations différentielles ou des équations aux différences récurrentes, le modèle est dit dynamique, respectivement à temps continu ou à temps discret [2].

2. Définition d'un processus et d'un modèle

2.1.Processus

Un processus est un système dynamique qui évolue dans le temps. Du point de vue d'un observateur, un processus correspond à un système physique envisagé dans le cadre de l'évolution des échanges réalisés avec son environnement**.**

Un processus est caractérisé par **:**

- Une ou plusieurs grandeurs de sortie, mesurables, qui constituent le résultat du processus,
- Une ou plusieurs grandeurs d'entrée (ou facteurs), qui peuvent être de deux types :
  - des entrées sur lesquelles il est possible d'agir (entrées de commande),
  - des entrées sur lesquelles il n'est pas possible d'agir (perturbations) ; ces dernières peuvent être aléatoires ou déterministes, mesurables ou non mesurables.

Les processus peuvent être de toutes natures : physique, chimique, biologique, écologique, financier, sociologique, …. etc [3].

Dans le cas général, un processus est un système dynamique traversé par des flux d'informations, d'énergie et de matière tout en étant soumis à des perturbations ayant l'une des trois formes précitées. La figure 1, fournit un exemple d'une telle représentation.

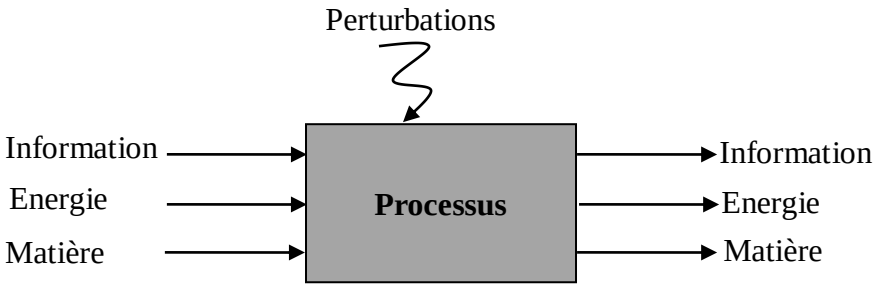


Figure 1 : Schéma synoptique d'un processus.

L'étude des processus a généralement pour objectif de :

- ✓ Pouvoir comprendre le fonctionnement du système et prévoir son comportement et ses performances face à une variation des entrées ; on parle alors d'analyse du système,
- ✓ Chercher à maîtriser les sorties et les performances du système en agissant sur les entrées ; il s'agit, dans ce cas, de synthèse de lois de commande.

2.2.Modèles

2.2.1. Qu'est-ce qu'un modèle ?

Un modèle est caractérisé par son domaine de validité, c'est-à-dire par le domaine de l'espace des entrées dans lequel l'accord entre les valeurs des sorties du processus calculées par le modèle, et leurs valeurs mesurées, est considéré comme satisfaisant compte tenu de l'utilisation que l'on fait du modèle**.**Un modèle est dit statique s'il est représenté par des équations mathématiques algébriques. Si ces équations sont des équations différentielles ou des équations aux différences récurrentes, le modèle est dit dynamique, respectivement à temps continu ou à temps discret**.**

2.2.2. Modélisation statique et modélisation dynamique :

- Un système statique a une sortie $y(t)$ indépendante des valeurs antérieures de l'entrée $u(T)$ avec $T<t$, pour tout t. La modélisation statique est un modèle qui réalise une relation algébrique entre ses entrées et ses sorties. Ce modèle est utilisé pour prédire les états stationnaires de la sortie d'un processus industriel ou pour relier des grandeurs qui sont indépendantes du temps [2].
- Un système dynamique a une sortie $y(t)$ qui dépend des valeurs antérieures de l'entrée $u(T)$ avec $T< t$**.** Pour modélisation dynamique **l**es entrées et les sorties sont reliées entre elles ; soit par des équations différentielles (modèle à temps continu) soit par des équations récurrentes ou aux différentielles (modèle à temps discret) Le pont diviseur de tension est donc un système statique, alors que le système masse-ressort est un système dynamique, qui nécessite de garder

en mémoire les valeurs antérieures de son entrée. Est également possible de classer les systèmes dynamiques dans deux catégories [2].

2.2.3. Buts d'une modélisation

Un modèle peut être utilisé soit **:**

- Pour simuler un processus **:** à des fins pédagogiques, de détection d'anomalies de fonctionnement, de diagnostic de pannes, de conception assistée par ordinateur, etc.,
- Pour effectuer la synthèse d'une loi de commande, ou pour être incorporé dans un dispositif de commande.

2.2.4. Classification des modèles

a. *Classification selon le mode de conception*

On distingue trois sortes de modèles en fonction des informations mises en jeu pour leur conception.

- ✓ **Les modèles de connaissance :** les modèles de connaissance sont construits à partir d'une analyse physique, chimique, biologique (ou autre suivant le type du processus), en appliquant soit les lois générales, fondées sur des principes (lois de la mécanique, de l'électromagnétisme, de la thermodynamique, de la physique quantique, etc.), soit les lois empiriques (finance, économie), qui régissent les phénomènes intervenant au sein des processus étudiés. Ces modèles ne comportent généralement pas de paramètres ajustables, ou des paramètres ajustables en très petit nombre**.** Dans la pratique, il est toujours souhaitable d'établir un modèle de connaissance des processus que l'on étudie. Néanmoins, il arrive fréquemment que le processus soit trop complexe, ou que les phénomènes qui le régissent soient trop mal connus, pour qu'il soit possible d'établir un modèle de connaissance suffisamment précis pour l'application considérée. On est alors amené à concevoir des modèles purement empiriques, fondés exclusivement sur les résultats de mesures effectuées sur le processus.
- ✓ **Les modèles « boîte noire » :** les modèles "boîte noire" sont construits essentiellement sur la base de mesures effectuées sur les entrées et les sorties du processus à modéliser. La modélisation consiste alors à utiliser,

pour représenter les relations entre les entrées et les sorties, des équations (algébriques, différentielles, ou récurrentes) paramétrées, et à estimer les paramètres, à partir des mesures disponibles, de manière à obtenir la meilleure précision possible avec le plus petit nombre possible de paramètres ajustables. Le domaine de validité d'un tel modèle ne peut pas s'étendre au-delà du domaine des entrées qui est représenté dans les mesures utilisées pour l'apprentissage.

- ✓ **Les modèles « boîte grise » :** lorsque des connaissances, exprimables sous forme d'équations, sont disponibles, mais insuffisantes pour concevoir un modèle de connaissance satisfaisant, on peut avoir recours à une modélisation "boîte grise" (ou modélisation semi physique) qui prend en considération à la fois les connaissances et les mesures. Une telle démarche peut concilier les avantages de l'intelligibilité d'un modèle de connaissance avec la souplesse d'un modèle comportant des paramètres ajustables.

b. ***Classification selon l'utilisation.***

Indépendamment de la classification précédente, on peut distinguer deux types de modèles en fonction de l'utilisation qui en est faite.

- ✓ **Les modèles de simulation (ou simulateurs) :** un modèle de simulation est utilisé de manière indépendante du processus qu'il représente. Il doit donc posséder un comportement aussi semblable que possible à celui du processus. De tels modèles sont utilisés pour valider la conception d'un système avant sa fabrication (conception assistée par ordinateur en mécanique, en microélectronique, ..), pour la formation de personnels (simulateurs de vols), pour la prévision à long terme, etc.Du point de vue de la structure du modèle, les sorties passées, mesurées sur le processus à modéliser, ne peut constituer des entrées du modèle. L'estimation des paramètres et l'utilisation du modèle constituent deux phases successives et distinctes (apprentissage non adaptatif)**.**
- ✓ **Les modèles de prédiction (ou prédicateurs) :** un modèle de prédiction est utilisé en parallèle avec le processus dont il est le modèle. Il prédit la sortie du processus à une échelle de temps courte devant les constantes de temps du processus. Les prédicateurs sont utilisés pour la synthèse de lois de

commande, ou dans le système de commande lui-même (commande avec modèle interne). Du point de vue de la structure du modèle, les sorties passées, mesurées sur le processus, peuvent constituer des entrées du modèle. L'estimation des paramètres et l'utilisation du modèle peuvent être effectuées simultanément si nécessaire (apprentissage adaptatif, utile notamment si les caractéristiques du processus dérivent dans le temps).

3. Choix du modèle d'un processus

La détermination d'un modèle mathématique d'un processus nécessite en général diverses hypothèses simplificatrices afin de limiter sa complexité. Dans chaque application, il apparaît nécessaire de faire un compromis entre la finesse et la précision du modèle à mettre en œuvre d'une part, et la limite de complexité admissible, compte tenu des objectifs fixe, d'autre part. Nous nous intéressons ici essentiellement aux processus pour lesquels les variables caractéristiques sont susceptibles de prendre un ensemble continu de valeurs appartenant à des intervalles fixés. De plus nous ne traiterons que des modèles de processus à paramètres localisés, c'est-à-dire, décrits par des équations différentielles ordinaires ou des équations récurrentes.

Il est important de toujours se souvenir que le choix du modèle d'un processus dépend de l'utilisation prévue pour ce modèle.

3.1.Modèle à temps continu

Dans ce type de modèle, le temps est une variable qui évolue continûment sur un intervalle $T \subset R$ prenant en croissant toutes les valeurs situées dans cet intervalle. C'est le cas par exemple d'un modèle dans l'espace d'état de la forme :

$$\begin{aligned} \dot{x} &= f(x,u,t,v) \\ y &= h(x,u,t,w) \end{aligned}, (1)$$

où $\dot{x}$ représente la dérivée totale dx/dt du vecteur d'état $x \in R^n$ par rapport au temps, $u \in R^l$ le vecteur d'entrée, $t \in T$ le temps, $v \in R^{n_v}$ un vecteur de perturbations, et $y \in R^m$ le vecteur des sorties.

Dans cette représentation on a :

$$f : R^n \times R^l \times T \times R^{n_v} \to R^n,$$
$$h : R^n \times R^l \times T \times R^{n_v} \to R^m, \qquad (2)$$
$$T = [t_0, t_0 + T_e], t_0 \in R, t_0 \prec\prec \infty, T_e \notin R^+$$

Dans beaucoup de problème, on prend $t_0 = 0$ ou $t_0 \succ 0$ mais cette restriction n'est en aucun nécessaire a priori.

3.2.Modèles à temps discontinu ou discret

Ce type de modèle est utilisé soit lorsque le processus étudié a naturellement un mode d'évolution séquentiel, soit lorsque-on a adopté une représentation du processus qui correspond à une observation des variables d'état ou de sortie à des instants discrets t_k du temps avec $k \in z$.

On a dans ce cas une représentation dite échantillonnée, particulièrement bien adoptée aux divers types de calculs à effectuer dans les problèmes de commande de processus par calculateur numérique.

notant :

$$x_k = x(t_k) \quad (3)$$

Alors, la description continue devient :

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, k, v_k), \\ y_k &= h(x_k, u_k, k, w_k). \end{aligned} \quad (4)$$

3.3.Modèle linéaire

Un tel modèle caractérise un processus susceptible d'être décrit par une équation différentielle ordinaire ou une équation récurrente à coefficients constants (stationnaire) ou non constants (non stationnaire) ou par un ensemble d'équation de ce type.

La propriété fondamentale des systèmes linéaires s'exprime par le principe de **s**uperposition : si $y_1(t)$ et $y_2(t)$ représentent respectivement l'évolution des sorties du processus pour les entrées $u_1(t)$ et $u_2(t)$ sur un horizon et pour un état initial donné, alors, à l'entrée $u(t)$:

$$u(t) = \alpha_1 u_1(t) + \alpha_2 u_2(t)\,; \quad (5)$$

Où α_1 et α_2 sont des constantes réelles, correspond la sortie :

$$y(t) = \alpha_1 y_1(t) + \alpha_2 y_2(t)\,. \quad (6)$$

Sur le même horizon et pour les mêmes conditions initiales.

3.4.Modèle non linéaire

Dans un modèle non linéaire, le théorème de superposition n'est plus valable. L'aspect non linéaire peut être intrinsèque et quasiment irréductible, comme dans la modélisation de la loi d'action de masse en chimie, ou correspondre des éléments à caractéristiques non linéaires à des systèmes linéaires comme dans le cas des systèmes à non linéarités séparables (figure .2).

Dans certains cas, lorsque le processus non linéaire est utilisé dans une plage de variation limitée de ses variables d'état, alors l'évolution sera effectuée autour d'une valeur x_0 donnée, il est donc possible d'effectuer une linéarisation autour de ce point de fonctionnement en prenant comme nouvelle variable l'expression $\Delta x = x - x_0$.

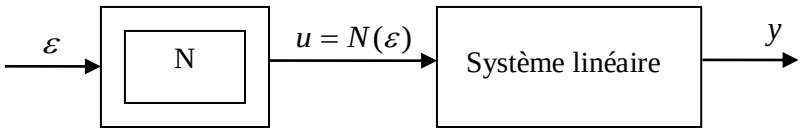


Figure. 2 : Système à non linéarité séparable.

Dans ce cas linéaire, on néglige alors dans le développement limité de la fonction $f(\cdot)$ les termes d'ordre supérieurs à 1.

Il vient par exemple pour la relation :

$$\dot{x} = f(x,u,t), \quad (7)$$

si la fonction f admet des dérivées partielles continues au premier ordre, le modèle linéarisé (8) sera valable pour (x,u) évoluant dans un voisinage de (x_0,u_0) :

$$\begin{aligned} f(x,u) &= f(x_0,u_0) + (x-x_0) f_x'(x_0,u_0) + (u-u_0) f_u'(x_0,u_0) \\ &= f(x_0,u_0) + \Delta x F_x + \Delta u F_u \end{aligned} \quad (8)$$

avec : $\Delta x = x - x_0$, $\Delta u = u - u_0$, et :

$$F_x = \left\{ \frac{\partial f}{\partial x} \right\},\ F_u = \left\{ \frac{\partial f}{\partial u} \right\} \quad (9)$$

Sont les matrices jacobéennes de f par rapport à x et à u.

Pour certains processus non linéaires et/ou non stationnaires, on adopte aussi parfois une représentation multi modèle, chaque modèle étant représentatif de l'évolution du processus dans un domaine limite de l'espace et du temps.

Ainsi un système linéaire non stationnaire peut être représenté par un ensemble de modèle linéaires à coefficients constants se succédant au cours du temps.

4. Identification des processus

L'identification est la détermination, sur la base de la connaissance d'un nombre fini d'entrées-sorties du système, d'un modèle appelé modèle d'identification, qui soumit aux mêmes entrées que le système fournit des sorties suffisamment proches de celui-ci. L'identification consiste à déterminer un ensemble d'équations –un modèle- décrivant le mieux possible le procédé. Il y a deux étapes d'identification, la première consiste à fixer la forme des équations, c'est l'étape qualitative, ou caractérisation, la seconde consiste à Trouver les valeurs numériques des coefficients qui interviennent dans ces équations, c'est l'étape quantitative, ou estimation des paramètres. Ces valeurs numériques sont déterminées pour que le comportement de modèle soit le plus proche de celui du système [1].

4.1.Les structures d'identification

La sortie d'un système dynamique dépend de son entrée et de son état ultérieur, c'est pour cela qu'il existe deux classes de modèle d'identification.

4.1.1. Identification parallèle

Dans le cas d'un système dynamique à temps discret, la sortie du modèle est calculée à partir de ses entrées et ses sorties passées :

$$\hat{y}(k+1) = f[\hat{y}(k), \hat{y}(k-1), \ldots, \hat{y}(k-n+1)\,; u(k), u(k-1), \ldots, u(k-m+1)] \quad (10)$$

Certaines précautions doivent être prises lors de l'identification, la principale étant d'utiliser des entrées bornées et sorties bornées, le système reste stable, il est dit BIBO (*Bonded Input Bonded Output*).

L'inconvénient de l'identification parallèle est que même avec un système borné, rien ne garantit que les paramètres vont converger et que l'erreur tend vers zéro[1].

La structure d'un identificateur parallèle est donnée par la figure suivante :

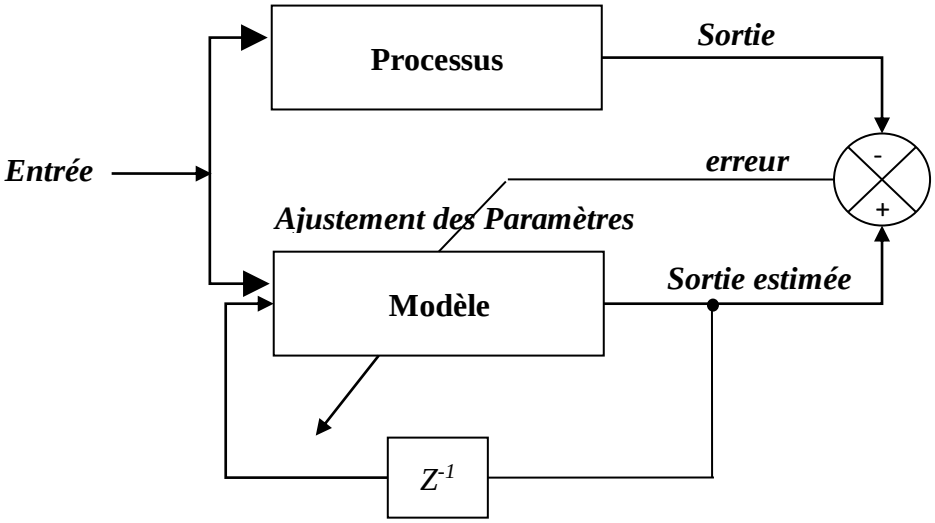


Figure. 3 : Structure d'identification parallèle.

4.1.2. Identification série-parallèle

La sortie du modèle est calculée à partir de ses entrées et la sortie du système à identifier :

$$\hat{y}(k+1) = f[y(k), y(k-1), \ldots, y(k-n+1); u(k), u(k-1), \ldots, u(k-m+1)] \quad (11)$$

Ce modèle a plus de chances de converger, car tous les signaux utilisés lors de l'identificationsont bornés[1].La structure d'un identificateur série- parallèle est donnée par la figure suivante :

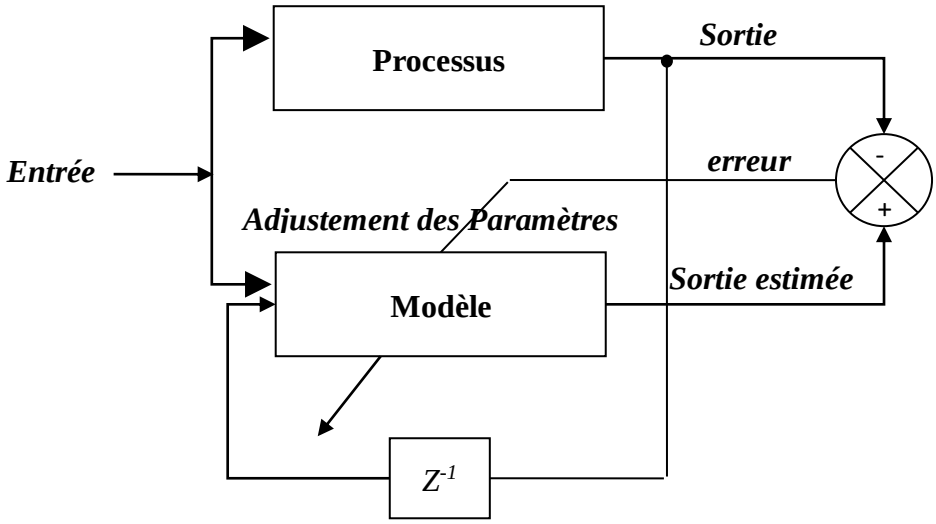


Figure. 4 : Structure d'identification série-parallèle.

5. Conclusion

Ce chapitre a été consacré aux notions de base relatives au processus et la modélisation, ou nous avons introduits quelques définitions, l'objectif de la modélisation, types des modèles et comment choisir un modèle. Le chapitre suivant sera consacré à la présentation des notions de base relatives aux algorithmes d'optimisation.

CHAPITRE II

RESEAUX DE NEURONES ARTIFICIELS

Les réseaux de neurones artificiels (Artificial neural networks en anglais) constituent une branche spécifique de la recherche en informatique et en neuro-informatique. Il existe différents types de réseaux de neurones artificiels, offrant différentes possibilités de traitement de l'information. Dans ce chapitre on va essayer de donner les notions essentielles pour comprendre les architectures, le fonctionnement et les applications des réseaux de neurones artificiels.

Sommaire

1. Introduction

Le cerveau humain a une fantastique puissance de traitement de l'information si l'on considère ses capacités à prendre en charge certaines taches nécessaires pour obtenir un comportement intelligent. La nature de l'intelligence a longtemps été un sujet difficile et controverse. D'une certaine manière, chacun a une idée assez vague de ce qu'est l'intelligence : la capacité à observer, à comprendre, à se souvenir, à résoudre des problèmes, à apprendre, à créer,…etc. Deux disciplines sont concernées par la définition et la modélisation du comportement intelligent : ce sont la psychologie cognitive et l'intelligence artificielle.

Les techniques de l'intelligence artificielle telles que les systèmes experts, la logique floue, les algorithmes génétiques et les réseaux de neurones artificiels (RNA) ont été largement utilisées dans le domaine de l'électronique.

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau.

2. Historique

Les recherches sur les méthodes neuronales de traitement de l'information en vue de modéliser le comportement du cerveau humain ne sont pas récentes, en **:**

- **1890 :** W. James, célèbre psychologue américain introduit le concept de mémoire associative et propose ce qui deviendra une loi de fonctionnement pour l'apprentissage sur les réseaux de neurones connus plus tard sous le nom de loi de Hebb.
- **1943 :** J. Mc Culloch et W. Pitts laissent leurs noms à une modélisation du neurone biologique (un neurone au comportement binaire). Ce sont les premiers à montrer que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes (tout au moins au niveau théorique).
- **1949 :** D. Hebb, physiologiste américain explique le conditionnement chez l'animal par les propriétés des neurones eux-mêmes. Ainsi, un conditionnement du type pavlovien tel que, nourrir tous les jours à la même heure un chien, entraînechez cet

animal la sécrétion de salive à cette heure précise même en l'absence de nourriture. La loi de modification des propriétés des connexions entre neurones qu'il propose explique en partie ce type de résultats expérimentaux **[7].** Le premier réseau de neurones artificiels apparaît en 1958, grâce aux travaux de Rosenblatt qui conçoit le perceptron. Ce dernier est inspiré du système visuel (en terme d'architecture neurobiologique) et possède une couche de neurones d'entrée une couche de sortie (décisionnelle). Ce réseau parvient à apprendre, à identifier des formes simples et à calculer certaines fonctions logiques **[8].**

- **1969 :** M. Minsky et S. Papert publient un ouvrage qui met en exergue les limitations théoriques du perceptron. Limitations alors connues, notamment concernant l'impossibilité de traiter par ce modèle des problèmes non linéaires. Ils étendent implicitement ces limitations à tout modèle de réseaux de neurones artificiels. Leur objectif est atteint, il y a abandon financier des recherches dans le domaine (surtout aux U.S.A.), les chercheurs se tournent principalement vers l'***IA*** (Intelligence Artificielle) et les systèmes à bases de règles **[7].**

Il faudra attendre le début des années 80 pour que l'intérêt pour ce domaine soit de nouveau présent **[7].** En effet, Hopfield démontre en1982 tout l'intérêt d'utiliser les réseaux récurrents (dits "feed-back») pour la modélisation des processus. Les réseaux récurrents constituent alors la deuxième grande classe de réseaux de neurones, avec les réseaux type perceptron (dit "feed-forward").

En parallèle des travaux de Hopfield, Werbos conçoit son algorithme de rétro- propagation qui ne sera pourtant popularisé qu'en 1986 par Rumelhart.

3. Domaines d'application des réseaux de neurones artificiels

Aujourd'hui, les réseaux de neurones artificiels ont de nombreuses applications dans des secteurs très variés **:**

- ✓ ***Traitement d'images*****:** reconnaissance de caractères et de signatures, compression d'images, reconnaissance de formes, cryptage, classification, etc.
- ✓ ***Traitement du signal*****:** filtrage, classification, identification de source, traitement de la parole etc.
- ✓ ***Contrôle*****:** commande de processus, diagnostic, contrôle de qualité, asservissement des robots, systèmes de guidage automatiques des automobiles et des avions etc.

- ✓ ***Défense*:** guidage des missiles, suivi de cible, reconnaissance du visage, radar, sonar, compression de données, suppression du bruit etc.
- ✓ ***Optimisation* :** planification, allocation de ressources, gestion et finances etc.
- ✓ ***Simulation*:** simulation du vol, simulation de boîte noire, prévision météorologique, recopie de modèle ….. etc.**[9].**

4. Présentation des Réseaux de Neurones

4.1.Définition

Un réseau de neurones peut être considéré comme un modèle mathématique de traitement réparti, composé de plusieurs éléments de calcul non linéaire (neurones), opérant en parallèle et connectés entre eux par des poids.

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit.

Les neurones artificiels sont souvent utilisés sous forme de réseaux qui diffèrent selon le type de connexions entre les neurones, une cinquantaine de types peuvent être dénombrée. En guise d'exemples nous citons **:** le perceptron de rosemblat, les réseaux de Hopfield, etc**.** Ces derniers sont les plus utilisés dans le domaine de la modélisation et de la commande des procédés. Ils sont constitués d'un nombre fini de neurones qui sont arrangés sous forme de couches. Les neurones de deux couches adjacentes sont interconnectés par des poids. L'information dans le réseau se propage d'une couche à l'autre, on dit qu'ils sont de type « feed-forward ». Nous distinguons trois types de couches **:**

- ***Couche d'entrée*:** les neurones de cette couche reçoivent les valeurs d'entrée du réseau et les transmettent aux neurones cachés. Chaque neurone reçoit une valeur, il ne fait pas donc de sommation.
- ***Couches cachées* :** chaque neurone de cette couche reçoit l'information de plusieurs couches précédentes, effectuent la sommation pondérée par les poids, puis la transforme selon sa fonction d'activation qui est en général une fonction sigmoïde. Par la suite, il envoie cette réponse aux neurones de la couche suivante.

- ***Couche de sortie*:** elle joue le même rôle que les couches cachées, la seule différence entre ces deux types de couches est que la sortie des neurones de la couche de sortie n'est liée à aucun autre neurone **[10].**

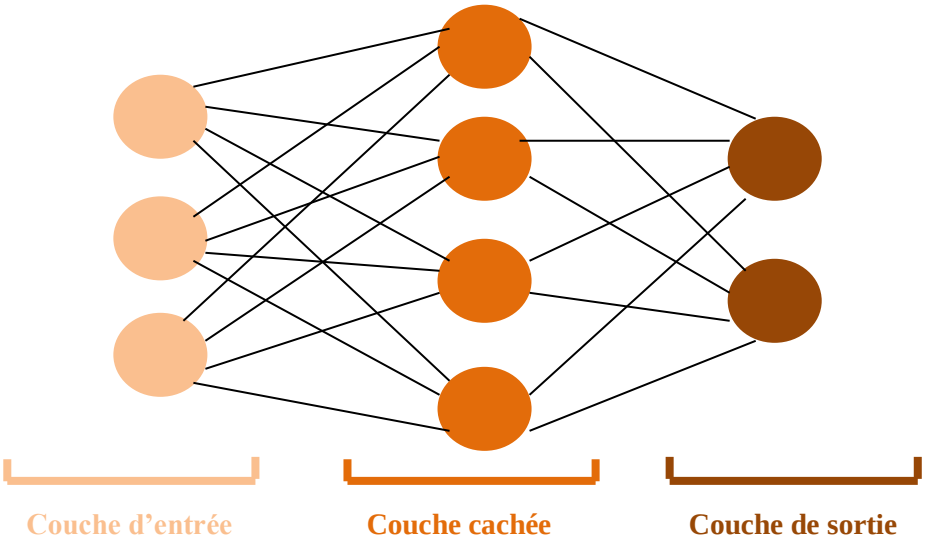


Figure II.1 : Diagramme du Réseau de neurones.

4.2.Neurone biologique

Le cerveau humain possède deux hémisphères latéraux reliés par le corps calleux et d'autres ponts axonaux, il pèse moins de deux kilogrammes et contient mille milliards de cellules, dont100 milliards sont des neurones constitués en réseaux.

Le neurone biologique (**figure. 2**) est une cellule vivante spécialisée dans le traitement des signaux électriques. Les neurones sont reliés entre eux par des liaisons appelées axones. Ces axones vont eux-mêmes jouer un rôle important dans le comportement logique de l'ensemble.

Ils conduisent les signaux électriques de la sortie d'un neurone vers l'entrée (synapse) d'un autre neurone. Les neurones font une sommation des signaux reçus en entrée et en fonction du résultat obtenu vont fournir un courant en sortie**[12].**

Les neurones sont des cellules nerveuses décomposables en 4 parties principales(**figure.2**).

Les dendrites, sur lesquelles les autres cellules entrent en contact synaptique : c'est par les dendrites que se fait la réception des signaux. Le corps de la cellule ou noyau, c'est l'unité de traitement. L'axone est la partie où passent les messages accumulés dans le corps de la cellule. Enfin, à la sortie du neurone on trouve les synapses, par lesquelles la cellule

communique avec d'autres neurones, ce sont des points de connexion par où passent les signaux de la cellule.

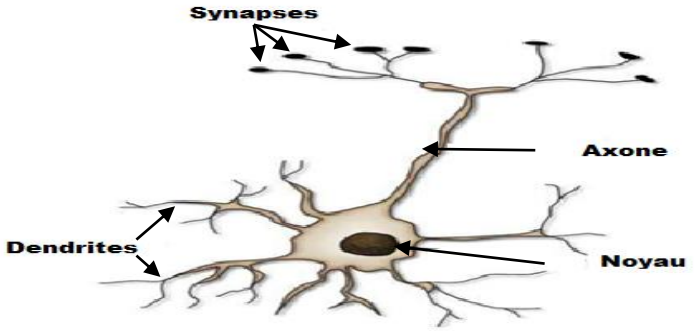


Figure. 2 : Neurone biologique **[13].**

Un neurone stimulé envoie des impulsions électriques ou potentielles d'action, à d'autres neurones. Ces impulsions se propagent le long de l'axone unique de la cellule. Au point de contact entre neurones, les synapses, ces impulsions sont converties en signaux chimiques. Quand l'accumulation des excitations atteint un certain seuil d'activation, le neurone engendre un potentiel d'action, pendant une durée de 1ms. Le neurone émettant le signal est appelé le neurone pré-synaptique et celui recevant ce signal, neurone post-synaptique.

4.3.Neurone formel

Un RNA est un assemblage d'éléments de structure identique appelés cellules (ou neurones) interconnectées à l'instar des cellules du système nerveux des vertébrés. Chaque point de connexion (appelé coefficient ou poids) entre deux cellules joue le rôle d'une synapse, l'élément principal d'interaction entre les neurones. Ces connexions ou poids synaptiques ont un rôle primordial dans le fonctionnement parallèle et adaptatif des neurones d'où la notion de réseaux connexionnistes. La représentation mathématique du neurone est introduite par McCulloch et Pitts (1943). Le neurone formel est considéré comme étant la pierre angulaire de la structure des RNA et comprennent cinq éléments importants **:**

- ***Les entrées*** **:** les valeurs, considérées comme entrées du réseau, sont généralement normalisées.

- ***Les poids et le biais* :** Les poids W_{k0}, $W_{k1} \dots W_{km}$ sont des valeurs reliant les neurones de deux couches différentes ou ceux de la même couche (selon le mode de connectivité). Le biais est représenté par x_0.

- ***La fonction de sommation (combinaison):***
  $Y = \sum_{i=0}^{m} x_i w_{ki} + x_0$.(1)
- ***La fonction de transfert (d'activation***
- ***La sortie* :** le résultat obtenu *Y***[11].**

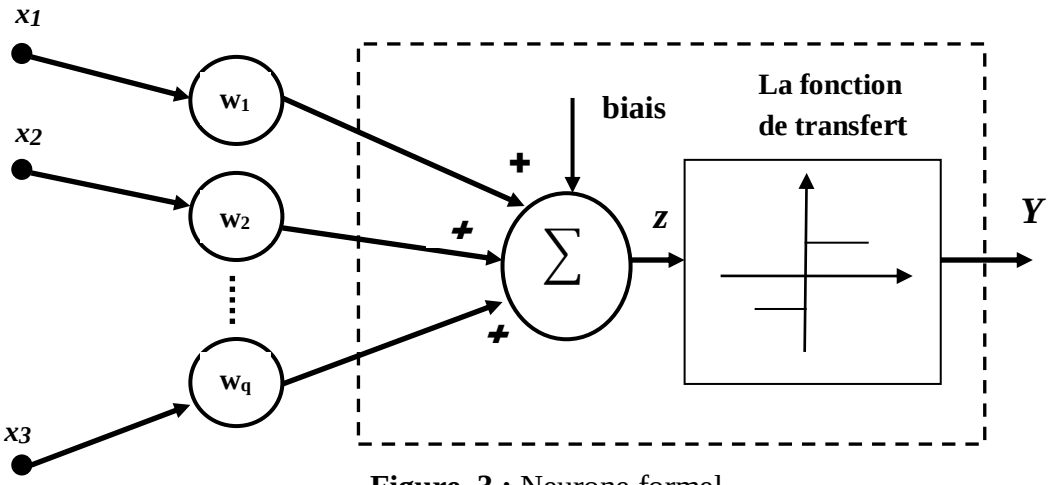


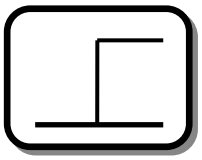
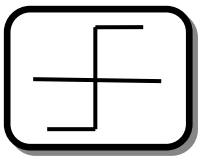
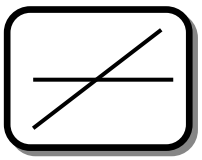
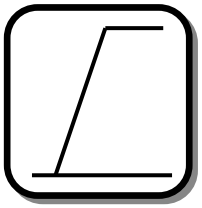
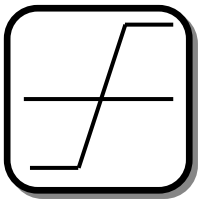
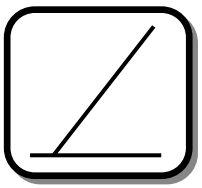
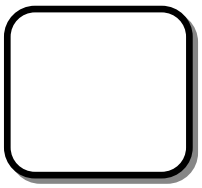
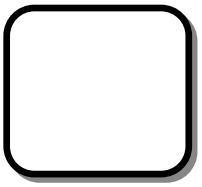
Figure. 3 : Neurone formel.

4.4.Fonction d'activation

La fonction d'activation (ou fonction de seuillage, ou encore fonction de transfert) sert à introduire une non-linéarité dans le fonctionnement du neurone. Les fonctions de seuillage présentent généralement trois intervalles **:**

- ✓ En dessous du seuil, le neurone est non actif (souvent dans ce cas, sa sortie vaut 0 ou -1).
- ✓ Aux alentours du seuil, une phase de transition.
- ✓ Au-dessus du seuil, le neurone est actif (souvent dans ce cas, sa sortie vaut 1) dans sa première version, le neurone formel était implémenté avec une fonction à seuil, mais de nombreuses versions existent**[14].** Ainsi, le neurone de McCulloch et Pitts a été généralisé de différentes manières, en choisissant d'autres fonctions d'activations, comme les fonctions énumérées dans le **(tableau .1).**

Tableau. 1 : Différentes fonctions d'activations utilisées dans les RNA **[9].**

Nom de la fonction	*Relation entrée/sortie*	*graphe*	*Nom MATALB*
Seuil	*y=0 si s <0* *y =1 si s ≥0*		Hardlim
Seuil symétrique	*y=–1 si s <0* *y =1 si s ≥0*		Hardlims
Linéaire	*y=s*		Purelin
Linéaire saturée	*y=0 si s ≤0* *y =s si 0 ≤s ≤1* *y =1 si s ≥1*		Satlin
Linéaire saturée symétrique	*y=–1 si s –<1* *y =s si –1 ≤s ≤1* *y =1 si s >1*		Satlins
Linéaire positive	*y=0 si s ≤0* *y =s si s ≥0*		Poslin
Sigmoïde	$y = \frac{1}{1+exp^{-s}}$		Logsig
Tangente hyperbolique	$y = \frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$		Tansig

Compétitive	*y=1 si s maximum* *y =0 autrement*	**C**	Compet

5. Architecture des réseaux de neurones

L'architecture d'un réseau de neurones est l'organisation des neurones entre eux au sein d'un même réseau. Autrement dit, il s'agit de la façon dont ils ordonnaient et connectés. La majorité des réseaux de neurones utilise le même type de neurones. Quelques architectures plus rares se basent sur des neurones dédiés. L'architecture d'un réseau de neurones dépend de la tâche à apprendre. Un réseau de neurones est en général composé de plusieurs couches de neurones, des entrées jusqu'aux sorties. On distingue deux grands types d'architectures de réseaux de neurones **:**

- Les réseaux de neurones non bouclés.
- Les réseaux de neurones bouclés **[9].**

5.1.Les réseaux de neurones non bouclés

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. Un réseau de neurones non bouclé est représenté graphiquement par un ensemble de neurones "connectés" entre eux, l'information circulante des entrées vers les sorties sans « retour en arrière »; si l'on représente le réseau comme un graphe dont les nœuds sont les neurones et les arêtes des « connexions » entre ceux-ci, le graphe d'un réseau non bouclé est acyclique. Le terme de "connexions" est une métaphore : dans la très grande majorité des applications, les réseaux de neurones sont des formules algébriques dont les valeurs numériques sont calculées par des programmes d'ordinateurs, non des objets physiques (circuits électroniques spécialisés) ; néanmoins, le terme de connexion, issu des origines biologiques des réseaux de neurones, est passé dans l'usage, car il est commode quoique trompeur. Il a même donné naissance au terme du connexionnisme **[9].**

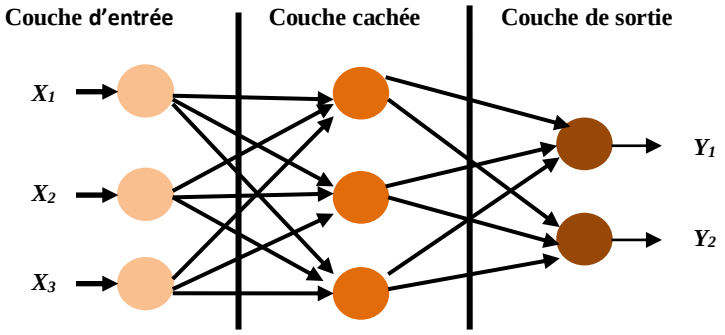


Figure.4 : Les réseaux de neurones non bouclés.

5.1.1. Réseaux de neurones monocouches

La structure d'un réseau monocouche est telle que des neurones organisés en entrée soient entièrement connectés à d'autres neurones organisés en sortie par une couche modifiable de poids **(figure. 5) [9].**

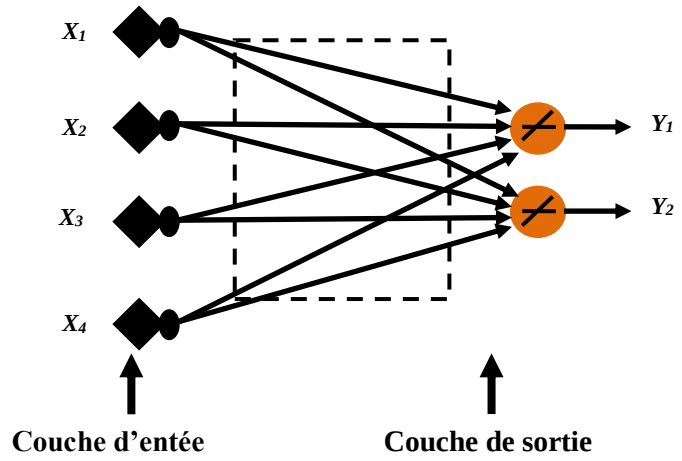


Figure. 5 : Schéma d'un réseau de neurones monocouche.

5.1.2. Réseaux de neurones multicouches

Les neurones sont arrangés par couches. Il n'y a pas de connexion entre neurones d'une même couche, et les connexions ne se font qu'avec les neurones de couches avalent. Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement. Ceci nous permet d'introduire la notion de sens de parcours de l'information (de l'activation) au sein d'un réseau et donc définir les concepts de neurones d'entrée, neurone de sortie. Par extension, on appelle couche d'entrée

l'ensemble des neurones d'entrée, couche de sortie l'ensemble des neurones de sortie. Les couches intermédiaires n'ayant aucun contact avec l'extérieur sont appelées couches cachées. La figure. 6 représente un réseau de neurones non bouclé qui a une structure particulière, très fréquemment utilisée **:** il comprend des entrées, deux couches de neurones cachés et des neurones de sortie. Les neurones de la couche cachée ne sont pas connectés entre eux. Cette structure est appelée Perceptron multicouche**.**

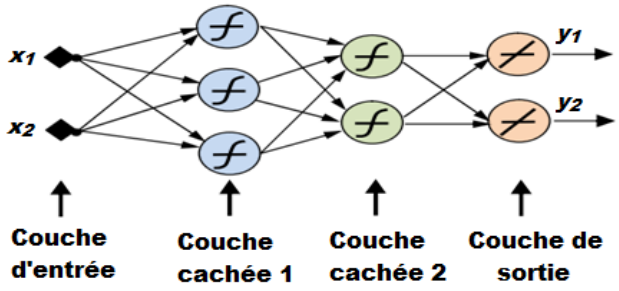


Figure. 6 : Schéma d'un réseau de neurones Perceptron multicouches.

On note aussi que les réseaux multicouches sont beaucoup plus puissants que les réseaux simples à une seule couche. En utilisant deux couches (une couche cachée et une couche de sortie), à condition d'employer une fonction d'activation sigmoïde sur la couche cachée, on peut entraîner un réseau à produire une approximation de la plupart des fonctions, avec une précision arbitraire (cela peut cependant requérir un grand nombre de neurones sur la couche cachée). Saufs dans des rares cas, les réseaux de neurones artificiels exploitent deux ou trois couches **[9].**

5.1.3. Réseaux de neurones à connexions locales

Il s'agit d'une structure multicouche, mais qui à l'image de la rétine conserve une certaine topologie. Chaque neurone entretient des relations avec un nombre réduit et localisé de neurones de la couche avalent. Les connexions sont donc moins nombreuses que dans le cas d'un réseau multicouche classique **(figure.7).**

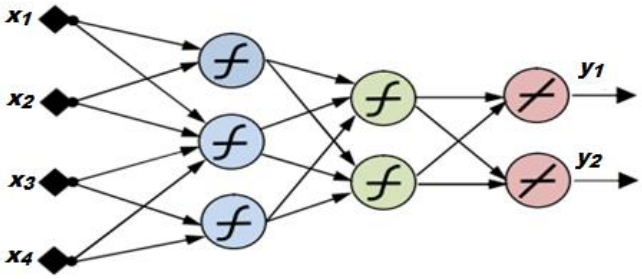


Figure. 7 : Schéma d'un réseau de neurones à connexions locales.

5.2.Les réseaux de neurones bouclés

Contrairement aux réseaux de neurones non bouclés dont le graphe de connexions est acyclique, les réseaux de neurones bouclés peuvent avoir une topologie de connexions quelconque, comprenant notamment des boucles qui ramènent aux entrées la valeur d'une ou plusieurs sorties. Pour qu'un tel système soit causal, il faut évidemment qu'à toute boucle soit associé un retard : 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. Il s'agit donc des réseaux de neurones avec retour en arrière (feedback network or récurrent network), **(Figure.8).**

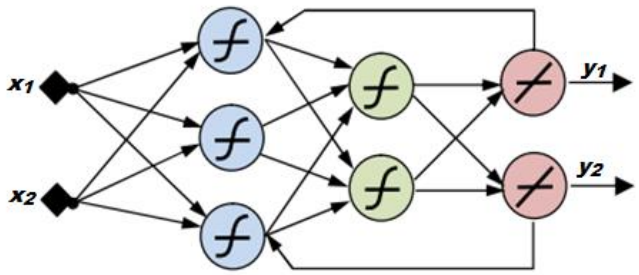


Figure. 8 : Schéma de réseau de neurones bouclé.

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 :**

$$x(k+1) = \varphi[x(k), u(k)] \quad (2)$$

$$y(k) = \psi[x(k), u(k)] \quad (3)$$

Où φ et ψ sont des fonctions non linéaires réalisées par un réseau de neurones non bouclé (mais pas obligatoirement, un perceptron multicouche), et ou*k* désigne le temps (discret). La forme canonique est représentée sur la figure 9**.** Tout réseau de neurones, aussi compliqué soit-il, peut être mis sous cette forme canonique, de manière complètement automatique **[9].**

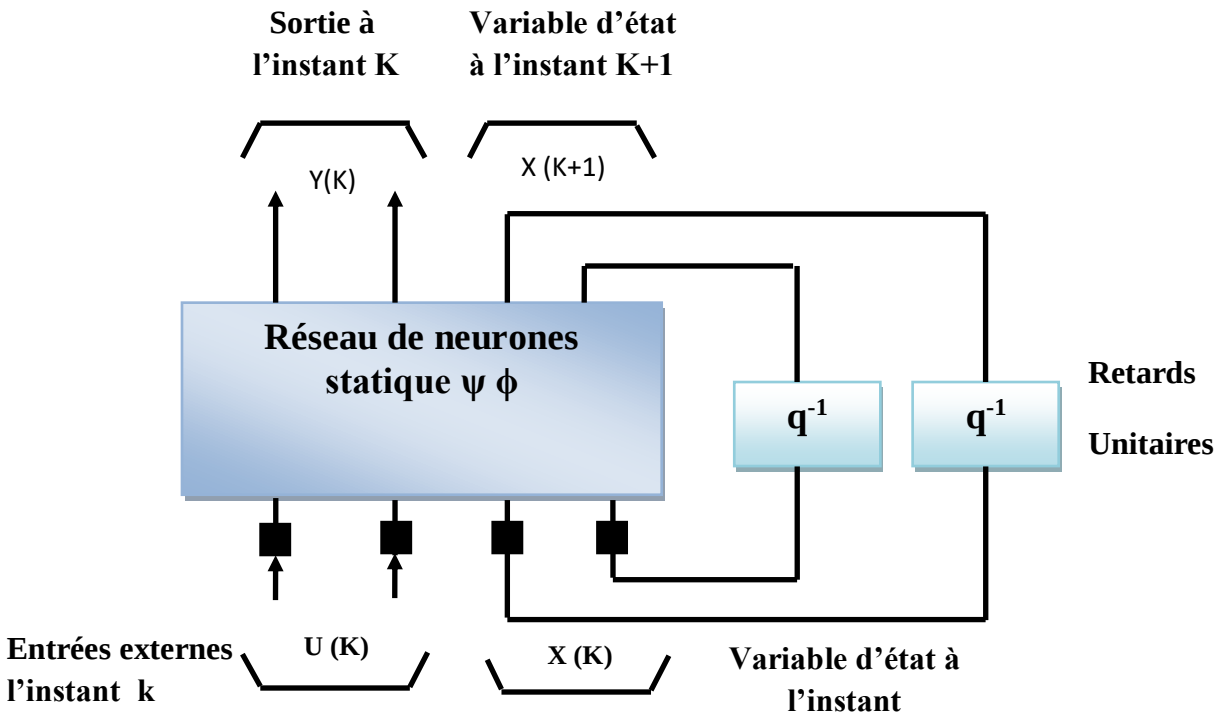


Figure. 9 : Forme canonique d'un réseau de neurones bouclé.

6. Apprentissage

Si le neurone est considéré comme étant la pierre angulaire du réseau de neurones artificiels, l'apprentissage peut être défini comme le processus le plus important pour la construction d'une structure performante du réseau de neurones en se déroulant sur l'adaptation des poids. Pour cela, il existe deux méthodes d'apprentissage **:** l'apprentissage supervisé et non supervisé. Les deux méthodes précédentes, fournissent au réseau neuronal une base d'apprentissage qui représente les données sur lesquelles le réseau peut apprendre et réaliser la phase d'apprentissage **[15].**

6.1.Apprentissage non supervisé

Que signifie le fait de former un réseau de neurones sans supervision ? Comme indiqué précédemment, le réseau de neurones est muni d'une base d'apprentissage, qui est une collection de valeurs d'entrée bien définies. Le réseau de neurones non supervisé n'est pas muni de données attendues ou sorties anticipées. L'apprentissage non supervisé est généralement utilisé pour entraîner les réseaux de neurones utilisés pour la classification et le data mining **[15].**

La figure suivante résume la méthode d'apprentissage non supervisé.

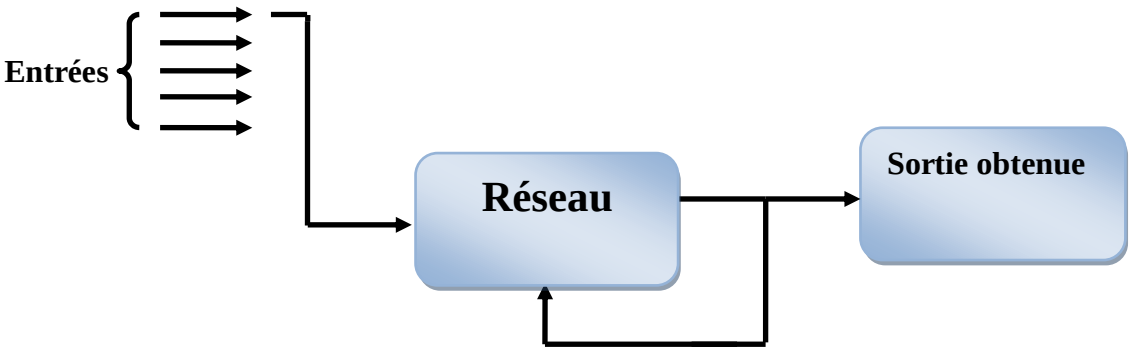


Figure. 10 : Apprentissage non supervisé.

6.2.Apprentissage supervisé

La méthode d'apprentissage supervisée est similaire à la méthode d'apprentissage non supervisé, dans laquelle les ensembles d'apprentissage sont fournis. Tout comme avec l'apprentissage non supervisé, les ensembles d'apprentissage spécifient les signaux d'entrée pour le réseau neuronal. La principale différence est que dans l'apprentissage supervisé, les résultats attendus sont fournis.

L'apprentissage supervisé est basé sur la comparaison directe entre l'entrée actuelle du RNA et la sortie désirée correcte, aussi connue comme la sortie cible. Il est souvent formulé comme la minimisation d'une fonction d'erreur comme l'erreur quadratique moyenne totale entre l'entrée actuelle et la sortie désirée additionnée à toutes les données disponibles. Une descente de gradient, basée sur l'algorithme d'optimisation telle que la rétro-propagation, peut alors être utilisée pour ajuster itérativement les connexions des poids dans le RNA afin de minimiser l'erreur **[15].**

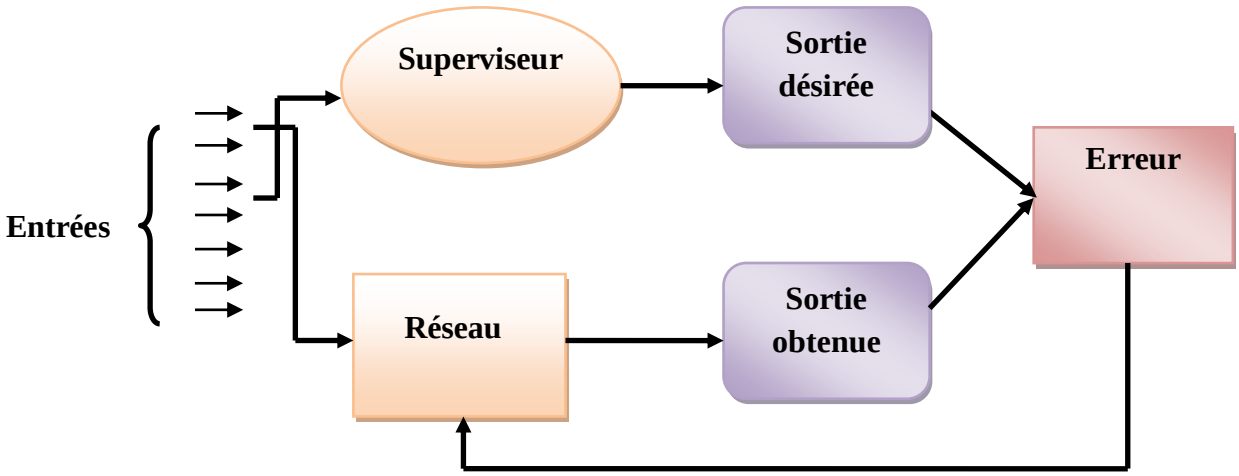


Figure.11 : Apprentissage supervisé.

6.3.Règles d'apprentissage

L'apprentissage est une procédure adaptative par laquelle les connexions (ou synapses) des neurones sont ajustés face à une source d'information. Il existe au moins 23 types de règles d'apprentissage qui peuvent être regroupés en trois catégories **:** les règles d'apprentissage supervisé, non supervisé, sont renforcées. Mais l'objectif fondamental de l'apprentissage reste le même : soit la classification, l'approximation de fonction ou encore la prévision. Dans tous les cas, la procédure adaptative de mise à jour des poids obéit essentiellement à un objectif : la recherche analytique du minimum de la fonction de coût (ou fonction d'erreur) dans un espace multidimensionnel. Il existe deux techniques de recherche du minimum de la fonction d'erreur **:** la première est une technique de recherche du minimal local basé sur la méthode du gradient décroissant ; la seconde est une recherche du minimum global fondée sur des approches statistiques ou génétiques **[15].**

7. Applications des réseaux de neurones artificiels

Les RNA sont considérés comme une nouvelle approche de traitement de l'information par apprentissage de cette information et la rend disponible à l'utilisation afin de résoudre un tel problème. Il existe de nombreux problèmes qui peuvent être résolus par un réseau de neurones. Toutefois, les réseaux de neurones sont généralement utilisés pour traiter des types particuliers de problèmes. Les quatre types de problèmes qui sont souvent résolus par les réseaux de neurones sont les suivants **:**

7.1.Classification

La classification est le processus de classement des entrées en groupes. Par exemple, une compagnie d'assurance peut vouloir classer les demandes d'assurance dans les différentes catégories de risques, ou une organisation en ligne peut vouloir de son système de messagerie de classer le courrier entrant dans des groupes de messages spam et non-spam**[15].**

7.2.Prédiction

La prédiction est une autre application des réseaux neuraux. Étant donné une série temporelle de données d'entrée, un réseau de neurones peut prédire les valeurs futures. La précision de la prédiction dépend de nombreux facteurs, tels que la quantité et la pertinence des données d'entrée. Par exemple, les réseaux de neurones sont généralement appliqués à des problèmes de prédiction de l'évolution des marchés financiers **[15].**

7.3.Reconnaissance des formes (pattern recognition)

Reconnaissance des formes est l'une des utilisations les plus courantes des réseaux neuraux. Pattern recognition est une forme de classification et est tout simplement la capacité de reconnaître un motif. Le modèle doit être reconnu même s'il n'est pas clair. Exemple : la reconnaissance des visages **[15].**

7.4.Optimisation

Une autre application des réseaux de neurones est l'optimisation qui peut être appliquée à de nombreux problèmes pour lesquels une solution est recherchée. Le réseau de neurones peut ne pas toujours trouver la solution optimale mais il cherche à trouver une solution acceptable.

Les RNA sont utilisés dans le domaine des sciences cognitives où l'on cherche à développer des modèles capables de manifester des capacités d'apprentissage et d'adaptation à leur environnement (Blayo et Verleysen 1996). Les deux concepts de base à l'origine des différents types de modèles neuronaux sont l'architecture et l'apprentissage **[15].**

8. Conclusion

Les réseaux de neurones sont considérés comme des systèmes imitant le fonctionnement du cerveau humain. Dans ce chapitre, nous avons vu la différence entre le neurone biologique et neurone formel ainsi que les différentes fonctions d'activation. Comme les RNA se caractérisent selon le domaine d'utilisation et des problèmes qu'ils peuvent résoudre. Ils se caractérisent aussi par leurs différentes architectures, traitées dans ce chapitre**.**

CHAPITRE III

ALGORITHMES D'OPTIMISATION

Dans ce mémoire de fin d'étude on va utiliser les réseaux de neurones artificiels et l'algorithme d'optimisation à base de biogéographie pour la modélisation des systèmes dynamiques. Dans le deuxième chapitre on a exposé les notions liées aux réseaux de neurones artificiels. Nous exposons de ce troisième chapitre les notions liées aux algorithmes optimisations.

Sommaire

1. Introduction

Un algorithme d'optimisation c'est une technique visant à résoudre des problèmes d'optimisation difficile (souvent issus des domaines de la recherche opérationnelle, de l'ingénierie ou de l'intelligence artificielle). Ces techniques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est-à dire l'extremum global d'une fonction. Elles se comportent comme des algorithmes de recherche tentant d'apprendre les caractéristiques d'un problème a fin d'en trouver une approximation de la meilleure solution. Il existe un grand nombre des algorithmes d'optimisations, allant de la simple recherche locale à des algorithmes complexes de recherche globale. Ces techniques sont souvent inspirées par des mécanismes biologiques ou biogéographiques. L'analogie entre un problème d'optimisation et ces phénomènes biologique set biogéographiques a été formalisée par plusieurs approches, à commencer par les algorithmes génétiques.

2. Problème d'optimisation

L'optimisation est un aspect fondamental de l'ingénierie et de la résolution de problèmes. L'objectif de l'optimisation est de chercher les valeurs d'un ensemble de paramètres pour maximiser ou minimiser les fonctions objectifs soumises à certaines contraintes. Un choix de valeurs, pour l'ensemble des paramètres, qui satisfont toutes les contraintes, est appelé une solution faisable. Les solutions faisables avec des valeurs de la fonction objectif qui sont meilleures que les valeurs de toutes les autres solutions possibles, sont appelées les solutions optimales [16].

Un problème d'optimisation peut être formulé sous la forme d'un problème de minimisation ou d'un problème de maximisation. Parfois, nous essayons de minimiser une fonction et parfois, nous essayons de maximiser une fonction. Ces deux problèmes sont facilement convertis l'un à l'autre [17] :

$$min_x f(x) \leftrightarrow max_x[-f(x)] \quad (1)$$

$$max_x f(x) \leftrightarrow min_x[-f(x)] \quad (2)$$

La fonction *f(x)* est appelée la fonction objectif, et le vecteur *x* est appelé la variable indépendante, ou la variable de décision. Notons, qu'en fonction de contexte, les termes« variable indépendante » et « variable de décision » se réfèrent parfois à l'ensemble du

vecteur *x*, et se réfèrent parfois à des éléments spécifiques dans le vecteur *x*. Les éléments de *x* sont aussi appelés les attributs de la solution. Le nombre des éléments de *x* est appelé la dimension du problème. Comme la montre les Équations (1) et (2), tout algorithme conçu pour minimiser une fonction peut facilement être utilisé pour maximiser une fonction, et tout algorithme conçu pour maximiser une fonction peut être facilement utilisé pour minimiser une fonction :

- ✓ Quand nous essayons de minimiser une fonction, nous appelons la valeur de la fonction : ***la fonction de coût***.
- ✓ Quand nous essayons de maximiser une fonction, nous appelons la valeur de la fonction : ***la fitness***.

3. Les types de problèmes d'optimisation

Il existe certains types particuliers de problèmes d'optimisation :

- ***Les problèmes généraux sans contraintes*** : où une fonction non linéaire est définie sur un ensemble de valeurs réelles sans contraintes.
- **Les problèmes généraux avec contraintes** : où une fonction non linéaire est définie sur un ensemble limité de valeurs réelles. Généralement, les problèmes d'optimisation sont des problèmes d'optimisation avec contraintes [17].
- ***Les problèmes d'optimisation multi-objectifs*** : dans lesquels un problème nécessite la résolution de plusieurs problèmes simultanément. Souvent, les solutions aux divers problèmes interfèrent entre elles, la meilleure solution est alors une sorte de compromis [18]. L'optimisation multi-objectif cherche à optimiser les composantes de valeurs de vecteur d'une fonction de coût. Contrairement à l'optimisation avec objectif unique, la solution à ce problème n'est pas un seul point, mais une famille de points connus comme l'ensemble Pareto-optimal. Chaque point de cette sur face est optimal dans le sens qu'aucune amélioration ne peut être obtenue en un composant de vecteur de coût. L'ensemble des solutions est appelé l'ensemble de Pareto [19].
- ***Les problèmes d'optimisation multimodale*** : ce sont ceux dans lesquels l'espace de recherche contient plusieurs optimums locaux et il est possible qu'il contient plus d'un optimum global. Ces problèmes sont intéressants, non seulement en raison du défi qu'ils représentent, en évitant les optimums locaux ou la localisation de plus d'un optimum global en même temps, mais parce

qu'il existe beaucoup de problèmes du monde réel présentant ces caractéristiques [20].

- ***Les problèmes combinatoires*** : il existe de nombreux problèmes d'optimisation pour lesquels les variables indépendantes sont limitées à un ensemble de valeurs discrètes. Ces problèmes sont appelés problèmes d'optimisation combinatoire [17].

4. Les méthodes d'optimisation classiques

Les méthodes d'optimisation classiques, qui sont connues par les méthodes évolutionnaires sont des méthodes ayant comme principe de faire évoluer une population de base (constituée de solutions au problème à résoudre) par des opérateurs de variations (croisement et mutation). Cette population est soumise à une sélection pour le croisement puis une sélection pour la survie. L'objectif est de faire converger les solutions (appelées « individus ») vers un optimum en respectant certains critères.

Dans les années 70, les premiers travaux sur l'évolution artificielle ont concerné les algorithmes génétiques (GA), les stratégies d'évolution (SE) et la programmation évolutive(PE). Ces trois types d'algorithmes ont utilisés des principes globalement communs care ils se sont tous inspirés des mêmes principes du néo-darwinisme : utilisation d'une population d'individus, évaluation des individus par une fonction, sélection des meilleurs et génération d'une nouvelle population avec des opérateurs de croisement et de mutation. Ensuite, dans les années 90 est apparue la programmation génétique (PG) qui introduit, notamment, des représentations arborescentes [21].

a. Les algorithmes génétiques

Les algorithmes génétiques (*GA : Genetic Algorithms*) sont des stratégies d'adaptation et des techniques d'optimisation globale[22]. Ce sont les premiers, les plus connus, et les plus utilisés parmi les méthodes évolutionnaires [17]. Les algorithmes génétiques ont été développés à l'origine dans les années 60 à l'université du Michigan par John Holland et son équipe [23], qui ont mené leur recherche sur des systèmes adaptatifs et robustes. Il sont été utilisés au début avec des représentations binaires, où les opérateurs de croisement et de mutation jouent un rôle majeur.

Les Algorithmes Génétiques forment une des principales classes des Algorithmes Evolutionnaires, proposés et développés par Hollande. Ils se basent sur les théories d'évolution naturelle modernes et utilisent une combinaison de reproduction : croisement et

mutation et de sélection, pour générer des individus de plus en plus adaptés à leur environnement, donc des solutions optimales**.**

b. La Programmation Evolutionnaire (PE)

Proposée par L.J. Fogel dans les années 60 également, et reprise par son fils D.B. Fogel dans les années 90, en Californie, USA. La Programmation Evolutionnaire est une des classes principales des Algorithmes Evolutionnaires. Elle se base sur le concept darwinien de l'évolution pour générer itérativement des solutions plus adaptées à leur environnement **[24].**

Bien que la Programmation Evolutionnaire ait été développée comme une méthode d'intelligence artificielle, son concept était loin des premières recherches en intelligence artificielle qui se basent sur la recherche des heuristiques simples. Au lieu de développer un ensemble complexe de lois de comportements dérivés de l'expertise humaine, la Programmation Evolutionnaire fait évoluer un ensemble d'individus faisant preuve d'un comportement optimal dans un environnement, statique ou dynamique, représenté par une fonction objective**.**

c. Les Stratégies d'Evolution (SE)

Elles ont été développées par I. Rechenberg et H.P. Schwefel, 1965 à Berlin. Les Stratégies d'Evolution constituent une autre classe d'Algorithmes Evolutionnaires. À l'inverse des autres approches, les Stratégies d'Evolution ont été conçues, à l'origine, comme des méthodes d'optimisation numériques. Leurs premières applications étaient l'optimisation d'un corps dans un tunnel de vent **[25].**

Abordant les problèmes d'optimisation continue, les Stratégies d'Evolution sont très similaires à la Programmation Evolutionnaire. Ils partagent la même philosophie : le problème d'optimisation est considéré dans sa globalité et aucun partitionnement n'est effectué, ils n'obéissent pas à la théorie des schèmes. Les deux méthodes mettent l'accent sur la mutation comme principal opérateur de recherche et utilisent une mutation auto-adaptative**.**

La seule différence notable avec la Programmation Evolutionnaire est l'utilisation du croisement par les Stratégies d'Evolution. Le croisement est utilisé comme second opérateur de recherche, son rôle n'était pas la manipulation des blocs élémentaires mais d'assurer plus de diversité dans la population.

d. La Programmation Génétique (PG)

Proposée par J. Koza en 1988 qui était auparavant un sous-groupe des *GA*. La spécificité de la PG est son espace de recherche, un espace de programmes le plus souvent représenté sous forme d'arbres. Cette technique cherche à atteindre un des vieux rêves des programmeurs **:** « ***écrire le programme qui écrit le programme*** »**[26].**

5. L'intelligence en essaims ou « *Swarm Intelligence* »

Certains chercheurs séparent l'intelligence en essaims de l'informatique évolutionnaire. Un algorithme d'intelligence en essaims est fondé sur le comportement des essaims qui existent dans la nature (par exemple, des essaims d'insectes ou d'oiseaux). L'optimisation par colonies de fourmis et par les essaims de particules sont les premiers algorithmes de l'intelligence en essaims, et certains chercheurs insistent sur le fait qu'ils ne doivent pas être considérés comme des algorithmes évolutionnaires. Cependant, d'autres auteurs considèrent l'intelligence en essaims comme étant un sous-ensemble de l'informatique évolutionnaire. Par exemple, l'un des pionniers de l'optimisation par les essaims de particules s'y réfère comme un algorithme évolutionnaire [27]. Plusieurs algorithmes ont été développés dans le cadre de l'intelligence en essaims tels que : les algorithmes de colonies de fourmis, le système de fourmis original ou Ant System, les essaims particulaires, les algorithmes inspirés des abeilles, l'algorithme des essaims de poissons artificiels, l'algorithme des chauves-souris, l'algorithme des essaims de lucioles et l'algorithme d'optimisation de l'exploration bactérienne….etc.

5.1. L'optimisation par essaim particulaire

L'optimisation par essaim particulaire (*PSO : Particle Swarm Optimisation*) a été introduite par Russel Elberhart et James Kennedy en 1995 [28]. Elle s'inspire des déplacements collectifs observés chez certains animaux sociaux tels que les poissons et les oiseaux migrateurs qui ont tendance à imiter les comportements réussis qu'ils observent dans leur entourage, tout en y apportant leurs variations personnelles. Elle trouve ses origines dans les travaux de Reynolds [29] et de Heppner et Granander [34] qui ont créé des modèles mathématiques permettant de simuler des vols groupés d'oiseaux et de bancs de poissons.

5.2. L'algorithme des essaims de luciole

L'algorithme d'optimisation basé sur les essaims de lucioles (insectes lumineux) (*FA : Firflay Algorithm*) a été développé par Xin-She Yang à l'université de Cambridge en 2007[31, 32, 33, 34]. FA a été inspiré du clignotement des lucioles. Chaque espèce de luciole a un motif unique de flashs. Bien que la gamme complète des fonctions flash n'a pas encore été déterminée, il est connu que le clignotant attire les compagnons. Dans certaines espèces de lucioles, les mâles sont attirés par des femelles sédentaires, dans d'autres espèces, la femelle copie le signal d'une espèce différente pour attirer les mâles de cette espèce. Le clignotement peut également être utilisé pour transmettre des informations entre les lucioles. L'algorithme FA s'inspire de l'idée de cette attraction et passage de l'information.

5.3.Optimisation par colonies de fourmis

L'optimisation par colonies de fourmis, (*ACO* : *Ant Colony Optimization*), conçue par Dorigo s'inspire comme son nom l'indique du comportement des fourmis lorsque celles-ci cherchent de la nourriture et optimisent le chemin entre leur nid et la nourriture trouvée. En effet, les fourmis utilisent leur environnement pour communiquer entre elles, il s'agit d'un mécanisme dît stigmergique grâce auquel elles déposent des phéromones sur le sol pour signifier aux autres fourmis le chemin qu'elles ont parcouru pour atteindre la nourriture. Ainsi, les autres pourront suivre la piste de phéromones pour retrouver la source de nourriture **[35].**

5.4.Optimisation par systèmes immunitaires artificiels

L'optimisation par systèmes immunitaires artificiels, (*AIS : artificial immune systems*), est née dans les années 1980 grâce au travail de Farmer, Packard et Perelson. L'AIS mime le fonctionnement du système immunitaire des êtres humains. En effet, ce dernier a pour but de protéger le corps d'agents pathogènes extérieur comme les bactéries ou les virus **[36].**

6. Autres méthodes évolutionnaires pour l'optimisation

Plusieurs techniques très récentes de l'informatique évolutionnaire ont été introduites ces dernières années mais n'appartiennent pas à l'intelligence en essaims tels que : l'optimisation basée sur la biogéographie, l'algorithme de recherche gravitationnelle et l'optimisation basée sur l'enseignement-apprentissage.

6.1.L'algorithme d'optimisation à base de biogéographie (BBO)

L'algorithme à base de biogéographie (*BBO : Biogeography-Based Optimization*), a été développé par Dan Simon en 2008 [37]. Inspiré par des études sur la répartition spatiale des espèces de plantes et d'animaux ainsi que les causes de leur répartition et de leur extinction. Elle traite de façon dont la richesse en espèces (nombres d'espèces) est maintenue dans un système d'île qui sont sujettes à l'immigration et sur lesquelles des espèces s'éteignent, quand une île ne peut pas facilement supporter la population d'une espèce, les membres migrent vers de nouvelles îles et subissent une spéciation. L'algorithme BBO manipule une population d'individus appelés îles (ou habitats). Chaque île représente une solution possible au problème à résoudre. La « fitness» de chaque île est déterminée par son HSI (Habitat Suitability Index), qui considère le caractéristique environnement a les telles que les précipitations, la température et la végétation qui constituent une mesure de la qualité d'une solution candidate. Ces caractéristiques peuvent être représentées quantitativement et sont appelées variables de l'indice d'aptitude (Suitability Index Variables ou SIV). Une bonne solution au problème d'optimisation est une île avec un grand nombre d'espèces, ce qui correspond à une île avec un faible HSI. Dans l'algorithme BBO, chaque habitat a ses propres taux d'immigration et d'émigration représentant les espèces qui viennent et sortent de l'île. Ces paramètres sont influencés par le nombre d'espèces(S) sur l'île.

6.1.1. Biogéographie

La Biogéographie est l'étude de la distribution de la biodiversité dans l'espace et dans le temps, ce qui permet aux nombreuses espèces animales de migrer vers différents habitats ou îles pour leur survie et une meilleure vie. La biogéographie a été étudiée dès le 19^{me} siècle par Alfred Wallaceet Charles Darwin.

Dans la science de la biogéographie, une île est définie comme la zone écologique habitée par des plantes particulière sou d'espèces animales et géographiquement isolée d'autres habitats. Chaque île a ses caractéristiques telles que la disponibilité alimentaire, les précipitations, la température, la diversité des espèces, la sécurité, etc. (Figure.). La qualité d'une île est mesurée par son indice d'adéquation (***Suitability Index***).Les îles avec un indice élevé sont plus adaptées à la vie et ont donc une grande population.

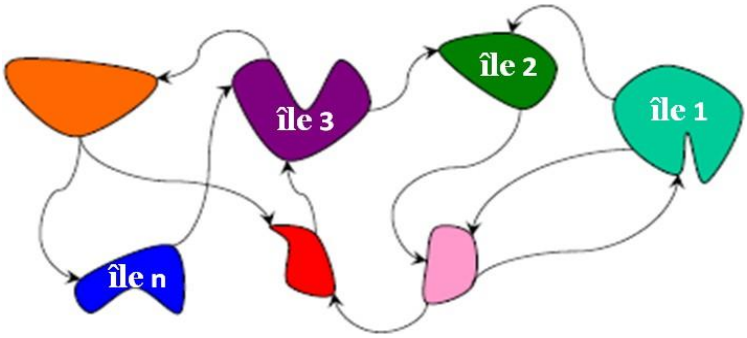


Figure .1 : Migration des espèces**.**

6.1.2. Principe de l'algorithme BBO

BBO, est un algorithme basé sur une population d'individus appelés île (ou habitats). Chaque île représente une solution possible au problème à résoudre. La « fitness » de chaque île est déterminée par son HSI (*Habitat Suitability Index*) qui est une mesure de la qualité d'une solution candidate, chaque île est représentée par des variables d'indice de qualité SIVs(*Suitability Index Variables*). Un HSI élevé d'une île signifie une bonne performance sur le problème d'optimisation, et un faible HSI signifie une mauvaise performance pour un problème d'optimisation**.**

Le fonctionnement de *BBO* est basé sur la migration et la mutation. La population initiale représente l'espace de recherche, elle est générée aléatoirement. L'évaluation de la population initiale engendre la migration de certains individus et les descendants vont être mutés. La migration permet de créer un nouvel ensemble d'individus et la mutation permet de fixer la proportion de la population qui sera renouvelée à chaque génération. Les meilleurs individus trouvés sont conservés par l'élitisme (sélection). Les nouveaux descendants remplacent les parents afin de former une nouvelle population**.**

6.1.3. Optimisation et biogéographie

L'application de la biogéographie dans l'optimisation est un exemple d'un modèle de processus naturel pour résoudre les problèmes d'optimisation. Ceci est similaire à ce qui s'est passé dans les dernières décennies avec les algorithmes génétiques, réseaux de neurones, les colonies de fourmis et d'autres domaines de l'intelligence informatique**.** L'optimisation à base de biogéographie (***BBO***) utilise un vocabulaire similaire à celui de la biogéographie (Tableau. 1) où chaque habitat est analogue à une solution du problème.

Les caractéristiques des solutions (variables de décision) sont appelées variables d'indice d'adéquation (SIV). L'indice d'adéquation de l'habitat (HSI) est analogue au fitness qui permet de mesurer l'adéquation de l'individu. Les habitats avec un HSI élevé ont tendance à avoir un grand nombre d'espèces, un taux d'immigration faible et un taux d'émigration élevé. Inversement, les habitats avec un HSI faible ont un faible nombre d'espèces, un taux d'immigration élevé et un taux d'émigration faible**.** L'algorithme BBO nécessite l'emploi de certains termes dont il est utile de préciser la définition.

Tableau. 1 : Terminologie de la BBO.

La Biogéographie	**Algorithme BBO**
Habitat ou île	Solution du problème
HSI	Qualité de la solution (fonctioncoût)
SIV	Les Variables du problème d'optimisation

- ✓ ***Variable d'indice d'adéquation***

Une variable d'indice d'adéquation ***(SIV: Suitability Index Variable)*** est une variable entière, réelle où booléenne qui caractérise l'habitabilité d'une île**.**

- ✓ ***Habitat***

Un habitat H est une solution du problème représentée généralement par un vecteur initialisé aléatoirement par des variables d'indice d'adéquation. Il est noté :

$$H = [SIV_1, SIV_2,...SIV_k] . \quad (3)$$

- ✓ ***Population*** **:**une population est un ensemble de *n* habitats**.**

$$Pop = [H_1, H_2,...H_n]. \quad (4)$$

- ✓ ***Indice d'adéquation de l'habitat***

Un indice d'adéquation de l'habitat ***(HSI)*** équivalant au fitness, associe une valeur pour chaque individu. Cette valeur a pour but d'évaluer le degré d'adaptation d'un individu à son environnement**.**

$HSI = f(H) = f([SIV_1, SIV_2,...SIV_k])$ (5)

- ✓ ***Taux d'immigration***

Le taux d'immigration $\boldsymbol{\lambda(Hi)}$ est le taux d'entrée des variables **(*SIV*)** à un habitat**.**

- ✓ ***Taux d'émigration***

Le taux d'émigration $\boldsymbol{\mu(H_i)}$ est le taux de sortie des variables **(*SIV*)** d'un habitat**.**

6.1.4. Étapes de l'algorithme d'optimisation basée sur la biogéographie

L'organigramme fonctionnel présenté dans la Figure.2, illustre l'organigramme générale de l'algorithme *BBO*. L'algorithme commence par initialiser les paramètres et la population initiale, il modifie cette population par des opérateurs spécifiques en construisant de nouvelle population jusqu'à atteindre une qualité (*HSI*) meilleure qu'un seuil préfixé ou un nombre maximal de générations g_{max}

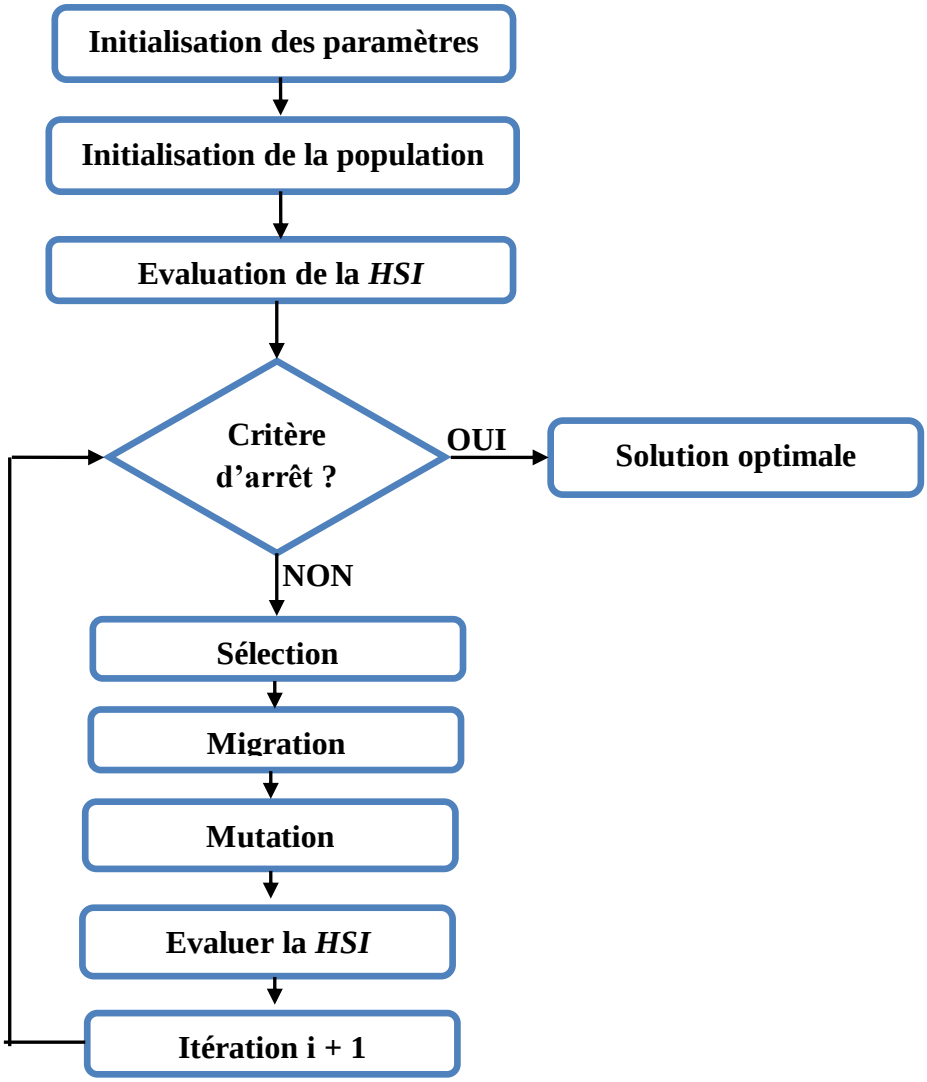


Figure. 2 : Organigramme de l'algorithme ***BBO***.

A. Initialisation

Dans cette étape, nous définissons les paramètres de contrôle (Tableau. 2) et nous générons aléatoirement une population initiale de *n* habitats. La taille de cette population reste constante tout au long de l'algorithme**.**

Tableau. 2 Paramètres de la ***BBO***.

Paramètres	***Notation***
Taille de la population	*Np*
Probabilité de mutation	*Pm*
Taille de la mémoire élite	n_{elit}

Taux d'immigration maximal	*I*
Taux d'émigration maximal	*E*
Nombre maximum de générations	*Gmax*

B. Évaluation de la fonction *HSI*

Pour évaluer la pertinence d'une solution par rapport à une autre, nous calculons la valeur de la ***HSI*** correspondante à chaque solution candidate**.**

C. Sélection

Pour appliquer les opérateurs de l'algorithme BBO comme n'importe quel autre algorithme d'optimisation, nous devons sélectionner les habitats ou individus candidats à cet opérateur. La sélection est fondée sur la qualité des habitats, est imée à l'aide de fonction d'adaptation. Il existe plusieurs méthodes de sélection**.**

- ***Sélection parroulette***

La population est représentée comme une roue de roulette, où chaque habitat est représenté par une portion qui correspond proportionnellement à sa valeur de ***HSI*** (fitness). La sélection d'un individu se fait en tournant la roue. L'un des inconvénients de ce type de sélection est de choisir presque toujours le même habitat s'il en existe un bien meilleur que tous les autres, ce qui cause une perte de diversité dans la population**.**

- ***Sélection parrang***

La sélection par rang trie d'abord la population par ***HSI***. Ensuite, chaque habitat se voit associé un rang en fonction de sa position. Ainsi le plus mauvais habitat aura le rang 1, le suivant 2, et ainsi de suite jusqu'au meilleur habitat qui aura le rang *n*. La sélection par rang d'un habitat est la même que par roulette, mais les proportions sont en relation avec le rang plutôt qu'avec la valeur de l'évaluation. Avec cette méthode de sélection, tous les habitats ont une chance d'être sélectionnés. Cependant, elle conduit à une convergence plus lente vers la bonne solution. Ceci est dû au fait que les meilleurs habitats ne diffèrent pas énormément des plus mauvais**.**

- ***Sélection par tournoi***

La sélection par tournoi est l'une des sélections les plus utilisées dans les algorithmes évolutionnaires. Le principe consiste à choisir un sous-ensemble d'individus (S individus)

aléatoirement dans la population, puis à sélectionner le meilleur individu dans ce groupe en fonction de son ***HSI***. Ce processus est répété jusqu'à l'obtention du nombre d'individus requis**.** Le nombre de participants à un tournoi, appelé la taille du tournoi, est utilisé pour faire varier la pression de cette sélection. Si ce nombre e s t grand, alors la pression sera forte et les faibles individus auront une petite chance d'être choisis. En général, un seul gagnant est choisi parmi les participants à un tournoi.

- ***Élitisme***

A la création d'une nouvelle population, il y a de grandes chances que les meilleurs habitats soient perdus après les opérations de migration et de mutation. Pour éviter cela, nous utilisons la méthode d'élitisme. Elle consiste à copier un ou plusieurs des meilleurs habitats dans la nouvelle génération. Ensuite, nous générons le reste de la population selon le mécanisme de reproduction usuel. L'opérateur de sélection est utilisé pour sélectionner des habitats pour appliquer les deux opérateurs d'évolution : la migration et la mutation**.**

D. Opérateur de migration

La migration est un opérateur probabiliste utilisé pour modifier chaque solution H_i en partageant des caractéristiques parmi les différentes solutions. L'idée de l'opérateur de migration est basée sur la migration en biogéographie, qui représente le mouvement des espèces entre les différents habitats. La probabilité qu'une solution est sélectionnée pour immigrer ou émigrer dépend de son taux d'immigration $\boldsymbol{\lambda_i}$ ou le taux d'émigration $\boldsymbol{\mu_j}$. Le processus de migration est déni par la relation :

$$H_i(SIV_k) \Leftarrow H_j(SIV_k). \qquad (6)$$

L'équation (6) représente comment une caractéristique ou ***SIV*** d'une solution H_i est remplacée par une caractéristique ou ***SIV*** d'une solution H_j par l'opération de migration. Dans la ***BBO***, chaque solution H_i a son propre taux d'immigration λ_i et taux d'émigration μ_i.

Ces deux taux sont calculés par l'équation (7) et l'équation (8) respectivement.

$$\lambda_i = I*(1-\frac{K_i}{n}) \qquad (7)$$

$$\mu_i = E*(\frac{K_i}{n}) \qquad (8)$$

I et *E* représentent les taux maximaux possibles d'immigration et d'émigration respectivement. k_i représente le rang du *i* habitat après le tri de tous les habitats en fonction de leur ***HSI***. *n* représente la taille de la population. La Figure. 4illustre deux solutions candidates S_1 et S_2 à un problème en utilisant des courbes d'immigration et d'émigration symétriques ($E = I$). S_1 représente une mauvaise solution et S_2 représente une meilleure solution. La probabilité d'immigration pour S_1 sera donc plus élevée que celle de S_2tandis que la probabilité d'émigration pour S_1 sera inférieure à la probabilité d'émigration pour S_2

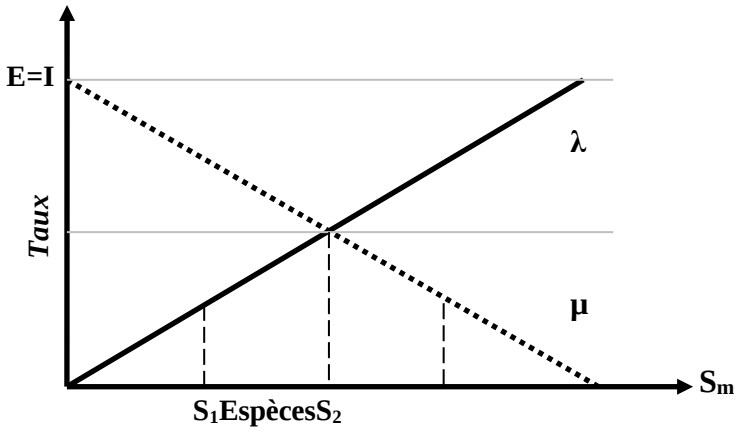


Figure .3 : Illustration de deux solutions candidates S1 et S2.

E. Opérateur de mutation

En plus de l'opérateur de migration, nous avons l'opérateur de mutation. La mutation est un opérateur probabiliste utilisé pour modifier un ou plusieurs ***SIV*** d'une solution sélectionnée aléatoirement en se basant sur sa probabilité d'existence P_{ig} pour la génération *g*. Cet opérateur aide à introduire de nouvelles caractéristiques et augmente la diversité dans la population. La probabilité de mutation mi est calculée en fonction de la probabilité de la solution exprimée dans l'équation (.9).

$$m_i = m_{max}\left(1-\frac{P_{ig}}{P_{max}}\right) \qquad (9)$$

Où :

$\boldsymbol{m_i}$**:** Le taux de mutation pour l'habitat *i*.

$\boldsymbol{m_{max}}$**:** Le taux maximum de mutation.

$\boldsymbol{P_{max}}$**:** La probabilité maximale d'existence. Avant d'appliquer l'opérateur de la mutation, nous devons calculer P_{ig}, la probabilité d'existence de l'habitat *i* dans la génération actuelle

g. Ce paramètre est calculé à partir de sa valeur dans la génération précédente par l'équation (10).

$$P_{ig} = P_{ig-1} + P_{ig} \qquad (10)$$

Les valeurs de P_i avant le début de l'algorithme sont initialisées à $1/n$ pour chaque habitat**.** P_i est le changement de probabilité d'existence de l'habitat *i*, il est donné par l'équation (**11**) :

$$\begin{cases} -(\lambda_i + \mu_i) P_i + \mu_{i+j} P_{i+1} i = 1 \\ \boldsymbol{P_i =} \quad -(\lambda_i + \mu_i) P_i + \lambda_{i-1} P_{i-1} \mu_{i+j} P_{i+1} \qquad 2 \leq i \leq n \\ -(\lambda_i + \mu_i) P_i + \lambda_i + jP_i - 1 \qquad i = n \end{cases} \qquad (11)$$

7. Conclusion

Ce chapitre a été dédié à la présentation des techniques d'optimisation et au juste l'algorithme d'optimisation à base de biogéographie, le fondement théorique de cette méthode a été explicitement présenté dans le but d'ouvrir la voie à l'algorithme de modélisation des systèmes dynamique qui sera développé dans le chapitre 4.

CHAPITRE IV

Modélisation Neuronale des Systèmes Dynamiques

Dans ce chapitre nous allons discuter une technique de modélisation des systèmes dynamiques à base des réseaux de neurones artificiels et l'algorithme d'optimisation à base de biogéographie. Cette technique intègre la capacité d'apprentissage des réseaux de neurones artificiels avec l'algorithme d'optimisation BBO pour approximer les systèmes dynamiques.

Sommaire

1. Introduction

Modéliser les systèmes complexes est un véritable défi. Pour y parvenir, beaucoup de chercheurs ont eu l'idée d'avoir recours à l'intelligence artificielle. Dans ce chapitre nous allons présenter une technique de modélisation des systèmes dynamiques à base des réseaux de neurones artificiels et l'algorithme d'optimisation à base de biogéographie. Cette technique comprend une interconnexion parallèle de deux sous modèles neuronaux. Le premier sous modèle neuronal est le modèle primaire, qui représente un modèle ordinaire à faible résolution. Pour surmonter le problème de la résolution et obtenir un modèle avec une résolution plus élevée, nous avons introduit un deuxième sous-modèle neuronal appelé le modèle d'erreur, qui représente la modélisation d'erreur entre la sortie du modèle primaire et la sortie réelle du système dynamique. Le modèle d'erreur représente l'incertitude dans le modèle principal, cette incertitude est minimisée par une simple soustraction de la sortie du modèle d'erreur de la sortie du modèle primaire, ce qui entraine une interconnexion parallèle entre les deux modèles. Cette interconnexion nous emmène à un seul modèle final unique et complet possédant une résolution plus élevée.

2. Méthode proposée pour la modélisation des systèmes dynamiques

Notre objectif est d'offrir un modèle neuronal fiable et efficace pour représenter les systèmes dynamiques. Un modèle primaire est initialement conçu à l'aide des données réelles d'entrée/ sortie du système dynamique en considération. Ensuite, l'erreur entre la sortie du système dynamique en considération et la sortie du modèle primaire est modélisée pour créer le modèle d'erreur. Ce dernier modèle représente les incertitudes dans le modèle primaire qui peuvent être facilement supprimés par une simple soustraction de la sortie du modèle d'erreur de la sortie du modèle Primaire. Les détails de cette méthode seront présentés dans les sections suivantes.

Un modèle neuronal peut représenter ou modéliser n'importe quel système ou fonction inconnu, $y = f(x)$, en utilisant les données d'entrée/sortie. L'architecture des réseaux de neurones utilisée est les réseaux de neurones non bouclés discutée dans la section.5 du chapitre II, dont l'idée est de trouver une relation entre l'entrée et la sortie du modèle neuronal en ajustant au fur et à mesure ses paramètres en utilisant des mécanismes d'ajustement (algorithme d'optimisation) de telle façon une fonction objectif atteint son minimum. Pour avoir l'objectif voulu notre méthode sera composée de trois étapes :

- ***Étape 1*** *: identification du modèle primaire,*
- ***Étape 2*** *: identification du processus d'erreur,*
- ***Étape 3*** *: conception du modèle final.*

a. Paramètres du modèle neuronal à ajuster

La conception du modèle neuronal pour la modélisation comprend la détermination des paramètres inconnus qui sont les poids de différentes couches du la structure du réseau de neurones artificiels utilisée (voir Figure. 1).

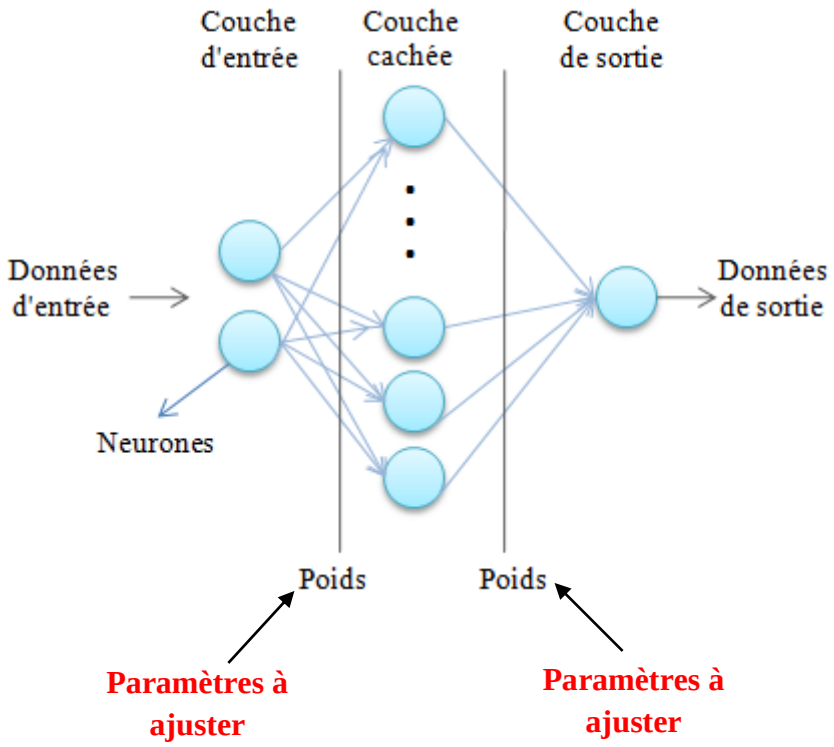


Figure. 1 : Paramètres du réseau de neurones à optimiser.

b. Fonction objectif

Le terme fonction objectif ou la fonction de coût est utilisé en optimisation mathématique pour désigner une fonction qui sert de critère pour déterminer la meilleure solution à un problème d'optimisation. La fonction objectif utilisée est le critère d'erreur quadratique moyenne (MSE : *Mean Square Error*) qui combine les valeurs réelles et estimées comme suit :

$$MSE = \frac{\sum_{k=1}^{N}(y_k - \hat{y}_k)}{N} = \frac{\sum_{k=1}^{N} e^2}{N}, \quad (1)$$

Avec y_k et $\hat{y}_k$ sont la mesure réelle et son estimation, respectivement, et N est la taille des données.

c. Identification du modèle primaire

Pour cette étape, l'ensemble de données d'entrée/sortie est utilisé pour déterminer le modèle neuronal primaire $\hat{f}_p$ pour le processus qui est le système dynamique en considération (voir Figure. 2).

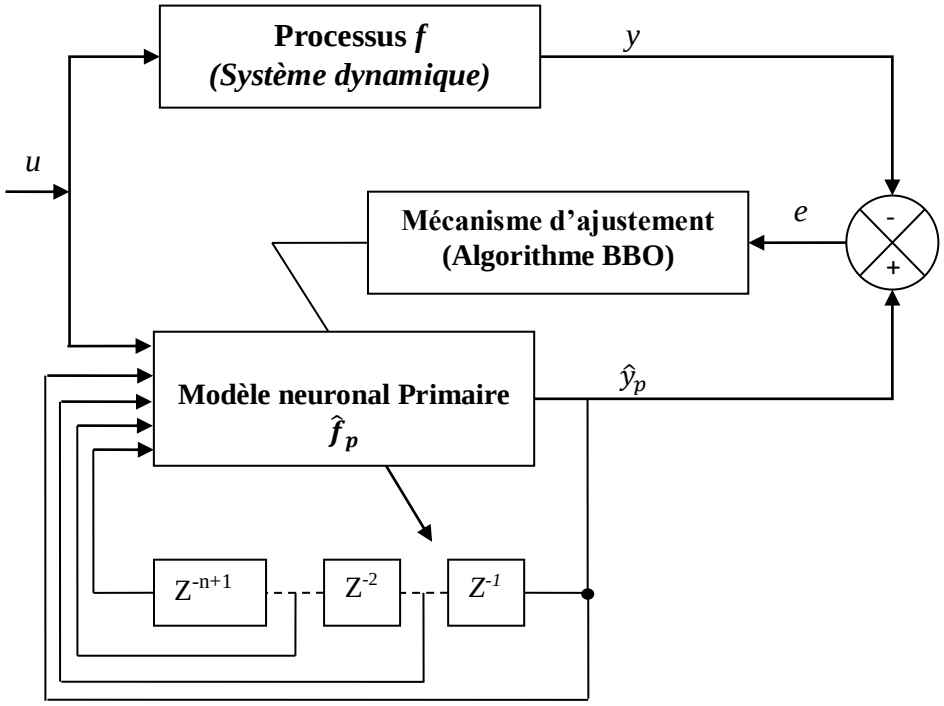


Figure. 2 : Identification du modèle primaire.

Avec :

*f:*fonction inconnue représentant la série temporelle que nous souhaitons identifier, ayant *y* comme sortie.

u : entrée du système

$\hat{y}_p$: la sortie du modèle primaire.

$e = \hat{y}_p - y$: erreur de modélisation, utilisée dans la fonction du cout (1) de l'algorithme d'optimisation (pour exciter le mécanisme d'ajustement).

Le principe de l'identificateur neuronal consiste en une adaptation en temps réel de $\hat{f}_p$. Le bloc mécanisme d'ajustement représenté dans la Figure. 2, est l'algorithme d'optimisation

à base de biogéographie (BBO) discuté dans la section 6.1 du chapitre III qui va ajuster les paramètres du modèle neuronal $\hat{f}_p$ de telle sorte que l'erreur e entre la sortie du processus y et la sortie du modèle primaire $\hat{y}_p$ atteint sa valeur minimale.

Pour identifier $\hat{f}_p$, on a utilisé un modèle autorégressifet moyenne mobile(ARMA) neuronal, qui est une technique de modélisation qui fait partie d'un groupe de formules de prédiction qui tentent de prédire la sortie d'un système en se basant sur les résultats précédents et l'entrée du système (ancienne sortie devient nouvelle entrée).

d. Identification du processus d'erreur

Le processus d'erreur (E_p) est défini comme une interconnexion parallèle entre le processus (système dynamique considéré) et le modèle neuronal primaire comme le montre la figure suivante :

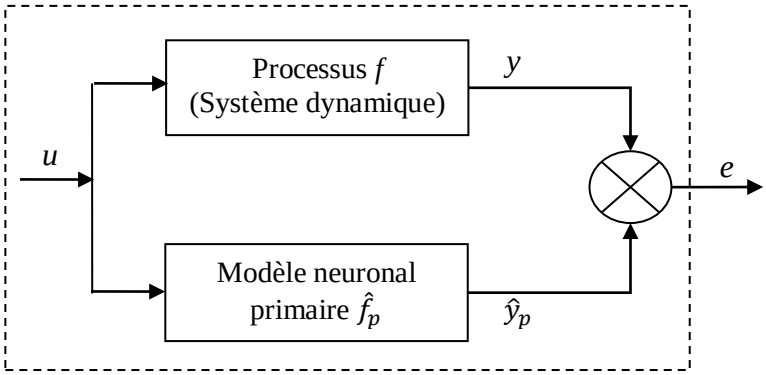


Figure. 3 : Processus d'erreur.

La sortie e du processus d'erreur (E_p) est définit comme suit :

$$e = \hat{y}_p - y. \qquad (2)$$

Une fois que la sortie du processus d'erreur (E_p) est obtenue (Equation. 2), un deuxième modèle neuronal peut être conçu pour la modélisation de l'erreur e. Ce modèle sera étiqueté par le modèle neuronal d'erreur. L'erreur e est considéré comme une série temporelle, donc il convient de concevoir son modèle en utilisant un modèle neuronal autorégressif (AR), qui tente de prédire la nouvelle sortie en fonction des résultats précédents.

La structure de l'étape 2 est illustré dans la Figure. 4, ou $\hat{e}$ est la sortie du modèle neuronal d'erreur et $e_1$est l'erreur de modélisation entre la sortie du processus d'erreur $e$et la sortie du modèle neuronal d'erreur $\hat{e}$.

La tache maintenant est d'ajuster les paramètres du modèle neuronal d'erreur jusqu'à ce que l'erreur e_1 atteint son minimum.

e. Conception du modèle final

Dans cette dernière étape, le modèle primaire $\hat{f}_p$ et le modèle d'erreur $\hat{E}_p$ sont interconnectés en une structure parallèle (voir figure. 5) afin d'obtenir le modèle final $\hat{f}_F$. Cette configuration nous permettra de compenser l'erreur résiduelle (erreur de modélisation) obtenu dans la première étape (modèle primaire), ce qui a comme

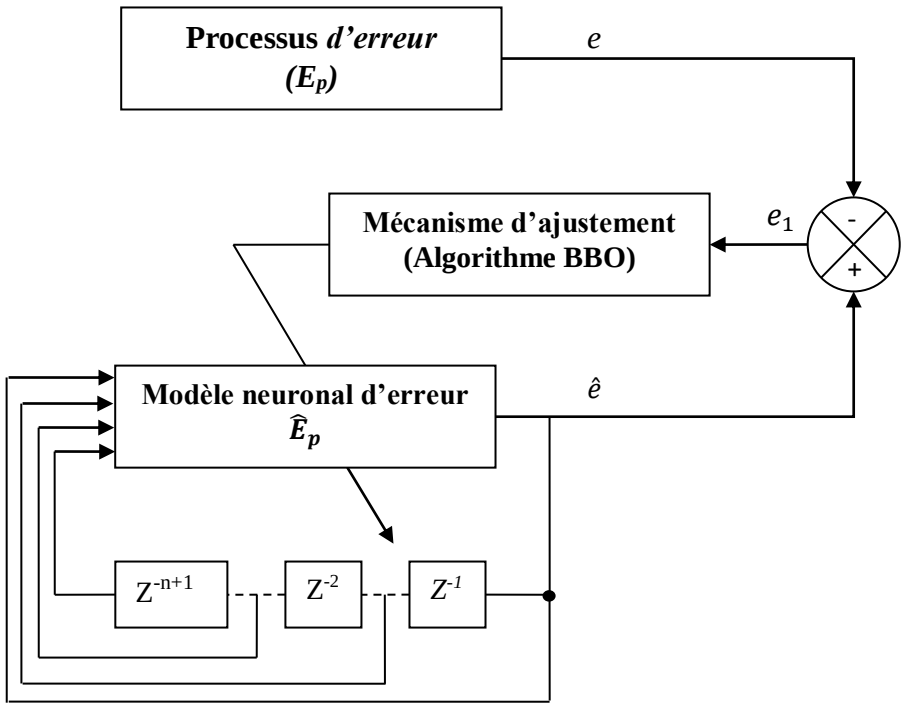


Figure. 4 : Identification du processus d'erreur.

Conséquence la minimisation de l'erreur de modélisation globale. La sortie du modèle final peut être décrite comme suit :

$$\hat{y} = \hat{y}_p - \hat{e}. \qquad (3)$$

Où $\hat{y}_p$ et $\hat{e}$ sont les sorties du modèle neuronal primaire et la sortie du modèle neuronal d'erreur, respectivement.

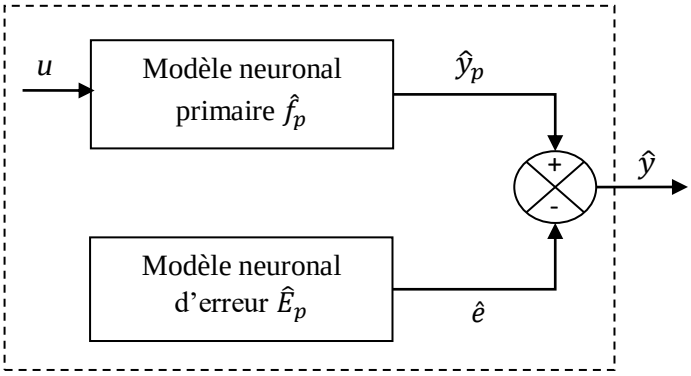


Figure. 5 : Modèle final $\hat{f}_F$.

3. Résultats de simulation

Dans cette section nous allons présenter et discuter les résultats de simulation de notre méthode appliquée pour la modélisation des systèmes dynamiques. Trois systèmes dynamiques [38] décrits ci-dessous vont être considérer pour modélisation :

1. ***Modèle I :***

$$y_p(k+1) = f\left[y_p(k), y_p(k-1), \dots, y_p(k-n+1)\right] + \sum_{i=0}^{m-1} \beta_i u(k-1) \quad (4)$$

2. ***Modèle II***

$$y_p(k+1) = \sum_{i=0}^{n-1} \alpha_i y_p(k-1) + g[u(k), u(k-1) \dots, u(k-m+1)] \quad (5)$$

3. ***Modèle II***

$$y_p(k+1) = f\left[y_p(k), y_p(k-1), \dots, y_p(k-n+1)\right] + g[u(k), u(k-1), \dots, u(k-m+1)] \quad (6)$$

a. Paramètres d'algorithme d'optimisation BBO

La taille de la population utilisée dans les algorithmes optimisations est un facteur important dans la détermination de la solution optimale. Lorsque la taille de la population augmente, la solution dans l'espace d'exploration est améliorée ; cependant cela augmente le temps de calcul. Dans ce travail, la taille de population correspondante à l'algorithme BBO a été fixé à 200. Les différents paramètres d'algorithme d'optimisation BBO sont choisis comme suit : le taux de mutation m = 0.01 et le taux d'immigration et le taux d'émigration sont similaires aux courbes de migration linéaire (Figure. 3 du chapitre III).

b. Modélisation du système I

Dans cette sous-section nous allons examiner l'efficacité la technique de modélisation neuronale sur un système compliqué (Équation 4), ou la non linéarité est associée

seulement aux sorties, en considérant le cas particulier régi par l'équation aux différences suivante :

$$y_p(k+1) = f\left[y_p(k), y_p(k-1)\right] + u(k), \quad (7)$$

Avec :

$$f\left[y_p(k), y_p(k-1)\right] = \frac{y_p(k)y_p(k-1)[y_p(k)+2.5]}{1+y_p^2(k)+y_p^2(k-1)}, \quad (8)$$

et:

$$u(k) = sin\frac{2\pi k}{25} \text{for } 1 \leq k \leq 5 \;\&\; 150 \leq k \leq 200. \quad (9)$$

$$u(k) = 0.5sin\frac{2\pi k}{10} + 0.5sin\frac{2\pi k}{5} \text{ for } 50 \leq k \leq 150. \quad (10)$$

La technique de modélisation des systèmes dynamiques discutée précédemment sera appliquée pour modéliser le système dynamique donné par les Equations 7, 8, 9 et 10. Les résultats de simulation sont illustrés dans les Figures 6-*a*, 6-*b* et 6-*c*, respectivement, telle que :

- ✓ **Figure. 6-*a*** : représente une superposition de la sortie du système dynamique et la sortie du modèle neuronal primaire.
- ✓ **Figure. 6-*b*** : représente une superposition de la sortie du système dynamique et la sortie du modèle neuronal final.
- ✓ **Figure. 6-*c*** : représente une superposition de l'erreur de modélisation, le modèle de l'erreur de modélisation et l'erreur de modélisation finale.

Par une simple inspection visuelle des Figure. 6-*a*, 6-*b* et 6-*c*, on peut remarquer que le modèle final est beaucoup mieux que le modèle primaire, où on constat que l'erreur de modélisation finale a été réduite ce qui permet d'améliorer efficacement le modèle final.

c. Modélisation du système II

Dans ce qui suit nous examinons l'efficacité de notre méthode sur un deuxième système plus compliqué (Équation. 5), ou la non linéarité est associée seulement à l'entrée en prenant en considération l'équation aux différences suivante :

$$y_p(k+1) = 0.3y_p(k) + 0.6y_p(k-1) + f[u(k)], \quad (11)$$

Avec :

$$f(u) = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u). \quad (12)$$

et :

$$u(k) = sin\frac{2\pi k}{250}. \quad (13)$$

La Figure. 7, représente les résultats finaux de l'identificateur neuronal. En inspectant les zooms, nous confirmons l'efficacité de notre méthode pour ce type de problèmes.

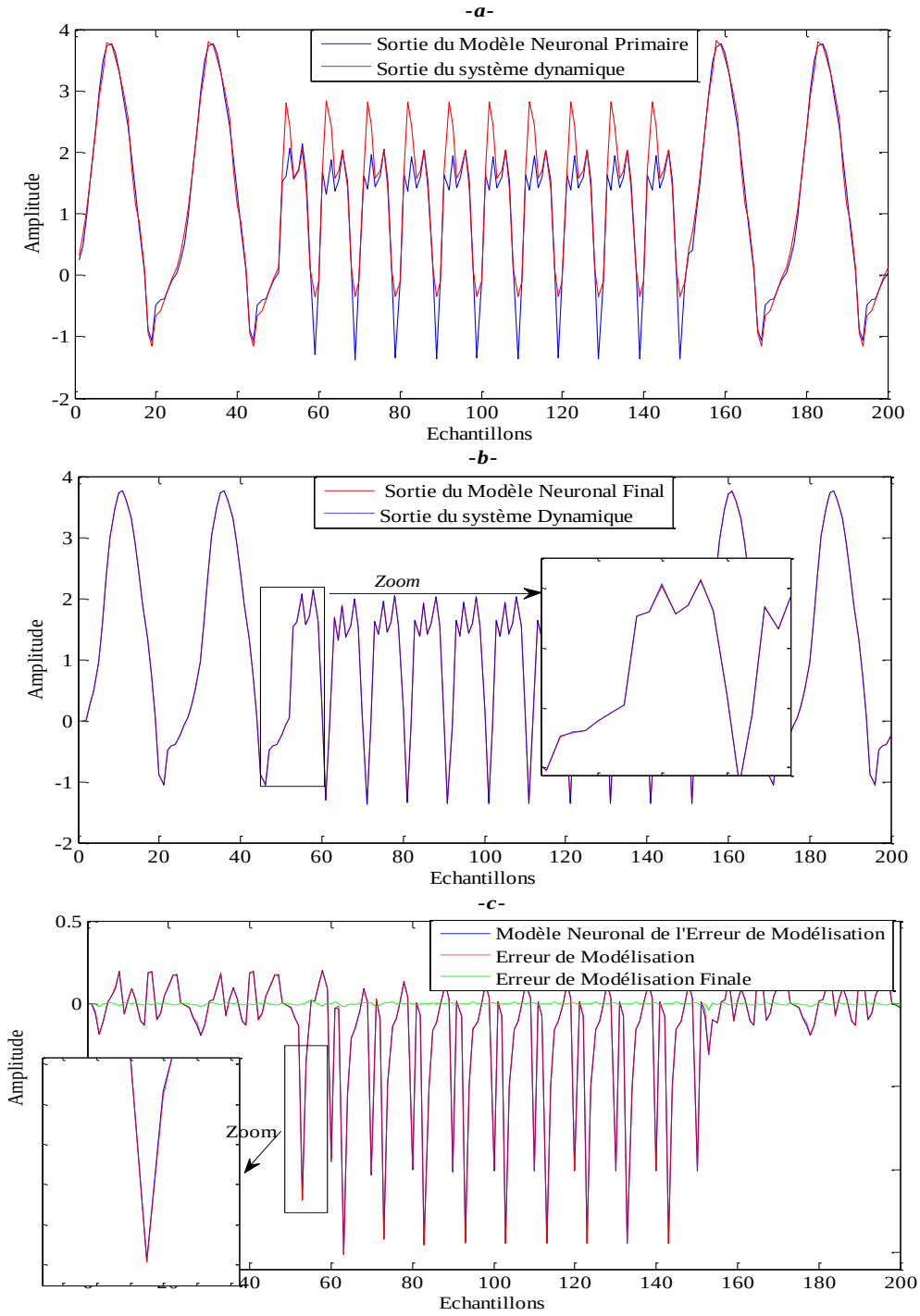


Figure.6 :Résultats de simulation du système dynamique Modèle I : *(a)* Modèle Primaire, *(b)* Modèle Final, *(c)* Erreurs de Modélisation.

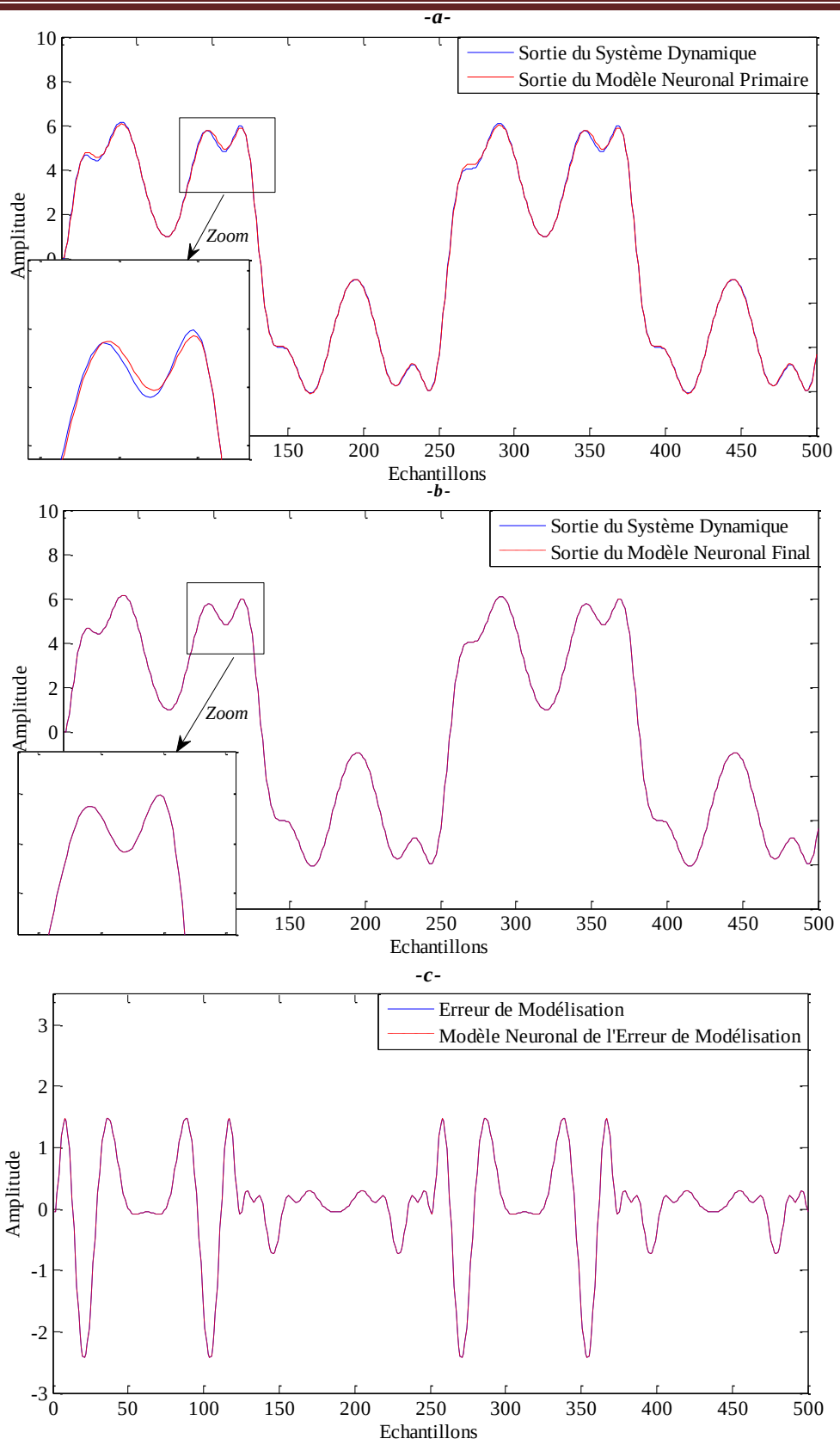


Figure.7:Résultats de simulation du système dynamique Modèle II : *(a)* Modèle Primaire, *(b)* Modèle Final, *(c)* Erreurs de Modélisation.

d. Modélisation du système III

Le modèle III (Équation. 6), pour lequel l'entrée et la sortie du système sont tous les deux compris dans la non linéarité, va être utilisé pour confirmer l'efficacité de notre méthode pour ce type de problèmes, en considérant le cas particulier décrit par l'équation aux différences suivante :

$$y_p(k+1) = f\left(y_p(k), u(k)\right) = \frac{y_p(k)}{1+y_p(k)^2} + u^3(k), \quad (14)$$

où la fonction inconnue *f*à identifier a une forme plus générale du fait que ses variables indépendantes sont les signaux d'entrée et de sortie.

Le signal d'entrée *u* est choisi comme suit :

$$u(k) = sin\left(\frac{2\pi k}{25}\right) + sin\left(\frac{2\pi k}{10}\right). \quad (15)$$

La Figure. 8 , représente les résultats finaux de l'identificateur neuronal. En inspectant les zooms, nous confirmons toujours l'efficacité de notre méthode pour ce type de problèmes.

De tous ces résultats de simulation on peut remarquer l'efficacité de la structure proposée pour la modélisation des systèmes dynamiques. La particularité ici est l'introduction d'un nouveau module s'appelle le modèle d'erreur qui permet de mieux laisser le modèle final du système dynamique considéré. La deuxième particularité est l'utilisation de l'algorithme d'optimisation BBO pour ajuster les paramètres de la structure proposée à base des réseaux de neurones artificiels, ce qui élimine le problème du minimum local rencontré chaque fois en utilisant des méthodes d'optimisation du gradient.

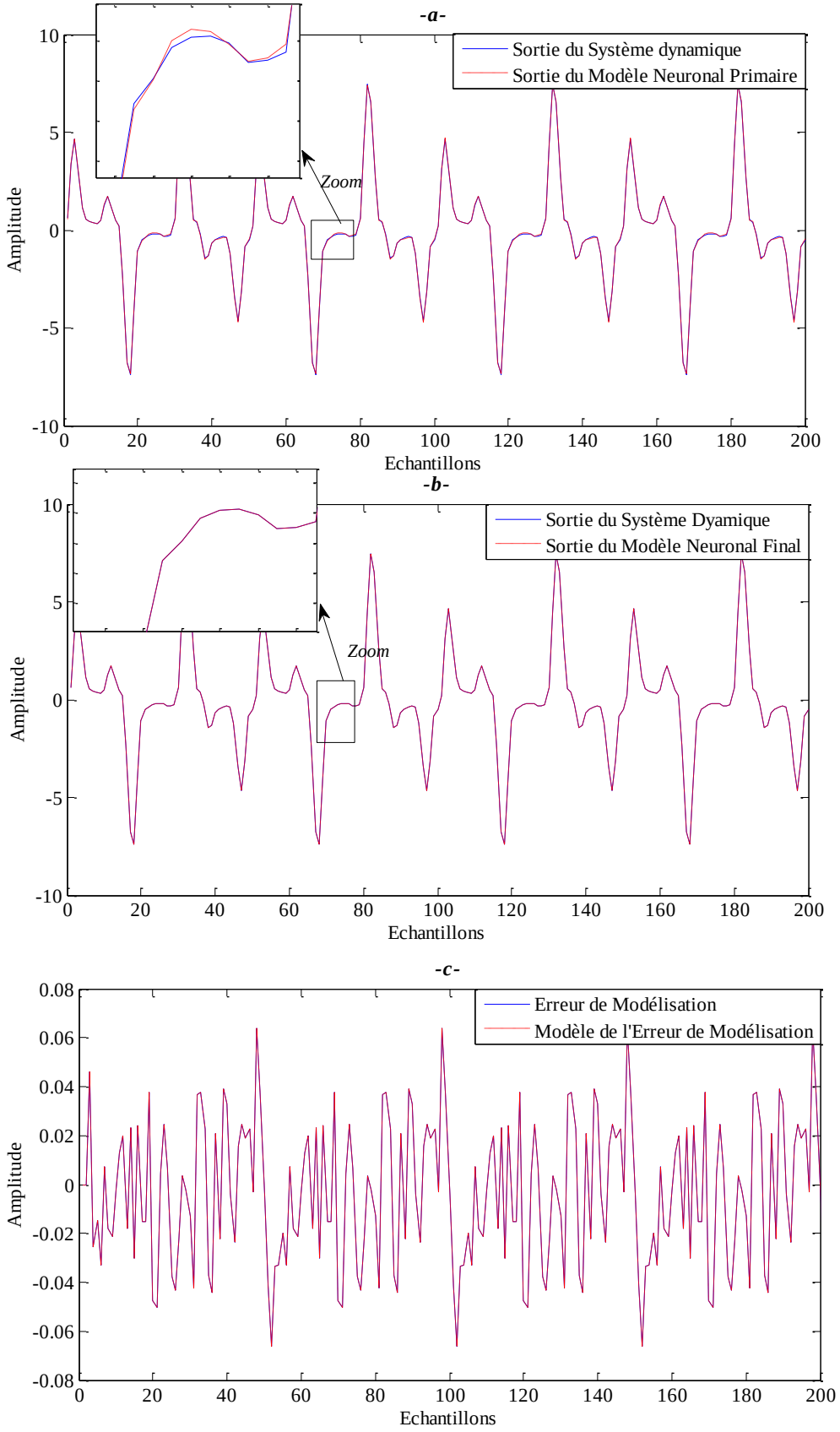


Figure.8:Résultats de simulation du système dynamique Modèle III: *(a)* Modèle Primaire, *(b)* Modèle Final, *(c)* Erreurs de Modélisation.

4. Conclusion

Dans ce chapitre, nous avons présenté une méthode de modélisation des systèmes dynamiques à base des réseaux de neurones artificiels et l'algorithme d'optimisation à base de biogéographie. L'approche permet de généraliser la notion d'identification en ajoutant un nouveau module d'identification, appelé modèle d'erreur. Le modèle neuronal d'erreur introduit a été utilisé comme un complément du modèle primaire pour améliorer la qualité du modèle, ce qui a fourni une bonne précision. L'optimisation des paramètres de la structure proposée a été assurée en utilisant l'algorithme d'optimisation BBO. Plusieurs résultats de simulations ont été introduits pour vérifier l'efficacité de notre méthode.

Conclusion générale

L'objectif de ce mémoire de fin d'étude a été de développer une technique de modélisation des systèmes dynamiques à base des réseaux de neurones artificiels et l'algorithme d'optimisation à base de biogéographie. Cette technique comprend une interconnexion parallèle de deux sous modèles neuronaux. Le premier sous modèle neuronal est le modèle primaire, qui représente un modèle ordinaire à faible résolution. Pour surmonter le problème de la résolution et obtenir un modèle avec une résolution plus élevée, nous avons introduit un deuxième sous-modèle neuronal appelé le modèle d'erreur, qui représente la modélisation d'erreur entre la sortie du modèle primaire et la sortie réelle du système dynamique. Le modèle d'erreur représente l'incertitude dans le modèle principal, cette incertitude est minimisée par une simple soustraction de la sortie du modèle d'erreur de la sortie du modèle primaire, ce qui entraine une interconnexion parallèle entre les deux modèles. Cette interconnexion nous emmène à un seul modèle final unique et complet possédant une résolution plus élevée.

L'approche développer dans le cadre de ce mémoire de fin d'étude permet de généraliser la notion d'identification en ajoutant un nouveau module d'identification, appelé modèle d'erreur. Le modèle neuronal d'erreur introduit a été utilisé comme un complément du modèle primaire pour améliorer la qualité du modèle, ce qui a fourni une bonne précision. L'optimisation des paramètres de la structure proposée a été assurée en utilisant l'algorithme d'optimisation BBO. Plusieurs résultats de simulations ont été introduits pour vérifier l'efficacité de la structure développée.

REFERENCES

[1] Habchi, F. (2013). Identification et commande des systèmes non linéaires par les techniques neuronales (Doctoral dissertation, Université abderrahmane mira béjaia).

[2] Merzouka, N. (2018). Etude des performances des réseaux de neurones dynamiques à représenter des systèmes réel: une approche dans l'espace d'état (Doctoral dissertation).

[3]Hammami, S. (2009). Sur la stabilisation de systèmes dynamiques continus non linéaires exploitant les matrices de formes en flèche: application à la synchronisation de systèmes chaotiques (Doctoral dissertation, Ecole centrale de Lille).

[4] Bonnet, S. (2008). Appoches numérique pour la commande des systèmes dynamiques. Thèse de doctorat, Université de Technologie de Compiegne, France.

[5]Saidi , M.(2012).Etude dynamique d'une application discrète du plan.(Magistère, université de m'sila)

[6]Mohamed, S. (2016). Une méthode topologique pour la recherche d'ensembles invariants de systèmes continus et à commutation (Doctoral dissertation).

[7] Touzet, C. (1992). Les réseaux de neurones artificiels, introduction au connexionnisme.

[8] H.Abdi. « *Les Réseaux de neurones* », édition presses universitaires de Grenoble, 1994.

[9] Youcef Djeriri (2017). Les réseaux de neurones artificiels, University of Sidi-bel-Abbes.

[10] Ammar, M. Y. (2007). Mise en œuvre de réseaux de neurones pour la modélisation de cinétiques réactionnelles en vue de la transposition batch/continu (Doctoral dissertation).

[11] Coulibaly, P., Anctil, F., & Bobée, B. (1999). Prévision hydrologique par réseaux de neurones artificiels: état de l'art. *Canadian Journal of civil engineering*, *26*(3), 293-304.

[12] Sorin, F., Broussard, L., &Roblin, p. (2001). Régulation d'un processus industriel par réseaux de neurones. Techniques de l'ingénieur. Informatique industrielle, 2(S7582), S7582-1.

[13] Rivals, I., Personnaz, L., Dreyfus, G., & Ploix, J. L. (1995). Modélisation, classification et commande par réseaux de neurones: principes fondamentaux, méthodologie de conception et illustrations industrielles. Les réseaux de neurones pour la modélisation et la commande de procédés, JP Corriou, ed.(Lavoisier Tec & Doc, 1995).

[14] Jayet, A. (2002). Affective Computing: Apport des Processus Emotionnels aux Systèmes Artificiels. Site.

[15] Dreyfus, G., Martinez, J. M., Samuelides, M., Gordon, M. B., Badran, F., Thiria, S., & Hérault, L. (2002). *Réseaux de neurones-Méthodologie et applications* (No. BOOK). Eyrolles.

[16] Mahamed GH Omran. *Particle swarm optimization methods for pattern recognition and image processing*. PhD thesis, 2006.

[17] Dan Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[18]James Kennedy. Swarm intelligence. In *Handbook of nature-inspired and innovativecomputing*, pages 187–219. Springer, 2006.

[19]Carlos M Fonseca, Peter J Fleming, et al. Genetic algorithms for multiobjectiveoptimization : Formulationdiscussion and generalization. In *Icga*, volume 93, pages416–423, 1993.

[20]Julio Barrera and Carlos A Coello Coello. Test function generators for assessing theperformance of PSO algorithms in multimodal optimization. *Handbook of Swarm Intelligence, Adaptation, Learning, and Optimization*, 8 :89–117, 2011.

[21]H. Azzag, F Picarougne, C Guinot, and G Venturini. Un survol des algorithmes biomimétiques pour la classification. *Classification et fouille de données, RNTI-C*, 1,2004.

[22]Jason Brownlee. *Clever algorithms : nature-inspired programming recipes*. JasonBrownlee, 2011.

[23]John H Holland. Adaptation in natural and artificial systems. an introductory analysiswith application to biology, control, and artificial intelligence. *Ann Arbor, MI :University of Michigan Press*, 1975.

[24] Sinha, Nidul, R. Chakrabarti, and P. K. Chattopadhyay. "Evolutionary programming techniques for economic load dispatch." *IEEE Transactions on evolutionary computation* 7.1 (2003): 83-94.

[25]Alavi, Maryam, and John C. Henderson. "An evolutionary strategy for implementing a decision support system." *Management Science* 27.11 (1981): 1309-1323.

[26] Nordin, Peter, Robert E. Keller, and Frank D. Francone. *Genetic programming*. Ed. Wolfgang Banzhaf. Springer, 1998.

[27]XH Shi, YC Liang, HP Lee, C Lu, and LM Wang. An improved GA and a novelPSO-GA-based hybrid algorithm. *Information Processing Letters*, 93(5) :255–261,2005.

[28]Kennedy J and Eberhart R. Particle swarm optimization. *Neural Networks, 1995.Proceedings., IEEE International Conference on*, 4 :1942–1948, 1995.

[29]Craig W Reynolds. Flocks, herds and schools : A distributed behavioral model. ACM SIGGRAPH computer graphics, 21(4) :25–34, 1987.

[30]Frank Heppner and Ulf Grenander. A stochastic nonlinear model for coordinatedbird flocks. *The ubiquity of chaos*, pages 233–238, 1990.

[31]Xin-She Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[32]Xin-She Yang. Firefly algorithm, levy flights and global optimization. *Research anddevelopment in intelligent systems XXVI*, pages 209–218, 2010.

[33]Xin-She Yang. Firefly algorithm, stochastic test functions and design optimisation.*International Journal of Bio-Inspired Computation*, 2(2) :78–84, 2010.

[34]Xin-She Yang. Firefly algorithms for multimodal optimization. In *Internationalsymposium on stochastic algorithms*, pages 169–178. Springer, 2009.

[35]Dorigo, Marco, and Gianni Di Caro. "Ant colony optimization: a new meta-heuristic." Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). Vol. 2. IEEE, 1999.

[36]Dasgupta, Dipankar, ed. Artificial immune systems and their applications. Springer Science & Business Media, 2012.

[37]Dan Simon. Biogeography-based optimization. IEEE transactions on evolutionary computation, 12(6) :702–713, 2008.

[38] Kumpati S Narendra and Kannan Parthasarathy. Identification and control of dynamicalsystems using neural networks. *IEEE Transactions on neural networks*, 1(1):4–27, 1990.

ملخص

الهدف من أطروحة نهاية الدراسة هذه هو توفير نموذج عصبي موثوق وفعال لتمثيل الأنظمة الديناميكية. تم تصميم النموذج الأساسي مبدئيًا باستخدام بيانات الإدخال / الإخراج الفعلية من النظام الديناميكي قيد الدراسة. بعد ذلك ، يتم نمذجة الخطأ بين ناتج النظام الديناميكي قيد الدراسة ومخرجات النموذج الأساسي لإنشاء نموذج الخطأ. يمثل النموذج الأخير حالات عدم يقين في النموذج الأساسي يمكن إزالتها بسهولة عن طريق طرح ناتج نموذج الخطأ من ناتج النموذج الأساسي.

Résumé

L'objectif de ce mémoire de fin d'étude est d'offrir un modèle neuronal fiable et efficace pour représenter les systèmes dynamiques. Un modèle primaire est initialement conçu à l'aide des données réelles d'entrée/ sortie du système dynamique en considération. Ensuite, l'erreur entre la sortie du système dynamique en considération et la sortie du modèle primaire est modélisée pour créer le modèle d'erreur. Ce dernier modèle représente les incertitudes dans le modèle primaire qui peuvent être facilement supprimés par une simple soustraction de la sortie du modèle d'erreur de la sortie du modèle Primaire.

Abstract

The objective of this end of study project is to provide a reliable and efficient neural model to represent dynamic systems. A primary model is initially designed using actual input / output data from the dynamic system under consideration. Then, the error between the output of the dynamic system under consideration and the output of the primary model is modeled to create the error model. The latter model represents uncertainties in the primary model that can be easily removed by simply subtracting the output of the error model from the output of the Primary model.