Université Larbi Ben M'hidi Oum-El-Bouaghi
Faculté des Science et Science Appliquées
Département de Génie Electrique

Mémoire

Présenté pour obtenir le diplôme de

MASTER

Filière: **Electronique**

Spécialité: **Electonique des Systèmes Embarqués**

Contrôle de la température d'une réaction chimique dans un CSTR par une méthode intelligente

Par :

Younssi Khaoula *&* ***Belabed Taqwa Imène***

Proposé et dirigé par :

Mr Djebabla. Ali

Juin 2023


بسم الله الرحمن الرحيم

Dédicace

Nous dédions ce modeste travail à nos chers parents

À nos frères et sœurs et toute la famille

À tous nos professeurs

À tous nos amis et nos collègues

Nous n’oublions pas les gens qui nous ont aidés

Remerciements

Tout d’abord, nous remercions Dieu de nous avoir donné la force, la volonté et le courage afin d’accomplir ce modeste travail.

Ces remerciements s’adressent à nos familles qui nous ont toujours aidées et Encouragées durant nos longues années d’études.

Nos remerciements vont à notre encadreur Dr. Djebabla Ali pour toute sa gentillesse, pour ses précieux conseils et pour sa patience avec nous, ainsi tous ceux qui nous ont aidés et soutenus dans notre travail.

Nos profonds remerciements pour les membres de jury qui ont accepté d’évaluer ce travail.

Finalement, nous tenons à exprimer notre profonde gratitude à l’ensemble des enseignants qui ont contribué à notre formation et à tous ce qui ont contribué pour la réalisation de ce mémoire

Liste des figures

Liste des tableaux

Liste des acronymes

RNA	Réseaux de Neurones Artificiels
MLP	Perceptron Multicouche
NNE	Emulateur de réseau neuronal
NNC	Contrôleur de réseau neuronal
PSO	Optimisation de l'essaim de particules
CSTR	Réacteur à cuve agitée continue

Contenu

Introduction générale

L'optimisation est un sujet central de la recherche scientifique, de nombreux problèmes d'aide à la décision peuvent être expliqués en termes de problèmes d'optimisation. Les problèmes d'identification, l'apprentissage supervisé des réseaux de neurones ou la recherche du chemin le plus court sont de même.

Les algorithmes d'optimisation des essaims de particules forment une classe de métaheuristiques, largement proposés pour résoudre les problèmes d'optimisation classés difficiles. Ces algorithmes s'inspirent du comportement collectif d'oiseaux et de bancs de poissons particulièrement sages.

Un réseau de neurones est un modèle d'apprentissage automatique basé sur l'apprentissage supervisé et non supervisé. Composés de structures cellulaires artificielles, ils représentent des modèles qui nous permettent d'aborder les problèmes de perception, de mémoire, d'apprentissage et de réflexion sous de nouvelles perspectives. C'est une alternative prometteuse pour éviter certaines limitations des algorithmes classiques. Car le traitement de l'information et les mécanismes inspirés des cellules nerveuses sont complètement différents de celle déjà rencontrées, utilisées dans les méthodes classiques. Cela nous permet de résoudre des problèmes qui étaient auparavant classés comme difficiles ou complexes.

L'optimisation d’entrainement des réseaux de neurones est un processus qui consiste à ajuster les paramètres d'une structure de celui-ci afin de minimiser une fonction objectif qui est généralement fonction de l'écart entre les prédictions du modèle et les résultats attendus.

Le réacteur chimique à cuve agitée continue ‘CSTR’ joue un rôle principal dans toutes les industries de procédés chimiques pour le mélange. Le CSTR est exploité dans de larges applications où les industries permettent des réactions liquides, gazeuses et solides, avec une agitation continue et une configuration en série pour la concentration et la température. Par conséquent, pour atteindre une concentration spécifique, la concentration totale de la solution est mélangée avec les proportions d'eau souhaitées.

Les contrôleurs conventionnels ont du mal à traiter les problèmes qui apparaissent dans les processus non linéaires complexes. Afin de résoudre ce problème et d'améliorer la réponse dynamique du CSTR, les performances du réacteur sont analysées à l'aide d'une technique de calcul souple dite ‘optimisation par essaim de particules’ (PSO).

Afin de remédier à cet handicap, l'objectif de notre travail se résume à la proposition d'une méthode de contrôle de ce type de processus complexe et fortement nonlinéaire. L'idée de base consiste à renforcer les capacités d'approximation universelle de la structure du réseau de neurones MLP par la capacité de collaboration de plusieurs particules pour la recherche d'un optimum dans un espace de recherche. Tout cela peut se concrétiser par une simple formulation telle que : 'optimiser l'entrainement d'un réseau MLP par un essaim de particules'.

Ce travail fait, l'objet de trois chapitres :

- Le premier chapitre est dédié à une brève présentation de la théorie des réseaux de neurones artificiels. Après avoir exposé un aperçu historique sur les RNAs, nous exposons les structures de base d'un réseau neuronal, puis expliquer la notion d'apprentissage qui est à la base de leur comportement intelligent, enfin clôturer par les principales méthodes d'émulation et de contrôle utilisant cet outil.
- Le deuxième chapitre, a été consacré à l'étude de la métaheuristique dite PSO, et à l'association de celle-ci aux réseaux de neurones MLP pour en déduire à une technique originale proposer pour l'émulation et le contrôle de systèmes complexes.
- Le troisième chapitre présente tous les résultats des tests après mise à l'épreuve de l'algorithme et afficher les performances de celui-ci dans le but de sa validation.

Chapitre 01

Emulation et contrôle par réseaux de neurones

1.1 Introduction

L'intelligence artificielle est à la base destinée à fabriquer des machines à apprendre et à mémoire distributive. Ce calcul est effectué en parallèle par un grand nombre d'unités, dites neurones formels, qui effectuent chacune des opérations élémentaires et contribuent ensemble au résultat simulant un comportement ou construisant une idée complexe.

Les réseaux de neurones représentent la concrétisation récente d'un vieux rêve : celui de la « machine intelligente », susceptible d'imiter et, bien sûr, de dépasser les capacités du cerveau humain, ce qui nous a motivé à entamer ce type d'étude pour découvrir cet outil qui est à la base de l'apprentissage, de la perception et du raisonnement déjà acquis par certains machines et processus qui nous entourent en industrie et dans notre vie quotidienne.

Ce premier chapitre est une base théorique portant sur le neurone formel, les principales structures de réseaux neuronales ainsi que leurs méthodes d'apprentissage afin de les exploiter dans divers domaines notamment en tant qu'approximateurs, émulateurs ou contrôleurs.

1.2 Introduction aux réseaux de neurones artificiels

1.2.1 Historique

En **1943**, *Mac Culloch* et *Pitts* ont proposé le premier modèle de neurone biologique. Ce dernier, appelé aussi neurone à seuil, a été inspiré des récentes découvertes, en biologie. Ce sont des neurones logiques (0 ou 1). Ces deux physiciens ont montré que les neurones formels peuvent réaliser des fonctions logiques. **[1]**

Les études des réseaux de neurones ont passé plusieurs générations comme suit :

- ✓ **1949 :** Introduction du terme connexionnisme, faisant référence à des modèles massivement parallèles et connectés, et la proposition de la « loi de *Hebb* ».
- ✓ **1958 :** Développement du modèle Perceptron.
- ✓ **1982 :** La démonstration l'intérêt des réseaux entièrement connectés, et la conception un mécanisme d'apprentissage pour les réseaux multicouches de type perceptron : la rétropropagation.
- ✓ **1986 :** Vulgarisation de l'algorithme de mécanisme d'apprentissage de réseaux multicouches de type perceptron par la Méthode de rétropropagation.

Depuis cette recherche, les applications des réseaux de neurones n'ont cessé de croître.

1.2.2 Modèle biologique

Les réseaux de neurones artificiels sont des modèles qui se rapprochent le plus possible du fonctionnement réel des réseaux de neurones biologiques, d'où découlent les associations implicites du cerveau. Le transfert de ces structures vers un système informatique permet au programme d'acquérir les connaissances tacites pour lui-même. Une meilleure compréhension du fonctionnement des neurones biologiques conduira à une meilleure compréhension de leurs équivalents en informatique.

Un neurone biologique est une cellule nerveuse qui peut transmettre des informations à d'autres neurones par diverses connexions (synapses). Sa base est constituée de structures ramifiées appelées dendrites qui reçoivent des signaux électriques d'autres neurones.

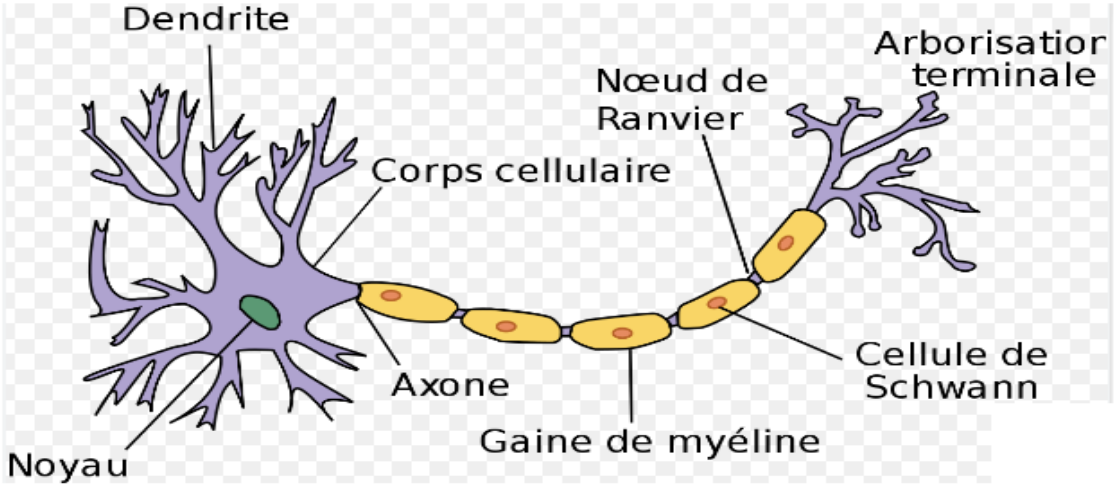

Figure 1.1: Schéma d'un neurone biologique

1.2.3 Rôle d'un neurone dans un réseau artificiel

Un réseau de neurones est un système d'opérateurs non linéaires interconnecté, qui reçoivent des signaux de l'extérieur via ses entrées et fournissent des signaux de sortie, Métaphore de la structure cérébrale, du traitement parallèle et de l'information distribuée, Ces réseaux de neurones comportent de multiples éléments de traitement appelés neurones.

Dans un réseau, chaque élément effectue un traitement indépendant et transmet le résultat de son analyse au nœud suivant. Ainsi, les informations fournies au réseau seront propagées couche par couches, de la couche d'entrée à la couche de sortie à travers plusieurs couches intermédiaires appelées couches cachées.

Par conséquent, chaque neurone d'une couche intermédiaire est connecté aux neurones de la couche précédente et ceux de la couche suivante. Les réseaux de neurones artificiels ont la capacité d'exploiter des données empiriques et de les rendre utilisables. Les résultats de traitement du réseau seront mémorisés dans des poids synaptiques, qui sont acquis par des processus d'adaptation ou d'apprentissage.

Le réseau de neurones n'est jamais programmé pour effectuer une tâche spécifique, mais il s'entraîne sur les données reçues, grâce au mécanisme d'apprentissage qui opère sur les composants du réseau, afin de réaliser au mieux la tâche souhaitée.

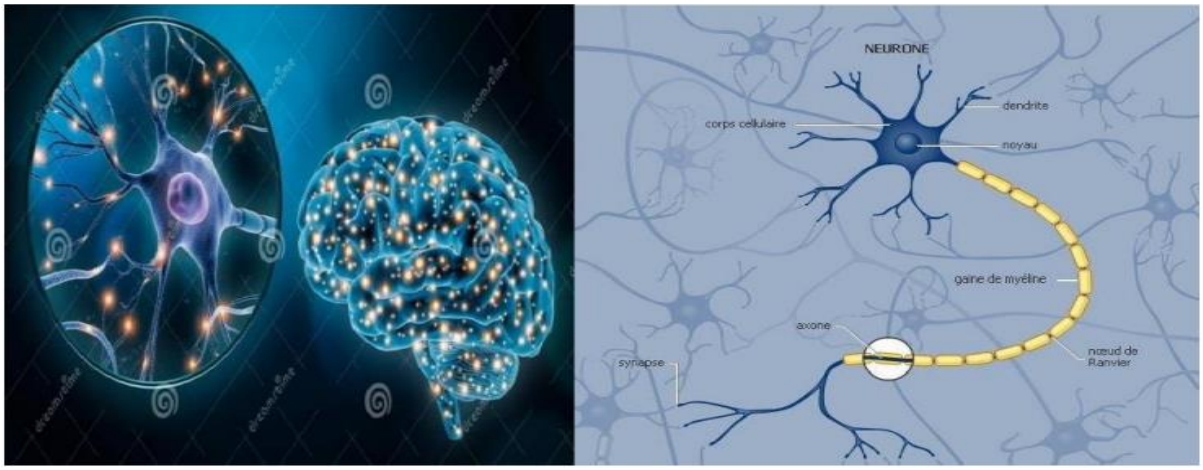

Figure 1.2: Structure d'un réseau de neurone biologique copiée du cerveau humain.

Le modèle mathématique ci-dessous de la figure 1.3 illustre toutes les composantes du neurone artificiel:

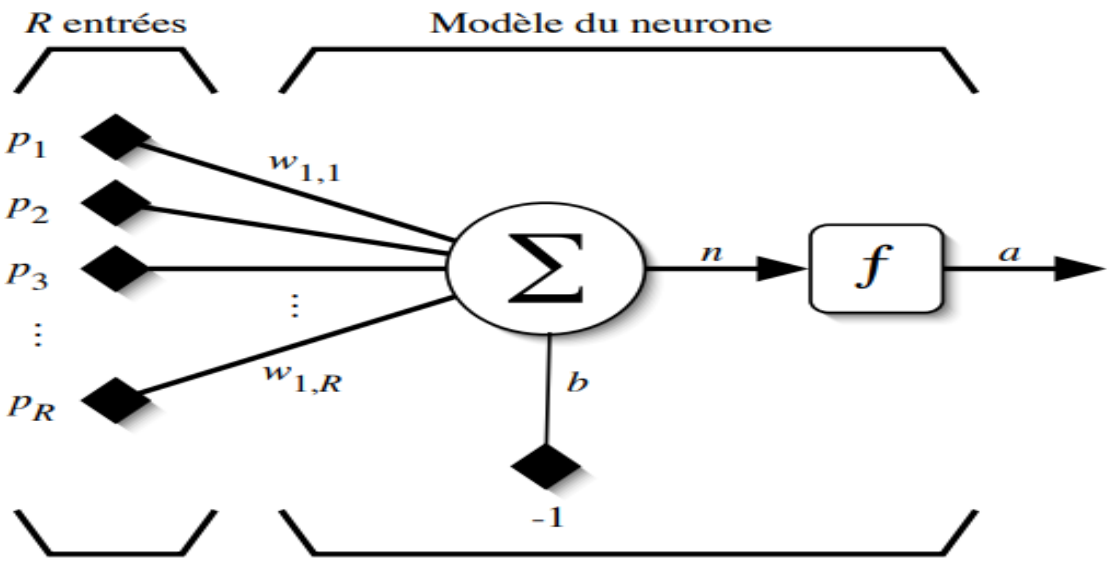

Figure 1.3: Modèle d'un neurone artificiel

Un neurone est essentiellement constitué d'un intégrateur qui effectue la somme pondérée de ses entrées. Le résultat **n** de cette somme est ensuite transformé par une fonction de transfert **f** qui produit la sortie **a** du neurone. Les **R** entrées du neurone correspondent au vecteur :

$$\boldsymbol{p} = [\boldsymbol{p}_1 \ \mathbf{p}_2 \ldots \ \boldsymbol{p}_\mathrm{R}]\boldsymbol{p}^\mathrm{T} \qquad \textbf{(1.1)}$$

alors que :

$$\boldsymbol{W} = [\omega_{1,1}\omega_{1,2} \ldots \omega_{1,\mathrm{R}}]^\mathrm{T} \qquad \textbf{(1.2)}$$

représente le vecteur des poids du neurone.

La sortie n de l'intégrateur est donnée par l'équation suivante : **[2]**

$$\boldsymbol{n} = \sum_{j=1}^{R} \omega_{1,\mathrm{j}}\, \boldsymbol{p}_\mathrm{j} - \boldsymbol{b} \qquad \textbf{(1.3)}$$

$$\boldsymbol{n} = \omega_{1,1}\boldsymbol{p}_1 + \omega_{1,2}\boldsymbol{p}_2 + \ \ldots \ + \omega_{1,\mathrm{R}}\boldsymbol{p}_\mathrm{R} - \boldsymbol{b} \qquad \textbf{(1.4)}$$

i : entrée du i^{ème} nœud de la couche précédente.

j : sortie du j^{ème} nœud vers la couche suivante.

R : Nombre d'entrées de la couche actuelle.

Qu'on peut aussi exprimer sous forme matricielle :

$$\boldsymbol{n} = \boldsymbol{W}^\mathrm{T}\,\mathbf{P} - \mathbf{b} \qquad \textbf{(1.5)}$$

1.2.4 Architecture des réseaux de neurones artificiels

L'architecture d'un réseau de neurones artificiel est définie par les connexions entre les neurones. Elle est spécifiée par les nombre d'entrées, de sorties et des nœuds ainsi que la façon selon laquelle sont interconnectés et organisés.

a. Les réseaux de neurones non bouclés

Un réseau de neurones non bouclé réalise une fonction algébrique de ses entrées, par composition des fonctions réalisées par chacun de ses neurones. Il est représenté graphiquement par un ensemble de neurones "connectés" entre eux, l'information circulante des entrées vers les sorties.

- **Réseaux de neurones monocouches (Perceptron)**

C'est le réseau le plus simple, il est caractérisé par :

- Les informations en entrée.
- Les neurones, que l'on représente généralement verticalement. En théorie, chacun pourrait avoir une fonction d'activation différente.
- Chacun des neurones est connecté aux informations d'entrée.

Le réseau de neurones possède ainsi **N** informations en entrée et **P** sorties, chaque neurone renvoyant sa sortie.

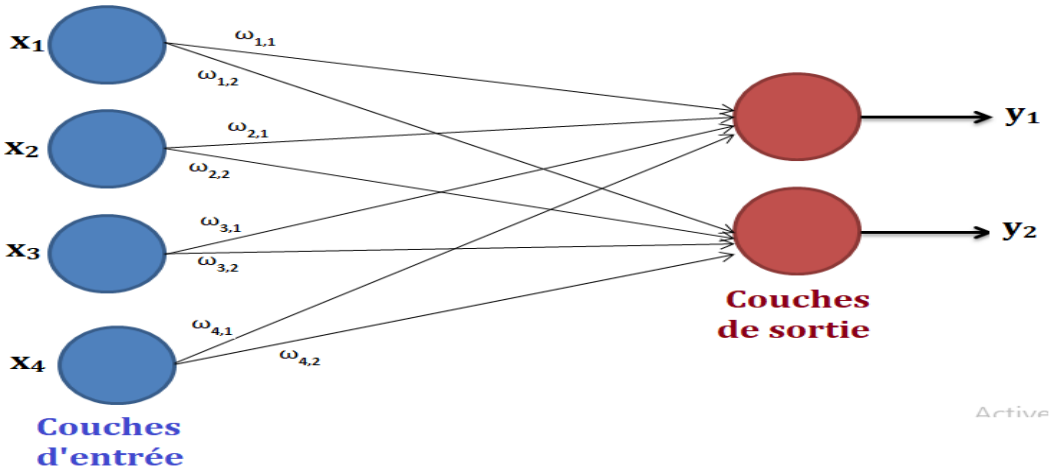

Figure 1.4: Schéma d'un réseau de neurones monocouche

- **Réseaux de neurones multicouches MLP**

Dans ce type de réseaux, les neurones de la première couche reçoivent toutes les informations d'entrée, la deuxième couche reçoit toutes les sorties des neurones de la première couche, et ainsi de suite jusqu'à ce que les neurones de sortie reçoivent les informations de la dernière couche cachée.

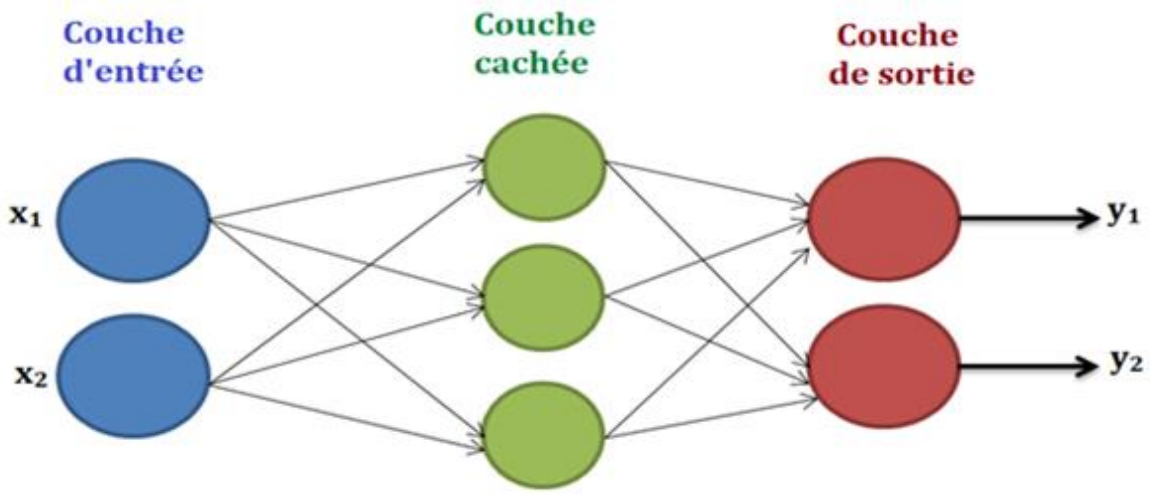

Figure 1.5 : Schéma d'un exemple de réseau de neurones MLP non bouclé

Pour un réseau MLP à une seule couche cachée, l'expression de la sortie est de la forme:

$$f(x) = \sum_{i=1}^{N} \omega_i * f(x_i) \qquad (1.6)$$

ω_i: $i^{ème}$ poids entre la couche cachée et la couche de sortie.

x_i : $i^{ème}$ entrée pondérée entre la couche d'entrée et la couche cachée.

f:fonction d'activation au niveau des nœuds de la couche cachée.

- **Réseaux de neurones à connexions locales**

Il s'agit d'une structure multicouche, mais conserve une certaine topologie. Chaque neurone entretient des relations avec un nombre réduit et localisé de neurones de la couche avale. Les connexions sont moins nombreuses que dans le cas d'un réseau multicouche MLP classique. **[3]**

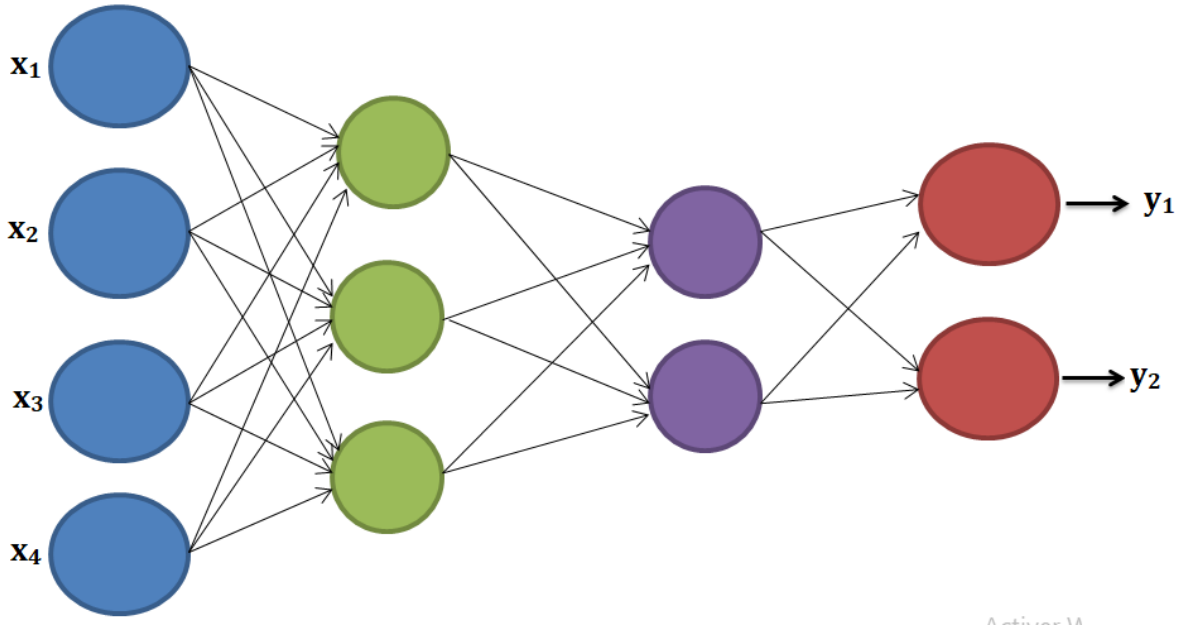

Figure 1.6: Schéma d'un exemple de réseau de neurones à connexions locales

b. 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. Il s'agit donc des réseaux de neurones avec retour en arrière (*feedback network ou récurrent network*). **[3]**

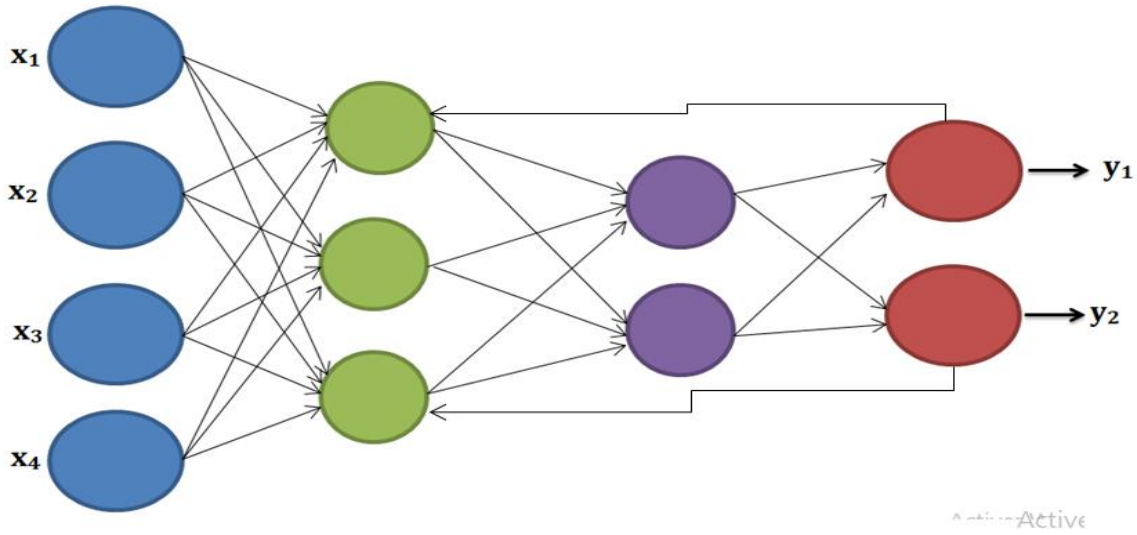

Figure 1.7: Schéma d'un réseau de neurones bouclé

1.2.5 Caractéristiques des réseaux de neurones artificiels

Les réseaux de neurones formels, possèdent une propriété remarquable qui est à l'origine de leur intérêt pratique dans des domaines très divers: ce sont des approximateurs universels parcimonieux. Sans entrer dans les détails mathématiques, la propriété d'approximation peut être énoncée de la manière suivante : toute fonction bornée suffisamment régulière peut être approchée avec une précision absolu, dans un domaine fini de l'espace de ses variables, par un réseau de neurones comportant une couche de neurones cachés en nombre fini, possédant tous la même fonction d'activation, et un neurone de sortie linéaire. **[4]** Cette propriété n'est pas spécifique aux réseaux de neurones: il existe bien d'autres familles de fonctions paramétrées possédant cette propriété.

La spécificité des réseaux de neurones réside dans le caractère parcimonieux de l'approximation: à précision égale, les réseaux de neurones nécessitent moins de paramètres ajustables que les approximateurs universels couramment utilisés, plus précisément, le nombre de poids varie linéairement avec le nombre de variables de la fonction à approcher, alors qu'il varie exponentiellement pour la plupart des autres approximateurs. **[5]**

En pratique, dès qu'un problème fait intervenir plus de deux variables, les réseaux de neurones sont, en général, préférables aux autres méthodes.

1.2.6 Apprentissage d'un réseau de neurones artificiels

L'apprentissage est un processus visant à améliorer les performances d'un système en se basant sur ses expériences passées. Cette méthode intervient lorsque le problème paraît trop compliqué à résoudre en temps réel, ou lorsqu'il paraît impossible de résoudre le problème de manière classique et rigoureuse. **[6]**

a. Types d'apprentissage

Il existe trois grandes méthodes d'apprentissage: l'apprentissage supervisé, l'apprentissage non supervisé, et l'apprentissage par renforcement.

- L'apprentissage supervisé :

C'est l'apprentissage le plus couramment utilisé et il est très bien maîtrisé. Son inconvénient est que l'agent n'est pas immédiatement autonome, puisqu'il a besoin d'un superviseur qui dans un premier temps lui indique la marche à suivre dans des situations qu'il pourra rencontrer. Si la base d'apprentissage est complète, l'agent saura réagir aux situations auxquelles il sera confronté. Cependant les situations doivent présenter une certaine constance : si l'agent est confronté à une situation entièrement nouvelle, il sera incapable de s'y adapter car aucun exemple donné par le superviseur n'y correspondra. **[6]**

- L'apprentissage non supervisé :

Contrairement à l'apprentissage supervisé, on ne fournit ici qu'une base d'entrée, et c'est le système qui doit déterminer ses sorties en fonction des similarités détectées entre les différentes entrées (règle d'auto-organisation). **[6]**

- L'apprentissage par renforcement :

Imaginant un agent, capable de percevoir et agir, qui va choisir ses actions en fonction de sa perception de son environnement et dont le but va être de maximiser une récompense qu'il obtient en fonction des actions qu'il réalise **[7].** Dans ce contexte, l'apprentissage par renforcement est une technique qui permet de trouver par un processus d'essais et d'erreurs, l'action optimale à effectuer pour chacune des situations « états» que l'agent va percevoir afin de maximiser ses récompenses **[8].** C'est une méthode d'apprentissage orienté objectif qui conduit à un contrôleur optimal pour la tâche spécifiée par les récompenses. Elle est aussi non supervisée car la récompense ne

donne pas l'action optimale à réaliser mais simplement une évaluation de la qualité de l'action choisie **[8]**.

1.2.7 Entrainement des réseaux de neurones

a. Réseau de neurones émulateur : NNE

L'émulateur neuronal du système est décrit par une fonction $\boldsymbol{\varphi}$ qui représente la sortie du réseau émulateur NNE en fonction de l'entrée donnée **u(t).**

La relation mathématique $\boldsymbol{\varphi}$ devrait ressembler à la fonction **f** du système émulé.

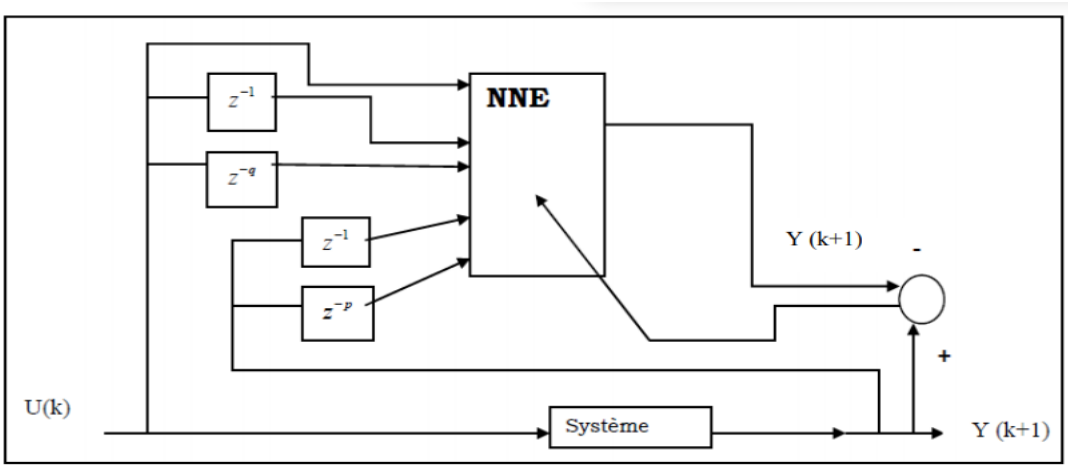

Figure 1.8 : Structure d'entrainement d'un réseau de neurones émulateur

Après entrainement, l'émulateur doit être capable de reproduire la même sortie que le système pour des entrées identiques. La formation est effectuée pour minimiser l'erreur entre la sortie du système et la sortie du réseau émulateur NNE.

La formation peut être menée en ligne ou hors ligne si suffisamment de données sont disponibles. Le but de la formation est de trouver des poids w_i qui minimisent l'erreur. Une fois la convergence atteinte, le système peut être simulé à l'aide du réseau NNE.

b. Réseau de neurones contrôleur : NNC

- Définition:

Le contrôle neuronal est l'étude des systèmes de commande à base de réseaux de neurones. Un réseau contrôleur est un réseau dont la sortie est le signal de commande u(t)**.** L'intérêt de ce type de contrôleurs est dû à la capacité d'apprentissage des réseaux de neurones. Un réseau de neurones contrôleur avec des poids initiaux aléatoires doit

apprendre la dynamique inverse du système en exploitant ses entrées-sorties à travers le temps.

Dans ce qui suit, nous considérons un système décrit par une équation récurrente non linéaire de la forme :

$$\boldsymbol{u(t+1) = f(y(t), y(t-1), \dots, y(t-p+1), u(t), \dots, u(t-q))} \qquad \textbf{(1.7)}$$

Où:

$\boldsymbol{f}$ **:** est une fonction non linéaire.

$\boldsymbol{p}$et$\boldsymbol{q}$**:** peuvent être connus ou bien estimés par $\hat{\mathbf{p}}$et$\hat{\mathbf{q}}$.

Généralement on suppose que $\hat{\mathbf{p}}$et$\hat{\mathbf{q}}$sont connus, et $\boldsymbol{f}$ inconnue.

Le but du contrôleur neuronal est de construire un réseau qui permet au système de suivre une référence fixe ou variable.

- Méthodes de contrôle par les réseaux de neurones :

On va présenter quelque méthodes répandues:

- Contrôle direct par modèle inverse :

Un réseau de neurones apprend la dynamique inverse d'un système. Le modèle inverse résultant (contrôleur) est mis en cascade avec le système à contrôler pour obtenir la fonction d'identité entre la sortie souhaitée $\mathbf{Y_d}$ et la sortie réelle du système **Y**. La reconstruction des commandes par le modèle inverse permet une confirmation directe des erreurs dans le vecteur de commande.

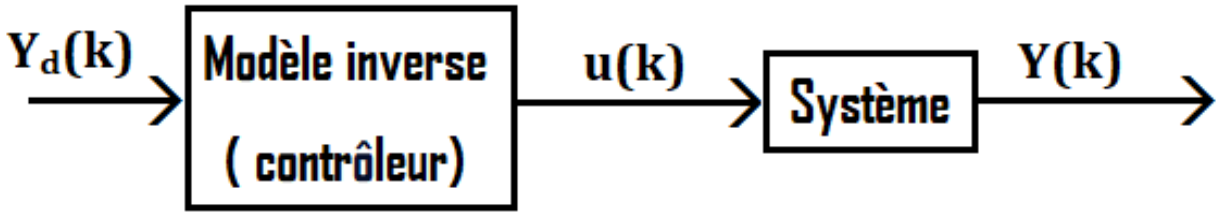

Figure 1.9 : Structure de contrôleur par modèle inverse

- Contrôle adaptatif direct :

Le contrôle adaptatif direct consiste en un apprentissage en ligne du réseau selon la configuration suivante :

Les performances en boucle fermée désirées du système sont spécifiées par un modèle de référence, défini par ses couples d'entrées /sorties (u(k), y(k)), le système contrôlé essaie de suivre le dit-modèle:

$$\mathbf{E_p} = \frac{1}{2}(\boldsymbol{yd(k) - y(k)})^2 \qquad \textbf{(1.8)}$$

Cette structure nécessite la connaissance du *Jacobien* du système (la dérivé de la fonction d'erreur $\mathbf{E_p}$), souvent pas facile à calculer et qui n'est pas disponible que si on a un modèle mathématique qui représente le système. **[9]**

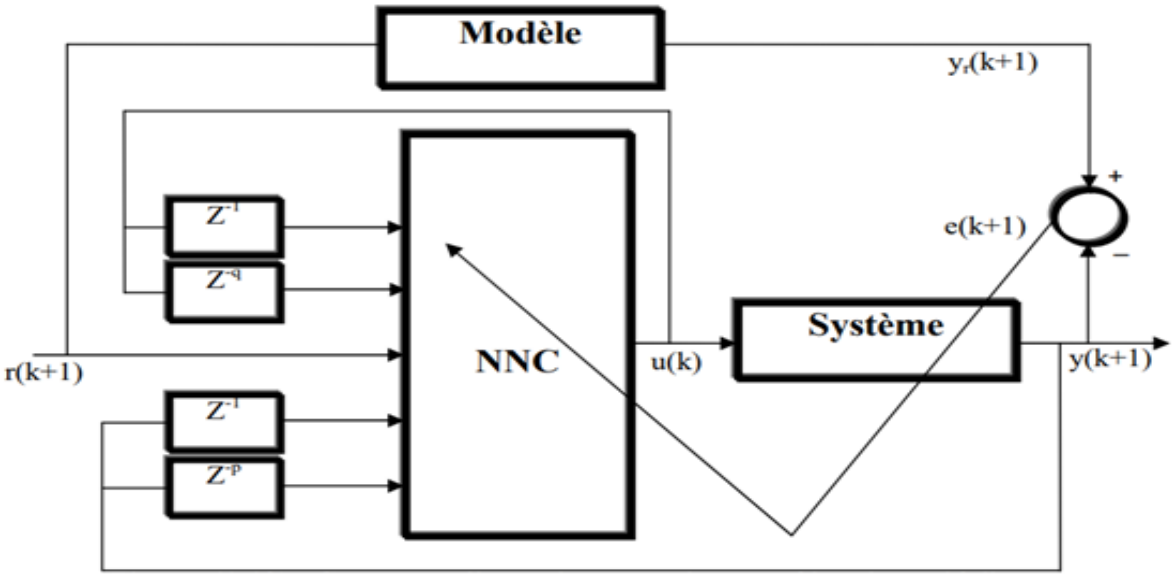

Figure 1.10: Structure d'entrainement du contrôle adaptatif direct

- Contrôle adaptatif indirect :

La structure générale de ce genre de contrôleur est représentéepar la figure elle contient deux réseaux de neurones : l'un est un émulateur qui a pour rôle d'estimer les paramètres du système, le second est le contrôleur, il génère la commande convenable en exploitant les résultats de l'émulateur.

En premier lieu, l'émulateur est entraîné hors ligne, émulant le système à contrôler par le calcul du *Jacobien* de la fonction d'erreur par rapport à la sortie du contrôleur neuronal.

En deuxième étape les deux réseaux, contrôleur et émulateur, sont entraînés en ligne pour suivre le modèle de référence. **[9]**

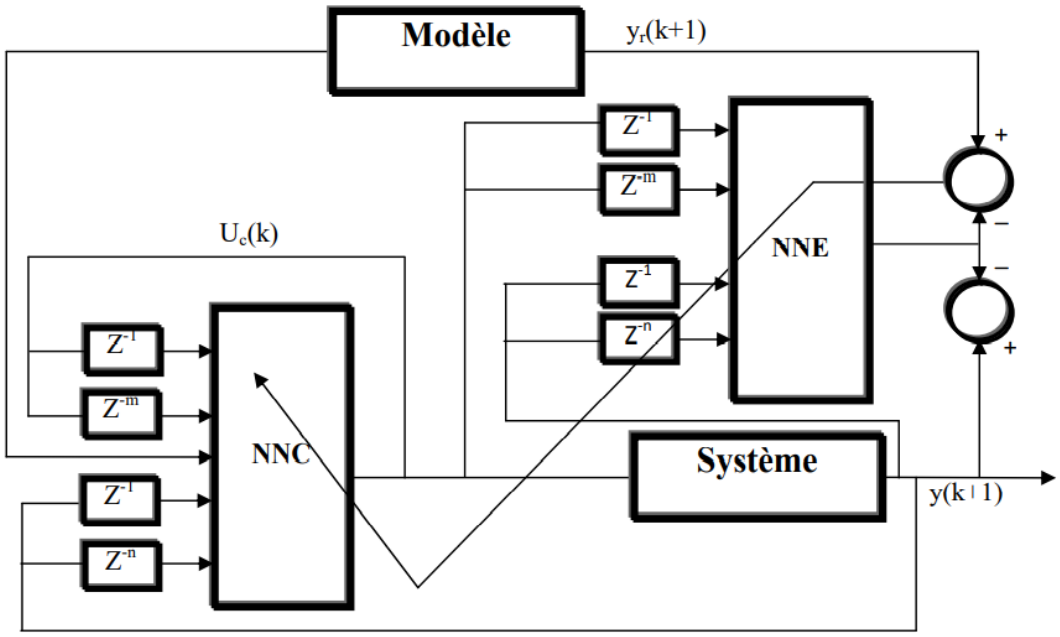

Figure 1.11: Structure d'entrainement du contrôle adaptatif indirect

1.2.8 Domaines d'application des réseaux de neurones artificiels

Les grands domaines d'application des réseaux de neurones découlent naturellement de leur propriété fondamentale :

- La régression non linéaire, ou modélisation de données statiques: il existe une immense variété de phénomènes statiques qui peuvent être caractérisés par une relation déterministe entre des causes et des effets.
- La modélisation de processus dynamiques non linéaires: modéliser un processus, c'est trouver un ensemble d'équations mathématiques qui décrivent le comportement dynamique du processus, c'est-à-dire l'évolution de ses sorties en fonction de celle de ses entrées.
- La commande de processus: commander un processus, c'est imposer à celui-ci un comportement défini à l'avance en fonction des signaux de commande.
- La classification: cette propriété remarquable que les réseaux de neurones partagent avec d'autres classifieurs, n'est malheureusement pas mise à profit dans la plupart des applications. **[10]**

1.3 Conclusion

Les réseaux de neurones artificiels sont devenus très importants dans de nombreux domaines de l'intelligence artificielle et de l'informatique en général. Voici quelques raisons pour lesquelles les RNA sont prometteurs à l'avenir :

- ✓ Ils sont très performants : Les RNA ont démontré leur capacité à résoudre des problèmes complexes et difficiles dans de nombreux domaines tels que le traitement de la parole, la reconnaissance de formes, la vision par ordinateur, la traduction automatique et bien d'autres. Les RNA ont souvent surpassé les performances des méthodes traditionnelles pour ces problèmes.

- ✓ Ils peuvent apprendre à partir de données : Les RNA sont des modèles d'apprentissage automatique qui peuvent apprendre à partir de données. Ils peuvent être entraînés sur des ensembles de données pour apprendre à effectuer des tâches spécifiques, telles que la classification d'images ou la reconnaissance de la parole. Cela les rend très utiles dans des domaines tels que l'analyse de données, la reconnaissance de formes et l'apprentissage automatique.

- ✓ Ils sont adaptatifs : Les RNA peuvent s'adapter à des situations changeantes et à des données nouvelles. Ils peuvent être entraînés à partir de nouvelles données pour améliorer leur performance et pour s'adapter à des changements dans les données d'entrée.

- ✓ Ils peuvent être utilisés pour des tâches diverses : Les RNA peuvent être utilisés pour de nombreuses tâches différentes, telles que la classification, la prédiction, la segmentation d'images, la reconnaissance de la parole, la traduction automatique, la génération de texte, etc. Cela les rend très polyvalents et utiles dans de nombreux domaines.

Actuellement les réseaux de neurones artificiels sont devenus très importants car ils sont performants, ils peuvent apprendre à partir de simples données, adaptatifs à toutes les situations et imprévus et utilisés pour une large variété de taches.

Chapitre 02

Optimisation d'entrainement des Réseaux de Neurones MLP par Essaim de Particules

2.1 Introduction

Le traitement par métaheuristique est une approche innovante pour la résolution de problèmes complexes qui offre de nombreux avantages, notamment en matière de rapidité, de précision et d'efficacité. Celles-ci sont souvent utilisées pour résoudre des problèmes d'optimisation, c'est-à-dire trouver la meilleure solution possible dans un espace de recherche prédéfini.

Les structures de réseaux de neurones peuvent être utilisées pour modéliser la fonction à optimiser, tandis que les métaheuristiques peuvent optimiser l'entrainement de ceux-ci. L'association de ces deux méthodes permettra-t-elle d'améliorer nettement les performances lors d'un contrôle d'un processus industriel complexe ? C'est dans ce sens que nous tenterons de proposer un algorithme d'optimisation qui exploite conjointement les points forts de ces deux outils pour aboutir à des résultats relativement meilleurs à ceux obtenus antérieurement.

2.2 Notion d'optimisation

2.2.1 Définition

L'optimisation est un ensemble de techniques permettant de trouver les valeurs des variables qui optimisent la fonction de réponse, également appelée fonction "objectif". Au niveau mathématique, cela équivaut à trouver les extrêmes d'une fonction. Le domaine des sciences appliquées consiste généralement à trouver des réponses optimales à des opérations industrielles ou à des expériences de laboratoire.

2.2.2 Problèmesd'optimisation

Le but d'un problème d'optimisation est de trouver une solution maximisant ou minimisant une fonction objective donnée. A chaque problème d'optimisation on peut associer un problème de décision dont le but est de déterminer s'il existe une solution pour laquelle la fonction objective soit supérieure ou égale (ou inférieur ou égale) à une valeur donnée.

Mathématiquement, un problème d'optimisation est défini par un ensemble de solutions possibles **S** dont la qualité peut être décrite par une fonction objectif **f**. Ensuite on essaie de trouver la solution **x'** avec la meilleure qualité **F(x')** (on peut alors minimiser ou maximiser **F(x)**). Les problèmes d'optimisation peuvent imposer des contraintes d'égalité (ou d'inégalité) sur S, ils peuvent être dynamiques si **F(x)** varie dans le temps, ou

être multiples si plusieurs fonctions objectifs doivent être optimisées.

Dans le cadre d'un problème de minimisation :

$$\boldsymbol{F(x') \leq F(x) \rightarrow F(x') = minF(x)} \qquad \textbf{(2.1)}$$

De même, pour un problème de maximisation :

$$\boldsymbol{F(x') \geq F(x) \rightarrow F(x') = maxF(x)} \qquad \textbf{(2.2)}$$

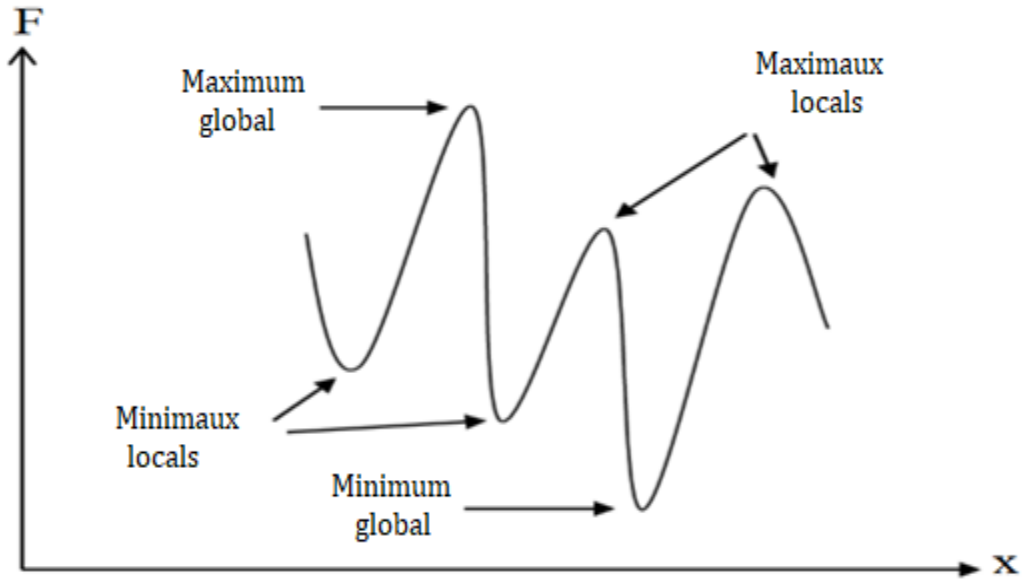

Figure 2.1: Points singuliers d'une fonction

2.2.3 Classification des methods d'optimisation

Il existe deux grandes familles de méthodes d'optimisation : méthodes exactes et d'approximation.

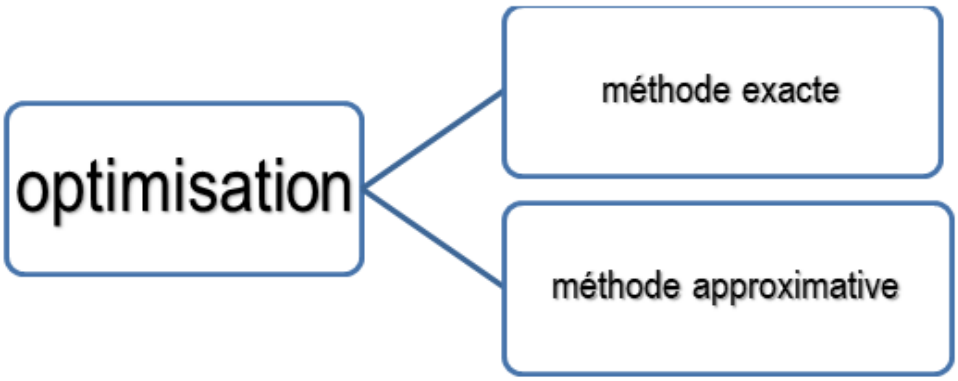

Figure 2.2: Classification des méthodes d'optimisation

a. Méthodes exactes

Les méthodes exactes constituent la base d'une pensée mathématique éprouvée. Elles sont généralement utilisées avec succès pour des problèmes facilement modélisables de nature linéaire et continue. Les méthodes exactes les plus couramment utilisées pour les problèmes d'ordonnancement sont :

- Programmation dynamique : Utilisée dans le cas où la fonction objectif possède la propriété de décomposition.
- Procédure par séparation et évaluation (*branch and bound*) : le principe est la décomposition du problème de façon arborescente en sous problèmes. Dans cette méthode nous trouvons la meilleure solution puisqu'on analyse tous les sous-ensembles de solutions possibles.
- Programmation linéaire : La programmation linéaire (PL) consiste à la résolution de problèmes complexes visant à obtenir le meilleur résultat possible en tenant compte de certaines contraintes. **[11]**

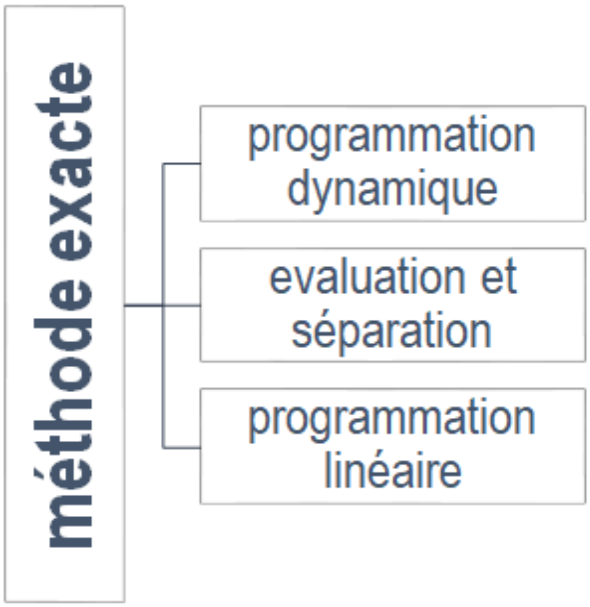

Figure 2.3: Classification des méthodes exactes

b. Méthodes approchées

Les difficultés rencontrées par les méthodes exactes pour la résolution des problèmes d'optimisation de grande taille en un temps raisonnable ont incité les chercheurs à trouver des alternatives connues sous l'appellation : méthodes approchées.

[12]. Ces méthodes peuvent être appliquées à n'importe quelle classe de problèmes. Basé sur la recherche de la solution idéale dans un délai raisonnable. Il existe deux classes d'heuristiques et de métaheuristiques.

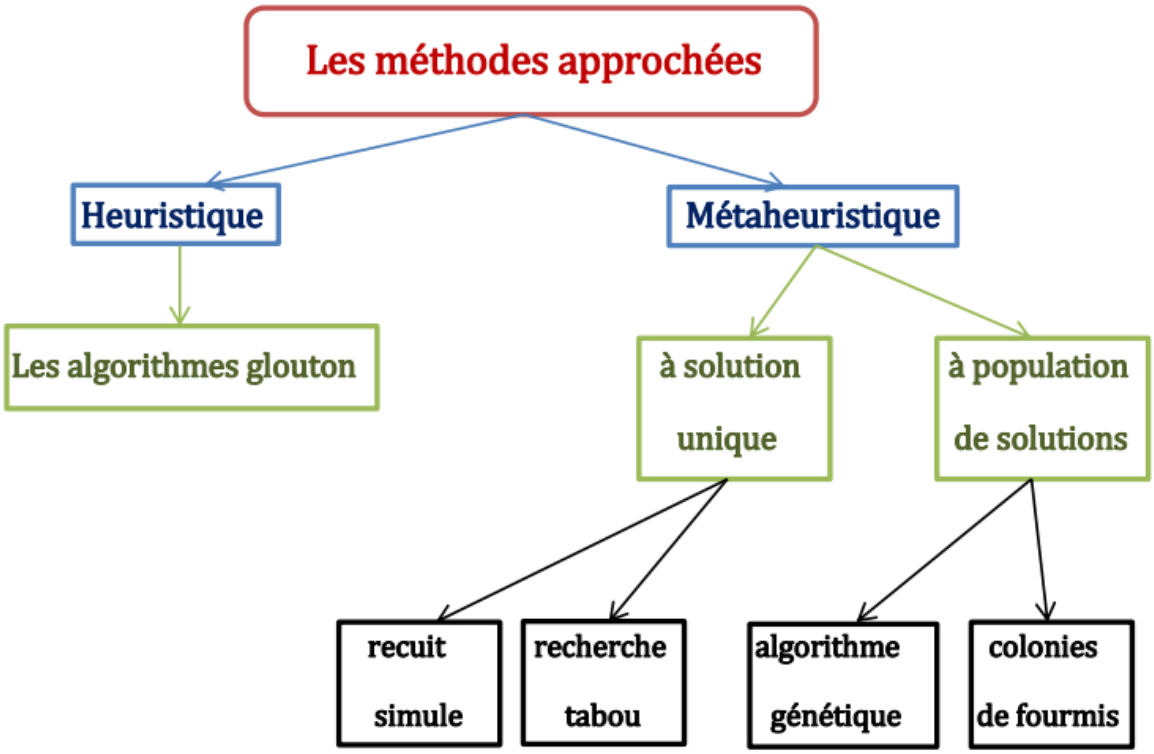

Figure 2.4: Classification des méthodes approchées

- **Définition des heuristiques** :

Les heuristiques sont des simples règles empiriques et, contrairement aux algorithmes, reposent sur des corrélations cumulatives entre les expériences et les résultats plutôt que sur une analyse scientifique trop complexe (car elles nécessitent la définition et l'attribution de nombreux facteurs). Plus simplement, ces règles utilisent les résultats précédents et leurs analogies pour optimiser les recherches futures en les examinant d'abord.

- **Définition des métaheuristiques** :

Les métaheuristiques sont des stratégies qui guident la recherche de solutions et explorent efficacement l'espace de recherche pour déterminer le point optimal. Les métaheuristiques peuvent être considérées comme des algorithmes probabilistes itératifs qui manipulent une ou plusieurs solutions à la recherche d'une solution optimale.

Les itérations continues doivent pouvoir passer des solutions de mauvaise qualité à

une qualité meilleures.

- **Métaheuristiques avec solution unique** :

Les métaheuristiques à base de solution unique dépendent du principe de la recherche au niveau du voisinage. Elles commencent par une solution initiale aléatoire, puis tentent à améliorer progressivement celle-ci en cherchant au dans l'espace de recherche. **[13]**

Les méthodes à solution unique englobent principalement la méthode de descente, la recherche locale itérée, la recherche à voisinage variable, la méthode du recuit simulé et la recherche tabou. Ces méthodes diffèrent essentiellement dans la manière d'exploiter le voisinage local par le choix des candidats et les critères de déplacement. **[14]**

- **Métaheuristiques avec une population de solutions** :

Les métaheuristiques à base d'une population de solutions débutent la recherche avec une panoplie de solutions. Elles s'appliquent sur un ensemble de solutions afin d'en extraire la meilleure qui représentera la solution du problème traité. L'idée d'utiliser un ensemble de solutions au lieu d'une seule solution renforce la diversité de la recherche et augmente la possibilité d'émergence de solutions de bonne qualité. **[15]**

2.3 Optimisation des réseaux MLP par essaim de particules PSO

2.3.1 Principe de la méthode PSO

L'optimisation par essaim de particules (PSO) est une méthode d'optimisation stochastique pour les fonctions non linéaires. Elle est basée sur la reproduction du comportement social développé par le *Dr Eberhart* et le *Dr Kennedy*. Il s'agit d'un algorithme d'optimisation qui fonctionne à l'aide de calculs évolutifs. Il est largement utilisé pour les problèmes d'optimisation.

L'optimisation par essaim de particules s'appuie sur un ensemble d'individus, appelés population, disposés d'une manière spécifique. Ces particules se déplacent dans l'espace de recherche, constituant des solutions potentielles. Chaque individu a une mémoire des solutions les mieux visitées et la capacité de communiquer avec les particules environnantes. Ainsi, la particule suit une direction basée sur les solutions les mieux visitées et influencées par les solutions de ses voisins. À partir de l'idéal local et empirique, toutes les particules convergeront typiquement vers l'optimum global du problème.

Une particule est caractérisée, à l'instant t, par :

1. $\vec{x}$**i(t):** sa position dans l'espace de recherche
2. $\vec{v}$**i(t):** sa vitesse de déplacement
3. $\vec{x}$ **Pbesti**: la position de la meilleure solution par laquelle elle est passée
4. $\vec{v}$ **vbesti**: la vitesse de la meilleure solution à son voisinage
5. **Gbesti**: la valeur fitness de sa meilleure solution
6. **Vbesti**: la valeur fitness de la solution d'espace la plus connue

Le déplacement de la particule i entre les itérations t et t+1 se fait selon :

$$\boldsymbol{V(t+1) = V(t) + C_1 r_1 \big(Pb(t) - X(t)\big) + C_1 r_1 \big(Pg(t) - X(t)\big)} \quad \textbf{(2.3)}$$

$$\boldsymbol{X(t+1) = X(t) + V(t+1)} \quad \textbf{(2.4)}$$

- ✓ $\mathbf{C_1}$ et $\mathbf{C_2}$: deux constantes qui représentent les coefficients d'accélération, elles peuvent être non constantes dans certains cas selon le problème d'optimisation posé.
- ✓ $\mathbf{r_1}$ et $\mathbf{r_2}$: deux nombres aléatoires tirés de l'intervalle [0,1].

2.3.2 Configuration des paramètres

Le PSO de base est influencé par des nombreux paramètres de contrôle, à savoir la dimension du problème, le nombre des particules, la disposition des particules, les coefficients de confiance, la vitesse maximale, le facteur d'inertie, le facteur de construction, la notion du voisinage et le critère d'arrêt.

- Taille de l'essaim :

Une série d'études empiriques a montré que le PSO peut trouver des solutions optimales avec de petites tailles d'essaim de 10 à 30 particules.

- Notion de voisinage :

La taille de voisinage définit la gamme des interactions sociales d'essaim. Plus le voisinage est petit, moins il y a d'interaction. Des voisinages plus petits entraînent une convergence plus lente, mais une convergence plus fiable vers la solution optimale. Plus la taille du quartier est petite, moins il est probable qu'il s'agisse d'un minimum local.

Le réseau de relations entre toutes les particules s'appelle la « topologie en essaim ». Diverses combinaisons ont été proposées, mais les plus couramment utilisées sont :

- Topologie en anneau : Chaque particule est associée à n particules (typiquement $n = 3$), et chaque particule tend à se déplacer vers la meilleure solution dans son voisinage local.
- Topologie radiale : La communication inter-particules passe par la particule centrale, et seule la particule centrale coordonne de manière optimale sa position. Au fur et à mesure que sa position s'améliore, l'information est transmise à ses congénères.
- Topologie en étoile : Chaque particule est connectée à toutes les autres particules pour compléter le réseau social. Un optimum de voisinage est un optimum global.

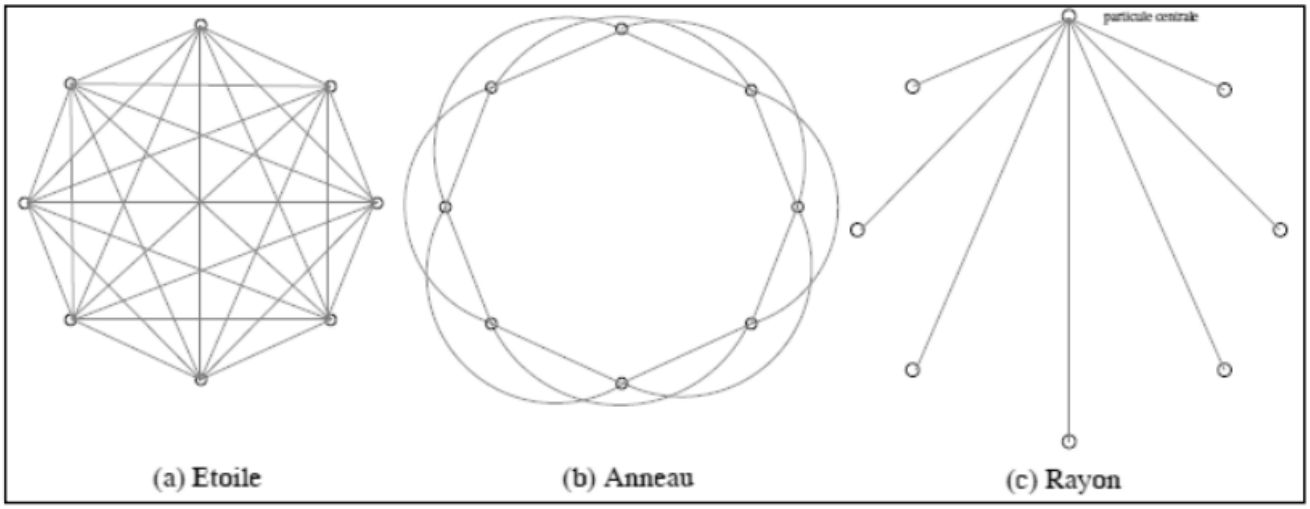

Figure 2.5: Différentes topologies pour un essaim de particules

- Nombre d'itérations :

Le nombre d'itérations pour obtenir une bonne solution dépend aussi du problème. Trop peu d'itérations peuvent entraîner l'arrêt prématuré de la recherche, et trop d'itérations introduisent une surcharge inutile (en supposant que les itérations sont la seule condition d'arrêt).

- Les coefficients d'accélération:

Affecte la valeur de déplacement maximale qu'une particule peut prendre au cours d'une itération. $\mathbf{C_1}$, $\mathbf{C_2}$ sont aussi appelés paramètres cognitifs ou sociaux. Les nombres aléatoires rand1 et rand2 sont utilisés pour influencer la nature stochastique de l'algorithme.

- Le coefficient d'inertie :

Développé par *Eberhart* et *al*, contrôle la quantité de vitesse actuelle qui doit être maintenue afin d'influencer le calcul de la nouvelle vitesse. La formule de calcul de la

vitesse de convergence est alors défini par :

$$\boldsymbol{V(t+1) = \omega\, V(t) + C_1 r_1\big(Pb(t) - X(t)\big) + C_2 r_2\big(Pg(t) - X(t)\big)} \qquad \textbf{(2.5)}$$

Le paramètre w régit donc les capacités de recherche globales et locales de l'essaim et influence le comportement de convergence de l'algorithme.

Les grands poids d'inertie ont tendance à faciliter l'exploration globale (explorer de nouveaux secteurs), tandis que les petites valeurs ont tendance à faciliter l'exploration locale (explorer le secteur actuel avec précision). Une bonne valeur de poids d'inertie équilibre généralement les capacités de recherche globales et locales, réduisant ainsi le nombre d'itérations nécessaires pour trouver la solution optimale.

- Le facteur de constriction :

Le facteur de constriction **k** a été proposé pour améliorer la convergence de l'algorithme, empêcher l'essaim d'exploser et contrôler la vitesse des particules pour éviter le problème de divergence d'essaim qui fait converger prématurément l'algorithme. La formule de vitesse est :

$$\boldsymbol{V(t+1) = k\, V(t) + C_1 r_1\big(Pb(t) - X(t)\big) + C_2 r_2\big(Pg(t) - X(t)\big)} \qquad \textbf{(2.6)}$$

Où :

$$\boldsymbol{k = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}} \qquad \textbf{(2.7)}$$

et : $\boldsymbol{\varphi = \varphi 1 + \varphi 2 \geq 4.0}$

$\boldsymbol{\varphi\ 1 = C_1\, r_1}$

$\boldsymbol{\varphi\ 2 = C_2\, r_2}$

Dans certains cas, les coefficients de constriction seuls ne peuvent pas converger vers la solution optimale pour un certain nombre d'itérations. Pour résoudre ce problème il peut être intéressant de fixer $\mathbf{V_{max} = X_{max}}$ en plus du facteur de contrainte. Cela améliorera les performances globales de l'algorithme.

2.3.3 Algorithme de base PSO

Soit **f(x)** la fonction objective à optimiser (fitness) et **N** le nombre de particules. Les étapes essentielles de l'optimisation par essaim de particules sont présentées par l'algorithme suivant :

Algorithme de base du PSO

Début

Initialisation aléatoire des positions et de vitesses de chaque particule.

Pour i = 1 à N **faire**

$$\boldsymbol{p_{best\,i} = x}$$

Fin pour

Calculer $g_{best\,i}$ selon :

$$\boldsymbol{g_{best}(k+1) = min_{\,p\,best\,i} f(p_{best}\,(k+1))} \qquad \text{avec : } \boldsymbol{1 \leq i \leq N}$$

Tant que la condition d'arrêt n'est pas satisfaite **faire**

Pour i = 1 à N **faire**

Mise à jour de la vitesse de chaque particule à l'aide l'équation :

$$\boldsymbol{vi(k+1) = \omega\, v_i(k) + c_1.r_1[p_{best\,i}(k) - x_i(k)] + c_2.r_2[g_{best\,i}(k) - x_i(k)]}$$

Mise à jour de la position de chaque particule utilisant l'équation :

$$\boldsymbol{x_i(k+1) = x_i(k) + v_i(k+1)}$$

Evaluer la fonction fitness (x_i).

Si $f(x_i) < f(p_{best\,i})$ **donc**

$$\boldsymbol{p_{best\,i} = x_i}$$

Fin si

Si $f(p_{best\,i}) < f(g_{best\,i})$ **donc**

$$\boldsymbol{g_{best\,i} = p_{best\,i}}$$

Fin si

Fin pour

Fin Tant que

Fin

2.3.4 Algorithme d'optimisation du réseau MLP par PSO

Etape 1: Déclaration des données

- nombre de nœuds la couche cachée **nc** taille du réseau MLP
- l'espace de recherche
- la taille de la population **N**
- nombre d'itération **i**
- Les coefficients **c_1 et c_2**
- Les variables stochastiques **r_1 et r_2**
- Le poids d'inertie **θ**

Etape 2: Initialisation

- Initialiser les vecteurs poids Wi de manière aléatoire
- Initialisation de la vitesse de recherche
- initialiser le best de la fonction objectif **fcb**

Etape 3: Loop

- For **i= 1 :1000** traitement itératif dans le temps
- For **t=1 : N** parcourir toutes les particules représentés par des réseaux de neurones
- for k=1 : nc parcourir tous les poids d'un RN représentant une particule
- Varier le poids dans un intervalle choisi
- Parcourir les nœuds avec les nouveaux poids
- Evaluer la performance de W_{ir} par fonction objectif **fc**
- Si **fc(i) < fcb(i)** **(**le cas où on a une **fc** à minimiser**)**
  - ✓ **$W_i = W_{ir}$**
  - ✓ **fcb(i) = fc(i)** (mise à jour du best la fonction objectif à l'itération i),
  - ✓ mise à jour des nouvelles, position et vitesse de la particule entrainée
  - ✓ mise à jour des positions des particules directement voisines (gauche et droite)
- Répétez jusqu'à atteindre un critère d'arrêt (ou fin des itérations)

2.4 Conclusion

Après étude théorique des deux outils ; 'Réseaux de neurones' et 'Essaim de particules', celle-ci nous a conduit à proposer un algorithme, utilisant la structure du réseau MLP dont l'entrainement est optimisé par l'essaim de particules, qui peut être utilisé pour l'émulation ou le contrôle des systèmes non linéaires.

Pour tester l'efficacité de ce dernier et valider notre tentative, nous allons le mettre à l'épreuve dans ce qui suit (chapitre prochain) en l'essayant : dans un premier temps pour l'émulation d'un processus supposé inconnu, défini par des entrées/sorties, puis, dans un second cas de contrôle d'un système choisi par sa forte nonlinéarité, défini par son modèle mathématique.

Chapitre 03

Simulation et interprétation

3.1 Introduction

Dans ce chapitre, nous allons mettre à l'épreuve l'algorithme proposé qui consiste à optimiser l'entrainement des réseaux de neurones MPL par essaim de particules. Cela a été planifié en deux étapes comme suit :

- Dans la première étape nous considérons que le processus dit '*Box and Jenkins'* est inconnu. Celui-ci est exprimé par 296 données d'entrées/sorties. Utilisant un réseau de neurones MLP pour représenter la structure de l'émulateur, nous essayons d'optimiser son entrainement pendant l'émulation:

- Par la méthode de rétropropagation de l'erreur de sortie
- Par la méthode d'essaim de particules

- Dans la deuxième étape nous utilisons le même réseau de neurones MLP pour représenter la structure d'un contrôleur, pour le contrôle de la température d'une réaction chimique dans un réacteur chimique ouvert CSTR complètement agité, connu par sa forte nonlinéarité. Les deux méthodes d'optimisation citées ci-dessous seront encore utilisées pour ce cas de figure.

3.2 Emulation du procédé « *Box & Jenkins* »

Le système *box & Jenkins*, est un processus dynamique représenté par 296 points présentés sous forme d'entrées/sorties du four à gaz, dont l'aire et le méthane sont combinés pour constituer un mélange de gaz contenant le $\mathbf{CO_2}$, l'entrée du four **X** correspond au débit du méthane et la sortie **Y** correspond à la concentration en pourcentage du $\mathbf{CO_2}$.

Le but de l'émulation du procédé est de comparer les performances des méthodes proposées. Pour émuler le processus "Box & Jenkins", nous avons utilisé une structure **MLP PSO** simple avec une couche d'entrée, une couche de sortie et une seule couche cachée, les entrées de notre réseau sont **X(i-4)** et **Y(i-1)** respectivement et **Y(i)** la sortie obtenue, Les poids $\mathbf{W_1(n)}$ et $\mathbf{W_2(n)}$ sont les poids reliant les couches d'entrée et cachée, et $\mathbf{W_3(n)}$ est le vecteur de poids reliant les couches cachée et de sortie.

3.2.1 Emulation du procédé par la rétropropagation BP

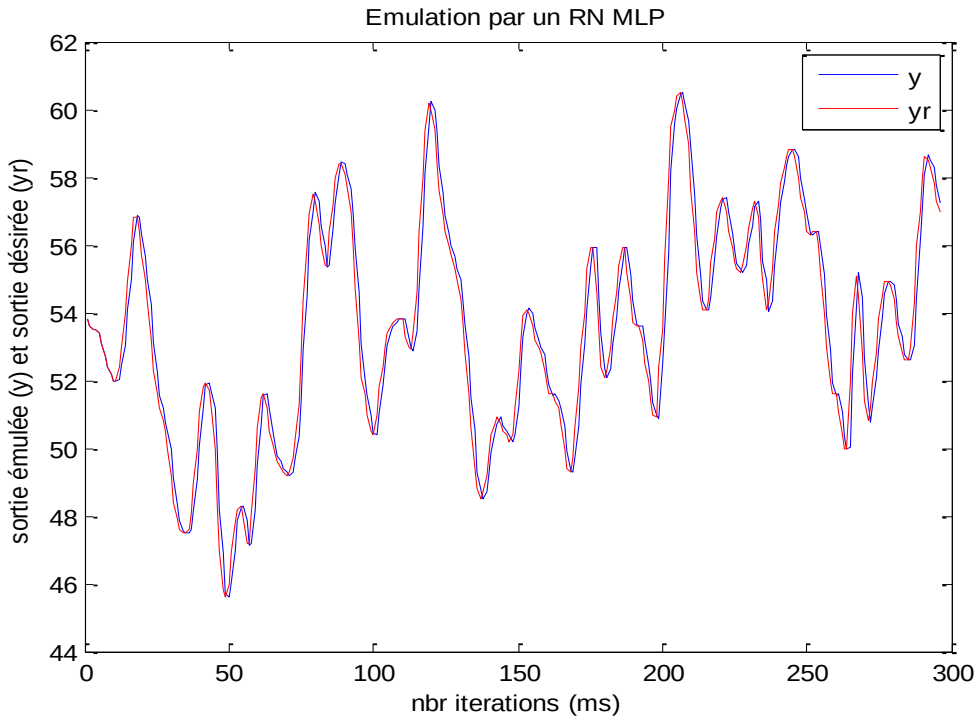

Figure 3.1: Signal de référence et signal sortie émulée par la rétropropagation

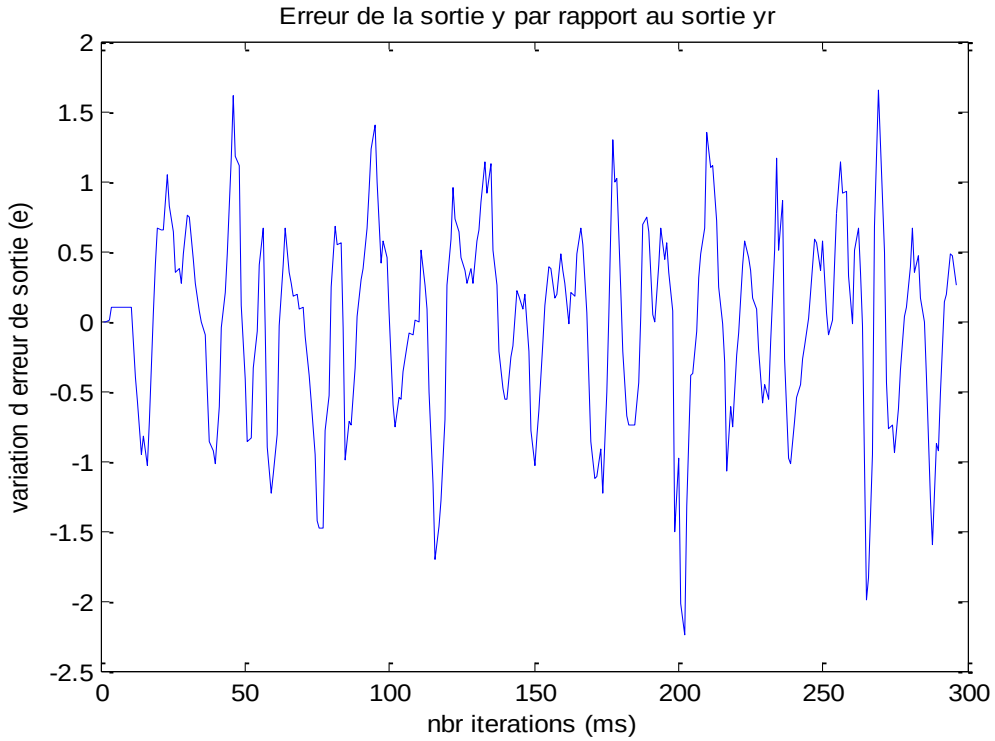

Figure 3.2: Signal d'erreur d'émulation par la rétropropagation

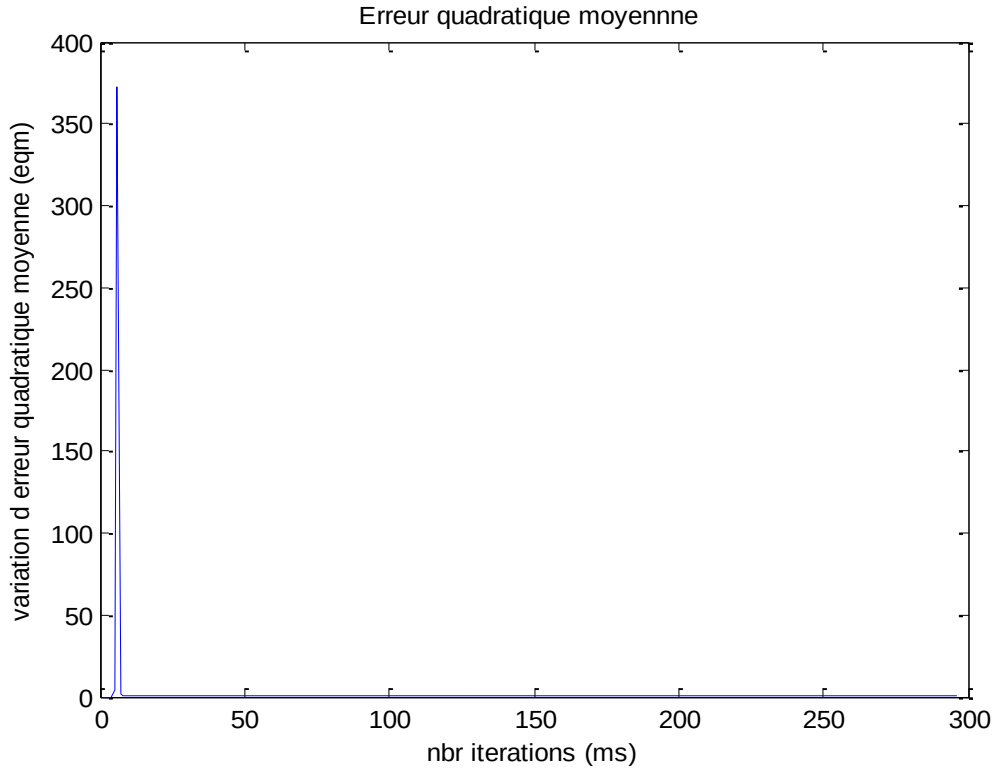

Figure 3.3 **:** Signal d'erreur quadratique moyenne d'émulation par la rétropropagation

- L'application de la rétropropagation pour l'émulation du procédé « *Box & Jenkins* », confirme l'efficacité de la méthode, le signal de sortie **Y(i)** suit convenablement le signal de référence **Yr(i).** Par conséquent l'erreur de sortie varie de -1.9 à 1.7 figure (3.1), et l'erreur quadratique moyenne est de l'ordre de 5*10^(-3).

Caractéristique	**valeur**
Structure d'émulateur	MLP 2-5-1
Méthode d'entrainement	BP
Nombre d'itérations	296
Erreur de sortie	varie de -1.9 à 1.7
Erreur quadratique moyenne	de l'ordre de 5.4*10^(-3)

Tableau 3.1: Caractéristiques de processus émulées par rétropropagation

3.2.2 Emulation du procédé Box & Jenkins par la méthode d'Essaim de particules utilisant une seule particule

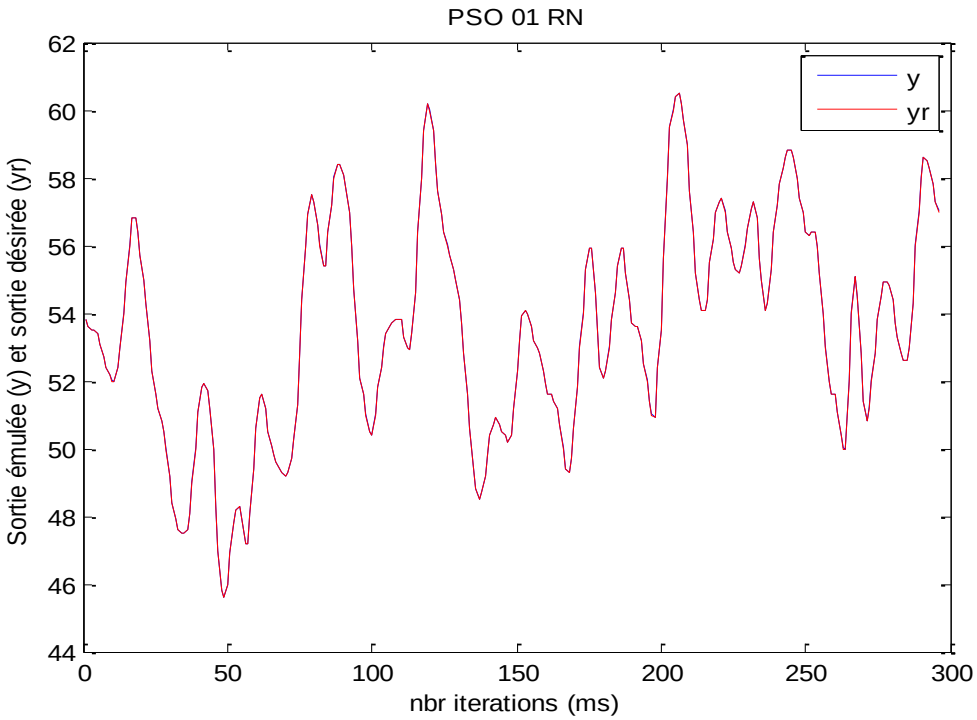

Figure 3.4 **:** Signal de référence et signal sortie émulée par une particule

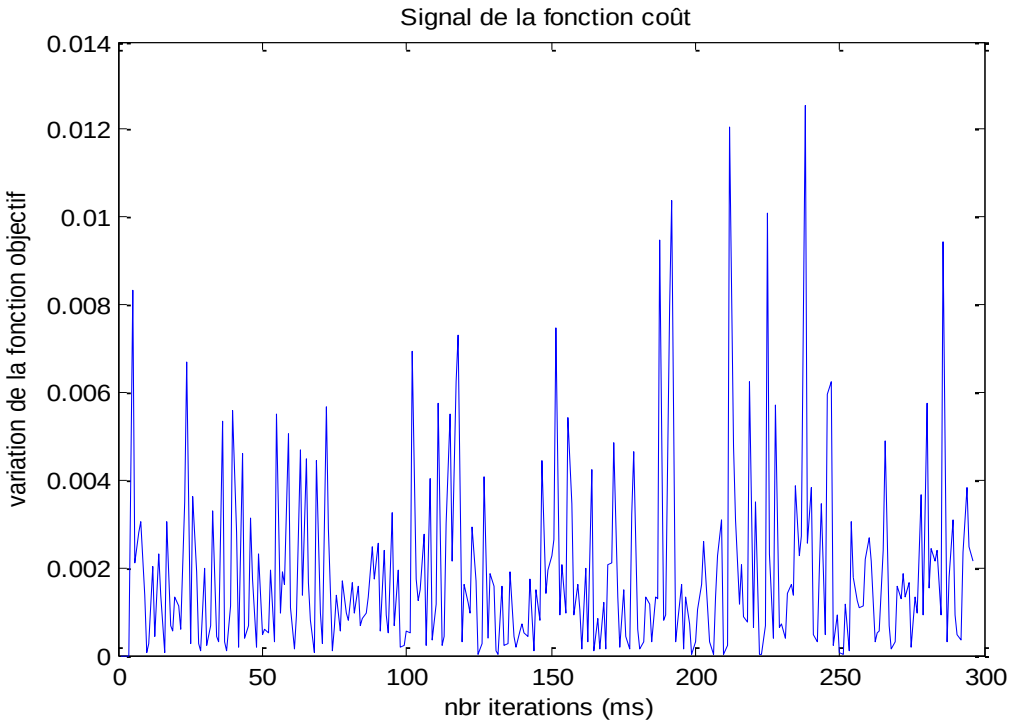

Figure 3.5: Signal de la fonction coût émulée par une particule

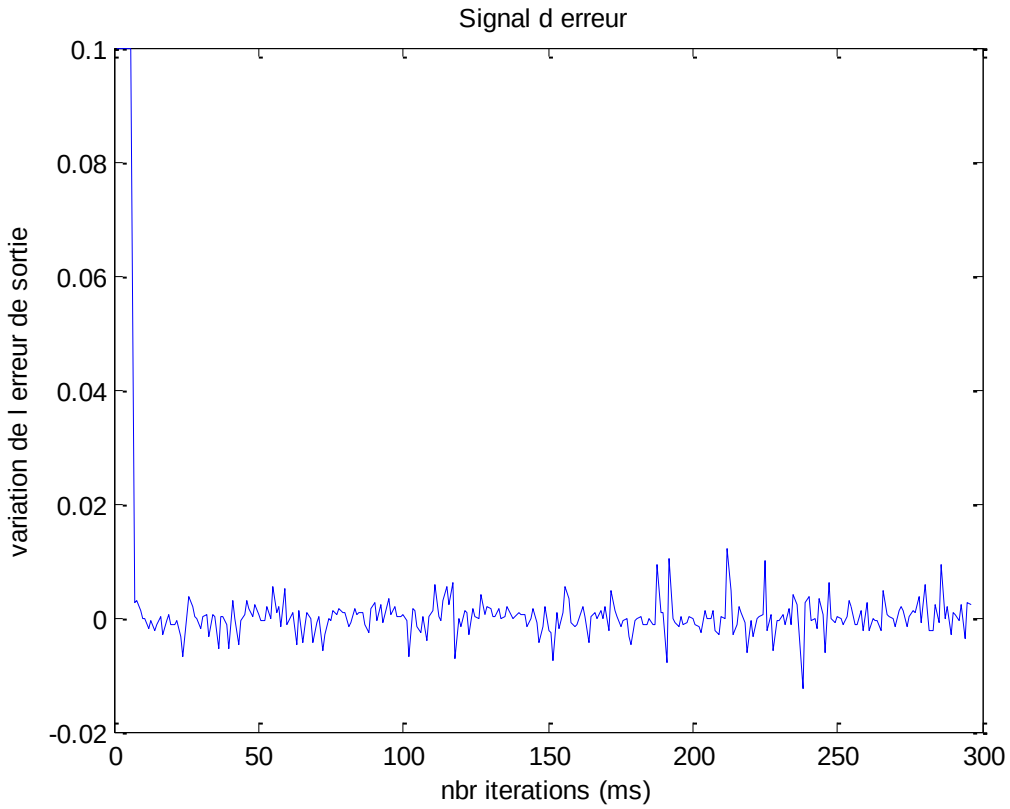

Figure 3.6 : Signal d'erreur de sortie émulée par une particule

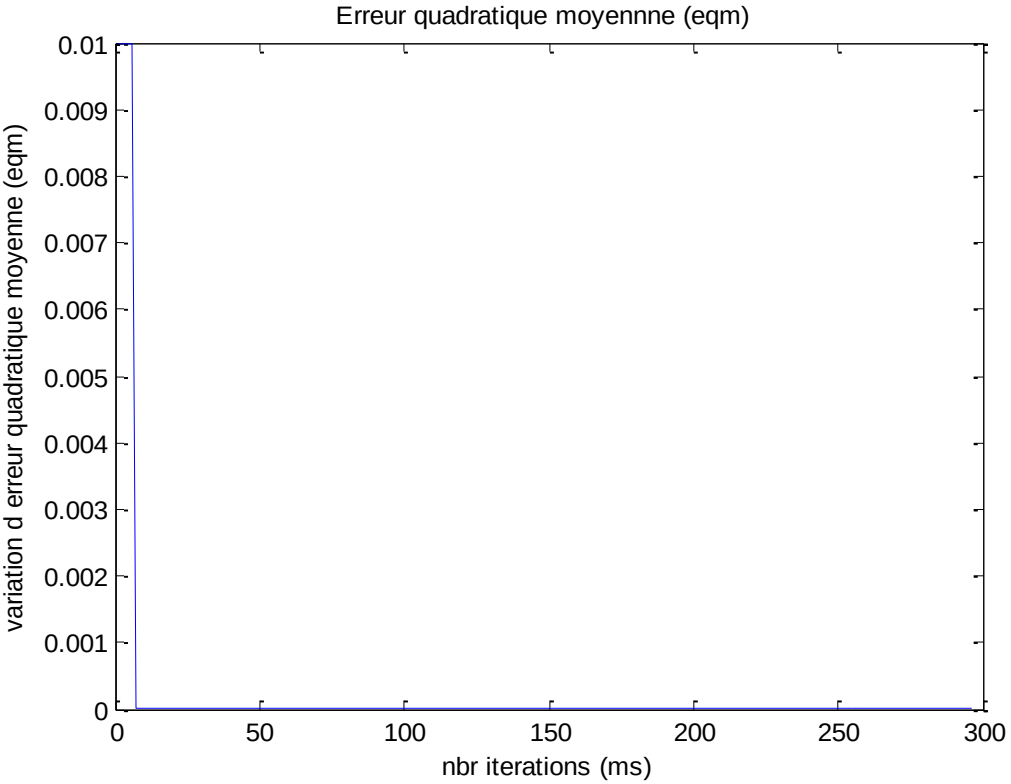

Figure 3.7 : Signal de l'erreur quadratique moyenne émulée par une particule

- D'après les résultats de l'émulation par une particule nous avons constaté que la technique de recherche de celle-ci améliore considérablement les performances comparées à celles obtenues par la BP, avec une erreur de sortie qui varie entre -0.012 et 0.0125 et la fonction objectif comprise entre 9.73*10^(-6) et 0.013.

Caractéristique	Valeur
Structure d'émulateur	MLP 2-5-1
méthode d'entrainement	PSO
Nombre d'itérations	296
Erreur de sortie	varie de -0.012 à 0.0125
Fonction objectif	varie de 9.73*10^(-6) à 0.013
Erreur quadratique moyenne	de l'ordre de 1.937*10^(-8)

Tableau 3.2 : Caractéristiques du processus émulé par un réseau MLP optimisé par une particule

3.2.3 Emulation du procédé par essaim de particules utilisant 0 5 particules

Chaque réseau MLP est considéré comme une particule de l'essaim, chaque particule va chercher sa meilleure performance, elle est influencée par celles de ses voisines directes, et est informée continuellement de la meilleure atteinte par tout le groupe, d'où la collaboration de toutes les particules pour la recherche de l'optimum dans tout l'espace de recherche prédéfini.

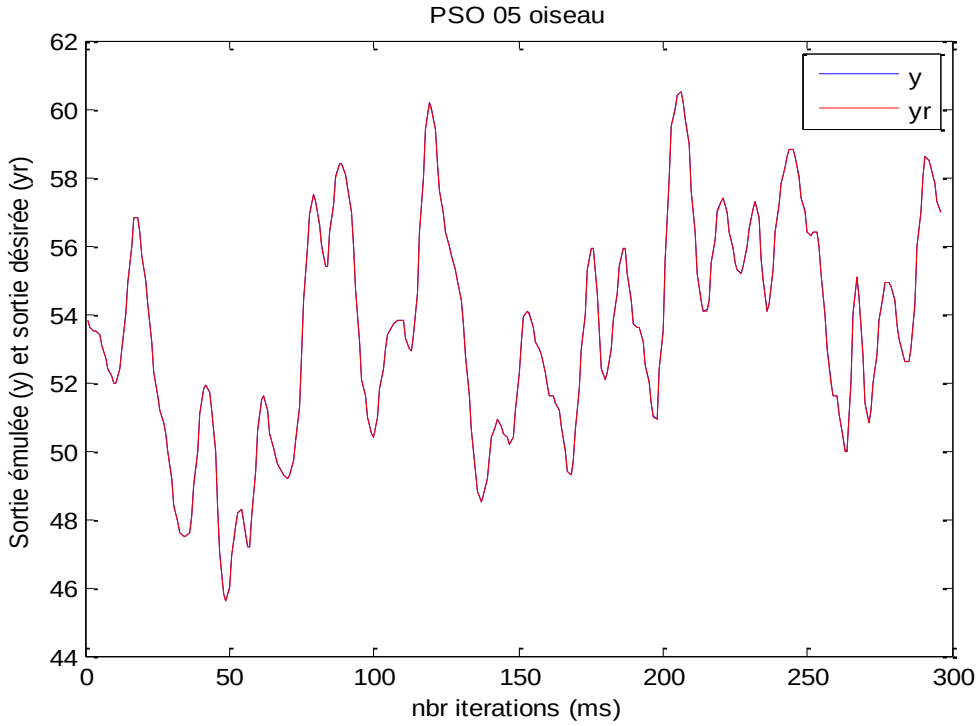

Figure 3.8 : Signal de référence et signal de sortie émulé par essaim de cinq particules

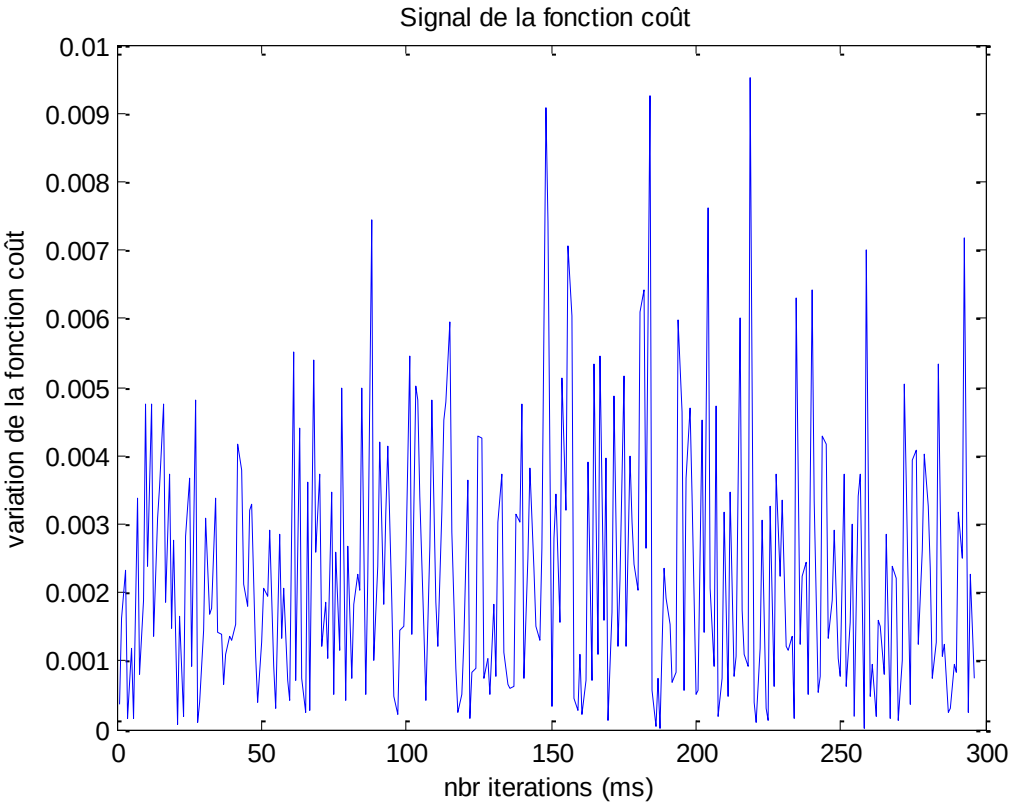

Figure 3.9: Signal de la fonction objectif émulée par l'essaim du cinq particules

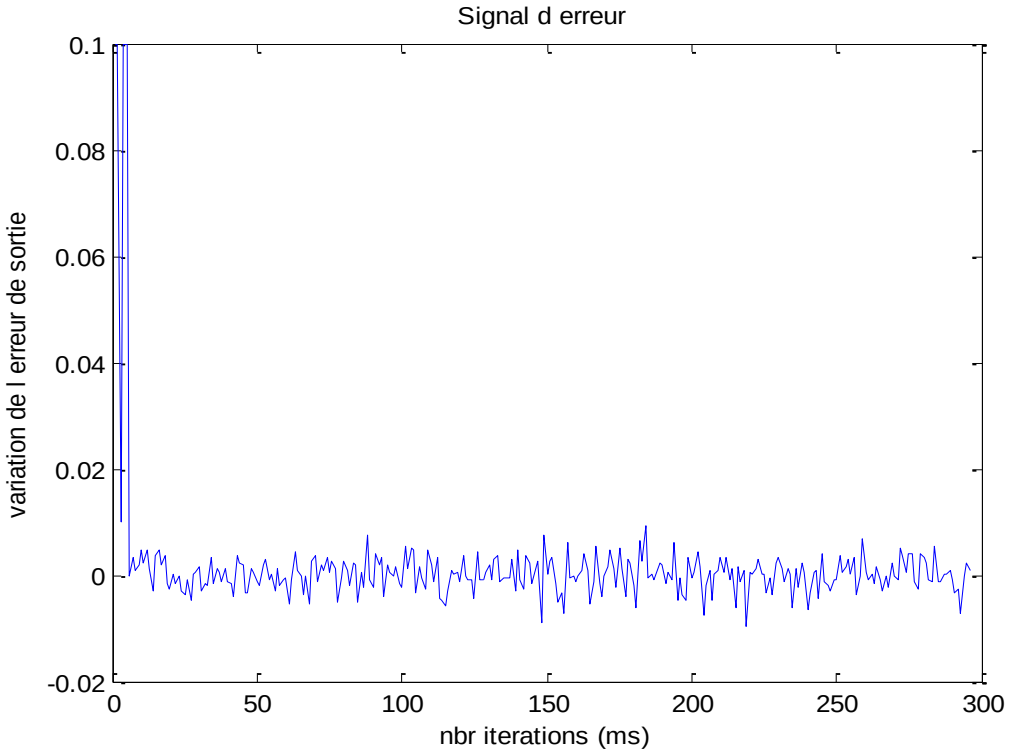

Figure 3.10 : Signal de l'erreur de sortie émulée par essaim de cinq particules

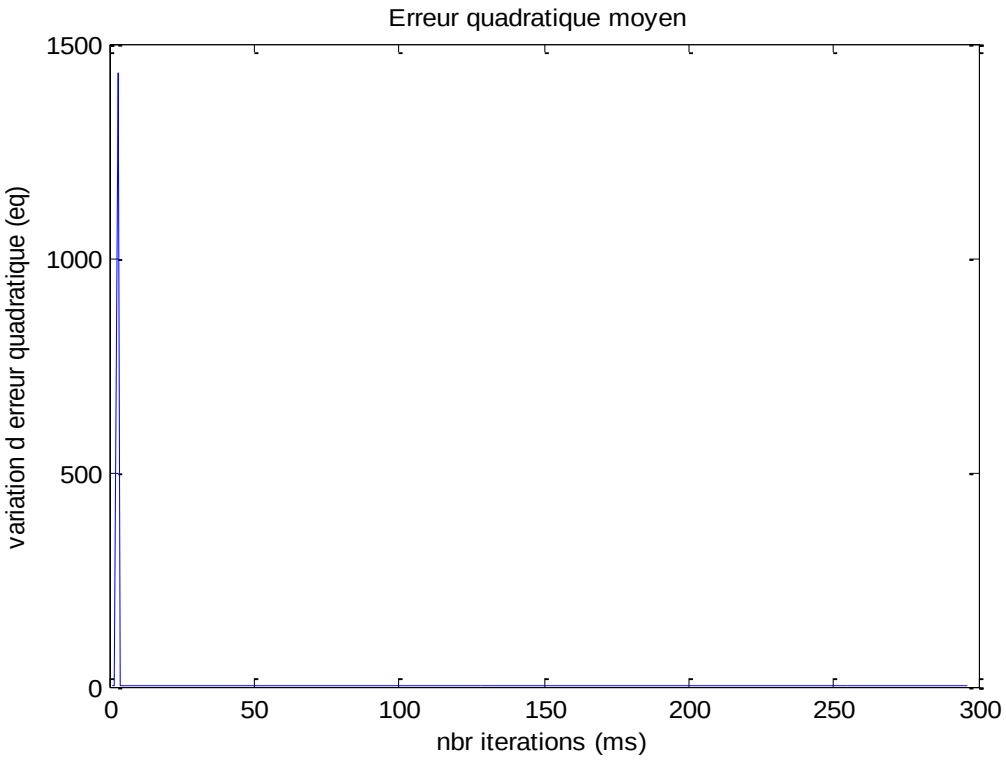

Figure 3.11: Signal de l'erreur quadratique moyenne émulée par essaim du cinq particules

- Les résultats de l'émulation du procédé « *Box & Jenkins* » utilisant un essaim de cinq particules s'avèrent très performants. Le signal émulé suit parfaitement le signal de référence. Les cinq particules de l'essaim ont coopéré pour atteindre ces performances affichées. Par conséquent la précision est considérablement améliorée, avec une erreur très faible de l'ordre de 0.01759 et la fonction objectif qui converge rapidement vers la valeur minimale de 9.68^(-6).

Caractéristique	**valeur**
Structure d'émulateur	MLP 2-5-1
Méthode d'entrainement	PSO
Nombre d'itérations	296
Erreur de sortie	de -0.01868 à 0.01759
Fonction objectif	varie entre 9.68*10^(-6) à 0.0098
Erreur quadratique moyenne	de l'ordre de 2.478*10^(-8)

Tableau 3.3 : Caractéristiques du procédé de la commande par réseau MLP utilisant PSO

3.3 Commande de la température d'une réaction chimique « CSTR » par un réseau MLP optimisé par PSO

3.3.1. Définition d'un réacteur chimique

Un réacteur à cuve agitée est un réacteur discontinu équipé d'une turbine ou d'un autre dispositif de mélange pour assurer un mélange efficace. En génie chimique, le nom CSTR est souvent utilisé pour le réacteur à cuve agitée idéal utilisé pour modéliser les variables manipulées nécessaires pour atteindre une certaine performance.

3.3.2. Présentation de processus

Un réacteur chimique continu parfaitement agité est le siège d'une réaction exothermique **A → B** d'ordre 1, qui fournit une chaleur de réaction**ΔH**. Le réactif **A** est alimenté en continu dans le réacteur avec un débit volumétrique **q**, et une concentration molaire d'alimentation $\mathbf{CA_0}$ et une température $\mathbf{T_0}$. Le contenu du réacteur est mélangé avec un agitateur motorisé. Un courant de sortie, qui contient un mélange des deux composés (le réactif **A** et le produit **B**), est retiré du réacteur en continu avec un débit volumique q, une composition **CA** et **CB** et une température **T**. Pour soutirer la chaleur générée par cette réaction exothermique, le réacteur est entouré d'une double enveloppe (indicée '**j**') avec une surface d'échange **A** et de volume constant **V** traversée par un fluide de refroidissement (eau) avec un débit $\mathbf{q_j}$ variable et une température d'entrée $\mathbf{TC_0}$ constante. Les paramètres du réacteur et les conditions opératoires utilisées dans notre étude sont ceux utilisé par Luyben. **[16]**

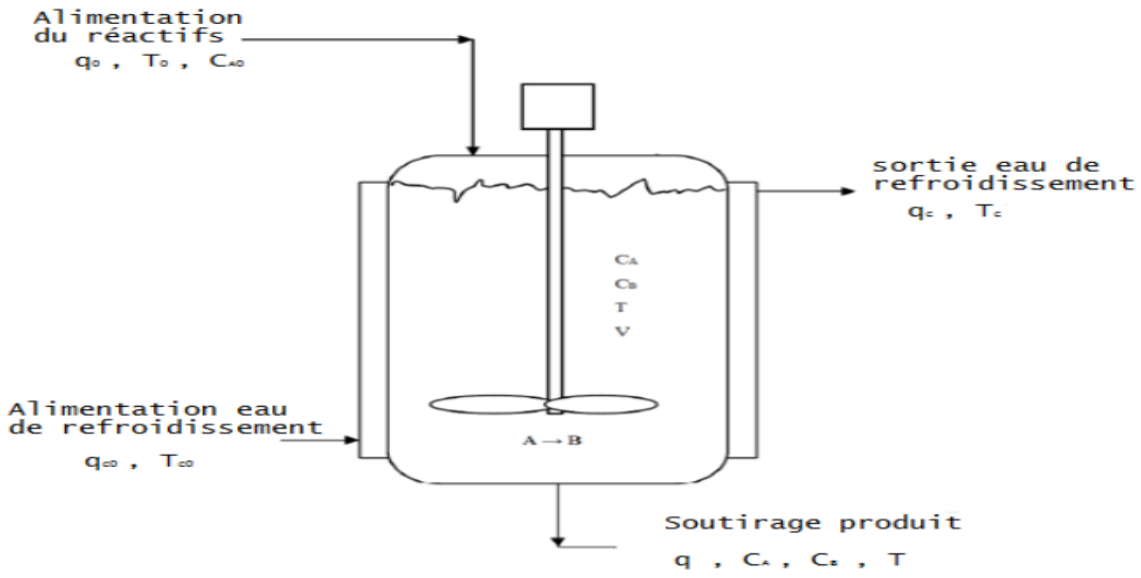

Figure 3.12 : Schéma d'un réacteur chimique avec double enveloppe

3.3.3. Modèle mathématique du CSTR

Le CSTR comporte un modèle non linéaire à trois équations différentielles mettant en œuvre la variation de cinq paramètres en fonction du temps ($\mathbf{C_A}$ concentration du réactif **A**, **T** température du réacteur, $\mathbf{T_j}$ température de la double enveloppe, **V** volume du mélange réactionnel).

Le modèle mathématique de ce processus est formulé en introduisant les équations constitutives appropriées pour établir le bilan masse-énergie.

$$\frac{dC_A}{dt} = \frac{q}{V}(C_{A0} - C_A) - K_0 C_A exp\left(\frac{E}{RT}\right) \qquad (3.1)$$

$$\frac{dT}{dt} = \frac{q}{V}(T_0 - T) - \left(\frac{\Delta H}{\rho C_p}\right) K_0 C_A exp\left(\frac{-E}{RT}\right) + \left(\frac{\rho_c C_{pc}}{C_p V}\right) q_c \left[1 - exp\left(\frac{-hA}{q_c \rho_c C_{pc}}\right)\right](T_{c0} - T) \qquad (3.2)$$

Où :

ΔH : chaleur de la réaction.

(-hA) : coefficient de transfert thermique.

$\mathbf{T_0}$: température d'alimentation.

$\mathbf{T_{c0}}$: température d'entré de liquide réfrigérant.

$\mathbf{C_A}$: produit (effluent) concentration de composant **A** dans le réacteur.

3.3.4 Résultat de la commande du processus

Les figures suivantes présentent les résultats de simulation du processus CSTR avec BP-MLP et MLP-PSO.

a. Contrôle du processus par la rétropropagation

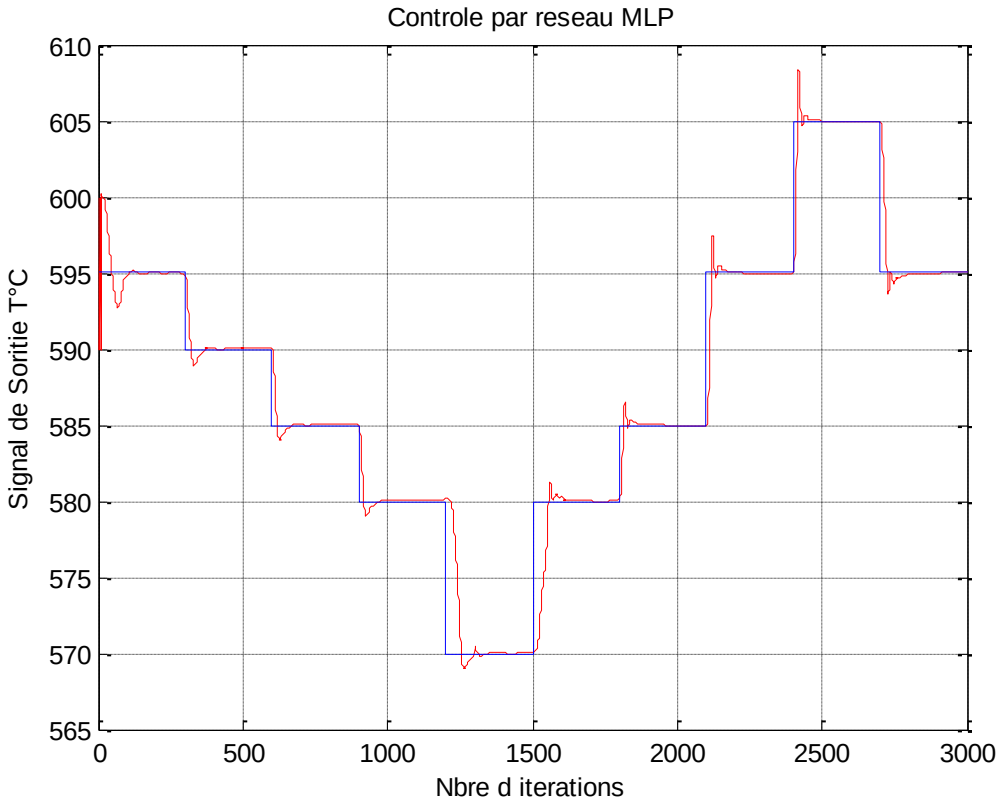

Figure 3.13 : Signal de sortie T et le signal de référence Tr

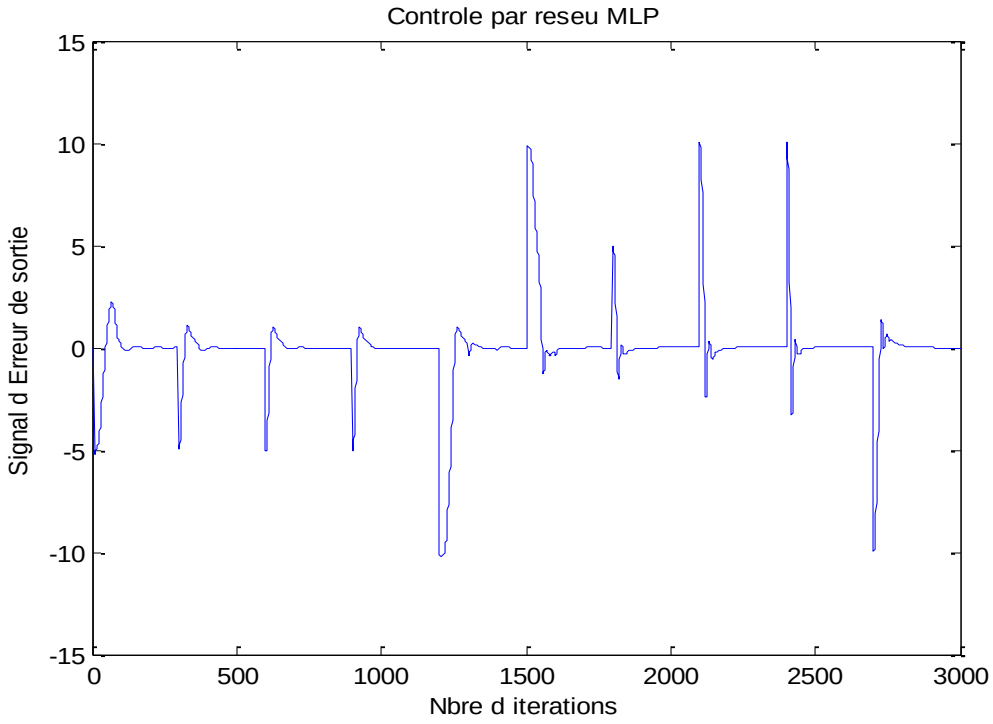

Figure 3.14 : Signal de l'erreur de sortie

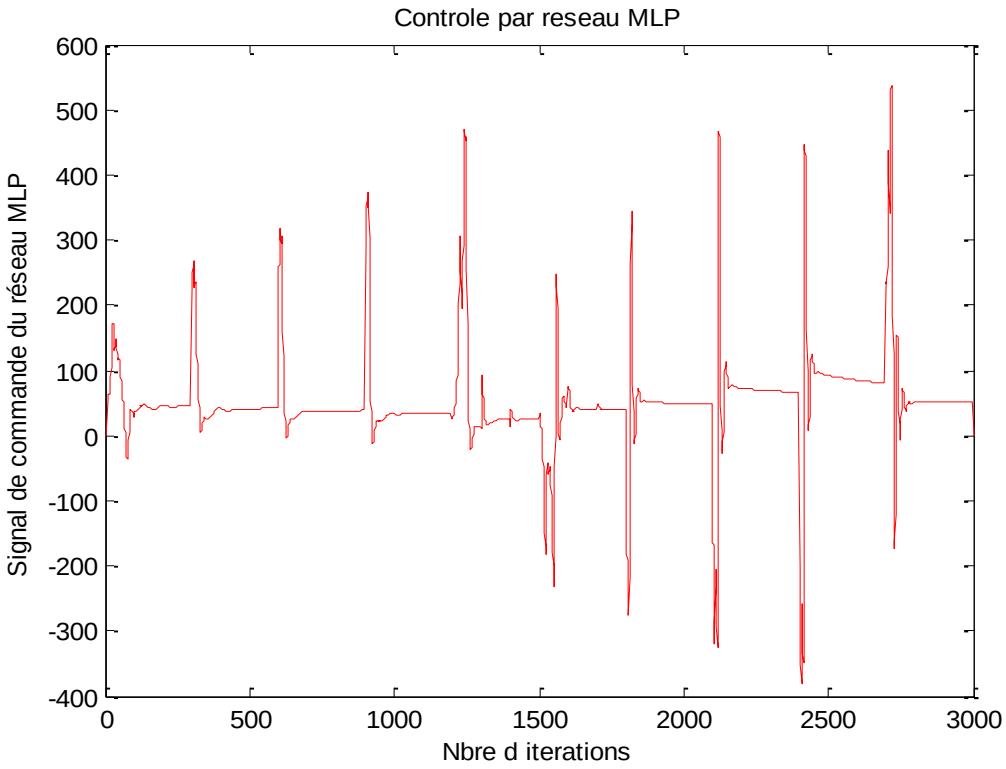

Figure 3.15 : Signal de la commande

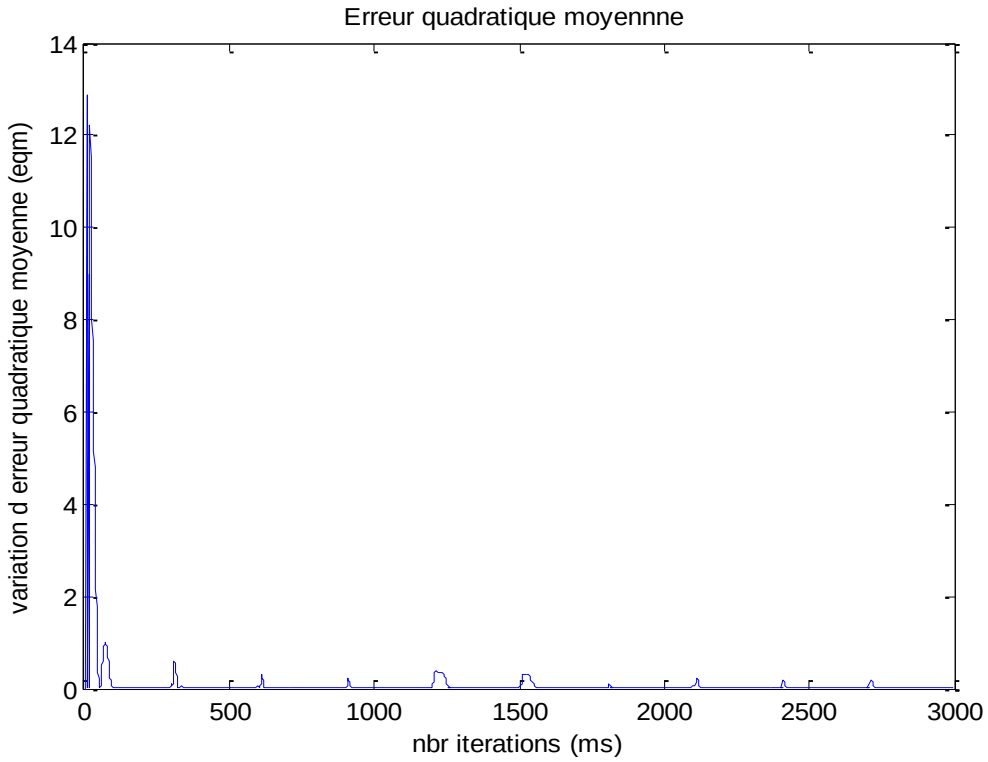

Figure 3.16 : Signal de l'erreur quadratique moyenne

- Le contrôle de la température d'une réaction par la rétropropagation, permet au signal de sortie de suivre le signal de référence choisi avec succès ainsi que l'apparition des oscillations pendant les commutations entre les différents points de fonctionnement (paliers de Tr), et l'erreur de sortie varie entre -10.19 et 9.88.

Caractéristique	valeur
Structure de contrôleur	MLP 1-5-1
Méthode d'entrainement	BP
Nombre d'itérations	3000
Erreur de sortie pendant les commutations	Varie de -10.19 à 9.88
Erreur de sortie autour des points de fonctionnement	de l'ordre de 0.25
L'erreur quadratique moyenne	de l'ordre de 0.9824

Tableau 3.4 : Caractéristiques de processus émulées par rétropropagation

b. Contrôle de la température avec MLPPSO utilisant cinq particules

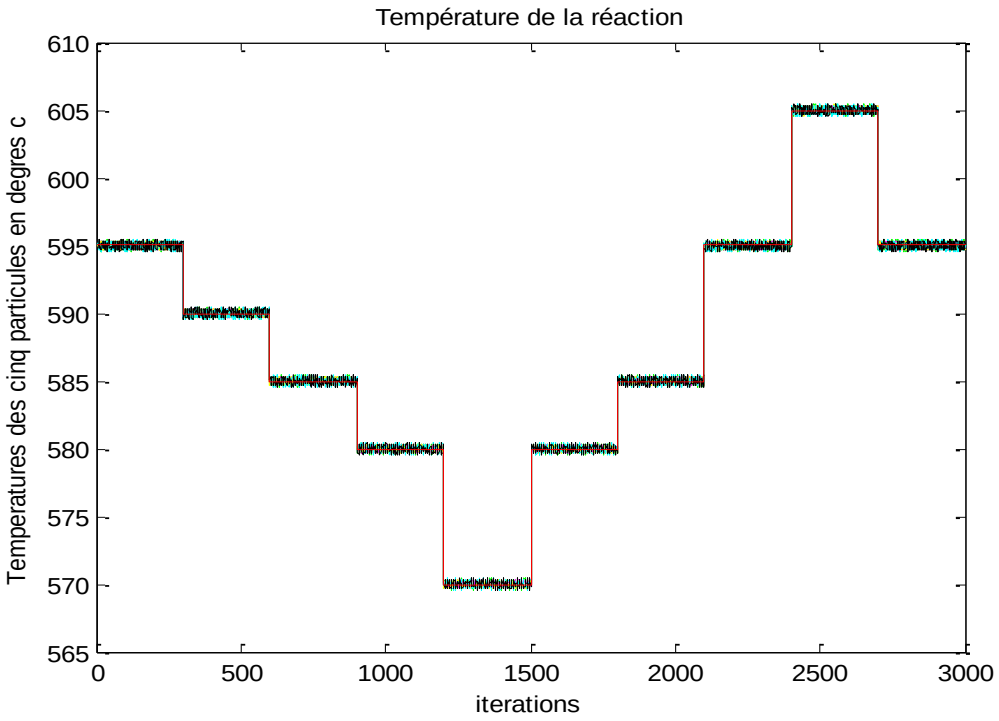

Figure 3.17: Signaux de températures des cinq particules avec la température de référence

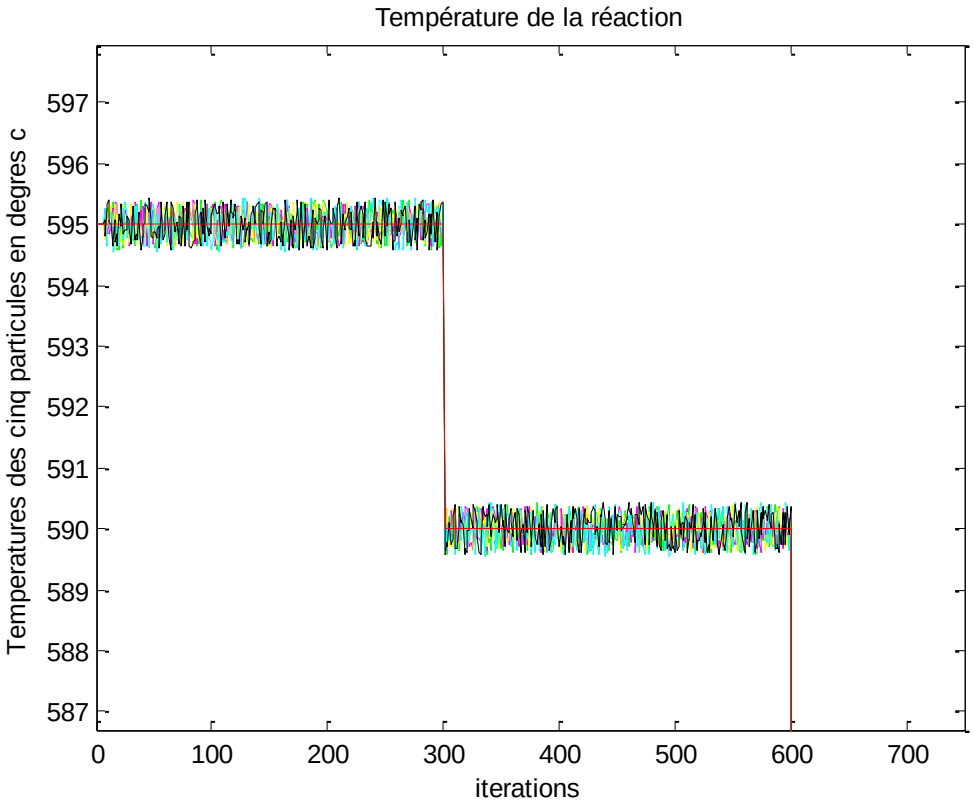

Figure 3.18 : Signaux de températures zoomés avec la température de référence

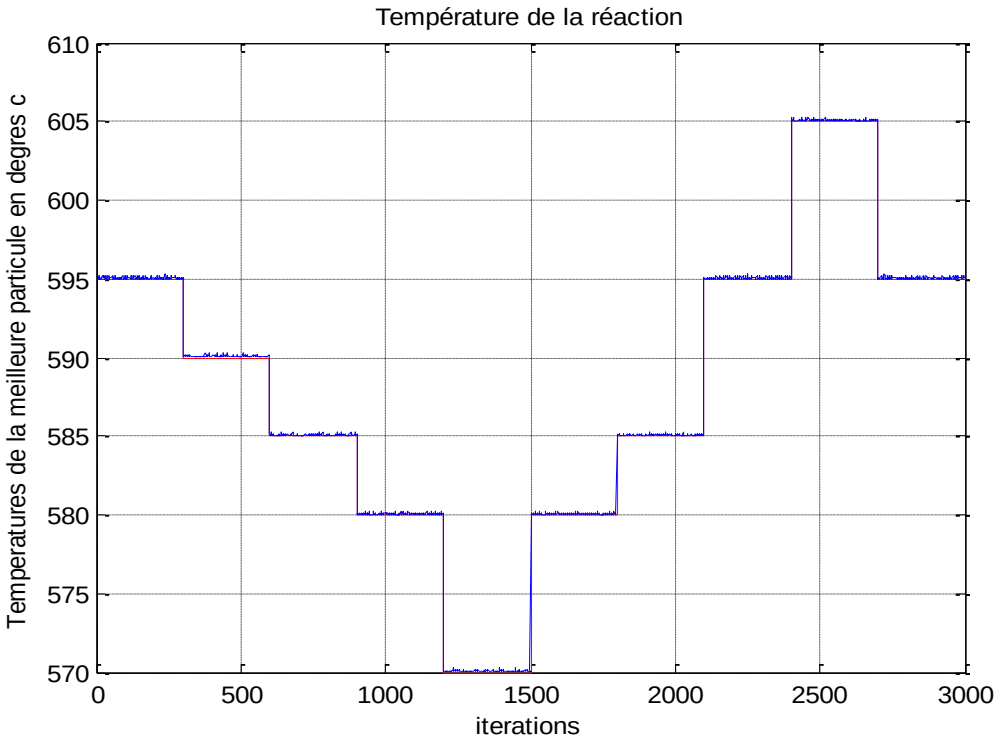

Figure 3.19 : Signal de la meilleure sortie T des cinq particules et le signal de référence

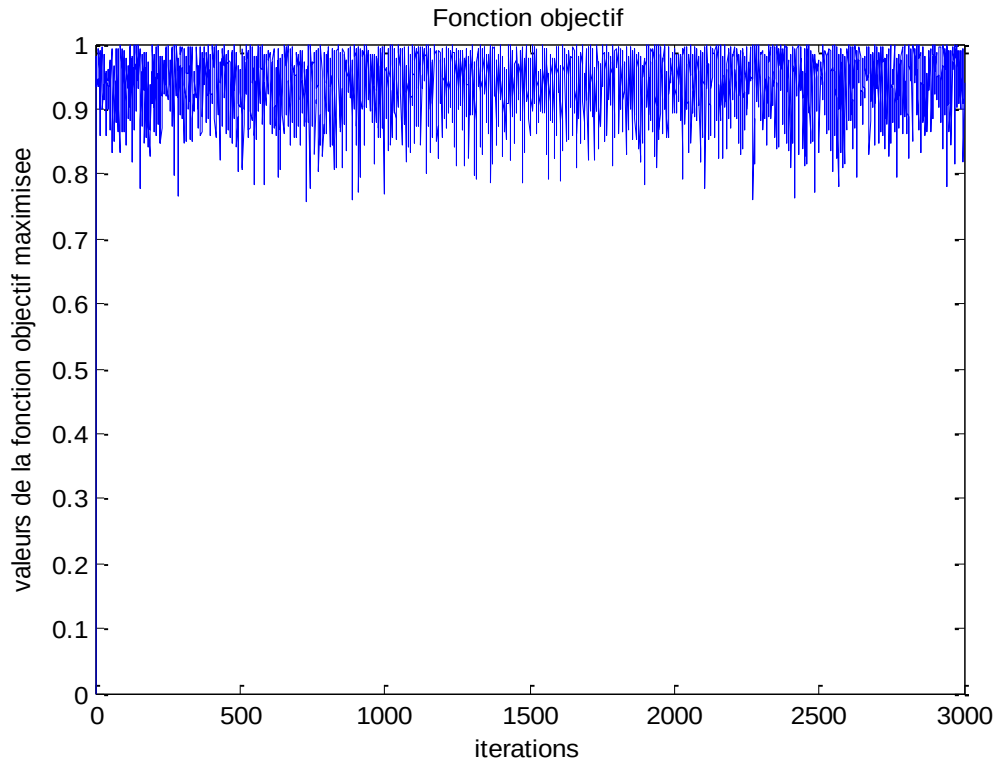

Figure 3.20 : Signal de la fonction objectif maximisée

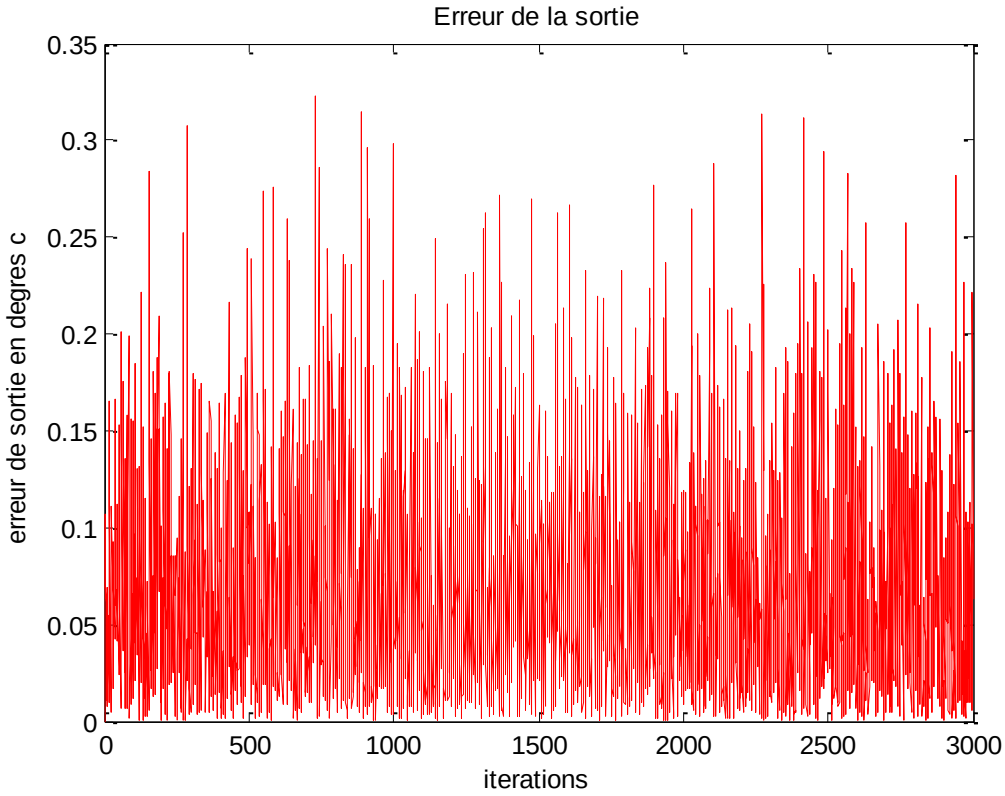

Figure 3.21: Signal de l'erreur de sortie

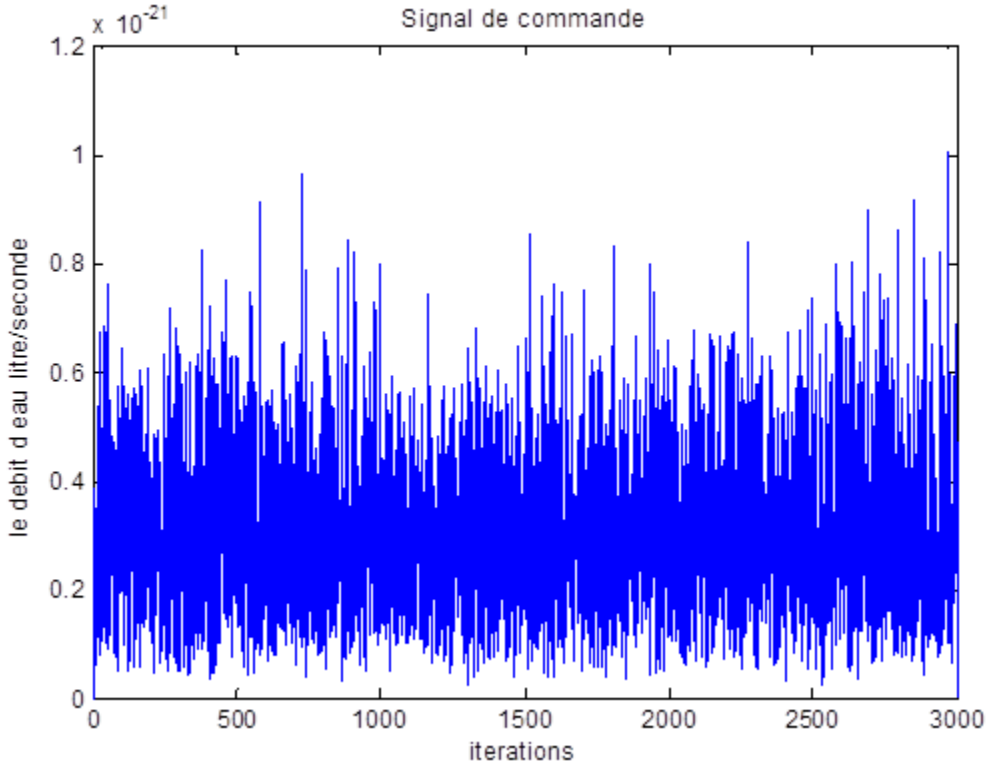

Figure 3.22 : Signal du débit q de la commande

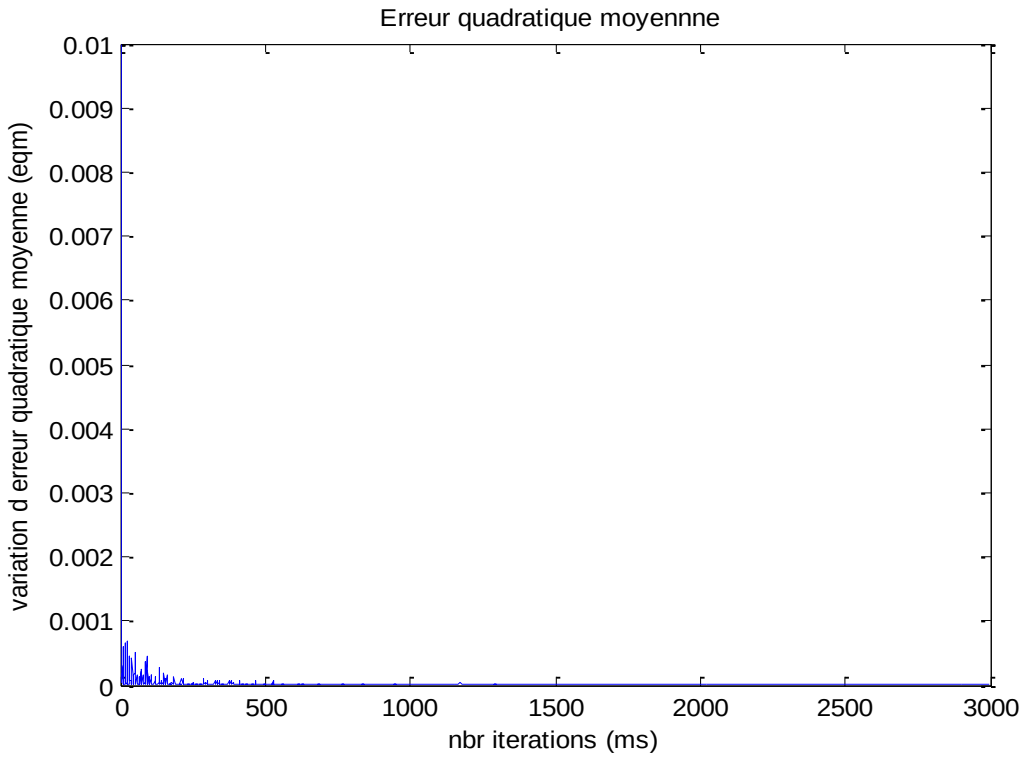

Figure 3.23 : Signal de l'erreur quadratique moyenne

- Les résultats obtenues pendant le contrôle de température de la réaction chimique dans un CSTR par réseaux de neurones optimisés par un Essaim de cinq particules montre la contribution de ces cinq particules pour atteindre des performances considérables, les signaux de sortie et de référence sont très proches et les commutations entres les différents points de fonctionnement sont quasi-parfaits.

Caractéristique	**valeur**
Structure de contrôleur	MLP 1-5-1
Méthode d'entrainement	PSO
Nombre d'itération	3000
Erreur de sortie	varie de 0.00026 à 0.31
Fonction objectif maximisée	varie entre 0.75 à 1
L'erreur quadratique moyenne	de l'ordre de 4.4*10^(-4)

Tableau 3.5: Caractéristiques du procédé de la commande par réseau MLP utilisant PSO

3.4 Conclusion

Dans ce chapitre nous avons mis notre algorithme à l'épreuve. Nous avons essayé d'émuler un processus exprimé par ses entrées/sorties, et de contrôler un second système représenté par son modèle mathématique fortement nonlinéaire.

Dans les deux cas, nous avons entrainé la structure MLP choisie par :

- La rétropropagation de l'erreur de sortie, que nous considérons comme une méthode relativement classique de référence, connue par ses performances dans divers domaines.
- La PSO qui s'articule sur la collaboration d'un ensemble de particules pour la recherche d'optimum dans un espace de recherche.

Les résultats de simulations ont prouvé que la technique proposée PSOMLP s'est avérée facile à implémenter et plus efficace que celle de référence comparée (BPMLP). Les résultats sont assez encourageants prometteurs.

Conclusion générale

Notre travail consiste à proposer une technique qui est à base d'outils d'intelligence artificielle pour l'émulation et le contrôle de processus complexes, l'objectif principal est d'améliorer les performances déjà atteintes par certaines méthodes connues dans la littérature.

Notre étude est portée sur l'hybridation des réseaux de neurone MLP avec l'une des métaheuristiques dite PSO connue par sa résolution des problèmes d'optimisations difficiles, qui comporte un nombre fini de solutions sous-optimales,

L'algorithme proposé PSOMLP est relativement simple à implémenter, pourtant il associe entre les spécifications de PSO, d'optimisation rapide et efficace et efficace et celles des réseaux MLP, d'approximation de n'importe quelle fonction complexe, ce qui a conduit lors des tests à des résultats extrêmement satisfaisantes. Depuis l'émulation du processus *'Box & Jenkins'* par une seule particule, ces résultats se sont encore améliorés par l'essaim de cinq particules. Lors du contrôle de la température d'un réacteur chimique dans un réacteur ouvert complètement agité CSTR, le contrôleur à base de la méthode PSOMLP a affiché des résultats similaires, très encourageants. Ce qui confirme l'efficacité et par conséquent la validation de la technique proposée.

Après le succès clairement affiché par l'optimisation de notre technique, nous pouvons viser à compléter notre étude en vérifiant l'influence de certains paramètres sur les performances de la recherche d'optimum, en l'occurrence, la topologie formée par les particules qui compose l'essaim ou la manière avec laquelle les particules communiquent entre eux dans un essaim (anneau, complètement connecte ...). Ou étudier ultérieurement d'autres hybridations d'outils récemment utilisé, qualifiés de performants, ayant un lien avec le contrôle des systèmes, l'intelligence artificielle, le *deep learning*, *machine learning.*

Bibliographie

[1] : **A. Chamekh**, "Optimisation des procédés de mise en forme par les réseaux de neurones artificiels", Thèse de doctorat en science, Université d'Angers, 2007. Français.

[2] : **M. Parizeau**, "RESEAUX DE NEURONES",Université LAVAL,2006, GIF-21140 et GIF-64326.

[3] : **Y. Djeriri**,"Les Réseaux de Neurones Artificiels",Université de Sidi Bel Abbès, 2017.

[4] : **K. HORNIK, M. STINCHCOMBE, H. WHITE**, "Multilayer Feedforward Networks are Universal Approximators", Neural Networks Vol. 2, pp. 359-366, 1989,

[5] : **K. HORNIK, M. STINCHCOMBE, H. WHITE, P. AUER,** "Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives", Neural Computation Vol. 6, pp. 1262-1275, 1994.

[6] : **H. Mezaache**, "Les réseaux de Neurones formels Et Les systèmes Neuro-Flous Pour l'apprentissage par renforcement", Université de Batna 2, 2008.

[7] : **Camille Besse Patin**, "Planification, Ordonnancement et Apprentissage par renforcement ", D.A.M.A.S. U. Laval, 2005.

[8] : **D. Filliat**, "Robotique Mobile", 2004.

[9] : **M. Benmeddour**,"Contrôle par réseaux de neurones", mémoire de master, Université d'Oum-El- Bouaghi,2012.

[10] : **G. DREYFUS**, "LES RÉSEAUX DE NEURONES", École Supérieure de Physique et de Chimie Industrielles de la Ville de Paris (ESPCI), Laboratoire d'Électronique 10, rue Vauquelin 75005 PARIS,2002.

[11] : **A. Omara, M. Zellouf,** "Ordonnancements coopératifs pour les chaînes logistiques", Thèse de master, Université M'Sila,2020.

[12] : **I. LARIBI,** "Résolution de problèmes d'ordonnancement de type Flow-Shop de permutation en présence de contraintes de ressources non-renouvelables ",Thèse de doctorat en LMD, Université de Tlemcen, 2018

[13] : **W. LABBI, M. BOUDHAR,** "Métaheuristiques pour un problème d'ordonnancement sous contraintes de préparation", mémoire de master, Université de Tlemcen, 2021

[14] **: Z. Deffas ,** " Métaheuristiques parallèles à solution unique pour la résolution du problème du Q3AP sur grille de calcul ", Mémoire de Magister en informatique option intelligence artificielle, université de Oum El-Bouaghi ,2010, p13.

[15] **:A. KARRAY,** "Contribution à l'ordonnancement d'ateliers agroalimentaires utilisant des méthodes d'optimisation hybrides", Ecole Centrale de Lille; École nationale d'ingénieurs de Tunis (Tunisie), Thèse de doctorat en science ,2011.

[16] **: Gourlia, Jean-Paul**, "Modélisation en génie des procédés, Techniques de l'ingénieur, Génie des procédés 1", 1995.

Résumé

La résolution de problèmes d'optimisation difficile est un domaine de recherche actif en informatique et en mathématiques appliquées. Les problèmes d'optimisation difficile sont des problèmes pour lesquels il est difficile voire impossible de trouver une solution exactes par les méthodes classiques.

Les techniques de résolution de problèmes d'optimisation difficile incluent l'utilisation des heuristiques qui cherchent à trouver des solutions de bonne qualité pour des problemes difficiles. Ces méthodes sont souvent basées sur des algorithmes d'optimisation à base de métaheuristiques.

Dans ce travail, nous avons choisi un reseau de neurones MLP pour l'utiliser comme structure de base. Concernant l'optimisation de l'entrainement de celui ci nous avons opte pour une méthode relativement classique utilisant la BP de l'erreur de sortie du réseau, et une méthode basée sur la PSO sur laquelle se base l'algorithme d'optimisation que nous proposons.

Ces deux méthodes utilisant la meme srtucture MLP sont mises a l'épreuve pour L'émulation du procédé dit "Box & Jenkins" et le controle de la température d'une réaction chimique dans CSTR .

Mots clés: problèmes d'optimisation difficile, heuristique, algorithms, métaheuristiques, reseau de neurons, MLP, entrainement, BP, température , CSTR

Abstract

Solving difficult optimization problems is an active area of research in computer science and applied mathematics. Hard optimization problems are problems for which it is difficult or even impossible to find an exact solution by classical methods.

Techniques for solving difficult optimization problems include the use of heuristics that seek to find good quality solutions for difficult problems. These methods are often based on optimization algorithms based on metaheuristics.

In this work, we have chosen an MLP neural network to use as a basic structure. Regarding the optimization of the training of this one, we have opted for a relatively classic method using the Bp of the network output error, and a method based on the PSO on which the optimization algorithm that we are based is based. propose.

These two methods using the same MLP structure are put to the test for the emulation of the so-called "Box & Jenkins" process and the control of the temperature of a chemical reaction in CSTR.

Keywords: hard optimization problems, heuristics, algorithms, metaheuristics, neural network, MLP, training, BP, temperature, CSTR

ملخص

يعد حل مشكلات التحسين الصعبة مجالًا نشطًا للبحث في علوم الكمبيوتر والرياضيات التطبيقية. مشاكل التحسين الصعبة هي المشاكل التي يصعب أو حتى من المستحيل إيجاد حل دقيق لها بالطرق التقليدية.

تتضمن تقنيات حل مشكلات التحسين الصعبة استخدام الأساليب التجريبية التي تسعى إلى إيجاد حلول جيدة الجودة للمشكلات الصعبة. غالبًا ما تستند هذه الطرق إلى خوارزميات التحسين بناءً على التحليل الفوقي.

في هذا العمل ، اخترنا شبكة عصبية MLP لاستخدامها كهيكل أساسي. فيما يتعلق بتحسين تدريب هذا ، اخترنا طريقة كلاسيكية نسبيًا باستخدام Bp لخطأ خرج الشبكة ، وطريقة تعتمد على PSO التي تستند إليها خوارزمية التحسين التي نعتمد عليها.

يتم اختبار هاتين الطريقتين باستخدام نفس بنية MLP لمحاكاة ما يسمى بعملية "Box & Jenkins" والتحكم في درجة حرارة تفاعل كيميائي في CSTR.

الكلمات الرئيسية: مشاكل التحسين الصعبة ، الاستدلال ، الخوارزميات ، metaheuristics ، الشبكة العصبية ، MLP ، التدريب ، BP ، درجة الحرارة ، CSTR