N° d'ordre :

UNIVERSITÉ * MOHAMED BOUDIAF * DE M'SILA

FACULTÉ DES SCIENCES ET SCIENCES DE L'INGENIEUR

DÉPARTEMENT D'ÉLECTRONIQUE

MÉMOIRE

Présenté pour l'obtention du diplôme de :

MAGISTER EN GÉNIE ÉLECTRONIQUE

Spécialité : Génie électronique

Option : Contrôle

Par

Abdelaziz AOUICHE

SUJET

REJECTION DES PERTURBATIONS DANS LES SYSTÈMES NON LINÉAIRES : ÉTUDE COMPARATIVE

Soutenue publiquement Devant le jury composé de :

Dr. Mohamed BOUAMAR,	Maître de conférence,	Université de M'sila,	Président
Dr. Farid BOUTTOUT,	Maître de conférence,	Université de M'sila,	Rapporteur
Pr. Djamel CHIKOUCHE,	Professeur,	Université de Sétif,	Examinateur
Pr. Djamel BENATIA,	Professeur,	Université de Batna,	Examinateur
M. Kheiredine CHAFAA,	Chargé de cours,	Université de M'sila	invité

2005-2006

remerciement

Je rends ma profonde gratitude à dieu tout puissant qui m’a aidé à réaliser ce modeste travail.

J’exprime ma profonde gratitude à mes parents pour leurs encouragements, leur soutien et pour les sacrifices qu'ils ont enduré.

Je remercie vivement tous les membres du jury qui ont accepté d'examiner mon travail de mémoire. J'exprime en particulier ma gratitude à Monsieur BOUTOUT Farid et CHAFAA Kheireddine mes rapporteurs pour leurs conseils et encouragements, leurs aides et leurs soutien.

J'adresse par ailleurs mes vifs remerciements à monsieur le doyen BOUAMAR Mohamed qui nous a accompagné durant tout le déroulement de magister.

Les autres membres permanents l'ensemble des enseignants du département d'électronique n'ont jamais été avares de leur temps ni de leurs connaissances, qu'ils en soient ici remerciés.

Je tiens enfin à adresser ma profonde gratitude à toutes les promotions d’électronique.

SOMMAIRE

CHAPITRE III : *IDENTIFICATION ET REJECTION DES PERTURBATIONS DANS LES SYSTEMES DYNAMIQUES*

CHAPITRE IV : *RÉSULTATS DE SIMULATION*

INTRODUCTION GÉNÉRALE

INTRODUCTION GÉNÉRALE

Les réseaux neuronaux artificiels ont été utilisés avec succès comme blocs structurels dans l'architecture de l'identification et la commande des systèmes dynamiques non linéaires [1], [2]. Cette architecture est basée sur l'entraînement du réseau utilisant les données d'entrée-sortie du système. En général, les paramètres du réseau sont ajustés en utilisant la méthode du Gradient. Pour une classe spécifique de systèmes non linéaires, existe une relation proportionnelle entre l'identification et la rejection de perturbation des systèmes non linéaires [3].

La logique floue soulève un large intérêt, tant théorique que pratique dans la commande des processus complexes et non linéaires. Cela est dû essentiellement à trois faits : 1) Les systèmes flous permettent une simple inclusion des informations qualitatives dans la conception des régulateurs et des modèles d'identification, 2) les systèmes flous n'exigent pas l'existence d'un modèle mathématique du processus à contrôler, et peu d'information suffit pour mettre en œuvre la boucle de commande et 3) les systèmes flous sont des systèmes non linéaires et de ce fait, sont adaptés à la commande des processus non linéaires [4]. La logique floue n'a pas été utilisée dans le problème de la réjection des perturbations et fera donc l'objet de la deuxième partie de notre étude.

Ce travail est une étude comparative entre deux méthodes (Réseaux de neurone et la logique floue).

Il est organisé en quatre chapitres principaux :

- Dans le chapitre I, nous exposons les éléments de base des réseaux de neurones, ces principales architectures, nous concentrons l'étude dans ce chapitre sur l'approche de rétro-propagation de l'erreur.
- Dans le chapitre II, après une présentation succincte du principe de la logique floue nous détaillons les différents types des règles floues. Nous détaillons, en particulier, les règles floues de type Mamdani et Takagi-Sugeno. Nous accentuons l'étude sur le modèle flou de TS d'ordre 0 (modèle singleton).

- Le chapitre III aborde le problème de l'identification et le principe de la rejection de perturbation ainsi que la relation entre eux.
- Dans le chapitre IV, des résultats de simulation pour des systèmes étudiés sont présentés et commentés. Nous comparons également les réseaux de neurones et la logique floue pour l'identification et la rejection des perturbations dans les systèmes non linéaire.

Une conclusion générale est donnée à la fin de ce mémoire avec des perspectives pour des études futures.

CHAPITRE I

LES RÉSEAUX DE NEURONES

I. Introduction

Dans la plupart des modélisations des systèmes industriels, des divergences persistent entre le comportement du système réel et l'évolution du modèle. Ces divergences sont dues, d'une part, aux manques de connaissances exhaustives sur le fonctionnement de l'équipement et, d'autre part, le modèle ne prend en compte qu'une partie des paramètres qui influent l'évolution de la sortie. Par ailleurs, dans certains cas, ce modèle est quasiment impossible à obtenir.

Les réseaux de neurones peuvent fournir une solution pour les problèmes de l'identification et de la rejection de la perturbation. Leur capacité de mémorisation, d'apprentissage et d'adaptation sont des fonctions utiles pour tout système non linéaire [5].

Ce chapitre est structuré en deux parties. Une première partie est consacrée à la présentation des réseaux de neurones artificiels. Nous commençons par donner une brève présentation de l'évolution historique de cet axe de recherche dont la première inspiration biologique remonte à 1890. Avant de présenter le principe de fonctionnement des neurones artificiels, nous décrivons les bases essentielles des neurones biologiques. Nous concluons cette partie en présentant les propriétés les plus importantes des réseaux de neurones artificiels.

La deuxième partie de ce chapitre est consacrée aux architectures neuronales les plus utilisées qui sont : *le Perceptron Multicouche*, les *Réseaux à Fonctions de Base Radiales*, les mémoires auto associatives (modèle de *Hopflield*), la carte de *Kohonen,* les réseaux compétitifs, les réseaux récurrents et l'associateur linéaire.

II. Eléments de base des réseaux de neurones

II. 1. Historique

L'origine de l'inspiration des réseaux de neurones artificiels remonte à 1890 lorsque ***James*** introduit le concept de mémoire associative. Il propose ce qui deviendra une loi de fonctionnement pour l'apprentissage des réseaux de neurones, connue plus tard sous le nom de loi de ***Hebb***. En *1949* ***Culloch*** et ***Pitts*** montrent que des réseaux de neurones formels simples peuvent réaliser des fonctions logiques arithmétiques et symboliques complexes.

C'est ensuite que ***Hebb*** physiologiste américain, présente en 1949 la propriété des neurones par le conditionnement chez l'animal. Ainsi, un conditionnement de type ***pavlovien*** tel que, nourrir tous les jours à la même heure un chien, entraîne chez 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.

Les premiers succès de cette discipline remontent à 1957, lorsque ***Rosenblatt*** développe le modèle du *Perceptron*. Il construit le premier neuro-ordinateur basé sur ce modèle et l'applique au domaine de la reconnaissance des formes. En 1960, l'automaticien ***Widrow*** développe le modèle ***Adaline*** (Adaptative Linear Element). Dans sa structure, le modèle ressemble au Perceptron, cependant la loi d'apprentissage est différente.

Celle-ci est à l'origine de l'algorithme de rétro-propagation de gradient très utilisé aujourd'hui. C'est un algorithme d'apprentissage adapté au Perceptron Multicouche. Grâce a cette découverte, nous avons la possibilité de réaliser une fonction non linéaire d'entrée/sortie sur un réseau, en décomposant cette fonction en une suite d'étapes linéairement séparables. En 1989, ***Moody*** et ***Darken*** exploitent quelques résultats de l'interpolation multivariables pour proposer le Réseau à Fonctions de base Radiales (*Radial Basis Function Network ; RBF*). Ce type de réseau se distingue par sa représentation locale [6].

II. 2. Fondements biologiques

Les premières études faites par les philosophes grecs pour localiser le siège de l'intelligence dans le corps humain attribuaient la capacité de raisonner au coeur et laissaient au cerveau un rôle très mineur. Cette croyance restera en vigueur jusque dans les écrits de ***Descartes*** qui réaffirme le rôle prépondérant du coeur dans la capacité du raisonnement. Cependant, à partir du dix-huitième siècle, ***La Mettrie*** et ***Cabanis*** affirment que le cerveau est l'organe de contrôle central du corps humain reprenant ainsi des études de ***Platon***, ***Hypocrate*** et ***Galien*** portées dans l'oubli.

Des études plus récentes, depuis le début du vingtième siècle et surtout après la deuxième guerre mondiale, s'intéressent au cerveau d'un point de vue microscopique et cellulaire. Il s'agit cette fois de relier les faits du comportement et les réactions électriques et chimiques qui se produisent dans notre cerveau. Ceci a permis d'isoler la

composante cellulaire de base du cerveau, ***le neurone*** qui est l'unité de traitement de l'information. Les neurones constituent environ quarante pour cent de la masse totale du cerveau ; le reste étant constitué de cellules de support pour protéger, nourrir et soutenir physiquement l'ensemble.

Il y a différents types de neurones regroupés dans différents organes pour réaliser différentes fonctions: acquérir un signal du monde extérieur (nerf optique), le traiter (cerveau) ou envoyer des stimuli aux muscles (moelle épinière). Cependant, tous les neurones possèdent les mêmes constituants élémentaires, les dendrites, le noyau et l'axone.

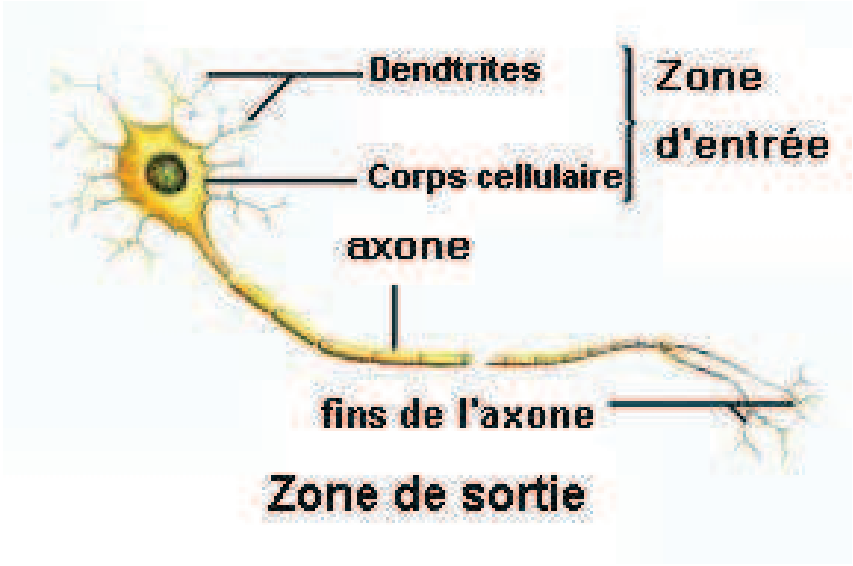


Figure I. 1 : un neurone

Chaque neurone reçoit un ensemble de potentiels excitateurs ou inhibiteurs par l'intermédiaire des synapses qui le relient aux neurones qui le précèdent. Les dendrites calculent une somme pondérée de leurs entrées. Selon le niveau d'activation obtenu, le neurone génère ou non un potentiel d'action qui se propage le long de l'axone. De cette manière, le neurone réalise ses cinq fonctions : ***recevoir*** des signaux, les ***intégrer, engendrer*** un influx nerveux et le ***conduire*** le long de l'axone pour le ***transmettre*** aux neurones suivants. Ce modèle biologique simple sert de base au modèle mathématique du neurone formel.

Les recherches en biologie ont des modèles de fonctionnement du neurone vivant beaucoup plus complexes. Le rôle du noyau s'est réduit, passant d'une unité de

transformation de l'information à une unité de maintien de la vie cellulaire, alors que dendrites, synapses et axone voyaient leur importance respective renforcée, passant de simples unités de transfert à des unités de traitement de l'information.

Le neurone génère non pas un potentiel unique mais des trains de potentiels et ce sont ces trains de potentiels qui transportent l'information. L'amplitude des potentiels est presque constante, la caractéristique de l'information étant contenue dans la forme fréquentielle du train d'impulsions généré par le neurone. Les dendrites réalisent des combinaisons complexes des signaux transmis par les synapses. L'axone module des trains de potentiels et les répartit différemment sur ses différentes terminaisons, l'ensemble forme donc un filtre analogique non linéaire [7].

II. 3. Modèle du neurone formel

Le modèle du neurone formel utilisé dans la plupart des réseaux actuels a été proposé en 1943 par ***McCulloch*** et ***Pitts*** [5], à partir des connaissances en neurobiologie de cette époque. La structure d'un neurone formel est représentée au Figure I.2 :

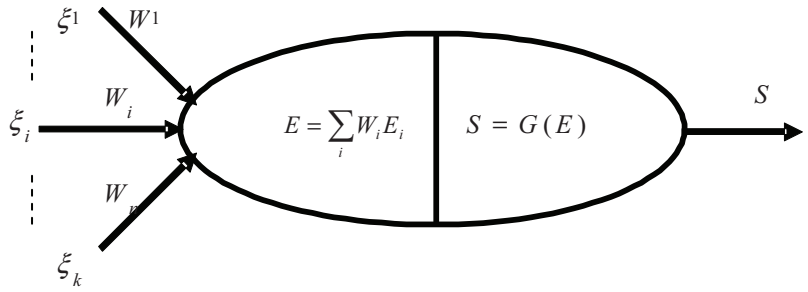


Figure. I.2: Modèle du neurone formel

Le neurone formel est un simple automate qui se compose de plusieurs entrées et d'une sortie. Les entrées et la sortie peuvent être binaires ou réelles. L'automate réalise une somme pondérée de ses entrées. La sortie est obtenue ensuite en appliquant à cette somme pondérée la fonction d'activation G .

Pour des sorties binaires, cette fonction est un seuil qui fait passer le neurone de l'état non activé ($S = 0$ ou $S = -1$) à l'état activé ($S = 1$). Pour des sorties continues la fonction de ***Heaviside*** (fonction de seuil) est remplacée par une fonction sigmoïde qui en plus d'être une approximation de celle-ci, elle est dérivable. Cette propriété est nécessaire pour appliquer l'algorithme d'apprentissage présenté plus loin.

Les poids w_i (*Weights*) sont des valeurs réelles qui constituent les paramètres du neurone formel. Ces paramètres sont fixés par l'opérateur. L'intérêt de ce modèle est

que l'on dispose également d'un algorithme d'apprentissage qui permet de modifier les valeurs des poids w_i pour faire apprendre au neurone la fonction que l'on désire réaliser. Les algorithmes d'apprentissage reposent sur la règle de ***Hebb*** : deux neurones reliés par une connexion qui s'activent simultanément renforcent les poids w_i de cette connexion. Le changement de valeur des poids est également fonction de l'erreur entre la sortie désirée par l'opérateur et la sortie proposée par le neurone.

Ce modèle est simple et peu fidèle à la réalité biologique telle que nous la connaissons maintenant. Il fonctionne en fait comme un automate non linéaire. Cependant, il est assez simple pour être largement utilisé tant en recherche que pour des applications commerciales. C'est autour de ce modèle du neurone formel que sont construits la plupart des réseaux utilisés à ce jour [8][9].

II. 4. Fonction d'activation ou de seuillage

Une fonction de transfert calcule la valeur de l'état du neurone, c'est cette valeur qui sera transmise aux neurones aval. Il existe plusieurs formes possibles pour la fonction de transfert. Les plus courantes sont présentées dans la Figure.I.3. On remarque qu'à la différence des neurones biologiques dont l'état est binaire, la plupart des fonctions de transfert sont continues, offrant une infinité de valeurs possibles comprises dans l'intervalle $[0,+1]$ ou $[-1,+1]$ [10].

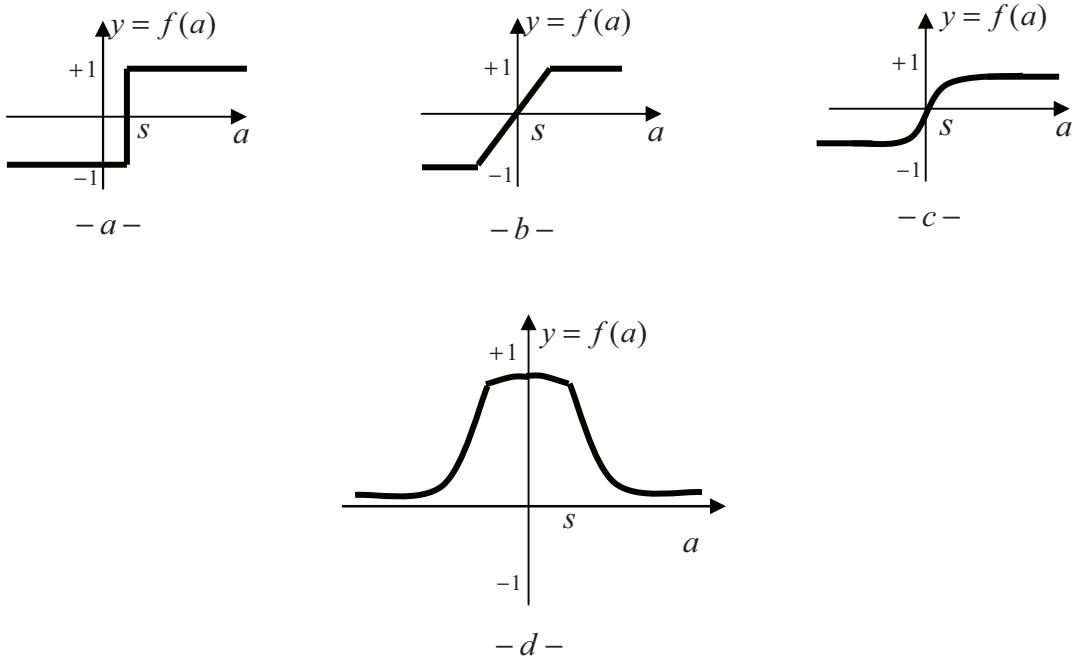


Figure I.3 : Différents types de fonctions de transfert pour le neurone artificiel.

-a- Fonction à seuil du neurone de ***Mc Culloch-Pitts***

-b- linéaire par morceaux du modèle ***Adaline*** de ***Widrow-Hoff***

-c- sigmoïde d'un réseau Perceptron Multicouches de ***Rosenblatt***

-d- gaussienne du réseau RFR de ***Moody-Darken***

III. Les réseaux de neurones

Un Réseau de Neurones Artificiel RNA (Artificial Neural Network ;ANN) est un ensemble de neurones formels (d'unités de calcul simples, de noeuds processeurs) associés en couches (ou sous-groupes) et fonctionnant en parallèle.

Dans un réseau, chaque sous-groupe fait un traitement indépendant des autres et transmet le résultat de son analyse au sous-groupe suivant L'information donnée au réseau va donc se propager couche par couche, de la couche d'entrée à la couche de sortie, en passant soit par aucune, une ou plusieurs couches intermédiaires (dites couches cachées). Il est à noter qu'en fonction de l'algorithme d'apprentissage, il est aussi possible d'avoir une propagation de l'information à reculons (*backpropagation*). Habituellement (excepté pour les couches d'entrée et de sortie), chaque neurone dans une couche est connecté à tous les neurones de la couche précédente et de la couche suivante, les RNA ont la capacité de stocker de la connaissance empirique et de la rendre disponible à l'usage. Les habiletés de traitement (et donc la connaissance) du réseau vont être stockées dans les poids synaptiques, obtenus par des processus d'adaptation ou d'apprentissage. En ce sens, les RNA ressemblent donc au cerveau car non seulement, la connaissance est acquise au travers d'un apprentissage mais de plus, cette connaissance est stockée dans les connexions entre les entités soit dans les poids synaptiques [8].

III.1. Propriétés des Réseaux de neurones artificiels

III.1.1. Apprentissage et mémoire

L'une des caractéristiques les plus complexes du fonctionnement de notre cerveau est bien la phase d'apprentissage. C'est une phase au bout de laquelle certaines modifications s'opèrent entre les connexions des neurones : certaines sont renforcées et d'autres affaiblies ou carrément inhibées. Le cerveau converge alors vers un comportement souhaité : par exemple l'apprentissage d'une langue, ou encore l'apprentissage par un enfant à reconnaître son environnement. Ceci nous emmène à la

notion de mémoire qui donne au cerveau la capacité de retrouver des expériences passées. Le cerveau possède plusieurs types de mémoires. Nous ne développons pas ces différents types de mémoires mais ce que nous retiendrons est que le cerveau humain procède par association. Cela permet par exemple de retrouver une information à partir d'éléments incomplets ou imprécis (bruités).

Par exemple, le fait de voir un bout d'une photographie qu'on connaît déjà est suffisant pour que notre cerveau soit capable de la reconnaître. Le mécanisme de l'association permet aussi au cerveau de converger vers un état à partir d'un autre état. Par exemple, le fait de passer devant une boulangerie nous fait rappeler qu'on devait acheter du pain. Cette deuxième importante caractéristique est aussi connue sous le nom de mémoire adressée par le contenu, dont le modèle de ***Hopfield*** s'en inspire. Par analogie avec les réseaux de neurones biologiques, les réseaux de neurones artificiels tentent de reproduire les caractéristiques les plus importantes du comportement biologique, à savoir *l'apprentissage*, *la généralisation* et *l'association*.

L'apprentissage des réseaux de neurones artificiels est une phase qui permet de déterminer ou de modifier les paramètres du réseau, afin d'adopter un comportement désiré. Plusieurs algorithmes d'apprentissage ont été développés depuis la première règle d'apprentissage de ***Hebb*** en 1949. Nous présentons les types l'apprentissage qui sont classés en deux catégories : *supervisé* et *non supervisé*.

Dans l'apprentissage *supervisé*, un superviseur (ou expert humain) fournit une valeur ou un vecteur ζ de sortie (appelé cible ou sortie désirée) que le réseau de neurones doit associer au vecteur d'entrée X. L'apprentissage consiste dans ce cas à modifier les paramètres du réseau de neurones afin de minimiser l'erreur entre la sortie cible et la sortie réelle du réseau de neurones.

Dans l'apprentissage *non supervisé*, les données ne contiennent pas d'informations sur une sortie désirée. Il n'y a pas de superviseur. Il s'agit de déterminer les paramètres du réseau de neurones suivant un critère à définir [5].

III.2. Le Perceptron Multicouches (PMC)

III.2.1. Le Perceptron simple

Comme nous l'avons énoncé au début de ce chapitre, le premier modèle de réseau de neurones a été introduit par **McCulloch** et **Pitts** en 1949. Les neurones de ce modèle sont binaires. Ils reçoivent des signaux excitateurs et inhibiteurs provenant des neurones

en amont afin d'effectuer une sommation. Si cette somme est supérieure à un seuil, le neurone est dans un état actif et émet un signal vers les autres neurones. Si ce n'est pas le cas, il est dans un état inactif et n'émet aucun signal.

Après la description de ce premier modèle de neurones, il fallait disposer d'un moyen de réaliser l'apprentissage. S'inspirant du conditionnement de type ***pavlovien*** chez l'animal, ***Donald O***. ***Hebb*** a introduit en 1949 la première règle d'apprentissage qui permet de modifier les poids synaptiques. Le principe de cette règle – connue sous la règle de ***Hebb*** .

« *Lorsqu'une connexion entre deux cellules est très forte, si la cellule émettrice s'active alors la cellule réceptrice s'active aussi. Pour lui permettre de jouer ce rôle déterminant lors du mécanisme d'apprentissage, il faut donc augmenter le poids de cette connexion. En revanche, si la cellule émettrice s'active sans que la cellule réceptrice le fasse, ou si la cellule réceptrice s'active alors que la cellule émettrice ne s'était pas activée, cela traduit le fait que la connexion entre ces deux cellules n'est pas prépondérante dans le comportement de la cellule réceptrice. On peut donc, dans la phase d'apprentissage, laisser un poids faible à cette connexion ».*

En d'autres termes, si deux neurones connectés entre eux sont activés en même temps alors on renforce la connexion qui les relie. Dans le cas contraire, elle n'est pas modifiée.

Cette règle d'apprentissage a été le point de départ des travaux *de* ***Rosenblatt*** (1958) qui a développé la première version d'un modèle neuronal très connu de nos jours, à savoir le *Perceptron*. C'est un réseau à deux couches (une couche d'entrée et une couche de sortie) de type ***feedforward*** (propagation avant). Les neurones de la couche d'entrée ont pour rôle de fournir au réseau les données externes. Chaque neurone de la couche de sortie effectue une somme pondérée de ses entrées :

$$o_i = f(a_i) = f(\sum_{k=1}^{k=N} w_{ik}\xi_k) \qquad \text{(I-1)}$$

Où w_{ik} est le poids de la connexion qui relie l'unité k à l'unité i, a_i est l'activation de l'unité i, f est la fonction d'activation des unités (Figure.I-4). Cette fonction d'activation est du type fonction à seuil avec l'expression suivante :

$$f(x) = \begin{cases} +1 & \text{si} \quad x \geq \theta \\ -1 & \text{si} \quad x \leq \theta \end{cases} \tag{I-2}$$

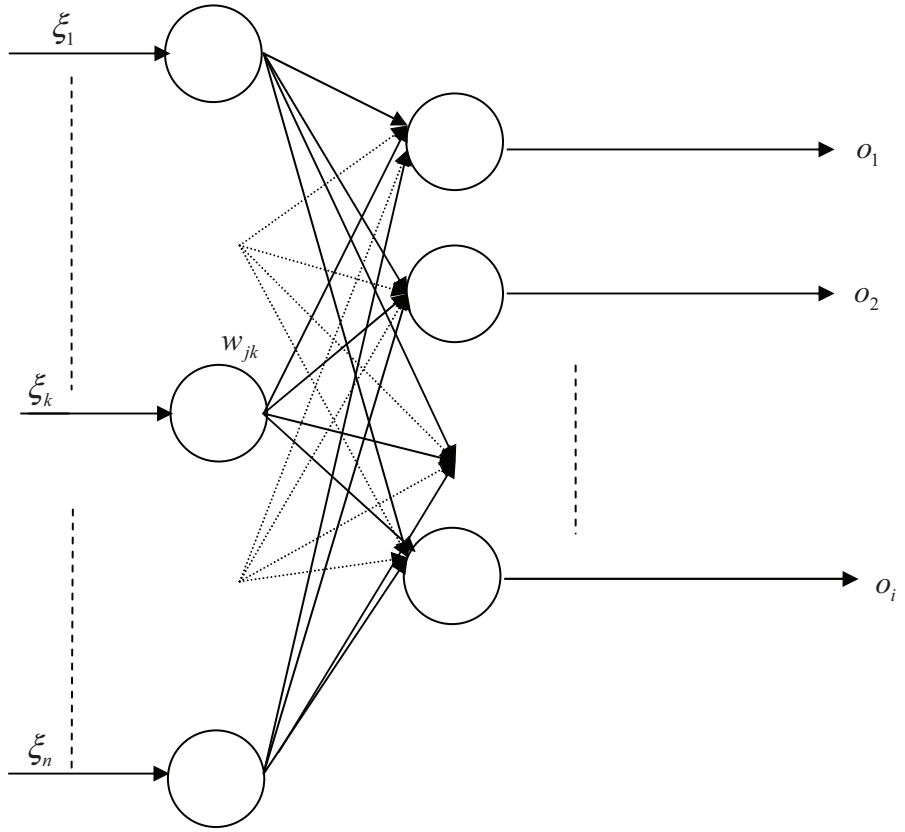


Figure I.4 : Perceptron Simple, modèle de ***Rosenblatt***

Le rôle de l'apprentissage est de modifier les poids des connexions entre les neurones d'entrée et ceux de sortie, de manière à obtenir une réponse que l'on souhaite reproduire par le réseau de neurones. ***Rosenblatt*** s'est inspiré de la règle de ***Hebb*** pour la modification des poids. Son principe est de rajouter, dans le cas où la sortie obtenue $o_i^{\wp}$ du réseau est différente de la sortie désirée $\xi_i^{\wp}$, une quantité Δw_{ik} aux poids de chaque connexion. Dans le cas contraire, les connexions demeurent inchangées. On peut exprimer ce principe par l'équation suivante :

$$w_{ik}^{nouveau} = w_{ik}^{ancien} + \Delta w_{ik} \tag{I-3}$$

où Δw_{ik} est la quantité ajoutée au poids w_{ik}. Pour chaque exemple $\wp$ de l'ensemble des exemples d'apprentissage, on peut ainsi écrire :

$$\Delta w_{ik} = \eta(\xi_i^{\wp} - o_i^{\wp})\xi_k^{\wp} \qquad \text{(I-4)}$$

Le paramètre η est appelé *taux d'apprentissage*. Il détermine la dynamique suivant laquelle les modifications vont avoir lieu.

Cette procédure d'apprentissage pour le *Perceptron* simple peut converger vers un état des poids des connexions donnant de bons résultats, à la seule condition que le problème soit linéairement séparable. Malheureusement, un grand nombre de problèmes rencontrés en pratique, ne sont pas linéairement séparables. Le *Perceptron* possède tout de même une bonne capacité de généralisation [5].

III.3. La rétro-propagation

Un des désavantages du *Perceptron* est qu'il minimise une erreur en tout ou rien à cause de sa fonction d'activation (I-2). Il ne prend donc pas en compte la notion de distance. De ce fait, il est très peu robuste. La règle d'apprentissage de ***Widrow-Hoff*** (règle de ***Delta***) ne travaille plus en tout ou rien mais minimise une fonction d'erreur quadratique, donc plus robuste. Malheureusement, cette règle ne peut s'appliquer que sur des réseaux à une seule couche de poids adaptatifs. C'est donc en étendant la règle de ***Widrow-Hoff*** que plusieurs équipes de chercheurs (***Le Cun,*** 1985) et (***Werbos***, 1974) ont développé un algorithme d'apprentissage appelé *rétro-propagation du gradient de l'erreur*, généralisé ensuite par l'équipe de ***Rummelhart*** en 1986. Cet algorithme fournit une façon de modifier les poids des connexions de toutes les couches d'un *Perceptron* Multicouches (*PMC*).

Soit le réseau à deux couches décrit par la Figure.I-5 dans lequel les unités de sortie sont notées o_j les unités cachées v_j et les unités d'entrée ζ_k Les connexions des unités d'entrée aux unités cachées sont notées w_{jk} et celles des unités cachées aux unités de sortie par w_{ij}.

L'entrée k a pour valeur $\xi_k^{\wp}$ lorsque la donnée $\wp$ est présentée au réseau. Ces valeurs peuvent être binaires (0/1 ou +1/-1) ou continues.

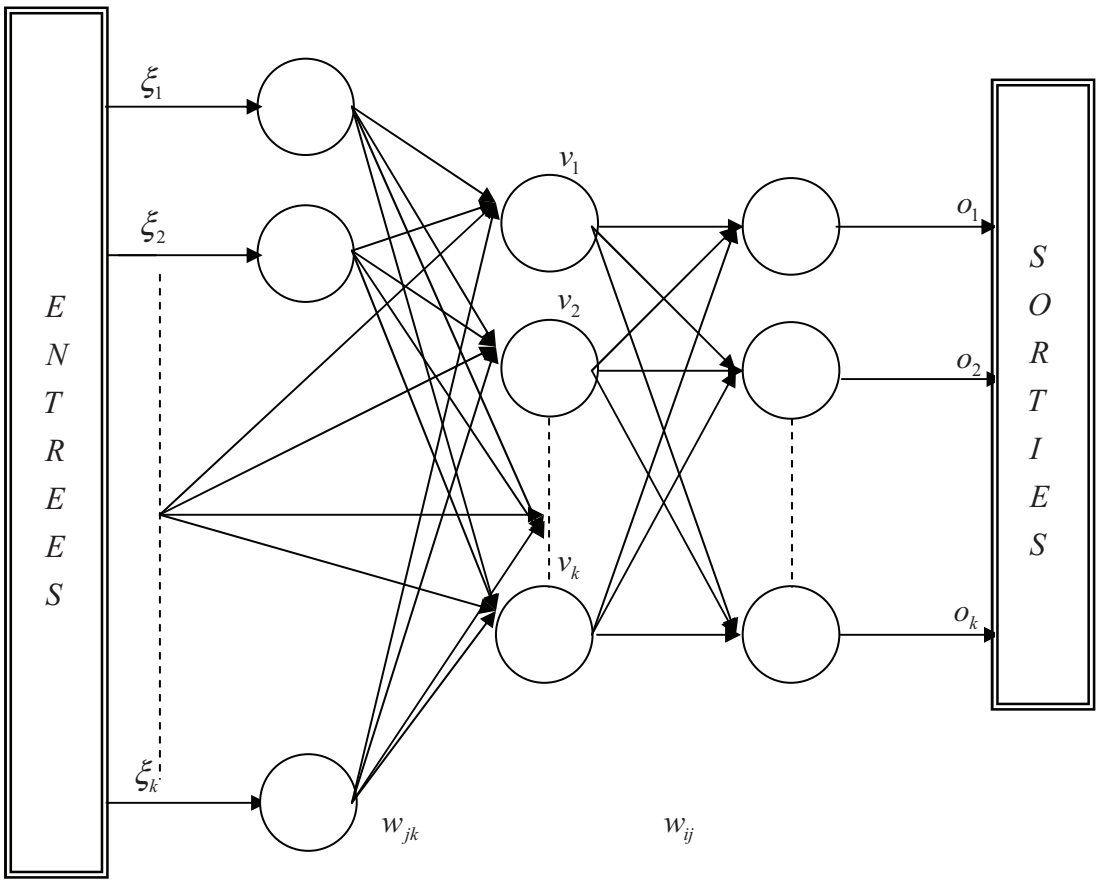


Figure I.5 : Perceptron Multicouches

Pour la donnée d'entrée $\wp$, la valeur de sortie de l'unité cachée *j* est donnée par :

$$v_j^{\wp} = f(a_j^{\wp}) = f(\sum_k w_{jk}\xi_k^{\wp}) \tag{I-5}$$

Les unités de sortie ont comme valeur :

$$o_i^{\wp} = f(a_i^{\wp}) = f(\sum_k w_{ik}v_j^{\wp}) \tag{I-6}$$

Les fonctions d'erreurs partielles et globales sont alors définies par :

$$E^{\wp} = \frac{1}{2}(o_i^{\wp} - \xi_i^{\wp})^2 \quad \text{et} \quad E = \sum_{\wp} E^{\wp} \tag{I-7}$$

La minimisation de la fonction d'erreur globale va se faire par une descente de gradient. Par conséquent, après la présentation de tous les vecteurs d'entrée de la base d'apprentissage, nous modifierons la valeur de chaque connexion par la règle.

$$\Delta w = -\eta \frac{\partial E}{\partial w} = -\eta \sum_{\wp} \frac{\partial E^{\wp}}{\partial w} \tag{I-8}$$

Cette règle d'apprentissage est généralement appelée la règle de ***delta*** *généralisée*.

Dans l'expression (I-7), seule la sortie o_i dépend du paramètre w_{ij}. Selon la position des poids des connexions, deux cas se présentent :

• *Cas des connexions entre la couche cachée et celle de sortie* $\left(w_{ij}\right)$:

Pour le cas des neurones de sortie, l'expression (I-8) devient fonction du paramètre w_{ij} qui influe uniquement sur la sortie du neurone d'indice i. Nous pouvons donc décomposer la dérivée de l'expression (I-8) par :

$$\frac{\partial E^{\wp}}{\partial w_{ij}} = \frac{\partial E^{\wp}}{\partial o_i^{\wp}} \frac{\partial o_i^{\wp}}{\partial a_i^{\wp}} \frac{\partial a_i^{\wp}}{\partial w_{ij}} = (o_i^{\wp} - \xi_i^{\wp}) f_i^{'}(a_i^{\wp}) v_j^{\wp} \tag{I-9}$$

L'expression (I-8) devient alors

$$\Delta w_{ij} = \eta \sum_{\wp} f_i^{'}(a_i^{\wp})(\xi_i^{\wp} - o_i^{\wp}) v_j^{\wp} \tag{I-10}$$

• *Cas des connexions entre la couche d'entrée et la couche cachée* $\left(w_{ik}\right)$:

Pour le cas des neurones cachés, l'expression (I-8) est fonction du paramètre w_{ik} qui influe non seulement sur la sortie du neurone j de la deuxième couche, mais aussi sur tous les neurones j de la couche de sortie (en aval) qui lui sont connectés. On obtient alors l'équation suivante :

$$\frac{\partial E^{\wp}}{\partial w_{jk}} = \frac{\partial E^{\wp}}{\partial v_j^{\wp}} \frac{\partial v_j^{\wp}}{\partial a_j^{\wp}} \frac{\partial a_j^{\wp}}{\partial w_{jk}} = \frac{\partial E^{\wp}}{\partial v_j^{\wp}} f_j^{'}(a_j^{\wp}) \xi_k^{\wp} \tag{I-11}$$

Le premier terme de cette expression devient alors :

$$\frac{\partial E^{\wp}}{\partial v_j^{\wp}} = \sum_i \frac{\partial E^{\wp}}{\partial o_i^{\wp}} \frac{\partial o_i^{\wp}}{\partial v_j^{\wp}} = \sum_i \frac{\partial E^{\wp}}{\partial o_i^{\wp}} \frac{\partial o_i^{\wp}}{\partial a_i^{\wp}} \frac{\partial a_i^{\wp}}{\partial v_j^{\wp}} = \sum_i (o_i^{\wp} - \xi_i^{\wp}) f_i^{'}(a_i^{\wp}) w_{ij} \tag{I-12}$$

On obtient alors la modification des poids :

$$\Delta w_{jk} = \eta \sum_{\wp} \left\{ f_j^{'}(a_j^{\wp}) \xi_i^{\wp} \left\{ \sum_i (\xi_i^{\wp} - o_i^{\wp}) f_i^{'}(a_i^{\wp}) w_{ij} \right\} \right\} \qquad \text{(I-13)}$$

Après avoir calculé la variation des poids des connexions pour tous les neurones de sortie via l'équation (I-10), on calcule alors la variation des poids des connexions de la couche cachée par l'équation (I-13). On met ainsi à jour les poids des connexions de la couche de sortie jusqu'à la couche d'entrée : on *rétropropage* ainsi le signal d'erreur. C'est de là que vient le nom de cet algorithme : *rétro-propagation du gradient de l'erreur*. Du fait de sommer les Δw_{ij} pour tous les vecteurs $\wp$ de la base d'apprentissage puis de remettre à jour les poids avec la variation totale ainsi calculée, l'algorithme est appelé gradient total. Une autre façon de faire, appelée version séquentielle, modifie les poids des connexions après chaque présentation d'un vecteur d'entrée $\wp$. Une version stochastique permet de prendre en compte les vecteurs d'apprentissage $\wp$ d'une façon aléatoire.

L'algorithme de rétro-propagation du gradient de l'erreur a permis de dépasser les limites du *Perceptron* simple. Il s'avère capable de résoudre un grand nombre de problèmes de classification et de reconnaissance des formes et a donné lieu à beaucoup d'applications. Cet algorithme souffre néanmoins de nombreux défauts, parmi lesquels :

- Temps de calcul excessif ; apprentissage très long.
- Une grande sensibilité aux conditions initiales, c'est-à-dire à la manière dont sont initialisés les poids des connexions.
- De nombreux problèmes sont dus à la géométrie de la fonction d'erreur: minimums locaux. Ce problème est en partie résolu avec le gradient stochastique, mais il subsiste quand même.
- Le problème de dimensionnement du réseau. La rétro-propagation apprend une base d'apprentissage sur un réseau dont la structure est fixée a priori. La structure est définie par le nombre de couches cachées, le nombre de neurones

par couches et la topologie des connexions. Un mauvais choix de structure peut dégrader considérablement les performances du réseau [5].

III.3.1. Adaline

C'est un autre type de réseau spécialisé dans le traitement du signal fonctionnant à partir d'une règle d'apprentissage. Cette règle permet de réduire l'erreur observée entre l'état du réseau et la valeur correcte attendue par une modification des connexions des neurones de façon à la minimiser (par un algorithme dit de descente de gradient) [11].

III.3.2. L'associateur linéaire

C'est un modèle de réseau à deux couches avec une fonction d'activation linéaire utilisant la règle de ***Hebb. D. Hebb*** a proposé en 1949 l'idée que le cerveau s'adapte à son environnement en modifiant l'efficacité des connexions entre ses neurones. Le principe de ***Hebb*** postule qu'une synapse améliore son efficacité seulement quand l'activité des deux neurones qu'elle relie est corrélée [11].

III.3.3. Réseaux récurrents

Permettent une interconnectivité quasi totale entre les neurones des différentes couches, y compris éventuellement en se bouclant sur eux-mêmes. Ils permettent la résolution de problèmes qui ne peuvent être résolus de façon algorithmique [12].

III.3.4. Réseau de Hopfield

Inspiré des modèles physiques des verres de spin sont constitués d'éléments bistables à connexions symétriques évoluant spontanément vers une réduction de l'énergie totale du système. Associé avec la règle de ***Hebb,*** ce réseau peut apprendre à mémoriser les exemples présentés en entrée sous la forme d'états stables En phase d'exploitation, les stimuli présentés en entrée évolueront dans le réseau vers l'état stable le plus ressemblant. Un réseau de ***Hopfield*** fonctionne ainsi comme un véritable classificateur à mémoire associative, cependant il existe une marge non négligeable d'erreurs due à l'existence d'états métastables, parfois très proches des états stables correspondant à une classification correcte [11].

III.3.5. Réseaux compétitifs

Sont des réseaux où chaque neurone d'entrée est relié à chaque neurone de sortie et chaque neurone de sortie inhibe tous les autres et s'auto excite. Cette architecture génère une compétition inter-neurones aboutissant à ce que le réseau a tendance à reproduire l'organisation topographique des formes d'entrée. En d'autres termes, si l'on présente à ce type de réseau des objets quelconques, le réseau va reproduire dans ses états internes ses traits structuraux.

Les réseaux compétitifs reproduisent une particularité du fonctionnement biologique des neurones, à savoir l'inhibition latérale. On sait en effet que lorsqu'un neurone biologique est excité, il transmet son excitation aux neurones voisins dans un rayon très court et inhibe par contre les neurones situés à plus grande distance [11].

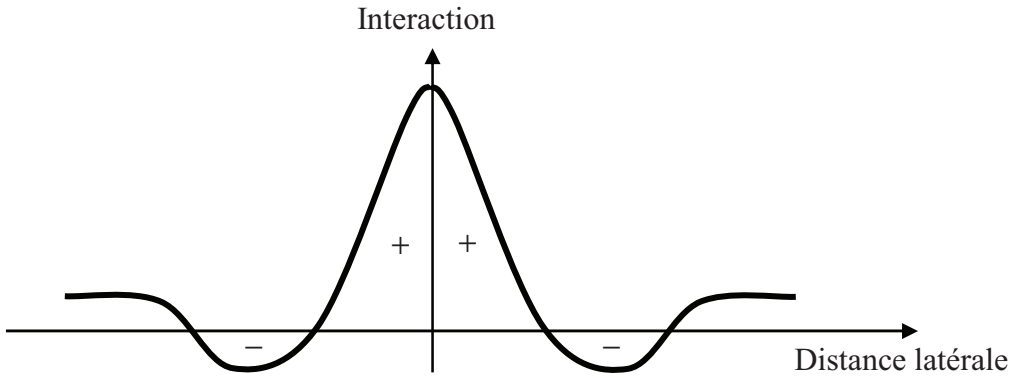


Figure I.6 Phénomène d'inhibition latérale dans les neurones biologiques et formels

Un neurone activé inhibe ses voisins. Il en résulte un phénomène de compétition inter-neurones dont le résultat est l'extraction des classes constituantes.

Biologiquement, le phénomène décrit par la Figure.I-6 est à la source des champs récepteurs en neurophysiologie sensorielle. Le degré d'interaction latérale est donc une fonction de la distance. Ces interactions latérales engendrent des réponses de groupes de neurones répartis autour du maximum local d'excitation. Les algorithmes d'apprentissage dans les réseaux compétitifs sont des simplifications algorithmiques utilisant l'idée de réponses localisées et d'interactions latérales. La forme la plus simple d'apprentissage compétitif modifie le vecteur poids du meilleur neurone à chaque étape de l'apprentissage [11].

III.3.6. Réseau de Kohonen

Il a été observé que, dans de nombreuses zones du cortex cérébral, des colonnes voisines ont tendance à réagir à des entrées similaires. Dans les aires visuelles, par exemple, deux colonnes proches sont en correspondance avec deux cellules proches de la rétine (***Hubel*** ,1977). Des observations identiques ont pu être faites dans le bulbe olfactif ou dans l'appareil auditif (***Knudsen***, 1979). Ces observations ont mené ***Kohonen*** (***Kohonen***, 1989) à proposer un modèle de *carte topologique auto-adaptative* qui permet de coder des motifs présentés en entrée, tout en conservant la topologie de l'espace d'entrée.

Dans la plupart des applications, les neurones d'une carte de ***Kohonen*** sont disposés sur une grille 2D (Figure I.7). Chaque neurone i de la carte effectue un calcul de la distance euclidienne entre le vecteur d'entrée ξ et le vecteur poids w_i .

Dans les réseaux de ***Kohonen***, la mise à jour des paramètres des neurones s'effectue sur tout un voisinage d'un neurone i. Un rayon de voisinage r représente donc la longueur du voisinage d'un neurone i en terme de nombre de neurones. On définit alors une fonction $h(i,k)$ égale à 1 pour tous les neurones k voisins du neurone i compris dans le rayon r et égale à zéro pour tous les autres neurones. L'algorithme d'apprentissage de la carte de ***Kohonen*** se présente comme suit :

- Initialiser aléatoirement les vecteurs w_i. On donne une valeur initiale au rayon r et au taux d'apprentissage η.
- Calculer la distance euclidienne entre le vecteur présenté ξ et le vecteur de poids de chaque neurone.
- Choisir le neurone k ayant la distance la plus petite.
- Les vecteurs de pondération de tous les neurones i de la carte de ***Kohonen*** sont alors mis à jour selon l'équation :

$$w_i \leftarrow w_i + \eta h(i,k)(\xi - w_i) \qquad \text{(I-14)}$$

Réduire la taille du voisinage et reprendre l'apprentissage du vecteur des pondérations. Après un long temps de convergence, le réseau évolue de manière à représenter au mieux la topologie de l'espace de départ. Il faut noter que la notion de conservation de la topologie est en fait abusive puisqu'en général, la taille du vecteur d'entrée est bien supérieure à la dimension de la carte (souvent égale à 2) et il est donc impossible de conserver parfaitement la topologie [6].

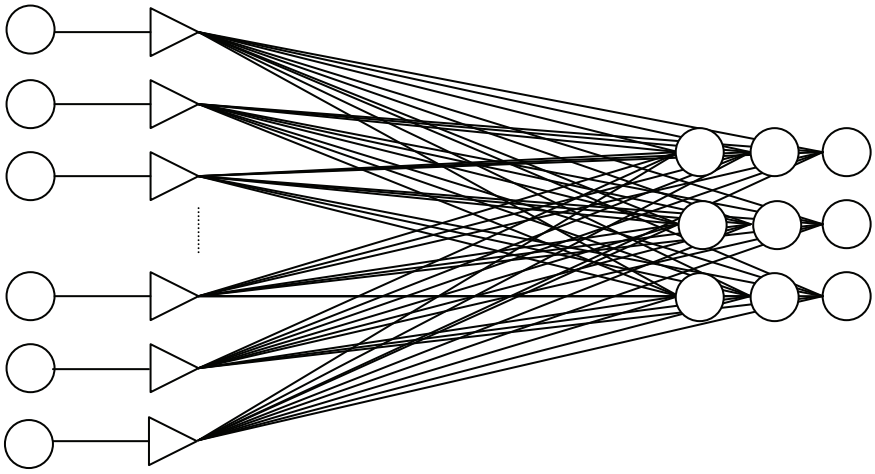


Figure I.7 : Carte topologique auto-adaptative de Kohonen.

III.3.7. Réseau de neurones à Fonctions de base Radiales (RFR)

Les réseaux de neurones à fonctions de base radiales sont des réseaux de type *feedforward* (à propagation avant) avec une seule couche cachée (Figure I.8). L'utilisation de ces réseaux remonte aux années soixante-dix par (***Hardy,*** 1971), (***Agteberg,*** 1974) et (***Schagen***, 1979) pour résoudre des problèmes d'interpolation multi variables. Les bases théoriques de ces réseaux ont été ensuite approfondies par (***Powell***, 1987), (***Poggio,*** 1989) et (***Moody,*** 1989). D'autres travaux se sont succédés où l'application des *RFR* a été élargie à d'autres domaines, à savoir la prédiction de l'évolution des systèmes dynamiques (***Broomhead,*** 1988), (***Casdagli***, 1989) et la classification de phonèmes (***Renals,*** 1989). La particularité de ces réseaux réside dans le fait qu'ils sont capables de fournir une représentation locale de l'espace grâce à des fonctions de base radiales $\phi(\| \|)$ dont l'influence est restreinte à une certaine zone de cet espace. $\| \|$ représente la norme euclidienne.

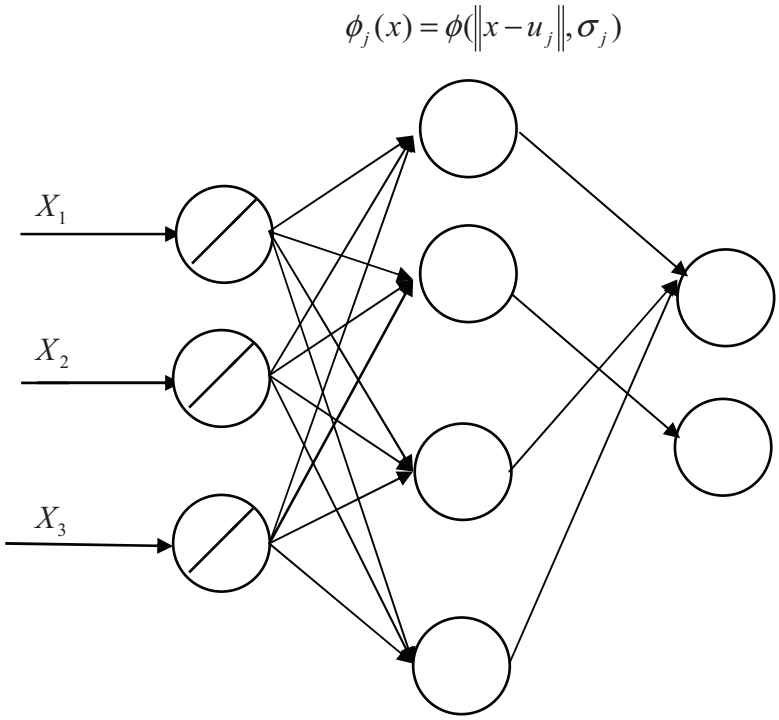


Figure I.8 : Réseau à fonctions de base radiales.

Deux paramètres sont associés à cette fonction de base : un vecteur de référence μ_j appelé centre ou *prototype* et la dimension σ_i du champ d'influence appelé *rayon d'influence*. La réponse de la fonction de base dépend donc de la distance du vecteur d'entrée X au vecteur prototype μ_j, et de la taille du champ d'influence :

$$\Phi_j(X) = \Phi_j\left(\|X - \mu_j\|, \sigma_j\right) \tag{I-15}$$

Où la fonction $\Phi_j(\|\ \|)$ est généralement maximale lorsque $X = \mu_j$ et décroît d'une façon monotone vers 0 quand $r = \|X - \mu_j\| \to \infty$. Pour qu'une fonction $\phi_j(r)$ soit utilisée comme fonction de base et résoudre le problème de l'interpolation, il faudrait qu'elle soit *définie positive* (***Micchelli***, 1986) [13].

IV. Réseaux de neurones et rejection de perturbation

Les réseaux de neurones peuvent fournir des solutions très intéressantes essentiellement pour l'identification. Les seuls travaux présentés dans (***S. Mukhopadhyay*** *et* ***K. S. Narendra***) et (***C. J. Goh*** et ***P. Podsiadlo***) traitent le

problème de la rejection de la perturbation. Après une phase d'apprentissage assez complexe, le réseau est capable de générer le modèle créé pour la rejection de la perturbation. La perturbation qui est considéré comme la sortie d'un système linéaire ou non linéaire. L'objectif principal et nécessaire est de trouver le (le plan dynamique augmenté ***ADP***) qui est nommé dans notre projet par la fonction f_{inc} par lequel la commande sera générée pour la rejection de la perturbation. Un *Perceptron* à trois couches permet d'identifier le ***ADP*** avec des résultats efficaces.

V. Conclusion

Ce chapitre a été dédié à la présentation des réseaux de neurones artificiels. Après avoir introduit leurs concepts de base, nous avons présenté les applications des réseaux de neurones, qui sont alors utilisés pour l'identification des systèmes dynamiques et pour donner un modèle sous forme d'une boîte noire nous avons particulièrement détaillé une seule architecture neuronale : *Le Perceptron Multicouches* (*PMC*) par laquelle nous avons concentré notre projet , dans le suivant chapitre on va entamer une autre méthode très efficaces aussi appelé la logique floue et en va consacrer notre étude sur l'approche de retro-propagation floue.

CHAPITRE II

LA LOGIQUE FLOUE

I. Introduction

La plupart des problèmes rencontrés sont modélisables mathématiquement. Cependant, ces modèles nécessitent des hypothèses parfois trop restrictives, rendant délicate l'application au monde réel. Les problèmes du monde réel doivent tenir compte d'informations imprécises et incertaines. Prenons l'exemple d'une climatisation, si on veut obtenir une température fraîche, on peut se demander quelle gamme de températures conviendra (la demande est imprécise). De plus la fiabilité des capteurs entre en jeu (la mesure de la température ambiante est incertaine). On voit apparaître la difficulté d'interprétation des variables linguistiques comme, frais, chaude,…ainsi que le traitement de ces données entachées d'incertitude [14].

Une approche fut développée en 1965 par ***Lofti A. Zadeh***, basée sur sa théorie des sous-ensembles flous (fuzzy sets), généralisant la théorie classique des ensembles. Dans la nouvelle théorie de *Zadeh*, un élément peut plus ou moins appartenir à un certain ensemble. Les imprécisions et les incertitudes peuvent ainsi être modélisées, et les raisonnements acquièrent une flexibilité que ne permet pas la logique classique : la Logique Floue était née [15].

Dans ce chapitre, nous développons la théorie des sous-ensembles flous. Ensuite, nous illustrons le raisonnement en logique floue. Enfin, nous présentons l'application de la logique floue dans la rejection de la perturbation.

II. Exemples introductifs

II. 1. L'age

Afin de mettre en évidence le principe fondamental de la logique floue, on présente un exemple simple, celui de la classification des personnes en trois ensembles «jeune», «entre deux âges » et « âgé ». Selon la logique classique (logique de Boole), qui n'admet pour les variables que les deux valeurs 0 et 1, une telle classification pourrait se faire comme le montre la Figure II.1. Toutes les personnes âgées de mois de 25 ans sont alors considérées des jeunes et toutes les personnes ayant plus de 50 ans comme des « âgées » (vieilles).

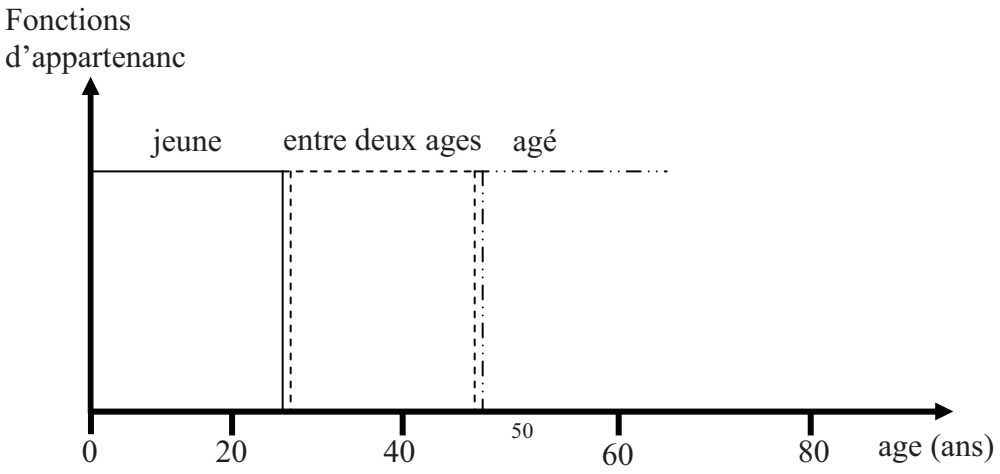


Figure II.1 : Classification des personnes en trois ensembles selon la logique classique

Cependant, une telle logique de classification n'est pas réaliste. Pourquoi une personne, lorsqu'elle a eu 50 ans, doit-elle être considéré comme appartenant à l'ensemble âgés?

Dans la logique floue, les variables peuvent prendre n'importe quelle valeur entre 0 et 1. La Figure II.2 montre une classification possible pour l'exemple précédent, cette fois-ci à l'aide de la logique floue. Ainsi, une personne de 25 ans appartient à l'ensemble «jeune» avec une valeur $\mu = 0.75$ de la fonction d'appartenance et à l'ensemble «entre deux âges» avec $\mu = 0.25$. Par contre une personne âgée de 65 ans appartient avec une valeur $\mu = 1$ de la fonction d'appartenance à l'ensemble «âgé».

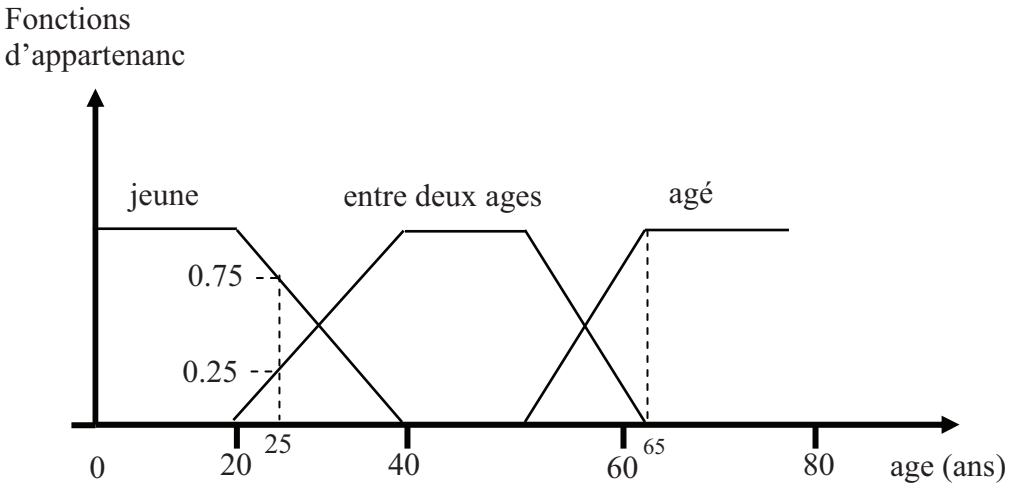


Figure II.2 : Classification des personnes en trois ensembles selon la logique floue.

Les grandeurs utilisées dans un système de réglages sont généralement générées par des capteurs. Il est nécessaire de convertir ces grandeurs en variables floues. Pour ce faire, on définit les deux notions suivantes :

- Les *fonctions d'appartenances* qui permettent de définir le degré de vérité de la variable floue en fonction de la grandeur d'entrée.
- Les *intervalles flous* qui déterminent le nombre de variables floues.

Dans l'exemple de la Figure II.2, on fait intervenir trois intervalles flous : «jeune» « entre deux age » et « âgé ». En outre, chaque intervalle fait référence à une fonction d'appartenance qui permet de définir le degré de vérité de la variable floue correspondante en fonction de la taille [16].

On peut évidemment choisir n'importe quelle forme pour les fonctions d'appartenance [17]. Cependant, en pratique, on utilise trois types de fonctions définies comme suit :

- ***La fonction triangulaire*** : Elle est caractérisée par trois paramètres (a, b, c) ; les sommets du triangle :

$$\mu(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right) \qquad \text{(II-1)}$$

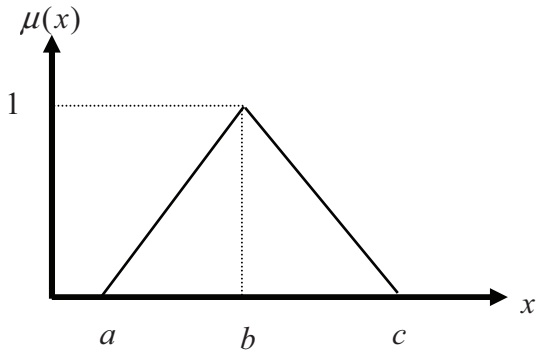


Figure II.3 : Fonction triangulaire

- ***La fonction trapézoïdale*** : définie par quatre paramètres (a, b, c, d) :

$$\mu(x) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right) \qquad \text{(II-2)}$$

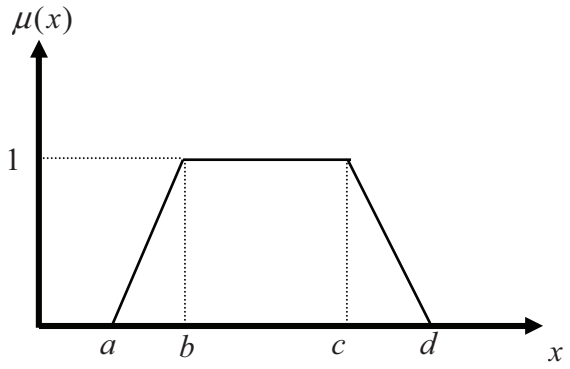


Figure II.4 : Fonction trapézoïdale

- ***La fonction Gaussienne*** : définie par c et σ le centre et l'épaisseur

$$\mu(x) = \exp\left(-(x-c)^2 / \sigma^2\right) \qquad \text{(II-3)}$$

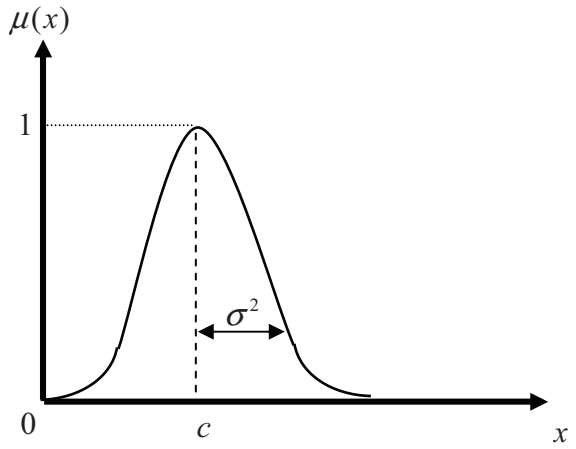


Figure II.5 : Fonction Gaussienne

Intervalles flous

Ces intervalles définissent le nombre de variables floues associées à une grandeur d'entrée. Dans le cas du réglage, trois à cinq intervalles s'avèrent suffisants. De façon générale, ils sont caractérisés à l'aide de symboles tels que ceux présentés dans le Tableau .1.

Symbole	Signification
NG	Négatif grand
NM	Négatif moyen
EZ	Négatif zéro
PM	Positif moyen
PG	Positif grand

Tableau 1 : Intervalles flous

Cas particulier : grandeur de sortie

La grandeur de sortie peut être définie à l'aide d'un certain nombre d'intervalles flous et diverses fonctions d'appartenance. Toutefois, en pratique, cette définition peut sembler assez lourde et le concepteur (l'*expert*) peut choisir d'associer une seule valeur à chaque intervalle flou. Par exemple, pour une grandeur à cinq intervalles flous, on peut définir les valeurs suivantes (Tableau 2) [18] :

Intervalle	Valeur en % du maximum
Très petit	0
Petit	25
moyen	50
grand	75
Très grand	100

Tableau 2 : Grandeur à cinq intervalles flous

III. Ensemble flou

Un ensemble flou est défini par sa fonction d'appartenance. Un point de l'univers x appartient à un ensemble A avec un degré d'appartenance, $0 \le \mu_A \le 1$. La Figure II.6 montre un ensemble de forme triangulaire [19].

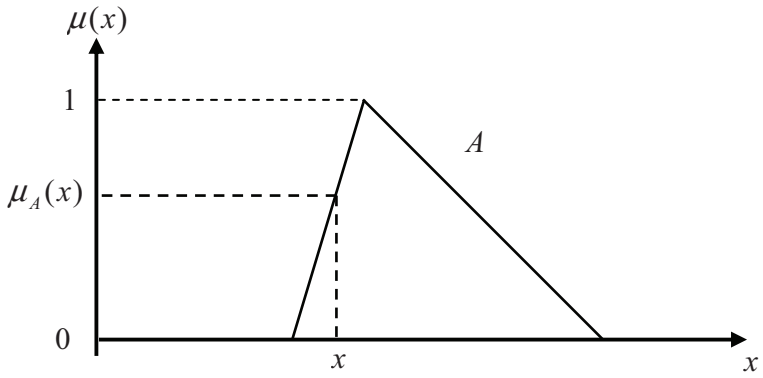


Figure II.6 : Un ensemble flou de forme triangulaire

III.1. Prototype d'un ensemble

Un point est un prototype d'un ensemble flou si son degré d'appartenance à cet ensemble vaut un [19].

III.2. Opérateurs

Les règles d'inférence font appel aux opérateurs ***et***, ***ou*** et ***non***, qui s'appliquent aux variables floues. Dans le cas de la logique binaire, ces opérateurs sont définis de façon simple et univoque. Dans le cas de la logique floue, la définition de ces opérateurs n'est plus univoque et on utilise le plus souvent les relations présentées dans le Tableau 3.

opérateur	**Opération sur le degré de vérité des variables**
Et	Minimum
	Produit
Ou	Maximum
	Valeur moyenne
non	Complément à 1

Tableau 3 : Opérateur binaire et flou

Les opérations ***minimum*** et ***maximum*** présentent l'avantage de la simplicité lors du calcul. Par contre, elles privilégient l'une des deux variables. Les opérations de ***produit***

et ***valeurs moyennes*** sont plus complexes à calculer mais elles produisent un résultat qui tient compte des valeurs des deux variables [20].

IV. Partitionnement

Le découpage du domaine de définition d'une variable en sous-ensembles flous est appelé partitionnement. Ces ensembles sont notés A_1 , A_2 [19].

IV.1. Partition flou forte

La partition de la variable X_i sera appelée une partition floue forte si : $\forall\, x \in X_i$, $\sum_j \mu_{A_j^i}(x) = 1$ [19].

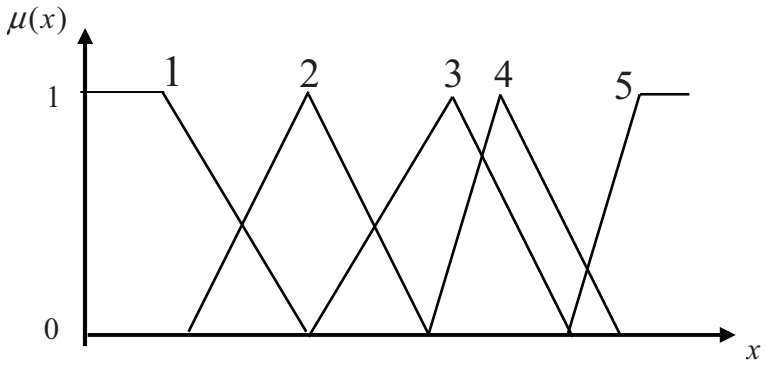


Figure II.7 : Exemple de partition floue forte

V. Règle floue

Une règle floue est de la forme ***SI*** *je rencontre telle situation* ***ALORS*** *j'en tire telle conclusion*. La situation, appelée prémisse ou antécédent de la règle, est définie par une combinaison de relations de la forme $x\ est\ A$ pour chacune des composantes du vecteur d'entrée. La partie conclusion de la règle est appelée conséquence, ou encore simplement conclusion [19].

V.1. Types de règles

On distingue deux types de règles floues

- **Mamdani**

Une règle floue de type *Mamdani,* dont la conclusion est un ensemble flou s'écrit : $SI\ x_1\ est\ A_1^i\ ET \ldots ET\ x_p\ est\ A_p^i\ ALORS\ y_1\ est\ C_1^i \ldots ET\ y_q\ est\ C_q^i$ où $A_j^i\ et\ C_j^i$ sont

des ensembles flous qui définissent le partitionnement des espaces d'entrée et de sortie [19].

- **Takagi-Sugeno**

Dans le modèle de *Sugeno* la conclusion de la règle est nette. Celle de la règle *i* pour la sortie *j* est calculée comme une fonction linéaire des entrées :

$y_j^i = p_{j0}^i + p_{j1}^i x_1 + p_{j2}^i x_2 + ... + p_{jp}^i x_p$, notée aussi : $y_j^i = f_j^i(x)$

La valeur numérique de la sortie de ce modèle est donnée par :

$$y = \frac{\sum_{l=1}^{M} \omega_l y_l}{\sum_{l=1}^{M} \omega_l} \qquad \text{(II-4)}$$

où ω_l est le degré de vérité du lème règle floue $SI - ALORS$ qui peut être obtenu par :

$$w_i = \mu_{A_1^i}(x_1) \wedge ... \wedge \mu_{A_p^i}(x_p) \qquad \text{(II-5)}$$

où $\mu_{A_j^i}(x_j)$ est le degré d'appartenance de la valeur x_j à l'ensemble flou A_j^i [19].

Remarque : *Pour une règle donnée i, son degré de vérité pour un exemple, également appelé poids, et noté w_i, résulte d'une opération de conjonction des éléments de la prémisse .*

Ou bien :

$$\omega_l = \min\left\{\mu_{A1}(x_1), \mu_{A2}(x_2), ..., \mu_{An}(x_n)\right\} \qquad \text{(II-6)}$$

Lorsque $p_{lj} = 0$ pour *j* = 1, 2,…, *n*, le système est dit « Modèle flou TS d'ordre zéro » ou système flou à conséquence singleton. Ainsi la valeur numérique de la sortie de ce modèle sera donnée par :

$$y = \frac{\sum_{l=1}^{M} \omega_l b_l}{\sum_{l=1}^{M} \omega_l} \tag{II-7}$$

Pour un choix de fonction d'appartenance Gaussienne de la forme :

$$\mu_{A_i^l}(x_i) = \exp\left(-\left(\left(x_i - c_i^l\right)/\sigma_i^l\right)^2\right) \tag{II-8}$$

Et en utilisant l'équation II-7, la sortie du système flou sera donnée par :

$$y = \hat{f}(\underline{x}) = \frac{\sum_{l=1}^{M} b_l \prod_{i=1}^{n} \exp\left(-\left(\left(x_i - c_i^l\right)/\sigma_i^l\right)^2\right)}{\sum_{l=1}^{M} \prod_{i=1}^{n} \exp\left(-\left(\left(x_i - c_i^l\right)/\sigma_i^l\right)^2\right)} \tag{II-9}$$

Pour toutes les simulations nous allons utiliser le modèle flou de TS d'ordre 0 (modèle singleton) dont la sortie est définie par l'équation II-9 pour la modélisation des systèmes non linéaires [19].

V.2. Règle incomplète

Une règle floue sera dite incomplète si sa prémisse est définie par un sous-ensemble des variables d'entrée seulement. La règle, « *SI* x_2 *est* A_2^1 *ALORS y est* C_2 » est incomplète car la variable x_1 n'intervient pas dans sa définition. Les règles formulées par les experts sont principalement des règles incomplètes. Formellement, une règle incomplète est définie par une combinaison implicite de connecteurs logiques *ET* et *OU* opérant sur l'ensemble des variables. Si l'univers de la variable x_1 est découpé en 3 sous-ensembles flous, la règle incomplète ci-dessus peut aussi s'écrire de la façon suivante : $SI\,(x_1\ est\ A_1^1\ OU\ x_1\ est\ A_1^2\ OU\ x_1\ est\ A_1^3)\ ET\ x_2\ est\ A_2^1\ ALORS\ y\ est\ C_2$ [19].

VI. Interface de fuzzification

Les opérateurs utilisés dans la commande floue agissent sur des sous-ensembles flous. Par conséquent, il est nécessaire de transformer les variables non floues provenant du mode extérieur en des sous-ensembles flous. Pour se faire, on utilise un opérateur dit

de fuzzification qui associe à une mesure de la variable x_0 une fonction d'appartenance particulière $\mu_{x0}(x)$.

Le choix de l'opérateur de fuzzification dépend de la confiance que l'on accorde aux mesures effectuées. Ainsi si la mesure x_0 est exacte, le sous-ensemble flou A_0 doit être représentée par un fait précis. Par conséquent, on utilise comme opérateur de fuzzification la transformation singleton. La fonction d'appartenance du sous-ensemble flous A_0 est alors définie par:

$$\mu_{x0} : U \to U, \mu_{x0}(x) = 1 \ Si \ x = x_0; \mu_{x0}(x) = 0 \ Si \ x \neq x_0$$

La Figure .II-8 montre l'aspect de cette fonction d'appartenance.

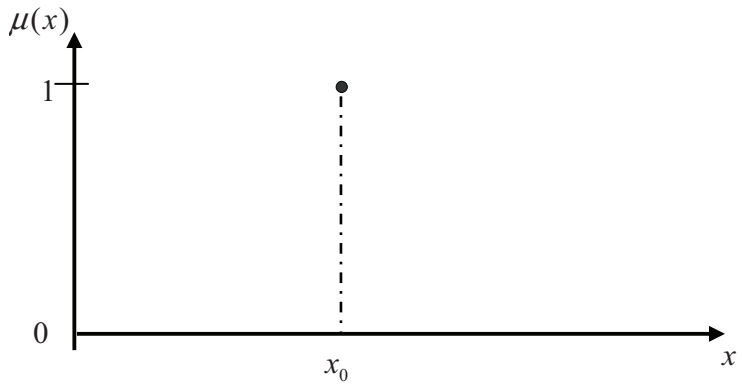


Figure.II-8: Méthode de fuzzification pour une mesure exacte

Ainsi, le sous ensemble floues A_0 réalisé par cette méthode de fuzzification ne comprend que l'élément x_0.

Par contre, si la mesure de la variable est incertaine, par exemple à cause de bruit, le sous-ensemble flou A_0 doit être représentée par un fait imprécis. On utilise alors une méthode de fuzzification qui associe à la variable mesurée x_0 une fonction d'appartenance telle que, par exemple:

$$\mu_{x0}(x) = \max\left\{0;1-\frac{|x-x_0|}{\varepsilon}\right\}$$

La représentation graphique de cette fonction est représentée par La Figure .II-9. Ce sous-ensemble flou comprend donc la mesure x_0 avec une appartenance unité et les valeurs voisines de x_0 avec une appartenance inversement proportionnelle à l'écart avec x_0.

La base du triangle (ε) est fonction de l'importance relative des erreurs de mesures. En effet, plus elles sont importantes, plus la mesure de la variable x_0 devient imprécise, et donc, plus le triangle doit s'élargir [21]-[22].

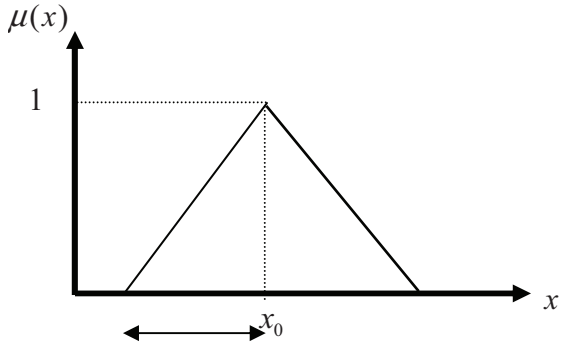


Figure. II-9: Méthode de fuzzification pour une mesure incertaine

VII. Déffuzification

Les valeurs obtenues lors de la combinaison des règles appliquées aux intervalles flous de la variable de sortie défini une fonction d'appartenance. Il s'agit de convertir cette information en une grandeur physique. Plusieurs façons de faire peuvent être envisagées mais, en pratique, on utilise surtout les deux méthodes suivantes :

- *Défuzzification par calcul du centre de gravité.*
- *Défuzzification par calcul du maximum* [18].

VII.1. Déffuzification par calcul du centre de gravité

Il s'agit de calculer le centre de gravité de la fonction d'appartenance de la variable de sortie :

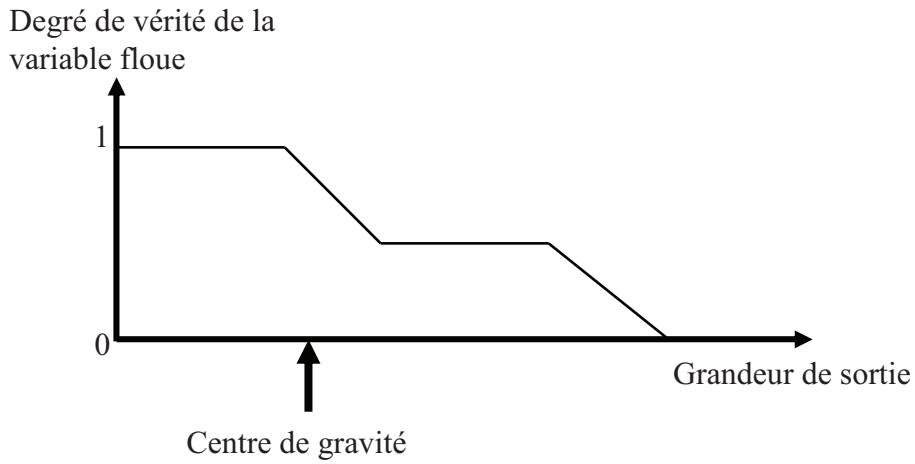


Figure.II-10 : Centre de gravité de grandeur de sortie

Le calcul du centre de gravité présenté dans la Figure. II-10 permet d'obtenir une seule valeur pour la grandeur de sortie. Son calcul est, cependant, relativement complexe puisqu'il nécessite le calcul d'une intégrale, ou dans le cas simple de fonctions d'appartenance en raies, d'une somme pondérée [18].

Le défuzzificateur détermine le centre de gravité $\overline{y}$ de B et utilise cette valeur comme sortie du FLS (Fuzzy Logic System) [18]. Le centre de gravité est donné par :

$$\overline{y} = \frac{\int_S y\, \mu_B(y)\, dy}{\int_S \mu_B(y)\, dy} \qquad \text{(II-10)}$$

Où S représente le support de$B$. Lorsque S est discrétisé, nous obtenons :

$$\overline{y} = \frac{\sum_{i=1}^{L} y_i \mu_B(y_i)}{\sum_{i=1}^{L} \mu_B(y_i)} \qquad \text{(II-11)}$$

VII.2. Déffuzification par calcul du maximum

Il s'agit de la façon la plus simple, du point de vue du volume de calcul, pour effectuer la défuzzification. Tout d'abord, la grandeur de sortie doit être normalisée (en pour-cent par exemple) et la définition des intervalles flous doit se résumer à une valeur : par exemple « petit » correspond à 0 et « moyen » à 0,5 (fonctions

d'appartenance en forme de raies). L'opération de défuzzification consiste à prendre d'abord le minimum entre la valeur produite par la règle concernée et la valeur de la variable floue de sortie. La valeur de sortie est définie par la valeur maximale des variables floues de sortie [18].

VIII. Algorithme de rétro-propagation à l'identification floue

Soit un système de paire entrée-sortie donnée par :

$(\underline{x}^p, d^p), \underline{x}^p \in U \subset R^n, d^p \in V \subset R$; notre tâche est de déterminer un système logique flou $f(\underline{x})$ dans la forme de l'équation II.3, et d'ajuster les paramètres b^l, c_i^l, σ_i^l de telle sorte que le critère :

$$J = \frac{1}{2}\left(d - \hat{f}(\underline{x})\right)^2 \tag{II-12}$$

soit minimisé, afin de déterminer un système flou $\hat{f}(x)$ sous la forme de l'équation II.7 pour un couple d'entrée-sortie (x, d) donné [23].

Pour l'ajustement des paramètres b^l selon l'algorithme d'adaptation paramétrique:

$$b^l(k+1) = b^l(k) - \alpha \left.\frac{\partial J}{\partial b^l}\right|_k \tag{II-13}$$

où $l = 1,2, \ldots, M$, $k = 0,1,2, \ldots,$ ($\hat{f}(\underline{x})$ sera noté par $\hat{f}$).

D'après la Figure. II.11, on déduit que $\hat{f}$ (et par conséquent J) dépend seulement de b^l à travers A, où $\hat{f} = A/B$, $A = \sum_{l=1}^{M} b^l \omega_l$, $B = \sum_{l=1}^{M} \omega_l$, tel que ω_l est donné par l'équation II-5 :

Alors, en dérivant J par rapport à b^l on aura :

$$\frac{\partial J}{\partial b^l} = -\left(d - \hat{f}\right)\frac{\partial \hat{f}}{\partial A}\frac{\partial A}{\partial b^l} = -\left(d - \hat{f}\right)\frac{1}{B}\omega_l \tag{II-14}$$

D'où :

$$\frac{\partial J}{\partial b^l} = -\frac{e}{B}\omega_l \tag{II-15}$$

Substituant (II-14) dans (II-13), on obtient l'algorithme d'ajustement des b^l comme suit :

$$b^l(k+1) = b^l(k) + \alpha \frac{e}{B}\omega_l \tag{II-16}$$

où $l = 1,2,\ldots,M$, $k = 0,1,2,\ \ldots$

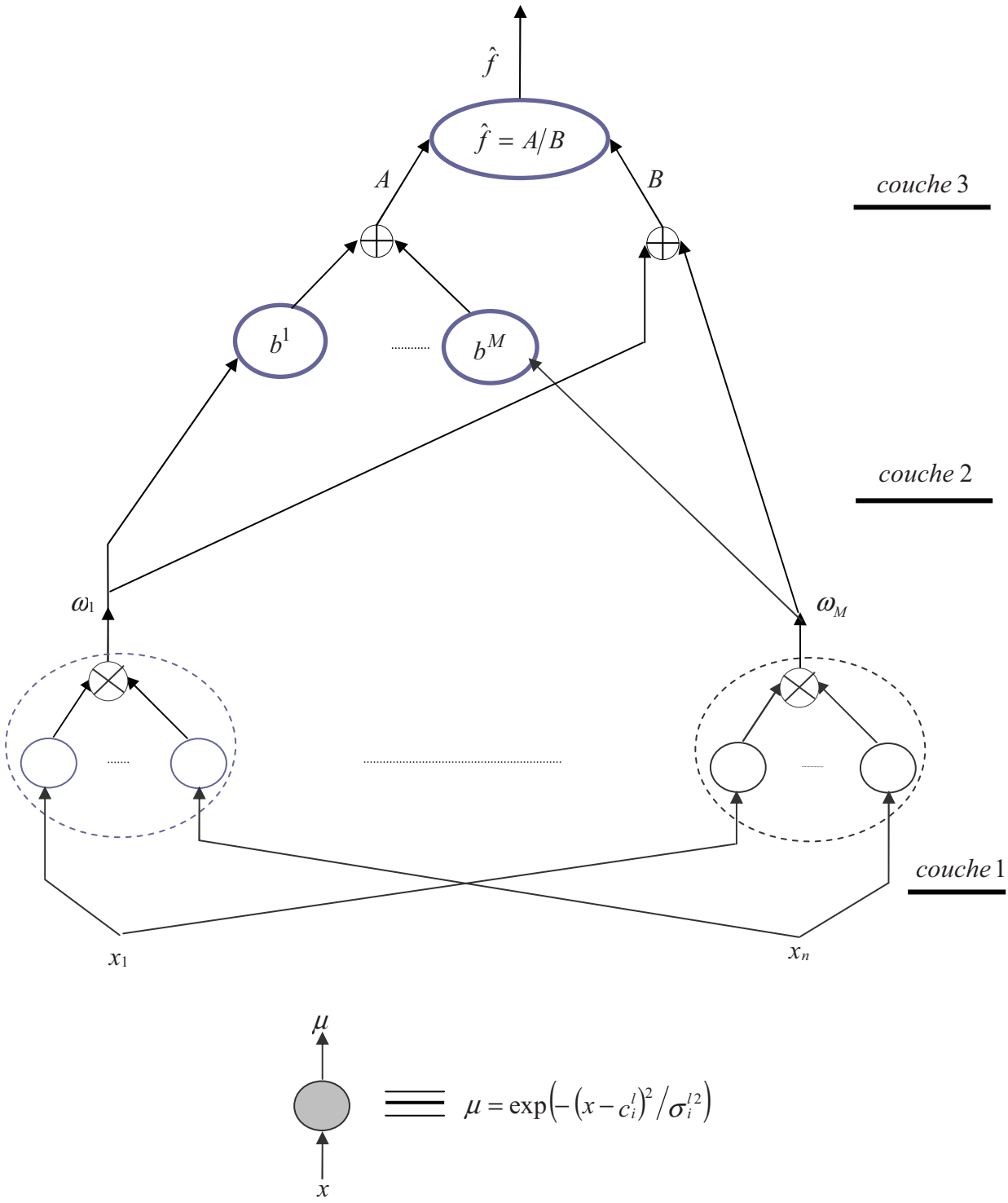


Figure. II.11 : Représentation en couches d'un système flou

Pour l'ajustement des paramètres c_i^l on a :

$$c_i^l(k+1) = c_i^l(k) - \alpha \frac{\partial J}{\partial c_i^l}\Big|_k \tag{II-17}$$

où $l = 1,2, \ldots, M$, $k = 0,1,2, \ldots$, $i = 1,2, \ldots, n$

D'après la Figure. II.11, on remarque que $\hat{f}$ (et par conséquent J) dépend seulement de c_i^l à travers ω_l , donc :

$$\frac{\partial J}{\partial c_i^l} = -\left(d - \hat{f}\right) \frac{\partial \hat{f}}{\partial \omega_l} \frac{\partial \omega_l}{\partial c_i^l} \tag{II-18}$$

Nous avons :

$$\frac{\partial \hat{f}}{\partial \omega_l} = \frac{B\left(\partial A/\partial \omega_l\right) - A\left(\partial B/\partial \omega_l\right)}{B^2} = \frac{\left(\partial A/\partial \omega_l\right) - \left(A/B\right)\left(\partial B/\partial \omega_l\right)}{B}$$

$$\frac{\partial \hat{f}}{\partial \omega_l} = \frac{b^l - \hat{f}}{B} \tag{II-19}$$

Et nous avons encore :

$$\frac{\partial \omega_l}{\partial c_i^l} = \frac{2\left(x_i - c_i^l\right)}{\sigma_i^{l2}} \omega_l \tag{II-20}$$

Par substitution de (II-19) et (II-20) dans (II-18) on aura :

$$\frac{\partial J}{\partial c_i^l} = \frac{-\left(d - \hat{f}\right)}{B} b^l - \hat{f} \frac{2\left(x_i - c_i^l\right)}{\sigma_i^{l2}} \omega_l \tag{II-21}$$

On obtient alors l'algorithme d'ajustement des paramètres c_i^l :

$$c_i^l(k+1) = c_i^l(k) + \alpha \frac{\left(d - \hat{f}\right)}{B} (b^l - \hat{f}) \frac{2\left(x_i - c_i^l\right)}{\sigma_i^{l2}} \omega_l \tag{II-22}$$

En utilisant la même méthode pour les écarts types σ_i^l des gaussiennes (largeurs), on obtient l'algorithme d'ajustement suivant :

$$\sigma_i^l(k+1) = \sigma_i^l(k) + \alpha \frac{e}{B}(b^l - \hat{f}) \frac{2\left(x_i - c_i^l\right)^2}{\sigma_i^{l3}} \omega_l \qquad \text{(II-23)}$$

L'algorithme d'apprentissage (II-16), (II-22) et (II-23) exécutent une procédure de rétro-propagation d'erreur [23].

Pour l'ajustement des paramètres b^l, l'erreur normalisée $\frac{e}{B}$ est rétro propagée vers la couche b^l ensuite ce dernier est mis à jours en utilisant (II-16) où ω_l est l'entrée de b^l (comme c'est montré en Figure II. 11).

Pour l'ajustement des paramètres c_i^l et σ_i^l, l'erreur normalisée $\frac{e}{B}$ ajuste $b^l - \hat{f}$ ensuite ω_l est rétro propagé vers l'unité de traitement de la couche l dont la sortie est ω_l ; puis c_i^l et σ_i^l seront mis à jours par (II-22) et (II-23).

Pour l'identification de tous les systèmes nous avons utilisé dans notre mémoire, le modèle flou décrit par (II-9) avec :

- ✓ La fuzzification singleton.
- ✓ Nombre de règles floues sont 20 parfois 40, et pour chaque entrée on ajuste deux paramètres des fonctions d'appartenance d'entrée correspondante et de sortie (gaussienne, singleton respectivement) qui sont le centre de la partie prémisse et le centre de la partie conséquence.
- ✓ Défuzzification par la méthode de la moyenne floue (fuzzy mean) [17].

IX. La logique floue et la rejection de perturbation

L'objectif principal et nécessaire est de trouver (le plant dynamique augmenté) qui est appelé dans notre mémoire par la fonction f_{inc} par lequel la commande floue sera généré pour la rejection de la perturbation, l'approche de rétro-propagation floue à trois couches permet d'identifier le f_{inc} avec des résultats efficaces.

X. Conclusion

Dans ce chapitre le système flou entraînable que nous appelons “Back-Propagation (rétro-propagation) de Système flou (BP FS),” été utilisé comme un identificateur pour les systèmes dynamique non linéaires. Il a été prouvé que Le BP FS est capable d’estimer toute vraie fonction continue non linéaire. En utilisant le fait que les systèmes flous peuvent être représentés comme un réseau à trois couches, l'algorithme de back-propagation a été développé pour former le système flou pour identifier des paires entrée-sortie désirées. Dans le prochain chapitre on va discuter la relation entre l’identification et la rejection de la perturbation.

CHAPITRE III

IDENTIFICATION ET REJECTION DES PERTURBATIONS DANS LES SYSTÉMES DYNAMIQUES

I. Introduction

L'identification et la commande des systèmes non linéaires ne sont pas des taches triviales à cause de la non linéarité qui les caractérise. La commande et l'identification doivent être appliqué par un système capable de parier à ce problème, La simulation étendue a montré qu'avec quelque information antérieure à propos d'un système non linéaire inconnu peut être contrôlé, tels identificateurs et contrôleurs résultent en performance satisfaisante [6].

Pour une classe spécifique de système non linéaires, il a été montré que les lois adaptatives donnent la stabilité globale du système total [10]. Dans ce chapitre, le problème de la rejection de la perturbation est traité, nous illustrons qu'il existe une grande relation entre la rejection de la perturbation et l'identification et la commande des systèmes non linéaires sachant que la perturbation considérées comme la sortie d'une équation différentielle linéaire ou non linéaire libre.

II. Étape de modélisation

II.1.Caractérisation

La première étape du processus de modélisation consiste à faire une hypothèse sur la structure du système, c'est-à-dire à choisir un type de relation mathématique f liant les entrées et les sorties. Les paramètres structuraux, au départ inconnus, seront déterminés numériquement dans l'étape suivante dite l'identification.

Dans ce choix de structure, on peut être guidé par :

- Une analyse physique conventionnelle du processus.
- Les expériences et le résultat qualitatif de tests simples.
- Des contraintes de calcul ou des contraintes économiques….

Cette étape, essentiellement qualitative, n'est éclairée et validée que par le reste de la procédure. C'est souvent la plus difficile ; elle fait appel à l'expérience [24].

II.2.Identification

L'identification de système est le processus de développer un modèle mathématique d'un système dynamique basé sur les données d'entrée et de rendement du processus réel. Ceci

signifie qu'il est possible de prélever l'entrée et les signaux de sortie d'un système et d'employer ces données pour produire un modèle mathématique.

Une étape importante dans la conception de système de commande est le développement d'un modèle mathématique du système à commander. Afin de développer un contrôleur, il faut qu'il soit possible d'analyser le système à commander. Cela est accompli en utilisant un modèle mathématique. Un autre avantage d'identification de système est évident si le processus est changé ou modifié. L'identification de système permet au vrai système d'être changé sans devoir calculer les équations dynamiques et modeler les paramètres une deuxième fois.

L'identification de système est concernée par les modèles développés. La Figure III.1 ci-dessous montre les entrées et le rendement d'un système.

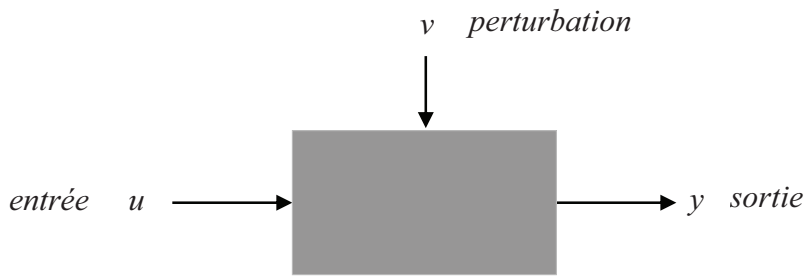


Figure III.1 : Système montrant l'entrée, la perturbation et les signaux de sortie

Dans cet exemple, le modèle mathématique est la boîte noire. Il décrit le rapport entre l'entrée et les signaux de sortie. Les systèmes présentés dans notre travail sont des processus non linéaires. Des méthodes non linéaires employant les réseaux neurologiques et la logiques floue, les algorithmes génétiques,…, peuvent être employées. Les études dans l'identification de système ont démontré que plusieurs méthodes sont réussies en modélisant beaucoup de systèmes non linéaires parmi eux les réseaux de neurone et la logique floue [25].

- **Processus non linéaires**

L'identification d'un processus consiste à déterminer un modèle. La construction de ce modèle est effectuée en deux étapes :

a. Étape qualitative

Elle consiste à fixer les formes des équations qui décrivent le processus. On propose les modèles suivants :

- ✓ **Modèle 1**

$$y(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-i) + g\left[u(k), u(k-1), ..., u(k-m+1)\right] \quad \text{(III-1)}$$

- ✓ **Modèle 2**

$$y(k+1) = f\left[y(k), y(k-1), ..., y(k-n+1)\right] + \sum_{i=0}^{m-1} \beta_i u(k-i) \quad \text{(III-2)}$$

- ✓ **Modèle 3**

$$y(k+1) = f\left[y(k), y(k-1), ..., y(k-n+1)\right] + g\left[u(k), u(k-1), ..., u(k-m+1)\right] \quad \text{(III-3)}$$

- ✓ **Modèle 4**

$$y(k+1) = f\left[y(k), y(k-1), ..., y(k-n+1); u(k), u(k-1), ..., u(k-m+1)\right] \quad \text{(III-4)}$$

où $u(k)$ est l'entrée du système à identifier et $y(k)$ sa sortie, les fonctions f et g sont supposées dérivables par rapport a leurs arguments [17].

b. Étape quantitative

Elle consiste à déterminer les valeurs numériques des paramètres (poids des connexions) qui interviennent dans le modèle d'identification. On doit définir un critère qui exprime quantitativement l'écart entre le système et le modèle. Ce critère devra être minimisé [26].

III. La procédure d'identification de système

Fondamentalement, l'identification de système est réalisée en ajustant les paramètres du modèle jusqu'à ce que le rendement du modèle soit semblable au rendement du vrai système. Ci-dessous, au Figure III-2, un organigramme expliquant le procédé d'identification de système [25].

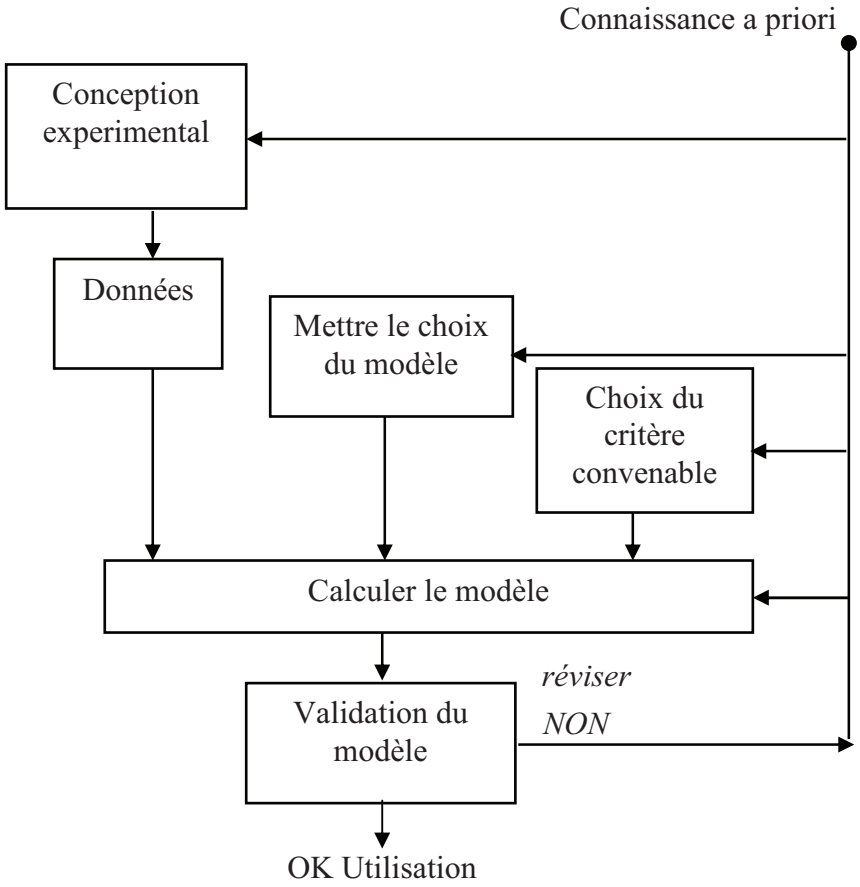


Figure III. 2 : Organigramme du procédé d'identification du système

Il y a trois étapes principales du procédé d'identification d'un système.

1) Produire des données expérimentales d'entrée-sortie du processus.
2) Choisir une structure modèle pour employer. Par exemple le modèle suivant correspond à la structure ARX

$$A.y(t) = Bu(t) + v(t)$$

3) les paramètres A et B seront ajustés jusqu'à ce que le rendement modèle soit semblable au rendement du processus. Dans l'identification, il n'y a aucune structure modèle parfaite à employer. Les modèles peuvent être développés en utilisant l'intuition de technologie ou *la connaissance à priori* du processus [25].

IV. Méthodes indirectes

IV.1.Identification de processus

Nous entamons cette classification par une présentation des méthodes d'identification de processus. Il ne s'agit pas à proprement parler de méthodes de commande, mais plutôt d'une première phase nécessaire dans les approches indirectes. Par identification d'un processus, nous entendons l'entraînement d'un réseau, à reproduire une fonction donnant les sorties ou l'état du processus à partir des entrées qui lui sont appliquées.

Le principe général de l'identification est simple et consiste à placer en parallèle le modèle d'identification et le processus à identifier, comme le montre la Figure III.3. Le modèle d'identification reçoit en entrée la commande $u(k)$ appliquée et éventuellement la sortie $y(k-1)$ précédente du processus. Il est entraîné à produire la nouvelle sortie (ou le nouvel état) $y(k)$ du processus, cette méthode d'identification est souvent appelée méthode série-paralléle. Elle est aussi souvent considérée comme plus stable car le modèle d'identification est régulièrement ((recalculé)) en utilisant l'état réel du processus, donné par :

$$\hat{y}(k+1) = \hat{f}\left(y(k), y(k-1), \ldots, y(k-n+1); u(k), u(k-1), \ldots, u(k-m+1)\right) \qquad \text{(III-5)}$$

Par opposition à la méthode parallèle présentée au Figure III.4. Dans cette dernière méthode, le modèle d'identification ne reçoit pas en entrée la sortie réelle $y(k-1)$ du processus mais la sortie $\hat{y}(k-1)$ qu'il a lui-même prédite au pas de temps précède.

Cette approche sera d'intérêt si la boucle donnant l'état du système est remplacée par une connexion récurrente du modèle d'identification. Ce dernier est alors libre de développer sa propre représentation de l'espace des états du processus. Elle est donnée par [17]:

$$\hat{y}(k+1) = \hat{f}\left(\hat{y}(k), \hat{y}(k-1), \ldots, \hat{y}(k-n+1); u(k), u(k-1), \ldots, u(k-m+1)\right) \qquad \text{(III-6)}$$

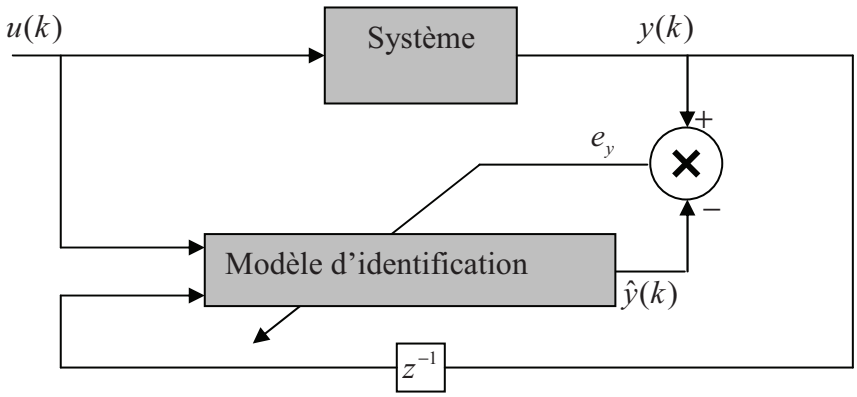


Figure III.3 : Identification de processus par la méthode série-paralléle.

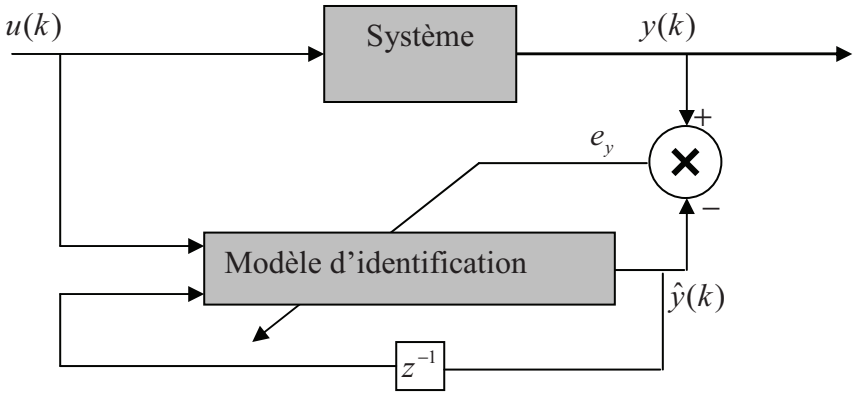


Figure III.4 : Identification de processus par la méthode parallèle.

V.Rejection de la perturbation

V.1.Problématique

Soit un système dans l'absence d'une perturbation externe :

$$x_p(k+1) = f_1\left[x_p(k), u(k)\right]$$

$$y(k) = h_1\left[x_p(k)\right] \qquad \text{(III-7)}$$

où $x_p(k) \in \Re^n$ est l'état du système à l'instant k, $u(k) \in \Re$ est l'entrée appliquée à l'instant k et $y(k) \in \Re$ est la sortie observée à l'instant k. On peut décrire le système par l'équation aux différences suivante :

$$y(k+1) = F_1[Y(k), U(k)] \tag{III-8}$$

où $Y(k) \in \Re^n = [y(k), y(k-1), ...y(k-n+1)]^T$ et $U(k) \in \Re^n = [u(k), u(k-1), ..., u(k-n+1)]^T$

La perturbation externe est représentée par l'équation suivante :

$$x_v(k+1) = f_2[x_v(k)]$$

$$v(k) = h_2[x_v(k)] \tag{III-9}$$

où $x_v(k) \in \Re^p$ est l'état de la perturbation généré de système et $v(k) \in \Re$ est la sortie correspondante. Désormais $v(k)$ c'est la perturbation considéré comme la sortie de système dynamique libre, telle que cette perturbation donnée par l'équation (III-9) soit représenté par une forme autorégressive décrite par :

$$v(k+1) = F_2[\overline{V}(k)]$$

où $$\overline{V}(k) = [v(k), v(k-1), ..., v(k-p+1)]^T$$

Si $v(k)$ est considéré aussi comme entée externe du système donnée par l'équation (III-7) alors le système global peut être représenté par :

$$x_p(k+1) = f_3[x_p(k), u(k), v(k)]$$

$$x_v(k+1) = f_2[x_v(k)]$$

$$v(k) = h_2[x_v(k)]$$

$$y(k) = h_1[x_p(k)] \tag{III-10}$$

Une représentation équivalente de III-11 est :

$$x_0(k+1) = f_4[x_0(k), u(k)]$$

$$y(k) = h_1\left[x_p(k) = h_4\left[x_0(k)\right]\right] \tag{III-11}$$

où $x_0 = \left[x_p^T, x_v^T\right] \in \Re^{n+p}$ est l'état de tout le système, telle que l'entrée et la sortie totales du système sont les mêmes que la perturbation libre.

L'objectif principal est de suivre la sortie d'un modèle de référence. Le dernier est représenté par la paire entrée-sortie $\{r(k), y_m(k)\}$, où $y_m(k)$ représente la sortie désirée. En d'autres termes, l'objectif théorique est de déterminer la commande d'entrée afin que l'erreur $e_1(k) \stackrel{\Delta}{=} y(k) - y^*(k)$ soit bornée et tend asymptotiquement vers zéro. Cependant, dans de nombreux cas pratiques, cependant, nous ne pouvons que d'essayer de minimiser l'erreur au sens indiqué dans [3].

Le problème de la rejection de la perturbation, a été étudié largement dans le cas linéaire. Le problème général de cas non linéaire, comme donné dans l'équation (III-10) et (III-11) (aussi bien que dans (III-13) de la forme entrée-sortie) est un problème complexe. Pour cela, les méthodes développées pour les systèmes linéaires seront étendues aux tels cas [3].

Dans ce mémoire, afin de considérer toutes les formes possibles des systèmes, nous utilisons les modèles donnés ci-dessous :

❖ **Étape 1**

$$y(k+1) = f\left[y(k),...,y(k-n+1)\right] + \sum_{j=0}^{n-1} \mathrm{B}_j\left[u(k-j) + v(k-j)\right]$$

$$v(k+1) = \sum_{i=0}^{p-1} \lambda_i v(k-i) \tag{III-12}$$

❖ **Étape 2**

$$y(k+1) = f\left[y(k),...,y(k-n+1)\right] + \mathrm{B}_0\left[u(k) + v(k)\right]$$

$$v(k+1) = g\left[v(k),...,v(k-p+1)\right] \tag{III-13}$$

❖ **Étape 3**

$$y(k+1) = \sum_{i=0}^{n} \alpha_i f_i[y(k),...,y(k-n+1)] + \sum_{j=0}^{n-1} \mathrm{B}_j[u(k-j) + v(k-j)]$$

$$v(k+1) = \sum_{i=0}^{p-1} \lambda_i v(k-i) \qquad \text{(III-14)}$$

Dans la présentation donnée dans l'étape 1, la fonction f est inconnue, et après la rejection de la perturbation elle devient augmenté, dans la simulation nous l'avons appelé f_{inc} ou (plant dynamique augmenté ADP) et doit être estimé. La logique floue Fs et les réseaux neuronaux (*MNN*) peuvent être utilisés pour le modèle de l'identification, depuis que les paramètres ajustables contribuent linéairement à la sortie [1].

Un cas spécial où la perturbation peut être décrit par un modèle non linéaire est considéré dans l'étape 2**.** Dans ce cas, la sortie $y(k+1)$ dépend explicitement de $u(k)$ seulement et pas sur ses valeurs passées.

Finalement, dans l'étape 3**,** le problème général de la rejection de la perturbation est considéré où la sortie $y(k+1)$ dépend explicitement de $u(k)$ et de ses valeurs passées. La perturbation linéaire est encore supposée additive [3].

Le trois différentes étapes sont considérées en détail dans la section suivante :

➢ ***Problème de la rejection de la perturbation***

Quand une perturbation externe est présente, le système total est décrit par les équations suivantes :

$$y(k+1) = \sum_{i=0}^{n} \alpha_i f_i[Y(k)] + D(z)[u(k) + v(k)]$$

$$v(k+1) = R(z)v(k) \qquad \text{(III-15)}$$

***NB** : La perturbation doit être dans la forme auto régressif :*

$$v(k+1) = \lambda_1 v(k) + \lambda_2 v(k-1)$$

où

$$R(z) = \lambda_0 + \lambda_1 z^{-1} + ... + \lambda_{p-1} z^{-(p-1)}$$

Cela implique que la perturbation est générée comme la sortie d'équation différentielle. Il est supposé que la perturbation est borné c'est à dire que le polynôme:

$R_1(z) = 1 - z^{-1}R(z)$ est stable.

De l'équation (III-15), on tire

$$D(z)v(k) = y(k+1) - \sum_{i=1}^{N} \alpha_i f_i[Y(k)] - D(z)u(k), \forall k$$

d'où on déduit :

$$\begin{aligned} D(z)v(k) &= D(z)R(z)v(k-1) \\ &= \lambda_0 \left\{ y(k) - \sum_{i=1}^{N} \alpha_1 f_i[Y(k-1)] - D(z)u(k-1) \right\} + ... \\ &+ \lambda_{p-1} \left\{ y(k-p+1) - \sum_{i=1}^{N} \alpha_i f_i[Y(k-p)] - D(z)u(k-p) \right\} \end{aligned} \quad \text{(III-16)}$$

La sortie du système donnée par (III-15) peut alors s'écrire comme suit :

$$y(k+1) = \sum_{i}^{\bar{N}} \bar{\alpha}_i \bar{f}_i[\bar{Y}(k)] + R(z)y(k) + \bar{D}(z)u(k)$$

$$y(k+1) = \sum_{i}^{\bar{\bar{N}}} \bar{\bar{\alpha}}_i \bar{\bar{f}}_i[\bar{Y}(k)] + \bar{D}(z)u(k) \quad \text{(III-17)}$$

Où le système (III-17) devient un plant dynamique augmenté (ADP) par le degré (p) telle que :

$$\bar{Y}(k) = [y(k), y(k-1), ..., y(k-p-n+1)]^T, \quad \bar{N} = (p+1)N, \quad \bar{\bar{N}} = (p+1)N + p$$

$\bar{D}(z) = \left[\bar{\beta}_0 + \bar{\beta}_1 z^{-1} + ... + \bar{\beta}_{n+p-1} z^{-(n+p-1)}\right].D(z)$ doit être stable. $\bar{\beta}_0 = \beta_0$. D'où, la même procédure peut être utilisée pour la commande adaptative des systèmes dans la présence des perturbations libres pour assurer que l'erreur d'asservissement tend ver zéro.

V.2. Étape 1

Un problème plus général est considéré ici, avec la sortie de système à l'instant $k+1$ dépend linéairement de l'entrée $u(k)$. En particulier, le système avec la perturbation additive, décrit par l'équation différentielle suivante, est considérée:

$$y(k+1) = f\left[Y(k)\right] + D(z)\left[u(k) + v(k)\right]$$

$$v(k+1) = R(z)v(k) \qquad \text{(III-18)}$$

où $u(k)$ et $y(k)$ sont, respectivement, l'entrée et la sortie du système à l'instant k. $Y(k), D(z)$ et $R(z)$ sont définis précédemment. La non linéarité f du système est supposée inconnue. Cependant, tant que les nombres entiers n et p et les coefficients $\{\beta_i\}_{i=\overline{1,n-1}}$ et $\{\lambda_i\}_{i=\overline{0,p-1}}$ sont supposés connus.

Dans ce cas, le système composé peut être décrit par une représentation entrée-sortie en éliminant $v(k)$ de (III-18).

Noter cela:

$$D(z)v(k-i) = y(k-i+1) - f\left[Y(k-i)\right] - D(z)u(k-i) \qquad \text{(III-19)}$$

(III-18) peut être écrit comme :

$$y(k+1) = f\left[Y(k)\right] + D(z)u(k) + \sum_{i=1}^{p} \lambda_{i-1} D(z)v(k-i) \qquad \text{(III-20)}$$

D'où la représentation entrée-sortie du système dans la présence de la perturbation devient :

$$y(k+1) = \bar{f}\left[\bar{Y}(k)\right] + \bar{D}(z)u(k) \qquad \text{(III-21)}$$

Où:

$$\bar{Y}(k) = \left[y(k), y(k-1),..., y(k-p-n+1)\right]^T$$

$$\bar{f}\left[\bar{Y}(k)\right] = f\left[Y(k)\right] + \sum_{i=1}^{p} \lambda_{i-1}(y(k-i+1) - f\left[Y(k-i)\right])$$

$$\bar{D}(z) = D(z)R_1(z) = D(z)\left[1 - z^{-1}R(z)\right] = \sum_{i=0}^{n+p-1} \bar{\beta}_i z^{-i} \qquad \text{(III-22)}$$

Comme le montre l'équation (III-21), la représentation entrée-sortie résultante dans la présence de la perturbation est augmentée par p comparer avec le système perturbation libre correspondant. Sachant que $\bar{\beta}_0$ est supposé connu et différent de zéro, comme décrit dans la section précédente, la commande d'entrée peut être calculée pour obtenir la réponse désirée.

Le problème considéré dans les trois étapes est que la fonction non linéaire $\bar{f}$ (ADP) n'est pas connue, et par conséquent elle va être estimée soit par un réseau de fonctions de bases radiales *RBFN* [27], ou par un réseau de neurones multicouches *MNN* [1] , ou par un système flou (FS) [23]. Par la suite le cas 1 correspond au cas où un *RBFN* est utilisé et cas 2 quand un *MNN* est utilisé [28]-[31].

Cas 1

Si un *RBFN* est utilisé pour approximer la fonction $\bar{f}$ ou (ADP) dans (III-22), la sortie du réseau avec les centres et les largeurs sont fixés exprimées comme :

$$N\left[\bar{Y}(k)\right] = \sum_{i=1}^{N} \alpha_i \Re_i\left[\bar{Y}(k)\right]$$

où N est le nombre de fonctions de la base utilisées.

$\Re_i$ Représente les fonctions de base et α_i étant les poids ajustables du *RBFN* [27], si la fonction désirée peut être représentée exactement par un *RBFN*, c'est possible de proposer un

RBFN pour approximer $\bar{f}$ dans un domaine compact. L'erreur de l'approximation résultante du *RBFN* converge seulement pour une petite perturbation bornée (petit ordre) [32]-[33].

Cas 2

Quand la fonction de la base radiale est utilisée comme expliqué dans le cas 1, l'erreur d'asservissement peut être petite. Cependant, le nombre des fonctions de la base du *RBFN* devient grand dans les problèmes d'augmentation d'ordre et la complexité. C'est en particulier dans le problème de la rejection de la perturbation parce que l'ordre du système total (i.e., la somme des ordres de système et le système générateur de la perturbation) peut être haut, même pour les systèmes d'ordres bas. On exigera que le nombre de fonctions de la base augmente exponentiellement avec la dimension de l'espace de l'entrée. Le placement adaptatif de fonction de base radiale pour s'occuper avec la complexité exponentielle est encore un problème de recherche ouvert [39]-[40]. Dans cette section, nous discutons comment les réseaux neuronaux du multicouche peuvent être utilisés dans les problèmes de la haute dimension de la rejection de la perturbation. L'usage d'un *MNN* devient obligatoire quand le système n'est pas linéaire dans la commande de l'entrée. La méthode du gradient (back propagation) sera utilisée pour l'ajustement des paramètres du réseau. Notre contribution porte sur la manière avec laquelle la logique floue est utilisée dans la rejection de la perturbation en conjonction avec la méthode de back-propagation.

V.3. Étape 2

Dans la section (V-2) la perturbation dynamique était supposée linéaire. La linéarité était essentielle à montrer l'existence d'une solution dans l'étape 1, dans cette section nous considérons un cas spécial d'un problème plus complexe où la perturbation est modelée comme la sortie d'un système libre non linéaire stable. Équations différentielles non linéaires bien connues l'équation, de *van der pol*. Sont des exemples de tels modèles non linéaires de la perturbation. Le cas spécial qui est considéré ici correspond à un système dont la représentation entrée-sortie de la perturbation-libre dépend explicitement seulement de $u(k)$ et non pas de ses valeurs passées. Dans ce cas particulier, une solution peut être trouvée pour le problème de la rejection complète de la perturbation, bien que la perturbation dynamique soit complexe et non linéaire.

Le système et la perturbation externe $v(k)$, peuvent être décrits par l'équation :

$$y(k+1) = f\left[Y(k)\right] + u(k) + v(k) \tag{III-23}$$

$$v(k+1) = g\left[v(k),...,v(k-p+1)\right] \tag{III-24}$$

où (III-24) est une équation différentielle non linéaire stable.

Pour éliminer la perturbation de (III-23), la description équivalente du système est obtenue comme :

$$y(k+1) = f\left[Y(k),U(k-1)\right] + u(k) \tag{III-25}$$

où $Y(k) = \left[y(k),..., y(k-n-p+1)\right]^T$ et

$$U(k-1) = \left[u(k-1),...,u(k-p)\right]^T$$

La représentation résultante de système donnée par (III-25) est encore linéaire dans la commande d'entrée $u(k)$. Une procédure semblable est utilisé au cas 2, qui sera adoptée pour identifier le système. Par conséquent, le modèle de l'identification est utilisé pour calculer la commande de l'entrée pour atteindre la réponse désirée [3], [35].

V.4. Étape 3

Nous considérons, a présent, le problème le plus général de l'étude, le plus difficile à résoudre qui est un système dont la représentation entrée-sortie dépend de ($u(k)$, $v(k)$) et ses valeurs passées. C'est pour cela que nous avons considéré une perturbation linéaire [3].

VI.Conclusion

L'identification et la commande des systèmes dynamiques non linéaires et la théorie de la rejection de la perturbation sont discutées dans ce chapitre. Vu les caractéristiques intrinsèques des *RBFN* et les inconvénients qui présentent, nous ne les utilisons pas dans la rejection de la perturbation. Les perturbations externes sont modelées comme la sortie linéaire ou non linéaire des systèmes dynamiques libres. La structure du contrôleur pour compenser l'effet du la perturbation sur la sortie de système a été fournie. Nous allons illustrer dans le prochain chapitre, par les résultats de simulation, comment les réseaux de neurone *(MNN)* et

la logique floue *(Fs)* seront utilisés pour arriver à des résultats efficaces. Pour chaque étape introduite précédemment, un ou deux exemples seront donnés.

CHAPITRE IV

RÉSULTATS DE SIMULATION

I. Introduction

Plusieurs méthodes d'identification ont été établies pour les systèmes non linéaires. L'avènement de la méthode des réseaux de neurones (MNN), et la logique floue (Fs) présentent des approches d'identification très efficace à la non linéarité des systèmes.

Dans ce chapitre on va appliquer les deux techniques d'identification sur quatre systèmes non linéaires selon leur complexité, et pour cela on va utiliser l'algorithme du rétro-propagation du gradient pour les deux méthodes qui est un algorithme d'apprentissage supervisé, ainsi que la structure d'identification série-parallèle et on va calculer la commande floue et neuronale.

En plus ce chapitre montre la rejection des perturbations des différents systèmes avec différents types des perturbations linéaires et non linéaires par les deux méthodes précédemment citées. En fin après que les résultats de la simulation seront donnés, une comparaison entre les deux méthodes sera faite.

II. Présentation des systèmes à identifier

- ***Système 1***

Le système 1 à identifier sera donné par l'équation récurrente suivante :

$$y(k+1) = 0.3y(k) + 0.6y(k-1) + f\left[u(k)\right] \qquad \text{(IV-1)}$$

Où la fonction inconnue f a la forme :

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

Pour identifier ce système, le modèle série parallèle suivant sera utilisé :

$$\hat{y}(k+1) = 0.3\ y(k) + 0.6\ y(k-1) + \hat{f}[u(k)]$$

- ***Système 2***

Le système 2 à identifier sera donné par l'équation récurrente suivante :

$$y(k+1) = f\left[y(k),\ y(k-1)\right] + u(k) \qquad \text{(IV-2)}$$

Où la fonction inconnue f a la forme :

$$f\left[y(k), y(k-1)\right] = y(k)\,y(k-1)\left(y(k)+2.5\right)/1+y^2\,(k)+y^2\,(k-1)$$

Le modèle d'identification sera décrit par l'équation suivante :

$$\hat{y}(k+1) = \hat{f}\left[y(k), y(k-1)\right] + u(k)$$

- ***Système 3***

Le système 3 à identifier sera donné par l'équation récurrente suivante :

$$y(k+1) = F[y(k)] + G[u(k)] \qquad \text{(IV-3)}$$

Où la fonction inconnue F est définie (pour la simulation) par :

$$F\left[y(k)\right] = y(k)/1+y^2(k)$$

et la fonction inconnue G est définie (pour la simulation) par :

$$G\left[u(k)\right] = u(k)^3$$

Le modèle d'identification est décrit par l'équation suivante :

$$\hat{y}(k+1) = \hat{F}\left[y(k)\right] + \hat{G}\left[u(k)\right]$$

- ***Système 4***

Le système 4 à identifier sera donné par l'équation récurrente suivante :

$$y(k+1) = f\left[y(k),\ y(k-1),\ y(k-2),\ u(k),\ u(k-1)\right] \qquad \text{(IV-4)}$$

Où la fonction inconnue f a la forme :

$$f\left(x_1, x_2, x_3, x_4, x_5\right) = \frac{x_1 x_2 x_3 x_5 \left(x_3 - 1\right) + x_4}{1 + x_3^2 + x_2^2}$$

Le modèle d'identification est décrit par l'équation suivante :

$$\hat{y}(k+1) = \hat{f}\left[y(k), y(k-1), y(k-2), u(k), u(k-1)\right]$$

III. Résultats :

- ***Système 1***

Pour le modèle flou nous avons pris les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k / 250)$.
- Pour l'identification nous avons trouvé 3000 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-1,1] le résultat est obtenu après 2.8473e+003s c'est-à-dire (47.4550 minute).
- 20 règles de la forme :

$$R_i : IF\ u(k)\ est\ A_i\ THEN\ y(k+1)\ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.2.
- Gain d'ajustement des prémisses 0.05.
- Gain d'ajustement des conséquences 0.01.

Pour le modèle neuronal nous avons choisis les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k / 250)$
- Pour l'identification nous avons trouvé 20000 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-1,1], le résultat est obtenu après 3.3111e+003s.
- 02 couches cachées.
- 20 neurones dans la première couche cachée.
- 10 neurones dans la deuxième couche cachée.
- Un seul neurone dans la couche de la sortie.
- La fonction d'activation utilisée dans les deux couches cachées est la fonction sigmoïde.
- Gain d'ajustement 0.1.

- ***Système 2***

Pour le modèle flou nous avons pris les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k / 25)$
- Pour l'identification nous avons trouvé 6000 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-2,2], le résultat est obtenu après 967.4380s.
- 20 règles de la forme.

$$R_i : IF\ \ y(k)\ est\ A_1^i\ \ AND\ \ y(k-1)\ est\ A_2^i\ \ THEN\ \ y(k+1)\ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.9.
- Gain d'ajustement des prémisses 0.4.
- Gain d'ajustement des conséquences 0.9.

Pour le modèle neuronal nous avons choisis les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k / 25)$
- Pour l'identification nous avons trouvé 10000 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-2,2], le résultat est obtenu après 1.3111 e+003s.
- 02 couches cachées.
- 20 neurones dans la première couche cachée.
- 10 neurones dans la deuxième couche cachée.
- Un seul neurone dans la couche de la sortie.
- La fonction d'activation utilisée dans les deux couches cachées est la fonction sigmoïde.
- Gain d'ajustement 0.1.

❖ ***Calcul La commande de poursuite***

Le modèle de référence est décrit par l'équation de différence de deuxième ordre :

$$y_m(k+1) = 0.6\,y_m(k) + 0.2\,y_m(k-1) + r(k)$$

Où $$r(k) = \sin(2\Pi k) / 25)$$

La commande de poursuite est calculé on utilisant la logique floue et les réseaux de neurone par:

$$u(k) = -\hat{f}\left[(y(k), y(k-1)\right] + 0.6y(k) + 0.2y(k-1) + r(k)$$

f : est la partie non linéaire identifié par les réseaux de neurone et la logique floue, la commande appliqué dans le système:

$$y(k+1) = \underbrace{f\left[(y(k), y(k-1)\right]}_{u(k)} = \hat{f}\left[(y(k), y(k-1)\right] + 0.6y(k) + 0.2y(k-1) + r(k)$$

- ***Système 3***

Pour le modèle flou, l'identification de F est pris les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k/25) + \sin(2\Pi k/10)$
- Pour l'identification nous avons trouvé 248 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-2,2], le résultat est obtenu après 41.2960s.
- 40 règles de la forme :

$$R_i : IF\ \ y(k)\ est\ \ A_i\ \ THEN\ \ y(k+1)\ est\ \ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.5.
- Gain d'ajustement des prémisses 0.5.
- Gain d'ajustement des conséquences 0.01.

Pour le modèle flou, l'identification de G est pris les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k/25) + \sin(2\Pi k/10)$
- Pour l'identification nous avons trouvé 2938 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-2,2], le résultat est obtenu après 268.1720s.
- 20 règles de la forme :

$$R_i : IF\ \ u(k)\ est\ \ A_i\ \ THEN\ \ y(k+1)\ est\ \ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.5.
- Gain d'ajustement des prémisses 0.9.
- Gain d'ajustement des conséquences 0.06.

Pour le modèle neuronal, l'identification de F on a choisis les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k/25) + \sin(2\Pi k/10)$
- Pour l'identification nous avons trouvé 2000 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-2,2], le résultat est obtenu après 139.3100s.
- 02 couches cachées.
- 20 neurones dans la première couche cachée.
- 10 neurones dans la deuxième couche cachée.
- Un seul neurone dans la couche de la sortie.
- La fonction d'activation utilisée dans les deux couches cachées est la fonction sigmoïde.
- Gain d'ajustement 0.1.

Pour le modèle neuronal l'identification de G on a choisis les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k/25) + \sin(2\Pi k/10)$
- Pour l'identification nous avons trouvé 50000 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-2,2], le résultat est obtenu après 1.5111 e+003s.
- 02 couches cachées.
- 20 neurones dans la première couche cachée.
- 10 neurones dans la deuxième couche cachée.
- Un seul neurone dans la couche de la sortie.
- La fonction d'activation utilisée dans les deux couches cachées est la fonction sigmoïde.
- Gain d'ajustement 0.1

❖ ***Calcul La commande de poursuite***

Le modèle de référence est décrit par l'équation de différence de premier ordre :

$$y_m(k+1) = 0.6\,y_m(k) + r(k)$$

Où $$r(k) = \sin((2\Pi k)/25)) + \sin((2\Pi k)/10))$$

La commande de poursuite est calculé on utilisant la logique floue et les réseaux de neurone par:

$$u(k) = \hat{G}^{-1}\left[-\hat{F}[y(k)] + 0.6\,y(k) + r(k)\right]$$

f : est la partie non linéaire identifié par les réseaux de neurone et la logique floue

La commande appliquée dans le système :

$$y(k+1) = G(k) \times \underbrace{\hat{G}^{-1}\left[-\hat{F}[y(k)] + 0.6y(k) + r(k)\right]}_{u(k)} + F[y(k)]$$

- ***Système 4***

Pour le modèle flou nous avons pris les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k/250)$ pour $k \le 500$ et $u(k) = 0.8\sin(2\Pi k/250) + 0.2\sin(2\Pi k/25)$ pour $k \ge 500$
- Pour l'identification nous avons trouvé 7000 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-1,1], le résultat est obtenu après 1.6076 e+003s.
- 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ AND\ y(k-1)\ est\ A_2^i\ AND\ y(k-2)\ est\ A_3^i$$
$$AND\ u(k)\ est\ A_4^i\ AND\ u(k-1)\ est\ A_5^i\ THEN\ y(k+1)\ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.5.
- Gain d'ajustement des prémisses 0.5.
- Gain d'ajustement des conséquences 1.

Pour le modèle neuronal nous avons choisis les paramètres suivants :

- L'entrée du système et du modèle doit être $u(k) = \sin(2\Pi k/250)$ pour $k \le 500$ et $u(k) = 0.8\sin(2\Pi k/250) + 0.2\sin(2\Pi k/25)$ pour $k \ge 500$.
- Pour l'identification nous avons trouvé 21000 itérations pour minimiser l'erreur avec une entrée aléatoire entre [-1,1], le résultat est obtenu après 2.3473e+003s.
- 02 couches cachées.
- 20 neurones dans la première couche cachée.
- 10 neurones dans la deuxième couche cachée.
- Un seul neurone dans la couche de la sortie.
- La fonction d'activation utilisée dans les deux couches cachées est la fonction sigmoïde.
- Gain d'ajustement 0.1.

IV. La simulation

❖ *Système 1*

a- identification floue ***b- identification neuronale***

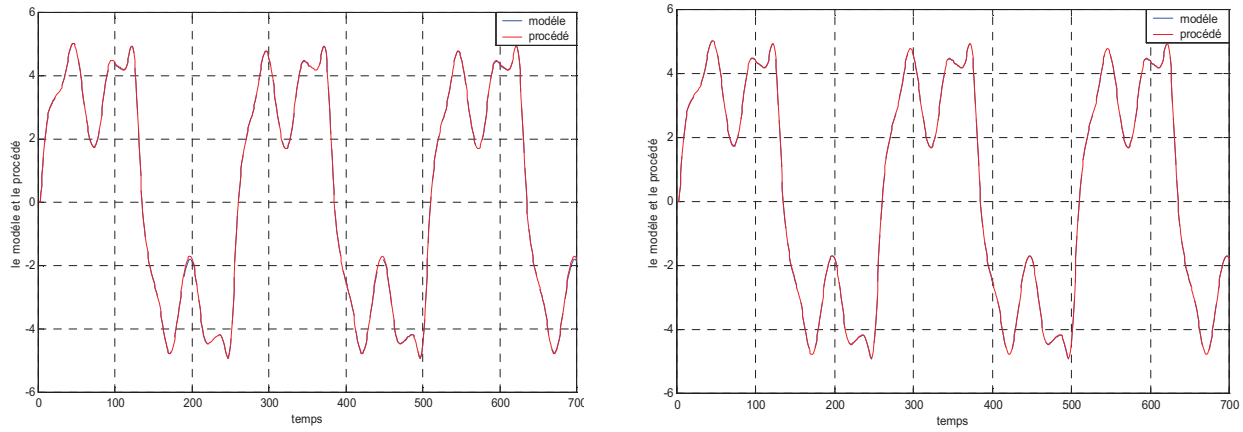


Figure.IV-1 : le modèle et le procédé 1

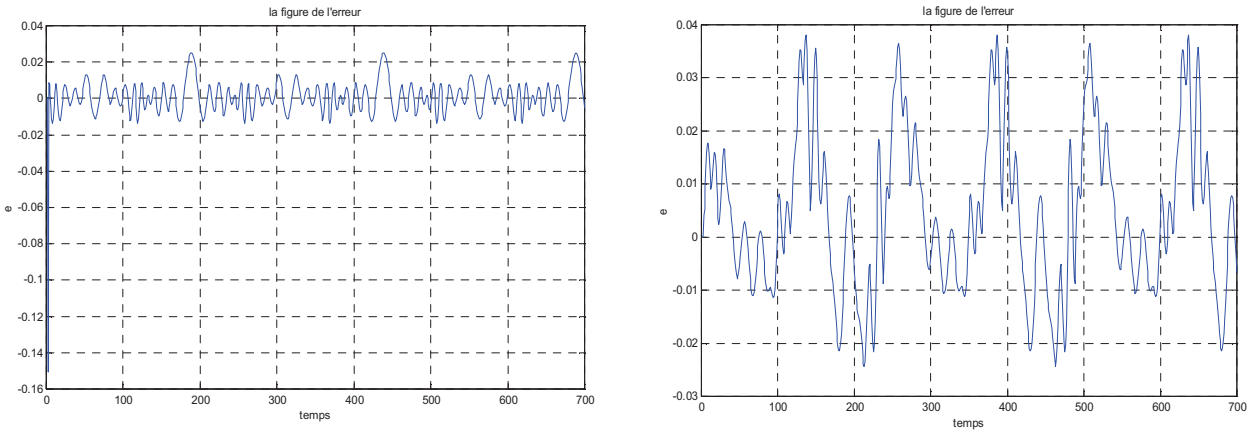


Figure.IV-2 : la somme des erreurs moyennes MSE

❖ *Système 2*

a- identification floue ***b- identification neuronale***

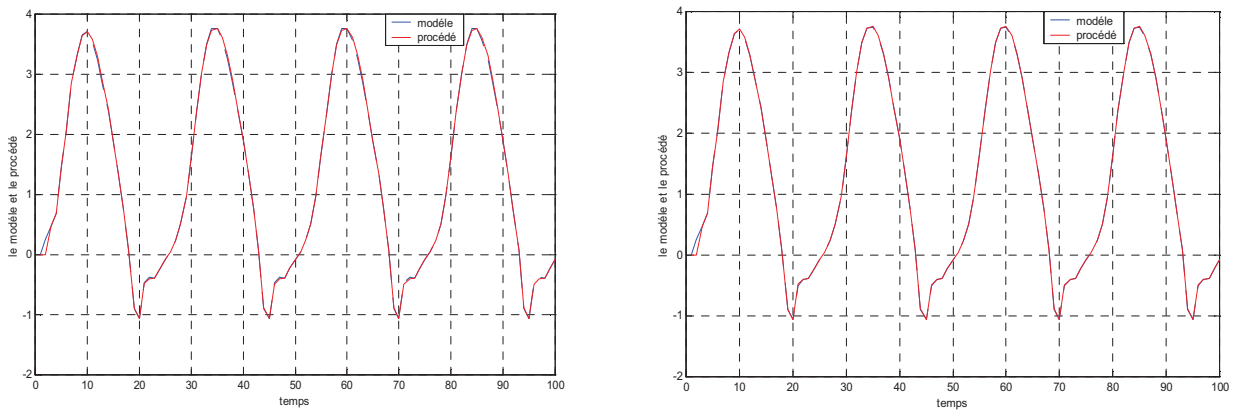


Figure.IV-3 : le modèle et le procédé 2

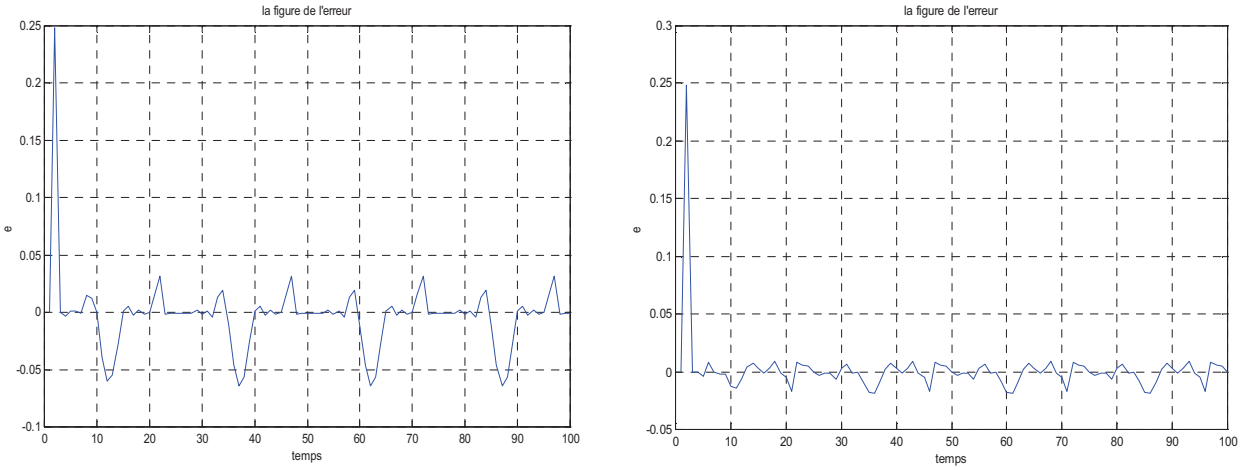


Figure.IV-4 : MSE entre le modèle et le procédé

❖ ***Système 3***

a- identification floue ***b- identification neuronale***

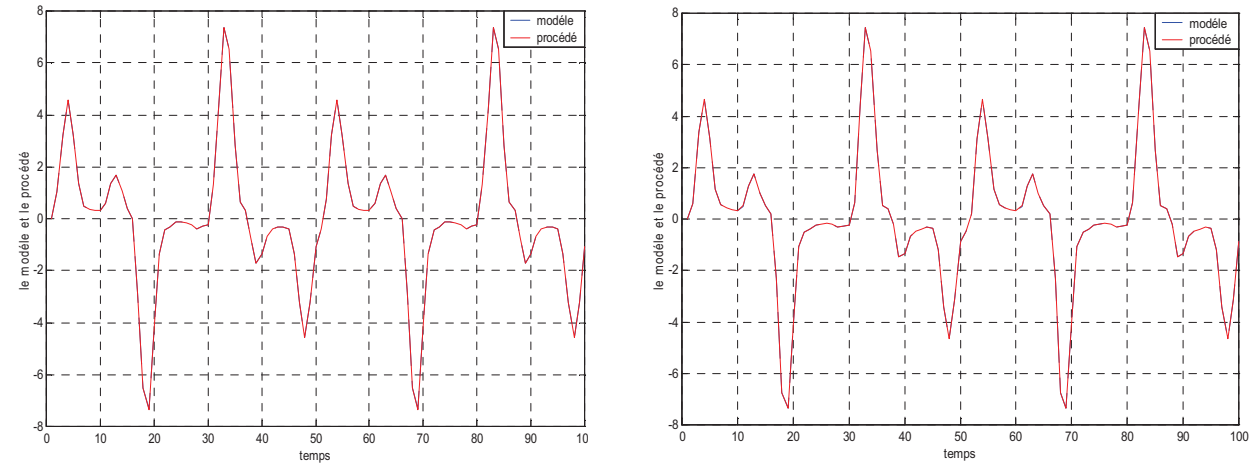


Figure.IV-5 : le modèle et le procédé 3

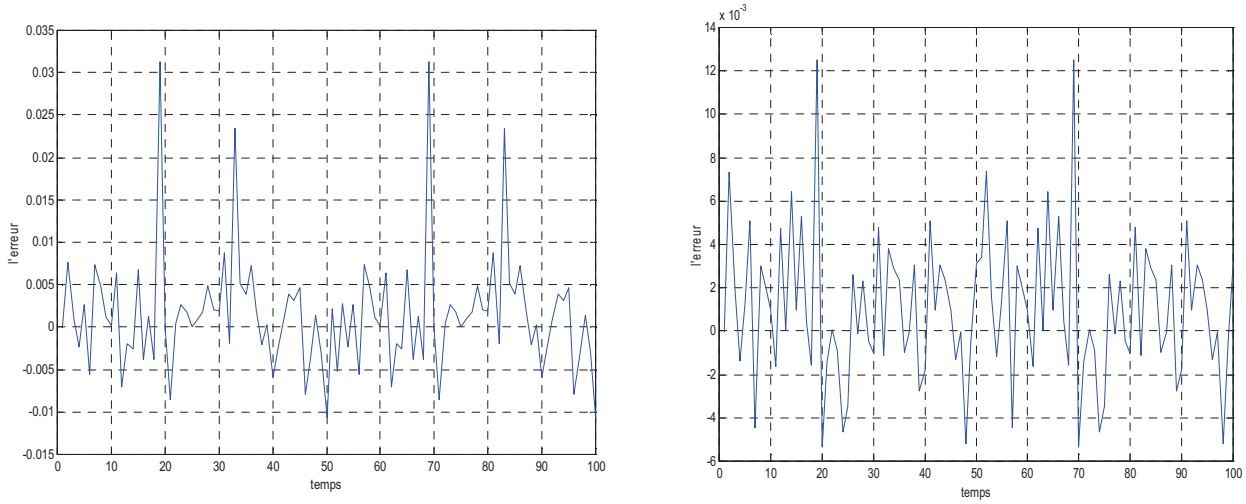


Figure.IV-6 : MSE entre le modèle et le procédé

❖ ***Système 4***

a- identification floue ***b- identification neuronale***

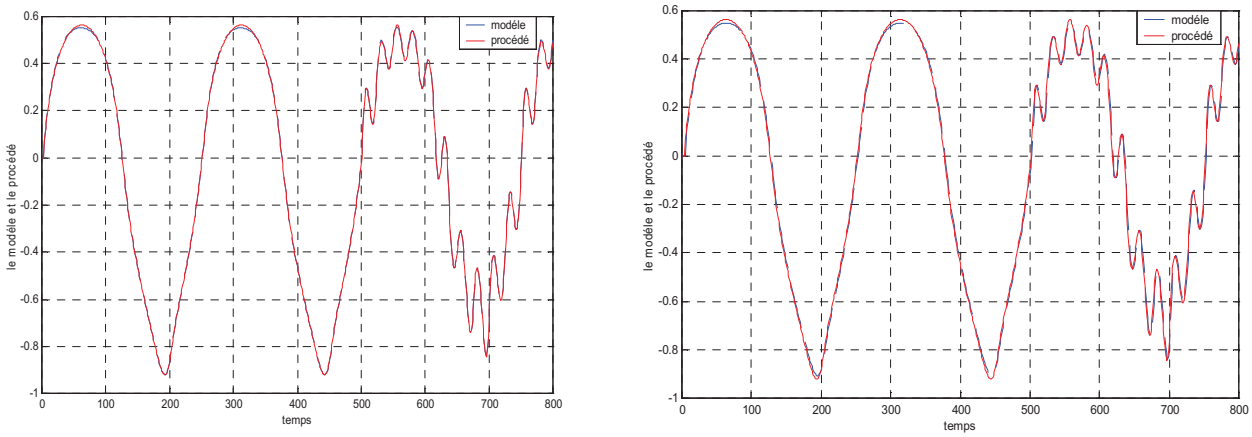


Figure.IV-7 : le modèle et le procédé 4

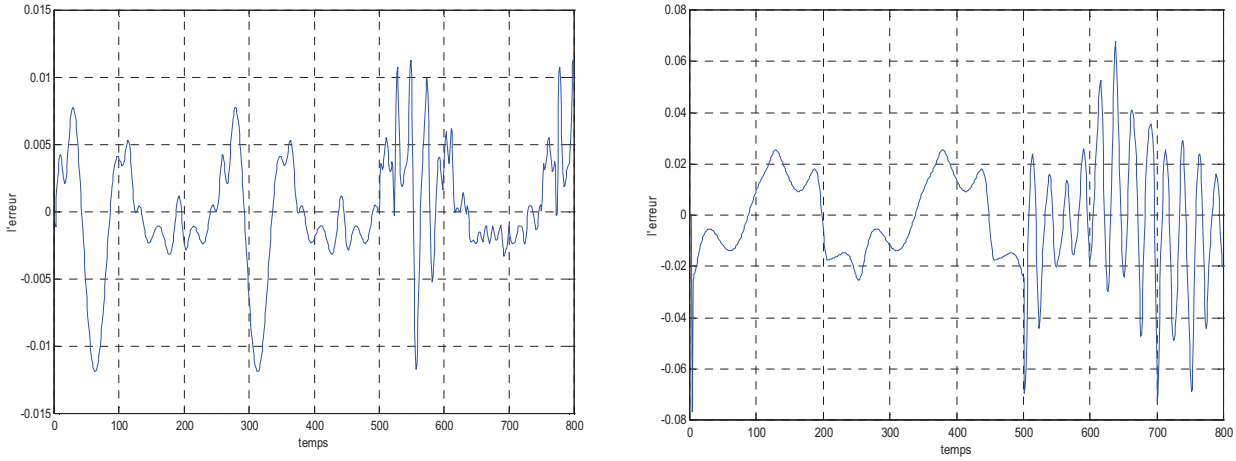


Figure.IV-8 : MSE entre le modèle et le procédé

V. la courbe de la commande floue de système 2

a- la commande floue ***b- la commande neuronale***

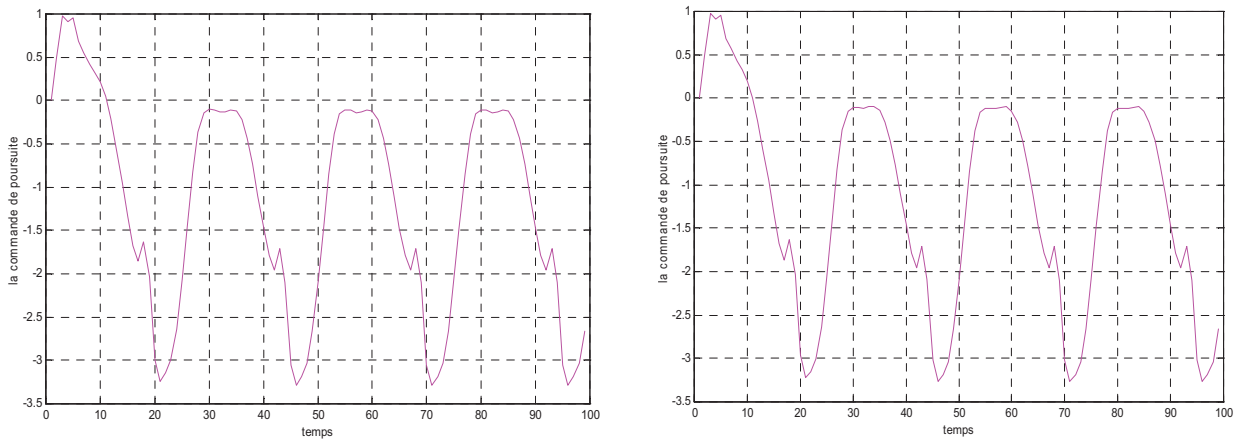


Figure.IV-9 : la commande de poursuite

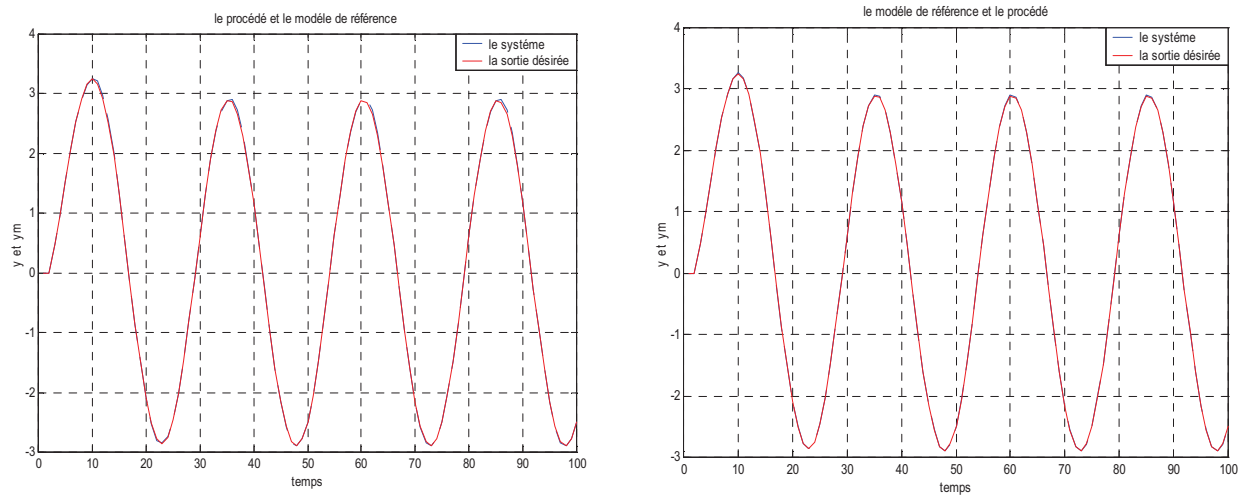


Figure.IV-10 : le système et le modèle de référence

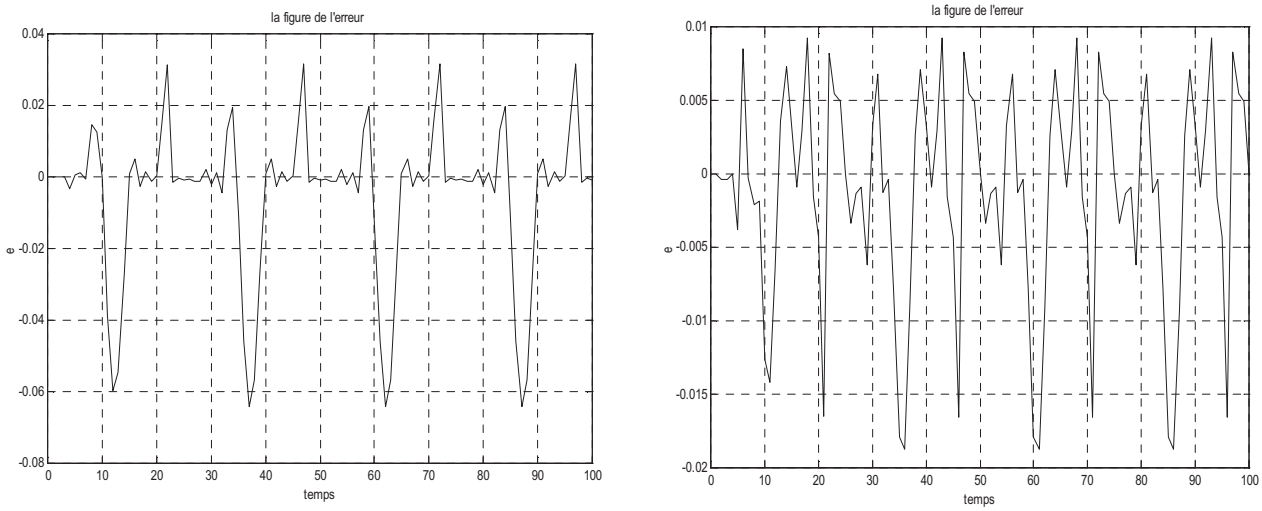


Figure.IV-11 : MSE entre le système et la sortie désirée

VI. La courbe de la commande floue de système 3

a- la commande floue ***b- la commande neuronale***

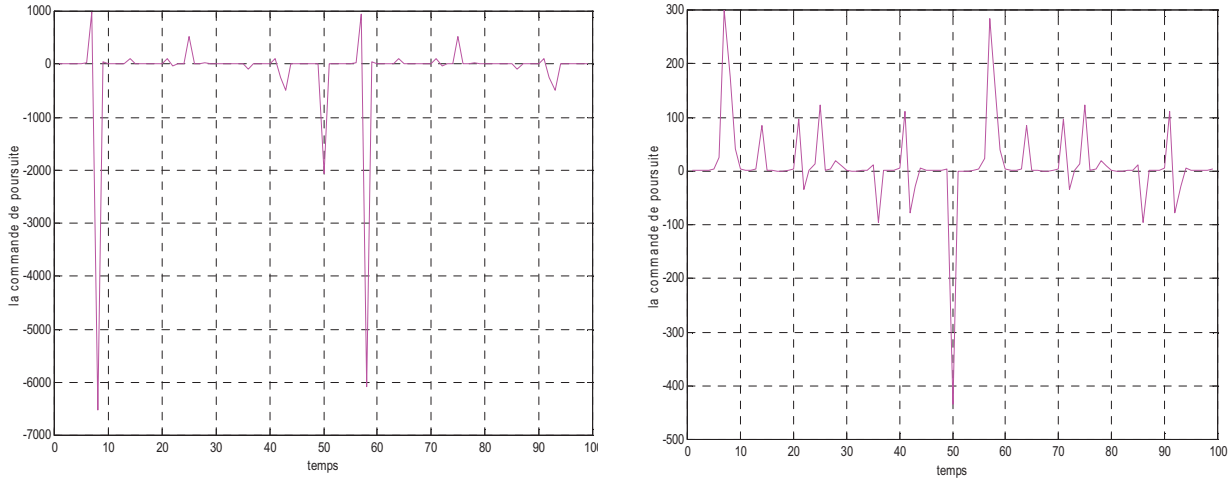


Figure.IV- 12:la commande de poursuite

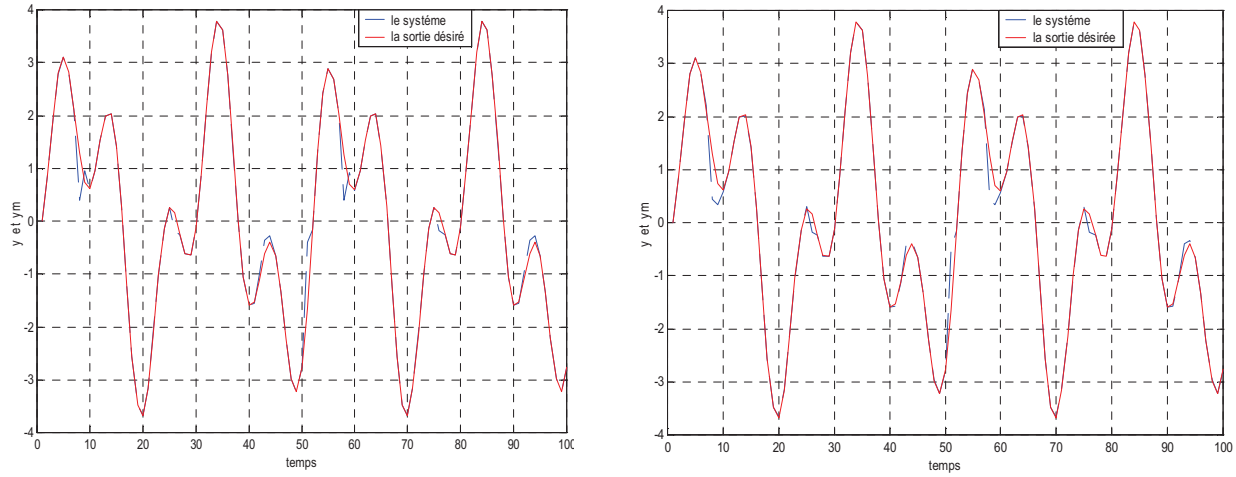


Figure.IV-13 : le système et le modèle de référence

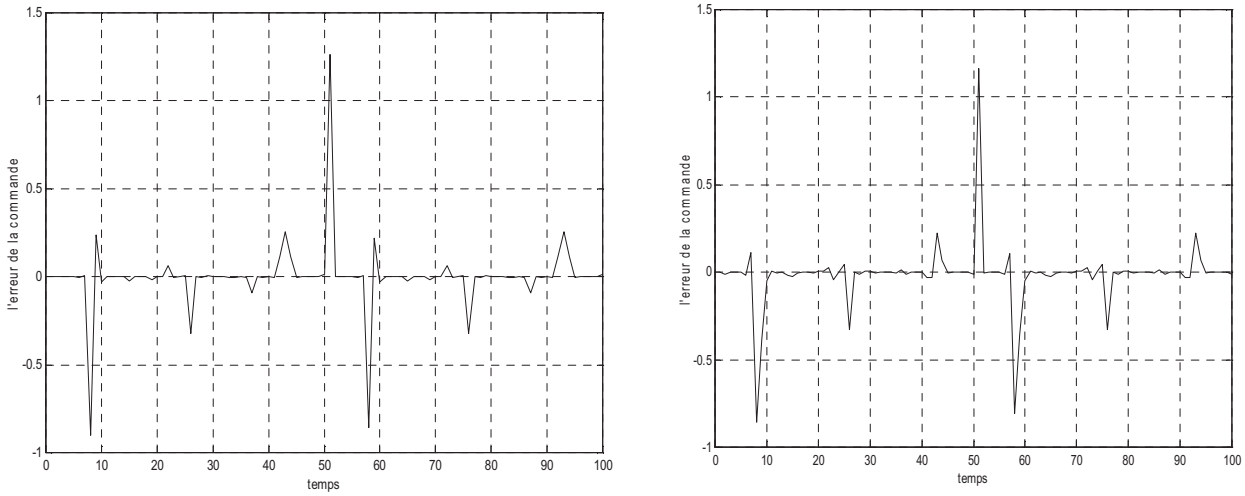


Figure.IV-14 : MSE entre le système et la sortie désirée

VII. La rejection de la perturbation par les réseaux de neurone et la logique floue

VIII. La simulation

VIII.1.Étape 1

VIII.1.1.La rejection de la perturbation par les réseaux de neurone

- ***Système 1-1***

Soit un système du premier ordre dans la présence d'une perturbation linéaire :

$$y(k+1) = \frac{y(k)}{1+y(k)^2} + u(k) + v(k) \qquad \text{(IV-5)}$$

$v(k)$: C'est la perturbation linéaire de deuxième ordre considéré comme la somme de deux sinusoïdes peut être modelé comme la sortie du système dynamique libre suivant :

$$\begin{aligned} v1(k+1) &= \cos(\Psi)v1(k) + \sin(\Psi)v2(k) \\ v2(k+1) &= \cos(\Psi)v2(k) - \sin(\Psi)v1(k) \\ v(k) &= v_1(k) \end{aligned} \qquad \text{(IV-6)}$$

Telle que $(2\Pi)/\Psi$ c'est la période de la perturbation sinusoïdale.

Dans la simulation Ψ a été choisis égal à $\Pi/6$ et les conditions initiale de la perturbation sont choisis comme $[v_{10}, v_{20}] = [1,0]$, donc c'est une perturbation d'amplitude égale à 1 et fréquence 12.

- ✓ ***La forme de cette perturbation***

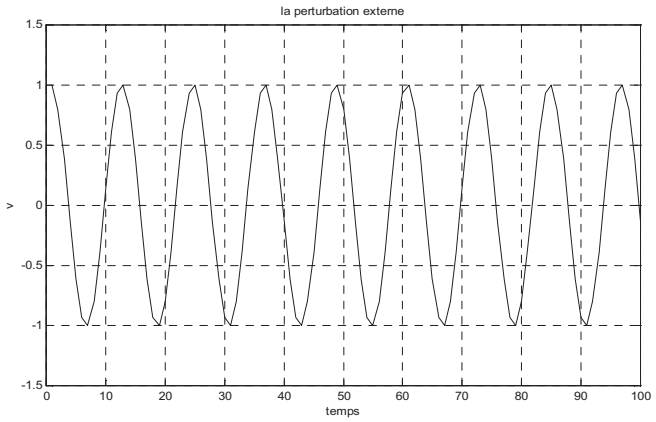


Figure.IV-15:la perturbation $v(k)$ externe de deuxième ordre

La représentation entrée-sortie du système dans la présence de la perturbation est :

$$y_{inc}(k+1) = f_{inc}[y(k), y(k-1), y(k-2), u(k-1), u(k-2)] + u(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k), y(k-1), ..., y(k-q), u(k-1), u(k-2), ..., u(k-q)] + u(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f_{inc}, pour une identification exacte on choisis $q \geq 2$, l'identification dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1,1].

La forme de ce réseau neuronal est décrite par : $N^3_{(2q+1),40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec $(2q+1)$ neurone d'entrée et une seule sortie. Avec un pas d'apprentissage égal à 0.05, les résultats sont obtenus après 6417 itérations et temps d'exécution égal à 1.4378e+003s. Des 3 différentes valeurs de q $(q=0, q=2, q=4)$ sont utilisées pour la comparaisons.

NB : y_{inc} Indique (increased) c'est-à-dire augmenté , y_p : Le procédé , y_m : le modèle de référence
L'objectif est d'atteindre la sortie désirée :

$$y_m(k+1) = 0.6y_m(k) + r(k)$$

Telle que l'entrée référence est :

$$r(k) = 0.5\sin((2\Pi k)/50)) + 0.5\sin((2\Pi k)/10))$$

✓ ***La commande d'entrée est calculé par***

$$u_r(k) = (y_m(k+1) - NN[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)])$$

Le résultat de la commande montre que :
Si $q=2$ rejection complète de la perturbation ; erreur très petite.
Si $q=4$ rejection pareil à $q = 2$.

VIII.1.2. La rejection de la perturbation par la logique floue

• ***Système 1-1***

La représentation entrée-sortie de système dans la présence de la perturbation est :

$$y_{inc}(k+1) = f_{inc}[y(k), y(k-1), y(k-2), u(k-1), u(k-2)] + u(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = Fs[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)] + u(k)$$

où $Fs(.)$ est réalisé par la logique floue comme une approximation à f_{inc}, pour une identification exacte on choisis $q \geq 2$, l'identification dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1, 1].

Pour l'identification floue on a choisi les paramètres suivants : après 844 itérations et temps d'exécution égal à 2901s pour minimiser l'erreur avec 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ AND\ y(k-1)\ est\ A_2^i\ AND\ y(k-2)\ est\ A_3^i$$
$$AND\ u(k-1)\ est\ A_4^i\ \ AND\ \ u(k-2)\ est\ A_5^i\ THEN\ y(k+1)\ \ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.9.
- Gain d'ajustement des prémisses 0.03.
- Gain d'ajustement des conséquences 0.1

✓ ***La commande d'entrée est calculé par***

$$u_r(k) = (y_m(k+1) - Fs[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)])$$

Le résultat de la commande montre que :

Si $q = 2$ donc rejection complète de la perturbation, erreur très petite.

a- rejection neuronale ***b- rejection floue***

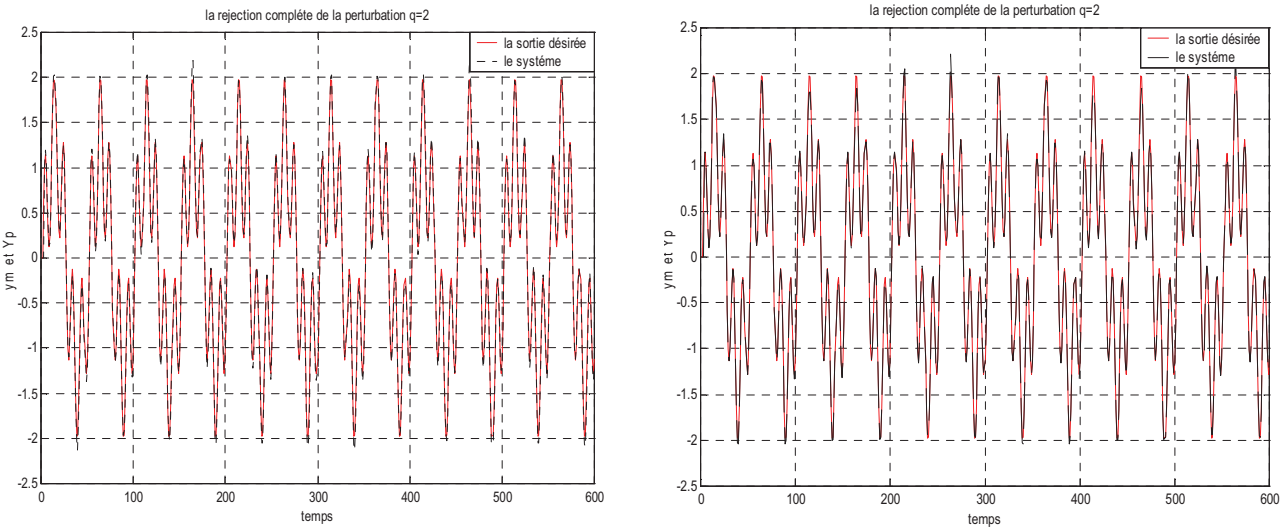


Figure.IV-16 : une rejection complète de la perturbation pour q égal à 2

a- la commande neuronale ***b- la commande floue***

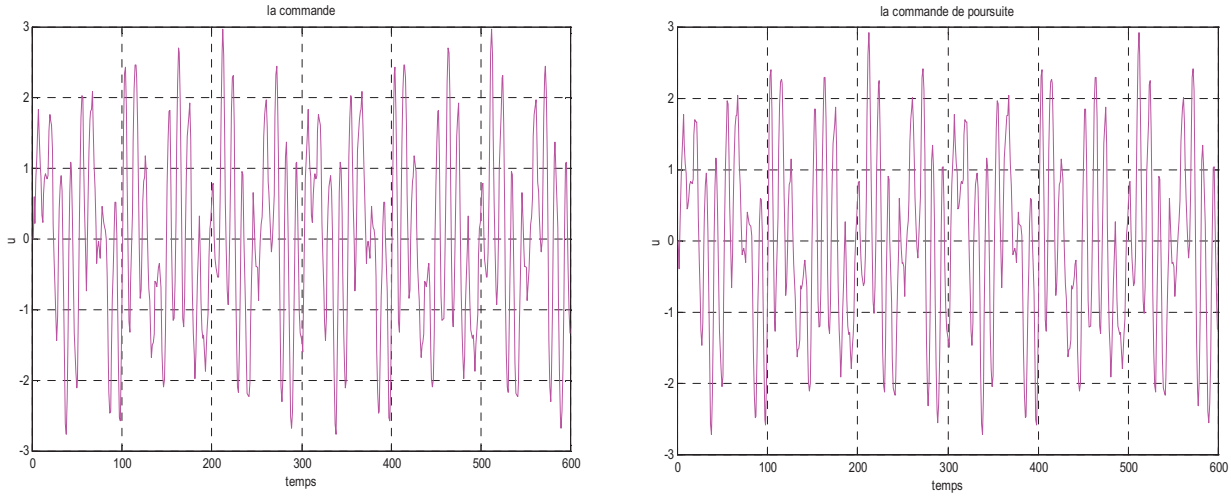


Figure.IV-17 : la commande de poursuite

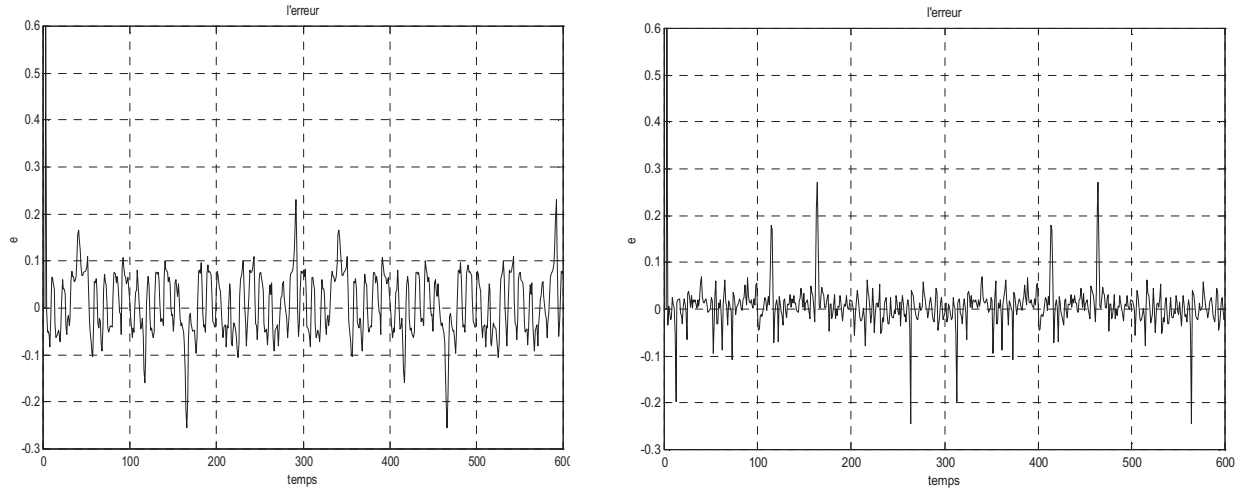


Figure.IV-18 :l'erreur entre y_p et y_m

VIII.1.3. Pas de rejection de la perturbation q=0

- ***Par les réseaux de neurone (MNN)***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k)] + u(k) + v(k)$$

Pour identifier le système, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k)] + u(k) + v(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f, l'identification se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1,1].

La forme de ce réseau neuronal est décrite par : $N^3_{1,40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec une seule entrée et une seule sortie. Avec un pas d'apprentissage égal à 0.05, les résultats sont obtenus après 10.000 itérations.

✓ ***La commande d'entrée est calculé par***

$$u(k) = y_m(k+1) - NN[y(k)]$$

Le résultat montre que :
Si $q = 0$ donc pas de réjection de la perturbation.

❖ ***Par la logique floue (Fs)***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k)] + u(k) + v(k)$$

Pour identifier le système, on propose le modèle d'identification :

$$\hat{y}(k+1) = Fs[y(k)] + u(k) + v(k)$$

où $Fs(.)$ est réalisé par la logique floue comme une approximation à f, pour l'identification floue on a choisi les paramètres suivants : après 2000 itérations et temps d'exécution égal à 698.8910 s pour minimiser l'erreur et 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ \ THEN\ y(k+1)\ \ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 1.
- Gain d'ajustement des prémisses 0.006.
- Gain d'ajustement des conséquences 0.005.

L'identification se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1, 1].

✓ ***La commande d'entrée est calculé par***

$$u(k) = y_m(k+1) - Fs[y(k)]$$

VIII.1.4. Pas de rejection de la perturbation

a- Par MNN ***b- Par Fs***

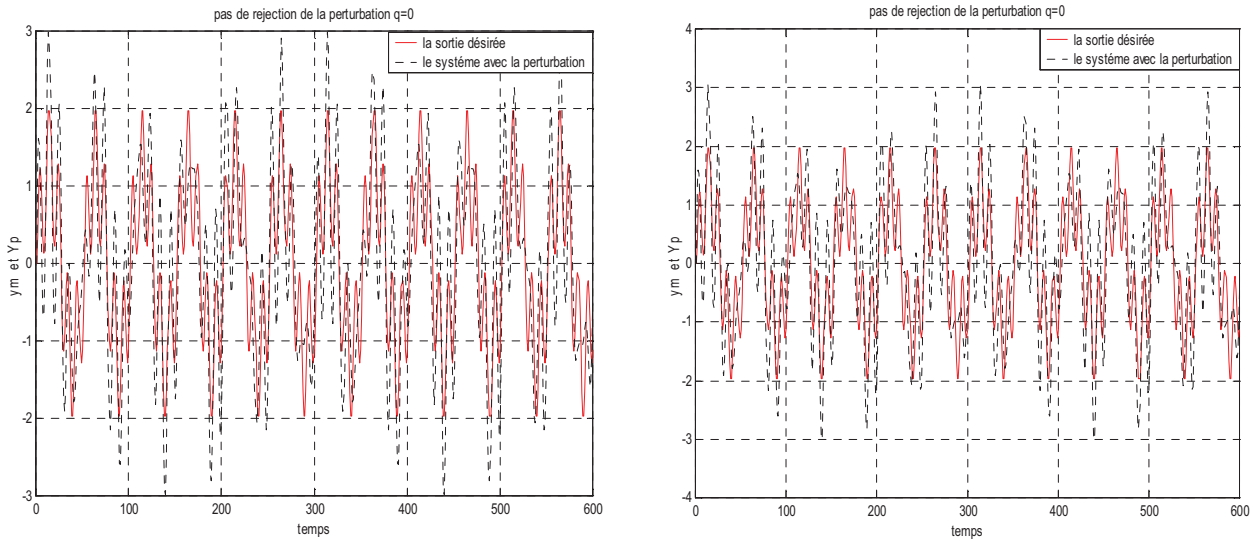
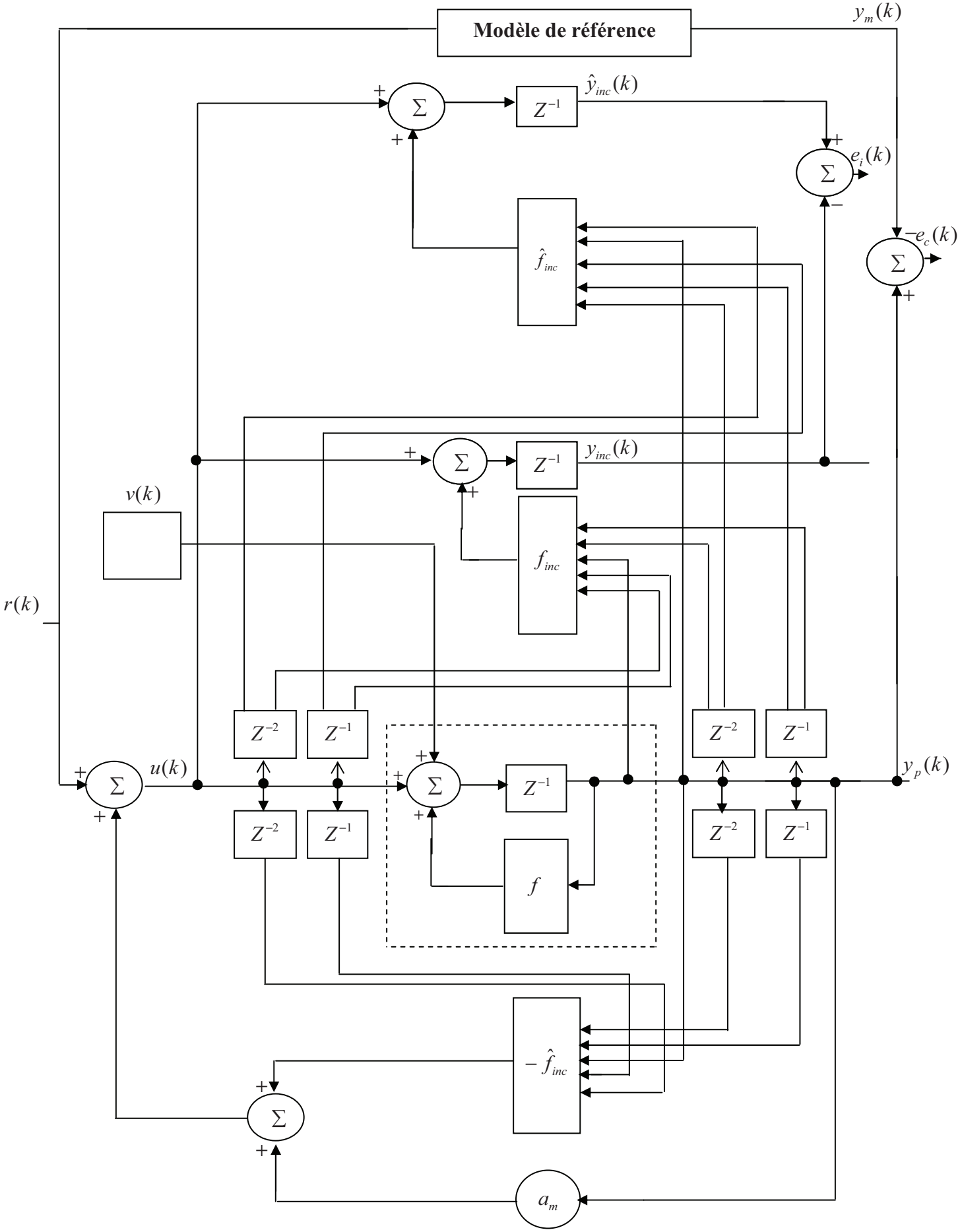
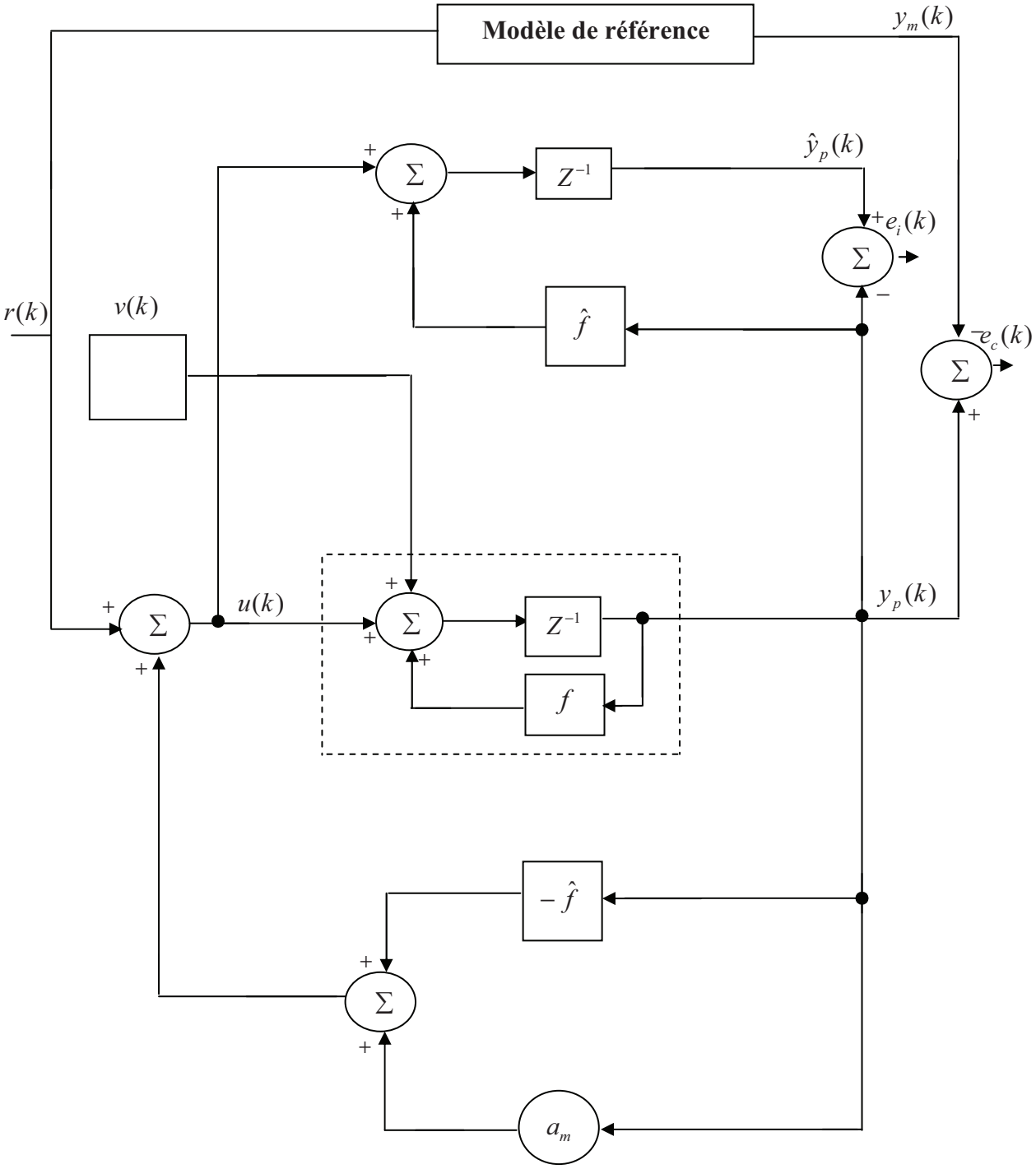


Figure.IV-19 : pas de rejection de la perturbation pour q égal à 0

VIII.1.5. La structure de tout le système q égal à 2



VIII.1.6. La structure du système ou q égal à 0



VIII.1.7. La rejection par les réseaux de neurone

- ***Système 1-2***

Soit le système du premier ordre plus complexe dans la présence d'une perturbation linéaire:

$$y(k+1)=\frac{\cos(y(k))}{1+\sin(y(k))^2}+\sin(y(k))\exp(-y(k)^2)+\frac{y(k)}{1+y(k)^2}+u(k)+v(k) \qquad (\text{IV-7})$$

$v(k)$: C'est la perturbation du quatrième ordre, c'est la somme de 2 sinusoïdes et peut être représenté par :

$$x_v(k+1)=\begin{bmatrix}\cos\psi_1 & \sin\psi_1 & 0 & 0\\ -\sin\psi_1 & \cos\psi_1 & 0 & 0\\ 0 & 0 & \cos\psi_2 & \sin\psi_2\\ 0 & 0 & -\sin\psi_2 & \cos\psi_2\end{bmatrix}x_v(k) \qquad (\text{IV-8})$$

$$v(k)=x_{v1}(k)+x_{v3}(k)$$

où $x_v(k)\in\Re^4$, ψ_1 et ψ_2 représente la période de 2 fonctions sinusoïdales constituent la perturbation telle que :

$\psi_1=\frac{\Pi}{6}$ et $\psi_2=\frac{\Pi}{12}$, $x_{vi}(k)$ c'est le i^{eme} élément de $x_v(k)$

- ✓ ***La forme de la perturbation***

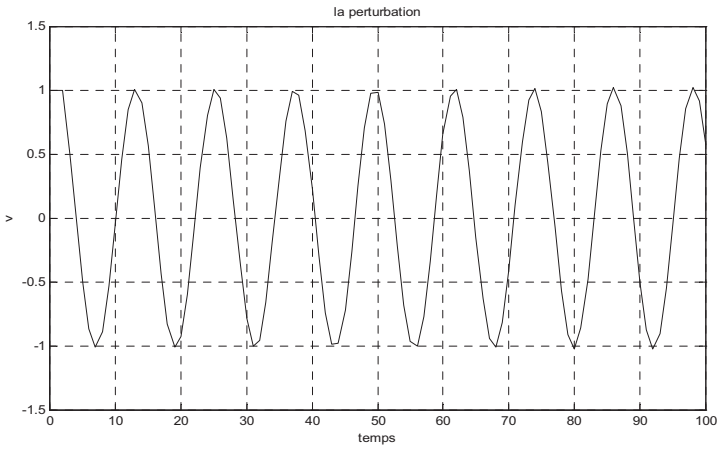


Figure.IV-20:la perturbation $v(k)$ externe de quatrième ordre

La représentation entrée-sortie du système dans la présence de la perturbation est :

$$y_{inc}(k+1) = f_{inc}[y(k), y(k-1),..., y(k-4), u(k-1), u(k-2),..., u(k-4)] + u(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k), y(k-1),..., y(k-4), u(k-1), u(k-2),..., u(k-4)] + u(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f_{inc}, pour une identification exacte on choisis $q \geq 4$, l'identification dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1,1].

La forme de ce réseau neuronal est décrite par : $N^{3}_{(2q+1),40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec $(2q+1)$ neurone d'entrée et une seule sortie, avec un pas d'apprentissage égal à 0.05 et temps d'exécution égal à 2.9717e+003s et nombre d'itérations égal à 12088 itérations la valeurs de q $(q = 4)$.

L'objectif est d'atteindre la sortie désirée :

$$y_m(k+1) = 0.6 y_m(k) + r(k)$$

Telle que l'entrée référence est :

$$r(k) = 0.25\sin(2\Pi k/25) + 0.25\sin(2\Pi k/10)$$

- ✓ ***La commande de poursuite d'entrée est calculé par***

$$u_r(k) = (y_m(k+1) - NN(..))$$

Si q=4 : rejection complète de la perturbation.

VIII.1.8. La rejection par la logique floue

- ***Système 1-2***

Pour identifier le système avec la perturbation, on propose le modèle d'identification floue :

$$\hat{y}(k+1) = Fs[y(k), y(k-1),..., y(k-4), u(k-1), u(k-2),..., u(k-4)] + u(k)$$

où $Fs(.)$ est réalisé par la logique floue comme une approximation à f_{inc}, pour une identification exacte on choisis $q \geq 4$, l'identification dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1, 1].

Pour l'identification floue on a choisi les paramètres suivants : après 8038 itérations et temps d'exécution égal à 1.8904e+003s pour minimiser l'erreur avec 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ AND\ y(k-1)\ est\ A_2^i\ AND\ y(k-2)\ est\ A_3^i\ AND\ y(k-3)\ est\ A_4^i\ AND\ y(k-4)\ est\ A_5^i$$
$$AND\ u(k-1)\ est\ A_6^i\ AND\ u(k-2)\ est\ A_7^i\ AND\ u(k-3)\ est\ A_8^i\ AND\ u(k-4)\ est\ A_9^i\ THEN\ y(k+1)\ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.9.
- Gain d'ajustement des prémisses 0.09.
- Gain d'ajustement des conséquences 0.09

✓ ***La commande de poursuite d'entrée est calculé par***

$$u_r(k) = (y_m(k+1) - Fs(..))$$

Le résultat de la commande montre que :
Si $q = 4$: rejection complète de la perturbation.

a- rejection neuronale ***b- rejection floue***

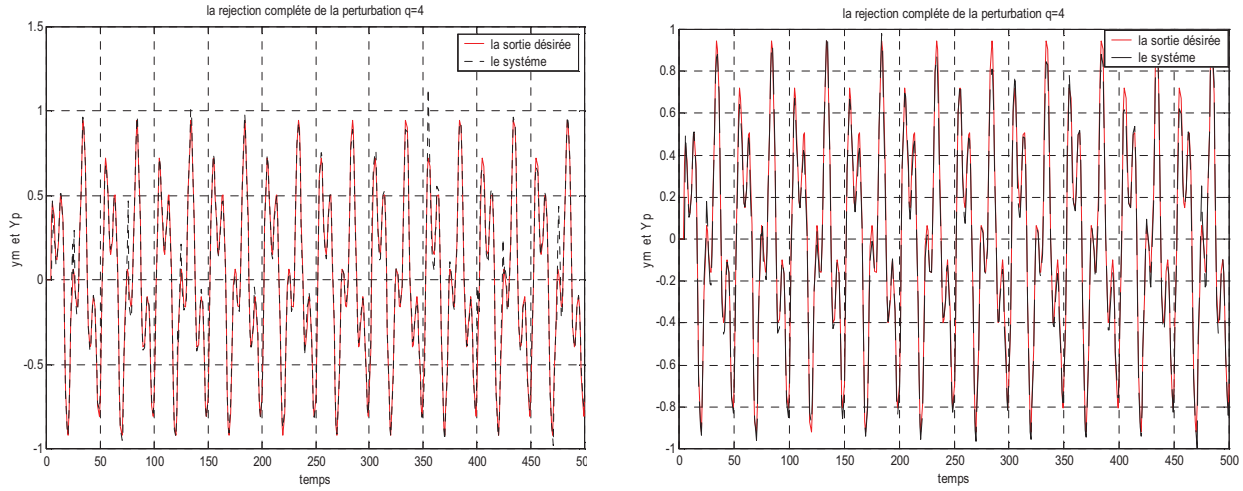


Figure.IV-21 : une rejection complète de la perturbation pour q égal à 4

a- la commande neuronale ***b- la commande floue***

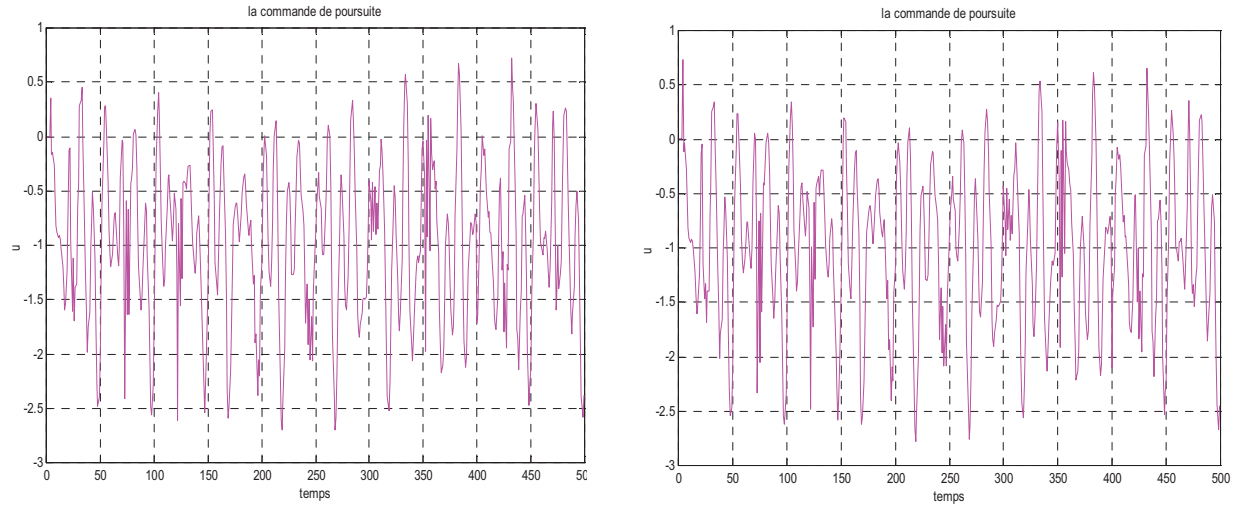


Figure.IV-22 : la commande de poursuite

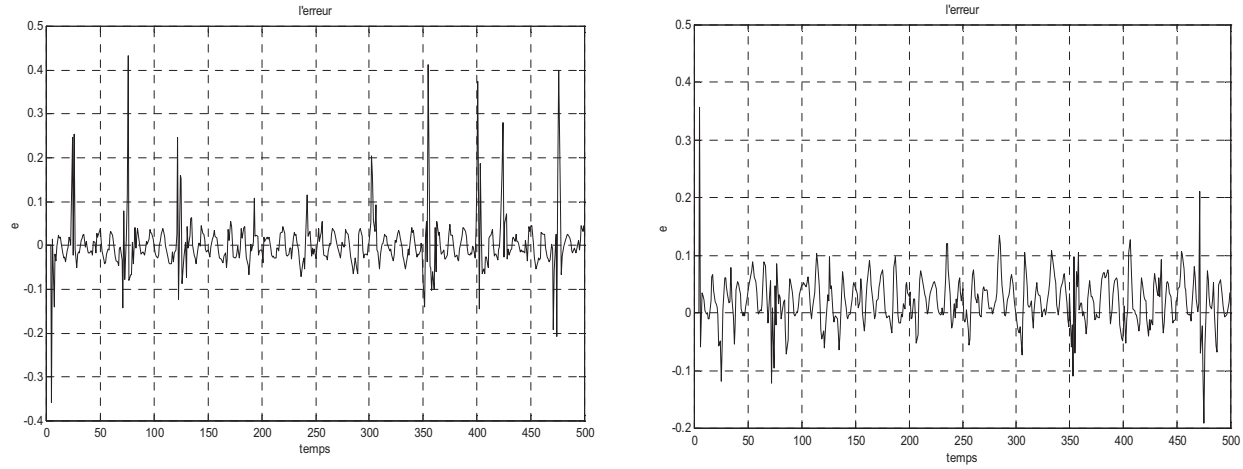


Figure.IV- 23 : l'erreur entre le système et le modèle de référence

VIII.1.9. Pas de rejection de la perturbation q=0

- ***Par MNN***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k)] + u(k) + v(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k)] + u(k) + v(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f, l'identification dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1,1].

La forme de ce réseau neuronal est décrite par : $N^{3}_{1,40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec une seule entrée et une seule sortie et avec un pas d'apprentissage égal à 0.01, la valeurs de q $(q = 0)$.

- ✓ ***La commande d'entrée est calculé par***

$$u(k) = (y_m(k+1) - NN(..))$$

Si $q = 0$ donc pas de réjection de perturbation.

- ***Par Fs***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k)] + u(k) + v(k)$$

Pour identifier le système, on propose le modèle d'identification :

$$\hat{y}(k+1) = Fs[y(k)] + u(k) + v(k)$$

où $Fs(.)$ est réalisé par la logique floue comme une approximation à f, pour l'identification floue on a choisi les paramètres suivants : pour minimiser l'erreur 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ \ THEN\ y(k+1)\ \ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.9.
- Gain d'ajustement des prémisses 0.6.
- Gain d'ajustement des conséquences 0.5.

L'identification se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1, 1].

✓ ***La commande d'entrée est calculé par***

$$u(k) = y_m(k+1) - Fs[y(k)]$$

Le résultat de la commande montre que :

Si $q = 0$ donc pas de réjection de la perturbation.

VIII.1.10. Pas de rejection de la perturbation

a- Par MNN ***b- Par Fs***

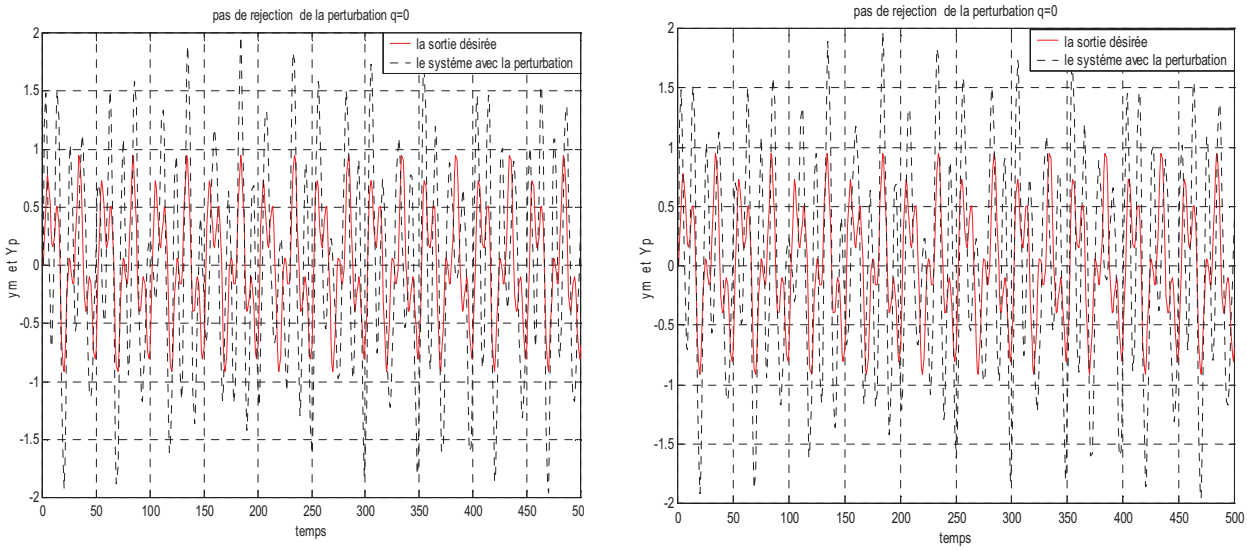
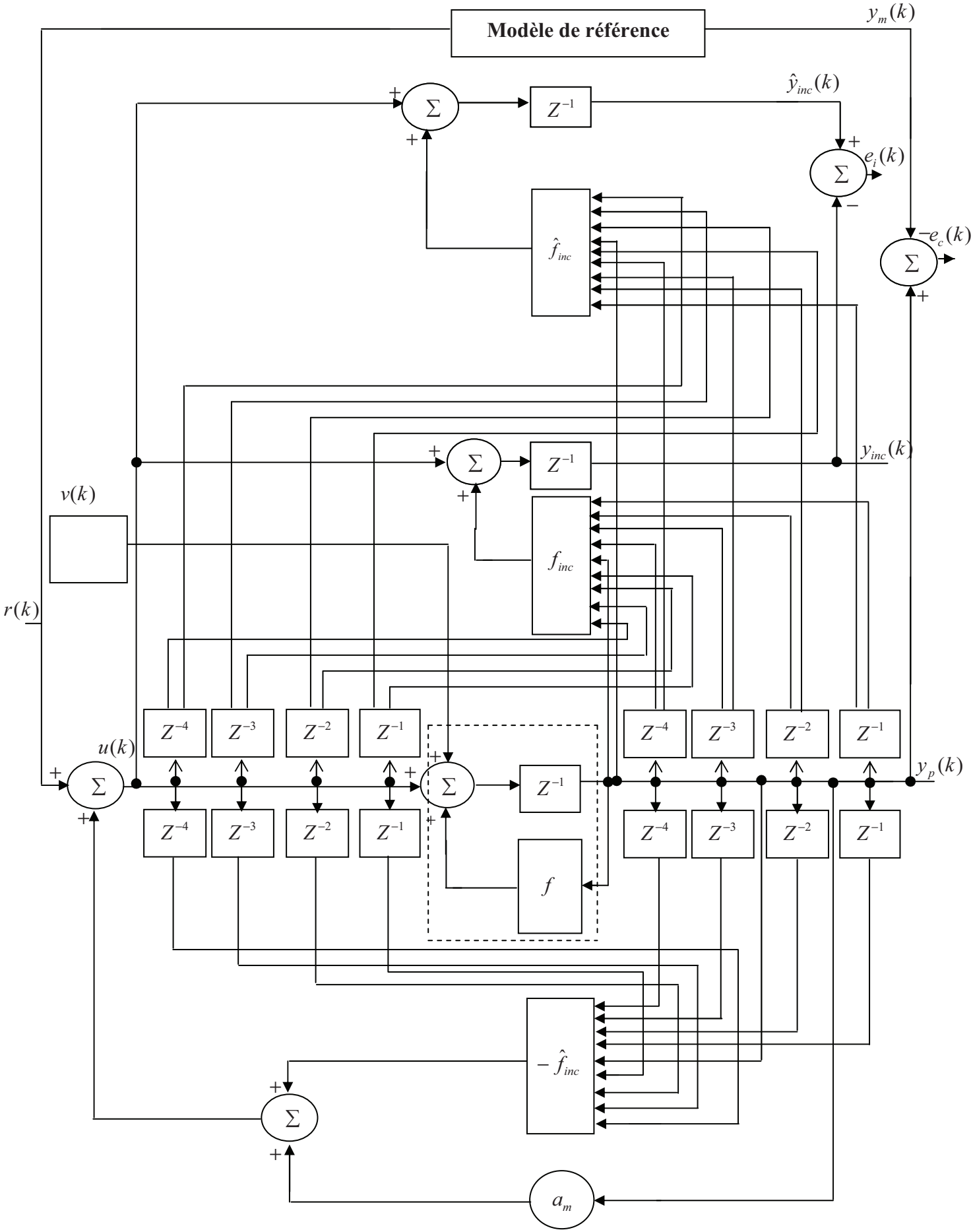
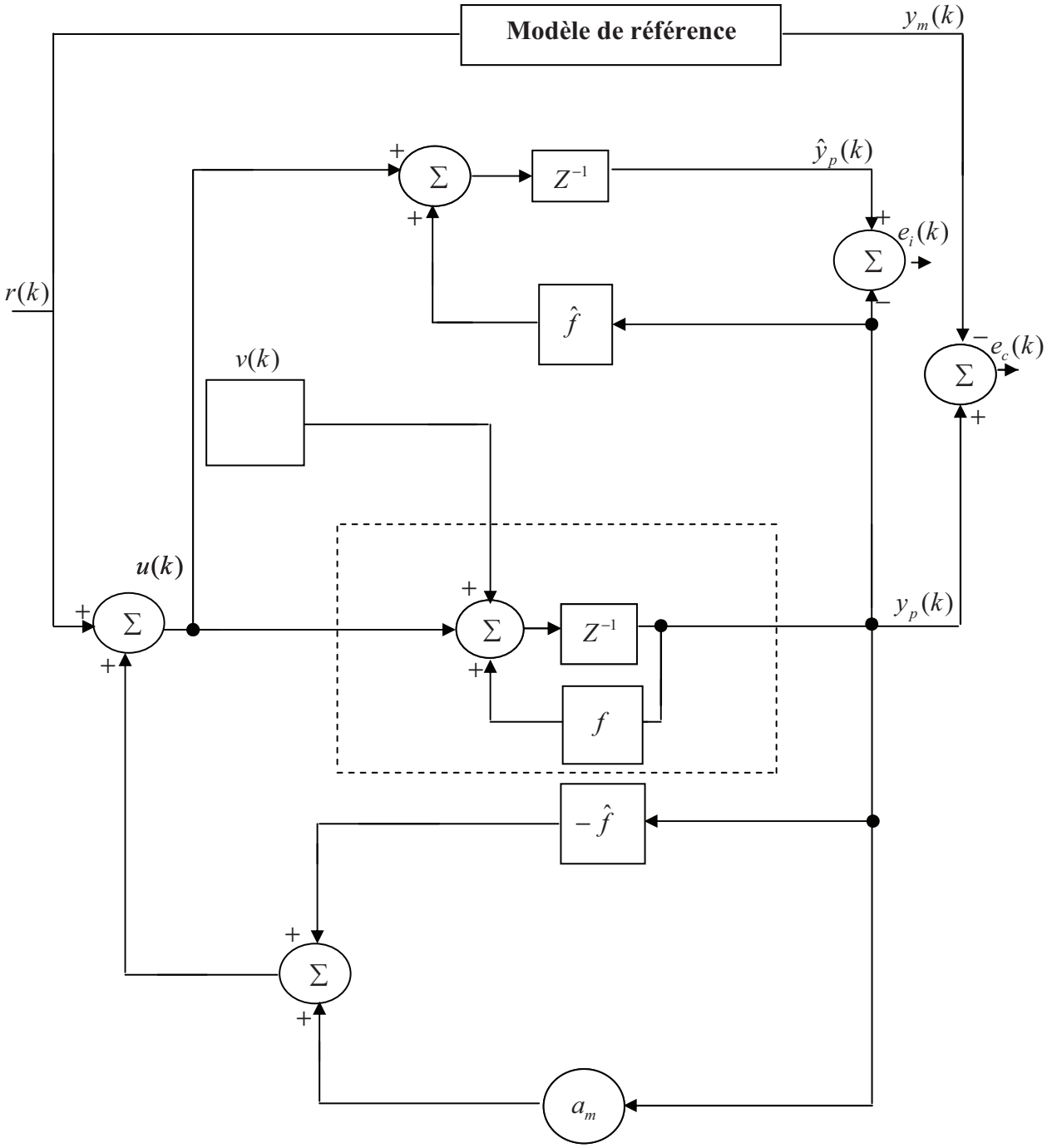


Figure.IV-24 : pas de rejection de la perturbation q égal à 0

VIII.1.11. La structure de tout le système pour q égal à 4



VIII.1.12. La structure du système où q égal à 0



VIII.2. Étape 2

VIII.2.1. La rejection de la perturbation par les réseaux de neurone

- ***Système 2-1***

Le système du premier ordre dans la présence d'une perturbation exponentielle :

$$y(k+1) = \frac{y(k)}{1+y(k)^2} + u(k) + v(k) \qquad \text{(IV-9)}$$

$v(k)$: c'est la perturbation non linéaire du premier ordre.

$$v(k+1) = \exp(-v(k)) - 2 \qquad \text{(IV-10)}$$

- ✓ ***La forme de la perturbation***

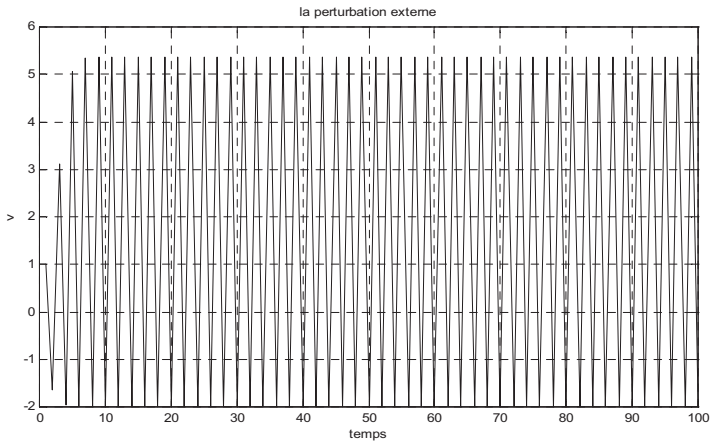


Figure.IV-25 : la perturbation non linéaire $v(k)$

La représentation entrée-sortie du système dans la présence de la perturbation est :

$$y_m(k+1) = f_{inc}\left[y(k), y(k-1), u(k-1)\right] + u(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k), y(k-1), ..., y(k-q), u(k-1), u(k-2), ..., u(k-q)] + u(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f_{inc}, pour une identification exacte on choisis $q \geq 1$, l'identification dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre $[-1.5, 1.5]$.

La forme de ce réseau neuronal est décrite par : $N^3_{(2q+1),40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec $(2q+1)$ neurone d'entrée et une seule sortie. Avec un pas d'apprentissage égal à 0.005, les résultats sont obtenus après 1544 itérations et temps d'exécution égal à 970.3910s.

L'objectif est d'atteindre la sortie désirée :

$$y_m(k+1) = 0.6y_m(k) + r(k)$$

Telle que l'entrée référence est :

$$r(k) = 0.5\sin((2\Pi k)/10)) + \sin((2\Pi k)/50))$$

✓ ***La commande d'entrée est calculé par***

$$u_r(k) = (y_m(k+1) - NN[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)])$$

Le résultat de la commande montre que :
Si $q = 1$ donc rejection complète de la perturbation, erreur très petite.

VIII.2.2. La rejection par la logique floue

- ***Système 2-1***

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = Fs[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)] + u(k)$$

Où Fs (.) est réalisé par la logique floue comme une approximation à f_{inc}.

Pour une identification exacte on choisis $q \geq 1$, l'identification dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1.5, 1.5].

Pour l'identification floue on a choisi les paramètres suivants : après 900 itérations et temps d'exécution 698.8910s et 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ AND\ y(k-1)\ est\ A_2^i\ AND\ u(k-1)\ est\ A_3^i\ THEN\ y(k+1)\ \ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 1.9.
- Gain d'ajustement des prémisses 0.5.
- Gain d'ajustement des conséquences 0.05

✓ ***La commande d'entrée est calculé par***

$$u_r(k) = (y_m(k+1) - Fs[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)])$$

Le résultat de la commande montre que :

Si $q = 1$ donc rejection complète de la perturbation, erreur très petite.

a- rejection neuronale ***b- rejection floue***

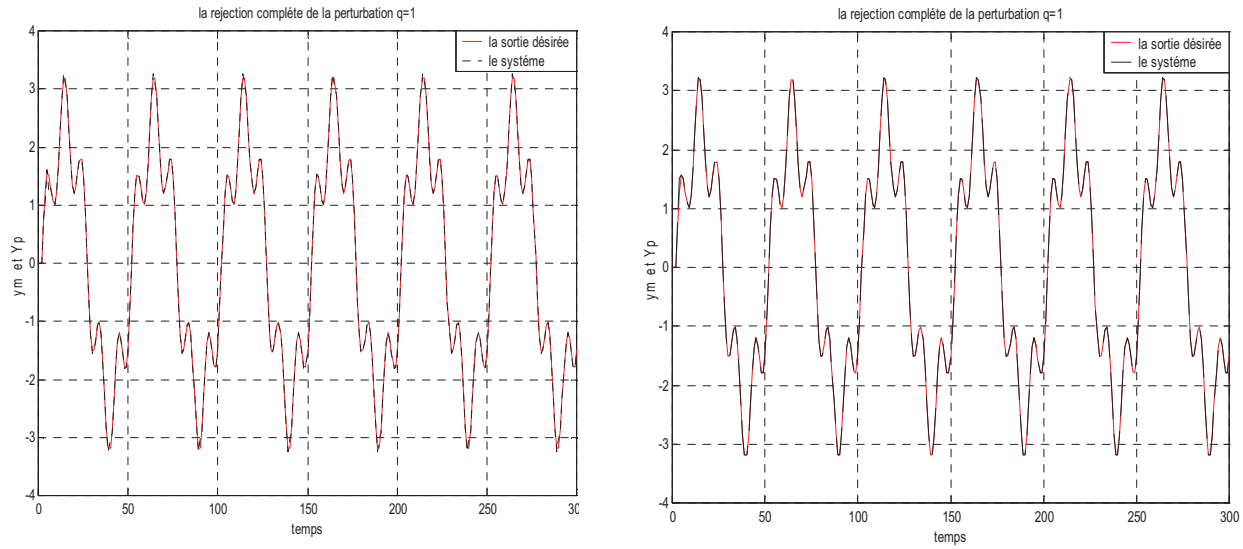


Figure.IV-26 : une rejection complète de la perturbation pour q égal à 1

a- la commande neuronale ***b- la commande floue***

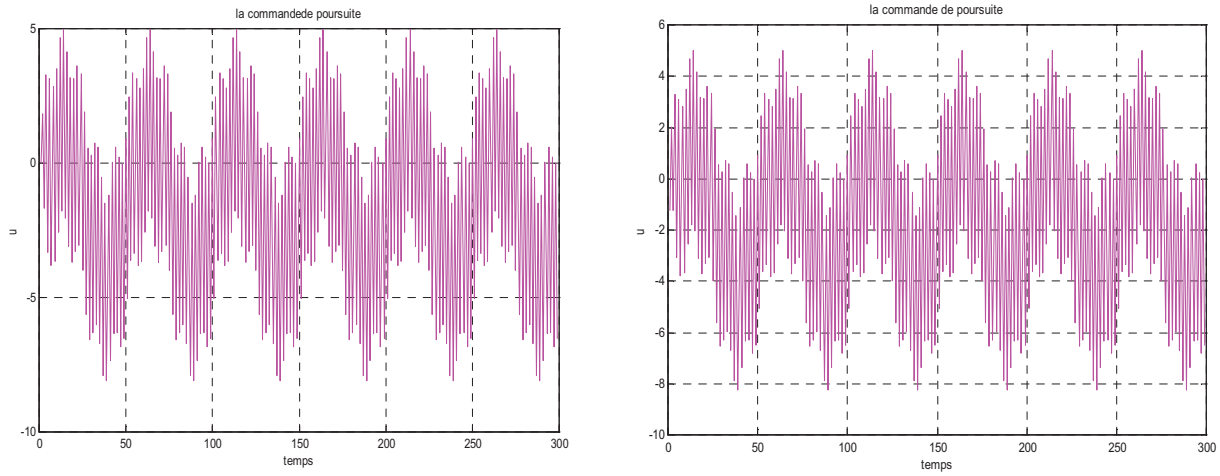


Figure.IV-27 : la commande de poursuite

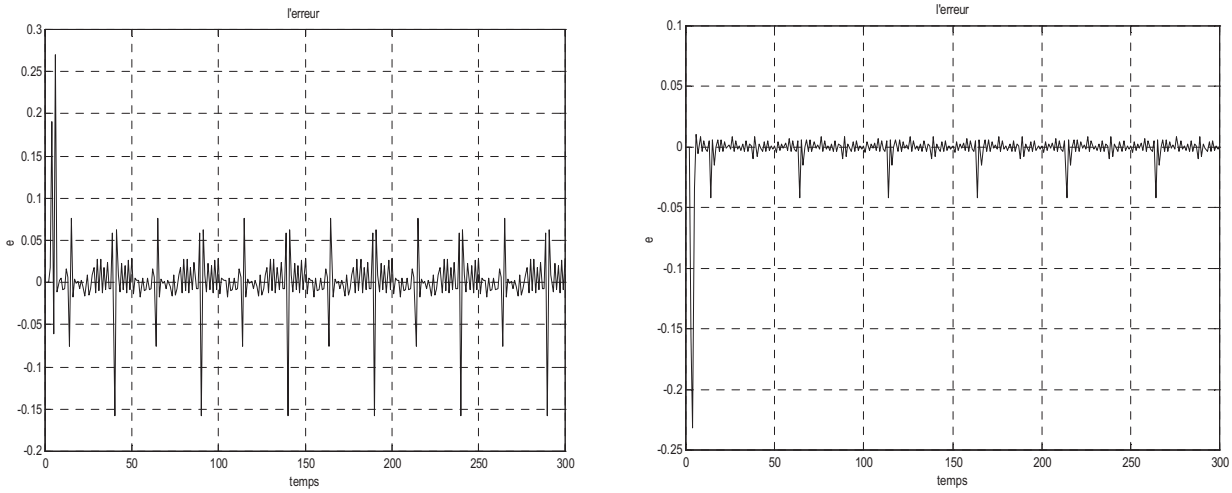


Figure.IV-28 :l'erreur entre y_p et y_m

VIII.2.3. Pas de rejection de la perturbation q=0

- ***Par MNN***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k)] + u(k) + v(k)$$

Pour identifier le système, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k)] + u(k) + v(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f, l'identification se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1.5, 1.5].

La forme de ce réseau neuronal est décrite par : $N^3_{1,40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec une seule entrée et une seule sortie et un pas d'apprentissage égal à 0.05, les résultats sont obtenus après 5.000 itérations.

✓ ***La commande d'entrée est calculé par***

$$u(k) = y_m(k+1) - NN[y(k)]$$

Le résultat de la commande montre que :
Si $q = 0$ donc pas de réjection de la perturbation.

❖ ***Par Fs***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k)] + u(k) + v(k)$$

Pour identifier le système, on propose le modèle d'identification :

$$\hat{y}(k+1) = Fs[y(k)] + u(k) + v(k)$$

où $Fs(.)$ est réalisé par la logique floue comme une approximation à f, pour l'identification floue on a choisi les paramètres suivants : après 1000 itérations et temps d'exécution 698.8910s pour minimiser l'erreur et 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ \ THEN\ y(k+1)\ \ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.9.
- Gain d'ajustement des prémisses 0.05.
- Gain d'ajustement des conséquences 0.05.

L'identification se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-1.5, 1.5].

✓ ***La commande d'entrée est calculé par***

$$u(k) = y_m(k+1) - Fs[y(k)]$$

Le résultat de la commande montre que :

Si $q = 0$ donc pas de réjection de la perturbation.

VIII.2.4. Pas de rejection de la perturbation

a- Par MNN ***b- Par Fs***

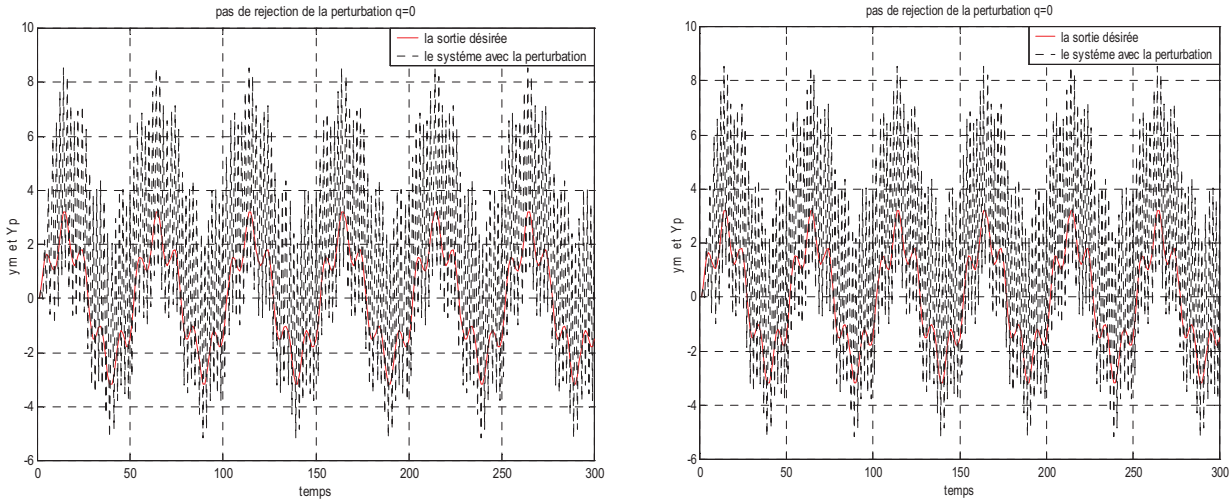
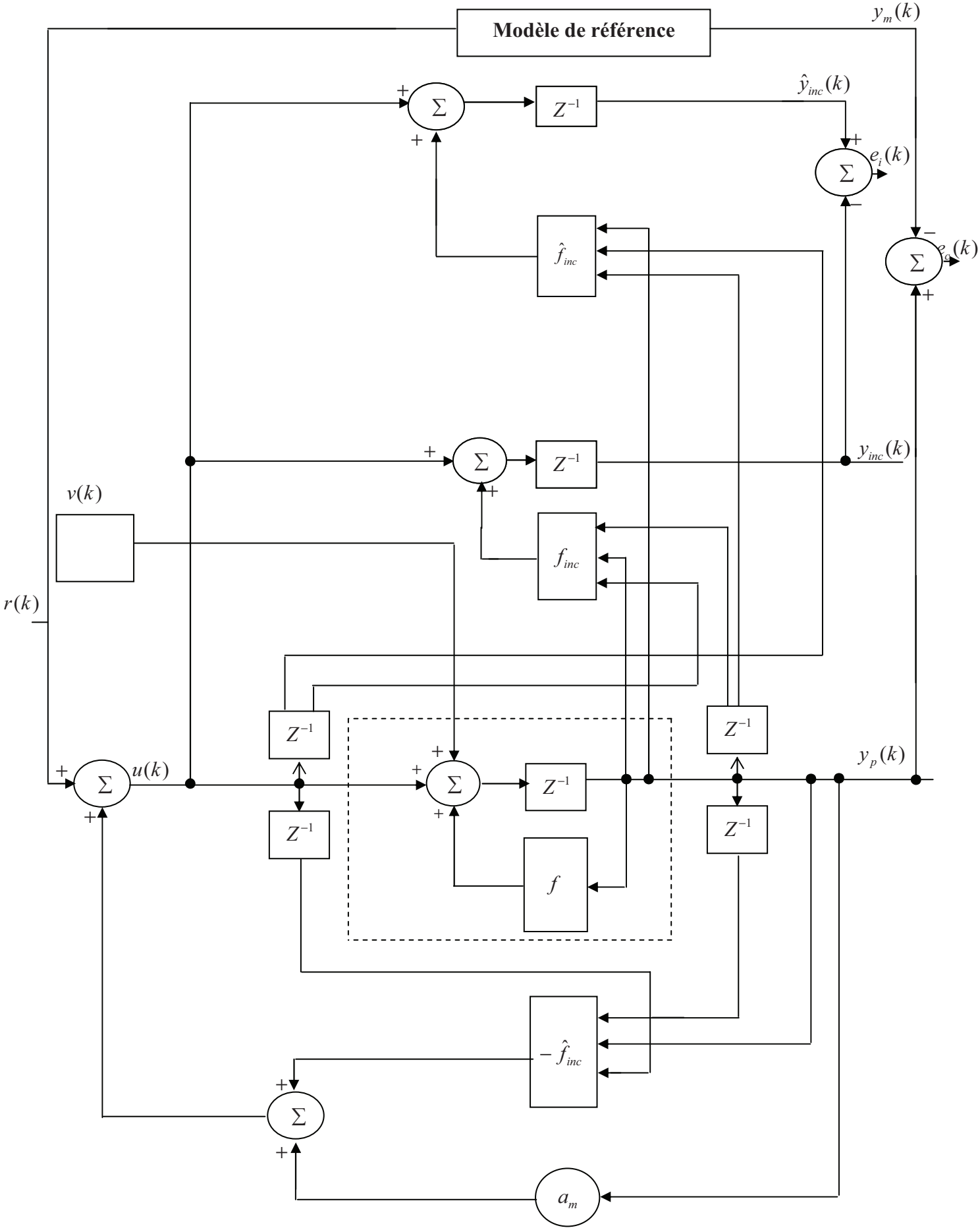
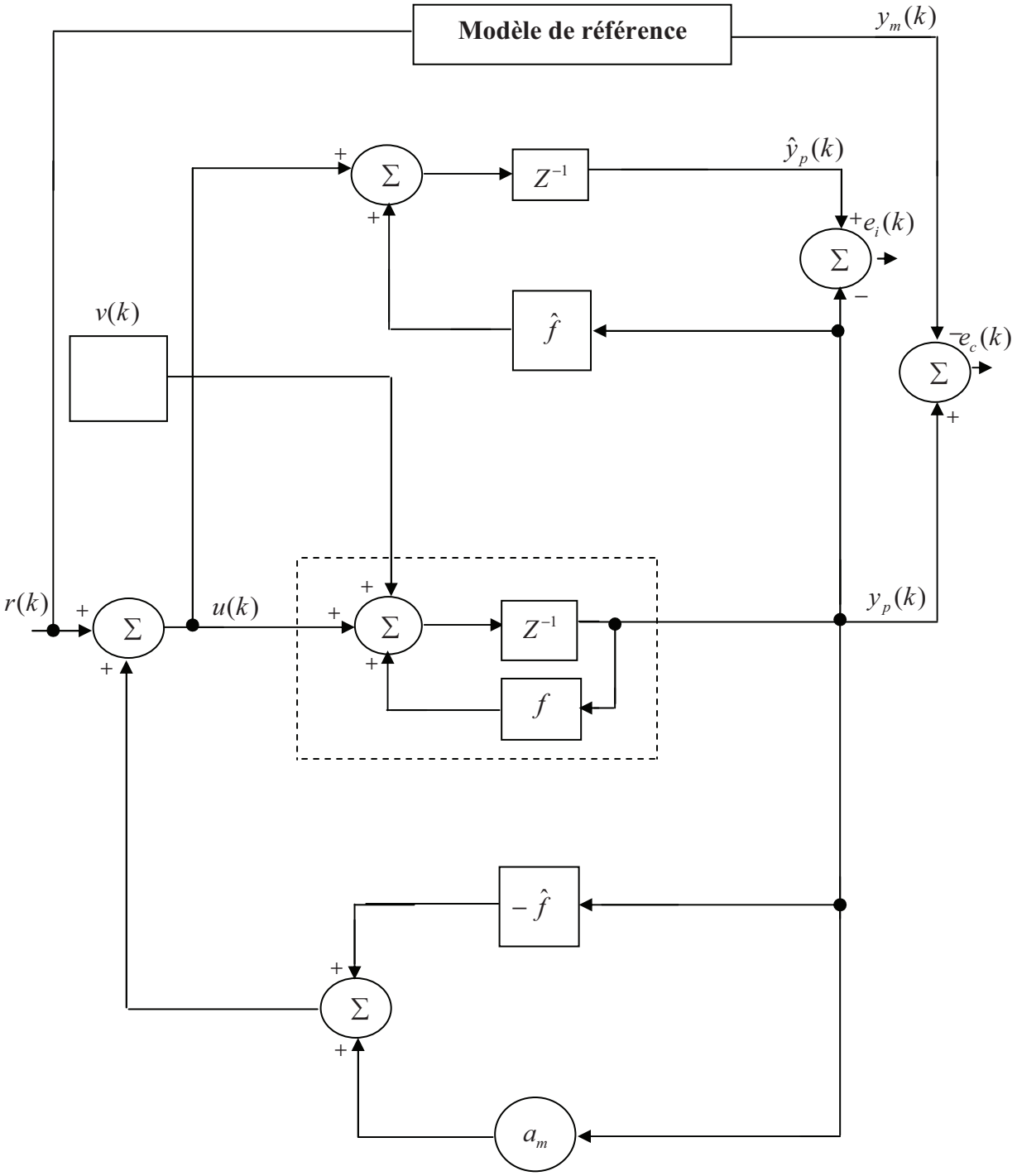


Figure.IV-29 : pas de rejection de la perturbation pour q égal à 0

VIII.2.5. La structure de tout le système q égal à 1



VIII.2.6. La structure de tout le système ou q égal à 0



VIII.2.7. La rejection de la perturbation par les réseaux de neurone

- ***Système 2-2***

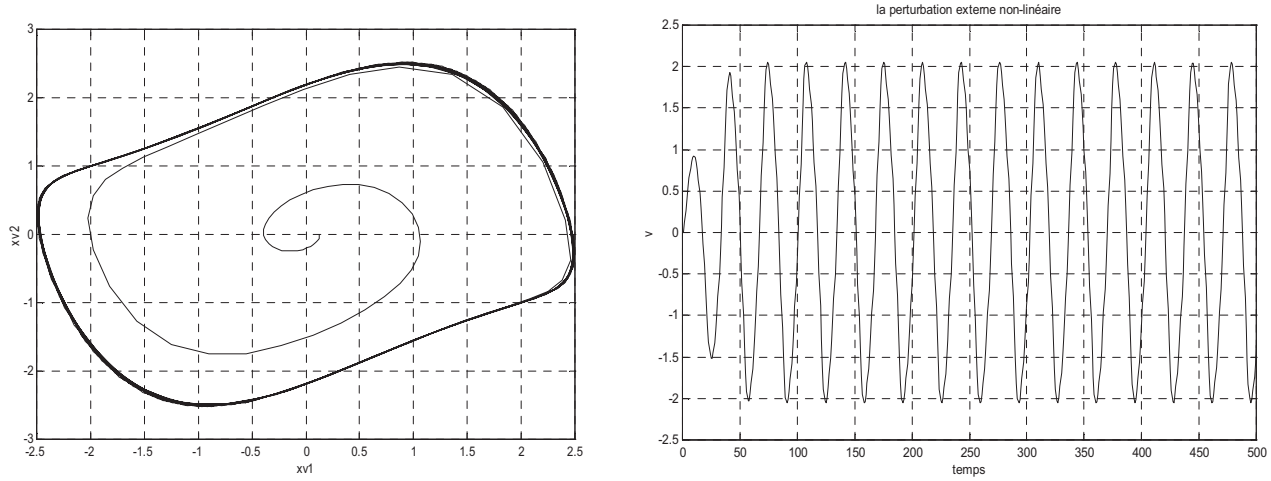
Le système du deuxième ordre dans la présence d'une perturbation non linéaire :

$$y(k+1)=\frac{\cos(y(k))}{1+\sin(y(k-1))^2}+\sin(y(k-1))\exp(-y(k)^2)+\frac{y(k)}{1+y(k)^2}+u(k)+v(k) \qquad \text{(IV-11)}$$

$v(k)$: c'est la perturbation externe du deuxième ordre définit comme la sortie de système non linéaire de van der pol.

$$\begin{aligned} x_{v1}(k+1) &= x_{v1}(k)+0.2x_{v2}(k) \\ x_{v2}(k+1) &= -0.2x_{v1}(k)+x_{v2}(k)-0.1(x_{v1}(k)^2-1)x_{v2}(k) \\ v(k) &= x_{v1}(k) \end{aligned} \qquad \text{(IV-12)}$$

✓ ***La forme de la perturbation***



(a) (b)

Figure.IV- 30 : :(a) perturbation : trajectoire dans $\Re^2$ et (b) x_{v1} en fonction de temps.

La représentation entrée-sortie du système dans la présence de la perturbation est :

$$y_{inc}(k+1)=f_{inc}[y(k),y(k-1),...,y(k-3),u(k-1),...,u(k-2)]+u(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k), y(k-1),..., y(k-q-1), u(k-1), u(k-2),..., u(k-q)] + u(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f_{inc}, pour une identification exacte on choisis $q \geq 2$, l'identification dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-2,2].

La forme de ce réseau neuronal est décrite par : $N^3_{(2q+2),40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec $(2q+2)$ neurone d'entrée et une seule sortie. Avec un pas d'apprentissage égal à 0.04 et après un temps d'exécution égal à 2.7227e+003s et nombre d'itérations égal à 32789, de 3 différentes valeurs de q $(q = 0, q = 2, q = 4)$ sont utilisées pour la comparaisons.

L'objectif est d'atteindre la sortie désirée :

$$y_m(k+1) = 0.6 y_m(k) + r(k)$$

Telle que l'entrée référence est :

$$r(k) = \sin((2\Pi k)/10)) + \sin((2\Pi k)/25))$$

✓ ***La commande d'entrée est calculé par***

$$u_r(k) = y_m(k+1) - NN[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)]$$

Telle que $u_r(k)$ est la commande de poursuite qui fait la rejection de la perturbation

Si $q = 2$ donc rejection complète erreur très petite.

Si $q = 4$ rejection pareil à $q = 2$.

VIII.2.8. La rejection par la logique floue

- ***Système 2-2***

Pour identifier le système avec la perturbation, on propose le modèle d'identification floue :

$$\hat{y}(k+1) = Fs[y(k), y(k-1),..., y(k-3), u(k-1),..., u(k-2)] + u(k)$$

où $Fs(.)$ est réalisé par la logique floue comme une approximation à f_{inc}, pour une identification exacte on choisis $q \geq 2$, l'identification dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-2, 2].

Pour l'identification floue on a choisi les paramètres suivants : après 3000 itérations et temps d'exécution égal à 2.2636e+003s pour minimiser l'erreur avec 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ AND\ y(k-1)\ est\ A_2^i\ AND\ y(k-2)\ est\ A_3^i\ AND\ y(k-3)\ est\ A_4^i$$
$$AND\ u(k-1)\ est\ A_5^i\ \ AND\ u(k-2)\ est\ A_6^i\ THEN\ y(k+1)\ \ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 1.
- Gain d'ajustement des prémisses 0.4.
- Gain d'ajustement des conséquences 0.05.

✓ ***La commande de poursuite d'entrée est calculé par***

$$u_r(k) = (y_m(k+1) - Fs(..))$$

$u_r(k)$: La commande de poursuite qui fait la rejection.

Si $q = 2$: rejection complète de la perturbation.

a- rejection neuronale ***b- rejection floue***

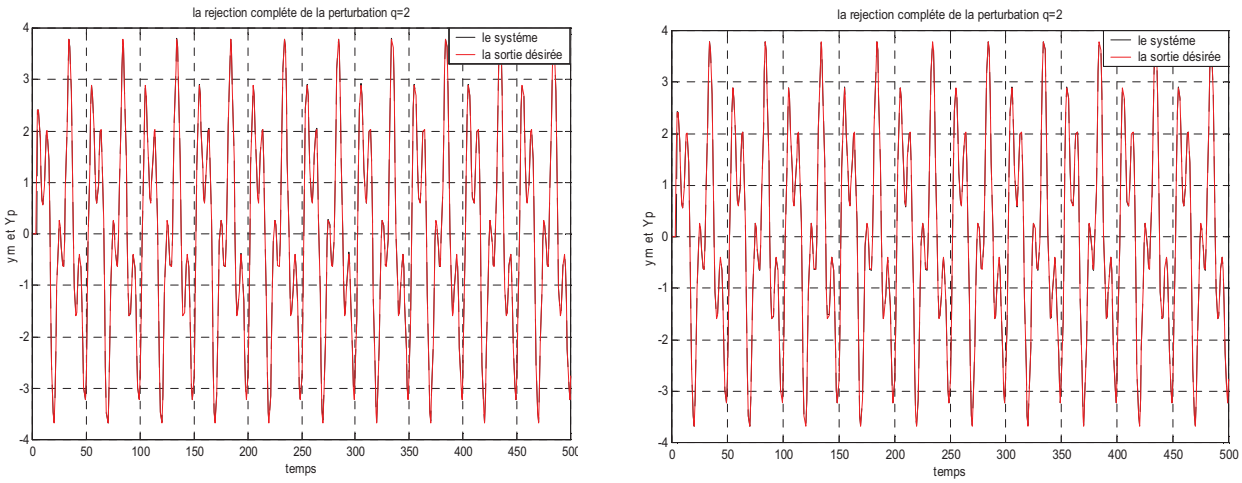


Figure.IV- 31: une rejection complète de la perturbation pour q égal à 2

a- la commande neuronale ***b- la commande floue***

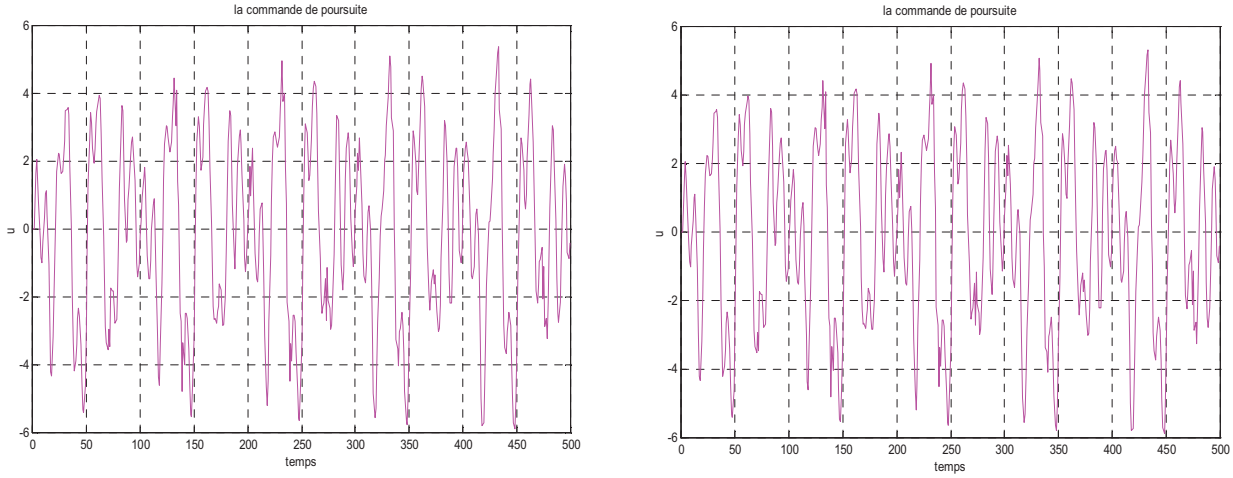


Figure.IV- 32 : la commande de poursuite

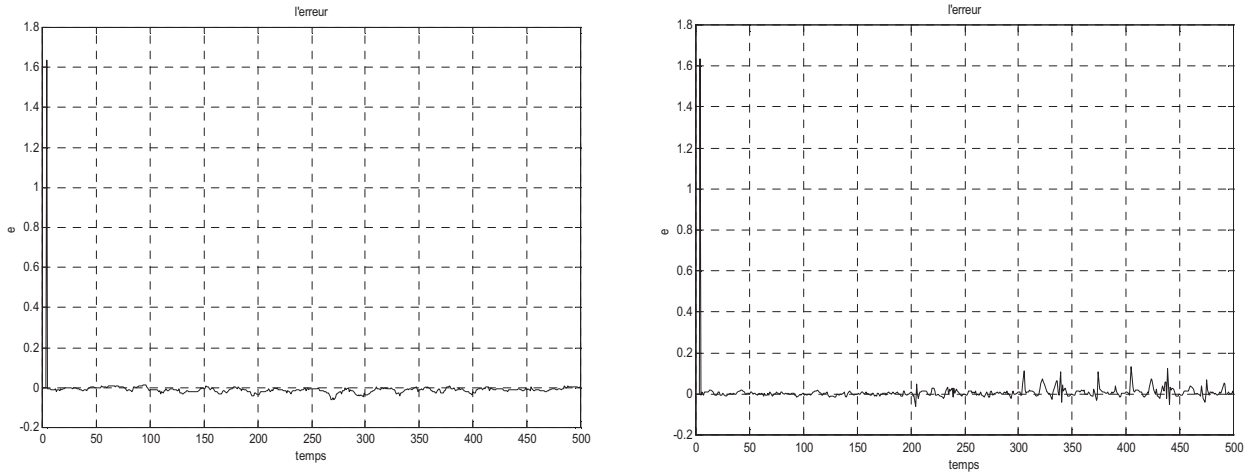


Figure.IV-33 :l'erreur entre le modèle du référence et le système

VIII.2.9. Pas de rejection de la perturbation q=0

- ***Par MNN***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k), y(k-1)] + u(k) + v(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k), y(k-1)] + u(k) + v(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f, l'identification se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-2,2].

La forme de ce réseau neuronal est décrite par : $N^{3}_{(2),40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec deux entrées et une seule sortie et un pas d'apprentissage égal à 0.05.

✓ ***La commande d'entrée est calculé par***

$$u(k) = y_m(k+1) - NN[y(k), y(k-1)]$$

Le résultat de la commande montre que :
Si $q = 0$ donc pas de réjection de la perturbation.

❖ ***Par Fs***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k), y(k-1)] + u(k) + v(k)$$

Pour identifier le système, on propose le modèle d'identification :

$$\hat{y}(k+1) = Fs[y(k), y(k-1)] + u(k) + v(k)$$

où $Fs(.)$ est réalisé par la logique floue comme une approximation à f, pour l'identification floue on a choisi les paramètres suivants : pour minimiser l'erreur 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ \ AND\ \ y(k-1)\ est\ \ A_2^i\ \ THEN\ y(k+1)\ \ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.5.
- Gain d'ajustement des prémisses 0.01.
- Gain d'ajustement des conséquences 0.05.

L'identification se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-2, 2].

✓ ***La commande d'entrée est calculé par***

$$u(k) = y_m(k+1) - Fs[y(k), y(k-1)]$$

VIII.2.10. Pas de rejection de la perturbation

a- Par MNN ***b- Par Fs***

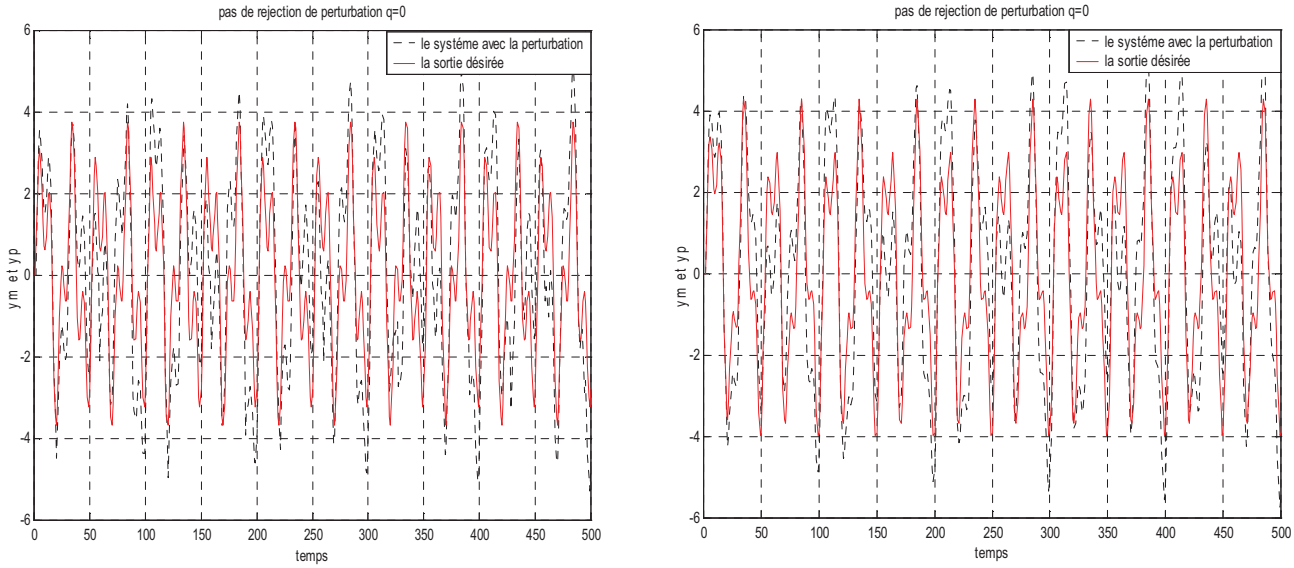
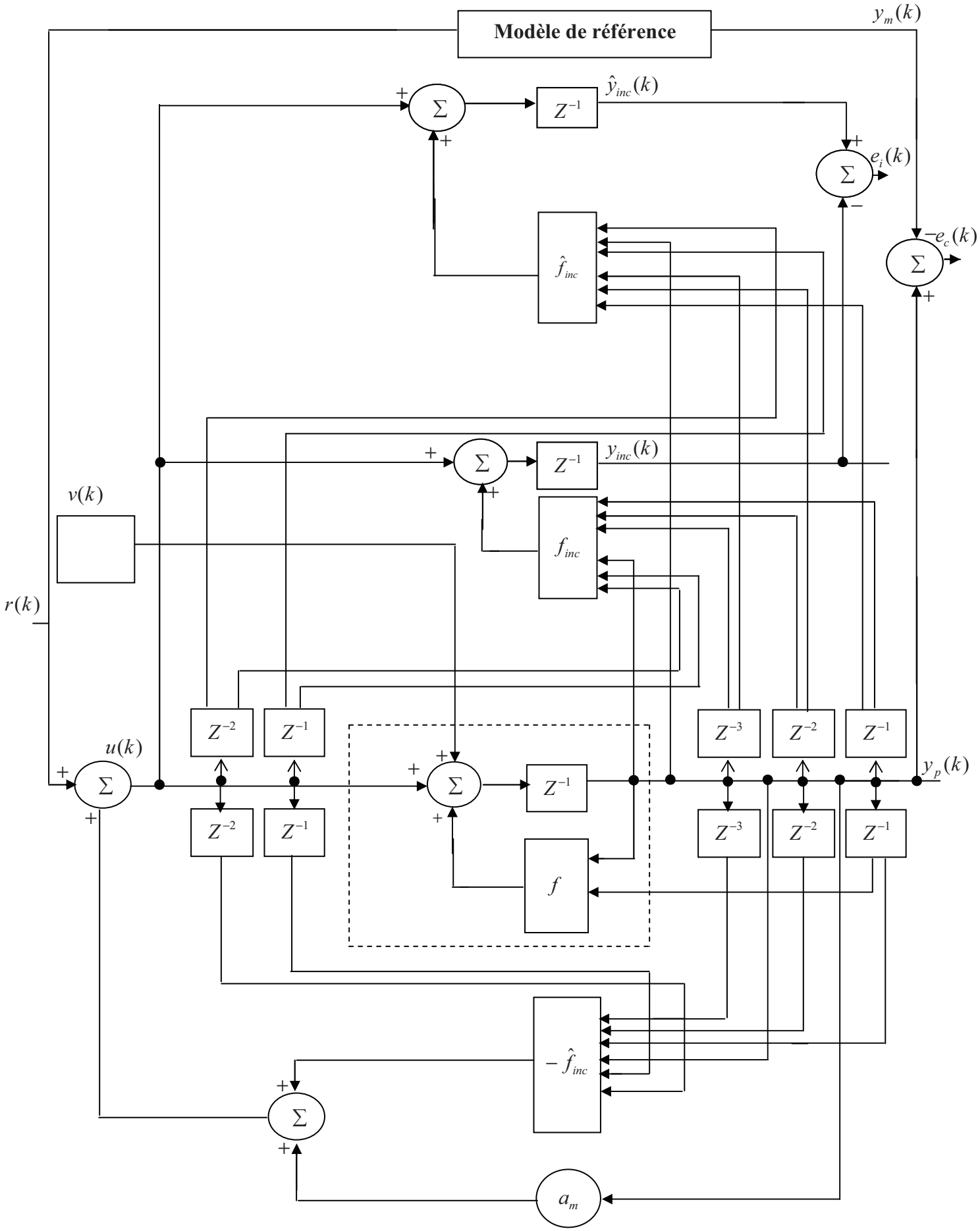
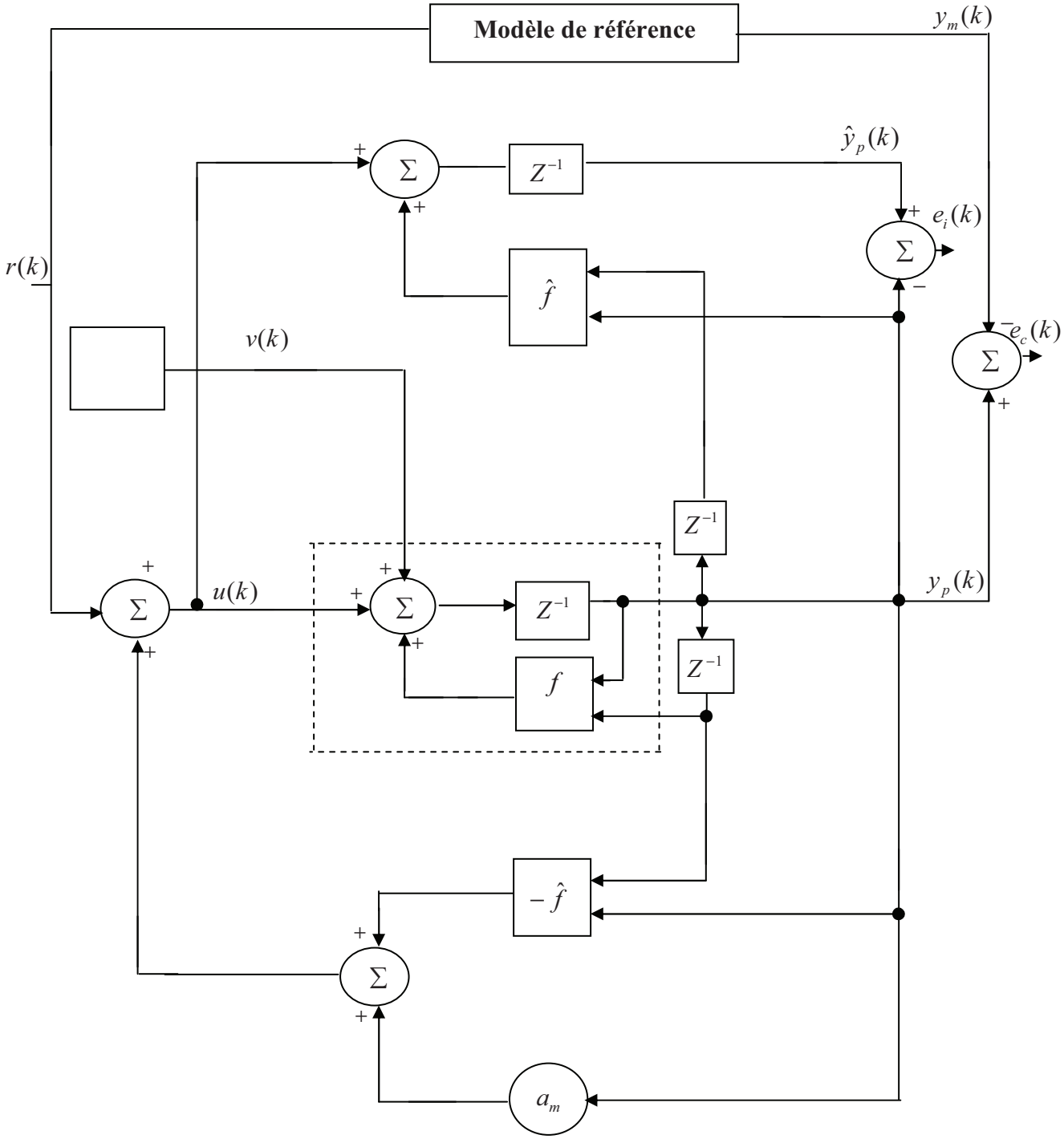


Figure.IV-34 : pas de rejection de la perturbation pour q égal à 0

VIII.2.11. La structure de tout le système où q égal à 2



VIII.2.12. La structure de tous le système où q égal à 0



VIII.3. Étape 3

VIII.3.1. La rejection de la perturbation par les réseaux de neurone

Le système du troisième ordre dans la présence d'une perturbation linéaire:

$$\begin{aligned} y(k+1) &= \theta_1\left\{y(k)+\sin[5y(k-1)]\exp^{-\cos[y(k-2)]}\right\} \\ &+ \theta_2\left\{y(k)+y(k-1)+y(k-2)+\sin[y(k)]\cos[2y(k-1)+3y(k-2)]\right\} \\ &+ D(z)[u(k)+v(k)] \end{aligned} \qquad \text{(IV-13)}$$

Sachant que :

$D(z)$: Un polynôme qui doit être stable donc :

$D(z) = 2.2Z^{-1} + \theta_3 Z^{-2} + \theta_4$, alors $\theta_1 = 0.05$, $\theta_2 = -0.5$, $\theta_3 = 1$, $\theta_4 = 1.3$.

$v(k)$: c'est la perturbation linéaire du deuxième ordre considéré comme un sinusoïdes et peut être représenté par :

$$\begin{aligned} v1(k+1) &= \cos(\Psi)v1(k) + \sin(\Psi)v2(k) \\ v2(k+1) &= \cos(\Psi)v2(k) - \sin(\Psi)v1(k) \\ v(k) &= v_1(k) \end{aligned} \qquad \text{(IV-14)}$$

Telle que $(2\Pi)/\Psi$ c'est la période de la perturbation sinusoïdale.

Dans la simulation Ψ a été choisis égal à $\Pi/6$ et les conditions initiale de la perturbation sont choisis comme $[v_{10}, v_{20}] = [1.2, -1.6]$ donc c'est une perturbation d'amplitude égal à 2 et fréquence 12 .

✓ ***La forme de la perturbation***

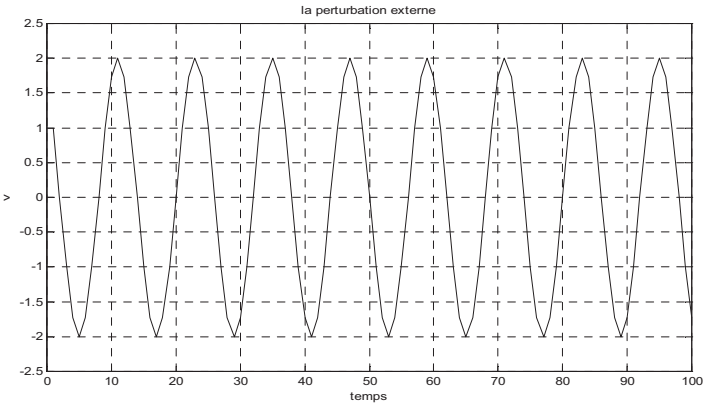


Figure.IV- 35:la perturbation $v(k)$ de deuxième ordre

La représentation entrée-sortie du système dans la présence de la perturbation est :

$$y_{inc}(k+1) = f_{inc}[y(k), y(k-1),..., y(k-4), u(k-1), u(k-2),...u(k-4))] + 2.2u(k-1) + u(k-2) + 1.3u(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)] + 2.2u(k-1) + u(k-2) + 1.3u(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f_{inc}, pour une identification exacte on choisis $q \geq 2$, parce que la perturbation de deuxième ordre et le système de troisième ordre donc le modèle f_{inc} à définir pour la rejection de la perturbation il faut qu'il soit d'ordre 5 et plus, l'identification de modèle f_{inc} dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-2,2].

La forme de ce réseau neuronal est décrite par : $N^3_{(2q+5),40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec $(2q+5)$ neurone d'entrée et une seule sortie. Avec un pas d'apprentissage égal à 0.007, les résultats sont obtenus après un temps d'exécution 2.9899e+003s et nombre d'itérations égal à 10603.

L'objectif est d'atteindre la sortie du référence y_m telle que :

$$y_m(k+1) = 0.6y_m(k) + 0.2y_m(k-1) + 0.1y_m(k-2) + r(k)$$

Et l'entrée référence est :

$$r(k) = 0.75\sin((2\Pi k)/50)) + \sin((2\Pi k)/20))$$

- ✓ ***La commande de poursuite est calculé par***

$$u(k) = (y_m(k+1) - NN[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)] - 2.2u(k-1) - u(k-2))/1.3$$

Telle que $e_c = y_m - y_p$ tend vers zéro

Le résultat de la commande montre que :

Si $q = 2$: une rejection complète de la perturbation.

VIII.3.2. La rejection par la logique floue

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = Fs[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)] + 2.2u(k-1) + u(k-2) + 1.3u(k)$$

où $Fs(.)$ est réalisé par la logique floue comme une approximation à f_{inc}, pour une identification exacte on choisis $q \geq 2$, l'identification de modèle f_{inc} dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-2,2].

Pour l'identification floue on a choisi les paramètres suivants : après temps d'exécution égal à 4.8322e+003s et nombre d'itérations égal à 2566 pour minimiser l'erreur avec 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ AND\ y(k-1)\ est\ A_2^i\ AND\ y(k-2)\ est\ A_3^i\ AND\ y(k-3)\ est\ A_4^i\ AND\ y(k-4)\ est\ A_5^i$$
$$AND\ u(k-1)\ est\ A_6^i\ AND\ u(k-2)\ est\ A_7^i\ AND\ u(k-3)\ est\ A_8^i\ AND\ u(k-4)\ est\ A_9^i\ THEN\ y(k+1)\ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 2.5.
- Gain d'ajustement des prémisses 0.1.
- Gain d'ajustement des conséquences 0.1.

✓ ***La commande de poursuite est calculé par***

$$u(k) = (y_m(k+1) - Fs[y(k), y(k-1),..., y(k-q), u(k-1), u(k-2),..., u(k-q)] - 2.2u(k-1) - u(k-2))/1.3$$

Si $q = 2$: une rejection complète de la perturbation.

a- rejection neuronale ***b- rejection floue***

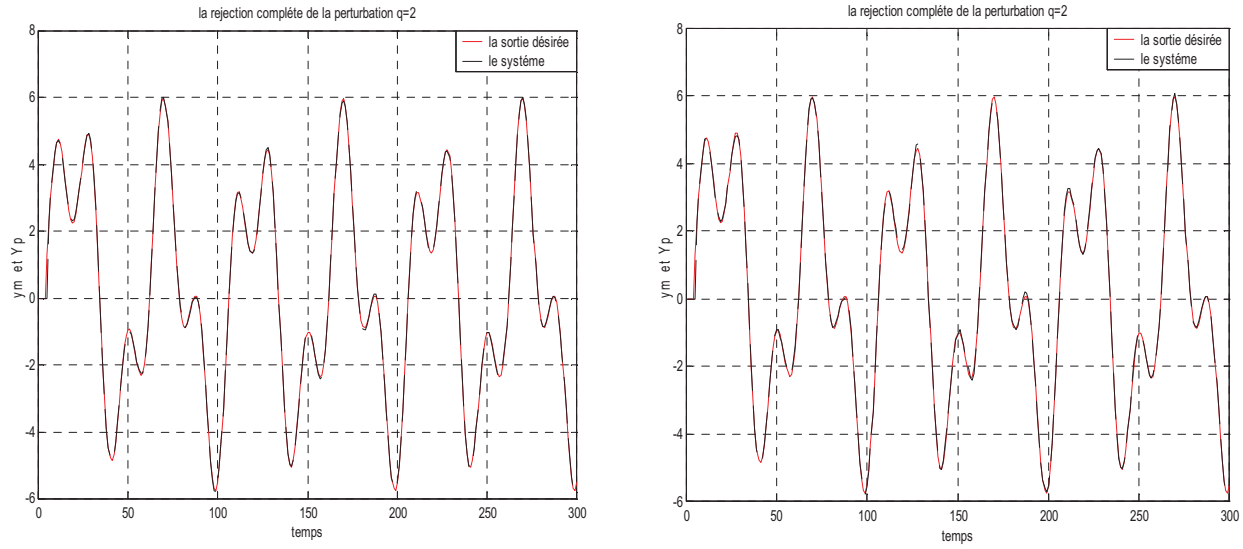


Figure.IV- 36 : la rejection complète de la perturbation pour $q = 2$; y_m identique à y_p

a- la commande neuronale ***b- la commande floue***

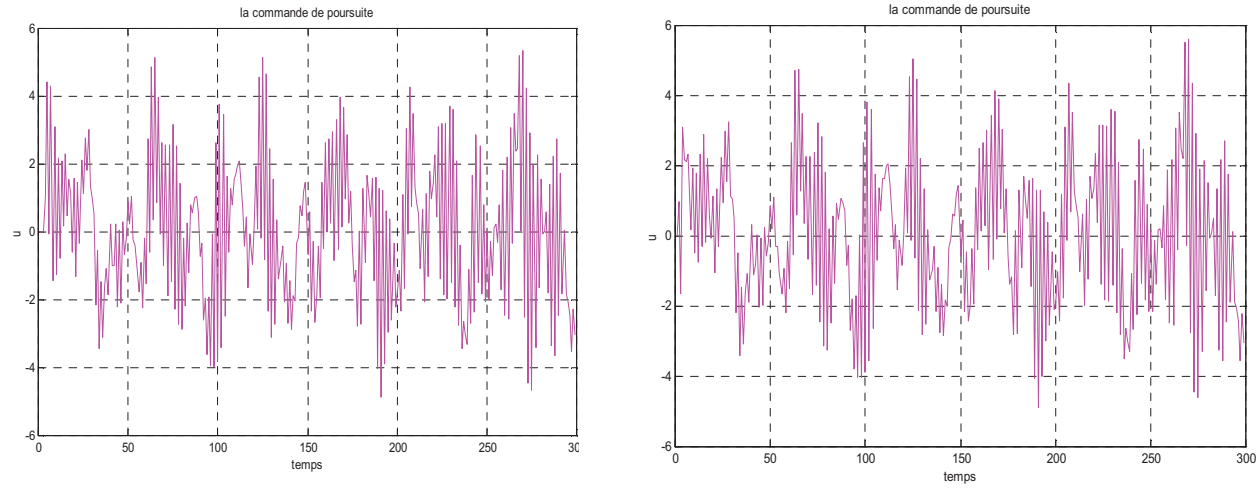


Figure.IV- 37 : la commande de poursuite utilisé pour la rejection de la perturbation

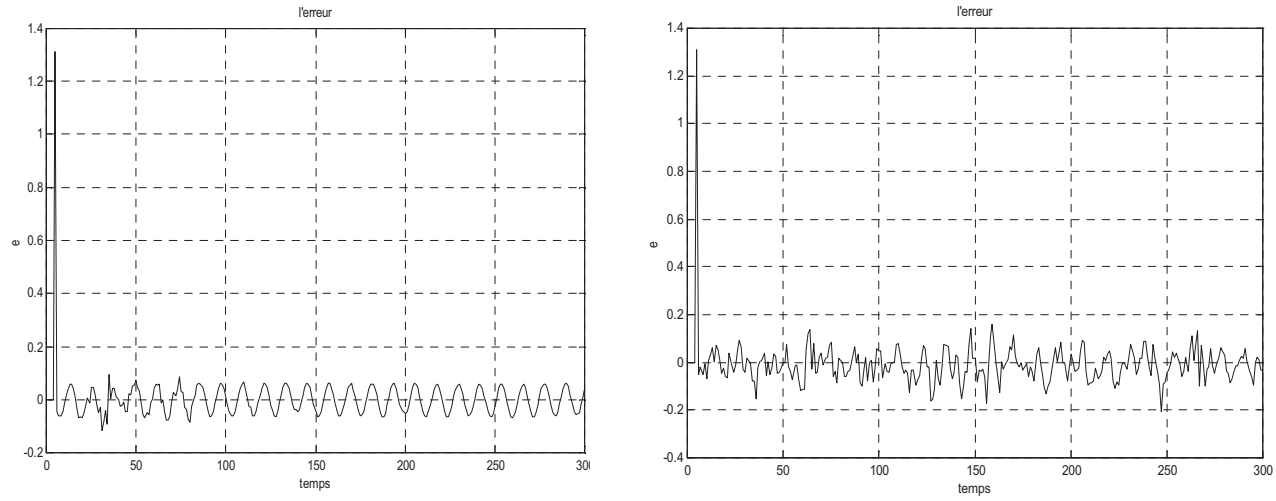


Figure.IV- 38 :l'erreur entre le modèle du référence et le système

VIII.3.3. Pas de rejection de la perturbation q=0

- ***Par MNN***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k), y(k-1), y(k-2), u(k-1), u(k-2)] + 2.2v(k-1) + v(k-2) + 1.3v(k) + 1.3u(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = NN[y(k), y(k-1), y(k-2), u(k-1), u(k-2)] + 2.2v(k-1) + v(k-2) + 1.3v(k) + 1.3u(k)$$

où $NN(.)$ est réalisé par MNN comme une approximation à f, l'identification de modèle f dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-2,2].

La forme de ce réseau neuronal est décrite par : $N^{3}_{(5),40,20,1}$ c'est-à-dire que le réseau comporte 3 couches avec cinq entrée et une seule sortie. Avec un pas d'apprentissage égal à 0.007, les résultats sont obtenus après un temps d'exécution égal à 6.8518e+003s et nombre d'itérations égales à 22123 itérations.

- ✓ ***La commande d'entrée est calculé par***

$$u(k) = (y_m(k+1) - NN[y(k), y(k-1), y(k-2), u(k-1), u(k-2)])/1.3$$

Le résultat de la commande montre :
Si $q = 0$: pas de rejection de la perturbation

- ***Par Fs***

La représentation entrée-sortie du système est :

$$y(k+1) = f[y(k), y(k-1), y(k-2), u(k-1), u(k-2)] + 2.2v(k-1) + v(k-2) + 1.3v(k) + 1.3u(k)$$

Pour identifier le système avec la perturbation, on propose le modèle d'identification :

$$\hat{y}(k+1) = Fs[y(k), y(k-1), y(k-2), u(k-1), u(k-2)] + 2.2v(k-1) + v(k-2) + 1.3v(k) + 1.3u(k)$$

où $Fs(.)$ est réalisé par la logique floue comme une approximation à f .

L'identification de modèle f dans la présence de la perturbation se fait on choisis l'entrée $u(k)$ par des valeurs aléatoires entre [-2,2].

Avec 20 règles de la forme :

$$R_i : IF\ y(k)\ est\ A_1^i\ AND\ y(k-1)\ est\ A_2^i\ AND\ y(k-2)\ est\ A_3^i$$
$$AND\ \ u(k-1)\ est\ A_4^i\ \ AND\ \ u(k-2)\ est\ A_5^i\ \ THEN\ y(k+1)\ \ est\ b_i$$

- Fonctions d'appartenances gaussiennes d'écart type 0.9.
- Gain d'ajustement des prémisses 0.002.
- Gain d'ajustement des conséquences 0.007

✓ ***La commande d'entrée est calculé par***

$$u(k) = (y_m(k+1) - Fs[y(k), y(k-1), y(k-2), u(k-1), u(k-2)])/1.3$$

Le résultat de la commande montre que :

Si $q = 0$: pas de rejection de la perturbation

VIII.3.4. Pas de rejection de la perturbation

a- Par MNN ***b-Par Fs***

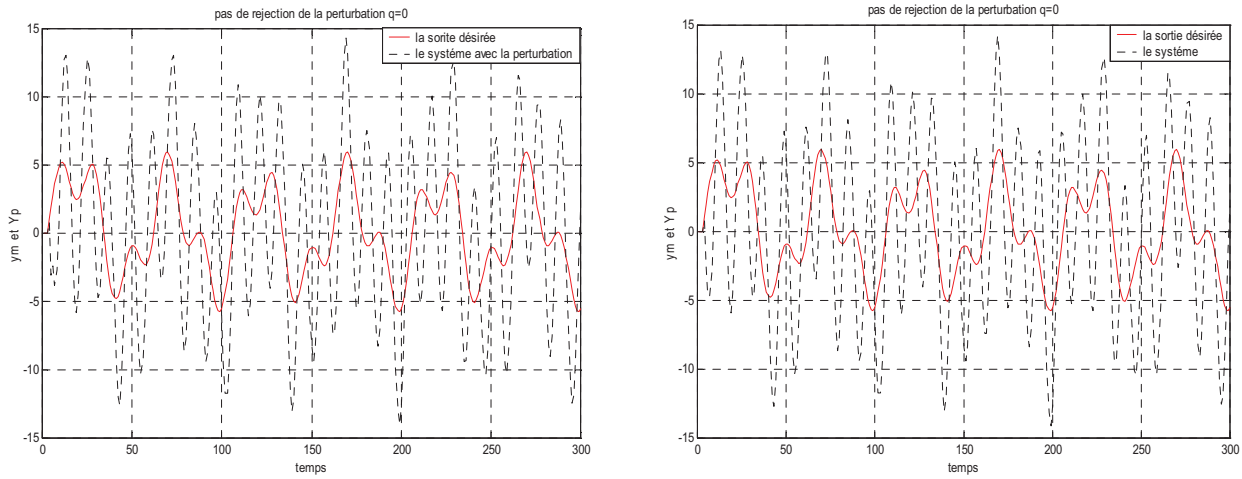
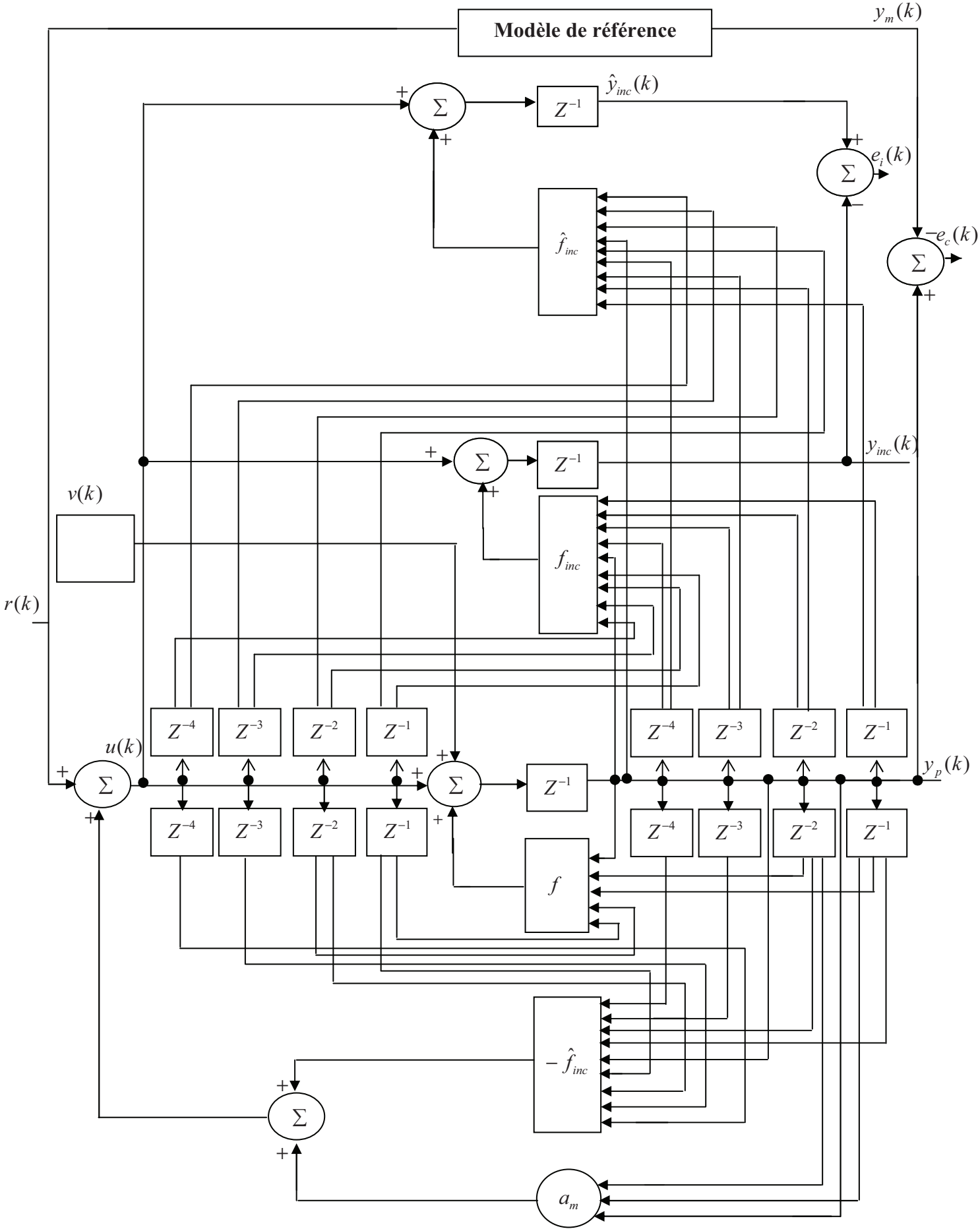
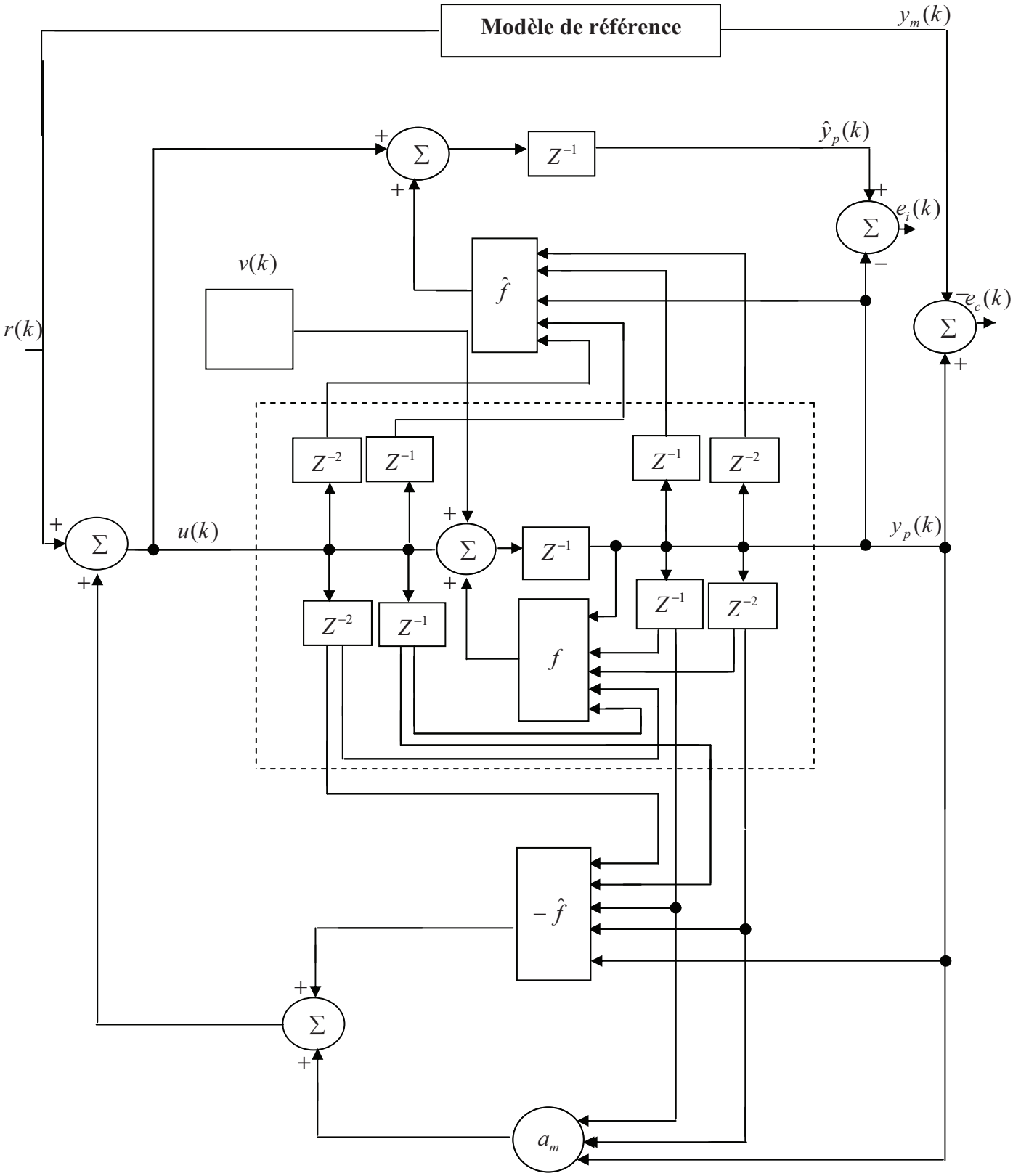


Figure.IV-39 : pas de rejection de la perturbation pour q égal à 0

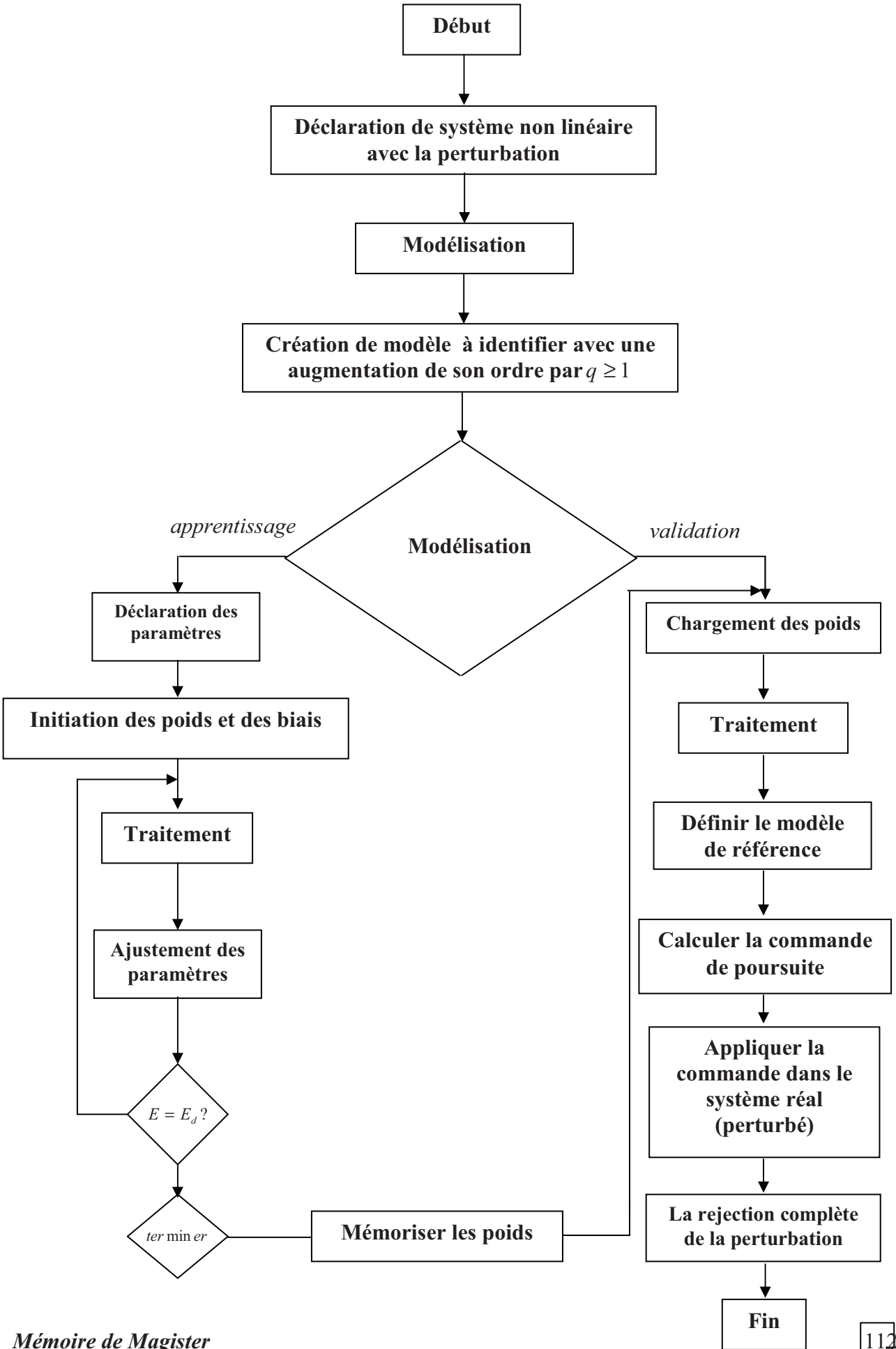
VIII.3.5. La structure de tout le système q égal à 2



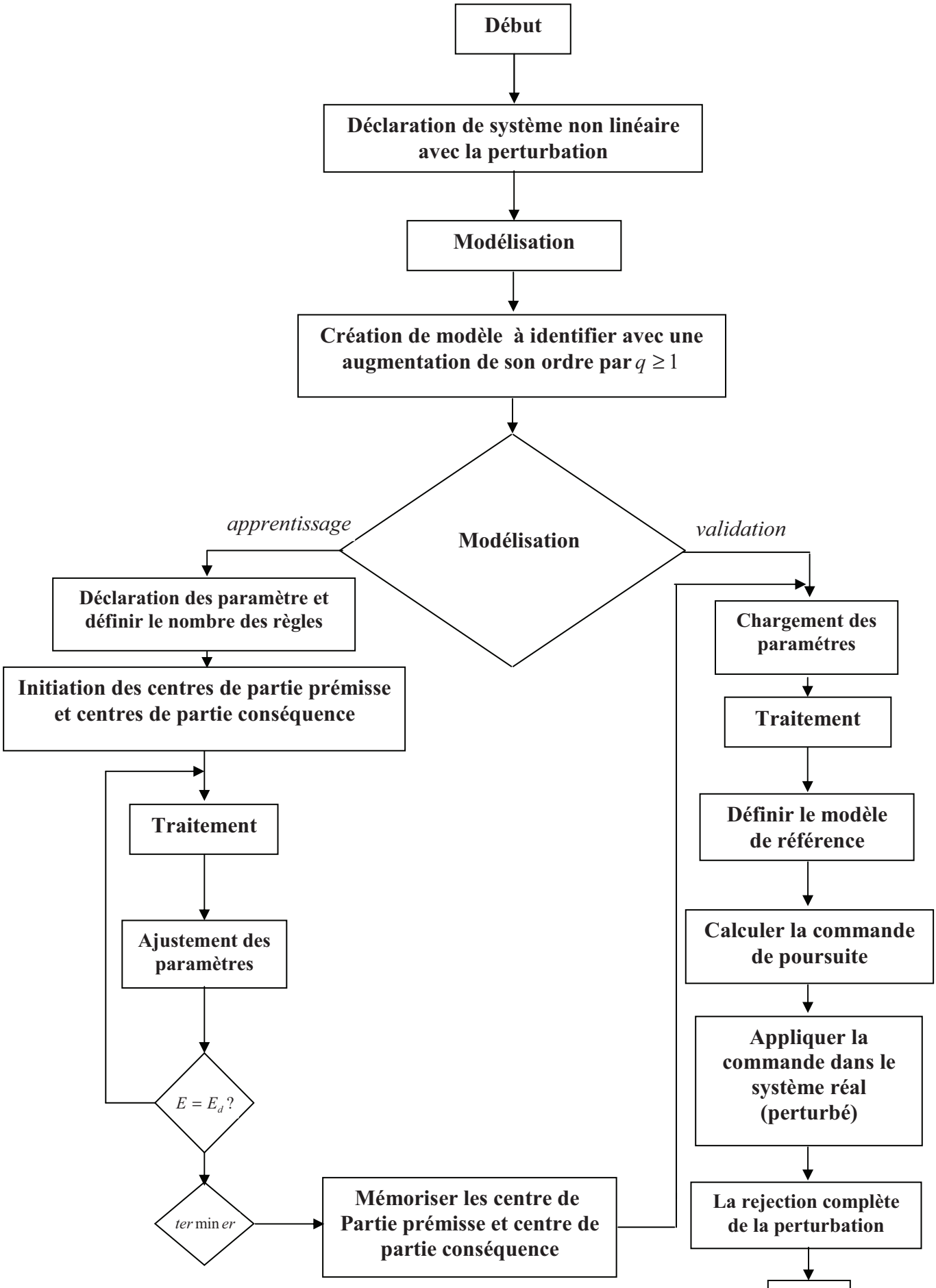
VIII.3.6. La structure de tout le système q égal à 0



IX. Organigramme dans le cas de rejection de perturbation par les réseaux de neurone



X. Organigramme dans le cas de rejection de perturbation par la logique floue



XI. Interprétation des résultats

XI.1. Identification

Nous avons étudié l'identification des systèmes non linéaires selon leur complexité. Le système 1 qui est le plus simple est un système TISO (Two Input Single Output), mais la partie non linéaire à identifier est un système SISO (Single Input Single Output) d'entrée $u(k)$ et de sortie $\hat{f}$. Le système 2 est un système TISO (Two Input Single Output), la partie à identifier est un système TISO d'entrée $[\, y(k), y(k-1)\,]$ et de sortie $\hat{f}$. Le système 3 est un système SISO complexe contient deux partie non linéaire à identifier F et G, F (SISO) d'entrée $[\, y(k)\,]$ et de sortie $\hat{f}$, G (SISO) d'entrée $[\, u(k)\,]$ et de sortie $\hat{g}$. Le dernier système est le plus complexe, car la partie non linéaire à identifier est un système MISO d'entrées $[\, y(k),\ y(k-1),\ y(k-2), u(k), u(k-1)\,]$ et de sortie $y(k+1)$.

La Figure (IV-1) montre les résultats de la simulation du système 1 pour les deux méthodes, dans la phase d'apprentissage, nous remarquons que les deux modèles convergents vers la sortie du système 1. Comme excitation nous avons pris pour les deux cas la même entrée aléatoire. Pour comparer les résultats entre les deux méthodes nous avons opté comme critère la moyenne de la somme des erreurs au carré MSE (Mean Square Error). Pour l'identification floue le MSE égal à 9.9702e-005, et pour l'identification neuronale le MSE égal à 2.3727e-004, ce qui est un avantage pour l'identification floue.

La Figure (IV-3) montre les résultats d'identification du système 2 pour les deux méthodes. Nous remarquons toujours une convergence des sorties des modèles vers la sortie du système 2. Comme excitation nous avons pris pour les deux cas la même entrée aléatoire. Pour l'identification floue nous avons trouvé MSE égal à 0.001, et pour l'identification neuronale nous avons obtenu MSE égal à 6.7267e-004, ce qui est un avantage pour l'identification neuronale mais si en comparant le temps de calcul nous allons trouver que pour l'identification floue 967.4380s avec 6000 itérations par contre pour l'identification neuronale 1.3111e+003s avec 10000 itérations .Ce qui est un inconvénient pour l'identification neuronale.

Pour la commande neuronale nous avons trouvé la somme des erreurs moyennes entre la sortie désiré et la sortie du système est de 5.6021e-005 pour la commande floue MSE égal à 4.7050e-004 (figure.IV-10), toujours un avantage pour les réseaux de neurone mais si en compare

le temps d'exécution aussi on trouve que la logique floue est plus rapide ce qui est un avantage pour la logique floue.

La Figure (IV-5) montre les résultats d'identification du système 3. La convergence des modèles vers le système est toujours assurée. Comme excitation nous avons pris pour les deux cas la même entrée aléatoire pour F et G. Pour l'identification floue de F et G nous avons trouvé MSE égal à 4.9295e-005, tandis que pour l'identification neuronale nous avons obtenu MSE égal à 1.2289e-005. Les résultats montrent que l'erreur d'identification floue atteint des pics supérieurs à ceux de l'identification neuronale.

Pour la commande floue MSE égal à 0.0366 et la commande neuronale MSE égal à 0.0339 sont plus proche (Figure.IV-13), mais par la comparaisons de nombre d'itérations et le temps de calcul on trouve que l'identification floue toujours plus rapide.

La Figure (IV-7) montre les résultats d'identification du système 4. La convergence des modèles vers le système est toujours assurée. Comme excitation nous avons pris pour les deux cas la même entrée aléatoire toujours. Pour l'identification floue nous avons trouvé MSE égal à 1.8362e-005, et pour l'identification neuronale nous avons obtenu MSE égal à 4.2842e-004. Ici sans discussion l'identification floue est très mieux.

XI.2. La rejection de la perturbation

- ***Étape 1***

Pour la rejection de la perturbation par les réseaux de neurone et la logique floue des systèmes non linéaires selon leur complexité et la complexité de la perturbation. Système 1-1 est un système SISO avec une perturbation sinusoïdale de deuxième ordre, la partie à identifier est un système MISO d'entrée $[y(k), y(k-1), y(k-2), u(k-1), u(k-2)]$ et de sortie $\hat{f}_{inc}$. Le Système 1-2 est un système SISO complexe mais avec une perturbation linéaire de quatrième ordre ce pour cela pour rejeter cette perturbation il faut que la partie non linéaire à identifier soit un système MISO d'entrée :

$[y(k), y(k-1), y(k-2), y(k-3), y(k-4), u(k-1), u(k-2), u(k-3), u(k-4)]$ et de sortie $\hat{f}_{inc}$.

❖ ***Étape 2***

Le système 2-1 qui est le même système 1-1 dans l'étape précédente mais avec une perturbation non linéaire de premier ordre, mais pour la rejection la partie non linéaire à identifier est un système MISO d'entrée $[y(k), y(k-1), u(k-1)]$ et de sortie $\hat{f}_{inc}$. Le système 2-2 est le plus complexe, car il est un système TISO et la perturbation non linéaire de van der pol donc la partie non linéaire à identifier est un système MISO d'entrées :

$[y(k), y(k-1), y(k-2), y(k-3), u(k-1), u(k-2)]$ et de sortie $\hat{f}_{inc}$.

❖ ***Étape 3***

Le dernier système c'est le plus difficile et plus complexe car il est un système MISO d'origine avec une perturbation sinusoïdale de deuxième ordre donc la partie à identifier est un système non linéaire MISO de 9 entrées :

$[y(k), y(k-1), y(k-2), y(k-3), y(k-4), u(k-1), u(k-2), u(k-3), u(k-4)]$ et de sortie $\hat{f}_{inc}$.

XII. La discussion

❖ ***Étape 1***

La Figure (IV-16) montre les résultats de la simulation du Système 1-1 pour les deux méthodes, dans la phase de la rejection, nous remarquons que le résultat de la commande montre que la sortie du système converge vers la sortie désirée pour les deux méthodes. Comme excitation nous avons pris pour les deux cas la même entrée aléatoire. Pour comparer les résultats entre les deux méthodes nous avons opté comme critère MSE. Pour la rejection floue le MSE égal à 0023, et pour la rejection neuronale le MSE égal à 0.005. Ce qui est un avantage pour la rejection de la perturbation par la logique floue.

La Figure (IV-21) montre les résultats de la rejection de la perturbation du système 1-2 par les deux méthodes. Nous remarquons toujours une convergence de la sortie du système vers la sortie désirée. Comme excitation nous avons pris pour les deux cas la même entrée aléatoire. Pour la rejection floue nous avons trouvé MSE égal à 0.0024, et pour la rejection neuronale nous avons obtenu MSE égal à 0.0038, ce qui est un avantage pour la rejection floue.

❖ ***Étape 2***

La Figure (IV-26) montre les résultats de la rejection de la perturbation du système 2-1 par les deux méthodes. Nous remarquons toujours une convergence de la sortie du système vers la sortie désirée. Comme excitation nous avons pris pour les deux cas la même entrée aléatoire. Pour la rejection floue nous avons trouvé MSE égal à 3.2069e-004s, et pour la rejection neuronale nous avons obtenu MSE égal à 0.0014, ce qui est un avantage pour la rejection floue. On trouve que la logique floue toujours mieux.

La Figure (IV-31) montre les résultats de la rejection de la perturbation du système 2-2. La convergence de la sortie du système vers la sortie désirée est toujours assurée. Comme excitation nous avons pris pour les deux cas la même entrée aléatoire toujours. Pour l'identification floue nous avons trouvé MSE égal à 0.0058, et pour l'identification neuronale nous avons obtenu MSE égal à 0.0056. Ici la rejection neuronale mieux mais si on compare le résultat par le temps de calcul on trouve que le temps d'exécution de la rejection floue égal à 2.2636e+003s après 3000 itérations et le temps d'exécution de la rejection neuronale égal à 6.27227e+003s après 32789 itérations donc une grande différence ce pour cela la logique floue est préférable.

❖ ***Étape 3***

La Figure (IV-36) montre les résultats de la rejection de la perturbation du système du l'étape 3. La convergence de la sortie du système vers le modèle de référence est toujours assurée. Comme excitation nous avons pris pour les deux cas la même entrée aléatoire toujours.

Pour la rejection floue nous avons trouvé MSE égal à 0.0077, et pour la rejection neuronale nous avons obtenu MSE égal à 0.0094. Ici la rejection neuronale est mieux aussi si on compare le résultat par le temps de calcul on trouve que le temps d'exécution de la rejection floue égal à 5.8322e+003s (97,2033 minutes) après 2866 itérations et le temps d'exécution de la rejection neuronale égal à 2.9899e+003s (49,8317 minutes) après 22123 itérations donc une grande différence, presque le double par rapport la rejection neuronale, ce pour cela les réseaux de neurone ici est préférable.

Selon notre étude, nous remarquons que les performances des identificateurs neuronaux sont plus précises que ceux des identificateurs flous. Cela ne peut pas être absolument affirmé d'une façon générale (système 1et 4) car notre étude a traité des cas particuliers des systèmes. Mais pour la rejection nous avons vu que la logique floue a montré son efficacité que ceux de réseaux de

neurone mais aussi pas affirmé d'une façon générale (système 2-2 et le système de l'étape 3), Cela n'empêche pas de citer quelques propriétés que nous avons remarquées dans notre travail :

- Les paramètres des identificateurs flous ont des significations physiques, ce qui n'est pas vrai pour le cas neuronal.
- Dans la logique floue on peut incorporer des connaissances d'un expert à notre identificateur.
- La vitesse d'exécution des algorithmes flous est plus rapide que ceux des réseaux de neurones.
- Les identificateurs neuronaux sont moins sensibles aux conditions initiales par rapport ceux de la logique floue.
- La capacité d'approximation de la logique floue est meilleure par rapport à celle de réseaux de neurone.
- Pour la rejection de la perturbation, la logique floue a montré son efficacité pour quelques systèmes moins complexes mais pour les systèmes plus complexes comme le dernier nous pensons que les réseaux de neurone sont préférables.

XIII. Conclusion

Dans ce chapitre nous avons illustré l'identification et la commande et plus la rejection de la perturbation par la logique floue et les réseaux de neurone par la simulation, nous avons pris quelques systèmes particulier de niveaux de complexité différent, ainsi dans ce chapitre nous avons discuté les résultats obtenus de la comparaison entre les deux méthodes étudié.

CONCLUSION GÉNÉRALE

CONCLUSION GÉNÉRALE

Nous avons présenté dans ce mémoire une méthode de réjection des perturbations dans les systèmes non linéaires, on a montré que l'objectif c'est l'identification de *ADP* (*augmented plant dynamics*), dans notre travail nous avons utilisé la logique floue et les réseaux de neurone pour cet objectif.

Nous avons comparé entre la commande neuronale et la commande floue pour l'identification et la rejection de la perturbation, sachant que la perturbation est considéré comme la sortie d'un système linéaire ou non linéaire libre.

Le premier chapitre a été désigné pour les réseaux de neurone artificiel, principe et architecture.

Le deuxième chapitre a été consacré pour la logique floue, notions de base, quelques explications théoriques.

Le troisième chapitre a été orienté pour l'identification, l'étape de la modélisation, et le principe de la rejection des perturbations.

Le quatrième chapitre c'est le dernier et le plus important dans ce mémoire, il contient la description des systèmes traités avec les types des perturbations rejeté ainsi l'interprétation des résultats obtenus par la simulation sur un pc.

Ce travail a montré que les réseaux de neurone et la logique floue sont très efficaces pour l'identification et la rejection complète de la perturbation, nous avons concentré sur les valeurs de MSE et le temps de calcul comme des indices de comparaison.

Les perspectives de ce travail sont nombreuses :

- ✓ Sur l'aspect de l'identification ils existent plusieurs méthodes récentes comme la logique floue type 2 et la méthode d'optimisation des particules (PSO) basé sur la technique (SWARM).
- ✓ Sur l'aspect système on essayant d'appliquer la méthode de la rejection de la perturbation sur les systèmes physiques ; pendule inversé, bras manipulateur,…

Bibliographie

[1]-*K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems Using neural networks" , IEEE ns. Neural Networks, vol. 1, pp. 4-27, Mar. 1990.*

[2]-*K. S. Narendra and S. Mukhopadhyay, "Intelligent Control Using Neural Networks", Proceedings of the American Control Conference, pp.1069-1073, 1991.*

[3]-*S Mukhopadhyay and K. S. Narendra. "Disturbance Rejection in Nonlinear Systems Using Neural Networks", Tech. Report No. 9114, Center for Systems Science, Yale University. Also to appear in IEEE Trans. on Neural Neiworks.*

[4]-*LEBOUKH djamel ET HEZZAT abdelghani '' Commande adaptative floue'', mémoire d'ingéniorat, Institut d'électronique. Université Mohamed Boudiaf de M'sila (2004).*

[5]-*Mohamed Ryad ZEMOURI '' Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques : Application à la maintenance '', Thèse de Doctorat, Ecole Doctorale Sciences Physiques pour l'Ingénieur et Microtechniques, l'université de franche-comte, 28 novembre 2003.*

[11]-*Benoît Virole '' réseaux de neurones et psychométrie'' Editions du Centre de Psychologie Appliquée – ECPA , Juin 2001*

[12]-*Zemouri R, Racoceanu D., Zerhouni N. – a – « Réseaux de neurones récurrents à fonctions de base radiales RRFR : Application à la surveillance dynamique », Revue Systèmes /JESA, Vol. 37, N°1, pp. 49-81, 2003.*

[13]-*Eric GAUTHIER '' Utilisation des Réseaux de Neurones Artificiels pour la Commande d'un Véhicule Autonome'', Thèse de Doctorat , Informatique, Systèmes et Communications) ,de l'institut national polytechnique de Grenoble, 25 Janvier 1999.*

[17]-*Khalida BENHARIRA Et Nawel HADJ BRAHIM '' étude comparative des méthodes d 'identification des systèmes non linéaires logique floue et réseaux de neurones '' mémoire d'ingéniorat, Institut d'électronique. Université Mohamed Boudiaf de M'sila (2004).*

[22]- *H. Bühler ''Réglage par la logique floue'', PPR 1994.*

[23]-*Li-Xin Wang and Jerry M. Mendel ,''back-propagation fuzzy system as nonlinear dynamic system identifiers '' Signal and Image Processing Institute Department of Electrical Engineering-Systems Universit,y of Southern California Los Angeles, CA 90089-2564. (1992).*

[24]-*Oulaya BENNIA et Lilia MOHAMADI ''identification des systèmes non linéaires par les réseaux de neurones '', mémoire d'ingéniorat, Institut d'électronique. Université Mohamed Boudiaf de M'sila (2002).*

[25]-*Tim Callinan '' Artificial Neural Network identification and control of the inverted pendulum '', Thèse de Doctorat, august 2003.*

[26]-*Saad REBOULI et Sofiane HAMOUDI ,'' identification et contrôle de la machine asynchrone par les réseaux de neurones flous'' mémoire d'ingéniorat, Institut d'électronique. Université Mohamed Boudiaf de M'sila (2004).*

[27]-*S.Chen,C.F.N,Cowan, and P.M. Grant ,'' orthogonal least squares algorithms for radial basis function networks ,'' IEEE Trans , Neural Neutworks ,vol.2,pp,302-309,19991.*

[28]-*K. S. Narendra and K. Parthasarathy, "Stable Adaptive Control of a Class of Discrete Time Nonlinear Systems Using Radial Basis Neural Networks", Center for Systems Science, Yale University, CT, Technical Report No. 9103, 1991.*

[**29]-***S.Chen ,C.F.N .Cowan ,and P.M ,Grant '' orthogonal least squares algorithm for radial basis function networks ,''IEEE trans neural networks ,vol2 ,pp.302-309,1991.*

[30]-*S.Lee and R.M.Kil ,''A Gaussian potential function network with hierarchically self-organization learning neural networks,'' Neural Networks ,vol ,4 ,pp.207-224,1991.*

[31]-*C,J.Goh and P.Podiasdlo '' On disturbance Rejection in nonlinear control'' ,Department of Mathematics, University of Western Australia , Proceedings of the American Control Conference, june 1994.*

[32]-*G.Cybenko ;''Approximation by superposition of sigmoidal functions'',mathematics of control ,signal and system ; vol .2 ,pp,303-314,1989.*

[33]-*K .Funahashi ,'' on the approximation realization of continious mappings by neural networks'',Neural Networks ,vol.2.pp.183-192,1989.*

[34]-*A.R.Gallant and H.white ,''there exixts a neural networks tha does not make avoidable mistakes '' in IEEE second Int, conf,Neural Networks, San Diego,CA ,1988,pp,657-664.*

[35]-*K.Hornij ,M,Stinchcombe, and H.White ,''Multilayer feed forwared networks are universal approximations'', neural networks ,vol.2,pp,183-192,1989.*

Webographie

[6]-‘*’Introduction aux réseaux de neurones’’,master2005-2006/www.apstat.com/documents/ nnet .pdf.*

[7]-*Les réseaux de neurones - Neuronebiologique.htm /http://www-igm.univ-mlv.fr/ ~dr/XPOSE2002/Neurones/index.php? .*

[8]-*Les réseaux de neurones -Introduction/http://www-igm.univ-lv.fr/~dr/ XPOSE2002 /Neurones/index.php?.*

[9]-*Les réseaux de neurones - Architecturedesreseauxdeneurones.htm /http://www-igm.univ-mlv.fr/~dr/XPOSE2002/Neurones/index.php?rubrique=Liens.*

[10]-*Formation aux réseaux de neurones netra – ESPC I/http: //www.Netra l. com / formation /espci/ -2fr.html.*

[14]-*la logique _ floue/intro-http://www. tn .refer .org /hebergement / cours / logique floue /intro.html#.*

[15]-*JITEC -Journal technologique BLofti ZADEH dans le JITEC-B.htm/ http :// www . thesame-innovation.com/Jitec/Jitec.php?Id=99.*

[16]-*logique _ floue - Exemples introductifs /http :// www.tn.refer.org / hebergement /cours/logique_floue/bibli.html.*

[18]-*Logique floue et processeur « flou » /http : //www.univ-perp.fr / SEE / ENS / I UPGS I /polit/chapo.html.*

[19]-*Petit glossaire de la logique floue/http://www.inra.fr/Internet/ Departements / MIA /M/fispro/fisprodocfr/node31.html.*

[20]-*la logique _ floue /les opérateurs flous/ http://www.tn.refer.org /hebergement / cours/logique_floue/_we_hinfo/opera.htm.*

[21]-*la logique floue/structure d'une commande floue /http://www.tn.Refer.org/ hebergement/cours/logique_floue/struc2.html.*

ملخص: طريقه الشبكـات العصبيـة و المنطـق الغامـض با شكا لهما المختلفة أستعملت بنجاح في السنوات الأخيرة في المطابقة و التحكم في الأنظمة اللاخطية بقسم واسع, في هذه المفكرة نتعرض إلى حل مشكله نزع الاضطرابات من الأنظمة اللاخطيه علما أن هذه الاضطرابات معتبره أيضا كانظمه ديناميكية حرة خطية أو لا خطية, الهدف من هذه المفكرة هو تحديد نموذج المطابقة وقانون التحكم الذي يُنقص فعل الاضطرابات على البرهان النظري لوجود حل لمشكل نزع الاضطرابات ,المخرج, العديد من المراحل المختلفة ستُناقش بالتفصيل سيوُضَح.
لقد أثبتت طريقه الشبكـات العصبيـة متعددة الطبقات وطريقه المنطـق الغامـض نموذج(تكاجي-سجونو) فعاليتهما في نزع الاضطرابات على الأنظمة اللاخطية التي تمت دراستها في هذه المفكرة.

***Résumé**: Les réseaux neuronaux et la logique floue avec ces architectures différentes ont été utilisés avec succès ces dernières années pour l'identification et la commande de systèmes non linéaires d'une classe large. Dans ce mémoire, nous considérons le problème de rejection des perturbations, une grande classe de perturbations, Lequel peut être modelé comme les sortie des systèmes dynamiques libre Linéaires ou non linéaires. L'objectif est déterminer le modèle de l'identification et la loi du commande qui minimise l'effet du la perturbation à la sortie. Plusieurs étapes de complexité croissante sont discutées en détail, la justification théorique été fournie pour l'existence de solutions au problème de rejection complète de la perturbation dans les cas spéciaux. Des systèmes dans une forme de simulation sont étudié utilisant des réseaux neuronaux multi layer et la logique floue le modèle de (Takagi-Sugeno) partout dans ce mémoire pour démontrer l'efficacité des méthodes suggérées.*

***Mots clé:** disturbance rejection, neural network, fuzzy logic, non-linear system, identification of non linear system.*

***Abstract**: Neural networks and the fuzzy logic with different architectures have been successfully used in recent years for the identification and control of a wide class of nonlinear systems. In this thesis, we consider the problem of disturbance rejection, a large class of disturbances, which can be modelled as the outputs of unforced linear or nonlinear dynamic systems. The objective is to determine the identification model and the control law to minimize the effect of the disturbance at the output. Several stages of increasing complexity of the problem are discussed in detail, theoretical justification is provided for the existence of solutions to the problem of complete rejection of the disturbance in special cases. Examples in the form of simulation studies using both multilayer neural networks, fuzzy logic model (Takagi-Sugeno) are presented throughout this thesis to demonstrate the effectiveness of the methods suggested.*

.