REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

D E R de Génie Electrique et Informatique
SPECIALITE : AUTOMATIQUE

THESE DE MAGISTER

SUR L'UTILISATION DES RESEAUX DE NEURONES ARTIFICIELS ET DES SYSTEMES FLOUS POUR LA LINEARISATION ET LA COMMANDE DE PROCESSUS CHIMIQUES NON LINEAIRES

Présentée par : **Mr. M'hamed Mouley HENICHE.**
Ingénieur d'Etat en Automatique (ENP).

Membres du jury :

Mr. Boucherit	*Président du jury*
Mr. Attari	*Directeur de thèse*
Mr. Boudjema	*Directeur de thèse*
Mr. Boukhetala	*Examinateur*
Mr. Chekireb	*Examinateur*
Mr. Farah	*Examinateur*

JUILLET 1997

REMERCIEMENTS

Je tiens à exprimer ma vive reconnaissance à messieurs M. Attari et F. Boudjema qui ont dirigé ce travail, pour la confiance qu'ils m'ont toujours accordée et pour les conseils qu'ils ont su me prodiguer en permanence. Leurs directives m'ont été bien précieuses.

Je ne saurai comment remercier monsieur M. Moudjahed dont j'ai pu, au cours de deux années passées à l'institut d'Electrotechnique du centre universitaire de Tiaret, apprécier non seulement sa compétence scientifique mais aussi et surtout ses qualités humaines. Son aide m'a été d'un apport appréciable.

Que mes amis, messieurs S. Lellou, M. Larbi et Y. Mihoub trouvent ici toute ma gratitude pour leur compréhension, leur aide, leur ambiance bien sympathique et leur soutien moral qui n'ont cessé de m'apporter tout au long de l'élaboration de ce travail.

Les professeurs qui ont bien voulu mobiliser leur temps et leur compétence pour juger ce travail, je les prie de recevoir ici l'expression de mes sincères remerciements.

SOMMAIRE

INTRODUCTION GENERALE

" Il coûte si peu aux grands à ne donner que des paroles et leur condition les dispense si fort de tenir les belles promesses qu'il vous en faites, que c'est modestie à eux de ne promettre pas encore plus largement."

J. De La Bruyère (1645-1696)

Les réseaux de neurones artificiels (RNA) et les systèmes flous (FLS) sont des outils qui offrent un ensemble de techniques efficaces développées ces dernières décennies, attirant ainsi l'attention des chercheurs dans divers domaines scientifiques. De ce fait, ils ne cessent de réaliser des performances remarquables. Ceci étant, il s'avère nécessaire de tester ces techniques pour la résolution de certains problèmes du monde réel. Les processus chimiques constituent un bon terrain de test vu les problèmes qui posent dans leur commande et même dans leurs systèmes de mesure.

En effet, les processus chimiques sont souvent fortement nonlinéaires, difficiles à commander et pas faciles à approximer par des modèles précis [72]. L'application de ces techniques à ce type de systèmes est bénéfique pour la recherche menée sur les RNA et les FLS, et aussi pour la commande de ces mêmes systèmes. La commande de certains systèmes chimiques tels que les réacteurs et les colonnes de distillation a été largement étudiée [71][75], ils ont fourni les principales applications à la commande adaptative. Des études ont été menées sur la convergence la stabilité et la robustesse de tels systèmes. Cependant, dans l'absence de ces propriétés, la commande traditionnelle a perdu de son efficacité. Encore, plus le système est fortement nonlinéaire, plus la qualité de la commande est discutable. Pour cela, l'utilisation des réseaux de neurones adaptatifs ainsi que les techniques de la logique floue peut stimuler la commande adaptative en lui suggérant de nouvelles approches, de nouveaux algorithmes et de nouvelles architectures. Notre travail est surtout orienté dans cette direction.

Dans le domaine de l'instrumentation, un bon nombre de travaux de recherche ont été menés, on citera la contribution de L.F. Pau dans la compréhension des signaux issus des instruments de mesure à l'aide des réseaux neuronaux [76], celle de R. Naidu pour la détection des erreurs dues aux capteurs lors de la commande des systèmes [77] et celle apportée par notre équipe pour la linéarisation des caractéristiques statiques de capteurs nonlinéaires à l'aide des RNA [13][58][59][60]. Dans ce contexte, des travaux ont été menés sur la linéarisation de capteurs sans avoir recours aux RNA ni aux FLS [57]. Notre travail consiste aussi à tester ces nouvelles techniques pour la linéarisation des caractéristiques statiques de capteurs chimiques tels que les capteurs ioniques, conçus pour mesurer la concentration d'un ion spécifique, mais qui présentent un problème quant à leur sensibilité à d'autres ions interférents présents dans la solution à analyser.

Ce manuscrit est donc organisé de la manière suivante :

- Le premier chapitre est consacré à la présentation des réseaux de neurones artificiels, certaines définitions de base et quelques notions sur la classification des RNA. Il comportera une présentation détaillée de deux algorithmes très utilisés pour l'apprentissage des réseaux neuronaux, à savoir l'algorithme de rétropropagation (BackPropagation) et la

méthode d'optimisation aléatoire (Random Optimisation Method) qu'on utilisera ultérieurement.

- Le deuxième chapitre nous présente un tour guidé sur les différentes méthodes de la commande neuronale, auxquelles les recherches ont abouti jusqu'à présent. Quatre approches principales seront présentées : la commande neuronale directe, la commande inverse, la commande adaptative neuronale et les critiques adaptatives.

- Le troisième chapitre nous fait une introduction assez détaillée aux les systèmes flous avec des définitions de base concernant les concepts des ensembles flous et de la logique floue. Il nous présentera les techniques du raisonnement flou permettant la synthèse des systèmes flous. Cette partie de notre travail comporte aussi une présentation des différentes techniques développées pour une commande floue des systèmes.

- Le quatrième chapitre concerne notre première application, à savoir la linéarisation de la caractéristique nonlinéaire d'une électrode sélective d'ion sodium à base de NASICON, à l'aide des RNA et des FLS. Nous mettrons en évidence la qualité de la linéarisation dans différents cas ainsi qu'une comparaison de ces deux techniques avec la linéarisation numérique présentée dans ce même chapitre.

- Le cinquième et dernier chapitre concerne notre seconde application, à savoir la commande neuronale puis floue d'un processus chimique qu'est un bioréacteur. Ce chapitre présentera notre étude menée pour la synthèse du neuro-contrôleur et du contrôleur flou, ainsi que les résultats de simulation qui donnent une idée assez claire sur la qualité de réglage obtenue par ces deux techniques.

- Enfin, nous terminerons notre travail par une conclusion générale inspirée de notre analyse de l'ensemble des résultats obtenus.

CHAPITRE 1

PRESENTATION DES RESEAUX DE NEURONES ARTIFICIELS

" J'admire la beauté de cette simplicité logique en laquelle je crois, faite d'ordre et d'harmonie que nous ne pouvons appréhender qu'avec humilité et de façon seulement imparfaite."

A. Einstein (1879-1955)

1.1 INTRODUCTION

Reconnaître un visage, identifier un manuscrit, percevoir une voix, s'adapter à des situations nouvelles mais de plus en plus complexes, s'auto-organiser..., sont quelques tâches qui fascinent les ingénieurs programmeurs, sans pour autant être assurées par le plus puissant des calculateurs.

L'incapacité de la nouvelle génération de micro-processeurs de traiter ces problèmes du monde réel, est due au fait que ces machines sont complètement inadéquates pour ce type de tâches. En effet, les recherches scientifiques menées ces dernières décennies dans le domaine neurobiologique, ont laissé les chercheurs penser à réaliser des systèmes capables d'imiter les fonctions extraordinaires du cerveau humain.

Les premiers travaux furent lancés en 1943 par McCulloch et Pitts, par la mise au point d'un ensemble de *neurones formels* capables de réaliser certaines fonctions logiques. Les recherches menées par Rosenblatt sur le *perceptron* (1958) et inspirées du système visuel humain, n'allèrent pas loin vu les limites théoriques qu'elles ont rencontré. Le domaine des réseaux de neurones était donc écarté et les chercheurs se sont penchés vers *l'intelligence artificielle*, un domaine qui était plus prometteur.

Le formalisme actuel des réseaux de neurones mené par Grossberg (1976), Kohonen (1979), Hopfield (1982), pour ne citer que ceux-là, semble donner de meilleurs résultats. De plus, la découverte d'algorithmes d'entraînement appropriés leur procure des perspectives d'application dans divers domaines: commande des processus, instrumentation, traitement du signal, prédiction financière...

Dans le présent chapitre, on présentera quelques définitions concernant les réseaux neuronaux, leur classifications, ainsi que les algorithmes d'apprentissage appropriés aux type de réseaux qu'on utilisera par la suite.

1.2 NEURONE BIOLOGIQUE

Le cerveau humain est constitué, à l'échelle microscopique, de cellules appelées neurones biologiques. La structure du neurone biologique comprend quatre parties principales, résumées dans la figure (1.1). Le fonctionnement de chaque partie s'effectue comme suit [1] :

- *Le corps cellulaire* :
  - Réception des influx nerveux.
  - Traitements des influx nerveux.
  - Activation du neurone.
- *Axone* :
  - Transmission du potentiel d'action.
- *Synapses :*
  - Transmission du signal provenant de l'axone par transformation électrique-chimique grâce aux neuro-transmetteurs.
  - Réception du signal par les dendrites par transformation

électrique-chimique grâce aux neuro-récepteurs.

- *Dendrites* :

  - Acheminement des signaux reçus des autres neurones vers le corps cellulaire.

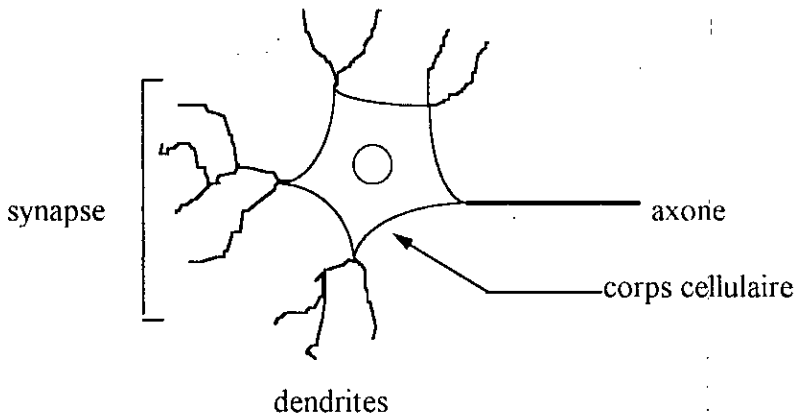


Fig. 1.1 neurone biologique.

1.3 NEURONE FORMEL

C'est le résultat de l'analogie directe qu'ont faite Mc Culloch et Pitts, en 1943, avec le neurone biologique [2]. Ce fut la première modélisation mathématique d'un neurone. Il s'agit d'un corps cellulaire qui exécute une somme pondérée des signaux d'entrée qui lui parviennent. Si cette somme dépasse un certain seuil, le neurone est activé, ou au niveau haut. Autrement, le neurone est dit désactivé, ou au niveau bas (Fig. 1.2).

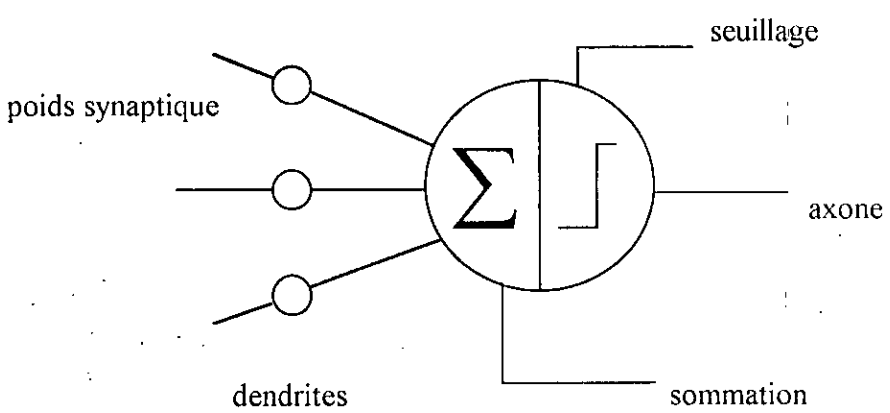


Fig. 1.2 neurone formel.

1.3.1 ELEMENT LINEAIRE ADAPTATIF (ADALINE)

L'élément linéaire adaptatif est le bloc de base dans l'architecture d'un réseau de neurones (Fig.1.3) [3]. Il est appelé ainsi car il permet une adaptation de ses poids au vu d'un certain comportement.

Cet élément reçoit à l'instant k un vecteur d'entrée X_k, à valeurs binaires ou continues, et une réponse désirée d_k. Après avoir calculé la somme pondérée S_k, on applique à celle-ci une *fonction* dite *d'activation* qui détermine l'état de la cellule, et donc la sortie du neurone.

Suivant l'application à laquelle on veut mêler le réseau de neurones, plusieurs types de fonctions d'activation peuvent être utilisées, parmi lesquelles on trouve:

- *Les fonctions binaires à seuil :* la fonction Heaviside ou la fonction signe.
- *Les fonctions linéaires à seuil :* la fonction de saturation.
- *Les fonctions sigmoïdes :* la fonction tangente hyperbolique entre autres.
- *Les fonctions probabilistiques :* la fonction Gaussienne [4].

Dans tous les cas, la fonction d'activation doit être saturable pour éviter des valeurs en sortie trop élevées qui peuvent déstabiliser le réseau. D'autre part, la fonction d'activation est à caractère *nonlinéaire*, ce qui rend les réseaux de neurones des systèmes nonlinéaires capables de simuler des fonctions complexes [5].

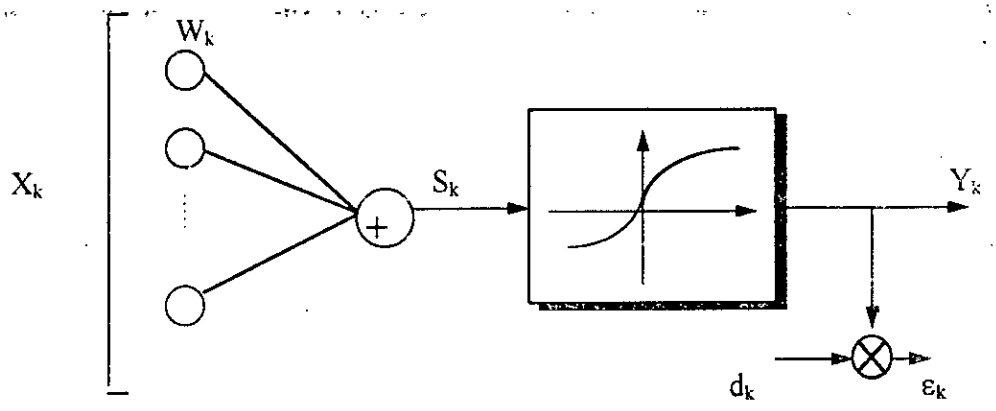


Fig. 1.3 élément linéaire adaptatif.

1.4 RESEAU DE NEURONES

L'assemblage de plusieurs éléments linéaires adaptatifs, où chaque élément est relié à d'autres par le biais de ses entrées et sorties, est appelé réseau de neurones. Un réseau neuronal n'est autre qu'un système *connexionniste* [6] complexe, formé de nombreuses interactions entres les éléments, leur permettant ainsi de travailler ensemble pour résoudre les problèmes. Ces éléments ont un comportement individuel simple, leurs influences mutuelles décident du comportement global du réseau.

Différents modèles de réseaux de neurones ont été établis, dont les plus importants sont résumés dans le tableau (1.1).

Année	Modèle	Résumé
1958	Le perceptron de Rosenblatt	Premier modèle des réseaux neuronaux, utilisé pour la reconnaissance des formes. Il comprenait: une rétine (couche d'entrée), une couche de cellules d'association (couche cachée), et une couche de cellules de décision (couche de sortie).
1960	Le Madaline de Widrow	Assemblage de plusieurs éléments adaptatifs sous forme de couches. Addition d'un terme bias à la somme pondérée et d'une fonction binaire à la sortie du neurone.
1982	Le réseau de Hopfield	Modélisation d'une mémoire associative. Le réseau est entièrement connecté, les neurones n'étant pas reliés à eux mêmes.
1983	La machine de Bolzmann	Amélioration de l'algorithme d'apprentissage d'un réseau de Hopfield.
1984	Le modèle de Kohonen	Le réseau se distingue par une auto-adaptation, une projection lui permettant de compresser les données et une bonne résistance aux bruits.

Tab. 1.1 principaux modèles des réseaux de neurones.

1.5 APPRENTISSAGE DES RESEAUX DE NEURONES

La particularité des réseaux de neurones est leur capacité à apprendre. Ils présentent une caractéristique très intéressante qu'est l'auto-organisation. Il peuvent en effet s'adapter à des situations nouvelles, en apprenant certaines caractéristiques de signaux d'entrée, et de là, synthétiser une mémoire associative leur permettant de présenter les sorties appropriées.

On entend par apprentissage des réseaux, l'opération qui consiste à modifier les poids des connexions du réseau lorsqu'on lui présente un vecteur d'entrée, la modification des poids se poursuit jusqu'à ce que ces derniers ne varient que d'une façon infime. Ce travail d'apprentissage en fait, est réalisé à l'aide d'algorithmes appropriés qu'on présentera par la suite.

Durant l'apprentissage, le réseau a la possibilité de trouver la relation qui existe entre les différentes entrées qu'on lui fait apprendre. De ce fait, il deviendra donc capable de généraliser sur des exemples qu'il n'a pas appris. Cette notion de généralisation est très importante, car les données réelles du problèmes sont très souvent bruitées [2].

1.5.1 PRINCIPE DE PERTURBATION MINIMALE

Tous les algorithmes itératifs utilisés pour l'apprentissage des réseaux, respectent un principe très important dont voila l'énoncé:

« La minimisation de l'erreur de sortie se fait de manière à ce que la perturbation créée sur les exemples déjà appris soit minimale » [3].
Ce principe intuitif a été à la base de tous les algorithmes d'apprentissage.

1.5.2 SUR-APPRENTISSAGE [7]

Les travaux expérimentaux menés sur les réseaux de neurones ont montré que durant l'apprentissage, ces derniers peuvent mémoriser les exemples, sans pour autant trouver le lien qui existe entre eux. En effet, si on dépasse un certain nombre d'exemples d'entraînement, le réseau peut perdre la capacité de généralisation, et donne ainsi des sorties non satisfaisantes en gardant même des réponses correctes pour les entrées déjà apprises.

Ce qu'on peut faire dans ce cas, c'est d'effectuer un test de généralisation chaque fois que le réseau aura appris un certain nombre d'exemples. Si le test commence à être négatif, on arrête l'introduction de nouveaux exemples, pour éviter ce qu'on appelle le *sur-apprentissage*. Cette explication est résumé dans un schéma graphique de la figure (1.4).

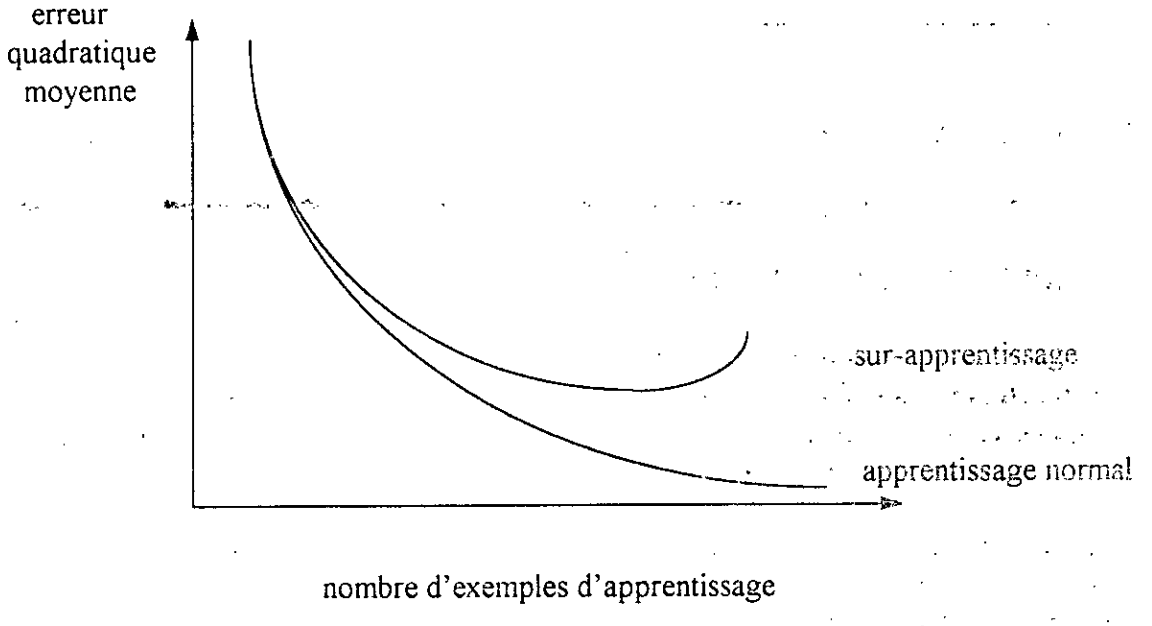


Fig. 1.4 phénomène de sur-apprentissage.

1.6 CLASSIFICATION DES RESEAUX DE NEURONES

1.6.1 CLASSIFICATION SELON LA STRUCTURE DU RESEAU

Les réseaux neuronaux peuvent être classés suivant leur architecture, donnant naissance à deux types de réseaux :

A. Réseaux multicouches

Le réseau est constitué de plusieurs couches, comportant chacune un nombre donné d'éléments adaptatifs. Les éléments d'une seule couche ne sont pas reliés entre eux. Le

neurone de chaque couche est relié à tous les neurones de la couche précédente par le biais de ses entrées et à tous les neurones de la couche suivante par le biais de sa sortie (Fig. 1.5).
Le réseau possède une couche d'entrée liée directement au vecteur d'entrée X, et une couche de sortie donnant les signaux accessibles du réseau. Les couches dont les sorties des éléments ne sont pas accessibles sont dites couches cachées. On les introduit dans le réseau pour augmenter sa capacité à simuler des fonctions de plus en plus complexes.
Dans ce type de réseaux, la propagation des signaux se fait vers l'avant, de la couche d'entrée vers la couche de sortie. On les nomme alors réseaux à propagation avant ou *feedforward networks*.

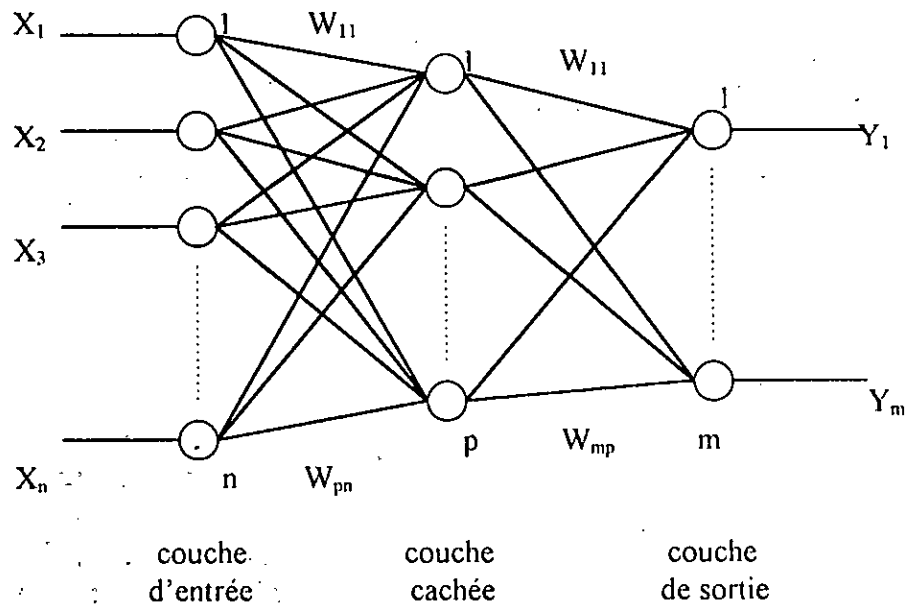


Fig. 1.5 réseau multicouches.

B. Réseaux entièrement connectés

Dans cette structure de réseaux, chaque neurone communique avec tous les autres, et même éventuellement avec lui même. La manière avec laquelle le réseau s'organise fut définie par D.O. Hebb, et est basée sur des déductions biologiques; elle stipule que l'activation simultanée de deux neurones tend à renforcer leur connexion [8].
La véritable approche de ce type de réseaux était établie par Hopfield en 1982, et selon lui le réseau doit chercher un état stable parmi plusieurs présentés lors de l'apprentissage. Bien qu'il semble parfois difficile à entraîner, ce réseau est le plus proche de la réalité.

1.6.2 CLASSIFICATION SELON LE TYPE D'APPRENTISSAGE

Les réseaux de neurones peuvent être classés suivant le type d'apprentissage qu'ils utilisent:

A. Apprentissage supervisé

C'est un apprentissage surveillé. Les poids synaptiques se modifient graduellement de manière à ce que le réseau délivre une réponse, la plus proche possible de celle désirée. Les exemples d'apprentissage seront alors formés de couples (entrées, sorties désirées). Parmi les algorithmes utilisés pour ce type d'apprentissage, on citera l'algorithme de rétropropagation, la méthode d'optimisation aléatoire et l'algorithme MADALINE III [3] qui est mathématiquement équivalent à la rétropropagation. Ce type d'apprentissage est surtout utilisé dans le domaine de la reconnaissance des formes.

B. Apprentissage non supervisé

C'est le type d'apprentissage où les exemples d'entraînement sont dépourvus de réponses désirées. Le réseau doit effectuer une auto-adaptation (Self-Organizing Neural Network). Il s'agit de trouver les relations qui existent entre les exemples présentés. Parmi les réseaux qui obéissent à ce type d'apprentissage, on citera les réseaux de Hopfield et la machine de Bolzmann. Ce type d'apprentissage est très utilisé dans le domaine de traitement des signaux.

1.7 MEMOIRE ASSOCIATIVE

Un réseau neuronal n'est autre qu'une mémoire associative. Son travail consiste en premier lieu à stocker ou à mémoriser des données sous forme d'entrées/sorties durant la phase d'apprentissage, et en second lieu de se rappeler, durant la phase de généralisation quand on présentera au réseau une entrée bruitée, d'un exemple déjà mémorisé. Si les entrées et sorties sont du même type, on parlera d'*auto-association.* Si les entrées et sorties ne sont pas du même type, on parlera d'*hétéro-association.*

Dans la phase de rappel, la notion la plus importante est celle de la décision. Le réseau doit décider si la donnée est plus proche de tel exemple appris plutôt que d'un autre. Au fait cette capacité est tout simplement déduite du fonctionnement du cerveau humain, puisque celui-ci peut, après avoir vu le visage d'une personne, l'associé automatiquement au nom de celle-ci.

1.8 ALGORITHMES D'APPRENTISSAGE

La phase d'apprentissage est une étape déterminante dans la conception du réseau de neurones. Pour cela, des algorithmes appropriés ont été élaborés et développés au fil des années pour être appliqués, chacun à un type spécifique de réseaux. Ces méthodes sont basées sur des techniques mathématiques connues appliquées dans divers domaines où ils ont prouvé leur efficacité.

Dans ce qui suit, deux principaux algorithmes d'apprentissage sont présentés :

- L'algorithme *de rétropropagation du gradient* (Backpropagation), découvert par Werbos en 1974 [12], faisant l'objet de sa thèse de doctorat, et développé par Rumelhart, Hinton et Williams en 1986 à MIT [2][3].

- La méthode d'*optimisation aléatoire*, découverte par Matyas en 1965 pour être de nouveau améliorée en 1981 par Solis et Wets à l'université de Kentucky [9][10].

D'autres algorithmes ont été conçus pour d'autres architectures de réseaux, dont on ne verra pas la suite ici. On citera par exemple: la machine de Boltzmann utilisée pour des réseaux partiellement ou entièrement connectés et la théorie de la résonance adaptative (ART) utilisée pour des réseaux ayant deux couches en interaction. A titre d'information, se référer à [2] et [7].

1.8.1 ALGORITHME DE RETROPROPAGATION (BP)

Cet algorithme est utilisé pour une topologie de réseau multicouches, l'apprentissage y est supervisé. Il est basé sur une méthode numérique dite de relaxation qui effectue une descente de gradient sur la surface d'erreur quadratique moyenne. Elle utilise donc les techniques de dérivées partielles. Ceci étant, on est amené à utiliser des fonctions d'activation dérivables appelées sigmoïdes,

exemple: $$F(x) = 1 / (1+\exp(-ax)) \qquad (1.1)$$

Le principe de la méthode est résumé dans la figure (1.6). Il s'agit de présenter au réseau un vecteur d'entrée et un vecteur de sortie désirée. Le réseau s'engagera à ce moment à calculer sa propre sortie par une propagation avant (forward) des calculs. Une fois la sortie calculée, celle-ci présente forcement une erreur par rapport à celle désirée. Cette erreur est utilisée pour l'adaptation des pondérations en faisant une propagation arrière (backward), de la couche de sortie vers la couche d'entrée, dans le but de minimiser cette erreur.

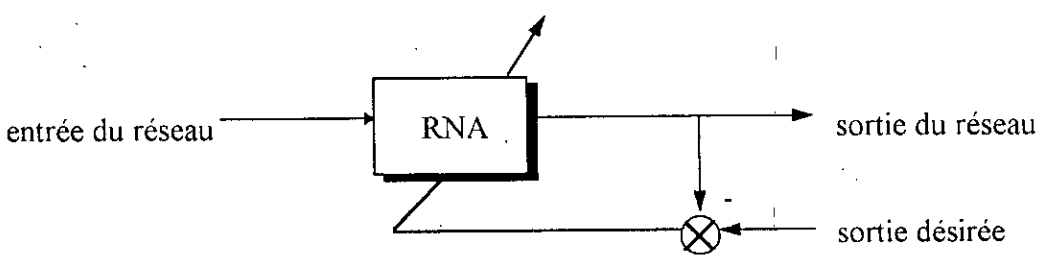


Fig. 1.6 Principe de la rétropropagation.

Résumé de l'algorithme BP

Soit un réseau à n neurones en entrée, m en sortie, p le nombre de couches cachées, et n_k le nombre de neurones de la couche cachée numéro k. μ étant le pas du gradient.

1. Initialiser tous les poids W_{ij} aléatoirement.

2. Présenter au réseau une forme $(x_1,.....,x_n)$ en entrée, et sa sortie désirée $(y_1,.....,y_m)$.

3. Calculer les sorties des neurones des couches cachées et de la couche de sortie par:

première couche cachée :

$$O_j^1(k) = f\left[\sum_{i=1}^{n} W_{ji}(k)x_i(k)\right] \tag{1.2}$$

deuxième couche cachée :

$$O_j^2(k) = f\left[\sum_{i=1}^{n1} W_{ji}(k)O_i^1(k)\right] \tag{1.3}$$

etc...

couche de sortie:

$$S_j(k) = f\left[\sum_{i=1}^{np} W_{ij}(k)O_i^p(k)\right] \tag{1.4}$$

4. Modifier les poids des connexions récursivement par :

$$W_{ji}(k+1) = W_{ji}(k) - \mu\, d_j(k)\, O_i(k) \tag{1.5}$$

avec:

$$d_j(k) = \left(S_i(k) - y_i(k)\right) f'\left[\sum_{i=1}^{np} W_{ji}(k)O_i^p(k)\right] \tag{1.6}$$

pour la couche de sortie,

$$d_j(k) = \sum_{i=1}^{n_{k+1}} d_i(k)W_{ji}(k) f'\left[\sum_{i=1}^{nk} W_{ji}(k)O_i^{nk}(k)\right] \tag{1.7}$$

pour la couche cachée numéro k.

5. Refaire les étapes 2 à 4 jusqu'à stabilisation du réseau.

Remarques

- L'expérience a montré qu'il est, généralement, préférable d'initialiser les poids à des valeurs comprises entre 0 et 1 pour éviter de déstabiliser le réseau dés le départ [2][14].

- Le pas du gradient est choisi entre 0 et 1, et on le diminuera au fur et à mesure pour atteindre une valeur fixe.

- Le choix des exemples d'apprentissage est très important, un choix judicieux de ces derniers pourra faciliter la généralisation sur des exemples non appris. Ceci a poussé les chercheurs à faire l'étude statistique des données d'entraînement pour trier celles qui sont le plus représentatives du phénomène qu'on souhaite avoir [7].

1.8.1.1 DIFFICULTES ET LIMITES DE L'ALGORITHME

Des questions restent posées quant à l'utilisation de cet algorithme, à savoir :

- Combien faut-il utiliser de couches, et combien doit-il y avoir de neurones par couche ?

Les réponses ne sont données que par l'expérience, et il n'existe aucune règle théorique précise qui puisse fixer le nombre de neurones ou de couches dans le réseau.

- Le problème de minimisation n'est pas facile à résoudre. En effet, la surface d'erreur peut présenter des caractéristiques peu satisfaisantes, telles que des minima locaux qui empêchent la convergence vers le minimum global, ainsi que des plateaux où les pentes sont très faibles.

- Le pas du gradient peut être difficile à choisir. S'il est faible, la convergence peut être très lente. S'il est élevé, l'erreur risque d'osciller sans pour autant converger.

- De plus, l'algorithme n'a aucune preuve théorique de sa convergence [1].

1.8.1.2 APPLICATIONS DE L'ALGORITHME (BP) [2]

Malgré les difficultés qu'on vient de citer, l'algorithme de rétropropagation s'est révélé assez performant dans la résolution de plusieurs problèmes, tels que :

- la reconnaissance des formes géométriques,
- le traitement et l'analyse des signaux,
- la commande des processus,
- la classification,

- le filtrage du bruit,
- le diagnostique médical,
- la prédiction financière,

pour ne citer que ceux-la.

C'est au fait l'algorithme qui a permis aux réseaux de neurones d'émerger après s'être éclipsé pendant un certain temps.

1.8.2 METHODE D'OPTIMISATION ALEATOIRE (ROM)

Cette méthode (Random Optimisation Method) est basée sur une technique de recherche aléatoire. Elle fut utilisée avec succès dans divers problèmes d'optimisation. Elle est performante quand la fonction est complexe et qu'on veut trouver son minimum global, sachant qu'elle présente des minima locaux. La nécessité d'utiliser cette méthode survient lorsque la dimension du réseau devient grande et le temps de calcul, avec l'algorithme de rétropropagation, peut être problématique [11].

La méthode d'optimisation aléatoire converge vers le minimum global d'une fonction avec une probabilité égale à 1 dans un ensemble compact [9].

L'idée de cet algorithme, dont la démonstration est présentée dans l'annexe b, est d'entacher les poids du réseau d'une séquence de bruit blanc et de calculer la sortie du réseau avec ces nouvelles pondérations. Si l'erreur entre celle-ci et la réponse désirée est inférieure à l'erreur précédente, on garde cette séquence de poids. Sinon on garde la séquence précédente. On refait l'opération jusqu'à obtention de l'erreur voulue. Voir figure (2.7).

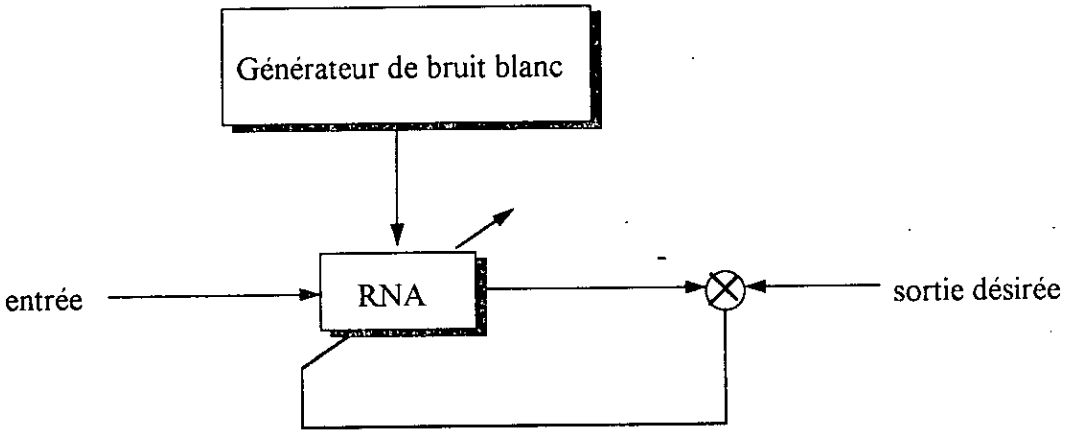


Fig. 1.7 adaptation des poids par ROM.

Résumé de l'algorithme ROM

Soit $f(x)$ la fonction à minimiser, g(k) le vecteur du bruit Gaussien et $b(k)$ la moyenne du bruit g(k) à l'instant k.

1. Choisir un point initial $x(0)$, $b(0)$ 0 pour k 0.

2. Générer un vecteur Gaussien $g(k)$.

3. (i) Si $f(x(k)+g(k)) < f(x(k))$ (1.8)

poser: $x(k+1) = x(k) + g(k)$ (1.9)

et $b(k+1) = 0.4\ g(k) + 0.2\ b(k)$ (1.10)

(ii) Si $f(x(k) + g(k)) \geq f(x(k))$ (1.11)

et $f(x(k) - g(k)) < f(x(k))$ (1.12)

poser: $x(k+1) = x(k) - g(k)$ (1.13)

et $b(k+1) = b(k) - 0.4\ g(k)$ (1.14)

Sinon, poser: $x(k+1) = x(k)$ (1.15)

et $b(k+1) = 0.5\ b(k)$ (1.16)

4. Si le minimum est satisfaisant, stopper les calculs, sinon poser $k=k+1$ et aller en 2.

Remarques

- Dans le cas de l'apprentissage du réseau neuronal, la fonction à minimiser est, bien entendu, l'erreur quadratique (la somme des carrés des erreurs commises sur chaque exemple d'apprentissage), qui est fonction des pondérations.

- Pour les besoins théoriques de l'algorithme, le vecteur poids doit appartenir à un ensemble compact [9]. Par exemple, on prendra W comme étant un vecteur dont les composantes sont inférieures à 100. Ce n'est que dans ce cas qu'on sera capable de trouver le minimum global de la fonction d'erreur.

- Durant la génération du bruit blanc (voir annexe c), le choix de la variance est capital pour améliorer la vitesse de convergence. Généralement, on la fixe à une valeur entre 0 et 1, pour des raisons purement expérimentales, et on la diminue au fur et à mesure qu'une stagnation de l'erreur est remarquée durant l'apprentissage.

1.8.3 COMBINAISON DES ALGORITHMES BP ET ROM

Des travaux de comparaison des algorithmes BP et ROM ont été effectués dans le but de déterminer le meilleur outil pour l'entraînement des réseaux multicouches [11][13]. Les résultats découlant de cette comparaison ont été avantageux à la méthode d'optimisation aléatoire du point de vue rapidité (nombre d'itérations relativement inférieur), et présentant une erreur finale inférieure à celle obtenue par l'algorithme BP.
Toutefois, ces résultats sont spécifiques à certaines applications, et la généralisation n'est pas pour bientôt. C'est pour cela qu'on continuera à utiliser les deux algorithmes pour l'entraînement des réseaux neuronaux.

Seulement, on peut mentionner une autre manière de faire concernant l'apprentissage. Elle concerne la combinaison des deux algorithmes [13]. En effet, la qualité de l'erreur peut être améliorée et de loin, si l'on procède de la manière suivante :

- Débuter l'apprentissage avec l'algorithme BP pendant un certain nombre d'itérations (généralement, jusqu'à ce que la convergence devient lente), bénéficiant ainsi d'un bon conditionnement des poids.

- Terminer l'apprentissage à l'aide de l'algorithme ROM, avec pour poids initiaux ceux obtenus à la fin de l'apprentissage par BP, bénéficiant ainsi de la convergence vers une erreur finale encore meilleure.

1.9 OPTIMISATION DES RESEAUX NEURONAUX

On dispose d'un réseau neuronal, d'un échantillon de données empiriques pour entraîner le réseau et d'un algorithme d'apprentissage approprié. Mais pour obtenir la meilleure généralisation possible, quelle doit être la taille du réseau ?

L'approximation de fonctions complexes nécessite un réseau d'une grande taille avec des unités cachées à fonctions d'activation nonlinéaires. Cependant, outre les inconvénients qu'il présente -simulation lente et trop d'exemples d'apprentissage-, un réseau sur-dimensionné risque de conduire à une mauvaise généralisation.

L'amélioration des capacités de généralisation a été essentiellement expérimentale. Aucune méthode ne permet de déterminer précisément la structure du réseau en fonction des exemples à apprendre pour fournir une généralisation optimale. Les travaux présentés par le mathématicien Vapnik [15] offrent un cadre théorique sur l'application de la minimisation structurelle aux réseaux neuronaux. Même si ces travaux ne sont pas encore validés pratiquement, ils constituent une avancée remarquable sur le plan théorique, et motivent actuellement beaucoup de recherches qui tentent d'expliquer les observations expérimentales [16].

1.10 CONCLUSIONS

Dans un réseau de neurones, le traitement des données ou des connaissances est parallèle et distribué. C'est à dire que toutes les unités du réseau travaillent simultanément, contribuant ainsi à un comportement intelligent.

Les réseaux neuronaux sont des systèmes difficiles à construire fournissant des résultats ou des performances des fois inexplicables. Par contre, ce sont des systèmes adaptatifs et leur capacité d'apprentissage leur permet de prendre en compte les nouvelles contraintes qui apparaissent, à l'inverse des calculateurs qui eux ne sont pas entraînés mais programmés. Sans oublier bien sûr, leur capacité de généralisation, une conséquence bénéfique qui résulte d'un bon apprentissage.

Aussi, ce sont des systèmes faciles à simuler par des langages souples, et on a nullement besoin de reprogrammer le fonctionnement du réseau si on ajoute des neurones supplémentaires.

Bien que la détermination de l'architecture optimale du réseau n'est pas encore bien cernée par des règles théoriques précises, l'expérience acquise durant les diverses applications nous permet d'améliorer les performances en jouant sur plusieurs paramètres, tels que le nombre de couches, le nombre de neurones par couche, le type de fonction d'activation, les coefficients d'apprentissage, etc.

CHAPITRE 2

STRUCTURES D'IDENTIFICATION ET DE COMMANDE NEURONALE

" La tache suprême du physicien est d'arriver à des lois élémentaires universelles telles que les cosmos puissent être construits à partir d'elles par pure déduction. Aucune voie logique ne conduit à ces lois, mais seule l'intuition qui repose sur une intelligence compréhensive. "

A. Einstein (1879-1955)

2.1 INTRODUCTION

Cette dernière décennie a vécu une croissance extraordinaire du nombre d'articles, de conférences et d'issues spéciales, consacrés à l'application des réseaux de neurones artificiels (RNA) dans le domaine de la commande des processus [18][31]. Ceci, bien sûr, indique le grand intérêt accordé à cette discipline.

L'utilisation des réseaux neuronaux pour la commande des systèmes est motivée par les caractéristiques et propriétés suivantes:

- Les RNA portent de grandes promesses pour résoudre les problèmes de commande nonlinéaire. Ceci provient de leur capacité à approximer des fonctions nonlinéaires arbitraires.
- Le traitement parallèle et distribué des données facilite leur implémentation et laisse espérer à obtenir de meilleures performances par rapport aux schémas de commande conventionnels.
- La possibilité d'une implémentation hardware par des circuits VLSI constitue un autre avantage pour ces derniers, puisqu'elle augmente leur vitesse de fonctionnement et permet d'accroître le nombre de réseaux à implémenter.
- Les RNA sont entraînés en utilisant des données issues du système à commander, un apprentissage on-line est de ce fait possible. Un réseau adéquatement entraîné peut généraliser sur des entrées qu'il n'a pas appris.
- Les RNA peuvent opérer simultanément sur des données quantitatives et qualitatives. Ils rassemblent donc, les avantages des systèmes traditionnels (données quantitatives), et des techniques de traitement issues de l'intelligence artificielle (données symboliques).
- Ce sont des systèmes pouvant avoir plusieurs entrées et plusieurs sorties, ils sont donc applicables aux systèmes multivariables.

Le plus important pour notre travail, c'est la capacité des réseaux de neurones à traiter des systèmes nonlinéaires. En effet, la grande diversité de ces derniers et leur complexité toujours croissante sont les raisons primaires pour lesquelles il n'existe pas de théorie de commande systématique et générale applicable pour tous les types de systèmes nonlinéaires. Un ensemble de méthodes, pour l'analyse et la synthèse de contrôleurs nonlinéaires, existent [32][33] et ont joué un rôle qu'on ne peut négliger, avec des performances satisfaisantes. Cependant, la capacité des réseaux à représenter des relations nonlinéaires, et ainsi, modéliser les systèmes dont les modèles sont inconnus , les place en pôle position pour la conception des contrôleurs nonlinéaires.

De plus, un outil capable de prendre en compte le changement de son environnement, de réduire l'incertitude, de générer et d'exécuter l'action de commande dans des situations défavorables tel un réseau de neurones, constitue un système de commande intelligent. Le changement dans l'environnement et dans les critères de performance, ainsi que la présence de perturbations non mesurables, sont quelques caractéristiques qui nécessitent une commande intelligente, et la commande neuronale en est une.

Après avoir cité les capacités dont jouissent les réseaux neuronaux, il va de soi que cet outil peut être utilisé, aussi bien en identification qu'en commande des processus. Cela permet de réaliser des prototypes neuronaux qui imiteront le comportement statique et dynamique des systèmes et faciliteront ainsi la synthèse de la commande.

Des recherches ont été menées dans le domaine de l'identification et de la commande neuronale [44]. On citera ceux de Narendra et al. [20][21][34] à l'université de Yale, ceux de Psaltis et al. à California Institute of Technology [23], ceux de Sbarbaro et al. à l'université de Glasgow [29] et ceux de Barto, Sutton et al. à MIT [31][35][36], pour ne citer que ceux là.

Ce chapitre passe en revue les structures de base d'identification et de commande neuronale, les plus développées en littérature, et dont quelques unes feront l'objet de notre application dans la suite de notre travail.

2.2 STRUCTURES D'IDENTIFICATION NEURONALE

L'aptitude des RNA à simuler des relations nonlinéaires, les met au premier rang pour qu'ils soient appliqués à l'identification des processus. Entraîner des réseaux en utilisant les entrées \ sorties du système nonlinéaire, n'est autre qu'un problème d'approximation de fonction. De la même manière qu'une fonction de transfert représente la boite noire d'un système linéaire, un réseau neuronal peut représenter la boite noire d'un système nonlinéaire.

Un certain nombre de résultats ont été publiés, montrant qu'un réseau multicouches peut bien approximer des fonctions continues [38][39].

Le problème d'identification neuronale est décrit, d'une manière générale, dans la figure (2.1). Ca consiste à ajuster les paramètres du modèle d'identification, dans le but d'optimiser une fonction performance basée sur l'erreur entre la sortie du modèle et celle du système.

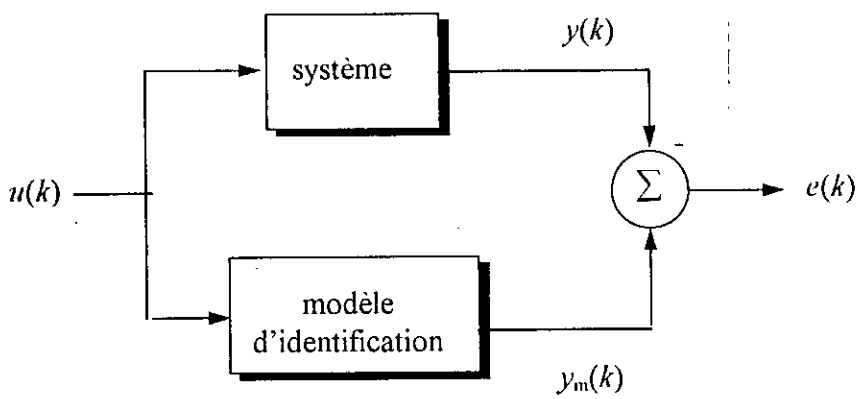


Fig. 2.1 schéma bloc général d'identification.

Une question importante concernant l'identification s'impose. C'est celle de l'identifiabilité du système [40], i.e., soit une structure de modèle particulière, le système en question peut-il être représenté par cette structure?. En l'absence d'un résultat théorique concret pour l'identification neuronale, on procède sous l'hypothèse que tout système, qu'on étudiera, appartient à la classe des systèmes que le réseau neuronal peut identifier.

Les principales structures d'identification par les RNA sont les suivantes :

2.2.1 IDENTIFICATION DU MODELE STATIQUE DU SYSTEME

Le RNA a pour rôle de reproduire la statique directe du système, soit alors :

$$y_p = f(u) \tag{2.1}$$

Il s'agit de placer le RNA en parallèle avec le système. L'erreur entre la sortie du système et celle du réseau (erreur de prédiction) est utilisée comme signal d'entraînement du réseau (fig. 2.2).

Puisque le RNA est concerné uniquement par la statique du système, il n'a donc pas besoin de poursuivre sa dynamique. Ceci étant, un apprentissage off-line est suffisant. De plus, la stabilité du système dans la plage de variation des données d'apprentissage, est exigée pour la convergence de l'erreur.

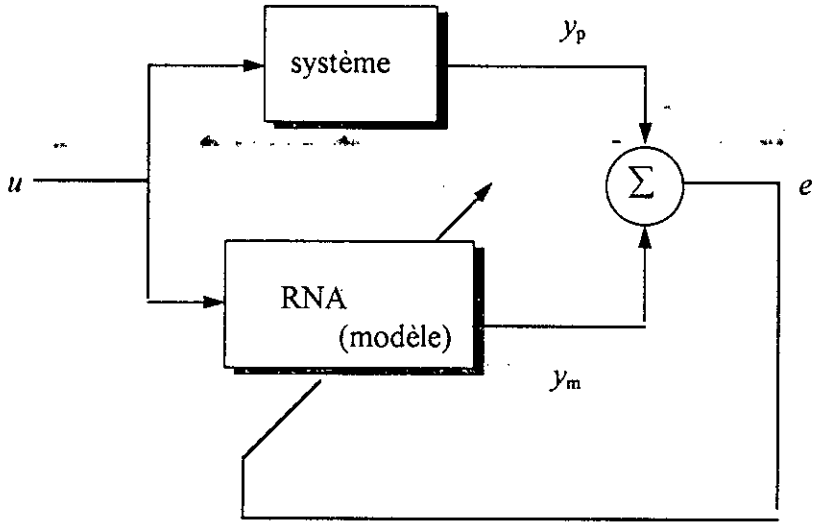


Fig. 2.2 identification du modèle statique du système.

2.2.2 IDENTIFICATION DU MODELE DYNAMIQUE DU SYSTEME

Dans le contexte de commande, l'aspect dynamique du système est important. La possibilité qui se présente est d'introduire cette dynamique dans le réseau lui même. L'idée est d'augmenter le vecteur d'entrée du réseau de signaux correspondant aux entrées et sorties des instants précédents.

Soit un système nonlinéaire d'ordre n gouverné par l'équation aux différences suivante :

$$y_p(k+1) = f[y_p(k),....,y_p(k-n+1);u(k),....,u(k-m+1)] \qquad (2.2)$$

Une approche évidente, pour la modélisation de ce système, est de choisir la structure du RNA identique à celle du système (fig. 2.3). Notant la sortie du réseau y_m, on a alors :

$$y_m(k+1) = \hat{f}[y_p(k),....,y_p(k-n+1);u(k),....,u(k-m+1)] \qquad (2.3)$$

Ici, $\hat{f}$ représente l'approximation de f par le réseau. Il faut noter que l'entrée du réseau est composée des sorties du système réel. Ce type de modèle (2.3) est appelé *série-parallèle* [20]. Une fois l'entraînement acheminé, si le réseau délivre une bonne représentation du système (i.e. $y_p \approx y_m$), il peut être utilisé indépendamment du système, et sera décrit par :

$$y_m(k+1) = \hat{f}[y_m(k),....,y_m(k-n+1);u(k),....,u(k-m+1)] \qquad (2.4)$$

Ce type de modèle (2.4) est dit *parallèle* [20].

De même que dans l'identification de la statique du système, celui-ci doit avoir des entrées \ sorties bornées (stables), dans le but d'assurer la convergence de l'erreur. L'apprentissage peut se faire aussi bien on-line qu'off-line.

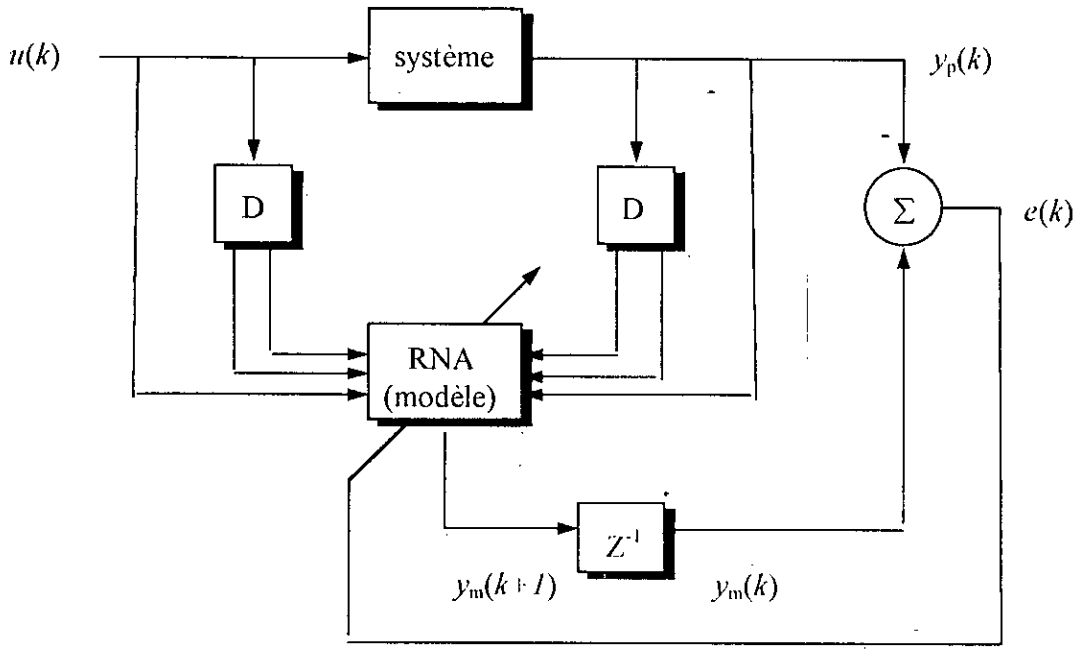


Fig. 2.3 identification du modèle dynamique nonlinéaire du système.

2.2.3 IDENTIFICATION DU MODELE D'ETAT

Considérons le système dynamique discret nonlinéaire d'ordre n, régit par l'équation d'état suivante :

$$x(k+1) = f[x(k), u(k)] \tag{2.5}$$

où $x(k) \in \chi \subset R^n$ and $u(k) \in R^m$.

Si f est inconnue, un réseau de neurones peut être conçu pour l'approximer, ayant pour exemples d'apprentissage, les états observés du système ainsi que l'action de commande (fig. 2.4). L'entraînement peut s'effectuer on-line.

Pour que l'identification soit possible, quelques informations doivent être connues à priori, telles que l'ordre du système et la nature de la fonction f. Les états du système, quant à eux, doivent être mesurables. De plus, f est assumée être bornée pour tout x et u appartenant à la classe des exemples d'apprentissage, pour la raison qu'on a cité auparavant.

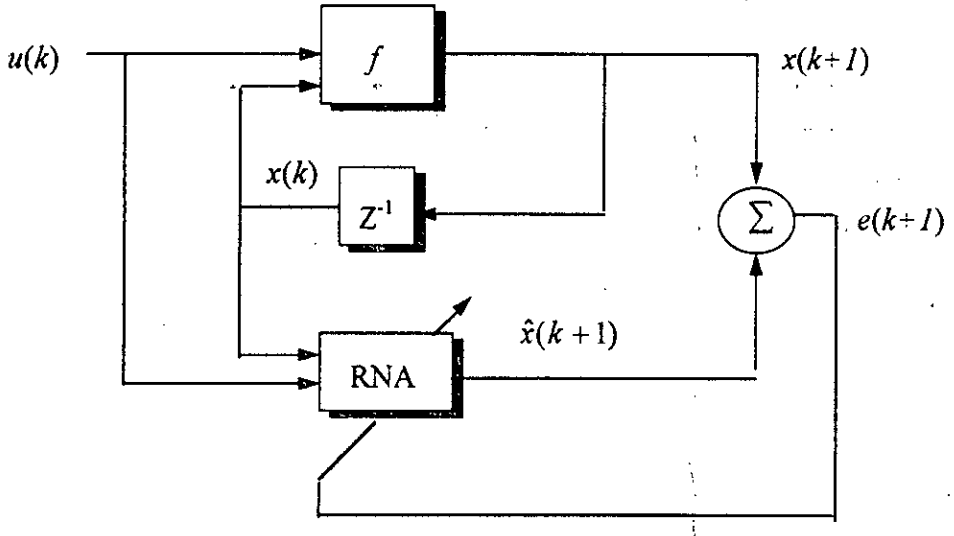


Fig. 2.4 identification du modèle d'état du système.

2.2.4 IDENTIFICATION DES PARAMETRES DU MODELE D'ETAT LINEAIRE

Une autre propriété intéressante, intrinsèque aux réseaux de neurones, est utilisée pour l'identification des paramètres du modèle d'état linéaire [25]. Elle diffère des autres méthodes présentées, du fait que c'est une identification paramétrique, alors que les autres sont des méthodes d'identification non-paramétriques, qui ont pour rôle d'imiter le comportement du système.

Cette méthode ce présente comme suit :

Soit le système linéaire discret d'ordre n décrit par l'équation d'état suivante :

$$x(k+1) = A\,x(k) + B\,u(k) \tag{2.6}$$

x : vecteur d'état du système.
u : vecteur des actions de commande.

L'objectif est d'identifier les matrices A et B qui représentent les paramètres du système, à l'aide d'un réseau neuronal adéquat. Prenons le cas d'un système SISO (le raisonnement est le même pour les systèmes multivariables). Le modèle (2.6) devient :

$$x(k+1) = \begin{bmatrix} a_{11} & a_{12} & & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ & & & \\ a_{n1} & a_{n2} & & a_{nn} \end{bmatrix} x(k) + \begin{bmatrix} b_1 \\ b_2 \\ \\ b_n \end{bmatrix} u(k) \tag{2.7}$$

Augmentons le vecteur x de l'ordre n à l'ordre n+1, en lui ajoutant le scalaire u. Ainsi, on obtient :

$$x(k+1) = \begin{bmatrix} a_{11} & a_{12} & & a_{1n} & b_1 \\ a_{21} & a_{22} & & a_{2n} & b_2 \\ & & & & \\ a_{n1} & a_{n2} & & a_{nn} & b_n \end{bmatrix} \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} = A' \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \tag{2.8}$$

On choisit un réseau ayant $[x(k),u(k)]$ pour entrées, et qui devra estimer $x(k+1)$ (fig. 2.5). Si ce réseau possède m couches cachées, alors sa sortie sera exprimée par :

$$\hat{x}(k+1) = \Gamma(W_{m+1}\,\Gamma(W_m \ldots\ldots \Gamma(W_1\,(x(k)u(k))^T)\ldots\ldots)) \tag{2.9}$$

avec W_i *(i=1,m+1)* représentent les matrices poids de la couche d'entrée (1) à la couche de sortie (m+1), et Γ représente le vecteur des fonctions d'activation appliquées aux sommes pondérées des neurones de chaque couche.

Jusqu'ici, on a fait juste une identification du modèle d'état du système. Seulement, si on choisit des fonctions d'activation identité, ce qui donne naissance à un *réseau de neurones linéaire*, la relation (2.9) devient :

$$\hat{x}(k+1) = W_{m+1}\,W_m \ldots\ldots W_1\,(x(k)u(k))^T \tag{2.10}$$

Une fois l'apprentissage terminé, on aura $\hat{x}(k+1) \approx x(k+1)$, et de ce fait :

$$A' = W_{m+1}\,W_m \ldots\ldots\, W_1 \tag{2.11}$$

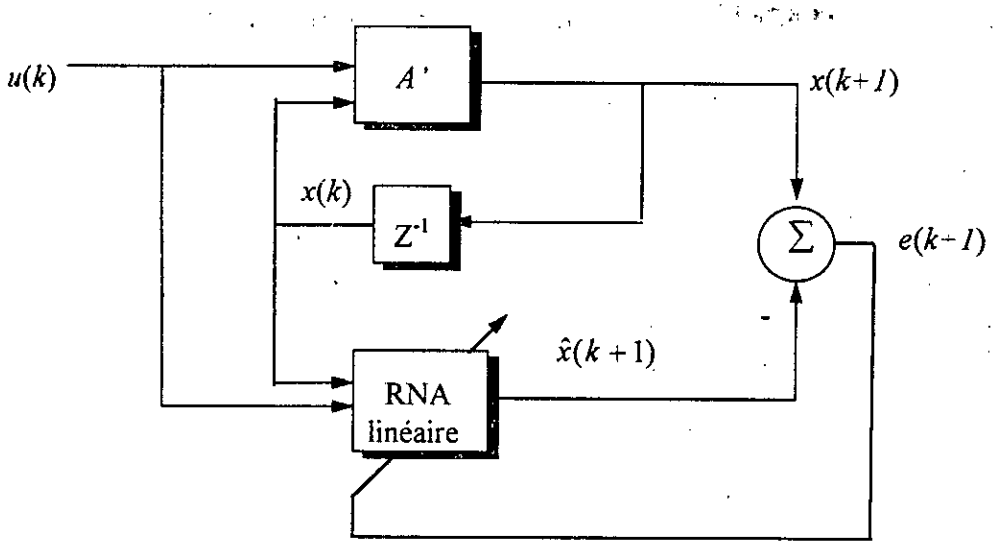


Fig. 2.5 identification paramétrique du modèle d'état linéaire.

Remarques

- Dans toutes les méthodes d'identification qu'on vient de voir, un choix concernant les conditions initiales des entrées \ sorties du système s'impose. Si le choix est le notre, la tâche est relativement facile, car il suffit d'initialiser les grandeurs d'apprentissage à des valeurs distribuées suivant la loi uniforme [20], donnant ainsi la chance à toutes les valeurs possibles d'apparaître durant l'entraînement du réseau. Dans certains problèmes pratiques, ceci n'est pas le cas, et une bonne identification du système nécessite un choix judicieux des commandes $u(k)$ correspondant à des observations spécifiques des états $x(k+1)$, favorisant ainsi des échantillons de commande par rapport à d'autres. Cette distribution des états initiaux est non-uniforme, rendant la tâche d'apprentissage plus difficile, sinon impossible [22].

- D'autre part, la qualité de l'erreur obtenue à la fin de l'apprentissage est d'une grande importance. En effet, si on utilise le modèle neuronal obtenu après identification pour la synthèse de la commande, on doit prendre en compte la valeur finale de l'erreur de prédiction tolérée pour l'obtention d'une bonne qualité de la commande. Cette question a suscité beaucoup d'intérêt et fait l'objet de diverses recherches [22].

2.3 STRUCTURES DE COMMANDE NEURONALE

Les modèles des systèmes dynamiques ainsi que leurs inverses, sont d'une grande utilité pour la commande. Dans la littérature, un bon nombre de structures de commande neuronale existent et sont bien établies et profondément analysées. Nous nous intéressons dans notre cas aux structures ayant une relation avec les modélisations directe et inverse du système. On assume que de tels modèles sont disponibles sous forme de réseaux neuronaux.

Beaucoup de structures de commande neuronale ont été proposées et utilisées, et c'est au delà de l'objectif de notre travail de fournir une vue d'ensemble de toutes les architectures de commande qui existent. Pour cette raison, on insiste surtout, sur les structures dont les propriétés sont, du point de vue de la théorie de la commande, les mieux analysées. Ces structures sont classées en quelques méthodes principales (fig. 2.6) :

- La commande supervisée, dont le but est d'imiter le comportement d'un opérateur humain à l'aide d'un RNA, fut utilisée pour la commande d'un système à balance (broom balancer) dans les années soixante par Widrow [41].

- La commande inverse, ayant pour rôle de reproduire la dynamique inverse du système, étudiée par Psaltis et al., est très utilisée dans la planification de la trajectoire et la commande de position et de force pour les bras de Robots manipulateurs [18].

- La commande adaptative neuronale, établie par Narendra et al., se basant sur la commande adaptative à modèle de référence (MRAC), est utilisée surtout en robotique. Le RNA a pour rôle d'identifier le processus et de prédire les sorties futures du système.

- La critique adaptative, qui n'est autre que la méthode d'apprentissage par renfort (reinforcement learning), dont le but est de maximiser une fonction performance ou de minimiser une fonction coût. Elle fut développée par Barto et al. [42], et sert à résoudre certains problèmes de commande difficiles, tels que le pendule inversé.

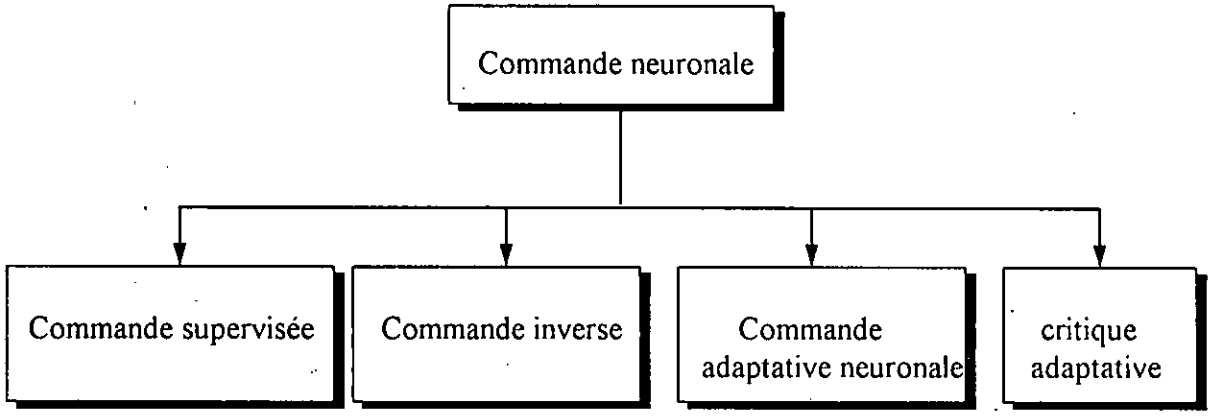


Fig. 2.6 classification des méthodes de commande neuronale.

2.3.1 COMMANDE SUPERVISEE

Il existe des situations où l'être humain fournit lui même les actions de commande pour une tâche particulière, et où il est difficile de concevoir un contrôleur en utilisant les techniques de commande standards. Dans certains cas, il est désirable de synthétiser un contrôleur qui imite les actions de l'opérateur humain. Ce type de commande est dit supervisé.

Le RNA se présente comme candidat efficace pour cette tâche. Le formalisme de la commande, ainsi que les connaissances concernant le système, sont délivrés par un expert. Dans ce cas, cependant, les entrées du réseaux correspondent aux sorties des différents capteurs, et sa sortie désirée correspond à l'action de commande appliquée par l'opérateur humain (fig. 2.7).

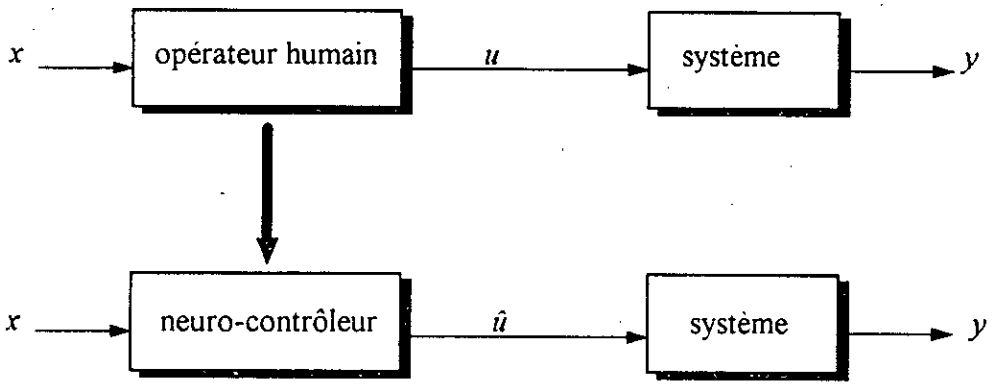


Fig. 2.7 principe de la commande supervisée.

2.3.2 COMMANDE INVERSE [23]

La commande inverse utilise le modèle inverse du système, simulé par le RNA. Celui-ci est, simplement, relié en cascade avec le système à contrôler, pour que le système global se comporte comme une fonction identité, qui lie la réponse désirée (la consigne) à la sortie du système. Le réseau est ainsi un contrôleur en boucle ouverte. Une conséquence immédiate de l'utilisation d'une commande en boucle ouverte est l'augmentation de la vitesse de réponse du système global.

Durant l'entraînement, les caractéristiques du système, initialement inconnues ou pas prises en compte, sont apprises par le réseau. De cette manière, les incertitudes de modélisation sont éliminées, et ainsi de bons résultats en commande peuvent être obtenus.

Dans ce qui suit, on présente trois méthodes d'apprentissage du neuro-contrôleur, et une version adaptée de l'algorithme de rétropropagation étendue aux problèmes dont l'erreur d'entraînement n'est pas celle mesurée à la sortie du réseau.

A. ARCHITECTURE D'APPRENTISSAGE INDIRECTE

La figure (2.8) montre un RNA utilisé en boucle ouverte, sa sortie u commande le système. Le neuro-contrôleur agit comme le modèle inverse du système. Le but de l'apprentissage est de trouver les poids du réseau qui poussent la sortie du système y à suivre celle désirée d, alors le RNA doit reproduire la commande u donnant y (i.e. $t \approx u$). Pour cela, on adapte les poids d'un autre réseau dans le but de minimiser l'erreur $\varepsilon_1 = u\text{-}t$, et ainsi minimiser l'erreur $\varepsilon = d\text{-}y$.

L'avantage de cette méthode et le fait que le réseau peut être entraîné seulement dans la région qui nous intéresse, puisqu'on utilise la réponse désirée d, et tous les autres signaux sont générés à partir du signal d. Malheureusement, cette méthode d'apprentissage n'est pas valide, parce que minimiser l'erreur ε_I ne veut pas dire, nécessairement, minimiser l'erreur ε. Cependant, cette technique reste intéressante, puisqu'elle peut être utilisée en collaboration avec l'une des procédures décrites ci-dessous, qui elles minimisent ε.

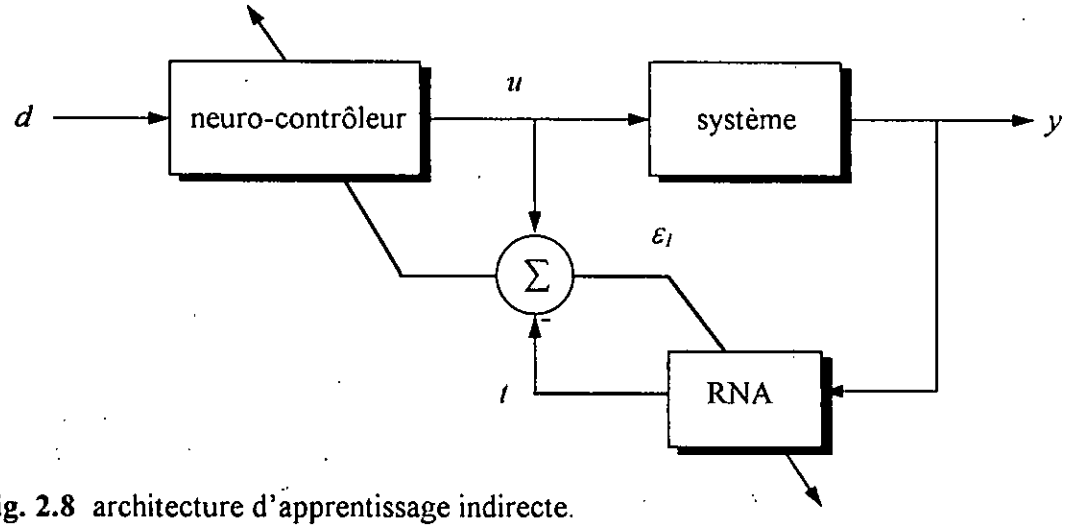


Fig. 2.8 architecture d'apprentissage indirecte.

B. ARCHITECTURE D'APPRENTISSAGE GENERALE (DIRECTE)

L'architecture montrée dans la figure (2.9) fournie une méthode d'apprentissage du neuro-contrôleur qui minimise l'erreur ε. Le réseau est entraîné pour reproduire u à partir de la sortie du système y. Une fois entraîné, le réseau doit être capable de générer la commande u à partir de la sortie désirée d, forçant la sortie du système y à suivre d.

Le succès de cette méthode dépend largement de l'aptitude du RNA à généraliser correctement sur des entrées pas nécessairement utilisées durant l'apprentissage. Dans cette architecture, on ne peut pas entraîner, sélectivement, le réseau à répondre correctement dans des régions qui nous intéressent par ce qu'on ne connaît pas, généralement, quelles sont les entrées u du système qui correspondent aux sorties désirées d. Ceci étant, l'apprentissage est effectué off-line, et une grande quantité de données d'apprentissage, non nécessaires, doivent être utilisées car les entrées, désirables et essentielles du système, sont inconnues. Une solution possible à ce problème est de combiner l'architecture d'apprentissage générale avec l'architecture spécialisée développée ci-dessous.

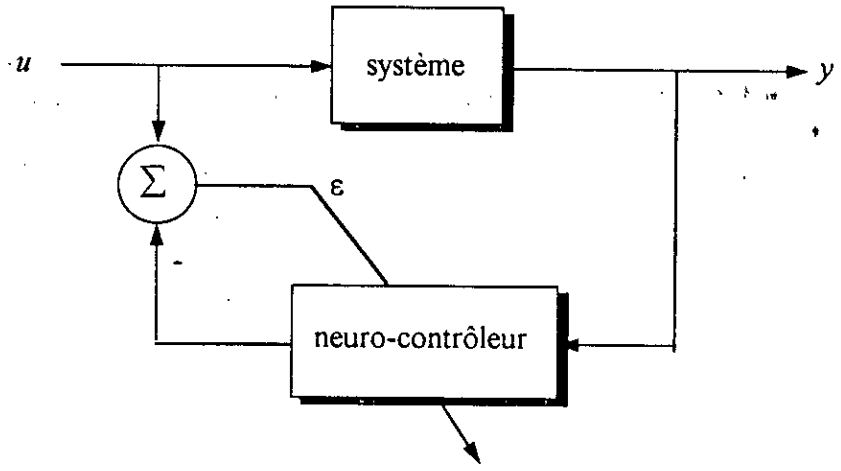


Fig. 2.9 architecture d'apprentissage générale.

C. ARCHITECTURE D'APPRENTISSAGE SPECIALISEE

Pour palier au problème sus mentionné, la figure (2.10) présente une architecture d'apprentissage du neuro-contrôleur pour qu'il opère uniquement dans les régions de spécialisation. Le réseau est entraîné à trouver la commande u qui oblige la sortie du système y à suivre la sortie désirée d. Cette architecture peut, spécifiquement, apprendre à fonctionner dans la région d'intérêt, et le réseau peut être entraîné on-line. Intuitivement, le système à commander ne doit présenter aucun problème de stabilité dans l'intervalle de variation des données d'apprentissage. De plus, si le modèle du système est inconnu, une identification neuronale de celui-ci est nécessaire pour pouvoir entraîner le neuro-contrôleur.

Comme on l'a déjà mentionné, une combinaison entre la structure générale et la structure spécialisée, est possible. Il suffit d'entamer l'apprentissage général pour approximer le comportement inverse du système, suivi de l'apprentissage spécialisé pour affiner les poids du réseau dans la région d'opération. La méthode générale a pour rôle de créer de meilleurs poids initiaux pour la méthode spécialisée. Cette combinaison est aussi bénéfique, puisqu'elle permet au réseau de s'adapter plus facilement, si jamais l'on change l'intervalle de fonctionnement. Enfin, en basculant d'une méthode vers l'autre, on peut parfois éviter de tomber dans des minima locaux.

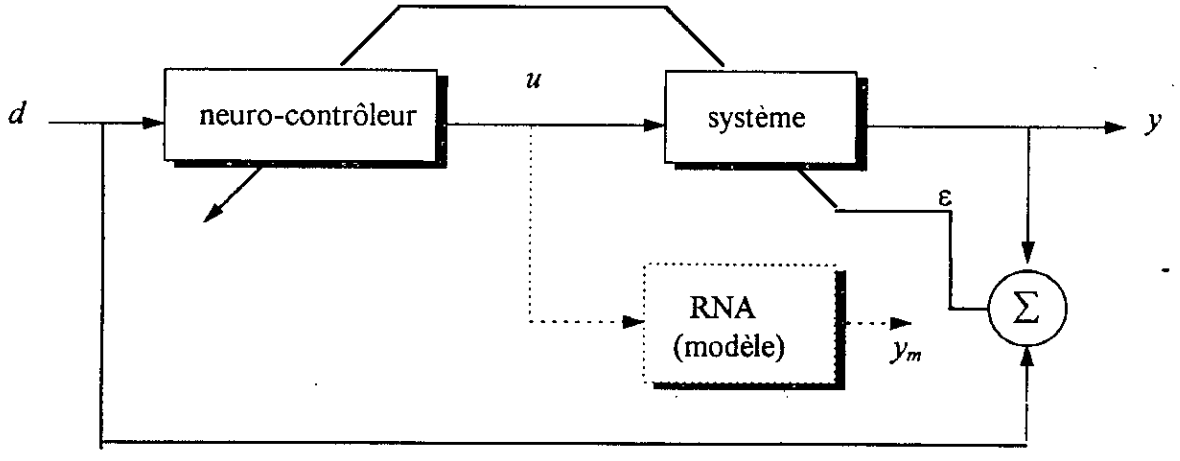


Fig. 2.10 architecture d'apprentissage spécialisée.

L'entraînement des réseaux multicouches, dans les deux méthodes précédentes, est accompli par l'algorithme de rétropropagation de base. Les poids W_{ab}, entre le neurone a de la couche m et le neurone b de la couche m-1, sont modifiés par :

$$\Delta W_{ab}^{m} = -\frac{\partial \|\varepsilon\|^2}{\partial W_{ab}^{m}} \tag{2.12}$$

$$\Delta W_{ab}^{m} = \mu \delta_a^m q_b^{m-1} \tag{2.13}$$

avec :

$$q_i^m = f_i^m(p_i^m) \tag{2.14}$$

et

$$p_i^m = \sum_j W_{ij}^m q_j^{m-1} \tag{2.15}$$

μ est le pas du gradient, q_k est la sortie du neurone k et p_k son entrée. δ_k est l'erreur rétropropagée donnée, pour la couche de sortie n, par :

$$\delta_a^n = f'^n(p_a^n)\,(u_a - t_a) \tag{2.16}$$

et pour toutes les autres couches, par :

$$\delta_a^m = f'^m(p_a^m) \sum_i \delta_i^{m+1} W_{ia}^{m+1} \tag{2.17}$$

Pour ce qui concerne l'apprentissage spécialisé, on ne peut pas appliquer la rétropropagation de l'erreur directement à cause de la position du système. L'idée est de considérer le système comme une couche supplémentaire non modifiable. Alors, l'erreur $\varepsilon = d - y$ est rétropropagée à travers le système en utilisant ses dérivées partielles au point de fonctionnement :

$$\delta_a^n = f'^n(p_a^n) \sum_i \delta_i^p \frac{\partial y_i}{\partial u_a} \tag{2.18}$$

$$\delta_a^p = d_a - y_a \tag{2.19}$$

avec y_i étant la $i^{ème}$ sortie du système correspondant à l'entrée u.

Il est clair que cette procédure nécessite la connaissance du jacobien du système. Si ceci n'est pas le cas, et qu'un RNA remplace le modèle du système, le calcul des δ_i^j s'effectue normalement avec le réseau modéle, sans pour autant modifier ses pondérations. Une fois arriver à la couche de sortie du neuro-contrôleur, l'adaptation des poids commence à avoir lieu [27].

2.3.3 COMMANDE INVERSE DYNAMIQUE [26][30]

La commande inverse qu'on vient de présenter, met à notre disposition un ensemble de techniques intéressantes pour la commande des systèmes statiques. Cependant, il n'est pas clair comment ces techniques peuvent-elles être appliquées pour la commande des systèmes dynamiques. La méthode qu'on va présenter peut être considérée comme une conception d'un contrôleur adaptatif, pour les systèmes nonlinéaires invariants, ayant une dynamique complètement inconnue.

La conception de notre neuro-contrôleur se fait en deux étapes: premièrement, une procédure pour la modélisation de la dynamique inverse du système on-line, puis une procédure pour générer les signaux de commande. Pour cela, deux hypothèses sont posées :

H1. L'ordre du système (le nombre des états) est connu.

H2. Les états sont mesurables.

A. Identification du modèle dynamique inverse du système

Considérons le système nonlinéaire discret d'ordre n, décrit par l'équation d'état suivante :

$$x(k+1) = f(x(k),u(k)) \tag{2.20}$$

$x(k)$ est l'état du système à l'instant k, et $u(k)$ est la commande au même instant.

De même, les états à l'instant k+2 sont fonctions de $u(k+1)$ et de $x(k+1)$ qui est, à son tour, fonction de $x(k)$ et de $u(k)$:

$$x(k+2) = f(x(k+1),u(k+1)) = f[f(x(k),u(k)),u(k+1)] \tag{2.21}$$

ceci implique que les états à l'instant k+2 sont déterminés par les états à l'instant k, et les actions de commande entre les instants k et k+2. En répétant ce raisonnement, on en déduit que les états à l'instant k+n sont déterminés par les états à l'instant k et les actions de commande entre k et k+n, i.e., :

$$x(k+n) = f_n(x(k),U) \tag{2.22}$$

où :

$$U = [u(k),u(k+1),....,u(k+n-1)]^T \tag{2.23}$$

On pose maintenant une troisième hypothèse sur le système :

H3. L'équation (2.22) est uniquement inversible pour U.

Alors U peut être fonction de $x(k+n)$ et de $x(k)$:

$$U = g(x(k),x(k+n)) \tag{2.24}$$

l'équation (2.24) est une relation fondamentale représentant la dynamique inverse du système. Elle implique que les actions de commande, qui conduisent les états $x(k)$ vers les états $x(k+n)$, sont contenues dans le vecteur U.

Maintenant, un RNA peut être utilisé pour approximer la fonction g, ayant $x(k)$ et $x(k+n)$ comme entrées et U comme sortie désirée (fig. 2.11). Si la dynamique du système est inconnue, le réseau peut être entraîné on-line, avec les signaux du système pour capturer la dynamique inverse de celui-ci. Si un modèle précis du système est connu, à priori, on peut simplement entraîner le réseau off-line avec les données générées à partir du modèle.

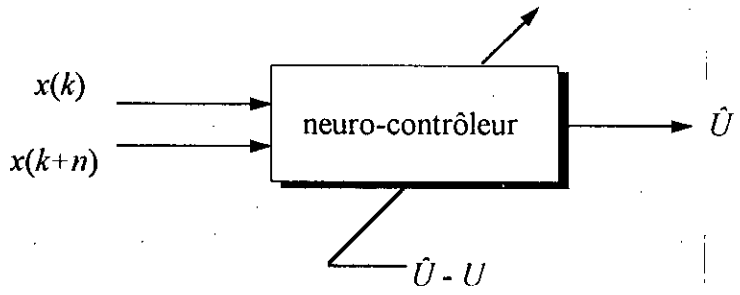


Fig. 2.11 modèle neuronal pour l'identification de la dynamique inverse du système.

B. Commande du système par le RNA

La section précédente nous a présenté la représentation neuronale de la dynamique inverse du système nonlinéaire. On explique maintenant comment le même réseau peut être utilisé pour générer les signaux de commande.

Assumons que le réseau a, complètement, appris la dynamique inverse du système de l'équation (2.24). Alors, donnant les mesures des états actuels $x(k)$ et des états futurs désirés x_d $(k+n)$, le réseau se charge de délivrer l'action de commande U par (fig.2.12) :

$$U = g\,(x(k), x_d(k+n)) \tag{2.25}$$

qui conduira l'état du système de $x(k)$ à $x_d(k+n)$ en, exactement, n étapes.

Le réseau utilisé pour la modélisation et la commande, possède 2n entrées et n sorties. Une architecture plausible pour ce réseau est de trois couches, avec 2n neurones dans la couche d'entrée, un certain nombre de neurones ayant des fonctions d'activation sigmoïdes, dans la couche cachée, et n neurones linéaires dans la couche de sortie. Les neurones à fonctions d'activation sigmoïdes sont utilisés pour capturer la nonlinéarité du système, et les neurones à fonctions d'activation linéaires sont utilisés pour permettre une approximation de U dans des intervalles arbitraires.

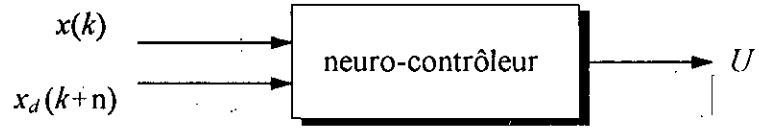


Fig. 2.12 génération des signaux de commande.

Remarque

Ici, le concept de système n'est pas limité au système à commander. Le réseau peut approximer la dynamique inverse du système et d'un contrôleur auxiliaire tel qu'un PID. Ceci est très intéressant, puisque ca nous permet de stabiliser le système, si jamais celui-ci est instable, à l'aide d'un contrôleur classique, puis utiliser le RNA pour commander le système global, qui lui est stable.

2.3.4 COMMANDE ADAPTATIVE NEURONALE [20]

Beaucoup de travaux ont été menés dans le domaine de la commande adaptative appliquée aux systèmes linéaires et, éventuellement, à quelques systèmes nonlinéaires. Dans ce cas, une connaissance, à priori, concernant le modèle du système, doit être disponible. Au contraire, peu de travaux ont été reportés sur la commande adaptative appliquée aux systèmes décrits par des équations différentielles (ou aux différences) nonlinéaires, et c'est par la commande de ces systèmes qu'on est concerné.

Deux approches distinctes existent pour la commande adaptative des systèmes [43]. Ce sont la *commande adaptative directe* et la *commande adaptative indirecte*. Dans la commande directe, les paramètres du contrôleur sont ajustés pour minimiser une norme liée à l'erreur de sortie. Dans la commande indirecte, les paramètres du système sont d'abord estimés, et les paramètres du contrôleur sont déterminés en assumant que les paramètres estimés du système sont les vrais paramètres. Les mêmes approches, utilisées pour la commande adaptative des systèmes linéaires, sont applicables aux systèmes nonlinéaires. Cependant, à la place des gains linéaires, des réseaux de neurones nonlinéaires sont utilisés. Ajoutons à cela, que c'est la technique de la commande adaptative à modèle de référence (MRAC) qui est utilisée.

A. Commande adaptative directe

La figure (2.13) montre la structure utilisée pour la commande adaptative directe du système. L'erreur entre la sortie du système et celle du modèle de référence est utilisée pour ajuster les poids du neuro-contrôleur.

B. Commande adaptative indirecte

La figure (2.14) montre la structure utilisée pour la commande adaptative indirecte du système. Dans ce cas, le modèle du système est inconnu, et les méthodes d'identification neuronale, présentées plus haut, sont utilisées on-line pour représenter son comportement. Les paramètres du neuro-contrôleur sont ajustés en se servant du modèle identifié.

Les erreurs e_i et e_c sont utilisées pour effectuer l'apprentissage du neuro-identificateur et du neuro-contrôleur réspectivement. Une fois le système identifié à un certain degré de précision, l'action de commande peut être enclenchée pour que la sortie du système suive la sortie d'un modèle de référence stable. Il est à noter que même si le système possède des sorties bornées pour des entrées bornées, la commande adaptative neuronale peut aboutir à des solutions non

bornées. Ainsi, pour une commande on-line, l'identification et la commande doivent procéder simultanément.

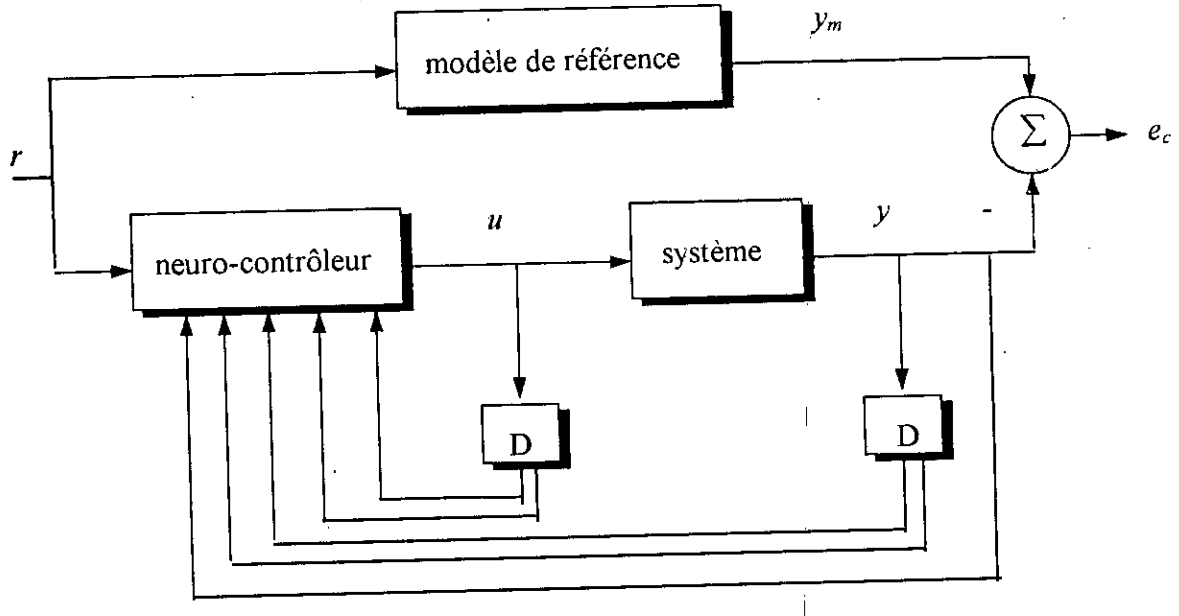


Fig. 2.13 commande adaptative neuronale directe d'un système.

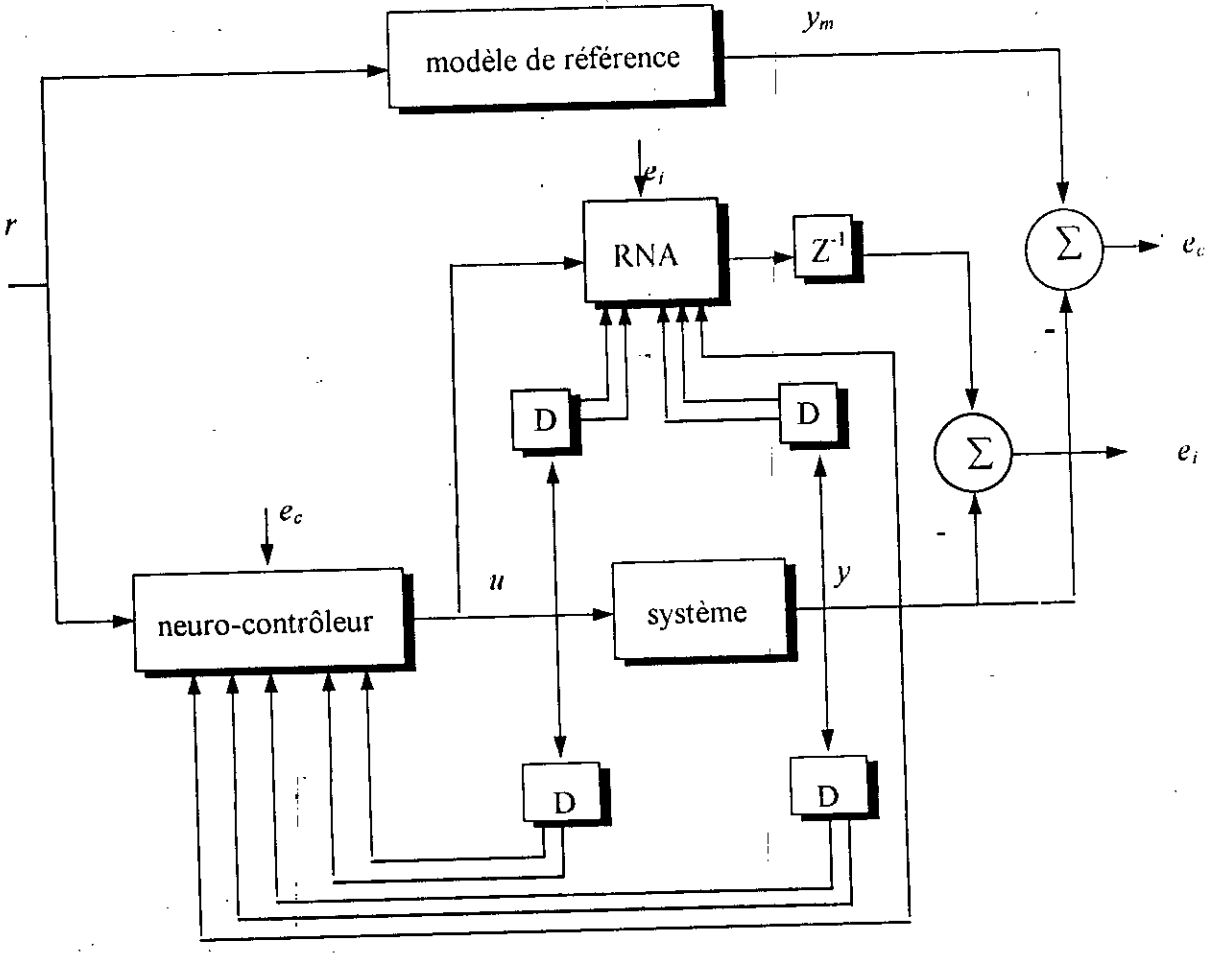


Fig. 2.14 commande adaptative neuronale indirecte d'un système nonlinéaire.

2.3.5 CRITIQUE ADAPTATIVE [28][37]

Les sections précédentes traitaient la manière avec laquelle les informations nécessaires pour l'apprentissage supervisé, peuvent être obtenues à partir des tâches de commande. Cependant, il existe certains problèmes de commande qui requièrent des méthodes qui ne font pas partie de celles utilisées en apprentissage supervisé. Supposons qu'on veut réaliser une commande neuronale pour améliorer la performance d'un système mesurée par un critère, qui en quelque sorte, évalue le comportement global du système. Par exemple, on peut vouloir minimiser l'énergie fournie par la commande pendant un certain temps. Ceci est possible en utilisant les méthodes d'apprentissage on-line, qui font partie de ce qu'on appelle: *l'apprentissage par renfort* (reinforcement learning).

L'apprentissage par renfort appartient à la classe des apprentissages non supervisés, on utilise donc la valeur du critère de performance, approximée par un réseau d'évaluation (critic network), comme signal supplémentaire pour l'entraînement d'un second réseau (action network), qui lui sert à contrôler le système (fig. 2.15). En d'autres termes, la sortie du réseau d'évaluation peut être considérée comme une deuxième fonction performance qui représente la somme des valeurs de la fonction performance originelle sur un intervalle de temps. Le réseau d'action à pour rôle de maximiser cette seconde fonction performance.

Ce type d'apprentissage pose, essentiellement, deux problèmes. Le premier est de construire un réseau d'évaluation capable d'estimer la performance du système et assez informatif pour permettre l'apprentissage. Le second est de déterminer comment ajuster les sorties du réseau d'action (neuro-contrôleur) pour améliorer cette performance. C'est pour ces raisons que certains chercheurs préfèrent les autres méthodes de commande neuronale par rapport à celle-ci [26] sans négliger, bien sûr, les bonnes performances de cette méthode, obtenues dans la commande de systèmes fortement nonlinéaires, tels que le pendule inversé [42]. Le résumé de cette méthode d'apprentissage est présenté dans l'annexe D.

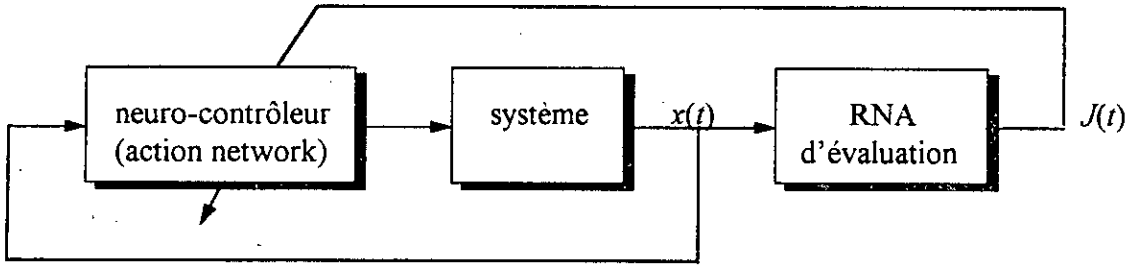


Fig. 2.15 système de critique adaptative simple.

2.4 COMMANDE NEURONALE ET THEORIE DE LA COMMANDE NONLINEAIRE [22]

Malgré leur avance, la plupart des méthodes de commande neuronale citées ci-dessus, sont de nature heuristique. Le succès réalisé en utilisant de telles méthodes, a suscité un nouvel intérêt, qui est de relier la théorie de la commande nonlinéaire aux techniques de synthèse des

neuro-contrôleurs. Le but est d'établir une méthodologie par laquelle les techniques de commande par réseaux de neurones peuvent être plus rigoureuses.

L'idée est d'utiliser les réseaux de neurones, comme outils nonlinéaires efficaces, pour la simulation de certaines fonctions, tout en se basant sur les techniques de synthèse de la commande nonlinéaire. L'avantage d'intégrer les RNA apparaît dans le cas où le modèle du système est inconnu.

2.4.1 STABILISATION PAR UN RETOUR LINEARISANT

Soit le système nonlinéaire discret, décrit par l'équation (2.5). On désire rendre notre système équivalent à un système linéaire, via les transformations suivantes :

- un changement de coordonnées dans l'espace d'état $z = \Phi(x)$, avec $\Phi(.)$ inversible et infiniment dérivable.
- Un retour d'état $u(k) = \Psi[x(k), v(k)]$.

Si ces deux transformations sont possibles, les techniques de la commande linéaire peuvent être utilisées pour contrôler le système. En appliquant les transformations précédentes, on aura :

$$z(k+1) = \Phi[x(k+1)] = \Phi[f(\Phi^{-1}(z(k)), \Psi(\Phi^{-1}(z(k)), v(k)))] \tag{2.26}$$

avec *z(k)* représentant les nouveaux états du système, et *v(k)* sa nouvelle entrée. Le système est alors dit linéarisable par un retour d'état nonlinéaire (feedback linearizable). Le modèle du système devient équivalent à :

$$z(k+1) = A\, z(k) + b\, v(k) \tag{2.27}$$

où A et b sont une paire commandable. En particulier, on peut choisir A et b ayant une forme canonique :

$$A = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ & & \vdots & & \\ 0 & 0 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ \\ 0 \\ 1 \end{bmatrix} \tag{2.28}$$

L'objectif est d'entraîner deux réseaux, RNA_Ψ et RNA_Φ, pour que le système global se comporte suivant le modèle (2.27), avec : $\hat{u} = RNA_\Psi(x,v)$ et $\hat{z} = RNA_\Phi(x)$. Le schéma bloc de la figure (2.16) illustre cette implémentation.

Puisque RNA_Φ est directement relié à la sortie désirée, ses poids peuvent être ajustés en utilisant la rétropropagation statique. Le modèle contient cependant, une boucle de retour, la rétropropagation dynamique doit être utilisée pour ajuster les poids de RNA_Ψ.

Cette méthode est efficace pour l'identification des fonctions Φ et Ψ quand le modèle du système est inconnu. Son inconvénient réside dans le temps de calcul durant l'apprentissage qui peut être très important.

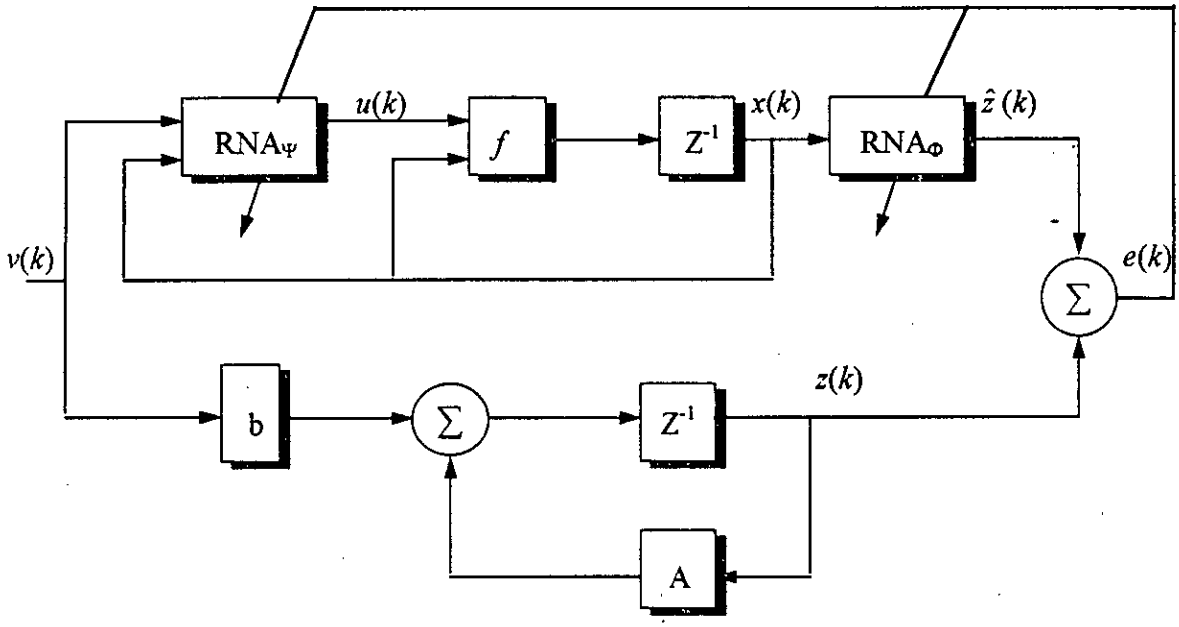


Fig. 2.16 architecture de la linéarisation par retour d'état.

2.4.2 STABILISATION DIRECTE

Bien qu'élégante, la linéarisation par retour d'état peut être appliquée qu'à une classe limitée de systèmes. De plus, même quand elle est possible, son implémentation décrite ci-dessus nécessite la rétropropagation dynamique, qui n'est pas très efficace du point de vue temps de calcul. Ainsi, des contrôleurs nonlinéaires, qui stabilisent directement le système, sont synthétisés et implémentés en utilisant les réseaux de neurones.

Théorème : Soit le système gouverné par l'équation d'état (2.5). Si ce système est localement commandable au voisinage d'un point d'équilibre, alors il existe un retour nonlinéaire $u(k) = g[x(k)]$ qui stabilise le système en n étapes autour de ce point d'équilibre.

g étant une fonction nonlinéaire, elle peut donc être approximée par un RNA.

Exemple : soit le système nonlinéaire décrit par :

$$x(k+1) = x(k) + u(k) + a\,u^2(k) \tag{2.29}$$

on veut stabiliser le système autour de l'origine.

$x(k+1) = 0 \Rightarrow x(k) + u(k) + a\,u^2(k) = 0$
ce qui donne :

$$u(k) = \frac{-1+\sqrt{1-4a\,x(k)}}{2a} = g(x(k)) \qquad (2.30)$$

La relation (2.30) sera approximée par un RNA et sera, par conséquent, utilisée pour générer les données d'apprentissage.

2.5 CONCLUSIONS

Ce chapitre nous a présenté un tour guidé concernant l'utilisation des réseaux de neurones artificiels dans le royaume de l'identification et de la commande des systèmes nonlinéaires. Les idées et techniques de base ont été présentées, tout en explorant le lien entre la science de la commande des systèmes et le domaine des réseaux neuronaux, sachant que ce nouvel outil reste ouvert à d'autres investigations et à des recherches futures.

D'autres méthodes de commande neuronale existent, et qu'on n'a pas présenté ici, car elles sont moins utilisées et sont encore dans l'étape d'exploration et d'expérimentation. Nous citerons: la commande par modèle interne, la commande prédictive neuronale et la commande adaptative en utilisant un indexe générique [19][24].

Les techniques de commande neuronale présentent une certaine souplesse quant à la connaissance du modèle du système à commander. En effet, l'utilisation des RNA nous épargne dans la majorité des cas, la connaissance du modèle du système. Il suffit donc d'identifier le comportement de celui-ci à l'aide de structures d'identification neuronale appropriées, ou alors acquérir un savoir faire capable de stabiliser le système via un apprentissage.

Néanmoins, des questions cruciales restent posées concernant la théorie de la stabilité en commande neuronale, le choix des données d'apprentissage (les signaux d'entraînement doivent couvrir, suffisamment, la plage de variation des entrées / sorties du système), et la robustesse (le maintien des propriétés telles que la stabilité et la convergence, dans l'absence d'un modèle dynamique du système). Pour cela, de nouveaux concepts, concernant la commande basée sur les RNA, doivent être explorés. Ajoutant à ceci qu'il est désirable de développer des critères capables de désigner les types de problèmes réels auxquels les réseaux neuronaux sont applicables.

CHAPITRE 3

INTRODUCTION AUX SYSTEMES FLOUS ET A LA COMMANDE FLOUE

" Plus loin vont les lois mathématiques dans la description de la réalité, plus elles sont incertaines. Et plus elles sont certaines, moins est leur description de la réalité. "

A. Einstein (1879-1955)

3.1 INTRODUCTION

Durant ces trois dernières décennies, la commande floue s'est révélée un domaine de recherche très attirant, faisant partie des applications de la théorie de la logique floue. S'appuyant sur les travaux de L. Zadeh [53], différentes synthèses de contrôleurs flous ont été élaborées. Depuis, les applications ont connu un essor important, dont les principales sont résumées dans le tableau (3.1) [46][50]. Une liste succincte des applications de la logique floue englobe [45]:

- en commande : transmission automatique (Nissan, Subaru), auto-stationnement d'un modèle de voiture (Tokyo Tech. Univ.), commande du vol d'avion (Rockwell Corp.), ...
- en optimisation : analyse du stock du marché (Yamaichi Securities),
- en traitement du signal : ajustement des images TV (Sony), reconnaissance de manuscrits (Sony Palm Top), autofocus des cameras vidéo (Sanyo/Fisher, Canon),...

L'intérêt attaché à cette discipline est dû au fait que la logique floue est plus proche de la manière avec laquelle l'être humain pense que la logique traditionnelle, puisqu'elle s'exprime en langage naturel. Elle fournit un moyen efficace pour capturer la nature inexacte et approximative du monde réel. En effet, la connaissance existe, en général, sous deux formes : une *connaissance objective*, qui est utilisée tout le temps dans la formulation des problèmes (e.g., les modèles mathématiques), et une *connaissance subjective*, qui représente l'information linguistique, impossible à quantifier par un modèle mathématique (e.g., les règles régissant le fonctionnement d'un système). La connaissance subjective est souvent ignorée, parfois utilisée pour évaluer les performances d'une conception. Cependant, les deux formes de connaissance peuvent être coordonnées d'une manière logique en utilisant la logique floue, dans le but d'améliorer les performances du système.

Justement, la logique floue est basée sur un *principe* dit *d'incompatibilité*, énoncé par le père fondateur de cette discipline, L. Zadeh, qui stipule [45] : « plus la complexité d'un système croit, notre aptitude à donner des explications précises et significatives de son comportement diminue, jusqu'à ce que précision et signification deviennent deux propriétés mutuellement exclusives, i.e., plus on veut tendre vers les problèmes du monde réel, plus le flou devient la solution ».

Ce présent chapitre nous donne une vue d'ensemble sur la théorie des ensembles flous, ainsi que la logique floue, qui sont à la base de la synthèse des systèmes flous, et parmi eux les contrôleurs flous.

Année	Auteurs	applications
1972	Zadeh	Approche linguistique
1974	Mamdani & Assilian	Commande d'une machine à vapeur
1980	Fukami et al.	L'inférence conditionnelle floue
1983	Sugeno & Takagi	Dérivation des règles de commande floue
1985	Togar & Watanabe	Processeur flou
1988	Dubois & Prade	Raisonnement approximatif
1991	Barrat et al.	Commande floue de la température d'un four

Tab. 3.1 principaux travaux dans le domaine de la logique floue.

3.2 CONCEPTS FLOUS

A l'inverse des concepts à valeur duale (vrai ou faux), les concepts flous sont de nature non précise, et donc pas claire. Une collection de termes à caractère flou, très utilisés dans les domaines de commande et de traitement du signal et communication, est présentée dans la table 3.2 [45]. Ces termes sont souvent utilisés dans des contextes flous, et donnent une information qui est plus significative que si l'on utilisait les valeurs nettes (numériques) de ces termes.

La corrélation est un exemple intéressant, car elle peut être exprimée mathématiquement, pour un ensemble de données, en calculant un nombre la représentant. On assume que la corrélation a été normalisée à des valeurs entre 0 et 1 et que pour un ensemble de données elle est égale à 0.15. En expliquant à une tierce personne la quantité de corrélation qui existe entre ces données, il est plus significatif de dire: « ces données ont une basse corrélation », que de lui dire la corrélation est de 0.15. En faisant ceci, on aura *fuzzifié* (rendu flou) la valeur nette 0.15 dans l'ensemble flou " basse corrélation ".

Terme	Usage contextuel
Corrélation	basse, moyenne, haute, parfaite
Erreur	large, moyenne, petite, grande, très large, très petite, presque nulle
Fréquence	haute, basse, très haute
Stabilité	stable (amorti, sur amorti, sous amorti), instable

Tab. 3.2 termes à usage contextuel flou.

3.3 QU'EST CE QU'UN SYSTEME FLOU ?

Un système flou (SF) est une relation nonlinéaire qui relie un vecteur de données d'entrée à un scalaire en sortie (le cas de plusieurs sorties peut être décomposé en sous systèmes flous de plusieurs entrées et une seule sortie). La figure (3.1) représente un système flou, largement utilisé en commande floue et dans les applications de traitement du signal. Il faut noter qu'un SF relie des entrées nettes (nombres) à des sorties nettes. Il contient quatre composantes: les règles, le fuzzificateur, l'engin d'inférence et le défuzzificateur.

- Les règles peuvent être fournies par des experts ou peuvent être extraites de données numériques. Dans les deux cas, elles sont de la forme « Si-Alors », e.g., « Si la température est basse, Alors la tension aux bornes de la résistance doit être grande ». Ceci étant, on a besoin de quantifier ce qu'on appelle les *variables linguistiques* (5°C, basse), à travers des *fonctions d'appartenance*. De plus, on a besoin de comprendre comment combiner plusieurs règles.

- Le fuzzificateur relie des nombres nets à des ensembles flous. On l'utilise pour pouvoir activer les règles qui sont exprimées à l'aide de variables linguistiques associées à des ensembles flous.

- L'engin d'inférence relie des ensembles flous à d'autres ensembles flous. Il s'occupe de la combinaison des différentes règles. Il existe plusieurs procédures d'inférence logique [48], mais seulement quelques unes parmi elles sont utilisées dans les applications de la logique floue.

- Le défuzzificateur relie des ensembles flous à des nombres nets en sortie. Dans le cas de la commande floue, ceci correspond à fournir une action de commande, à partir d'un ensemble flou auquel la commande appartient.

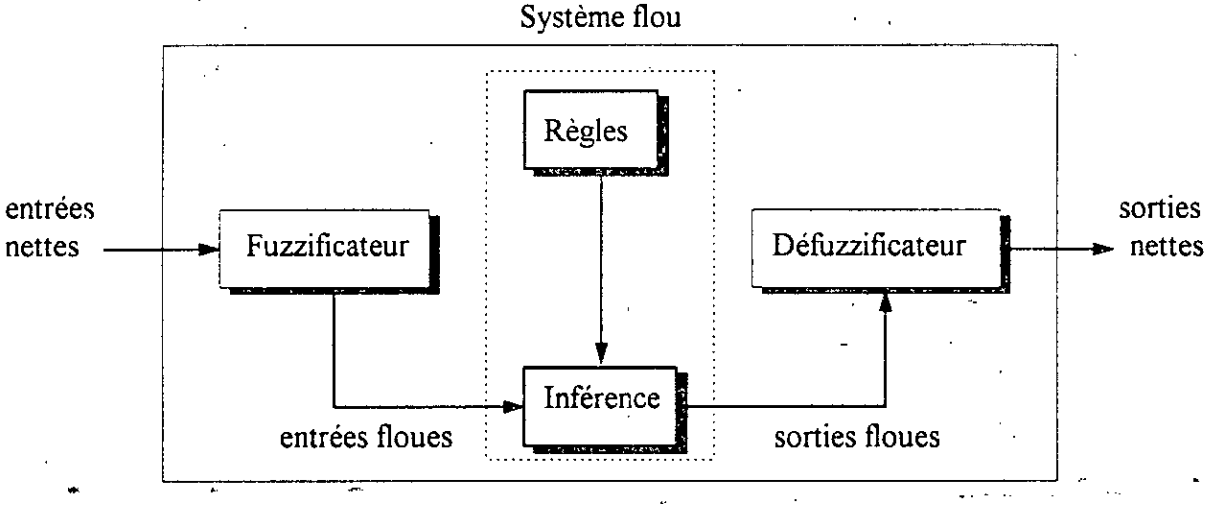


Fig. 3.1 schéma bloc représentant un système flou.

3.4 APPRET SUR LES ENSEMBLES FLOUS [45]

3.4.1 ENSEMBLES NETS

Rappelons qu'un ensemble net A peut être définit en donnant la liste de ses éléments ou en identifiant ces derniers à partir des caractéristiques les liant à cet ensemble. Soit alors :

$A = \{ x / x$ vérifie certaines conditions $\}$.

De même, on peut introduire une fonction d'appartenance binaire (0,1), notée $\mu_A(x)$, telle que:

$$\mu_A(x) = \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si } x \notin A \end{cases} \qquad (3.1)$$

Exemple : Considérons l'ensemble des systèmes physiques U. Soit A un sous ensemble de U représentant, uniquement, les systèmes linéaires. Qu'un système donné soit linéaire ou pas (appartenant à A ou pas), ça n'a rien de flou dans cette définition. Ainsi, si un système est nonlinéaire la valeur de la fonction d'appartenance pour celui-ci est nulle.

3.4.2 ENSEMBLES FLOUS

Un ensemble flou F est caractérisé par une fonction d'appartenance $\mu_F(x)$ qui prend ses valeurs dans l'intervalle [0,1]. Cette fonction nous donne le degré d'appartenance d'un élément à l'ensemble F.

Exemple : Considérons l'ensemble des fréquences variant des plus basses fréquences aux plus hautes fréquences. Soit les deux sous ensembles F et F' représentant les basses fréquences et les moyennes fréquences, respectivement. Une fréquence égale à 100 Hz peut appartenir aux deux ensembles F et F' avec différents degrés de similarité (fig. 3.2). Dans ce contexte, la notion du flou est apparente, puisque les termes basses et moyennes fréquences sont imprécis (pas nets).

Un ensemble flou F *dans* U peut être représenté comme un ensemble de paires, constituées de l'élément x et de la valeur de sa fonction d'appartenance : $F = \{(x, \mu_F(x)) / x \in U\}$.

Si U est continu, F est noté par :

$$F = \int_U \mu_F(x) / x \tag{3.2}$$

Si U est discret, F est noté par :

$$F = \sum_U \mu_F(x) / x \tag{3.3}$$

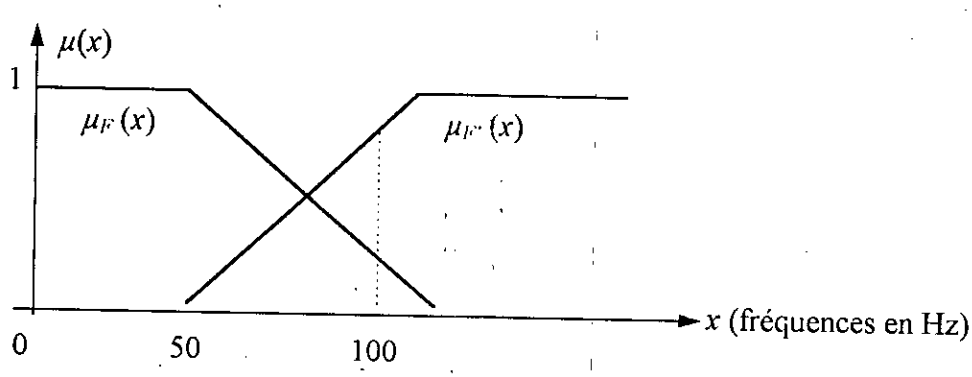


Fig. 3.2 fonctions d'appartenance des fréquences aux ensembles flous: « basses fréquences » et « moyennes fréquences ».

3.4.3 VARIABLES LINGUISTIQUES

Ce sont des variables dont les valeurs ne sont pas des nombres, mais des mots ou des phrases exprimés en langage naturel ou artificiel. La raison pour laquelle on utilise des mots ou des phrases à la place des nombres, est que le caractère linguistique est moins spécifique que le caractère numérique.

Une variable linguistique u est, généralement, décomposée en un ensemble de termes, $T(u)$, qui couvre tout son domaine de variation.

Exemple : la pression (u), peut être interprétée comme variable linguistique. Elle peut être décomposée en l'ensemble des termes suivant :

T (pression) = { faible, basse, moyenne, forte, élevée }. Chaque terme dans T est caractérisé par un ensemble flou dans le domaine de variation U = [100 psi, 2300 psi], dont les fonctions d'appartenance sont montrées dans la figure (3.3).

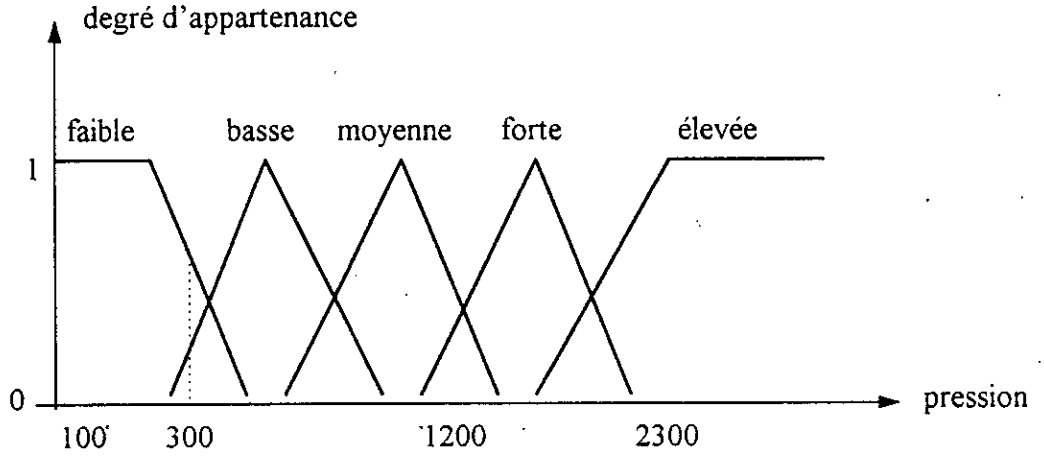


Fig. 3.3 fonctions d'appartenance de T (pression) = { faible, basse, moyenne, forte, élevée }.

3.4.4 FONCTIONS D'APPARTENANCE

Dans les applications de la logique floue, les fonctions d'appartenance sont associées avec les termes qui apparaissent dans les antécédents ou les conséquents des règles, comme on verra par la suite. Les formes les plus utilisées pour ces fonctions sont triangulaires, trapézoïdales, linéaires par morceaux et Gaussiennes. Elles sont, généralement, normalisées entre 0 et 1.

Ces fonctions sont choisies, arbitrairement, par l'utilisateur. Ainsi, deux fonctions d'appartenance choisies par deux utilisateurs, peuvent être différentes, dépendantes de leurs expériences et perspectives. Récemment, quelques fonctions d'appartenance sont conçues par des procédures d'optimisation.

Le choix du nombre de fonctions d'appartenance à utiliser lors d'une application, est très important. En effet, plus grande est la résolution en utilisant plus de fonctions d'appartenance, plus grand et plus complexe est le temps de calcul. Le chevauchement qui existe entre les fonctions d'appartenance constitue un avantage pour la conception des systèmes flous. Cela vient du fait qu'« un verre peut être partiellement rempli et partiellement vide en même temps ». De cette manière, on est capable de distribuer nos décisions sur plus d'une classe d'entrées, ce qui rend les systèmes flous robustes.

Pour des raisons de terminologie, notons que l'ensemble des éléments x pour lesquels $\mu_F(x)>0$, est appelé *support* de l'ensemble flou F, l'élément x pour lequel $\mu_F(x) = 0.5$ est appelé *crossover point*, et l'ensemble flou qui a pour support un seul élément x_0, avec $\mu_F(x_0) = 1$, est appelé *fuzzy singleton*.

Remarque

Il existe ce qu'on appelle des modificateurs linguistiques, dont le rôle est de modifier le sens d'un terme ou plus généralement, d'un ensemble flou. Par exemple, si basse température est un ensemble flou, alors très basse température, plus ou moins basse température et extrêmement basse température sont des exemples de modificateurs appliqués à cet ensemble flou. Ces modificateurs agissent directement sur la fonction d'appartenance de l'ensemble flou qu'ils modifient. Par exemple :

- La concentration : si basse température possède une fonction d'appartenance $\mu_F(x)$, alors *très* basse température a une fonction d'appartenance $\mu_{F'}(x) = [\mu_F(x)]^2$.
- La dilatation : si basse température possède une fonction d'appartenance $\mu_F(x)$, alors *moins* basse température a une fonction d'appartenance $\mu_{F'}(x) = [\mu_F(x)]^{1/2}$.

3.4.5 OPERATIONS SUR LES ENSEMBLES FLOUS

Pour les ensembles flous, l'union, l'intersection et le complément sont définis par le biais de leurs fonctions d'appartenance. Soient A et B deux ensembles flous définis par leurs fonctions d'appartenance $\mu_A(x)$ *et* $\mu_B(x)$. L'une des définitions concernant l'union floue et l'intersection floue est donnée par :

$$\mu_{A\cup B}(x) = \max\left[\mu_A(x), \mu_B(x)\right] \tag{3.4}$$

$$\mu_{A\cap B}(x) = \min\left[\mu_A(x), \mu_B(x)\right] \tag{3.5}$$

la fonction d'appartenance pour le complément flou est donnée par :

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{3.6}$$

Evidemment, ces trois définitions sont motivées par leurs semblables concernant les ensembles nets. Il y a à noter qu'un élément x peut appartenir à l'ensemble et à son complément simultanément, ce qui n'est pas le cas pour les ensembles nets.

Les opérateurs « max » et « min » ne sont pas les seuls utilisés pour modéliser l'union et l'intersection. Zadeh propose deux autres opérateurs pour l'union et l'intersection floues, appelés somme algébrique et produit algébrique, respectivement :

$$\mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x) \tag{3.7}$$

$$\mu_{A\cap B}(x) = \mu_A(x)\mu_B(x) \tag{3.8}$$

D'autres opérateurs qui ont une base axiomatique, étaient introduits et nommés: t-conorm pour l'union floue ($\oplus$), et t-norm pour l'intersection floue (*) [46]. Seulement, la plupart des applications des ensembles flous utilisent le min ou le produit algébrique pour l'intersection floue, et le max pour l'union floue.

3.4.6 RELATIONS ET COMPOSITIONS FLOUES

De même qu'une relation nette, une relation floue représente le degré de présence ou d'absence d'une association entre deux ou plusieurs ensembles flous. Ici, on limitera notre attention à deux ensembles U et V, et aux relations binaires $R(U,V)$. Une relation floue est, elle même, un ensemble flou dans l'espace $U\times V$. Elle est caractérisée par une fonction d'appartenance $\mu_R(x,y)$ où $x\in U$ et $y\in V$ et $\mu_R(x,y) \in [0,1]$. Les opérateurs définis plus haut, concernant l'union, l'intersection et le complément des ensembles flous, sont applicables aux relations floues.

La composition de deux relations floues qui partagent un même ensemble, $P(U,V)$ et $Q(V,W)$, est notée par :

$$R(U,W) = P(U,V) \circ Q(V,W) \tag{3.9}$$

et est définie dans $U\times W$.

Définitions

- La composition max-min des relations $P(U,V)$ et $Q(V,W)$ est définie par la fonction d'appartenance $\mu_{P\circ Q}(x,z)$, telle que :

$$\mu_{P\circ Q}(x,z) = \max_y\left[\min(\mu_P(x,y),\mu_Q(y,z))\right] \tag{3.10}$$

- La composition max-produit des relations $P(U,V)$ et $Q(V,W)$ est définie par la fonction d'appartenance $\mu_{P\times Q}(x,z)$, telle que :

$$\mu_{P\times Q}(x,z) = \max_y\left[\mu_P(x,y)\mu_Q(y,z)\right] \tag{3.11}$$

Une formule mathématique générale pour la composition de deux relations, motivée par les équations (3.10) et (3.11), est la composition *sup-star* suivante :

$$\mu_{P\circ Q}(x,z) = \sup_{y\in V}\left[\mu_P(x,y) * \mu_Q(y,z)\right] \tag{3.12}$$

Bien qu'il est possible d'utiliser d'autres opérateurs t-norm, les compositions sup-star les plus utilisées sont le sup-min et le sup-produit. La figure (3.4) nous donne une interprétation, sous forme de schéma bloc, de la composition sup-star [45].

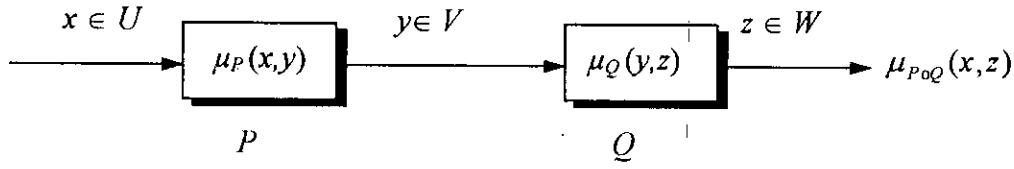


Fig. 3.4 interprétation en schéma bloc de la composition sup-star.

Supposons que la relation P est définie que dans l'ensemble U, telle que $\mu_P(x,y)$ devient $\mu_P(x)$ et $V = U$. La figure (3.4) se réduit à la figure (3.5) et nous montre comment un ensemble flou peut activer une relation floue, ce qui constitue un cas spécial d'une implication floue, comme on le verra par la suite. Dans ce cas, la fonction d'appartenance concernant la composition sup-star devient :

$$\mu_{P \circ Q}(z) = \sup_{x \in U} \left[\mu_P(x) * \mu_Q(x,z) \right] \tag{3.13}$$

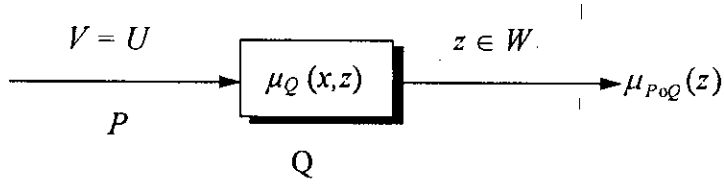


Fig. 3.5 interprétation en schéma bloc de la composition sup-star, quand la première relation est juste un ensemble flou.

3.5 APPRET SUR LA LOGIQUE FLOUE [45]

Les règles constituent un bloc important dans le fonctionnement d'un SF. Elles sont exprimées sous forme d'implications logiques, i.e., des propositions « si u est A, alors v est B », où la partie Si de l'implication est appelée antécédent et la partie Alors est appelée conséquent. Dans la logique traditionnelle, toute proposition est générée en utilisant trois opérations fondamentales : la conjonction ($\wedge$), la disjonction ($\vee$) et la négation ($\sim$). Une règle représente une relation de type spécial entre les ensembles flous A et B, sa fonction d'appartenance est notée $\mu_{A \to B}(x,y)$. Notre rôle est de faire un choix approprié de cette fonction d'appartenance. Pour se faire, on utilisera l'équivalence mathématique qui existe entre la logique et la théorie des ensembles, illustrée dans la table (3.3).

Logique	Théorie des ensembles
$\wedge$	$\cap$
$\vee$	$\cup$
$\sim$	$(-)$

Tab. 3.3 équivalence mathématique entre la logique et la théorie des ensembles.

En utilisant le fait que :

$$(A \rightarrow B) \leftrightarrow \sim [A \wedge (\sim B)]$$
et $$(A \rightarrow B) \leftrightarrow (\sim A) \vee B$$

on obtient les fonctions d'appartenance $\mu_{A \rightarrow B}(x,y)$:

$$\begin{aligned} \mu_{A \rightarrow B}(x,y) &= 1 - \mu_{A \cap \bar{B}}(x,y) \\ &= 1 - \min\left[\mu_A(x),\ 1 - \mu_B(y)\right] \end{aligned} \tag{3.14}$$

et

$$\begin{aligned} \mu_{A \rightarrow B}(x,y) &= \mu_{\bar{A} \cup B}(x,y) \\ &= \max\left[1 - \mu_A(x),\ \mu_B(y)\right] \end{aligned} \tag{3.15}$$

La fonction d'appartenance $\mu_{A \rightarrow B}(x,y)$ mesure le degré de vérité de la relation d'implication entre x et y.

Bien sûr, la logique floue puise ses notions dans la logique nette (traditionnelle). Cependant, utiliser les formules (3.14) et (3.15), dans les applications liées à logique floue, est inadéquat, parce que ces relations ne vérifient pas le principe de causalité [45], qui est une caractéristique importante dans la modélisation des systèmes physiques. Ceci étant, d'autres relations d'implication, qui pallient à ce problème, étaient définies par :

Mamdani : $$\mu_{A \rightarrow B}(x,y) = min[\ \mu_A(x), \mu_B(y)\ ] \tag{3.16}$$

Larsen : $$\mu_{A \rightarrow B}(x,y) = \mu_A(x)\ \ \mu_B(y) \tag{3.17}$$

Les raisons du choix de ces définitions ne sont pas basées sur le principe de causalité, bien qu'ils la vérifient, mais sur leur simplicité de calcul. Aujourd'hui, les implications min et produit sont les plus utilisées dans les applications de la logique floue.

3.5.1 LES REGLES D'INFERENCE MODUS PONENS ET MODUS TOLLENS

En logique traditionnelle, il existe deux règles d'inférence qui permettent, justement, d'interpréter une règle comme étant un système avec des entrées et sorties.

A. Modus Ponens - la règle se présente comme suit :

premier antécédent : « x est A »
deuxième antécédent : « si x est A, alors y est B »

conséquence: « y est B »

l'inférence Modus Ponens peut, aussi, être exprimée par : $(A \wedge (A \rightarrow B)) \rightarrow B$.

B. Modus Tollens - la règle se présente comme suit :

premier antécédent : « y n'est pas B »
deuxième antécédent : « si x est A, alors y est B »

conséquence: « x n'est pas A »

Bien que l'inférence Modus Ponens joue un rôle important dans les applications de la logique floue, l'inférence Modus Tollens n'est pas encore très utilisée.
La version généralisée du Modus Ponens est :

premier antécédent : « x est A^* »
deuxième antécédent : « si x est A, alors y est B »

conséquence: « y est B^* »

Les ensembles flous A^* et B^* ne sont pas forcement les ensembles A et B, mais ils sont similaires. Une interprétation, sous forme de système, du Modus Ponens généralisé, est donnée dans la figure (3.6), de laquelle on peut conclure que le Modus Ponens n'est autre qu'une composition floue où la première relation floue est l'ensemble flou A^*. Par conséquent, la fonction $\mu_{B^*}(y)$ est obtenue de la composition sup-star, en comparant la figure (3.5) et (3.6) :

$$\mu_{B^*}(y) = \sup_{x \in A^*} \left[\mu_{A^*}(x) * \mu_{A \to B}(x,y) \right] \tag{3.18}$$

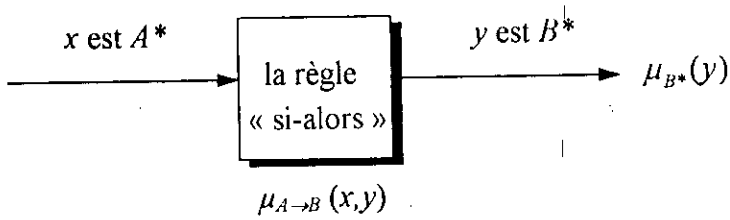


Fig. 3.6 interprétation, sous forme de schéma bloc, du Modus Ponens généralisé.

3.6 SYSTEMES FLOUS

Maintenant, on va présenter plus en détails, les quatre éléments de la figure (3.1), représentant un système flou. Ainsi, on sera capable de faire la conception de chaque élément et surtout, trouver la formule mathématique qui lie l'entrée du système flou à sa sortie.

3.6.1 LES REGLES

Un ensemble de règles floues consiste en une collection de propositions SI-ALORS qui peuvent être exprimées de la manière suivante :

$$R^{(l)} : \text{ SI } u_1 \text{ est } F_1^l \text{ et } u_2 \text{ est } F_2^l \text{ et } \ldots u_p \text{ est } F_p^l, \text{ ALORS } v \text{ est } G^l \qquad (3.19)$$

où l = 1,2,...,M, F_i^l et G^l sont des ensembles flous dans R (ensemble des nombres réels). Le vecteur $u = (u_1, u_2, \ldots, u_p)^T$ et le scalaire v sont des variables linguistiques. Ces règles sont à antécédents multiples. L'exemple suivant montre comment les règles peuvent être extraites à partir du problème à traiter.

Exemple [45]: La figure (3.7) montre le système balle et poutre. La poutre pivote en lui appliquant un couple au centre de rotation et la balle est libre de se mouvoir le long de la poutre. La commande $u(t)$ est l'accélération de θ. Le problème est de concevoir un contrôleur qui conduit la balle à l'origine et la maintient en ce point.
Ce système est nonlinéaire et possède quatre variables d'état, $r(t)$, $dr(t)/dt$, $\theta(t)$, $d\theta(t)/dt$. On peut alors concevoir quatre règles générales ayant quatre entrées et une sortie:

- $R^{(1)}$: SI (la position radiale) r est positive et (la vitesse radiale) $dr(t)/dt$ est proche de zéro, et (la position angulaire) θ est positive et (la vitesse angulaire) $d\theta(t)/dt$ est proche de zéro, ALORS (la commande) u est négative.
- $R^{(2)}$: SI r est négative et $dr(t)/dt$ est proche de zéro, et θ est négative et $d\theta(t)/dt$ est proche de zéro, ALORS u est positive.
- $R^{(3)}$: SI r est positive et $dr(t)/dt$ est proche de zéro, et θ est négative et $d\theta(t)/dt$ est proche de zéro, ALORS u est positive grande.
- $R^{(4)}$: SI r est négative et $dr(t)/dt$ est proche de zéro, et θ est positive et $d\theta(t)/dt$ est proche de zéro, ALORS u est négative grande.

Ces règles sont déduites à partir de l'expérience acquise durant le fonctionnement du système.

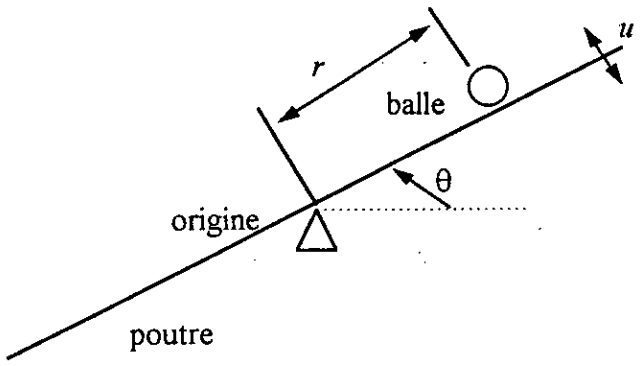


Fig. 3.7 le système balle et poutre.

3.6.2 L'ENGIN D'INFERENCE FLOUE

L'engin d'inférence floue à pour rôle de combiner les règles floues de l'ensemble des règles, en utilisant les principes de la logique floue déjà évoqués. Chaque règle est interprétée comme une implication floue, avec $F_1^l \times F_2^l \times \ldots \times F_p^l \equiv A$ et $G^l \equiv B$. On traite l'engin d'inférence floue

comme un système qui lie des ensembles flous à d'autres ensembles flous par le biais de $\mu_{A\to B}(x,y)$, où x = u et $y = v$. Par conséquent :

$$\begin{aligned}\mu_{R(l)}(x,y) &= \mu_{A\to B}(x,y) \\ &= \mu_{F1l}(x_1) * \ldots * \mu_{Fpl}(x_2) * \mu_{Gl}(y)\end{aligned} \tag{3.20}$$

où * est un opérateur t-norm (minimum ou produit). La règle $R^{(l)}$ possède p entrées, données par l'ensemble flou A_x, ayant pour fonction d'appartenance :

$$\mu_{Ax}(x) = \mu_{x1}(x_1) * \ldots * \mu_{xp}(x_p) \tag{3.21}$$

Chaque règle $R^{(l)}$ détermine un ensemble flou $B^l = A_x \circ R^{(l)}$, tel que (voir la relation (3.12)) :

$$\begin{aligned}\mu_{B^l}(y) &= \mu_{Ax} \circ R^{(l)}(y) \\ &= \sup_{x\in Ax}\left[\mu_{Ax}(x) * \mu_{A\to B}(x,y)\right]\end{aligned} \tag{3.22}$$

Cette équation nous donne la relation entre l'ensemble flou qui excite une règle de l'engin d'inférence, et l'ensemble flou qui constitue la sortie de cet engin. L'ensemble flou final B = Ax o[$R^{(1)}$, $R^{(2)}$, ..., $R^{(M)}$] qui est déterminé par l'activation de toutes les règles, est obtenu en combinant les différents $\mu_{Bl}(y)$ (l = 1,2,...,M). En général, ces fonctions d'appartenance sont combinées en utilisant un opérateur t-conorm (maximum), i.e., $B = B^1 \oplus B^2 \oplus \ldots \oplus B^M$.
La question qui reste posée est, comment trouver la sortie du système flou y_0 à partir de $\mu_{Bl}(y)$? C'est l'objet de la défuzzification.

3.6.3 FUZZIFICATION

Le fuzzificateur relie un vecteur net (numérique) $x = (x_1, x_2, \ldots, x_n)^T$ à un ensemble flou A*. Le fuzzificateur le plus utilisé est le singleton fuzzifier qui n'est autre qu'un fuzzy singleton, c'est à dire que A* a pour support x' tel que :

$$\mu_{A^*}(x) = \begin{cases} 1 & \text{si } x = x' \\ 0 & \text{si } x \neq x' \end{cases} \tag{3.23}$$

en utilisant un singleton fuzzifier, le supremum dans la relation de composition (3.22), disparaît et on aura :

$$\mu_{B^l}(y) = \mu_{Ax} \circ R^{(l)}(y) = \mu_{A\to B}(x',y) \tag{3.24}$$

Le singleton fuzzifier n'est pas le seul utilisé, spécialement quand les données sont bruitées. Un nonsingleton fuzzifier est un autre moyen pour palier à ce problème. Il s'agit d'utiliser un ensemble flou A* pour lequel $\mu_{A^*}(x') = 1$, et $\mu_{A^*}(x)$ décroît de l'unité quand x s'éloigne de x'. Des fonctions d'appartenance Gaussiennes ou triangulaires sont utilisées concernant A*. Plus large est le support de ces fonctions, plus grande est l'incertitude sur x'.

3.6.4 DEFUZZIFICATION

Le défuzzificateur produit une sortie nette (numérique) pour le système flou, à partir de l'ensemble flou fourni par l'engin d'inférence. Plusieurs défuzzificateurs ont été proposés dans la littérature; cependant, ils n'ont aucune base scientifique [45]. Puisqu'on est intéressé par l'application de la logique floue en commande, le critère du choix du défuzzificateur est sa simplicité de calcul. Ce critère nous mène vers les candidats suivants :

A. Défuzzificateur maximum

Ce défuzzificateur nous fournit une sortie y pour laquelle $\mu_B(y)$ est un maximum. Cependant, ce type de défuzzificateur présente un certain inconvénient lorsqu'il existe tout un intervalle des y pour lesquels $\mu_B(y)$ est un maximum.

B. Défuzzificateur moyenne des maxima

Après examen de l'ensemble flou B, ce défuzzificateur détermine, d'abord, les valeurs de y pour lesquelles $\mu_B(y)$ est un maximum, puis calcule la moyenne de ces valeurs pour sortie. Malheureusement, ceci peut aussi mener à des résultats aberrants. La figure (3.8) illustre une situation dans laquelle la sortie du défuzzificateur est une valeur de y pour laquelle $\mu_B(y)$ est nulle.

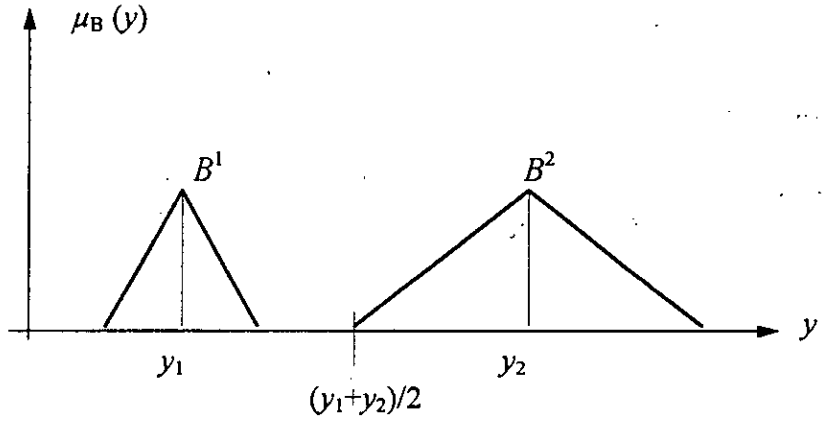


Fig. 3.8 exemple dans le quel la défuzzification par la moyenne des maxima n'a aucun sens.

C. Défuzzificateur centroïde

Ce défuzzificateur détermine le centre de gravité (centroïde) $\bar{y}$ de B, et donne cette valeur en sortie du SF. On obtient alors :

$$\bar{y} = \frac{\sum_{i=1}^{I} y_i \mu_B(y_i)}{\sum_{i=1}^{I} \mu_B(y_i)} \tag{3.25}$$

I étant le nombre de valeurs de y constituant le support de $\mu_B(y)$.

D. Défuzzificateur centroïde modifié

Soit $\bar{y}^l$ le centre de gravité de l'ensemble flou B, associé à l'activation de la règle R(l). Ce défuzzificateur évalue en premier $\mu_{Bl}(y)$ au point $\bar{y}^l$, puis calcule la sortie du système flou par:

$$y_h = \frac{\sum_{l=1}^{M} \bar{y}^l \mu_{Bl}(\bar{y}^l)}{\sum_{l=1}^{M} \mu_{Bl}(\bar{y}^l)} \tag{3.26}$$

Ce défuzzificateur a pour avantage de prendre en considération la forme de chaque fonction d'appartenance $\mu_{Bl}(y)$ individuellement.

3.6.5 FONCTIONS FLOUES DE BASE (FUZZY BASIS FUNCTIONS)

Maintenant, on veut trouver une formule mathématique $y = f(x)$, qui lie l'entrée du système flou x à sa sortie y. Si on choisit le singleton fuzzifier, la composition max-produit, l'inférence produit et le défuzzificateur centroïde modifié, il est facile de montrer que :

$$y = f(x) = \frac{\left[\sum_{l=1}^{M} \bar{y}^l \prod_{i=1}^{p} \mu_{F_i^l}(x_i)\right]}{\left[\sum_{l=1}^{M} \prod_{i=1}^{p} \mu_{F_i^l}(x_i)\right]} \tag{3.27}$$

pour obtenir (3.27) il suffit de remplacer dans (3.26) la formule de $\mu_{Bl}(\bar{y}^l)$, où :

$$\mu_{Bl}(\bar{y}^l) = \mu_{A \to B}(x', \bar{y}^l) = \left[\prod_{i=1}^{p} \mu_{F_i^l}(x'_i)\right] \mu_{Gl}(\bar{y}^l)$$

$$= \prod_{i=1}^{p} \mu_{F_i^l}(x'_i) \tag{3.28}$$

et on assume que $\mu_{Gl}(\bar{y}^l) = 1$.

La relation (3.27) peut être représentée par :

$$y = f(\mathrm{x}) = \sum_{l=1}^{M} \bar{y}^{l} \phi_{l}(\mathrm{x}) \tag{3.29}$$

ϕ_l (x) sont appelées *fonctions floues de base* (fuzzy basis functions), et sont données par :

$$\phi_{l}(\mathrm{x}) = \frac{\left[\prod_{i=1}^{p} \mu_{F_i^l}(x_i)\right]}{\left[\sum_{l=1}^{M} \prod_{i=1}^{p} \mu_{F_i^l}(x_i)\right]} \tag{3.30}$$

En faisant ceci, on arrive à placer les systèmes flous dans la perspective de l'approximation des fonctions, une caractéristique qu'ils partagent avec les réseaux de neurones artificiels. Notons que ces fonctions de base sont nonlinéaires, ce qui confère aux systèmes flous le caractère de nonlinéarité. De plus, à l'inverse des fonctions de base classiques (e.g., les polynômes de Laguerre et les fonctions trigonométriques), les fonctions floues de base sont les seules à réunir des informations numériques et linguistiques.

De même que les RNA, les systèmes flous sont des approximateurs de fonctions à succès. Cependant, rien ne peut spécifier le système flou adéquat (e.g., le nombre de règles et le nombre de fonctions d'appartenance), ce qui est aussi valable pour les RNA concernant le nombre de couches et le nombre de neurones par couche.

Remarque [45]

A partir des détails donnés concernant les quatre éléments du système flou, on constate qu'il y a plusieurs possibilités sur lesquelles on peut jouer pour concevoir un système flou. On peut choisir le type de fuzzification (singleton ou nonsingleton), les formes des fonctions d'appartenance (triangulaires, trapézoïdales, Gaussiennes, linéaires par morceaux), les paramètres des fonctions d'appartenance (fixes ou variant dans le temps), le type d'inférence (minimum, produit), le type de composition (max-min, max-produit) et le défuzzificateur (moyenne des maxima, centroïde, centroïde modifié). Ceci mène à 2^{15} = 32768 systèmes flous possibles.

3.7 CONTROLEURS FLOUS

La commande floue (linguistique) s'est révélée, durant ces dernières années, un domaine de recherche actif. Elle est basée sur la logique floue, qui est plus proche du savoir faire humain et du langage naturel. Un contrôleur flou n'est autre qu'un système flou conçu pour commander un processus (fig. 3.9). Il est constitué essentiellement d'un ensemble de règles de commande floues, reliées par des concepts d'implication floue et d'inférence floue. Vu dans cette perspective, le contrôleur flou parait très utile quand le système à commander est complexe, ou quand les informations disponibles sont interprétées qualitativement, ou sont incertaines.

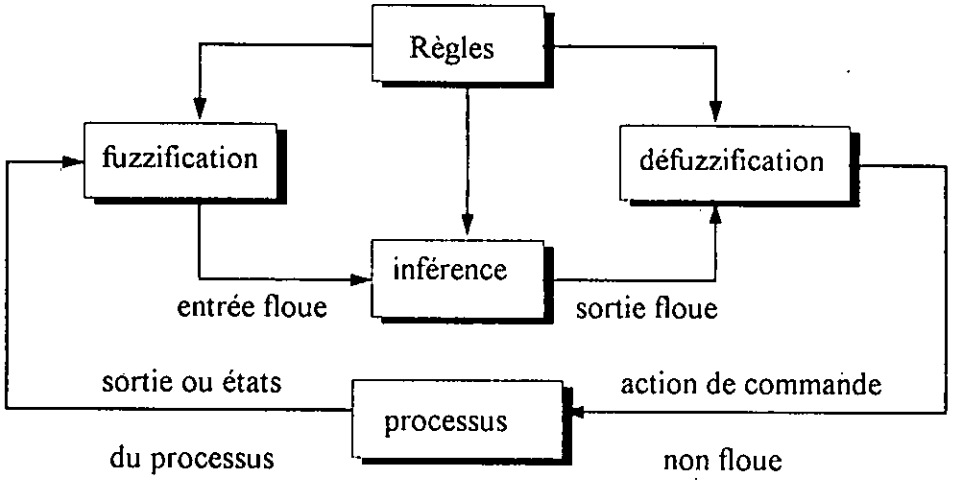


Fig. 3.9 schéma bloc général d'un contrôleur flou.

3.7.1 LES FONCTIONS D'APPARTENANCE

Dans les applications de commande, il existe deux manières pour définir les fonctions d'appartenance, dépendant du domaine de variation des variables linguistiques s'il est discret ou continu. La première est une définition numérique dans laquelle on dresse une table qui délivre la valeur de la fonction d'appartenance de chaque point du domaine discret. La deuxième est une définition fonctionnelle, telle que la fonction Gaussienne :

$$\mu_f(x) = \exp\left[\frac{-(x-u_f)^2}{2\sigma_f{}^2}\right] \tag{3.31}$$

En général, l'intersection entre les fonctions d'appartenance est choisie au crossover point, i.e., 0.5, pour qu'à chaque moment, il existe une seule fonction dominante supérieure à 0.5. De plus, si les données sont trop bruitées, les fonctions d'appartenance doivent être assez larges pour réduire la sensibilité aux bruits [46].

3.7.2 LES REGLES DE COMMANDE FLOUE

Les règles de commande floue sont formulées en termes linguistiques, et sont basées sur le choix des variables d'état du système, ainsi que les variables de commande. L'expérience et la connaissance du problème jouent un rôle important durant ce choix. En particulier, la sélection des variables linguistiques et de leurs fonctions d'appartenance, a un effet considérable sur les performances du contrôleur flou. Généralement, les variables linguistiques utilisées par un contrôleur flou sont les états du système, les erreurs sur les états, la dérivée des erreurs, l'intégrale des erreurs, etc.

Pour établir les règles de commande floue, on peut se baser sur quatre techniques reportées dans la littérature [30][46]. Ces techniques ne sont pas mutuellement exclusives et leur combinaison peut mener à de meilleurs résultats. Ces techniques d'établissement des règles de commande floue sont :

- *Expérience d'un expert et connaissance du problème de commande :* la plupart des contrôleurs flous ont été conçus en faisant appel à l'expérience de l'être humain et à la connaissance du problème par des ingénieurs spécialisés. On peut dire que la commande floue est l'une des premières applications des systèmes experts. Dans certains cas, où un opérateur joue un rôle important dans la commande du processus, il serait intéressant de trouver son savoir faire par des interrogations, et de formuler des protocoles de commande. Cependant, cette méthode présente certains inconvénients. Premièrement, il est souvent possible qu'un opérateur ne peut pas bien expliquer ces secrets linguistiquement. Deuxièmement, il est difficile d'écrire des règles de commande, même basées sur l'expérience, si le processus est complexe. De plus, cette technique est essentiellement heuristique, alors il est difficile de donner une procédure générale de conception.

- *Modélisation des actions de commande d'un opérateur :* dans certains systèmes de commande industriels, où la relation entrée/sortie n'est pas bien connue ou est imprécise, des opérateurs humains entraînés peuvent commander ces systèmes, sans avoir de modèles quantitatifs en mémoire. En effet, l'opérateur humain utilise un ensemble de règles floues pour contrôler le système. Il est donc possible d'exprimer les règles de l'opérateur comme implications floues « si-alors », en employant des variables linguistiques. En pratique, ces règles peuvent être déduites à partir des observations des actions du contrôleur humain, sous forme de données entrées/sorties.

- *Modélisation floue du processus :* la description linguistique des caractéristiques dynamiques du système à commander, peut être vue comme un modèle flou du processus. En utilisant ce modèle flou, on peut générer un ensemble de règles floues pour atteindre certaines performances. Bien que compliquée, cette technique est beaucoup plus rigoureuse pour la conception d'un contrôleur flou; malheureusement, elle n'est pas encore entièrement développée.

- *Apprentissage :* certains contrôleurs flous sont capables de créer leurs règles de commande et de les modifier en se basant sur l'expérience. C'est ce qu'on appelle les contrôleurs à auto-organisation (self-organising controllers). Ce type de contrôleur possède une structure hiérarchique basée sur deux ensembles de règles. Le premier est un ensemble général, et le second est construit en imitant la manière dont l'être humain s'entraîne pour créer et générer l'ensemble général des règles, dans le but de réaliser une performance désirée du système. Un exemple intéressant de ce type de contrôleur est le modèle flou d'une voiture, proposé par Sugeno [54].

3.7.3 STRATEGIES DE FUZZIFICATION

Dans les applications de la commande floue, les grandeurs observées sont nettes (numériques). Le contrôleur flou étant basé sur la théorie des ensembles flous, la fuzzification

est donc nécessaire. L'expérience acquise durant la conception des contrôleurs flous suggère les techniques suivantes pour effectuer la fuzzification :

- En général, le fuzzificateur convertit une valeur nette en un fuzzy singleton. Cette stratégie a été largement utilisée en commande floue, puisqu'elle est intuitive et facile à implémenter. Il s'agit de concevoir un singleton fuzzifier qu'on a présenté plus haut avec les systèmes flous.

- Dans certaines applications, les données observées sont bruitées. Dans ce cas, le fuzzificateur doit convertir les données probabilistiques en nombres flous, et l'utilisation de fonctions d'appartenance s'avère indispensable. L'une des techniques qui existent, consiste à utiliser un triangle isocèle comme fonction de fuzzification. La médiane de ce triangle correspond à la valeur moyenne de la donnée, et sa base est égale à deux fois l'écart type de cette variable. Les autres techniques consistent à choisir des fonctions d'appartenance adéquates telles que la fonction Gaussienne.

- Dans d'autres applications, où il s'agit de contrôler des systèmes dits hybrides, i.e., des systèmes qui ont des grandeurs bruitées et d'autres non bruitées, on peut combiner les deux stratégies citées ci-dessus.

3.7.4 STRATEGIES DE DEFUZZIFICATION

La défuzzification a pour objectif de produire une action de commande non floue, qui représente le mieux la distribution de l'action de commande floue donnée par l'engin d'inférence. Les techniques de défuzzification, déjà présentées avec les systèmes flous, sont celles utilisées dans la conception des contrôleurs flous.

Les travaux de recherche effectués par Rutherford et Braae [47], menés sur les différents défuzzificateurs, ont montré que le défuzzificateur centroïde donne, en général, les meilleurs résultats. Cependant, le défuzzificateur moyenne des maxima donne une performance meilleure en régime transitoire, alors que le défuzzificateur centroïde est meilleur en régime permanent. De plus, le défuzzificateur centroïde fourni une erreur quadratique moyenne inférieure à celle obtenue par le défuzzificateur moyenne des maxima.

3.8 CONCLUSIONS

Durant ce chapitre, consacré à la présentation des systèmes flous et des contrôleurs flous, on a démontré que ces systèmes sont nonlinéaires, reliant des entrées nettes (numériques) à une sortie nette. On a fourni les formules mathématiques qui décrivent ces systèmes, tout en expliquant qu'ils forment un outil efficace pour l'approximation de fonctions, une propriété qu'ils partagent avec les réseaux de neurones artificiels.

Les systèmes flous jouissent de l'avantage de pouvoir manipuler une connaissance linguistique, chose qui n'est pas possible avec les RNA. Cette forme de connaissance est très utile, surtout quand on ne dispose pas de beaucoup de données numériques pour effectuer l'apprentissage. Aussi, ce sont des systèmes riches en possibilités, i.e., on peut décider de plusieurs paramètres pour les concevoir, tels que les fonctions d'appartenance, le type de fuzzification, le type de défuzzification, le type d'inférence, et la relation de composition.

L'idée de la commande floue est caractérisée par une stratégie de commande issue du savoir faire humain qualitatif. Cependant, il n'est pas suffisant, en commande, d'exprimer le protocole qualitativement. On doit exécuter l'idée qualitative dans une situation réelle. Les ensembles flous ont justement pour rôle de réunir une connaissance qualitative et quantitative en utilisant les fonctions d'appartenance. Ainsi, l'exécution de l'ensemble des règles floues peut être réalisée numériquement par la méthode du raisonnement flou. De plus, le format d'une implication étant linguistique, on peut insérer une variété d'idées utiles dans le contrôleur.

En commande floue, on peut facilement concevoir un contrôleur qui opère pour réaliser plusieurs objectifs à la fois, tels que, la stabilité, l'optimisation de l'énergie de l'action de commande, la sécurité, etc. Ceci est accompli par l'établissement d'un certain nombre de règles qui obéissent à un premier critère, et d'autres qui obéissent à un second critère. La coordination des différents objectifs est assurée par le raisonnement flou.

Peut être, le seul inconvénient de la commande floue est le manque d'un cadre théorique et d'outils analytiques qui peuvent optimiser l'architecture du contrôleur flou, car notre sens de raisonnement n'est pas, tout le temps, le bon.

CHAPITRE 4

LINEARISATION NEURONALE ET FLOUE D'UNE ELECTRODE SELECTIVE D'IONS A MEMBRANE DE NASICON

" Quand le seul outil qu'on possède est un marteau, tous nos problèmes doivent ressembler à des clous. "

Source inconnue

4.1 INTRODUCTION

L'opération de mesure joue un rôle capital en faveur du bon fonctionnement des processus physiques. Les activités à caractère industriel ont pu, grâce à elle, évoluer rapidement et le domaine de la robotique en est un véritable témoin. En régulation automatique, une bonne mesure est liée directement à la qualité de réglage qu'on désire obtenir, et par ceci à un contrôle fiable des grandeurs physiques associées au système. Les outils prodigués par l'électronique ont permis à ce que toute grandeur physique qu'on veut évaluer soit traduite en une grandeur électrique, ce qui est du rôle du capteur [56].

Parmi tous les capteurs, se distinguent les capteurs ioniques qui sont largement utilisés dans les processus chimiques, l'industrie alimentaire, la surveillance de l'environnement et les applications biomédicales. L'inconvénient majeur de ce type de capteurs est leur grande sensibilité à la grandeur de perturbation, à savoir la présence dans la solution d'ions interférents qui gênent une bonne estimation de l'ion principal. Par exemple, si on désire mesurer la concentration de l'ion sodium dans le sang, les ions potassium, calcium et autres introduiront une erreur de mesure sur l'ion sodium. De plus, les capteurs ioniques sont fortement nonlinéaires, et jusqu'à présent les méthodes d'interpolation sont les plus utilisées pour corriger ces nonlinéarités. La chute des prix des microprocesseurs a permis le remplacement des techniques de linéarisation analogique par des techniques numériques. Dans ce contexte, les réseaux de neurones artificiels ont été suggérés comme outils de linéarisation de capteurs [58][59][60].

Dans ce chapitre, deux méthodes de linéarisation de capteurs ioniques, avec la prise en considération de la grandeur de perturbation, sont proposées. La première méthode est basée sur la conception d'un réseau de neurones dont le comportement désiré est celui de la caractéristique inverse du capteur. La seconde méthode, quant à elle, consiste à concevoir un système flou qui a le même objectif que le réseau de neurones. Les performances des deux techniques seront présentées et feront l'objet de comparaison.

4.2 NONLINEARITE DES CAPTEURS

Le modèle général du capteur est défini par (fig. 4.1) :

$$y = f(x,g) \tag{4.1}$$

y étant un signal électrique et x la grandeur à mesurer. Dans le cas d'un capteur passif, où il y a variation d'impédance, celle-ci est traduite en signal électrique à l'aide d'un conditionneur [56].

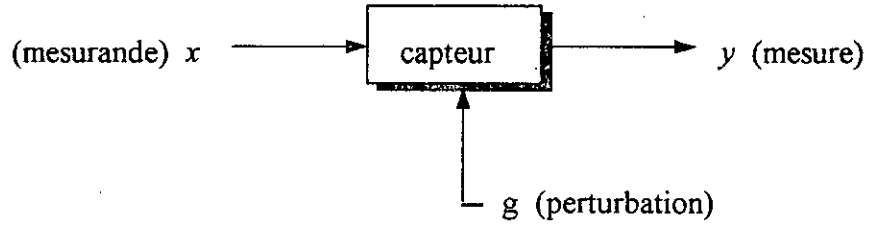


Fig. 4.1 schéma bloc d'un capteur.

La variation de la mesure y est donnée par le développement limité autour du point (x_0,g_0) à l'ordre 2 :

$$\Delta y = \frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial g}\Delta g + \frac{\partial^2 f}{2\partial x^2}(\Delta x)^2 + \frac{\partial^2 f}{2\partial g^2}(\Delta g)^2 + \frac{\partial^2 f}{2\partial x \partial g}\Delta x \Delta g + \frac{\partial^2 f}{2\partial g \partial x}\Delta g \Delta x \qquad (4.2)$$

le terme : $S = \frac{\partial f}{\partial x}$ est appelé sensibilité statique du capteur.

le terme : $S_g = \frac{\partial f}{\partial g}$ est appelé sensibilité au bruit.

le terme : $S_d = \frac{\partial^2 f}{\partial x^2}$ est appelé la dérive de la sensibilité en fonction de x.

Le capteur est dit nonlinéaire si sa sensibilité au mesurande x est dépendante de celui-ci, c-à-d :

$$\frac{\partial^2 f}{\partial x^2} \neq 0$$

En régime dynamique, on dira que le capteur est nonlinéaire, s'il est régit par une équation différentielle nonlinéaire. Dans ce cas, on parlera de sensibilité dynamique, qui dépend de la fréquence du mesurande, ou fonction de transfert. Les exemples de capteurs nonlinéaires sont multiples, on citera les résistances thermiques, les capteurs de débit d'air, les capteurs de composition gazeuse et bien sûr les capteurs ioniques.

4.3 POURQUOI LINEARISER LA CARACTERISTIQUE D'UN CAPTEUR

Telle est la question qu'un utilisateur se pose à chaque fois qu'il doit utiliser un capteur. C'est vrai que tout dépend de l'application à laquelle on veut joindre le capteur. Parfois, on a pas besoin de linéariser pour des causes qu'on juge valables et qui servent à un bon fonctionnement de cet instrument; mais souvent ce n'est pas le cas, et une linéarisation adéquate s'avère indispensable.

Quand la caractéristique du capteur est linéaire, le signal de sortie peut dépasser le champ de visualisation; mais ce problème peut être réglé par simple mise à l'échelle des grandeurs [57]. Seulement, si le capteur est nonlinéaire on se trouve confronté à d'autres genres de problèmes peut être plus difficiles à régler. De plus, la linéarisation a l'avantage d'augmenter la précision des capteurs et leur étendue de mesure.

Généralement, on approxime les parties nonlinéaires de la caractéristique du capteur par des segments de droites autour de points de fonctionnement. Cette méthode a le désavantage d'avoir une étroite étendue de mesure, qui diminue encore plus si le capteur est fortement nonlinéaire. En régulation, la linéarisation d'un bloc nonlinéaire permet de simplifier la synthèse de la loi de commande, en utilisant la théorie des systèmes linéaires.

4.4 METHODES CLASSIQUES DE LINEARISATION DES CAPTEURS [57]

En instrumentation, il existe deux principales méthodes de linéarisation des capteurs nonlinéaires : la *linéarisation analogique* et la *linéarisation numérique*.

- La première méthode utilise des circuits analogiques nonlinéaires placés après le conditionneur du capteur. Ces circuits ont pour entrée le signal de mesure du capteur et produisent en sortie un signal linéaire en fonction du mesurande. Ils peuvent être conçus à base d'éléments électroniques nonlinéaires comme les diodes et les transistors [70], dans le but d'approximer la fonction inverse de la caractéristique du capteur, obtenue par étalonnage. Cette méthode est utilisée quand il s'agit de concevoir des circuits linéarisants à faible coût et à grande vitesse de traitement. Néanmoins, elle est limitée par le fait qu'elle n'est opérationnelle que si le signal de mesure dépend uniquement du mesurande, ou quand les grandeurs de perturbation sont invariables dans le temps. Autrement dit, si le capteur a une grande sensibilité aux bruits, il sera difficile de le linéariser par un circuit analogique.

- La deuxième méthode est basée sur l'implémentation d'un programme dans une mémoire ROM sous forme d'une table de mesures prises sur le capteur, qui permet de nous délivrer la valeur du mesurande à chaque fois qu'on lui donne la valeur de la mesure . Ceci étant, le signal de sortie du capteur doit être conditionné et digitalisé à travers un conditionneur A/D. L'avantage de cette méthode n'est pas dérisoire, puisqu'elle permet de linéariser tout en tenant compte de l'influence des perturbations, chose qui n'est pas possible avec le linéarisateur analogique. De cette façon, les mesures deviennent, sans aucun doute, plus précises. Le principe de la méthode consiste, tout simplement, à opérer une interpolation linéaire ou quadratique à partir des données du tableau contenant les entrées / sorties du capteur.

4.5 ELECTRODES SELECTIVES D'IONS (ISE)

4.5.1 PRESENTATION

Les électrodes sélectives d'ions (Ion Selective Electrodes) sont des capteurs chimiques basés sur la potentiométrie, une méthode électrochimique qui consiste à mesurer la différence de potentiel entre deux électrodes plongées dans une solution [55]. Ces capteurs sont fonction de plusieurs paramètres, nous citerons : la sensibilité de la membrane, la référence interne du capteur, la concentration de l'ion à mesurer, la pureté de la solution à analyser, la température, etc...

En potentiométrie, on distingue en général trois sortes d'électrodes [65] :

- les électrodes de mesure du pH,
- les électrodes de mesure d'oxydo-réduction (exemple : électrode de platine),
- les électrodes "spécifiques" utilisant des membranes sélectives.

Les membranes sélectives ont été introduites après que certains verres destinés aux mesures du pH n'aient pas donné satisfaction [65]. Ces verres ont présenté une sensibilité considérable aux ions alcalins tels que Na^+, K^+, Li^+. Quelques chercheurs ont mis à profit cette propriété pour la réalisation d'électrodes spécifiques aux ions Na^+, K^+, etc... La sensibilité à l'ion à

mesurer est fournie par un échange direct d'ions, entre la membrane de verre et la solution analysée.

4.5.2 POURQUOI LA MEMBRANE DE NASICON

Les électrodes de verre présentent un certain défaut de fragilité et ne possèdent pas les commodités de fabrication désirées [62]. De plus, elles sont instables dans un milieu humide, et ne peuvent pas être utilisées dans des solutions aqueuses [66]. Ceci a poussé les chercheurs à remplacer ces membranes de verre par une autre membrane robuste, plus sélective, facile à miniaturiser et adaptable dans divers domaines d'application tels que le biomédical, l'industrie chimique et les mesures dans les milieux exposés aux chocs. Cette membrane est constituée d'une céramique appelée le NASICON (Sodium Super Ionic Conductor), qui a pour formule $Na_{1+x}\ Zr_2\ Si_x\ P_{3-x}\ O_{12}$, avec $0 \leq x \leq 3$, il présente donc une composition assez large. Il a été synthétisé pour la première fois par Hong en 1976 [67].

L'avantage de ce matériau est sa structure rigide à sites de conduction tridimensionnels [69]. C'est une céramique qui se fritte à une température pas trop élevée (1000° à 1200°C) et possède une bonne stabilité à l'air humide ou dans une solution aqueuse. De plus, on a trouvé que c'est un très bon conducteur ionique à 25°C ($\sigma \approx 10^{-3}$ S/cm pour $1.8 \leq x \leq 2.2$). Le choix du NASICON comme membrane sensitive est aussi justifié par le fait qu'il présente une flexibilité quant à la fabrication des ISE, puisque c'est une céramique qui peut être usinée facilement, sa grande conductivité ne nécessite pas l'utilisation de membranes fines, et donc toutes les géométries sont possibles. On peut ajouter à cela que la sensibilité de cette membrane aux ions alcalins tels que K^+, Li^+ est minime relativement aux membranes de verre.

Concernant le régime dynamique des capteurs à membrane de NASICON, on a remarqué que le temps de réponse du capteur pour une variation de concentration du sodium de 0.1 à 1 mol/litre était inférieur à 20 ms [66], donc une réponse assez rapide.

Un exemple de capteur à membrane de NASICON est présenté dans la figure (4.2).

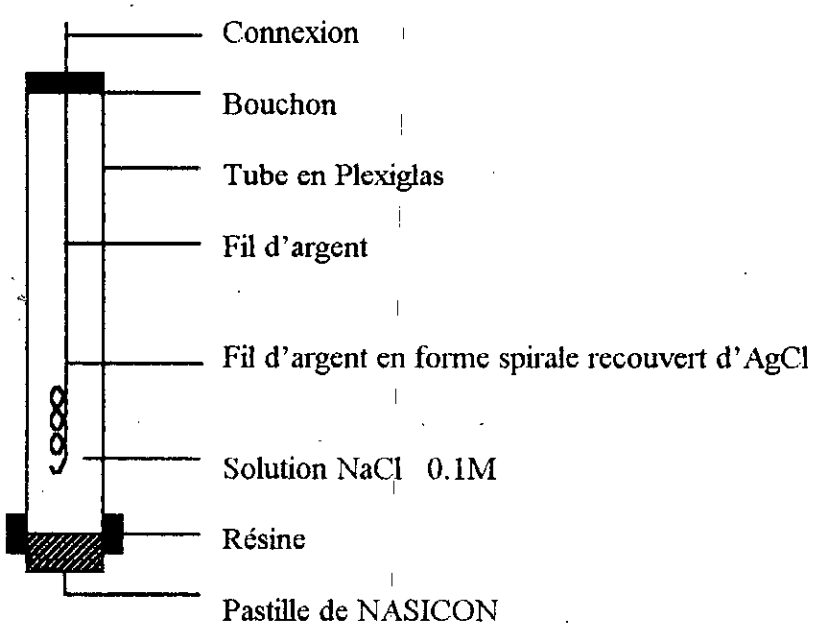


Fig. 4.2 exemple d'un capteur à membrane de NASICON.

4.5.3 MODELISATION

Une électrode sélective d'ion délivre à sa sortie une f.e.m en réponse à une concentration d'un type d'ion spécifique. Cette f.e.m est donnée par la **relation de Nernst**, et est mesurée par rapport à une tension référence E_0 associée à la solution interne du capteur :

$$E = E_0 + S\,Log\,A \tag{4.3}$$

A : est l'activité de l'ion à mesurer,

S : est la pente de la réponse de l'électrode. Théoriquement, elle est calculée par :

$$S = 2.3\,\frac{RT}{ZF} \tag{4.4}$$

R : est la constante de Bolzmann, $R = 8.314$ J/mol.K

T : est la température absolue,

Z : est le nombre de charge de l'ion à mesurer,

F : est la constante de Faraday, $F = 96500$ C

Il n'existe point d'ISE qui répond exclusivement à l'ion spécifié. La présence d'autres ions dans la solution peut affecter sérieusement les performances de celle-ci. L'interférence entre l'ion principal et les ions secondaires peut prendre plusieurs formes dépendant du type de membrane utilisée. Dans ce cas, le comportement de l'électrode est régit par une équation, utilisée pour la première fois par Nickolskii pour l'électrode de verre, et plus connue maintenant sous le nom de **relation de Nickolskii-Eisenmann** qui est la suivante :

$$E = E_0 + S\,Log\left(a_i + K_{ij}\,a_j^{Z_i/Z_j}\right) \tag{4.5}$$

où Z_i et Z_j sont les nombres de charge de l'ion principal i et l'ion interférent j, respectivement. a_i et a_j sont les activités respectives des ions i et j. K_{ij} est appelé coefficient de sélectivité potentiométrique qui est une mesure de l'aptitude de l'électrode à distinguer l'ion primaire i de l'ion interférent j. L'activité d'un ion i dans une solution est relié à sa concentration C_i par la relation :

$$a_i = \gamma_i C_i \tag{4.6}$$

où γ_i est le coefficient d'activité, il dépend des ions présents dans la solution.

La figure (4.3) illustre le principe de mesure d'une électrode sélective d'ion.

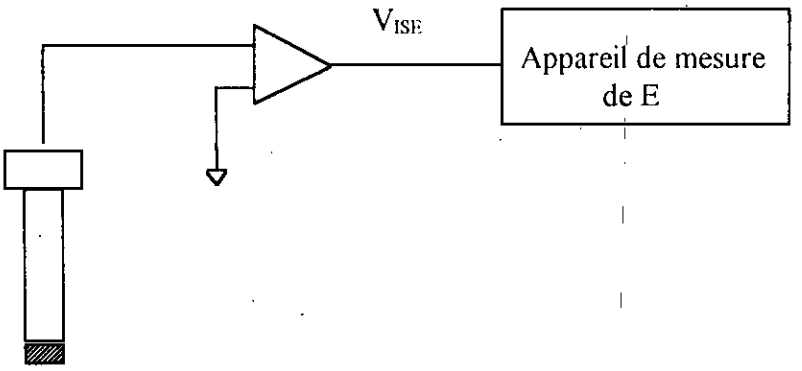


Fig. 4.3 schéma illustrant le principe de mesure par une ISE

4.6 LINEARISATION NEURONALE DE L'ISE A MEMBRANE DE NASICON

4.6.1 PRINCIPE DE FONCTIONNEMENT

Comme dans le cas de la linéarisation analogique, le RNA doit simuler la caractéristique inverse de l'ISE, qui est nonlinéaire. L'assemblage en cascade du capteur et du RNA, après apprentissage, se comportera comme un bloc linéaire (Fig. 4.4). Le signal numérique à la sortie de l'ADC est traité par le réseau neuronal implanté dans une EPROM qui travaille en simultanée avec un micro-processeur pour délivrer un signal proportionnel au mesurande.

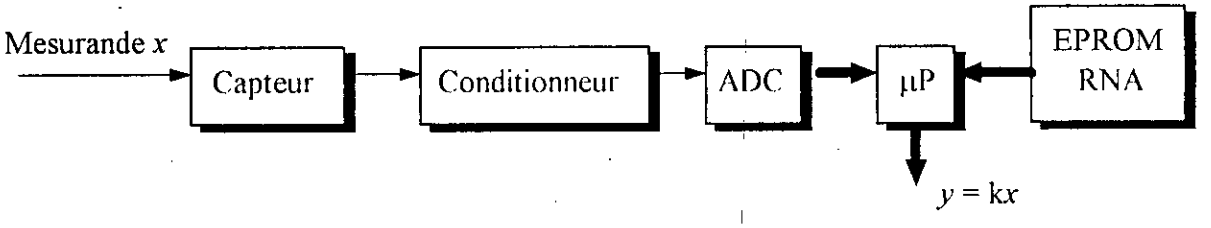


Fig. 4.4 insertion d'un linéarisateur neuronal dans une chaîne instrumentale.

Le rôle du RNA étant de reproduire la caractéristique statique inverse du capteur, il doit alors avoir pour exemples d'apprentissage, des couples (entrée, sortie désirée) pratiques, c-à-d, issus de la caractéristique du capteur obtenue par des mesures réelles.

Concernant le capteur ionique, notre travail va consister à entraîner un RNA pour simuler la caractéristique inverse de l'électrode dans deux cas distincts, à savoir :

- la conception d'un linéarisateur neuronal en l'absence d'ions interférents (cas de la relation de Nernst). A ce moment, le réseau possède une seule entrée et une seule sortie (Fig. 4.5).

- La conception d'un linéarisateur neuronal en présence d'ions interférents (cas de la relation de Nickolskii-Eisenmann). Dans ce cas, le réseau possède deux entrées et une seule sortie (Fig. 4.6).

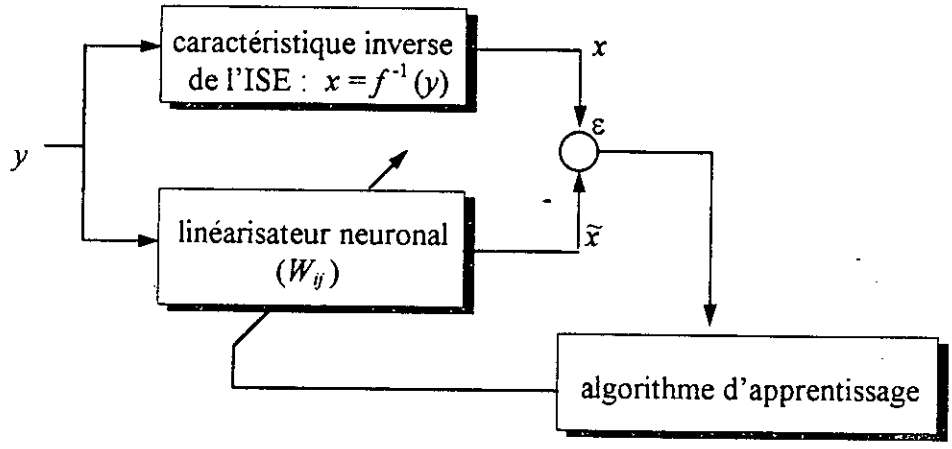


Fig. 4.5 apprentissage du linéarisateur neuronal en l'absence d'ions interférents.

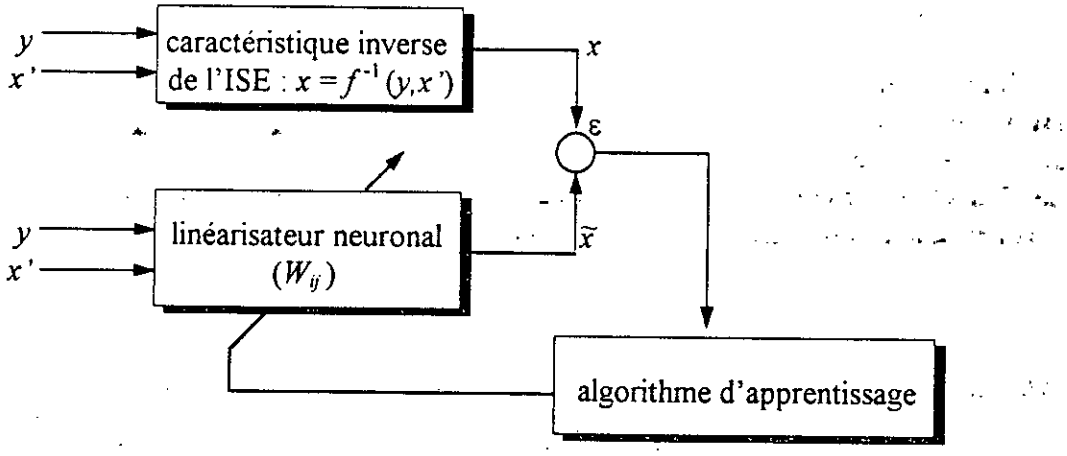


Fig. 4.6 apprentissage du linéarisateur neuronal en présence d'un ion interférent.

4.6.2 LINEARISATION DE L'ISE EN L'ABSENCE D'IONS INTERFERENTS

La caractéristique inverse de l'électrode est, dans ce cas, la fonction inverse de la relation de Nernst. Cette électrode à membrane de NASICON est, bien sûr, conçue pour la mesure de la concentration de l'ion Na^+, on a donc :

$$E = cste + S\,Log(a_{Na^+}) \tag{4.7}$$

la nonlinéarité de cette relation réside dans la partie logarithmique. On se propose donc, de simuler par RNA la fonction inverse de cette nonlinéarité, à savoir :

$$a_{Na^+} = 10^{Log(a_{Na^+})} \tag{4.8}$$

ou alors :

$$a_{Na^+} = 10^{\left(\frac{E-cste}{S}\right)} \tag{4.9}$$

Ceci étant, on a opté pour un réseau neuronal siso à deux couches cachées dont chacune comporte 5 neurones nonlinéaires ayant la tangente hyperbolique pour fonction d'activation (Fig. 4.7).

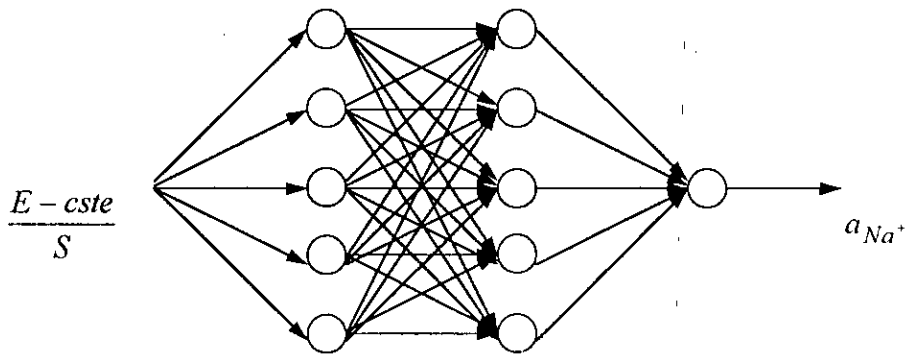


Fig. 4.7 structure du réseau linéarisateur dans le cas de la relation de Nernst.

Remarque

L'avantage de cette structure du réseau est sa capacité à prendre en charge la grandeur de perturbation qu'est la température. En effet, l'entrée du réseau est fonction du coefficient S qui lui est fonction de la température T (voir l'équation 4.4). L'utilisation d'un capteur de température adéquat (une thermistance par exemple), nous permettra de linéariser la caractéristique de l'électrode en question, sur une large plage de température. Dans notre cas, la plage de température envisagée va de 0° à 50°C.

A. Apprentissage

Le RNA a été entraîné pour reproduire la relation nonlinéaire (4.9) dans un intervalle de variation de la concentration de l'ion Na^+ allant de 10^{-3} m/l à 1m/l. Les données d'entraînement ont été choisies par discrétisation, avec un pas constant, du domaine de variation de $[Na^+]$. Ceci permettra au réseau de bien assimiler les différentes parties de la caractéristique. L'algorithme d'apprentissage utilisé est la rétropropagation classique avec un pas du gradient variable. L'entraînement a duré prés de 50000 itérations, pour aboutir à une erreur quadratique moyenne de l'ordre de 10^{-4}.

Les résultats d'apprentissage sont illustrés graphiquement, dans la figure (4.8) sous forme d'erreurs relatives commises sur les exemples appris. De là, on peut remarquer que ces erreurs sont en majorité inférieures à 5%.

B. Généralisation

Pour valider les résultats obtenus par le RNA durant l'apprentissage, des tests ont été effectués sur des données non apprises, et cela en prenant en compte la variation de la

température. La figure (4.9) nous présente les résultats délivrés par le réseau pour deux températures différentes (T = 288K et T = 308K). Le RNA a présenté un comportement similaire à celui de la phase d'entraînement, à savoir des erreurs relatives en sortie inférieures à 5% pour la plupart des données.

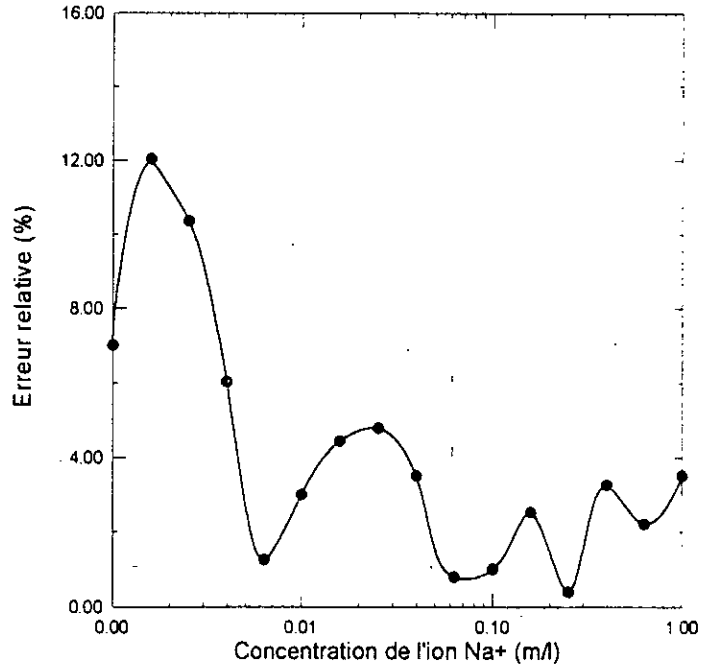


Fig. 4.8 résultats d'apprentissage du RNA pour l'approximation de la relation inverse de Nernst.

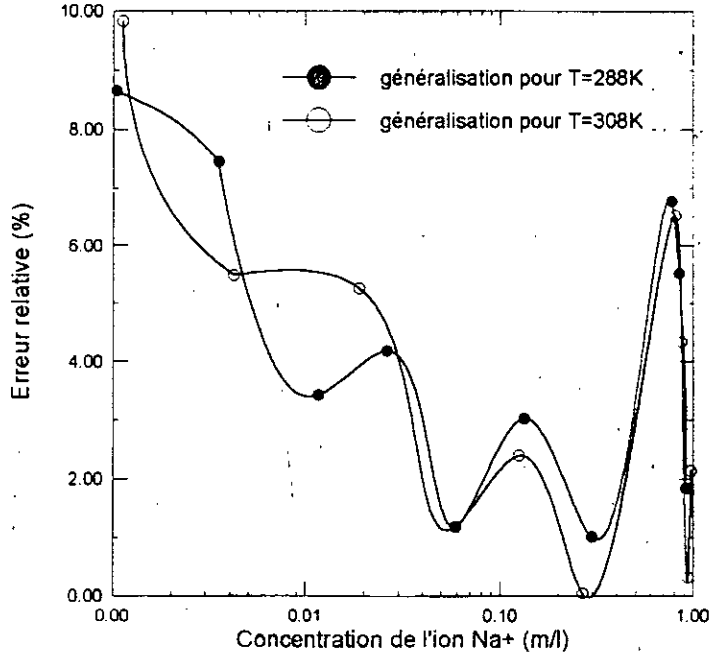


Fig. 4.9 résultats de généralisation du RNA pour l'approximation de la relation inverse de Nernst.

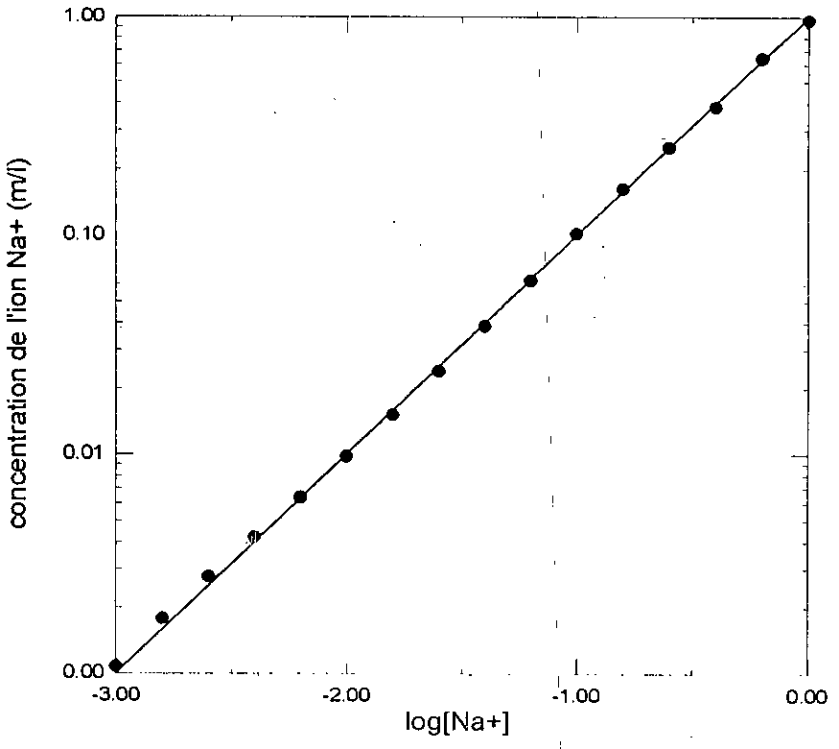


Fig. 4.10 approximation de la relation nonlinéaire : $[Na^+] = f(\text{Log}[Na^+])$ par le RNA.

Pour bien apprécier la qualité de la linéarisation neuronale, la figure (4.10) nous montre l'approximation de la relation inverse de Nernst : $[Na^+] = f(\text{Log}[Na^+])$ par le RNA de la figure (4.7).

C. Comparaison avec les linéarisations numériques du premier et second ordre

Les approximations numériques du premier et second ordre appliquées à la relation inverse de Nernst nous ont permis de situer la qualité de l'approximation obtenue par réseau de neurones (Fig. 4.11). On remarque que l'approximation numérique du premier ordre donne des résultats très satisfaisants, avec des erreurs relatives qui ne dépassent pas les 3% en grande partie. Néanmoins, on note au passage la contre performance obtenue par l'approximation numérique du second ordre, avec des erreurs relatives voisines des 36%. Ceci peut être expliqué par le faite que la caractéristique, régit par la relation de Nernst, est quasiment linéaire entre les points de discrétisation; une approximation linéaire est donc meilleure qu'une approximation quadratique. Pour ce qui est de l'approximation neuronale, on peut distinguer qu'elle est proche de l'approximation numérique du premier ordre, et donc comparable à celle-ci.

Pour que la comparaison soit objective, les trois approximations ont été soumises aux mêmes exemples.

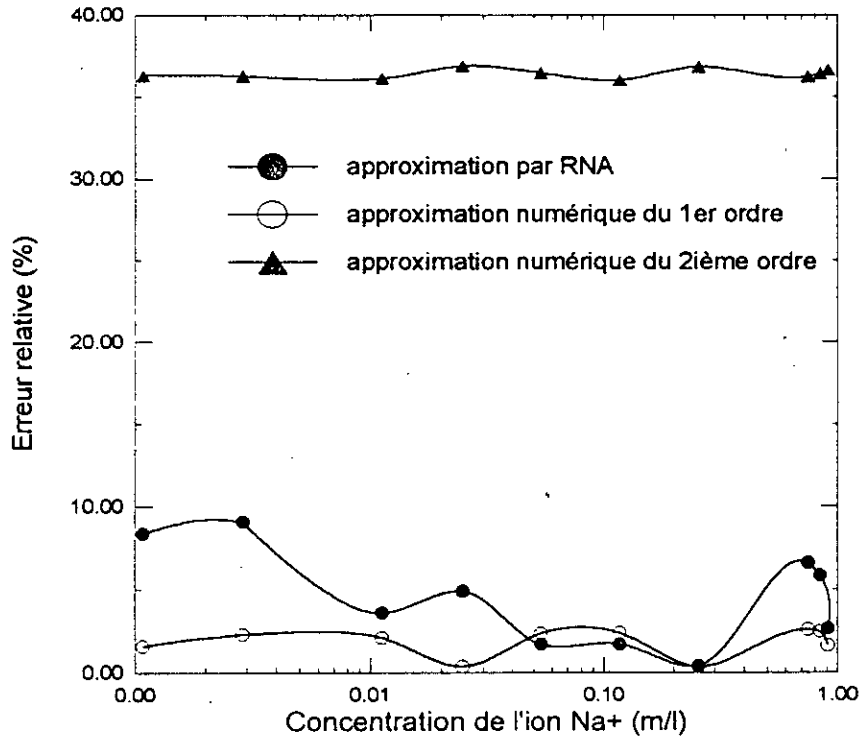


Fig. 4.11 comparaison des résultats des approximations numériques du premier et second ordre avec les résultats obtenus par le RNA.

4.6.3 LINEARISATION DE L'ISE EN PRESENCE DE L'ION INTERFERENT K^+

Comme on l'a mentionné plus haut, toute électrode, et particulièrement l'électrode à membrane de NASICON, est sensible à d'autres types d'ions tels que l'ion potassium K^+. Dans ce cas, le potentiel de l'ISE est donné par l'équation de Nickolskii-Eisenmann suivante :

$$E = cste + S\,Log\left(a_{Na^+} + k_p\,a_{K^+}\right) \tag{4.10}$$

de même que dans le cas de la relation de Nernst, la nonlinéarité de la relation (4.10) se situe dans la partie logarithmique. La caractéristique nonlinéaire inverse que le RNA doit approximer est, alors :

$$a_{Na^+} = 10^{\left(\frac{E-cste}{S}\right)} - k_P\,a_{K^+} \tag{4.11}$$

Le coefficient de sélectivité k_P est une grandeur intrinsèque à l'électrode, cela veut dire que chaque électrode possède son propre coefficient de sélectivité. La mesure de ce coefficient a suscité beaucoup de travaux de recherche [63][64], et il existe aujourd'hui plusieurs méthodes pour évaluer ce facteur potentiométrique, dont il n'y aura pas de suite ici, vu que ce n'est pas

l'objectif de notre travail. Ces méthodes ont abouti au fait que ce coefficient dépend de plusieurs paramètres, notamment des concentrations respectives de l'ion principal et de l'ion interférent. Concernant la membrane de NASICON, les travaux menés par K. Hafit [65], au laboratoire d'ionique de l'Ecole Polytechnique de Grenoble, ont démontré que pour une concentration variable de l'ion principal Na^+ et une concentration fixe de l'ion interférent K^+, le coefficient de sélectivité k_P est variable.

Alors, pour bien cerner le problème de linéarisation, il fallait prendre ce phénomène en considération. Ceci étant, la suite du travail a été divisée en deux parties : une linéarisation dans le cas de k_P variable et la concentration de K^+ fixe en un premier temps, puis une linéarisation dans le cas de k_P fixe et la concentration de K^+ variable en un second temps.

4.6.3.1 Linéarisation dans le cas de k_P variable et $[K^+]$ fixe

Les données sur les quelles on a travaillé sont des données réelles [65]. Dans ce cas, pour une concentration de l'ion interférent K^+ égale à 1m/l, et une concentration de Na^+ variant de 5×10^{-4} à 5×10^{-1} m/l, les différentes valeurs du coefficient potentiométrique k_P sont reportées dans le tableau (4.1) pour une membrane de NASICON (x=2) frittée à 1200°C.

$[Na^+]$ (m/l)	$[5\times10^{-4}, 5\times10^{-3}[$	$[5\times10^{-3}, 5\times10^{-2}[$	$[5\times10^{-2}, 5\times10^{-1}]$
k_P	8×10^{-3}	2×10^{-2}	6×10^{-2}

Tab. 4.1 valeurs de k_P en fonction de $[Na^+]$, avec $[K^+]$=1m/l.

Dans ces conditions, le RNA doit simuler la caractéristique donnée par la relation (4.11). Cette fois ci, on a opté pour un réseau à une seule entrée et une seule sortie, identique à celui de la figure (4.7). la différence réside dans le fait qu'on a partagé l'intervalle de variation de $[Na^+]$ en trois sous intervalles comme c'est le cas dans le tableau (4.1). le réseau neuronal doit être entraîné dans chaque sous intervalle à part. Il aura ainsi, trois matrices de pondération distinctes. Si l'on veut, c'est un réseau à structure variable. Pour choisir quelle structure utiliser, il suffit de savoir dans quelle plage de mesure on se situe (Fig. 4.12).

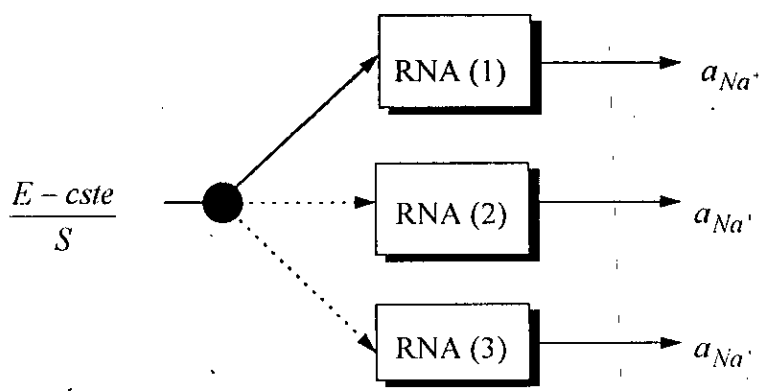


Fig. 4.12 principe de la linéarisation neuronale dans le cas de k_P variable.

A. Apprentissage

Les données d'apprentissage sont générées par la relation (4.11), que le RNA doit approximer. La concentration de l'ion interférent K^+ est constante et égale à 1m/l, tandis que la concentration de l'ion principal Na^+ varie de 5×10^{-4} à 5×10^{-1} m/l. Comme on l'a déjà mentionné, le réseau neuronal est entraîné dans trois sous intervalles distincts de $[Na^+]$, relatifs aux trois valeurs du coefficient de sélectivité k_P. Le premier sous intervalle allant de 5×10^{-4} à 5×10^{-3} m/l, le second allant de 5×10^{-3} à 5×10^{-2} m/l, et le troisième allant de 5×10^{-2} à 5×10^{-1} m/l. A la fin de l'apprentissage de chaque sous intervalle, on garde les pondérations du RNA relatives à cette étape, et on entame un nouvel entraînement avec le sous intervalle suivant.

L'algorithme utilisé est la rétropropagation classique avec un pas de gradient variable. L'entraînement a duré prés de 25000 itérations dans chacune des trois étapes. L'erreur quadratique moyenne s'est stabilisée aux alentours de 10^{-4}. Les résultats d'apprentissage sont illustrés dans le graphe de la figure (4.13). On a remarqué que le réseau a trouvé quelques difficultés à apprendre la partie de la caractéristique relative au deuxième sous intervalle, présentant des erreurs relatives inférieures à 10% en majorité, sauf pour certains exemples, où on a des erreurs proches de 20%. Concernant les deux autres sous intervalles, le RNA a présenté un bon comportement, avec des erreurs relatives inférieures à 2.6% pour la première plage, et inférieures à 5% pour la troisième plage de variation.

La figure (4.14) montre, à son tour, la qualité de l'approximation de la relation : $[Na^+] = f(\mathrm{Log}([Na^+] + k_P.\,[K^+]))$.

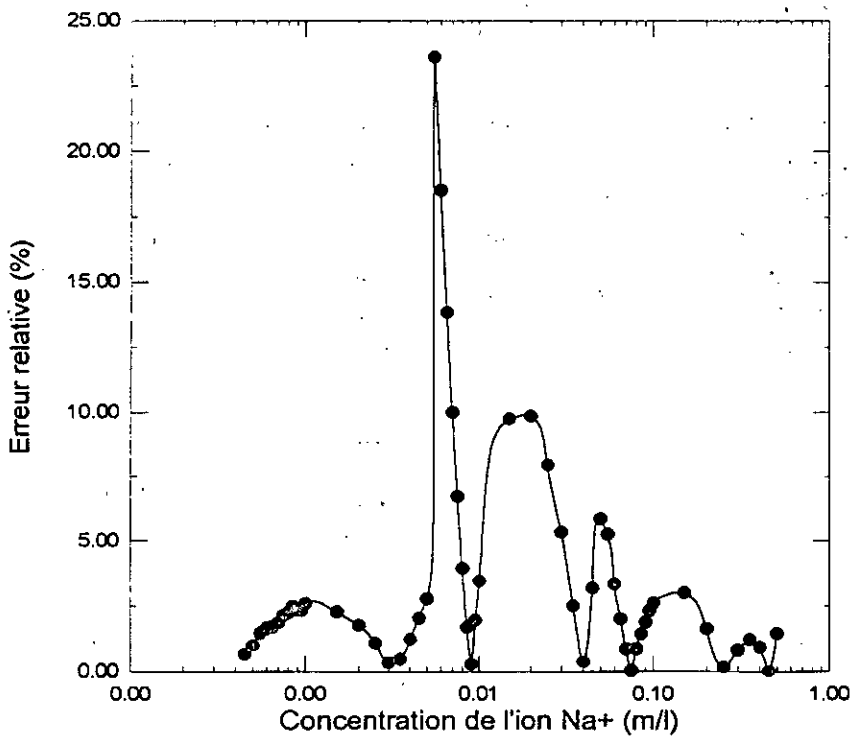


Fig. 4.13 résultats d'apprentissage du RNA pour l'approximation de la relation de Nickolskii-Eisenmann dans le cas de k_P variable.

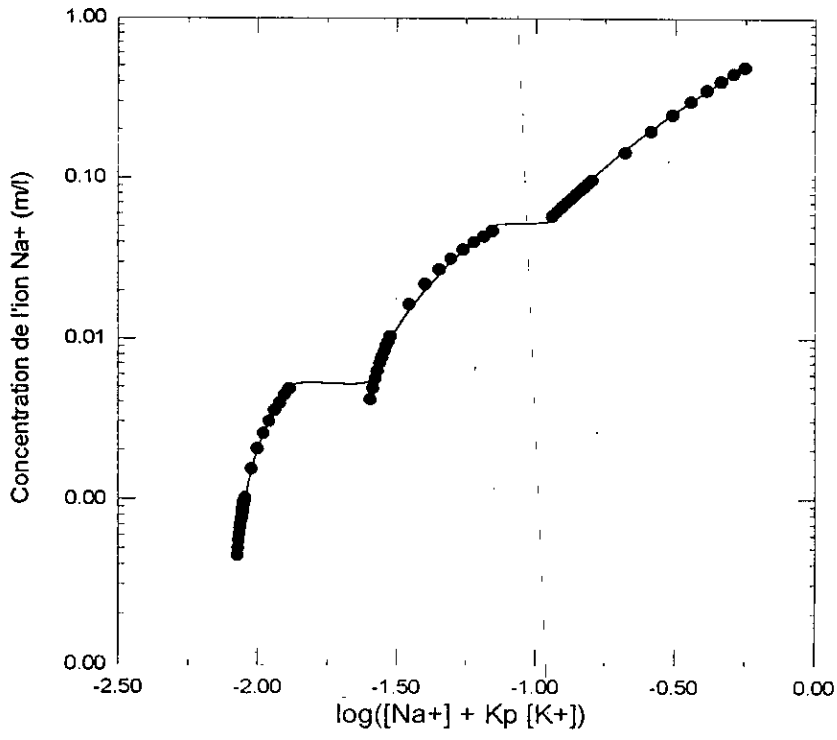


Fig. 4.14 approximation de la relation nonlinéaire : $[Na^+] = f(\text{Log}([Na^+] + k_P\,[K^+]))$ par le RNA.

B. Généralisation

Après la phase d'apprentissage, le RNA a été testé avec des exemples non appris, et ses résultats ont été comparés avec ceux des approximations numériques du premier et second ordre (Fig. 4.15). les meilleurs résultats furent donnés par l'approximation numérique du premier ordre, avec des erreurs relatives ne dépassant pas les 2%. Le RNA a fourni des résultats appréciables avec des erreurs relatives inférieures à 5% pour la plupart des exemples, et souvent très proches des résultats de l'approximation numérique du premier ordre. L'approximation numérique du second ordre , quant à elle, a présenté des erreurs relatives plus élevées, inférieures à 10%, mais allant jusqu'à 30% pour quelques exemples.

4.6.3.2 Linéarisation dans le cas de k_P fixe et $[K^+]$ variable

C'est au fait le cas le plus répondu, puisqu'en réalité on a affaire à des mesures dans des solutions où l'ion interférent est de concentration variable. La consultation des travaux de K. Hafit [65], nous a permis d'effectuer nos simulations en utilisant des données réelles, aux quelles il a abouti au cours de ses essais. En effet, pour une membrane de NASICON (x=2) frittée à 1200°C, le coefficient de sélectivité k_P est égal à 1.58×10^{-3}, pour $[K^+]$ variant de 10^{-4} à 1m/l.

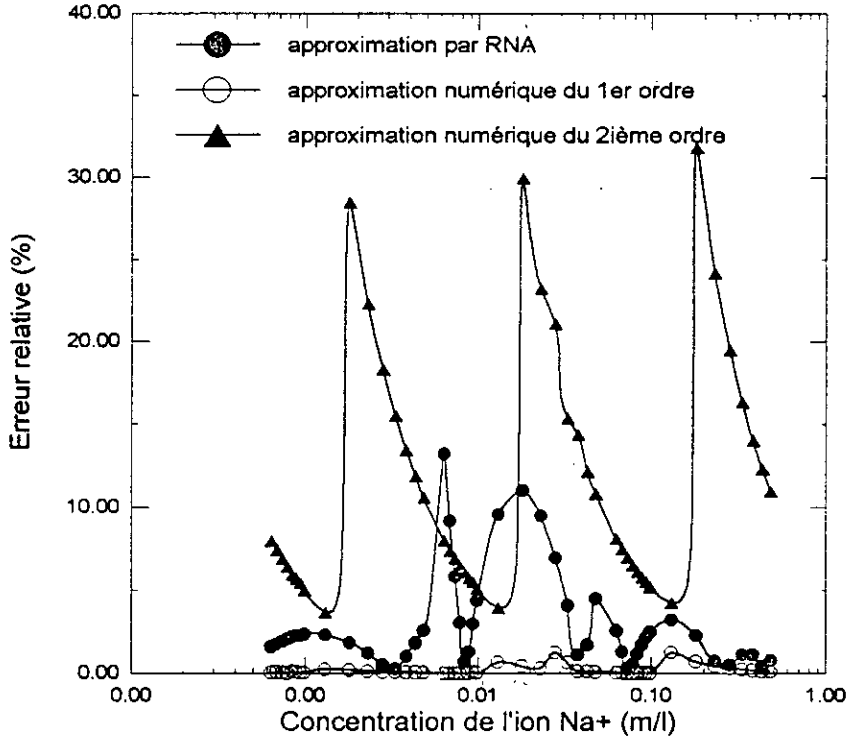


Fig. 4.15 comparaison des résultats des approximations numériques du premier et second ordre avec les résultats de généralisation du RNA dans le cas de k_P variable.

Dans ce cas, le RNA a toujours pour rôle d'approximer la relation inverse de Nickolskii-Eisenmann (4.11), mais cette fois-ci le linéarisateur neuronal est un réseau à deux entrées à savoir, $\frac{E - cste}{S}$ et a_{K^+}, et à une seule sortie a_{Na^+}. La structure du RNA choisie est de deux couches cachées à 10 neurones chacune, ayant la tangente hyperbolique pour fonction d'activation (Fig. 4.16).

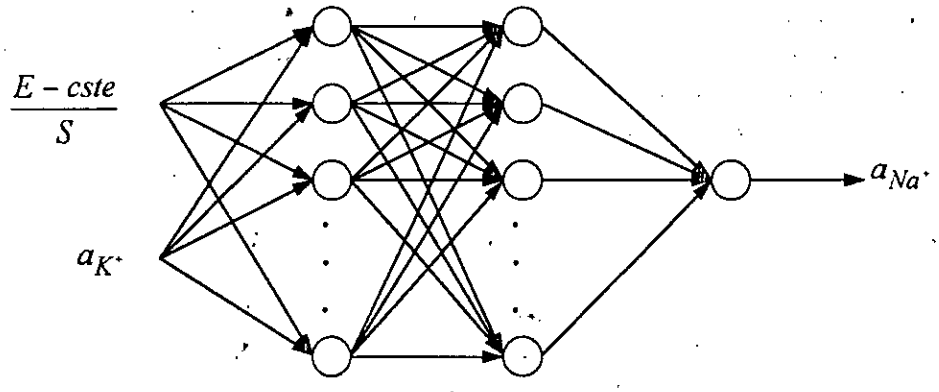


Fig. 4.16 structure du réseau linéarisateur dans le cas de $[K^+]$ variable.

A. Apprentissage

Les exemples d'apprentissage sont générés à l'aide de la relation (4.11). Les intervalles de variation des concentrations des ions : principal Na^+ et interférent K^+ sont de 5×10^{-3} à 1m/l et

de 10^{-4} à 10^{-1} m/l, respectivement. L'algorithme d'apprentissage utilisé est la rétropropagation classique avec un pas de gradient variable.

L'apprentissage a duré prés de 50000 itérations, aboutissant à une erreur quadratique moyenne d'environ 10^{-4}. La figure (4.17) nous offre une idée claire sur la qualité des résultats d'apprentissage, obtenus par le RNA. En effet, on peut remarquer que les erreurs relatives commises sur les exemples d'entraînement sont inférieures à 10%, dans l'intervalle de $[Na^+]$ allant de 5×10^{-3} à 10^{-2} m/l, et inférieures à 4%, dans l'intervalle de $[Na^+]$ allant de 10^{-2} à 1m/l. De plus, on peut distinguer aisément que ces erreurs relatives sont quasiment identiques, quand la concentration de K^+ varie de 10^{-4} à 10^{-1} m/l. Le RNA est presque insensible à la variation de la concentration de l'ion interférent. Le tableau (4.2) met en évidence ce qu'on vient d'affirmer, puisque la moyenne des erreurs relatives est presque constante pour différentes valeurs de $[K^+]$.

$[K^+]$ (mol/l)	moyenne des erreurs relatives (%)
10^{-4}	3.280
10^{-3}	3.286
10^{-2}	3.360
10^{-1}	4.00

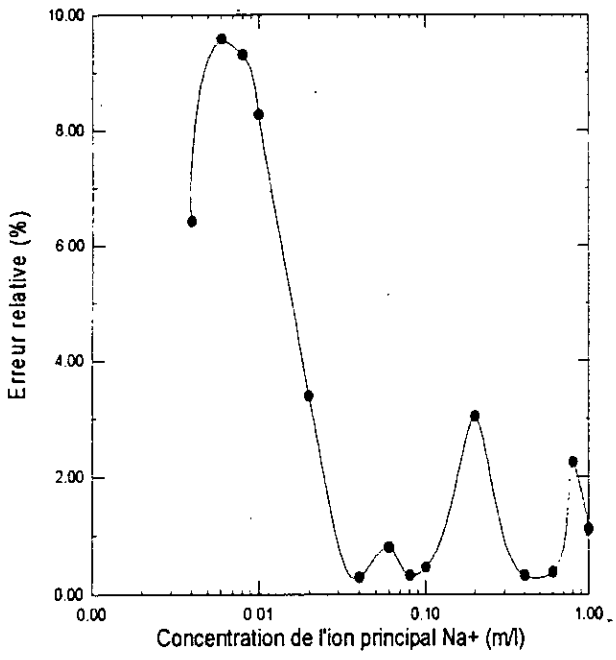
Tab. 4.2 moyenne des erreurs relatives obtenues par le RNA dans le cas de $[K^+]$ variable.

La figure (4.18) montre la qualité de la linéarisation neuronale de la relation de Nickolskii-Eisenmann : $[Na^+] = f(\mathrm{Log}([Na^+] + k_P [K^+]))$, dans le cas de $[K^+] = 10^{-1}$ m/l.

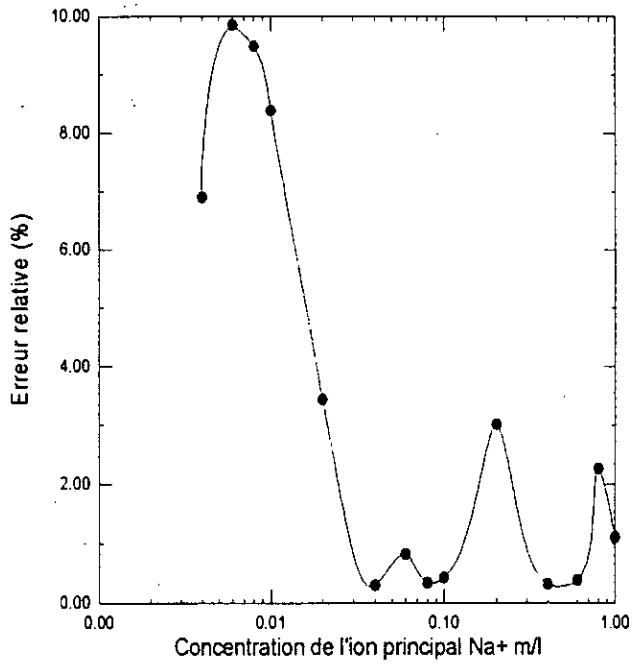
B. Généralisation

De même que les deux étapes précédentes, le RNA a été testé avec des exemples non appris, générés par la relation (4.11). Dans les mêmes conditions (mêmes exemples), ont été testées les approximations numériques du premier et second ordre. La figure (4.19) nous permet de comparer, graphiquement, les résultats des trois méthodes dans le cas de $[K^+]$ variable.

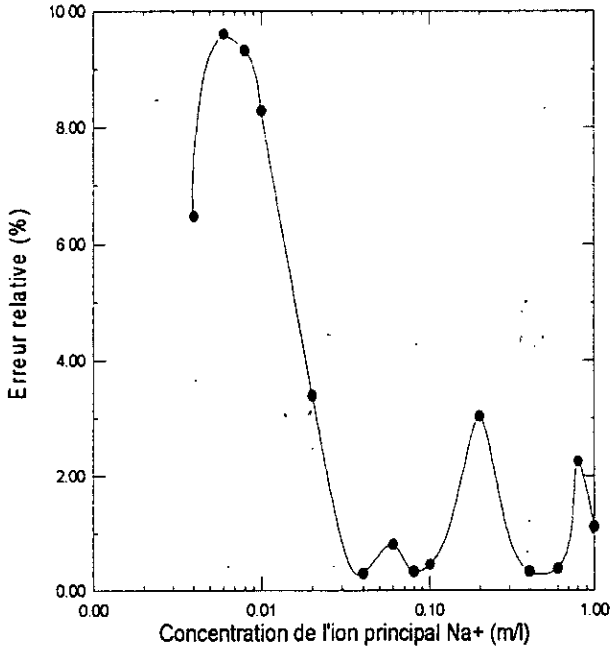
Les erreurs relatives commises par l'approximation numérique du premier ordre sont, globalement, inférieures à 6%. Les erreurs du RNA, quant à eux, sont en majorité inférieures à 3% et sont, même, meilleures que celles de l'approximation numérique du premier ordre dans l'intervalle de variation de $[Na^+]$ allant de 2×10^{-2} à 1m/l. Les résultats fournis par l'approximation numérique du second ordre sont moins performants, les erreurs oscillent autour de 40%.



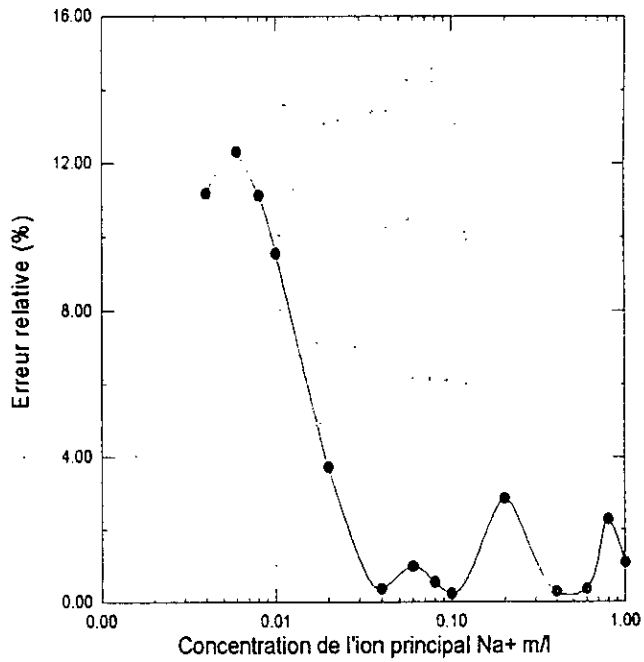
(a)



(c)



(b)



(d)

Fig. 4.17 résultats d'apprentissage du RNA dans le cas de $[K^+]$ variable :
(a) $[K^+]=10^{-4}$ m/l, (b) $[K^+]=10^{-3}$ m/l, (c) $[K^+]=10^{-2}$ m/l, (d) $[K^+]=10^{-1}$ m/l.

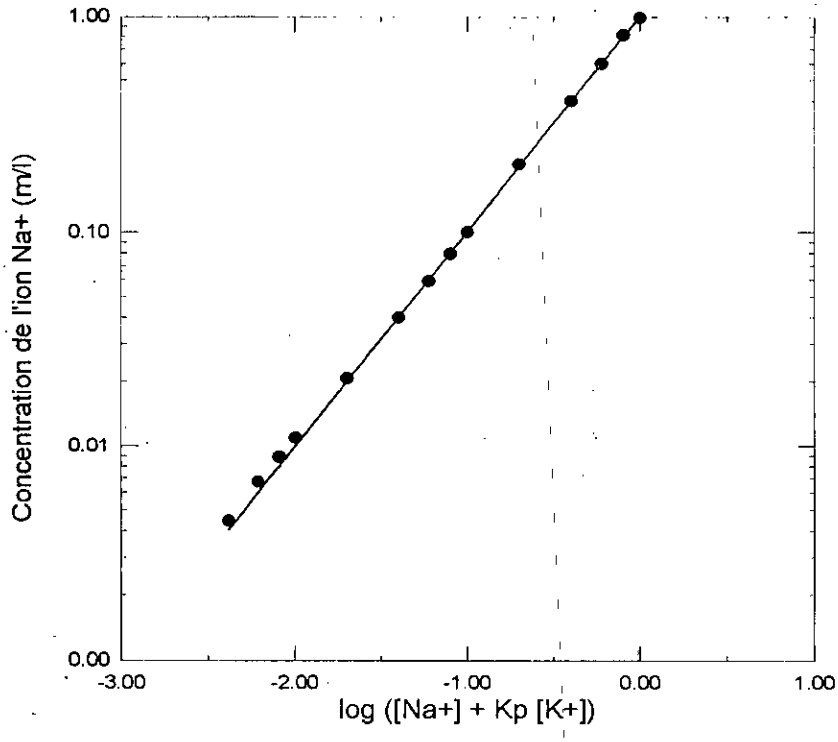


Fig. 4.18 approximation de la relation nonlinéaire : $[Na^+] = f(\text{Log}([Na^+] + k_P\,[K^+]))$ par le RNA, cas de $[K^+] = 10^{-1}$ m/l.

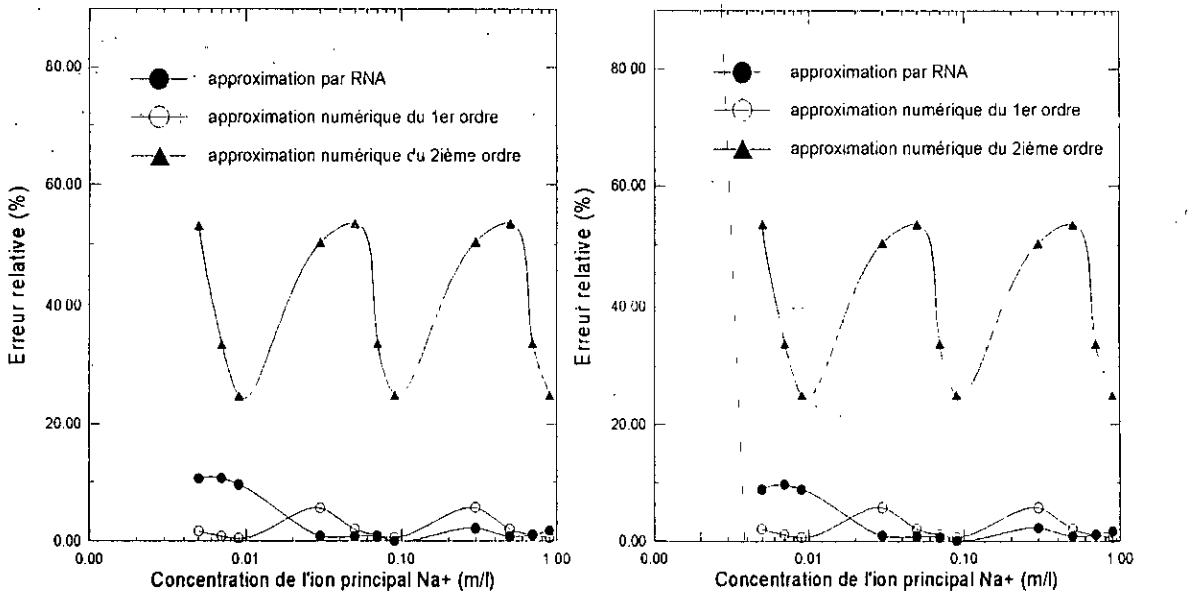


Fig. 4.19 comparaison des résultats des approximations numériques du premier et second ordre avec les résultats de généralisation du RNA dans les cas : (a) $[K^+]=5\times10^{-4}$ m/l, (b) $[K^+] = 5\times10^{-2}$ m/l.

4.7 LINEARISATION FLOUE DE L'ISE A MEMBRANE DE NASICON

4.7.1 PRINCIPE DE FONCTIONNEMENT

Les systèmes flous (Fuzzy Logic Systems) sont des approximateurs universels, ils sont donc applicables à la reproduction de fonctions nonlinéaires en général, et à la linéarisation des caractéristiques de capteurs en particulier. C'est, justement, cette capacité des FLS qu'on a voulu tester.

Le principe de fonctionnement du FLS, pour linéariser la caractéristique du capteur ionique, ne diffère pas du principe de fonctionnement du linéarisateur neuronal étudié précédemment. Le linéarisateur flou a aussi pour rôle de reproduire la caractéristique inverse de l'ISE, et une fois relié en cascade au capteur, le bloc global se comportera comme un système linéaire (Fig. 4.20).

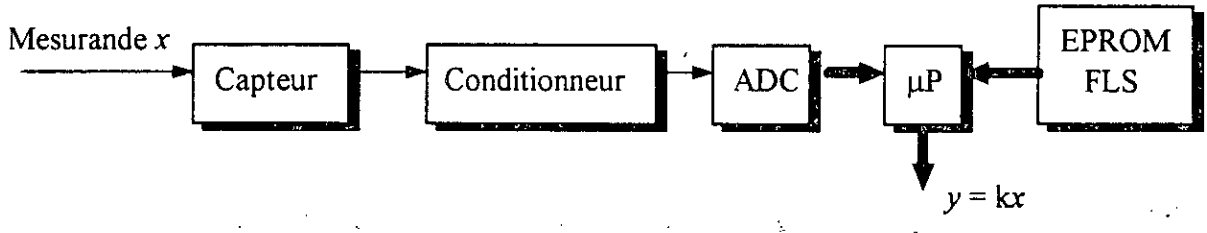


Fig. 4.20 insertion d'un linéarisateur flou dans une chaîne instrumentale.

Le déroulement du travail est aussi divisé en deux étapes à savoir, une linéarisation en l'absence d'ions interférents (relation de Nernst). Dans ce cas, le FLS possède une entrée et une sortie (Fig. 4.21), et une linéarisation en présence d'un ion interférent (relation de Nicolskii-Eisenmann). Dans ce second cas, le FLS possède deux entrées et une sortie (Fig. 4.22). Dans les deux cas, la base de règles du FLS est générée à partir d'observations opérées sur l'évolution de la caractéristique inverse de l'ISE.

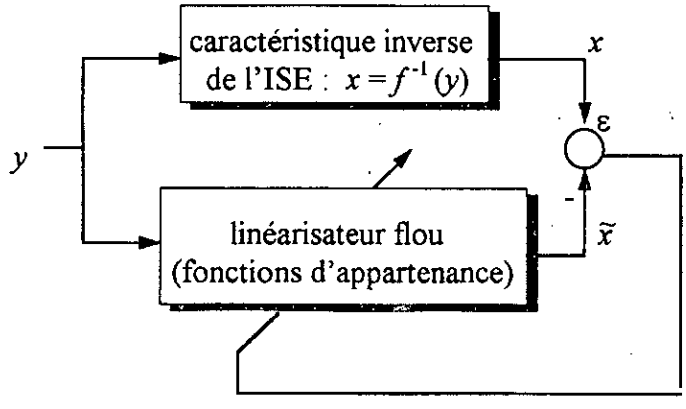


Fig. 4.21 adaptation du linéarisateur flou en l'absence d'ions interférents.

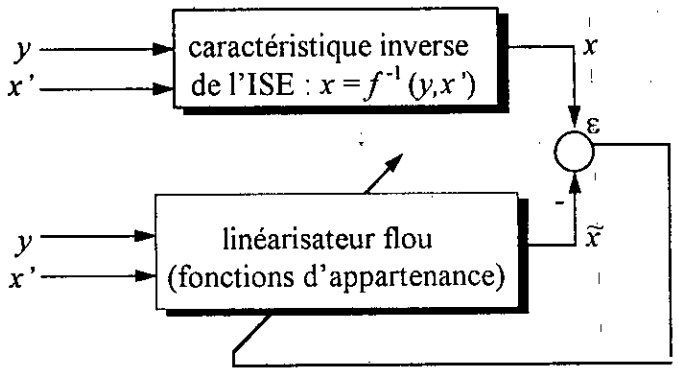


Fig. 4.22 adaptation du linéarisateur flou en présence d'un ion interférent.

Il est clair, à partir des figures (4.21) et (4.22), qu'il y a une adaptation du FLS en vue de minimiser l'erreur d'approximation. En effet, le choix à priori des fonctions d'appartenance des entrées et sorties n'est pas toujours adéquat et ne donne pas forcement de bons résultats. C'est pour cela, qu'on est amené à modifier ces fonctions d'appartenance pour apporter des améliorations. Bien que cette méthode d'adaptation est en partie heuristique, on ne peut négliger qu'elle est basée sur des observations du comportement du FLS à travers ses différentes étapes de fonctionnement c-à-d, la fuzzification, l'inférence et la défuzzification. C'est en quelque sorte un apprentissage du linéarisateur flou.

4.7.2 LINEARISATION DE L'ISE EN L'ABSENCE D'IONS INTERFERENTS

En l'absence d'ions interférents, la tension délivrée par l'ISE est régie par la relation de Nernst (4.7). La nonlinéarité logarithmique inverse que le FLS doit approximer est donnée par les relations (4.8) et (4.9). La figure (4.23) montre le mode d'opération du FLS.

Le système flou choisi, possède un singleton fuzzifier, une inférence max-produit, une composition max et une défuzzification centroïde. Les fonctions d'appartenance sont triangulaires avec pour point d'intersection 0.5, étalés sur l'intervalle de variation de [Na^+] allant de 10^{-3} à 1m/l concernant les fonctions d'appartenance de sortie, et sur l'intervalle de variation de Log(a_{Na^+}) allant de -3 à 0 concernant les fonctions d'appartenance d'entrée.
Le nombre de règles établies est de trente (30), de la forme :« si Log (a_{Na^+}) est A, alors a_{Na^+} est B », A et B étant des ensembles flous allant de très petit à très grand.

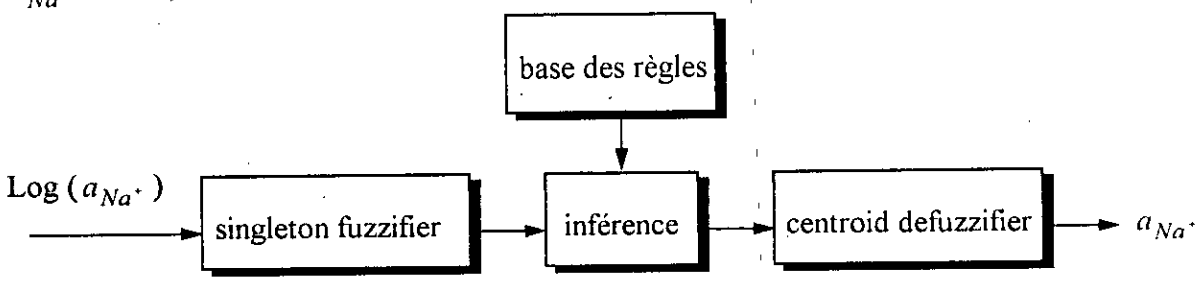


Fig. 4.23 schéma bloc montrant le mode d'opération du linéarisateur flou en l'absence d'ions interférents.

A. Résultats de la linéarisation

Les résultats obtenus par le linéarisateur flou sont schématisés, graphiquement, dans la figure (4.24). Le FLS s'est révélé très efficace pour reproduire la relation inverse de Nernst, puisque les erreurs relatives commises sur les sorties ne dépassent pas les 0.9%, et allant même jusqu'à 7×10^{-6} % pour une variation de $[Na^+]$ allant de 10^{-3} à 1m/l, un intervalle assez large.

La qualité de la linéarisation floue est, à son tour, présentée dans la figure (4.25), illustrant l'approximation de la relation : $[Na^+] = f(\text{Log}[Na^+])$ par le FLS.

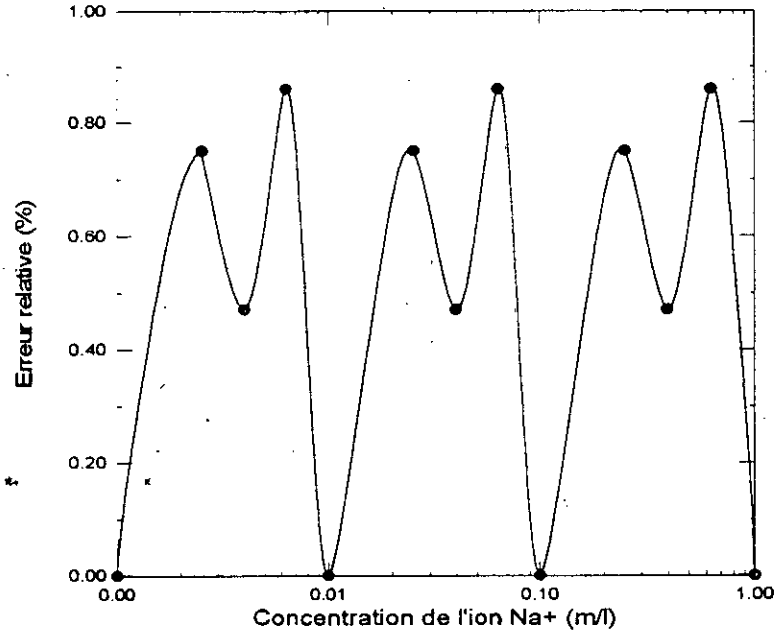


Fig. 4.24 résultats de la linéarisation floue en l'absence d'ions interférents.

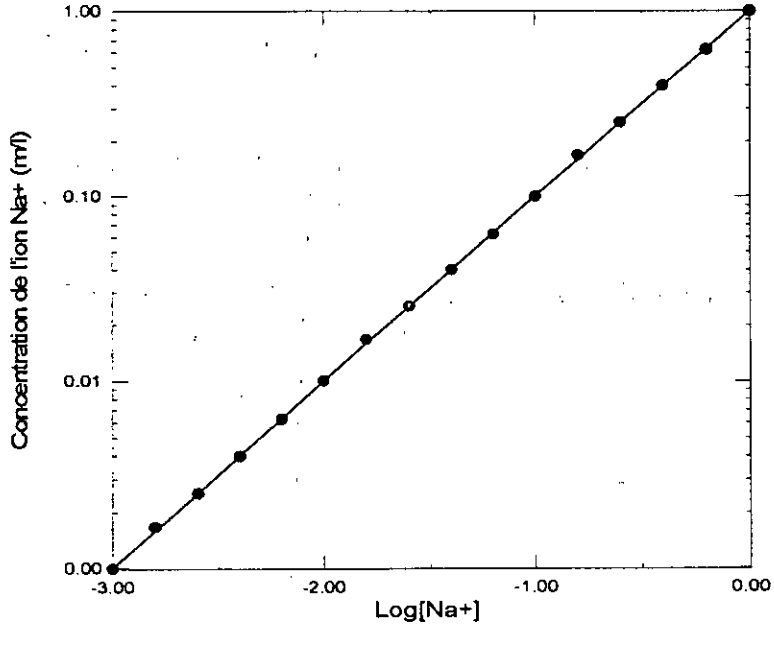


Fig. 4.25 approximation de la relation $[Na^+] = f(\text{Log}[Na^+])$ par le FLS.

B. Comparaison avec la linéarisation neuronale

Les résultats obtenus par le linéarisateur neuronal, durant la phase d'apprentissage, ont été comparés avec ceux obtenus par le linéarisateur flou. La figure (4.26) révèle que l'approximation floue fournit des erreurs relatives inférieures à celles fournies par le linéarisateur neuronal. Ceci peut être justifié par le fait que la conception du FLS présente beaucoup plus de flexibilité, quant au choix des fonctions d'appartenance, et est basée sur la bonne connaissance de la variation de la fonction à approximer, à savoir la relation inverse de Nernst. En d'autres termes, c'est notre expérience personnelle qui est transférée directement au FLS. Cette connaissance est un peu difficile à mettre à profit lors de la conception du linéarisateur neuronal, qui apprend par lui même la variation de la fonction à travers ses exemples d'entraînement, et de ce fait la conception est plus difficile dans le cas du linéarisateur neuronal que dans le cas du linéarisateur flou.

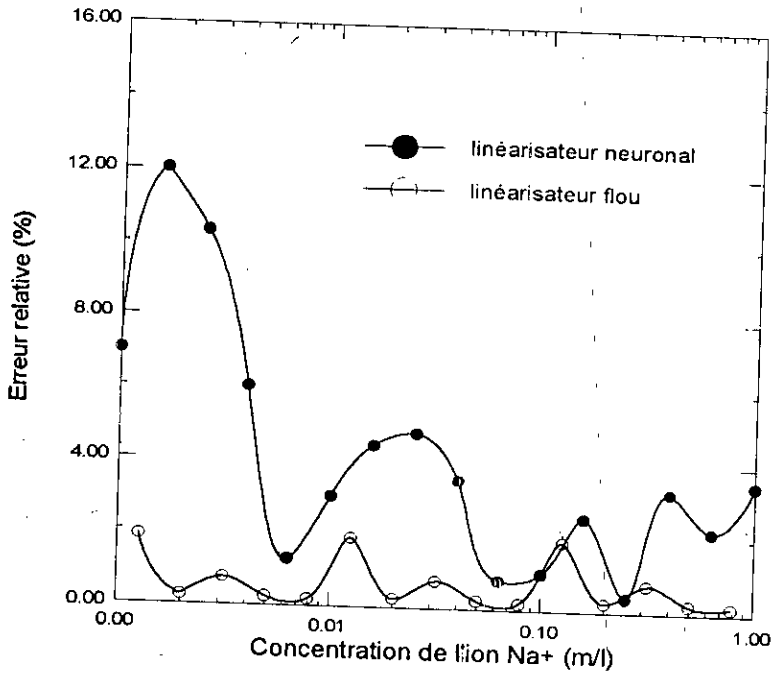


Fig. 4.26 comparaison des résultats de la linéarisation floue avec ceux de la linéarisation neuronale, en l'absence d'ions interférents.

4.7.3 LINEARISATION DE L'ISE EN PRESENCE DE L'ION INTERFERENT K^+

Ici, la caractéristique directe de l'électrode est régie par la relation de Nickolskii-Eisenmann (4.10). Comme dans le cas de la linéarisation neuronale, le FLS a pour rôle de reproduire la nonlinéarité inverse du capteur ionique, résumée dans la relation (4.11).

4.7.3.1 Linéarisation dans le cas de k_P variable et $[K^+]$ fixe

Les données sur lesquelles on travaille sont issues de la relation (4.11), et la variation du coefficient de sélectivité k_P s'opère suivant le tableau (4.1) sus mentionné. Ceci étant, l'intervalle de variation de la concentration de Na^+ va de 5×10^{-4} à 5×10^{-1} m/l. La concentration de K^+ est égale à 1m/l.

La structure du système flou choisie est résumée dans la figure (4.27). Celui-ci possède une entrée ($Log\left(a_{Na^+} + k_p\, a_{K^+}\right)$), et une sortie (a_{Na^+}). Il est constitué d'un singleton fuzzifier, d'une inférence max-produit, d'un défuzzificateur centroïde et d'une base de règles formulées à partir d'observations faites sur la variation de la relation inverse de Nickolskii-Eisenmann. Le nombre de règles utilisées est de trente deux (32), toutes de la forme :

« si $Log\left(a_{Na^+} + k_p\, a_{K^+}\right)$ est A, alors a_{Na^+} est B »,

A et B étant des ensembles flous allant de très petit à très grand.

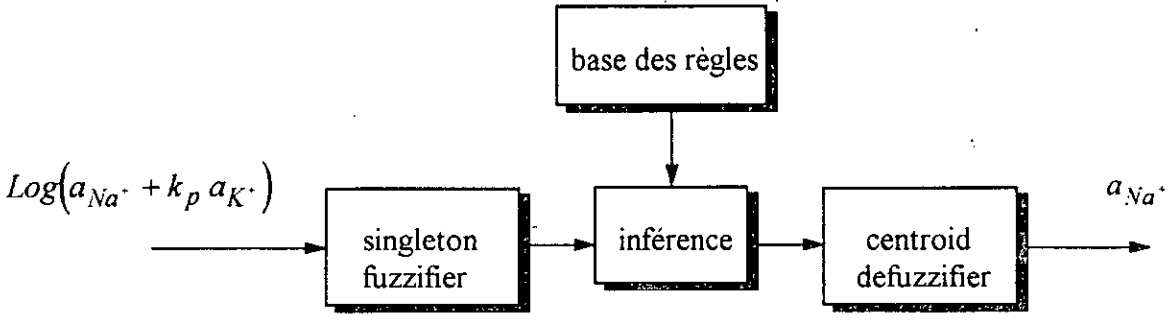


Fig. 4.27 schéma bloc montrant le mode d'opération du linéarisateur flou en présence de l'ion interférent K^+ (cas de k_P variable).

A. Résultats de la linéarisation

la figure (4.28), qui présente les erreurs relatives commises par le linérisateur flou, montre, une fois encore, la grande capacité de ce système à reproduire ce type de nonlinéarité. En effet, ces erreurs ne dépassent pas, en majorité, les 0.5%, allant même jusqu'à 5×10^{-3} %. Le FLS prouve donc qu'il est capable « d'absorber » la variation du coefficient k_P.

Pour mieux voir la qualité de la linérisation floue, dans ce cas, la figure (4.29) nous illustre graphiquement l'approximation de la relation : $[Na^+] = f(\mathrm{Log}([Na^+] + k_P\,[K^+]))$ par le FLS.

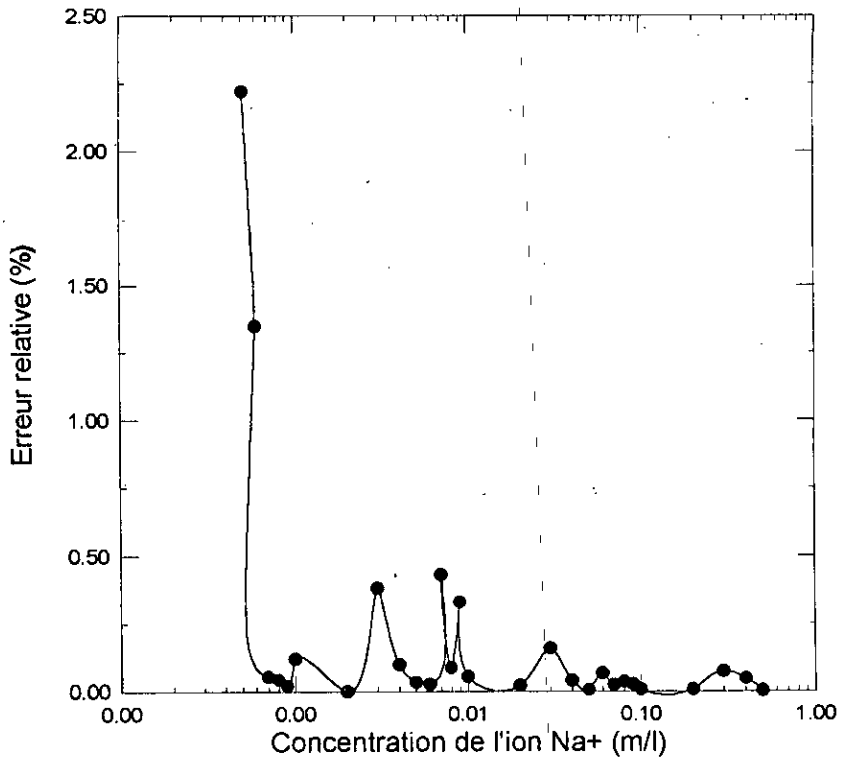


Fig. 4.28 résultats de la linéarisation floue en présence de l'ion interférent K^+ (cas de k_P variable).

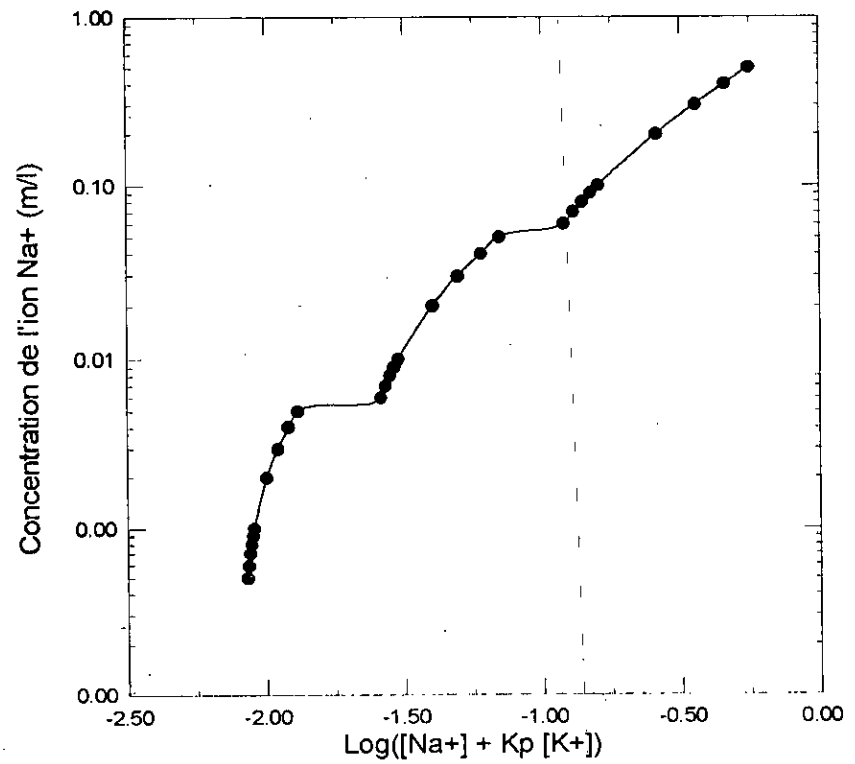


Fig. 4.29 approximation de la relation nonlinéaire : $[Na^+] = f\,(\mathrm{Log}([Na^+] + k_P\,[K^+]))$ par le FLS.

B. Comparaison avec la linéarisation neuronale

La comparaison des résultats de la linéarisation neuronale avec ceux de la linéarisation floue a abouti aux résultats de la figure (4.30). Les erreurs relatives du linéarisateur flou sont en majorité inférieures à 2%, et celles du linéarisateur neuronal sont, en majorité, inférieures à 5%. De ce fait, les résultats fournis par le linéarisateur flou sont légèrement meilleurs par rapport aux résultats du linéarisateur neuronal, sauf sur l'intervalle de $[Na^+]$ allant de 10^{-1} à 5×10^{-1} m/l, où le RNA s'est révélé un peu meilleur. Les raisons de cette différence entre les deux techniques de linéarisation sont les mêmes que celles mentionnées durant la linéarisation en l'absence d'ions interférents.

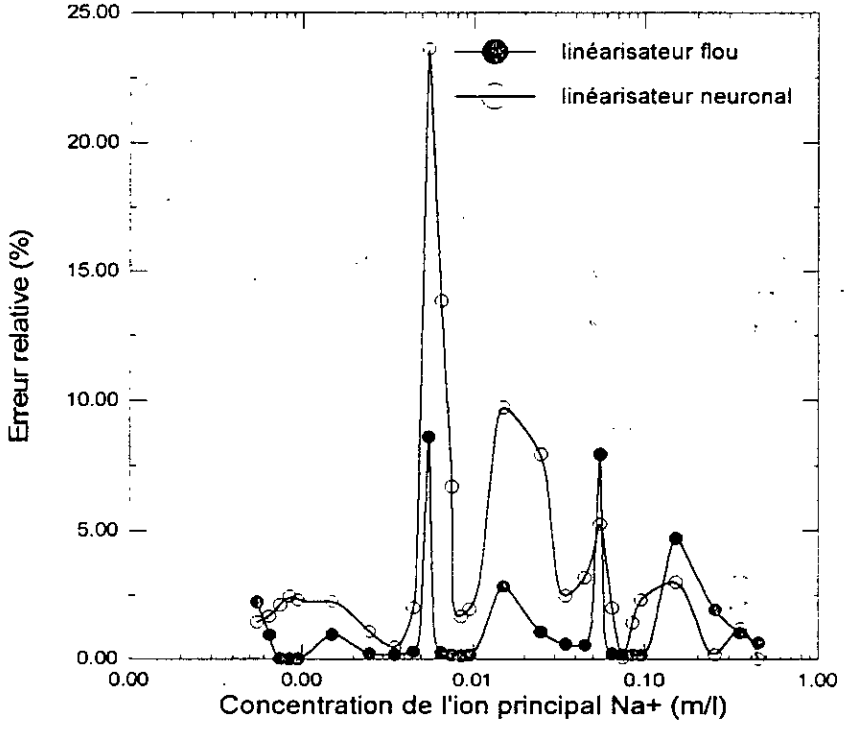


Fig. 4.30 comparaison des résultats de la linéarisation floue avec ceux de la linéarisation neuronale (cas de k_P variable).

4.7.3.2 Linéarisation dans le cas de k_P fixe et $[K^+]$ variable

On désire, toujours, reproduire la relation inverse de Nickolskii-Eisenmann (4.11) à l'aide d'un système flou, mais dans ce cas, la solution à analyser présente une concentration variable de K^+. Le linéarisateur flou utilise les mêmes données pratiques que son prédécesseur (le linéarisateur neuronal) à savoir, un coefficient de sélectivité égal à 1.58×10^{-3}. Ceci étant, le FLS possède deux entrées, $Log\left(a_{Na^+} + k_p\, a_{K^+}\right)$ et a_{K^+}, et une seule sortie a_{Na^+} (Fig. 4.31). Il est, à son tour, constitué d'un singleton fuzzifier, d'une inférence max-produit, d'un défuzzificateur centroïde, et d'une base de règles formulées à partir des observations faites sur la variation de la relation inverse de Nickolskii-Eisenmann. Le nombre de règles utilisées est de cinquante deux (52), toutes de la forme :

« si $Log\left(a_{Na^+} + k_p\, a_{K^+}\right)$ est A, et a_{K^+} est B, alors a_{Na^+} est C »,

A, B et C étant des ensembles flous allant de très petit à très grand.

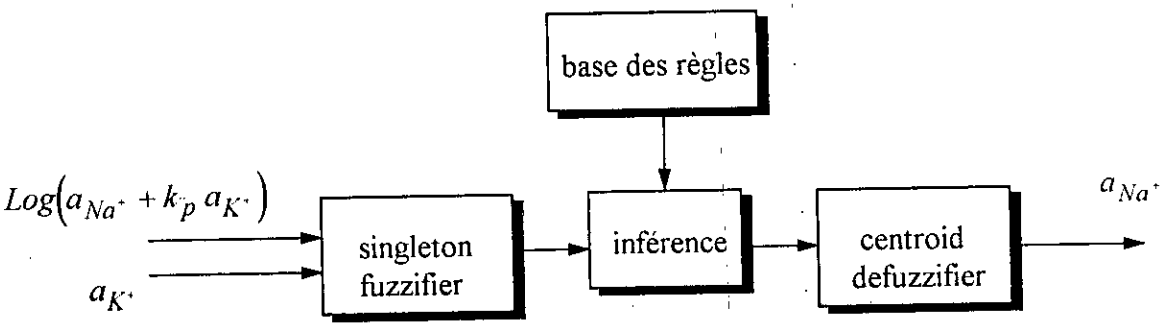


Fig. 4.31 schéma bloc montrant le mode d'opération du linéarisateur flou dans le cas de $[K^+]$ variable.

A. Résultats de la linéarisation

Le linéarisateur flou s'est montré assez performant pour simuler une nonlinéarité de type dur, à deux variables, sur deux intervalles très larges allant de 5×10^{-3} à 1m/l pour $[Na^+]$, et de 10^{-4} à 1m/l pour $[K^+]$. En effet, la figure (4.32) nous révèle, graphiquement, les erreurs relatives obtenues par le FLS pour diverses valeurs de la concentration de K^+. Ces erreurs ne dépassent pas, en majorité, les 0.2%, pour $[K^+]$ allant de 10^{-4} à 10^{-2} m/l, et sont inférieures, pour la plupart, à 0.8% pour $[K^+]$ allant de 10^{-2} à 1m/l.

Bien que les erreurs relatives, commises par le FLS, croient avec l'augmentation de $[K^+]$, marquant ainsi la légère sensibilité du linéarisateur flou à la variation de l'ion interférent, le FLS a réussi à garder ses performances dans les limites souhaitées.

La figure (4.33) montre la qualité de la linéarisation floue pour deux valeurs extrêmes de $[K^+]$ à savoir, 10^{-4} et 1m/l. On peut remarquer l'aptitude du FLS à suivre la déviation de la droite, due à la variation de $[K^+]$.

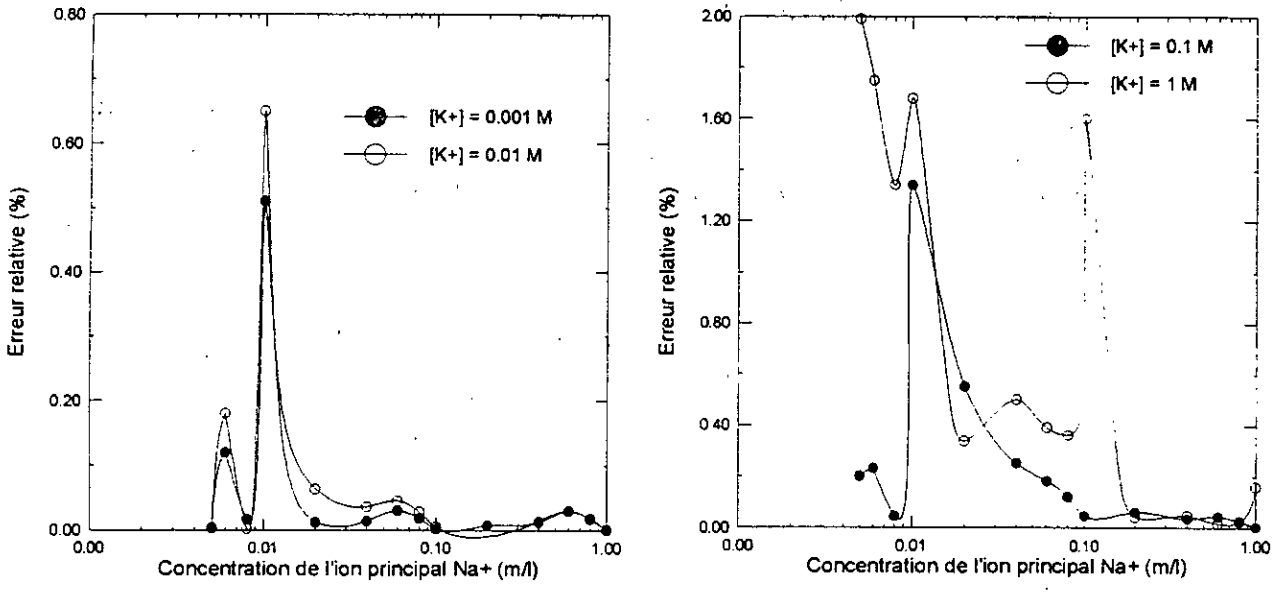


Fig. 4.32 résultats du linéarisateur flou dans le cas de $[K^+]$ variable.

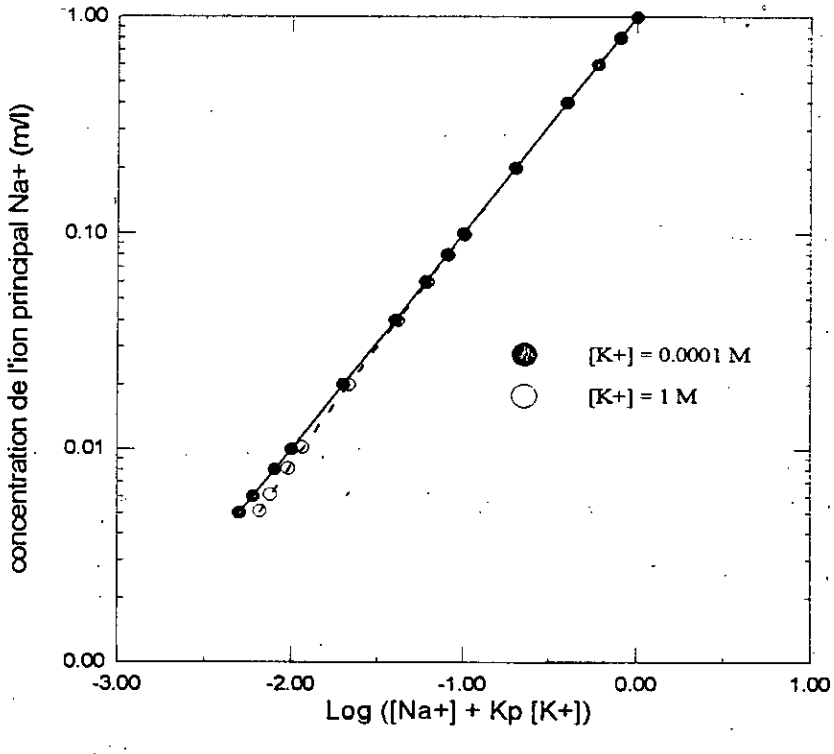
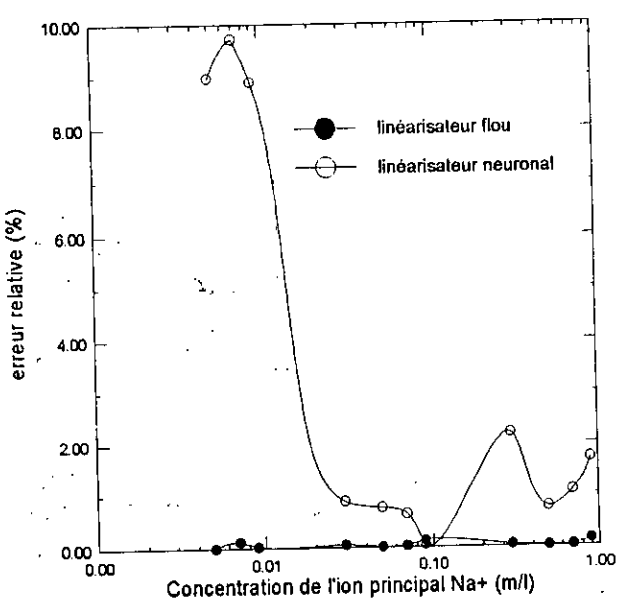


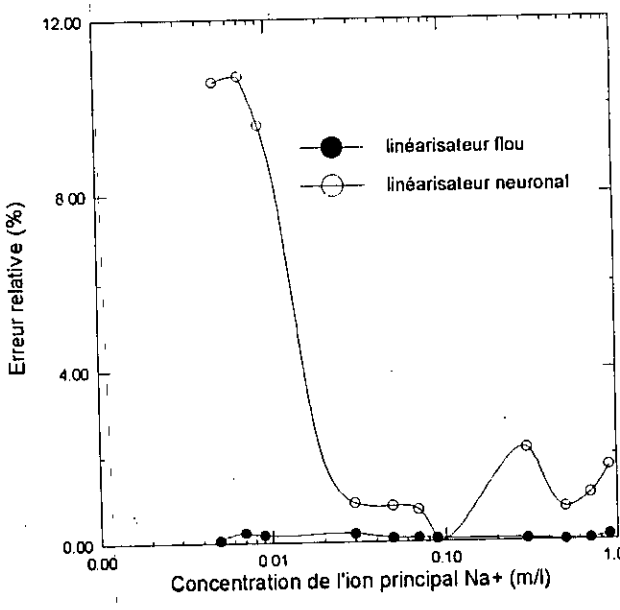
Fig. 4.33 approximation de la relation nonlinéaire : $[Na^+] = f(\text{Log}([Na^+] + k_P\,[K^-]))$ par le FLS dans le cas de $[K^+]$ variable.

B. Comparaison avec la linéarisation neuronale

Comme ca a été le cas pour les deux étapes précédentes, une comparaison entre les résultats de la linéarisation neuronale et ceux de la linéarisation floue, a été effectuée. Le fruit de cette comparaison est illustré dans la figure (4.34) qui révèle que sur les 11 exemples de test pris, le FLS s'est montré nettement meilleur dans la plage de variation de $[Na^+]$ allant de 5×10^{-3} à 2×10^{-2} m/l, et légèrement meilleur pour $[Na^+]$ allant de 2×10^{-3} à 1m/l. Notons au passage que ceci est dû, comme on l'a déjà mentionné, à la différence de conception des deux systèmes. Le FLS a présenté une plus grande flexibilité quant à sa capacité de recevoir directement l'expérience acquise par le conceptionneur, lui même, durant son observation de la fonction à approximer.



(a)



(b)

Fig. 4.34 comparaison des résultats de la linéarisation floue avec ceux de la linéarisation neuronale : (a) $[K^+]=5\times10^{-3}$ m/l, (b) $[K^+]=5\times10^{-2}$ m/l.

4.8 CONCLUSIONS

Le travail effectué le long de ce chapitre, concernant la linéarisation de la caractéristique statique d'un capteur ionique, nous a permis, en plus de tester la faisabilité de cette méthode originale, d'en sortir avec les remarques et conclusions suivantes :

Les deux techniques de linéarisation (neuronale et floue) proposées, nous ont permis, implicitement, de linéariser la caractéristique de l'ISE dans un intervalle assez large de variation de la température ambiante. Ceci bien sûr, constitue un atout considérable vu que ce capteur est sensible à la variation de la température. Il suffit donc d'utiliser un capteur de température adéquat afin de calculer l'entrée exacte du linéarisateur neuronal ou flou.

De plus, et le linéarisateur neuronal et le linéarisateur flou permettent une linéarisation dans des intervalles de variation assez larges de la concentration de l'ion principal Na^+. Ceci nous éloigne des techniques de linéarisation autour d'un point de fonctionnement, qui elles sont limitées à des intervalles de variation trop étroits, et qui diminuent encore plus si on a affaire à de fortes nonlinéarités.

Quelques difficultés ont été rencontrées lors de l'apprentissage du RNA, dans le cas d'un coefficient de sélectivité variable. Le réseau neuronal a présenté un comportement différent, qu'on ne peut justifier, sur les trois parties distinctes de la caractéristique. Un comportement qu'on n'a pas remarqué avec le linéarisateur flou. Néanmoins, il faut souligner au passage qu'avec un peu plus de temps et en modifiant la structure du réseau en jouant sur les fonctions d'activation, le nombre de neurones et le conditionnement des données d'entraînement, on peut sûrement améliorer les résultats. Dans la même optique, pour pouvoir améliorer les résultats du linéarisateur flou, plusieurs structures du FLS ont été testées en modifiant les fonctions d'appartenance ainsi que les règles d'inférence.

Pour ce qui est de la linéarisation dans le cas de $[K^+]$ variable (présence d'une grandeur de perturbation), les deux méthodes de linéarisation se sont révélées capables d'inhiber l'influence de cette perturbation sur une large plage de variation, et ceci en la prenant comme entrée supplémentaire, délivrée par un capteur adéquat.

Enfin, pour mieux illustrer la qualité des linéarisations neuronale et floue, celles-ci ont fait l'objet de comparaisons. En effet, le RNA a présenté des résultats meilleurs que ceux de la linéarisation numérique du second ordre, et très proches de ceux de la linéarisation numérique du premier ordre. Le linéarisateur flou, quant à lui, a fourni de bons résultats relativement au RNA, et s'est placé en pôle position. Ceci est expliqué par le fait que le système flou a présenté une plus grande souplesse lors de sa conception, puisqu'il s'agissait de lui transmettre directement un savoir faire acquit par l'opérateur humain à travers des observations faites sur la caractéristique statique du capteur ionique.

CHAPITRE 5

COMMANDE NEURONALE ET FLOUE D'UN BIOREACTEUR

" Le génie est fait d'un pourcent d'inspiration et de quatre vingt dix neuf pourcent de transpiration."

T.A. Edison (1847-1931)

5.1 INTRODUCTION

Après avoir testé les réseaux de neurones artificiels et les systèmes flous comme outils de linéarisation des caractéristiques nonlinéaires de capteurs, et dû aux performances obtenues par ces techniques dans ce domaine comme dans d'autres, il s'avère indispensable de les appliquer à des problèmes réels de commande de systèmes nonlinéaires. La commande des processus chimiques offre un bon terrain d'essai et de test pour ces nouvelles techniques. En effet, les systèmes chimiques sont souvent fortement nonlinéaires et difficiles à commander par les méthodes conventionnelles.

La commande des processus chimiques tels que les réacteurs et les colonnes de distillation a été largement étudiée [71], ils ont fourni à la commande adaptative les principales applications. Cependant, ces techniques de commande deviennent limitées si le système chimique à contrôler tend à être complexe. C'est pour cette raison que l'utilisation des réseaux de neurones artificiels et des méthodes de la logique floue peut, non seulement améliorer les performances du système, mais aussi stimuler la commande adaptative en suggérant de nouvelles approches ainsi que des architectures de commande efficaces.

Cette partie de notre travail est réservée à l'application des RNA et des FLS à la commande d'un bioréacteur qui constitue un domaine attractif permettant de tester la faisabilité et l'efficacité de ces deux méthodes. De nouvelles architectures de commande seront présentées et étudiées, les réponses temporelles du système permettront de discuter des performances de celui-ci. Enfin, les deux techniques de commande feront l'objet de comparaison.

5.2 MOTIVATIONS POUR LA COMMANDE DES PROCESSUS CHIMIQUES

Malgré le progrès enregistré par les contrôleurs auto-ajustables et la commande adaptative à modèle de référence, il existe certains problèmes dans l'industrie chimique pour lesquels les techniques courantes sont inadéquates, à titre d'exemple nous citerons les bioréacteurs. Pour la plupart des processus chimiques, un large nombre de données (entrées / sorties) sont disponibles, mais il est difficile de formuler des modèles précis [72]. C'est justement pour ce type de problèmes que les réseaux neuronaux et les systèmes flous sont sollicités. Quelques processus chimiques sont fortement nonlinéaires, ces nonlinéarités peuvent être intrinsèques au fonctionnement physique ou chimique du processus, ils peuvent aussi être dues au couplage de sous systèmes simples. Le système global est par contre multivariable et dur à commander.

Les systèmes à réactions chimiques (réacteurs) présentent un problème de commande largement étudié. Bien que de tels réacteurs sont décrits par des équations simples, ils peuvent présenter des comportements complexes tels que de multiples états permanents et un comportement chaotique [73][74]. Ce sont des systèmes difficiles à modéliser à cause des organismes vivants qui sont à l'intérieur. La difficulté de les contrôler est aussi liée au fait qu'on ne peut pas souvent mesurer online les concentrations des éléments chimiques produits ou métabolisés. De plus, les réacteurs chimiques, et particulièrement les bioréacteurs peuvent avoir différents régimes de fonctionnement, à savoir si les « bugs » (bactéries ou levure) sont en mode de croissance ou de production [72], en plus d'une dynamique complexe. Le problème à résoudre est donc double : déterminer un modèle représentatif du système et trouver une loi de commande qui fait face aux incertitudes du modèle nonlinéaire.

5.3 DESCRIPTION DU PROBLEME DE COMMANDE D'UN BIOREACTEUR

5.3.1 MODELISATION

Le bioréacteur peut être considéré comme relativement simple car il possède peu de variables, mais encore difficile à commander à cause de ses fortes nonlinéarités qu'on ne peut modéliser d'une manière précise. Sous sa forme la plus simple, le bioréacteur est un réservoir contenant de l'eau et des cellules (e.g., bactéries ou levures) qui consomment des éléments nutritifs (substrat) et produisent certains produits (désirés et non désirés) et plus de cellules. Le bioréacteur est d'autre part compliqué, car les cellules peuvent changer leur taux de croissance et de production radicalement, suivant la température et la concentration des déchets (e.g., l'alcool). Pour un début, un modèle relativement simple du bioréacteur constitue le meilleur choix.

Justement, la version la plus simple du problème est un réservoir à flux continu dans lequel la croissance des bactéries dépend uniquement des élément nutritifs donnés au système. La grandeur à contrôler est la concentration des bactéries dans le réservoir. Un tel bioréacteur est régi par le système d'équations différentielles suivant [72] :

$$\begin{cases} \dfrac{dC_1}{dt} = -C_1 w + C_1(1-C_2)\, exp\,(C_2/\gamma) \\ \dfrac{dC_2}{dt} = -C_2 w + C_1(1-C_2)\, exp\,(C_2/\gamma)\dfrac{1+\beta}{1+\beta-C_2} \end{cases} \tag{5.1}$$

où C_1 et C_2 sont respectivement, la concentration relative des bactéries et la concentration relative convertie du substrat, avec :

$$C_2 = \frac{S_f - S}{S_f} \tag{5.2}$$

où S_f est la concentration relative du substrat dans le flux et S est la concentration relative du substrat dans le réacteur (voir Fig. 5.1). L'action de commande w est le taux du flux à travers le réacteur. La première équation montre que la variation de la quantité de bactéries dans le réservoir est égale à la quantité de bactéries qui sort de celui-ci ($C_1\, w$) plus la quantité qui s'ajoute due à la croissance des cellules ($C_1\,(1 - C_2)\, exp\,(C_2/\gamma)$). La seconde équation montre que la variation de la quantité de substrat (C_2) dans le réservoir est égale à la quantité du substrat qui quitte celui-ci plus la quantité de substrat métabolisé par les bactéries. Les constantes β et γ déterminent le taux de croissance des cellules et le taux de consommation des éléments nutritifs respectivement. Ce modèle n'est pas complètement réel, mais il fournit un système simple dont la commande est un vrai défi.

Ce système est difficile à commander pour différentes raisons : les équations qui régissent son fonctionnement sont fortement nonlinéaires. Un comportement optimal du système est proche de la région d'instabilité. De plus, le problème présente une multiplicité : deux

différentes valeurs de l'action de commande (flux w) peuvent mener au même état permanent concernant la quantité de cellules donnée.

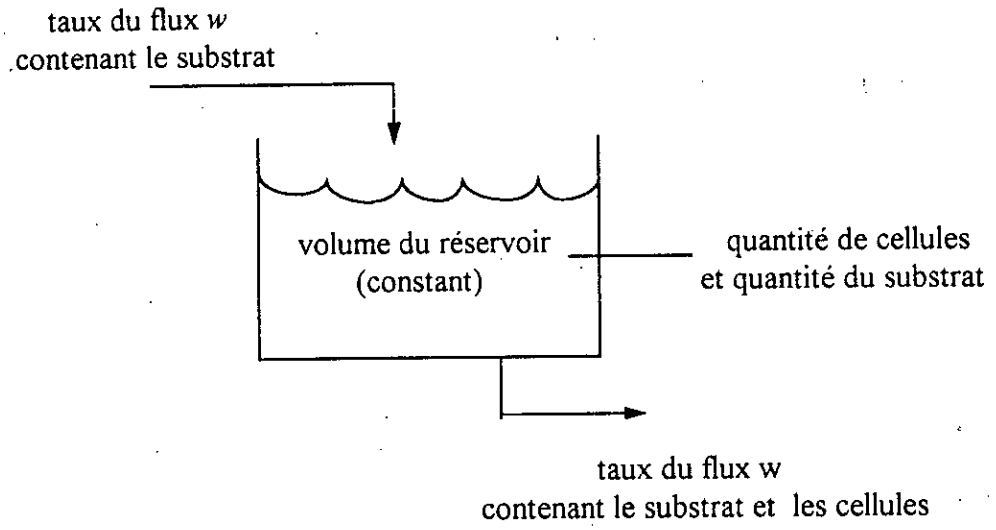


Fig. 5.1 diagramme représentant le bioréacteur.

Il est à noter que ce système possède les contraintes suivantes :

- $0 \leq C_1 , C_2 \leq 1$ (grandeurs relatives)
- $0 \leq w \leq 2$

La version discrète du modèle continu (5.1) est donnée par le système suivant :

$$\begin{cases} C_1(t+1) = C_1(t) + T\left(-C_1(t)\, w(t) + C_1(t)\ (1 - C_2(t))\ exp\ (C_2(t)/\gamma)\right) \\ C_2(t+1) = C_2(t) + T\left(-C_2(t)\, w(t) + C_1(t)\ (1 - C_2(t))\ exp\ (C_2(t)/\gamma)\dfrac{1+\beta}{1+\beta - C_2(t)}\right) \end{cases} \quad (5.3)$$

où T est la constante d'échantillonnage. Durant la simulation, on prendra pour paramètres les valeurs suivantes [72] : $\beta = 0.02$, $\gamma = 0.48$, $T = 0.01$s. Les valeurs initiales $C_1(0)$, $C_2(0)$ et $w(0)$ sont des variables aléatoires, suivant la loi uniforme.

5.3.2 COMPORTEMENT DU SYSTEME EN BOUCLE OUVERTE

Avant d'entamer la commande du bioréacteur, nous avons jugé utile de présenter le comportement de ce système en boucle ouverte, c'est à dire l'évolution dans le temps des états du système (C_1 (t) et C_2 (t)) due à une sollicitation donnée du flux w.

Au fait, ce système possède trois comportements différents dans trois intervalles distincts du flux w :

- Dans l'intervalle de w allant de 0 à 0.75, les états du système sont stables et ne dépendent pas de leurs valeurs initiales. Le même flux w mène aux mêmes états permanents quels que soient leurs états initiaux. La figure (5.2) montre les réponses du système pour deux valeurs différentes du flux w, à savoir 0.40 et 0.75.

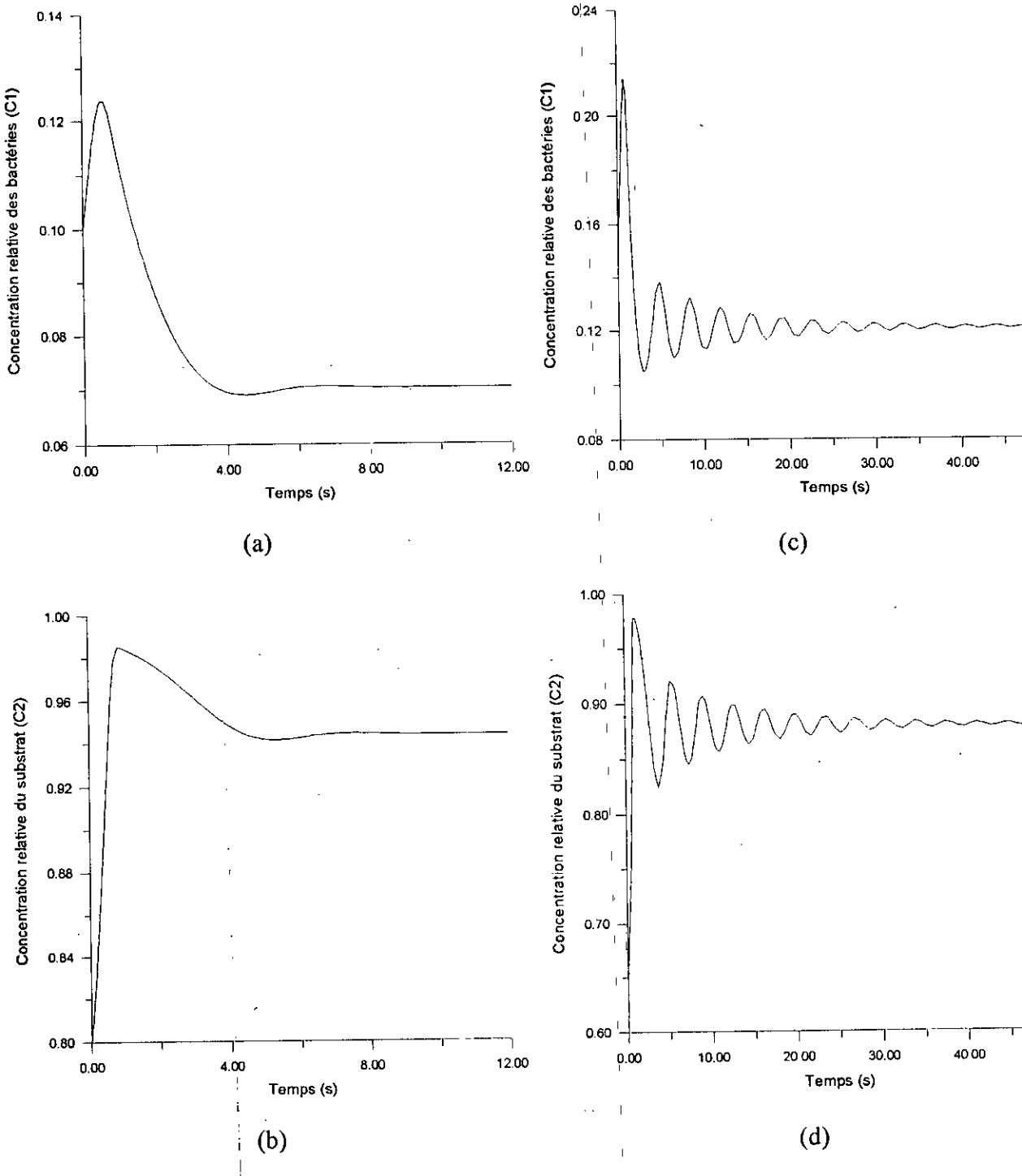


Fig. 5.2 réponses du système : (a) et (b) cas où $w = 0.40$, (c) et (d) cas où $w = 0.75$.

- Dans l'intervalle allant de 0.75 à environ 1.20 le système est oscillatoire, et donc considéré comme instable. De plus, quelle que soit la valeur du flux w dans cet intervalle, les états présentent des réponses différentes pour des valeurs initiales différentes. La figure (5.3) montre l'instabilité du système pour une valeur du flux égale à 0.85. La figure (5.4) montre les réponses du système pour une valeur du flux égale à 1.20, et présente l'influence des états initiaux sur celles-ci.

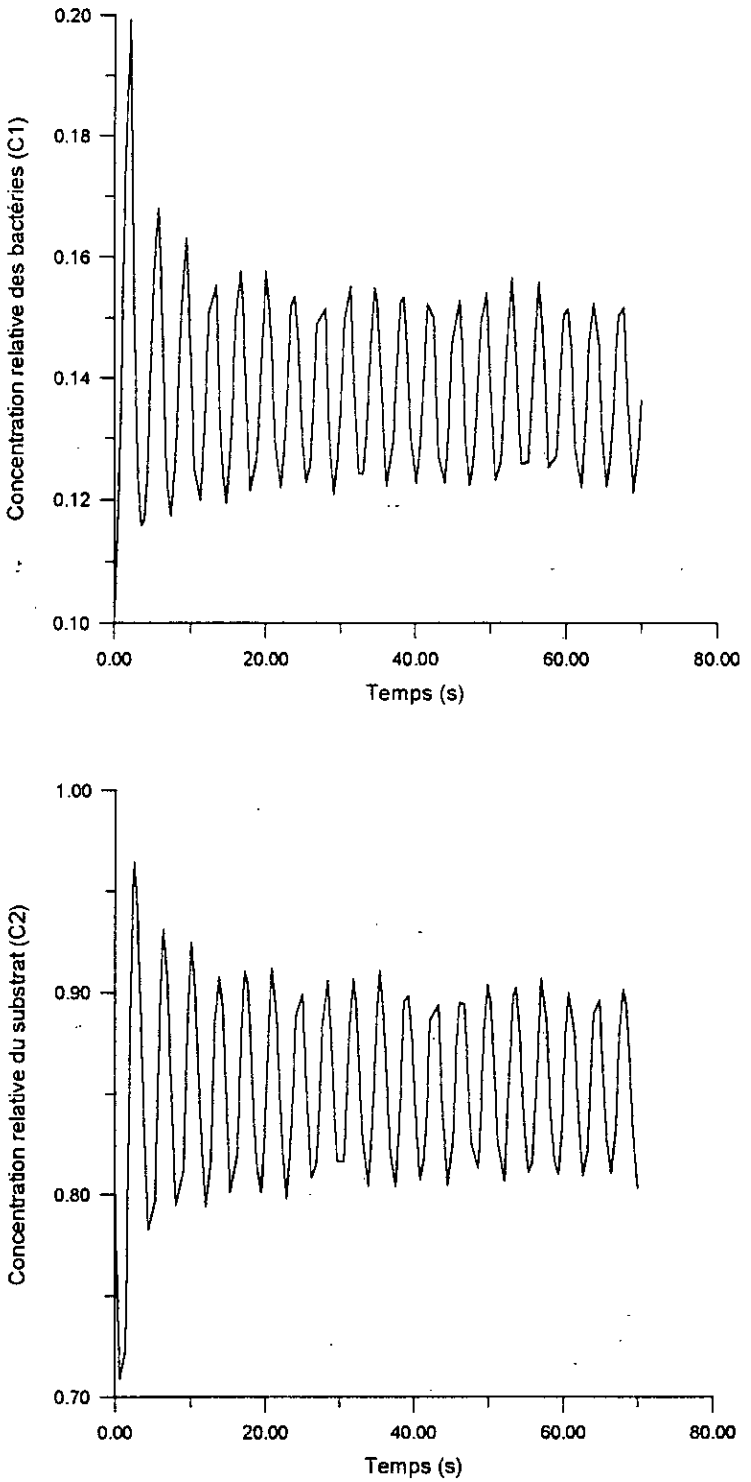
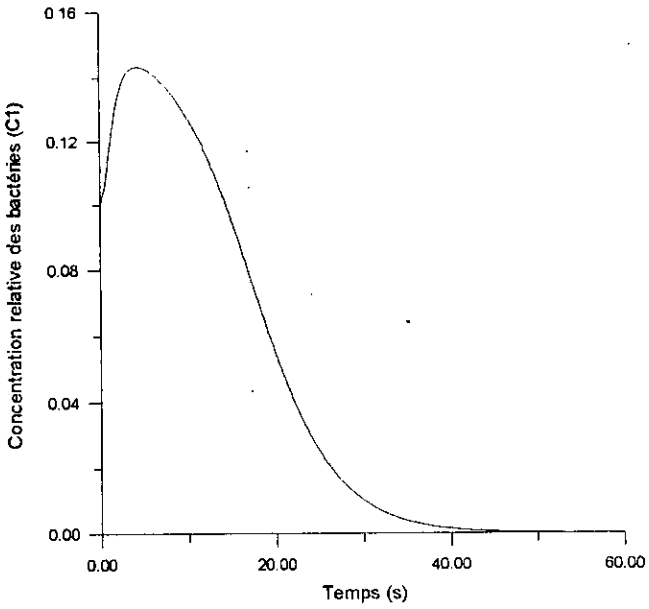
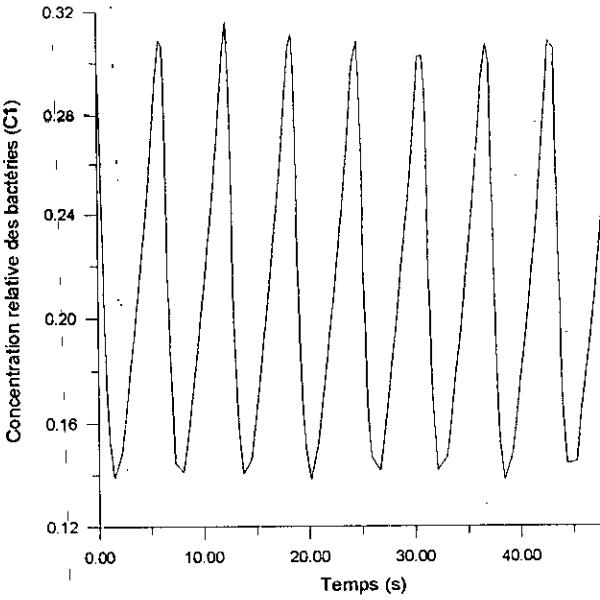


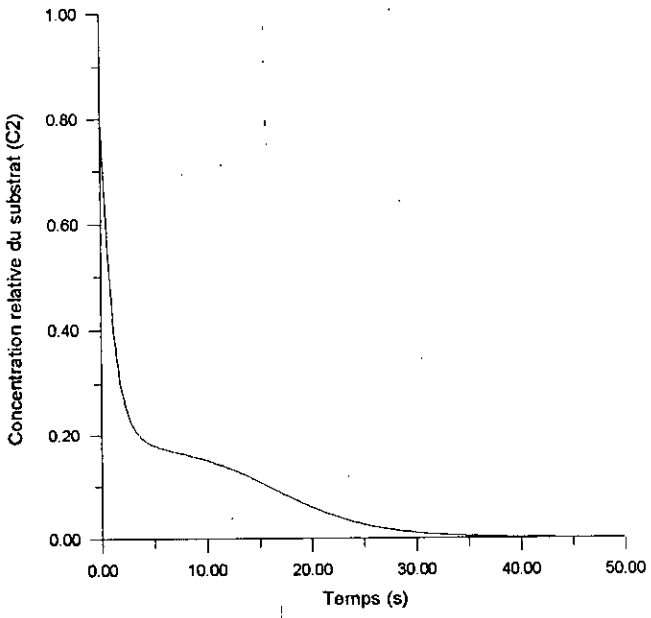
Fig. 5.3 réponses du système pour une valeur du flux $w = 0.85$.



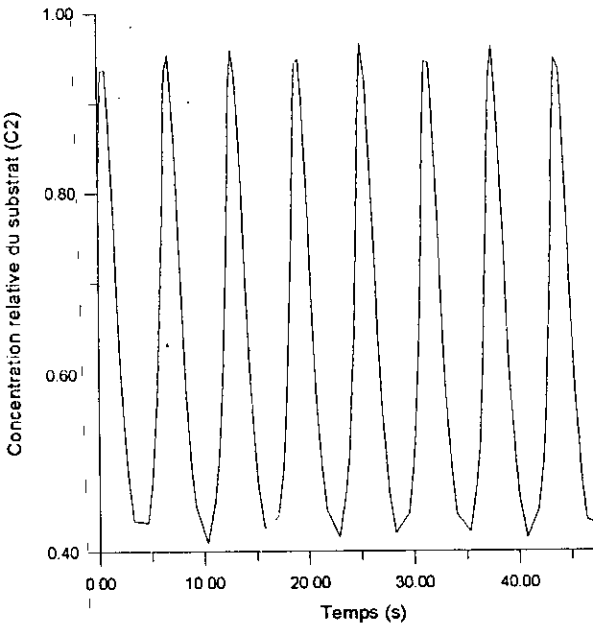
(a)



(c)



(b)



(d)

Fig. 5.4 réponses du système pour une valeur du flux $w = 1.20$. (a) et (b) cas où $C_1(0) = 0.1$ et $C_2(0) = 0.8$. (c) et (d) cas où $C_1(0) = 0.3$ et $C_2(0) = 0.8$.

- Dans l'intervalle supérieur à 1.20 et allant jusqu'à 2 concernant le flux w, le système revient à la stabilité et est indépendant des valeurs initiales des états. La figure (5.5) montre les réponse du système pour une valeur de w égale à 1.30. Plus le flux tend vers sa valeur supérieure 2, la concentration des cellules dans le réservoir tend à s'annuler rapidement.

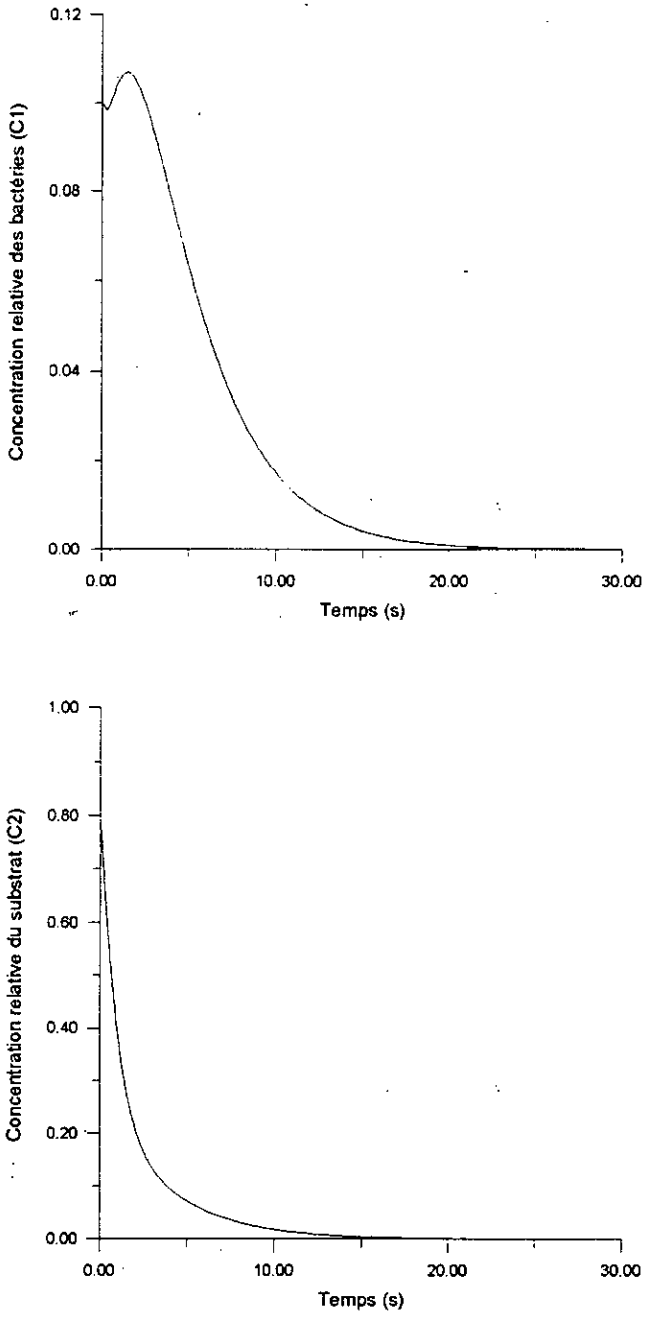


Fig. 5.5 réponses du système pour une valeur du flux $w = 1.30$.

H. Ungar [72] a défini deux types de problèmes pour la commande du bioréacteur. Dans le premier problème, il propose de commander le système dans sa région de stabilité, i.e., pour un flux variant de 0 à 0.75. Dans cet intervalle, un maximum de production de bactéries est atteint

pour $w = 0.75$. Pour le second problème, il propose de commander le système dans sa région d'instabilité, i.e., pour un flux supérieur à 0.75. Dans cet intervalle, un maximum de production de bactéries est atteint pour une valeur de w aux alentours de 1.20. Le fait que le système est instable rend ce problème plus difficile à résoudre que le premier.

5.4 COMMANDE NEURONALE DU BIOREACTEUR

5.4.1 COMMANDE DU BIOREACTEUR DANS SA PLAGE DE STABILITE

En régime statique (permanent), le bioréacteur est régi par le système d'équations issues du modèle dynamique (5.1) en annulant les dérivées respectives de C_1 et C_2, soit alors :

$$\begin{cases} C_1(1-C_2)\,exp\,(C_2/\gamma) - C_1 w = 0 \\ C_1(1-C_2)\,exp\,(C_2/\gamma)\dfrac{1+\beta}{1+\beta-C_2} - C_2 w = 0 \end{cases} \tag{5.4}$$

ce modèle est utilisé pour générer les états permanents du système. Le neuro-contrôleur doit fournir l'action de commande w qui devra mener la grandeur C_1 (t) d'un état initial à un état final désiré. Ce neuro-contrôleur ne tiendra pas compte de la dynamique du système vu que celui-ci est stable.

Pour ce type de régulation, nous avons opté pour une commande neuronale inverse spécialisée que nous avons présenté en détail dans le chapitre 2. Cette commande se prête bien pour la résolution de ce problème. En effet, le RNA a pour rôle de représenter la statique inverse du système. L'architecture d'apprentissage du neuro-contrôleur est présentée dans la figure (5.6). Une fois l'apprentissage effectué, on aura affaire à une commande en boucle ouverte du bioréacteur. Le neuro-contrôleur sera relié en cascade avec le système.

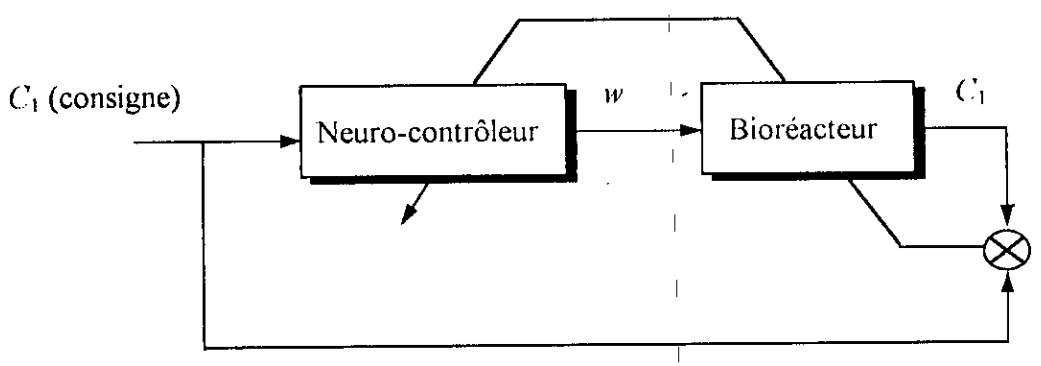


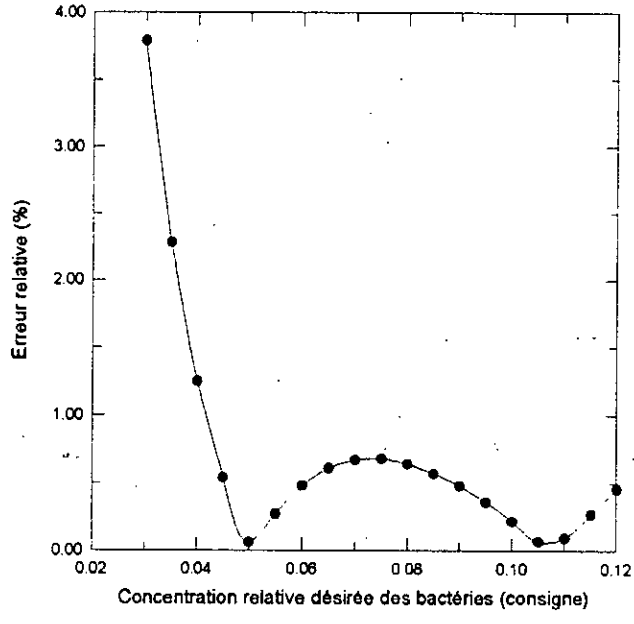
Fig. 5.6 architecture d'apprentissage du neuro-contrôleur pour la commande du bioréacteur dans sa plage de stabilité.

Le réseau neuronal choisi est mono entrée mono sortie possédant deux couches cachées ayant 5 neurones chacune. Chaque neurone possède une fonction d'activation du type tangente hyperbolique.

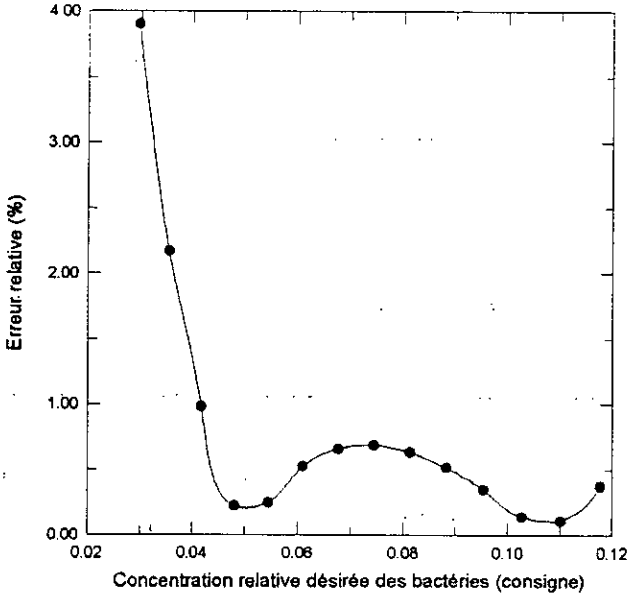
A. Apprentissage et généralisation

L'entrée (consigne) et la sortie (action de commande) du RNA varient dans les intervalles respectifs]0, 0.12] et]0, 0.75]. L'apprentissage s'est effectué pendant près de 30000 itérations aboutissant à une erreur quadratique de l'ordre de 6.78×10^{-6} sur l'ensemble des exemples d'entraînement. La figure (5.7) illustre les résultats d'apprentissage et de généralisation sur des exemples non appris. L'ensemble des erreurs relatives est en majorité inférieur à 1%.

La figure (5.8), quant à elle, nous montre graphiquement l'évolution des erreurs relatives commises par le RNA sur l'action de commande *w*. Celles-ci sont en majorité inférieures à 2%.



(a)



(b)

Fig. 5.7 (a) résultats d'apprentissage du RNA, (b) résultats de généralisation du RNA.

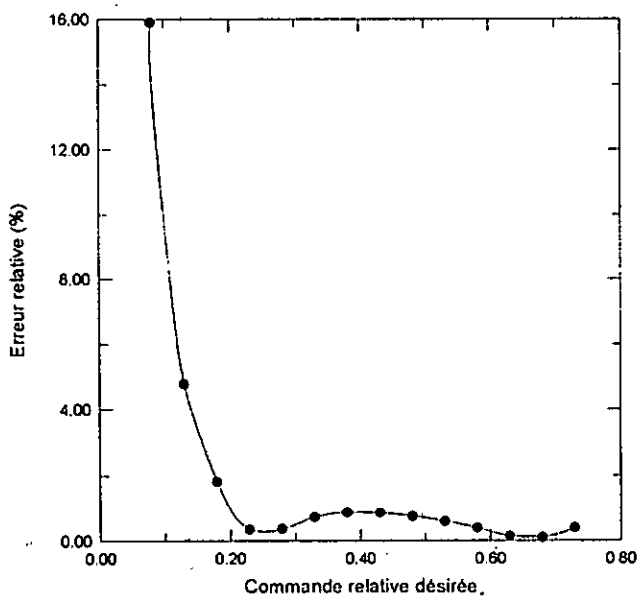


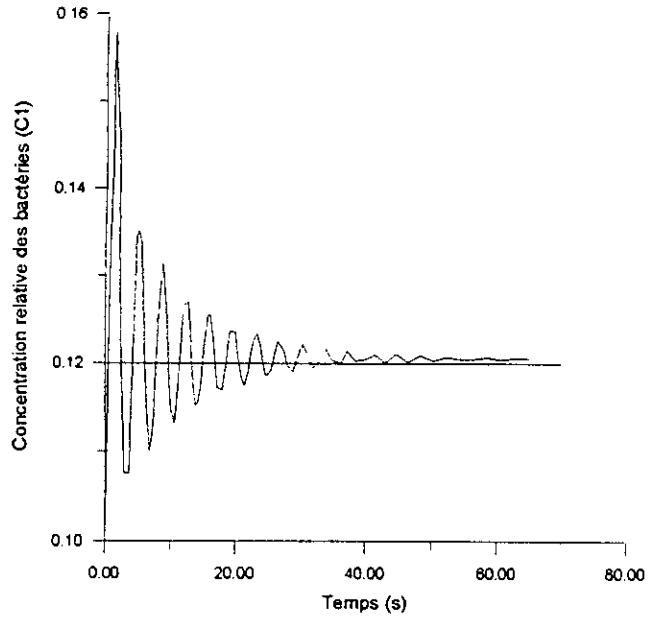
Fig. 5.8 erreurs commises par le RNA sur l'action de commande w.

B. Résultats de simulation

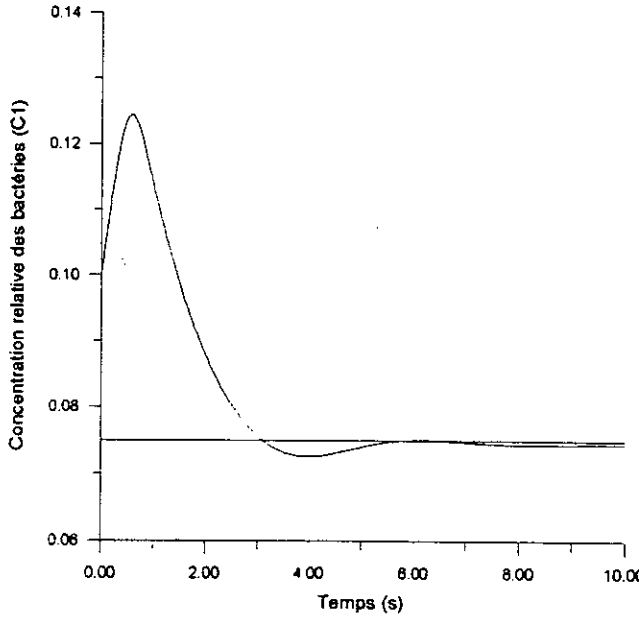
Les réponses du système sont obtenues à partir de l'action de commande délivrée par le neuro-contrôleur due à une concentration de bactéries désirée (valeur de consigne). En un premier temps, on a testé l'aptitude du RNA à conduire le système vers n'importe quelle état voulu (différentes valeurs de consigne). La figure (5.9) montre les réponses du bioréacteur pour deux valeurs de consigne distinctes, à savoir $C_1 = 0.12$ et $C_1 = 0.075$. Dans les deux cas la concentration relative des bactéries se stabilise autour de sa valeur désirée. On peut aussi remarquer la stabilité de la concentration relative du substrat.

En un second temps, on a testé l'aptitude du neuro-contrôleur à palier au problème des conditions initiales. Le RNA devrait être insensible aux variations de celles-ci pour obtenir une bonne qualité de réglage. La figure (5.10) nous présente les réponses du système pour une valeur de consigne $C_1 = 0.11$ pour deux conditions initiales différentes, à savoir C_1 (0) = 0.2 et C_2 (0) = 0.6, puis C_1 (0) = C_2 (0) = 0.4. En effet, le neuro-contrôleur s'est montré insensible à la variation des conditions initiales.

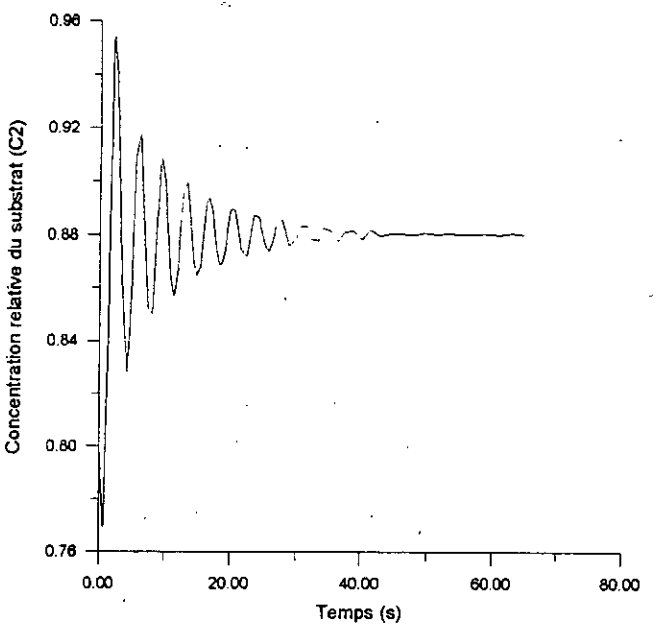
En un troisième temps, on a testé l'aptitude du neuro-contrôleur à palier au problème des perturbations externes. On entend par ceci, la variation brusque de l'une des concentrations (bactéries ou substrat) à l'intérieur du bioréacteur. On citera par exemple l'ajout involontaire d'une quantité de substrat ou de bactéries dans le réservoir. La figure (5.11) nous illustre les réponses du système pour une valeur de consigne $C_1 = 0.12$. Une fois que le système ait atteint son régime permanent, une perturbation de l'ordre de 50% est provoquée à t = 60s sur la concentration des bactéries et sur la concentration du substrat. L'action de commande w étant maintenue par le neuro-contrôleur, les concentrations respectives reviennent à leurs valeurs désirées.



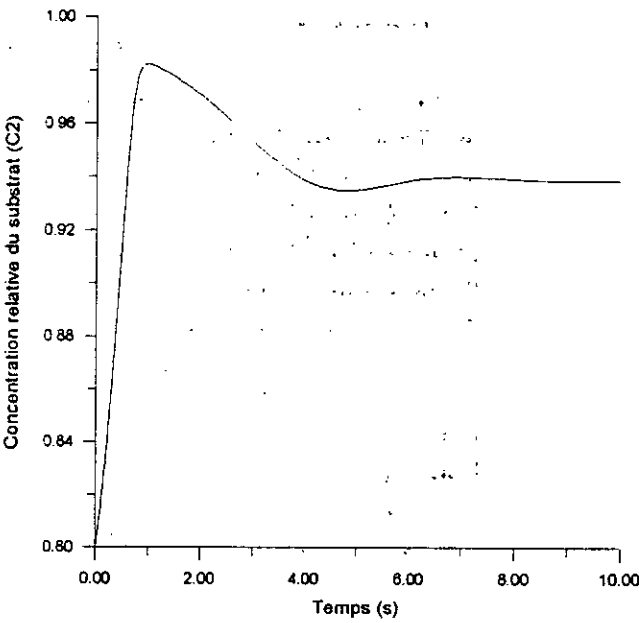
(a)



(c)

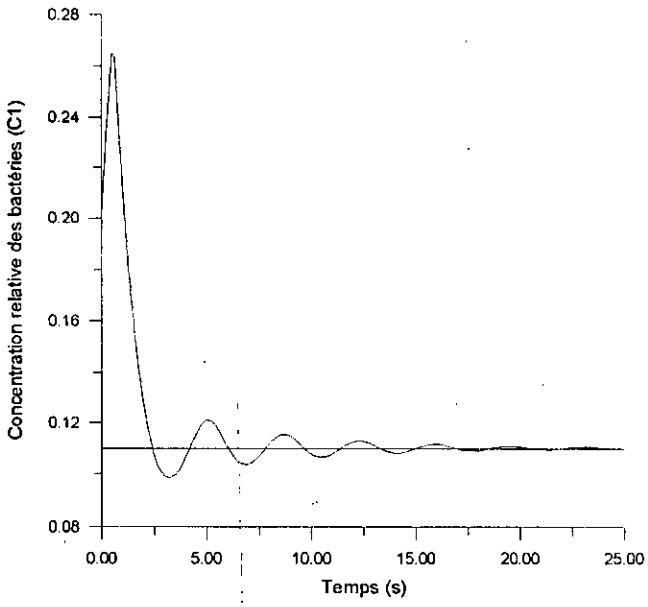


(b)

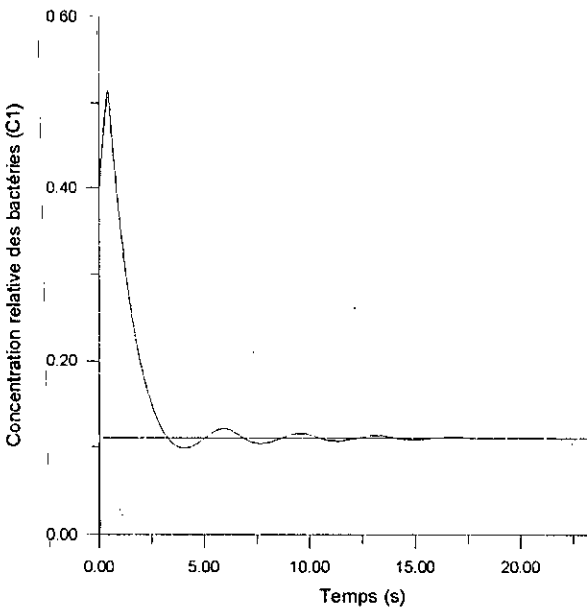


(d)

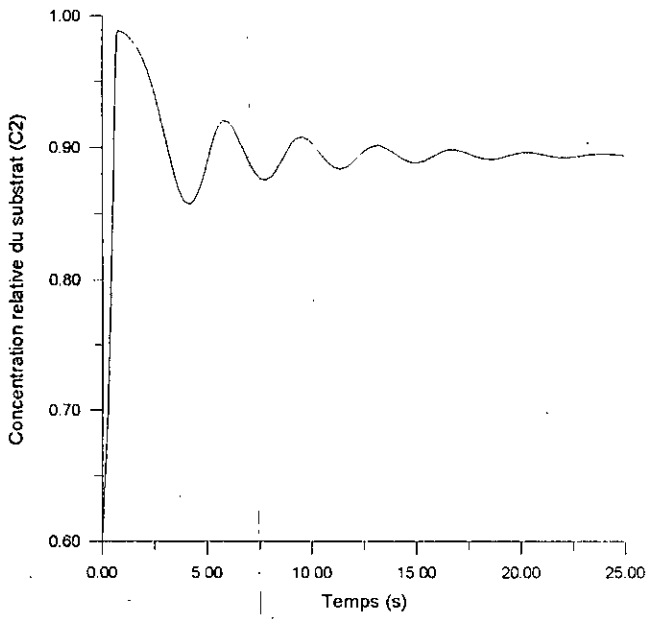
Fig. 5.9 réponses du système pour deux valeurs de consignes différentes : (a) et (b) $C_1 = 0.12$, (c) et (d) $C_1 = 0.075$.



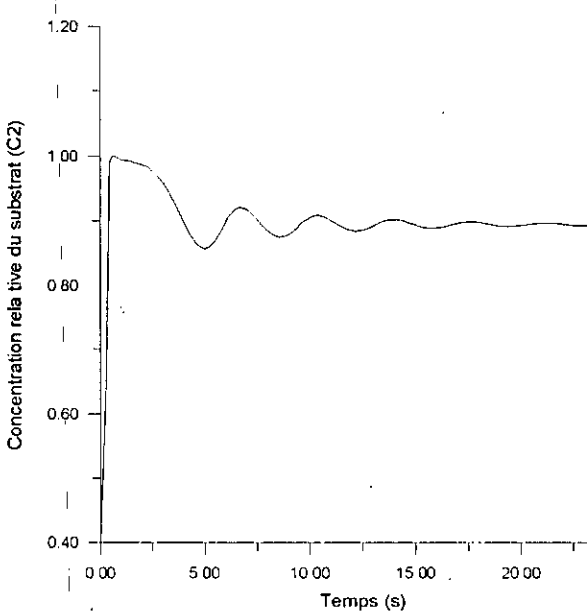
(a)



(c)

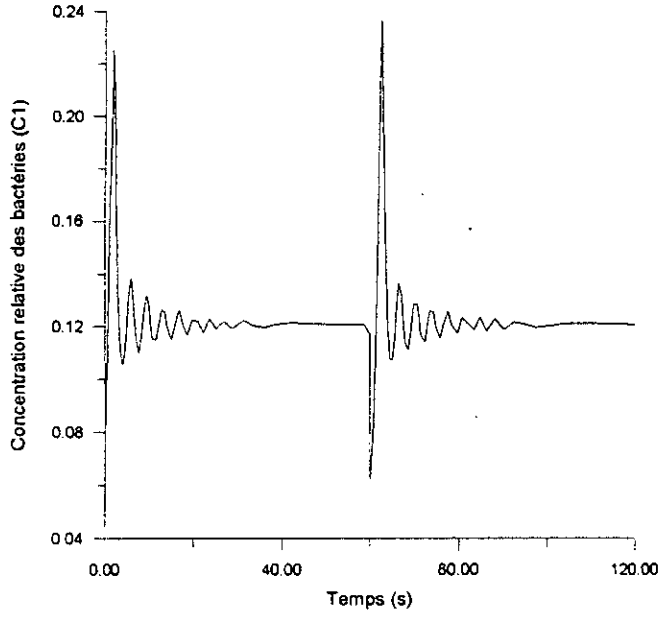


(b)

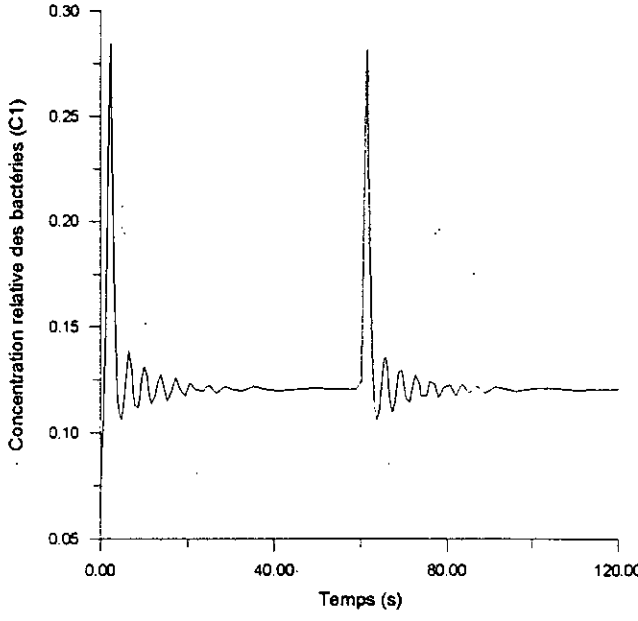


(d)

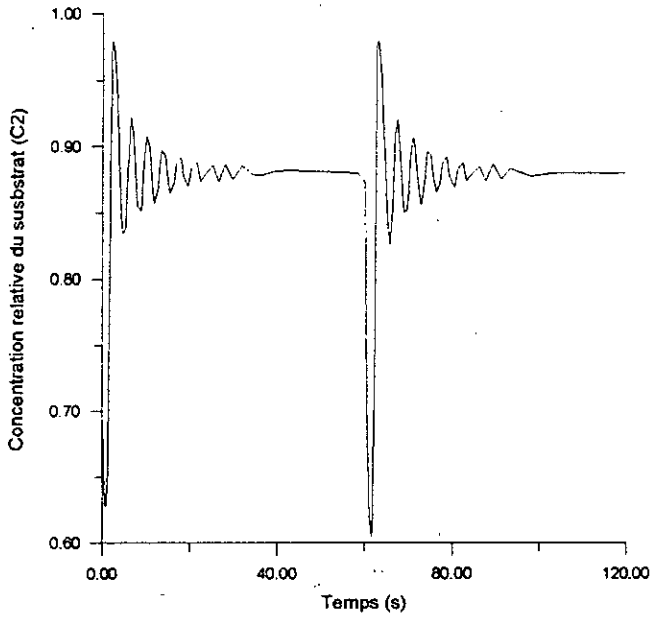
Fig. 5.10 réponses du système pour deux conditions initiales distinctes : (a) et (b) C_1 (0) = 0.1, C_2 (0) = 0.6, (c) et (d) C_1 (0) = C_2 (0) = 0.4.



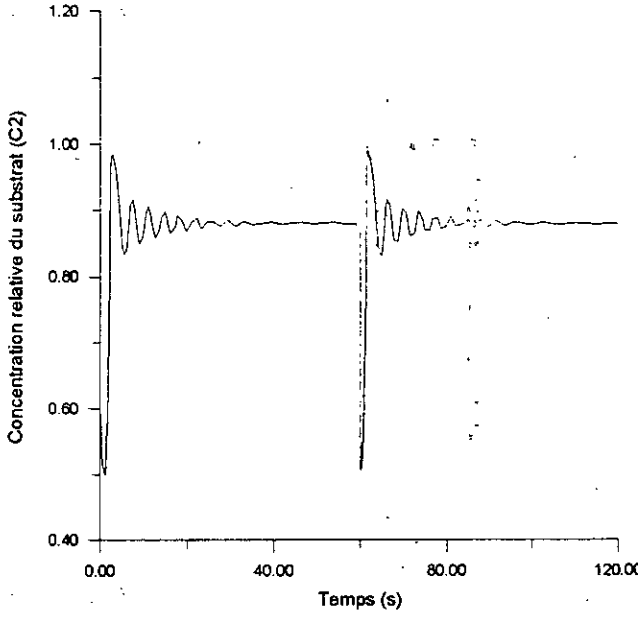
(a)



(c)



(b)



(d)

Fig. 5.11 réponses du système aux perturbations externes : (a) et (b) perturbation de 50% sur C_1 , (c) et (d) perturbation de 50% sur C_2.

5.4.2 COMMANDE DU BIOREACTEUR DANS SA PLAGE D'INSTABILITE

Dans cette deuxième partie de notre travail, il s'agit de concevoir une commande neuronale qui a le rôle de rendre le système stable dans un intervalle de fonctionnement où il est instable, c'est à dire pour une variation de l'entrée w (flux) allant de 0.75 à 1.20. La conception de cette commande s'avère indispensable, puisque le maximum de production de bactéries est atteint pour un flux w aux alentours de 1.20. De plus, l'évolution dans le temps des différentes concentrations dépend de leurs états initiaux. Ceci étant, il a fallut trouver une technique judicieuse qui non seulement doit stabiliser le système, mais aussi qui doit être réalisable. Les autres techniques de commande neuronale qu'on a présenté dans le chapitre 2 se sont révélées inadéquates du point de vue applicabilité.

L'idée se présente comme suit : pour pouvoir stabiliser la concentration des bactéries, et par la même, la concentration du substrat à l'intérieur du bioréacteur, il faudra trouver un contrôleur capable de délivrer une action de commande w qui mènera l'état du système d'un état quelconque à un état désiré et le maintenir dans cette position. En effet, à partir de la première équation du modèle discret (5.3) :

$$C_1(k+1) = C_1(k) + T\left(-C_1(k)\, w(k) + C_1(k)\left(1 - C_2(k)\right) exp\left(C_2(k)/\gamma\right)\right) \qquad (5.5)$$

on pose $C_1(k+1) = C_{10}$, où C_{10} est la concentration de bactéries désirée. Par ce raisonnement, on veut trouver l'action de commande $w(k)$ qui mènera l'état du système à l'instant k vers l'état C_{10} à l'instant $k+1$. Soit alors :

$$w(k) = \left(1 - C_2(k)\right) exp\left(C_2(k)/\gamma\right) + \frac{C_1(k) - C_{10}}{T\, C_1(k)} \qquad (5.6)$$

Cette loi de commande est considérée comme un retour d'état nonlinéaire. La relation (5.6) peut être exprimée sous la forme :

$$w(k) = f_1\left(C_2(k)\right) + f_2\left(C_1(k)\right) \qquad (5.7)$$

où f_1 et f_2 sont deux fonctions nonlinéaires. Pour reproduire ces nonlinéarités, on utilisera deux réseaux neuronaux distincts. L'ensemble des deux réseaux formera notre contrôleur neuronal.

L'architecture de commande est présentée dans la figure (5.12). Les RNA devraient estimer les valeurs de f_1 et f_2 après apprentissage. La sommation est effectuée par un simple sommateur ou par un neurone à deux entrées ayant des pondérations unité et une fonction d'activation linéaire.

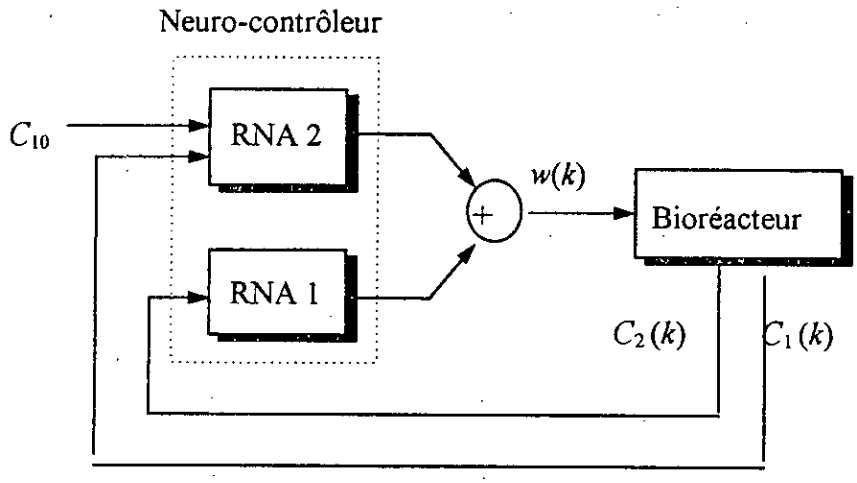


Fig. 5.12 architecture de la commande neuronale dans le cas où le système est instable.

A. Apprentissage et généralisation

En premier lieu on a commencer par entraîner le RNA1 en utilisant des entrées / sorties issues du modèle de la fonction nonlinéaire f_1. Le réseau utilisé possède une entrée (C_2 (k)) et une sortie. Il est doté de deux couches cachées à cinq neurones chacune. Les neurones de chaque couche possèdent une fonction d'activation tangente hyperbolique ainsi qu'une entrée bias. La plage de variation de l'entrée (concentration relative du substrat) va de 0 à 1. L'erreur quadratique obtenue après environ 10000 itérations était de 1.67×10^{-4}. La figure (5.13) montre les erreurs relatives commises par le RNA1 sur les exemples d'apprentissage et ceux de la généralisation, celles-ci sont inférieures à 1% pour les données d'entraînement et en majorité inférieures à 2% pour les données de la généralisation.

Le RNA2 est entraîné en utilisant des entrées / sorties issues du modèle de la fonction nonlinéaire f_2. Le réseau choisi possède deux entrées (C_{10} et C_1 (k)) et une sortie. Il est muni de deux couches cachées à cinq neurones chacune ayant une fonction d'activation tangente hyperbolique. En fait, dans cette étape on a considéré deux cas différents dans le but de faire une étude plus détaillée du problème.

En effet, d'après le modèle (5.6) la nonlinéarité f_2 dépend de la valeur de C_{10} ainsi que de la constante de temps T. Il a fallut donc préparer le RNA2 avec trois fichiers de pondérations différents à savoir, un apprentissage dans le cas où $C_{10} = 0.20$ et $T = 0.5$s, puis un apprentissage dans le cas où $C_{10} = 0.15$ et T = 0.5s (variation de C_{10}) et enfin un apprentissage dans le cas où $C_{10} = 0.20$ et $T = 1$s (variation de T). Les résultats d'apprentissage et de généralisation du premier cas sont illustrés dans la figure (5.14). L'apprentissage a duré près de 30000 itérations aboutissant à une erreur quadratique de 6.67×10^{-4}. La figure (5.15) quant à elle, présente les résultats d'apprentissage et de généralisation du deuxième cas. L'entraînement a duré près de 10000 itérations aboutissant à une erreur quadratique de 2.3×10^{-3}. Enfin, la figure (5.16) nous montre les résultats d'apprentissage et de généralisation du troisième cas. L'apprentissage a duré près de 20000 itérations pour aboutir à une erreur quadratique de 3×10^{-4}. Il faut noter que dans les trois cas le réseau neuronal a trouvé des

difficultés à apprendre certains exemples, c'est pour cette raison qu'on trouve des erreurs relatives qui vont jusqu'à 20% ou un peu plus.

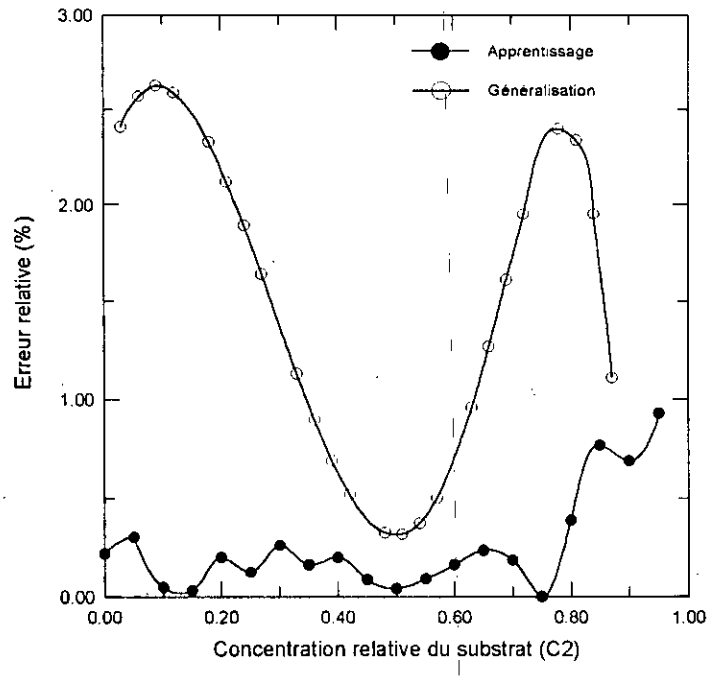


Fig. 5.13 résultats d'apprentissage et de généralisation du RNA1 pour approximer la fonction nonlinéaire f_1.

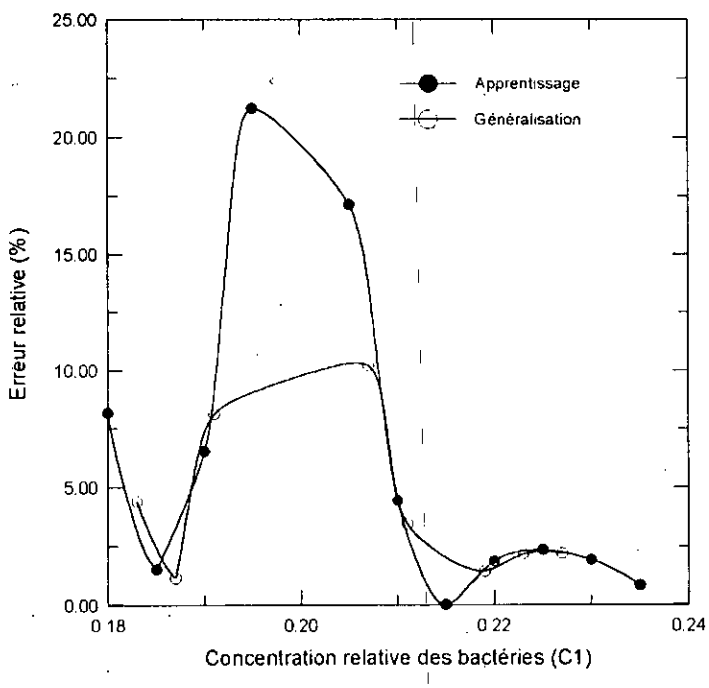


Fig. 5.14 résultats d'apprentissage et de généralisation du RNA2 pour approximer la fonction nonlinéaire f_2. Cas où $C_{10} = 0.20$ et $T = 0.5$ s.

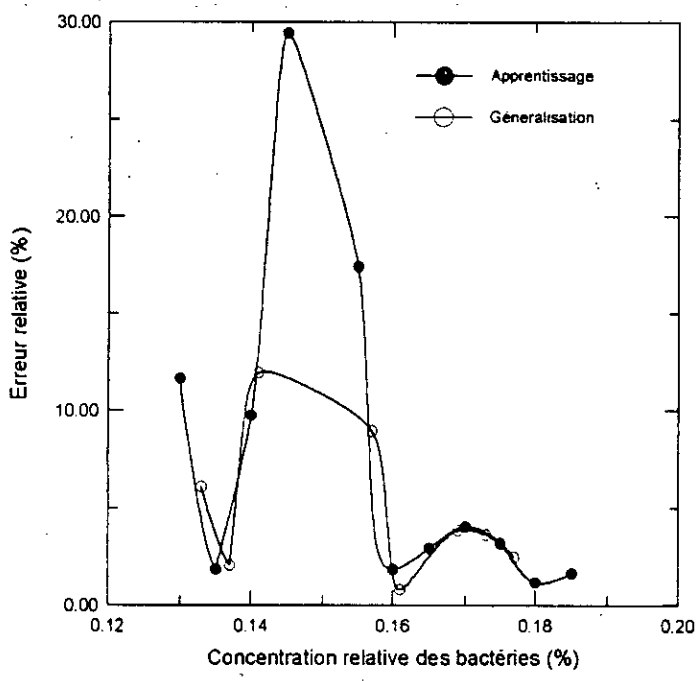


Fig. 5.15 résultats d'apprentissage et de généralisation du RNA2 pour approximer la fonction nonlinéaire f_2. Cas où $C_{10} = 0.15$ et $T = 0.5$ s.

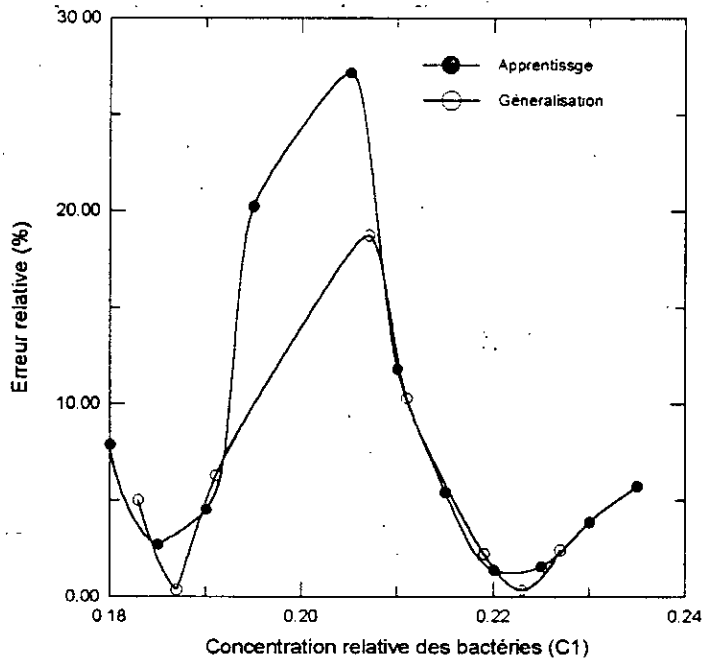


Fig. 5.16 résultats d'apprentissage et de généralisation du RNA2 pour approximer la fonction nonlinéaire f_2. Cas où $C_{10} = 0.20$ et $T = 1$s.

B. Résultats de simulation

D'après sa structure, le neuro-contrôleur doit agir à la fin de chaque période T (constante d'échantillonnage). Bien que le système chimique est relativement lent, et pour éviter tout problème pratique de retard provoqué par le neuro-contrôleur, on a fait en sorte à ce que ce dernier agisse à la fin de chaque période $T_0 > T$ (T= 0.01s) et maintienne son action de commande tout au long de cette période.

Comme déjà mentionné, le RNA2 a été d'abord entraîné pour une valeur de $T_0 = 50T$ = 0.5s. Dans cette phase initiale, le neuro-contrôleur a pour rôle de stabiliser le système autour de C_{10} = 0.20. Les figures (5.17) et (5.18) montrent les réponses du bioréacteur ainsi que l'action de commande délivrée par le régulateur neuronal pour deux valeurs de conditions initiales différentes, à savoir C_1 (0) = 0.18 et C_2 (0) = 0.70 puis C_1 (0) = 0.22 et C_2 (0) = 0.60. On peut remarquer que le résultat est assez convainquant puisque le neuro-contrôleur arrive à stabiliser la concentration des bactéries à la valeur désirée avec une bonne précision. De plus, il est insensible aux variations des conditions initiales.

Dans la seconde étape, on utilise le deuxième fichier poids du RNA2 pour tester l'aptitude du neuro-contrôleur à stabiliser le système autour du point C_{10} = 0.15 avec des conditions initiales C_1 (0) = 0.13 et C_2 (0) = 0.60. La figure (5.19) présente les réponses du système dans ce cas.

La troisième étape de la simulation consiste à étudier l'influence de la constante de temps T_0 sur le comportement du neuro-contrôleur. Cette fois-ci, on prend $T_0 = 100T$ = 1s. Les figures (5.20) et (5.21) illustrent les réponses du bioréacteur ainsi que l'action de commande délivrée par le régulateur neuronal pour deux valeurs de conditions initiales différentes, à savoir C_1 (0) = 0.18 et C_2 (0) = 0.70 puis C_1 (0) = 0.22 et C_2 (0) = 0.60. On peut remarquer que le neuro-contrôleur peut aussi stabiliser la concentration des bactéries autour de sa valeur désirée. Cependant, l'influence de T_0 apparaît sur la dynamique du système puisque celui-ci devient moins rapide et est sujet à plus d'oscillations, vu que l'action de commande adéquate prend plus de temps à venir. La précision du réglage quant à elle reste inchangée. Il faut ajouter à cela que le fait de prendre en compte le régime transitoire par le neuro-contrôleur, la réponse du système devient plus rapide que dans le cas de la commande en régime statique.

Dans la suite de notre travail, on a testé l'aptitude du neuro-contrôleur à palier au problème des perturbations externes, c'est à dire à la variation brusque de l'une des concentrations (bactéries ou substrat) à l'intérieur du bioréacteur. La figure (5.22) nous illustre les réponses du système pour une valeur de consigne C_{10} = 0.20. Une fois que le système ait atteint son régime permanent, une perturbation est provoquée à t = 8s sur la concentration des bactéries (une augmentation jusqu'à 0.22) et sur la concentration du substrat (une augmentation jusqu'à 0.60). On peut remarquer qu'à ce moment-là le neuro-contrôleur détecte cette déviation et déclenche une action de commande susceptible de reconduire la concentration des bactéries à sa valeur désirée.

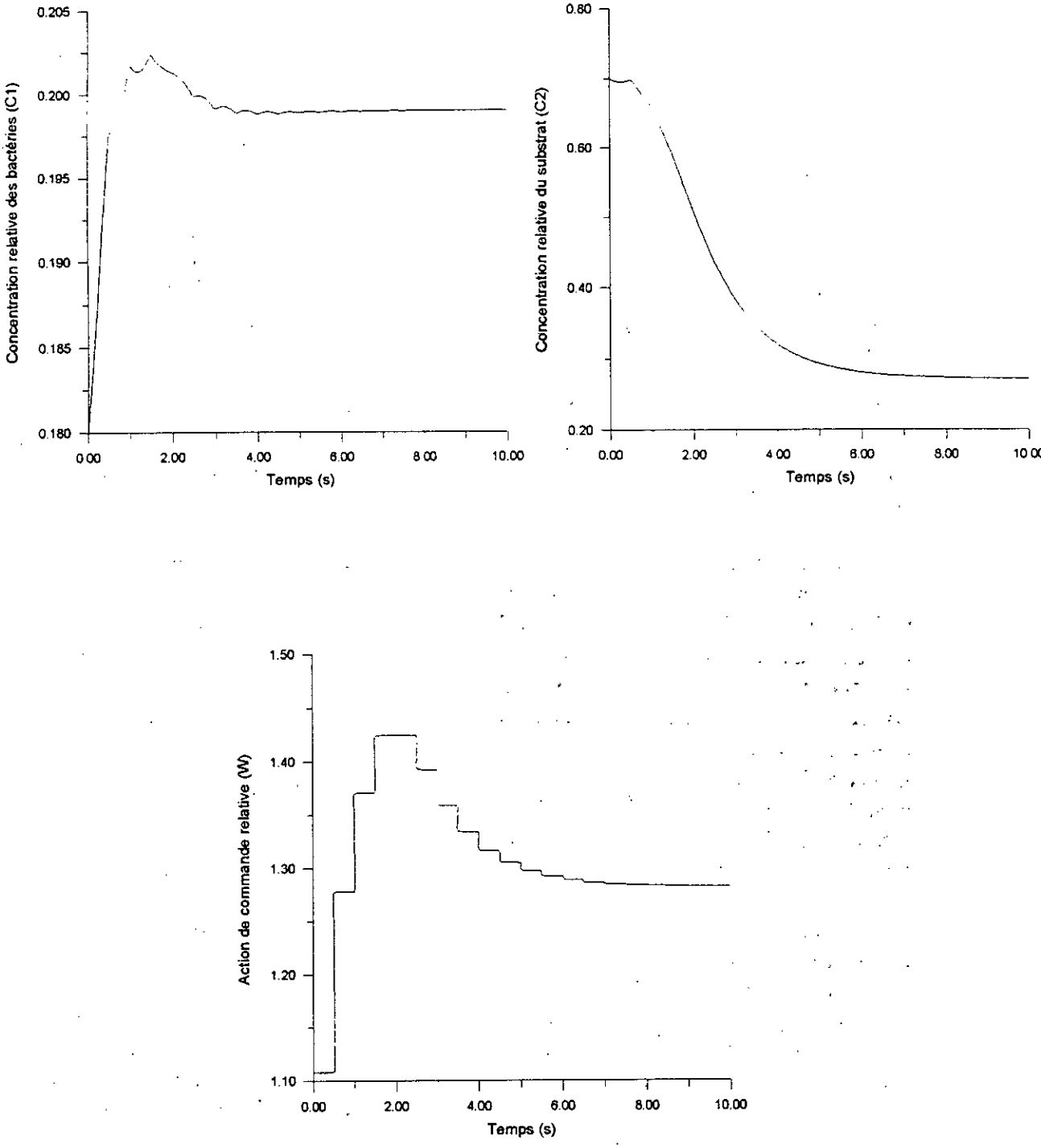
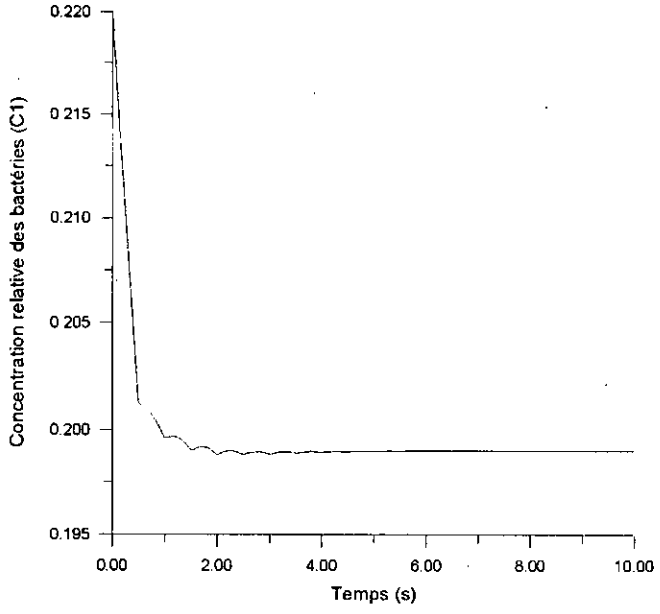
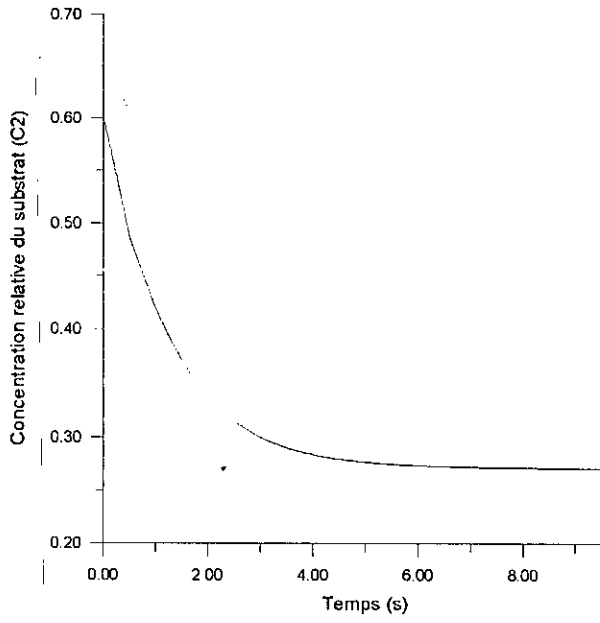


Fig. 5.17 réponses du système ainsi que l'action de commande délivrée par le neuro-contrôleur dans le cas où $C_{10} = 0.20$, $C_1(0) = 0.18$, $C_2(0) = 0.70$ et $T_0 = 0.5$s.





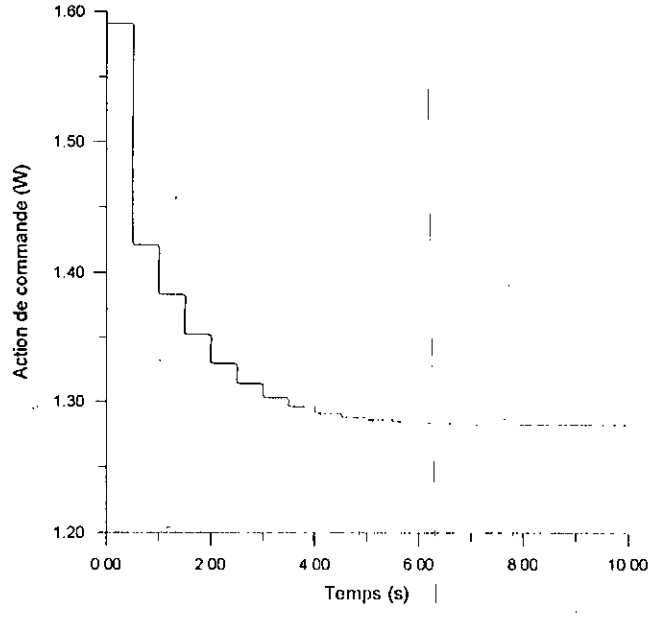


Fig. 5.18 réponses du système ainsi que l'action de commande délivrée par le neuro-contrôleur dans le cas où $C_{10} = 0.20$, $C_1(0) = 0.22$, $C_2(0) = 0.60$ et $T_0 = 0.5$s.

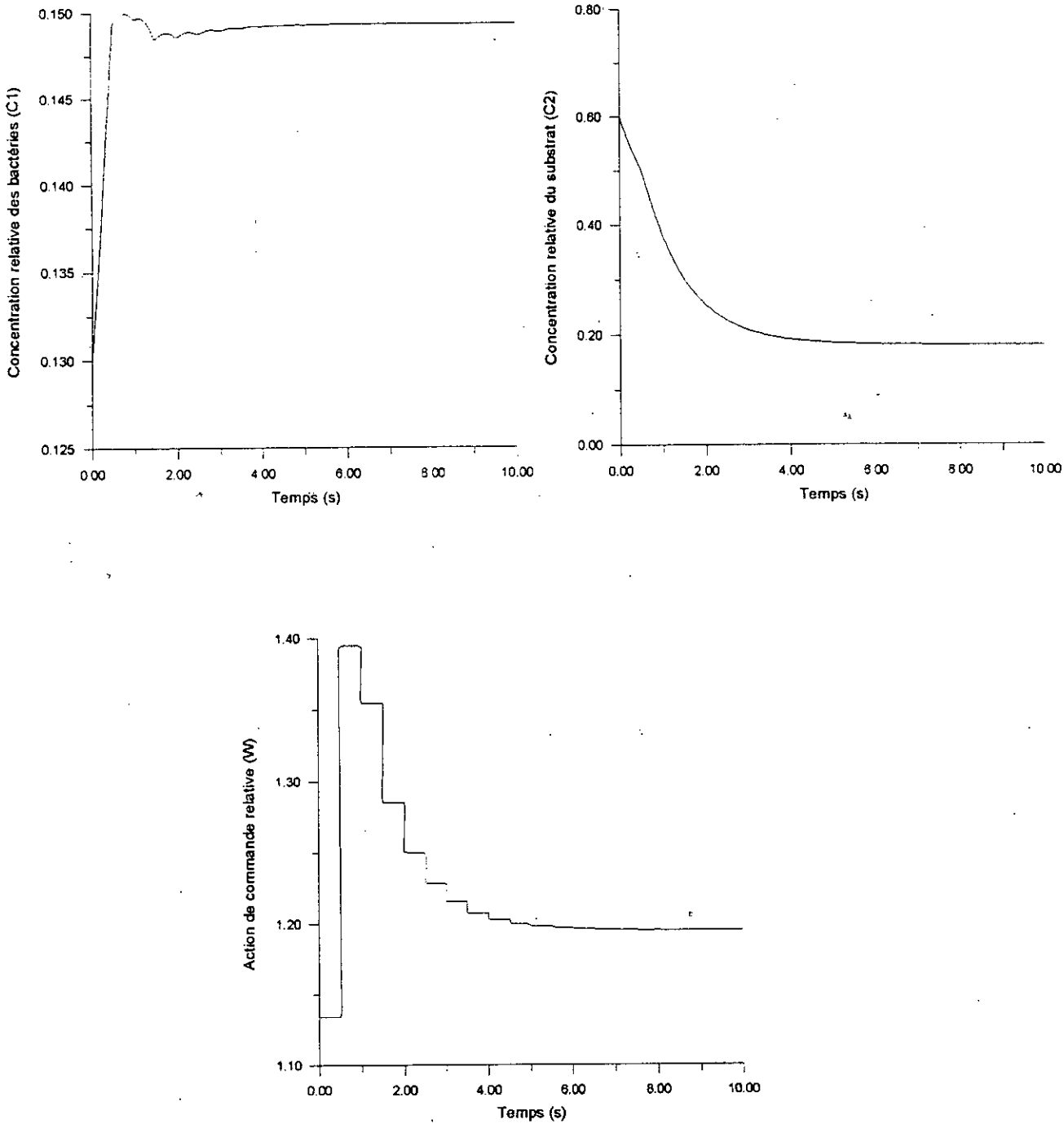
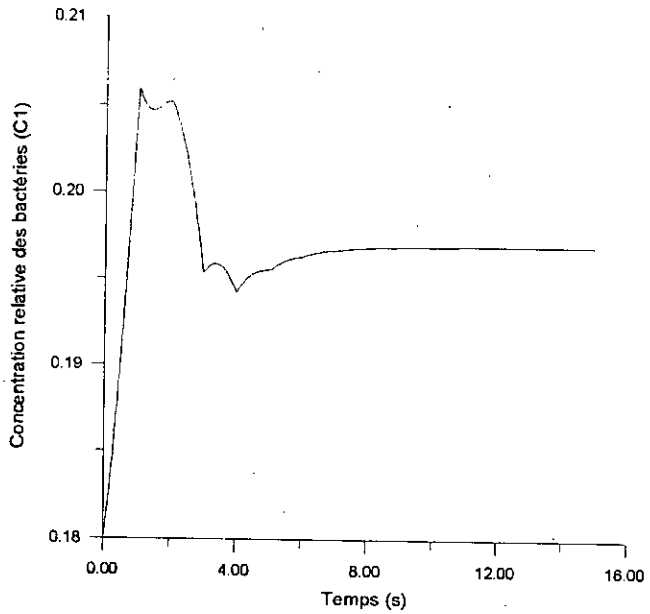
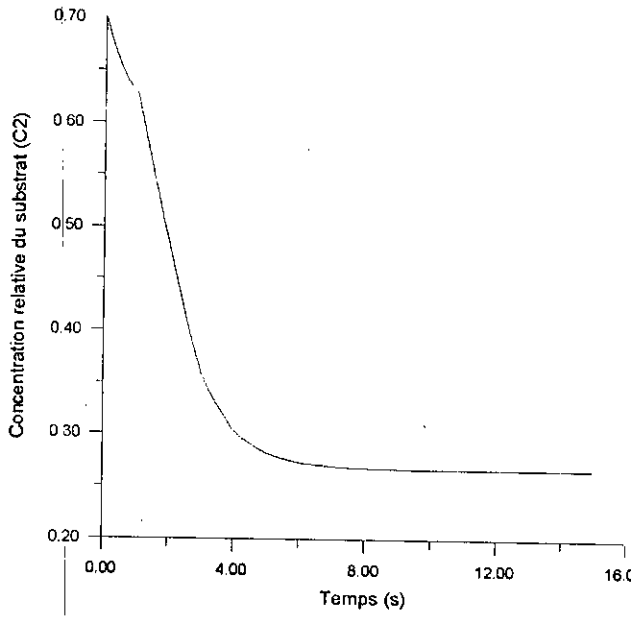


Fig. 5.19 réponses du système ainsi que l'action de commande délivrée par le neuro-contrôleur dans le cas où $C_{10} = 0.15$, $C_1(0) = 0.13$, $C_2(0) = 0.60$ et $T_0 = 0.5$s.





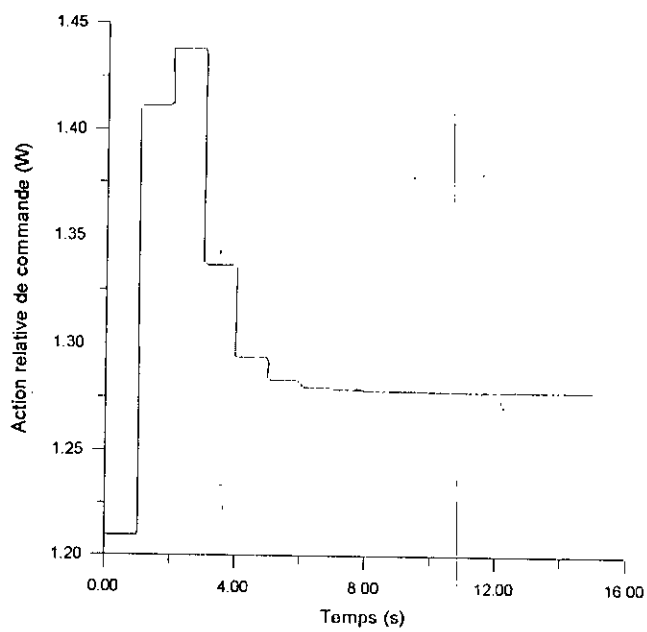
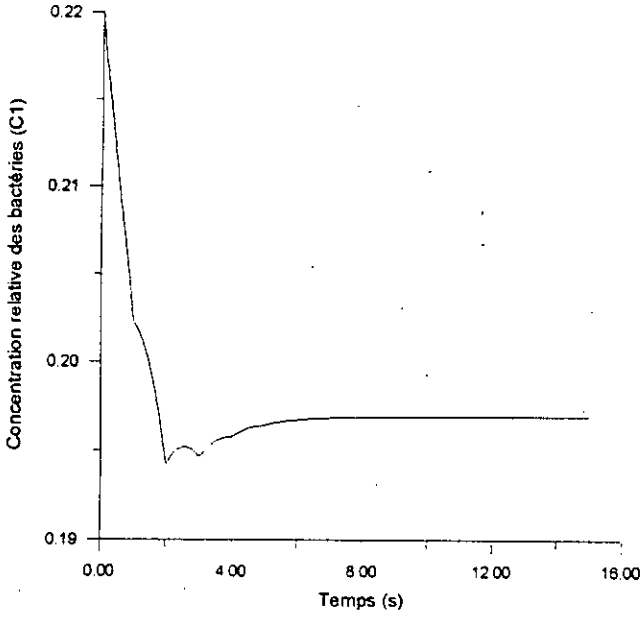
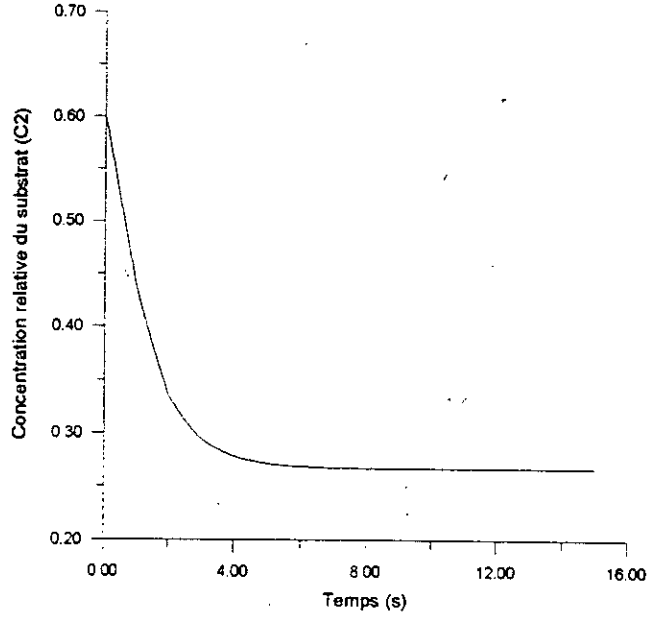


Fig. 5.20 réponses du système ainsi que l'action de commande délivrée par le neuro-contrôleur dans le cas où $C_{10} = 0.20$, $C_1(0) = 0.18$, $C_2(0) = 0.70$ et $T_0 = 1$s.





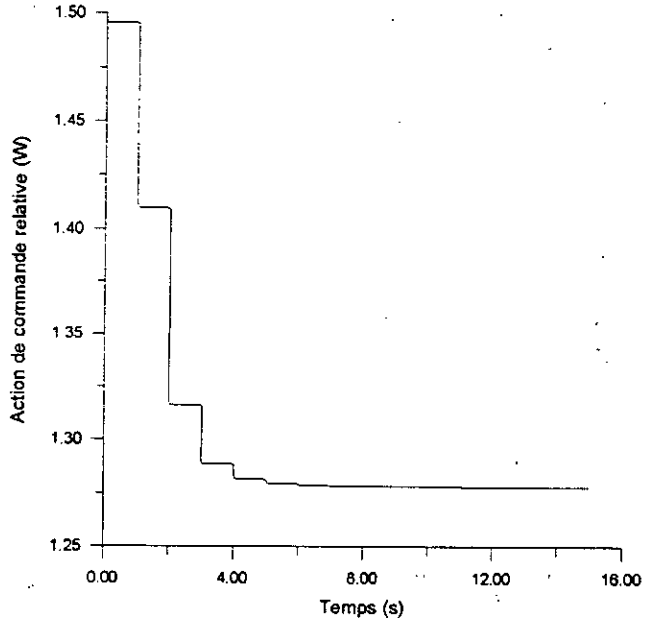
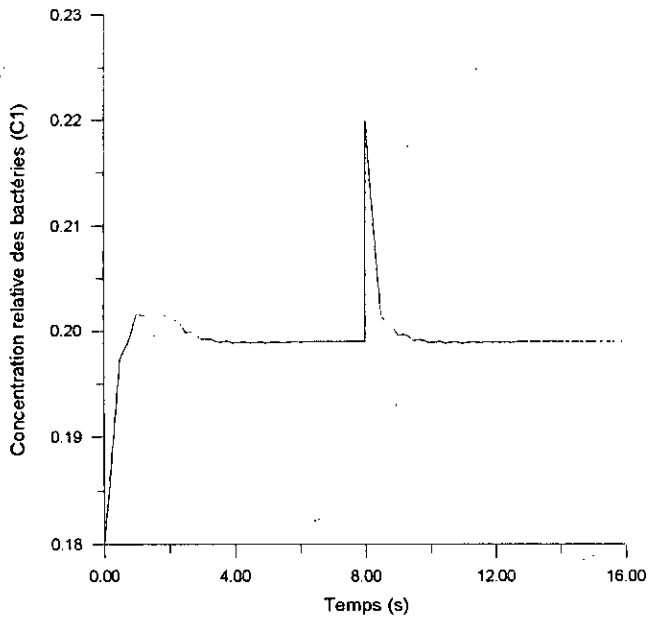
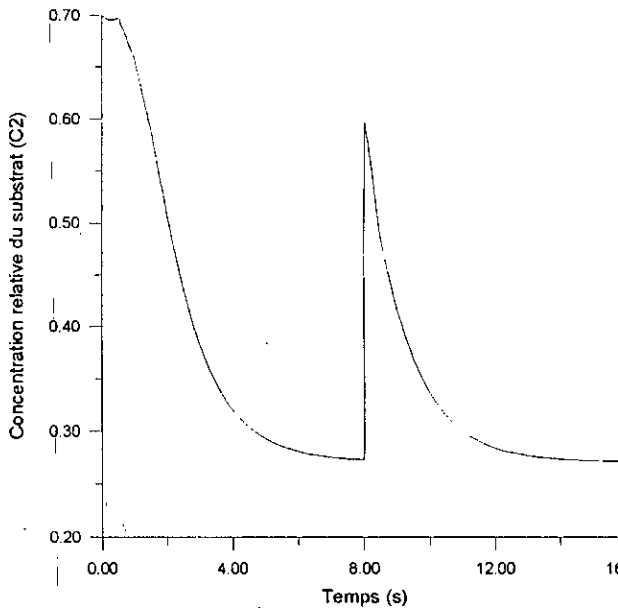


Fig. 5.21 réponses du système ainsi que l'action de commande délivrée par le neuro-contrôleur dans le cas où $C_{10} = 0.20$, $C_1(0) = 0.22$, $C_2(0) = 0.60$ et $T_0 = 1$s.





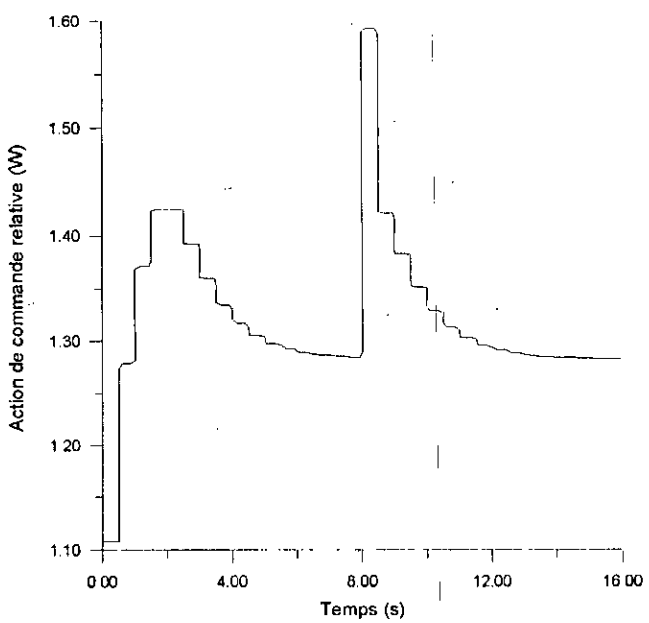


Fig. 5.22 réponses du système perturbé à t = 8s ainsi que l'action de commande délivrée par le neuro-contrôleur dans le cas où $C_{10} = 0.20$, $C_1(0) = 0.22$, $C_2(0) = 0.60$ et $T_0 = 1$s.

5.5 COMMANDE FLOUE DU BIOREACTEUR

5.5.1 COMMANDE DU BIOREACTEUR DANS SA PLAGE DE STABILITE

De même que dans le cas de la commande neuronale, le modèle (5.4) du bioréacteur sert à générer les états permanents du système dans sa plage de stabilité, i.e., pour une action de commande w variant de 0 à 0.75. A partir des intervalles de variation de l'entrée du système (la commande) et de sa sortie (la concentration relative des bactéries), on doit construire les fonctions d'appartenance d'entrée et de sortie de notre système flou servant à contrôler le bioréacteur. En effet, le contrôleur flou a pour rôle de reproduire la statique inverse du bioréacteur (voir fig. 5.23).

Dans ce cas, le contrôleur flou possède une entrée (C_1) et une sortie (w). Pour se faire, on a utilisé 7 fonctions d'appartenance en entrée et 7 autres en sortie. Ceci a abouti à la formulation de 7 règles du type:

« si C_1 est A alors, w est B »

avec : A et B des ensembles flous variants de très petit à très grand.

La figure (5.24) nous présente graphiquement les résultats du contrôleur flou sous forme d'erreurs relatives commises sur l'action de commande w. Nos testes se sont effectués sur un intervalle de w allant de 0.1 à 0.75 et donc une variation de la concentration relative des bactéries allant de 3×10^{-2} à 0.12. On peut remarquer aisément que la majorité des exemples sur lesquels on a testé le contrôleur flou ont présenté une erreur relative inférieure à 1%. La reproduction de la statique inverse du système par le contrôleur flou est donc assez appréciable.

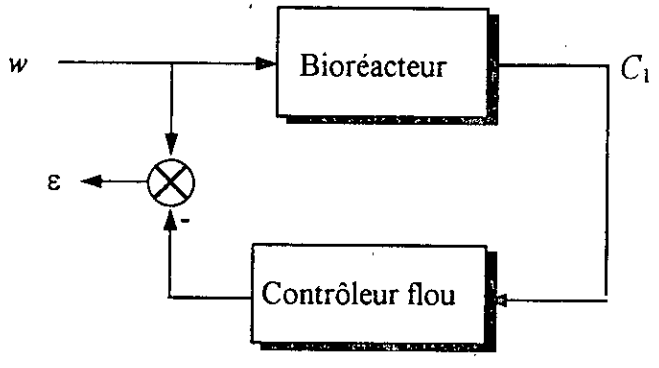


Fig. 5.23 principe de fonctionnement du contrôleur flou pour la commande du bioréacteur en régime statique.

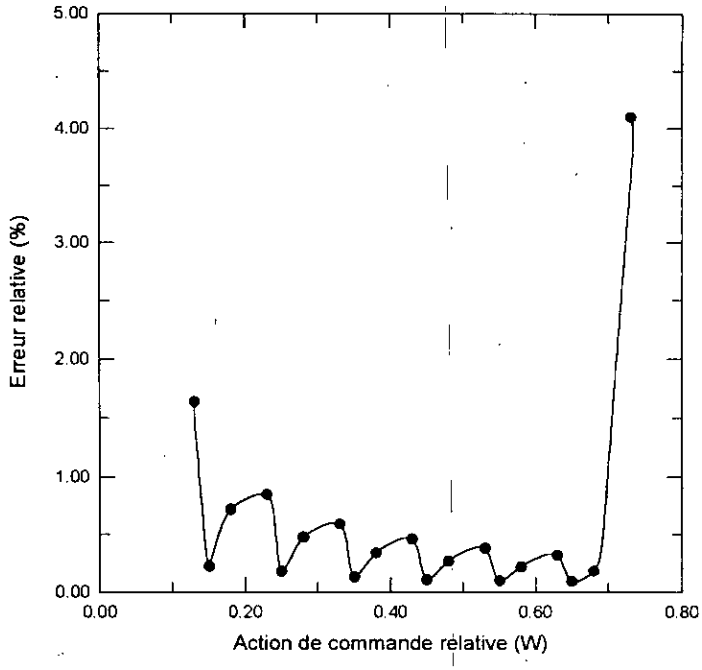


Fig. 5.24 erreurs relatives commises par le contrôleur flou sur l'action de commande dans le cas du régime statique.

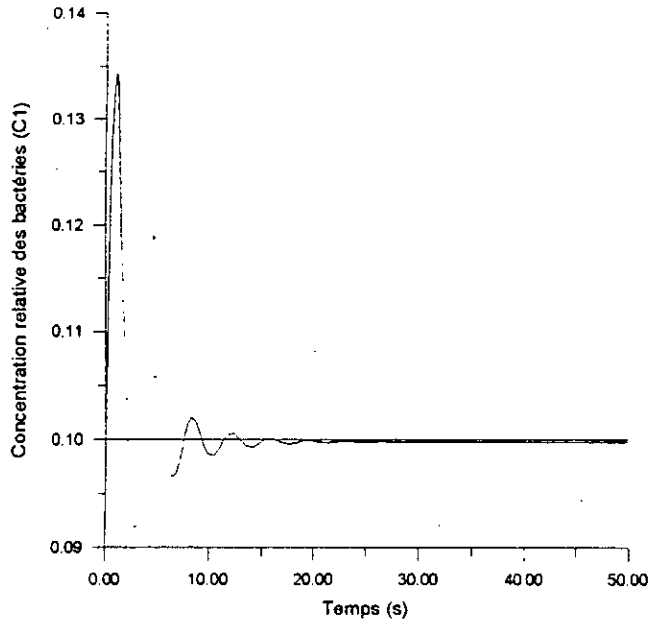
Résultats de simulation

Après avoir testé la qualité de reproduction de la statique inverse du bioréacteur par le contrôleur flou, ce dernier a été relié en cascade avec le système afin de délivrer l'action de commande adéquate qui conduira la concentration des bactéries vers sa valeur désirée (la consigne). C'est au fait une commande en boucle ouverte.

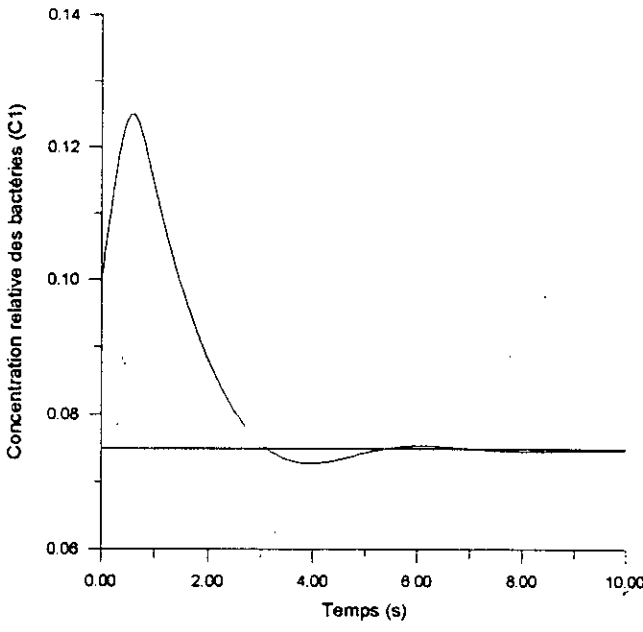
La première étape consistait à tester l'aptitude du contrôleur flou à mener l'état du système vers n'importe quelle valeur de consigne entre 0 et 0.12 qui représente un maximum de production de bactéries dans la plage de stabilité. La figure (5.25) présente les réponses du système pour deux valeurs de consigne différentes, à savoir $C_1 = 0.10$ et $C_1 = 0.075$.

La deuxième étape consistait à tester l'aptitude du contrôleur flou à conduire le système vers sa valeur de consigne quelles que soient les conditions initiales. La figure (5.26) illustre les réponses du système pour deux valeurs de conditions initiales différentes, à savoir $C_1(0) = 0.2$, $C_2(0) = 0.6$ et $C_1(0) = C_2(0) = 0.4$.

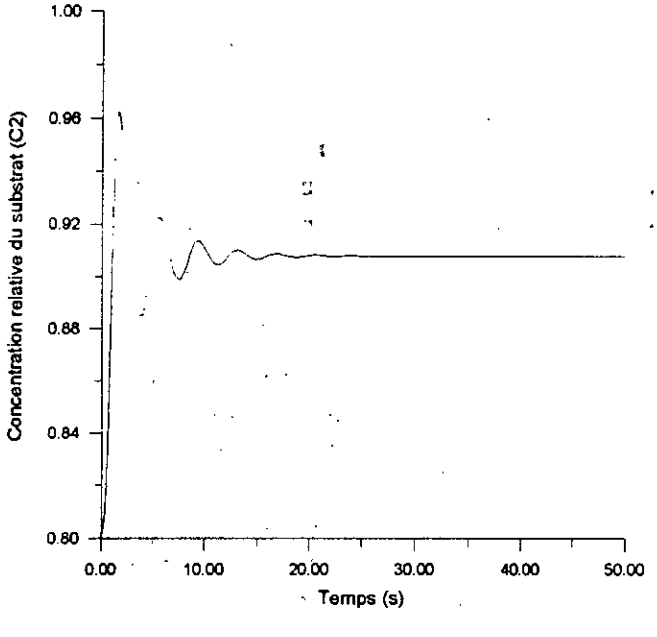
La deuxième étape consistait à tester l'aptitude du contrôleur flou à palier au problème de variation brusque de la concentration relative des bactéries ou du substrat causée par une perturbation externe. La figure (5.27) montre les réponses du système dans le cas où celui-ci est perturbé durant son régime permanent à l'instant t = 25s. On peut distinguer que la concentration relative des bactéries revient vers sa valeur de consigne ($C_1 = 0.10$), l'action de commande adéquate étant maintenue par le contrôleur flou. Dans tous les cas, la qualité de réglage est très appréciable.



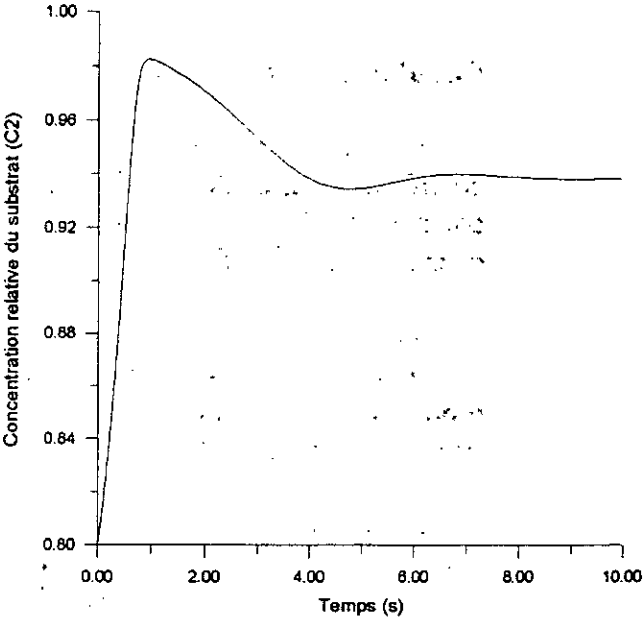
(a)



(c)

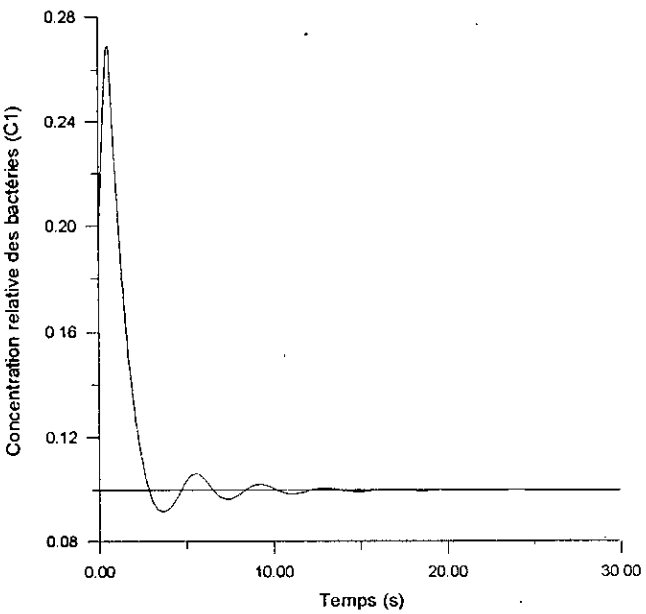


(b)

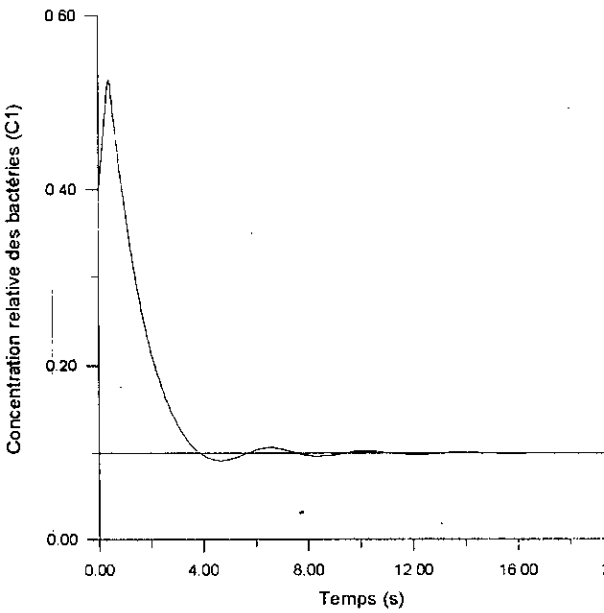


(d)

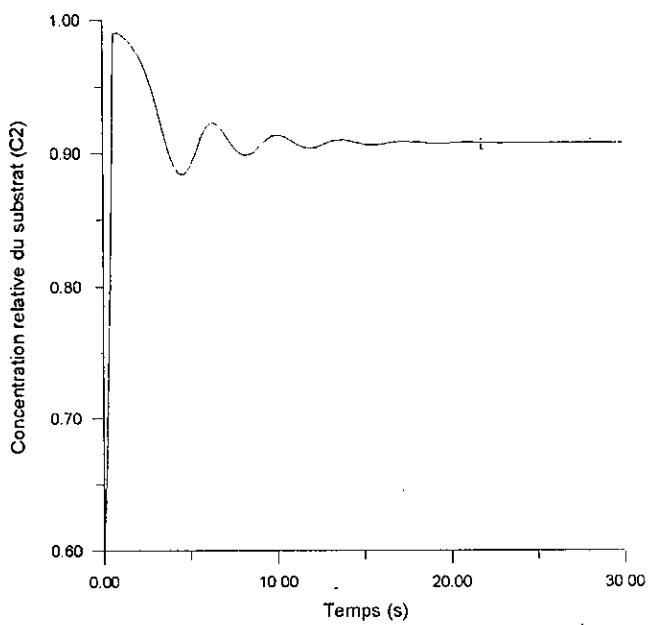
Fig. 5.25 réponses du système dans le cas d'une commande floue en régime statique : (a) et (b) $C_1 = 0.10$, (c) et (d) $C_1 = 0.075$.



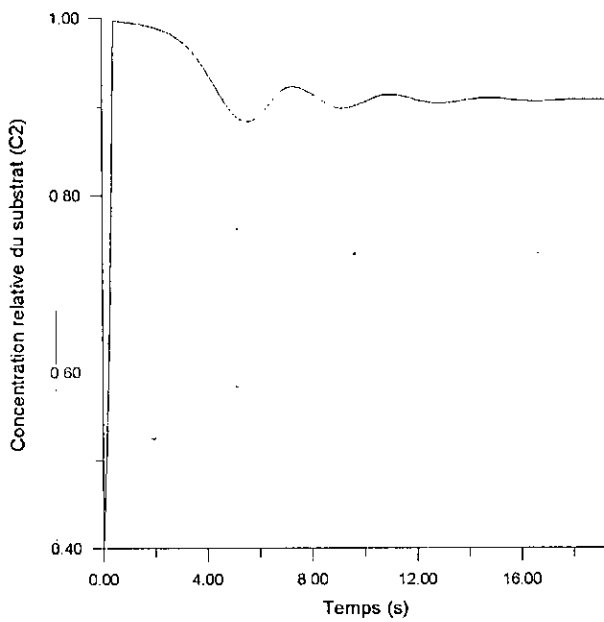
(a)



(c)

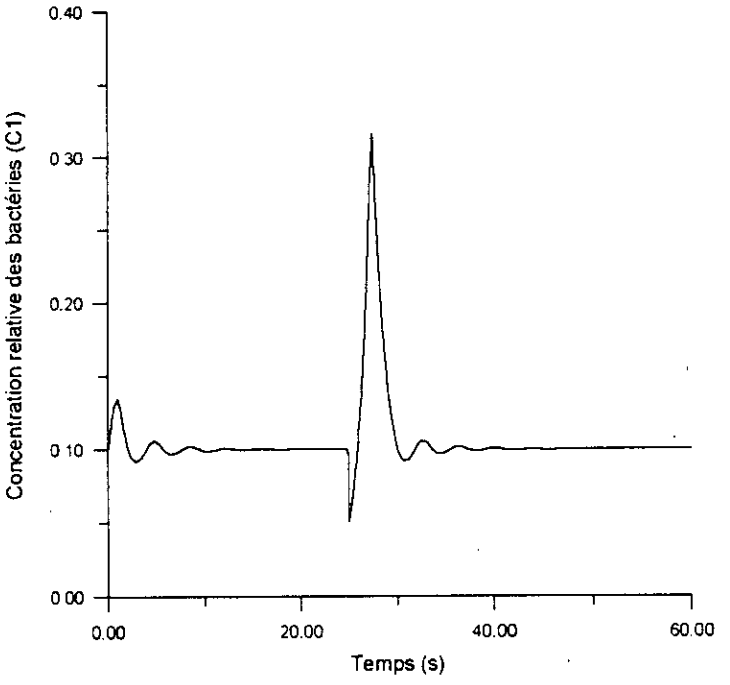


(b)



(d)

Fig. 5.26 réponses du système dans le cas d'une commande floue en régime statique : (a) et (b) $C_1(0) = 0.20$, $C_2(0) = 0.60$, (c) et (d) $C_1(0) = C_2(0) = 0.40$.



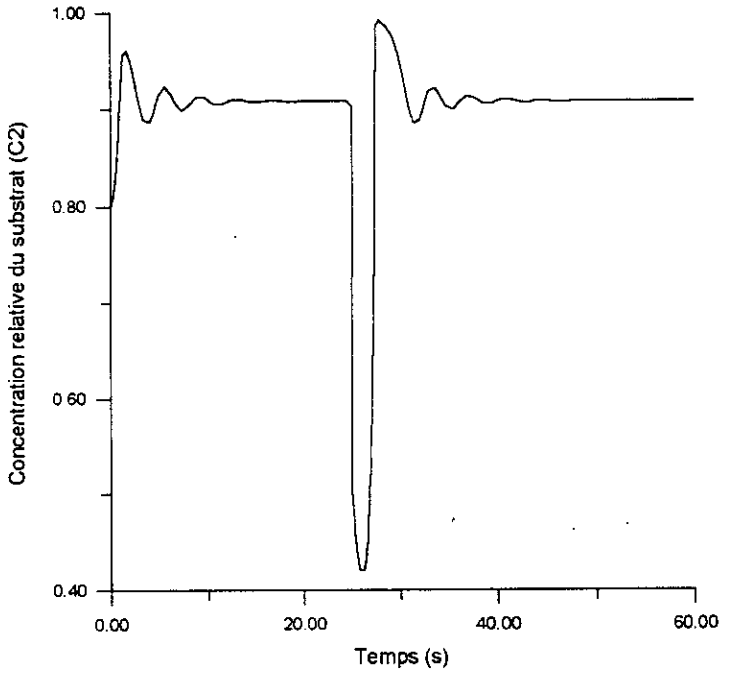


Fig. 5.27 réponses du système perturbé à t = 25s dans le cas de la commande floue en régime statique.

5.5.2 COMMANDE DU BIOREACTEUR DANS SA PLAGE D'INSTABILITE

L'objectif de cette partie de notre étude ne diffère pas de celui convoité en commande neuronale. Il s'agit de concevoir une commande floue qui a le rôle de stabiliser le système dans un intervalle de fonctionnement où il est instable, c'est à dire pour une variation de l'entrée w (flux) allant de 0.75 à 1.20. La conception de cette commande semble un centre d'intérêt puisque les systèmes flous peuvent réaliser les mêmes performances obtenues par les RNA et donc nous permet de comparer ces deux techniques et de tirer profit de leur efficacité concernant l'approximation de modèles nonlinéaires.

L'idée comme déjà exposée est de stabiliser la concentration des bactéries à l'intérieur du bioréacteur. La loi de commande qui nous garantie cette stabilité est obtenue en suivant les mêmes étapes de la partie (5.4.2), i.e., à partir de l'expression (5.5) on tire le retour d'état adéquat explicité dans les relations (5.6) et (5.7). Dans ce cas, notre contrôleur flou sera formé de deux systèmes flous, l'un pour représenter $f_1(C_2(k))$ et l'autre pour représenter $f_2(C_1(k))$, un sommateur est utilisé pour nous délivrer l'action de commande w. L'architecture de commande est présentée dans la figure (5.28).

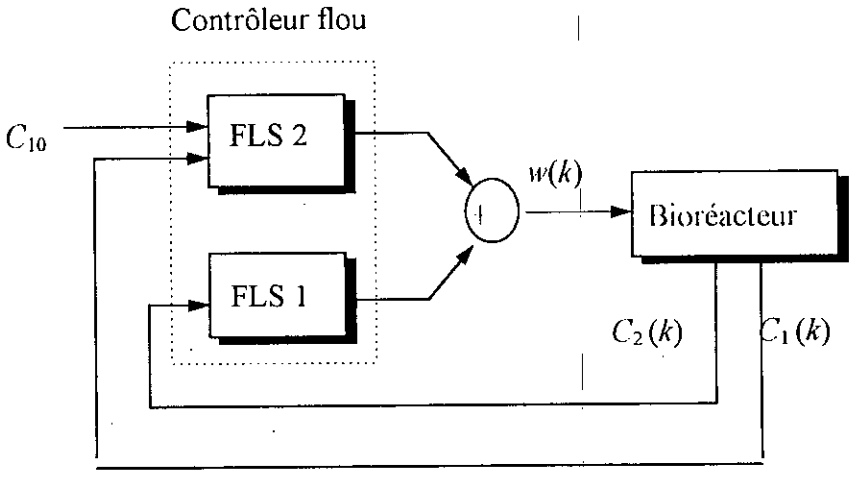


Fig. 5.28 architecture de la commande floue dans le cas où le système est instable.

Pour tester le FLS1 on a utilisé des entrées / sorties issues du modèle de la fonction nonlinéaire f_1. Le système flou utilisé est doté de 10 fonctions d'appartenance en entrée et de 10 autres en sortie. Ceci a mené à la formulation de 10 règles floues. La plage de variation de l'entrée (concentration relative du substrat) va de 0 à 1. La figure (5.29) montre les erreurs relatives commises par le FLS1 sur les exemples de test, la plupart de celles-ci sont inférieures à 2%.

Le FLS2 est testé en utilisant des entrées / sorties issues du modèle de la fonction nonlinéaire f_2. Le système flou choisi possède deux entrées (C_{10} et $C_1(k)$) et une sortie. De même que dans le cas de la commande neuronale, on a considéré deux cas différents, du fait que la nonlinéarité f_2 dépend de la valeur de C_{10} ainsi que de la constante de temps T_0.

Le FLS2 a été préparé avec trois types de fonctions d'appartenance différents, à savoir un ensemble de fonctions d'appartenance au nombre de 12 dans le cas où $C_{10} = 0.20$ et $T_0 = 0.5$s,

puis un autre ensemble de fonctions d'appartenance au nombre de 10 dans le cas où $C_{10} = 0.16$ et $T_0 = 0.5$s (variation de C_{10}) et enfin un troisième ensemble de fonctions d'appartenance au nombre de 8 dans le cas où $C_{10} = 0.20$ et $T_0 = 1$s (variation de T_0). Les résultats du test du premier cas sont illustrés dans la figure (5.30). La figure (5.31) quant à elle, présente les résultats du test du deuxième cas. Enfin, la figure (5.32) nous montre les résultats du test du troisième cas. Dans les trois cas, le système flou a abouti à des erreurs relatives inférieures à 2% en grande partie.

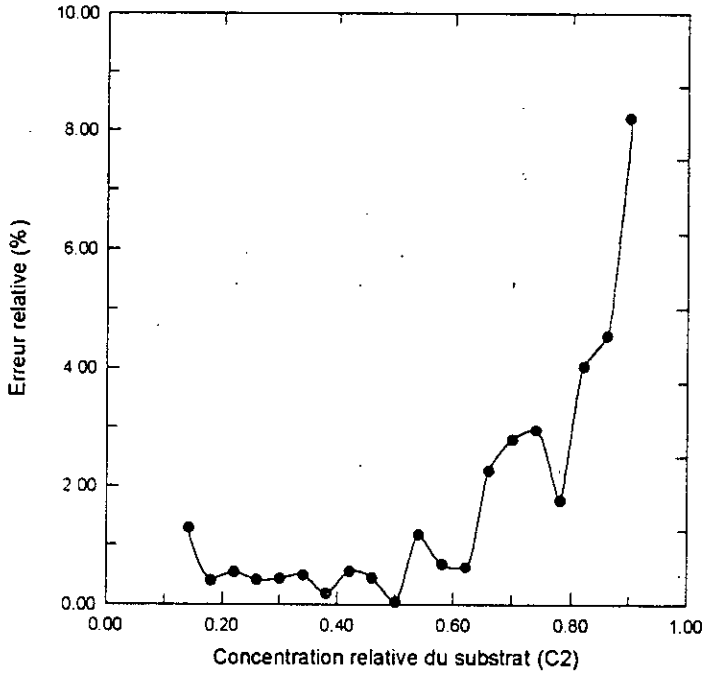


Fig. 5.29 résultats du test du FLS1 pour approximer la fonction nonlinéaire f_1.

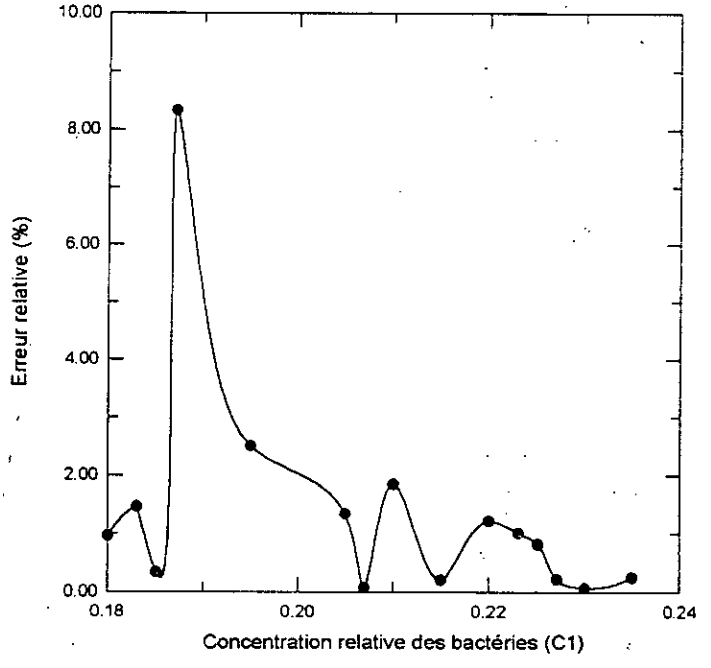


Fig. 5.30 résultats du test du FLS2 pour approximer la fonction nonlinéaire f_2. Cas où $C_{10} = 0.20$ et $T_0 = 0.5$ s.

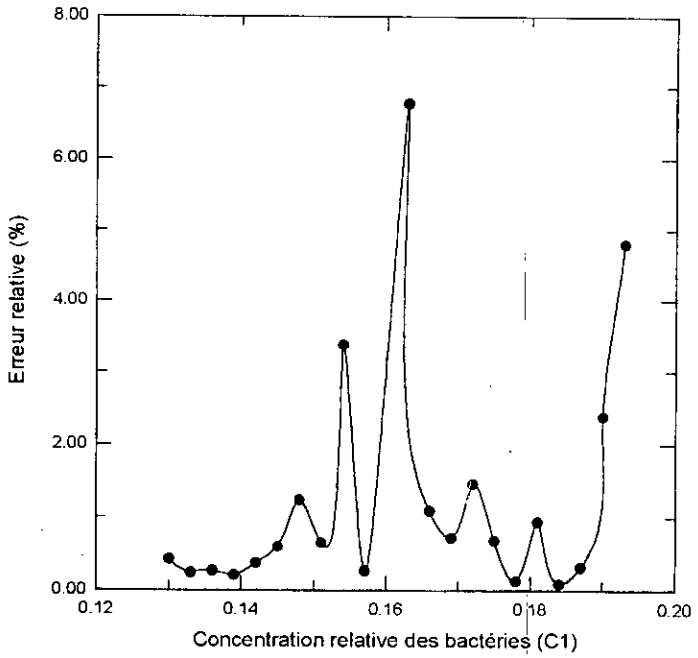


Fig. 5.31 résultats du test du FLS2 pour approximer la fonction nonlinéaire f_2. Cas où C_{10} = 0.16 et T_0 = 0.5 s.

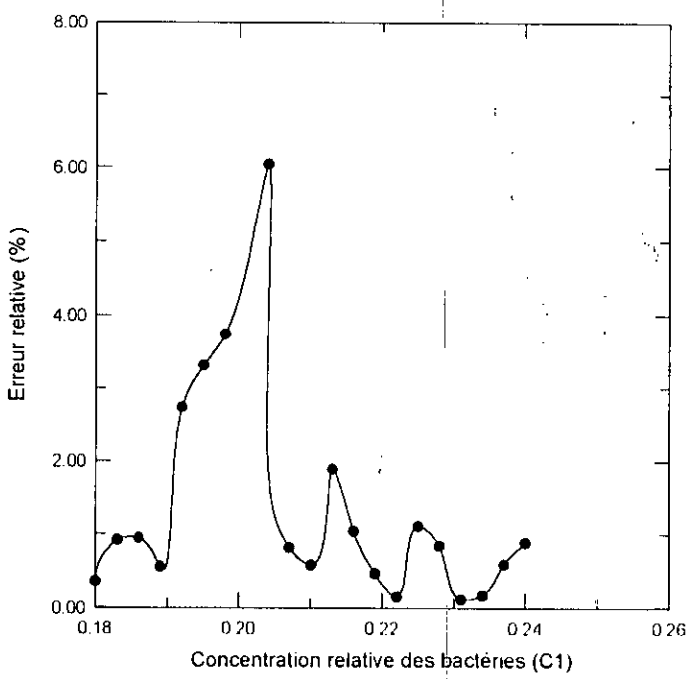


Fig. 5.32 résultats du test du FLS2 pour approximer la fonction nonlinéaire f_2. Cas où C_{10} = 0.20 et T_0 = 1 s.

Résultats de simulation

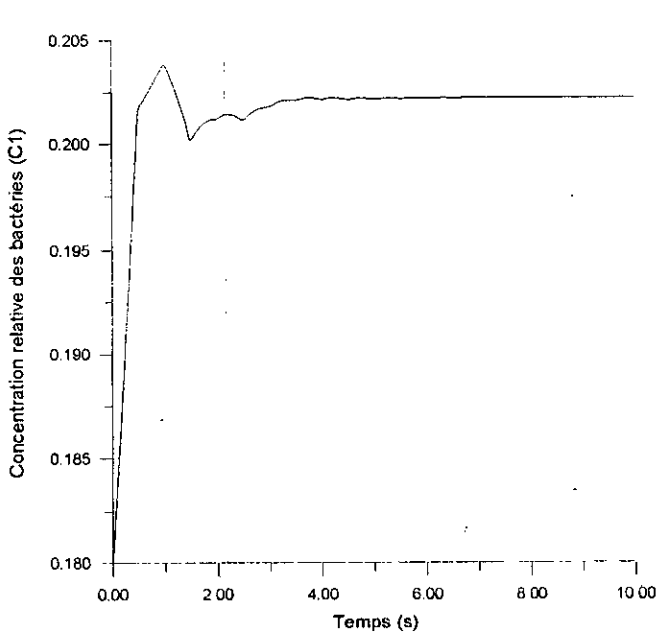
Le contrôleur flou délivre une action de commande à la fin de chaque période $T_0 > T$ et maintient celle-ci constante tout au long de cette période pour les raisons qu'on a cité plus haut avec la commande neuronale concernant un éventuel conflit d'avance ou de retard pouvant être provoqué par notre contrôleur flou.

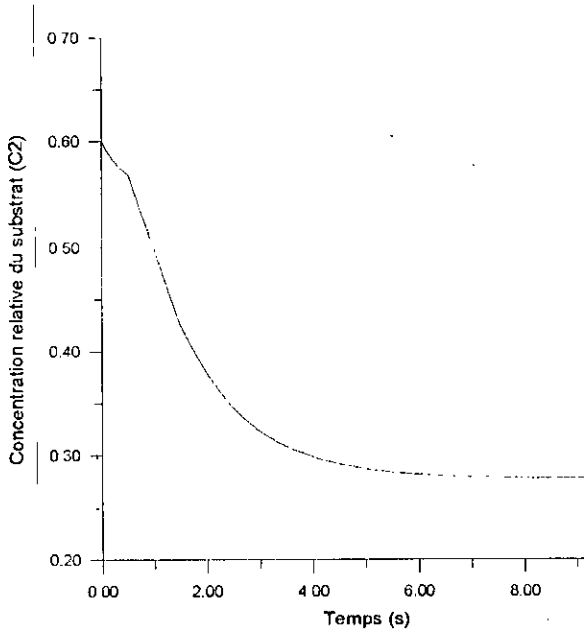
Dans le cas où $T_0 = 50T = 0.5$s. Le contrôleur flou a d'abord pour rôle de stabiliser le système autour de $C_{10} = 0.20$. La figure (5.33) montre les réponses du bioréacteur ainsi que l'action de commande délivrée par le régulateur flou pour des valeurs de conditions initiales C_1 (0) = 0.18 et C_2 (0) = 0.60. On peut distinguer que le contrôleur flou arrive à stabiliser la concentration des bactéries à la valeur désirée avec une assez bonne précision.

Pour la même valeur de T_0, on a testé l'aptitude du contrôleur flou à stabiliser le système autour du point $C_{10} = 0.16$ avec deux valeurs de conditions initiales différentes, à savoir C_1 (0) = 0.18 et C_2 (0) = 0.60 puis C_1 (0) = 0.13 et C_2 (0) = 0.60. Les figures (5.34) et (5.35) présentent les réponses du système des deux cas. Le régulateur flou se montre capable d'absorber l'influence des conditions initiales.

L'autre étape de la simulation consistait à voir l'influence de la constante de temps T_0 sur le comportement du contrôleur flou. On a donc pris $T_0 = 100T = 1$s. La figure (5.36) illustre les réponses du bioréacteur ainsi que l'action de commande délivrée par le régulateur flou pour les valeurs de conditions initiales suivantes : C_1 (0) = 0.18 et C_2 (0) = 0.60. Le contrôleur flou peut aussi stabiliser la concentration des bactéries autour de sa valeur désirée. L'influence de T_0 apparaît sur la dynamique du système, celui-ci devient moins rapide et sujet à plus d'oscillations, vu que l'action de commande, comme on l'a précisé, prend plus de temps à être appliquée. La précision du réglage quant à elle reste inchangée.

Enfin, on a testé l'aptitude du contrôleur flou à palier au problème des perturbations externes (variation brusque de l'une des concentrations) à l'intérieur du bioréacteur. La figure (5.37) nous illustre les réponses du système pour une valeur de consigne $C_{10} = 0.20$. Une fois que le système ait atteint son régime permanent, une perturbation est provoquée à t = 7s sur la concentration des bactéries (une augmentation jusqu'à 0.22) et sur la concentration du substrat (une augmentation jusqu'à 0.60). On peut remarquer que le contrôleur flou s'adapte rapidement à cette déviation et déclenche une action de commande susceptible de reconduire la concentration des bactéries vers sa valeur de consigne.





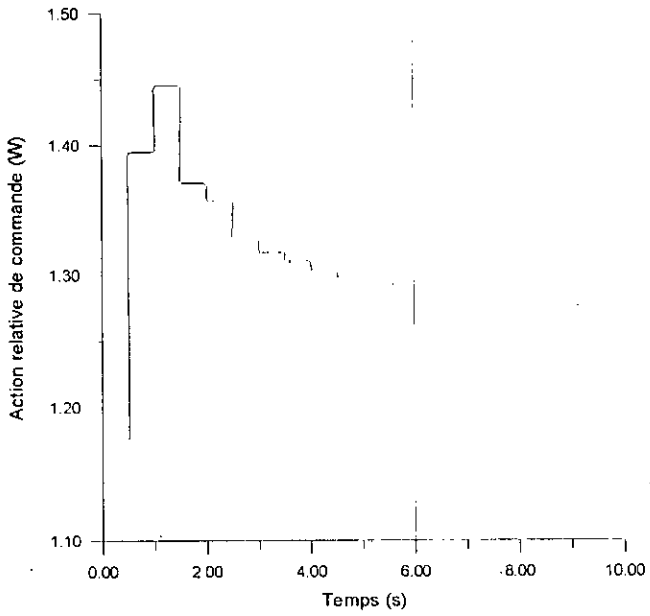
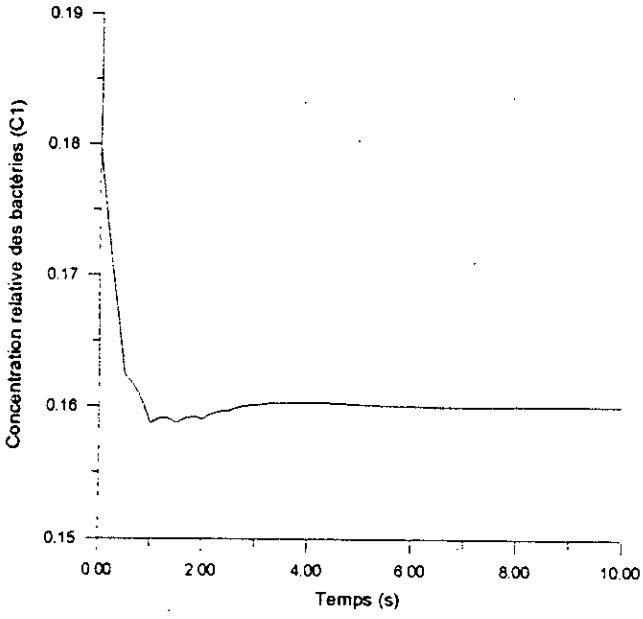
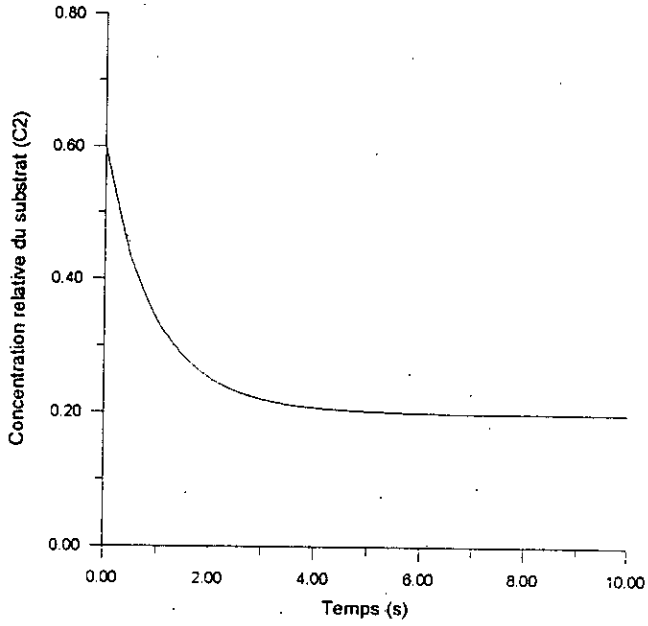


Fig. 5.33 réponses du système ainsi que l'action de commande délivrée par le contrôleur flou dans le cas où $C_{10} = 0.20$, $C_1(0) = 0.18$, $C_2(0) = 0.60$ et $T_0 = 0.5$s.





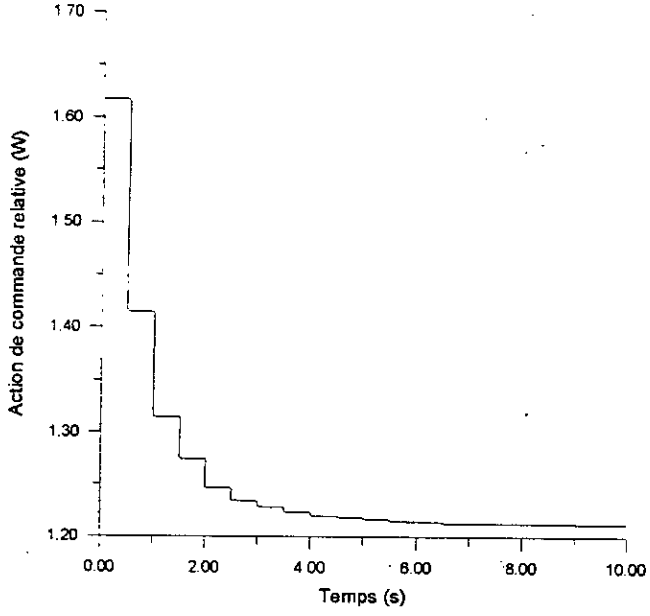
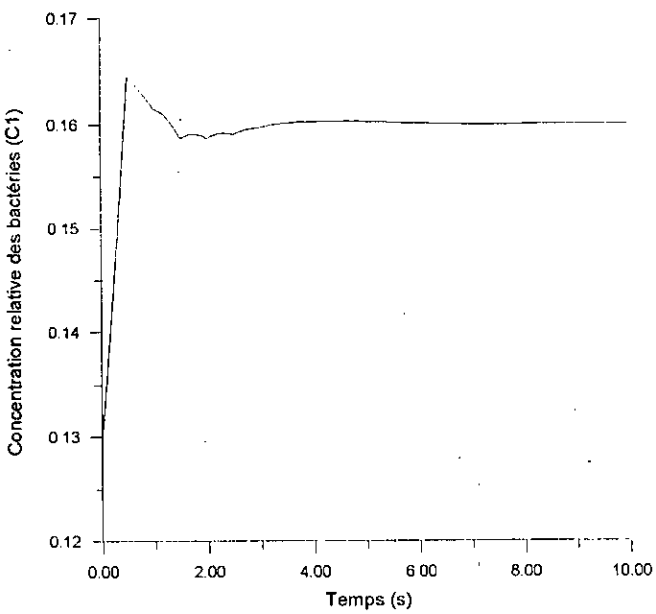
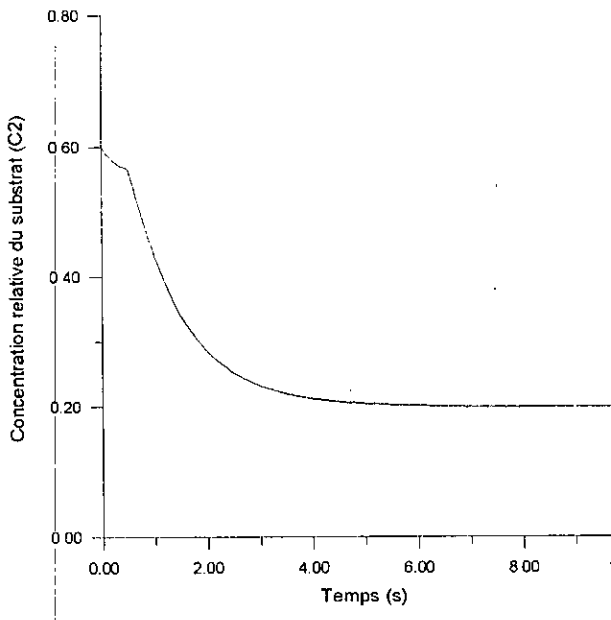


Fig. 5.34 réponses du système ainsi que l'action de commande délivrée par le contrôleur flou dans le cas où $C_{10} = 0.16$, $C_1(0) = 0.18$, $C_2(0) = 0.60$ et $T_0 = 0.5$s.





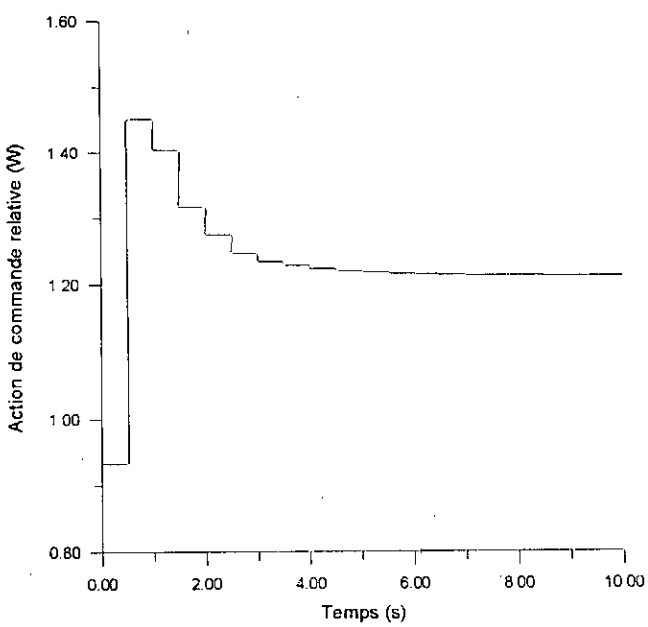
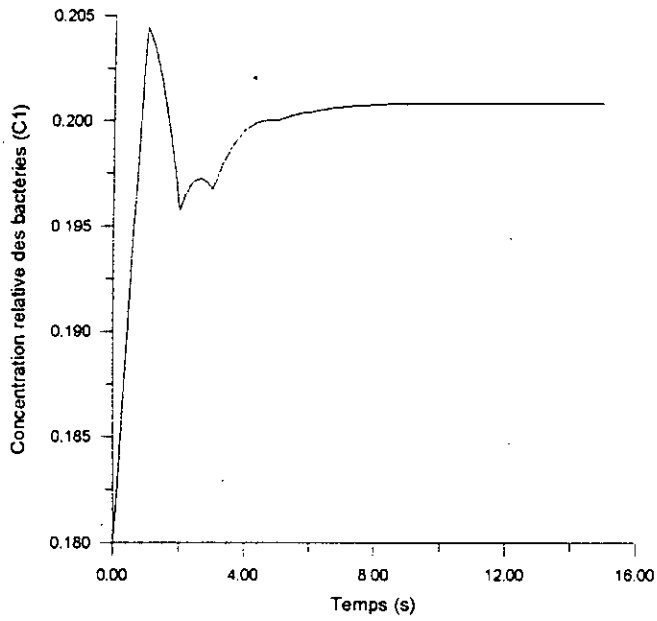
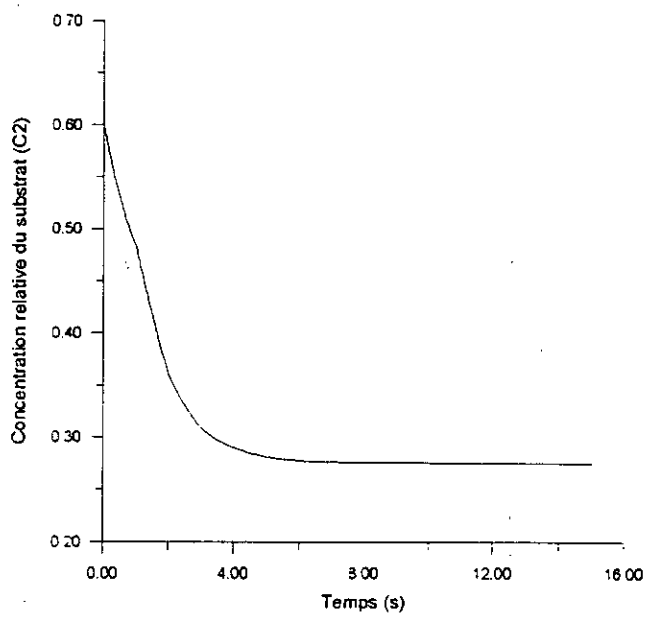


Fig. 5.35 réponses du système ainsi que l'action de commande délivrée par le contrôleur flou dans le cas où $C_{10} = 0.16$, $C_1(0) = 0.13$, $C_2(0) = 0.60$ et $T_0 = 0.5$s.





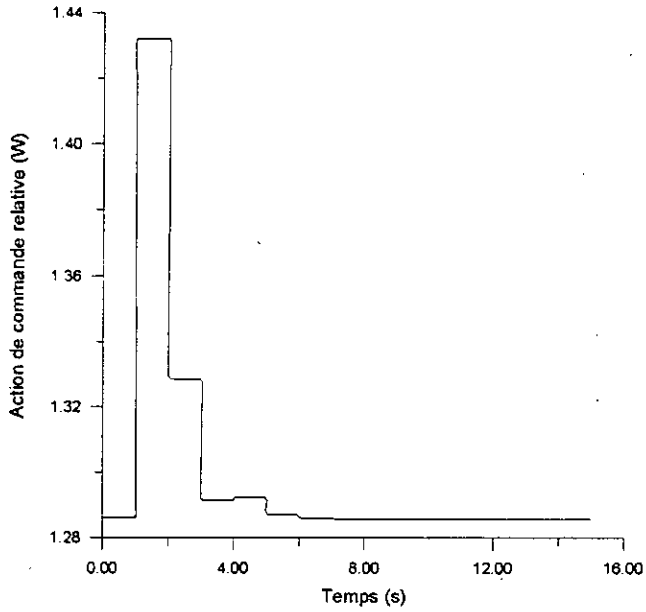


Fig. 5.36 réponses du système ainsi que l'action de commande délivrée par le contrôleur flou dans le cas où $C_{10} = 0.20$, $C_1(0) = 0.18$, $C_2(0) = 0.60$ et $T_0 = 1$s.

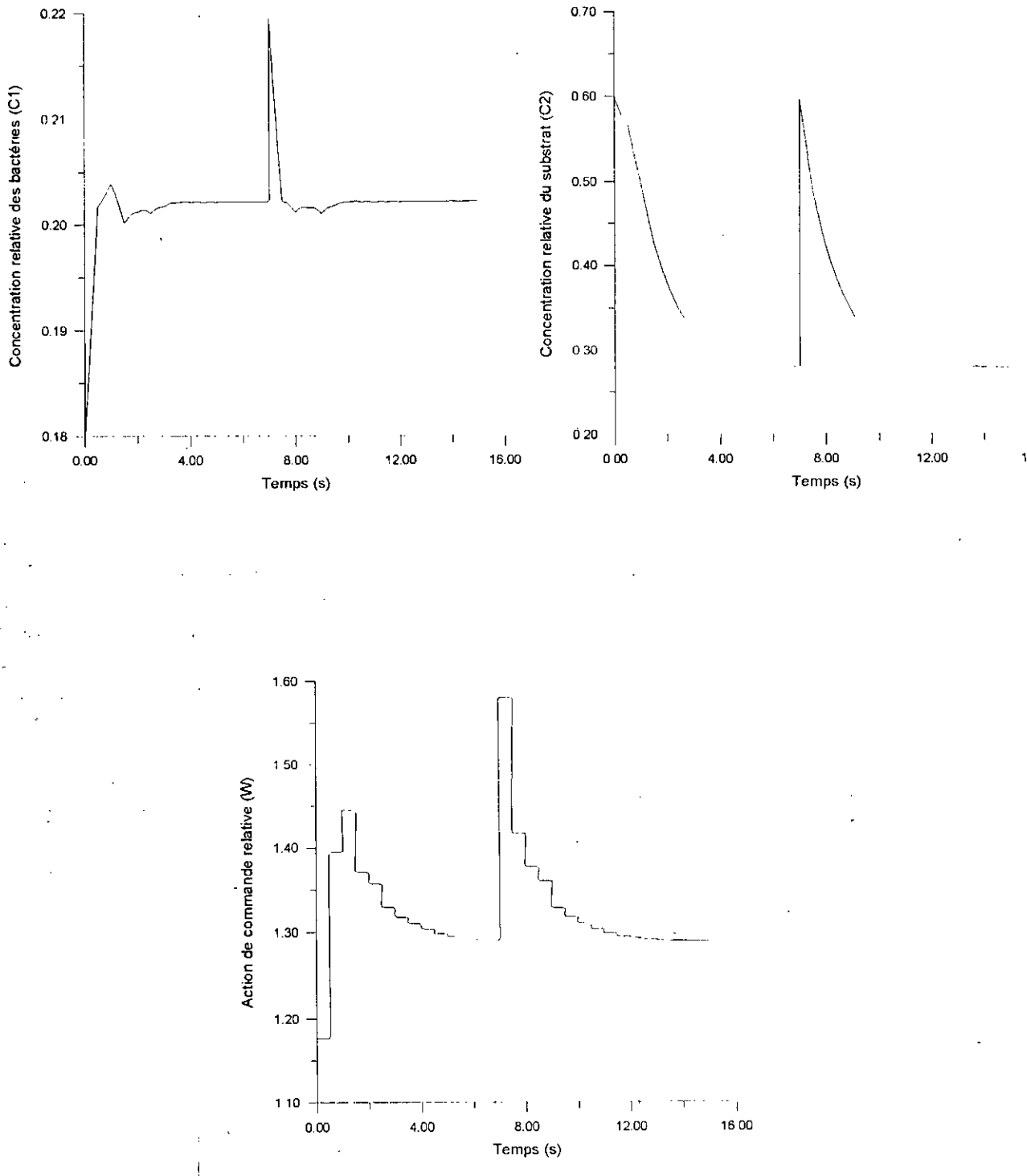


Fig. 5.37 réponses du système perturbé à t = 7s ainsi que l'action de commande délivrée par le contrôleur flou dans le cas où C_{10} 0.20, $C_1(0) = 0.22$, $C_2(0) = 0.60$ et $T_0 = 0.5$s.

5.6 COMPARAISON DES DEUX TECHNIQUES DE COMMANDE

Au vu des résultats de simulation fournis par les deux techniques de commande, à savoir les réseaux neuronaux et les systèmes flous, les performances obtenues par les deux méthodes sont comparables et quasiment similaires. Ceci est dû au fait que chaque système, que ce soit le RNA ou le FLS, est basé dans sa conception sur la reproduction d'un modèle nonlinéaire qu'est le régulateur recherché, et comme on le sait maintenant, les deux systèmes sont surtout convoités pour la résolution de ce type de problème aussi bien. Chacun d'eux est conçu en faisant appel à une expérience acquise au préalable après analyse du comportement du bioréacteur en boucle ouverte.

Pour ce qui est de la première partie de la régulation, i.e., la commande du système dans sa plage de stabilité, le neuro-contrôleur et le contrôleur flou agissent de la même manière, c'est à dire le maintien de l'action de commande adéquate qui conduit la concentration des bactéries vers sa valeur souhaitée. On peut vérifier aisément que les réponses du bioréacteur dans les deux cas sont identiques tant en régime transitoire qu'en régime permanent.

Dans la deuxième partie de la régulation, i.e., la commande du système dans sa plage d'instabilité, les deux techniques ont été d'abord comparées pour la valeur de $T_0 = 0.5$s, puis pour la valeur de $T_0 = 1$s. Pour le premier cas, la figure (5.38) montre les réponses du bioréacteur soumis aux deux techniques de commande dans le cas où $C_{10} = 0.20$ avec des conditions initiales $C_1(0) = 0.22$ et $C_2(0) = 0.60$. Avec les deux régulateurs, la concentration des bactéries se stabilise aux alentours de 0.20 avec une précision peu meilleure obtenue par le neuro-contrôleur. Ceci est dû à la légère différence qui existe entre les paliers de l'action de commande neuronale et floue, qui est due à son tour à une reproduction différentes des nonlinéarités f_1 et f_2. Pour le second cas, la figure (5.39) illustre les réponses du bioréacteur soumis aux deux techniques de commande dans le cas où $C_{10} = 0.20$ avec les mêmes conditions initiales $C_1(0) = 0.22$ et $C_2(0) = 0.60$. La concentration des bactéries se stabilise aux alentours de 0.20 avec une précision peu meilleure obtenue cette fois-ci par le contrôleur flou pour les raisons qu'on a cité ci-haut. Au fait, ces légères différences ne favorisent pas une méthode par rapport à une autre. Elles sont juste dues à la précision obtenue dans la conception des deux régulateurs (la précision de l'apprentissage pour le RNA et le choix des paramètres tels que les fonctions d'appartenance pour le système flou) d'une part et le changement de la valeur de T_0 d'autre part. Il faut noter au passage que la dynamique du système ainsi que son temps de réponse sont pratiquement pareils pour les deux régulateurs.

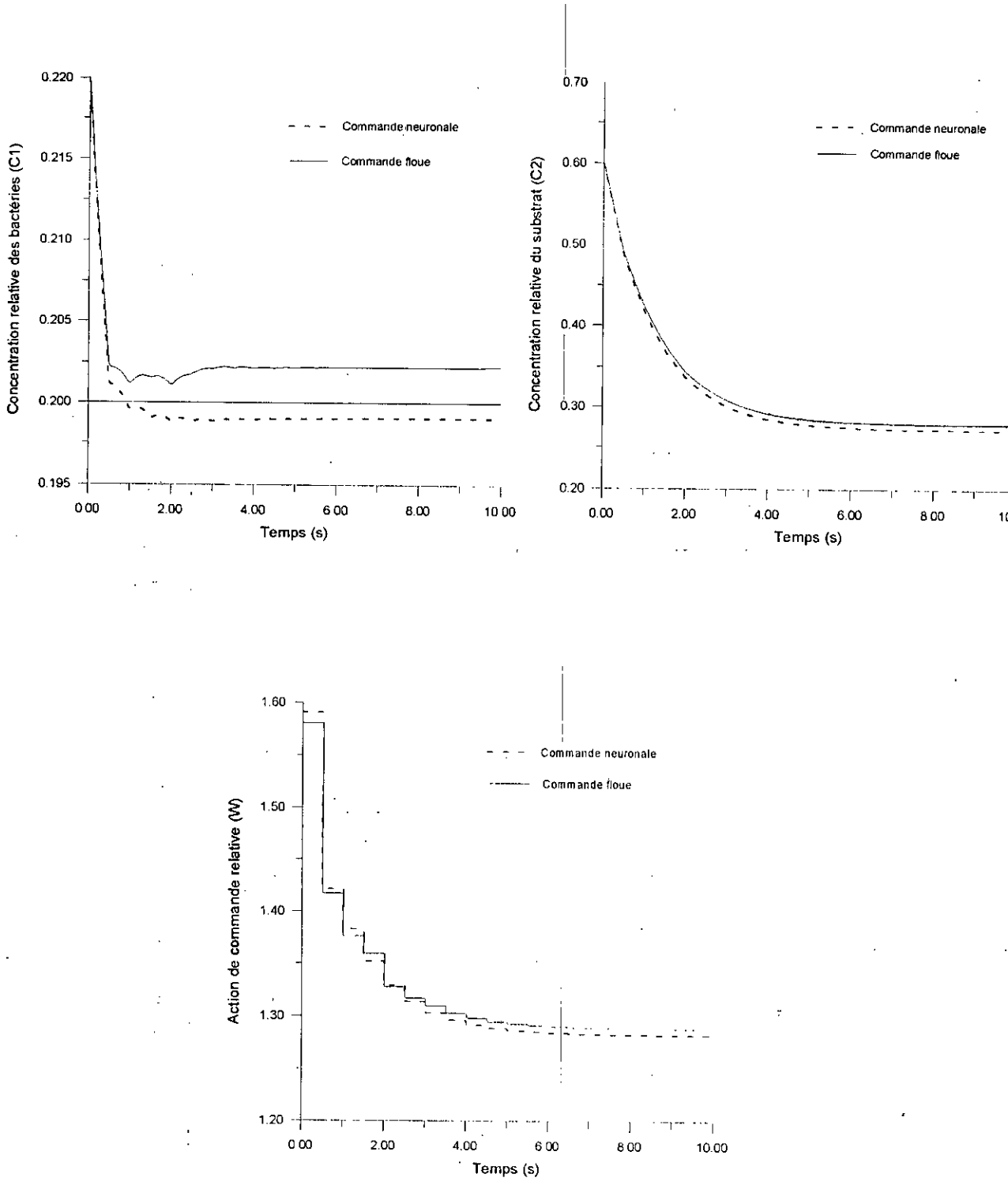


Fig. 5.38 réponses du système ainsi que l'action de commande délivrée par le neuro-contrôleur et le contrôleur flou dans le cas où $C_{10} = 0.20$, $C_1(0) = 0.22$, $C_2(0) = 0.60$ et $T_0 = 0.5$s.

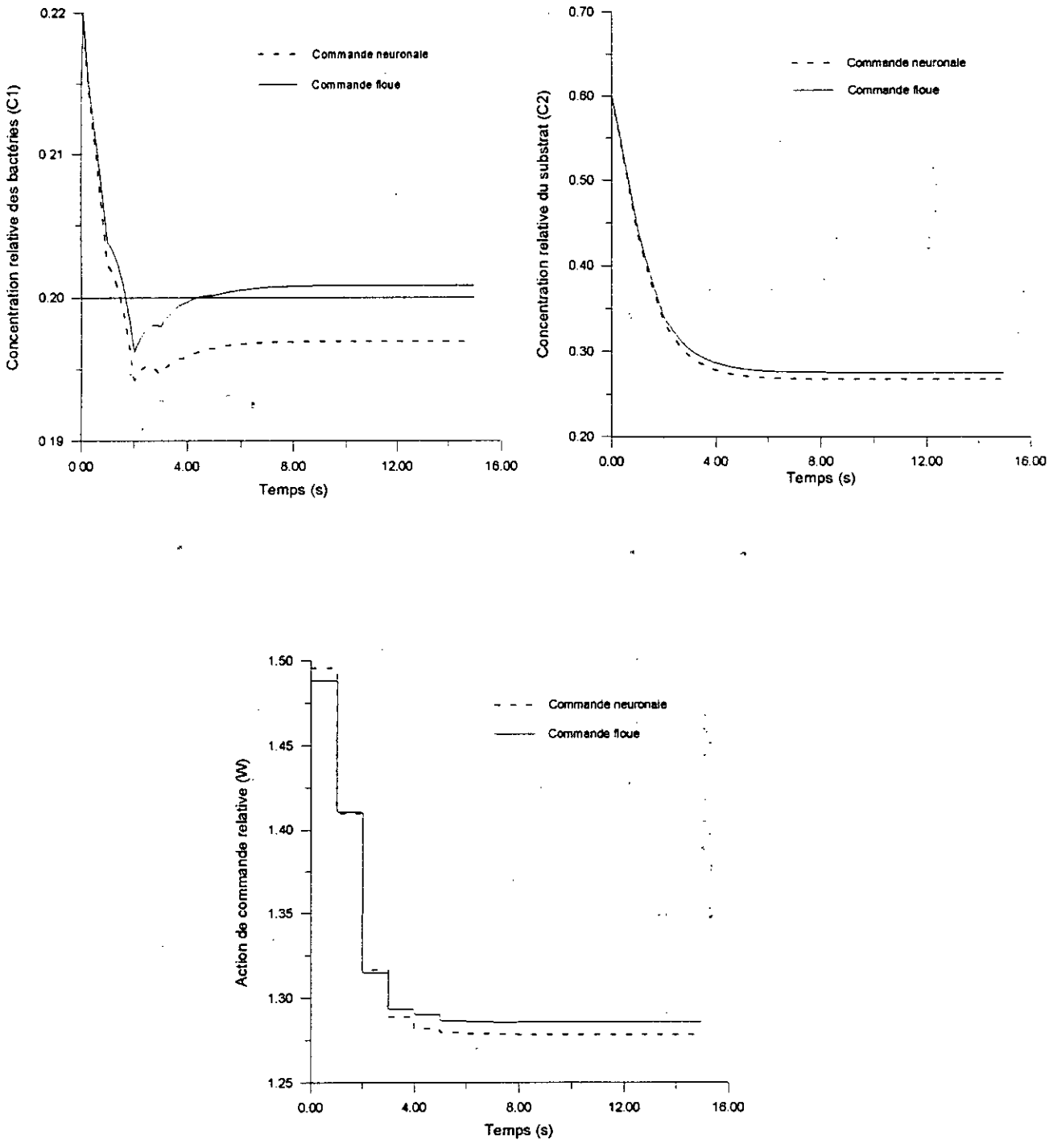


Fig. 5.39 réponses du système ainsi que l'action de commande délivrée par le neuro-contrôleur et le contrôleur flou dans le cas où $C_{10} = 0.20$, $C_1(0) = 0.22$, $C_2(0) = 0.60$ et $T_0 = 1$s.

5.7 CONCLUSIONS

Dans ce présent chapitre nous nous sommes intéressés à l'application des réseaux de neurones artificiels et de la logique floue à la commande des systèmes nonlinéaires, plus particulièrement aux systèmes qui présentent un défit quant à leur commande par les techniques classiques. Parmi ceux-là les systèmes chimiques qui présentent un intérêt industriel trop important. L'étude du neuro-contrôleur et du contrôleur flou pour la commande d'un système chimique comme le bioréacteur nous a permis d'effectuer une analyse assez détaillée dont les résultats sont les suivants :

- Contrairement aux méthodes de commande classiques, la synthèse de la commande neuronale ou floue ne nécessite pas la connaissance détaillée du modèle mathématique du système, mais se contente seulement de son comportement qualitatif en boucle ouverte.

- Comme dans le cas de la linéarisation des caractéristiques de capteurs, le choix des fonctions d'activation ainsi que les règles d'inférence floue reste subjectif.

- Les performances du contrôleur flou et du neuro-contrôleur dépendent respectivement du nombre de règles floues établies et de l'architecture du réseau choisie. Bien sûr, celles-ci peuvent être améliorées en augmentant le nombre de règles ou le nombre de neurones. Néanmoins, un compromis doit être fait car plus important est le nombre de règles ou le nombre de neurones, plus lent devient notre contrôleur nécessitant des moyens de calcul puissants pour être appliqué en temps réel.

- Le neuro-contrôleur et le contrôleur flou se sont montrés robustes vis-à-vis des variations des conditions initiales des états du bioréacteur. De plus, ils se sont révélés bons réjecteurs de bruit.

- L'utilisation de ces deux techniques de commande nous a permis de stabiliser le système dans une plage de fonctionnement ou il est instable et présente une dynamique compliquée. Elles nous offrent des performances considérables telles que la rapidité et surtout la gestion de la consommation d'énergie concernant l'action de commande.

- Enfin, la comparaison des deux techniques reste à ce stade subjective et ne peut être généralisée pour la commande de n'importe quel système.

CONCLUSION GENERALE

" Si j'ai appris une chose au cours de ma longue vie c'est que toute notre science confrontée à la réalité apparaît comme primitive et enfantine, et pourtant c'est ce que nous possédons de plus précieux."

A. Einstein (1879-1955)

L'objectif de notre travail a été la mise en oeuvre de deux techniques de pointe, qui sont actuellement des axes de recherche prometteurs pour la communauté scientifique. On sous entend par ceci les réseaux de neurones artificiels et les systèmes flous. Cette mise en oeuvre a concerné deux centres névralgiques de l'automatique, à savoir l'instrumentation (les capteurs) et la commande des systèmes nonlinéaires, qui jusqu'à présent ne possède ni outils de conception standards ni cadre théorique général. Dans les deux cas, notre choix s'est penché vers les systèmes chimiques (le capteur ionique et le bioréacteur), un domaine qui suscite beaucoup d'intérêt vu sa complexité relative.

Pour cela, une large recherche bibliographique a été faite dans le but d'étudier ces nouveaux outils (les RNA et les FLS) en qualité d'approximateurs universels de fonctions nonlinéaires, et de connaître les différentes approches existantes concernant la commande neuronale et la commande floue. A l'issue de cette recherche, on a su que ces systèmes sont parfois difficiles à construire, et que la détermination d'une architecture optimale les concernant n'est pas encore bien établie par des règles théoriques précises. Par contre, ce sont des systèmes adaptatifs capables de prendre en compte diverses contraintes. De plus, l'expérience acquise de leurs applications permet d'améliorer leur performance. Dans le domaine de la commande, l'utilisation de ces outils nous épargne, dans la majorité des cas, la connaissance détaillée du modèle du système et on se contente de l'évolution de ses entrées et sorties. Néanmoins, certaines questions doivent être investies telles que la théorie de la stabilité en commande neuronale ou floue.

Le premier test effectué avec ces deux techniques concernait la linéarisation de la caractéristique statique d'un capteur d'ions sodium à membrane de NASICON, sous l'effet d'une grandeur de perturbation due à la présence d'ions interférents de potassium dans la solution à analyser. Les linéarisateurs neuronal et flou ont permis une linéarisation sur de larges plages de variation des concentrations des ions en question, ainsi que sur une large plage de variation de la température ambiante (de 0 à 50°C). Les deux techniques de linéarisation ont pu opérer dans deux cas distincts de fonctionnement du capteur, à savoir un coefficient potentiométrique variable et une concentration de l'ion interférent variable. Dans les deux cas, le RNA et le FLS se sont comportés aussi bien. Ces mêmes techniques ont été comparées avec la linéarisation numérique appliquée au capteur ionique et ont donné des résultats comparables à la linéarisation numérique du premier ordre et meilleurs que ceux de la linéarisation numérique du second ordre. Pour ce fait, on a mentionné que les performances des deux linéarisateurs peuvent être améliorées en adaptant leurs paramètres internes. Il faut noter que les techniques de linéarisation proposées nous éloignent des méthodes de linéarisation autour d'un point de fonctionnement qui elles sont limitées.

Le second test effectué concernait la commande d'un bioréacteur qui est un système chimique fortement nonlinéaire et constitue un problème de commande délicat vu sa complexité. Les simulations en boucle ouverte ont permis la compréhension du comportement qualitatif du système et de là, la formulation des problèmes de commande à résoudre. On a pu constater que les performances du contrôleur flou et du neuro-contrôleur dépendent respectivement du nombre de règles floues et de l'architecture du réseau choisie. Celles-ci peuvent être améliorées en augmentant le nombre de règles ou le nombre de neurones. Toutefois, il faut respecter le dilemme qualité de réglage et temps de calcul. Le neuro-contrôleur et le contrôleur flou se sont révélés robustes vis-à-vis des variations des conditions initiales des états du bioréacteur, et bien sûr bons rejecteurs de bruit du à des perturbations externes. Les deux techniques de commande nous ont permis de stabiliser le système dans une plage de fonctionnement où il est instable et présentant une dynamique complexe. La rapidité des réponses ainsi que la bonne gestion de la consommation d'énergie concernant l'action de commande sont d'autres performances obtenues par les deux régulateurs.

Enfin et à titre de perspectives, on pourra suggérer, pour des études ultérieures, les points suivants :

- En ce qui concerne la linéarisation des capteurs ioniques, des recherches peuvent être menées quant à la linéarisation en présence de plusieurs ions interférents simultanément tels que les ions potassium et les ions lithium. Une autre étude pourra traiter la linéarisation avec prise en compte de la dynamique du capteur.

- En ce qui concerne la commande du bioréacteur, on suggère de commander le système sur toute sa plage de fonctionnement et pourquoi pas prendre un modèle beaucoup plus général mais beaucoup plus difficile que celui pris dans notre étude. On mentionnera aussi la possibilité de commander le système en bénéficiant des atouts des deux techniques à la fois, on sous entend par ceci l'utilisation de la commande neuro-linguistique.

BIBLIOGRAPHIE

" Pour faire de grandes choses il ne faut pas être un si grand génie, il ne faut pas être au dessus des hommes, il faut être avec eux."

Charles De Secondât Montesquieu (1689-1755)

[1] Davalo, E. and Naim, P. *Des Réseaux de Neurones.* France : Eyrolls, 1990. 232 p.

[2] Freeman, J.A. and Skapura, D.M. *Neural Networks.* Massachusets : Addison -Wesley, 1992. 401p.

[3] Widrow, B. and Lehr, M.A. 30 Years of Adaptive Neural Networks : Perceptron, Madaline, Backpropagation. *Proc. of the IEEE*, September 1990, Vol. 78, N° 9, p. 1415-1442.

[4] Sanner, R.M. and Slotine, J.E. Gaussian Networks for Direct Adaptive Control. *IEEE Trans. on Neural Networks*, November 1992, Vol. 3, N° 6, p. 837-863.

[5] Heniche, M.M., Boudjema, F. and Attari, M. Simulating Nonlinear Functions With Artificial Neural Networks. In *Proc. of the Int. Conf. on Signal and Systems*, Algiers, September 1994, p. II.25-II.29.

[6] Binse, M. Systèmes Connexionnistes. Lille (Fr.) : Laboratoire d'informatique, Janvier 1990, Pub. N° I.T 178.

[7] Hammerstrom, D. Working With Neural Networks. *IEEE Spectrum Comp. App.*, July 1993, p. 46-53.

[8] Lemberg, H. Les Réseaux Neuronaux Artificiels. *INFOPC-Langages et Systèmes*, N° 71, p. 20-28.

[9] Baba, N. A New Approach for Finding the Global Minimum of Error Function of Neural Networks. *Pergamon Press*, 1989, Vol. 2, p. 367-373.

[10] Solis, F.J. and Wets, R.J.B. Minimization by Random Search Techniques, Mathematics of Operations Research, *The Institute of Management Science*, Feb. 1981, Vol.6, N° 1, p. 19-29.

[11] Boudaoud, N. and Boukhobza, T. *Modélisation Nonlinéaire et Commande Neurolinguistique par Régimes Glissants de la Machine Synchrone en Monovariable.* Projet de fin d'études : Ecole Nationale Polytechnique, Département Génie Electrique (Automatique), 1993. 143 p.

[12] Werbos, P. Backpropagation Through Time : What it Does and How to Do it. *Proc. of the IEEE*, October 1990, Vol. 78, N° 10, p. 1550-1560.

[13] Heniche, M.M. *Sur la Linéarisation de Capteurs à l'Aide des Réseaux de Neurones Artificiels.* Projet de fin d'études : Ecole Nationale Polytechnique, Département de Génie Electrique (Automatique), 1994. 93 p.

[14] Shun-Ichi, A. Mathematical Foundations of Neurocomputing. *Proc. of the IEEE*, September 1990, Vol. 78, N° 9, p. 1443-1463.

[15] Vapnik, V.N. Estimation of Dependences Based on Empirical Data. *Springer Series in Statistics*, Springer-Verlag, 1982.

[16] Baum, E.B. and Haussler, D. What Size Net Gives Valid Generalization. Massachusets Institute of Technology : *Neural Computation*, 1989, p. 151-160.

[17] Kunt, M. *Traitement Numérique des Signaux*. Paris : Dunod, 1981, Traité d'Electricité. 403 p.

[18] Fukuda, T. Theory and Applications of Neural Networks for Industrial Control Systems. *IEEE Trans. on Industrial Electronics*, December 1992, Vol. 39, N° 6, p. 472-489.

[19] Hunt, K.J. et al. Neural Networks for Control Systems - A Survey. *Automatica*, 1992, Vol. 23, p. 1083-1112.

[20] Narandra, K.S. and Parthasarathy, K. Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Trans. on Neural Networks*, March 1990, Vol. 1, N° 1, p. 4-27.

[21] Narandra, K.S. and Mukhopadhayay, S. Intelligent Control Using Neural Networks. *IEEE Control Systems Magazine*, April 1992, p. 11-18.

[22] Levin A.U. and Narandra, K.S. Control of Nonlinear Dynamical Systems Using Neural Networks : Controllability and Stabilization. *IEEE Trans. on Neural Networks*, March 1993, Vol. 4, N° 2, p. 192-206.

[23] Psaltis, D. et al. A Multilayered Neural Network Controller. *IEEE Control Systems Magazine*, April 1988, p. 17-21.

[24] Ichikawa. Y. and Sawa, T. Neural Network Application for Direct Feedback Controllers. *IEEE Trans. on Neural Networks*, March 1992, Vol. 3, N° 2, p. 224-231.

[25] Yamada, T. and Yabuta, T. Nonlinear Network Controller for Dynamic System. *Proc. of the IEEE*, 1990, p. 1244-1249.

[26] Li, W. and Slotine, J.E. Neural Network Control of Unknown Nonlinear Systems. Massachusets Institute of Technology : Nonlinear Systems Laboratory, *TA6-9:00*, p. 1136-1141.

[27] Nguyen, D.H. and Widrow, B. Neural Networks for Self-Learning Control Systems. *IEEE Control Systems Magazine*, April 1990, p. 18-23.

[28] Anderson, C.W. Learning to Control an Inverted Pendulum Using Neural Networks. *IEEE Control Systems Magazine*, April 1989, p. 31-36.

[29] Hoffer, D.S. et al. Neural Control of a Steel Rolling Mill. *IEEE Control Systems Magazine*, June 1993, p. 69-75.

[30] Dote, Y. Fuzzy and Neural Network Controller. *Proc. of the IEEE*, 1990, p. 1314-1343.

[31] Miller, W.T. et al. *Neural Networks for Control.* Massachusets : The MIT Press, 1990. 524 p.

[32] Isidori, A. *Nonlinear Control Systems : An Introduction.* New York : Springer Verlag, 1989. 479 p.

[33] Gilles, J. and Decaulne, P. *Systèmes Asservis Nonlinéaires.* Paris : Dunod, 1988. 219 p.

[34] Mukhopadhayay, S. and Narandra, K.S. Disturbance Rejection in Nonlinear Systems Using Neural Networks. *IEEE Trans. on Neural Networks*, January 1993, Vol. 4, N° 1, p. 63-72.

[35] Barto, A.G. *Neural Networks for Control.* Massachusets : The MIT Press, 1990. Connexionist Learning for Control, p. 5-58.

[36] Werbos, P.J. *Neural Networks for Control.* Massachusets : The MIT Press, 1990. Overview of Disigns and Capabilities, p. 59-65.

[37] Werbos, P.J. *Neural Networks for Control.* Massachusets : The MIT Press, 1990. A Menu of Disigns for Reinforcement Learning, p. 67-95.

[38] Hornik, K.M. et al. Multilayered Feedforward Networks Are Universal Approximators. *IEEE Trans. on Neural Networks*, 1989, Vol. 2, p. 359-366.

[39] Lippmann, R.P. An Introduction to Computing With Neural Nets. *IEEE ASSP Magazine*, Vol. 4, p. 4-22.

[40] Ljung, L. *System Identification : Theory for The User.* New Jersey : Prentice-Hall, INC., Englewood Cliffs, 1987. 519 p.

[41] Widrow, B. and SMITH, F.W. Pattern Recognition Control Systems. *Proc. COINS*, 1963.

[42] Barto, A.G., Sutton, R.S. and Anderson, C.W. Neuron Like Adaptive Element that Can Solve Difficult Learning Problem. *IEEE Trans. Syst. Man Cybern.*, 1983, Vol. SMC-13, p. 834-846.

[43] Aström, K. And Wittenmark, B. *Adaptive Control.* New york : Addison-Wesley, 1989. 526 p.

[44] Hamzi, M. and Labiod, S. *Identification et Commande des Systèmes Dynamiques par Réseaux de Neurones.* Projet de fin d'études : Ecole

Nationale Polytechnique, Département de Génie Electrique (Automatique), 1995. 108 p.

[45] Mendel, J.M. Fuzzy Logic Systems for Engineering : A Tutorial. *Proc. of The IEEE*, March 1995, Vol. 83, N° 3, p.345-377.

[46] Lee, C.C. Fuzzy Logic in Control Systems : Fuzzy Logic Controller - Part I. *IEEE Trans. Syst. Man Cybern.*, March/April 1990, Vol. 20, N° 2, p. 404-418.

[47] Lee, C.C. Fuzzy Logic in Control Systems : Fuzzy Logic Controller - Part II. *IEEE Trans. Syst. Man Cybern.*, March/April 1990, Vol. 20, N° 2, p. 419-433.

[48] Kosko, B. *Neural Networks and Fuzzy Systems*. New Jersey : Prentice-Hall, INC., Englewood Cliffs, 1992. 451 p.

[49] Yamada, S.I. et al. Fuzzy Control of The Roof Crane. *Proc. of The IEEE*, 1989, p. 709-714.

[50] Barrat, J.P. and Lecluse, Y. Exemple d'Application de la Logique Floue : Commande de la Température d'un Four Pilote. Université de Caen (Fr.) : *Techniques de L'ingenieur,* Traité Mesures et Contrôle, 1995, p. 1-10. R 7 428.

[51] Herrero, R. et al. A Highly Nonlinear Fuzzy Control Alghorithm for Servo Systems Positioning. *Proc. of The IFAC Intelligent Components and Instruments for Control Applications*, Malaga (Spain), 1992, P. 93-98.

[52] Boscolo, A. et al. Fuzzy Controller for Generally Loaded DC Electric Motor. *Proc. of The IFAC Intelligent Components and Instruments for Control Applications*, Malaga (Spain), 1992, P. 99-104.

[53] Tuner, R. *Logique pour l'Intelligence Artificielle*. Paris : Masson, 1986.

[54] Sugeno, M. And Murakami, K. An Experimental Study on Fuzzy Parking Control Using a Model Car. In *Industrial Applications of Fuzzy Control*. Amsterdam : North-Holland, 1985, p. 125-138.

[55] Buck, R.P. Electrochemistry of Ion-Selective Electrodes. Lausanne : *Sensors and Actuators*, 1 (1981), p. 197-260.

[56] Asch, G. *Les Capteurs en Instrumentation*. Paris : Dunod, 1982. 788 p.

[57] Attari, M. Methods for Linearization of Nonlinear Sensors. *Proc. CMMNI-4, Fourth Maghrebin Conference on Numerical Methods of Engineering*, Algiers (Algeria), Nov. 1993, Vol. 1, p. 344-350.

[58] Attari, M., Boudjema, F. and Heniche, M.M. Linearizing a Thermistor Characteristic in the Range of Zero to 100°C With Two Layers Artificial Neural

ANNEXES

" Je me fais l'impression de n'avoir été qu'un enfant jouant sur la plage et s'y amusant a y trouver de temps en temps un galet particulièrement lisse ou un coquillage plus joli que les autres, tandis que s'étendait devant moi, inconnu, le grand océan de la vérité."

Isaac Newton (1642-1727)

Network. *Proc. IEEE Inst. Meas. Tech. Conference (IMTC/95),* Waltham, Massachusets (USA), April 1995, p. 119-122.

[59] Attari, M., Boudjema, F. and Heniche, M.M. An Artificial Neural Network to Linearize a G (Tungesten vs. Tungesten 26% Rhenium) Thermocouple Characteristic in the Range of Zero to 2000°C. *Proc. IEEE International Symposium on Industrial Electronics, ISIE'95*, Athens (Greece), July 1995, Vol. 1, p. 176-180.

[60] Attari, M., Heniche, M.M. and Boudjema, F. A Two Dimensional Intelligent Calibration of an Ion Sensor. *Proc. IEEE Inst. Meas. Tech. Conference (IMTC/96),* Brussels (Belgium), June 1996, Vol. II, p. 788-791.

[61] Covington, A.K. Ion Selective Electrode Methodology. *CRC Press*, Vol. 1, 1979.

[62] Koryta, J. and Stulik, K. *Ion-Selective Electrodes.* Cambridge : Cambridge University Press, 1983.

[63] Hulanicki, A. et al. The Study of Processes Influencing Apparent Selectivity for Solid-State Ion Selective Electrodes. *Pergamon Press, 5th Symposium on Ion Selective Electrodes*, Oxford, 1988, p. 411-424.

[64] Macca, C. and Cakrt, M. Determination of Selectivity Coefficients of Ion-Selective Electrodes by Means of Linearized Multiple Standard Addition Techniques. Holland : *Elsevier Science Publishers* B.V., 1983, p. 51-60.

[65] Hafit, K. *Etudes des Performances de Capteurs Potentiométriques a Ions Sodium Utilisant des Membranes de NASICON.* Thèse de Doctorat : Institut National Polytechnique de Grenoble (Fr.), Laboratoire d'Ionique et d'Electrochimie des Solides, Mars 1992. 128 p.

[66] Fabry, P. and Siebert, E. NASICON : A Sensitive Membrane for Ion Analysis. Laboratoire d'Ionique et d'Electrochimie des Solides de Grenoble (Fr.), p. 111-124, CNRS-UA 1213.

[67] Fabry, P. et al. NASICON, an Ionic Conductor for Solid-State Na^+ Selective Electrode. *Sensors and Actuators*, 15 (1988), p. 33-49.

[68] Kleitz, M. et al. New Compounds for ISFETS. North-Holland, Amsterdam, *Solid-State Ionics* 22 (1987), p. 295-303.

[69] Siebert, E. et al. Na^+ Exchange at the NASICON / Water Interface. Lausanne : *Elsevier Sequoia S.A., J. Electroanal. Chem.*, 286 (1990), p. 245-251.

[70] Hasler, M. *Circuits Nonlinéaires.* Lausanne : Presses Polytechnique Romandes, 1985.

[71] Morari, M. and Zafiriou, E. *Robust Process Control.* New Jersey : Prentice-Hall, Englewood Cliffs, 1989.

[72] Ungar, L.H. *Neural Networks for Control.* Massachusets : The MIT Press, 1990. A Bioreactor Benchmark for Adaptive Network-based Process Control, p. 387-402.

[73] Agrawal, P. et al. Theoretical Invistigations of Dynamic Behavior of Isothermal Continuous Stirred Tank Biological Reactors. *Chemical Engineering Science*, 1982. 37:453.

[74] Agrawal, P. and Seborg, D.E. Self-tuning Controllers for Nonlinear Systems. *Automatica*, Vol. 23, N° 2, 1987, p. 209-214.

[75] Abida, L. et al. Modélisation et Commande Adaptative d'une Colonne de Distillation. *In Proc. of the Int. Conf. on Signal and Systems*, Algiers, September 1994, p. VI.1-VI.5.

[76] Pau, L.F. and Johansen. Neural Network Signal Understanding for Instrumentation. *IEEE Trans. On Inst. & Meas.*, Aug. 1990, Vol. 39, N° 4, p. 558-564.

[77] Naidu, R. and MCavoy, T.J. Use of Neural Networks for Sensor Failure Detection in a Control System. *IEEE Control Systems Magazine*, April 1990, p. 94-105.

ANNEXES

" Je me fais l'impression de n'avoir été qu'un enfant jouant sur la plage et s'y amusant a y trouver de temps en temps un galet particulièrement lisse ou un coquillage plus joli que les autres, tandis que s'étendait devant moi, inconnu, le grand océan de la vérité."

Isaac Newton (1642-1727)

ANNEXE -A-

Démonstration mathématique de l'algorithme de rétropropagation

On note:

$X = (x_1, x_2, \ldots, x_n)$ entrées du réseau.
$Y = (y_1, y_2, \ldots, y_m)$ sorties désirées du réseau.
$S = (s_1, s_2, \ldots, s_m)$ sorties réelles du réseau.

f : fonction sigmoïde de chaque neurone, f' sa dérivée.

O_j : sortie du neurone j.
I_i : entrée du neurone i. (Somme pondérée)

μ : pas du gradient.

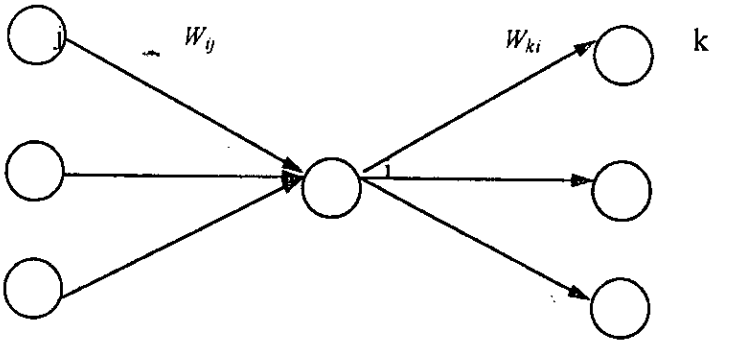


Fig. A.1 connexions du neurone i avec les neurones de la couche précédente et suivante.

On définit la fonction coût à minimiser par :

$$J(w) = \frac{1}{2}\sum_{i=1}^{m}(s_i - y_i)^2 \qquad (A.1)$$

la formule d'adaptation des poids est, par définition :

$$w_{ij}(k+1) = w_{ij}(k) - \mu\frac{\partial J}{\partial w_{ij}} \qquad (A.2)$$

on a :

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial I_i}\frac{\partial I_i}{\partial w_{ij}} \tag{A.3}$$

or :

$$\frac{\partial I_i}{\partial w_{ij}} = \frac{\partial(\sum_p w_{ip} O_p)}{\partial w_{ij}} = O_j \tag{A.4}$$

on a donc :

$$\frac{\partial J}{\partial w_{ij}} = d_i O_j \tag{A.5}$$

avec :

$$d_i = \frac{\partial J}{\partial I_i} \tag{A.6}$$

Pour les neurones de la couche de sortie :

$$d_i = \frac{1}{2}\frac{\partial \sum_j (s_i - y_j)^2}{\partial I_i} = (s_i - y_i)\frac{\partial s_i}{\partial I_i} \tag{A.7}$$

$$d_i = (s_i - y_i) f'(I_i) \tag{A.8}$$

Pour les neurones des couches cachées :

$$d_i = \sum_h \frac{\partial J}{\partial I_h}\frac{\partial I_h}{\partial I_i} = \sum_h d_h \frac{\partial I_h}{\partial I_i} \tag{A.9}$$

avec h l'indicatif des neurones reliés à la sortie du neurone i.

On aura donc :

$$d_i = \sum_h d_h w_{hi} f'(I_i) \tag{A.10}$$

La règle de modification sera alors :

$$w_{ij}(k+1) = w_{ij}(k) - \mu d_i O_j \tag{A.11}$$

avec d_i calculés à l'aide des formules (A.8) et (A.10).

ANNEXE -B-

Méthode d'optimisation aléatoire [10]

Soit f une fonction définie de R^n vers R et S un sous ensemble de R^n. On veut trouver le point x de S, pour lequel f présente un minimum.

- étape 1 : choisir $x(0)$ dans S (poser k=0)
- étape 2 : générer un bruit Gaussien $g(k)$ dans l'espace $(R^n,B,\mu(k))$
- étape 3 : poser $x(k+1) = D(x(k),g(k))$
  choisir $\mu(k+1)$

avec: $\mu(k)$ correspondant à une fonction de distribution définie dans R^n, et B un sous ensemble de Borel.

De cet algorithme, on va tirer la première hypothèse de convergence vers le minimum:

H1. $f(x) \geq f(D(x,g))$.

Si le minimum de f existe pour un point où la fonction est discontinue, il y a de fortes chances de ne pas trouver ce point, à moins qu'on effectue un test point par point du domaine S.
C'est pour cela qu'on remplace la recherche du minimum par celui de a défini par:

$$a = \min \{t: v[x \in S / f(x) < t] > 0\} \tag{B.1}$$

v(A) est le volume de l'ensemble A, appelé généralement la mesure de Lebesgue.

On défini, alors, la région d'optimalité par :

$$R(e,m) = \begin{cases} (x \in S / f(x) < a + e) & \text{si a est fini} \\ (x \in S / f(x) < M) & \text{si a} \rightarrow \infty \end{cases} \tag{B.2}$$

avec e>0 et M<0.

Pour démontrer la convergence de l'algorithme, on ajoute une autre hypothèse :

H2. Pour n'importe quel sous ensemble A de S, où v(A) > 0, on a :

$$\prod_{k=0}^{\infty} [1 - \mu_k(a)] = 0 \tag{B.3}$$

c'est à dire, la probabilité d'omettre l'ensemble A répétitivement, lors de la génération de g(k), doit être nulle.

Théorème de convergence

On suppose la fonction f mesurable, et les hypothèses H1 et H2 satisfaites.

$$[x(k)]_{k=0}^{\infty} \tag{B.4}$$

est la suite générée par l'algorithme.

Alors:

$$\lim_{k \to \infty} p[x(k) \in R(e,M)] = 1 \tag{B.5}$$

car de l'hypothèse H1, si $x(k)$ et $g(k)$ appartiennent à $R(e,M)$, ceci implique que :

$$x(k') \in R(e,M) \quad \text{pour tout } k' \geq k+1 \tag{B.6}$$

ainsi : $$p[x(k) \in R(e,M)] \quad 1 - p[x(k) \in S / R(e,M)] \tag{B.7}$$

$$1 - p[x(k) \in S / R(e,M)] \geq 1 - \prod_{i=0}^{k} (1 - \mu_i(R(e,M)) \tag{B.8}$$

d'où : $$1 \geq \lim_{k \to \infty} p[x(k) \in R(e,M)] \geq 1 - \lim_{k \to \infty} \prod_{i=0}^{k} (1 - \mu_i(R(e,M))) \tag{B.9}$$

donc :

$$\lim_{k \to \infty} p[x(k) \in R(e,M)] = 1 \tag{B.10}$$

Remarque

Il existe plusieurs choix de la norme $D(x(k),g(k))$. Dans notre cas :

$$D(x(k),g(k)) = x(k) \pm g(k) \tag{B.11}$$

ANNEXE -C-

Génération des signaux numériques [17]

Chaque signal déterministe $x(t)$ peut être numérisé en l'échantillonnant, soit alors :

$$x(kT) = x(t)_{t=kT} \tag{C.1}$$

avec T: période d'échantillonnage.

Génération des signaux pseudo-aléatoires

Tout signal pseudo-aléatoire a pour relation de récurrence la relation suivante :

$$x(k-1) = [a\,x(k)]\ mod\ p \tag{C.2}$$

a et p étant des entiers.

p : nombre premier.
a : racine primitive de p.

Tout signal généré de la relation (C.2) est périodique, et de période p-1. De plus le signal :

$$x'(k) = x(k)/p \tag{C.3}$$

a pour densité de probabilité, la loi uniforme [0,1].

Génération de signaux possédant d'autres densités de probabilité

A partir d'un signal pseudo-aléatoire (C.2) à distribution uniforme, on peut générer d'autres signaux pseudo-aléatoires ayant d'autres distributions.

Exemple :

$$y(k) = \sqrt{2\sigma^2 \ln(1/x(k))} \tag{C.4}$$

si $x(k)$ suit la loi uniforme, alors $y(k)$ suit la loi de Reyleigh. De plus :

$$z(k) = y(k)\cos(2\pi\, x(k+1)) \tag{C.5}$$

suit la loi normale de Gauss.

Génération d'un bruit Gaussien

De la relation (C.3), la suite :

$$x_1(t) = x(t) / (2^n - 1) \tag{C.6}$$

suit la loi uniforme. Avec :

$$x(k+1) \equiv a\, x(k)\ [2^n - 1] \tag{C.7}$$

on prend :

$a = 131$
$x(0) = 12375$
$n = 16$

On calcule :

$$R(k) = \sqrt{2\sigma^2 \ln(1 / x_1(k))} \tag{C.8}$$

puis :

$$z(k) = R(k) \cos(2\pi\, x_1(k+1)) + b \tag{C.9}$$

$z(k)$ est le signal représentant le bruit Gaussien de moyenne b et de variance σ^2.

ANNEXE -D-

Règles de modification des poids dans l'apprentissage par renfort [30]

Dans ce type d'apprentissage, on a affaire à deux réseaux. Un réseau d'action (neuro-contrôleur), et un réseau d'évaluation.

Modification des poids du réseau d'action

Soit un neurone appartenant au réseau, ayant pour entrée le vecteur $X(k) = (x_1(k),...,x_n(k))$. Sa sortie $y(k)$ est calculée par :

$$y(k) = f\left[\sum_{i=1}^{n} W_i(k)x_i(k)\right] \qquad (D.1)$$

Les poids w_i changent en fonction du temps comme suit :

$$w_i(k+1) = w_i(k) + \alpha \hat{r}(k)e_i(k) \qquad (D.2)$$

avec :

α constante positive déterminant le taux de changement de W_i.

$\hat{r}(k)$ valeur du renfort à l'instant k, délivrée par le réseau d'évaluation.

$e_i(k)$ éligibilité à l'instant k correspondant à la ième entrée.

L'éligibilité e_i est calculée en utilisant l'équation aux différences suivante :

$$e_i(k+1) = \delta e_i(k) + (1-\delta)y(k)x_i(k) \qquad (D.3)$$

avec : $0 \le \delta < 1$

On note que chaque synapse a sa propre éligibilité.

Modification des poids du réseau d'évaluation

Pour produire la valeur du renfort $\hat{r}(k)$, le réseau d'évaluation doit déterminer une prédiction $p(k)$ d'un éventuel renfort, c'est à dire une fonction du vecteur d'entrée $X(k)$, à savoir :

$$p(k) = \sum_{i=1}^{n} v_i(k)\, x_i(k) \tag{D.4}$$

les $v_i(k)$ représentent les poids du réseau d'évaluation. Ces poids sont ajustés de manière à faire converger $p(k)$ vers une prédiction assez précise. La règle d'adaptation est la suivante :

$$v_i(k+1) = v_i(k) + \beta\left[r(k) + \gamma p(k) - p(k-1)\right]\bar{x}_i(k) \tag{D.5}$$

avec :

β constante positive déterminant le taux de changement de v_i.

$$0 < \gamma \leq 1$$

r(k) est le signal de renfort généré par le critère de performance,

$\bar{x}_i$ est la trace de x_i calculée par :

$$\bar{x}_i(k+1) = \lambda\,\bar{x}_i(k) + (1-\lambda)x_i(k) \tag{D.6}$$

où $0 \leq \lambda < 1$.

la sortie du réseau d'évaluation, représentant le signal de renfort approximé, est donnée par :

$$\hat{r}(k) = r(k) + \gamma p(k) - p(k-1) \tag{D.7}$$

c'est la même expression qui apparaît dans la relation (D.5).