N° d'ordre : 2400

THESE

Présentée à

L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE
Spécialité : GENIE ELECTRIQUE

par

Christophe FORGEZ
Ingénieur EUDIL

METHODOLOGIE DE MODELISATION ET DE COMMANDE PAR RESEAUX DE NEURONES POUR DES DISPOSITIFS ELECTROTECHNIQUES NON LINEAIRES

Soutenue le 8 Décembre 1998, devant la Commission d'Examen :

MM.	J. FAUCHER	Président et rapporteur
	C. IUNG	Rapporteur
	J.P HAUTIER	Directeur de thèse
Mme	B. SEMAIL	Examinatrice
MM	J. EVIN	Examinateur
	L. LORON	Examinateur
	F. PIRIOU	Examinateur

Remerciements

Les travaux présentés dans ce mémoire, ont été menés au Laboratoire d'Electrotechnique et d'Electronique de Puissance de Lille, dirigé par Monsieur le Professeur Christian ROMBAUT.

Je souhaite remercier les Professeurs qui, malgré leurs lourdes occupations, m'ont fait l'honneur de rapporter sur ce travail : Monsieur Jean FAUCHER, Professeur à l'ENSEEIHT de Toulouse, et Monsieur Claude IUNG, Professeur à l'ENSEM de Nancy.

Mes remerciements s'adressent également à Monsieur Luc LORON, Maître de Conférences Habilité à Diriger des Recherches à l'Université de Technologie de Compiègne, ainsi qu'à Monsieur Jean EVIN, PDG de la société AUTINOR LOGILIFT, qui ont bien voulu examiner mon travail.

Je tiens à remercier Monsieur Francis PIRIOU, Professeur à l'Université de Lille1, pour l'intérêt qu'il a bien voulu accorder à mon travail au travers de sa collaboration.

Que Monsieur Jean Paul HAUTIER, Professeur à l'ENSAM et directeur de ma thèse, trouve ici toute ma gratitude pour son expérience et les conseils scientifiques dont j'ai toujours bénéficié dans la bonne humeur.

Mes plus vifs remerciements vont à Madame Betty SEMAIL, Professeur à l'EUDIL, qui m'a encadré tout au long de ces trois années. Qu'elle trouve ici l'expression de ma sincère reconnaissance pour sa sollicitude, les qualités scientifiques et humaines dont elle a toujours fait preuve à mon égard.

Je tiens également à remercier vivement toute l'équipe du L2EP et plus particulièrement :

Monsieur Xavier CIMETIERE pour l'aide qu'il m'a apportée par son soutien informatique.

Les enseignants et doctorants du laboratoire, tout particulièrement Messieurs Bruno FRANCOIS, Emmanuel DELMOTTE, Frédéric GILLON, Francky HEMBERT, Christophe SAUDEMONT et Hervé ROISSE, pour l'atmosphère chaleureuse, amicale et néanmoins de travail, qu'ils ont su créer, ainsi que pour leurs nombreux conseils éclairés tant d'ordre théorique que pratique.

Madame Annick PENNEQUIN qui contribue largement au développement et au maintien de cette ambiance si agréable au sein du Laboratoire.

à ma famille

Chapitre 1
Position du problème

Chapitre 2
Les réseaux de neurones artificiels

Chapitre 3
La modélisation neuronale de systèmes électrotechniques

Chapitre 4
Commandes neuronales

INTRODUCTION GENERALE

Le domaine de l'électrotechnique recèle différents types de phénomènes non linéaires de par la construction et la constitution des machines ou leurs alimentations. Par conséquent, des outils de modélisation et de commande permettant de prendre en compte des non linéarités s'imposent. Bon nombre de méthodes capables de les traiter existent, mais souffrent de ne pas apporter de solutions universelles: soit elles apportent une modélisation complète et fine mais ne sont pas utilisables en commande, c'est le cas par exemple des calculs par éléments finis, soit elles s'appliquent parfaitement à la commande mais ne présentent pas de réel intérêt en modélisation, citons à titre d'exemple la logique floue. Nous avons, pour notre part, axé notre recherche sur la détermination d'un outil capable de répondre à ces attentes. C'est pourquoi nous nous sommes intéressés aux réseaux artificiels de neurones, non pas parce qu'ils bénéficient d'un effet de mode, mais parce qu'ils semblent correspondre à l'outil envisagé. Nous nous sommes efforcés d'appliquer les propriétés de modélisation neuronale à différents phénomènes rencontrés en génie électrique et notamment en électromécanique et électromagnétisme. L'objet de cette thèse est de montrer à la fois les avantages des réseaux de neurones tout en évoquant leurs limites.

La principale force des réseaux de neurones, qu'on peut cataloguer dans l'intelligence artificielle, réside dans leur capacité d'apprentissage. Cet apprentissage permet de modéliser des caractéristiques linéaires, non linéaires ou discontinues, à partir d'échantillons, en utilisant des méthodes d'optimisation non linéaires telles que la rétropropagation du gradient, les algorithmes génétiques, ... ces méthodes sont nécessaires pour déterminer au mieux tous les degrés de liberté du réseau de neurones.

L'intelligence artificielle, largement exploitée dans les ouvrages et autres articles scientifiques nourrit l'ambition des chercheurs dans leur quête de l'outil offrant la modélisation universelle. Néanmoins, cet aspect séduisant de telles méthodes généralistes présente le double inconvénient de nécessiter des temps de calculs importants tant à l'apprentissage qu'à l'émulation à cause d'une architecture naturellement gourmande face aux dispositions technologiques actuelles. Nous proposons alors une méthode d'apprentissage, basée sur une régression non linéaire ainsi que sur une optimisation de la structure neuronale.

Cette approche, certes moins généraliste, présente l'avantage de ramener les temps d'apprentissage de quelques heures à quelques minutes voire quelques secondes. Nous insistons également sur l'allégement des architectures neuronales en vue de leur implantation dans des codes de calculs pour des simulations ou des asservissements en temps réel.

En matière de commande, les possibilités offertes par les modèles neuronaux sont nombreuses ; il est alors important d'élaborer un formalisme de construction de commandes linéarisantes à partir ces modèles. Cette démarche s'inscrit dans la recherche d'une commande universelle capable de contrôler n'importe quel système sans besoin de connaissance sur ce dernier. En effet, s'il est possible de modéliser des parties ou la totalité d'un système, il est alors intéressant d'utiliser ces modèles directement dans la commande. Avant d'appliquer cette technique, encore faut il déterminer, au sens des entrées-sorties, quelle doit être la caractéristique à apprendre en vue de l'utilisation en commande. Pour cela, nous avons fait appel au graphe informationnel causal qui permet de déterminer la structure de la commande par inversion directe ou indirecte du graphe. Des apports au point de vue représentation des graphes informationnel causaux sont suggérés notamment pour prendre en compte le choix de la technique de numérisation et de l'univocité des systèmes non linéaires.

Néanmoins le respect de la non inversion des causalités limite notre ambition et nous a contraint à explorer deux types de commandes neuronales : la commande neuronale partielle et la commande neuronale totale.

Enfin des résultats expérimentaux sont présentés tant en modélisation qu'en commande, montrant la qualité de notre modélisation neuronale et prouve la faisabilité et l'efficacité de tels types de contrôleurs.

La première partie est consacrée à la position du problème. Après avoir inventorié et répertorié les différents systèmes, nous nous attacherons à exposer les méthodes de modélisation qui s'adaptent aux différents cas, en particulier à la catégorie des systèmes non linéaires. Cet exposé nous amène naturellement à placer la méthode de modélisation neuronale en position d'approche universelle, quant à l'établissement d'un modèle et son utilisation en commande.

Le second chapitre porte sur la description du fonctionnement des réseaux de neurones, des différentes architectures existantes ainsi que sur les principes d'apprentissage. Des méthodes usuelles d'apprentissage y sont exposées afin de mettre en lumière leurs avantages et inconvénients. Au sein de ce chapitre, une seconde partie explique, démontre et formalise notre méthode dite d'initialisation qui permet d'optimiser la structure mathématique en fonction de la nature de la caractéristique à modéliser.

Le troisième chapitre applique les concepts précédemment développés, relatifs à la modélisation neuronale. Trois applications concrètes sont traitées : modélisation d'une inductance saturable, de cycles majeurs d'hystérésis et de charges mécaniques.

Le dernier chapitre définit naturellement la commande non linéaire d'un système quelconque grâce à l'inversion causale permise par la phase d'apprentissage préalable. Cette démarche nous paraît quasi irréalisable de manière aussi formelle avec toute autre technique. En revanche, nous évoquons les limites de validité de ce concept, afin de fixer le cadre des applications possibles. Enfin pour pallier aux erreurs d'apprentissage ou aux éventuelles évolutions des paramètres du processus, une méthode d'adaptation en ligne de la commande neuronale totale est également présentée.

Chapitre 1

Position du problème

1 Introduction

Les choses ne sont pas forcément comme elles nous apparaissent et ne sont pas toujours comme on voudrait qu'elles soient. Ce constat nourrit l'idée que nous ne maîtrisons pas la compréhension de notre univers, ne serait ce que par le manque flagrant de connaissances face à l'infini. Néanmoins, malgré l'aspect chaotique de notre univers, les scientifiques s'attachent actuellement à penser qu'il existe malgré tout un ordre dans ce chaos, c'est à dire que chaque cause a une conséquence. Les principales questions qui viennent alors à l'esprit sont les suivantes : peut-on représenter ce chaos et comment s'y prendre ?

Evidemment, ces questions métaphysiques apparaissent bien loin des préoccupations techniques quotidiennes. Cependant, il apparaît que, dans notre domaine d'investigation, les modèles utilisés, dits cartésiens, ne représentent pas toujours très bien la réalité des phénomènes physiques. Le problème ici soulevé est alors de savoir s'il existe une représentation globale pour un système ; comment l'établir de sorte que celle-ci soit la plus réaliste possible et ne soit pas influencée par les a priori de son concepteur ? De plus, à l'aide de cette représentation, est-il possible d'imposer réellement une commande à des systèmes complexes?

Pour aborder ces questions, nous redéfinissons dans un premier temps quels sont les différents types de systèmes afin d'évaluer les difficultés et les possibilités de représentation. Plusieurs méthodes de modélisation sont présentées afin de donner un aperçu des techniques employées pour décrire une caractéristique physique. Nous essayons d'en faire une synthèse dans le but de proposer un modèle de représentation général basé sur le formalisme neuronal. Dans un second temps, nous déterminerons quelle technique employer pour commander un processus donné. Nous insistons sur le fait qu'une commande ne peut raisonnablement être conçue qu'avec l'aide d'un modèle de connaissance de ce processus. En définitive, ce chapitre explicite les possibilités quant à l'élaboration d'un modèle général et son utilisation en commande.

2 Représentation des systèmes

2.1 Graphe Informationnel Causal (G.I.C)

Afin de définir quelles sont les grandeurs d'états ainsi que les causalités qui interviennent au sein d'un système il est nécessaire d'utiliser une représentation des modèles de connaissance dont nous disposons. Nous utilisons à cet effet le graphe informationnel causal (G.I.C) [MASON 53][HAUTIER 96]. Le G.I.C est une représentation symbolique entre les relations des grandeurs physiques d'un système. Cette transcription est en fait un modèle de comportement descriptif des interdépendances entre les différentes grandeurs évolutives dans le temps de ce modèle. Le graphe relie entre eux les éléments de transformation de ces grandeurs que nous appelons processeurs et qui sont représentés par une simple ou double 'bulle' selon qu'ils sont linéaires ou non, comme l'illustre la figure I.2.1.

a) *b)*

Fig I.2.1 : Processeur linéaire a) et non linéaire b) du graphe informationnel causal

Le processeur agit suivant le principe de causalité, c'est à dire qu'un effet est toujours produit par une cause. De plus, on distingue les relations rigides c'est à dire non dépendantes du temps, représentées sur la figure I.2.2 par une double flèche et les relations causales, dépendantes du temps, représentées par une simple flèche dans le sens de la causalité.

$S = K * E$ avec K constant

$$\beta_n \frac{d^{(n)}S}{dt^{(n)}} + \cdots \beta_1 \frac{dS}{dt} + \beta_0 = E$$

a) *b)*

Fig I.2.2 : Relation rigide a) relation causale b)

La symbolisation des flèches est à rapprocher de la théorie des ensembles. En effet la double flèche décrit une relation à opération bijective, c'est à dire qu'un élément de l'ensemble de sortie provient du résultat d'une opération sur l'ensemble des entrées. Dans ces conditions, il est possible de déterminer quel est le résultat de l'opération d'entrée

correspondant à une valeur de sortie ; en revanche, la réciprocité n'existe pas (ou la bijection entrée-sortie) dans le cas de plusieurs entrées. Dans le cas d'une relation causale, la simple flèche indique que la relation n'est pas univoque, c'est à dire qu'à une même sortie peut correspondre plusieurs entrées différentes. Prenons l'exemple du volume d'eau débité par une pompe remplissant une cuve. A un instant donné, on constate un volume d'eau de 1000 litres. Cette information ne peut cependant pas nous renseigner sur le débit de la pompe car nous ne connaissons ni la durée de remplissage ni la valeur du volume initial.

Cette exemple nous indique qu'une relation causale ne peut être inversée directement en vue d'établir quelle est l'entrée correspondant à une sortie. Cette remarque est en fait à l'origine de la théorie de la contre réaction, qui traduit la nécessité de comparer une grandeur d'état par rapport à une référence pour pouvoir 'remonter' le temps, c'est à dire inverser indirectement la causalité. Ainsi, la représentation d'un système à l'aide du G.I.C permet de déterminer les structures de commande appropriées.

2.2 Classification des systèmes

Les systèmes physiques sont le siège de phénomènes dynamiques, que l'on peut interpréter au travers d'équations différentielles, linéaires ou non. Avant de modéliser un système, il convient de définir sa nature afin d'approcher au mieux la réalité. Rappelons simplement qu'un modèle de connaissance n'en est qu'une interprétation. Dans ces conditions, les hypothèses incontournables le rendent nécessairement imparfait.

2.2.1 Les systèmes linéaires

Les systèmes linéaires qui se définissent par leurs propriétés de proportionnalité et de continuité, sont représentables par les équations différentielles suivantes :

- sous forme continue

$$a_{na}\frac{d^{na}y}{dt^{na}}+\cdots+a_1\frac{dy}{dt}+a_0y=b_{nb}\frac{d^{nb}u}{dt^{nb}}+\cdots+b_1\frac{du}{dt}+b_0u \qquad (1.1)$$

- ou sous forme discrète

$$\alpha_{n\alpha}y(t-n\alpha Te)+\cdots+\alpha_1y(t-Te)+\alpha_0y(t)=\beta_{n\beta}u(t-n\beta Te)+\cdots+\beta_1u(t-Te)+\beta_0u(t) \qquad (1.2)$$

Classiquement, ces systèmes sont mis sous forme d'équations d'état :

$$\begin{aligned} \dot{X} &= AX + BU \\ Y &= CX + DU \end{aligned} \tag{1.3}$$

où X représente le vecteur des états du système, U le vecteur de commande, A la matrice de transition, B la matrice d'application directe des commandes, Y le vecteur de sortie.

2.2.2 Les systèmes non linéaires

Les systèmes non linéaires se caractérisent par l'absence ou non des propriétés des systèmes linéaires. En d'autres termes, un système linéaire est un cas particulier de l'ensemble des systèmes non linéaires. Néanmoins, si la proportionnalité et la continuité ne sont pas forcément le lot des systèmes non linéaires, la constance est de rigueur, c'est-à-dire que les lois qui régissent un système non linéaire sont immuables en fonction du temps.

Ces systèmes sont mis sous la forme d'équations d'état :

$$\begin{aligned} \dot{X} &= f(X,U) \\ Y &= g(X,U) \end{aligned} \tag{1.4}$$

avec f et g des fonctions non linéaires quelconques.

2.2.3 Les systèmes non stationnaires

Les systèmes non stationnaires sont les plus difficiles à caractériser et à modéliser. Leurs structures peuvent être linéaires ou non linéaires, mais changeantes en fonction du temps. Ils peuvent également être mis sous la forme d'équations d'état de manière instantanée ou sur une fenêtre temporelle :

$$\begin{aligned} \dot{X} &= f(X,U,t) \\ Y &= g(X,U,t) \end{aligned} \tag{1.5}$$

D'après les définitions précédentes, il est possible d'affirmer que les systèmes linéaires et non linéaires sont des cas particuliers des systèmes non stationnaires.

***Remarque** : les systèmes non stationnaires dont l'évolution des paramètres au cours du temps dépend des états du système, pourront être qualifiés de systèmes non linéaires. Prenons deux exemples différents afin de mettre en relief la nuance précédente. Par exemple, au cours d'un fonctionnement prolongé ou intensif d'une machine électrique, les résistances ont tendance à*

chauffer et, par conséquent, à augmenter de valeur ; dans ce cas, l'évolution des paramètres du système est une fonction dépendante du courant dont la valeur efficace peut dépendre du temps; par conséquent, le système est qualifié de non stationnaire.
Autre exemple, considérons un train qui roule sur une voie horizontale puis monte une pente. On pourrait penser que le couple résistant est alors une grandeur aléatoire, considérée comme une perturbation. Cependant, la pente que doit monter le train, se situe à une position fixe dans l'espace, c'est pourquoi le couple résistant devient fonction de la position du train, par conséquent de sa vitesse. Ce système n'est donc pas non stationnaire mais non linéaire. Le choix de considérer ce genre de système comme non linéaire accroît plus ou moins la difficulté de le modéliser .

Cette remarque insiste donc sur la possibilité du choix de représentation d'un système par différents modèles. La figure I.2.3 illustre les différentes possibilités de modélisation.

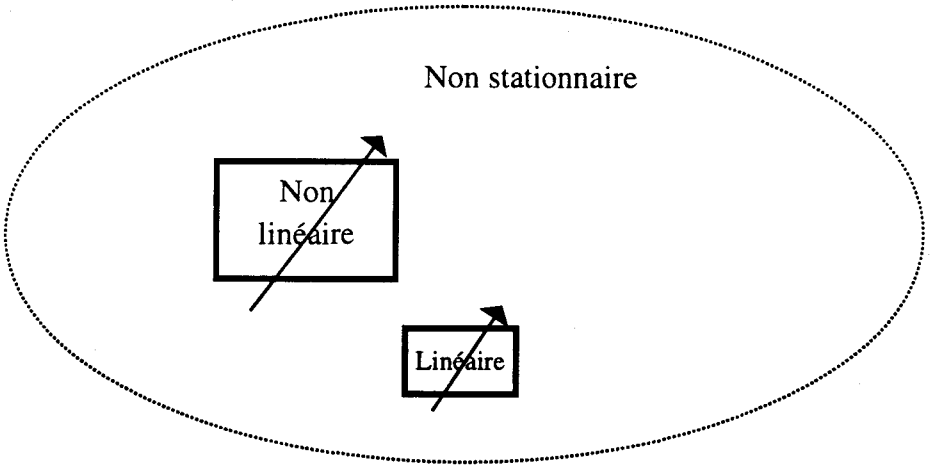


Fig I.2.3 : Possibilités de représentation d'un système non stationnaire

Bien qu'un système non stationnaire soit dépendant du temps, on peut symboliser l'univers de ce système par un ensemble, mais non borné. Ce même système peut être représenté en partie par un modèle non linéaire que l'on modifie au cours du temps afin de couvrir le domaine du système non stationnaire. De même, on peut recourir à un modèle linéaire temporellement adaptatif. La décision de prendre un modèle plutôt qu'un autre dépend alors essentiellement de la puissance de calcul mise à disposition ainsi que de la connaissance dont on dispose du système.

3 Modélisation de caractéristiques rigides non linéaires

Nous allons montrer dans ce paragraphe qu'on peut utiliser différentes approches selon le compromis temps de calcul-précision que l'on souhaite respecter afin de modéliser des fonctions non linéaires. Ces fonctions non linéaires peuvent être symbolisées par des processeurs comme ceux de la figure I.2.1.b. Les différentes approches sont présentées ici par ordre croissant de complexité. Le but est de montrer que ces méthodes ont en commun la mise en série de fonctions qui conduit naturellement à la structuration neuronale.

3.1 Approche par 'morceaux'

La première approche consiste à linéariser partiellement ou globalement les fonctions à caractériser. La linéarisation globale, par plans d'expériences par exemple, donne des résultats approchés pour la modélisation, mais offre un excellent compromis temps de calcul-précision lorsqu'elle est utilisée en tant que méthode d'optimisation [GILLON 97]. La linéarisation partielle consiste à considérer la fonction par morceaux qu'on approche par des droites ou des hyperplans suivant la dimension de la caractéristique réelle. Le principe de partager un domaine d'étude en plusieurs ensembles peut évidemment être utilisé pour caractériser des fonctions multivariables. Néanmoins, si ce principe permet de déterminer un modèle au sein des différents domaines, il reste à définir comment s'effectue le passage d'un modèle à un autre. La fusion des différents modèles peut être formalisée en utilisant l'approche multi-modèles [DELMOTTE F 97]. La logique floue, que nous décrirons au travers d'un paragraphe, est fondée sur cette approche.

3.2 Séries de fonctions

La seconde approche consiste à modéliser des caractéristiques quelconques à partir d'équations paramétriques dont la complexité varie à la fois selon celle de la caractéristique réelle, mais également en fonction de l'usage et ou de la finesse du modèle. Cette approche s'étend donc du modèle paramétrique à la décomposition en séries de fonctions.

La méthode la plus directe est de décrire les lois fondamentales d'un système afin d'en établir un modèle de connaissance. Ces lois, si elles ne sont pas connues, peuvent être modélisées à l'aide d'une interpolation polynomiale (si le nombre d'échantillons n'est pas trop important) ou à l'aide d'une régression (si le nombre d'échantillons devient important).

Dans ce cas l'objectif de la modélisation est d'ajuster les paramètres du modèle de façon à minimiser l'erreur entre ce dernier et les valeurs réelles de la fonction à estimer. A partir de méthodes numériques il est possible de déterminer un ajustement optimal non biaisé des coefficients du modèle en vue d'identifier les paramètres physiques du système [RICHALET 91].

3.3 Formalisme flou

Nous allons au cours de ce paragraphe, décrire comment il est possible de mettre en place un modèle flou. Etant donné que la logique floue fait partie intégrante de l'intelligence artificielle, ce paragraphe nous permettra de nous inspirer de cet aspect et d'expliquer par la suite pourquoi nous avons choisi d'utiliser les réseaux de neurones.

La logique floue travaille avec des ensembles approximatifs : 'plutôt grand', 'assez petit', ... Ces ensembles peuvent être traduits mathématiquement par une fonction d'appartenance qui définit, pour tout élément d'une variable, le degré d'appartenance à ces mêmes ensembles. Ce degré varie de 0 à 100% et, pour des raisons de facilité et de rapidité dans l'exécution des calculs, les fonctions utilisées sont triangulaires ou trapézoïdales ; on peut néanmoins utiliser des fonctions gaussiennes, mais cela accroît les temps de calcul.

Il est également nécessaire de définir quelles sont les variables ainsi que les ensembles flous de ces variables sur l'univers étudié. Toutes les valeurs des variables en jeu doivent être considérées, c'est à dire qu'il ne peut y avoir de 'trous' dans les ensembles flous définis. En effet, un système flou comporte des règles associées à des ensembles et, par conséquent, il ne peut les extrapoler sur des ensembles non définis. Cela implique le chevauchement des fonctions d'appartenance. Une fois les ensembles définis, on élabore le profil des prémisses (triangulaire, trapéizoïdale, ...), c'est à dire les degrés d'appartenance des éléments aux différents ensembles flous. Cette étape s'appelle la 'fuzzyfication', et s'applique aux variables d'entrée et de sortie du système. Nous allons décrire cette étape au travers de l'illustration de la figure I.3.1, qui décrit la 'fuzzyfication' d'un système à deux entrées et une sortie. Les domaines des trois variables sont partagés en sept ensembles *(GN, MN, PN, ZE, PP, MP, GP)*. Tous les ensembles se chevauchent de manière à assurer une équiprobabilité de chance qu'un élément appartienne à un ensemble. Par exemple la valeur -7 de la variable *X1* appartient à 70% à l'ensemble *PN*, et à 30% à l'ensemble *ZE*.

Ensuite, on définit les règles qui permettent d'associer les ensembles d'entrée aux ensembles de sortie. Ces règles sont élaborées à partir de la base de connaissance du système.

Comme dans le cas de la logique classique, on dispose d'opérateurs pour mettre en équations ces règles : *(NON, ET, OU)*, mais au lieu de ne traiter que des 0 ou 1, la logique floue traite des valeurs comprises entre 0 et 1, telles que :

$$\begin{aligned} A\ \textbf{ET}\ B &= min(\mu A(e),\mu B(e)) \\ A\ \textbf{OU}\ B &= max(\mu A(e),\mu B(e)) \\ \textbf{NON}\ A &= 1-\mu A(e) \end{aligned} \tag{1.6}$$

où $\mu A(e)$ indique le degré d'appartenance de l'élément e à l'ensemble flou A.

La 'défuzzyfication' est l'opération symétrique de la 'fuzzyfication'. Elle consiste à calculer une valeur unique de la variable à partir des différents degrés d'appartenance aux ensembles de sortie, obtenus par l'application des règles. Ce calcul est également une évaluation floue.

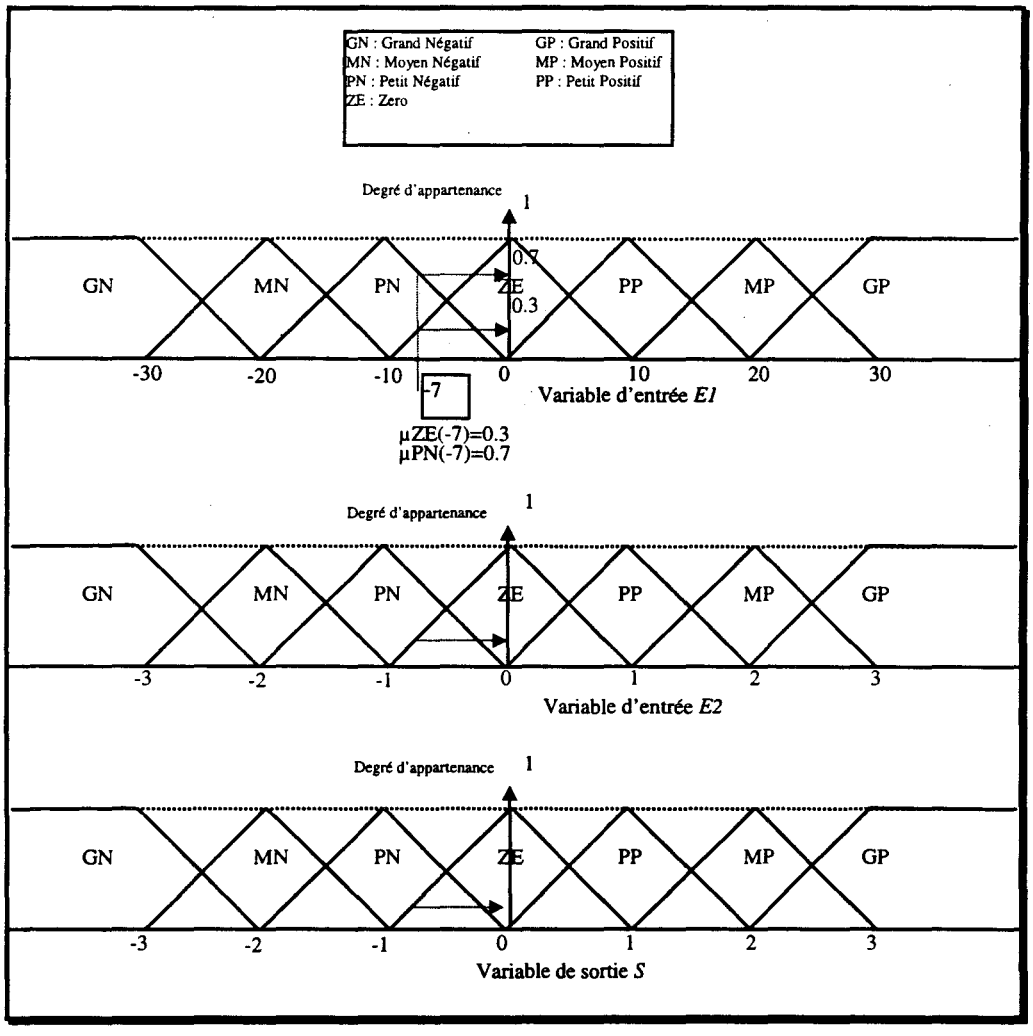


Fig I.3.1 : Fuzzyfication

Une démarche consiste à prendre la valeur maximale des degrés d'appartenance, mais cette méthode est délaissée au profit du calcul du centre de gravité de ces degrés. Cette démarche a l'intérêt de ne pas recentrer les valeurs de sortie sur les valeurs nominales des ensembles de sortie. Le calcul d'un centre de gravité s'effectue de la façon suivante :

$$e = \frac{\sum_{i=1}^{n} v_i \mu(v_i)}{\sum_{i=1}^{n} \mu(v_i)} \qquad (1.7)$$

avec $\mu(v_i)$ la valeur d'appartenance, et v_i la valeur e de la variable E.

L'élaboration d'un modèle flou se décompose en trois phases : la 'fuzzyfication', l'évaluation des règles et la 'défuzzyfication' (fig I.3.2) [DUBOIS 95].

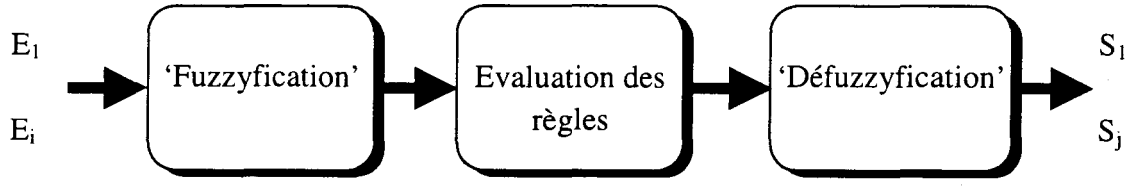


Fig I.3.2 : Etapes de l'élaboration d'un modèle flou

Ainsi, un modèle flou permet de reconstituer de manière universelle n'importe quelle caractéristique stationnaire [WANG 92a].

L'intérêt porté à la logique floue, depuis trois décennies, est certainement dû à sa simplicité d'emploi, de compréhension et de calcul. Néanmoins le choix des ensembles, des prémisses et des règles relève essentiellement de l'intuition. On comprend aisément que la forme et le nombre des fonctions d'appartenance influent sur la qualité de la modélisation. Pour rendre plus systématique la conception d'un modèle flou, des recherches ont permis de mettre en œuvre une méthode d'extraction automatique des règles et de génération des ensembles simplement à partir des données (fig I.3.3) [COX 97] [WANG 92b]. Cette méthode est fondée essentiellement sur des méthodes itératives, similaires aux méthodes d'apprentissage des réseaux de neurones exposées au chapitre suivant. Elles consistent à établir des règles entre toutes les variables d'entrée et de sortie et d'éliminer systématiquement les variables qui ne jouent pas de rôle au sein des différentes règles.

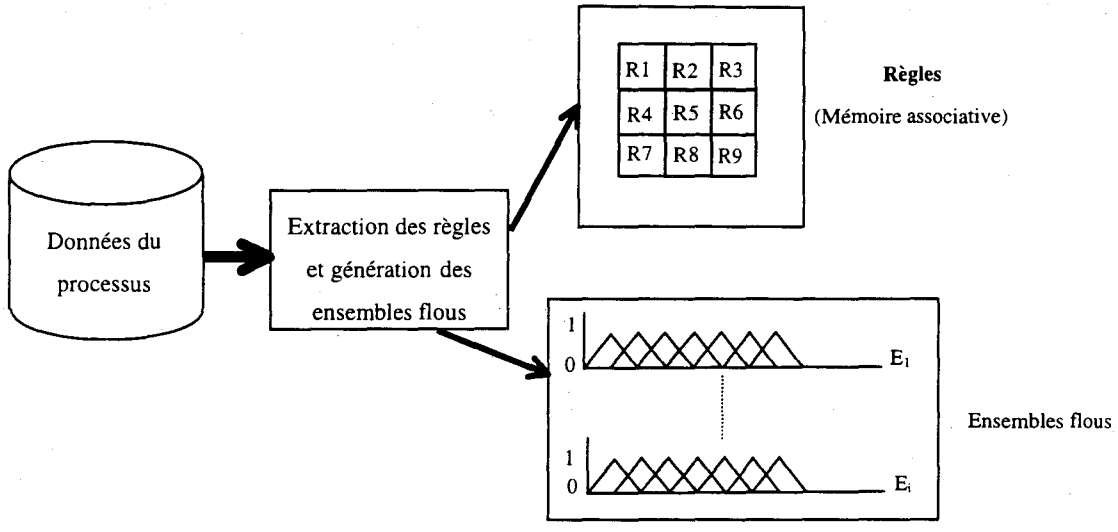


Fig I.3.3 : Extraction automatique des ensembles et des règles flous

3.4 Synthèse

Le lecteur ne manquera pas de trouver dans la suite de ce travail de nombreuses idées inspirées à la fois de la logique floue et des décompositions en série de fonctions. Ceci n'est pas le fait intentionnel d'imiter des méthodes qui semblent apporter des solutions dans le domaine de la modélisation non linéaire, mais est issu d'une réflexion qui conduit à penser que toutes les méthodes exposées précédemment sont liées entre elles par des constructions mathématiques et qu'elles ne sont finalement que des cas particuliers les unes des autres, ce qu'illustre la composition figure I.3.4.

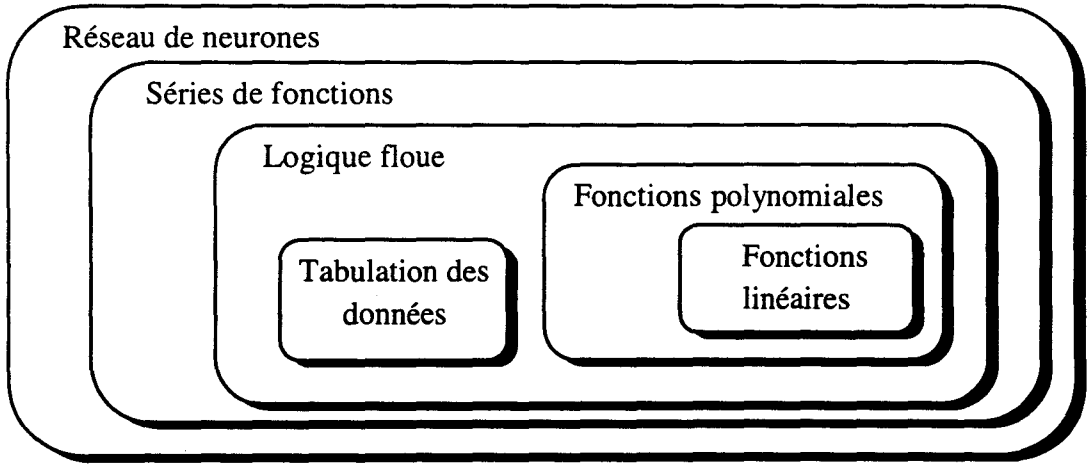


Fig I.3.4 : Appartenances mutuelles des différents modèles

Sur cette figure nous avons placé les réseaux de neurones au plus haut niveau dans la hiérarchie, car nous verrons par la suite que ceux ci possèdent les propriétés des autres outils de modélisation.

Nous invitons également le lecteur à noter que le développement en séries de fonctions se rapproche le plus de la modélisation neuronale. Ceci permettra d'emprunter des propriétés bien connues des séries de fonctions pour les extrapoler aux réseaux de neurones.

4 Méthodes de commande pour prendre en compte les non linéarités

4.1 Nécessité d'une commande non linéaire

Les systèmes électrotechniques sont, par nature, des systèmes non linéaires et ce à plusieurs titres. Les machines électriques, par exemple, sont régies par des phénomènes physiques non linéaires, telle la saturation magnétique des matériaux. De plus, comme ces machines, chargées mécaniquement, subissent les non linéarités des systèmes qu'elles entraînent, ceci se répercute sur les grandeurs de réglage.

En ce qui concerne leur alimentation, elle provient soit de turboalternateurs, (dans ce cas les grandeurs électriques sont sinusoïdales), ou d'une association source - convertisseur statique. Les signaux sont alors bien souvent discontinus et, dans le cas d'une source continue, on observe généralement les comportements suivants :

- la source continue est construite à partir d'une source alternative redressée, auquel cas on observe généralement des résidus d'alternances.
- la source continue provient d'une batterie, dont l'état de charge est difficilement identifiable en raison de la complexité des phénomènes chimiques qui interviennent selon l'utilisation et le vieillissement [CAUMONT 97].
- la source est supposée parfaitement continue mais une utilisation en régime dynamique peut la rendre non linéaire, ainsi a-t-on des fluctuations des tensions aux bornes des condensateurs en entrée d'un onduleur.

Rappelons également que dans les systèmes électrotechniques, les convertisseurs statiques interfacent une alimentation au système à piloter. Comme l'électronique de puissance est une électronique de commutations, qui subit les limitations intrinsèques à

l'alimentation, celle-ci génère, en amont comme en aval, des perturbations en ligne, ainsi qu'un rayonnement électromagnétique non négligeable, ce qui rend le système global encore plus complexe.

La figure I.4.1 résume ce paragraphe en indiquant la présence des non linéarités tant en amont qu'en aval des systèmes électrotechniques.

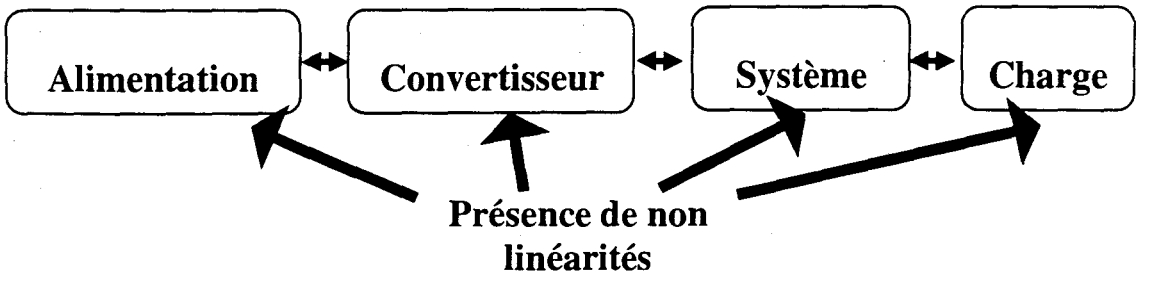


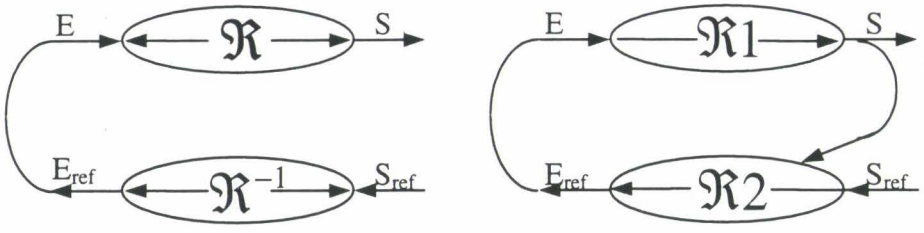
Fig I.4.1 Schéma indiquant la présence de nonlinéarités

4.2 Définition d'une structure de commande

Comme nous l'avons vu précédemment, le principe d'inversion du processus pour sa commande, réside dans le changement des sens des flux des grandeurs physiques et nécessite le respect de quelques règles. En dépit de la méthode adoptée, définir une commande sous une forme ou sous une autre, est la recherche implicite d'un modèle inverse du processus à conduire. En d'autres termes, c'est l'expression d'une volonté d'inverser les causalités : 'puisque l'on connaît l'effet de la cause, il n'y a qu'à créer la bonne cause pour avoir le bon effet'.

Pour un processeur élémentaire donné, l'inversion d'une relation entrée-sortie détermine une relation de commande pour le dispositif physique ainsi modélisé. La commande revient donc à permuter l'orientation des variables concernées et à déterminer le modèle mathématique inverse du processus. En pratique, le principe de causalité naturelle oblige à réaliser cette opération avec des artifices différents selon la nature et la complexité du modèle mathématique ; en effet, comme l'évolution de tout système provient d'une intégration globale, vouloir imposer cette évolution suppose la possibilité de pouvoir disposer de la dérivation qui, par essence, est une opération physiquement irréalisable. La structure de commande s'obtient, de manière systématique, en appliquant les principes précédents au modèle GIC du processus à commander. En partant de la grandeur à asservir, le chemin causal du processus est 'remonté', en y inversant les relations caractéristiques des processeurs

rencontrés, et ceci, jusqu'à la grandeur de réglage. La technique d'inversion consiste, pour un processeur quelconque, à permuter les orientations de la grandeur à commander, de la grandeur de réglage et de la causalité de la relation par traitement direct ou indirect. Les autres grandeurs influentes sont conservées, mais agissent généralement en compensation ou en linéarisation. Lorsqu'elles sont inaccessibles, on peut recourir à l'observation ou vérifier que les propriétés de l'inversion indirecte suffisent à limiter leurs effets.



a) *b)*

Fig I.4.2 : Commande d'un système rigide a) et d'un système causal b)

La figure I.4.2 illustre les deux types d'inversion qu'on rencontre dans la commande des systèmes linéaires. Ce formalisme peut bien sûr être étendu aux systèmes non linéaires; néanmoins il est nécessaire de rajouter aux relations non linéaires la notion d'inversibilité en plus de la notion de rigidité. Prenons l'exemple d'une saturation, bien que cette relation non linéaire soit rigide, elle n'est pas pour autant inversible, car non univoque. Nous détaillerons, dans le quatrième chapitre, le formalisme à adopter pour les systèmes non linéaires.

Cet énoncé suppose implicitement une commande monovariable, mais s'applique évidemment au cas multivariable.

4.3 Relations de commande

4.3.1 Commande par modes glissants

Une fois la structure de commande définie à l'aide de la méthodologie décrite précédemment, il reste alors à déterminer les relations de commande, c'est à dire quel type de correcteur nous devons (ou voulons) utiliser.

La manière la plus intuitive de commander un système quelconque est la commande 'tout ou rien' encore désignée sous le vocable de commande à mode glissants [BÜHLER 86]. L'électronique de puissance est très bien adaptée à cette technique car elle dispose d'un

organe de commutation, l'interrupteur électronique, qui s'adapte tout à fait aux commandes envoyées sur le processus, c'est à dire aux commandes booléennes. En fait, cette technique consiste simplement à comparer la sortie du processus par rapport à la consigne, si cette dernière est supérieure à la sortie il faut augmenter la valeur de la commande, dans le cas contraire, on diminue la valeur de la commande. L'avantage de la commande tout ou rien est de présenter un caractère universel par rapport à n'importe quel type de processus linéaire, non linéaire ou même non stationnaire. Néanmoins, la commande booléenne est équivalente à une correction de type proportionnelle dont le gain serait infini. Dans ces conditions, les effets des non linéarités et des perturbations sont implicitement rejetés et la commande est naturellement robuste. Cette méthode a donc l'avantage de faire suivre instantanément la grandeur de réglage sur la grandeur de consigne et, de plus, permet de réduire l'ordre du système global. Cette technique est très simple à mettre en œuvre mais présente l'inconvénient de solliciter de façon intensive les interrupteurs des convertisseurs statiques lorsque le système est proche du régime permanent car la fréquence à laquelle s'applique la commande tend naturellement vers une valeur infiniment grande. Une solution permettant de contrecarrer cet aspect néfaste est d'introduire un hystérésis dans la loi de commande. Toutefois, la valeur rendue libre de la fréquence pose des problèmes de nuisances harmoniques et, un cycle d'hystérésis de largeur trop importante fait perdre au système toutes les qualités précitées de sorte que la solution est assez peu utilisée en électrotechnique.

4.3.2 Commande par logique floue

4.3.2.1 Utilisation d'un modèle inverse

La commande par logique floue est fondée sur l'expérience ou l'analyse d'un expert. C'est une manière 'intelligente' de tenir compte des caractéristiques d'un système sans pour autant devoir recourir à une identification des paramètres ; cette technique permet d'appréhender naturellement des problèmes non linéaires.

Pour concevoir une commande floue, l'expert doit définir les règles de commande, c'est à dire ce qu'il ferait dans tel ou tel cas. Ces règles sont 'fuzzyfiées' et 'défuzzyfiées' comme nous l'avons vu au paragraphe 3.2.3 [DUBOIS 95].

Néanmoins, l'expert doit déterminer les variables mises en jeu à la fois dans ses règles et dans son correcteur. Pour cela il est nécessaire de compter le nombre de variables d'état.

Prenons par exemple un système linéaire du premier ordre constitué par un circuit *r,l*. Dans la mise en équation de ce circuit interviennent le courant *i* et sa dérivée $\frac{di}{dt}$, la commande restant la tension *u*. Il est alors possible de définir un correcteur flou comme l'indique le schéma de la figure I.4.3.

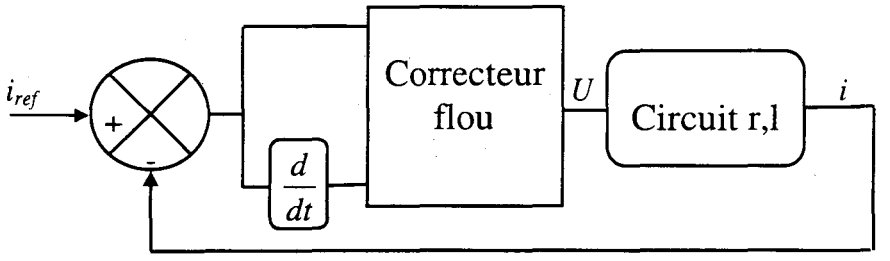


Fig I.4.3 : Schéma d'une commande floue

Le simple fait de trouver une dérivée dans la réalisation du schéma de commande signifie que la conception même du correcteur est également fondée sur l'utilisation d'un modèle inverse. Néanmoins, comme on est également confronté à l'incapacité d'inverser des relations causales, le calcul de la commande ne peut s'effectuer qu'en terme de variations (ou écarts entre une consigne et la grandeur de sortie) par rapport à un point de fonctionnement.

Dans le cas de systèmes d'ordre supérieur à un, on se limite à deux variables d'état floues, pour n'avoir que la dérivée première à calculer ; en effet la prise en compte des dérivées d'ordre supérieur rend l'asservissement d'autant plus sensible au bruit.

4.3.2.2 Elaboration de la commande floue

Pour comprendre comment est élaborée une commande floue, illustrons le cas d'un système à deux variables d'état *E1, E2* dont le couple de valeurs est représenté par le point noir de la figure I.4.5. On considère que la table des règles ci-dessous a été préalablement définie par un expert. Les notations sont les mêmes que celles employées à la figure I.3.1.

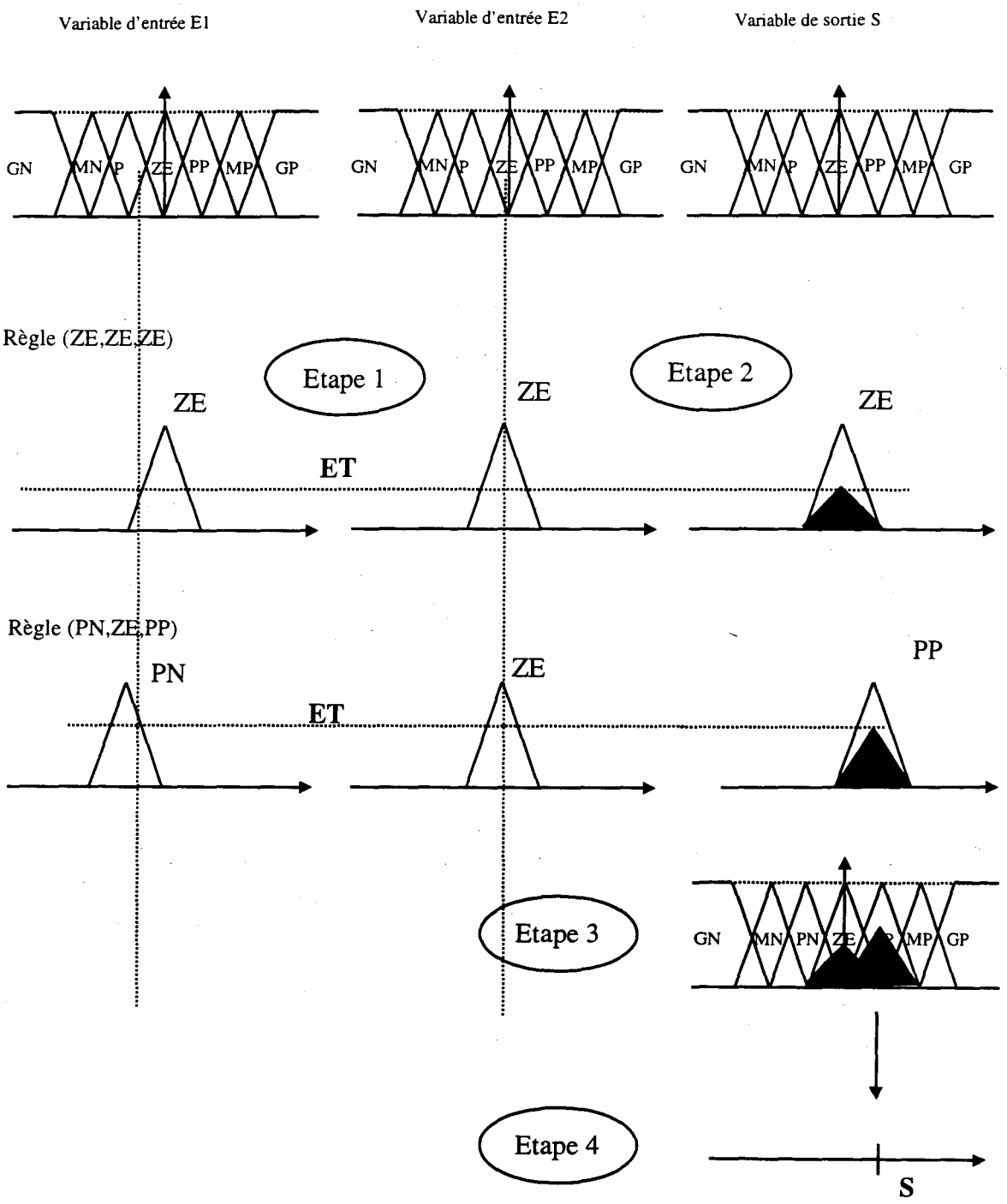


Fig I.4.4 : Etapes de la détermination d'une commande floue

E1\E2	GN	MN	PN	ZE	PP	MP	GP
GN				GP			ZE
MN				MP		ZE	
PN				PP	ZE		
ZE	GP	MP	PP	ZE	PN	MN	GN
PP			ZE	PN			
MP		ZE		MN			
GP	ZE			GN			

Fig I.4.5 : Table des règles de commande

Dans l'exemple présenté, le point de fonctionnement se situe sur deux règles *: (ZE,ZE, ZE)* et *(ZE,PN,PP).* Il est nécessaire de les 'défuzzyfier' pour connaître leurs degrés de pertinence. Ainsi l'étape 1 de la figure I.4.4 décrit la 'défuzzyfication' des variables *E1* et *E2* ; il s'agit de déterminer quels sont les degrés d'appartenance de ces variables respectivement aux ensembles *ZE* et *ZE* pour la première règle, et aux ensembles *ZE* et *PN* pour la seconde. L'étape 2 consiste à transposer ces degrés d'appartenance sur l'ensemble de sortie suivant les règles définies par l'équation *I.7.* Ensuite, on récupère le résultat de chaque règle concernant le point de fonctionnement actuel (étape 3). La commande qui sera appliquée au système est calculée à partir des résultats obtenus soit grâce à la méthode des min-max soit par le calcul du barycentre (étape 4). Toutes ces étapes sont représentées graphiquement sur la figure I.4.4.

5 Méthode neuronale

5.1 Un modèle général

Nous proposons d'utiliser les réseaux neuronaux dans le but de profiter des propriétés des autres modèles (cf fig I.3.6) tout en maintenant une approche généraliste. Certes, l'émergence de l'intelligence artificielle, en particulier la logique floue depuis les années 1970, a permis d'établir des modèles généralistes ; il n'en reste pas moins que les formalismes sont essentiellement fondées sur l'intuition. Rappelons que la logique floue est à l'origine une aide à la décision dont les règles ont été définies au préalable par un expert. Bien que le recours à cet expert ne soit plus aussi indispensable qu'autrefois, grâce aux recherches menées

sur l'extraction automatique de règles, l'optimalité du système flou dépend essentiellement de l'intuition de son concepteur. C'est pourquoi il semble intéressant de comparer la technologie floue à la technologie neuronale, car elles sont toutes deux issues de la même famille, l'intelligence artificielle, mais diffèrent dans l'élaboration des modèles. L'avantage des réseaux de neurones par rapport à la logique floue est le non recours à un expert pour définir les règles. Le concepteur intervient à un niveau hiérarchiquement supérieur dans l'élaboration d'un modèle ou d'un correcteur neuronal. Alors qu'en logique floue le concepteur peut se 'tromper' sur l'exactitude des règles, avec l'approche neuronale, ces règles, ou plus précisément les équations, sont issues directement d'un apprentissage à partir d'échantillons ; celles-ci sont forcément très proches de la réalité.

Dans la littérature, les réseaux neuronaux et la logique floue sont généralement associés, car ils s'empruntent mutuellement des propriétés. L'apprentissage neuronal sert à l'extraction automatique des règles, les ensembles flous permettent aux réseaux de neurones de focaliser l'apprentissage sur certaines parties du domaine d'échantillonnage.
La figure I.5.1 illustre la différence entre la modélisation floue (*fig I.5.1a*) qui, à plusieurs ensembles d'échantillons, associe d'autres ensembles au travers de règles, tandis que l'approche neuronale (*fig I.5.1b*) associe des échantillons à d'autres échantillons par l'intermédiaire d'une seule relation. Par conséquent, si on définit n'importe quelle caractéristique comme une relation entre un ensemble de départ et un ensemble d'arrivée, on redéfinit la nature même d'une fonction, ce qui pourrait justifier l'universalité des réseaux de neurones vis à vis des autres méthodes de modélisation illustrées par la figure I.3.4.

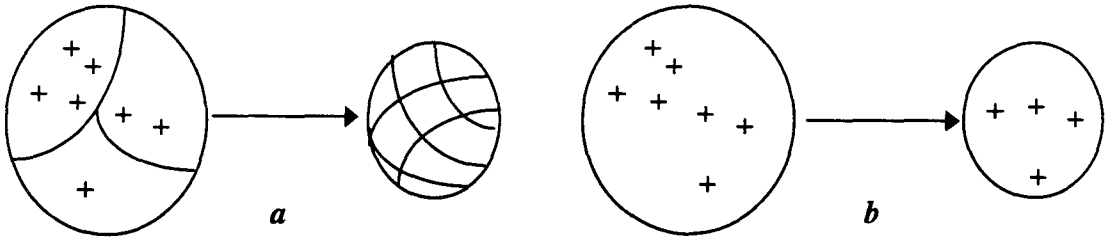


Fig I.5.1 : Distinction entre la logique floue et les réseaux de neurones

Par ailleurs, il n'est pas interdit d'utiliser le réseaux de neurones non plus en temps que modèle global, mais comme élément particulier d'un autre modèle et ceci pour des raisons d'optimalité.

5.2 Simplification de l'élaboration du modèle

L'idée la plus fondamentale de la technique neuronale réside dans la simplification de la méthode permettant l'élaboration d'un modèle à la fois valable pour la simulation et la commande du processus. En fait, l'immense majorité des commandes utilise des modèles analytiques nécessitant une identification paramétrique préalable. Une fois le modèle obtenu, celui-ci sert de base au calcul des correcteurs. Dans la commande neuronale, il est possible d'introduire directement dans le schéma de commande, le modèle obtenu par l'apprentissage des échantillons sans passer par une identification paramétrique. La figure I.5.2 illustre le 'raccourci' pris par la technique que nous allons développer.

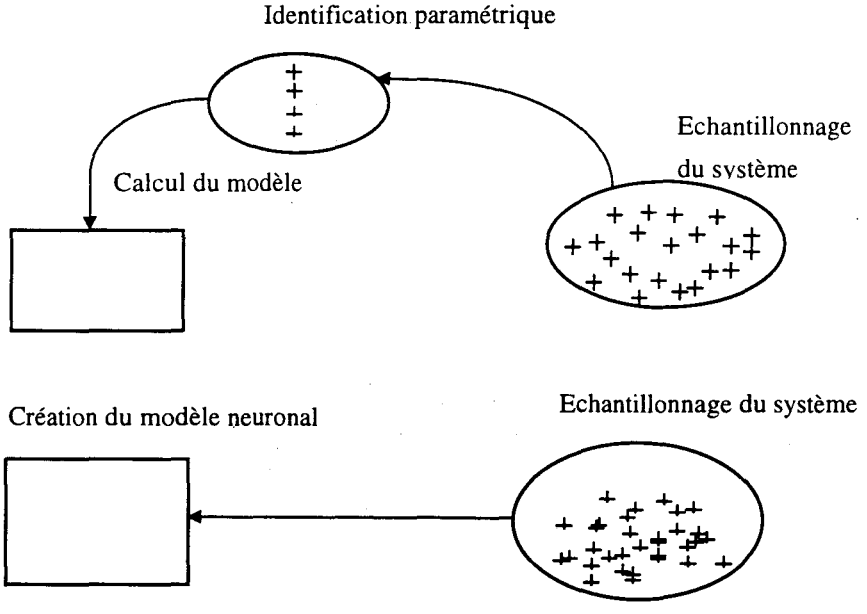


Fig I.5.2 : Schématisation du 'raccourci' pris par la technique neuronale dans l'établissement d'un modèle pour la commande

Un autre intérêt de la modélisation neuronale réside dans la symbolisation graphique des équations d'un modèle. En effet nous verrons qu'il est possible d'utiliser la morphologie d'une caractéristique quelconque pour concevoir la structure mathématique du modèle. Cette technique se concrétise au travers de la construction de l'architecture neuronale. Elle permet en effet de 'placer des fonctions' d'approximation là où cela est nécessaire. Ce principe sera notamment très utile dans l'optimisation du modèle tant du point de vue qualitatif que quantitatif.

5.3 Commande neuronale

Les non linéarités sont généralement négligées dans la commande des processus industriels du fait de la complexité des techniques automatiques proposées pour les prendre en compte. Ces techniques supposent une bonne identification paramétrique au préalable, ce qui implique de connaître préalablement le type de non linéarité rencontrée, ou bien le modèle reste non linéaire mais simplifié et est généralement identifié comme un gain équivalent : c'est la méthode du premier harmonique. Généralement on suppose que les réglages sont suffisamment robustes pour limiter les effets des non linéarités qui sont considérées comme des perturbations extérieures. Les correcteurs sont donc calculés de manière à réguler les variations du système engendrées par ces perturbations. Selon la nature des perturbations, la dynamique de ces correcteurs doit être choisie plus ou moins grande, avec le risque de fatigue que cela entraîne le cas échéant sur l'organe de puissance. Pour pallier ce défaut, il est possible de mettre en œuvre des commandes adaptatives, mais dans ces conditions, on peut considérer que les correcteurs sont en perpétuelle évolution. En particulier, il leur est impossible de profiter de l'expérience acquise au cours des fonctionnements précédents pour anticiper le comportement du processus.

L'idée d'utiliser les réseaux de neurones en commande réside dans la capacité de cet outil à modéliser de façon générale n'importe quel processus, voire chaotique [AUSSEM 95]. Il s'avérera très vite nécessaire d'optimiser l'architecture des modèles neuronaux afin de ne pas alourdir les calculs, ce qui limiterait leurs applications dans les commandes en temps réel. Toutefois, le graphe informationnel causal nous permettra de localiser les besoins spécifiques dans les chaînes d'action en ce qui concerne le recours aux modèles neuronaux.

6 Conclusion

Nous avons décrit au cours de ce chapitre comment définir la nature d'un système, en vue de le modéliser et de le commander. Nous avons décomposé les systèmes en trois catégories : les systèmes linéaires, non linéaires et non stationnaires. Il apparaît que la variable temps, composante principale des systèmes non stationnaires, n'est pas bornée, ce qui rend ce genre de système impossible à représenter par une équation. Néanmoins, si on possède une connaissance globale du système, c'est à dire si on connaît toutes les variables qui entrent en jeu, ainsi que toutes les causes aux effets, alors ces systèmes apparaissent

comme linéaires ou non. Ainsi, tel le mythe de la caverne de Platon, les phénomènes physiques ne sont pas forcément comme ils nous apparaissent. Par conséquent, comme 'le non stationnaire' ne peut se représenter que sur un horizon temporel réduit et comme 'le linéaire' n'est qu'un cas particulier 'du non linéaire', nous nous sommes limités à l'étude des systèmes non linéaires selon les remarques précédentes. Dans l'objectif de représenter ces systèmes, nous avons répertorié les différentes méthodes usuelles, afin de montrer que ces techniques, liées entre elles par des constructions mathématiques, ne satisfont finalement que le choix entre la localité ou la globalité d'une représentation. Nous en avons, par conséquent, déduit qu'un modèle universel pouvait généraliser toutes ces méthodes et satisfaire ce choix. L'attrait de cette quête réside essentiellement dans la systématisation de l'élaboration d'une représentation d'un phénomène à partir d'échantillons prélevés. Nous développerons au second chapitre la construction d'une telle représentation.

Les modèles évoqués précédemment n'ont de raisons d'exister qu'au travers de leurs utilisations soit en simulation, pour prédire les performances ou les états d'un système, soit en vue de le commander. Sur ce dernier point, nous nous sommes interrogés sur la nécessité de connaître un système pour le contrôler et nous avons introduit la notion d'inversion des causalités. Les lois de la physique sont représentées grâce à l'introduction du graphe informationnel causal (G.I.C.) qui, de plus, conduit systématiquement au fondement de la commande. Cet outil nous permettra, notamment au chapitre IV, de mettre en valeur la nécessité d'une commande à base de modèles non linéaires.

Il s'avère néanmoins difficile de synthétiser de façon globale la commande d'un système ne serait ce que pour tenir compte des limitations technologiques. C'est pourquoi, la recherche d'un formalisme unique de commande est une tâche délicate, car chaque système est unique.

Chapitre 2

Les réseaux de neurones artificiels

1 Introduction

Le cerveau naturel est le plus mystérieux de tous les organes biologiques qui composent un être vivant. Son fonctionnement commence à être bien compris par les scientifiques. Néanmoins la gestion des informations qui lui sont communiquées, leur arrangement, leur interprétation, la capacité et la motivation d'utilisation de ces données, toutes ces actions sont autant de mécanismes difficilement définissables et reproductibles de manière artificielle. Cependant les découvertes effectuées sur les neurones, composants de base d'un cerveau, ont donné lieu à des recherches en intelligence artificielle. L'avènement de l'informatique a contribué au développement de cette science qui semble trouver son intérêt essentiellement dans la modélisation de phénomènes non linéaires mal connus.

On se rend compte, au travers des découvertes réalisées sur le cerveau ou à partir de simples observations sur le comportement humain ou animal, combien l'apprentissage est inhérent à l'intelligence. C'est cette phase d'apprentissage qui permet de 'ranger' les informations perçues de telle manière qu'elles soient utilisables pour résoudre des problèmes nouveaux.

Dans un premier temps nous allons définir les bases du neuromimétisme et exposer les différentes topologies existantes. Nous décrirons quelques méthodes classiques d'apprentissage afin d'en dégager les avantages et les inconvénients. Ensuite, nous développerons une méthode basée sur une initialisation qui permettra de réduire considérablement les temps de calculs tant pour l'apprentissage que pour l'émulation. Cette partie comportera une démonstration intuitive et une démonstration mathématique avant d'aboutir sur une étude de cas.

2 Un peu d'histoire

Les origines historiques des réseaux de neurones remontent au début des années 40. L'idée de base est alors de simuler le fonctionnement d'un neurone biologique par des automates linéaires à seuils interconnectés. En 1943, McCulloch et Pitts réussissent à montrer que leurs modèles, composés d'automates à états finis, sont capables de calculer certaines fonctions logiques, ce qui fait naître l'idée d'ordinateur universel [McCulloch 43].

Il faut attendre 1958 pour voir la naissance, grâce à Rosenblatt, d'une méthode analytique rigoureuse d'adaptation des poids au sein d'un modèle multicouches appelé **perceptron** [Rosenblatt 62]. D'autres réseaux similaires appelés 'adalines' furent inventés à la même époque par Widrow [Widrow 62].

Rosenblatt a démontré la convergence d'un algorithme itératif d'adaptation des poids pour les perceptrons mono-couche. Héritant des idées précédemment décrites, le perceptron était en mesure d'apprendre n'importe quelle fonction logique à partir d'expériences grâce à la modification des poids synaptiques. Ces résultats n'ont pas manqué à l'époque de susciter l'enthousiasme et de nourrir de grandes ambitions.

Néanmoins les difficultés du perceptron à résoudre certains problèmes comme la modélisation du OU exclusif furent rapidement mises en évidence [Minsky 69]. Des chercheurs se tournèrent alors vers un nouveau courant de pensée, celui de la représentation symbolique, qui est à la base des systèmes experts. Cependant cette forme d'intelligence artificielle a vite montré ses limites notamment au niveau du mécanisme d'apprentissage.

Quelques chercheurs restés fidèles aux réseaux de neurones ont permis la reformulation plus riche par le biais de travaux centrés autour des mémoires associatives. Les recherches se sont alors plutôt orientées vers des considérations statistiques du traitement de l'information. Il faut attendre Hopfield, en 1982, pour déboucher sur une représentation symbolique usuelle à ce jour des réseaux de neurones [Hopfield 82]. Il apporte également à ce formalisme, le bouclage du réseau sur lui-même, permettant ainsi une représentation de la dynamique des systèmes. Le développement le plus significatif de ces vingt dernières années est incontestablement celui d'un algorithme d'apprentissage formalisé et convergent qui est la rétropropagation. Cette méthode, inventée par Werbos en 1974 [Werbos 74], fût redécouverte et mise à l'honneur par Rumelhart&Cie en 1985 [Rumelhart 85]. Elle permet en outre l'utilisation de réseaux multicouches et par conséquent l'amélioration de l'émulation. Cependant cette méthode n'est pas une panacée en matière d'apprentissage en raison de temps

de calculs excessifs, d'autant plus qu'elle se limite aux réseaux de neurones non bouclés. Néanmoins, elle va contribuer au regain d'intérêt des chercheurs pour les réseaux de neurones artificiels.

D'autres avancées en matière d'architecture ont été réalisées, notamment dans le cadre des récurrences temporelles essentielles dans la modélisation des systèmes chaotiques.
Nous recommandons vivement la lecture des ouvrages suivants sur lesquels se fondent cet historique ainsi que les considérations topologiques et mathématiques détaillées dans les paragraphes suivants [DAVALO 89][HERTZ 91][ABDI 94][JODOUIN 94] [Patterson 96].

3 Le neurone formel

Le neurone formel est un processeur mathématique élémentaire qui applique à son état une fonction ϕ de nature linéaire ou non. On appelle état, ou activité, d'un neurone la somme ou le produit de ses entrées pondérées. Il représente en quelque sorte l'information synaptique globale qui arrive sur ce neurone. La figure ci-dessous en est sa représentation graphique :

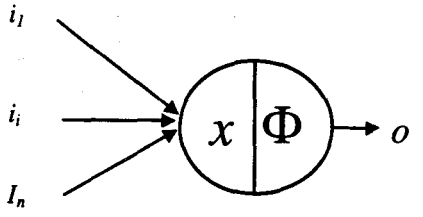


Fig II.3.1 : Le neurone

x est **l'état** ou **l'activité**, o la sortie, i_i les entrées
Φ est la **fonction d'activation** du neurone
i_n entrée n

Le neurone réalise une sommation puis une transformation comme l'indique les équations suivantes :

$$x = \sum_{i=1}^{n} i_i$$
$$o = \Phi(x) \qquad (2.1)$$

Les fonctions d'activation sont généralement croissantes et bornées. A titre d'exemple, la figure II.3.2 représente trois types de fonction d'activation.

- la fonction signe correspondant à un neurone dont la sortie est binaire,
- la caractéristique linéaire est généralement utilisée en entrée et en sortie de réseau afin de distribuer ou de collecter les informations,
- la tangente hyperbolique, appelée sigmoïde, est la fonction la plus souvent employée du fait de sa caractéristique non linéaire et de sa dérivabilité continue sur le domaine. C'est en fait la version continue de la fonction signe.

Il existe d'autres fonctions employées, notamment les fonctions à base radiale dont nous évoquerons les propriétés ultérieurement.

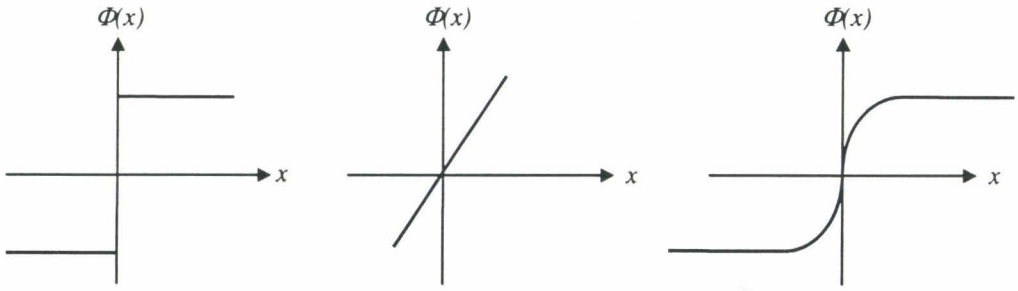


Fig II.3.2 : exemples de fonctions d'activation

Chaque neurone, utilisé dans un réseau, se voit connecté à un biais, c'est à dire à une valeur constante, pour l'une de ses entrées. Ceci permet au neurone de translater son domaine d'activité et d'ajuster son seuil d'efficacité. L'effet de polarisation obtenu est indispensable si l'on veut exploiter correctement l'aptitude du réseau neuronal à reproduire n'importe quelle non-linéarité sur tout l'espace d'un univers donné.

4 Les réseaux non bouclés

4.1 Notion de couches

On appelle couche un ensemble de n neurones possédant chacun $p+1$ entrées (p entrées + un biais).

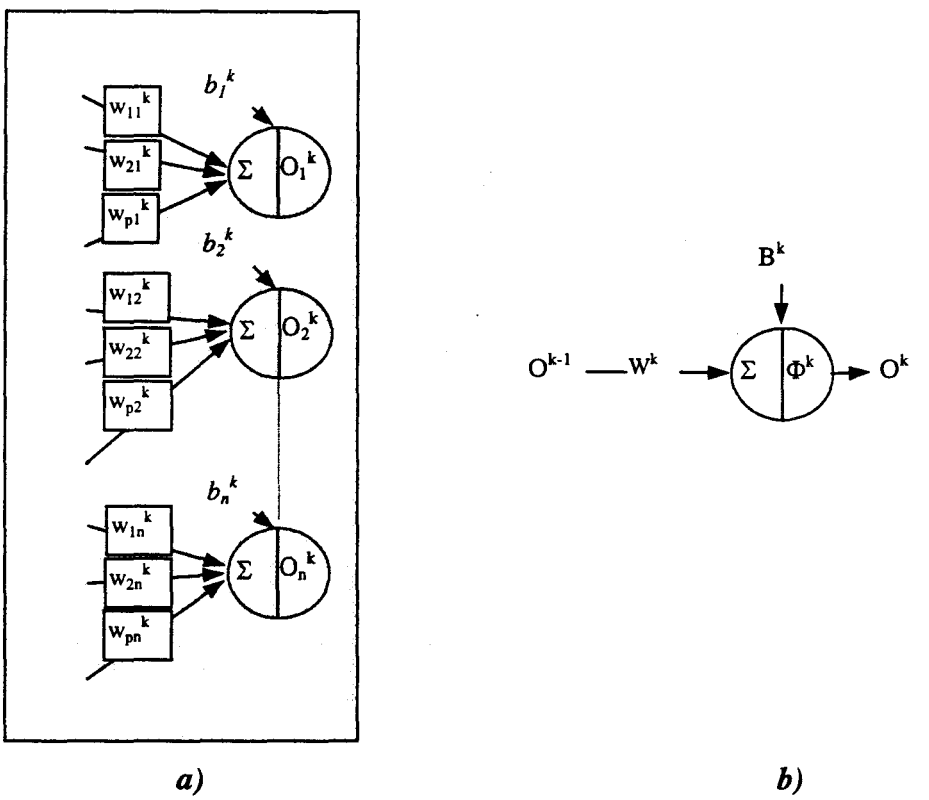
La figure II.4.1 représente une couche k composée de n neurones. Les équations (2.2) ci dessous expriment l'activité et la sortie d'un neurone j de cette couche.

$$x_j^k = \sum_{i=1}^{p} i_i w_{ij}^k + b_j^k$$
$$o_j^k = \Phi_j^k(x) \qquad (2.2)$$

Notations :

w_{ij}^k poids reliant le neurone i de la couche $k-1$ au neurone j de la couche k.

b_j^k biais du neurone j de la couche k.



a) *b)*

Fig II.4.1 : Représentation naturelle a) et matricielle b) d'une couche de neurones

De la même façon on peut exprimer matriciellement la transformation des informations au sein d'une couche (2.3).

$$\begin{aligned} X^k &= O^{k-1}W^k + B^k \\ O^k &= \Phi^k(X^k) \end{aligned} \tag{2.3}$$

4.2 Réseau monocouche

Chaque neurone est capable de reproduire sa propre fonction d'activation. La connexion en couches puis en réseau, va permettre de démultiplier leurs capacités en profitant des propriétés de toutes les fonctions d'activation. Ainsi, par ajustement des poids et des biais, qui sont autant de degré de liberté, le réseau sera capable d'approcher n'importe quelle fonction.

On peut donc considérer le réseau de neurones comme 'une boite noire' capable de synthétiser n'importe quelles sorties fonctions d'entrées. Il paraît alors naturel de créer un réseau dont le flux d'informations transite de l'entrée vers la sortie ; c'est pourquoi on organise une architecture dite à couches. En disposant les neurones en plusieurs couches, nous construisons un réseau de type *feedforward*, dans lequel les connexions n'existent que d'une couche vers la suivante (figure II.4.2.a) ; le système ainsi défini est donc statique. Au sein de celui-ci, les neurones pourront être différents d'une couche à l'autre, ainsi qu'à l'intérieur d'une même couche, comme nous le verrons plus loin. Les entrées et les sorties sont pourvues de neurones à fonctions d'activation linéaires pour respectivement distribuer et récolter les informations. Lorsque l'état d'un neurone est formé par la somme des entrées, on le symbolise par le signe somme. De même, pour les fonctions d'activation, on représente la fonction soit par une lettre soit par un dessin décrivant la fonction.

Sur la représentation naturelle (figure II.4.2), les poids et les biais n'ont pas été représentés par souci de clarté. Les neurones de la couche d'entrée et de sortie ne servent respectivement qu'à distribuer et à recueillir les informations. C'est pour cette raison que les fonctions d'activation de ces neurones sont toujours linéaires. De plus la couche d'entrée, qui n'a qu'un rôle de distribution, possède des poids unitaire et aucun décalage, les biais sont donc nuls. Par ailleurs, on appelle également **couche cachée** toute couche n'appartenant ni à la couche d'entrée ni à la couche de sortie.

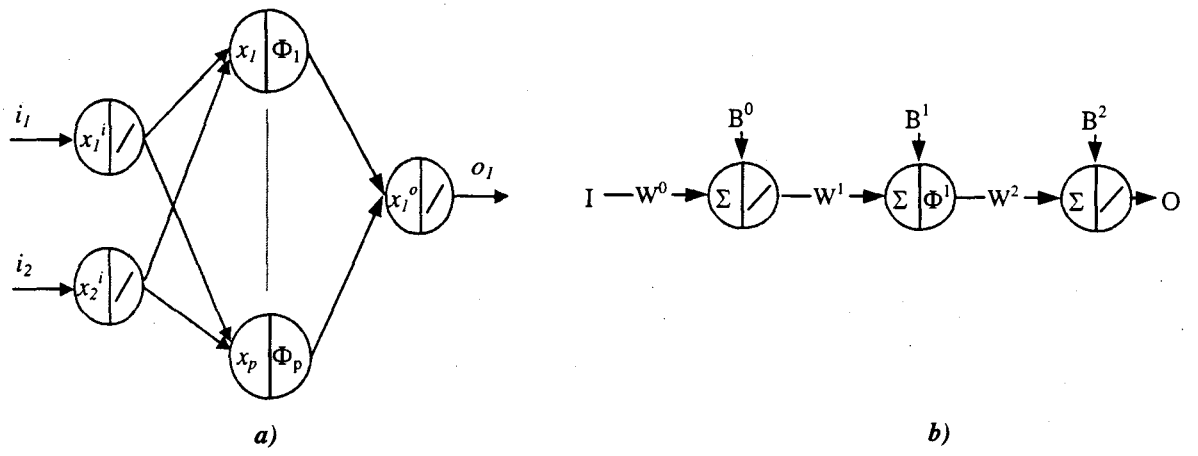


Fig II.4.2 : Représentation naturelle a) et matricielle b) d'un réseau mono couche

Etant donné le rôle des couches d'entrée et de sortie, la matrice W^0 est la matrice identité, les matrices B^0 et B^2 sont nulles. Ainsi, dans l'exemple de la figure II.4.2, la sortie du réseau s'exprime matriciellement de la façon suivante (2.4) :

$$O = \Phi(IW^1 + B^1)W^2 \qquad (2.4)$$

4.3 Réseau multicouche

La qualité d'émulation augmente généralement avec le nombre de neurones. Dans le cas des réseaux monocouches, la recherche de l'amélioration conduit à augmenter l'unique couche cachée. Afin d'éviter une hypertrophie de cette couche, on utilise généralement des réseaux possédant plusieurs couches cachées (figure II.4.3). Il a en effet été montré que la qualité d'approche se renforce avec l'adjonction de couches cachées supplémentaires [HORNIK 88] [BAUM 89] [CHESTER 90].

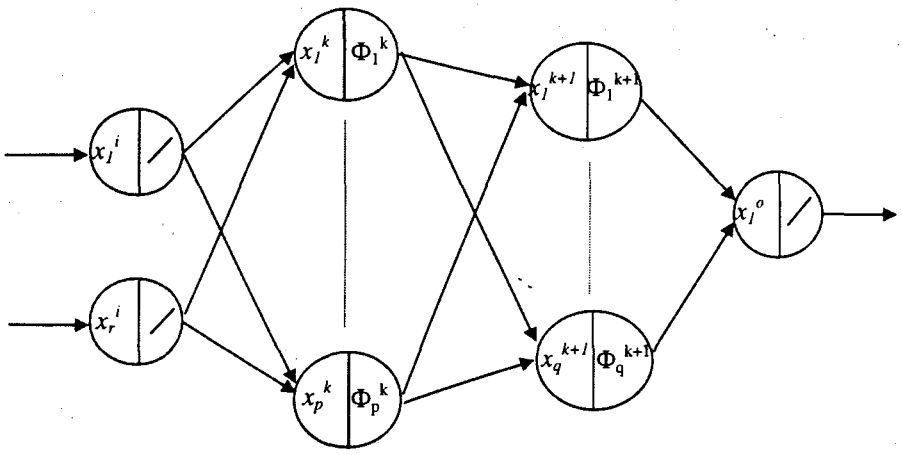


Fig II.4.3 : Exemple d'un réseau à deux couches cachées

Nous définissons pour chaque neurone *i* de la couche k l'état x_i^k et la sortie o_i^k. Les couches *k* et *k+1* comportent respectivement *p* et *q* neurones, nous avons les équations (2.5) :

$$\begin{aligned} x_i^{k+1} &= \sum_{j=1}^{p} o_j^k . w_{ji}^{k+1} + b_i^{k+1} \\ o_i^{k+1} &= \Phi_i^{k+1}(x_i^{k+1}) \end{aligned} \tag{2.5}$$

où w_{ji}^{k+1} est le poids allant du neurone *j* couche *k* au neurone *i* couche *k+1*, et b_i^{k+1} est le poids de l'entrée de polarisation (biais).

Il s'en déduit l'expression matricielle (2.6) du comportement de la couche *k+1*, en notant x^{k+1} et o^{k+1} les vecteurs des états et des sorties des neurones de la couche *k+1* :

$$\begin{aligned} \mathrm{X}^{\mathrm{k+1}} &= \mathrm{O}^{\mathrm{k}} . \mathrm{W}^{\mathrm{k+1}} + \mathrm{B}^{\mathrm{k+1}} \\ \mathrm{O}^{\mathrm{k+1}} &= \Phi^{\mathrm{k+1}}(\mathrm{X}^{\mathrm{k+1}}) \end{aligned} \tag{2.6}$$

Avec les matrices suivantes (2.7) :

$$W^{k+1} = \begin{bmatrix} w_{1,1}^{k+1} & w_{1,2}^{k+1} & w_{1,q}^{k+1} \\ & & \\ w_{p,1}^{k+1} & w_{p,2}^{k+1} & w_{p,q}^{k+1} \end{bmatrix}, B^{k+1} = \begin{bmatrix} b_1^{k+1} \\ \\ b_q^{k+1} \end{bmatrix}^T, \Phi^{k+1}(X^{k+1}) = \begin{bmatrix} \Phi_1^{k+1}(x_1^{k+1}) \\ \Phi_2^{k+1}(x_2^{k+1}) \\ \\ \Phi_q^{k+1}(x_q^{k+1}) \end{bmatrix} \tag{2.7}$$

A partir des expressions précédentes, il est facile de décrire le cas général d'un réseau à N couches sous la forme matricielle suivante (2.8) :

$$O = \Phi^N((((.....(X^1W^1+B^1).....)W^{N-2} + B^{N-2})W^{N-1} + B^{N-1})W^N + B^N) \qquad (2.8)$$

5 Les réseaux bouclés

Jusqu'à présent, nous nous sommes restreints à l'architecture des réseaux non bouclés. Nous abordons désormais une topologie de réseaux caractérisés par des connexions récurrentes. On peut distinguer deux familles de réseaux bouclés, les réseaux à rebouclage externe et les réseaux à rebouclage interne ou complètement bouclés.

5.1 Le réseau bouclé extérieurement

L'introduction d'un bouclage externe permet au réseau de neurones de prendre en compte en plus des entrées courantes, un certain nombre d'états passés. Cette architecture est notamment très utile pour la conception des modèles de type NARMAX [CONNOR 94]. Ce type d'architecture est également utilisé pour la modélisation de séquences temporelles complexes, qu'on utilise par exemple pour la reconnaissance de phonèmes.

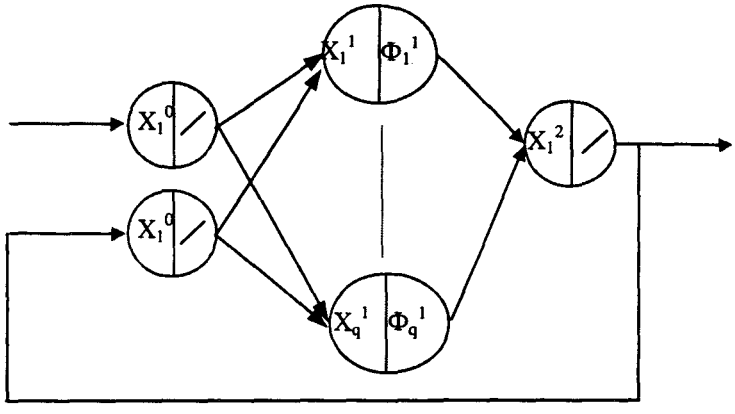


Fig II.5.1 : Exemple d'architecture statique récurrente de type Hopfield

Ce type de réseau permet, entre autres, la réalisation de modèles de processus non linéaires du type :

$$\begin{aligned} X_{k+1} &= f(X_k, U_k) \\ Y_{k+1} &= CX_{k+1} \end{aligned} \qquad (2.9)$$

où X, Y sont des vecteurs d'états et f une fonction non linéaire.

Les réseaux de type Hopfield sont également organisés en couche, ce qui permet d'accroître le nombre de degrés de liberté, c'est à dire la taille mémoire, en fonction à la fois du nombre de neurones et de couches. Comme ce type de réseau est régi par les équations décrites précédemment, il est possible de trouver une fonction de Lyapounov qui atteste de la stabilité et de la convergence vers des attracteurs.

5.2 Le réseau complètement bouclé

Cette famille de réseaux est constituée de neurones connectés entre eux de toutes les manières possibles, y compris par des rebouclages sur un même neurone. Dans cette architecture, les neurones ne sont plus organisés en couches (figure II.5.2). On définit néanmoins quels sont les neurones qui servent d'entrées et de sorties. Comme l'architecture n'est plus organisée en couche, le sens des flux d'informations se repère, non plus seulement dans l'espace, mais également dans un repère temporel. Il est à noter que l'introduction de bouclage interne rend beaucoup plus complexe la phase d'apprentissage. De plus, dans le cadre de notre application aux systèmes électrotechniques, il s'avère que cette architecture complexe n'est pas nécessaire, compte tenu de la connaissance que nous avons des systèmes. C'est pourquoi, nous n'utiliserons pas ici la topologie complètement bouclée, cette dernière n'étant citée dans ce paragraphe qu'au titre de l'état de l'art.

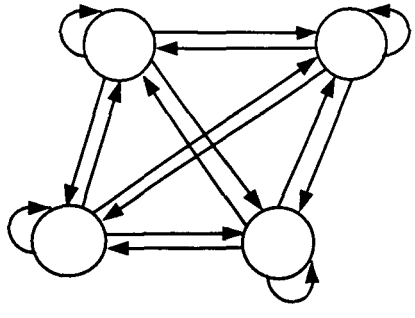


Fig II.5.2 : Exemple d'architecture statique complètement bouclée

6 Apprentissage

L'intérêt d'un réseau de neurones réside dans sa capacité à composer avec ses fonctions d'activation pour approcher n'importe quelle fonction, linéaire ou non, à partir d'échantillons expérimentaux. Les poids et les biais sont autant de paramètres à régler, dont une seule combinaison offrira une approche optimale au sens des moindres carrés. En effet, on utilise généralement la somme des écarts quadratiques comme critère d'ajustement, d'une part car ce dernier ne peut pas s'annuler par simple retranchement des erreurs et, d'autre part, la somme des écarts quadratiques possède des propriétés statistiques liées à la définition même de la variance. Afin d'aboutir à cette synthèse, il est nécessaire d'effectuer une phase dite 'd'apprentissage', qui consiste à trouver les poids et les biais optimaux pour une architecture de réseau donnée. Comme les fonctions d'activation et les fonctions à approcher sont généralement de nature non linéaire, les méthodes employées s'inspirent des méthodes d'optimisation non linéaires. Les plus connues sont **le recuit simulé, les algorithmes génétiques, la méthode des directions conjuguées et la descente de gradient**. L'apprentissage est réalisé suivant le schéma de la figure II.6.1 de sorte que l'erreur ε entre la fonction réelle et la fonction estimée soit réduite au maximum; on parle dans ce cas d'apprentissage supervisé, l'algorithme d'optimisation étant alors le superviseur.

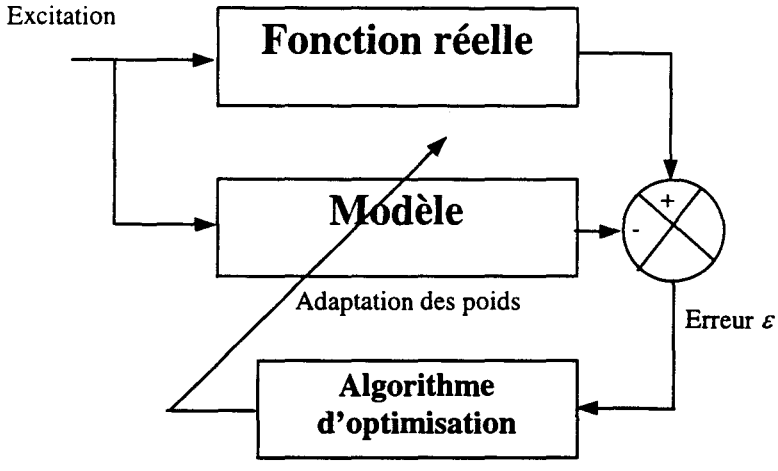


Fig II.6.1 : Schéma d'apprentissage supervisé

Ainsi, on décompose l'ensemble global d'échantillons en deux sous-ensembles : le premier, celui des échantillons d'apprentissage, sert à l'ajustement des poids. Le second,

appelé ensemble des échantillons de validité, permet de tester les performances de l'apprentissage obtenues sur le même domaine de modélisation.

Nous allons donner à titre indicatif le principe des méthodes du recuit simulé et des algorithmes génétiques pour montrer les avantages et les inconvénients de ces deux méthodes. L'apprentissage par descente de gradient sera détaillé ultérieurement.

6.1 Le recuit simulé

Le principe de cette méthode consiste à balayer toutes les possibilités paramétriques possibles et à ne retenir que la combinaison qui minimise au mieux l'erreur d'apprentissage [BONNEMOY 91]. Par exemple, considérons un réseau constitué uniquement de deux poids qu'il faut ajuster pour approcher une fonction quelconque et dont l'erreur ε est représentée (figure II.6.2) en fonction de la valeur des poids.

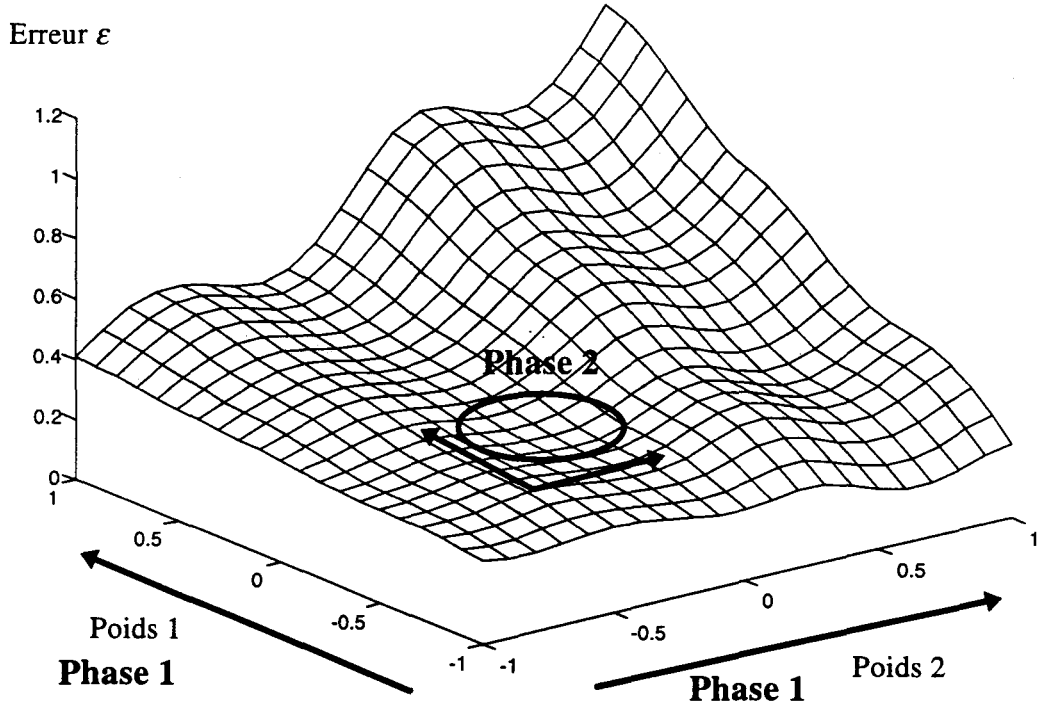


Fig II.6.2 : Représentation de la fonction erreur d'apprentissage en fonction de la valeur des différents poids

L'ensemble des couples est examiné (phase 1 sur la figure), afin de trouver un doublet de poids minimisant l'erreur d'apprentissage. On examine alors les couples de poids présents autour de chaque minimum en affinant la distance entre chaque couple (phase 2) puis, autour d'un de ces minimum, les opérations sont réitérées en affinant de plus en plus cette distance jusqu'à ce que l'erreur soit acceptable. Comme on peut s'en douter, si le recuit simulé à l'avantage d'assurer la convergence vers le minimum global (à condition de prendre un pas suffisamment petit), il n'en reste pas moins 'gourmand' en temps de calcul.

6.2 Les algorithmes génétiques

L'apprentissage par algorithmes génétiques est basé sur l'observation de l'évolution de la nature au cours du temps. Ainsi l'évolution naturelle est décrite comme une mutation aléatoire ou accidentelle des gênes d'un individu ; la nature ne conservera cette mutation que si elle apporte une amélioration à l'adaptation de cet individu dans son environnement. Pour transposer ce principe à l'apprentissage des réseaux neuronaux, il faut dans un premier temps coder chaque poids sous forme binaire. Ainsi cette représentation va constituer un 'chromosome' du réseau. Ensuite on opère de façon aléatoire des mutations (figure II.6.3.a) ; il suffit de modifier par exemple un gêne d'un chromosome en remplaçant un 1 en 0 ou vice versa. On peut également croiser des morceaux entiers de chromosome ce qu'on appelle le '*crossover*' (figure II.6.3.b).

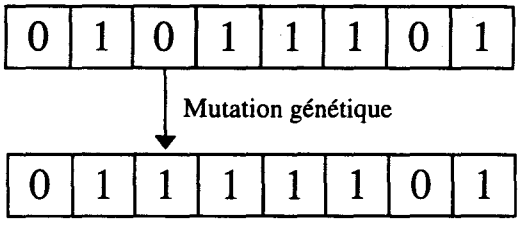


Fig II.6.3.a : Représentation d'une mutation génétique d'un chromosome

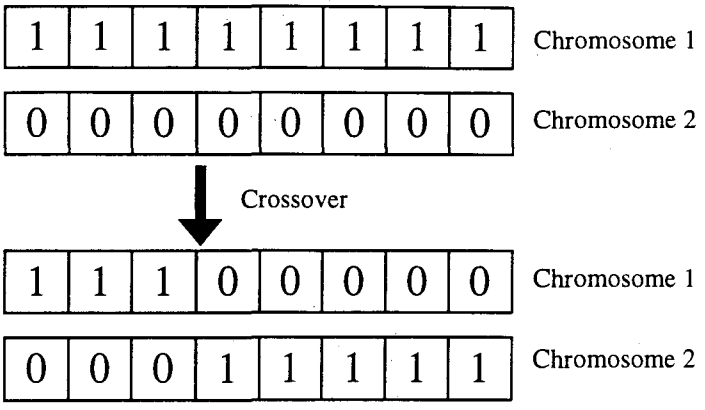


Fig II.6.3.b : Représentation d'un crossover

Les changements des valeurs des poids se font essentiellement par ces deux méthodes et de façon aléatoire [PATTERSON 96].

7 La rétropropagation du gradient

7.1 Principe

Considérons un réseau comprenant N entrées et M sorties. Au cours de cette procédure d'apprentissage, un vecteur $I=\{I_0,I_1,....,I_N\}$ est présenté à l'entrée du réseau ; le vecteur de sortie $O=\{O_1,O_2,.....,O_M\}$ qui en résulte est alors différent de celui désiré soit $T=\{T_1,T_2,.....,T_M\}$. Chaque composante I_i, O_i ou T_i est un vecteur composé de n échantillons. Un vecteur erreur E entre les vecteurs T et O est introduit et les poids sont ajustés de façon à minimiser l'erreur globale au sens des moindres carrés (2.10)

$$E=\frac{1}{2}\sum_{i=1}^{M}(O_i-T_i)^2, \qquad (2.10)$$

L'algorithme de rétropropagation du gradient (2.11) définit chaque nouveau poids en fonction de l'ancien [RUMELHART 86] :

$$w_{ij}^k(new)=w_{ij}^k(old)-\mu\frac{\partial E}{\partial w_{ij}^k(old)}, \qquad (2.11)$$

μ étant le coefficient d'apprentissage (plus μ est grand, plus la modification des poids se fera rapidement). L'ajustement des coefficients se fait en commençant par les poids de la couche de sortie pour finir par les poids de la couche d'entrée, d'où le nom de rétropropagation. Une

fois que tous les poids et les biais ont été modifiés, on représente à nouveau le vecteur I en entrée et on recalcule le vecteur en sortie du réseau. Si l'erreur E est toujours jugée trop grande, on réitère la procédure d'ajustement précédemment décrite, jusqu'à ce que E respecte le critère de précision fixé (figure II.7.1).

Ce processus d'apprentissage peut être utilisé 'hors ligne' ou 'en ligne'. On parle d'apprentissage hors ligne lorsque le réseau n'est pas sollicité en tant que modèle ou commande pendant la phase d 'apprentissage. Lorsque ce dernier doit, en même temps qu'il apprend, établir la valeur d'une commande ou d'une prédiction, on parle alors d'apprentissage en ligne. Il est à noter que dans le cas d'un apprentissage hors ligne, la rétropropagation du gradient traite des vecteurs contenant des informations sur l'ensemble du domaine d'apprentissage, c'est à dire sur des milliers d'échantillons. Dans le cas d'un apprentissage en ligne, on ne peut se permettre de traiter des milliers de valeurs : le calcul étant trop lourd, il peut être nuisible au bon déroulement d'une commande en temps réel. Tout ceci reste bien sûr relatif aux moyens de calcul mis en oeuvre et aux constantes de temps des systèmes contrôlés.

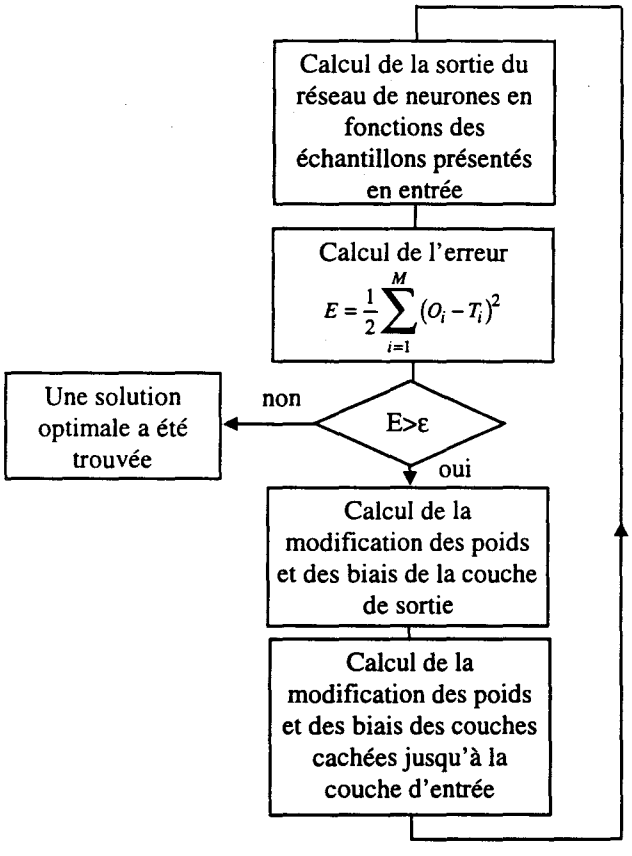


Fig II.7.1 : Déroulement de la rétropropagation du gradient de l'erreur

7.2 Calcul du gradient pour un réseau non bouclé

7.2.1 Rétropropagation hors ligne

Considérons la fraction de réseau représentée figure II.7.2. Le gradient de l'erreur par rapport au poids w_{ji} sur la couche $k+1$ de sortie, est tout d'abord calculé au moyen des dérivées partielles (2.12) :

$$\frac{\partial E}{\partial w_{ji}^{k+1}} = \frac{\partial E}{\partial x_i^{k+1}} \frac{\partial x_i^{k+1}}{\partial w_{ji}^{k+1}} \tag{2.12}$$

et en faisant intervenir l'état x de chaque neurone

$$\frac{\partial x_i^{k+1}}{\partial w_{ji}^{k+1}} = \frac{\partial \sum_{p=1}^{P} w_{ji}^{k+1} o_p^k}{\partial w_{ji}^{k+1}} = o_j^k \tag{2.13}$$

on obtient finalement :

$$\frac{\partial E}{\partial w_{ji}^{k+1}} = \frac{\partial E}{\partial x_i^{k+1}} o_j^k \tag{2.14}$$

Pour calculer le gradient de l'erreur par rapport à l'état d'un neurone, il vient :

$$\begin{aligned} \frac{\partial E}{\partial x_i^{k+1}} &= \frac{1}{2} \frac{\partial \sum_{m=1}^{M} \left(t_m^{k+1} - o_m^{k+1}\right)^2}{\partial x_i^{k+1}} \\ &= \frac{1}{2} \frac{\partial \left(\left(t_1^{k+1} - o_1^{k+1}\right)^2 + \cdots + \left(t_i^{k+1} - o_i^{k+1}\right)^2 + \cdots + \left(t_n^{k+1} - o_n^{k+1}\right)^2 \right)}{\partial x_i^{k+1}} \\ &= -\left(t_i^{k+1} - o_i^{k+1}\right) \frac{\partial o_i^{k+1}}{\partial x_i^{k+1}} \end{aligned} \tag{2.15}$$

Le terme $\frac{\partial o_i^{k+1}}{\partial x_i^{k+1}}$ représente la dérivée de la fonction d'activation du neurone i de la couche $k+1$. Pour cette raison, les fonctions d'activation doivent être choisies continûment dérivables sur leur domaine de définition.

L'expression de la loi modifiant **les poids de la couche de sortie** est alors :

$$\boxed{w_{ji}^{k+1} = w_{ji}^{k+1} + \mu\left(t_i^{k+1} - o_i^{k+1}\right)\frac{\partial o_i^{k+1}}{\partial x_i^{k+1}} o_j^k} \tag{2.16}$$

Les biais sont également réajustés par une loi similaire :

$$\boxed{b_{ji}^{k+1} = b_{ji}^{k+1} + \mu\left(t_i^{k+1} - o_i^{k+1}\right)\frac{\partial o_i^{k+1}}{\partial x_i^{k+1}}} \tag{2.17}$$

En ce qui concerne les couches cachées, le gradient de l'erreur se détermine à partir des dérivées partielles déjà calculées précédemment. Soit pour une couche cachée l :

$$\frac{\partial E}{\partial x_i^l} = \sum_{m=1}^{M}\left(\frac{\partial E}{\partial x_m^{l+1}}\frac{\partial x_m^{l+1}}{\partial x_i^l}\right) = \sum_{m=1}^{M}\left(\frac{\partial E}{\partial x_m^{l+1}}\frac{\partial x_m^{l+1}}{\partial o_i^l}\frac{\partial o_i^l}{\partial x_i^l}\right) \tag{2.18}$$

en développant :

$$\begin{aligned}\frac{\partial E}{\partial x_i^l} &= \sum_{m=1}^{M}\left(\frac{\partial E}{\partial x_m^{l+1}}\frac{\partial\left(w_{1m}^{l+1}o_1^l + \cdots + w_{im}^{l+1}o_i^l + \cdots + w_{pm}^{l+1}o_p^l\right)}{\partial o_i^l}\frac{\partial o_i^l}{\partial x_i^l}\right) \\ &= \sum_{m=1}^{M}\left(\frac{\partial E}{\partial x_m^{l+1}} w_{im}^{l+1}\frac{\partial o_i^l}{\partial x_i^l}\right)\end{aligned} \tag{2.19}$$

Par conséquent, la modification des poids au sein des couches cachées s'exprime de la façon suivante :

$$\boxed{w_{ji}^l = w_{ji}^l + \mu\frac{\partial o_i^l}{\partial x_i^l}\sum_{m=1}^{M}\left(\frac{\partial E}{\partial x_m^{l+1}} w_{im}^{l+1}\right) o_j^{l-1}} \tag{2.20}$$

Les biais sont également réajustés par une loi similaire

$$b_{ji}^{l} = b_{ji}^{l} + \mu \frac{\partial o_i^l}{\partial x_i^l} \sum_{m=1}^{M} \left(\frac{\partial E}{\partial x_m^{l+1}} w_{im}^{l+1} \right) \tag{2.21}$$

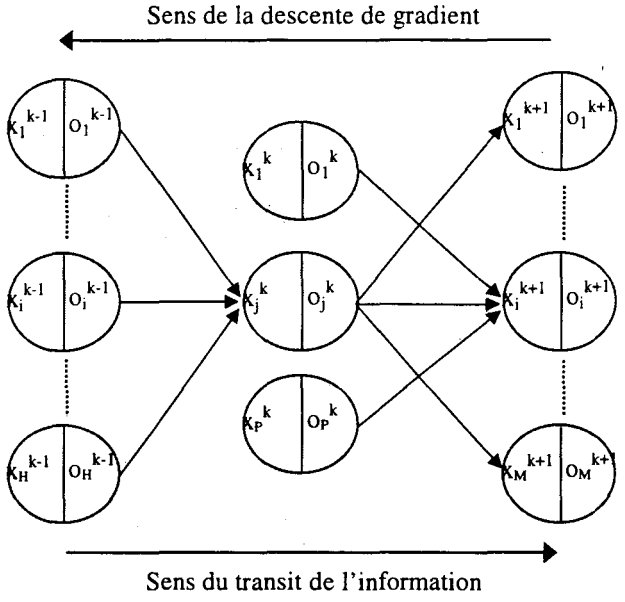


Fig II.7.2 : Couches illustrant la descente de gradient

7.2.2 Rétropropagation en ligne

La rétropropagation en ligne utilise les expressions et l'algorithme vus précédemment. La seule différence est que les vecteurs sur lesquels s'effectuent les opérations, sont de dimensions réduites, voire unitaires. Un exemple de cette méthode sera présenté au quatrième chapitre dans le paragraphe concernant l'adaptation en ligne.

7.3 Calcul du gradient pour un réseau bouclé

Dans ce paragraphe nous distinguerons les réseaux extérieurement bouclés des réseaux complètement bouclés.

7.3.1 Réseaux extérieurement bouclés

Le réseau de neurones extérieurement bouclé est un réseau mono ou multi couches dont la ou les sorties sont rebouclées sur la couche d'entrée. Ce type de réseau est intéressant pour des applications où le temps joue un rôle important, comme la prédiction ou le contrôle

de processus dynamiques. Ainsi les neurones peuvent tenir compte de plusieurs entrées et états précédents d'un modèle. L'exemple d'un réseau comportant p entrées retardées et q sorties retardées réinjectées en entrée du réseau est donné figure (II.7.3).

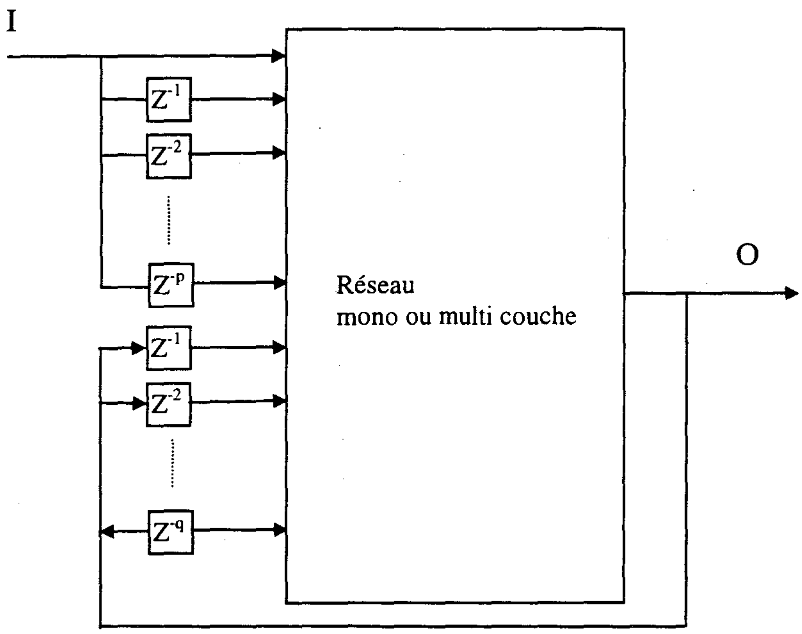


Fig II.7.3 Réseau neuronal extérieurement bouclé

7.3.1.1 Apprentissage hors ligne

L'apprentissage hors ligne utilisant la méthode de 'descente de gradient' est identique à celle employée pour les réseaux non bouclés. Dans le cas de la structure présentée figure II.7.3, si on dispose de n échantillons, le vecteur présenté en entrée doit néanmoins comporter le nombre de composantes correspondant aux récurrences du réseau (2.22) :

$$\begin{pmatrix} i(1) & i(0) & i(-1) & \cdots & \cdots & \cdots & i(1-p) & o(1) & o(0) & o(-1) & \cdots & \cdots & \cdots & \cdots & o(1-q) \\ i(2) & i(1) & i(0) & \ddots & \ddots & \ddots & \vdots & o(2) & o(1) & o(0) & \ddots & \ddots & \ddots & \ddots & \vdots \\ i(3) & i(2) & i(1) & \ddots & \ddots & \ddots & \vdots & o(3) & o(2) & o(1) & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ i(n) & i(n-1) & i(n-2) & \cdots & \cdots & \cdots & i(n-p) & o(n) & o(n-1) & o(n-2) & \cdots & \cdots & \cdots & \cdots & o(n-q) \end{pmatrix} \quad (2.22)$$

7.3.1.2 Apprentissage en ligne

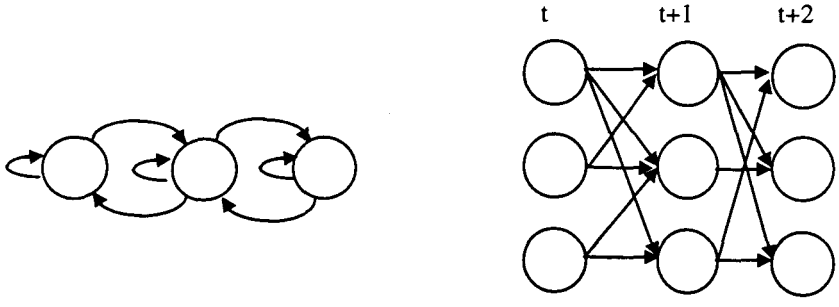
En raison du bouclage de la sortie vers l'entrée du réseau, les équations de la rétropropagation en ligne de ce type d'architecture devront en partie utiliser les équations de la 'Real Time Recurent Learning' (RTRL) que nous verrons un peu plus loin.

7.3.2 Réseaux complètement bouclés

Deux techniques d'apprentissage sont disponibles pour entraîner ce type d'architecture : il s'agit de la rétropropagation dans le temps, 'Back-Propagation Through Time' ou BPTT, pour la première méthode, et l'apprentissage en temps réel, 'Real Time Recurent Learning' ou RTRL pour la seconde.

7.3.2.1 La 'Back-Propagation Through Time'

La BPTT consiste à 'déplier' l'architecture autant de fois qu'il y a de récurrence pour obtenir un réseau multi couches non bouclé. Chaque récurrence constitue ainsi une couche, les couches de sortie et d'entrée sont respectivement la plus récente et la plus ancienne (figure II.7.4).



a) *b)*

Fig II.7.4 a) Réseau récurrent b) Réseau récurent 'déplié' dans le temps

Cet artifice permet donc de convertir n'importe quelle architecture récurrente en non récurrente, ce qui permet, pour l'apprentissage, de se ramener à une rétropropagation classique du gradient. Le principal défaut de la BPTT est que le 'dépliage' ne peut s'effectuer que sur un

intervalle de temps fini puisque la récurrence temporelle est remplacée par une représentation spatiale.

7.3.2.2 La 'Real Time Recurent Learning' méthode

La principale difficulté des réseaux bouclés réside dans le fait que l'erreur d'apprentissage à une itération donnée est non seulement due à la mauvaise estimation des poids mais également, à l'erreur qui en découle sur les sorties précédentes [PINEDA 86 et 89]. De plus, nous avons vu au paragraphe 5, que l'architecture neuronale complètement bouclée n'est pas organisée en couches. Cette notion de couche se substitue alors à la notion de temps, c'est à dire que l'information qui provenait d'une couche inférieure est maintenant interprétée comme une information passée (c'est à dire à l'instant *t-1*) provenant d'un neurone quelconque. Par commodité, et du fait de nos applications, on ne s'attardera que sur les architectures récurrentes organisées en couches. Par conséquent la rétropropagation récurrente en temps réel découle des équations classiques :

$$W_{ji}(t+1) = W_{ji}(t) - \mu \frac{\partial E(t)}{\partial W_{ji}(t)} \tag{2.23}$$

$$E_i = \begin{cases} T_i - O_i & \text{si i est un neurone de sortie} \\ 0 & \text{sinon} \end{cases}, \tag{2.24}$$

$$\frac{\partial E(t)}{\partial W_{ji}(t)} = -\sum_{k \in S} \left(T_k(t) - O_k(t)\right) \frac{\partial O_k(t)}{\partial W_{ji}(t)} \tag{2.25}$$

$$\frac{\partial O_k(t+1)}{\partial W_{ji}(t)} = \frac{\partial O_k(t+1)}{\partial X_k(t)} \frac{\partial X_k(t)}{\partial W_{ji}(t)} \tag{2.26}$$

$$\frac{\partial X_k(t)}{\partial W_{ji}(t)} = \frac{\partial \sum_{l \in S} W_{lk} O_l(t)}{\partial W_{ji}(t)} = \delta_{ik} O_j(t) + \sum_{l \in S} W_{lk} \frac{\partial O_l(t)}{\partial W_{ji}(t)} \tag{2.27}$$

Par conséquent, les mises à jour respectivement des poids et des biais s'expriment de la façon suivante, sachant que l'état initial du réseau ne dépend pas des poids :

$$\begin{cases} \dfrac{\partial O_k(t_0)}{\partial W_{ji}} = 0 \qquad \forall k \\ \dfrac{\partial O_k(t+1)}{\partial W_{ji}} = \dfrac{\partial O_k(t)}{\partial X_k}\left(\delta_{ik} O_j(t) + \sum_{l \in S} W_{lk} \dfrac{\partial O_l(t)}{\partial W_{ji}(t)}\right) \end{cases} \tag{2.28}$$

$$W_{ji}(t+1) = W_{ji}(t) + \mu \sum_{k \in S}\left(\left(T_k(t) - O_k(t)\right)\frac{\partial O_k(t+1)}{\partial W_{ji}}\right) \tag{2.29}$$

8 L'apprentissage par initialisation

8.1 Initialisation

La méthode d'initialisation consiste en fait à trouver tous les poids et biais par simple résolution des équations qui représentent le réseau neuronal, en une seule itération alors que les méthodes d'apprentissage classiques en nécessitent plusieurs. Pour cela, on utilise toujours un réseau monocouche afin de simplifier les équations. Il y a en réalité deux sortes de coefficients à trouver : les poids et biais d'entrée, les poids de sortie.

8.1.1 Poids et biais d'entrée

Soit une variable d'entrée x_i normalisée dont le domaine de variation est représenté figure II.8.1. On effectue un découpage en n sous-domaines ; les poids et les biais d'entrée sont fixés par rapport aux moyennes et variances des variables d'entrée tels que :

$$w_{ij} = \frac{1}{\sqrt{\sigma x_{ij}}} \text{ et } b_{ij} = \frac{-\mu x_{ij}}{\sqrt{\sigma x_{ij}}} \tag{2.30}$$

où μx_{ij} est la moyenne des échantillons de la variable d'entrée x_i sur la $j^{ième}$ portion du domaine de x_i, de même σx_{ij} est la variance des échantillons de la variable d'entrée x_i sur la $j^{ième}$ portion du domaine de x_i. En effet si une variable d'entrée x_i est connectée sur n neurones de la couche cachée, alors l'idée est de faire travailler chacun de ces n neurones sur une portion bien précise du domaine de cette variable.

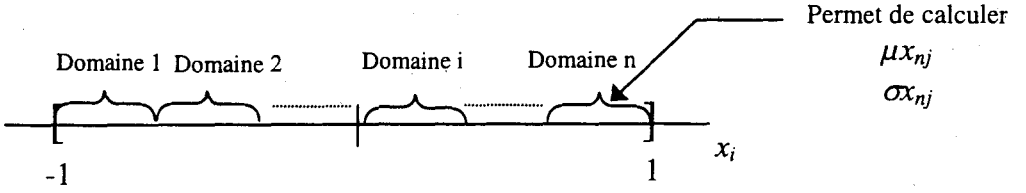


Fig II.8.1 : Découpage du domaine en n sous domaines équivalents

8.1.2 Poids de sortie

Comme le réseau comporte une seule couche cachée, il est possible de déterminer les poids de sortie en résolvant toutes les équations du système. Considérons un réseau mono couche illustré par sa représentation matricielle figure II.8.2.

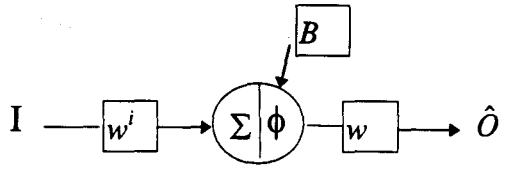


Fig II.8.2 : Représentation matricielle d'un réseau mono couche

Etant donné que les poids et les biais d'entrée, respectivement W^i et B sont préalablement fixés, on peut alors calculer la sortie Oh de la couche cachée telle que $Oh = \Phi\left(IW^i + B\right)$. On cherche alors la matrice des poids W de manière à ce que la sortie du réseau $\hat{O}$ approche au mieux le vecteur réel désiré O. Pour cela, il suffit de résoudre :

$$\hat{O} = O = Oh * W \tag{2.31}$$

La matrice W se trouve par pseudo inversion (*pinv*) du vecteur de sortie de la couche cachée :

$$\boxed{W = pinv(Oh) * O} \tag{2.32}$$

8.2 Pseudo-inversion de matrice :

Nous allons montrer que la méthode de pseudo inversion de matrice de Moore Penrose permet d'obtenir le meilleur estimateur au sens des moindres carrés. Considérons un réseau ne possédant qu'une seule couche cachée, et dont les poids et les biais d'entrée ont été préalablement fixés. La représentation matricielle de celui ci est illustré par la figure II.8.2.

Nous savons que le meilleur estimateur au sens des moindres carrés doit minimiser la somme des erreurs quadratiques, ce qui revient à minimiser la trace :

$$\min E\left\{\left\|O - Oh * W\right\|^2\right\} = \min tr\left\{P_{OO} - P_{OOh}W - W^T P^T{}_{OOh} + W^T P_{OhOh} W\right\} \tag{2.33}$$

avec

$$\begin{aligned} P_{OOh} &= E\left\{(O - m_O)(Oh - m_{Oh})^T\right\} \\ P_{OO} &= E\left\{(O - m_O)(O - m_O)^T\right\} \\ m_{Oh} &= E\{Oh\} \qquad m_O = E\{O\} \end{aligned} \tag{2.34}$$

Si P_{OhOh} est régulière on peut utiliser l'égalité suivante :

$$\begin{aligned} &P_{OO} - P_{OOh}W - W^T P^T{}_{OOh} + W^T P_{OhOh} W = \\ &P_{OO} - P_{OOh} P^{-1}{}_{OhOh} P^T{}_{OOh} + \left[W^T - P_{OOh} P^{-1}{}_{OhOh}\right] P_{OhOh} \left[W^T - P_{OOh} P^{-1}{}_{OhOh}\right]^T \end{aligned} \tag{2.35}$$

ce qui conduit à écrire l'erreur quadratique sous la forme :

$$\begin{aligned} &E\left\{\left\|O - Oh * W\right\|^2\right\} = \\ &tr\left[P_{OO} - P_{OOh} P^{-1}{}_{OhOh} P^T{}_{OOh}\right] + tr\left(\left[W^T - P_{OOh} P^{-1}{}_{OhOh}\right] P_{OhOh} \left[W^T - P_{OOh} P^{-1}{}_{OhOh}\right]^T\right) \end{aligned} \tag{2.36}$$

Par conséquent cette expression sera minimale pour :

$$\boxed{W^T = P_{OOh}P^{-1}{}_{OhOh}} \tag{2.37}$$

De cette expression, on déduit que la matrice W est issue de la méthode de pseudo inversion de Moore Penrose qui a la propriété de minimiser l'erreur au sens des moindres carrés. La pseudo inversion du vecteur Oh donne en effet

$$W = \left(Oh^T Oh\right)^{-1} Oh^T O \tag{2.38}$$

Ce que nous venons de montrer est valable pour n'importe quel vecteur Oh donné, c'est à dire pour n'importe quelle fonction d'activation choisie au sein de la couche cachée. Nous verrons ultérieurement qu'un choix judicieux des fonctions d'activation peut encore accroître la qualité de l'apprentissage.

9 Structuration

Nous allons exposer dans ce paragraphe la méthode préconisée pour optimiser la phase d'apprentissage. Jusqu'à présent, nous avons considéré le réseau neuronal comme une boîte noire capable de déterminer un lien entre des entrées et des sorties, ce qui en fait un estimateur universel. Cette vue de l'esprit est intéressante pour certaines disciplines, par exemple la météorologie, où les modèles climatiques ne sont pas vraiment connus [AUSSEM 95]. Ceci n'est pas le cas pour le génie électrique où des modèles validés existent [SILVESTER 70] [SEGUIER 77][OSTOVIC 89][LOUIS 91]. En effet, on dispose des composants dont les caractéristiques nous sont familières : résistance, inductance, capacité et interrupteur (diode, thyristor, ...). Chaque composant pris indépendamment des autres constitue un système très simple ; par contre, la connexion de ces différents composants peut s'avérer complexe et non linéaire. Il en est de même pour les systèmes mécaniques qui se composent essentiellement de ressorts et de masses. Cependant, comme on dispose de modèles de connaissance validés, il est possible de séparer les équations qui régissent les lois physiques en parties continues, discontinues, linéaires ou non. Ainsi, il apparaît judicieux d'utiliser une structure de réseau

calquée sur la décomposition des équations. On parlera dans ce cas de réseau spécialisé à l'approche d'une fonction d'un type bien établi.

Nous allons montrer ici combien la structure neuronale choisie est importante pour obtenir une bonne qualité de modélisation. Ensuite nous passerons en revue les différents cas de caractéristiques à modéliser et nous donnerons les structures neuronales adéquates.

9.1 Importance de la structure mathématique :

Nous allons chercher quel doit être l'estimateur $\hat{y}$ optimal d'une fonction qui minimise la somme des erreurs quadratiques, c'est à dire l'espérance mathématique, entre l'estimateur optimal $\hat{y}$ et la fonction réelle à approcher y. Il est alors évident de penser que la valeur de cette espérance mathématique va dépendre essentiellement de la structure mathématique prise pour l'estimateur, c'est à dire de la nature et du nombre de ou des fonction(s) choisies pour cet estimateur. Ainsi, si les fonctions d'approximation sont choisies judicieusement et de manière optimale, alors l'inégalité suivante est vérifiée :

$$\boxed{E\left\{\left\|Y-\hat{Y}(X)\right\|^2\right\} \leq E\left\{\left\|Y-g(X)\right\|^2\right\}} \qquad (2.39)$$

ce qui signifie en fait que l'estimateur optimal possède la propriété de minimiser l'erreur quadratique d'estimation et ceci par rapport à n'importe quelle autre fonction *g(X)*. Il peut cependant exister des cas où l'équation (2.39) ne se vérifie pas, notamment lorsqu'on s'impose une structure pour l'estimateur $\hat{Y}$ qui ne correspond en aucune façon à la structure de la fonction réelle. C'est à dire que si *g(X)* ressemble de par son équation à la fonction réelle à modéliser, alors l'inégalité (2.39) change de sens. Autrement dit, il vaut mieux approcher une droite *y* par une droite *g(X)* que par une estimation à base de fonctions quelconques. Cette remarque traduit l'idée que l'optimalité de l'estimateur est relative à la structure de ce dernier par rapport à celle de la fonction réelle. Elle sera développée dans le paragraphe concernant la structuration.

9.2 Equations linéaires

Compte tenu de l'importance de la structuration du réseau, montrée dans le paragraphe précédent, nous allons maintenant passer en revue les différents types d'équations rencontrées et proposer une structure pour chacune pour chacune d'elles.

Considérons une équation linéaire du type :

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_p x_p \tag{2.40}$$

Nous définissons respectivement les matrices X et Y définissant les tableaux des n échantillons dont on dispose ainsi que β le vecteur des coefficients reliant les matrices X et Y.

$$X=\begin{pmatrix} 1 & x_{11} & \cdots & \cdots & \cdots & x_{p1} \\ \vdots & \vdots & & & & \vdots \\ \vdots & \vdots & & & & \vdots \\ 1 & x_{11n} & \cdots & \cdots & \cdots & x_{pn} \end{pmatrix}$$ correspondant aux valeurs suivantes $$Y=\begin{pmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{pmatrix}$$

Ainsi le problème se résume à la résolution d'une équation matricielle :

$$Y=X\beta \tag{2.41}$$

Connaissant les matrices X et Y, il est alors aisé de trouver le vecteur β solution de cette équation par inversion de la matrice X (ou plus exactement par pseudo inversion car la matrice X n'est pas forcément une matrice carrée) telle que :

$$\left(X^t X\right)^{-1} X^t Y = \beta \tag{2.42}$$

$$\beta = X^+ Y \tag{2.43}$$

où X^+ désigne la pseudo inverse de X.

9.3 Equations non linéaires à structure connue

9.3.1 Equations non linéaires linéarisables

9.3.1.1 Equation à variables séparées

Nous appellerons équations à variables séparées, les équations linéairement dépendantes de variables, de puissances ou de produits de variables.

Considérons une équation non linéaire du type :

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_1^{\,k} + \alpha_3 x_1^{\,j} + \cdots + \alpha_p x_1 x_2 \qquad (2.44)$$

Cette expression peut être linéarisée par changement de variable en posant

$\begin{cases} z_1 = x_1 \\ z_2 = x_1^{\,k} \\ z_3 = x_1^{\,j} \\ \cdots \\ z_p = x_1 x_2 \end{cases}$, on obtient l'expression linéaire $y = \alpha_0 + \alpha_1 z_1 + \alpha_2 z_2 + \cdots + \alpha_p z_p$ ce qui nous ramène au problème précédent en utilisant les matrices d'échantillons suivants :

$$X = \begin{pmatrix} 1\, x_{11} & x^k{}_{11} & x^j{}_{11} & \cdots & x_{11}x_{21} \\ \vdots\ \vdots & & & & \vdots \\ \vdots\ \vdots & & & & \vdots \\ 1\, x_{1n} & x^k{}_{1n} & x^j{}_{11} & \cdots & x_{1n}x_{2n} \end{pmatrix} \text{ correspondant aux valeurs suivantes } Y = \begin{pmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{pmatrix}$$

9.3.1.2 Equation à variables non séparées

Nous appellerons équations à variables non séparées, les équations linéairement dépendantes de fonctions non linéaires, par exemple, la fonction (2.46) :

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 \ln(x_1) \qquad (2.45)$$

Cette équation peut être linéarisée en posant comme variable $x_2 = \ln(x_1)$, d'où,

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 \tag{2.46}$$

la régression linéaire se fera alors sur les matrices

$X=\begin{pmatrix} 1 & x_{11} & \ln(x_{11}) \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ 1 & x_{1n} & \ln(x_{1n}) \end{pmatrix}$ correspondant aux valeurs suivantes $Y=\begin{pmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{pmatrix}$

9.3.2 Equations non linéaires non linéarisables

Il s'agit par exemple d'équations dont les paramètres à identifier sont cachés dans une fonction du genre $y = f_{nonlinéaire}\left(x_1, x_2, \cdots, x_p, \alpha_1, \cdots, \alpha_q\right)$, par exemple $y = \alpha_0 \sin(w x_1) + \alpha_1 x^2{}_1$

Ne connaissant pas le paramètres *w*, il est alors impossible de linéariser l'équation précédente. Dans ces conditions, il est impératif de se référer au traitement des équations non linéaires à structure inconnue.

9.4 Equations non linéaires à structure inconnue

Nous nous intéressons maintenant aux équations non linéaires dont la structure est inconnue; trois cas sont étudiés, le niveau de complexité augmentant avec le manque de connaissance sur l'expression de l'équation.

9.4.1 Variables et nature des fonctions connues

Deux cas peuvent se présenter ; les fonctions peuvent être complètement non linéaires mono ou multi variables, ou bien les fonctions, que l'on appelle hétérogènes, sont composées en partie de termes linéarisables et d'autres termes non linéarisables.

9.4.1.1 Fonctions non linéaires mono ou multivariable

Il s'agit de fonctions du type

$$y = f_{nonlinéaire}(x_1) \text{ ou } y = f_{nonlinéaire}(x_1, x_2, \cdots, x_p) \tag{2.47}$$

Comme on ne connaît pas l'expression de la fonction non linéaire, il est alors impossible de linéariser cette expression. Par conséquent, il est nécessaire de développer une structure capable d'approcher ou d'interpoler $f_{nonlinéaire}$ par une série de fonctions g_i de même nature c'est à dire non linéaires. Nous reviendrons sur le choix de ces fonctions dans le paragraphe suivant. L'équation estimée pourra alors s'exprimer de la façon suivante :

$$y = \sum_{i=0}^{m} \alpha_i g_i(x_1) \text{ ou, pour le cas multivariable, } y = \sum_{i=0}^{m} \alpha_i g_i(x_1, \cdots, x_p)$$

ce qui peut se traduire par une équation linéaire en prenant comme changement de variable respectivement dans les cas mono et multivariables $z_i = g_i(x_1)$ et $z_i = g_i(x_1, \cdots, x_p)$

Dans ce cas la matrice X définie plus haut devient la suivante :

$$X = \begin{pmatrix} g_1(x_{11}, \cdots, x_{p1}) & \cdots & \cdots & g_m(x_{11}, \cdots, x_{p1}) \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ g_1(x_{1n}, \cdots, x_{pn}) & \cdots & \cdots & g_m(x_{1n}, \cdots, x_{pn}) \end{pmatrix}$$

Bien entendu, il est encore possible de simplifier cette matrice si on connaît les variables qui interviennent dans la ou les fonction(s) à approcher. Par exemple, considérons la fonction suivante à modéliser : $y = f_{nonlinéaire}(x_1, x_3) + h_{nonlinéaire}(x_2)$. La séparation des rôles des trois variables peut ici être mise à profit pour simplifier la procédure d'apprentissage.

$$X=\begin{pmatrix} g_1(x_{11},x_{31}) & \cdots & \cdots & g_m(x_{11},x_{31}) & k_1(x_{21}) & \cdots & \cdots & k_l(x_{2n}) \\ \vdots & & & \vdots & \vdots & & & \vdots \\ \vdots & & & \vdots & \vdots & & & \vdots \\ g_1(x_{1n},x_{3n}) & \cdots & \cdots & g_m(x_{1n},x_{3n}) & k_1(x_{21}) & \cdots & \cdots & k_l(x_{2n}) \end{pmatrix}$$

Modélisation de $f_{nonlinéaire}(x_1,x_3)$

Modélisation de $h_{nonlinéaire}(x_2)$

9.4.1.2 Fonctions hétérogènes mono ou multivariable

Il s'agit d'équations composées à la fois de fonctions linéaires et non linéaires, mono ou multivariables du type

$$y = f_{nonlinéaire}(x_1,x_2,\cdots,x_p) + g_{linéaire}(x_1,x_2,\cdots,x_p)$$

Dans ce cas il est nécessaire de scinder cette équation en deux parties, l'une linéaire et l'autre non linéaire ; la partie non linéaire doit être elle même décomposée en une série linéarisable comme nous l'avons vu précédemment. Une fois de plus on retombe sur une forme linéarisable et la matrice X devient alors la suivante

$$X=\begin{pmatrix} g_1(x_{11},\cdots,x_{p1}) & \cdots & \cdots & g_m(x_{11},\cdots,x_{p1})\,\phi & \cdots & \cdots & \phi \\ \vdots & & & \vdots \quad \vdots & & & \vdots \\ \vdots & & & \vdots \quad \vdots & & & \vdots \\ g_1(x_{1n},\cdots,x_{pn}) & \cdots & \cdots & g_m(x_{1n},\cdots,x_{pn})\,\phi & \cdots & \cdots & \phi \end{pmatrix}$$

où ϕ représente une variable, une puissance ou un produit de variables.

9.4.2 Nature inconnue des fonctions

Si maintenant on se place dans le cas où on ne connaît ni l'expression ni la nature des fonctions, il ne reste qu'une solution, approcher cette équation par une fonction non linéaire généraliste, c'est à dire capable d'émuler n'importe quelle fonction. Dans ce cas, la structure doit être également généraliste, c'est à dire une série de fonctions non linéaires. Si, de plus, on ne connaît ni le rôle des différentes variables ni dans quelles fonctions elles interviennent (en

cas d'hétérogénéité), alors effectivement il est impossible d'éliminer certaines variables, ce qui n'autorise pas la spécialisation.

9.5 Représentation symbolique des structures

Pour résumer ce paragraphe, nous allons représenter les correspondances entre les différentes matrices X et la structure neuronale adaptée.

9.5.1 Equation linéaire

$$X=\begin{pmatrix} 1\,x_{11} & \cdots & \cdots & \cdots & x_{p1} \\ \vdots\ \vdots & & & & \vdots \\ \vdots\ \vdots & & & & \vdots \\ 1\,x_{11n} & \cdots & \cdots & \cdots & x_{pn} \end{pmatrix}$$ correspond à la structure

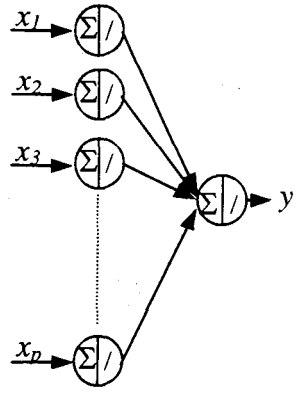


Fig II.9.1 : Structure neuronale linéaire

9.5.2 Equation non linéaire à structure connue

9.5.2.1 Equation non linéaire linéarisable

Reprenons à titre d'exemple la matrice X du paragraphe 9.3.1.1 :

$$X=\begin{pmatrix} 1\,x_{11} & x^{k}{}_{11} & x^{j}{}_{11} & \cdots & x_{11}x_{21} \\ \vdots\ \vdots & & & & \vdots \\ \vdots\ \vdots & & & & \vdots \\ 1\,x_{1n} & x^{k}{}_{1n} & x^{j}{}_{11} & \cdots & x_{1n}x_{2n} \end{pmatrix}$$ correspond à la structure

avec $\phi 1 = x^k$ et $\phi 2 = x^j$

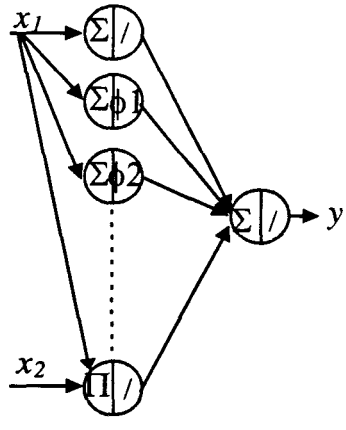


Fig II.9.2 : Structure neuronale non linéaire spécialisée

9.5.2.2 Equation non linéaire non linéarisable

Soit une fonction non linéaire du type

$$y = f_{nonlinéaire}(x_1, x_3) + h_{nonlinéaire}(x_2)$$

la matrice d'échantillons devient la suivante :

$$X=\begin{bmatrix} g_1(x_{11},x_{31}) & \cdots & \cdots & g_m(x_{11},x_{31}) & k_1(x_{21}) & \cdots & \cdots & k_l(x_{2n}) \\ \vdots & & & \vdots & \vdots & & & \vdots \\ \vdots & & & \vdots & \vdots & & & \vdots \\ g_1(x_{1n},x_{3n}) & \cdots & \cdots & g_m(x_{1n},x_{3n}) & k_1(x_{21}) & \cdots & \cdots & k_l(x_{2n}) \end{bmatrix}$$

Modélisation de $f_{nonlinéaire}(x_1, x_3)$

Modélisation de $h_{nonlinéaire}(x_2)$

La structure neuronale correspondante est alors la suivante :

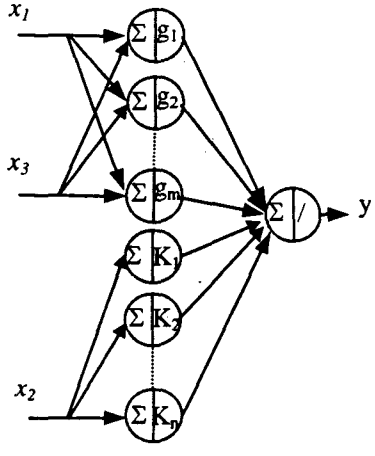


Fig II.9.3 : Structure neuronale non linéaire homogène généralisée

Reprenons maintenant le cas d'une équation non linéaire hétérogène, c'est à dire une équation comportant une partie non linéaire et une autre linéaire ou linéarisable.

$$X=\begin{pmatrix} g_1(x_{11},\cdots,x_{p1}) & \cdots & \cdots & g_m(x_{11},\cdots,x_{p1})\,\phi & \cdots & \cdots & \phi \\ \vdots & & & \vdots & \vdots & & \vdots \\ \vdots & & & \vdots & \vdots & & \vdots \\ g_1(x_{1n},\cdots,x_{pn}) & \cdots & \cdots & g_m(x_{1n},\cdots,x_{pn})\,\phi & \cdots & \cdots & \phi \end{pmatrix}$$

La structure neuronale correspondante devient alors la suivante :

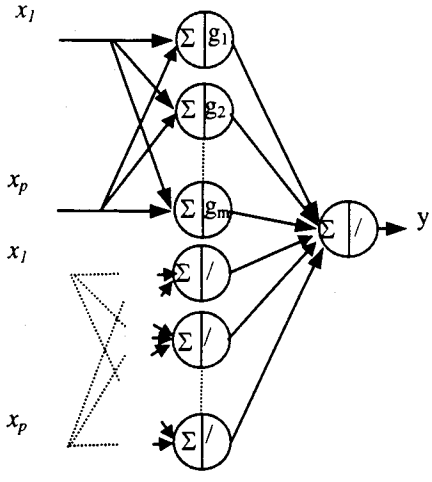


Fig II.9.4 : Structure neuronale non linéaire hétérogéne et généralisée

10 Choix des fonctions d'activation

10.1 Fonctions d'activation orthogonales

Lorsque les fonctions d'activation utilisées sont du type de celles présentées figure II.3.2, l'apprentissage réalisé à l'aide de la méthode de rétropropagation du gradient de l'erreur peut amener la solution optimale ainsi trouvée dans un minimum local. Tout dépend en fait des valeurs des poids initialisés aléatoirement au départ de la phase d'apprentissage. Considérons, par exemple, un réseau à un seul neurone avec un seul poids w à déterminer. Supposons que le profil de l'erreur en fonction de la valeur du poids soit celui présenté figure II.10.1.a.

La figure présente des minima locaux et un seul minimum global. Par conséquent, si la valeur du poids au départ de la phase d'apprentissage est la valeur *D1*, la solution optimale qui sera obtenue sera la valeur *A1*. Par contre, si la valeur de départ a la valeur *D2*, la

rétropropagation conduira au minimum global autrement dit à la valeur de w qui minimisera de façon optimale l'erreur entre la fonction réelle et la fonction apprise.

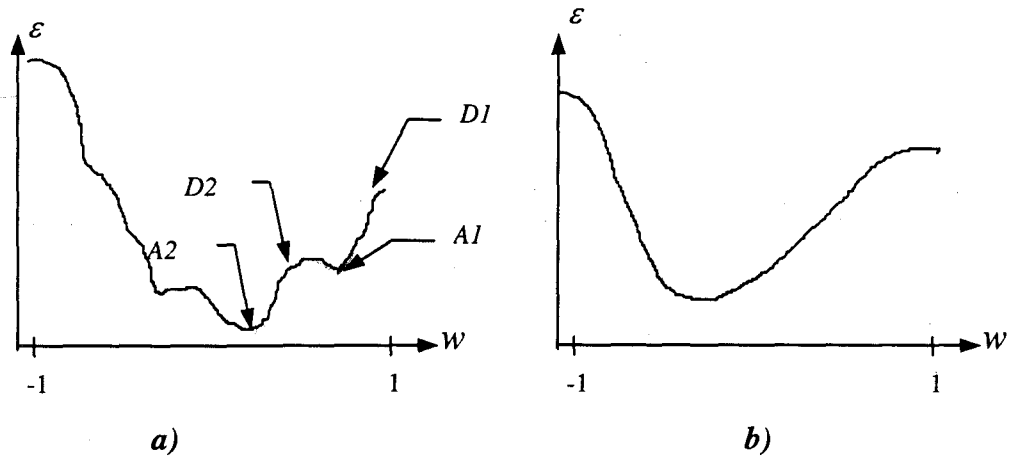


Fig II.10.1 : Minimum global et minimum local.

C'est pourquoi il peut sembler judicieux d'utiliser des fonctions d'activation dont le profil d'erreur ne présente qu'un seul minimum local qui est de surcroît global (figure II.10.1.b). Les fonctions orthogonales présentent cette propriété [FRANÇOIS 96], [BORNE 92].

10.2 Fonctions radiales

Pour notre part, nous utiliserons essentiellement des fonctions à bases radiales, *Radial Basis Functions* (RBF), qui ne sont autres que des fonctions gaussiennes. Le double intérêt d'utiliser ce type de fonction est de donner un caractère stochastique aux fonctions d'activation et de centrer celles ci sur des domaines de travail distincts. L'importance des variables aléatoires gaussiennes réside dans leurs propriétés mathématiques et dans leur capacité à modéliser de nombreux phénomènes naturels [HARTMAN 90] [PARK 91]. On peut ainsi obtenir des domaines d'appartenance semblables à ceux rencontrés en logique floue. Nous allons démontrer que ce type de fonctions couplées à une régression linéaire permet d'obtenir un estimateur optimal au sens des moindres carrés.

10.2.1 Démonstration intuitive

10.2.1.1 Fonction à une dimension

Une courbe peut être considérée comme une suite infinie de points. Par conséquent on peut écrire l'équation de la fonction estimée :

$$\hat{f}(t) = \sum_{\Delta=-\infty}^{+\infty} \delta(t-\Delta) f(t) \qquad (2.48)$$

$$\delta(0) = 1 \quad \text{et} \quad \delta(t) = 0 \ \ \forall t \neq 0$$

Cette équation classique, utilisée en traitement de signal, traduit en fait l'échantillonnage d'un signal par multiplication de ce signal avec un peigne d'impulsions comme l'illustre la figure II.10.2.

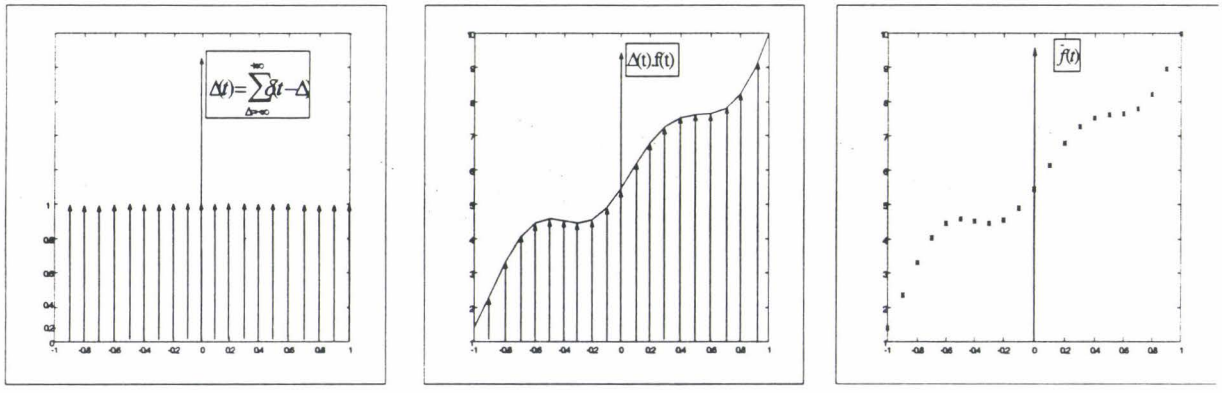


Fig II.10.2 : Echantillonnage par un peigne d'impulsions

Si on pouvait disposer d'un réseau à une couche cachée avec un nombre infini de neurones à fonctions d'activation impulsionnelles, l'estimation de la fonction à identifier serait parfaite (2.49). Il suffirait que chaque fonction impulsion approche un seul point en la pondérant par l'ordonnée du point à apprendre.

$$\lim_{\Delta \to 0} \sum_{k=-\infty}^{\infty} f(k\Delta) = f(t) \qquad (2.49)$$

Malheureusement les performances actuelles de l'informatique ne permettent pas cette configuration et ne le permettront sans doute jamais en raison de la taille toujours finie des mémoires. C'est pourquoi en limitant le nombre de neurones, nous devons élargir les intervalles de travail des fonctions d'activation, à moins d'accepter une détérioration de la qualité de l'estimation. En d'autres termes, il faut augmenter la variance des fonctions d'activation. Néanmoins, si les fonctions d'activation sont des créneaux alors la recomposition de la fonction à identifier sera échelonnée.

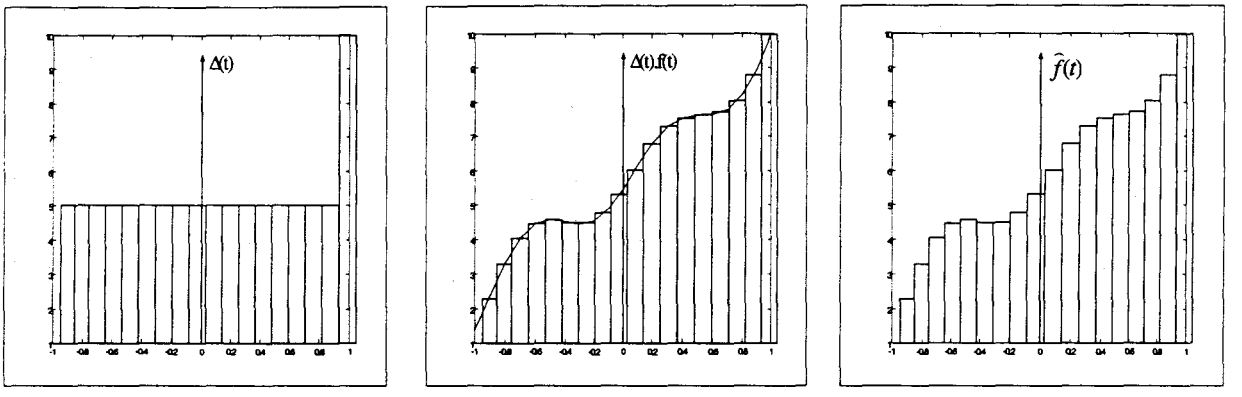


Fig II.10.3 : Echantillonnage à l'aide de fonctions étagées

Dans le but de réduire considérablement le nombre de neurones , il est préférable d'utiliser des fonctions d'activation de type gaussiennes d'où la dénomination RBFNN *(Radial Basis Functions Neural Network)*. Ceci à l'avantage d'optimiser le travail de chaque neurone sur une portion d'intervalle et de donner un caractère statistique aux échantillons (probabilité qu'a x d'avoir comme valeur y). Etant donné que la majorité des phénomènes physiques obéissent à des lois normales, c'est à dire gaussiennes, ou convergent vers ce type de loi, on comprend alors l'intérêt d'une modélisation par RBFNN. De cette façon on crée un ensemble de fonctions et domaines d'appartenance semblable à celui utilisé en logique floue. Il est à noter qu'en logique floue, les fonctions d'appartenance sont généralement discontinues (triangle) afin de simplifier les calculs, alors qu'avec un réseau de neurones on utilise majoritairement des fonctions continues (II.10.4).

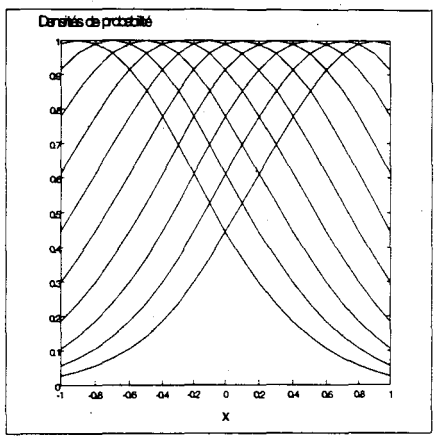


Fig II.10.4 : Ensemble de fonctions d'appartenance

10.2.1.2 Fonctions de dimensions supérieures

Ce qui vient d'être montré pour une fonction à une dimension, est également vrai pour une fonction de dimension supérieure (II.10.5). Nous nous arrêterons dans ces explications intuitives aux fonctions à deux dimensions, puisqu'au delà la visualisation n'est plus possible. Par contre, une généralisation aux fonctions à n dimensions sera proposée au paragraphe suivant.

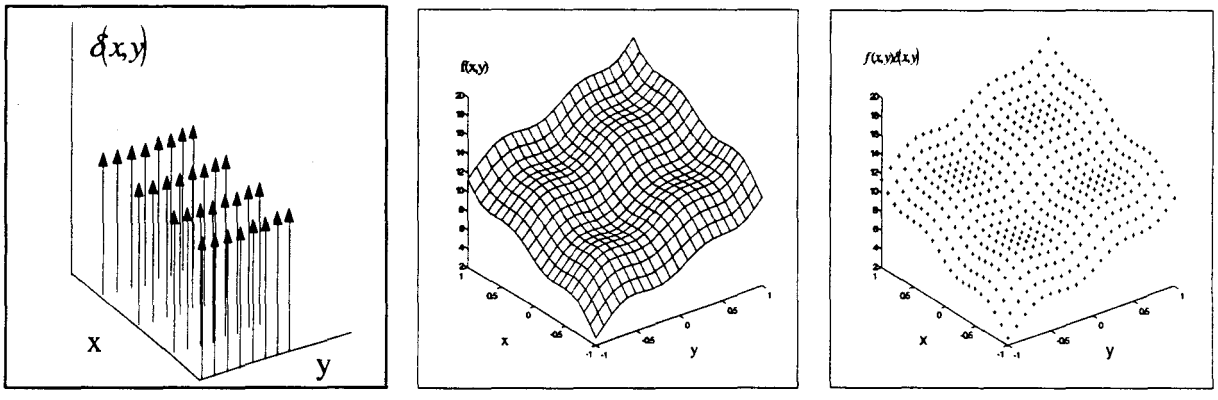


Fig II.10.5 : Echantillonnage par un peigne de Dirac à deux dimensions

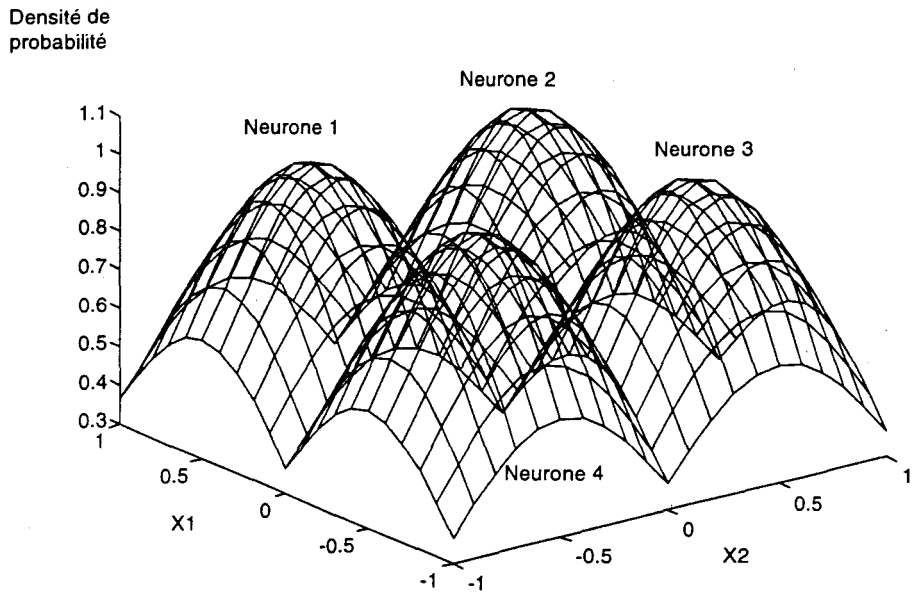


Fig II.10.6: Ensemble de fonctions d'appartenance neuronal à deux dimensions

Dans le cas de la figure II.10.6, l'ensemble du domaine d'échantillonnage est couvert par seulement quatre fonctions d'activation. Il est bien évident que la qualité de l'estimation sera croissante en fonction du nombre de fonctions d'activation. On notera également la répartition homogène des fonctions d'activation sur le domaine d'application.

10.2 2 Démonstration mathématique

Nous allons montrer que si X et Y sont des vecteurs aléatoires gaussiens, l'estimateur optimal $\hat{y}(x)$ est linéaire. Le fait que les vecteurs soient gaussiens signifie que les variables sont normalement distribuées. Ce que nous allons démontrer ici n'est en fait que le théorème 'central limite' qui explique qu'une variable normalement distribuée est engendrée par l'addition de fluctuations nombreuses, indépendantes et normalement distribuées. Ceci va nous permettre en fait d'affirmer que si les fonctions d'activation sont gaussiennes, la somme de celles-ci pondérées par les poids de sortie (on considère toujours un réseau à une seule couche cachée) est capable d'approcher n'importe quelle autre fonction dont les échantillons sont normalement distribués.

10.2.2.2 Cas des vecteurs gaussiens

Soit un vecteur Z décomposable en deux vecteurs X et Y

$$Z = \begin{bmatrix} Y \\ X \end{bmatrix} \tag{2.50}$$

Un vecteur aléatoire Z de dimension n est gaussien (ou normal) si sa loi de densité de probabilité $N(m,P)$ est de la forme :

$$f_Z(z) = \frac{1}{(2\pi)^{\frac{n}{2}} |P|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(z-m)^T P^{-1}(z-m) \right) \tag{2.51}$$

où m et P représentent respectivement la moyenne de Z et P la matrice de covariance de la variable centrée Z-m. On peut noter les matrices m et P :

$$\begin{aligned} m &= \begin{bmatrix} my \\ mx \end{bmatrix} \\ P &= \begin{bmatrix} P_{YY} & P_{YX} \\ P_{YX} & P_{XX} \end{bmatrix} \end{aligned} \tag{2.52}$$

Suivant la loi de Bayes (multivariable)

$$f_{Y/X}(y,x)) = f_Z(z) |f_X(x)|^{-1} \tag{2.53}$$

on définit la matrice inverse des covariances :

$$P^{-1} = \begin{bmatrix} S_{YY} & S_{YX} \\ S^T{}_{YX} & S_{XX} \end{bmatrix} \tag{2.54}$$

$$\text{et } f_{Y/X}(y,x) = \frac{|P_{XX}|^{1/2}}{(2\pi)^{n/2}|P|^{1/2}} exp\left\{-\frac{1}{2}(z-m)^T \begin{bmatrix} S_{YY} & S_{YX} \\ S^T{}_{YX} & S_{XX} - P^{-1}{}_{XX} \end{bmatrix}(z-m)\right\} \quad (2.55)$$

en développant le terme dans l'exponentielle :

$$-\frac{1}{2}\left[y - m_Y - P_{YX}P^{-1}{}_{XX}(x-m_X)\right]^T \Delta^{-1}\left[y - m_Y - P_{YX}P^{-1}{}_{XX}(x-m_X)\right] \quad (2.56)$$

avec Δ tel que :

$$|P| = |P_{XX}||\Delta| \quad (2.57)$$

on en déduit que *E(Y/X)* est un vecteur aléatoire gaussien de loi normale $N(m_Y + P_{YX}P^{-1}{}_{XX}(x-m_X), \Delta)$

L'estimateur optimal est donc linéaire de la forme

$$\boxed{\hat{Y} = P_{YX}P^{-1}{}_{XX}X + \left[m_Y - P_{YX}P^{-1}{}_{XX}m_X\right]} \quad (2.58)$$

Si *X* et *Y* sont normalement distribués, alors nous venons de montrer que le lien optimal entre *X* et *Y* est linéaire et peut être obtenu par une pseudo inversion.

Dans le cadre d'un réseau de neurones à une couche, on remplace dans les équations ci-dessus la variable aléatoire *X* par le vecteur *Oh*. Puisque l'on utilise des fonctions d'activation RBF dans la couche cachée, on force le vecteur *X* à être gaussien Pour obtenir l'estimateur optimal on utilisera une pseudo-inversion de matrice qu'on retrouve dans le terme : $P_{YX}P^{-1}{}_{XX}$.

10.3 Mise en œuvre pratique

Notons tout d'abord que les fonctions d'appartenance du réseau de neurones peuvent être de nature continue (loi de distribution normale) ou discontinue (créneau ou triangle).

Cette remarque permet d'améliorer la qualité de l'estimation lorsque l'allure de la fonction à identifier est pressentie comme le préconise la structuration.

En utilisant une pseudo inversion de matrice, la qualité de l'estimation aux points d'échantillonnage est optimale au sens des moindres carrés. Néanmoins, si la densité de répartition des données n'est pas homogène sur tout le domaine d'identification et que les fonctions d'activation sont concentrées aux endroits où la population d'échantillons est la plus nombreuse, alors l'estimation sera toujours optimale aux points d'échantillonnage mais fortement dégradée ailleurs (figure II.10.7). Par conséquent, on veillera à ce que la somme des valeurs des fonctions d'activation soit à peu prés constante quel que soit le point du domaine, ce qui implique une répartition spatiale homogène des fonctions d'activation et des échantillons (figure II.10.9). De même, si les fonctions d'activation sont réparties de façon homogène sur l'espace de travail, mais que les échantillons ne le sont pas (figure II.10.8), alors la qualité d'approche de ces données sera moyenne ; de plus, les capacités d'extrapolation seront faibles.

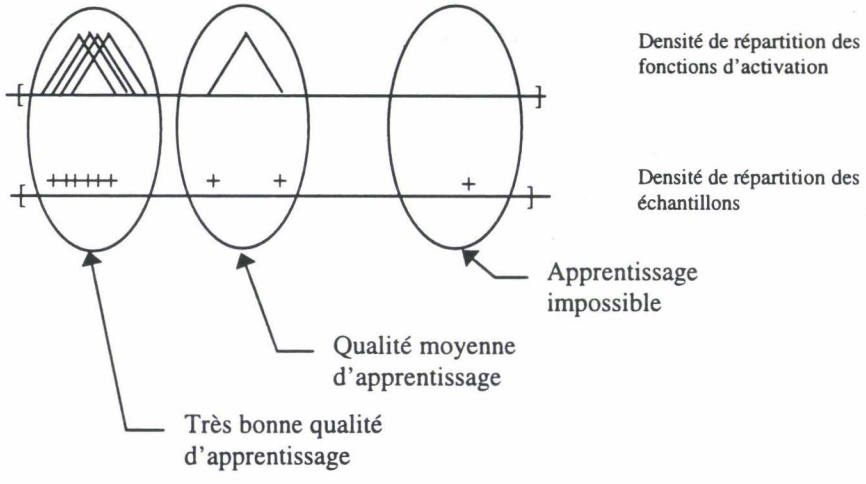


Fig II.10.7 : Mauvaises répartitions des échantillons et des fonctions d'activation

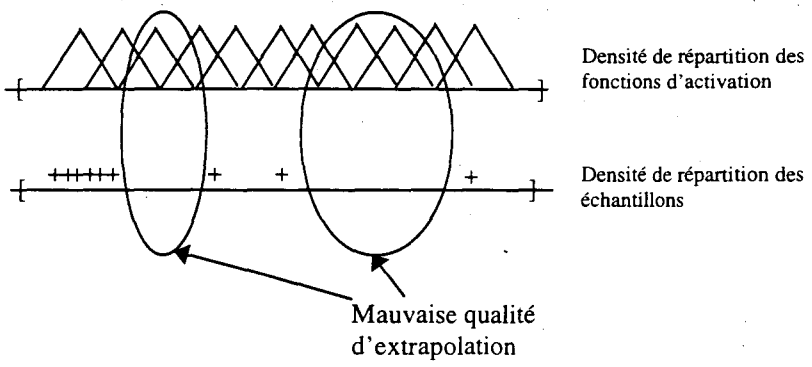


Fig II.10.8 : Mauvaises répartitions des échantillons et répartition homogène des fonctions d'activation

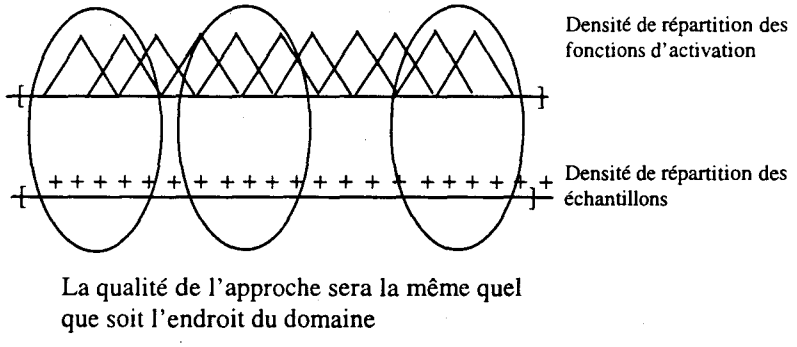


Fig II.10.9 : Répartitions homogènes des échantillons et des fonctions d'activation

De même, en calculant les poids et les biais conformément au paragraphe 8.1.1, il se peut que dans le cas de fonctions à base radiale, les bases ne se recouvrent pas ou pas suffisamment. Ceci dégrade en fait les capacités d'extrapolation entre ces fonctions (figure II.10.10)

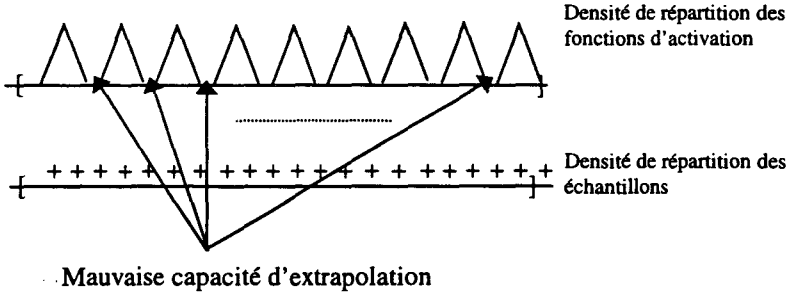


Fig II.10.10 : Mauvaise initialisation des fonctions d'activation

Pour pallier cette difficulté, nous proposons un algorithme d'initialisation dont la caractéristique majeure réside dans le fait que le choix des poids et biais d'entrée se fait de façon itérative de manière à augmenter plus ou moins la variance des fonctions RBF pour obtenir un 'auto recouvrement' de ces fonctions (Fig II.10.11).

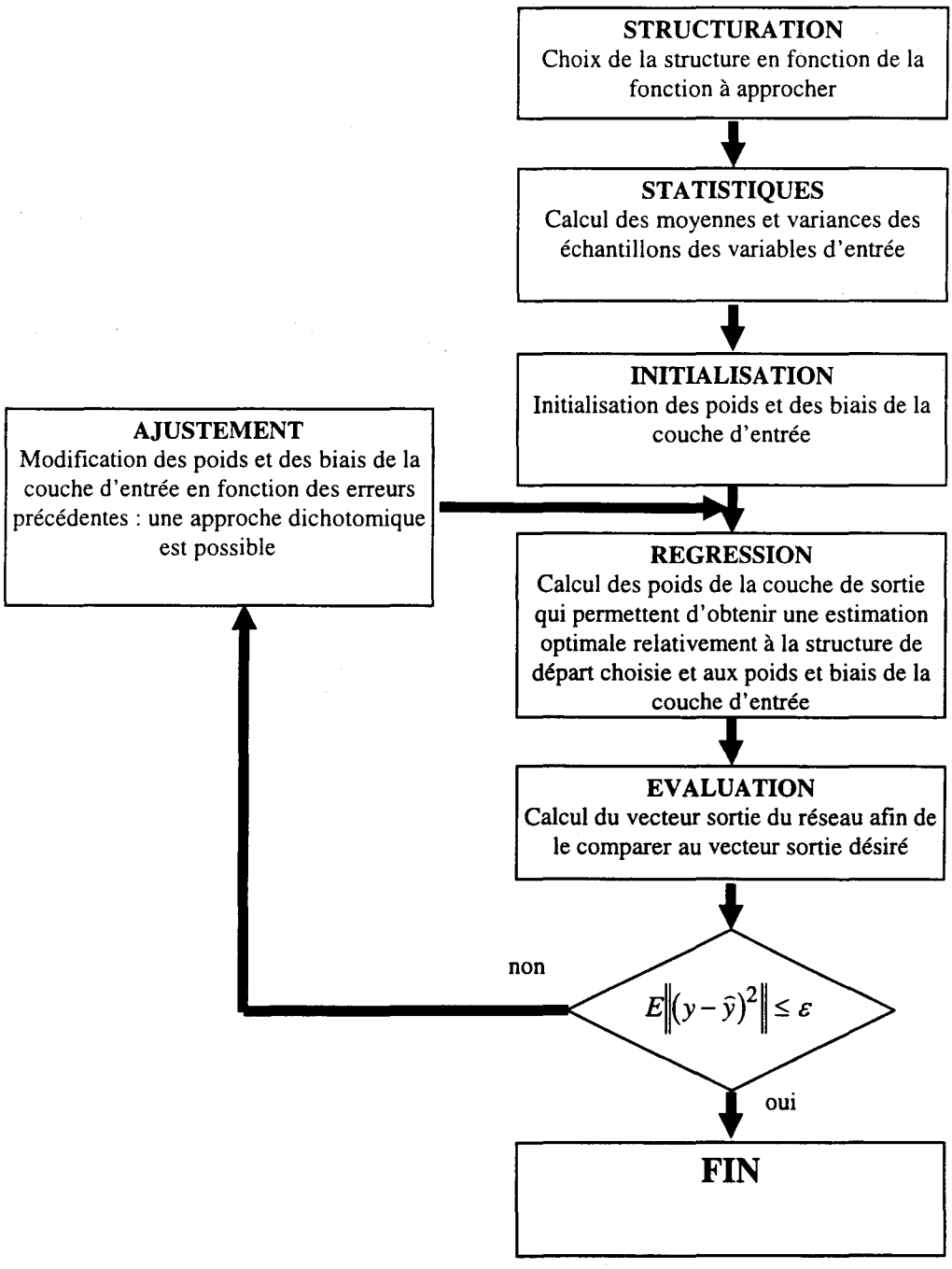


Fig II.10.11 : Organigramme de l'algorithme d'initialisation

11 Conclusion

Dans ce chapitre, nous avons énoncé les fondements des réseaux de neurones, les architectures statiques et dynamiques ainsi que différentes méthodes classiques d'apprentissage. Nous avons montré quelles étaient les inconvénients de ces méthodes, notamment par rapport au temps de calcul requis. Une méthode d'apprentissage dite par initialisation a été proposée afin d'optimiser le temps de calcul d'une part, améliorer la qualité d'apprentissage d'autre part. En effet, nous avons insisté dans ce chapitre sur l'importance que revêt le choix de la structure mathématique pour l'apprentissage d'une caractéristique quelconque. De plus, nous avons prouvé que le choix des fonctions d'activation de type gaussien est optimal pour l'approximation de fonctions non linéaires dont la structure mathématique est inconnue. Ce chapitre a contribué ainsi à montrer que les réseaux de neurones étaient des estimateurs universels et pouvaient être utilisés en temps que modèles, à condition de disposer d'informations sur le système, données sous forme d'échantillons. Nous verrons dans les chapitres suivants comment utiliser ces réseaux et cette méthode d'apprentissage pour modéliser ou commander des systèmes électrotechniques non linéaires.

Chapitre 3

La modélisation neuronale de systèmes électrotechniques

1 Introduction

Nous venons de voir au second chapitre que les réseaux de neurones sont capables de modéliser n'importe quelle fonction récurrente ou non. Cette modélisation mathématique, a priori séduisante, semble prometteuse quant à ses facultés d'approcher des phénomènes dont on ne connaît pas les relations internes, ou dont on ne souhaite pas identifier les paramètres. C'est pourquoi, il semble opportun de tester l'approche neuronale sur les phénomènes électromagnétiques et électromécaniques. L'intérêt de cette modélisation réside à la fois dans la conception de structure capable d'émuler un système et ou de le commander. Nos efforts se porteront sur la qualité de la modélisation tout en gardant à l'esprit la notion de coût en temps de calcul.

Après avoir évoqué les utilisations inhérentes aux différentes architectures neuronales ainsi que les limites du domaine de validité des modèles neuronaux, nous appliquons les principes de l'approche neuronale sur la modélisation de la saturation magnétique, puis sur la modélisation de cycles majeurs d'hystérésis ; enfin, nous appliquons ce principe à la modélisation de charges mécaniques non linéaires.

2 Classification des modèles

Nous pouvons établir un classement des différents modèles simplement en fonction de deux critères : la précision et le temps de calcul pour l'obtenir. La méthode des éléments finis est classée en tête selon ces deux critères (figure III.2.1). Néanmoins, en raison d'un temps de

calcul généralement long, ce type de modélisation ne peut être utilisé que pour simuler les systèmes, à des fins d'analyse ou de conception. Il ne peut donc pas être utilisé en commande compte tenu des exigences temporelles du calcul 'en ligne' (figure III.2.2). En ce qui concerne le réseau neuronal généraliste, c'est à dire celui dont l'architecture n'a pas été structurée en fonction de la caractéristique à apprendre et, de plus, qui a été entraîné par une descente de gradient, il ne réalise pas le bon compromis temps de calcul précision. Enfin nous pouvons classer le modèle paramétrique et le réseau neuronal spécialiste, tel que nous l'avons défini au chapitre précédent, quasiment dans la même gamme de compromis puisque les structures sont pratiquement les mêmes. Néanmoins, du fait de la capacité du réseau de neurones à prendre en compte des non linéarités dont on ignore la structure, en contrepartie, ce modèle, plus complexe, requiert un temps de calcul plus long. Toutefois, la précision du modèle neuronal dépasse bien souvent celle obtenue par un modèle analytique.

De ces différents compromis, on peut préconiser l'utilisation de tel ou tel modèle dans les domaines allant de la conception à la commande (figure III.2.2). Il est manifeste que les temps de calcul relatifs à la méthode des éléments finis les empêchent d'être utilisés dans le domaine de la commande. Quant aux autres types de modèles, leur utilisation dans les différents domaines dépend essentiellement de la complexité de la caractéristique réelle du système à modéliser.

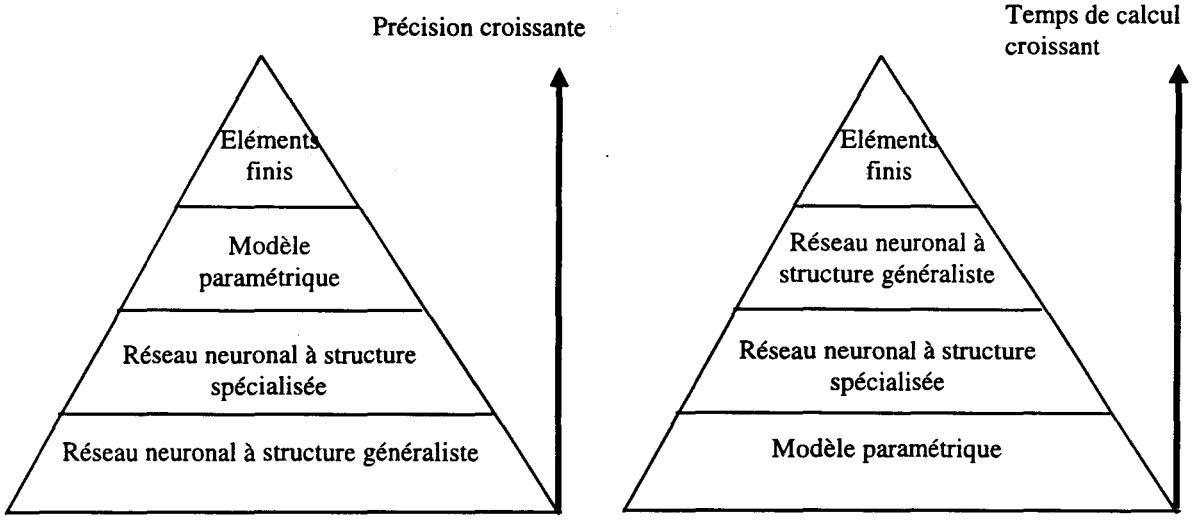


Fig III.2.1 : Compromis précision temps de calcul

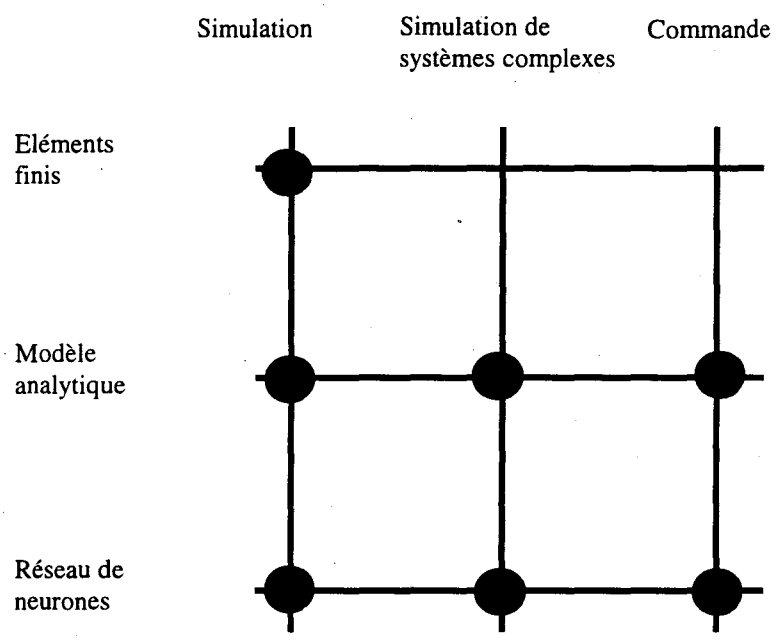


Fig III.2.2 : Utilisations préconisées des différents modèles

3 Utilisation appropriée des réseaux

Le choix de l'architecture neuronale qui doit être utilisée est intimement lié à l'utilisation du modèle : la structure du réseau sera différente selon que celui-ci sera utilisé en commande ou en simulation. Le graphe informationnel causal présenté au premier chapitre permet d'une part de définir quelle est la structure neuronale à adopter et, d'autre part, il met en évidence les variables entrant en jeu .

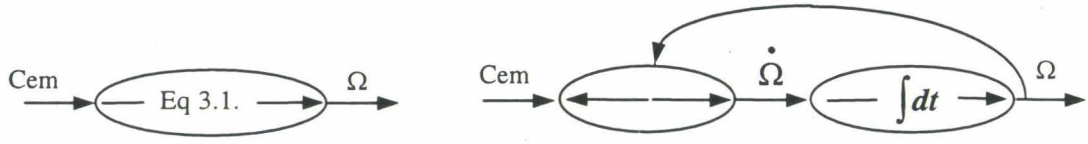
3.1 Architectures employées

Dans le cas de la modélisation d'un système complexe, le graphe informationnel causal offre la possibilité de représenter une relation complexe par plusieurs relations élémentaires. Le choix de l'architecture neuronale utilisée pour la modélisation de ces relations dépend essentiellement de leur rigidité ou de leur causalité. Si on s'en tient aux réseaux feedforward et extérieurement bouclés (type Hopfield), en omettant par conséquent les réseaux complètement récurrents, les réseaux de neurones apparaissent dans le formalisme G.I.C, comme des relations rigides. On peut alors approcher n'importe quelle relation, rigide ou causale, pour peu qu'on puisse extraire la rigidité de ces équations.

Prenons par exemple le cas d'un couple électromagnétique C_{em} transmis par un moteur sur l'axe de son rotor, en considérant un coefficient de frottement f et un moment d'inertie J; l'équation de la mécanique se traduit de la façon suivante :

$$J\frac{d\Omega}{dt} + f\Omega = Cem \qquad (3.1)$$

Cette équation, traduite sous forme de G.I.C, peut être représentée de deux façons (figure III.3.1a et b) : la première (figure III.3.1.a) exprime la causalité globale de l'équation (3.1) tandis que la seconde décompose cette équation en deux relations, l'une rigide, l'autre causale, en faisant apparaître l'accélération de façon naturelle.



a) b)

Fig III.3.1 Extraction d'une relation rigide d'une relation causale continue

En vue d'une implantation sur calculateur, il est nécessaire de transformer ce graphe causal 'continu' en graphe causal 'discret'. Une représentation discrète pourrait être celle de la figure III.3.2.

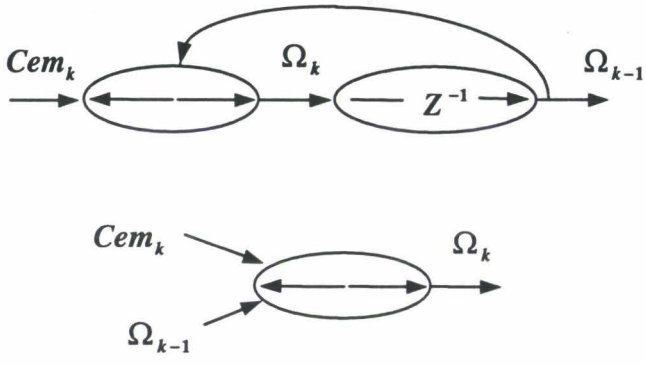


Fig III.3.2 : Représentation discrète

Par conséquent on s'aperçoit d'après le graphe informationnel causal 'discret' que si on désire un modèle capable de donner la vitesse en fonction du couple à chaque instant *k,* il est nécessaire d'utiliser la vitesse à l'instant *k-1*. Ceci implique, pour une modélisation neuronale, l'utilisation d'un réseau récurrent de type Hopfield, où la récurrence sera traitée à l'extérieur du réseau.

Par contre, si on désire un modèle capable de donner le couple électromoteur à chaque instant *k*, on peut faire glisser sur le graphe informationnel discret les grandeurs d'un côté ou de l'autre de l'opérateur graphique, puisque ce dernier représente une relation rigide. On déduit de cette figure que le couple électromoteur donné à un instant *k*, est fonction des vitesses actuelle et précédente, résultat en accord avec la causalité naturelle qui ne peut considérer que la dérivée à gauche. Par conséquent, l'architecture neuronale requise est du type feedforward. Celle ci sera notamment très employée en commande (figure III.3.3).

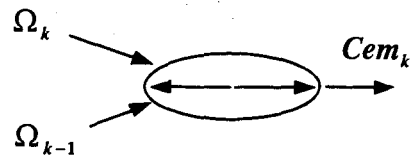


Fig III.3.3 Modèle utilisé en commande

Toutes ces remarques sont applicables aux systèmes non linéaires. Nous représenterons les relations rigides et causales non linéaires par des doubles cercles. Dans l'exemple précédent, supposons cette fois que le couple de frottements est non linéairement dépendant de la vitesse, l'équation (3.1) devient :

$$J\frac{d\Omega}{dt} + f(\Omega)\Omega = Cem \qquad (3.2)$$

Par application des principes vus précédemment, le graphe se représente de la façon suivante (figure III.3.4) :

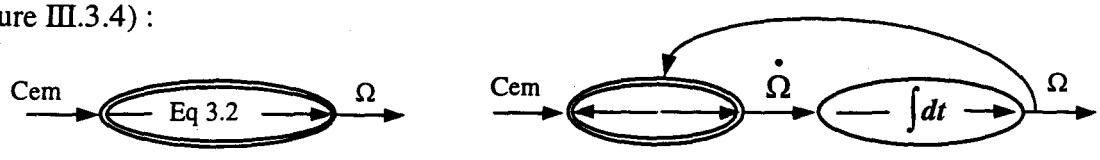
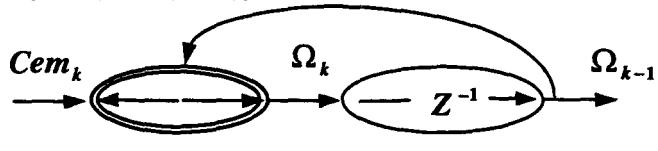


Fig III.3.4 Représentation continue d'une relation non linéaire causale

De même pour la représentation discrète :



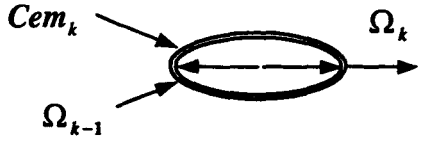
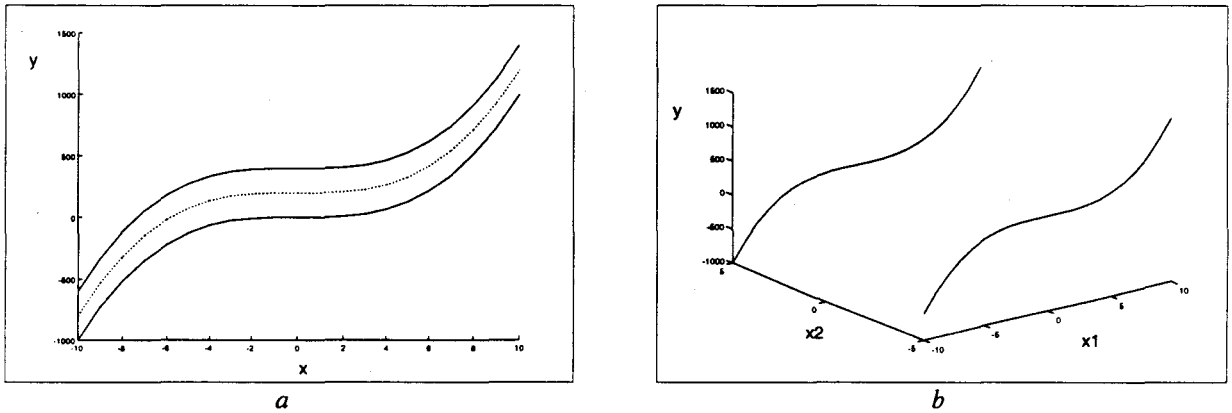


Fig III.3.5 : Représentation discrète d'une relation causale non linéaire

3.2 A propos de l'univocité

Nous avons exposé au second chapitre les méthodes permettant d'obtenir le meilleur estimateur au sens des moindres carrés. Celles ci ont toutes en commun la propriété de minimiser la somme des écarts quadratiques entre les données réelles et les estimations issues d'une structure mathématique choisie en guise de modèle. Néanmoins cette approximation universelle peut être faussée par le manque d'univocité du lien entrées - sorties. Prenons l'exemple d'une relation liant une variable y à une variable x. Considérons que les données réelles sont celles exposées figure III.3.6.a en trait plein. Il apparaît que cette relation n'est pas univoque car, pour une même valeur de x, correspondent plusieurs valeurs de y. Si on essaye de trouver une relation optimale au sens des moindres carrés entre x et y, on obtient la courbe moyenne en pointillée sur la figure III.3.6.a. Par contre, si on différencie les deux courbes à l'aide d'une deuxième variable d'entrée, alors on obtient une relation univoque non plus entre x et y, mais entre $x1$,$x2$ et y. C'est à dire que pour chaque couple ($x1$,$x2$) correspond une seule valeur de y et vice versa.



a *b*

Fig III.3.6 : Relation non univoque a) rendue univoque b) par un changement de variable

Ainsi la symbolique des graphes causaux se justifie une fois de plus. En effet nous avons vu dans le chapitre 'Position du problème', que seules les relations rigides pouvaient être inversées directement. L'inversion impose également que ce genre de relation soit univoque ; par conséquent, si on désire utiliser un réseau de neurones pour approcher une relation rigide en vue de l'inversion, il faut au préalable garantir que cette relation rigide soit réellement univoque, sinon il faut impérativement la transformer soit au moyen d'une application, soit grâce à un changement de variable comme l'illustre la figure III.3.7.

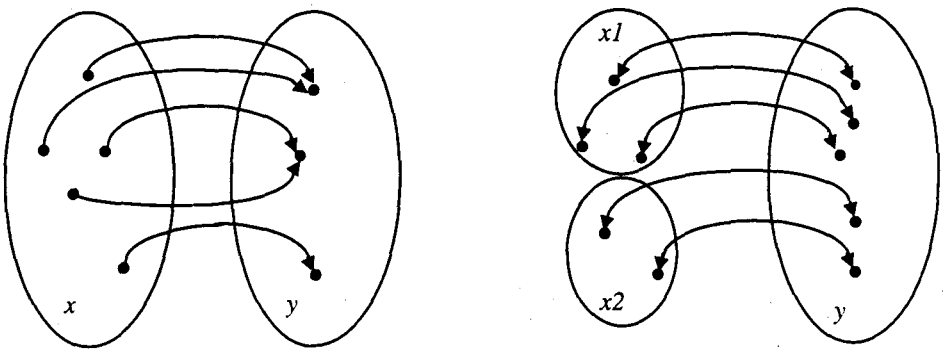


Fig III.3.7 : Transformation en relation univoque

4 Domaine de validité du modèle

4.1 Au sein du domaine d'échantillonnage

Un modèle est bon tant qu'il n'est pas faux!. Cette Lapalissade pourrait en faire rire plus d'un si on ne se posait la question de savoir jusqu'où le modèle est correct. En effet cette trivialité nous amène à penser naturellement que, puisqu'il est évalué à partir d'échantillons, le modèle sera correct sur cet espace d'échantillonnage. Seulement, si l'échantillonnage n'a pas été effectué de manière homogène, c'est à dire si les échantillons ne sont pas équitablement répartis sur l'espace des variables mises en jeu, le modèle aura une précision importante là où la densité d'échantillons sera la plus grande et inversement. Il est certes possible de tirer parti de cet inconvénient pour renforcer l'apprentissage d'une zone bien déterminée, ou pour effectuer un filtrage d'informations bruitées [PIERLOT 96]. Néanmoins, suivant la méthode d'apprentissage proposée au second chapitre (cf chapitre II paragraphe 8), le fait de spécialiser la topologie du réseau sur un problème donné, revient à imposer l'allure de la fonction d'approximation. Par ailleurs l'augmentation de la quantité d'échantillons autour d'un domaine particulier peut s'avérer inutile si la spécificité a déjà été prise en compte grâce à l'adjonction d'un neurone à fonction d'activation particulière.

Afin de vérifier la validité du modèle sur l'espace d'échantillonnage on procède de la façon suivante. Avant la phase d'apprentissage, on extrait une partie des échantillons recueillis. On forme ainsi un premier groupe nécessaire à l'apprentissage et un second qui sera utilisé exclusivement pour tester la validité du modèle. Il est à noter bien entendu que les

échantillons du second groupe n'entrent pas dans la phase d'apprentissage afin de ne pas influencer le test de validité

4.2 En dehors du domaine d'échantillonnage

Une fois que la validité a été montrée à l'intérieur de l'espace d'échantillonnage, on peut alors se poser la question de savoir si le réseau est capable d'extrapoler la caractéristique. Puisque toute la théorie du neuromimétisme est fondée sur l'apprentissage, il est raisonnable de penser que les réseaux neuronaux sont incapables d'évaluer quelque chose qu'ils n'ont pas appris ou qui se situe en marge de leur connaissance. C'est pourquoi deux précautions sont nécessaires afin d'éviter les effets néfastes d'un fonctionnement hors limites.

- La première concerne la nature des fonctions d'activation : celles ci doivent être bornées pour éviter les divergences rapides en dehors du domaine des fonctions reconstituées.
- La seconde concerne l'utilisation des modèles neuronaux en commande. Etant donné la qualité hasardeuse des modèles en dehors de leur domaine d'échantillonnage, il sera impératif de contraindre l'utilisation des modèles sur leurs domaines d'apprentissage au moyen de limitations sur les grandeurs d'entrée. Ce point sera plus amplement discuté dans le cadre de la stabilité des commandes neuronales au quatrième chapitre.

4.3 Validité au cours du temps

Même si les modèles neuronaux sont capables d'approcher de façon quasiment parfaite une caractéristique quelconque, il résulte néanmoins des erreurs de modélisation. Si le réseau est de type récurent, par accumulation, alors ces erreurs risquent d'entraîner le modèle dans des zones de fonctionnement inconnues du réseau. C'est pourquoi, il est nécessaire dans certains cas d'adapter le modèle au cours du temps. Cette opération permet de :

- forcer les entrées du réseau à l'intérieur du domaine d'apprentissage,
- continuer l'apprentissage afin de remettre à jour le modèle vis à vis des variations temporelles des caractéristiques qu'on désire modéliser.

Cette adaptation au cours du temps devra toujours être réalisée par rapport à un modèle de référence ou par rapport à des échantillons prélevés sur le processus. Elle ne peut généralement se faire que grâce à la méthode de rétropropagation du gradient.

5 Application 1 : Inductance saturable

Dans ce paragraphe nous allons nous intéresser à la modélisation de la saturation magnétique par réseaux de neurones. L'idée maîtresse est de disposer d'un modèle non linéaire simple d'une charge RL magnétiquement saturée, dans l'espoir de l'intégrer dans des schémas de commande ou de surveillance. En effet, cet élément magnétique se retrouve à plusieurs reprises dans les graphes informationnels causaux qui décrivent les lois de la physique des machines électriques [DEGOBERT 97][LEMAIRE-SEMAIL 97].

5.1 Modélisation de l'inductance saturable

5.1.1 Rappels

Au sein d'un matériau magnétique, l'excitation H(A/m) et le champ *B(T)* sont reliés par la formule

$$B = \mu H = \mu_0 \mu_r H \tag{3.3}$$

avec μ_0 la perméabilité magnétique du vide et μ_r la perméabilité magnétique relative du matériau. Considérons le circuit magnétique de la figure III.5.1, en effectuant les hypothèses classiques de l'approche analytique (absence de fuites, répartition homogène du champ *B*), le flux du champ magnétique à travers une surface *S* s'exprime :

$$\phi = BS \tag{3.4}$$

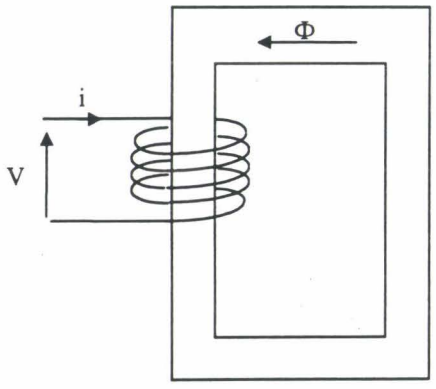
De plus, l'excitation magnétique circulant le long d'un circuit de longueur l est proportionnelle au courant i et au nombre de spires n de la bobine :

$$Hl = ni \tag{3.5}$$

Le flux capté par les n spires peut alors s'écrire :

$$\Phi = n\phi = \frac{\mu n^2 S}{l} i = L_p i \tag{3.6}$$

avec L_p l'inductance propre du circuit. Néanmoins, comme la perméabilité magnétique μ dépend de l'organisation cristalline du matériau, elle même dépendante non linéairement de H, l'inductance propre devient fonction du courant qui la traverse : en négligeant le phénomène d'hystérésis on notera $L(i)$.



V: tension aux bornes de l'inductance

i : courant dans l'inductance

B : champ magnétique

H : excitation magnétique

Φ : flux magnétique

n : nombre de spires

μ : perméabilité absolue du matériau

S : section moyenne d'une spire

l : longueur moyenne du circuit

L_p : inductance propre

Fig III.5.1 : Montage décrivant la loi de Faraday.

La loi d'Ohm s'exprime de façon classique :

$$U = Ri + \frac{d\Phi}{dt} \tag{3.8}$$

Deux représentations sont alors utilisables selon que l'on choisit le courant ou le flux comme variable d'état.

5.1.1.1 Variable d'état 'courant'

Dans cette optique, le flux doit être exprimé en fonction du courant :

$$\frac{d\Phi}{dt} = \frac{d(L(i)i)}{dt} = \frac{d(L(i))}{di}\frac{di}{dt}i + L(i)\frac{di}{dt} = \frac{di}{dt}\left(\frac{d(L(i))}{di}i + L(i)\right) \tag{3.9}$$

d'où

$$\boxed{(\Re) : U = Ri + \frac{di}{dt}\left(\frac{d(L(i))}{di}i + L(i)\right)} \tag{3.10}$$

soit encore

$$\boxed{U = Ri + \frac{di}{dt}L^*} \tag{3.11}$$

avec L^* l'inductance incrémentale ou inductance dynamique telle que $L^* = \left(\frac{d(L(i))}{di}i + L(i)\right)$.

La représentation sous forme de G.I.C de ce système est donné figure III.5.2.

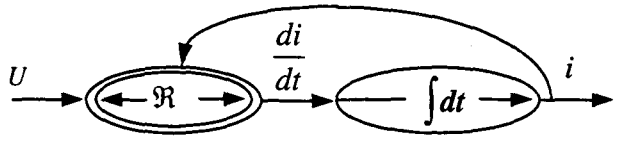


Fig III.5.2 : Graphe causal d'une inductance saturable

Cette solution est facilement exploitable à condition de connaître $L^*(i)$. Si tel n'est pas le cas une autre mise en équation est utilisable.

5.1.1.2 Variable d'état 'flux'

On propose alors une méthode fondée sur la mise sous forme d'une équation différentielle du premier ordre du flux. De cette manière les relations non linéaires restent rigides et ne dépendent que de $L(i)$.

$$\begin{aligned} U &= R\frac{\Phi}{L(i)} + \frac{d\Phi}{dt} \\ i &= \frac{\Phi}{L(i)} \end{aligned} \quad \Leftrightarrow \quad \boxed{\begin{aligned} \frac{d\Phi}{dt} &= U - Ri \qquad (\Re 1) \\ i &= \frac{\Phi}{L(i)} \qquad (\Re 2) \end{aligned}} \tag{3.12}$$

Ce qui donne en représentation graphique :

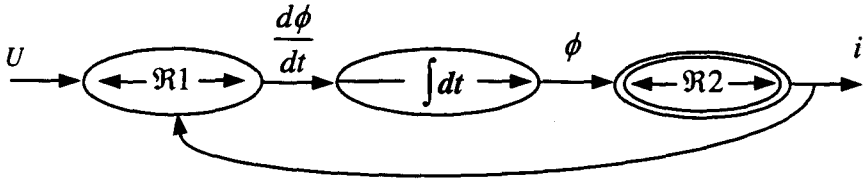


Fig III.5.3 : Graphe causal des équations liant le flux et le courant au sein d'une inductance saturable

La solution de ce système nécessite la connaissance de la relation $\Re 2$ plus accessible que la relation L*(i) ; elle nous a également amené à utiliser les méthodes de discrétisation de Runge Kutta et de substitution.

L'identification de $\Re 2$ peut se faire à partir d'essais expérimentaux ou de calculs par éléments finis, grâce à un modèle neuronal.

En effet, nous disposons de quelques valeurs de l'inductance (signalés sur les graphes III.5.5 par des ronds 'o'); il est cependant nécessaire de concevoir un modèle capable de donner des valeurs quel que soit le courant. Les propriétés d'approche stochastique des réseaux neuronaux évoquées au second chapitre, conviennent parfaitement bien dans ce cas.

5.1.2 Modèle neuronal et résultats

Nous utilisons la méthode d'apprentissage par 'initialisation' décrite au second chapitre. On choisit donc un réseau comportant une seule couche cachée. Morphologiquement, la courbe *L(i)* apparaît comme étant une fonction non linéaire du courant. Par conséquent, le réseau n'aura qu'une seule entrée, le courant et des fonctions d'activations non linéaires.

L'architecture retenue est donc celle de la figure III.5.4, les fonctions g_i étant des fonctions d'activation non linéaires :

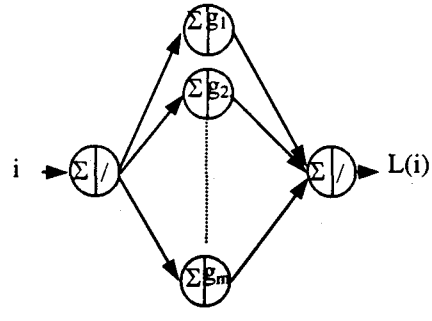
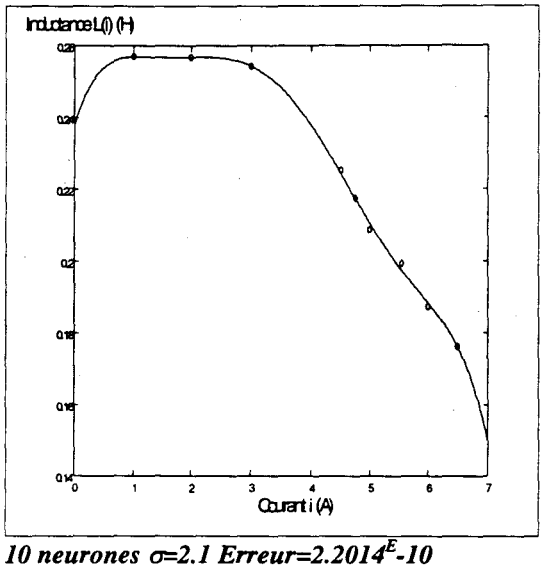


Fig III.5.4 : Architecture neuronale employée

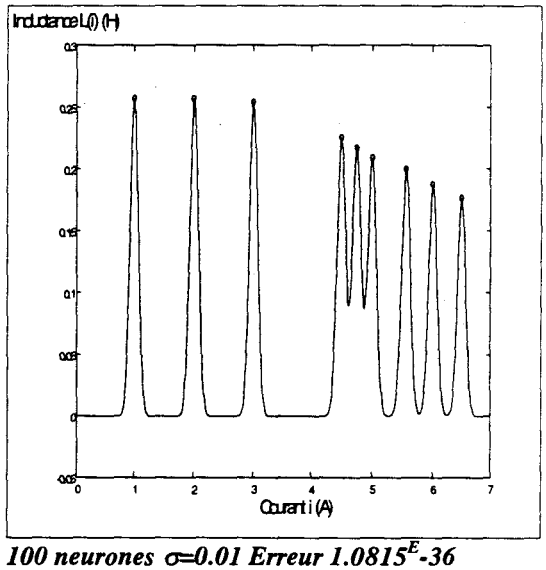
Les figures III.5.5 montrent différents résultats obtenus avec l'architecture présentée précédemment pour différentes quantités de neurones et différentes largeurs de base pour les fonctions RBF. La figure III.5.5.a représente le résultat obtenu avec 10 neurones à fonction d'activation RBF (trait continu), une variance large (σ=2.1). On remarque que la faible quantité d'échantillons influence la qualité d'apprentissage notamment aux limites du domaine. Ceci s'explique par le fait que le réseau sollicite extrêmement ses capacités d'extrapolation mais ne peut restituer ce qu'il n'a pas appris. Ce dernier propos est également vérifié sur la figure III.5.5.b où le nombre de neurones est excessif et la variance très petite (σ=0.01). On remarque alors que ce qui a été appris n'est autre que l'information qui était disponible. Dans ce cas, à cause d'une variance trop petite le réseau n'est pas capable d'extrapoler ailleurs. Afin de pallier le manque d'information, nous avons décidé d'en rajouter

au moyen d'une interpolation linéaire entre les échantillons d'origine. Les informations rajoutées sont désignées par des '+' sur la figure III.5.5.c et sont utilisées dans les figures *c* et *d*. Sur la figure III.5.5.c on remarque que, malgré une densité d'information plus importante, la qualité d'apprentissage est déplorable en raison d'une variance choisie trop petite pour les fonctions RBF. Par contre, dans la dernière illustration, toutes les conditions sont réunies pour obtenir une bonne approche de la courbe réelle



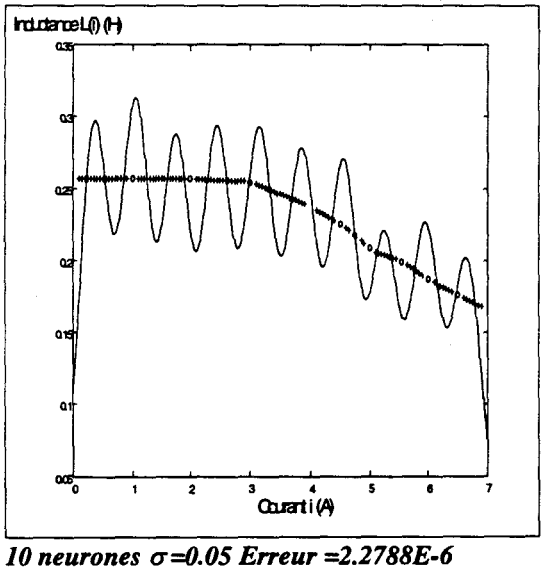
10 neurones σ=2.1 Erreur=2.2014^{E}-10

a)



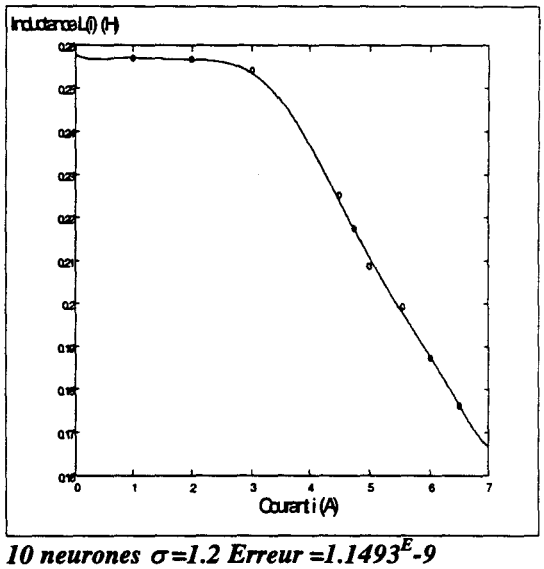
100 neurones σ=0.01 Erreur 1.0815^{E}-36

b)



10 neurones σ=0.05 Erreur =2.2788E-6

c)



10 neurones σ=1.2 Erreur =1.1493^{E}-9

d)

Fig III.5.5 : Résultats d'apprentissage

On remarque également d'après les figures III.5.5 que l'erreur quadratique issue de la comparaison entre les valeurs apprises et les valeurs réelles ne donne pas une information

correcte sur la qualité d'apprentissage. En fait on s'aperçoit qu'un réseau neuronal est capable de très bien apprendre ce qu'on lui demande d'approcher. Pour, généraliser, si les informations ne sont pas suffisamment denses, le réseau est confronté à ses propres capacités d'extrapolation. Pour pallier ce phénomène deux solutions sont envisageables :

- La première consiste à prendre plus d'échantillons pour couvrir un peu mieux le domaine étudié. Si on ne peut obtenir plus d'informations, il est nécessaire de générer des échantillons, avant la phase d'apprentissage, par le biais de méthodes d'extrapolation linéaire ou non linéaire entre les échantillons de base.

- La seconde consiste à utiliser des fonctions dont les rayons d'action s'étendent plus largement sur l'ensemble du domaine d'étude.

5.2 Modélisation globale d'une charge RL non linéaire

Dans ce qui précède, nous avons établi un modèle basé sur des échantillons de la caractéristique *L(i)*. Néanmoins, à partir de mesures physiques, c'est la caractéristique *u(i)* qui est directement accessible. L'évolution de l'inductance peut s'en déduire mais pas de façon immédiate. Nous allons donc essayer de mettre au point une topologie de réseau capable de reproduire le modèle global de la charge. On peut noter que la composante résistive d'une telle charge peut également varier de façon non linéaire.

5.2.1 Topologie du réseau

Afin de déterminer quelle est la topologie optimale pour apprendre des caractéristiques d'inductances saturables, il faut discrétiser l'équation différentielle qui décrit les lois physiques. Ainsi en appliquant la méthode d'Euler sur l'équation (3.10), et en appelant *Te* la période d'échantillonnage, on arrive à :

$$U = Ri + \frac{di}{dt}\left(\frac{d(L(i))}{di}i + L(i)\right) \Leftrightarrow U_k = Ri_k + \frac{\left(i_{k+1} - i_k\right)}{Te}\left(L'(i_k)i_k + L(i_k)\right) \qquad (3.13)$$

avec L' la dérivée par rapport à i de $L(i)$. La représentation sous forme de graphe informationnel causal présentée figure III.5.6. indique bien les variables mises en jeu dans les diverses relations.

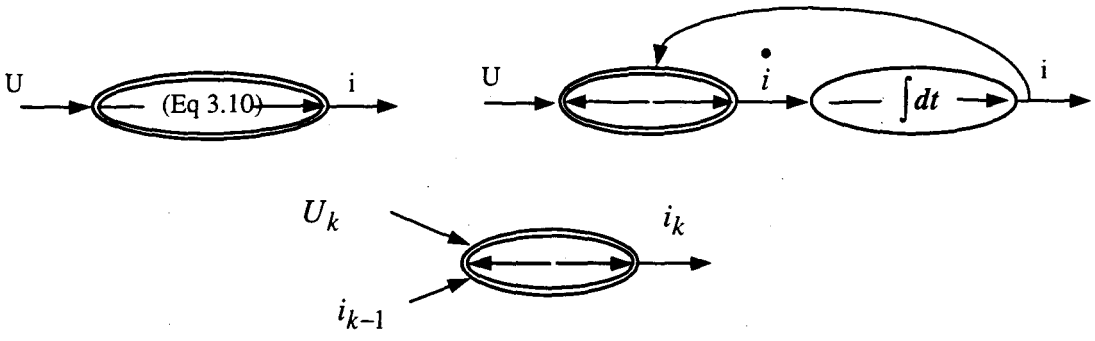


Fig III.5.6 : Graphe causal des équations liant la tension et le courant au sein d'une inductance saturable

Pour définir et multiplier la structure neuronale du modèle de l'inductance il est nécessaire de connaître le nombre de variables de l'équation de i_k et la nature de cette fonction par rapport aux variables mises en jeu.

Du graphe figure III.5.6 on peut déduire *ik* en fonction de *Uk* et *ik-1* :

$$i_k = \lambda_{nonlinéaire}(U_k, i_{k-1}) \tag{3.15}$$

Par conséquent pour approcher la relation (3.15) nous avons besoin d'utiliser un réseau dont les fonctions d'activation non linéaires à deux dimensions sont connectées sur la tension U_k et le courant i_k issu de la sortie du réseau. Ceci nous donne l'architecture de la figure III.5.7 :

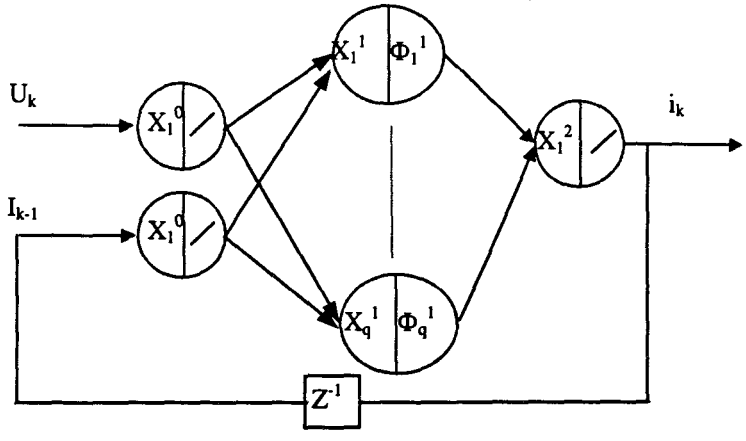


Fig III.5.7 Architecture neuronale retenue

5.2.2 Echantillonnage du plan de phase

Avant de commencer la phase d'apprentissage, il faut disposer d'échantillons. Cette fois, ceux ci se situent dans un espace à trois dimensions, c'est à dire dans l'espace U_k, ik, i_{k-1}. Pour cela, il est nécessaire d'exciter le processus de telle manière qu'on puisse collecter des courants i_k quel que soit le courant précédent i_{k-1}, et ceci pour toutes les tensions U_k. dans les limites physiques du processus. De façon pratique, cela se concrétise par l'application de rampes de tension sur l'inductance dont la fréquence varie afin de balayer tout l'espace d'état (figure III.5.8).

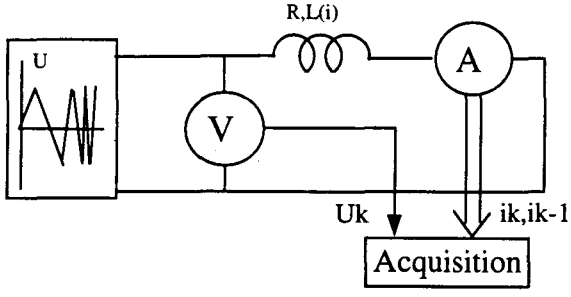


Fig III.5.8 Principe d'acquisition des échantillons

Le résultat de cette échantillonnage est illustré figure III.5.9.

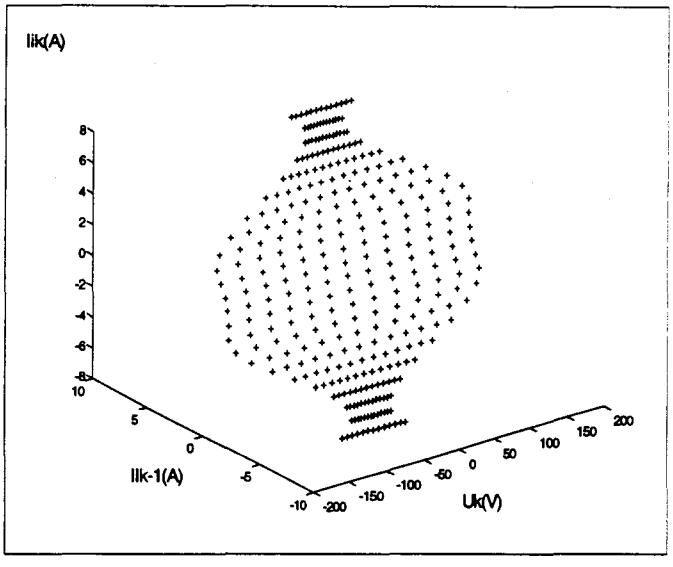


Fig III.5.9 : Echantillonnage de l'espace U_k, i_k, i_{k-1}

5.3 Comparaison des modèles.

Si on utilise la modélisation développée au paragraphe §5.1 comportant une architecture monocouche à dix neurones utilisant des fonctions d'activation RBF, le bilan du nombre de coefficients et de fonctions est le suivant :

10 coefficients pour les poids d'entrée,

10 coefficients pour les biais d'entrée,

10 coefficients pour les poids de sortie,

10 fonctions non linéaires.

Dans le cas d'une modélisation §5.2 on peut établir un premier bilan :

Si on considère que le réseau a deux entrées et que les fonctions d'activation sont des fonctions RBF à deux dimensions réparties sur 100 parcelles du domaine d'étude, alors cette architecture nécessite :

100 coefficients pour les poids d'entrée,

100 coefficients pour les biais d'entrée,

100 matrices 2x2 pour les biais d'entrée (car les fonctions RBF sont à deux dimensions),

100 fonctions RBF à deux dimensions,

100 coefficients pour les biais d'entrée.

Ce bilan très lourd peut être allégé en utilisant les propriétés des fonctions RBF. En effet, au lieu de placer une fonction RBF à deux dimensions sur chacune des 100 parcelles, il est possible de placer 10 fonctions RBF à une dimension sur un axe du plan des variables d'entrées, puis 10 autres sur un axe orthogonal au premier, et enfin d'effectuer le produit de ces différentes fonctions comme l'indique la figure III.5.10 La propriété utilisée est celle des fonctions de densités de probabilités marginales telle que, lorsque des aléas numériques (autrement dit des variables) sont indépendants, alors la densité de probabilité globale ou la fonction de répartition globale est respectivement égale au produit des densités marginales ou fonctions de répartition marginales :

$$f(x,y) = f(x\bullet) * f(\bullet\, y) \text{ ou } F(x,y) = F(x\bullet) * F(\bullet\, y) \qquad (3.16)$$

avec f la densité de probabilité et F la fonction de répartition qui n'est autre que l'intégrale de f. Cette propriété est fondamentale car elle permet non seulement de transformer une architecture composée de fonctions RBF multidimensionnelles en produit de fonctions RBF unidimensionnelles, mais est également valable pour les sigmoïdes puisque ce type de fonctions est l'intégrale d'une fonction Gauss ; elle représente, par conséquent, une fonction de répartition F.

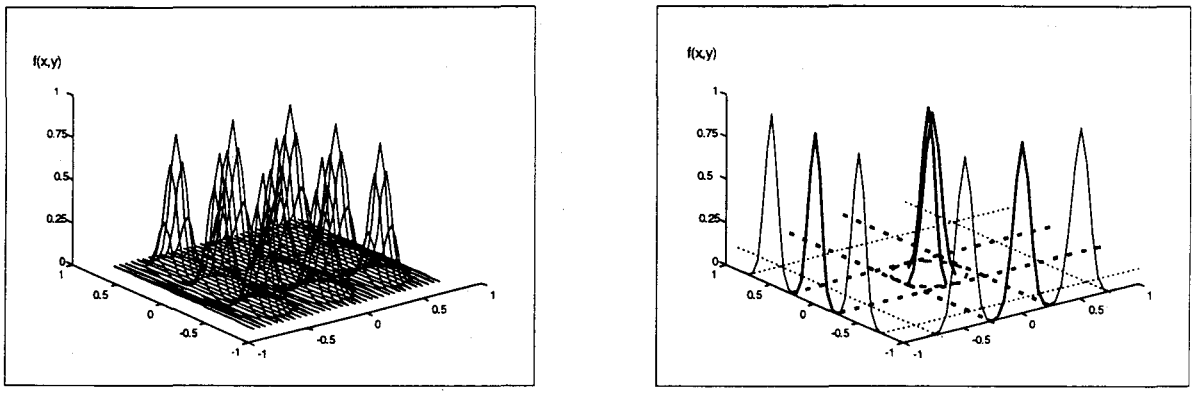


Fig III.5.10 Visualisation de la propriété des densités marginales

L'architecture du réseau de neurones peut alors être simplifiée comme l'indique la figure III.5.11.

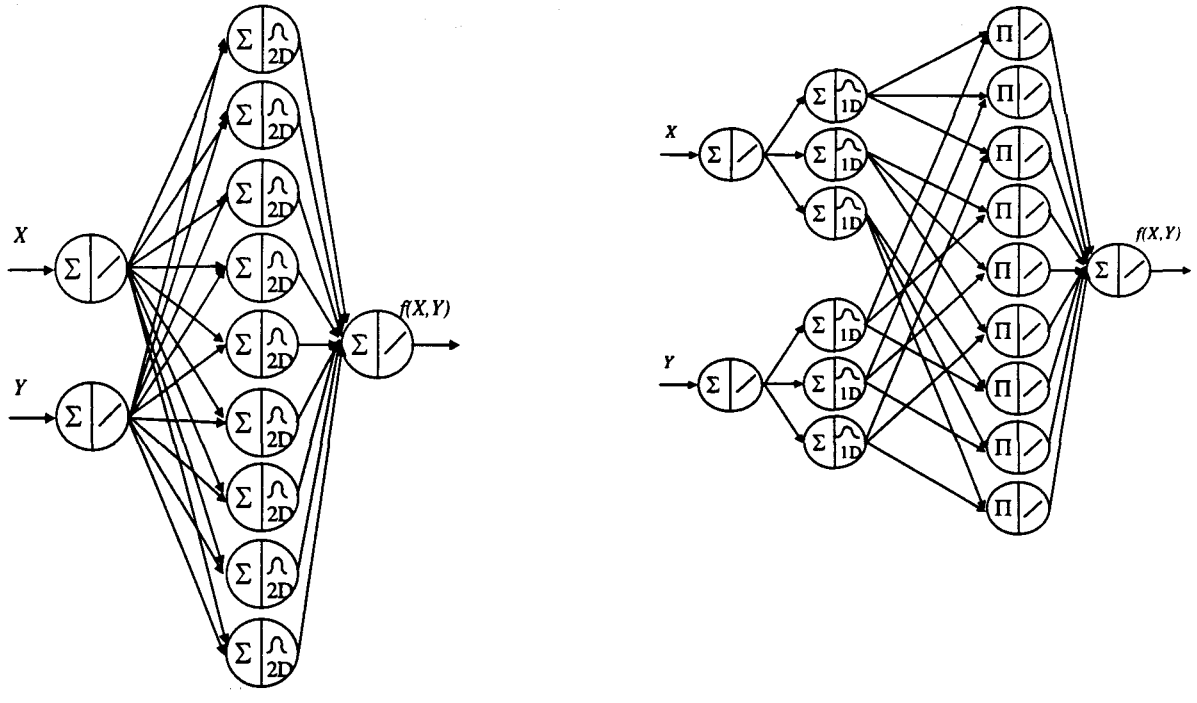


Fig III.5.11 Conséquence sur l'architecture de la propriété des densités marginales

Le bilan se transforme alors comme suit :

20 coefficients pour les poids d'entrée,
20 coefficients pour les biais d'entrée,
20 fonctions RBF unidimensionnelles,
100 produits à effectuer,
100 coefficients pour les poids de sortie.

Sachant que les fonctions non linéaires du type exponentielle sont issues de calculs itératifs à l'intérieur des processeurs, la propriété des fonctions marginales va réduire énormément les temps de calcul.

La figure III.5.12 montre les réponses temporelles à une excitation en tension sinusoïdale, dans le cas d'une modélisation unidimensionnelle (caractéristique L(i) cf § 5.1) et d'une modélisation bidimensionnelle, notée 2D, (cf § 5.2). Les résultats sont tout à fait comparables, néanmoins si le réseau récurrent offre l'avantage de modéliser le système complet et qu'il permet de prendre en compte la résistance sans avoir à en connaître la valeur, c'est au prix de calculs plus lourds et plus coûteux.

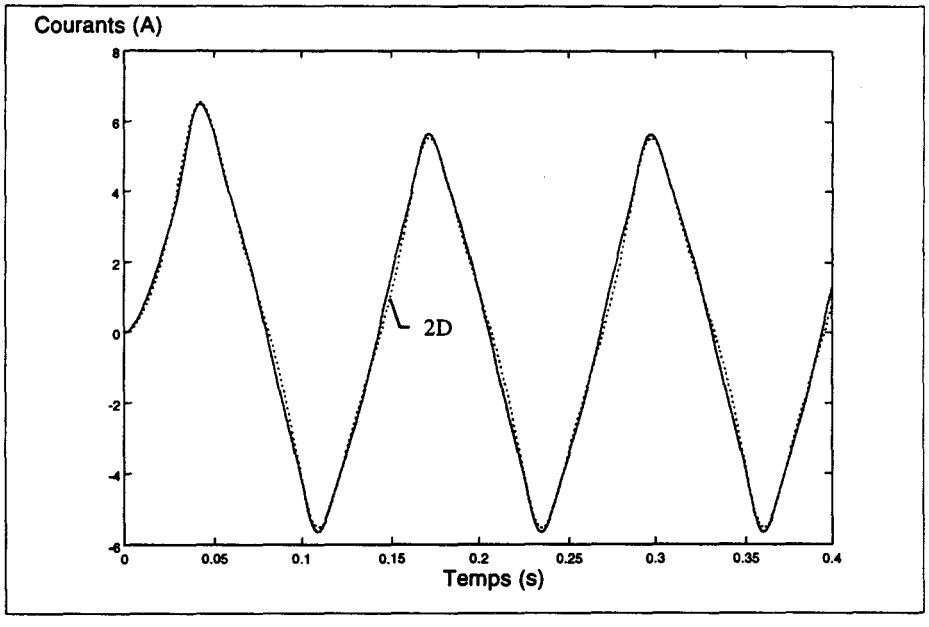


Fig III.5.12 Simulations d'une inductance saturable

6 Application 2 : Matériau magnétique, hystérésis

Dans ce paragraphe, nous allons nous intéresser à la modélisation des phénomènes non linéaires au sein des matériaux magnétiques.

6.1 Phénomène de saturation magnétique

Considérons tout d'abord le seul phénomène de saturation magnétique : la valeur du champ B à un instant donné est uniquement dépendante de celle de l'excitation magnétique H au même instant. Le modèle est alors unidimensionnel, il comporte une entrée excitation H et une sortie le champ B. Comme la caractéristique est non linéaire, nous utiliserons un réseau monocouche dont les fonctions d'activation sont non linéaires et continues. La figure III.6.1 montre la qualité de modélisation qu'il est possible d'obtenir grâce à la méthode d'initialisation. Les résultats présentés ici sont issus de mesures expérimentales effectuées sur un transformateur haute fréquence. Certes le modèle neuronal utilisé pour la modélisation de la caractéristique $B(H)$ est plus complexe que des modèles algébriques [Marrocco 77] dont les paramètres sont déterminés au moyen d'une identification au sens des moindres carrés; l'intérêt de cette modélisation neuronale est de montrer l'universalité de la méthode.

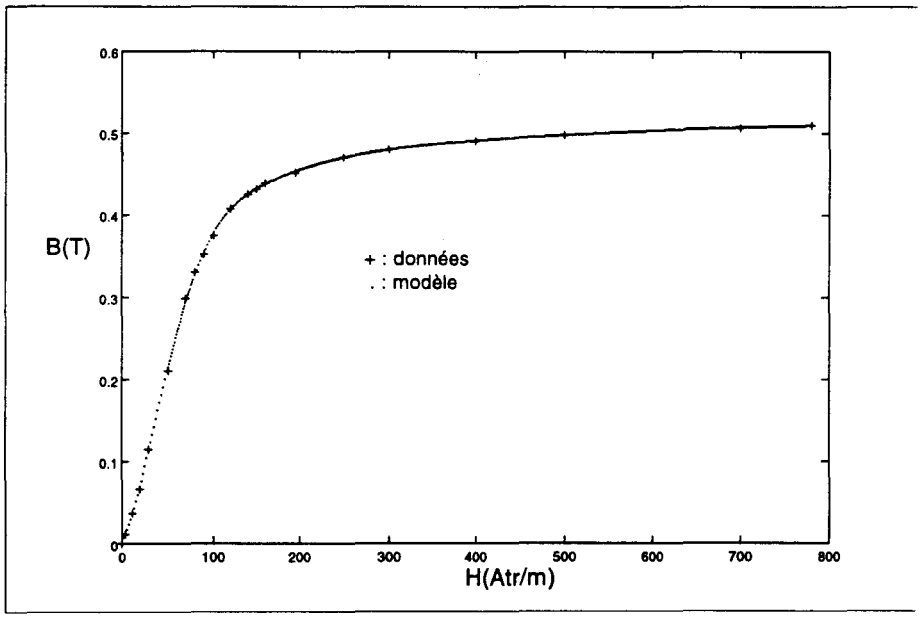


Fig III.6.1 : Caractéristique réelle et apprise B(H)

6.2 Phénomène d'hystérésis

Nous allons maintenant développer une méthode permettant la modélisation d'un cycle d'hystérésis majeur.

Plusieurs méthodes existent pour modéliser l'hystéresis magnétique : on peut citer la méthode de Preisach [TORRE 90]. Ces méthodes, généralement implantées dans des codes de calcul de champ, nécessitent un temps de résolution très important. Il est alors intéressant d'étudier dans ce sens la solution neuronale.

Contrairement à ce que nous avons considéré au paragraphe §5.1, le phénomène d'hystérésis se caractérise par un champ B dont le niveau évolue suivant la valeur et la dérivée de la valeur de l'excitation H. Si une telle caractéristique est apprise par le réseau neuronal à une seule entrée, on obtient évidemment la caractéristique de saturation moyenne (figure III.6.2). Cela montre bien que la relation $B(H)$ n'est pas univoque en ce qui concerne les cycles d'hystéresis et que, par conséquent, la structure neuronale choisie est mal adaptée à ce problème. Il est manifeste que la topologie du réseau doit prendre en compte une entrée supplémentaire afin d'intégrer l'information relative au sens de variation de l'excitation dans la fonction neuronale. Nous aurions ainsi deux entrées, l'une correspondant à H à un instant donné et l'autre, correspondant à H à l'instant précédent. Dans ce cas nous rencontrons trois inconvénients :

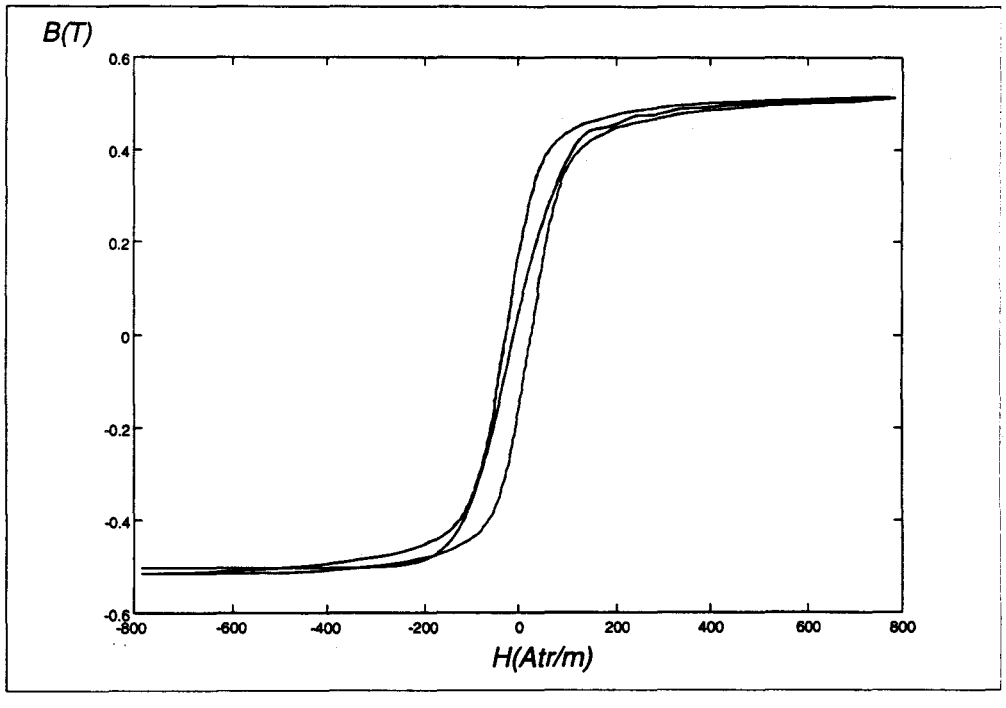


Fig III.6.2 : Cycles d'hystérésis réel et moyen(20 neurones σ=0.2 et sigmoïdes)

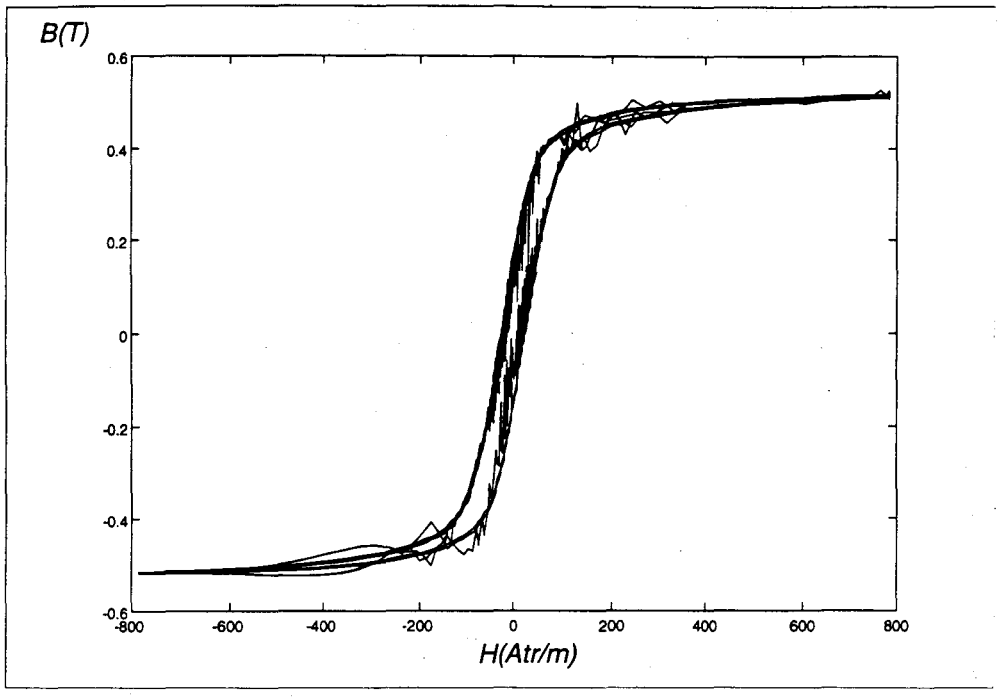


Fig III.6.3 : Cycles d'hystérésis réel et appris avec un réseau à deux entrées (10 neurones par dim σ=0.25 et sigmoïdes)

• le premier est que le réseau, s'il a deux entrées, va nécessiter une plus grande quantité de neurones, ce qui va nuire, soit à la qualité d'apprentissage, soit au temps de calcul.

• le second est un problème de statistique. En effet, la méthode d'initialisation n'est en fait qu'une simple régression linéaire appliquée sur une structure non linéaire. Dans le cas de l'hystérésis de la figure III.6.3 on remarque que les chemins aller et retour du cycle majeur sont très rapprochés. Lors du calcul des poids de sortie, la régression risque de 'confondre' l'aller et le retour avec du bruit éventuel de mesure.

• le troisième inconvénient est que la représentation d'un cycle d'hystérésis dans le trièdre (H_k,H_{k-1},B_k) dépend essentiellement de la fréquence de H. La figure III.6.4 schématise trois cycles d'hystérésis dans le plan (H_k,H_{k-1}) à des fréquences différentes. Il est manifeste que le réseau de neurones ayant appris cette caractéristique sera incapable d'extrapoler la sortie pour d'autres valeurs de la fréquence et pour d'autres valeurs maximales de l'excitation H. Certes, cette représentation permet de caractériser la saturation magnétique d'un matériau en fonction de la valeur maximale de H ainsi que de sa fréquence. Mais elle nécessite à la

fois de disposer de cycles d'hystérésis recueillis pour plusieurs fréquences, et ne permet de tenir compte que d'une seule valeur d'induction maximale.

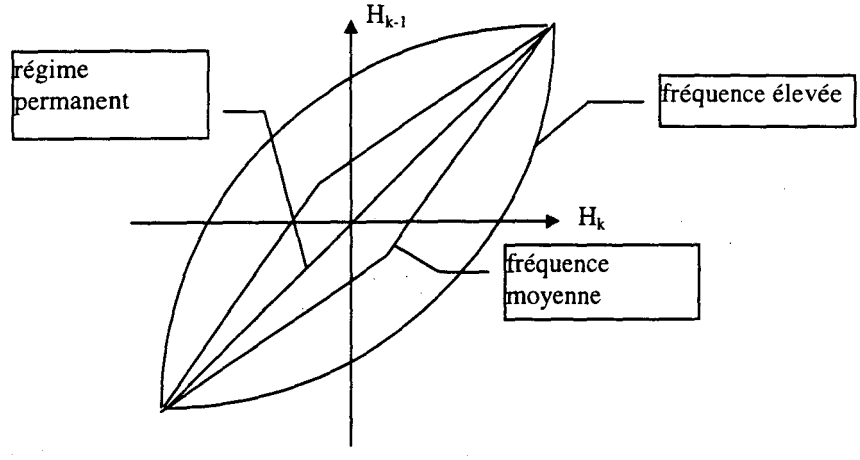


Fig III.6.4 : Représentation d'hystérésis dans le plan (H_k,H_{k-1}) en fonction de divers fréquences

Pour pallier ces inconvénients, une première méthode consiste à utiliser une structure comportant deux réseaux à une seule entrée; le premier sera chargé de reproduire la caractéristique magnétique croissante par exemple et le second la caractéristique décroissante. Dans ces conditions, la distinction des 'chemins' ne se fera plus de façon interne au réseau comme précédemment, mais de façon logique, à l'extérieur du réseau comme l'indique le schéma de la figure III.6.5. Si cette méthode est beaucoup moins générale que la première, elle a le mérite d'être efficace à en juger le résultat de la figure III.6.6.

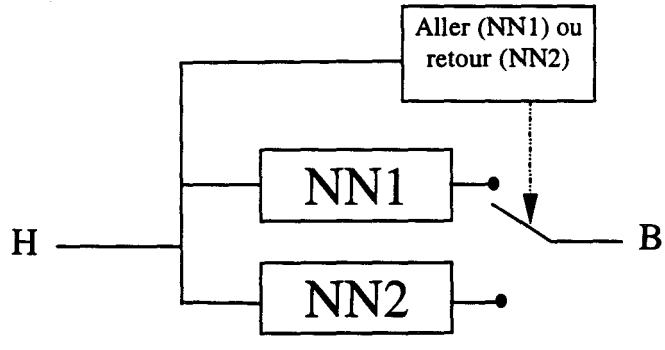


Fig III.6.5 : Reconstitution du cycle d'hystérésis à l'aide de deux réseaux SISO

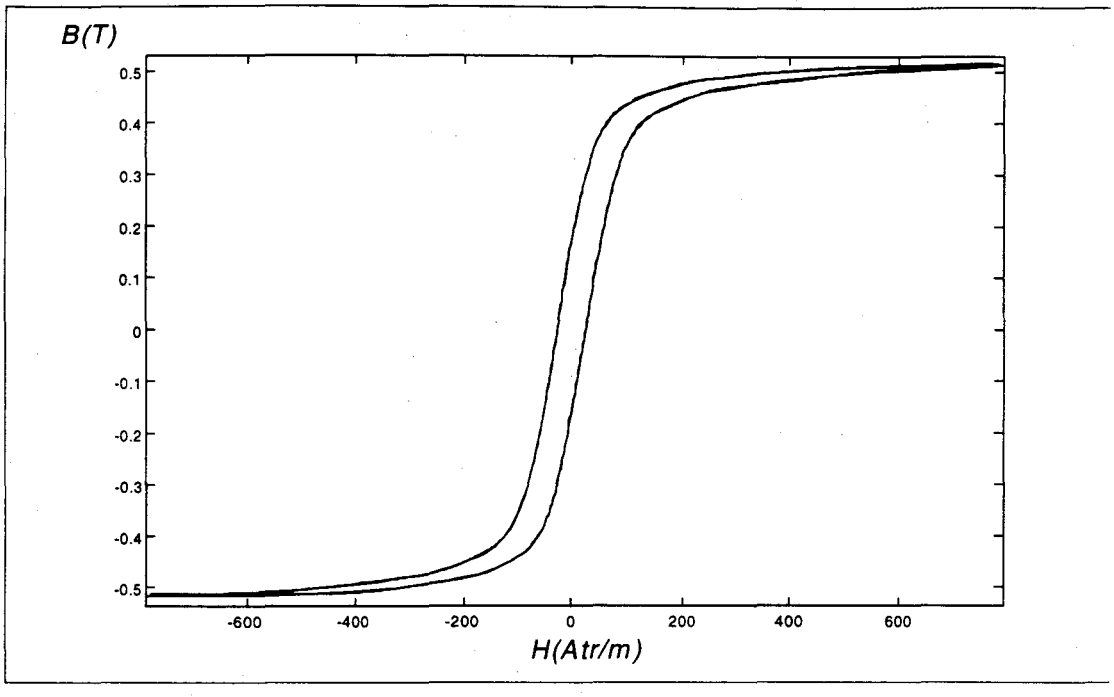


Fig III.6.6 : Cycles d'hystérésis réel et appris

Néanmoins, cette méthode ne permet pas de prendre en compte plusieurs valeurs maximales d'induction de cycles. Nous avons donc envisagé une autre représentation. Afin de différencier à la fois l'aller et le retour du cycle ainsi que les différentes valeurs maximales possibles de l'excitation magnétique, nous remplaçons la variable H_{k-1} par la variable $H_{max}signe(H_k-H_{k-1})$: en effet cette variable permet de véhiculer une information sur la valeur maximale de l'excitation ainsi que sur le parcours emprunté. Ainsi, on peut représenter une série de cycles majeurs d'excitations maximales différentes dans un trièdre $(H_k, H_{max}signe(H_k-H_{k-1}), B_k)$ comme l'illustre la figure III.6.7. Pour illustrer différents cycles majeurs nous avons utiliser le modèle de Preisach mis en place au L2EP et qui nous a permis d'obtenir un grand nombre d'échantillons [DEBRUYNE 96]

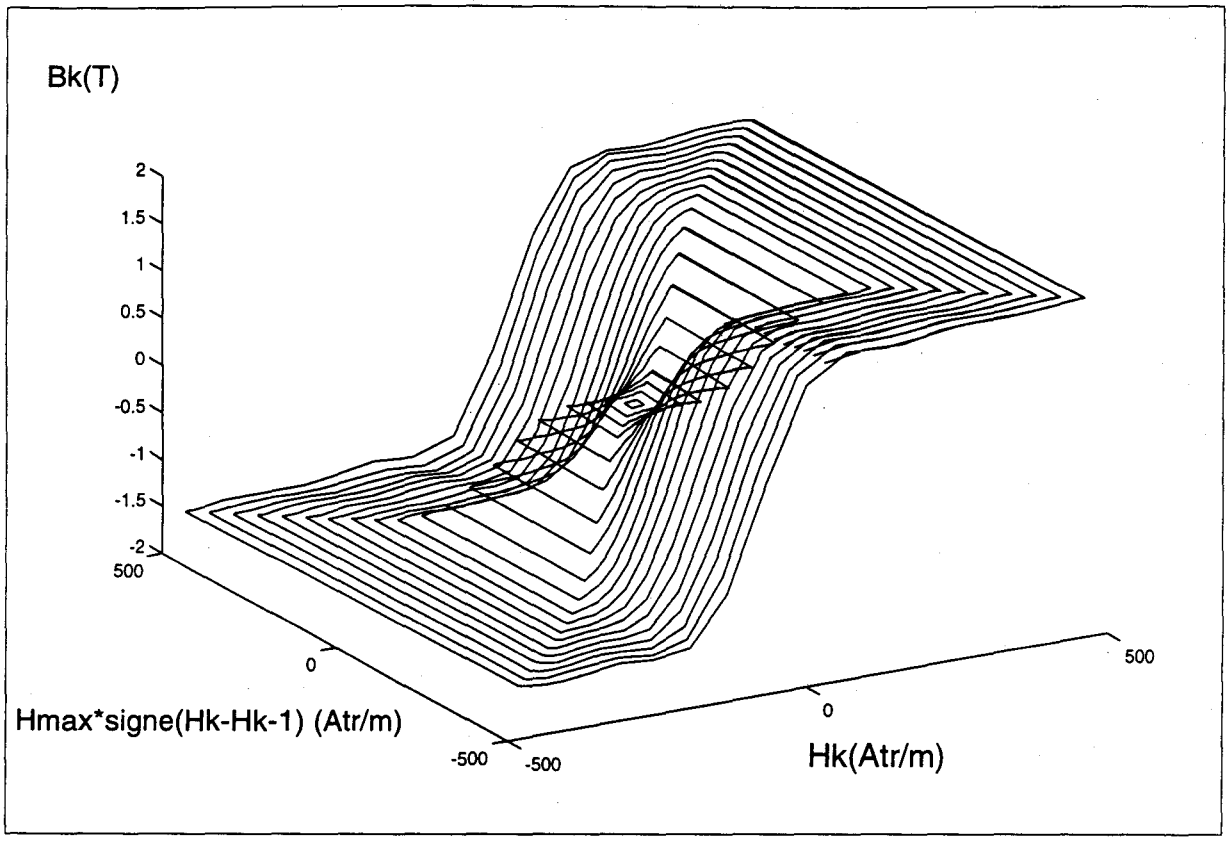


Fig III.6.7 : Représentation de cycles majeurs issus du modèle de Preisach et appris dans un espace permettant l'univocité

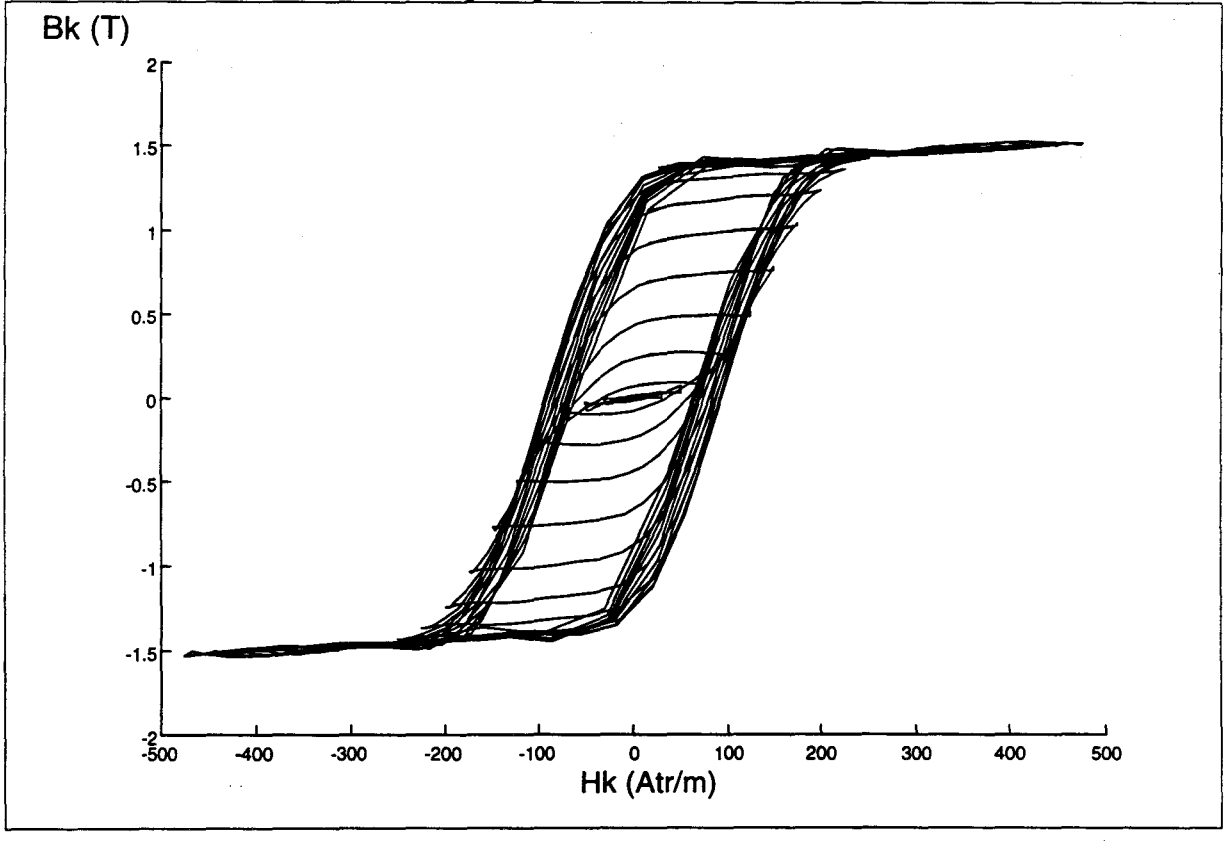


Fig III.6.8 : Projection de la représentation précédente dans le plan B(H)

Nous avons extrait de ces courbes un des cycles pour montrer la qualité d'apprentissage de cette méthode de modélisation (fig III.6.9). Les échantillons qui étaient à apprendre sont représentés par des croix, la ligne continue représente l'interprétation neuronale obtenue à partir de ces échantillons. La qualité d'interpolation est excellente, ce qui paraît normal puisque le réseau travaille dans son domaine d'apprentissage. Il faut alors vérifier que l'extrapolation à la fois au sein du domaine et en dehors de ce dernier reste correcte. La figure III.6.10 illustre deux types de résultats : une interpolation au sein du domaine d'échantillons, et une estrapolation en dehors. On s'aperçoit alors que le modèle neuronal est à même d'extrapoler avec une précision relativement bonne, à condition que l'estimation se fasse à proximité de la zone d'échantillonnage, c'est à dire de zone où l'on dispose de connaissances.

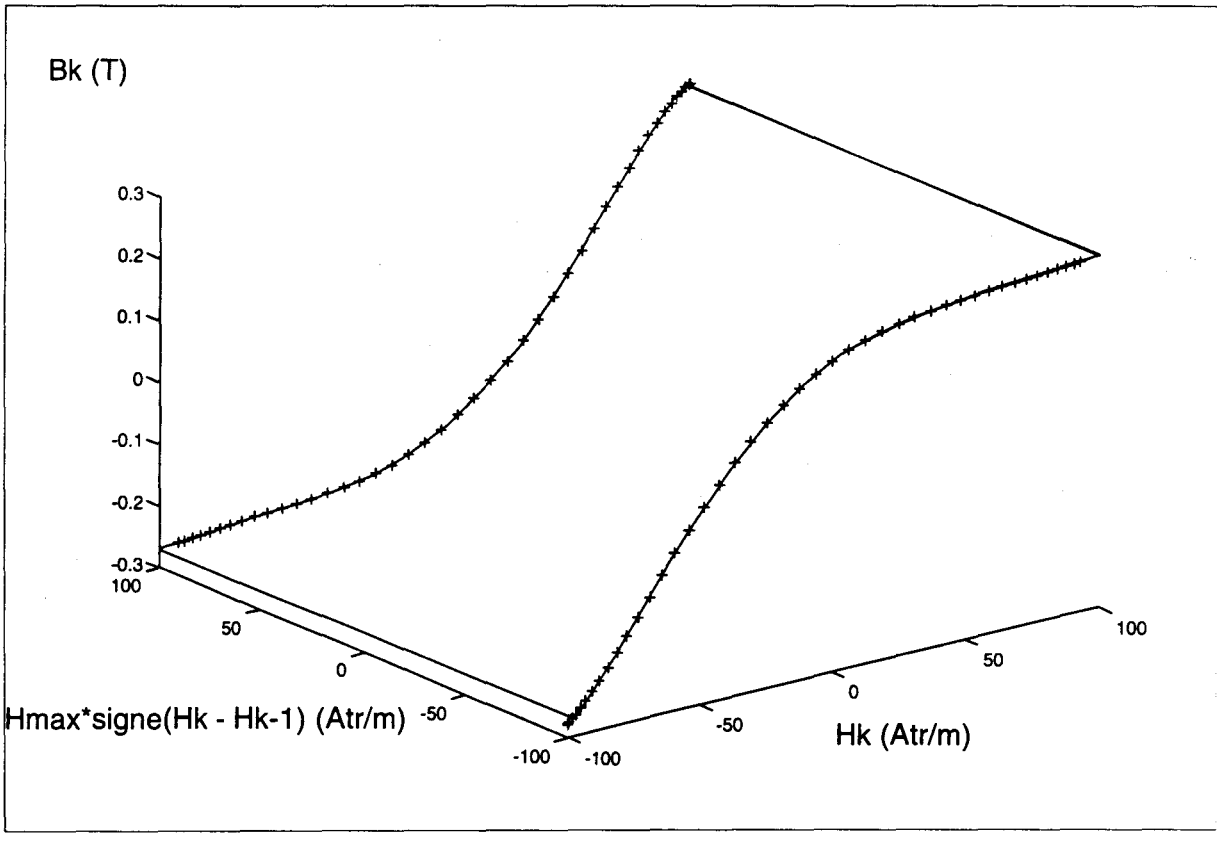


Fig III.6.9 : Hystérésis estimé en fonction d'un cycle appris

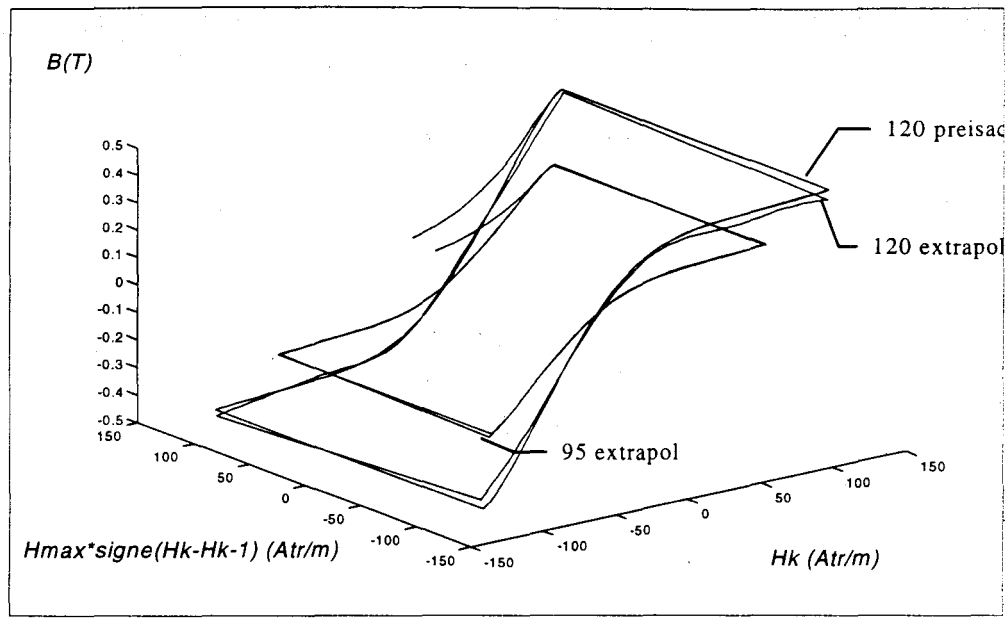


Fig III.6.10 : Extrapolation d'hystérésis

Nous avons testé cette technique de modélisation sur un cycle d'hystérésis issu de mesures expérimentales afin de prouver la validité de notre théorie sur l'apprentissage de cycles d'hystérésis. On constate d'après la figure III.6.11 que le modèle neuronal obtenu est très proche du cycle réel, ce qui prouve l'efficacité de notre technique d'apprentissage.

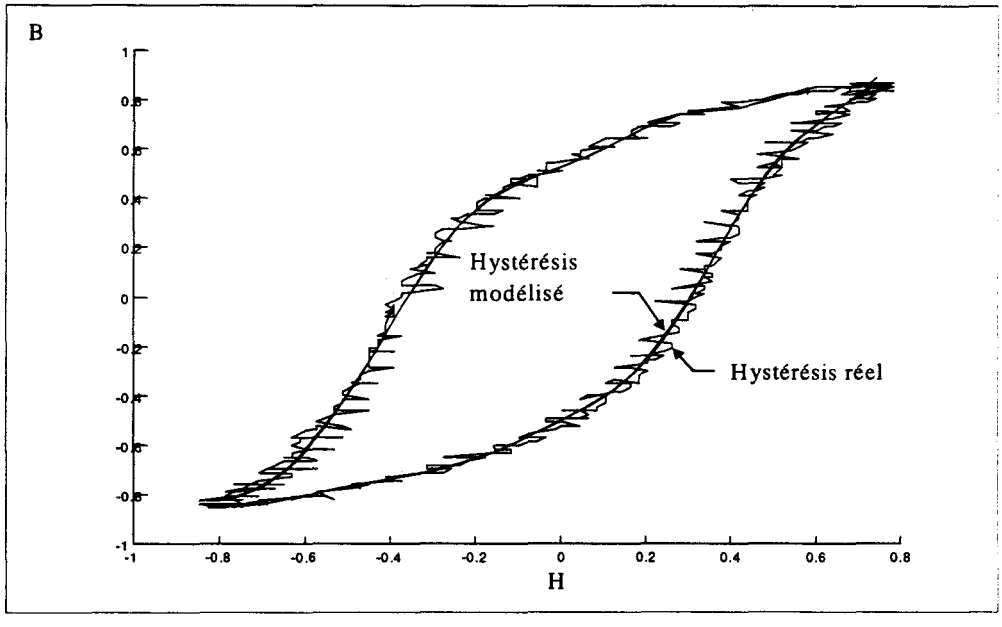


Fig III.6.11 : Modélisation d'un cycle d'hystérésis réel normalisé

7 Application 3 : Couples de charge mécanique

Nous allons nous intéresser dans ce paragraphe à la modélisation de systèmes mécaniques en rotation ou en translation, accouplés à des arbres moteurs. Les modèles sont élaborés essentiellement en vue d'établir directement les lois de commande.

7.1 Couples de charge linéaire

Dans un premier temps, nous allons nous intéresser aux systèmes linéaires et montrer que la modélisation neuronale dédiée au non linéaire fonctionne a fortiori pour ce genre de système.

7.1.1 Système d'ordre un

Pour illustrer ce passage nous modélisons la charge naturelle que constitue le rotor de la machine tournante, composée d'une inertie de moment J et de frottements visqueux de coefficient f. Dans cette partie nous négligerons volontairement les frottements secs qui rendent le système non linéaire ; ce cas sera évoqué dans le paragraphe 7.2. Le système étudié peut donc être schématisé de la façon suivante (figure III.7.1) avec C_{em} le couple électromoteur, Ω la vitesse.

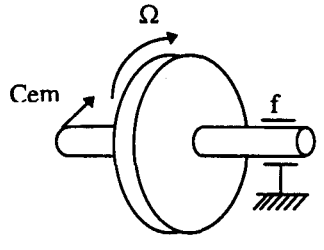


Fig III.7.1 : Couple de charge comportant une inertie et des frottements

Le bilan des forces est régit par loi physique :

$$J\dot{\Omega} + f\Omega = Cem \tag{3.17}$$

soit après discrétisation :

$$J\left(\frac{\Omega_k - \Omega_{k-1}}{Te}\right) + f\Omega_k = Cem_k \tag{3.18}$$

De cette équation, on peut extraire un modèle soit pour la commande, soit pour la modélisation du processus (cf paragraphe III.3.1).

$$Cem_k = f1_{linéaire}(\Omega_k) + f1_{linéaire}(\Omega_{k-1}) \tag{3.19}$$

La topologie du réseau est alors la suivante :

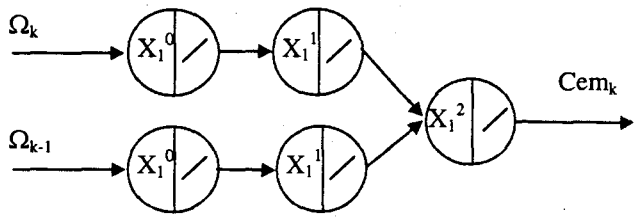


Fig III.7.2 : Structure neuronale permettant de déterminer le couple en fonction de la vitesse pour un système linéaire

Pour la modélisation du processus, l'équation (3.18) se transforme comme suit :

$$\Omega_k (f + \frac{J}{Te}) = Cem_k + \frac{J}{Te}\Omega_{k-1} \tag{3.20}$$

$$\Omega_k = g1_{linéaire}(Cem_k) + g2_{linéaire}(\Omega_{k-1}) \tag{3.21}$$

Le réseau employé est cette fois celui de la figure III.7.3.

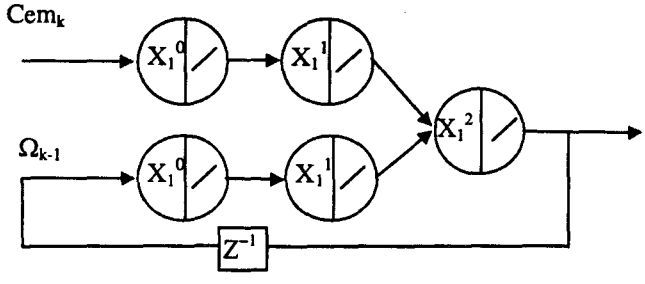


Fig III.7.3 Structure neuronale permettant de déterminer la vitesse en fonction du couple et de la vitesse précédente pour un système linéaire

7.1.2 Système d'ordre supérieur

Nous allons illustrer le cas d'un système d'entraînement avec conversion rotation translation, utilisé dans le cas d'un tour à usinage très grande vitesse (figure III.7.4).

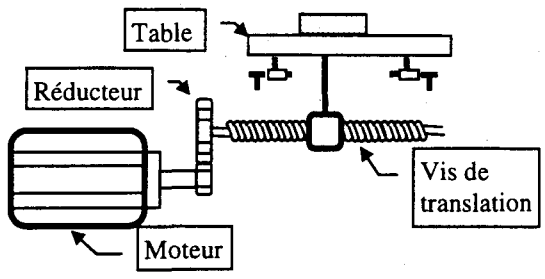


Fig III.7.4 : Schéma d'un tour à usinage

Ce dispositif peut être caractérisé par les paramètres du schéma de la figure III.7.5 :

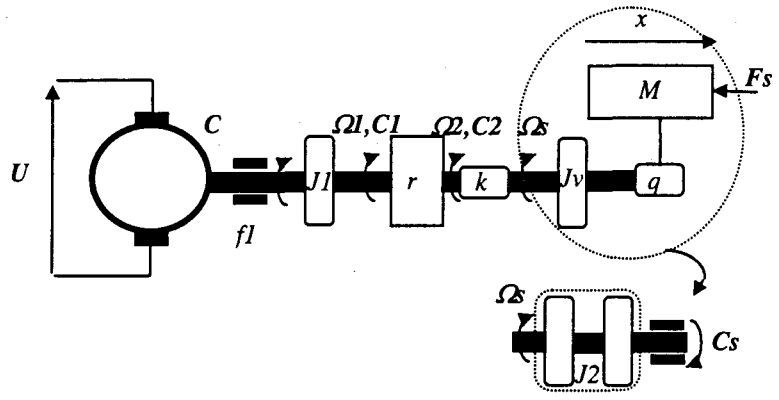


Fig III.7.5 : Représentation de l'axe de conversion rotation translation d'un tour à usinage

avec *J1, J2, Jv* les moments d'inertie, *f1, f2* les coefficients de frottements, *r* le coefficient de réduction, *k* le coefficient de raideur, *U* la tension d'alimentation, *C* le couple électromoteur, *C1, C2* les couples de transmission, *M* la masse de la pièce à usiner.

Cette fois la fonction de transfert mécanique est d'ordre trois, elle est établie à partir des équations (3.22).

$$\begin{cases} (J_1 s + f_1)\Omega_1 = c - c_1 \\ \Omega_2 = r\Omega_1 \\ c_1 = rc_2 \\ \frac{1}{k} pc_2 = \Omega_2 - \Omega_s \\ (J_2 s + f_2)\Omega_2 = c_2 - c_s \end{cases} \tag{3.22}$$

le couple électromoteur *c* s'exprime alors :

$$c = \Omega_s\left(rf_2 + \frac{1}{r}f_1\right) + \dot{\Omega}_s\left(rJ_2 + \frac{1}{r}J_1 + \frac{1}{rk}f_1 f_2\right) + \ddot{\Omega}_s\left(\frac{1}{r}J_1 f_2\right) + \dddot{\Omega}_s \frac{1}{rk}J_1 J_2 + rc_s + \frac{1}{rk}(J_1 p + f_1)\left(\frac{1}{k}pc_s\right)$$

Pour simplifier, nous écrirons cette équation sous la forme :

$$c = a\Omega_s + b\dot{\Omega}_s + c\ddot{\Omega}_s + d\dddot{\Omega}_s \tag{3.24}$$

où a,b,c et d sont des coefficients constants.

Ensuite l'équation (3.24) est discretisée puisque les dérivées successives de la vitesse ne sont pas mesurables directement. La méthode d'Euler est bien adaptée à cette transformation et reste simple à mettre en oeuvre. Il faut alors considérer les dérivées en tant que variations.

$$\frac{d^n}{dt^n} \Leftrightarrow \left(\frac{Z^0 - Z^{-1}}{Te}\right)^n \tag{3.25}$$

où Te est la période d'échantillonnage.

Par conséquent le couple discret devient :

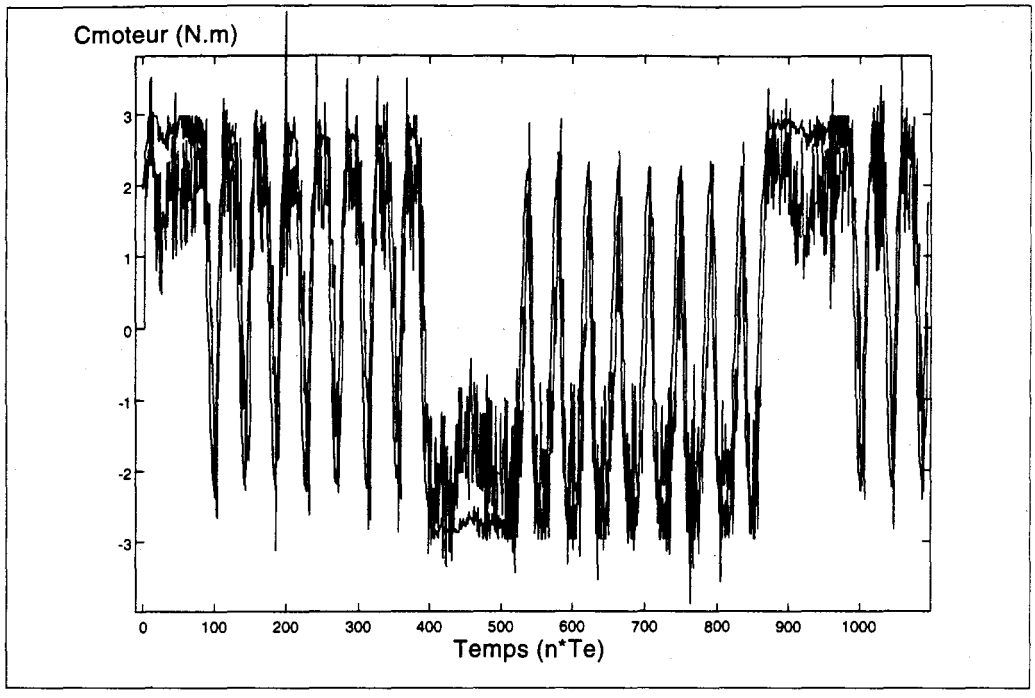
$$c_k = f(\Omega_k) + g(\Omega_{k-1}) + h(\Omega_{k-2}) + k(\Omega_{k-3}) \tag{3.26}$$

Si l'on désire modéliser le processus, l'équation utilisée sera la suivante :

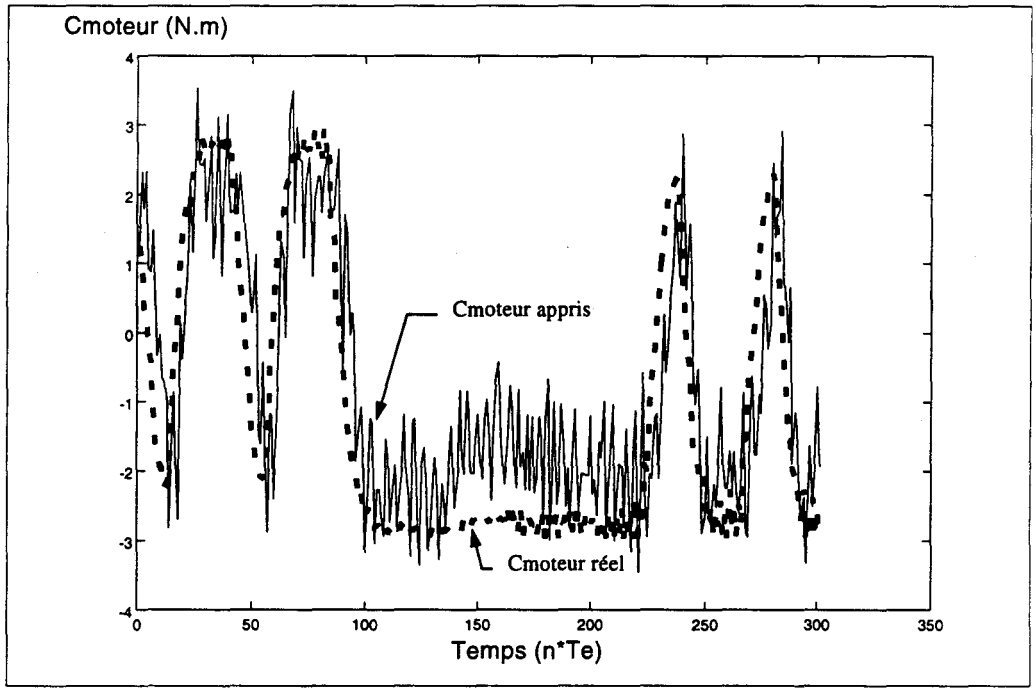
$$\Omega_k = \Phi(c_k) + \Gamma(\Omega_{k-1}) + \mathrm{H}(\Omega_{k-2}) + \Lambda(\Omega_{k-3}) \tag{3.27}$$

Les figures ci-dessous (III.7.6 a et b) montrent le résultat d'apprentissage du profil de couple en fonction du temps, correspondant à une succession de réponses indicielles en vitesse pour ce type de charge mécanique. Les résultats présentés sont expérimentaux : les échantillons proviennent du banc expérimental de la figure III.7.4 et correspondent à des aller retour du chariot.

La topologie de ces réseaux est comparable à celle des réseaux linéaires utilisés plus hauts mais avec une couche cachée un peu plus large.



a)



b)

Fig III.7.6 : Apprentissage à partir d'échantillons expérimentaux d'un couple de charge (dicrétisation par la méthode d'euler)

On s'aperçoit que certaines portions de la trajectoire sont très mal apprises. La première idée qui vient à l'esprit est de dire que le modèle paramétrique (3.22) ne correspond pas tout à fait à la réalité. Ces imperfections du modèle de connaissance engendreraient alors

des erreurs. Plus probablement ces erreurs proviennent de la méthode de discrétisation. En effet, la méthode d'Euler est correcte tant que la période d'échantillonnage reste petite. Néanmoins, si la numérisation doit s'opérer sur une dérivée énième, alors dans ce cas le calcul de la variation s'étend sur n périodes, et l'approximation d'Euler devient discutable. C'est pourquoi une autre méthode de numérisation doit être employée, on pourra utiliser par exemple l'approximation de TUSTIN.

Celle ci définit le changement de variable suivant dont la démonstration est rappelée en annexe :

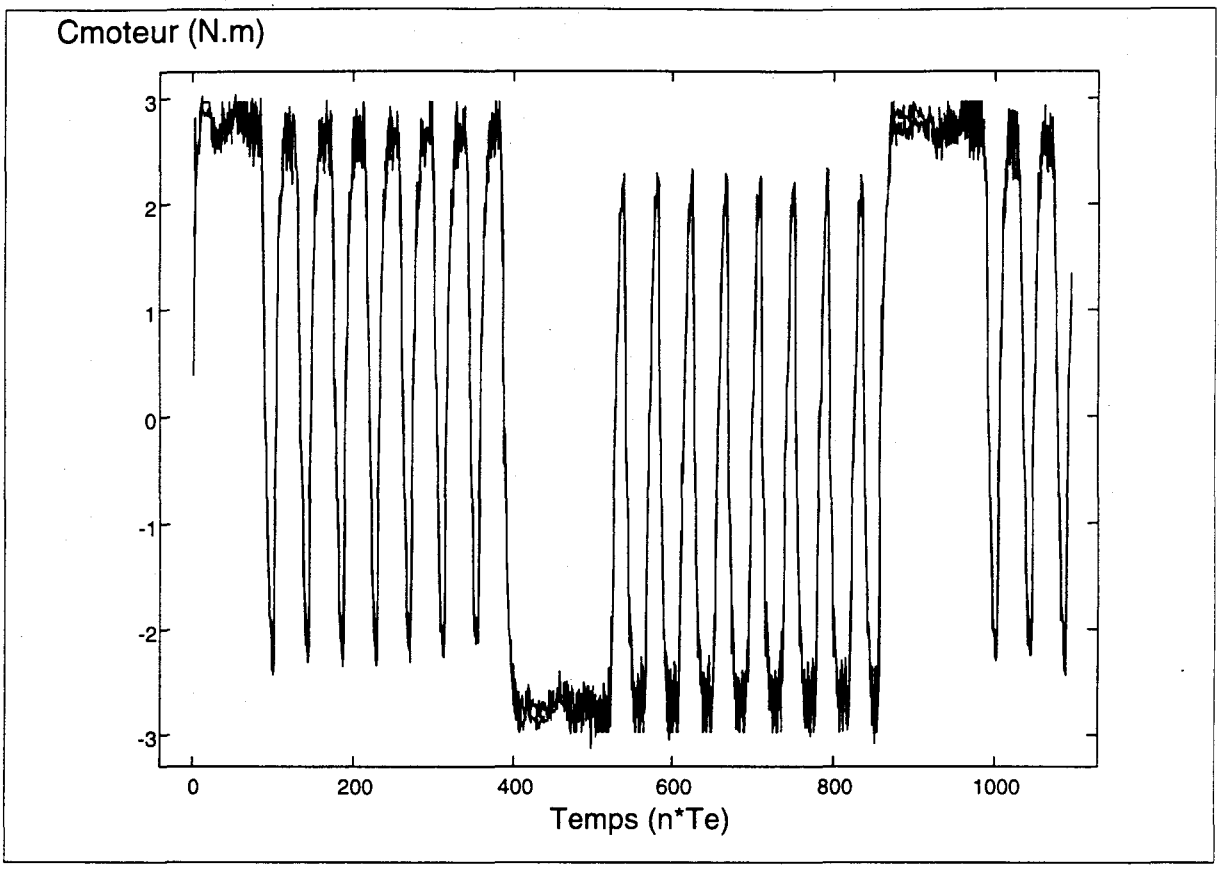
$$z = \frac{1+\frac{Te}{2}s}{1-\frac{Te}{2}s} \text{ soit } s = \frac{2}{Te}\frac{z-1}{z+1} \tag{3.28}$$

où s est la variable de Laplace, z l'opérateur retard, et Te la période d'échantillonnage.

Le modèle de connaissance discret devient alors :

$$\Phi(c_k) + \Gamma(c_{k-1}) + \mathrm{H}(c_{k-2}) + \Lambda(c_{k-3}) = f(\Omega_k) + g(\Omega_{k-1}) + h(\Omega_{k-2}) + k(\Omega_{k-3}) \tag{3.29}$$

La figure III.7.8 montre qu'avec ce modèle l'approche est beaucoup plus satisfaisante. L'enseignement à retenir de cet exemple est qu'il faut impérativement tenir compte de l'ordre du système.



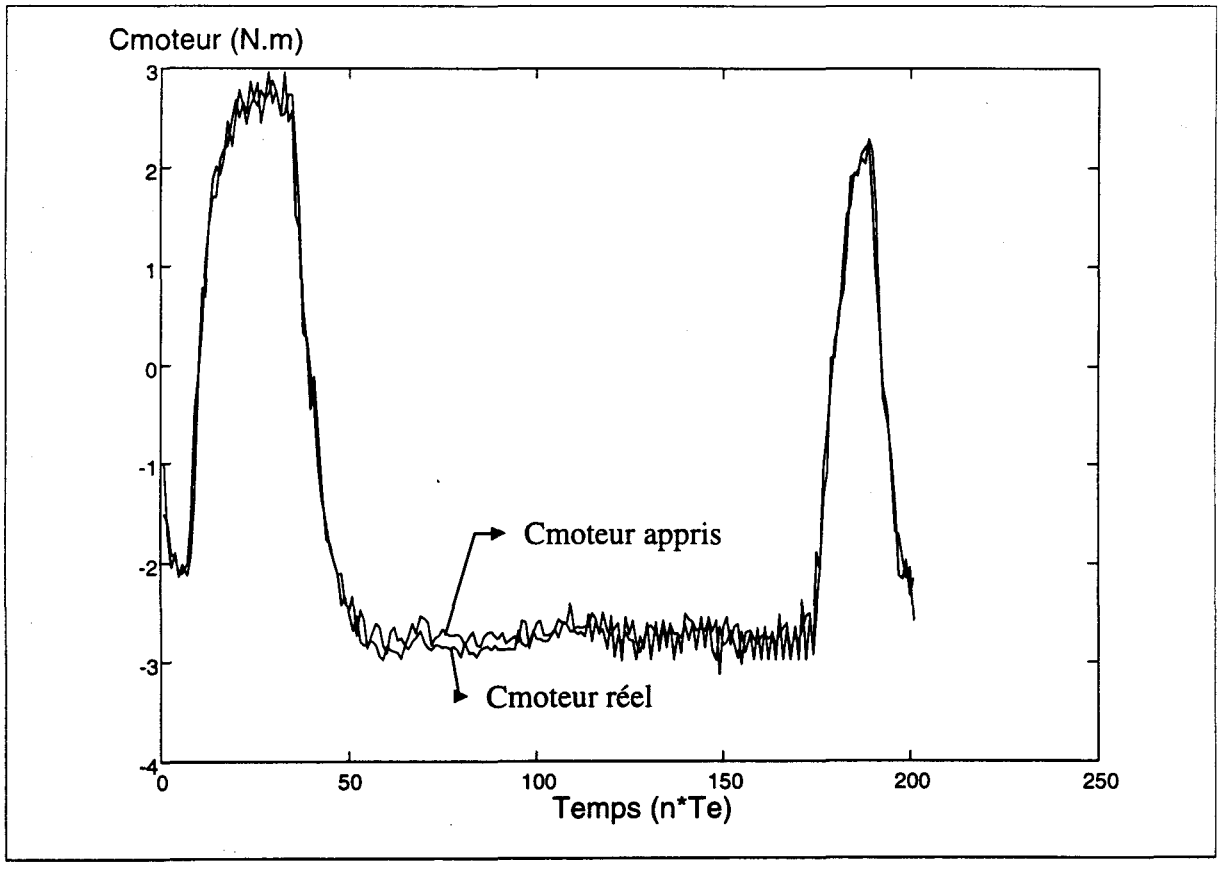


Fig III.7.8 : Apprentissage à partir d'échantillons expérimentaux d'un couple de charge (dicrétisation par la méthode des trapèzes)

7.2 Couple de charge non linéaire d'ordre un

Dans ce paragraphe, nous allons nous intéresser, au cas d'une charge rotative, non linéaire, d'ordre un, accouplée à une machine électrique. Nous présenterons des résultats de modélisation d'une charge de type ventilateur issue de simulations, puis des modélisations de charges réelles, obtenues grâce à un banc de tests. Tous les modèles sont prévus pour être utilisés directement dans l'axe de commande.

7.2.1 Définition des grandeurs d'états

Afin de déterminer quelles sont les grandeurs qui vont intervenir dans la modélisation d'une charge mécanique non linéaire, on se reporte, comme dans le cas de charges linéaires, aux lois de la mécanique et au bilan des forces :

$$J\dot{\Omega} = -f\Omega - C_r + C_{em} \tag{3.30}$$

avec J moment d'inertie du moteur, f coefficient de frottements du moteur, C_{em} couple électromoteur et C_r le couple résistant du type

$$C_r = C_0 signe(\Omega) + F\left(\Omega^2\right) \tag{3.31}$$

l'équation (3.30) devient donc après discrêtisation par la méthode d'Euler :

$$\boxed{C_{em_k} = \left(\frac{J}{T_e} + f\right)\Omega_k - \frac{J}{T_e}\Omega_{k-1} + Cr_k} \tag{3.32}$$

Ainsi, le modèle du couple électromoteur est de la forme :

$$C_{em} = h\left(\Omega_k, \Omega_{k-1}\right) \tag{3.33}$$

ou

$$C_{em} = g\left(\Omega_k, \frac{\Omega_k - \Omega_{k-1}}{Te}\right) \tag{3.34}$$

7.2.2 Topologie du réseau

Nous pouvons voir d'après les équations (3.33 et 3.34) que les entrées de ce réseau sont la vitesse et l'accélération. Afin d'alléger la structure, il est bon de remarquer que la fonction qui relie le couple électromoteur aux grandeurs précédemment citées est composée d'une fonction non linéaire sur la vitesse et d'une fonction linéaire sur l'accélération. Par conséquent la topologie restreinte est celle de la figure III.7.9. On remarque que des fonctions d'activation à seuil ont été rajoutées volontairement dans un souci de qualité d'apprentissage des frottements secs, car aucune autre fonction d'activation usuelle ne possède des propriétés permettant d'approcher au mieux cette discontinuité.

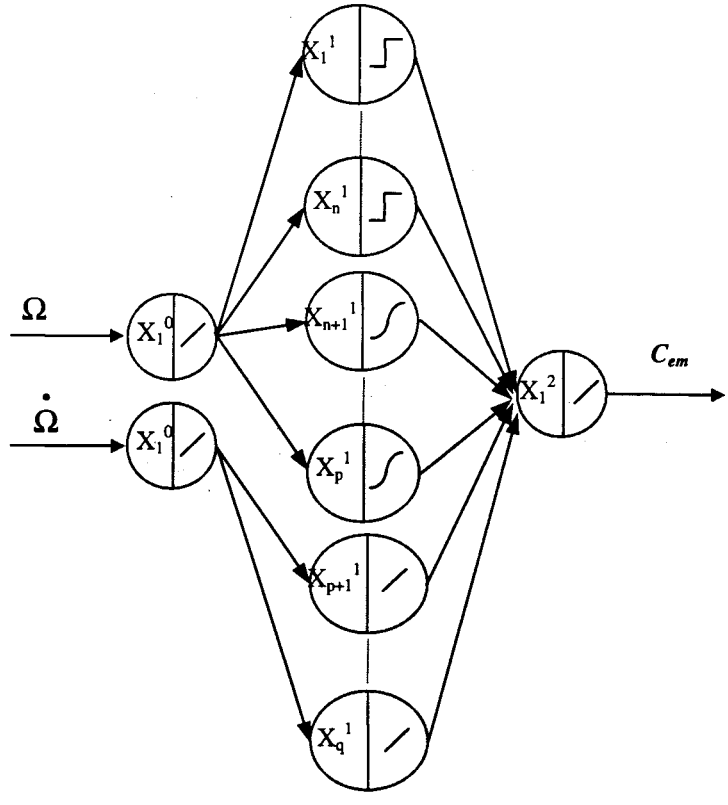


Fig III.7.9 : Structure retenue pour la modélisation du couple de charge non linéaire équation (3.32)

Certes, cette structure est très spécialisée sur une morphologie donnée mais présente l'avantage de ne pas être trop gourmande en temps de calcul. Si la charge n'était absolument pas connue, alors on pourrait utiliser une structure comprenant des fonctions d'activation discontinues, non linéaires et linéaires connectées à la fois sur la vitesse et l'accélération.
La figure III.7.10 montre la qualité d'apprentissage qui résulte de la méthode d'initialisation. On remarque sur cette coupe que les erreurs d'apprentissage sont minimes. La discontinuité

est parfaitement bien modélisée. Néanmoins, on peut s'apercevoir que l'apprentissage ne reste valable que dans la zone qui a été échantillonnée. Au delà, l'extrapolation neuronale diverge de la courbe réelle.

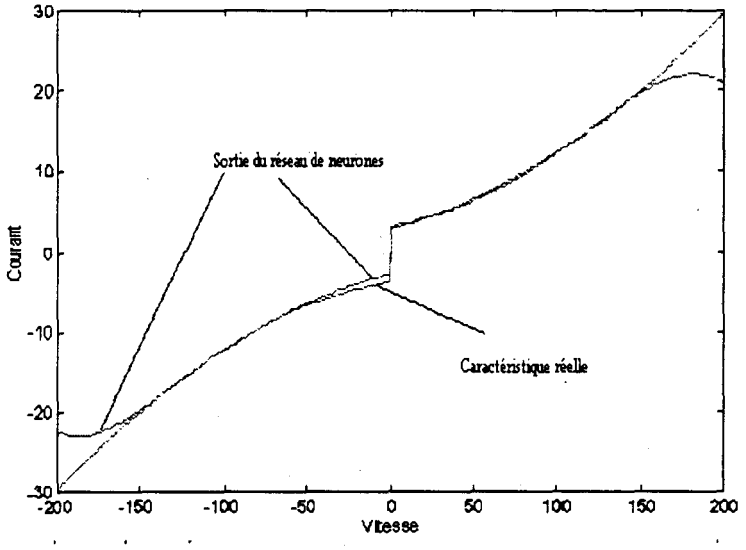


Fig III.7.10 : Résultat d'apprentissage avec la méthode d'initialisation

Nous avons également testé notre méthode de modélisation sur une charge mécanique réelle fortement discontinue. La figure III.7.11.b montre le résultat de modélisation neuronale obtenu à partir des échantillons présentés figure III.7.11.a. L'efficacité de cette méthode est d'autant plus grande que pour un résultat de cette qualité, l'apprentissage n'a duré que quelques secondes en utilisant le langage Matlab non compilé.

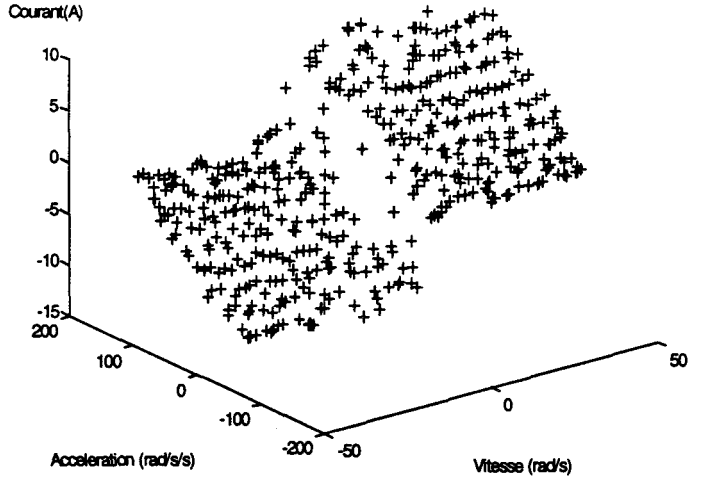


Fig III.7.11.a : Echantillonnage d'une caractéristique mécanique réelle fortement discontinue

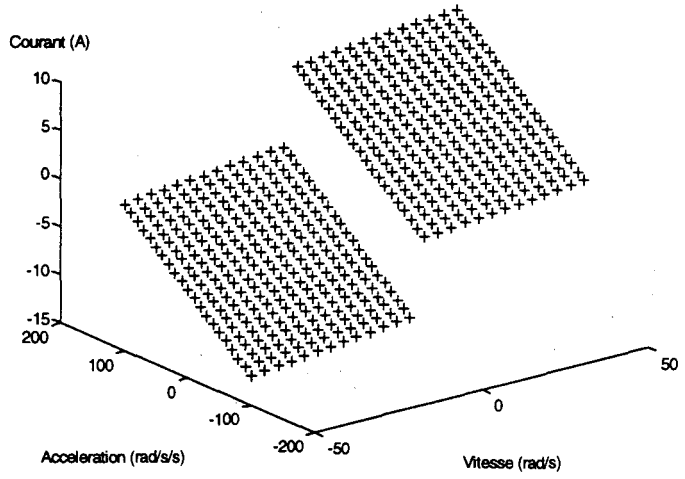


Fig III.7.11.b : Modèle d'une caractéristique mécanique réelle fortement discontinue

8 Conclusion

Nous avons vu au cours de ce chapitre les performances de modélisation des réseaux de neurones appliqués à la caractérisation de certains phénomènes électromagnétiques et électromécaniques. La qualité de l'approche neuronale laisse envisager des perspectives intéressantes, notamment en ce qui concerne la modélisation des cycles d'hystérésis en vue d'une implantation dans les codes de calculs par éléments finis pour l'évaluation des pertes dans les machines électriques. L'avantage de ces modèles par rapport au modèle de Preisach se situe au niveau du temps de calcul puisque le modèle neuronal de cycles d'hystérésis n'est pas fondé sur des itérations. D'autre part, le formalisme de l'approche neuronale rend plus convivial et surtout systématique la caractérisation des phénomènes de saturation magnétique au sein des machines. L'application de l'approche neuronale au couple de charge mécanique laisse également envisager de belles perspectives en ce qui concerne l'utilisation des modèles neuronaux en tant que correcteur. Néanmoins, nous insistons sur le fait que la qualité de modélisation reste intimement liée au choix de la structure du modèle de connaissance sur lequel est calquée l'architecture neuronale. Cette remarque conforte l'idée selon laquelle la qualité et les temps d'apprentissage doivent être optimisés en fonction de la connaissance dont on dispose sur les caractéristiques à modéliser.

A partir de ces modèles compétitifs, nous allons maintenant examiner comment les utiliser en tant que tel dans la commande des systèmes électromagnétiques et électromécaniques.

Chapitre 4

Commandes neuronales

1 Introduction

Nous avons vu aux chapitres précédents comment obtenir un modèle neuronal à partir d'échantillons et de la connaissance a priori d'un système. Ce quatrième chapitre a pour but de développer une technique permettant d'utiliser directement le modèle dans la commande du système. Pour ceci, il est important de connaître la manière dont est mis en oeuvre le modèle approché du processus et pour en avoir une idée, il est intéressant de s'inspirer des commandes classiques, largement validées et d'y intégrer les réseaux de neurones en temps qu'outils d'aide à la commande non linéaire. C'est pourquoi en rappelant les principes de base de différentes commandes, nous nous appliquons à expliquer l'intérêt ou les inconvénients inhérents à chacune d'elles, en considérant notamment les problèmes de faisabilité et de stabilité rencontrés.

Dans un premier temps, nous rappelons quels sont les intérêts d'une commande neuronale ainsi que ses limitations. Puis, nous développerons les commandes par correction, anticipation ou compensation. Nous exposons une méthode d'adaptation en ligne permettant de modifier les paramètres du correcteur neuronal pour pallier les variations éventuelles du processus ou corriger des erreurs liées à la modélisation.

2 Utilisation des neurones en commande

2.1 Regain d'intérêt mitigé

L'utilisation des réseaux neuronaux en commande de processus induit des contraintes en termes de temps de calcul. Les progrès réalisés dans la rapidité des processeurs dédiés aux calculs intensifs permettent, à l'heure actuelle, l'implantation des réseaux neuronaux dans des systèmes temps réel de traitement d'images ou d'informations. Le prix des DSP (Digital Signal Processor) en constante diminution rend attractif l'utilisation de plus en plus prononcée de l'intelligence artificielle dans la commande de processus sophistiqués. Cependant,

contrairement aux algorithmes de la logique floue facilement intégrables sur circuits spécialisés, ceux des réseaux neuronaux ne font pas encore l'objet d'un large développement de circuits intégrés. Certes, certains constructeurs de circuits intégrés commence à développer des circuits spécialisés dans le cadre des mémoires associatives, mais cette tendance est supplantée par la logique floue beaucoup plus facilement intégrable. En fait, la non standardisation des problèmes à traiter met un frein à l'intégration des réseaux neuronaux. En effet, la technique neuronale consiste en général à utiliser un grand nombre de neurones à fonctions d'activation non linéaires, puis à appliquer un algorithme d'apprentissage itératif. La quantité de neurones alors requise, nécessite un stockage mémoire des poids très important, ce qui technologiquement peut se réaliser actuellement, puisque l'on peut concevoir des mémoires de plusieurs mega octets sur une même puce. En revanche, le calcul des fonctions d'activation non linéaires complique la tâche puisqu'il nécessite soit autant d'unités arithmétiques que de neurones (trop sophistiqué), soit un multiplexage des neurones sur une seule unité arithmétique en contre partie d'un temps de réponse plus long. Pour ces raisons, les réseaux neuronaux ne sont pas encore disponibles sur le marché des circuits intégrés d'autant plus qu'il faudrait implanter l'algorithme d'apprentissage, ce qui accroîtrait davantage la complexité d'intégration.

2.2 Prise en compte des phénomènes mal connus

Bien que les systèmes électromécaniques soient généralement connus, il reste néanmoins parfois des indéterminations comportementales, soit faute de connaissances 'a priori' sur certains éléments du système, soit en cas de non stationnarité du processus lorsque les systèmes sont fortement non linéaires ou dépendant du temps. Néanmoins, on peut penser que les phénomènes mal connus agissent comme des perturbations sur les processus qu'on caractérise. Ceci nous permet de considérer que le processus devient fonction des perturbations et que celles ci sont intrinsèquement contenues dans des échantillons prélevés sur le système. Ainsi, une modélisation neuronale basée sur des échantillons doit permettre de prendre en compte des perturbations, pour peu que celles ci soient fonction des variables du modèle de connaissance.

3 Limitations

Nous avons vu dans le chapitre concernant la modélisation neuronale, que toute fonction pouvait être approchée. Néanmoins, l'implantation pratique des réseaux de neurones a mis en évidence quelques limitations quant à leur utilisation en commande.

3.1 Simplicité des structures

Pour une utilisation en temps réel au moyen d'un unique processeur, il est primordial de garder au réseau de neurones une taille modeste. Ceci pose une première limite dans l'utilisation de structures trop complexes en temps réel. Par conséquent, la commande neuronale de systèmes sera d'autant moins efficace que le modèle sera sophistiqué; d'où la nécessité de bien spécialiser l'architecture du réseau au problème à traiter.

De plus, nous avons vu que la validité du modèle est généralement réduite au domaine d'apprentissage. C'est pourquoi, une limitation du domaine d'application s'impose également. Comme la circonscription de ce domaine considéré en un contour est bien souvent non linéaire, l'apprentissage de ce dernier nécessite l'usage d'un second réseau neuronal. Donc, plus le modèle sera complexe et d'ordre élevé, plus le contour de sa circonscription le sera aussi.

3.2 Systèmes d'ordre élevé

L'utilisation des réseaux neuronaux en commande se heurte au problème de reconstitution des grandeurs d'état nécessaires à l'inversion de causalité à reproduire. En effet, toute commande étant basée sur le principe d'inversion des causalités, le principe de la commande neuronale consiste à utiliser les propriétés de modélisation des réseaux de neurones pour établir un modèle inverse du processus en vue de l'utiliser directement dans la commande. Il est nécessaire de reconstituer les dérivées successives de la sortie du processus ; celle de la dérivée première est possible à déterminer moyennant filtrage, ce n'est pas le cas pour les dérivées d'ordre supérieur qui sont assujetties au bruit de mesure, ce qui rend leur reconstitution particulièrement délicate. Dans le cas des systèmes non linéaires, comme le principe de superposition ne s'applique plus, il n'est pas non plus possible d'utiliser une correction, à moins bien sûr de considérer le système comme linéaire. Nous verrons dans les

paragraphes suivants qu'il est bien souvent possible de faire apparaître, au sein du G.I.C, les non linéarités comme des perturbations, et de les compenser plutôt que de les corriger.

4. Correction neuronale

Nous allons, dans un premier temps, expliquer comment en utilisant le modèle inverse d'un système linéaire, on peut effectuer une correction.

4.1 Processus à pôles et zéros stables

Pour un système linéaire représentable par une fonction de transfert, les racines du dénominateur et du numérateur sont respectivement désignées pôles et zéros. Lorsque les racines sont à parties réelles négatives ou positives, elles sont respectivement qualifiées de stables et d'instables.

Considérons un système linéaire stable exprimé par une fonction de transfert W de type ARMAX

$$W = \frac{\sum_{i=0}^{n} a_i s^i}{\sum_{j=0}^{m} b_j s^j} \qquad (4.1)$$

avec $m \geq n$

La technique classique pour asservir un tel système est la simplification des pôles par des zéros, des zéros par les pôles et le placement de pôles afin d'imposer le comportement et le temps de réponse du système asservi. Le schéma de contrôle est représenté par la figure IV.4.1.

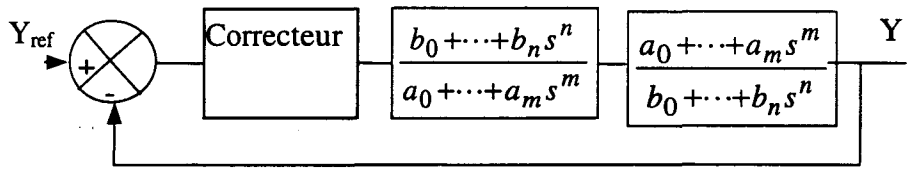


Fig IV.4.1 Correction linéaire

Dans le cadre d'une commande continue il est évident que la compensation des pôles et des zéros doit impérativement être englobée dans le placement de pôles pour éviter les problèmes de causalité inverse. En effet si n<m alors, le modèle inverse du processus est non causal. Par conséquent pour réaliser l'inversion du processus il faut augmenter artificiellement le degré du dénominateur du modèle inverse en profitant notamment du placement de pôles. Le schéma global devient alors le suivant :

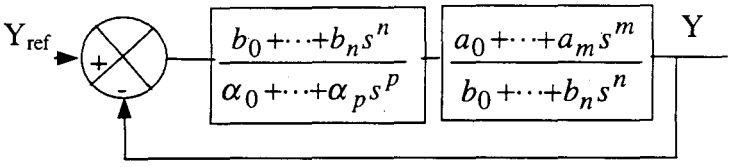


Fig IV.4.2 Correction linéaire (réalisable)

avec $p>n$.

Lorsque le processus dépasse l'ordre un, ce type de correcteur est implanté sur calculateur. Il est donc nécessaire de discrétiser le processus qui s'exprime de la façon suivante :

$$W(z)=\frac{Y(z)}{U(z)}=\frac{\sum_{i=0}^{n}\alpha_i z^i}{\sum_{j=0}^{m}\beta_j z^j} \tag{4.2}$$

De cette transmitance, on peut tirer une équation récurrente :

$$Y(z)\sum_{j=0}^{m}\beta_j z^j = U(z)\sum_{i=0}^{n}\alpha_i z^i \tag{4.3}$$

$$Y_k\beta_0 + Y_{k-1}\beta_1+\cdots+Y_{k-\mu}\beta_\mu = U_k\alpha_0 + U_{k-1}\alpha_1+\cdots+U_{k-\eta}\alpha_\eta \tag{4.4}$$

On en déduit la loi de commande suivante

$$U_k=\frac{1}{\alpha_0}(Y_k\beta_0 + Y_{k-1}\beta_1+\cdots+Y_{k-\mu}\beta_\mu - U_{k-1}\alpha_1-\cdots-U_{k-\eta}\alpha_\eta) \tag{4.5}$$

Par conséquent, si on désire utiliser un réseau de neurones pour décrire cette équation, l'architecture récurrente à topologie linéaire s'impose pour son apprentissage. Néanmoins, seul le nombre de variables ainsi que l'ordre des récurrences émanent de ce modèle de connaissance, par conséquent les paramètres α_i et β_i restent indéterminées. L'équation résultant de la phase d'apprentissage est donc la suivante :

$$U_k = Y_k NW2_{11} + Y_{k-1} NW2_{21} + \cdots + Y_{k-\mu} NW2_{\mu 1} - U_{k-1} NW2_{\mu+1,1} - \cdots - U_{k-\eta} NW2_{\mu+\eta,1} \quad (4.6)$$

conformément au réseau de la figure IV.4.3.

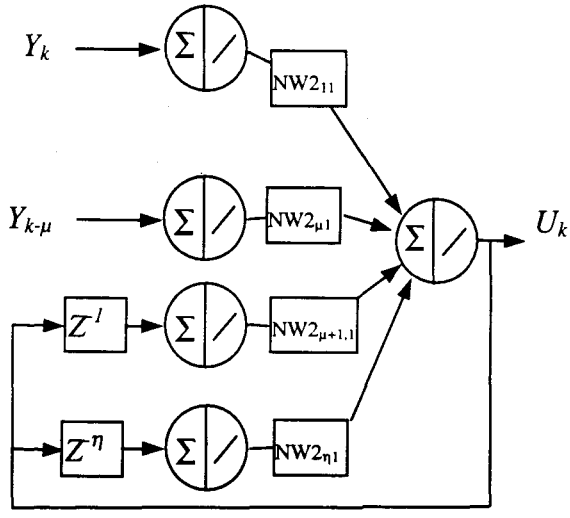


Fig IV.4.3 Structure neuronale utilisée pour la correction d'un système linéaire à pôles et zéros stables

On peut remarquer que la structure du modèle correspond exactement à celle du processus, et si les poids d'entrée sont unitaires, les poids de sortie *NW2* correspondent aux rapports $\frac{\beta_i}{\alpha_0}$.

Il est également possible d'apprendre les caractéristiques du correcteur usuel qui sert à la phase d'échantillonnage. Celui ci, même s'il est loin d'être parfait, donne la possibilité au correcteur neuronal d'asservir le processus comme un correcteur classique. Rien n'empêche par la suite d'adapter le réseau de neurones afin qu'il compense au mieux le processus. Avec cette technique, on peut se passer de la phase d'échantillonnage; la phase d'adaptation en ligne sera expliquée ultérieurement.

4.2 Processus à pôles et/ou zéros instables

Si le processus possède des zéros ou des pôles instables, il n'est pas possible de compenser les pôles par des zéros et vice versa. Par conséquent, on ne peut utiliser le modèle inverse directement dans la commande de sorte que le réseau neuronal soit mis en oeuvre lors d'une étape intermédiaire, pour identifier les paramètres de la fonction de transfert.

Considérons la fonction de transfert du processus :

$$W_{bo} = \frac{\sum_{i=0}^{m} a_i s^i}{\sum_{j=0}^{n} b_j s^j} \tag{4.7}$$

Si le système est à pôles et/ou zéros instables, on suppose qu'il existe un correcteur optimal, capable de le stabiliser, dont la fonction de transfert est la suivante :

$$W_{correcteur} = \frac{\sum_{i=0}^{m} n_i s^i}{\sum_{j=0}^{n} d_j s^j} \tag{4.8}$$

La fonction de transfert en boucle fermée est la suivante :

$$W_{bf} = \frac{\sum_{k=0}^{m} n_k s^k \sum_{i=0}^{m} a_i s^i}{\sum_{l=0}^{n} d_l s^l \sum_{j=0}^{n} b_j s^j + \sum_{k=0}^{m} n_k s^k \sum_{i=0}^{m} a_i s^i} \tag{4.9}$$

Si on désire qu'elle soit du type

$$W_{équ} = \frac{1}{\sum_{l=0}^{2*m} p_l s^l} \tag{4.10}$$

alors il faut choisir les paramètres n_k et d_l de façon optimale. Il suffit pour cela de résoudre l'équation matricielle suivante :

$$\begin{pmatrix} b_0 & 0 & 0 & 0 & 0 & a_0 & 0 & 0 & 0 & 0 \\ b_1 & b_0 & 0 & 0 & 0 & a_1 & a_0 & 0 & 0 & 0 \\ \ddots & \ddots & b_0 & 0 & 0 & \ddots & \ddots & a_0 & 0 & 0 \\ \ddots & \ddots & \ddots & b_0 & 0 & \ddots & \ddots & \ddots & a_0 & 0 \\ b_m & \ddots & \ddots & \ddots & b_0 & a_m & \ddots & \ddots & \ddots & a_0 \\ 0 & b_m & \ddots & \ddots & \ddots & 0 & a_m & \ddots & \ddots & \ddots \\ 0 & 0 & b_m & \ddots & \ddots & 0 & 0 & a_m & \ddots & \ddots \\ 0 & 0 & 0 & b_m & \ddots & 0 & 0 & 0 & a_m & \ddots \\ 0 & 0 & 0 & 0 & b_m & 0 & 0 & 0 & 0 & a_m \end{pmatrix} \begin{pmatrix} d_0 \\ \vdots \\ \vdots \\ \vdots \\ d_m \\ n_0 \\ \vdots \\ \vdots \\ \vdots \\ n_m \end{pmatrix} = \begin{pmatrix} p_0 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ p_{2m+1} \end{pmatrix} \tag{4.11}$$

ou pour simplifier l'écriture :

$$\boldsymbol{AX} = \boldsymbol{P} \tag{4.12}$$

avec A la matrice des paramètres b_i et a_i, X la matrice des n_i et d_i qu'on cherche à déterminer et P la matrice des coefficients de la fonction de transfert qu'on cherche à reproduire.
Comme la matrice A n'est pas une matrice carrée, la résolution impose d'utiliser une pseudo inversion de la matrice A notée A^+ [BORNE 92]. Ainsi,

$$X = PA^+ \tag{4.13}$$

De cette manière, on obtient un vecteur solution X, dont les composantes sont les paramètres optimaux du correcteur et du filtre d'entrée. L'architecture de cette commande est présentée figure IV.4.4.

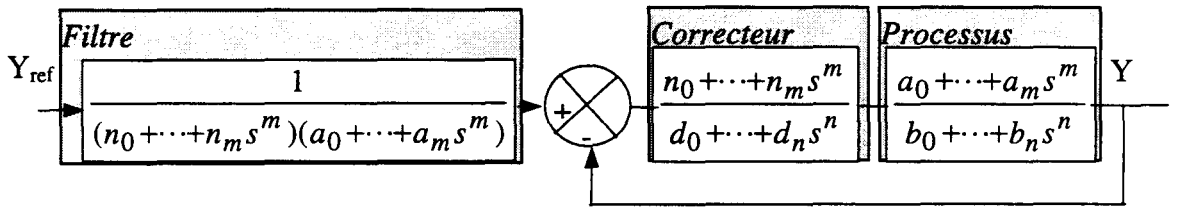


Fig IV.4.4 : Correction d'un système linéaire instable

5 Commandes neuronales

5.1 Méthodologie pour l'inversion des causalités

5.1.1 Systèmes d'ordre un

Nous nous sommes intéressés dans la formalisation de la commande d'un système causal d'ordre un. Nous prenons, pour illustrer les différentes techniques de commande, le cas général d'un asservissement d'un système liant une grandeur de sortie Y à une grandeur de commande U. Nous considérons que ce système est non linéaire afin de généraliser notre présentation. Sa représentation globale sous forme de G.I.C est la suivante (Fig IV.5.1):

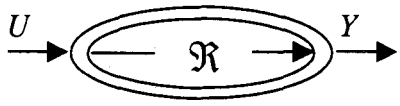


Fig IV.5.1 : Processeur causal non linéaire

Nous considérons que ce système d'ordre un est régi par une équation différentielle non linéaire, où les variables d'états sont dissociables, du type (4.13) :

$$\Re : U = f(Y) + g\left(\dot{Y}\right) \tag{4.13}$$

5.1.1.1 Inversion indirecte globale

Le formalisme d'inversion du G.I.C, en vue de l'établissement de la commande, nous indique que la causalité de ce système ne peut être inversée directement; par conséquent, le graphe de commande devient le suivant (Fig IV.5.2) :

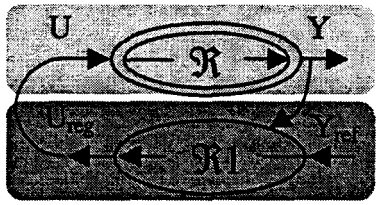


Fig IV.5.2 : Correction linéaire d'une relation non linéaire

Le fait d'utiliser un correcteur linéaire de type PI, PID, ne convient pas forcément à la bonne poursuite de la sortie du processus par rapport à la référence. En fait tout dépend de la nature et de la 'dureté' de la non linéarité. C'est pourquoi nous proposons une décomposition en plusieurs processeurs élémentaires en vue d'établir d'autres commandes permettant de prendre en compte les non linéarités.

5.1.1.2 Commande neuronale partielle

Nous pouvons représenter le G.I.C de la figure IV.5.1 de façon plus éclatée: nous considérons qu'au moins une relation rigide puisse être extraite de la relation causale globale comme l'indique le graphe de la figure IV.5.3.

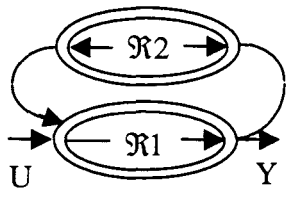


Fig IV.5.3 : Extraction de la relation rigide

Le formalisme d'inversion du G.I.C nous permet alors de concevoir un second schéma de commande composé d'une inversion indirecte et d'une anticipation de la relation rigide comme l'indique la figure IV.5.4.

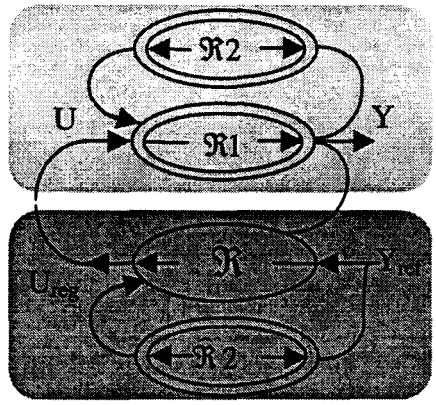


Fig IV.5.4 : Commande neuronale partielle

Nous remarquons que dans ce schéma, seule la non linéarité de la relation rigide est complètement prise en compte dans la commande. La non linéarité de la relation causale doit être linéarisée afin de déterminer au mieux la correction linéaire $\Re$; dans ces conditions, l'influence de cette non linéarité de la relation causale se répercutera sur l'allure et le temps de réponse en régime transitoire. En revanche, l'avantage d'une telle commande est de compenser les non linéarités du régime permanent; ainsi, un réseau de neurones peut approcher la relation $\hat{\Re}2$ et être placé en lieu et place de cette relation dans le schéma de commande.

5.1.1.3 Commande neuronale totale

Afin de pallier à l'absence de contrôle total au cours des régimes transitoires, nous avons décomposé en processeur élémentaire le G.I.C de la figure IV.5.3 afin de déterminer un troisième schéma de commande capable de prendre en compte toutes les non linéarités. Le G.I.C du processus associé à sa commande est alors représenté figure IV.5.5

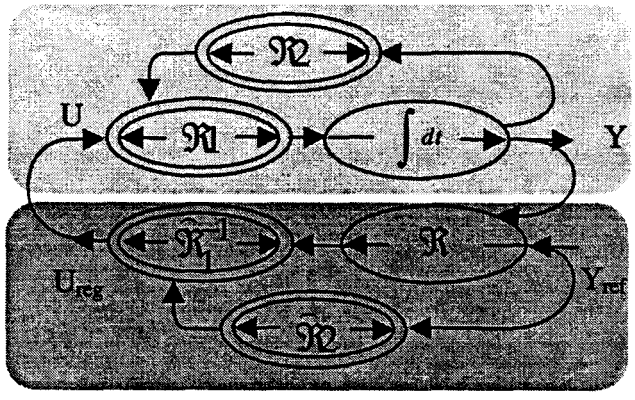


Fig IV.5.5 : Commande neuronale totale

Comme dans le cas du G.I.C de la commande neuronale partielle, nous avons extrait une première relation rigide $\Re 2$, puis nous avons décomposé la relation non linéaire causale restante en une relation non linéaire rigide $\Re 1$ et une relation linéaire causale qui n'est autre qu'un intégrateur. Par conséquent la relation $\Re 2$ peut être compensée par sa relation estimée $\hat{\Re}2$ comme dans le cas de la commande neuronale partielle puis, la relation $\Re 1$ peut être corrigée par sa relation inverse estimée $\hat{\Re}_1^{-1}$. Ainsi, toutes les non linéarités peuvent être prise en compte complètement. L'unique relation causale étant un intégrateur pur, celui ci peut être

corrigé à l'aide d'un correcteur $\Re$ qui permettra de fixer le temps de réponse du système ainsi bouclé. Il est alors possible d'approcher les relations $\widehat{\Re}_1^{-1}$ et $\widehat{\Re}2$ par un réseau de neurones. L'avantage de cette commande est alors cette fois de prendre en compte toutes les non linéarités du système et de corriger le système sans avoir à identifier le moindre paramètre. Le réseau de neurones se comporte comme un abaque capable de donner le couple de réglage exact pour passer d'un point de fonctionnement à un autre.

La principale difficulté réside dans la phase d'échantillonnage, car ce type de commande nécessite la connaissance de la commande U dans l'espace des phases $(Y, \dot{Y})$ et impose par conséquent la reconstitution de cette dernière. Celle ci est réalisée par un calcul aux variations (à gauche) qui nécessite un filtrage ou lissage préalable de la vitesse afin d'éviter les perturbations liées aux bruits. En effet, si la période d'échantillonnage Te est très petite et que la dérivée se calcule par variation $\frac{Y_k - Y_{k-1}}{Te}$, le moindre bruit de mesure sur Y peut faire changer le signe de la dérivée et, par conséquent, errone les calculs. Cependant, comme ce filtrage entraîne inexorablement un déphasage, le modèle neuronal du système sera forcement biaisé.

5.1.2 Généralisation aux systèmes d'ordre *n*

Nous allons donc généraliser les trois types de commande à un système d'ordre n. Nous prenons comme précédemment le cas d'un asservissement d'un processus liant une grandeur de commande U et une grandeur de sortie Y.

5.1.2.1 Inversion indirecte globale

La représentation d'un système d'ordre n non linéaire est exactement la même que celle présentée figure IV.5.1. En utilisant le principe d'inversion du G.I.C, on obtient le même schéma de commande que celui de la figure IV.5.2: dans ce cas, la détermination du correcteur nécessite alors la linéarisation du système par un hyperplan. La nature des non linéarités implique des déformations des trajectoires de poursuite.

5.1.2.2 Commande neuronale partielle

Afin de prendre en compte les non linéarités en régime permanent, nous allons extraire une relation rigide liée à la variable de sortie du système. La représentation de ce G.I.C est identique à celle de la figure IV.5.3. Néanmoins, la relation causale non linéaire restante est d'ordre *n*. Dans ces conditions le correcteur devra assurer au mieux la poursuite de cette dernière relation de façon à amener le système global dans un régime proche du régime permanent, puis la compensation $\hat{\Re}2$ assurera la diminution, voire l'annulation des effets des non linéarités en régime permanent. Par conséquent, un réseau de neurones peut approcher la relation $\Re 2$ et être mis en lieu et place de la relation $\hat{\Re}2$. Cette commande ne requiert en fait que l'échantillonnage du système et l'apprentissage de cette caractéristique en régime permanent. En revanche, les paramètres de la relation causale non linéaire restante devront être identifiés.

5.1.2.3 Commande neuronale globale

Afin de prendre en compte toutes les non linéarités du système, nous décomposons le G.I.C en autant de processeurs élémentaires simples que nécessaire, afin de ne faire apparaître que des intégrateurs purs et des relations rigides, regroupées en une relation rigide $\Re$. Ainsi, l'inversion du G.I.C permet de déterminer la commande (figure IV.5.6) dans laquelle une relation non linéaire est requise $\hat{\Re}^{-1}$ afin de compenser toutes les non linéarités, et où les relations $\Re 1$ à $\Re n$ permettent l'inversion indirecte des intégrateurs purs et offrent ainsi la possibité de contrôler les transitoires.

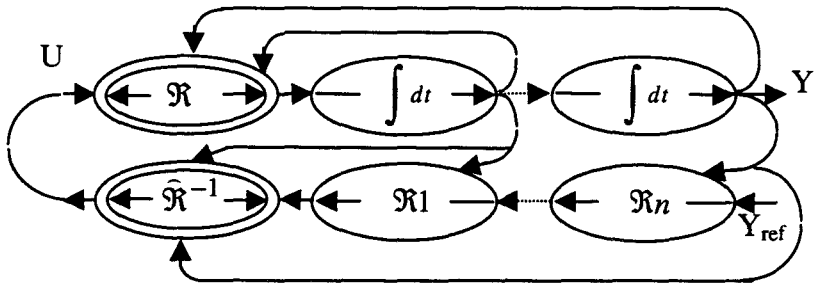


Fig IV.5.6 : G.I.C d'une commande neuronale totale d'un système d'ordre n

L'utilisation de ce type de commande impose de recréer les dérivées successives de la sortie nécessaires aux comparaisons opérées dans les relations $\Re_1$ à $\Re_n$. Face au problème

de bruit de mesure, la première idée qui vient à l'esprit est de filtrer ou lisser les variables d'entrée puis de calculer les dérivées par calculs successifs des variations. Cette méthode reste valable jusqu'à la dérivée première à condition que les transitoires ne soient pas trop rapides, mais au-delà de la dérivée seconde la détermination correcte des variations s'avère délicate. C'est pourquoi, il est impératif dans ce cas d'avoir recours à un modèle interne du processus capable d'en estimer la sortie ainsi que ses dérivées. On obtient alors le schéma de commande ci-dessous (figure IV.5.7) :

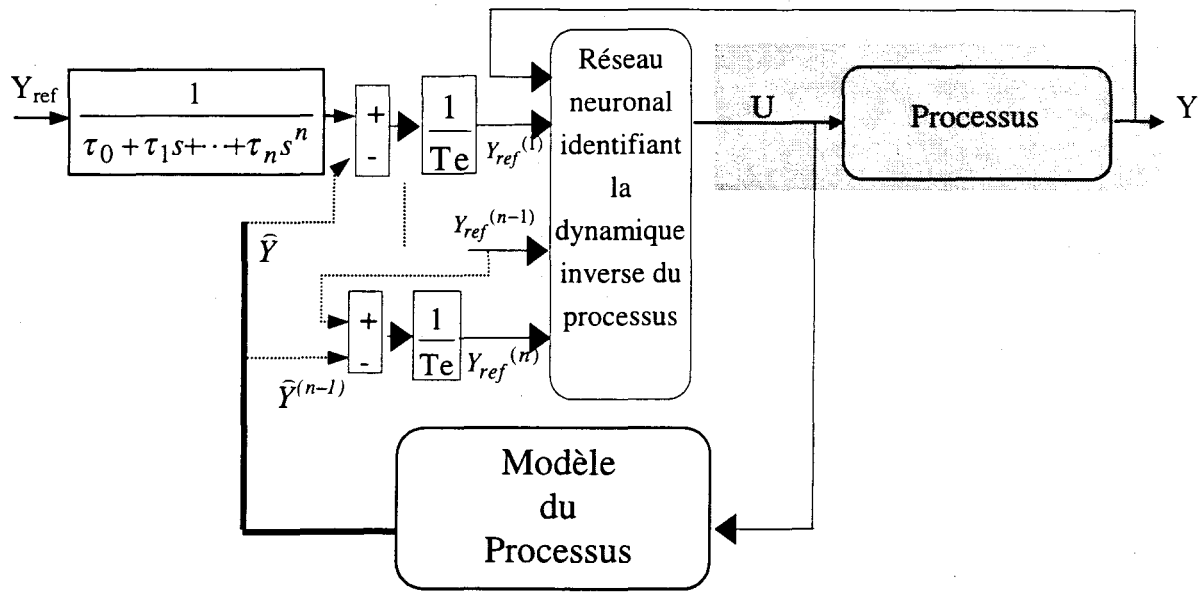


Fig IV.5.7 : Schéma de commande neuronale totale d'un système non linéaire d'ordre n

La fonction de transfert placée sur la consigne permet d'imposer le comportement en poursuite du processus une fois asservi. Néanmoins, comme son nom l'indique, le modèle du processus ne reflète pas exactement la réalité, il faut corriger les imperfections des estimations fournies ; pour cela l'estimateur doit être transformé alors en observateur [BORNE 93].
Par ailleurs, comme le concepteur de l'asservissement n'est pas censé connaître la valeur des paramètres du processus, le réseau de neurones s'impose également dans l'élaboration du modèle. Deux structures neuronales sont alors possibles dans ce cas :

La première solution consiste à rajouter une matrice de gain d'observation dont les coefficients sont calculés pour minimiser les erreurs commises par l'estimateur. La difficulté majeure réside dans le choix des gains de correction qui théoriquement nécessitent une connaissance du modèle. C'est pourquoi une seconde solution basée sur une technique d'adaptation en ligne est proposée.

Il s'agit donc de modifier en ligne les poids du réseau de neurones jouant le rôle d'estimateur. Ceci implique une acquisition de mesures de points de fonctionnement, dont la répartition doit être homogène afin de posséder une connaissance du système sur la totalité de l'espace d'état. A partir de ces mesures, il faut calculer la surface de régression qui servira au nouvel apprentissage, ce calcul nécessite une inversion de matrice importante si on utilise la méthode d'initialisation présentée au second chapitre. En temps réel, ceci risque d'être contraignant à moins de posséder un processeur numérique très puissant, ou de pouvoir soit dérouter le calcul sur un autre processeur dédié à cette tâche, soit interrompre si c'est possible le processeur qui réalise la commande. Il ne faut pas oublier que si cette technique est adoptée, les calculs à effectuer seront doubles puisque, si l'estimateur est adapté, le correcteur doit l'être également. En pratique, nous préférons plutôt adopter la rétropropagation pour adapter en ligne le modèle et le correcteur. Néanmoins comme les structures neuronales différent, la première est récurrente l'autre pas, les algorithmes d'apprentissage ne seront pas les mêmes ce qui implique une distorsion au niveau de l'adaptation.

Ces considérations handicapent fortement la commande neuronale totale pour des systèmes d'ordre supérieur à un.

6 Applications

Nous allons illustrer au travers de deux applications, la prise en compte d'une charge mécanique non linéaire et la prise en compte d'une inductance non linéaire, les différentes commandes exposées précédemment.

6.1 Charge mécanique non linéaire

6.1.1 Commande neuronale partielle

6.1.1.1 Structure de commande

Considérons le cas du réglage de la boucle de vitesse d'une machine électrique. Pour simplifier les explications, nous négligeons la dynamique de la boucle de courant. On suppose que la charge mécanique accouplée à la machine est non linéaire en fonction de la vitesse et linéaire en fonction de l'accélération. Etant donné que nous ne souhaitons compenser que les non linéarités d'une part, qu'il est très périlleux de chercher à reconstituer une dérivée d'autre

part, nous ne considérons comme perturbation dans le processus, que la partie concernant les régimes permanents. On peut donc décomposer le G.I.C de la boucle de vitesse ainsi commandée :

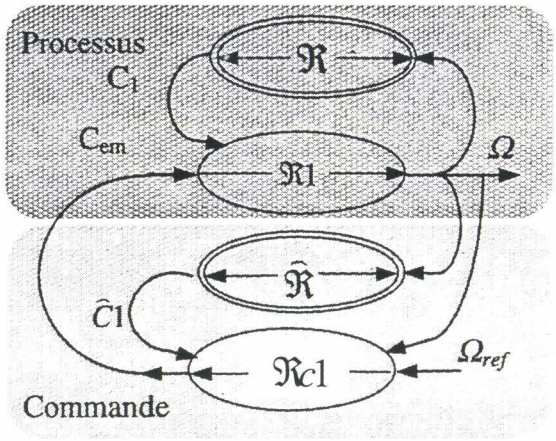


Fig IV.6.1 : Boucle de vitesse avec commande neuronale partielle

$$\begin{aligned} \Re &\rightarrow C_1 = f(\Omega) + C_r sign(\Omega) \\ \text{avec } \Re 1 &\rightarrow C_{acc} = J\dot{\Omega} \\ Cem &= Cacc + C1 \end{aligned} \tag{4.14}$$

où *Cem* est le couple électromoteur, *Cacc* le couple d'accélération et *C1* le couple perturbateur. La relation $\Re c1$ décrit la correction de la relation $\Re 1$. Elle doit compenser le pôle du processus par un zéro et placer un pôle pour imposer la dynamique de la boucle fermée. Etant donné que $\Re 1$ représente un intégrateur pur, on peut se contenter de mettre un simple gain pour le correcteur, par rapport à une réponse indicielle.

$$\begin{aligned} \Re c1 &\rightarrow Cacc_{reg} = K(\Omega ref - \Omega) \\ \hat{\Re} &\rightarrow \hat{C}1 = g(\Omega) \end{aligned} \tag{4.15}$$

6.1.1.2 Schéma de commande

Le réseau de neurones est, ici encore, en lieu et place de la relation non linéaire $\Re$. Le schéma de commande global devient alors le suivant :

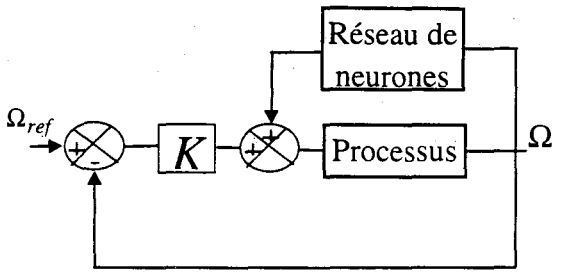


Fig IV.6.2 : Schéma de commande avec compensation neuronale

6.1.1.3 Stabilité

Pour étudier la stabilité nous linéarisons le processus ainsi que la perturbation.

$$\varepsilon_c = C1 - \hat{C}1 = \varepsilon_\Omega \Omega \tag{4.16}$$

Le processus compensé a donc comme fonction de transfert équivalente

$$\frac{\Omega(s)}{Cacc(s)} = \frac{1}{\varepsilon_\Omega + Js} \tag{4.17}$$

soit en boucle fermée si le correcteur principal est un simple gain K :

$$\frac{\Omega(s)}{\Omega_{ref}(s)} = \frac{1}{1 + \frac{\varepsilon_\Omega}{K} + \frac{J}{K}s} \tag{4.18}$$

Le système sera stable tant que la condition (4.19) ci-dessous sera vérifiée :

$$\varepsilon_\Omega > -K \tag{4.19}$$

Etant donné que le réseau de neurones est sensé avoir appris correctement le couple perturbateur, l'erreur ε_Ω est sensée être petite, et la condition ci-dessus sera d'autant plus vérrifiée que le gain K sera d'autant plus grand.

6.1.1.4 Résultats

Le résultat de la figure IV.6.3 montre l'efficacité d'une commande neuronale partielle grâce notamment à la compensation de la perturbation non linéaire dans le cas d'un asservissement en vitesse d'une machine électrique accouplée à une charge fortement non linéaire du type présenté figure III.7.10. On remarque des oscillations au démarrage ce qui prouvent un manque de précision sur la modélisation de la perturbation non linéaire au niveau des vitesses très faibles.

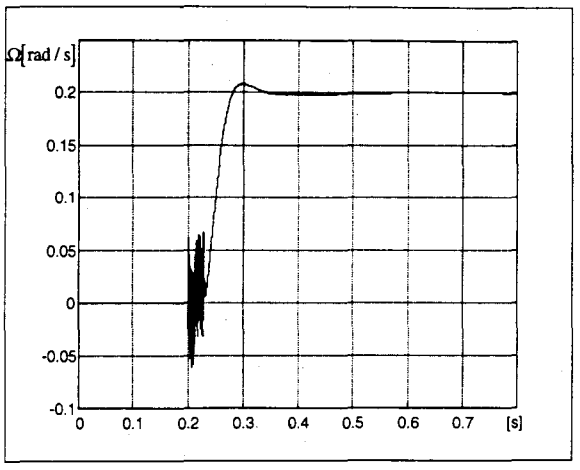


Fig IV.6.3 : Résultat d'une commande neuronale partielle

6.1.2 Commande neuronale totale

6.1.2.1 Structure de commande

Le principe est de disposer d'un modèle inverse du processus aussi proche que possible de la réalité, puis de le mettre en oeuvre comme un abaque capable de donner la valeur de la commande pour passer d'un point de fonctionnement à un autre.

Nous allons nous appuyer sur l'exemple explicité au paragraphe 5.1.1.3, en supposant cette fois qu'il s'agit d'une charge mécanique non linéaire accouplée à une machine électrique. Les relations du processus et de commande sont les suivantes :

$$\dot{\Omega} = \frac{1}{J}\left(Cem - f\Omega - Cr\right) \qquad (4.20)$$

où J est l'inertie, f les frottements, C_r le couple de charge introduisant les non linéarités et C_{em} le couple électro-moteur de la machine.La représentation graphique du système et de sa commande est montrée figure IV.6.4.

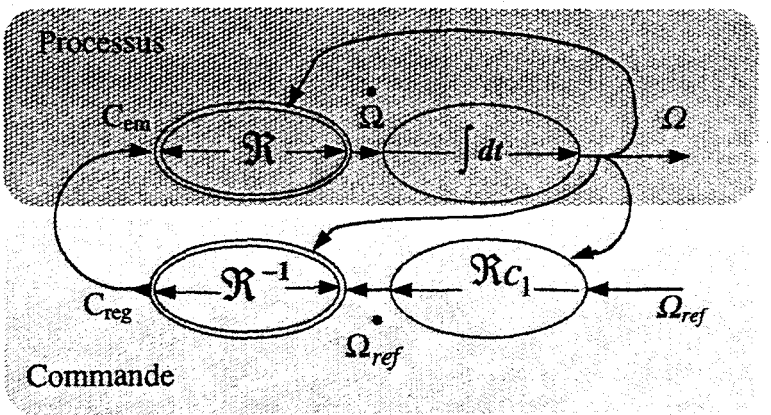


Fig IV.6.4 : G.I.C d'une boucle d'asservissement de vitesse

Ici encore, la structure de commande s'obtient par simple inversion des relations qui régissent le système. La relation liant la vitesse au couple électromagnétique est causale et non linéaire ; néanmoins, il est possible d'extraire de cette relation la partie rigide ; la partie causale n'est autre qu'une intégration, tout comme pour un système linéaire. De ce fait, la relation rigide peut être prise en compte dans la boucle d'asservissement par la relation rigide inverse directe à condition qu'il y ait univocité. Quant à l'intégrateur, il ne peut être inversé qu'indirectement. Dans cette commande, le réseau de neurones est capable de prendre en compte toutes les non linéarités et agit comme un gain variable fonction de l'accélération de réglage et de la vitesse à laquelle la machine tourne.

6.1.2.2 Topologie de réseau

Etant donné que l'on souhaite utiliser un modèle inverse de la relation non linéaire $\Re$, la topologie neuronale choisie doit être optimisée comme nous l'avons vu au chapitre III (réseau pour lequel les fonctions d'activation non linéaires ont été judicieusement choisies). La principale difficulté réside également dans la phase d'échantillonnage et concerne la mesure ou plutôt la reconstitution de l'accélération.

6.1.2.3 Schéma de commande

Nous avons testé le schéma de commande (figure IV.6.5) sur une machine asynchrone commandée vectoriellement.

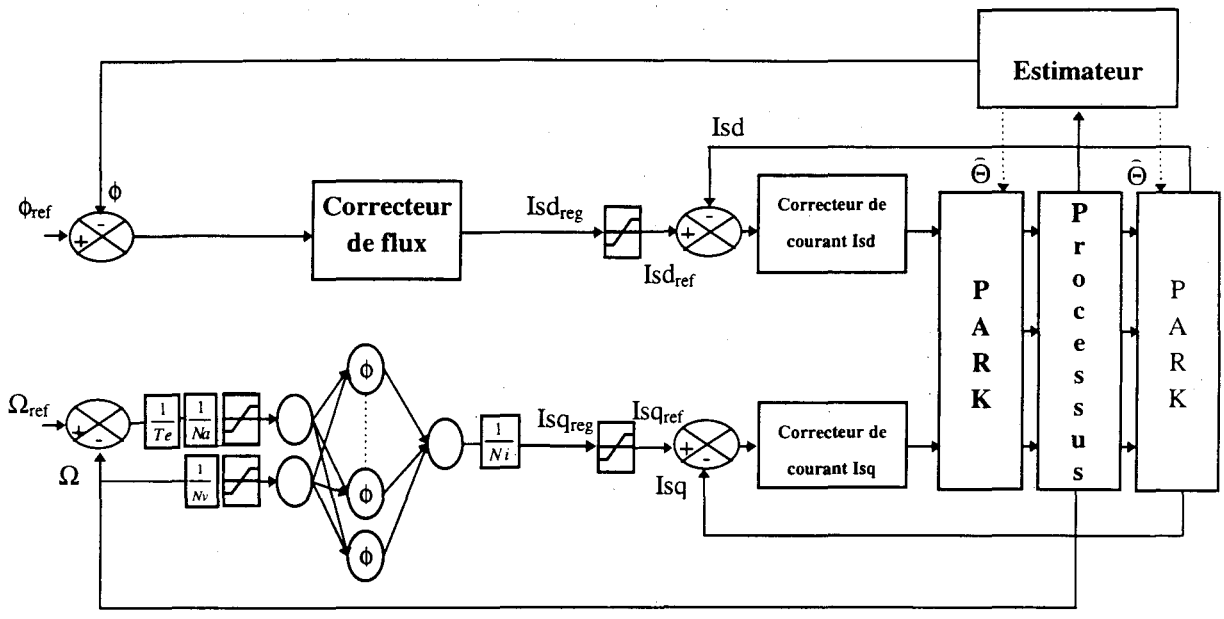


Fig IV.6.5 : Schéma de commande neuronale

La machine est commandée dans les axes *dq* obtenus grâce à une transformation de Park, qui permet ainsi de découpler les commandes du flux et du couple [LEONARD 85].

Les figures IV.6.6 montrent les réponses en vitesse, simulées et expérimentales d'une telle commande. Deux cas de charge ont été envisagés : l'une en simulation de type 'ventilateur' dont l'allure est présentée figure III.7.10, l'autre en expérimentation est une charge linéaire comportant un fort frottement sec à l'origine (figure III.7.11). Bien que les charges soient différentes on obtient les mêmes types de réponses, c'est à dire des réponses du premier ordre qui prouvent que la commande ainsi mise en oeuvre, linéarise bien le processus. En revanche, on peut remarquer que dans les deux cas, il persiste une erreur statique due aux erreurs d'apprentissage et comme le système est non linéaire, la valeur de cette erreur dépend du point de fonctionnement. La cause de cette erreur sera analysé au paragraphe suivant.

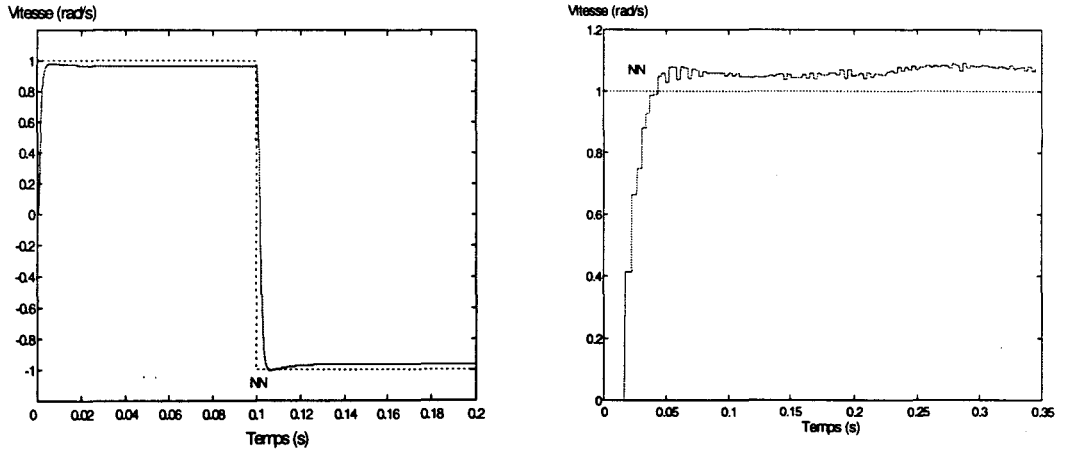


Fig IV.6.6 : Commande neuronale totale simulation (a) et expérimentation (b)

6.1.2.4 Stabilité

L'étude de la stabilité d'un système non linéaire n'est pas chose aisée ; de plus, lorsque l'on parle de stabilité pour les systèmes non linéaires, il faut commencer par préciser quel type de stabilité nous étudions. En effet, pour ce genre de système, on peut distinguer la stabilité asymptotique, la stabilité exponentielle, locale,... Des méthodes ont été développées pour vérifier les critères de stabilité, notamment la méthode des fonctions candidates de Lyapounov. Néanmoins, cette méthode nécessite la connaissance du processus. Or, dans le cas de notre commande neuronale, nous ne connaissons pas le processus, par conséquent, nous proposons une méthode fondée sur une linéarisation des équations du modèle de connaissance. Nous appliquons cette approche au cas de la commande neuronale en vitesse d'une charge mécanique, explicitée au paragraphe précédent.

L'équation du correcteur neuronal donnant le couple de référence en fonction de la vitesse et de l'accélération de référence peut se ramener à:

$$f\left(\dot{\Omega}_{ref}\right) + g(\Omega) + \hat{C}_r = C_{ref} \tag{4.21}$$

où *f()* et *g()* sont des fonctions neuronales d'approximation ainsi que le couple de frottement sec $\hat{C}_r$.

Posons maintenant le modèle de connaissance du processus servant d'exemple :

$$J\dot{\Omega} + f\Omega^2 + C_r = C_{em} \tag{4.22}$$

f est le coefficient de frottements visqueux, J le moment d'inertie réelle. Le modèle neuronal est par nature inexact et de plus, depuis l'identification, le processus peut avoir évolué. Des écarts interviennent donc entre ces caractéristiques. La loi de commande impose alors :

$$f\left(\dot{\Omega}_{ref}\right) + g(\Omega) + \hat{C}_r = J\dot{\Omega} + C_r + f\Omega^2 \tag{4.23}$$

En supposant que les fonctions remplies par les réseaux de neurones sont de même natures que celles du processus, g(W) est une caractéristique quadratique de la vitesse. Nous définissons alors ε_Ω et ε_{Cr} les expressions linéaires respectives entre les fonctions réelles et estimées.

$$\frac{K}{Tc}\left(\Omega_{ref} - \Omega\right) + \varepsilon_\Omega\Omega + \varepsilon_{Cr} = J\dot{\Omega} \tag{4.24}$$

Par ailleurs le coefficient $\frac{K}{Tc}$ représente le gain du correcteur $\Re 1$ de la figure IV.6.4. Nous obtenons finalement une équation différentielle du premier ordre en Ω.

$$\frac{K}{Tc}\Omega_{ref} + \varepsilon_{Cr} = \left(\frac{K}{Tc} - \varepsilon_{\Omega}\right)\Omega + J\dot{\Omega} \tag{4.25}$$

Ainsi nous pouvons écrire que si ε_{Ω} est petit ainsi que Tc, alors l'équation précédente reste une équation différentielle stable du premier ordre tant que $\frac{K}{Tc} - \varepsilon_{\Omega} \geq 0$. La stabilité du système reste difficile à apprécier car elle dépend des paramètres du système et des erreurs de modélisation qui sont a priori inconnues. Néanmoins, nous pouvons supposer au vu des résultats de modélisation que le correcteur restitue une bonne estimation de la charge réelle et comme la période d'échantillonnage de la commande Tc, est faible, alors $\frac{K}{Tc} > |\varepsilon_{\Omega}|$ ce qui rend le système stable.

L'équation linéarisée précédemment permet d'expliciter l'erreur statique qui se produit en régime permanent, et qu'il est possible d'estimer par un calcul à la limite, en fonction des erreurs de modélisation :

$$\lim_{t\to\infty}\Omega = \frac{\frac{K}{Tc}\Omega_{ref} + \varepsilon_{Cr}}{\left(\frac{K}{Tc} - \varepsilon_{\Omega}\right)} \tag{4.26}$$

Une adaptation en ligne s'impose alors afin d'annuler cette erreur statique due aux erreurs de modélisation. De plus, cette adaptation permettra également la prise en considération des variations éventuelles du processus.

6.2 Charge inductive

6.2.1 Structures de commande

Considérons le réglage du courant dans une bobine saturable. Ce système peut être décrit par le modèle de connaissance :

$$\begin{aligned} &\Re 2 : U - \frac{d\Phi}{dt} = Ri \\ &\Re : \Phi = L(i)i \end{aligned} \tag{4.27}$$

où R est la résistance, Φ le flux et L l'inductance. Nous pouvons alors représenter le G.I.C de ce processus ainsi que la commande qui en découle (figure IV.6.7).

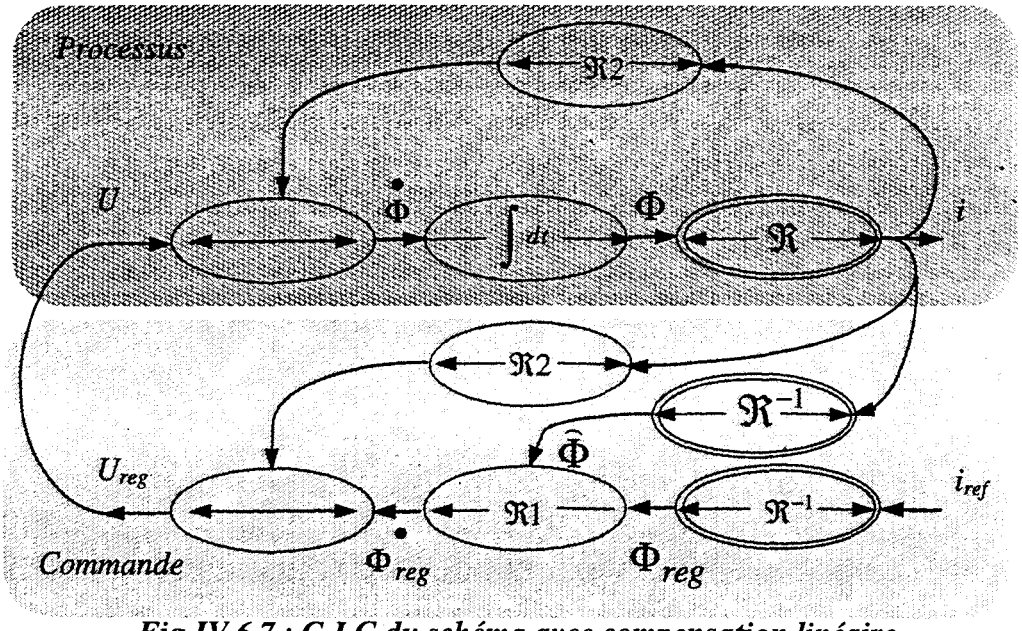


Fig IV.6.7 : G.I.C du schéma avec compensation linéaire

Dans ce schéma de commande le processus n'est pas linéarisé à tout instant puisque les relations non linéaires $\Re$ et $\Re^{-1}$ se compensent de part et d'autre d'une relation causale équivalente. Nous proposons par conséquent un autre schéma (figure IV.6.8) qui linéarise cette fois la boucle de courant.

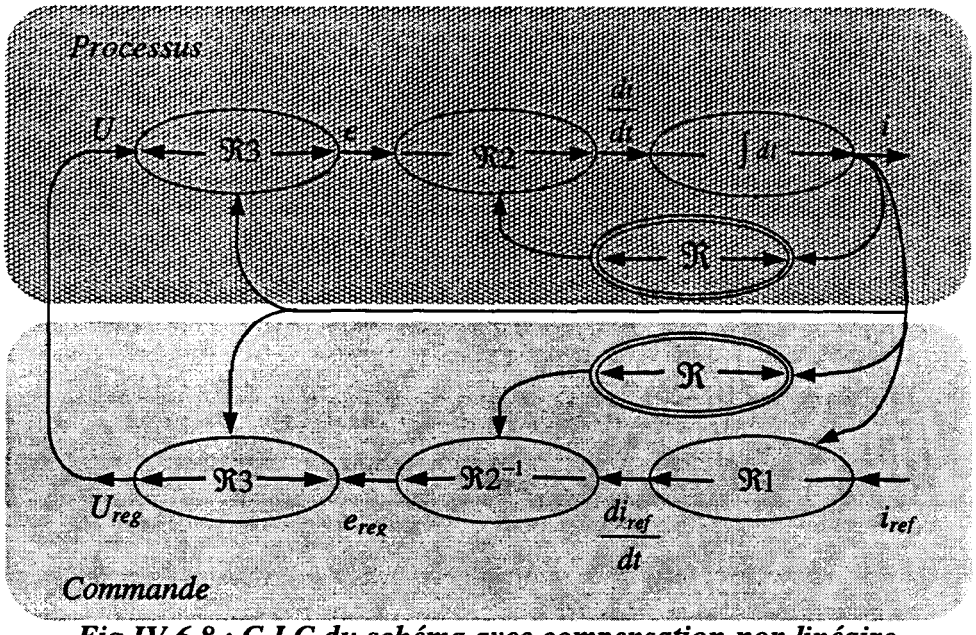


Fig IV.6.8 : G.I.C du schéma avec compensation non linéaire

Nous définissons une f.e.m e telle que :

$$\Re 3 : e = U - Ri \tag{4.28}$$

L'inductance incrémentale L_d résulte de la dérivation de l'inductance non linéaire par rapport au temps :

$$\Re : L_d = \frac{dL(i)i}{dt} = \frac{dL(i)}{di}\frac{di}{dt}i + L(i)\frac{di}{dt} \tag{4.29}$$

La relation $\Re 2$ définit alors le lien entre la dérivée du courant par rapport au temps et la f.e.m.

$$\Re 2 : \frac{\mathrm{di}}{\mathrm{dt}} = \frac{e}{L_d} \tag{4.30}$$

6.2.2 Stabilité

En supposant que la résistance suit convenablement identifiée et en linéarisant les inductances incrémentales réelles L_d et estimées $\hat{L}_d$, la boucle d'asservissement du courant s'apparente à celle ci dessous

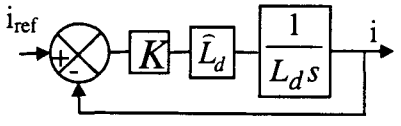


Fig IV.6.9 : Schéma résultant de la compensation

Dans cette configuration, la différence entre les inductances réelles et estimées va influencer uniquement la dynamique de l'asservissement, mais ne rendra en aucun cas instable cette boucle

6.2.3 Résultats

Les résultats expérimentaux présentés ci-dessous illustrent les deux structures de commande décrites précédemment. Les calculs d'asservissement et d'acquisition étant faits à partir d'une simple carte PC, nous avons limité le réseau à cinq neurones pour approcher les fonctions non linéaires nécessaires aux compensations.

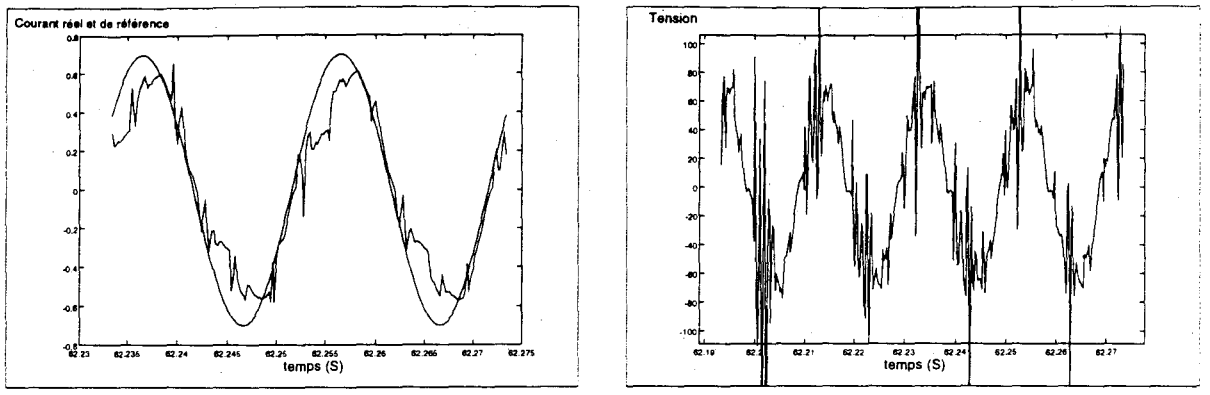


Fig IV.6.10 : Résultats obtenus avec la première structure de commande

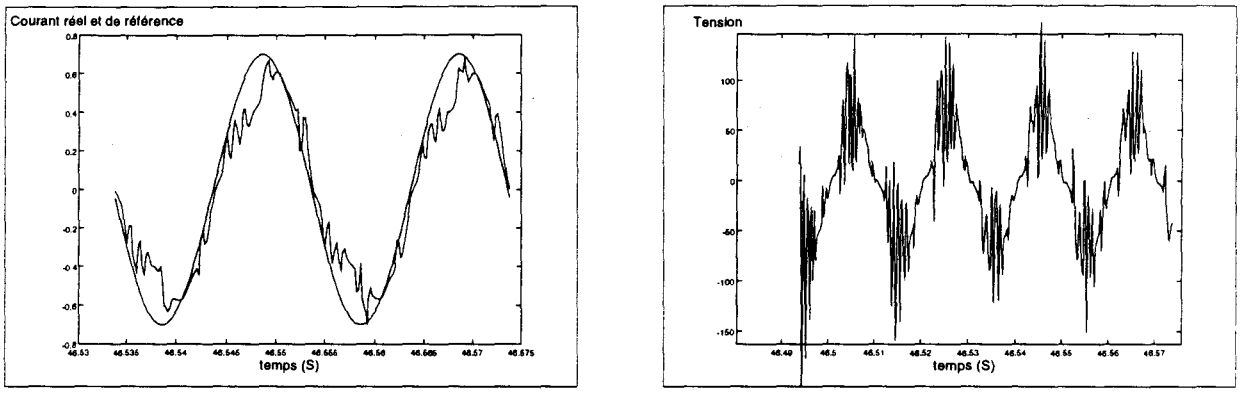


Fig IV.6.11 : Résultats obtenus avec la seconde structure de commande

Les résultats semblent similaires, néanmoins ont peut remarquer que la forme en 'tétines' de la tension est plus nette dans le second cas, ce qui tend à prouver que ce schéma de commande linéarise la boucle de courant contrairement à la première structure. On peut remarquer pour les deux asservissements que les courbes de courants réels ne 'collent' pas exactement à la référence à poursuivre. Ceci est notamment dû à l'effet de l'hystérésis qui n'est pas compensé et qui ne peut pas l'être en raison du faible nombre de neurones imposé par la puissance de calcul disponible.

7 Adaptation en ligne

Comme nous l'avons vu au paragraphe IV.6.1.2.4 les erreurs d'apprentissage sur les réseaux neuronaux utilisés ultérieurement en commande neuronale totale entraînent

l'apparition d'erreurs statiques sur les réponses indicielles. C'est pourquoi il est important de modifier les poids du correcteur neuronal pour pallier ce problème et parer à d'éventuelles variations paramétriques du processus. On se limitera dans cette partie au cas de la commande neuronale totale.

7.1 Adaptations directe et indirecte

Considérons une commande qui repose sur une structure en boucle(s) comportant comparateur(s) et correcteur(s). Le but de cette commande est d'imposer la sortie d'un processus suivant la référence désirée. Si l'un des correcteurs est mal calculé, il est alors intéressant d'ajouter un mécanisme d'adaptation (apprentissage en ligne) pour redimensionner la commande. Cette adaptation peut être directe ou indirecte.

L'adaptation indirecte (Fig IV.7.1) consiste dans un premier temps à identifier les paramètres du processus puis dans un second temps à tenir compte de leurs variations en recalculant les paramètres du correcteur. Ce mécanisme ne convient pas aux correcteurs ou aux modèles neuronaux que nous utilisons, puisque ceux-ci identifient pas des paramètres mais plutôt une structure.

L'adaptation directe (Fig IV.7.2) permet en revanche de modifier les paramètres d'un modèle ou d'un correcteur suivant une loi déterministe d'adaptation sans recourir à une identification du processus.

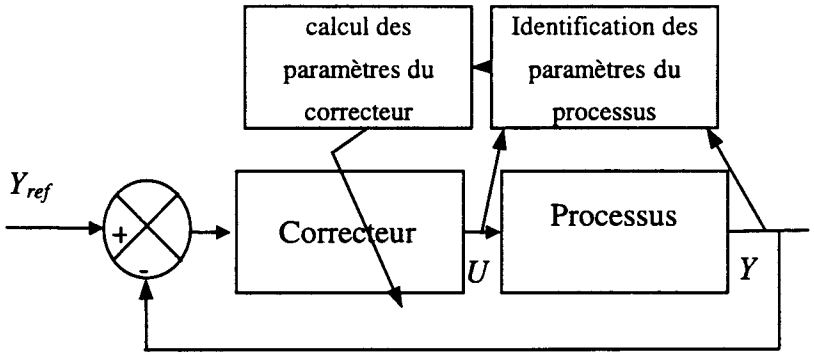


Fig IV.7.1 Schéma d'adaptation indirecte

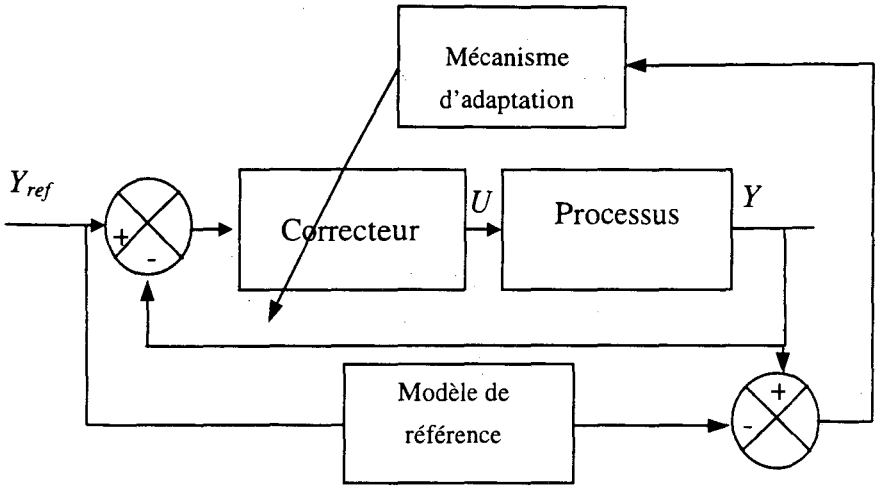


Fig IV.7.2 : Schéma d'adaptation directe avec modèle de référence

7.2 Calcul d'une loi d'adaptation

L'adaptation directe en ligne est réalisée en minimisant un critère quadratique J qui s'écrit en fonction des écarts quadratiques entre la sortie du processus Y et la sortie du modèle Y^* (4.31).

$$J = \frac{1}{2} E\left\{\left(Y^*(k) - Y(k)\right)^T Q\left(Y^*(k) - Y(k)\right)\right\} \tag{4.31}$$

avec Q une matrice de pondération

La mise à jour des poids du réseau, placés dans le vecteur Θ, est réalisée par une approche itérative de type 'descente de gradient' de la façon suivante :

$$\Theta(k+1) = \Theta(k) - \mu \frac{\partial J}{\partial \Theta} \tag{4.32}$$

avec μ le coefficient d'apprentissage.

Cette méthode est une méthode de résolution quasi newtonienne d'équations non linéaires. Développons la formule précédente pour affiner les calculs d'adaptation :

$$\frac{\partial J}{\partial \Theta} = \frac{\partial\left\{\frac{1}{2} E\left\{\left(Y^*(k) - Y(k)\right)^T Q\left(Y^*(k) - Y(k)\right)\right\}\right\}}{\partial \Theta} \tag{4.33}$$

soit encore :

$$\frac{\partial J}{\partial \Theta} = \frac{\partial\left\{\left(Y^*(k) - Y(k)\right)\right\}}{\partial \Theta} Q\left(Y^*(k) - Y(k)\right) \tag{4.34}$$

puisque la sortie du modèle ne dépend pas de Θ, on a :

$$\frac{\partial J}{\partial \Theta} = -\frac{\partial Y(k)}{\partial \Theta(k)} Q\left(Y^*(k) - Y(k)\right) \tag{4.35}$$

La mise à jour des poids devient alors la suivante :

$$\Theta(k+1) = \Theta(k) + \mu \frac{\partial Y(k)}{\partial \Theta(k)} Q\left(Y^*(k) - Y(k)\right) \tag{4.36}$$

Comme la sortie du processus dépend indirectement du vecteur Θ des poids à travers la loi de commande $U(k)$, la loi de mise à jour devient :

$$\boxed{\Theta(k+1) = \Theta(k) + \mu \frac{\partial U(k)}{\partial \Theta(k)} \frac{\partial Y(k)}{\partial U(k)} Q\left(Y^*(k) - Y(k)\right)} \tag{4.37}$$

avec $\dfrac{\partial U(k)}{\partial Y(k)}$ le vecteur des dérivées partielles de la sortie du réseau neuronal par rapport au différents poids :

$$\frac{\partial U(k)}{\partial \Theta(k)} = \left\{ \frac{\partial U(k)}{\partial NW1(1,1)}, \cdots, \frac{\partial U(k)}{\partial NB1(1,1)}, \cdots, \frac{\partial U(k)}{\partial NW2(1,1)}, \cdots \right\} \tag{4.38}$$

Le calcul du terme $\dfrac{\partial Y(k)}{\partial U(k)} Q$ nécessite la connaissance a priori du processus. N'étant pas dans ce cas de figure nous regroupons le terme $\mu \dfrac{\partial Y(k)}{\partial U(k)} Q$ au sein d'un même coefficient d'apprentissage μ' choisi empiriquement. Plus ce coefficient sera élevé plus la modification des coefficients sera rapide, et inversement si le coefficient est petit. Néanmoins, lorsque le processus se stabilise dans un régime permanent identique à celui de la consigne qu'on

cherche à suivre, cela ne signifie pas que le modèle neuronal ait parfaitement appris le nouveau processus. En fait la modification des coefficients du réseau neuronal aura été telle que le processus aura été contraint d'évoluer jusqu'à ce que les écarts entre le processus et le correcteur se soient annulés.

7.3 Choix du référentiel d'adaptation

Le choix du modèle n'est pas sans influence sur la stabilité de l'adaptation. En effet si ce dernier ne reflète pas la dynamique et le comportement du processus asservi, il provoquera alors une erreur non nulle entre la sortie du processus et la sortie du modèle, même si le correcteur correspond effectivement au modèle dynamique inverse du processus. Par conséquent, il faut accorder un soin tout particulier à l'élaboration du modèle.

7.3.1 Adaptation selon un modèle interne

La première démarche consisterait à placer un modèle du processus en sortie du correcteur (figure IV.7.4). D'après la remarque faite ci-dessus, si le processus est non linéaire, alors il faut que le modèle le soit également. La réalisation du modèle peut se faire au moyen d'un réseau de neurone qu'on nomme réseau émulateur. Néanmoins, l'approche neuronale du processus n'étant pas parfaite en raison des erreurs d'apprentissage, le modèle risque d'engendrer des erreurs dans l'adaptation du correcteur. De plus, l'utilisation d'un réseau émulateur alourdit l'implantation en temps réel. Par conséquent, le schéma d'adaptation ci-dessous ne convient pas, d'autant plus que si le processus a évolué, l'émulateur du processus est également faussé et ne peut donc servir de référentiel à une adaptation en ligne.

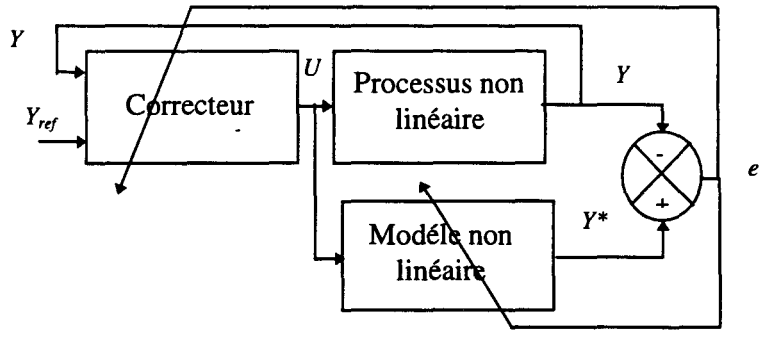


Fig IV.7.4 : Schéma d'adaptation selon un modèle interne

7.3.2 Adaptation selon un modèle de référence

Puisque le correcteur est un réseau neuronal qui est supposé avoir appris la parfaite dynamique inverse du processus non linéaire, alors le processus en boucle fermée est supposé linéaire comme l'indique de façon simpliste le G.I.C ci dessous.

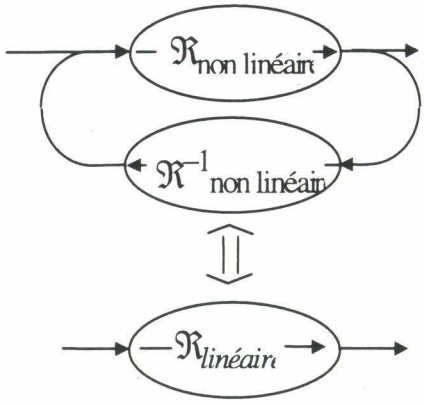


Fig IV.7.5 : Linéarisation d'un processus non linéaire grâce à la commande neuronale totale

Par conséquent on peut utiliser un modèle linéaire de référence dans le schéma d'adaptation. Les conditions d'utilisation de ce modèle sont les mêmes que celles requises pour l'utilisation du correcteur neuronal ; c'est à dire que la validité du modèle linéaire n'est effectué que lorsqu'on utilise le correcteur neuronal dans son domaine d'apprentissage.

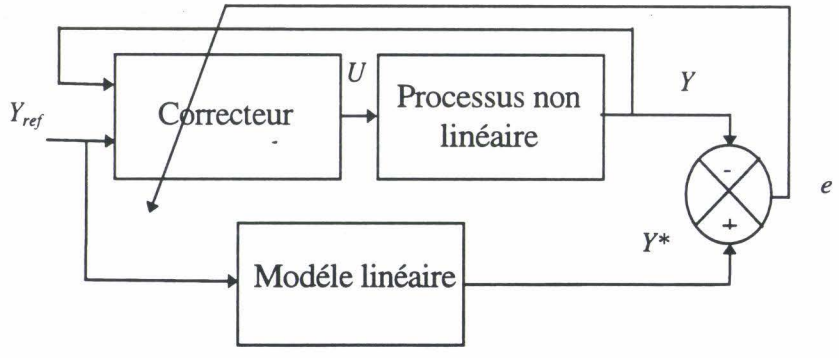


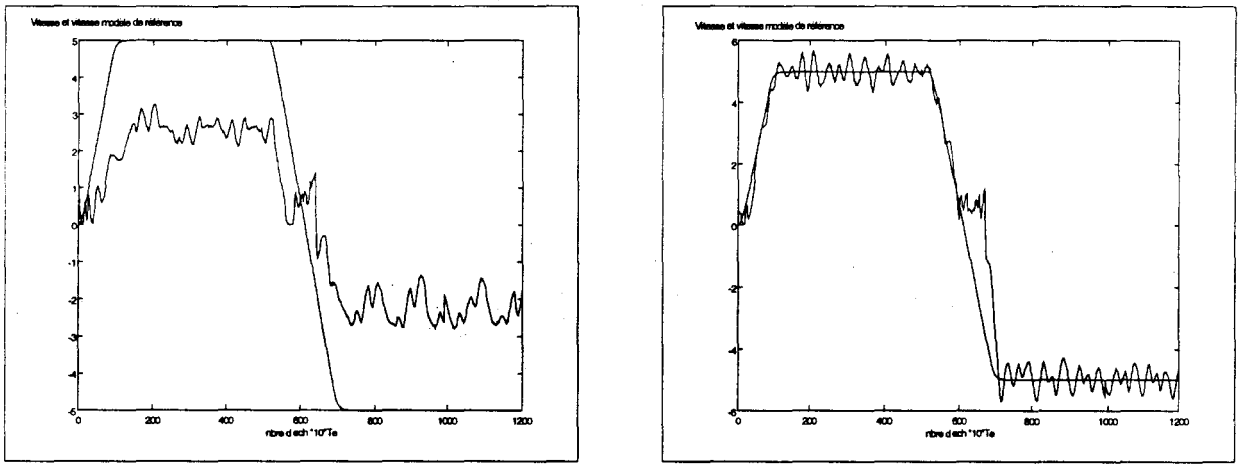
Fig IV.7.6 : Schéma d'adaptation selon un modèle de référence

7.4 Résultats expérimentaux

Dans le cadre des expérimentations le schéma de d'adaptation employé est celui de la figure IV.7.6 dont le modèle de référence est linéaire et peut être régi par une équation récurrente du premier ordre :

$$Y^*(k) = -a_m Y^*(k-1) + b_m Y_{ref} \qquad (4.39)$$

Les résultats ci dessous sont tous expérimentaux, issus d'un banc de tests. Les graphes montrent les performances d'une commande neuronale respectivement non adaptative figure IV.7.7.a et adaptative IV.7.7.b, implantées sur DSP et dont le but est de commander une machine asynchrone entraînant une charge mécanique dont le frottement sec est six fois plus important que celui appris initialement. Ils montrent également la sensibilité par rapport au bruit de la commande neuronale totale.



a) *b)*

Fig IV.7.7 : Résultat d'une commande non adaptative a) et adaptative b) dans le cas où la charge a varié

7.5 Choix du coefficient d'adaptation

La loi de mise à jour des coefficients du réseau nécessite l'emploi du coefficient μ' qui permet d'augmenter ou diminuer la vitesse d'adaptation. Pour des raisons de stabilité ce coefficient doit être pris positif, sinon la mise à jour des coefficients va se faire dans le sens opposé à la minimisation de l'erreur, autrement dit si μ est négatif on va maximiser l'erreur.

Les résultats des figures IV.7.8 ont tous été obtenus dans les mêmes conditions c'est à dire que l'adaptation en ligne a pour but de réduire uniquement l'erreur statique inhérente à la

commande neuronale totale, la charge réelle étant celle apprise par le réseau. Ces figures montrent à quel point la valeur du coefficient d'adaptation joue un rôle important dans la vitesse de convergence du processus réel par rapport au modèle de référence. Nous pouvons voir sur la figure IV.7.8.c que l'écart est complètement annulé en dépit, il est vrai, de fortes oscillations après le transitoire. Cette remarque nous indique que plus le coefficient d'adaptation est important plus l'erreur entre le processus réel et le modèle de référence est rapidement annulée, mais plus le système ainsi commandé sera sensible aux variations du processus et aux changements de consigne. Cette remarque peut conduire notamment à l'utilisation d'un coefficient d'adaptation variable en fonction de l'erreur. Comme le coefficient μ est généralement choisi de façon empirique, il est alors envisageable d'utiliser une variation floue du coefficient d'adaptation.

Dans le cas d'un processus non linéaire et constamment en évolution au cours du temps (par conséquent non stationnaire), l'usage d'un réseau de neurones non linéaire n'est dans ce cas plus très astucieux. En effet si le correcteur doit être modifié en permanence, rien ne sert de mémoriser la caractéristique de la courbe. Une méthode consiste à utiliser un modèle linéaire tangent au processus et à faire varier cet hyperplan en fonction des erreurs mesurées et prédites entre la trajectoire du processus et du modèle [CANART 97]. Cette méthode est efficace mais présente l'inconvénient de nécessiter des inversions de matrices en ligne, ce qui requiert un temps de calcul assez long. Elle n'est donc pas encore très bien adaptée à la commande de systèmes rapides comme les systèmes électriques.

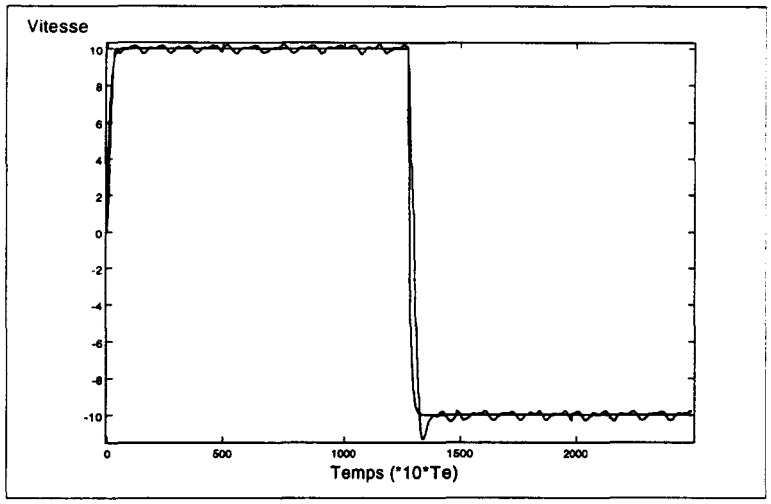


Fig IV.7.8.a : μ=.0001

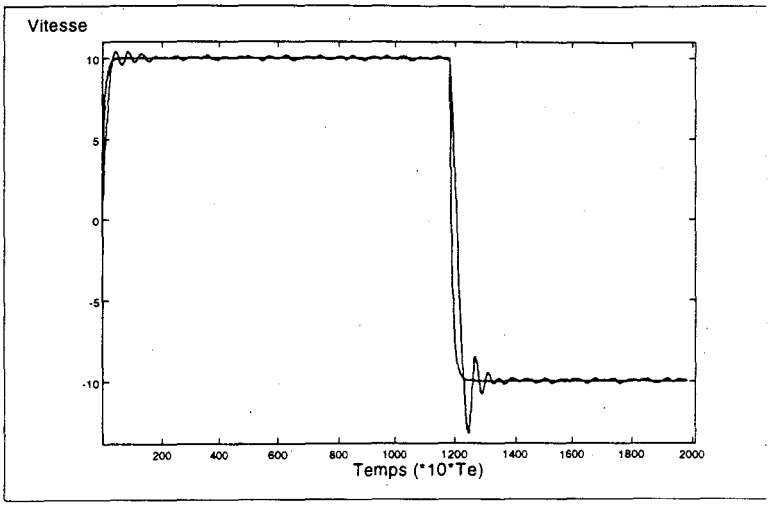


Fig IV.7.8.b : μ=0.0005

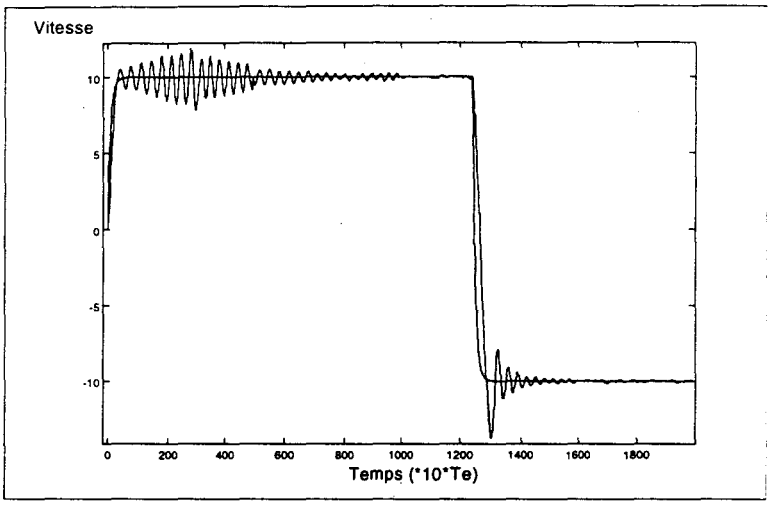


Fig IV.7.8.c : μ=0.001

Fig IV.7.8 : Influence du coefficient d'adaptation

8 Conclusion

Nous avons proposé dans ce chapitre différentes méthodes pour utiliser les réseaux de neurones en commande. L'idée principale restant l'utilisation des réseaux neuronaux en vue d'établir un modèle inverse global du processus ou d'une partie du processus utilisable directement dans la commande. La nécessité d'inverser des causalités pour générer les lois de commande impose la génération de dérivées successives des états. Ceci limite l'utilisation systématique du modèle inverse en tant que correcteur. Cette contrainte restreint une utilisation des réseaux neuronaux aux relations du premier ordre.

Ces remarques sont valables pour les systèmes linéaires et non linéaires. En ce qui concerne l'étude de la stabilité des différentes commandes, celle-ci s'avère empirique

puisque l'utilisation des réseaux neuronaux empêche d'évaluer la robustesse des commandes proposées étant donné que la modélisation neuronale masque à la fois les équations et les valeurs des paramètres du système.

Néanmoins, les bonnes performances des commandes neuronales laissent envisager une application systématique des réseaux neuronaux dans tous les types de corrections. De plus, ces solutions sont capables de prendre en compte toutes sortes de non linéarités et on peut supposer qu'un phénomène rajoutant d'autres non linéarités sera pris en compte intrinsèquement dans le modèle. Par conséquent, il pourrait être judicieux d'utiliser des réseaux de neurones pour remplacer les correcteurs traditionnels de courant et de flux au sein d'une commande vectorielle d'une machine asynchrone. En effet, on peut espérer que l'information de mauvaise orientation du flux soit contenue dans les mesures de tensions et de courants.

CONCLUSION GENERALE

Afin de prendre en compte les nombreux aspects non linéaires inhérents aux systèmes électrotechniques, il est nécessaire de recourir à des modèles de simulation et de commande, qui les prennent en compte de ces non linéarités. Après avoir fait l'inventaire des divers principes de modélisation, nous avons montré qu'il était possible de considérer le réseau artificiel de neurones comme un estimateur universel, les autres techniques étant considérées comme des cas particuliers de ces derniers. Nous avons également discuté au cours du premier chapitre, de la nécessité des commandes non linéaires et, après avoir dressé le constat qu'aucune panacée n'existait dans ce domaine, nous avons alors proposé l'utilisation des réseaux artificiels de neurones comme solution de commande non linéaire. Nous avons donc investi nos efforts dans la détermination d'une commande à partir d'un modèle neuronal, évitant ainsi de recourir à une identification paramétrique du processus à contrôler.

Les principes de base des réseaux de neurones ainsi que les différentes architectures ont été exposées au second chapitre. Le principe d'apprentissage qui est le fondement même des réseaux neuronaux et qui permet l'approche de n'importe quelle caractéristique y a été présenté; différentes techniques d'apprentissage ont été évoquées et ont permis de mettre en lumière le coût important en temps de calcul. Pour pallier cet inconvénient majeur nous avons proposé une méthode dite d'initialisation basée sur une régression non linéaire. Celle-ci permet, entre autre, d'optimiser la structure mathématique du réseau neuronal en fonction de la caractéristique à apprendre, ce qui permet de réduire considérablement les temps d'apprentissage et d'augmenter la qualité de modélisation.

Cette technique démontrée et formalisée au second chapitre a été mise en pratique sur différentes applications. Nous avons tout d'abord testé la méthode d'apprentissage sur la modélisation d'inductances saturables, ce qui a permis de soulever les problèmes liés au coût et à la qualité des modèles. Au vu des bonnes performances, nous l'avons appliquée sur la modélisation de cycles majeurs d'hystérésis; là encore, nous avons insisté sur l'optimisation de la structure neuronale. Les résultats tout à fait encourageants nous permettent d'envisager la simulation complète d'une machine asynchrone avec prise en compte du phénomène de saturation magnétique. Cette méthode nécessitera vraisemblablement au préalable une simulation plus fine par éléments finis, afin d'acquérir des échantillons des caractéristiques

magnétiques de la machine. Une fois la phase d'apprentissage effectuée, le modèle neuronal de la machine devrait nous permettre de simuler les régimes dynamiques de la machine avec un gain de temps par rapport aux méthodes par éléments finis, tout en ayant une qualité de modélisation comparable. Nous avons également appliqué ces concepts à la caractérisation de charges mécaniques en vue d'utiliser ultérieurement le modèle neuronal en commande.

Le dernier chapitre a exposé comment à partir d'un modèle neuronal il est possible de commander un processus qu'il soit linéaire ou non. Deux commandes principales ont été proposées : commande par anticipation et commande avec compensation. La stabilité de ces commandes a été démontrée. Afin de remédier aux erreurs d 'apprentissage qui entraînent des erreurs de convergence, nous avons alors proposé une commande adaptative.

Nous avons montré tout au long de ce travail quels étaient les avantages et les inconvénients des réseaux artificiels de neurones tant en commande qu'en modélisation des systèmes électrotechniques et électromécaniques non linéaires. Au vu des performances obtenues liées à l'optimisation des structures neuronales, il peut alors être intéressant d'établir une bibliothèque de structures neuronales afin de répondre aux différents besoins de modélisation : modélisation des couples de charges discontinus, des jeux au sein des entraînements mécaniques, des 'durs' mécaniques,.., de même, pour les phénomènes électromagnétiques. A partir des modèles de connaissance a priori, en utilisant cette bibliothèque de modèles il serait alors possible de concevoir une commande idéale dans les limites d'utilisation des réseaux neuronaux en contrôle.

Puisque la commande par réseaux de neurones linéarise les systèmes non linéaires, celle-ci peut également permettre la détection de défauts par simple comparaison de la sortie d'un processus idéalement linéarisé par rapport à un modèle de référence lui même linéaire. Toute erreur indiquera la présence d'un changement dans le processus, provenant par exemple d'un défaut de fonctionnement.

Si l'intelligence artificielle semble prometteuse, nous devons rester modeste quant aux performances; le cerveau humain n'a pas encore révélé tous ses mécanismes et étant donné que l'homme s'inspire de sa propre nature, il reste encore bon nombre de découvertes à effectuer en connexionisme; par exemple, nous possédons à ce jour un grand nombre de

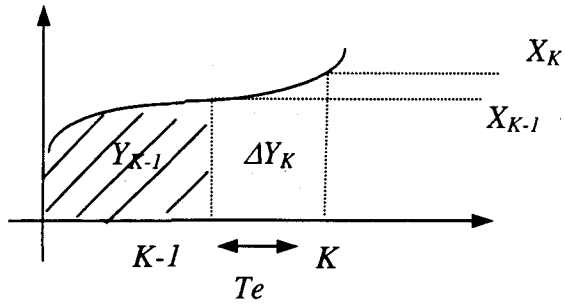
techniques pour mémoriser un fait, une caractéristique,..., ce que nous avons qualifié par 'apprentissage'. Néanmoins il est encore extrêmement difficile de formaliser la motivation de cet apprentissage et c'est pourquoi un expert est encore nécessaire pour évaluer ce qui doit être appris et comment il doit être appris. Afin de bien mettre en valeur la qualité des résultats que nous avons obtenus avec de faibles quantités de neurones, nous pouvons oser une comparaison entre le cerveau humain et le microprocesseur. Le microprocesseur Pentium mis en service en 1993 par Intel possède 3.1 millions de transistors et consomme une puissance de 16 watts [INTEL 93]. Sachant que le réseau humain possède 10^{11} neurones reliés chacun à 10^4 voisins [KANDEL 87] et, en considérant qu'un transistor puisse émuler une connexion, il faudrait alors 3.10^8 microprocesseurs pour atteindre la complexité du cerveau humain dont la consommation frôlerait les 5 gigawatts. Cette ultime remarque nous permettra de philosopher quant à la situation dominante ou dominée de l'Homme vis à vis de sa technologie.

ANNEXES

Approximation de TUSTIN

On montre que z est l'opérateur d'intégration par la méthode des trapèzes.

Démonstration :



Méthode des trapèzes

$$Y_k = Y_{k-1} + \Delta Y_k$$

$$Y_k = Y_{k-1} + \left(\frac{x_k + x_{k-1}}{2}\right) Te$$

en posant

$$Y_{k-1} = z^{-1} Y_k$$

$$x_{k-1} = z^{-1} x_k$$

il vient

$$Y_k (1 - z^{-1}) = \frac{x_k}{2}\left(1 + z^{-1}\right) Te$$

d'où $\frac{Y_k}{x_k} = \frac{Te}{2}\frac{1 + z^{-1}}{1 - z^{-1}} \Leftrightarrow \frac{1}{s}$

Bibliographie

[ABDI 94] 'Les Réseaux de neurones' collection sciences et technologies de la connaissance', H. ABDI édition Plug 1994

[AUSSEM 95] 'Théorie et applications des réseaux de neurones récurrents et dynamiques à la prédiction, à la modélisation et au contrôle adaptatif des processus dynamiques', A. AUSSEM, Thèse en informatique, Université René Descartes, Paris V, 1995

[BAUM 89] 'What size net gives valid generalization ?', E.B. Baum, D. Haussler, Neural Computation, vol 1, pp151-160, Spring 1989

[BONNEMOY 91] 'Simulated annealing method : global optimisation in Rn', C. BONNEMOY, S. B. HAMMA, APII, vol 25, pp477-496, 1991

[BORNE 92] 'Modélisation et identification des processus', P. BORNE, G. DAUPHIN-TANGUY, J.P. RICHARD, F. ROTELLA, I. ZAMBETTAKIS, tomes 1 et 2, Editions Technip, 1992

[BORNE 93] 'Analyse et régulation des processus industriels', P. BORNE, G. DAUPHIN, J.P RICHARD, F. ROTELLA, tome 1, Editions Technip, 1993
TANGUY, J.P. RICHARD, F. ROTELLA, I. ZAMBETTAKIS, tomes 1 et 2, Editions Technip, 1992

[BÜHLER 86] 'Réglage par mode de glissement', H. BÜHLER, Presses Polytechniques Romandes, 1986

[CANART 97] 'Contribution à la commande adaptative et prédictive', R. CANART, Thèse de doctorat en automatique et informatique industrielle, USTL, LAIL, 1997

[CAUMONT 97] 'Détermination de l'état de charge d'une batterie Pb-Acide en utilisation véhicule électrique', O. CAUMONT, Thèse de doctorat de génie électrique, USTL, L2EP Lille 1997

[CHESTER 90] 'Why two hidden layers are better than one', D.L. Chester, Proceedings of the IJCNN-90, 1990, Wachington, DC, pp 265-8

[CONNOR 94] 'Recurrent neural networks and robust time series prediction', CONNOR J.T., MARTIN R.D., ATLAS L.E., IEEE Trans. Neural Networks, vol 5, no. 2, 1994

[COX 97] 'La logique floue : Pour les affaires et l'industrie', Earl D. Cox, International Thomson Publishing 1997

[DEBRUYNE 96] 'Characterisation and modelling of hysteresis phenomenon' H. DEBRUYNE, S. CLENET, F. PIRIOU, Electrimacs 96, Vol 1, pp 327-331

[DELMOTTE F 97] 'Analyse multi-modèles', F. DELMOTTE, Thèse de doctorat en automatique et informatique industrielle, Université des Sciences et Techniques de Lille, LAIL, 1997

[DEGOBERT 97] , 'Formalisme pour la commande des machines électriques alimentées par convertisseurs statiques. Application à la commande numérique d'un ensemble machine asynchrone-commutateur de courant', Ph. DEGOBERT, Thèse de l'U.S.T.L, juin 1997, L2EP Lille

[DUBOIS 95] 'Utilisation de la logique floue dans la commande des systèmes complexes', L. DUBOIS, Thèse de doctorat en automatique et informatique industrielle, Université des Sciences et Techniques de Lille, LAIL, 1995

[DROESBEKE 97] 'Plans d'expériences : Applications à l'entreprise' J.J DROESBEKE, J. FINE, G. SAPORTA, Editions Technip 1997

[FLANDRIN 93] 'Temps - fréquence',P. FLANDRIN, Traité des Nouvelles Technologies, Hermes, 1993

[FORGEZ 98A] 'Comparison between neural compensation and internal model control for induction machine drive', I. STEFAN, C. FORGEZ, B. LEMAIRE SEMAIL, J.P. HAUTIER : ICEM 4-6 sep 1998, Istanbul (Turquie), vol 2, pp 1330-1334

[FORGEZ 98B] 'Adaptive speed control of an AC machine coupled to a non linear load torque using neural network', C. FORGEZ, B. LEMAIRE SEMAIL, J.P. HAUTIER : ICEM 4-6 sep 1998, Istanbul (Turquie), vol 1 , pp 608-613

[FORGEZ 96] 'Induction machine control with neural network to consider non linear loads', C. FORGEZ, B. LEMAIRE SEMAIL, J.P. HAUTIER : ELECTRIMACS 17-19 sep 1996, St Nazaire, vol 2, pp 375-380

[FRANÇOIS 96] 'Orthogonal Considerations in the Design of Neural Networks for Function Approximation',B. FRANÇOIS, Mathematics and Computers in Simulation 41, Elsevier, July 1996, in press, pp 95-108

[GILLON 97] 'Modélisation et optimisation par plans d'expériences d'un moteur à commutations électroniques', F. GILLON, Thèse de doctorat de génie électrique, Université des Sciences et Techniques de Lille, L2EP, 1997

[HARTMAN 90] 'Layered neural networks with gaussian hidden units as universal approximations', HARTMAN E.J., KEELER J.D. and KOWALSKI J.M, Neural Computation, vol 2, pp. 210-5, 1990

[HAUTIER 96] 'Le graphe Informationnel Causal, outil de modélisation et de synthèse des commandes de processus électromécaniques' J.P. HAUTIER, J. FAUCHER, Bulletin de l'Union des Physiciens n°785, cahier de l'Enseignement Supérieur, Vol 90, Juin 1996, pp 167-189

[HORNIK 88] 'Multilayer feedforward networks are universal approximators', K. HORNIK, M. Stinchcombe, H.White, Dept. Economics University of California, San Diego, CA, Discussion pap., Dept. Economics, juin 1988

[INTEL 93] 'Intel Pentium Processor User's Manual', Intel Corporation Literature Sales Mt Prospects, IL, 1993

[JODOUIN 94] 'Les Réseaux neuromimétiques', J. F. JODOUIN collection informatique edition Hermes 1994

[KANDEL 87] 'Principles of Neural Science', E.R KANDEL, J.H SCHWARTZ, New York, Elsevier, North Holland, 1981

[LEONARD 85] 'Control of electrical drives', W. LEONARD, Springer Verlag 1985

[LEMAIRE-SEMAIL 97]'Contribution à la modélisation et à la commande des machines asynchrones', B. LEMAIRE-SEMAIL, Thèse d'habilitation, U.S.T.L, décembre 1997, L2EP Lille.

[LOUIS 91] 'Modélisation des machines à courant alternatif au sens du premier harmonique', J.P. LOUIS, polycopié de cours de DEA, ENS Cachan, 1991

[MARROCCO 77] 'Analyse numérique des problèmes en électrotechnique', A. MARROCCO, Ann. Sc. Math. Québec, 1977, pp 271-296

[OSTOVIC 89] 'Dynamics of Saturated Electric Machines', V.OSTOVIC, Springer-Verlag, 1989

[PARK 91] 'Universal approximation using radial basis function networks', PARK J. and SANDBERG I.W., Neural Computation, vol 3, pp. 246-257, 1991

[MASSON 53] 'Feedback Theory - Some properties of signal flows graphs', S.J Mason, Proceeding IRE 1953

[PATTERSON 96] 'Artificial Neural Networks, Theory and Applications', Dan W. PATTERSON, Prentice Hall 1996

[PIERLOT 96] 'Application des réseaux neuronaux à la commande en vitesse d'une charge mécanique entraînée par machine asynchrone', N. PIERLOT, Thèse de doctorat de génie électrique, U.S.T.L, L2EP Lille, décembre 1996.

[PINEDA 87] 'Generalisation of back-propagation to recurent neural networks', PINEDA F.J, Physical Review Letters 59, pp 2229-2232, 1987

[PINEDA 89] 'Recurrent back-propagation and the dynamical approach to adaptive neural computation', PINEDA F.J, Neural Computation 1, pp 161-172, 1989

[RICHALET 91] 'Pratique de l'identification', J. RICHALET, Hermès Paris 1991

[ROSENBLATT 62]'Principles of neurodynamics', F. ROSENBLATT, New York, Spartan, 1962

[RUMELHART 86] 'Learning internal representations by error propagattion', RUMELHART D.E., HINTON G.E., WILLIAMS R.J., Parallel Distributed Processing Explorations in the Microstructure of Cognition, Eds. RUMELHART D.E and MCCLELLAND J.L., Cambridge, MA : MIT Press, Bradfords Books, vol. 1, pp.318-362, 1986.

[SEGUIER 77] 'Electrotechnique industrielle', G.SEGUIER, F.NOTELET, Technique et Documentation, Paris 1977

[SILVESTER 70] 'Finite element solution of saturable magnetic fields problems', P.SILVESTER, M.V.K. CHARI, IEEE Trans. P.A.S 89, pp 1642-1651, 1970

[TORRE 90] 'Preisach modeling and reversible magnetization', E. DELLA TORRE,J. OTI, G. KADAR, IEEE Trans. On Magnetics, Vol 26, pp 3052-3058, 1990.

[WANG 92a] 'Fuzzy basis functions, universal approximation, and orthogonal least-squares learning', L.X WANG, MENDEL, IEEE Transactions on Systems, Man and Cybernetics, Vol 3 n°5 pp 807-814, 1992

[WANG 92b] 'Generating fuzzy rules by learning from examples', L.X WANG, J. MENDEL, IEEE Transactions on Systems, Man and Cybernetics, Vol 22 pp 1414-1427, 1992