REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE BATNA
FACULTE DES SCIENCES DE L'INGENIEUR
DEPARTEMENT D'ELECTRONIQUE

MEMOIRE

Présenté Pour L'obtention Du Diplôme De
Magister En Microélectronique
Option : IC Design

THEME

Conception D'un Capteur De Pression Intelligent

Par Hafiane Mohamed Lamine
Electro_lamine@yahoo.fr

Soutenue le 28/12/2005 devant le jury :

Pr. Farida Hobar	Président	**Université de Constantine**
Dr. Dibi Zohir	Rapporteur	**Université de Batna**
MC. Benhaya Ablelhamid	Examinateur	**Université de Batna**
MC. Ounissi Abdelhamid	Examinateur	**Université de Batna**
MC. Mahamdi Ramdane	Examinateur	**Université de Batna**

2005

AVANT PROPOS

Ce travail a été effectué au sein du Laboratoire d'Electronique Avancée (LEA) de la Faculté des Sciences de l'Ingénieur de l'Université de Batna. A cet effet, je tiens à remercier son Directeur le professeur ***Bouguechal Nour-Eddine*** *qui m'a accueilli dans ce laboratoire.*

Mes vifs remerciements vont à mon encadreur le docteur ***Dibi Zohir*** *pour le sujet, la confiance qu'il ma témoigné, le suivi constant de ce travail. Je lui exprime toute ma gratitude et mon estime.*

J'exprime également mes remerciements à madame ***Hobbar Farida****, Professeur au département d'Electronique de la Faculté des Sciences de l'Ingénieur de l'université de Constantine, qui ma fait l'honneur de s'intéresser à ce travail et de présider son jury.*

Que messieurs ***Benhaya Abdelhamid, Ounissi Abdelhamid*** *et* ***Mahamedi Ramdane,*** *tous maître de conférences au département d'Electronique de l'Université de Batna, trouvent ici toute ma reconnaissance de m'avoir honorer en acceptant d'examiner ce modeste travail.*

Je tiens également à présenter ma profonde gratitude à monsieur ***Djeffal Fayçal*** *doctorant au département d'Electronique de l'université de Batna et membre du laboratoire (LEA), pour les discussions fructueuses que nous avons eu.*

Résumé

A travers ce mémoire, nous avons montré que les réseaux de neurones artificiels (ANNs) se substituent à un capteur de pression capacitif (CPS) du fait qu'ils reproduisent fidèlement son comportement dans un milieu dynamique. En se basant sur des résultats réalisés par des chercheurs au laboratoire LAAS (France) sur les capteurs miniatures type CPS, nous avons formé deux bases de données, la première pour l'apprentissage du réseaux par l'algorithme de la rétropropagation des erreurs et la deuxième pour le test et la validation du modèle. Le modèle à base des réseaux de neurones du CPS (modèle ANN), ainsi obtenu, a été implanté sur le simulateur SPICE, ce qui nous a permis de simuler son fonctionnement sur un environnement électrique et par conséquent évaluer ces performances. Nous avons également présenté, les réseaux de neurones comme un composant de correction (modèle inverse) pour le capteur CPS placé dans un environnement dynamique. Ce composant, synthétisé en langage VHDL, permet de réduire les erreurs de mesure dûes aux effets de température, de non linéarité et d'hystérésis. Il s'agit donc d'associer à ce capteur une structure neuronale intelligente (en code VHDL) permettant d'effectuer des corrections et des compensations en temps réel afin d'obtenir une réponse précise sur une large gamme de pression (0 à 6 bar) et une plage de température entre -10°C et 90°C.

Mots clés : Capteur de pression capacitif (CPS), réseaux de neurones artificiels (ANNs), perceptron multicouche (MLP), non linéarité, dérive en température, hystérésis, capteur intelligent, modélisation directe et inverse, SPICE, VHDL.

Liste des symboles

ANN : Réseaux de neurones artificiels (Artificial Neural Network)

ANN-model: Modèle à base des réseaux de neurones

INV_ANN : Modèle inverse à base des réseaux de neurones

MLP : Multicouche perceptron (multilayer perceptron)

CPS : Capteur de pression capacitif (capacitive pressure sensor)

SCI : Interface de capacité commutée (switched capacitor interface)

FS : Réponse pleine échelle (Full Scale)

P : Pression

P^- : Pression précédente

C : Capacité

T : Température

NL : Non linéarité

H_P : Hystérésis en pression

V_P : Tension de sortie du CPS&SCI

V_P- : Tension précédente

Z^{-1} : Délai temporaire

S_P : Tension de sortie d'INV_ANN

EQM : Erreur quadratique moyenne

BP : Algorithme de la rétropropagation des erreurs (back propagation)

Table des matières

Introduction

Dans plusieurs applications relatives aux chaînes de mesure et de contrôle on fait assez souvent appel aux capteurs de pression capacitifs (CPS). Ces derniers présentent l'avantage d'une grande résolution, une faible consommation électrique, une grande précision et une possibilité d'intégration. Les inconvénients par contre résident dans la non linéarité de leur réponse, la dépendance de la température, en plus, du phénomène d'hystérésis. L'exploitation de ce type de capteur nécessite alors, l'association d'un circuit de correction et de compensation en temps réel afin d'obtenir une réponse linéaire, complètement indépendante de la température et tient compte de l'effet d'hystérésis. La structure : capteur, circuit de conditionnement et circuit de correction, fait appel au concept du capteur intelligent. Dans la littérature, on donne le nom « capteur intelligent » à un composant qui regroupe plusieurs étages complémentaires dans le but est de fournir une information précise sur une grandeur physique à mesurée (pression dans notre cas), donc directement exploitable. En effet, le développement considérable de la capacité d'intégration des composants microélectroniques et la miniaturisation des capteurs à permis d'assembler sur un seul chip, les différents étages constituant le capteur intelligent, à savoir le capteur de pression lui-même, le circuit de conditionnement et le circuit de correction du signal de sortie, ce qui est largement étudié dans [1,2].

Un capteur intelligent peut être défini selon deux points de vue, fonctionnel et technologique :

- Point de vue fonctionnel :

  Les capteurs intelligents sont des dispositifs capables de détecter, de mesurer, de traduire et de traiter les données collectées en vue de les communiquer à d'autres organes du système dans lequel ils sont intégrés.

- Point de vue technologique :

  Le capteur intelligent correspond principalement à l'intégration, dans le corps du capteur, d'un organe de calcul interne (microprocesseur, micro-contrôleur), d'un système de conditionnement du signal (programmable ou contrôlé) et d'une interface de communication.

Dans ce mémoire, nous présentons les réseaux de neurones artificiels (ANNs) comme un modèle de correction (à matérialisé sur un circuit FPGA ou ASIC), afin de corriger la réponse du capteur de pression capacitif (CPS) placé dans un environnement dynamique. En se basant sur les avantages qu'offrent les ANNs à savoir, une grande adaptation aux divers problèmes, le développement des modèles ANNs nécessite moins de données par rapport au autres méthodes statistiques classiques (moindre carré, interpolation polynomial.. etc.), et leur capacité d'approximation pour toutes fonctions mathématiques, en plus de l'aspect de généralisation. Dans cette optique, nous avons noté plusieurs travaux qui traitent l'utilisation des ANN pour la correction de la réponse du CPS : [3, 4, 5, 6], ces travaux donnent de bon résultats, cependant, l'effet d'hystérésis n'est pas pris en compte.

Notre travail est divisé en deux grandes parties : la première consiste au développement d'un modèle à base des réseaux de neurones (ANNs) pour le CPS et l'insérer sous forme d'un composant dans la bibliothèque du simulateur électrique SPICE. Ce modèle exprime le comportement du capteur dans un environnement dynamique, il remplace aussi le modèle analytique qui est limité à des conditions relativement moins dynamique.

La seconde partie est consacrée au développement d'un deuxième modèle à base des réseaux de neurones et à son implantation sur un circuit FPGA sous forme d'un code écrit en langage VHDL. Le but assigné à ce composant est la correction du signal de sortie du capteur. Il est à noter que la différence entre les deux modèles développés est que : le premier est une modélisation comportemental du capteur de pression capacitif (CPS), par contre le deuxième modèle joue le rôle d'un composant de correction matérialisé par un code en langage de description VHDL sur un circuit FPGA ou ASIC (figure 1).

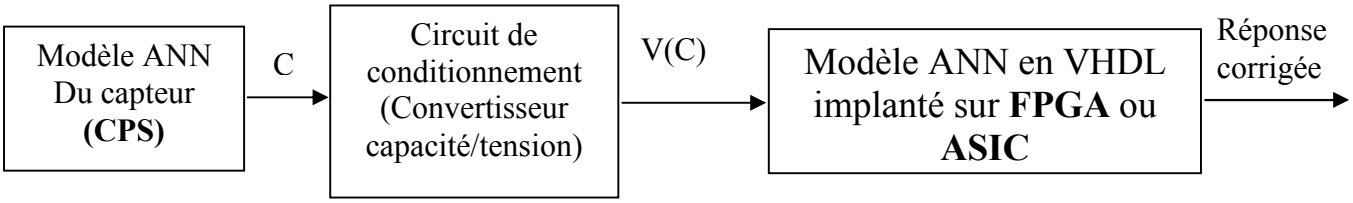


Figure 1 : Structure du capteur intelligent simulé sur l'environnement SPICE

Afin de remplir la tache prévue, nous avons fait appel à deux outils de développement informatiques qui sont : le MATLAB et le simulateur SPICE.

MATLAB : est utilisé pour le la mise au point des modèles à base des réseaux de neurones (l'apprentissage, l'optimisation), en plus de tout les calculs mathématiques.

Simulateur SPICE : une fois que les modèles ANNs sont obtenus (nombre de neurones, nombre de couches et les poids des connexions) par le MATLAB, on procède à l'implantation de ces derniers dans le simulateur SPICE qui est utilisé comme un environnement de simulation pour le capteur intelligent (capteur, circuit de correction).

Dans le premier chapitre de ce mémoire, nous introduisons des notions théoriques sur les réseaux de neurones et nous détaillons d'avantage les concepts utilisés au cours de notre travail.

Nous décrirons dans le deuxième chapitre le fonctionnement du capteur de pression capacitif (CPS), sa structure technologique, en suite nous nous attachons à étudier son modèle analytique et sa limitation dans un milieu dynamique.

Le troisième chapitre, est consacré au développement d'un modèle neuronal qui permet de reproduire fidèlement le comportement du CPS, et de montrer l'efficacité de ce modèle par rapport au modèle analytique existant dans la mesure où le capteur CPS est placé dans un environnement dynamique. En plus, nous proposons une implantation du modèle ANN obtenu sur le simulateur SPICE, avec les résultats de simulation obtenus.

Le développement d'un correcteur du signal de sortie à base des réseaux de neurones et sa matérialisation sur un circuit FPGA en utilisant le langage VHDL, ont fait l'objet du dernier chapitre.

Introduction

L'objectif de ce chapitre est multiple : il s'agit tout d'abord de rappeler les définitions fondamentales relatives aux réseaux de neurones ainsi que leur propriétés mathématiques. Nous décrirons ensuite les principaux types des réseaux de neurones. Finalement nous nous attacherons à détailler le type de réseaux de neurones utilisé dans notre thèse (MLP), et plus particulièrement ces propriétés et sa mise en oeuvre. Au long de ce chapitre nous avons cherché a éclairé les concepts généraux des réseaux de neurones et détailler d'avantage les notions auquels nous avons fait appel pour élaboré notre travail.

I. Les neurones biologiques et les neurones artificiels

Les réseaux de neurones artificiels (formels) sont à l'origine, une tentative de modélisation mathématique du cerveau humain. Les premiers travaux datent de 1943 et sont l'oeuvre de MM. *Mac Culloch et Pitts* [7]. Ils présentent un modèle assez simple pour les neurones et explorent les possibilités de ce modèle. L'idée principale des réseaux de neurones artificiels est de donner une unité simple, un neurone, qui est capable de réaliser quelques calculs élémentaires. On relie ensuite entre elles un nombre important de ces unités et on essaye de déterminer la puissance de calcul du réseau ainsi obtenu.

Le modèle biologique illustré dans la figure I.1 décrit un modèle simple du neurone biologique qui a servi à la mise en place des premiers neurones formels.

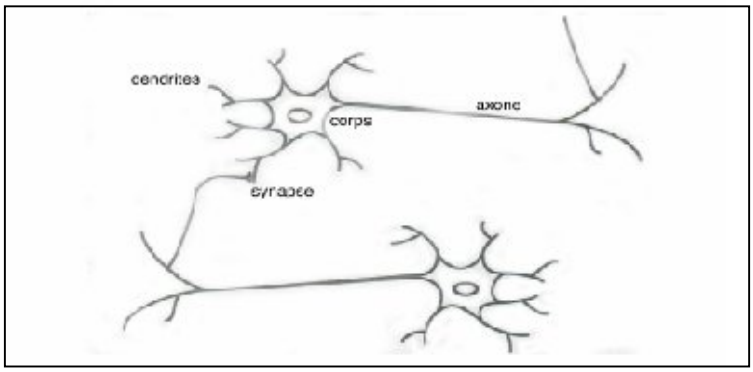


Figure I.1 : Le neurone biologique

Dans le cerveau, les neurones sont reliés entre eux par l'intermédiaire d'axones et de dendrites. En première approche, on peut considérer que ces sortes de filaments sont conductrices d'électricité et peuvent ainsi véhiculer des messages depuis un neurone vers un autre. Les dendrites représentent les entrées du neurone et son axone sa sortie.

Un neurone émet un signal en fonction des signaux qui lui proviennent des autres neurones. On observe en fait au niveau d'un neurone, une intégration des signaux reçus au cours du

temps, c'est-à-dire une sorte de sommations des signaux. En général, quand la somme dépasse un certain seuil, le neurone émet à son tour un signal électrique [7].

La notion de synapse explique la transmission des signaux entre un axone et une dendrite. Au niveau de la jonction. Quand un signal arrive au niveau de la synapse, un signal électrique est émis de l'autre côté et on a donc une transmission. En fait, suivant le type de la synapse, l'activité d'un neurone peut renforcer ou diminuer l'activité de ces voisins. On parle ainsi de synapse excitatrice ou inhibitrice [7].

Un neurone formel est un processeur très simple, simulé sur ordinateur ou réalisé sur circuit intégré, imitant grossièrement la structure et le fonctionnement d'un neurone biologique. Le plus simple concept d'un neurone est un automate binaire qui réalise une somme 'S' pondérée de ses entrées et compare cette somme à un seuil 'B_0'.

- Si $S>B_0$ la sortie du neurone vaut +1, le neurone est dit actif
- Si $S< B_0$ la sortie vaut -1, le neurone est dit inactif

Généralement le type de neurones le mieux adaptés aux tâches de traitement du signal ou de classification, est celui dont la sortie n'est pas binaire mais une fonction algébrique non linéaire, paramétrée, à valeurs bornées.

II. Neurone artificiel élémentaire

La figure I.2 montre la structure d'un neurone artificiel. Chaque neurone reçoit un nombre variable d'entrées en provenance des neurones amonts. A chacune de ces entrées est associée un poids ' W_i ' abréviation de Weight (poids en Anglais) représentatif de la force de la connexion, le seuil W_0 peut être envisagé comme le coefficient de pondération de l'entrée X_0, dont la valeur est fixée à 1. Chaque neurone élémentaire est doté d'une fonction de transfert (fonction d'activation) qui donne une sortie unique 'Y', et se ramifie ensuite pour alimenter un nombre variable de neurones avals.

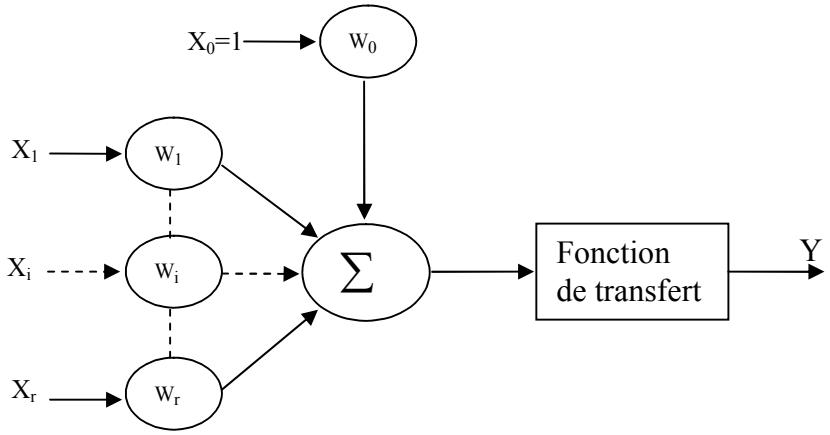


Figure I.2 : Structure d'un neurone

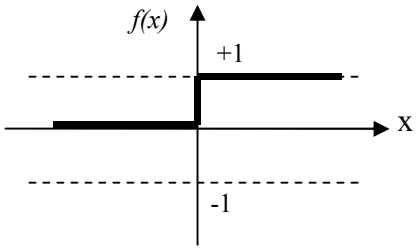
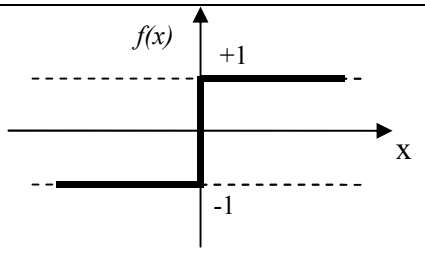
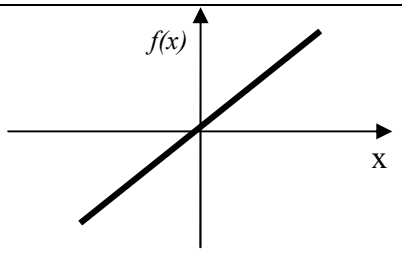
On distingue deux phases : la première est le calcul de la somme pondérée des entrées ' X_i ', à partir de cette valeur, une fonction de transfert ' *f* ' calcule la valeur de l'état du neurone selon l'expression suivante :

$$Y = f(W_0 + \sum_{i=1}^{r} W_i \cdot X_i) \qquad \text{(I.1)}$$

Ou bien :

$$Y = f(\sum_{i=0}^{r} W_i \cdot X_i) \qquad \text{(I.2)}$$

C'est cette valeur qui sera transmise aux neurones avals. Il existe de nombreuses formes possibles pour la fonction de transfert. Les plus courantes sont présentées sur le tableau I.1, avec leurs équations mathématiques. On remarquera qu'à la différence des neurones biologiques dont l'état est binaire, la plupart des fonctions de transfert sont contenue et offrant une infinité de valeurs comprises dans l'intervalle [0, +1] ou [-1, +1].

Catégorie	Type	Equation	Allure
Seuil	Binaire (fonction de Heaviside)	$f(x) \begin{cases} 0 \text{ si } x<0 \\ 1 \text{ si } x\geq 0 \end{cases}$	
	Signe	$f(x) \begin{cases} 1 \text{ si } x>0 \\ -1 \text{ si } x\leq 0 \end{cases}$	
Linéaire	Identité	$F(x)=x$	

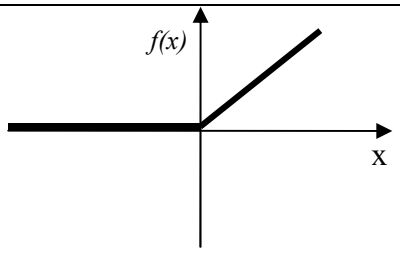
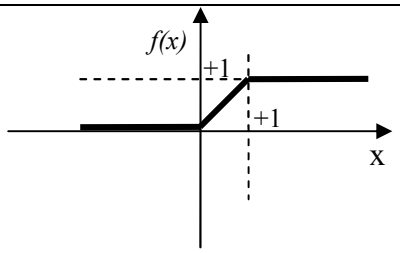
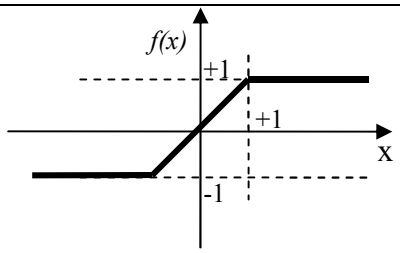
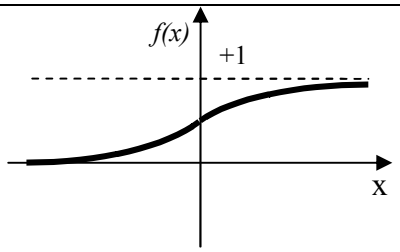
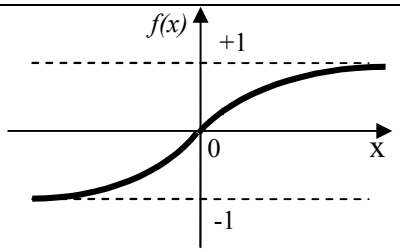
Linéaire	Linéaire positif	$f(x)\begin{cases} 1 \text{ si } x<0 \\ x \text{ si } x\geq 0 \end{cases}$	
	Saturé positif	$f(x)\begin{cases} 0 \text{ si } x < 0 \\ 1 \text{ si } x \geq 1 \\ x \text{ si non} \end{cases}$	
	Saturé symétrique	$f(x)\begin{cases} -1 \text{ si } x \leq -1 \\ 1 \text{ si } x \geq 1 \\ x \text{ si non} \end{cases}$	
Non linéaire	Logistique (sigmoïde)	$f(x)=\frac{1}{1+e^{-X}}$	
	Tan-sigmoïde (tanh)	$f(x)=\frac{2}{1+e^{-X}}-1$	

Tableau I.1 : Les fonctions de transfert

III. Présentation des réseaux de neurones

Un neurone élémentaire est limité en ces applications, en effet, un neurone réalise une simple fonction non linéaire, paramétrée, de ses variables d'entrées. L'intérêt des neurones réside dans la propriété qui résultent de leur association dans une structure, par une certaine logique d'interconnexion, cette structure est appelée : le réseau de neurones ou bien par l'abréviation ANN (Artificiel neural network). Le comportement collectif ainsi obtenu permet de réaliser des fonctions d'ordre supérieur par rapport à la fonction élémentaire réalisé par un neurone.

Dans un tel réseau, les entrées d'un neurone sont soit les entrées du réseau globale, soit les sorties d'autres neurones. Les valeurs des poids du réseau sont, en général, déterminées par une opération dite l'apprentissage [8].

Suivant la logique d'interconnexion choisie, les réseaux de neurones se distinguent en deux grande familles : les réseaux non bouclé (statique) et les réseaux bouclé (dynamique), les figures I.3 et I.4 illustrent le schéma synoptique des deux réseaux respectivement.

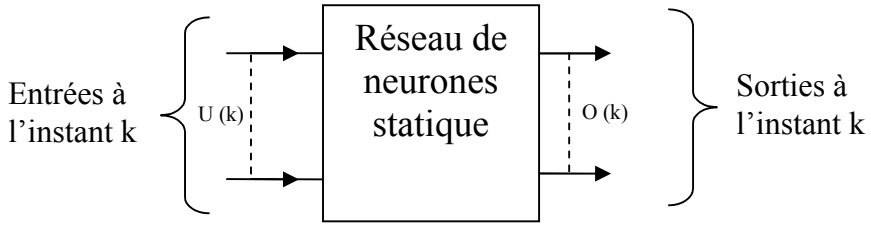


Figure I.3 : Schéma d'un réseau de neurones non bouclé (Statique)

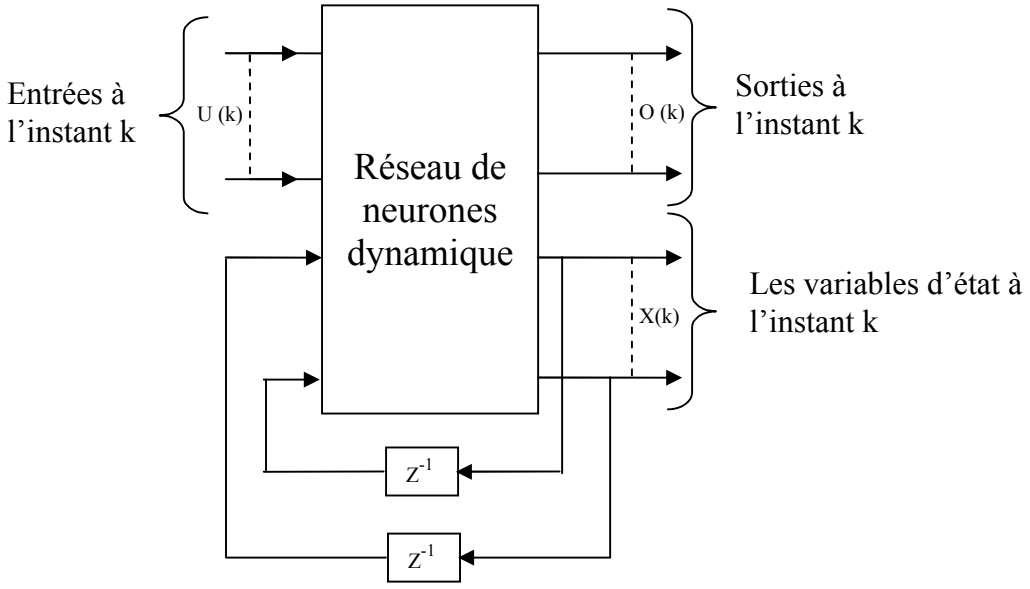


Figure I.4 : Schéma d'un réseau de neurones bouclé (dynamique)

III.1 Les réseaux de neurones bouclé

L'architecture la plus général pour un réseau de neurones est bien les réseaux bouclés, dont le graphe des connexions est cyclique, lorsqu'on se déplace dans le réseau en suivant le sens des connexions, il est possible de trouver au moins un chemin qui revient à son point de départ. La sortie d'un neurone du réseau peut donc être fonction d'elle-même, alors la notion

du temps est explicitement prise en considération [8]. A chaque connexion d'un neurone bouclé est attaché un retard, multiple entier de l'unité de temps choisi (figure I.5).

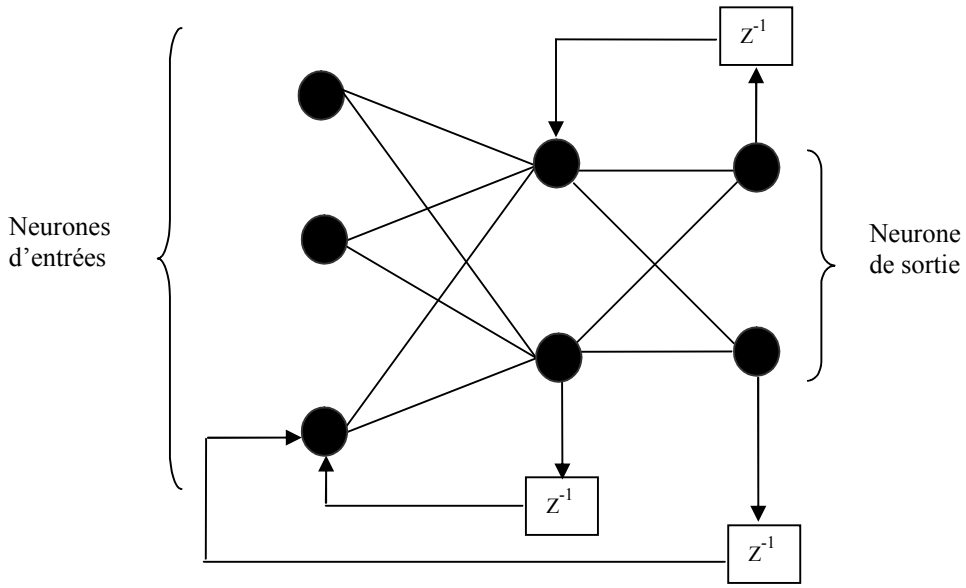


Figure I.5: Exemple illustratif d'interconnexion d'un réseau dynamique

III.2 Les réseaux de neurones non bouclé (statique)

Un réseau de neurones non bouclé réalise une ou plusieurs fonctions algébriques de ses entrées par composition des fonctions réalisées par chacun de ses neurones. Ce réseau est représenté graphiquement par un ensemble de neurones connecté entre eux figure I.6, dans un tel réseau, le flux de l'information circule des entrées vers les sorties sans "retour en arrière", si on se déplace dans le réseau, à partir d'un neurone quelconque, en suivant les connexions, on ne peut pas revenir au neurone de départ [8]. Les neurones qui effectuent le dernier calcul de la composition de fonction sont les neurones de sortie, ceux qui effectuent des calculs intermédiaires sont les neurones cachés.

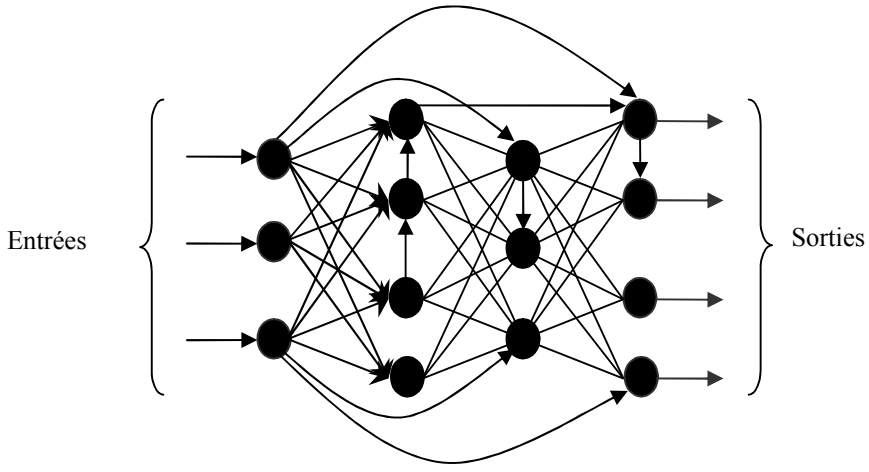


Figure I.6 : Exemple illustratif d'interconnexion d'un réseau statique

Une autre façon d'assembler les neurones entre eux consiste à constituer des couches de neurones en interdisant toute connexion entre neurones de la même couche et aussi les connexions entre les neurones de deux couches non consécutives. Les réseaux de neurones à couches sont aussi appelés perceptrons multicouches ou bien par l'abréviation MLP (Multilayer Perceptron). La figure I.7 montre un exemple d'un MLP composé de : trois neurones d'entrées, une couche cachée composé de 4 neurones et un neurone de sortie. Cette architecture (MLP) est particulièrement utilisée dans notre travail car elle possède des propriétés mathématiques intéressantes, que nous présenterons dans le paragraphe IV.2.

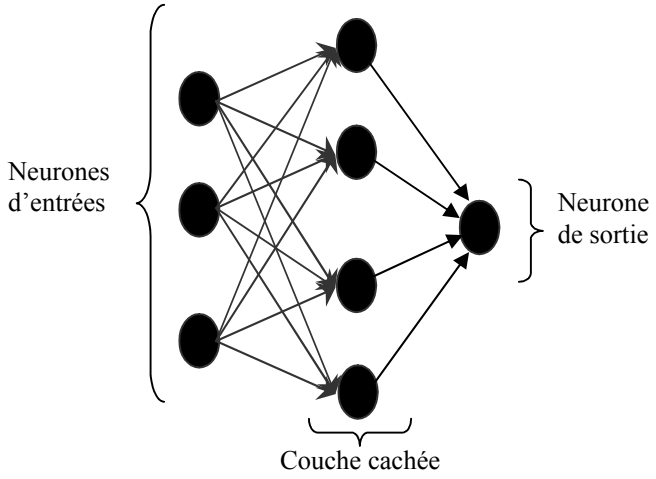


Figure I.7 : Exemple illustratif d'un réseau multicouche (MLP)

III.3 L'apprentissage des réseaux de neurones

L'apprentissage est le but principal du développement de modèles à base des réseaux de neurones. Il est réalisé par la modification des poids de connexion du réseau, généralement par des algorithmes spécifiques, afin d'obtenir des valeurs optimales appropriées à ces poids. A la fin de cette opération on converge vers un fonctionnement de réseau, le plus possible adapté au problème qu'on désire résoudre, toute en fournissant des exemples d'apprentissage. Ces derniers doivent être suffisamment représentatifs, autrement dit : il faudra qu'ils couvrent aussi complètement que possible le domaine de fonctionnement désiré pour le réseau.

Un échantillon d'apprentissage pour un réseau est constitué de N exemples, chacun est composé d'un vecteur des entrées et d'un vecteur des sorties désirées correspondantes à l'entrée. Suivant la règle utilisé pour l'apprentissage, on distingue deux principaux types d'apprentissages : non supervisé et supervisé.

III.3.1 L'apprentissage non supervisé

Le réseau doit détecter des points communs aux exemples présentés, par la modification des poids, afin de fournir la même sortie pour des entrées aux caractéristiques proches. L'apprentissage non supervisé est bien adapté à la modélisation des données complexes (images, sons, ...), généralement des données symboliques [7], où l'on possède des règles moins précises qui gouverne le comportement de systèmes à modélisé par les réseaux de neurones.

III.3.2 L'apprentissage supervisé

Comme nous l'avons vu précédemment, un réseau de neurones non bouclé réalise une fonction algébrique entre ses entrées et ses sorties. Donc on peut affecter à un tel réseau la tâche qui consiste à réaliser une fonction algébrique non linéaire, on fournit à ce réseau un couple (entrée, sortie) et on modifie les poids en fonction de l'erreur entre la sortie désirée et la sortie obtenue, figure I.8.

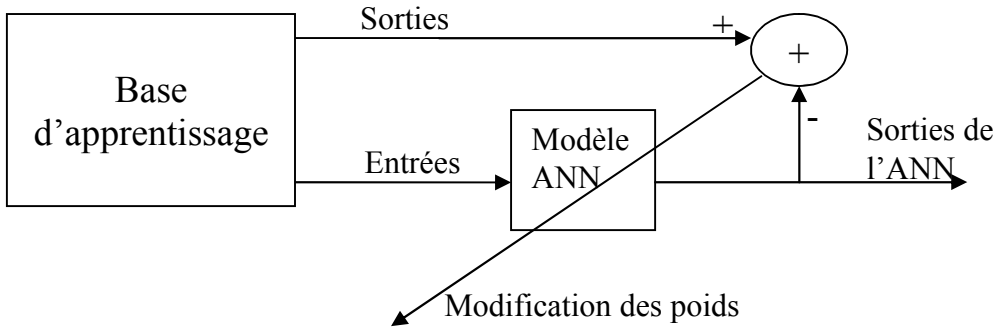


Figure I.8 : Apprentissage supervisé d'un réseau de neurones (ANN)

On peut divisé la fonction algébrique réalisé par ce réseau en deux parties : fonction connue analytiquement, où le réseau réalise la tâche d'approximation et une fonction inconnue analytiquement, mais pour laquelle on dispose de valeurs, en nombre fini, si ces valeurs résultent de mesures effectuées sur un processus physique, chimique, etc. Le réseau dans ce cas réalise une modélisation statique ou une régression [8].

Nous nous limitons, dans ce chapitre, à l'apprentissage supervisé et plus particulièrement à la modélisation statique. L'algorithme d'apprentissage utilisé dans notre travail : est la rétropropagation des erreurs (détaillé au paragraphe IV.3.1) car ce dernier est le mieux adapté à la modélisation statique par le perceptron multicouches MLP.

IV. Le perceptron multicouche MLP

Parmi les types des réseaux de neurones les plus utilisés on trouve le MLP avec son algorithme d'apprentissage, la rétropropagation des erreurs.

Le perceptron multicouche est un réseau orienté de neurones artificiels organisé en couches, où l'information voyage dans un seul sens, de la couche d'entrée vers la couche de sortie. La figure I.9 donne un exemple d'un réseau contenant une couche d'entrée, deux couches cachées et une couche de sortie. La couche d'entrée présente toujours une couche virtuelle associée aux entrées du système, elle ne contient aucun neurone. Les couches suivantes sont des couches de neurones. Dans l'exemple illustré (figure I.9), il y a 3 neurones d'entrées, 4 neurones sur la première couche cachée, trois neurones sur la deuxième couche cachée et 4 neurones sur la couche de sortie.

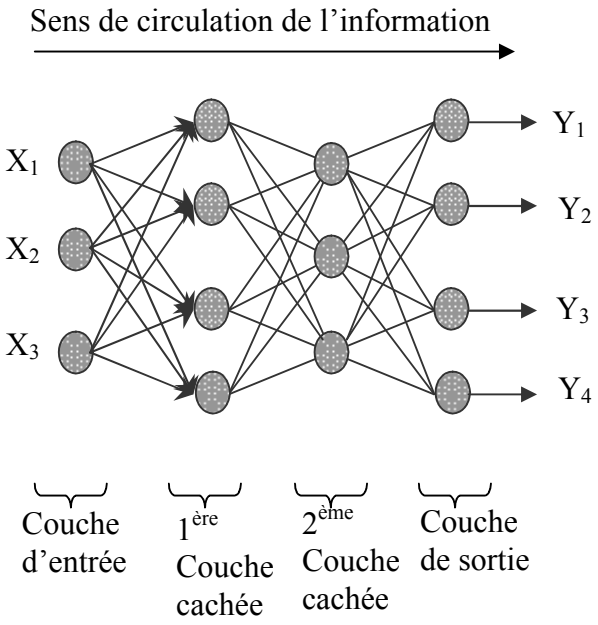


Figure I.9 : Exemple d'un réseau de type MLP

IV.1 Mise en œuvre du réseau de neurones MLP

La mise en œuvre des réseaux de neurones comporte à la fois une partie conception, dont l'objectif est de permettre de choisir la meilleure architecture possible, et une partie de calcul numérique, pour réaliser l'apprentissage d'un réseau de neurones. Dans le cas général, un MLP peut posséder un nombre de couches quelconque et un nombre de neurones par couche également quelconque, mais en vue de perfectionner le fonctionnement du MLP d'un côté et minimisé au maximum le temps de calcul d'autre part, on doit cherché une architecture optimale au point de vue nombre de couche et nombre de neurones par couche.

A partir d'une architecture de réseau de neurones donnée et des exemples disponibles (la base d'apprentissage), on détermine les poids optimaux, par l'algorithme de la rétropropagation des erreurs, pour que la sortie du modèle s'approche le plus possible du fonctionnement désiré.

IV.2 Propriétés fondamentales du MLP

La famille des réseaux de neurones à une couche de neurones cachée possède la propriété d'approximation parcimonieuse. Cela signifie qu'elle est capable d'approcher n'importe quelle fonction bornée et suffisamment régulière, en utilisant moins de paramètres ajustables que les familles de fonctions usuelles telles que les polynômes [8].

Dans l'optique d'une modélisation statistique, on utilise les réseaux de neurones pour approcher la fonction de régression du processus. L'intérêt de la parcimonie est alors de limiter le nombre d'exemples nécessaires pour obtenir une bonne estimation de la fonction de régression.

IV.2.1 L'approximation universelle

La propriété d'approximation universelle a été démontrée dans [9], et peut s'énoncer de la façon suivante :

« Toute fonction bornée suffisamment régulière peut être approchée uniformément, avec une précision arbitraire, dans un domaine fini de l'espace de ses variables, par un réseau de neurones comportant une couche de neurones cachée en nombre fini, possédant tous la même fonction d'activation, et un neurone de sortie linéaire » [8].

IV.2.2 La parcimonie

Lorsqu'on veut modéliser un processus à partir de données, on cherche toujours à obtenir les résultats les plus satisfaisants possibles avec un nombre minimal de paramètres ajustables. Dans cette optique, il est montré que [10] :

Si le résultat de l'approximation (c'est-à-dire la sortie du réseau de neurones) est une fonction non linéaire des paramètres ajustables, elle est plus parcimonieuse que si elle est une fonction linéaire de ces paramètres. De plus, pour des réseaux de neurones à fonction d'activation sigmoïdale, l'erreur commise dans l'approximation varie comme l'inverse du nombre de neurones cachés, et elle est indépendante du nombre de variables de la fonction à approcher. Par conséquent, pour une précision donnée, donc pour un nombre de neurones cachés donné, le nombre de paramètres du réseau est proportionnel au nombre de variables de la fonction à approcher [8].

L'avantage des réseaux de neurones par rapport aux autres techniques d'approximation (tels que les polynômes) est d'autant plus sensible que le nombre de variables de la fonction à approcher est grand : pour des problèmes faisant intervenir une ou deux variables, on peut généralement utiliser indifféremment des réseaux de neurones ou des polynômes. En revanche, pour des problèmes présentant trois variables où plus, il est généralement avantageux d'utiliser les réseaux de neurones [8], nous verrons dans le chapitre III comment on s'est servi de cette propriété pour la conception de modèle à base du réseau de neurones pour le CPS.

Rappelons ici, que cette propriété est démontrée de manière générale et pourrait se révéler inexacte pour un problème particulier. Elle constitue néanmoins une justification fondamentale de l'utilisation des réseaux de neurones. Nous allons montrer dans le paragraphe suivant que la technique des réseaux de neurones MLP est généralement utilisée comme une méthode de modélisation statistique.

IV.2.3 Modélisation statistique

Les réseaux de neurones, en raison de leurs propriétés fondamentales, mentionnées précédemment, sont des bons candidats pour réaliser une approximation de la fonction de régression. C'est ce qui justifie l'utilisation pratique des réseaux de neurones : la recherche d'une approximation de la fonction de régression à partir d'un nombre fini de points expérimentaux (exemple d'apprentissage). L'utilisation des réseaux de neurones entre donc complètement dans le cadre de méthodes statistiques d'approximation d'une fonction de régression. De telles méthodes ont été largement développées pour les fonctions de régression linéaires. L'avantage des réseaux de neurones réside dans leur capacité à modéliser des processus non linéaires et multi variables [8].

IV.3 L'apprentissage des réseaux MLP

L'apprentissage neuronal fait appel à des exemples de comportement. Soit une base d'apprentissage constituée de N exemples, chacun étant constitué d'un vecteur *x(n)* appliqué aux entrées du réseau, et du vecteur '*d(n)*' des valeurs désirées correspondantes pour les sorties, le vecteur '*y(n)*' correspond à la sortie du réseau pour l'entrée *' X(n)'*. On suppose aussi que le réseau de neurones possède un nombre '*r*' de neurones de sortie.

L'apprentissage d'un réseau de neurones est défini comme un problème d'optimisation qui consiste à trouver les coefficients du réseau minimisant une fonction d'erreur globale (fonction de coût). La définition de cette fonction de coût est primordiale, car celle-ci sert à

mesurer l'écart entre les sorties désirées du modèle et les sorties du réseau observées. La fonction la plus couramment utilisée, et dont nous nous sommes servi lors de nos travaux, est la fonction dite fonction d'erreur quadratique, dont la définition est :

Pour chaque exemple n ($n \in N$) on calcule une fonction d'erreur quadratique :

$$e(n) = \frac{1}{2}\sum_{j=1}^{r}[d_j(n) - y_j(n)]^2 \qquad \text{(I.3)}$$

Pour tout l'ensemble d'apprentissage N on peut définir la fonction de coût (appelée aussi l'erreur quadratique moyenne EQM):

$$E(n) = \frac{1}{N}\sum_{n=1}^{N} e\ (n) \qquad \text{(I.4)}$$

Le principe de l'algorithme d'apprentissage (la rétropropagation) est de calculer la contribution des poids du réseau à cette erreur [11].

IV.3.1 L'algorithme de la rétropropagation des erreurs

L'apprentissage du MLP est attaché à l'algorithme de la rétropropagation des erreurs, cet algorithme, utilisée par les réseaux multicouches, consiste simplement en une descente de gradient, qui est une méthode d'optimisation universelle. On cherche à minimiser une fonction de coût (qui représente l'erreur entre la sortie désirée et la sortie obtenue), en suivant les lignes de plus grande pente [11].

La mise en œuvre de cet algorithme nécessite un enchaînement des opérations mathématiques données comme suit [12] :

Soit le couple $(\vec{x}(n), \vec{d}(n))$ désignant la nième donnée d'apprentissage du réseau où :

$$\vec{x}(n) = < x_1(n), ..., x_P(n) > \qquad \text{(I.5)}$$

$$\vec{d}(n) = < d_1(n), ..., d_q(n) > \qquad \text{(I.6)}$$

Correspondent respectivement aux « p » entrées et aux « q » sorties désirées du système. L'algorithme de la rétropropagation consiste alors à mesurer l'erreur entre les sorties désirée, et les sorties observées $\vec{y}(n)$:

$$\vec{y}(n) = < y_1(n), ..., y_q(n) > \qquad \text{(I.7)}$$

Résultat de la propagation vers l'avant des entrées $\vec{x}(n)$, et à rétropropager cette erreur à travers les couches du réseau en allant des sorties vers les entrées [11].

L'algorithme de rétropropagation procède à l'adaptation des poids neurone par neurone en commencent par la couche de sortie. Soit l'erreur observée $e_j(n)$ pour le neurone de sortie j et la donnée d'entraînement (apprentissage) n :

$$e_j(n) = d_j(n) - y_j(n) \tag{I.8}$$

L'indice j représente le neurone pour lequel on veut adapter les poids.

L'objectif de l'algorithme est d'adapté les poids des connexions du réseau de manière à minimiser la somme des erreurs sur tous les neurones de sortie.

Soit $E(n)$ la somme des erreurs quadratique observées sur l'ensemble C des neurones de sorties :

$$E(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{I.9}$$

La sortie du neurones j est définie par :

$$y_j(n) = \sigma[\sum_{i=0}^{r} w_{ji}(n).y_i(n)] \tag{I.10}$$

Où : σ est la fonction d'activation du neurone j, w_{ji} est le poids de la connexion entre le neurone i de la couche précédente et le neurone j de la couche courante, et y_i est la sortie du neurone i. On suppose ici que la couche précédente contient r neurones numérotés de 1 à r, le poids w_{j0} correspond au biais (seuil) du neurone j et que l'entrée $y_0(n) = 1$ (figure I.10).

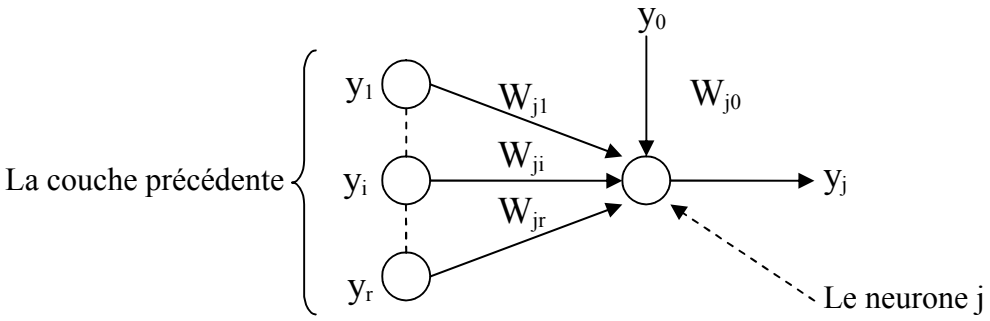


Figure I.10 : Modèle du neurone j

Pour corriger l'erreur observée, il faut modifier le poids $w_{ji}(n)$ dans le sens opposé au gradient $\frac{\partial E(n)}{\partial w_{ji}(n)}$ de l'erreur (figure I.11).

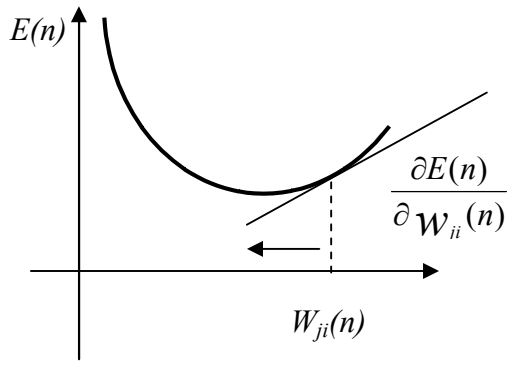


Figure I.11 : Gradient de l'erreur total

Cette dérivée partielle représente un facteur de sensibilité :

- si on change un peu $w_{ji}(n)$, $E(N)$ change beaucoup, alors on change beaucoup $w_{ji}(n)$ dans le sens inverse de cette dérivée car cela devrait nous rapprocher beaucoup du minimum local.
- si non, on dois changer seulement un peu $w_{ji}(n)$ pour corriger l'erreur car on est tout prés de ce minimum.

Par la règle de chaînage des dérivées partielle on obtient :

$$\frac{\partial E\ (n)}{\partial w_{ji}(n)} = \frac{\partial E\ (n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{I.11}$$

Avec :

$$v_j(n) = \sum_{i=0}^{r} w_{ji}(n) . y_i(n) \tag{I.12}$$

Et on exprime la variation de poids $\Delta w_{ji}(n)$ sous la forme :

$$\Delta w_{ji}(n) = -\eta . \frac{\partial E\ (n)}{\partial w_{ji}(n)} \tag{I.13}$$

Avec : $0 \le \eta \le 1$ représentant un taux d'apprentissage ou gain de l'algorithme

Evaluons maintenant chacun des termes du gradient :

- $$\frac{\partial E\ (n)}{\partial e_j(n)} = \frac{1}{2} . \frac{\partial e_j^2(n)}{\partial e_j(n)} = e_j(n) \tag{I.14}$$

- $$\frac{\partial e_j(n)}{\partial y_j(n)} = \frac{\partial\ [d_j(n) - y_j(n)]}{\partial y_1(n)} = -1 \tag{I.15}$$

- $$\frac{\partial y_j(n)}{\partial v_j(n)} = \frac{\partial[\sigma(v_j(n))]}{\partial v_j(n)} \quad \text{(I.16)}$$

On suppose que σ est fonction sigmoïde, ce qui donne :

$$\sigma\ (v_j(n)) = \frac{1}{1+\exp(-v_j(n))} \quad \text{(I.17)}$$

Alors

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \frac{\exp(-v_j(n))}{(1+\exp(-v_j(n)))^2} \quad \text{(I.18)}$$

Or

$$\frac{\partial y_j(n)}{\partial v_j(n)} = y_j(n).[1-y_j(n)] \quad \text{(I.19)}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = \frac{\partial\ [\sum_{i=0}^{r} w_{ji}(n).y_i(n)]}{\partial w_{ji}(n)} = y_i(n) \quad \text{(I.20)}$$

Nous obtenons donc :

$$\frac{\partial E\ (n)}{\partial w_{ji}(n)} = -e_j(n)\, y_j(n).[1-y_j(n)].y_i(n) \quad \text{(I.21)}$$

et la règle du 'delta' pour la couche de sortie s'exprime par :

$$\Delta w_{ji}(n) = -\eta.\frac{\partial E\ (n)}{\partial w_{ji}(n)} = \eta\,\delta_j(n).y_i(n) \quad \text{(I.22)}$$

Avec :

$$\delta_j(n) = e_j(n)\, y_j(n).[1-y_j(n)] \quad \text{(I.23)}$$

Considérons maintenant le cas des neurones sur la dernière couche caché (le cas des autre couches cachées est semblable).

- La variable n désignera toujours la donnée d'entraînement
- Les indices i et j désignerons respectivement (comme précédemment) un neurone sur la couche précédente et un neurone sur la couche courante.
- L'indice k servira maintenant à désigner un neurone sur la couche suivante

Reprenons l'expression de la dérivée partielle de l'erreur totale E(n) par rapport à W_{ji} mais on ne dérivant pas par rapport à l'erreur $e_j(n)$, car celle-ci est inconnue dans le cas d'une couche cachée.

$$\frac{\partial E\ (n)}{\partial w_{ji}(n)} = \frac{\partial E\ (n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{I.24}$$

Par rapport aux résultats obtenus pour la couche de sortie, les deux derniers termes de cette équation restent inchangés, seul le premier terme sera évalué.

$$\frac{\partial E\ (n)}{\partial y_j(n)} = \frac{\partial\ [\frac{1}{2}\sum_{k\in c} e_k^2(n)]}{\partial y_j(n)} \tag{I.25}$$

$$\frac{\partial E\ (n)}{\partial y_j(n)} = \sum_{k\in c} [e_k(n) \cdot \frac{\partial e_k(n)}{\partial y_j(n)}] \tag{I.26}$$

$$\frac{\partial E\ (n)}{\partial y_j(n)} = \sum_{k\in c} [e_k(n) \cdot \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)}] \tag{I.27}$$

$$\frac{\partial E\ (n)}{\partial y_j(n)} = \sum_{k\in c} [e_k(n) \cdot \frac{\partial\ [d_k(n) - \sigma(v_k(n))]}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)}] \tag{I.28}$$

$$\frac{\partial E\ (n)}{\partial y_j(n)} = \sum_{k\in c} [e_k(n).(-y_k(n).[1-y_k(n)]) \cdot w_{kj}] \tag{I.29}$$

Ce qui donne :

$$\frac{\partial E\ (n)}{\partial y_j(n)} = -\sum_{k\in c} [\delta_k(n) \cdot w_{kj}(n)] \tag{I.30}$$

En substituant l'équation II.30 dans II.24, on obtient :

$$\frac{\partial E\ (n)}{\partial w_{ji}(n)} = -y_j(n)[1-y_j(n)]\sum_{k\in c} [\delta_k(n) \cdot w_{kj}(n)]\, y_i(n) \tag{I.31}$$

et :

$$\Delta w_{ji}(n) = -\eta \cdot \frac{\partial E\ (n)}{\partial w_{ji}(n)} = \eta\, \delta_j(n) \cdot y_i(n) \tag{I.32}$$

avec :

$$\delta_j(n) = y_j(n).[1-y_j(n)].\sum_{k\in c}[\delta_k(n) \cdot w_{kj}(n)] \tag{I.33}$$

On peut démontré que les équations II.32 et II.33 sont valide pour toutes les couches cachées. Cependant, pour la première couche cachée du réseau, il faut substituer la variable $Y_i(n)$ par l'entrée du réseau $X_i(n)$.

IV.3.2 Mise en œuvre de l'algorithme

Nous allons maintenant utiliser ce qui précède pour résumer la mise en œuvre de l'algorithme de rétropropagation standard [12].

Phase 1 : initialisation de tous les poids à de petite valeurs aléatoire dans l'intervalle [-0.5, 0.5] ;

Phase 2 : pour chaque donnée d'entraînement n :

a. calculer des sorties observées en propageant les entrées vers l'avant ;

b. ajuster les poids en rétropropageant l'erreur observée :

$$w_{ji}(n) = w_{ji}(n-1) + \Delta w_{ji}(n) = w_{ji}(n-1) + \eta \delta_j(n).y_i(n) \qquad \text{(I.33)}$$

où le gradient local es défini par **:**

$$\begin{cases} \delta_j(n) = e_j(n)\, y_j(n).[1 - y_j(n)] & \text{pour la couche de sortie} \\ \delta_j(n) = y_j(n).[1 - y_j(n)].\sum_{k \in c}[\delta_k(n).w_{kj}(n)] & \text{pour une couche cachée} \end{cases}$$

avec $0 \le \eta \le 1$ est le taux d'apprentissage

- le choix de η est empirique,
- si η est trop petit, le nombre d'itérations peut être très élevé,
- si η est trop grand, les valeurs de la suite risquent d'osciller autour du minimum sans converger.

phase 3 : répéter les étape 1 et 2 jusqu'à un nombre maximum d'itération ou jusqu'à ce que la valeur de l'erreur quadratique moyenne (EQM) soit inférieur à un certain seuil ;

En effet, le but d'atteindre EQM inférieur à un seuil n'est pas sûre, alors pour éviter le problème de la boucle ouverte, on fixe un nombre d'itérations maximum, généralement l'ordre des centaines, dans ce cas là l'algorithme cherche à minimiser EQM en N_i itérations successive tel que : N_i est inférieur au nombre d'itérations maximum.

Nous verrons dans le chapitre III, la mise en œuvre détaillée de l'algorithme ainsi que les résultats pratiques obtenus.

IV.4 Optimisation de l'architecture

Le nombre d'unités cachées joue un rôle crucial dans le contrôle de la capacité du réseau de neurones. Si le nombre de couche caché est trop petit, alors le réseau possède trop peu de paramètres et ne peut capter toutes les dépendances qui servent à modéliser la fonction de régression. À l'inverse, si l'on choisit une valeur trop grande pour le nombre de couche cachée, alors le nombre de paramètres du modèle augmente et il devient possible, pendant la phase d'optimisation des paramètres, de modéliser certaines relations qui ne sont que le fruit de fluctuations statistiques propres à l'ensemble d'entraînement utilisé plutôt que des relations fondamentales de dépendance entre les variables d'entrée. Il faut comprendre que les réseaux

de neurones sont des approximateurs universels, c'est-à-dire qu'ils peuvent modéliser n'importe quelle fonction si le nombre d'unités cachées est suffisant. Autrement dit, un réseau de neurones peut apprendre par coeur un ensemble d'entraînement [7].

Après avoir entraîné quelques modèles, chacun avec un nombre différent d'unités cachées, on peut comparer les erreurs d'entraînement et de validation. On obtient généralement le résultat suivant :

L'erreur d'entraînement diminue au fur et à mesure que le nombre d'unités cachées augmente. L'erreur de validation, quant à elle, est élevée lorsque le nombre d'unités cachées est faible, décroît avec l'augmentation du nombre d'unités cachées, atteint un minimum pour un certain nombre optimal d'unités cachées, puis croît lorsque le nombre d'unités devient trop grand. C'est donc l'utilisation d'un ensemble de validation, distinct de l'ensemble d'entraînement, qui nous permet de choisir le nombre optimal d'unités cachées ou neurones. Alors le nombre d'unité caché est en général déterminé selon une procédure itérative, suivant le succès de validation [7].

IV.5 Validation du modèle

En plus de l'ensemble d'apprentissage, un second ensemble appelé ensemble de validation, est utilisé à la fin de chaque phase d'apprentissage. On mesure non seulement l'erreur d'apprentissage mais aussi l'erreur de validation, c'est-à-dire l'erreur totale commise sur tous les exemples de l'ensemble de validation. Cette erreur de validation est calculée une fois que la phase d'optimisation des poids est terminée. A partir d'un ensemble d'apprentissage de N exemple on tire un sous ensemble N_0, généralement N_0 représente 10 à 25% de l'ensemble N. On réserve N_0 et on fait l'apprentissage des réseaux à partir de la base composé de $(N - N_0)$ exemple puis on valide le modèle par l'ensemble N_0. Il y a plusieurs techniques concernant le choix de l'ensemble N_0, la plus utilisé consiste à tiré N_0 d'une manière à couvrir au maximum l'ensemble d'apprentissage ce qui est traduit mathématiquement par : le facteur de l'espérance mathématique pour l'ensemble N_0 ($N_0 \in N$) est le plus élevé possible.

V. Conclusion

A travers ce chapitre nous avons exploré la capacité des réseaux de neurones artificiels dans le domaine de la modélisation physique et nous avons mis en valeur leur grande adaptation aux différents problèmes liés au régime dynamique. Du fait de leur forte non linéarité et l'aspect de généralisation qu'ils présentent, les réseaux de neurones artificiels sont de bons candidats à la modélisation du comportement du CPS en régime dynamique.

Introduction

L'objectif assigné à ce chapitre est de décrire le comportement du capteur de pression capacitif (CPS) miniature réalisé sur un substrat de silicium. En premier lieu nous présentons son architecture de base, puis nous caractérisons sa réponse et nous terminons par l'étude de son modèle analytique.

Le capteur de pression capacitif, décrit dans ce chapitre, repose sur le principe de la variation d'une capacité en fonction de la pression appliquée. Parmi les problèmes technologiques liés à la mesure de pression par le CPS, on trouve celui du comportement thermique, en effet, sa réponse est largement dépendante de la température. En plus du caractère non linéaire de la réponse, et le phénomène de l'hystérésis prélevé, dont nous allons tenir compte.

I. Principe de fonctionnement du CPS

Le principe de base d'un capteur de pression réside dans une déformation sous l'effet d'une contrainte mécanique qui engendre une variation d'une grandeur électrique. On peut distinguer deux grande famille de capteur de pression : le type piézorésistif où la mesure de la pression se fait par la mesure d'une variation d'une résistance et le deuxième type, capacitif basé sur la variation d'une capacité en fonction de la pression appliquée (figure II.1). Ce second type est caractérisé par une haute sensibilité, une faible consommation électrique et une grande résolution par rapport au premier, cependant, il présente une réponse non linéaire, une dérive en température et une faible variation par rapport à la capacité d'offset [13,14].

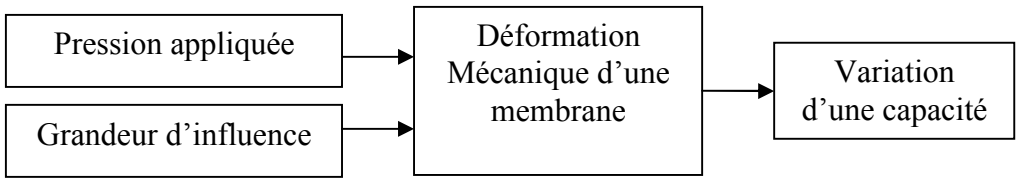


Figure II.1 : Principe de fonctionnement d'un capteur de pression

I.1 Structure du CPS

La structure de base est composée d'une armature fixe déposée au fond d'une cavité creusée sur un substrat et d'une armature déformable, appelée « membrane » [15,16], figure II.2. Le matériau utilisé pour l'armature déformable est le silicium compte tenu de son excellent comportement mécanique et de son micro usinage précis [17]. La cellule est réalisée à partir

d'une plaque de silicium suffisamment dopé pour que la résistivité de cette électrode soit faible (environ 0.008 Ω.cm). La membrane est fixée au substrat par soudure thermoélectrique.

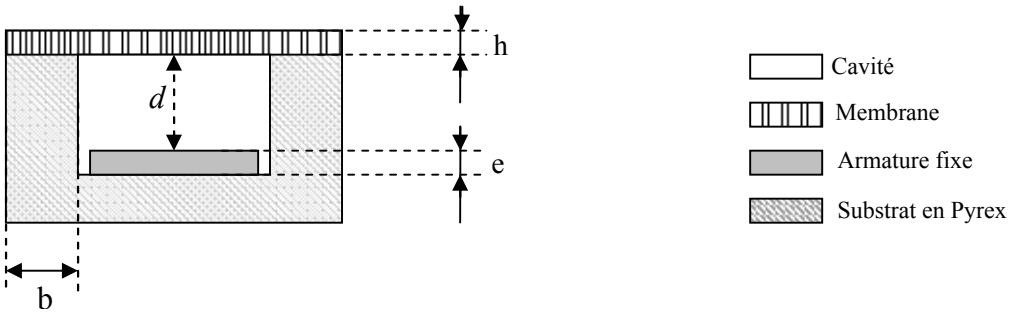


Figure II.2 : Vue en coupe de la structure du CPS

Avec :

h : L'épaisseur de la membrane

e : L'épaisseur de l'armature fixe

d : La distance entre les deux armatures

A : Surface de l'armature fixe

b : Largeur de la soudure

I.2 Mécanisme de détection

En l'absence d'une pression appliquée (P=0), les deux armatures sont parallèles. la capacité du repos est donnée alors par la relation :

$$C_0 = \varepsilon.\frac{A}{d} \qquad \text{(II.1)}$$

Où :

ε : la permittivité

A : la surface des armatures fixe

d : la distance séparant les deux armatures

Lorsqu'on applique une pression P, la membrane déformable fléchie comme indiqué à la figure II.3. La distance séparant les deux armatures varie en tous points de la membrane [18], dans ce cas là la relation entre la capacité et la pression appliquée devient :

$$C(p) = \varepsilon.\iint_A \frac{\partial A}{d - w(x,y,p)} \qquad \text{(II.2)}$$

Où :

∂A : représente un élément de la surface

W(x, y, P) : la déflexion en fonction de la pression au point de cordonnées (x, y) de la membrane ; l'origine du repère étant définie au centre géométrique de la face inférieure de la membrane à P=0.

Cette relation montre que si la pression augmente la distance entre les armatures diminue et par conséquent la capacité augmente.

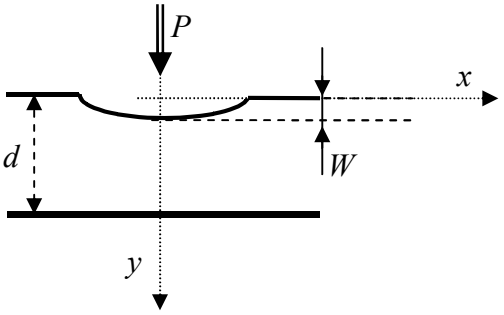


Figure II.3 : Illustration de la déformation de la membrane

II. Réponse du CPS en pression paramétrée en température

Afin d'évaluer la réponse du CPS en fonction de la pression dans un milieu où la température varie, nous avons représenté la variation de la capacité du CPS pour une plage de pression de 1 à 6 bars pour différentes températures (figure II.4), en se basant sur des travaux expérimentaux [19].

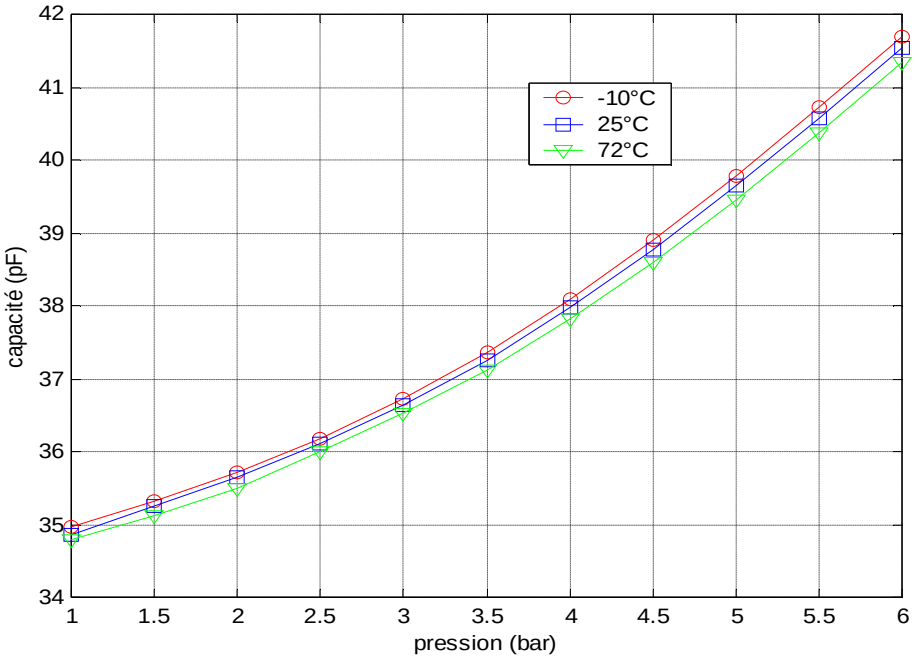


Figure II.4 : Variation de la capacité en fonction de la pression à différentes températures [19]

On remarque que la variation de la capacité est une fonction non linéaire de la pression appliquée et dépend de la température, ce qui donne un système multi variable. Nous avons regroupé au tableau II.1, les différents paramètres géométriques du CPS utilisé [19].

Epaisseur de la membrane **(h)**	50 µm
Epaisseur de l'armature fixe **(e)**	0.1 µm
Distance entre les deux armatures **(d)**	1.7 µm
Surface de l'armature fixe **(A)**	6.327 mm^2
Largeur de la soudure **(b)**	500 µm

Tableau II.1 : Principaux paramètres géométriques du CPS [19]

III. Etude du modèle analytique

En examinons les résultats expérimentaux du CPS [19], sa réponse peut être modélisée par une droite à laquelle on ajoute un terme de non linéarité (figure II.5) en plus de l'hystérésis. Donc la réponse du CPS peut être mise sous la forme :

$$C(P,T)=C_0(T)+S(T).P+NL(P,T)+H_p(P,P^-,T) \quad \text{(II.3)}$$

Où :

- $NL(P,T)$: exprime le terme de la non linéarité qui dépend de la pression appliquée et de la température.
- $C0(T)$: Capacité d'offset qui dépend de la température
- $S(T)$: Sensibilité à la pression du CPS et qui dépend de la température
- $H_p(P,P^-,T)$: Exprime l'hystérésis en pression qui dépend de la pression appliquée P, de la pression précédente P^- et de la température.

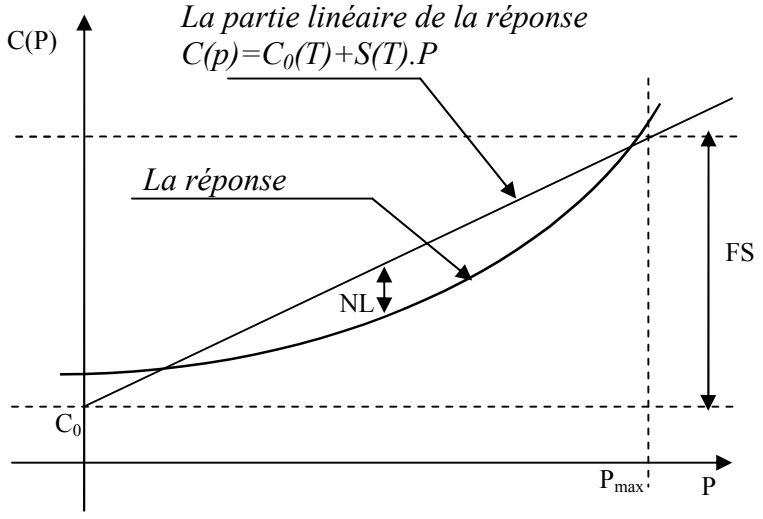


Figure II.5 : Définition graphique du modèle analytique

III.1 Dérive en température

Dans ce paragraphe on examine l'effet de la température sur la réponse du CPS. Physiquement, la déformation de la membrane peut être causée aussi bien par la pression que par la température, en plus, la température influe aussi sur le facteur d'élasticité de la membrane et par conséquent la sensibilité à la pression [20].

Conformément au modèle analytique proposé, nous avons étudié la drive thermique de chacun des paramètres de ce modèle. A savoir celle de la capacité d'offset 'C_0' et la sensibilité en pression 'S'. Le comportement thermique de chaque paramètre peut être évaluer par le calcul de son coefficient thermique qui est défini par :

$$TC(X) = \frac{1}{X} \cdot \frac{\partial X}{\partial T} \tag{II.4}$$

Où X est une variable qui représente un des paramètres du modèle. Ce coefficient peut être exprimé soit en pourcentage par degré Celsius (% / °C), soit en partie par million par degré Celsius (ppm/ °C). Vu que pratiquement les coefficients sont relativement faible par rapport aux valeurs nominales, X(T) peut être remplacée par X(T=25°C).

III.1.1 Dérive thermique de la capacité d'offset

Un capteur est dit « au repos » si l'ensemble des pressions s'exerçant sur la membrane est nulle, donc la capacité du CPS se réduit à une capacité appelée capacité d'offset. La variation de la capacité d'offset introduite par la variation de la température 'T', à une pression nulle, dans un intervalle de -10 à 90 °C est représentée sur la figure II.6 [21].

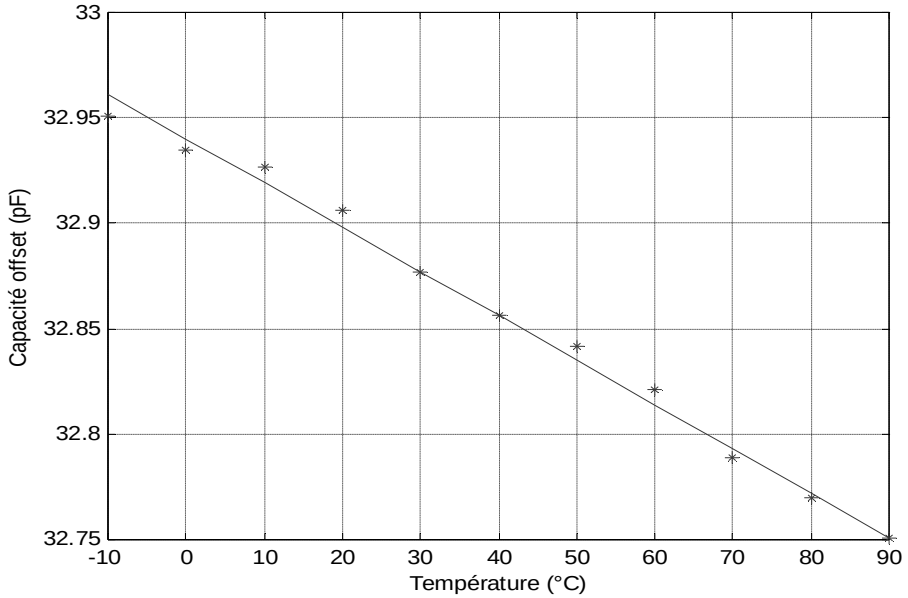


Figure I1.6 : Dérive thermique de la capacité d'offset [21]

La dérive de la capacité d'offset peut être approchée à une droite linéaire, le coefficient de température est constant est donné par : $TC[C_0(T)]$ = -58.8 ppm/°C . On peut ainsi attribuer à la variation de la capacité d'offset en fonction de la température un modèle linéaire, ce qui se traduit par l'expression II.5.

$$C_0(T)=C_0+g_1.T \qquad \text{(II.5)}$$

Où :

C_0 : la capacité d'offset à la température T=0 °C

g_1 : la dérive en température

C_0 et g_1 dépendent des paramètres géométriques du CPS.

III.1.2 Dérive de la sensibilité

La sensibilité d'un capteur est un paramètre important qui caractérise le comportement d'un capteur. Cependant elle varie également en fonction de la température [21].

Nous avons reporté, sur la figure II.7, la dérive thermique de la sensibilité '*S (T)*'.

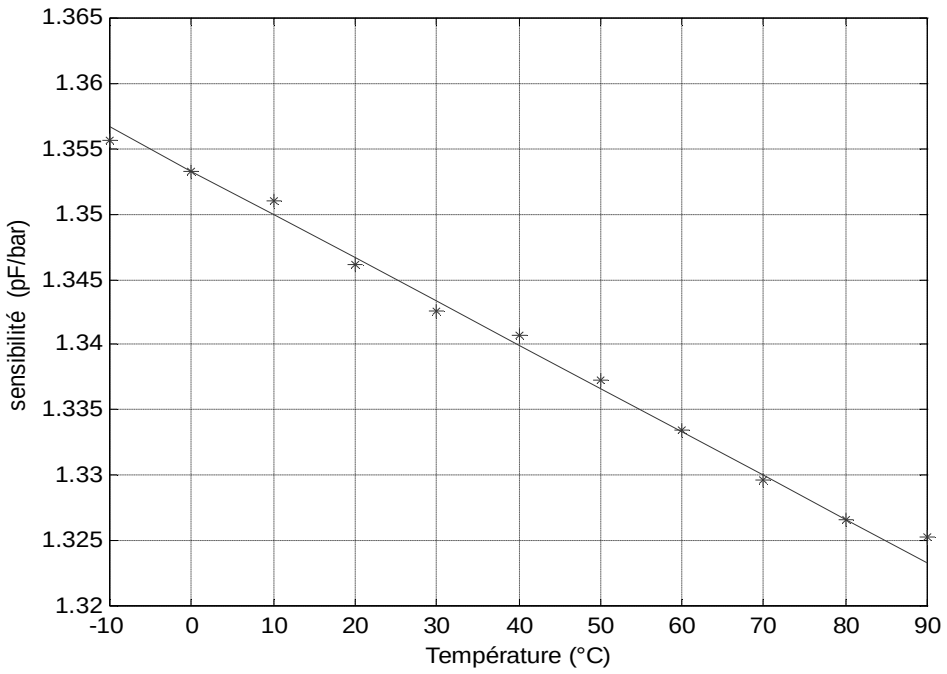


Figure I1.7 : Dérive thermique de la sensibilité en pression du CPS [21]

De la même manière que pour la capacité d'offset, nous pouvons approché la dérive thermique de la sensibilité par une régression linéaire (expression II.6). Le coefficient de température est considéré constant, il est donné par : $TC[C_0(T)]$ = -222.2 ppm/°C.

$$S(T)=S_0+g_2.T \qquad \text{(II.6)}$$

Où :

S_0 : Sensibilité à la température T=0 °C

g_2 : Dérive en température de la sensibilité

S_0 et g_2 dépendent des paramètres géométriques du CPS.

III.2 La non linéarité

La non linéarité de la réponse en pression à une température constante est calculée en faisant la différence entre les points de mesure $C_m(P)$ et la droite des moindre carrés de ses mêmes points $C_L(P)$. Cette différence est exprimée en pourcentage par rapport à la réponse pleine échelle (FS).

$$NL(\%FS) = 100.\frac{C_m - C_L}{FS} \qquad \text{(II.7)}$$

La figure II.8 illustre la variation de la non linéarité à température ambiante pour un cycle de pression de1 à 6 bars [19].

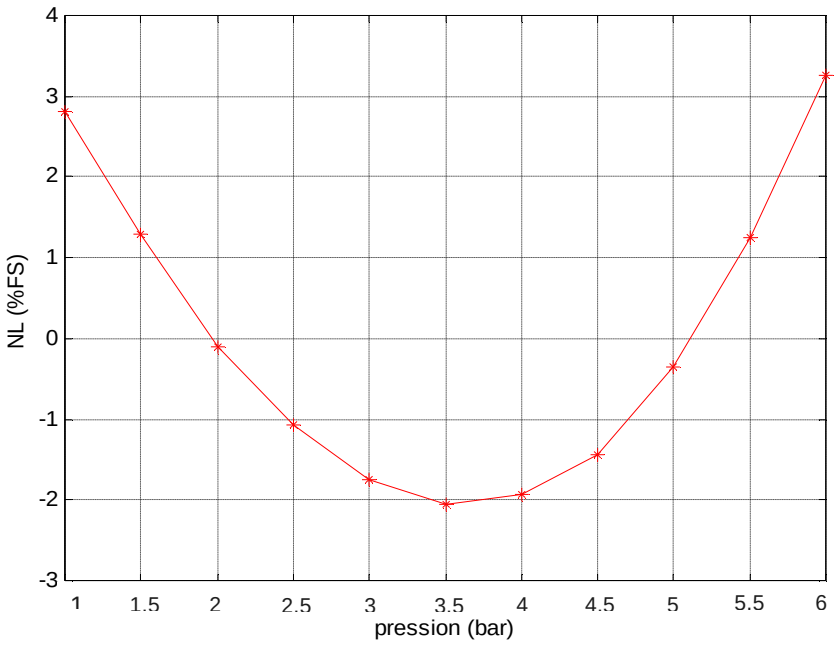


Figure II.8 : La non linéarité en fonction d'un cycle de pression à température ambiante [19]

En pratique la température a une faible implication sur la non linéarité par rapport à la pression, ce qui est traduit par (II.8) :

$$\frac{\Delta NL}{\Delta T}\Big|_{P=const} << \frac{\Delta NL}{\Delta P}\Big|_{T=const} \qquad \text{(II.8)}$$

La non linéarité se réduit à une fonction de la pression, nous avons alors modélisé la non linéarité par une régression polynomiale d'ordre 3, donné par l'expression (II.9).

$$NL(P) = 0.0484.P^3 + 0.3174.P^2 - 4.2135.P + 6.7282 \quad \text{(II.9)}$$

III.3 Hystérésis en pression

Si le capteur est placé dans un environnement dynamique, l'effet d'hystérésis apparaît. En effet, la réponse du capteur peut avoir deux valeurs déférentes pour une même pression suivant un cycle de pression (croissant ou décroissant). La différence entre ces deux valeurs donne la valeur de l'hystérésis en pression. Elle est exprimée en pourcentage par rapport à la réponse pleine échelle (FS) selon l'équation (II.10) [19].

$$H_P = 100.\frac{C(P_c) - C(P_d)}{FS} \quad \text{(II.10)}$$

Avec

- P_c : la pression croissante
- P_d : la pression décroissante

Lorsqu'on tient compte de l'effet d'hystérésis, la réponse du capteur est une fonction de la pression appliquée et de la pression précédente. Les figures II.9 et II.10, illustrent ces variations en fonction d'un cycle de pression.

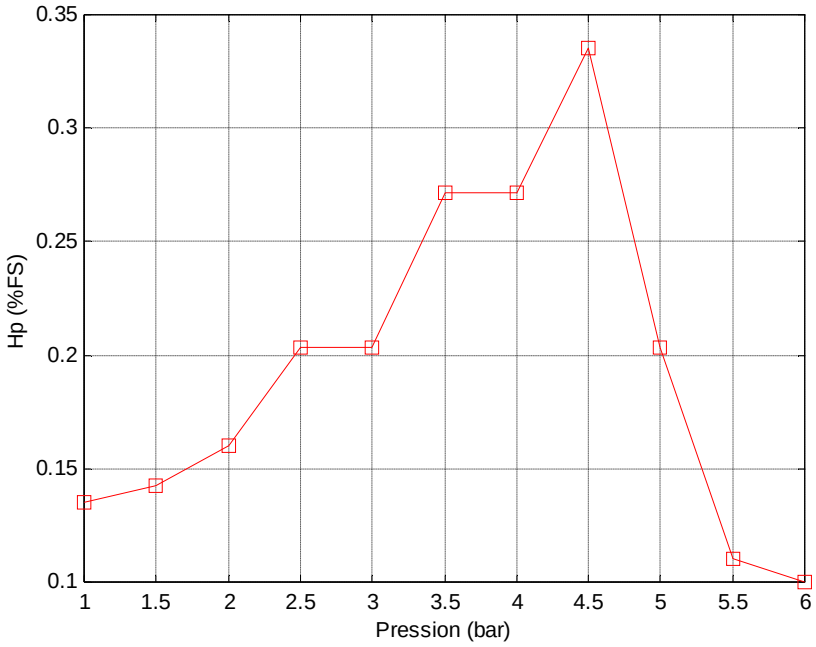


Figure II.9 **:** *Variation de la valeur d'hystérésis en fonction d'un cycle de pression [19]*

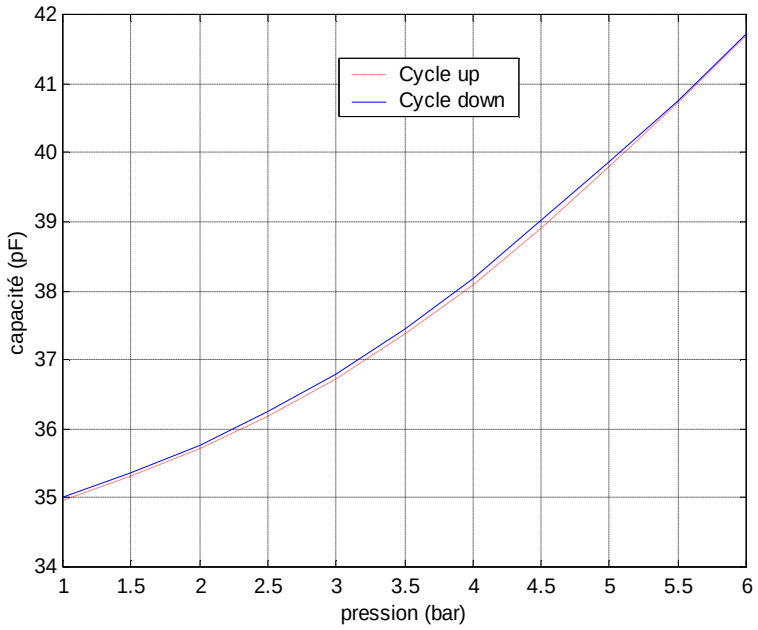


Figure II.10 : Variation de la capacité en fonction d'un cycle de pression [19]

La valeur d'hystérésis dépend également de la température. La figure II.11 représente la variation de la valeur maximale de l'hystérésis en fonction de la température.

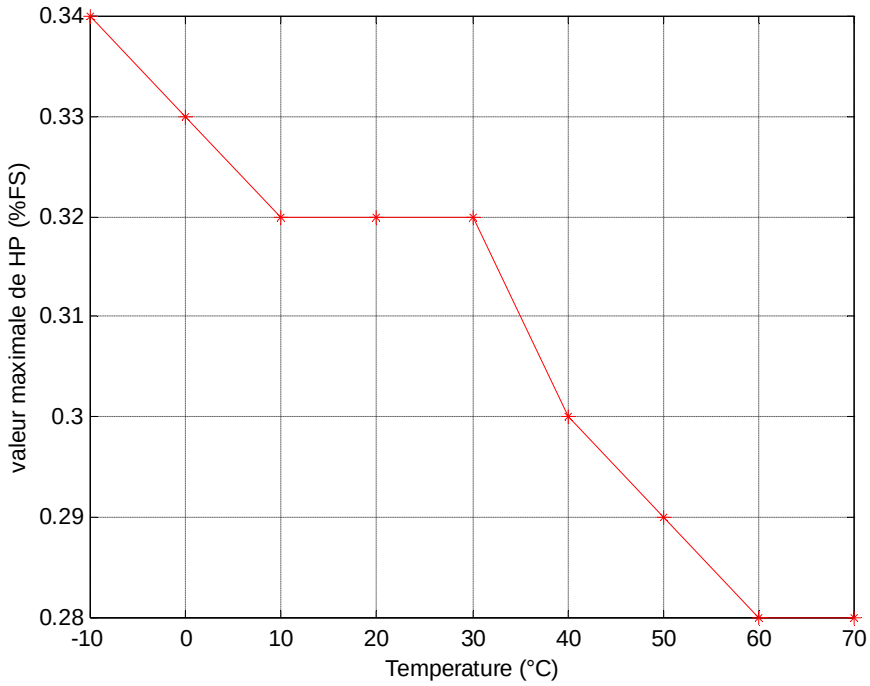


Figure II.11: Variation de la valeur maximale d'hystérésis en fonction de température [19]

Il est à noter que dans le cas où le capteur est placé dans un environnement caractérisé par des variations lentes en pression et en température. On peut dire que $P \approx P^{-}$ ce qui implique que H_p est négligeable.

III.4 Influence des paramètres géométriques sur le comportement thermique

Le comportement thermique du CPS est causé essentiellement par la variation des caractéristiques mécaniques des matériaux utilisé tel que : le facteur d'élasticité et la déformation thermique de la membrane au silicium [17]. Ces variations en fonction de la température dépendent des paramètres géométriques qui définissent l'architecture du CPS [21].

Pour mettre en évidence l'influence des paramètres géométriques nous avons donné sur le tableau II.2 quelque exemples [21].

Les paramètres géométriques					C_0	TC[C_0(T)]	TC[S(T)]
H (µm)	b (µm)	e (µm)	d (µm)	A (mm^2)	(pF)	(ppm/°C)	(ppm/°C)
50	1000	0.1	0.8	2.89	32.05	-45	-444
50	1000	1	0.8	5.33	59.06	-227	-377
50	750	0.1	1.7	2.91	32.3	-50	-378
50	750	1	0.8	5.71	63.32	-263	-239
50	500	1	0.8	6.13	67.91	-352	-193
50	250	0.1	1.7	6.33	32.98	-174	-188

Tableau II.2 :Influence des paramètres géométriques sur le comportement thermique du CPS [21]

IV. Limitation du modèle analytique

A partir des résultas montré au paravent, le comportement du CPS dans un environnement relativement moins dynamique peut être modélisé par l'expression (II.11).

$$C(P,T) = (C_0 + g_1.T) + (S_0 + g_2T).P + NL(p) \qquad \text{(II.11)}$$

Pour des conditions en pression et en température relativement moins dynamiques l'approche analytique exprime fidèlement la variation de la capacité en fonction de la température. Cependant, dans un milieu dynamique la variation de la capacité devient une fonction

complexe multi variable de la pression ‘P’, la pression précédente ‘P^{-}’ et de la température ‘T’. Elle est caractérisé par une variation fortement non linéaire.

V. Conclusion

Le modèle analytique étudié dans ce chapitre présente l’avantage de la simplicité en plus du bon suivi de la loi de variation présenté par le CPS. Cependant, son utilisation dans un milieu dynamique, montre son incapacité à suivre la loi de variation du moment que l’approche analytique ne tient pas en compte de l’effet d’hystérésis. En se basons sur les notions des ANNs étudiées au chapitre I et les résultats expérimentaux collectés, nous allons tenté de modéliser le comportement du CPS dans un environnement dynamique, au chapitre III.

Introduction

Après avoir décrit le comportement du CPS (chapitre II) et mis en évidence la limitation de son modèle analytique d'un côté, et explorer la capacité des réseaux de neurones dans la modélisation statistique, d'autre côté (chapitre I). Nous allons exposer à travers ce chapitre un modèle à base de réseaux de neurones (modèle ANN) plus précis que l'approche analytique. Ce modèle doit tenir compte de l'effet de l'hystérésis lorsque le capteur est placé dans un milieu dynamique. L'implantation du modèle ANN sur le simulateur SPICE ainsi que les résultats de simulation obtenus sont également reportés.

L'objectif visé à travers la construction d'un modèle à base d'ANN pour le CPS, est de reproduire fidèlement son comportement lors de la phase de conception du capteur intelligent (chapitre IV). Autrement dit ce modèle servira à la simulation comportementale du capteur intelligent, vu que le modèle analytique du CPS est limité à des conditions moins dynamique.

I. Modélisation du CPS à base des réseaux de neurones

Les ANN sont utilisés, en instrumentation, pour modéliser des systèmes complexes du fait de leur aspect fortement multi variable et de leur forte non linéarité. Les ANN sont très efficace pour des problèmes à caractère dynamique, qu'on ne peut pas traitée dans le domaine statistique avec des modèles analytiques. De plus les ANNs offrent l'avantage de la simplicité d'implantation et un temps de calcul relativement faible par rapport aux modèles numériques [22].

Pour concevoir le modèle ANN du CPS, en question, nous avons utilisé l'interface MATLAB au cours de la phase de conception et d'optimisation. Les résultas obtenus (l'architecture optimale et les poids du réseau) sont utilisés pour l'implantation du modèle sur le simulateur SPICE. La modélisation par les réseaux de neurones de notre capteur (CPS), doit passer par l'enchaînement des étapes suivantes :

a. Choix d'une base de données,
b. Séparation des bases d'apprentissage et de test,
c. Entraînement d'un réseau de neurones sur la base d'apprentissage avec l'algorithme de rétropropagation (BP),
d. Mesure de la performance du modèle obtenue avec la base de test.

I.1 Base d'apprentissage et base de validation (test)

Comme nous avons vu auparavant, l'apprentissage d'un réseau de neurones nécessite une base de données assez représentative sur l'espace de fonctionnement. Le capteur étudié travaille sur une gamme de pression de 0 à 6 bars et un intervalle de température entre -10 à 90°C, alors la base de données doit couvrir cet ensemble de valeur.

En se basant sur des résultats expérimentaux [19, 21], on construit une base de données composée de 800 exemples, chaque exemple est représenté par un vecteur $\vec{x}$ *(P, P*$^{-1}$*, T, C)* :

Où :

P → la pression, mesurée en bar, appliquée au CPS à l'instant « t »,

P^{-1} → la pression précédente, mesurée en bar, appliquée au CPS à l'instant « t-τ », avec : τ un délai temporaire,

T → la température de l'environnement où le CPS est placé, mesuré en (°C),

C → la réponse du CPS, mesuré en (pF).

Après avoir construit la base de données on procède à la phase de séparation entre la base d'apprentissage et la base de test (validation). On doit noter ici qu'il n'y a pas une règle précise concernant cette séparation, néanmoins, d'une manière générale la base de test représente entre 10 et 25% de la base de données, suivant le problème étudié. Les deux bases de données ainsi obtenues par cette séparation doivent impérativement couvrir l'espace de fonctionnement. Dans le cas pratique, si la température de travail du CPS, varie entre -10°C et 90°C, alors les deux bases englobent des différentes valeurs distribuées au long de cet intervalle, le même principe est appliqué à l'intervalle de variation de la pression (0 à 6 bar). Dans notre cas la base d'apprentissage est composée de 650 éléments, quand à la base de test elle est composée de 150 éléments.

Il est important de ne pas utiliser aucun élément de la base de test pendant toute la durée de l'apprentissage. Cette base est réservée uniquement à la mesure finale de la performance. Autrement dit, elle sert à vérifier si le réseau de neurones a une bonne performance sur les exemples qu'il n'a pas appris (base de test). Avec les réseaux de neurones, il existe toujours le risque de sur-apprentissage, c'est-à-dire, quand le réseau a pris trop de paramètres pour représenter une fonction qui n'est pas très complexe. La base de validation permet de mettre en évidence le problème s'il se présente. Le sur apprentissage se traduit par une augmentation de l'erreur sur la base de validation [7].

I.2 L'apprentissage du réseau de neurones

Une fois que les deux bases sont crée (apprentissage et validation), il faudra définir une architecture du réseau de neurones (MLP). Le nombre des neurones dans la couche de sortie est déterminé par le nombre de sorties du système à modéliser, le CPS possède une seule sortie C (capacité) donc un neurone pour la couche de sortie. Quand aux couches cachées nous avons envisagé deux couches, 4 neurones pour la première couche et 5 neurones pour la deuxième couche cachée, cette architecture a été choisie par un processus d'optimisation qui sera détaillé dans le paragraphe I.3. La fonction d'activation de la sortie est l'identité, pour la couche cachée, nous avons choisi une fonction d'activation sigmoïde.

Le schéma de la figure III.1 illustre le processus d'apprentissage du modèle ANN.

Stat. CPS → C'est le modèle statistique (résultats expérimentaux),

C_m → la valeur de la capacité mesurée,

C → la valeur de la capacité observée à la sortie du modèle ANN,

P → la pression,

T → la température,

Z^{-1} → délai temporaire,

Modèle ANN → le modèle du CPS à base du réseau de neurones.

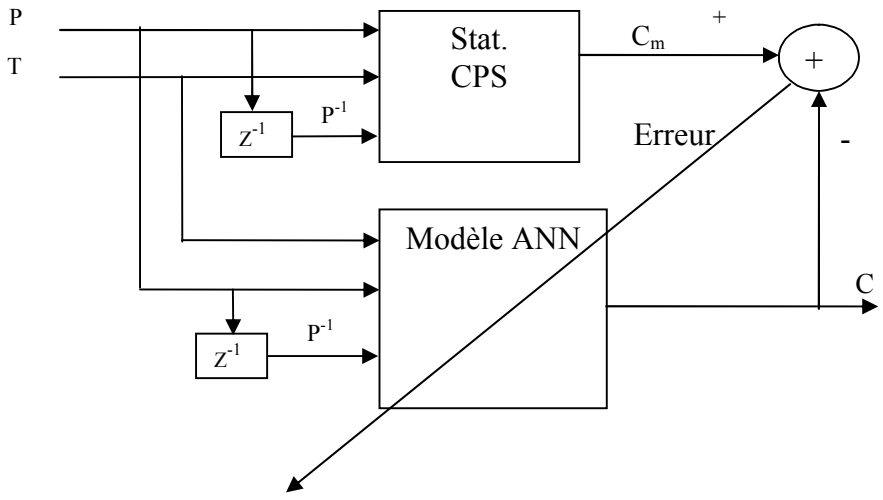


Figure III.1 : Schémas de la modélisation directe du CPS

Pour l'obtention de la pression précédente nous avons introduit un délai temporaire Z^{-1} de valeur 'τ', le choix de cette valeur est déterminée à partir du caractère dynamique de l'environnement où le CPS est placé. Si la variation de la pression est relativement lente, alors

‘τ’ prend de grandes valeurs, dans le cas inverse (variation de pression rapide) ‘τ’ prend de faibles valeurs.

Il est important de noter, que le fait d’extraire la pression précédente P^{-1} par un délai temporaire Z^{-1} est complètement différent du concept du réseaux de neurones bouclé, cité dans le chapitre I, car le délai temporaire dans les réseaux bouclés est introduit à la sorties des neurones afin de réinjecté l’état des sorties à l’entrée du réseau (boucle de retour).

I.2.1 Mise au point de l’algorithme d’apprentissage

Nous avons vu au chapitre I, que les poids de la structure MLP (perceptron multi couches), ont fait l’objet d’une optimisation par l’algorithme de la rétropropagation des erreurs. Dans ce paragraphe nous détaillerons d’avantage la mise en œuvre de cet algorithme et les résultats obtenus.

Soit un réseau MLP composé de L+1 couches (L couche cachées plus une couche de sortie) chaque couche contient R^q neurones avec :

q → un indice qui représente le numéro de la couche, si q=1 le cas de la couche de sortie, q=L+1 c’est le cas de la dernière couche cachée (dans notre cas L=2, R^1=1, R^2=5, R^3=4),

Supposons la notation suivante :

w_{ji}^{q} → le poids de connexion entre le neurone ‘ j ’ de la couche ‘q’ et le neurone ‘i’ de la couche précédente q-1,

H_0 → le nombre d’itération maximum,

h → un indice qui représente le numéro d’itération,

EQM → l’erreur quadratique moyenne (la fonction du coût) sur l’ensemble d’apprentissage N,

Y_j → la sortie ‘j’ du réseau

S → un seuil qui représente la valeur minimal de EQM qu’on désire obtenir,

N → le nombre d’éléments de la base d’apprentissage (dans notre cas N=650),

n → un indice qui représente le numéro d’élément de la base d’apprentissage.

La figure III.2 représente l’organigramme de l’algorithme de la rétropropagation des erreurs, cet organigramme est composé essentiellement de trois boucles imbriquées l’une dans l’autre. La première boucle sert au contrôle du nombre d’itération ‘h’, si se dernier dépasse le nombre d’itération maximum H_0 sans atteindre le but EQM<S, alors le programme n’a pas pu

optimisé les poids du réseaux. La deuxième boucle contrôle le nombre d'exemples d'apprentissage, si n=N alors l'apprentissage se fait sur tout l'ensemble N, ce qui permet de passer au calcul de EQM. Quand à la dernière boucle elle contrôle la propagation de l'erreur sur les différentes couches du réseau.

On remarque que la fin de l'algorithme peut être causé par la condition : l'erreur quadratique moyenne EQM est inférieure au seuil S ou bien on attend le nombre maximum d'itérations.

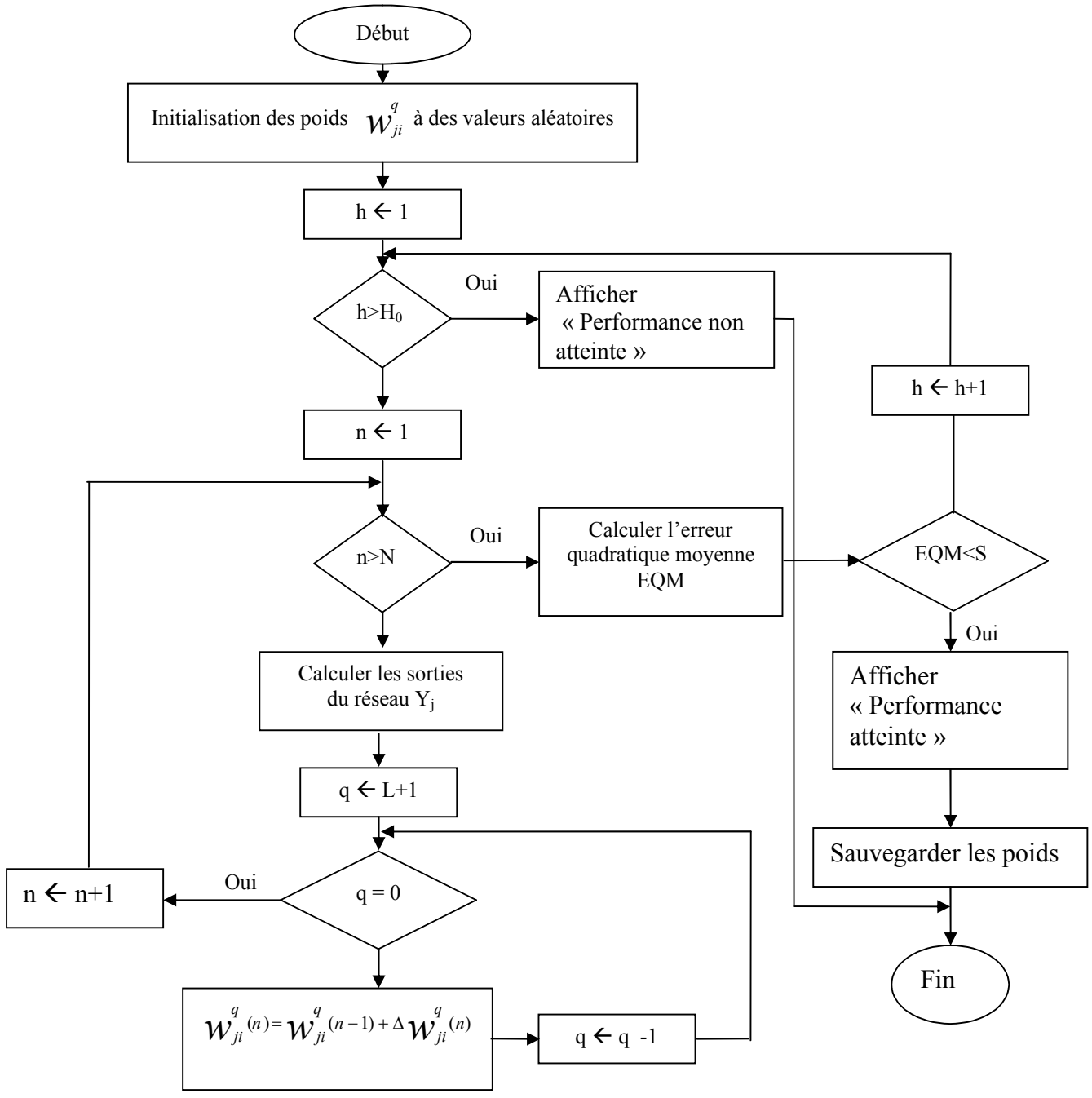


Figure III.2 : Organigramme de la rétropropagation des erreurs

L'organigramme précédant est interprété par un programme structuré en MATLAB. Afin d'évaluer l'influence du seuil S sur le nombre d'itérations nécessaire pour obtenir EQM<S. Nous avons choisi plusieurs valeurs pour S. Les figures III.3, III.4, III.5 et III.6, représentent l'évolution du EQM en fonction du nombre d'itérations pour des différentes valeurs du seuil

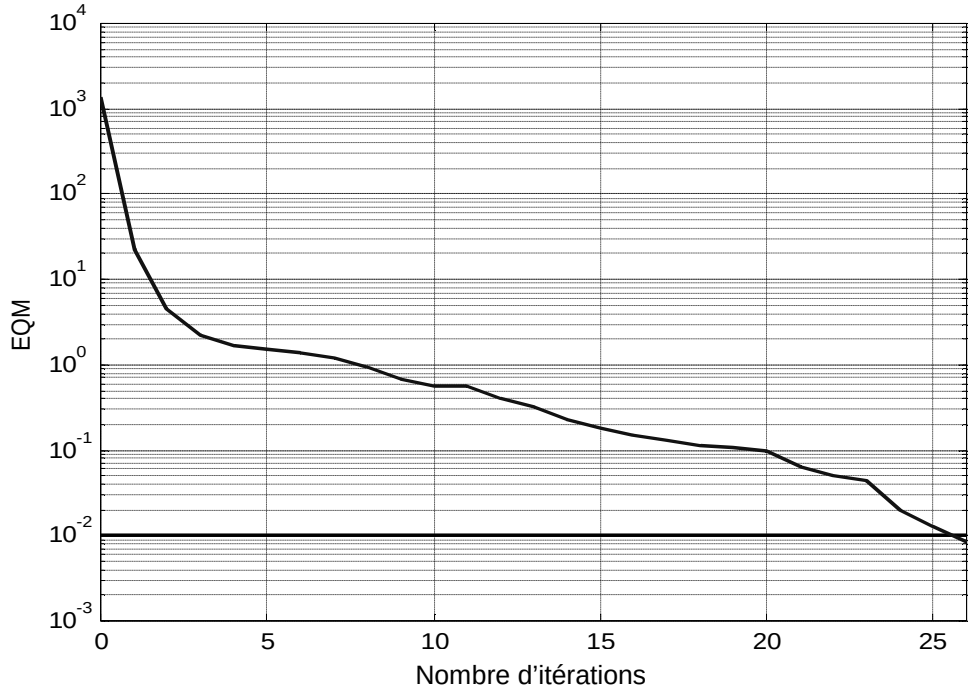


Figure III.3 : Evolution de l'erreur EQM en fonction du nombre d'itérations pour S=0.01

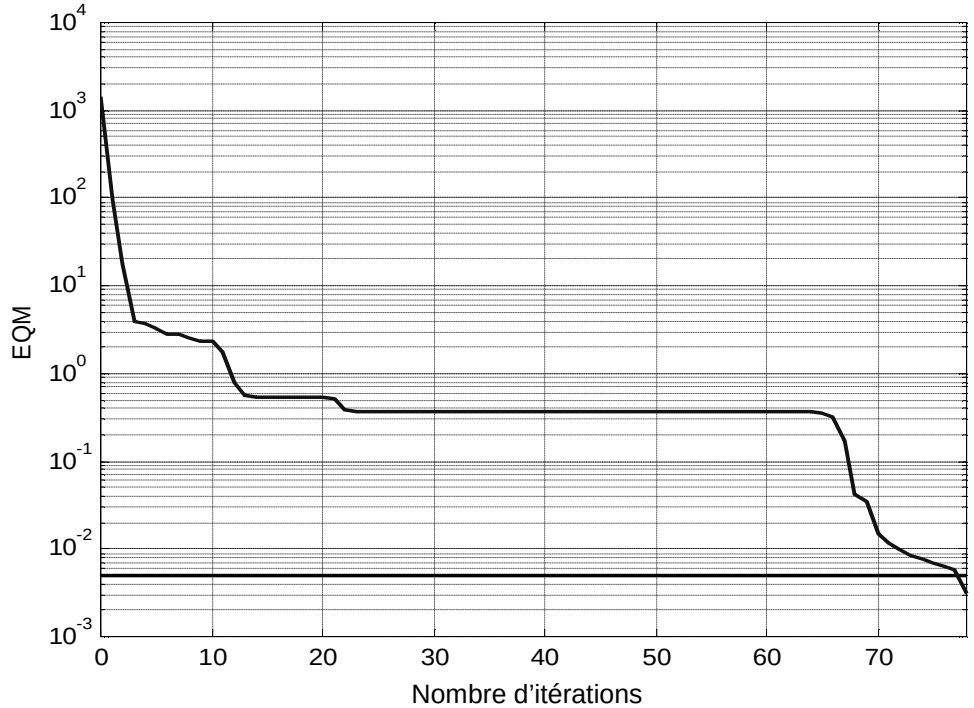


Figure III.4 : Evolution de l'erreur EQM en fonction du nombre d'itérations pour S=0.05

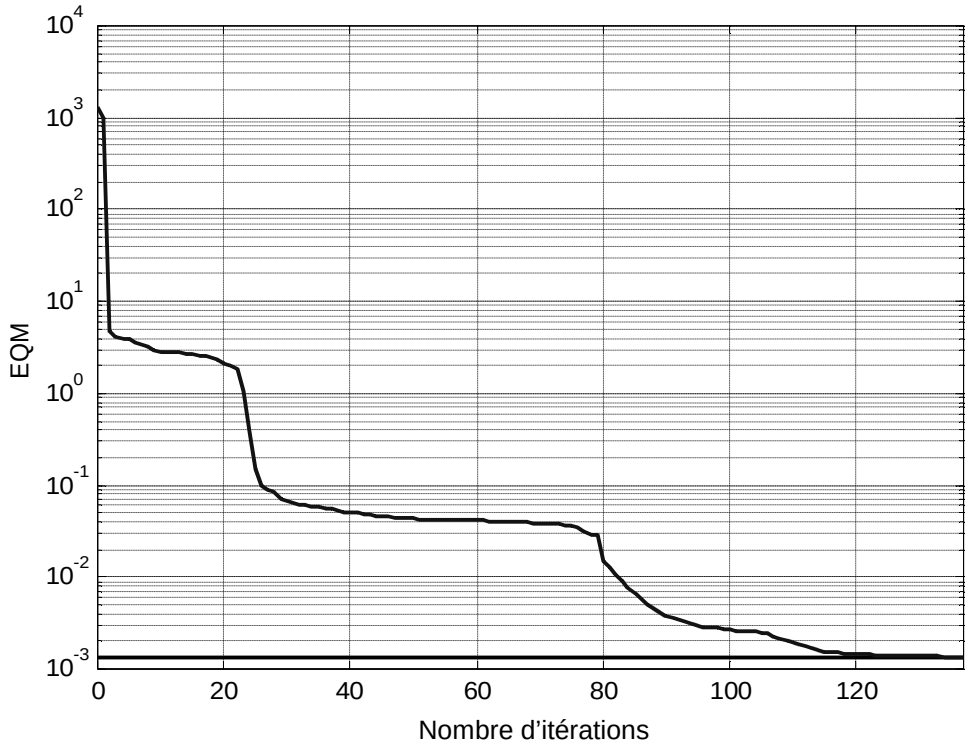


Figure III.5 : Evolution de l'erreur EQM en fonction du nombre d'itérations pour S=0.001

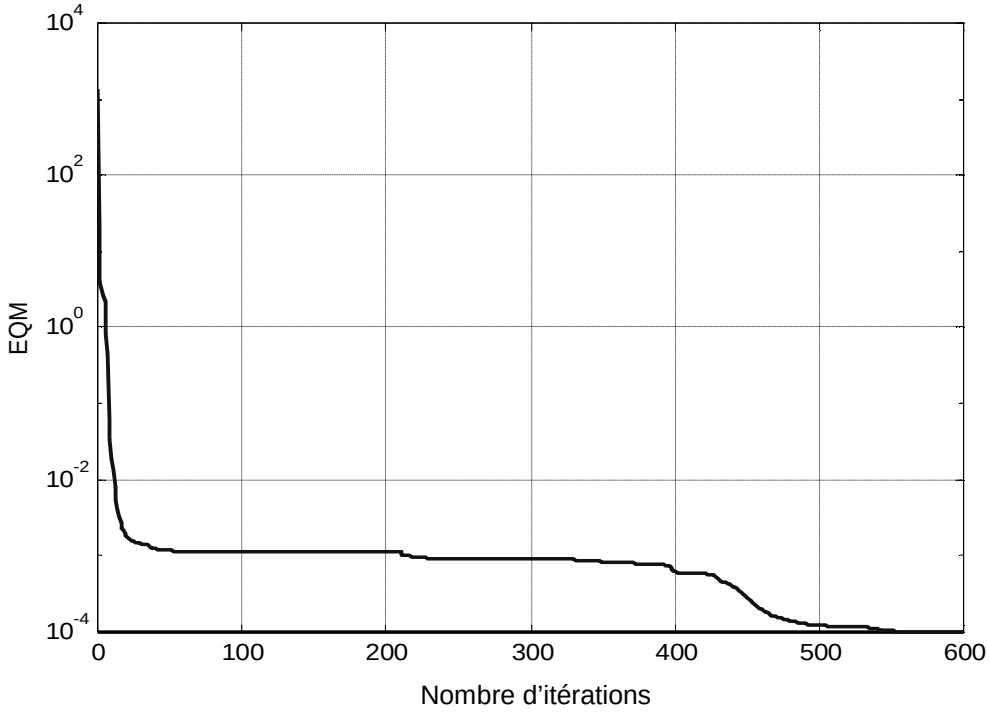


Figure III.6 : Evolution de l'erreur EQM en fonction du nombre d'itérations pour S=0.0001

On remarque que plus le seuil 'S' est faible, plus le nombre d'itérations est important et inversement. Le tableau III.1 résume les résultas obtenus, ces résultats concordent avec la théorie.

Seuil S (EQM<S)	Nombre d'itérations
0.0001	580
0.001	137
0.002	95
0.005	78
0.01	28
0.05	23

Tableau III.1 : Nombre d'itérations maximum en fonction du seuil

I.3 Optimisation de l'architecture

La phase d'optimisation est une phase primordiale da la conception des ANN. Il s'agit de trouver le nombre optimal des couches cachées et le nombre de neurones par couches pour que le modèle ANN exprime fidèlement la variation de la réponse du CPS.

Notre choix s'est porté sur deux couches cachées, il faudra encore déterminer le nombre de neurones dans chaque couche cachée. Nous avons testé le réseau avec un nombre de neurones entre 1 et 6 neurones pour chaque couche, puis nous avons observé la variation de l'erreur globale (apprentissage et test). L'organigramme de la figure III.7 montre l'enchaînement du processus d'optimisation. La base 1 et la base 2 représentent respectivement la base d'apprentissage (650 exemple) et la base de test (150 exemple). Les performances du processus sont fixés à un seuil d'apprentissage S_0=0.0001 (EQM<S_0) et une erreur globale du test inférieur à 0.005. Quand au nombre d'itérations maximum il est fixé à 1000 itérations.

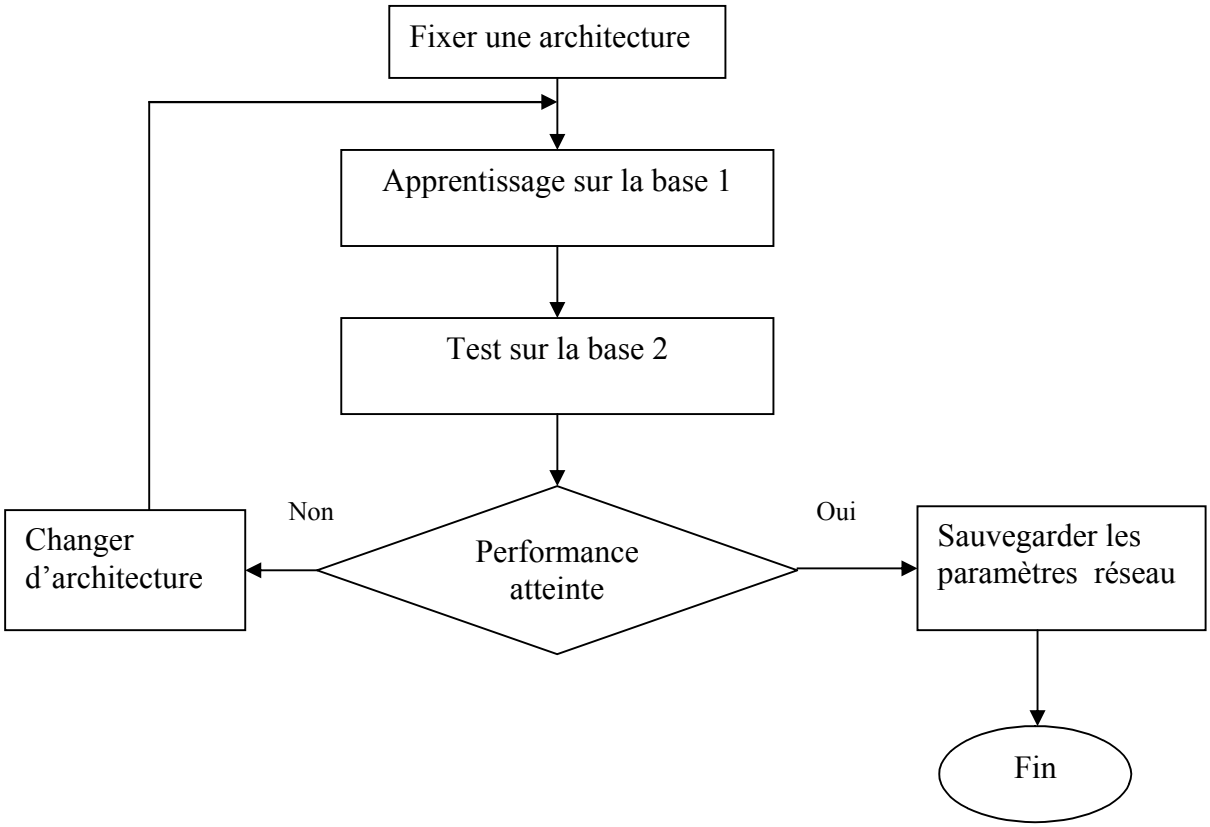


Figure III.7 : Processus d'optimisation

Après avoir tester plusieurs possibilités d'architecture et mesurer à chaque fois les performances du modèle ANN, nous avons reporté ses résultats sur le tableau III.2. Pour les architectures numérotées de 1 à 7 on remarque que l'algorithme n'a pas pu atteindre le seuil S_0=0.0001 fixé auparavant, or que, à partir de l'architecture numéros 8 l'algorithme d'apprentissage a atteint le seuil S_0, cependant l'erreur de test converge vers le minimum dans l'architecture numéros 10. Les résultas, ainsi obtenus, montre l'existence de différences notables sur les performances du modèle ANN par rapport à différentes architectures qui justifient un tel choix. Alors nous avons retenu l'architecture numéros 10.

N°	Nombre de neurones pour la première couche cachée	Nombre de neurones pour la deuxième couche cachée	EQM Test	EQM Apprentissage
01	3	0	1458.5	0.0001 (non atteinte)
02	4	0	1322.6	0.0001 (non atteinte)
03	1	2	4.4862	0.0001 (non atteinte)
04	1	5	1.5528	0.0001 (non atteinte)
05	2	2	4.1099	0.0001 (non atteinte)
06	3	2	0.25	< 0.0001 (non atteinte)
07	4	2	1.9700	< 0.0001 (non atteinte)
08	4	3	0.09	< 0.0001 (atteinte)
09	4	4	0.1270	< 0.0001 (atteinte)
10	4	5	0.0011	< 0.0001 (atteinte)
11	5	5	0.02	< 0.0001 (atteinte)
12	5	6	0.05	< 0.0001 (atteinte)
13	6	6	0.9	< 0.0001 (atteinte)

Tableau III.2 : Variation de l'erreur globale du test en fonction de différentes architectures

I.4 Modèle finale

Pour modéliser le capteur CPS nous avons résumé sur le tableau III.3 les paramètres optimisés du modèle ANN.

Paramètre	Valeur optimisée			
Architecture	Feed-forward MLP (perceptron multi-couches)			
Couche cachée	2			
Règle d'apprentissage	Rétropropagation des erreurs (Back propagation)			
Nombre de Neurones	Couche d'entrée			3
	1ère couche			4
	2ème couche			5
	Couche de sortie			1
La fonction de transfert	1ère couche			Sigmoid
	2ème couche			Sigmoid
	Couche de sortie			Linéaire
Définition des entrées		P (bar)	P^{-1} (bar)	T (°C)
	max	6	6	90
	min	0	0	-10
Définition des sorties		C_p (pF)		
	max	42		

	Min	34	
EQM du test	0.0011		
EQM d'apprentissage	< 0.0001		
Erreur maximale sur un exemple	< 0.1 % (FS)		
Base des données	Base d'apprentissage		650 (P, P^{-1}, T, C)
	Base de test et Validation		150 (P, P^{-1}, T, C)

Tableau III.3 : Paramètres optimisés du réseau de neurones

La figure III.8 montre la représentation symbolique du modèle ANN optimisé.

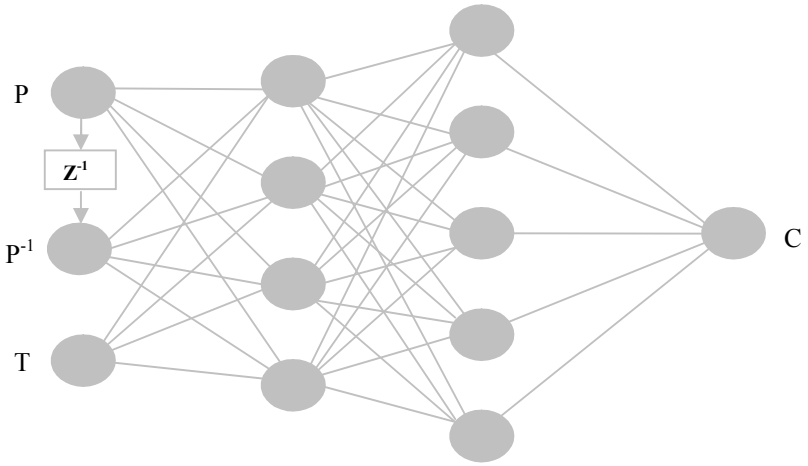


Figure III.8 **:** *l'architecture optimisée*

Nous avons conçu un modèle à base des réseaux de neurones (Modèle ANN), pour modéliser le comportement du CPS dans un enivrement dynamique, dans un but que ce modèle exprime le terme d'hystérésis à l'inverse du modèle analytique. Pour illustrer cette différence on fait varier la pression dans un cycle (croissant, décroissant), puis on observe la variation de la capacité (réponse) du CPS. La figure III.9 montre la différence entre le modèle analytique et le modèle ANN. Nous verrons également dans la section suivante (implantation du modèle) que le modèle ANN exprime fidèlement le terme de la non linéarité (*NL(P,T)*) de la réponse du CPS et sa dépendance de la température.

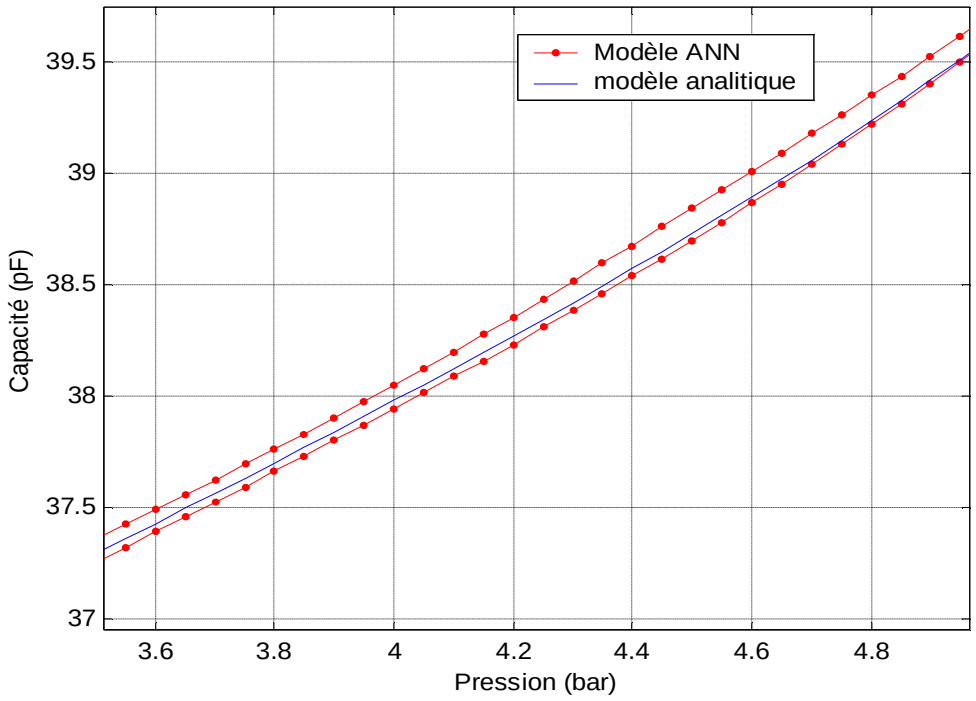


Figure III.9 : Modèle analytique et le modèle ANN à 20 °C

II. Implantation du modèle ANN sur SPICE

L'évaluation des performances d'un circuit avec le logiciel SPICE avant sa réalisation nous permet d'obtenir un temps de conception réduit et limiter ainsi les coûts de production.

La qualité des résultats obtenus de la conception et de la simulation des circuits électroniques, dépendent de la précision des modèles utilisés.

Dans se cadre, nous proposons une implantation du modèle ANN sur le simulateur SPICE, elle va nous permettre de tester les performances du modèle ANN du CPS.

II.1 Présentation du simulateur SPICE

SPICE (**S**imulation **P**rogram with **I**ntegrated **C**ircuit **E**mphasis) est un simulateur électrique standard qui permet l'analyse statique, transitoire et dynamique des circuits électronique. Les circuits peuvent contenir des différents dispositif électronique (résistances, condensateurs, inductances, des sources de tension et de courant, ainsi que des modèles de dispositifs actifs : diodes, transistors bipolaires, transistors à effet de champ à jonction…etc.). Le simulateur a été développé à l'université de Berkeley à partir des années 1967 et jusqu'en 1990. La première version BIAS3 en 1967 pour l'analyse statique des transistors bipolaires puis CANCER (Computer **A**nalysis of **N**onlinear **C**ircuits, **E**xcluding **R**adiation) en 1970, SPICE1

en 1973, SPICE2 en 1975 et SPICE3 en 1980. Il est actuellement figé, avec sa dernière version SPICE3F5. C'est un simulateur électrique de deuxième génération fonctionnellement construit sur quatre méthodes numériques de base [23]:

1. Une méthode numérique de formulation des équations: la MNA,
2. Une méthode numérique de résolution d'un système linéaire: la décomposition LU,
3. Une méthode numérique d'intégration: trapézoïdale ou Gear2,
4. Une méthode numérique d'analyse non linéaire: Newton-Raphson.

Aujourd'hui, la plupart des simulateurs électriques, sont des descendants du simulateur de deuxième génération SPICE. Les simulateurs commerciaux les plus utilisés actuellement sont: HSPICE créé par Meta-Software, ELDO créé par Anacad, SPECTRE créé par Cadence, PSPICE créé par Micro-sim, SPICE3F5 distribué par l'université de Berkeley.

II.2 Netlist d'implantation du modèle ANN du CPS

Le modèle ANN du CPS a été exprimé par le biais des composants ABM (Analog Behavioral Modelling) de la bibliothèque de PSPICE, où chaque neurones du réseau est implanté dans une boite ABM. Le délais temporaire est implanté lui aussi sur une boite ABM. On suppose la notation suivante :

- B_{ni} : la matrice du bias (seuil W_0)

n : numéro de la couche

i : numéro du neurone

- W_{nji} : la matrice des poids

n : numéro de la couche

j : numéro du neurone

i : numéro du neurone de la couche précédente

t : le délais temporaire

```
*------------------------- Modèle CPS -----------------------------
*--------- la sortie C  ‘ pin 1, pin2 ’
.EXTERNAL OUTPUT 1
.EXTERNAL OUTPUT 2
*--------------------  la couche d’entrée
.EXTERNAL INPUT P
.EXTERNAL INPUT T
*----------      (retard Z-1)   P → P-
E_CPS_E1        CPS_P- 0 LAPLACE { V(P) } = { exp(-
+   t*S) }
```

```
*--------------------  la première couche cachée
E_ABM1        N41207 0 VALUE { 1/(1+exp(-
+  (B11+W111*V(P)+W112*V(P-)+ W113*V(T) )))
+  }
E_ABM2        N41511 0 VALUE { 1/(1+exp(-
+   (B12+W121*V(P)+W122*V(P-)+ W123*V(T))))
+   }
E_ABM3        N41208 0 VALUE { 1/(1+exp(-
+  (B13+W131*V(P)+W132*V(P-)+ W133*V(T) )))
+  }
E_ABM4        N41512 0 VALUE { 1/(1+exp(-
+   (B14+W141*V(P)+W142*V(P-)+ W143*V(T))))
+   }
*--------------------  la deuxième couche cachée
E_ABM5        N41663 0 VALUE {
+   1/(1+exp(-(B21+
+   W211*V(N41207)+W212*V(N41511))))+
+   W213*V(N41208)+W214*V(N41512))))    }
E_ABM6         N41696 0 VALUE {
+   1/(1+exp(-(B2+
+   W221*V(N41207)+W222*V(N41511))))+
+   W223*V(N41208)+W224*V(N41512))))    }
E_ABM7        N41761 0 VALUE {
+   1/(1+exp(-(B23+
+   W231*V(N41207)+W232*V(N41511))))+
+   W233*V(N41208)+W234*V(N41512))))    }
E_ABM8    N41973    0 VALUE {
+   1/(1+exp(-(B24+
+   W241*V(N41207)+W242*V(N41511))))+
+   W243*V(N41208)+W244*V(N41512))))    }
E_ABM9        N41725 0 VALUE {
+   1/(1+exp(-(B25+
+   W251*V(N41207)+W252*V(N41511))))+
+   W253*V(N41208)+W254*V(N41512))))    }
---------------- la couche de sortie
E_ABM10        N394804 0 VALUE {
+   B31+ W311*V(N41663)+W312*V(N41696) +
+   W313*V(N41761) +W314*V(N41973)+
+   W315*V(N41725)   }
* -------   la mise en échelle (C  en pF)
C_ABM11        1 2 VALUE { (V(N394804) * 1E-12)    }
*-------------- END
```

II.3 Convertisseur Capacité / Tension (SCI)

La variation de la capacité du CPS en fonction de la pression ne peut pas être directement exploitée par une chaîne de mesure ou de contrôle. En effet, cette variation est passive et elle doit être convertie à une autre grandeur active telle que une tension ou un courant. Donc on fait appel à une interface SCI (Switched Capacitor Interface), cette interface délivre une tension proportionnelle à la capacité du CPS (Figure III.10).

Lorsque Φ=0 la capacité C(P) ce charge avec la tension de V_{ref} au moment où la capacité Cs se décharge à la masse. Pour Φ=1 la charge $Q=V_{ref}.C_p$ stockée dans la capacité C(P) est transférée à la capacité Cs ($V_0.C_s=V_{ref}.C_P$) qui produit une tension de sortie $V_0 = K.C_p$ avec $K=V_{ref} / Cs$. Finalement la tension V_0 est amplifiée avec un gain en tension 'G' ce qui donne $V_p = G. V_0$.

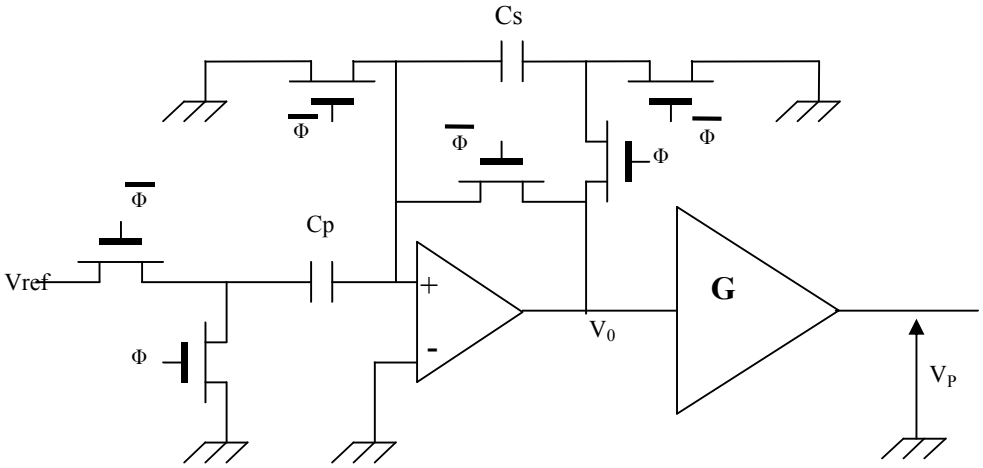


Figure III.10 : Schéma électrique du SCI

II.4 Résultats de simulation

En premier lieu, nous avons placé le CPS dans un simple circuit (charge et décharge d'une capacité), composé d'une résistance et un générateur de tension carré (impulsion) monté en série avec le CPS (Figure III.11), dans le but est de confirmer si le capteur CPS se comporte comme une capacité. En effet, le capteur CPS est caractérisé par la variation d'une capacité C en fonction de la pression appliquée et de la variation de la température.

Choisissons des valeurs fixes pour la pression et pour la température (par exemple P=3.22 bar et T=45,6 °C), ensuite on procède à une simulation temporaire « Time domain » les résultats obtenus sont représentés sur la figure III.12. L'analyse des résultats permet de conclure que le capteur se comporte comme une capacité ordinaire.

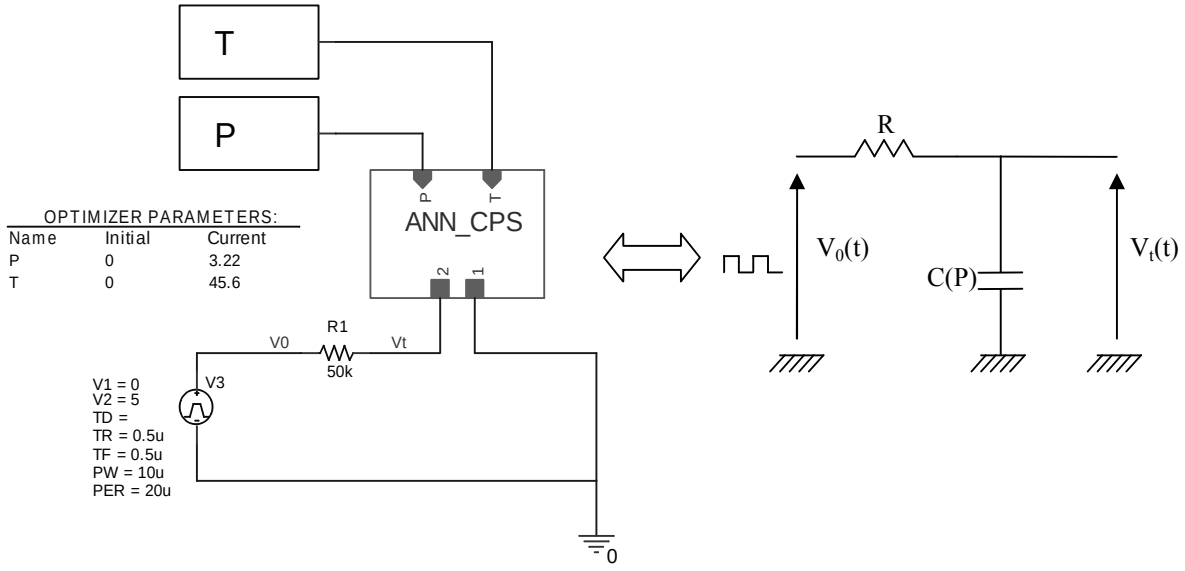


Figure III.11 : Schéma de simulation environnement PSPICE

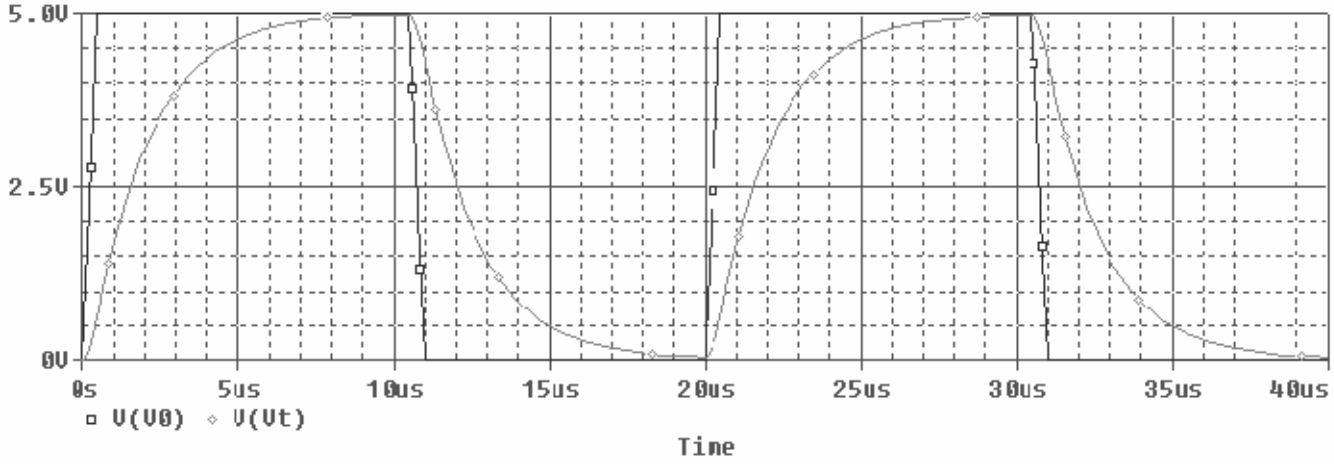


Figure III.12 : Résultats de simulation

Le deuxième circuit de simulation proposé est celui de la figure III.13, plus pratique que le premier, il s'agit d'associer le CPS avec l'interface SCI (étudiée précédemment). SCI joue le rôle d'un circuit de conditionnement pour le CPS, généralement cette interface est la plus utilisée pour les capteurs de type capacitif. A partir de ce circuit on procède à une série de test.

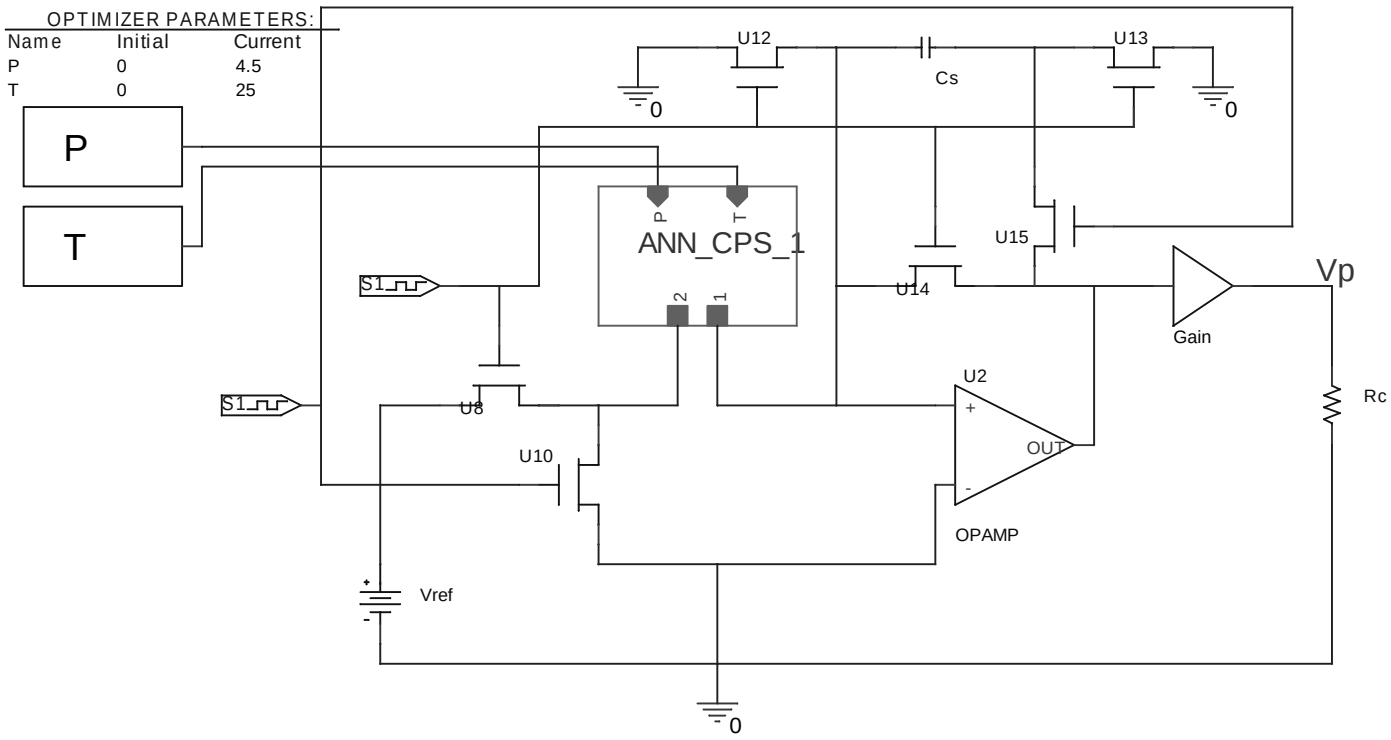


Figure III.13 : Schéma de simulation du CPS associé à l'interface SCI

→ Fixons la température à 25°C (T=25) et on fait varier la pression dans une plage de 0 à 6 bar**,** par l'analyse « DC- SWEEP/ Parametric Sweep » on obtient les résultats représentés sur la figure III.14.

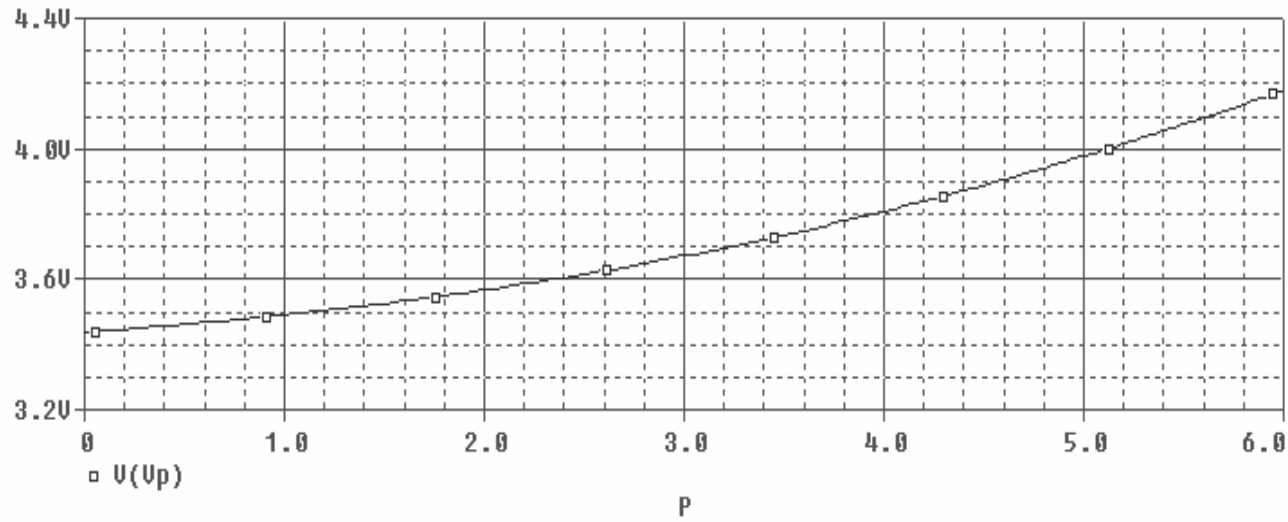


Figure III.14 : Variation de la tension de sortie Vp en fonction de la pression P à une température de 25°C

→ Par analogie avec le 1er test, on procède au deuxième. Fixons la pression à 4.5 bar puis faisant varier la température dans un intervalle de -10 à 90°C. La figure III.15 illustre le résultat obtenu.

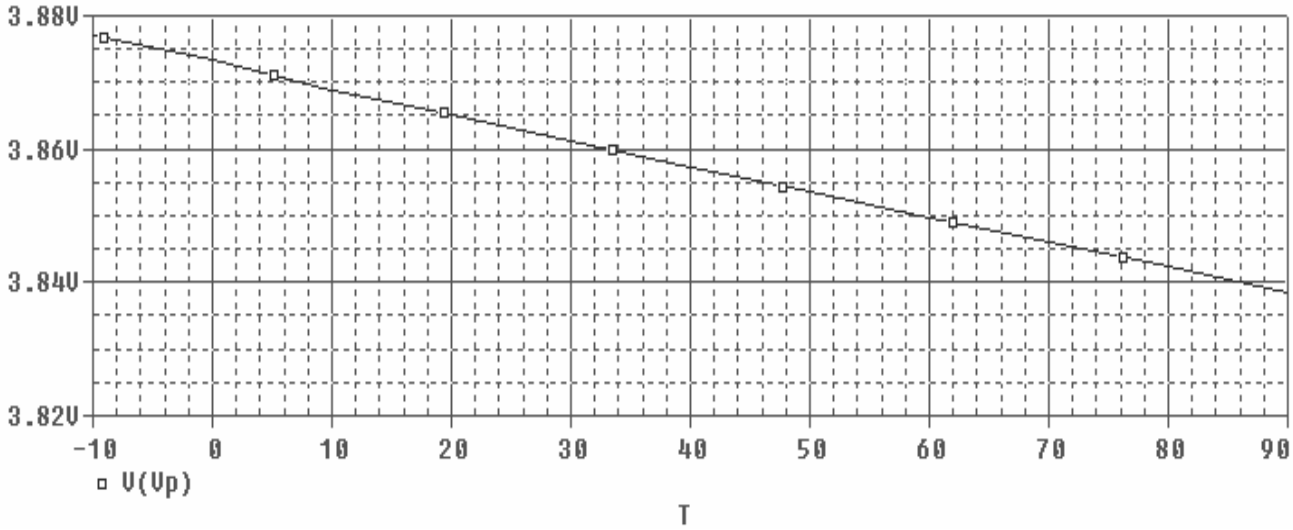


Figure III.15 : Variation de Vp en fonction de la température T à une pression de 4.5 bar

→ Afin de tester le comportement dynamique du modèle ANN, nous avons fait varier la pression en fonction du temps selon un cycle de 20s, (10s pour le cycle en pression croissante et 10s pour le cycle en pression décroissante), la figure III.16 et la figure III.18, toute en maintenons la température fixe (T= 25° C). Utilisons l'analyse « Time Domain » du SPICE on obtient les résultats représenté aux figures III.17 et III.19. On remarque que pour des mêmes valeurs de pressions (cycle croissant, cycle décroissant), on obtient deux valeurs différentes de capacité, ce qui est expliqué par l'apparition de l'effet d'hystérésis en régime dynamique.

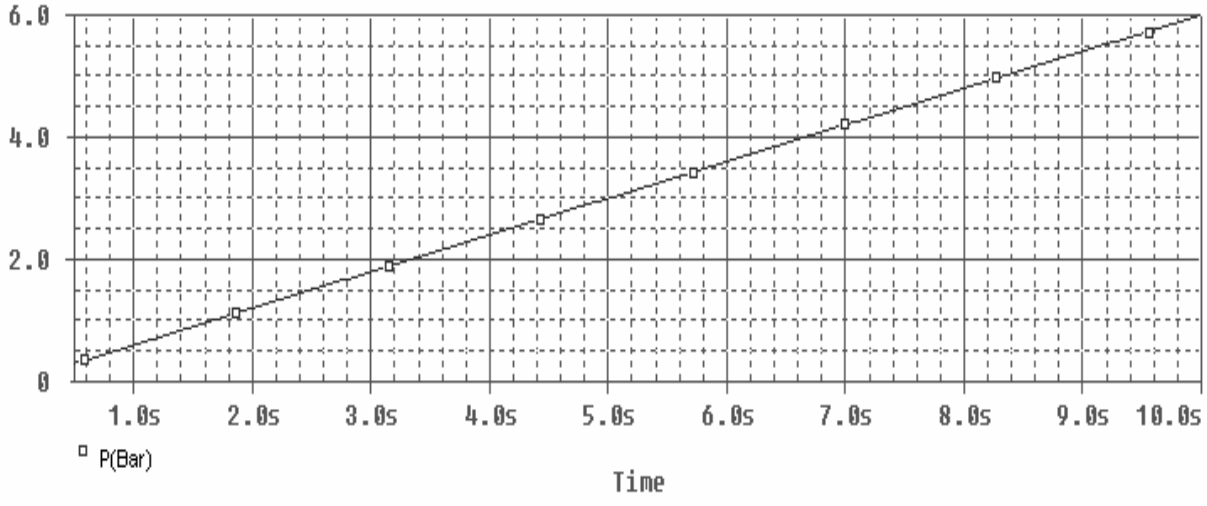


Figure III .16 : Variation de la pression P en fonction du temps (cycle croissant)

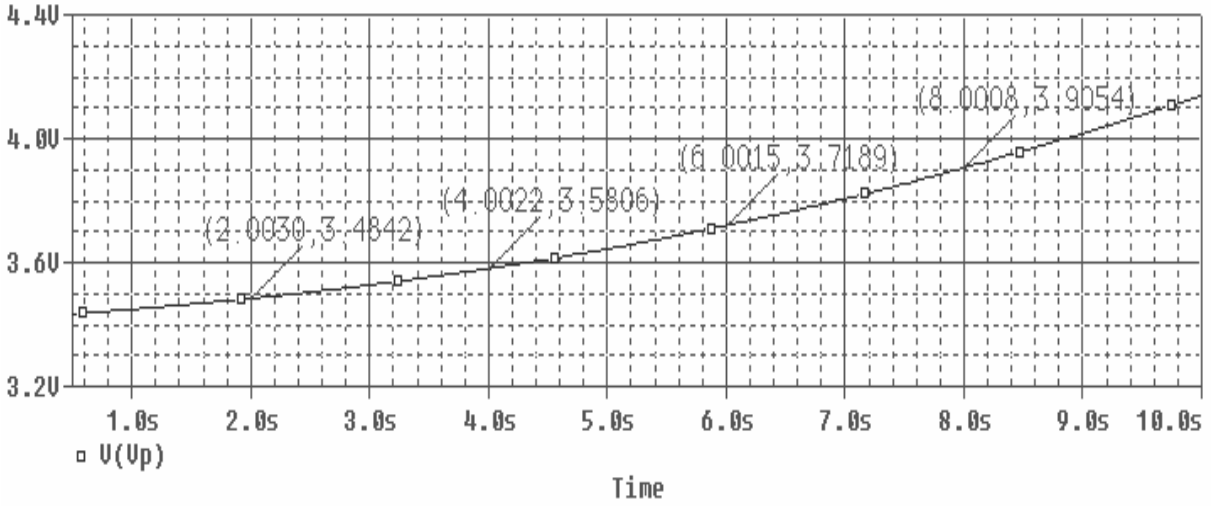


Figure III.17 : Variation de Vp en fonction du temps (cycle croissant)

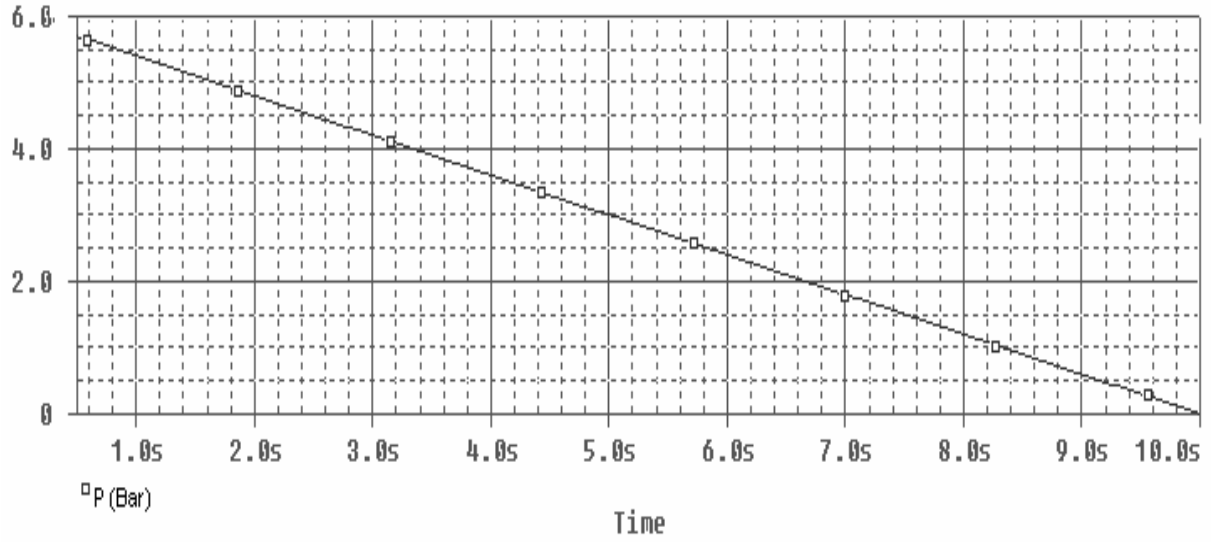


Figure III.18 : Variation de la pression P en fonction du temps (cycle décroissant)

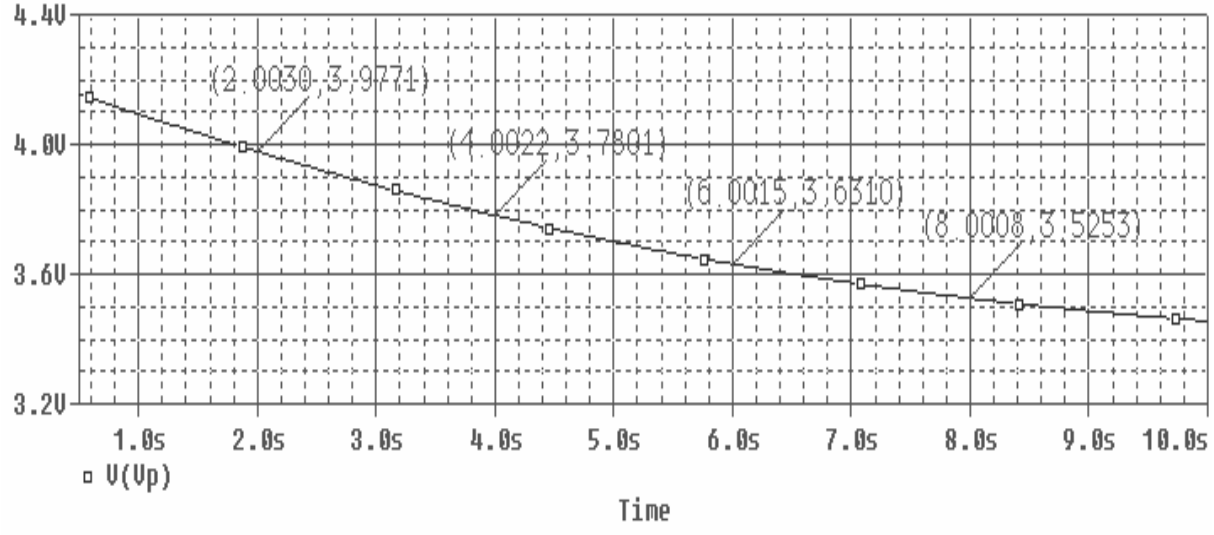


Figure III.19 : Variation de Vp en fonction du temps (cycle décroissant)

En analysons les résultats obtenus des tests précédents, on conclue que : le modèle ANN conçu et implanté exprime fidèlement le comportement du CPS. Il tient compte de la non linéarité, la dépendance à la température et de l'effet d'hystérésis.

III. Conclusion

Le modèle ANN développé au cours de ce chapitre présente l'avantage, par rapport au modèle analytique étudié au chapitre II, la reproduction fidèle du comportement du CPS. L'implantation de ce modèle sous forme d'un composant électronique dans la bibliothèque du simulateur SPICE nous a permis d'évaluer ses performances dans un environnement électrique. En plus, il joue le rôle d'une plate forme dans le développement d'un composant de correction de la sortie du CPS dans un environnement dynamique, qui sera exposé au chapitre IV.

Introduction

Dans le cadre de ce mémoire, il nous a été assigné de concevoir un composant de mesure permettant la correction de la réponse d'un capteur de pression capacitif (CPS), afin d'obtenir une sortie linéaire par rapport à la pression appliquée et complètement indépendante de la température. De plus, ce composant doit délivrer une réponse unique pour une même pression appliquée, quelque soit la valeur de la pression précédente, ce qui se traduit par l'élimination de l'effet d'hystérésis. L'ensemble : capteur, circuit de conditionnement et le composent de correction, est dit capteur intelligent.

Plusieurs techniques sont employées pour la correction de la réponse du capteur CPS. Concernent la linéarisation de la réponse, plusieurs travaux ont été effectués [24,25], en exploitant certains oscillateurs où le capteur est employé comme élément passif. Dans ce cas la sortie du système est la fréquence d'oscillation ou bien la période, cette sortie est linéaire en fonction de la pression appliquée à 1% par rapport à la réponse pleine échelle (FS). Une autre technique repose sur le placement de deux capteurs CPS différents, sur une structure push-pull type capacitif [26], cette méthode donne une sortie linéaire, dans le taux de la non linéarité est inférieur à 0.5% par rapport à la réponse pleine échelle (FS). Cependant, ces techniques de linéarisation supposent que le capteur travaille à température ambiante, dans ce cas l'erreur due à l'effet de la température n'est pas incluse dans la correction de la réponse.

En ce qui concerne la compensation de l'effet de la température on trouve des travaux [27], qui consistent à mesurer la température par un capteur de température et utiliser cette valeur dans une boucle thermique de compensation. Pour obtenir un système de linéarisation et de compensation simultanément, on fait appel aux techniques de traitement numérique de signal par le biais des circuits programmable tel que les microcontrôleurs [28].

Dans notre cas, nous proposons les réseaux de neurones comme un composant de correction de la sortie du CPS. Nous avons noté plusieurs travaux qui traite l'utilisation des ANN pour la correction de la réponse du CPS : [3, 4, 5, 6], ces travaux donnent de très bon résultats, mais l'effet d'hystérésis n'est pas pris en compte.

Il est important de mentionner, que la conception d'un modèle à base des réseaux de neurones pour la correction de la réponse du CPS, diffère complètement de la modélisation par les réseaux de neurones du CPS (étudiée au chapitre III). En effet, le modèle ANN précédemment développé approxime une fonction mathématique qui exprime la capacité du CPS en fonction de la pression appliquée, de la pression précédente et de la température. Cependant, le modèle ANN du capteur intelligent (modèle inverse), présenté dans se chapitre,

joue le rôle d'un composant de mesure, implanté dans un circuit qui permet de corriger la sortie du CPS. Pour ne pas confondre entre les deux modèles, on suppose la notation suivante :

→ Le modèle à base du réseaux de neurones du capteur CPS est dit : modèle ANN

→ Le modèle ANN du capteur intelligent ou le modèle inverse est dit : modèle INV-ANN

I. Développement du modèle à base des réseaux de neurones (INV-ANN)

L'utilisation des réseaux de neurones comme des composants de mesure, permettant de corriger la réponse des capteurs, a connu un essor considérable au cours de ces dernières années. En effet, les ANN présentent l'avantage d'une grande adaptation aux différents problèmes causé par la non idéalité des capteurs [29].

Nous avons examiné le problème de la non linéarité, la dépendance à la température et l'effet d'hystérésis de la réponse du capteur CPS (chapitre II), pour lequel nous avons développé un modèle à base du réseau de neurones pour le CPS dans milieu dynamique (chapitre III). Après avoir mis au point le modèle du CPS à base des réseaux de neurones nous pouvons alors, construire une deuxième base de données (à base du premier modèle) pour l'apprentissage du modèle INV-ANN. Cette dernière est composée de 1200 éléments pour l'apprentissage et 250 éléments pour le test et la validation, arrangés de la manière suivante : (V, V_p^{-1}, T, P). On remarque que la dimension de la deuxième base d'apprentissage obtenue, est presque deux fois la dimension de la première base d'apprentissage (modèle ANN). Pour le développement du modèle INV-ANN, nous avons utilisé l'interface MATLAB, le schémas de l'apprentissage du capteur intelligent est illustré en figure IV.1.

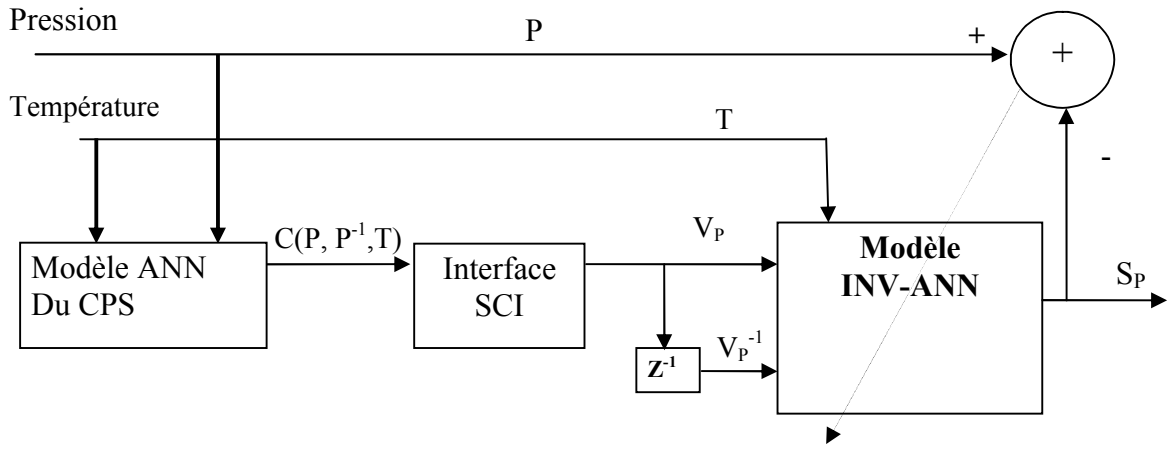


Figure IV.1 : Schéma de modélisation du capteur intelligent

Où :

$C(P,P^{-1},T)$	→	Sortie du CPS
V_P	→	Tension de sortie de l'interface SCI
P	→	Sortie désirée
S_P	→	Sortie du modèle ANN
Z^{-1}	→	Un délai temporaire
Modèle INV-ANN	→	Modèle du capteur intelligent (modèle inverse)

Par analogie avec la modélisation du CPS (chapitre III), on procède à la modélisation du capteur intelligent. A la fin de la phase d'apprentissage et d'optimisation on obtient une architecture semblable à celle du modèle ANN. Les algorithmes d'apprentissage et d'optimisation ont été largement étudiés au chapitre III. La figure IV.2 illustre l'évolution de l'erreur globale sur la base d'apprentissage en fonction du nombre d'itérations.

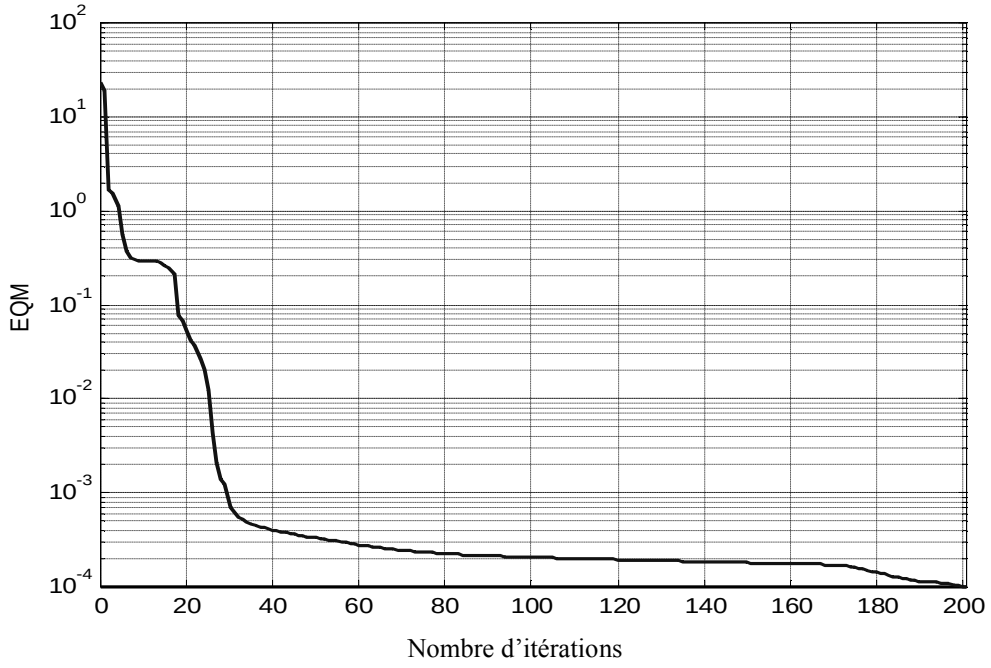


Figure IV.2 : Evolution de l'erreur EQM en fonction du nombre d'itérations pour S=0.0001

Le tableau IV.1 résume tous les paramètres de l'architecture et d'optimisation. Quand à la figure IV.3 elle donne une représentation symbolique du modèle INV-ANN ainsi optimisé.

Paramètre	Valeur optimisée			
Architecture	Feed-forward MLP (perceptron multi-couches)			
Couche cachée	2			
Règle d'apprentissage	Rétropropagation des erreurs (Back propagation)			
Nombre de Neurones	Couche d'entrée			3
	1ère couche			4
	2ème couche			5
	Couche de sortie			1
Fonction de transfert	1ère couche			Sigmoid
	2ème couche			Sigmoid
	Couche de sortie			Linéaire
Définition des entrées		V_p (mV)	V_P^{-1} (mV)	T (°C)
	Max	4100	4100	90
	Min	3500	3500	-10
Définition des sorties		S_p (pF)		
	Max	4100		
	Min	3500		
EQM du test	0.002			
EQM d'apprentissage	< 0.0001			
Erreur maximale sur un exemple	< 0.1 % (FS)			
Base de données	Base d'apprentissage	1200 (V_P, V_P^{-1}, T, S_p)		
	Base de test et Validation	250 (V_P, V_P^{-1}, T, S_p)		

Tableau IV.1 : Paramètres optimisés du réseau de neurones du modèle INV-ANN

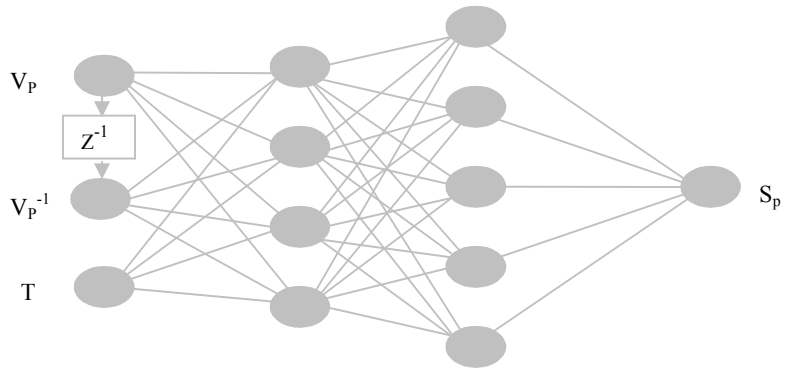


Figure IV.3 **:** *Architecture optimisée*

II. Tests et résultats

Afin de tester les performances de notre capteur intelligent (figure IV.4), nous avons envisagé quatre tests différents, qui englobe tous les problèmes précédemment étudiés (la non linéarité, la dépendance à la température et l'effet de l'hystérésis) ; finalement on valide le

modèle du capteur intelligent par un test pour des valeurs arbitraire attribué à la pression et à la température.

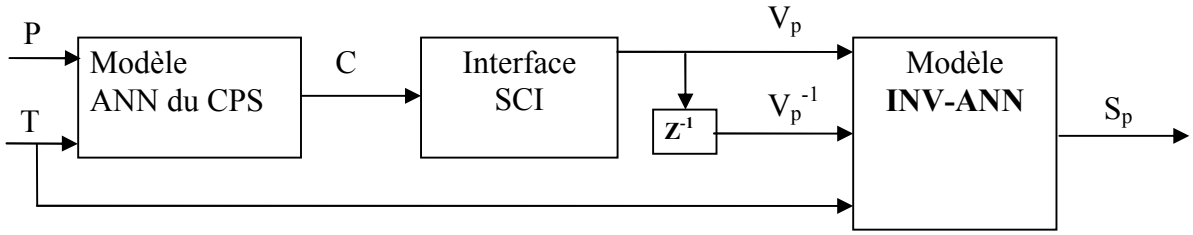


Figure IV.4: Schémas du capteur intelligent

II.1 Test pour un cycle de pression

Fixons la valeur de la température à 20°C, puis faisons varier la pression sur un intervalle de 1 jusqu'à 6 bar, les résultats de cette expérience sont représentés sur les figures IV.5 et IV.6. On remarque que la non linéarité du capteur intelligent est très faible par rapport à celle du CPS.

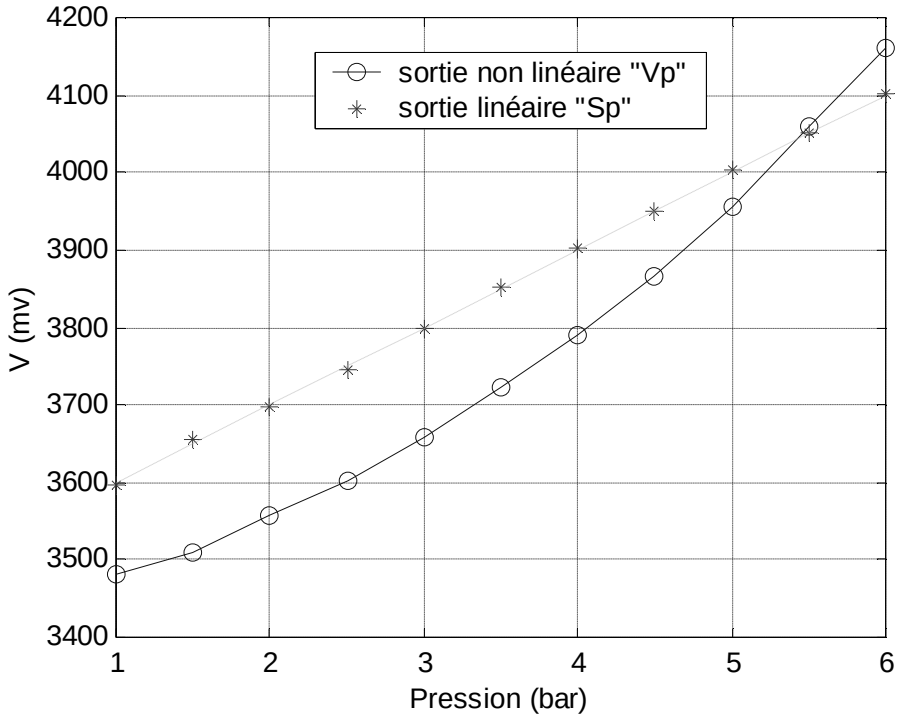


Figure IV.5 : Réponse du CPS&SCI et réponse du capteur intelligent à T=20°C

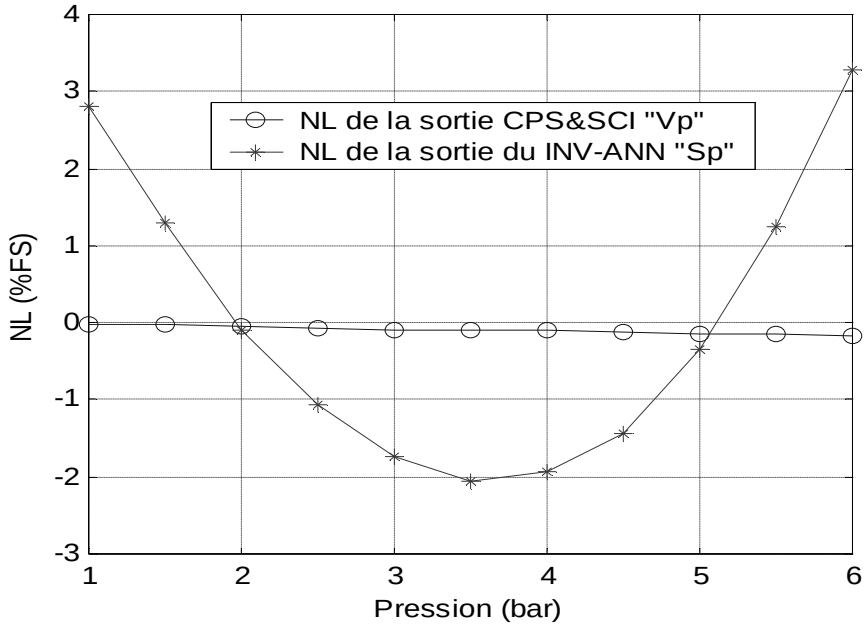


Figure IV.6 : Comparaison entre la non linéarité de la réponse du CPS&SCI et celle du capteur intelligent

II.2 Test pour un cycle de température

Ce test consiste à tester l'indépendance à la température du capteur intelligent. En premier lieu on maintient la pression constante (4.5 bar), puis on fait varier la température sur une plage de -10 à 90°C. En examinons les résultas da la figure IV.7, on remarque que le capteur intelligent et complètement indépendant de la température.

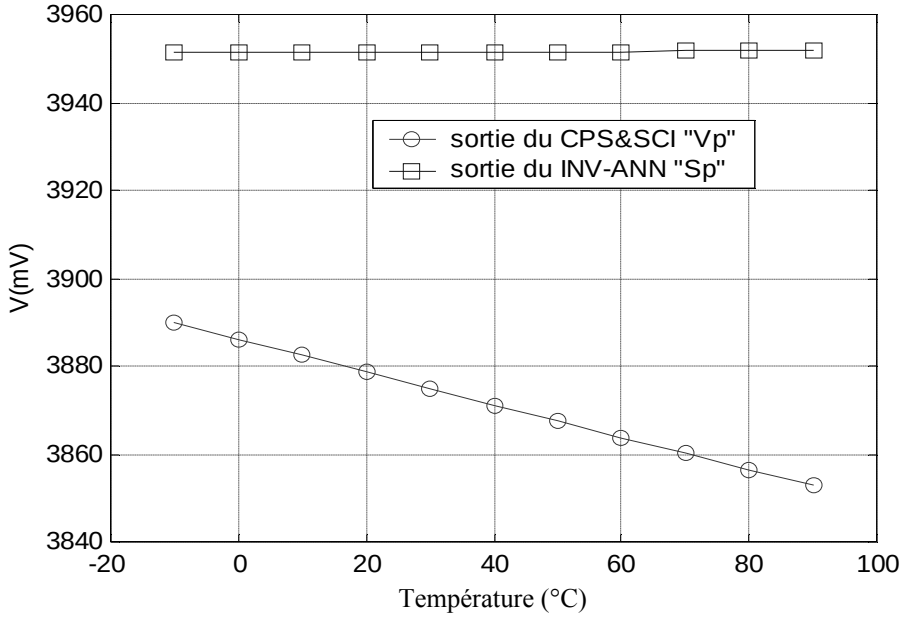


Figure IV.7 : Réponse du CPS&SCI et du Capteur intelligent en fonction de T à P=4.5 bar

II.3 Test de l'hystérésis

Le but de ce test est de comparer les valeurs de l'hystérésis du CPS avec celle du capteur intelligent. Les résultats obtenus sont représentés sur la figure IV.8. On remarque que l'effet de l'hystérésis relatif au capteur intelligent est très faible par rapport au CPS. Il est inférieur à l'erreur d'apprentissage du réseau de neurones (0.1 FS), il peut alors être négligé.

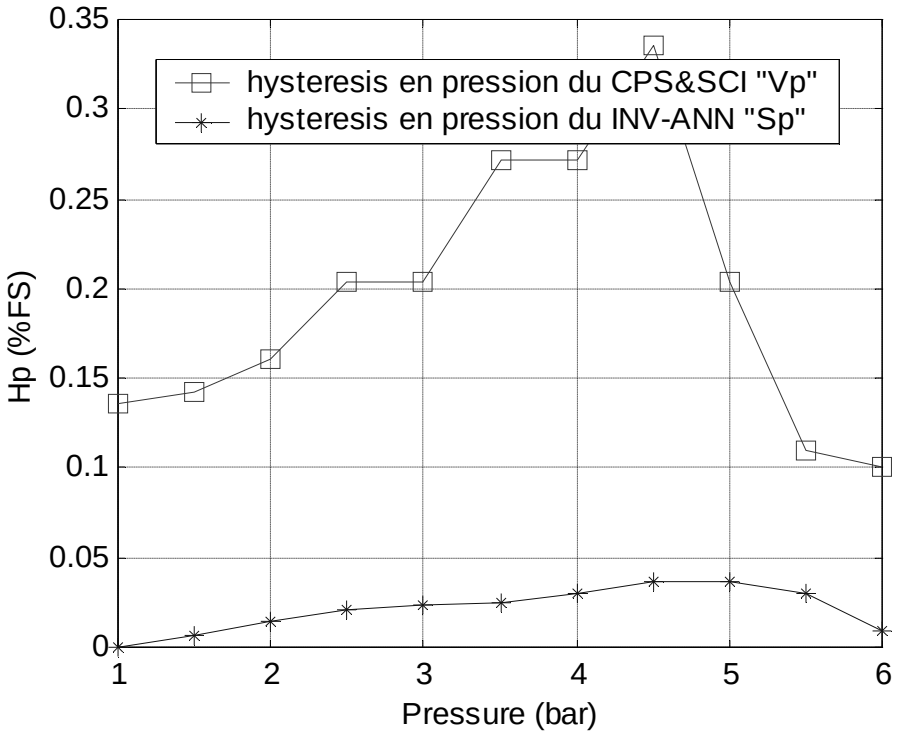


Figure IV.8**:** *Comparaison entre l'hystérésis en pression du CPS&SCI et du capteur intelligent*

II.4 Test pour des valeurs arbitraires

Ce test sert à la validation finale du modèle proposée, il est basé sur le tirage au hasard (par programme) des valeurs de pression et de température à partir d'un ensemble de données (0 à 6 bar, -10 à 90 °C). Le tableau IV.2 englobe l'ensemble des valeurs de ce test et la réponse du système. Après simulation on obtient les résultats représentés en figure IV.9. L'analyse de ces résultas valide le modèle conçu du capteur intelligent.

T (°C)	P (bar)	Sortie du CPS&SCI « V_P » x 100 mV	Sortie du INT_ANN « $\overline{S}_P$ » x 100 mV
-05.2	1.29	35.16	36.31
-05.2	2.08	35.77	37.08
-05.2	2.98	36.70	37.98
-05.2	2.50	36.07	37.46
-05.2	3.88	37.89	38.91
-05.2	4.25	38.47	39.27
15.7	5.02	39.72	40.02
15.7	5.65	40.90	40.67
15.7	5.87	41.33	40.86
15.7	5.02	39.62	40.05
15.7	4.56	38.77	39.57
15.7	4.31	38.36	39.32
45.6	4.12	37.99	39.13
45.6	3.75	37.49	38.76
45.6	3.48	37.13	38.48
45.6	3.22	36.78	38.21
45.6	2.75	36.21	37.71
45.6	2.06	35.61	37.08
65.3	2.15	35.64	37.15
65.3	2.25	35.79	37.31
65.3	1.98	35.51	36.10
65.3	1.50	35.12	36.49
65.3	1.78	35.33	36.79
65.3	1.02	34.77	36.01

Tableau IV.2 : Valeurs du test

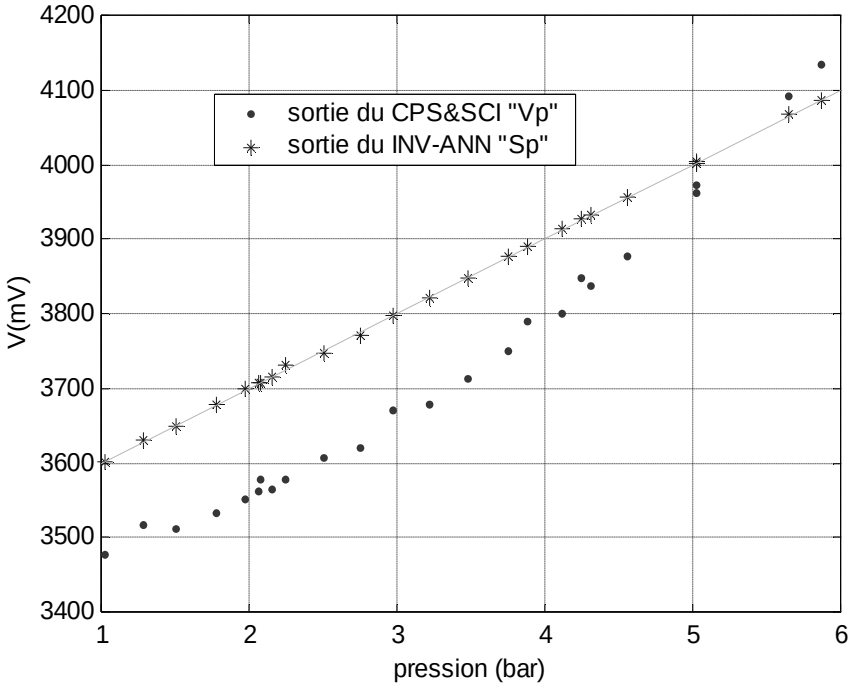


Figure IV.9 : Réponse du CPS&SCI et du Capteur intelligent à des valeurs arbitraires

II.5 Discutions des résultats

Les résultas de simulation obtenus, montrent que : le modèle INV-ANN développé a permis de corriger la sortie du capteur CPS, tout en éliminant les différents problèmes liés à la mesure de la pression par ce type de capteur, tels que la non linéarité, la dépendance à la température en plus de l'instabilité de type hystérésis. L'association d'un tel élément de correction donne une sortie précise et linéaire par rapport à la pression appliquée. Nous exposons au paragraphe suivant, la technique de matérialisation du modèle INV-ANN sur un circuit FPGA (code VHDL), ainsi que le schéma de simulation complet du capteur intelligent.

III. Implantation numérique du réseau de neurones

Après avoir mis au point le modèle du capteur intelligent (INV-ANN), on procède dans ce qui suit à l'implantation du réseau de neurones obtenu.

Il y a plusieurs technique d'implantation qu'on peut regrouper selon deux grandes familles : implantation analogique et implantation numérique. L'implantation analogique offre l'avantage d'une surface de silicium minimale en plus d'un temps de propagation des informations minime par rapport à l'implantation numérique. Cette dernière présente l'avantage d'une grande précision de calcul. Nous nous intéressons en ce qui nous concerne à l'implantation numérique des ANNs.

Dans des travaux concernant le développement d'un capteur de pression intelligent [3, 4], l'implantation numérique du modèle à base des ANNs a été proposé sous forme d'un programme structuré en assembleur exécuté par un microcontrôleur figure IV.10. Chaque neurone est représenté par un sous programme où le calcul se fait neurone par neurone et couche par couche par un processus séquentiel. L'inconvénient de cette méthode réside dans le temps de calcul qui est très important. Dans notre cas nous proposons une mise en parallèle du processus par un circuit FPGA, dans ce cas, chaque neurone est un bloc indépendant des autre neurones et les sorties des neurones sont simultanément calculés ce qui offre un faible temps de calcul par rapport à l'implantation sur des microcontrôleurs.

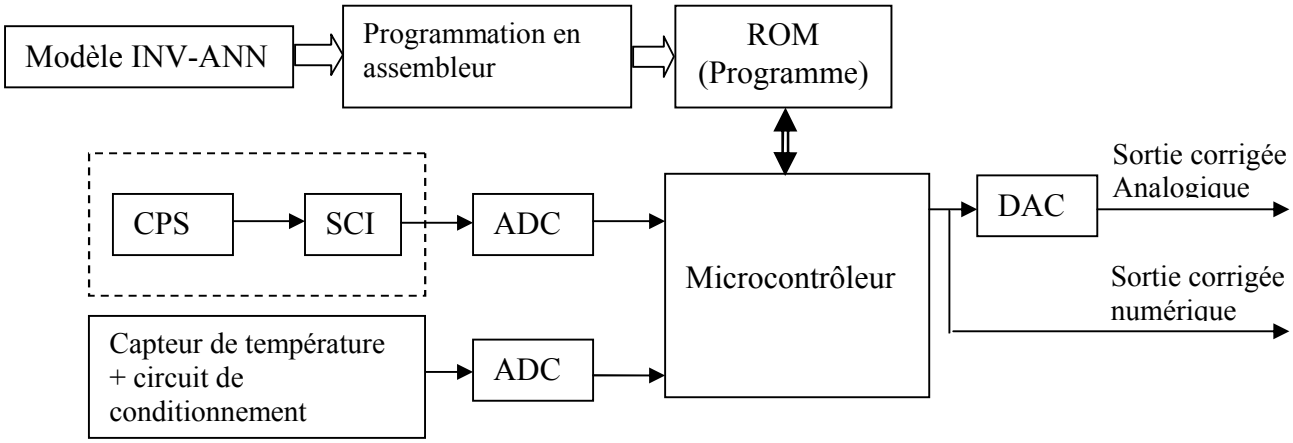


Figure IV.10 : Schéma du capteur intelligent réalisé par un Microcontrôleur

L'implémentation du réseau de neurones a été simulée sous forme matérielle dans un circuit FPGA par le biais du langage VHDL. Pour la mise en œuvre et l'implantation des réseaux de neurones du capteur intelligent nous avons envisagé la solution de la mise en parallèle de processeurs [30]. La figure IV.11 illustre le schéma du capteur intelligent réalisé par un circuit FPGA.

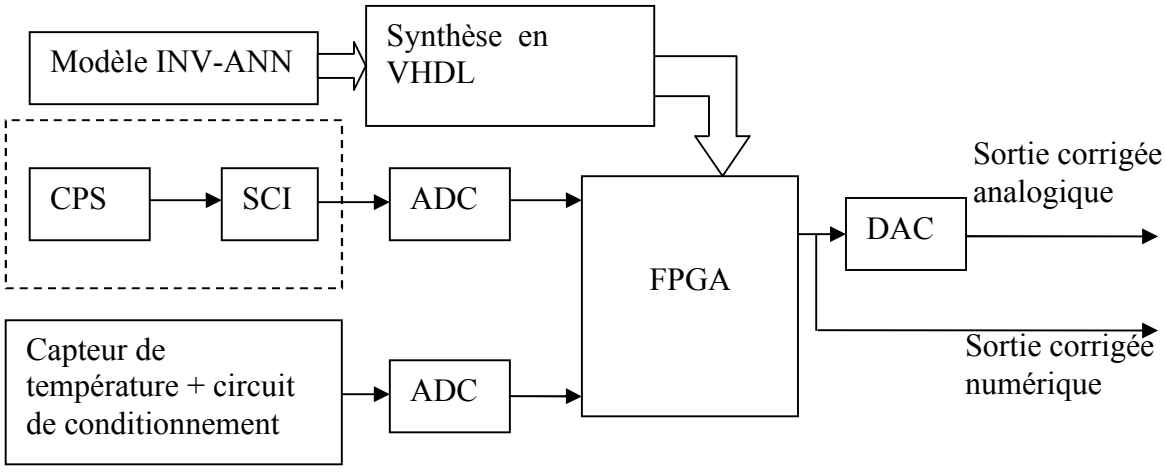


Figure IV.11 : Schéma du capteur intelligent matérialisé par un circuit FPGA

III.1 Présentation de la synthèse logique par le langage VHDL

L'abréviation **VHDL** signifie **V**HSIC **H**ardware **D**escription **L**anguage (**VHSIC** : **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit). Développé dans les années 80 aux États-Unis, le langage de description VHDL est devenu ensuite une norme IEEE numéro 1076 en 1987. Révisée en

1993 pour supprimer quelques ambiguïtés et améliorer la portabilité du langage, cette norme est vite devenue un standard en matière d'outils de description de fonctions logiques.

Le langage VHDL permet la description de tous les aspects d'un système matériel (hardware system), son comportement, sa structure et ses caractéristiques temporelles, par un système matériel. A ce jour, on utilise le langage VHDL pour :

- Concevoir des ASICs (circuit intégré à application spécifique),
- Programmer des composants programmables du type PLD, CPLD et FPGA,
- Concevoir des modèles de simulations numériques ou des bancs de tests.

La réalisation des fonctions logiques simple ou complexe se faisait à travers des fonctions logiques élémentaires contenues dans les circuits intégrés des familles 74xxx ou CD4000. Les expérimentations se limitaient souvent aux fonctions proposées par les fabricants de ces circuits car la mise en oeuvre de fonctions regroupant plusieurs de ces circuits nécessitait le câblage de nombreuses connexions et éventuellement la réalisation de câblage imprimé.

L'apparition des circuits logiques programmables de type PLD (*Programmable Logic Device*), CPLD (*Complexe PLD*) ou FPGA (*Field Programmable Gate Array)*) a permis de s'affranchir de cette limitation. En effet, l'utilisateur peut créer, dans ces circuits, toutes les fonctions logiques qu'il souhaite avec comme limitations, la place disponible dans le circuit choisi et la vitesse de fonctionnement de celui-ci.

Les outils de développement mis à la disposition des utilisateurs par les fabricants des circuits logiques programmables doivent donc permettre de passer de la description du comportement d'une fonction logique à son câblage dans le circuit et cela de la manière la plus simple possible. La figure IV.12 illustre un flux de conception d'un circuit logique programmable.

Notre choix s'est porté sur le VHDL pour la matérialisation du réseau de neurones, car il présente l'avantage d'un langage standard qui peut être implanté dans la plus part des circuits de type FPGA proposés par les fabricants. Nous avons utilisé l'éditeur VHDL intégré dans le simulateur SPICE (où nous avons déjà implanté le modèle du CPS), ce qui nous offre la possibilité de simuler les différents éléments ou modèles dans le même environnement.

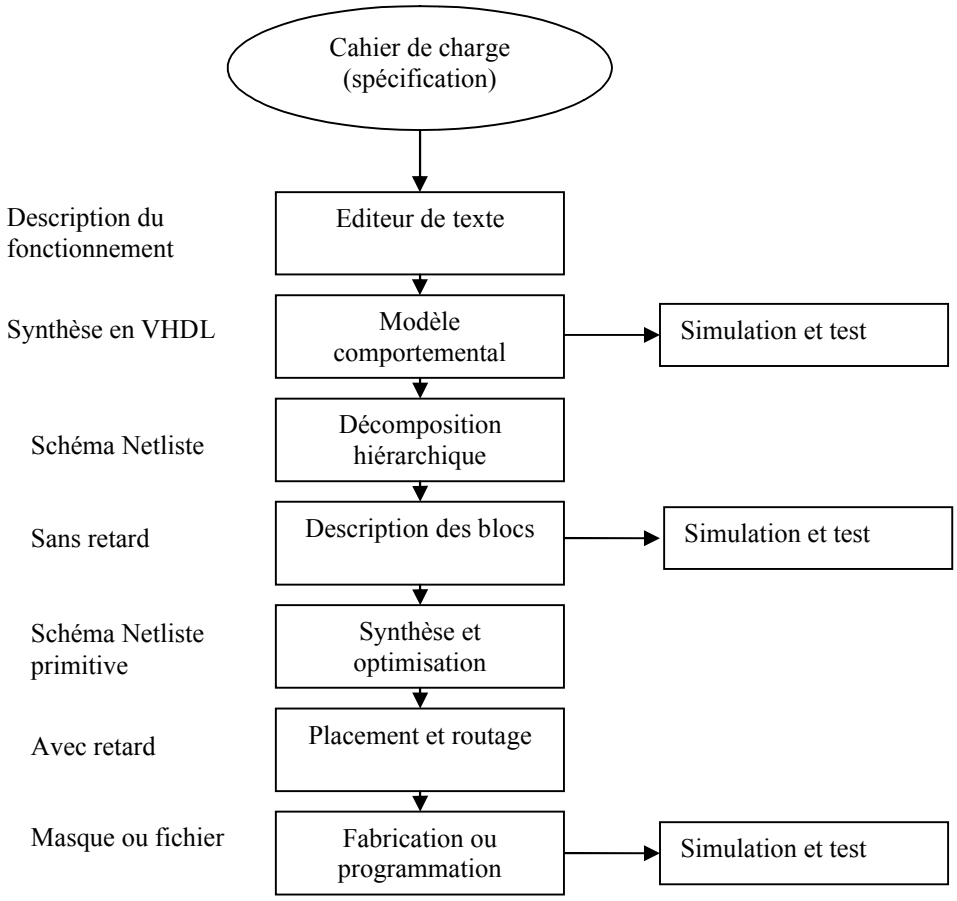


Figure IV.12 : Flux de conception

Le modèle d'un circuit logique en VHDL est composé d'un ensemble d'unités de conception. Chaque unité représente une partie du programme qui peut être compilée séparément. Cet aspect modulaire est la base de la structuration de la description. Le support du programme VHDL est un fichier texte qui peut contenir une ou plusieurs unités. L'unité de conception est composée de deux parties :

Entité (ENTITY) : définie les signaux d'entrées-sorties, leur type ainsi que leur mode (lecture seule, écriture seule, lecture et écriture) ainsi que les procédures éventuellement associées (par exemple: vérifications de relations temporelles).

L'architecture (ARCHITECTURE) : est relative à une entité. Elle contient les fonctionnalités et éventuellement les relations temporelles du modèle dans une description en vue de la synthèse.

III.2 Réalisation d'un neurone assemblable

Afin de réaliser le réseau de neurones développé auparavant, on réalise, en premier lieu, un neurone élémentaire qu'il peut être associé par la suite pour construire un réseau. La fonction

mathématique exprimée par le neurone doit être exprimée par des simples calculs élémentaires (addition et multiplication), ce qui donne la possibilité de synthétiser le neurone par le langage VHDL.

La description structurelle du neurone en VHDL permet de spécifier certaines caractéristiques telles que le nombre d'entrées, la fonction d'activation et la longueur du mot binaire.

Le neurone effectue d'abord le produit des données en entrée 'X', mémorisées dans une zone de mémoire (RAM) avec les poids 'W' correspondants qui sont dans une autre zone de mémoire (PROM) puis il les additionne. Le résultat de ces opérations est soumis à une unité qui approxime une fonction d'activation par segment. La figure IV.13 montre l'architecture d'un neurone élémentaire.

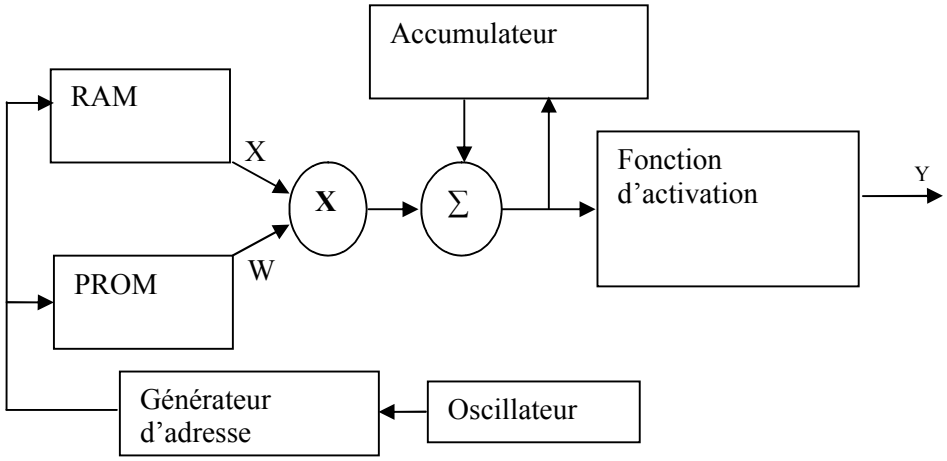


Figure IV.13 : L'architecture d'un neurone élémentaire

Plusieurs possibilités existent pour matérialiser la fonction sigmoïde. Elle pourrait être exprimée par son développement limité tronqué avec une précision arbitraire qui est chère en temps de calcul et en surface de silicium, ou par une table de valeurs arrangé dans une zone de mémoire dans laquelle à chaque valeur de X correspond une valeur de Y. Les valeurs de X étant utilisées pour adresser cette mémoire. Si X est un nombre sur 16 bits et Y est un nombre sur 8 bits, alors il faudrait 64 K Octets de cases mémoires par neurone ce qui requiert une surface de silicium plus importante. L'alternative est d'utiliser une approximation linéaire par intervalle de la fonction de transfert [31]. Dans ce cas, il s'agit d'une table de valeurs combinée avec une approximation linéaire. Dans notre cas, nous avons utilisé une approximation linéaire par intervalle où la fonction sigmoïde est divisée en 18 segments (Figure IV.14). Avec une erreur relative inférieur à 0.15 % (FS).

La fonction sigmoïde est symétrique par rapport au point (X=0,Y=0.5), ce qui implique que la fonction ne requiert que le calcul pour des valeurs positives de X. Pour les valeurs de X négatives, la sortie Y' associée se déduit de l'équation suivante : Y' = 1 - Y où Y est la valeur calculée pour X>0. Le tableau IV.3 résume les segments d'approximation de la fonction sigmoïde.

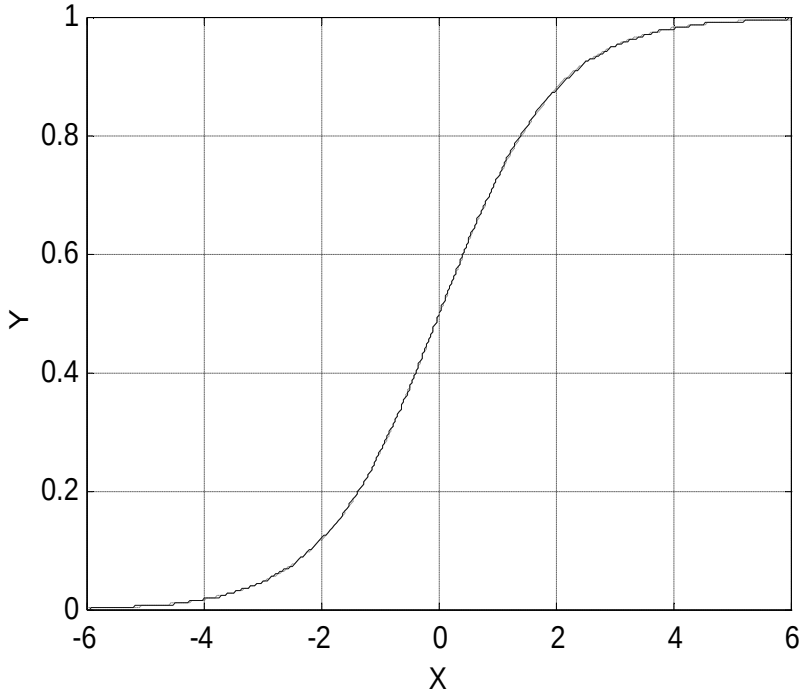


Figure IV.14 : L'approximation de la fonction sigmoïde

L'intervalle de X	L'approximation Y
$\|X\| > 6$	Y=1
$4.50 < \|X\| \leq 6.00$	$Y= 0.0055.\|X\| + 0.0345$
$3.76 < \|X\| \leq 4.50$	$Y= 0.0158.\|X\| + 0.0816$
$3.00 < \|X\| \leq 3.76$	$Y=0.0324.\|X\| + 0.1430$
$2.51 < \|X\| \leq 3.00$	$Y=0.0936.\|X\| +0.3070$
$1.81 < \|X\| \leq 2.51$	$Y=0.0567.\|X\| + 0.2165$
$1.26 < \|X\| \leq 1.81$	$Y=0.1464.\|X\| + 0.4032$
$0.50 < \|X\| \leq 1.26$	$Y=0.2069.\|X\| + 0.4773$
$0 < \|X\| \leq 0.50$	$Y=0.2456.\|X\| + 0.4995$
Si X < 0	Y'=1-Y

Tableau IV.3: Les segments d'approximation de la fonction sigmoïde

On remarque que le neurone élémentaire est traduit par un simple calcul de base (adition et multiplication). Ces deux opérations de base sont matérialisés par un code VHDL (voire l'annexe B), finalement on obtient un neurone sous forme d'un bloc synthétisé en VHDL. Une fois la synthèse du neurone élémentaire est obtenu on peut assemblé ces éléments de base pour obtenir le réseau de neurones en question.

IV. Schéma de simulation global du capteur intelligent

Nous avons regroupé dans ce paragraphe, toutes les données précédemment étudiées dans un seul schéma sur le simulateur SPICE (figure IV.15), il s'agit du modèle ANN du capteur CPS, l'interface SCI et le code en VHDL du INV-ANN. Cet ensemble représente les différents blocs du capteur intelligent implanté dans la bibliothèque SPICE. Les résultats de simulations obtenus confirment les résultats de simulation et de test (effectués avec le MATLAB) présenté au paragraphe IV.2. La figure IV.16 illustre la réponse du capteur intelligent à un cycle de pression à température ambiante.

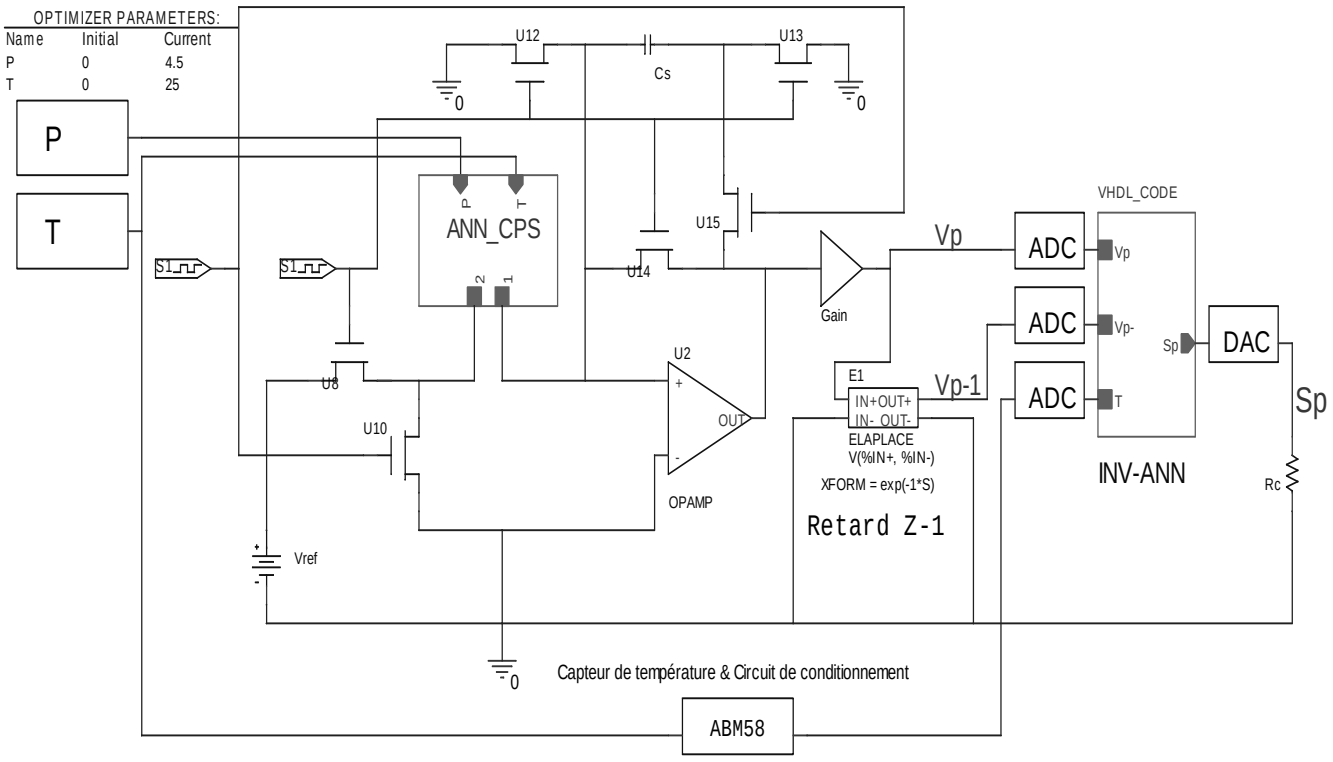


Figure IV.15 : Capteur intelligent

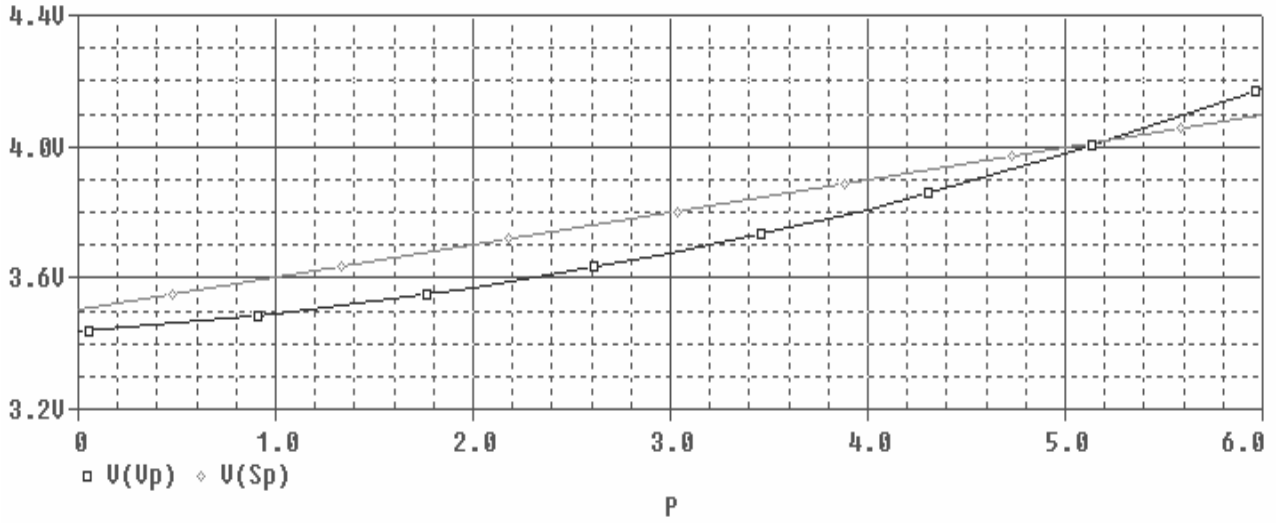


Figure IV.16 : Réponses du capteur intelligent à 25 °C

V. Conclusion

En examinons les résultats de simulation obtenus nous pouvons confirmer que l'association de notre composant de correction modélisé à base des ANNs (INV-ANN) et le matérialiser par un code VHDL, a apporté des améliorations considérables sur le fonctionnement du CPS. En effet, il délivre une sortie linéaire par rapport à la pression appliquée, complètement indépendante de la température en éliminant l'effet d'hystérésis. Le signal de sortie obtenu peut être éventuellement exploité par d'autre processus.

Conclusions et perspectives

Tout au long de ce travail nous avons cherché à développer des modèles à base des réseaux de neurones, que ce soit pour le capteur CPS ou pour son circuit de correction. Le modèle ANN du CPS obtenu exprime fidèlement le comportement du capteur CPS dans un milieu dynamique, du fait qu'il tient compte de l'effet d'hystérésis qui n'est pas pris en compte dans le modèle analytique. L'insertion de ce modèle dans la bibliothèque du simulateur SPICE nous a permis de simuler le fonctionnement du CPS dans des conditions dynamiques.

En ce qui concerne le modèle du circuit de correction (INV_ANN), nous avons mis en évidence son implantation sur un circuit FPGA par le biais d'un programme écrit en code VHDL. Finalement nous avons regroupé sur un seul circuit le CPS, le circuit de conditionnement (SCI) et le circuit de correction implanté sur FPGA. En examinant les résultats obtenus, nous pouvons dire que notre capteur intelligent donne d'excellents résultats dans un milieu dynamique. En effet, il délivre une sortie linéaire par rapport à la pression appliquée, complètement indépendante de la température avec élimination de l'effet d'hystérésis, qui peut être exploiter directement par un processus industrielle.

La modélisation par les réseaux de neurones nous a révélé leur grande capacité d'adaptation aux différents phénomènes dynamiques à condition de posséder une base de données assez représentative sur le phénomène à étudier.

Comme perspectives, de ce travail, une extension du modèle ANN du CPS tenant compte des paramètres géométrique et physique des matériaux, utilisé pour sa fabrication, peut être envisagée. Nous obtenons ainsi des structures précises et fiables destinées à des applications spécifiques. Les méthodes et les techniques développées au cours de notre mémoire peuvent être étendues à d'autre type de capteur afin d'améliorer leur performances et obtenir des réponses précises sur les grandeurs physiques mesurées.

Bibliographie :

[1] **U. Schöneberg**
CMOS Integrated Capacitive Pressure Transducer with On-chip Electronics and Digital Calibration Capability,
The 6th Int. Conference on Solid-State Sensors and Actuators (Transducers'91), pp. 304-307, 1991.

[2] **G. BLASQUEZ, P. PONS, N. FABRE, V. CONEDERA, C. SOLANO, Ph. DONDON and C.ZARDINI**
Faisabilité d'un capteur de pression et de température conçu selon les principes des systèmes intelligents,
Rapport de recherche LAAS N° 91376, SITEF'91, capteur intelligent et micro actionneurs intégrés, Toulouse (France), pp 63-67 ,1991

[3] **Jagdish C. Patra, Adriaan van den Bos**
Auto-calibration and -compensation of a capacitive pressure sensor using multilayer perceptrons,
ISA Transactions, 39, PP 175-190, (2000)

[4] **J.C. Patra**
An artificial neural network-based smart capacitive pressure sensor,
Measurement 22 , PP 113-121, (1997).

[5] **Jagdish C.Patra, Ganapati Panda, Rameswar Baliarsingh**
Artificial neural network-based nonlinearity estimation of pressure sensors,
IEEE transaction on instrumentation and measurement 43 (6), PP 874-881, (1994).

[6] **Jagdish C. Patra, Adriaan van den Bos**
Modelling of an intelligent pressure sensor using functional link artificial neural networks,
ISA Transactions 39, PP 15-27, 2000

[7] **Paul BOURRET, James REGGIA, Manuel SAMUELIDES**
Réseaux neuronaux une approche connexionniste de l'intelligence artificielle,
Ouvrage de l'édition TEKNEA, 1991

[8] **G. DREYFUS, M. MARTINEZ, M. SAMUELIDES, M. B. GORDON, F. BADRAN, S. THIRIA, L. HERAULT.**
Réseaux de neurones méthodologie et application,
Ouvrage de l'édition Eyrolles, 2002

[9] **HORNIK K., STINCHCOMBE M., WHITE H.**
Multilayer feedforward networks are universal approximators,
Neural Networks 2 : PP 359-366. (1989)

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

[11] **D. E. RUMELHART**
Learning representations by back-propagation errors,
The MIT Press, vol. 1, Cambridge. (1986)

[12] **Marc Parizeau**
Réseaux de neurones,
Automne 2004 presse de l'université de LAVAl (2004)

[13] **T. KUDOH, S. SHOJI, M. ESASHI**
An integrated miniature capacitive pressure sensor,
Sensors and Actuators A29 (1991), pp. 185-193.

[14] **Ph. MENINI, Ph. DONDON, G. BLASQUEZ, P. PONS, P. FAVARO, C. DOUZIECH**
Characterisation and modelling analyse of a capacitive pressure sensor based on a silicon/ Pyrex sensing cell and BICMOS A/D integrated circuit,
Sensors and Actuators 85 (2000), pp. 90-98.

[15] **Y.S LEE, K.D. WISE**
A batch- fabricated silicon capacitive pressure transducer with low temperature sensitivity,
IEEE transaction on electronics devices, Vol. ED-29, NO.1

[16] **G. BLASQUEZ, P. PONS, A. BOUKABACHE**
Capabilities and limits of silicon pressure sensors,
Sensors and Actuators, 17 (1989), pp. 387-403.

[17] **Kurt E. PETERSEN**
Silicon as a mechanical material,
Proceedings of the IEEE, Vol. 70, n° 5, 1982, pp. 420-457.

[18] **G. BLASQUEZ, Y. NACIRI, N. BEN MOUSSA, P. PONS**
Static response of miniature capacitive sensor with square or rectangular silicon diaphragm,
Revue de physique appliquée, 22 (1987), pp. 43-46.

[19] **Philippe MENINI**
Faisabilité d'un capteur de pression capacitif miniature sur silicium,
Thèse de l'université Paul Sabatier de Toulouse, LAAS – CNRS, (1998)

[20] **X. CHAUFFLEUR**
Modélisation par la méthode des éléments finis du comportement thermomécanique de capteur de pression capacitifs et piézorésistifs en silicium,
Thèse de l'université Paul Sabatier de Toulouse, (1998)

[21] **Cyril DOUZIECH**
Comportement thermique des capteurs de pression capacitifs au silicium,
Thèse de l'université Paul Sabatier de Toulouse, LAAS - CNRS (1998)

[22] **Leonard Meijer**
Neural Network Applications in device and Subcircuit Modelling for Circuit Simulation,
Thesis of Philips Research Laboratories in Eindhoven, the Netherlands (2003).

[23] **T. QUARLES, A.R. NEWTON, D.O. PEDERSON, and A. SANGIOVANNI-VINCENTELLI.**
SPICE 3 Version 3F5,
User's manual University of California Berkeley CA 94720, 1996.

[24] **A. Hjensmo, A. Hanneborg, J. Gakkestad and H. von der Lippe**
A CMOS front-end circuit for a capacitive pressure sensor,
Sensors and actuators, A21-A23, PP 102 -107, (1990)

[25] **A. Hanneborg, T.E. Hansen, P.A. Ohlckers, E. Carlson, B. Dahl and O.Holwech**
An integrated capacitive pressure sensor with frequency,
Sensors and actuators, A9, PP 354 -361, (1986)

[26] **Xinxin Li, Minhang Bao, Shaoqun Shen**
Study on linearization of silicon capacitive pressure sensor,
Sensors and actuators, A63, PP 1 - 6, (1997)

[27] **A.ETTOUHAMI, A. ESSAID, N. OUAKRIM, L. MICHEL and M. LIMOURI**
Thermal buckling of silicon capacitive pressure sensor,
Sensors and actuators, A57, PP 167 - 171, (1996)

[28] **P.T. Kolen**
Self-calibration /compensation technique for microcontroller-based sensor arrays,
IEEE transaction on instrumentation and measurement. 43 (4), PP 620–623, (1994).

[29] **ARPAIA, P. – DAPONTE, P. – GRIMALDI, D. – MICHAELI, L.**
ANN-Based Error Reduction for Experimentally Modelled Sensors,
IEEE Trans. on Instrumentation and Measurement, vol. 51, no. 1, pp. 23-30, (2002).

[30] **PJC. Clare**
Design and tuning of FPGA implementations of neural networks
SPIE Proc., vol. 3069, pp 129-136, (1997).

[31] **H. Amin**
Piecewise linear approximation applied to non linear function of neural network,
IEE Proc- Circuits Devices Syst., vol 144(6), pp 313-317, (1997).