N° d'ordre:19/2013-M/ELN

الجمهورية الجزائرية الديمقراطية الشعبية

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et Informatique

جامعة هواري بومدين للعلوم والتكنولوجيا

U S T H B

Mémoire

présenté pour l'obtention du diplôme de **MAGISTER**

en : ELECTRONIQUE

Spécialité : *Contrôle des Processus et Robotique*

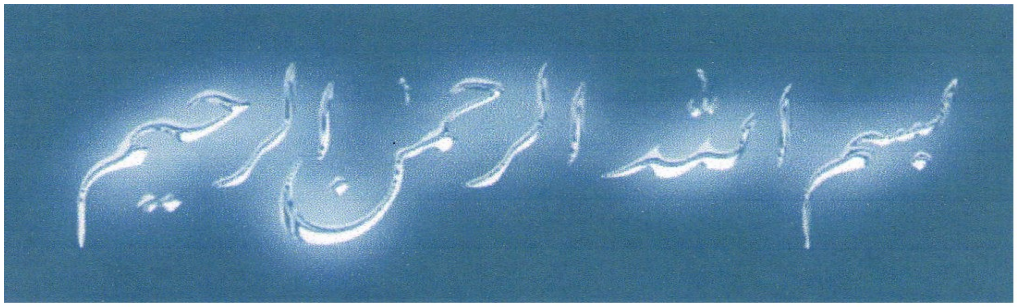
par : Ahmed BOUSSOUFA

THEME

Contribution à l'Elaboration d'Architectures Non-linéaires Embarquées pour le Calibrage des Capteurs

Soutenu publiquement le : 21/03/2013 devant le jury composé de :

Farès BOUDJEMA	*Professeur , à l'ENP*	Président
Mokhtar ATTARI	*Professeur , à l'USTHB*	Directeur de mémoire
Halim SAYOUD	*Professeur , à l'USTHB*	Examinateur
Azzedine NACER	*Maître de Conférences/A, à l'USTHB*	Examinateur


بسم الله الرحمن الرحيم

ملخص:

تعتبر معايرة أجهزة الاستشعار (الحساسات) مهمة ضرورية من أجل الحصول على قياسات دقيقة. ولقد تم اقتراح عدة تقنيات مختلفة في المؤلفات لطرق المعايرة بما في ذلك طرق المعايرة التناظرية والرقمية. إن مفهوم الاستشعار الذكي الذي يدمج المعالج الدقيق يفضل استخدام الطرق الرقمية التي تعتبر أكثر كفاءة ومرونة. في هذا العمل،تم تناول طريقتين رقميتين: الأولى هي أسلوب متعدد الحدود التدريجي الذي يبدو أنه أكثر كفاءة من متعدد حدود لاغرانج من حيث عدد نقاط المعايرة و خطأ المعايرة. الطريقة الثانية هي عن طريق المعايرة باستخدام الشبكة العصبية الاصطناعية، أداة قوية لنمذجة النظم غير الخطية. كانت النتائج مرضية و تم تنفيذها في متحكم PIC16F877A.

Summary :

Sensor calibration is an essential task in order to get accurate measurements. Different techniques have been proposed in the literature for sensor calibration methods including analog and digital methods. The concept of intelligent sensor incorporating a microprocessor favors the use of numerical methods that are more efficient and flexible. In this work, we have addressed two numerical methods: the first is the progressive polynomial calibration method which seems to be more efficient than the Lagrange polynomial in terms of necessity of calibration points and calibration error. The second method is the calibration by artificial neural network, a powerful tool for modeling nonlinear systems. The results are satisfactory and they have been implemented in a microcontroller PIC 16F877A.

Résumé :

La calibration du capteur est une tâche essentielle afin d'avoir des mesures précises. Des différentes techniques ont été proposées dans la littérature pour la calibration des capteurs comprenant les méthodes analogiques et numériques. Le concept de capteur intelligent intégrant un microprocesseur favorise l'utilisation des méthodes numériques qui sont plus efficaces et plus flexibles. Dans ce travail, on a abordé deux méthodes numériques : la première est la méthode polynomiale progressive qui semble d'être plus efficace que la méthode polynomiale de Lagrange en terme de nécessité de points de calibrage et d'erreur de calibration. La deuxième méthode est la calibration par réseau de neurones artificiels, un outil très puissant pour la modélisation des systèmes non linéaires. Les résultats obtenus sont satisfaisant et ils sont implémentés dans un microcontrôlleur PIC 16F877A.

Sommaire

INTRODUCTION:

Les capteurs peuvent être considérés comme les yeux et les oreilles de tout système qui exige des informations sur son environnement. Dans de nombreux cas, les êtres humains prennent part à un tel système. Ils peuvent agir comme un capteur qui fournissent des données à une machine ou même de contrôler la réponse d'une machine. Dans d'autres cas, les capteurs sont tenus de fournir aux êtres humains des informations qu'ils ne peuvent pas observer directement, comme les radiations. Avec la croissance récente des capacités des systèmes automatiques et de traitement de l'information, il ya une demande pour plus d'informations, à des vitesses plus élevées et de sources plus diverses, à des coûts beaucoup plus bas que jamais. C'est cette demande croissante d'informations qui anime actuellement le développement de capteurs à faible coût, haute performance. Ces exigences avec les avancées technologiques de ces dernières décennies, principalement dans le domaine de la micro-électronique, ont contribué de manière décisive à l'émergence du concept de capteur intégré intelligent

Comme les techniques impliquées dans la fabrication de capteurs et de circuits de conditionnement peuvent souvent être pas optimales en raison des caractéristiques intrinsèques de ces composants qui introduisent des imperfections ou des erreurs dans le système d'étalonnage de capteurs,la caractéristiques de transfert du capteur n'est pas idéale et y comprenant plusieurs erreurs indésirables. Ces erreurs doivent être corriger afin d'avoir des mesures précises. Le processus de correction des ces erreurs s'appelle la calibration de la caractéristiques de transfert du capteur. Dans ce travail, on a discuté différentes techniques de calibration ont été discutées comprenant les méthodes analogiques et numériques. Parmi eux, on a abordé deux méthodes pour calibrer un capteur de température

Thermocouple type K : la première est la méthode polynomiale progressive, une solution attrayante pour la linéarisation des capteurs intelligents, car elle nécessite un nombre réduit de points d'étalonnage, un petit nombre d'emplacements de mémoire pour stocker les coefficients d'étalonnage, et une simple évaluation algorithmique des coefficients d'étalonnage, sans itérations mathématiques. En outre, cette méthode permet la mise en œuvre d'une procédure d'étalonnage étape par étape en utilisant un algorithme répétitif qui est particulièrement bien adapté à mettre en œuvre pour une calibrage dynamique et adaptatif des données mesurées. La deuxième méthode consiste à utiliser les réseaux de neurones artificiels. Les réseaux de neurones est un outil très puissant pour modéliser les systèmes non linéaires car il peut être prouvé qu'un modèle de réseau neuronal artificiel peut approximer n'importe quelle fonction non-linéaire dans n'importe quel degré de précision [HAY 94]. Des différentes architectures et topologies et différents algorithmes ont été discutés comprenant les réseaux multicouches MLP, les réseaux BMLP (Bridge multilayer perceptron) et les réseaux FCN (Fully connected feedforward neural network). Ainsi, on a discuté des différents algorithmes comprenant les algorithmes du première ordre et de deuxième ordre.

Chapitre 1

Les Techniques d'étalonnage et de calibrage des capteurs

INTRODUCTION

Les Capteurs intelligents et intégrés sont actuellement dans pratiquement tous les domaines de la vie quotidienne. Génie biomédical, circuits de communication et les applications automobiles sont quelques exemples. Beaucoup de capteurs intelligents sont utilisés dans les systèmes embarqués, qui augmente généralement les besoins de faible superficie, faible dissipation de puissance et de haute performance à un coût minimal et un temps de développement réduit. Un capteur intelligent générique peut être vu comme une unité composée d'un capteur, un circuit de conditionnement du signal et très probablement due à la technologie actuelle, un convertisseur A/D associé à un microprocesseur, de préférence incorporés dans un seul circuit intégré [HOS97] . La tâche principale d'un capteur est de convertir une grandeur physique en un signal électrique qui peut être lu et traité par des circuits électroniques. En raison de la technologie en cause et les caractéristiques non idéaux des composants utilisés dans la fabrication de ces capteurs et les circuits de conditionnement, des effets indésirables tels que l'offset, la non-linéarité et la déviation peuvent généralement être trouvés, et doivent être fixés intérieurement ou extérieurement au système de capteur.

1.1 LES CAPTEURS INTELLIGENTS ET INTÉGRÉS

Les avancements technologiques de ces dernières décennies, principalement dans le domaine de la micro-électronique, ont contribué de manière décisive à l'émergence au concept de capteur intelligent intégré.

Selon J.M. Favennec [FAV87], un capteur intelligent est composé principalement par un transducteur, circuit de conditionnement, une alimentation électrique, une capacité de traitement interne, une interface de communication et un identifiant du capteur. Ce principe est illustré à la figure 1-1.

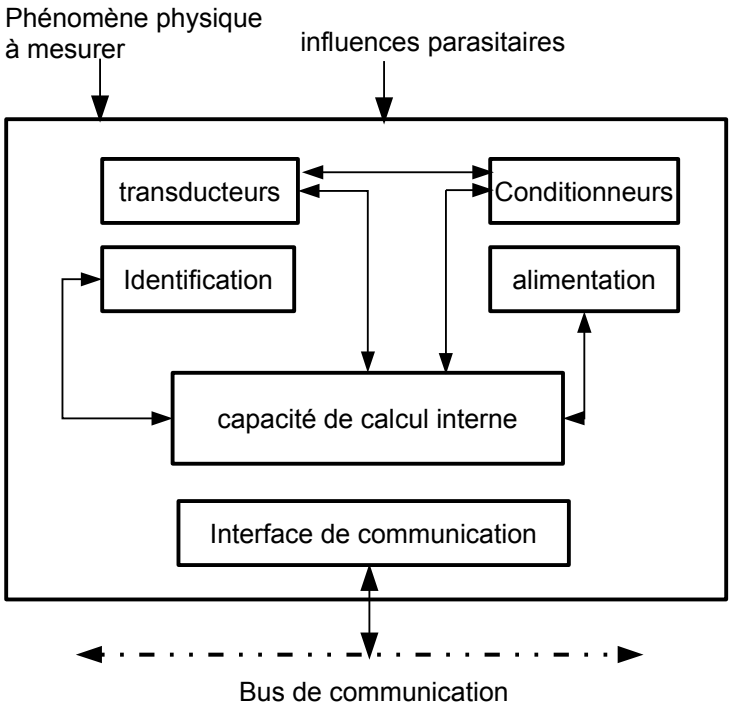


Figure 1-1. Le principe du capteur intelligent, tiré de [FAV87]

Les caractéristiques de base qui ont mis les capteurs intelligents dans une nouvelle catégorie de capteurs peuvent être résumées comme suit:

• La valeur physique mesurée par ordinateur directe (corrigée contre les influences parasites) dans le niveau du capteur (plutôt que au niveau de l'ordinateur central).

• Le signal numérique au lieu d'analogique.

• Identification du capteur .

• Communication avec un réseau ou un ordinateur hôte.

• L'intégration de fonctions supplémentaires au niveau local.

• Possibilité d'auto-diagnostic pour un fonctionnement correct et la préparation des données pour la maintenance.

Dans [HUI94], le concept du capteur intelligent intégré est révisé à nouveau sous le point de vue industriel. Avec la révolution de l'automatisation, suivie par la révolution industrielle et d'information, le besoin de capteurs est devenu très large. Les problèmes associés à ces capteurs ont également augmentés, chaque capteur ayant ses propres particularités. Selon [HUI94], un capteur intégré à puce est défini comme un circuit intégré, sans composants externes, y compris les fonctions suivantes:

• Détection par un ou plusieurs capteurs.

• Interfaçage, de conditionnement du signal, conversion analogique-numérique, et la standardisation du format des bus de sortie.

• Calibrage: offset, l'échelle et la linéarité, la sensibilité croisée.

• Intelligence: auto-test, l'auto-identification.

Ce concept peut être vu dans la figure 1-2.

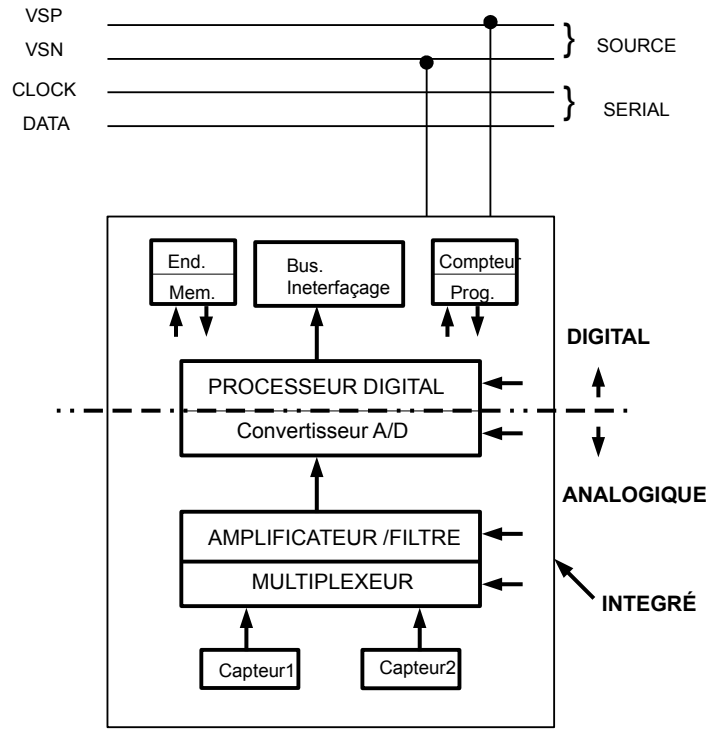


Figure 1-2. Fonctions d'un capteur à puce intégré tiré de [HUI94]

La normalisation des signaux de sortie du capteur signifie de normaliser le point de zéro, le facteur

d'échelle, la non-linéarité et la sensibilité croisée. L'étalonnage des capteurs contribue considérablement au coût du capteur. Souvent, le coût de l'étalonnage dépasse le coût de fabrication primaire. L'importance des micro-systèmes intégrés a continué de croître en raison de la combinaison de deux facteurs: les progrès de la technologie des capteurs de silicium et l'introduction de techniques nouvelles pour la conception des circuits d'interface. D'autres exemples d'application de ce concept sont présentés dans [HOS97, HAB97, BAL96], où l'intégration de conditionnement de signaux et de fonctions de traitement peut améliorer les performances du système de capteurs, de réduire l'effet de non-idéalités et aussi de faciliter la mise en œuvre d'algorithmes de traitement du signal et fonctions de contrôle, de test et de surveillance d'interfaçage.

Beaucoup de ces capteurs sont basés sur des matériaux qui génèrent des petits changements de courant, tension, résistance ou de capacité en réponse à un ou plusieurs stimulus. En outre, le signal de sortie peut être fortement affecté par d'autres paramètres indésirables tels que la température ou la pression. En général, le capteur doit avoir une sortie standard, où les écarts indésirables sont compensés. Le test d'étalonnage et d'auto-test devrait avoir lieu périodiquement. La manière de réaliser ce concept est de combiner dans un seul circuit intégré, un dispositif de détection et d'un certain nombre de composants micro-électroniques. Cet ensemble est souvent désigné comme un capteur intelligent [HUI96].

1.2 CARACTÉRISTIQUES DE LA COURBE DE TRANSFERT DU CAPTEUR :

Comme noté dans la section 1.1, les techniques impliquées dans la fabrication de capteurs et de circuits de conditionnement peuvent souvent être pas optimales en raison des caractéristiques intrinsèques de ces composants qui introduisent des imperfections ou des erreurs dans le système d'étalonnage de capteurs. Pendant le processus de fabrication, ces systèmes de capteurs doivent être testés, individuellement ou par échantillonnage, afin d'assurer une réponse dans une certaine précision prédéfinie [HOR97-DOE90]. Parmi les principales différences dans l'étalonnage de la courbe de capteurs lors de l'essai sont les suivantes:

- **Décalage (Offset):** se produit quand une entrée physique nulle (ou moins) est appliquée mais la valeur de sortie mesurée n'est pas nulle, (Figure 2-3a).

- **Facteur d'échelle (erreur de gain) :** survient lorsque la sensibilité du capteur n'est pas souhaitée, une valeur maximale de la quantité d'entrée physique ne correspond pas à une valeur maximale du signal électrique de sortie (Figure 2-3a).

- **Non linéarité:** lorsque la valeur de sortie du capteur ne varie pas linéairement avec la valeur d'entrée de quantité physique à mesurer (Figure 1-3a).

- **Sensibilité croisée (cross sensitivity):** la courbe d'étalonnage du capteur varie lorsque les mesures sont faites dans différentes conditions environnementales (température, par exemple), de sorte que le capteur est sensible non seulement au signal d'entrée, mais aussi à d'autres paramètres (figure 1-3b).

- **Hystérésis:** survient lorsque la courbe d'étalonnage du capteur est différente pour les différentes valeurs en fonction de valeurs croissantes et décroissantes du signal d'entrée physique (Figure 1-3c).

- **Déviation (Drift):** se produit lorsque la courbe de calibration du capteur change lentement avec le temps (figure 1-3d).

- **Comportement dynamique:** en raison de la coexistence des éléments dissipatifs, et le stockage inertiel, les capteurs peuvent présenter des comportements différents de leurs courbes de réponse en fonction de la fréquence caractéristique du signal d'entrée.

Dans la Figure 1-3, les erreurs principales de l'étalonnage sont expliquées sous forme graphique. Habituellement, ces erreurs sont plus grandes que permise par la précision désirée. Pour corriger ces erreurs, il est nécessaire d'avoir un étalonnage individuel. Les erreurs de gain, d'offset et de la non-linéarité sont observées dans presque tous les capteurs. Erreurs d'hystérésis et de changement de la sensibilité avec le temps sont observées dans seulement quelques capteurs.

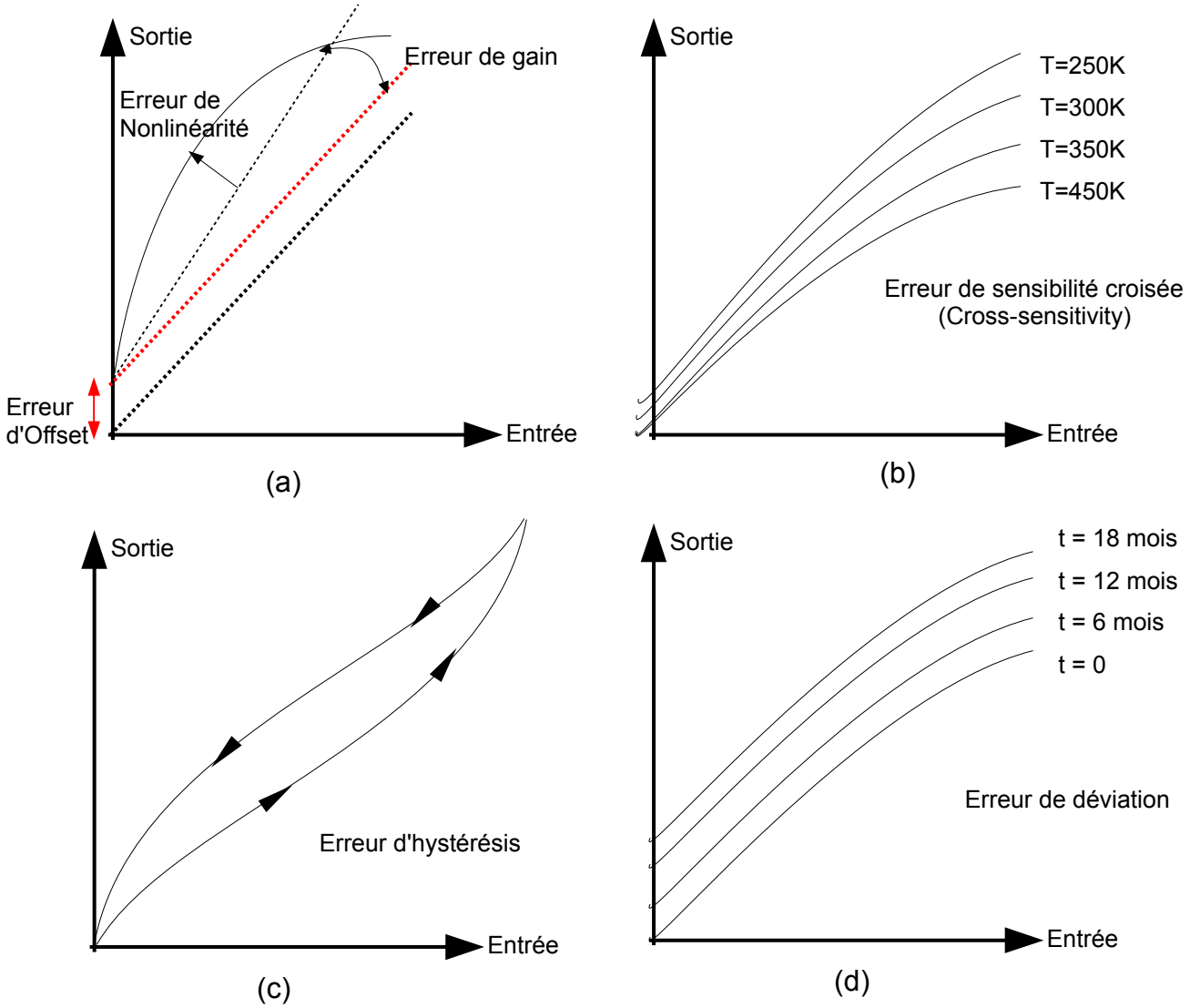


Figure 1-3. Les erreurs possibles dans un capteur .

1.3 MÉTHODES DE CALIBRAGE ET DE LINÉARISATION :

L'étalonnage ou le calibrage des systèmes de capteurs peut être fait en utilisant les deux technologies analogique et numérique. La linéarisation du capteur est une étape principale dans le processus d'étalonnage et de conditionnement de capteur. La linéarisation peut être définie comme la compensation de la caractéristique de transfert du capteur, ou la correction de la non-linéarité entre le signal de sortie du capteur et la quantité mesurée physiques liée [ATT 94].

1.3.1 Méthodes Analogiques:

Parmi les méthodes analogiques utilisées pour linéariser les capteurs, on peut distinguer celles qui agissent directement sur le signal de sortie du capteur, et celles qui agissent sur la source d'alimentation du capteur en cas d'un capteur fabriqué à base d'élément passif.

1.3.1.1. Linéarisation par un circuit analogique non-linéaire

La figure 1-5 est un exemple où la correction de la non-linéarité d'un capteur de température thermocouple est réalisée en utilisant un circuit analogique. Le signal de sortie du capteur a une réponse de type exponentiel. Après que le signal provienne du capteur, il passe à travers un circuit analogique qui a une réponse logarithmique, on obtient donc un signal de sortie corrigé avec une réponse linéaire.

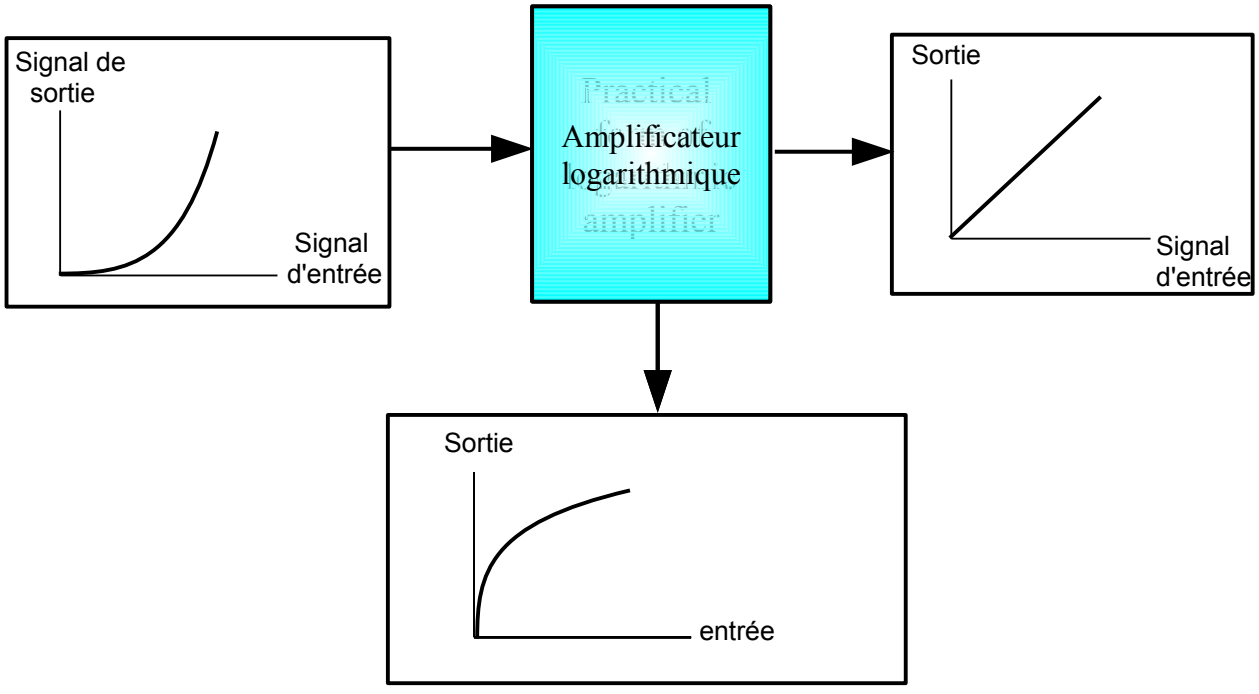


Figure1.5 linéarisation par circuit analogique.

Nikhil et.al ont récemment utilisé ce type d'amplificateur-log pour calibrer des thermocouples de type T, J et G. Le circuit de linéarisation de base amplificateur-log présenté a été encore amélioré en utilisant la propriété ratio-métrique d'ampli-op pour compenser les variations dues à la température [Mon 2009]. Nikhil et.al ont obtenu des bonnes performances concernant les thermocouples de type T et J, alors que pour le thermocouple de type G, le résultat obtenu n'était pas satisfaisant connaissant que ce dernier thermocouple a une caractéristique fortement non linéaire.

1.3.1.2. Linéarisation par réaction sur la source d'alimentation d'un capteur

Un autre exemple d'un circuit de linéarisation agissant sur la source d'alimentation est présenté dans [IVA01] où un circuit intégré analogique de linéarisation du capteur de pression (jauge de contrainte) placé dans un pont de Wheatstone est présenté. La linéarisation est réalisée par l'introduction d'un circuit de Feedback qui module la tension d'excitation du pont proportionnellement au signal de sortie déséquilibré (c-à-d: non linéaire) en ajoutant ou en retranchant une quantité suffisante pour corriger les erreurs de non linéarité positif ou négatif.

$$V_{EXCITE} = V_{EXCITE(0)} \pm V_{BRout} * K_{LIN} \qquad (1.1)$$

En d'autres termes, si la non-linéarité présentée est causée par une diminution (augmentation) de la sensibilité, nous devons ajouter (soustraire) la quantité $V_{BRout}*K_{LIN}$ à la tension d'alimentation du pont pour hausser (baisser) la sensibilité et donc corriger l'erreur de la non-linéarité . Les avantages de cette invention est d'éviter la nécessité d'échanger les connexions des câbles pour établir la polarité de la correction de la tension d'excitation du pont et d'éviter la dépendance de la constante de linéarisation K_{LIN} sur les variations des résistances absolues dans la puce de circuits intégrés.

Lors de l'utilisation des méthodes analogiques, il n'est pas toujours possible de trouver un circuit qui intègre pleinement la fonction inverse du capteur que nous souhaitons corriger [HOR97], de sorte que la compensation, lorsqu'elle est réalisée, n'atteint pas la résolution désirée. En revanche, les méthodes numériques d'étalonnage, en dépit de consommer une plus grande surface lorsqu'elles sont appliquées, soutiennent une grande variation dans le type de fonction de transfert. En outre, les circuits numériques sont constamment développés et peuvent offrir un plus large éventail d'avantages dans le contexte de capteurs intelligents. Des tâches telles que l'auto-calibrage et auto-test sont en général plus facilement mises en œuvre, en réduisant les coûts de production et de tests, intégration et automatisation des

installations, d'autre part, réduisant le coût de conception, les méthodes numériques font une excellente solution. Pour ces raisons, les méthodes numériques sont d'une grande importance et seront abordées prochainement dans ce travail.

En outre, l'étalonnage de capteurs en utilisant la technologie numérique a été systématiquement abordé par la communauté scientifique et de l'industrie, en prenant un nombre important d'ouvrages publiés. D'eux, comme on le verra ci-dessous, nous pouvons avoir une idée générale de ce qui a été fait jusqu'à ce jour, et les diverses techniques employées dans la compensation numérique.

1.3.2 Méthodes numériques:

1.3.2.1. Méthode de table de correspondance (Look up table):

La méthode la plus simple est la méthode de linéarisation par table, où les points de calibrage sont stockés dans une mémoire morte [BRI96]. La sortie du capteur est relié à un convertisseur A/D et le signal numérique est utilisé pour adresser la mémoire contenant les points d'étalonnage. Cette méthode est simple et rapide, mais l'espace de mémoire utilisée augmente exponentiellement avec le nombre de bits requis par l'application, puisque pour chaque bit supplémentaire de la résolution, l'espace requis pour l'adressage double.

La méthode de la table, ici dénommée LUT (Look Up Table) a son diagramme avec le processus de linéarisation à la figure 1-5. Le signal d'entrée à calibrer f(x) du capteur est converti dans le domaine numérique par un convertisseur A/D. Le signal quantifié f [n] adresse la mémoire, qui fournit un signal de sortie calibré g[n]. les avantages de cette méthode sont la simplicité et la rapidité, mais l'inconvénient principal est la zone utilisée par la mémoire.

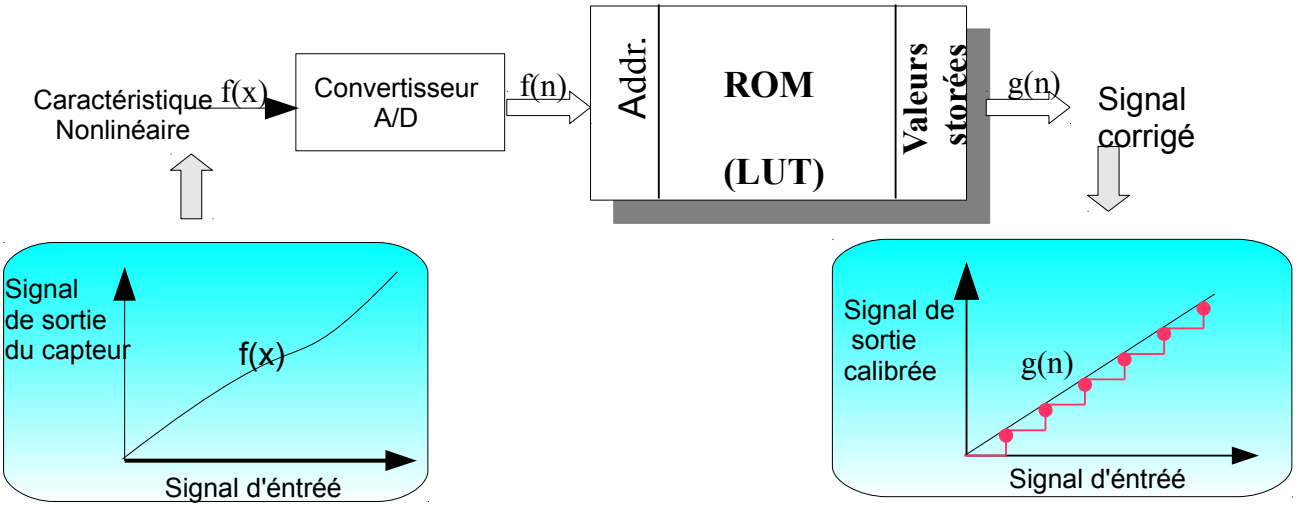


Figure 2-5.Linéarisation par table (Look up table (LUT)).

Le convertisseur de type sigma-delta:

En 1994, Malcovati et.al [MAL94] ont présenté une interface pour les capteurs intelligents en utilisant la conversion A/D sigma-delta et la calibration programmable. Cette méthode d'étalonnage est une variante de la méthode de table (Look Up Table) afin de réduire la taille de mémoire utilisée pour stocker les données d'étalonnage du capteur. La table d'étalonnage a été insérée dans le milieu de l'étage de filtrage du convertisseur, comme le montre la Figure 2-6. Certains cycles de traitement supplémentaires sont nécessaires, en échange de la zone de mémoire gagnée, aussi la mémoire doit fonctionner à la fréquence de sur-échantillonnage du convertisseur. Évidemment, cette méthode est limitée aux convertisseurs de type sigma-delta.

La topologie d'un ADC sigma-delta est essentiellement composée de deux étapes principales: la modulation et le filtrage. Dans la première étape, le modulateur sigma-delta transforme le signal d'entrée en un flux de bits avec la fréquence de sur-échantillonnage ***fs*** [FRA07]. Autrement dit, le flux de bits à la sortie du modulateur est une représentation numérique du signal d'entrée sous la forme de zéros et de uns. La deuxième étape est généralement un filtre de décimation type passe-bas SINC, qui vise à filtrer le flux de bits afin d'éliminer le bruit de quantification introduit par le modulateur, de récupérer le signal d'entrée et de diminuer la fréquence de sur-échantillonnage.

Le gain de cette méthode est d'amener la ROM dans l'étape de filtrage de convertisseur sigma-delta. Pour cela, il est nécessaire d'insérer un nouveau filtre entre le modulateur et le filtre décimateur . Ce nouveau filtre est un filtre SINC opérant sur la fréquence de sur-échantillonnage, qui est bien élevée, est encore raisonnable pour les circuits numériques. Ainsi, le signal d'entrée V_{in} du capteur (Figure 2-6, point a) est transformé par le modulateur sigma delta à un flux de bits (point b). Le flux de bits traverse ensuite le filtre SYNC transversale, qui en raison de mauvais filtrage effectué, est transformé en un flux numérique de mots ***FSN*** (point c) qui peut être quantifié avec un petit nombre de bits. Le nombre de bits du mot numérique qui définit le nombre d'adresses d'entrée de la table (ROM), qui sera utilisé pour corriger la forme d'onde du flux de mots. Après la table est appliquée au signal ***FSN***, on obtient un signal compensé ***NR*** (point d), qui présente également un rapport signal sur bruit très faible en raison de composantes de haute fréquence spectrale conclues par la quantification du processus de modulateur sigma-delta. Enfin, dans la dernière étape, le signal ***NR*** est traité par le filtre de décimation pour avoir en sortie le signal corrigé (point e). Avec la topologie de la figure 2-6, cela pourrait être un compromis entre la résolution et la complexité du matériel.

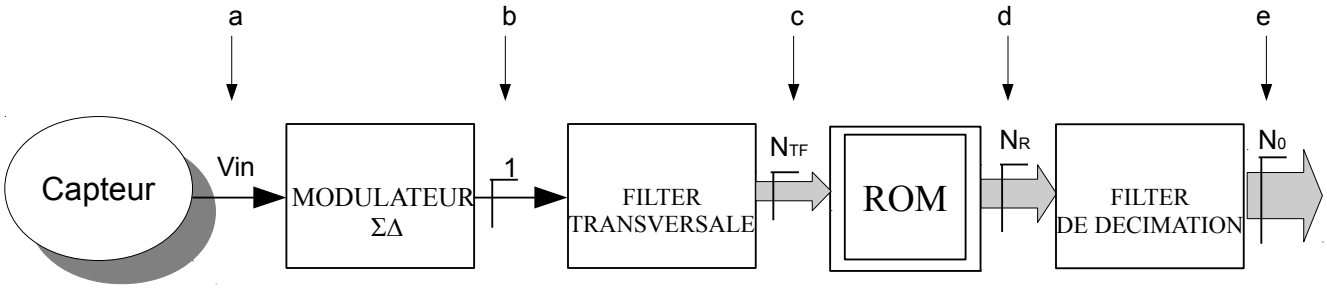


Figure 1-6.Linéarisation par convertisseur sigma-delta

Comme indiqué dans [MAL94], des simulations a été faites dans Matlab 5.3 et un prototype a été mis en œuvre, qui a démontré que, par comparaison avec la méthode directe conventionnelle (Look Up Table), la taille du tableau (ROM) peut être réduite considérablement en maintenant la même résolution. Dans le prototype, un convertisseur ADC a été mis en œuvre en utilisant un 12-bit modulateur sigma-delta avec un filtre du deuxième ordre SINC. La ROM utilisée pour stocker les données d'étalonnage est de type 6 bits abordée par 12 bits de données (768 bits). Le filtre de sortie est un filtre de décimation SINC deuxième ordre. traditionnellement, pour maintenir la même résolution, il faudrait une ROM avec 12 bits d'adresses par 12 bits de données pour un total de 49,152 bits. Par conséquent, la topologie présentée avait réduit de 64 fois la taille de la ROM qui contient les données d'étalonnage. Cependant, la surface utilisée par le filtre décimateur SINC augmente, laissant la solution finale avec une réduction de superficie de 2 à 5 fois.

1.3.2.2. méthode mathématique directe:

Une linéarisation par une méthode mathématique directe est effectuée lorsque une équation mathématique explicite de la caractéristique de transfert du capteur est connue avec précision et peut être résolue. Par exemple, la fonction de transfert d'une RDT de platine est décrit par deux équations polynomiales distinctes: une pour les températures inférieures à 0°C et d'autre pour des températures supérieures à 0°C. Ces équations sont:

$$\begin{aligned} R_{RTD}(T) &= R_0[1 + A*T + B*T^2 + C(1-100^0C)*T^3] \quad (pour \quad T<0^0C) \\ R_{RTD}(T) &= R_0[1 + A*T + B*T^2] \quad (pour \quad T<0^0C) \end{aligned} \tag{1.2}$$

où:

T: température;

R_0, A, B, C sont des constants. [AN-709]

Ainsi, pour la région des températures positives, l'équation est simplement une quadratique. La solution se fait par:

$$T_{RTD}(r) = \frac{-A + \sqrt{A^2 - 4B\left(1 - \frac{r}{R_0}\right)}}{2B} \qquad (1.3)$$

où A, B, et R0 sont définis précédemment et ***r*** est la résistance de la RTD.

Un inconvénient de la méthode mathématique directe pour la linéarisation est qu'elle nécessite la puissance en virgule flottante et les fonctions racines carrées, comme ceux trouvées dans la bibliothèque mathématique du compilateur C51 de Keil (http://www.keil.com). Ces fonctions mathématiques à virgule flottante généralement ajoute plus de 1 Ko à la taille du code.

Une précision similaire ou meilleure peut être obtenue avec une taille plus petite de code globale en utilisant la méthode d'approximation linéaire par morceaux décrit prochainement dans ce chapitre. Cependant, si la bibliothèque de fonctions mathématiques sont nécessaires pour d'autres opérations dans le programme, la technique mathématique directe pourrait être la meilleure solution [AN-709].

1.3.2.3.Méthode d'interpolation linéaire par morceaux:

Comme mentionné précédemment, l'inconvénient de la méthode de Look-up-table est que la zone de mémoire utilisée augmente exponentiellement avec le nombre de bits requis par l'application. Une solution pour minimiser cet inconvénient est basée sur les techniques de linéarisation par morceaux. Dans ce cas, la gamme d'entrée du capteur est divisée en un petit nombre de sous-intervalles où on suppose que le capteur a un comportement linéaire entre ses sous-intervalles. Le nombre de points d'étalonnage et la taille de mémoire nécessaire sont alors considérablement réduits. Pour chaque sous-intervalle du plage d'entrée, des différents coefficients de correction d'offset et de gain sont utilisés pour obtenir le signal du capteur linéarisé [PER09].

Pour comprendre comment cette méthode est appliquée dans la pratique, prenons un exemple d'un capteur de température basé sur un élément de RDT. Supposons d'abord que le tableau des coefficients déjà existe. Chaque coefficient dans le tableau est tout simplement un point sur la fonction de transfert, représentée par une résistance et une température. Ainsi, le tableau prend la forme ***{r0, t0; r1, t1, r2, t2, ... rn , tn}***. Compte tenu de ce tableau, la tâche du microcontrôlleur (MCU) en temps réel est d'abord de

déterminer les deux coefficients qui sont plus proches du point en question, appelant ces coefficients ***(r_m, t_m)*** et ***(r_n, t_n)***, et puis interpoler linéairement entre ces deux points pour obtenir la température. La formule d'interpolation linéaire réelle pour cette sous-intervalle (valable uniquement pour des valeurs de r entre r_m et r_n) prendra alors la forme:

$$T_{SEG}(r) = t_m + (r - r_m) * \left(\frac{t_n - t_m}{r_n - r_m}\right) \qquad (1.4)$$

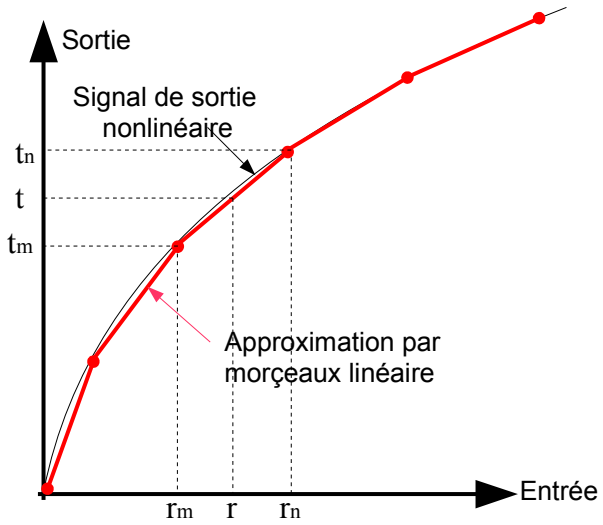


Figure 1-7. Méthode d'approximation par morçeaux linéaires (Piecewise linear approximation).

La technique d'interpolation linéaire par morceaux est souvent utilisée pour linéariser la sortie d'un capteur. Cette technique présente de nombreux avantages dont la vitesse d'exécution est plus rapide en utilisant une simple table de mémoire avec un simple programme qui réduit considérablement les exigences de RAM [AN-942a]. Toutefois, les principaux inconvénients de cette méthode sont liés à sa capacité réduite de linéarisation des caractéristiques fortement non-linéaires et avec les discontinuités importantes qui sont introduites dans le dérivé du signal du capteur linéarisé qui sont particulièrement gênant dans les applications de contrôle par Feedback [PER09].

1.3.2.4 Approximation de courbe et les méthodes d'interpolation polynomiale:

L'ajustement de courbe peut être réalisé soit par lissage (curve fitting) ou par interpolation polynomiale (polynomial interpolation). La différence entre le lissage et l'interpolation polynomiale est que pour le première méthode, les données sont supposées d'avoir certain caractère aléatoire pour

que l'on n'a pas forcément envie d'ajuster une fonction qui passe "exactement" à travers l'ensemble de points de données, mais une courbe ou une fonction est souhaitée que représente meilleure les points de données (figure 1.8(a)), alors que les techniques d'interpolation ont été développées pour l'obtention d'une courbe lisse passant exactement par chaque point de données (figure 1.8(b)) [HAU09].

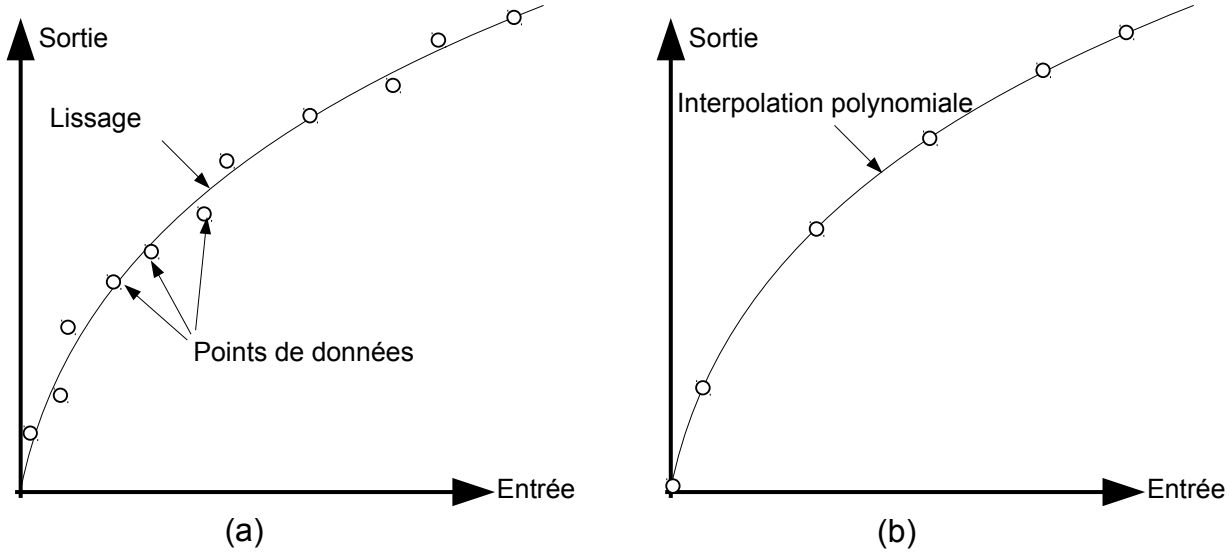


Figure 1-8. (a)Lissage (Curve fitting)
(b)Interpolation Polynomiale

Le lissage est généralement effectué par la minimisation d'erreur des moindres carrés. La fonction la plus générale d'approximation peut être exprimée comme suit:

$$y = C_1 f_1(x) + C_2 f_2(x) + \ldots + C_n f_n(x) \qquad (1.5)$$

où les C_i sont des constantes et les $f_i(x)$ sont des fonctions générales de la variable indépendante x. En pratique, plusieurs formes spécifiques de $f_i(x)$ sont populaires. Parmi ces formes, figure la régression polynomiale en utilisant une série entière en x ou des fonctions sinusoïdales de fréquences croissantes.

Les méthodes d'interpolation sont basées sur les polynômes. Un polynôme de degré n est uniquement déterminé par n +1 points. Le nombre de points utilisé détermine le degré du polynôme. La forme générale d'un polynôme de degré n est:

$$P(x) = C_n * X^n + C_{n-1} * X^{n-1} + ... + C_1 * X^1 + C_0 \qquad (1.6)$$

Quand une sortie du capteur est mesurée pour un petit ensemble de signal d'entrée connu, un algorithme est ensuite utilisé pour calculer une fonction de transfert du capteur $v = f_{est}(x)$ en utilisant

des techniques de lissage ou d'interpolation (figure 1.6). La fonction de transfert inverse est alors calculée de sorte que le signal d'entrée peut être dérivée de la sortie du capteur $(f_{est})^{-1}$. L'autre possibilité est de trouver une fonction d'approximation de courbe pour la fonction de transfert du capteur inverse immédiatement $x = (f^{-1})_{est}(v)$.

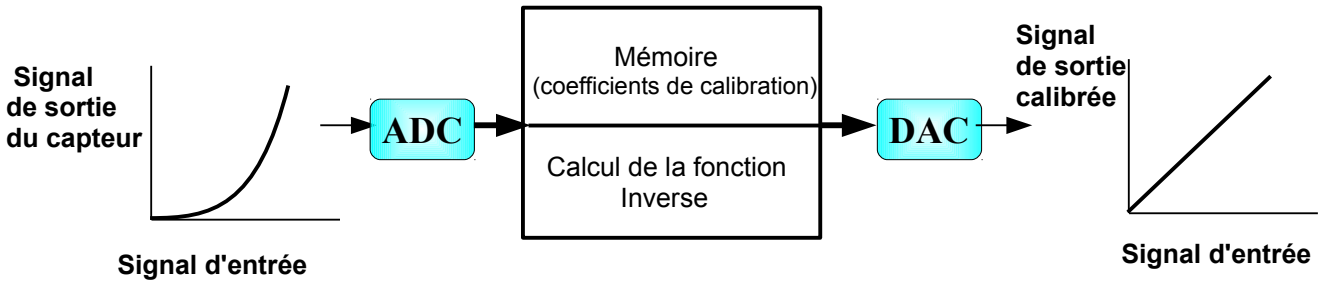


Figure 1.10.Linéarisation du capteur par le calcul de la fonction inverse de la caractéristique du capteur en utilisant la méthode de lissage (curve fitting). Tiré de [AMR05]

Les avantages d'ajustement de courbe sont qu'une fonction de transfert inverse est obtenue pour la gamme complète du signal et les besoins en mémoire sont faibles en raison d'un faible nombre de coefficients. L'inconvénient peut être que des polynômes d'ordre parfois plus élevés sont nécessaires pour obtenir la précision souhaitée. Ceci implique des calculs plus complexes et peuvent également demander des calculs de haute précision (arithmétique en virgule flottante). En général, le degré du polynôme est maintenu bas [Paš 2005].

D. Šaponjic et A. Zigic ont utilisé un polynôme de second ordre pour approcher la fonction de réponse inverse du capteur de pression piézorésistif [SAP01]. La tension de sortie étant non linéaire et dépendante de la température, c'est à dire, $V_S = f(p, T)$, le signal de capteur idéal de sortie a la forme suivante:

$$V_{pc} = g(V_s) = g(f(p,T)) = k * p \qquad (1.7)$$

La fonction ***g*** de compensation (dans le cas idéal, g est la fonction inverse de la fonction ***f*** d'entrée) est approchée sous la forme du polynôme de second ordre suivant:

$$V_{pc} = a_2(T) * V_s + a_1(T) * V_s + a_0(T) \qquad (1.8)$$

L'expression (1.8) permet de corriger la non-linéarité du capteur à une température donnée ***T***. Pour prendre en compte les variations de la tension ***Vs*** avec la température, il est nécessaire d'obtenir le

correspondant coefficients ai (i = 0, 1, 2) du polynôme à un température donnée et à les remplacer dans la formule (1.8) afin de calculer la tension compensée V_{PC}. La dépendance de la température pour chacun des coefficients ai (i = 0, 1, 2) est également approchée par un polynôme de second ordre, c'est-à-dire:

$$a_i(T) = b_{2i} * T^2 + b_{1i} * T + b_{0i} \qquad (1.9)$$

En substituant l'expression (3) dans (2), on obtient la formule de calcul de la valeur compensée ***Vpc*** sous la forme matricielle:

$$V_{pc} = [V_S^2 \quad V_1 \quad 1] \begin{bmatrix} b_{22} & b_{12} & b_{02} \\ b_{21} & b_{11} & b_{01} \\ b_{20} & b_{10} & b_{00} \end{bmatrix} \begin{bmatrix} T^2 \\ T \\ 1 \end{bmatrix} \qquad (1.10)$$

Calibrage polynomiale progressive:

En 1997, G. van der Horn et J.H. Huijsing ont présenté une méthode de linéarisation utilisée pour le calibrage des capteurs intelligents [HOR97]. Cette méthode a été créée afin d'augmenter les performances et la fiabilité des systèmes de mesure. Une fois le calibrage des capteurs avec les méthodes classiques est un processus fastidieux et augmente considérablement les coûts de production car elles nécessitent des ajustements manuels et individuels lors de l'application d'un grand nombre de stimulus d'entrée, cette solution intégrée est proposée afin de minimiser ces problèmes. Une méthode d'étalonnage et des options pour intégrer le concept de capteur intelligent à la fois matérielles et logicielles sont proposées. Un avantage de la méthode proposée dans [HOR97] est qu'elle ne nécessite pas une large gamme de données d'étalonnage pour être stockés dans une table de mémoire. Cette méthode est implémentée grâce à un algorithme non itératif . Dans une première étape, des coefficients d'étalonnage sont trouvés, similaire à un filtre IIR, et la prochaine étape est de mettre le régime en opération. Le signal de référence est un capteur avec une courbe d'étalonnage idéal, et l'étalonnage se fait progressivement.

Dans la figure 1-7, nous pouvons voir graphiquement comment le processus de calibration polynomiale progressive se produit. La première mesure $f_0(x_1)$ est utilisée pour corriger le décalage, ce qui signifie que la fonction de transfert est décalée , comme indiqué dans la Fig.1(a).

$$f_1(x) = f_0(x) + a_1 * y_{ref} \qquad (1.11)$$

La seconde mesure $f_1(x_2)$ est utilisée pour corriger l'erreur de gain, sans affecter l'étalonnage de

décalage. Ce résultat est obtenu par la rotation de la fonction autour du point d'étalonnage précédent, comme indiqué dans la Fig.1(b).

$$f_2(x) = f_1(x) + a_2 * (f_1(x) - x_1) \qquad (1.12)$$

Tout le reste des mesures sont utilisées pour corriger l'erreur de non-linéarité, qui est réalisée par "plier" la fonction de telle manière que les points d'étalonnage précédentes restent fixes, comme le montre les Fig.1(c, d).

$$f_3(x) = f_2(x) + a_3*(f_2(x) - x_2)*(f_1(x) - x_1) \qquad (1.13)$$

$$f_4(x) = f_3(x) + a_4*(f_3(x) - x_3) *(f_2(x) - x_2)*(f_1(x) - x_1) \qquad (1.14)$$

Avec la correction d'erreur de non-linéarité, de gain et de décalage, cette méthode peut aussi être mise en œuvre en deux dimensions, corrigeant ainsi les erreurs dues à la sensibilité croisée comme la température [KHA97, HOR 97].

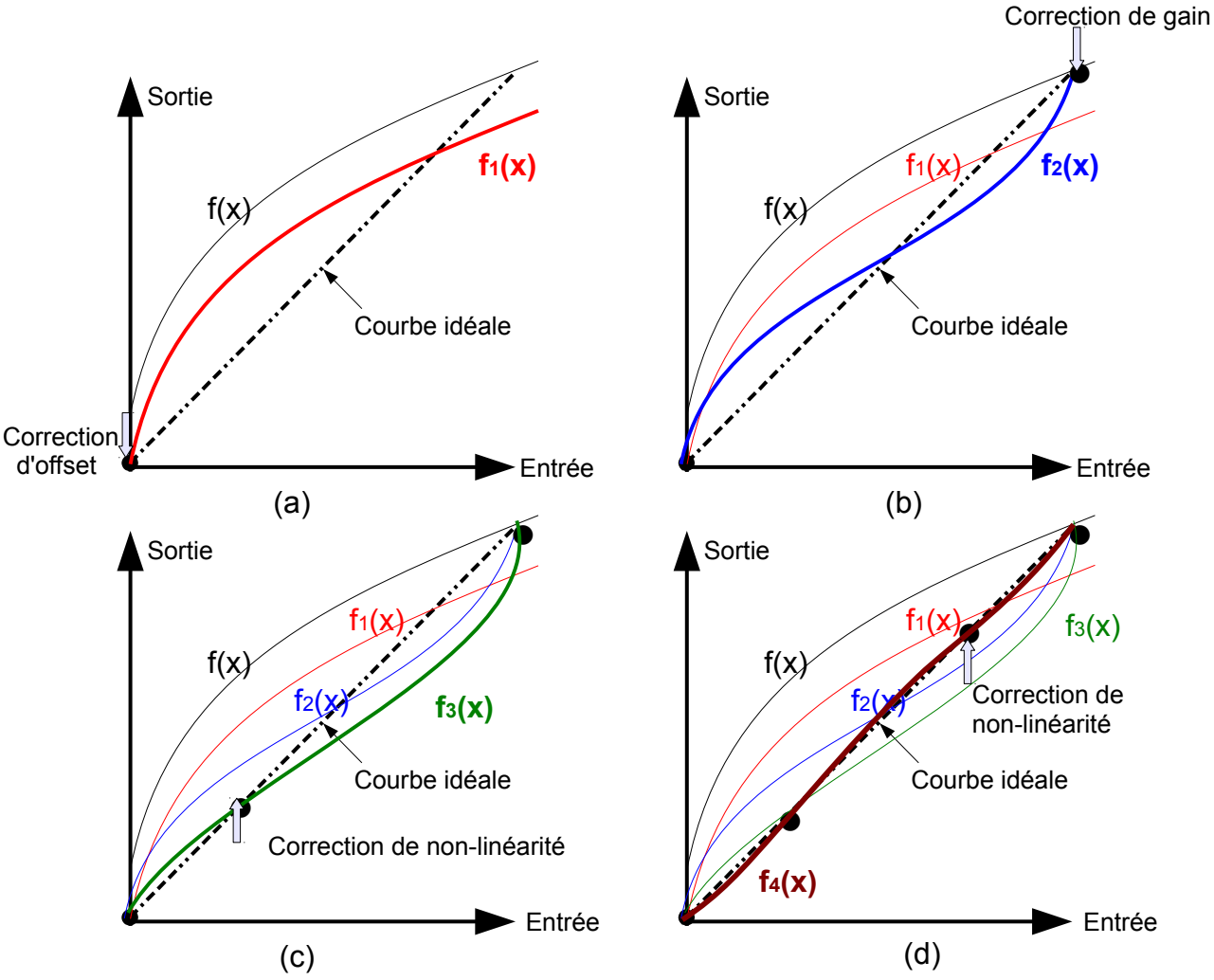


Figure 1-10. Calibrage par méthode polynomiale progressive.

Description	Mesure	Fonction de transfert	égalisation
1ère calibration	$f_0(x_1)$	$f_1(x) = f_0(x) + a_1 * y_{ref}$	$f_1(x_1) = x_1$
2ème calibration	$f_1(x_2)$	$f_2(x) = f_1(x) + a_2 * (f_1(x) - x_1)$	$f_2(x_2) = x_2$
3ème calibration	$f_2(x_3)$	$f_3(x) = f_2(x) + a_3*(f_2(x) - x_2)*(f_1(x) - x_1)$	$f_3(x_3) = x_3$
4ème calibration	$f_3(x_4)$	$f_4(x) = f_3(x) + a_4*(f_3(x) - x_3) *(f_2(x) - x_2)*(f_1(x) - x_1)$	$f_4(x_4) = x_4$

Table1.1.Équations de calibration pour la méthode de calibration polynomiale progressive

Les avantages de cette méthode est qu'elle donne une bonne linéarisation pour un nombre minimum de points, a des exigences peu de mémoire due à un petit nombre de coefficients et utilise un algorithme répétitif (étape par étape de calibration). Les inconvénients sont que quelques connaissances sont nécessaires lors du choix des mesures d'étalonnage [AMR05].

Des améliorations à la méthode de calibration polynomiale progressive ont été rapportées dans la littérature telle que présentée dans [PER09], où les points d'étalonnage sont choisis en fonction de la densité de probabilité des points de mesure. Une autre est présentée dans [JOS08] où des points d'étalonnage fixes sont réajustées de façons différentes afin de minimiser l'erreur de non-linéarité.

1.3.2.5. Linéarisation avec convertisseur analogique-numérique (ADC) non linéaire

Cette méthode consiste à employer un convertisseur A/D, ayant une caractéristique non-linéaire qui est l'inverse de la caractéristique du capteur tel que la non-linéarité du capteur est retirée. De cette façon, la conversion A/D et la linéarisation sont exercées conjointement en utilisant une seule unité physique. Beaucoup d'architectures intéressantes ADC non-linéaires ont été proposées telles que celles basées sur la caractéristique I_C-V_{BE} du transistor, la caractéristique V-T de capacité et d'un amplificateur du gain programmable, ayant une commune approche l'adaptation dynamique de la gamme d'entrée du convertisseur en fonction de la caractéristique non-linéaire du capteur.

Le conversion A/D non-linéaire est une alternative intéressante car elle offre de grosses économies en termes de superficie (surface de silicium). Cela est dû au fait qu'un bon convertisseur ADC non-linéaire conçu (ie, correspondant à l'inverse de la caractéristique du capteur) exige un nombre minimum de bits pour atteindre une certaine résolution [ANT03].

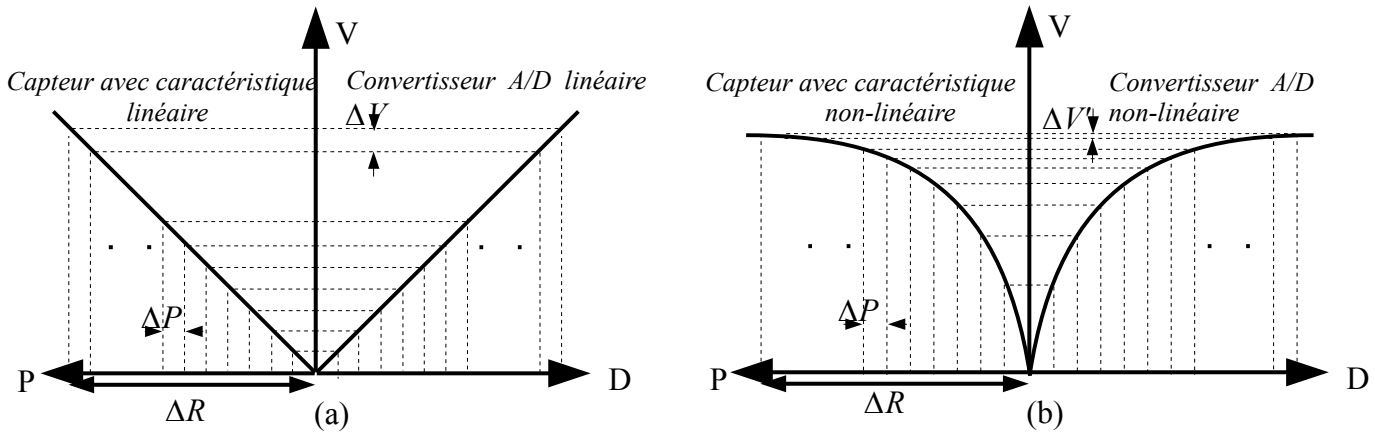


Fig. 1.8 (a) capteur lineaire et convertisseur A/D linéaire.
(b) capteur nonlinéaire avec convertisseur A/D linéaire.Tiré de [ANT 2003]

Parmi les architectures possibles pour les ADC non-linéaires , on trouve le convertisseur dont la caractéristique est une approximation linéaire par morceaux d'une fonction non linéaire (par exemple l'inverse de la caractéristique du capteur). J. Antonio et al. [ANT03] ont présenté une nouvelle 8-bit CMOS convertisseur A/D avec une caractéristique linéaire par morceaux. L'architecture qui en résulte peut être appliquée à la linéarisation de caractéristiques non-linéaires d'une grande variété de capteurs. Une mise en œuvre très compacte est obtenue, puisque la conversion A/D et la linéarisation sont effectuées simultanément à la fois par le même circuit et parce que les deux étages de conversion partagent la plupart du matériel requis.

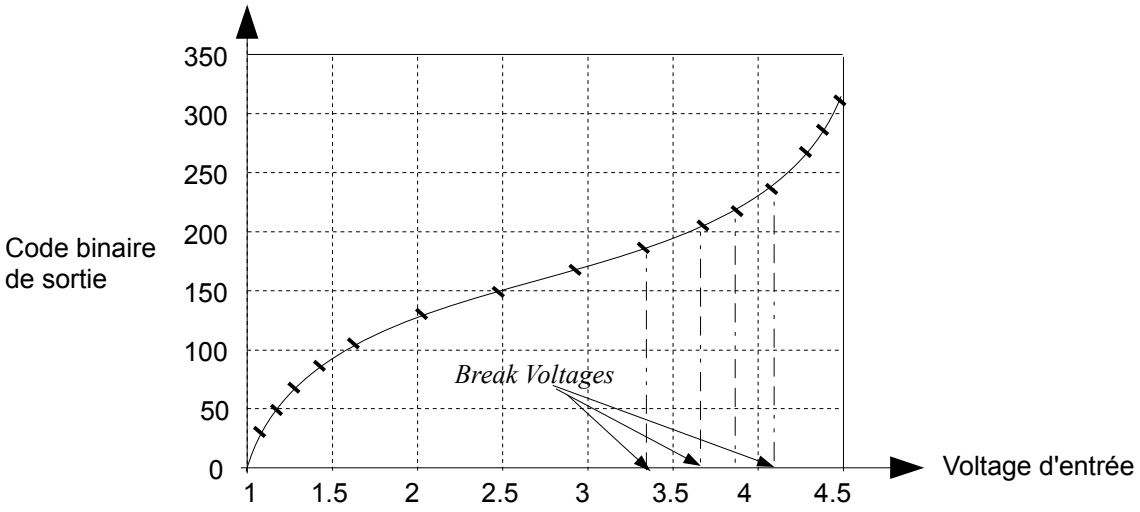


Fig. 1-12. Characteristique linéaire par morceaux d'un A/D.

1.3.2.6.Calibrage avec le filtrage adaptatif

En 1998, Jahn et Car [JAH98] a présenté une nouvelle topologie pour la compensation numérique de non-linéarité dans les circuits analogiques basée sur un filtre adaptative. Outre la linéarisation des caractéristiques du capteur, la méthode compense également les étapes d'amplification et de filtrage anti-aliasing. Un des principaux avantages de cette méthode est d'augmenter les performances des circuits analogiques CMOS construits avec la technologie numérique.

La méthode de [JAH98] utilise des filtres FIR (Finite Impulse Response) avec des coefficients linéaires et non-linéaires associés à l'algorithme LMS (Least Mean Square) que doivent être trouvés pour mieux corriger la fonction d'entrée. Dans la figure 2-8, nous avons le système de linéarisation. Dans l'étape d'entraînement (zone verte), le capteur est excité à l'aide du bruit blanc. Après convergence vers l'erreur minimal prévu, il a été en fonctionnement continu. Cette mise en œuvre a utilisé 10 coefficients pour le filtre linéaire et 4 coefficients pour les filtres non-linéaires (carré et au cube).

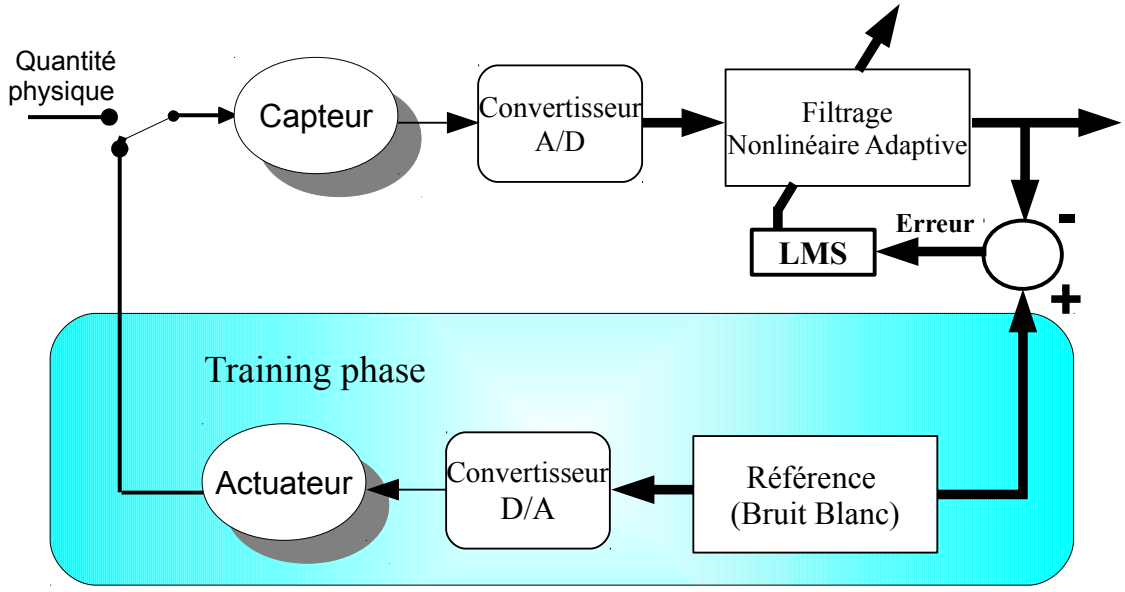


Figure 1-13. Calibrage avec filtrage adaptative.

Un autre exemple de linéarisation de capteurs par filtrage adaptatif est présenté dans [JEN97], où un système de détection est démontrée qui utilise deux ADC ΣΔ pour l'étalonnage de la non-linéarité du capteur. Le système intègre à la fois un grand et un petit capteur, et exploite les fonctionnalités de chacun. Le grand capteur ***CL*** fournit le rapport ***SNR*** (Signal to Noise Ratio) désiré à un débit de données, tandis que le petit capteur ***CS*** est utilisé pour effectuer l'étalonnage de linéarité continu à un taux inférieur de données. Cet étalonnage compense à la fois la non-linéarité du capteur ainsi que les

grands changements lents environnementaux tels que la température. L'algorithme de calibrage est une technique récursive des moindres carrés basée sur un modèle polynôme de Tchebychev pour la caractéristique du capteur.

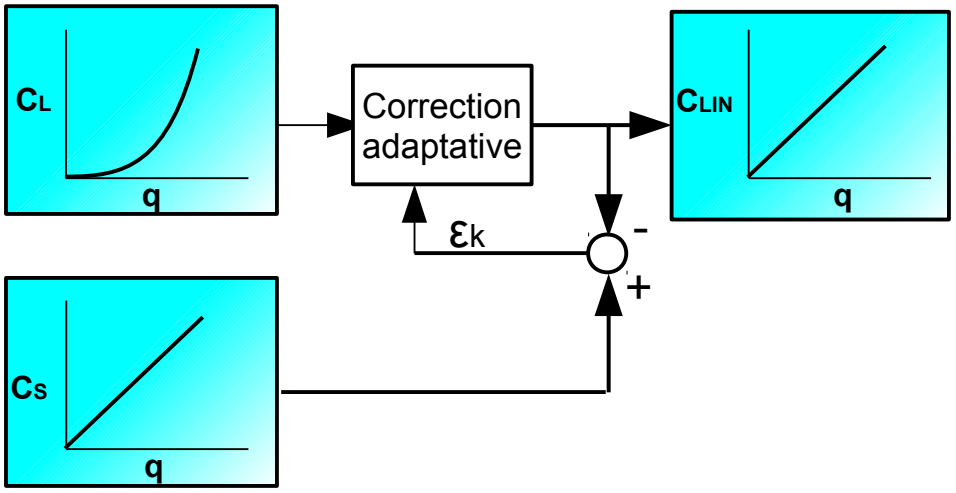


Fig. 1-14. Exemple d'un calibrage adaptive d'un capteur C_L avec un signal de référence C_S.

1.3.2.7.Réseaux de neurones artificiels:

Dans les dernières années, les réseaux neuronaux artificiels (ANN) ont acquis une large attention dans les applications d'ingénierie. Une de ses principales applications est la modélisation des systèmes non-linéaires, car il peut être prouvé qu'un modèle de réseau neuronal artificiel peut approximer n'importe quelle fonction non-linéaire dans n'importe quel degré de précision [HAY94]. Une grande variété de topologies de réseaux de neurones peuvent être imaginées, toutefois, la vaste majorité des applications de réseaux de neurones sont des réseaux multicouches MLP.

La figure 1.11 montre un réseau de neurones multicouches feedforward à **n** entrées,une seule couche cachée avec **Nc** neurones cachés et $\mathbf{N_0}$ neurones de sortie. Il est à noter que les réseaux feedforward sont statiques; si les entrées sont constants, alors même que les sorties.

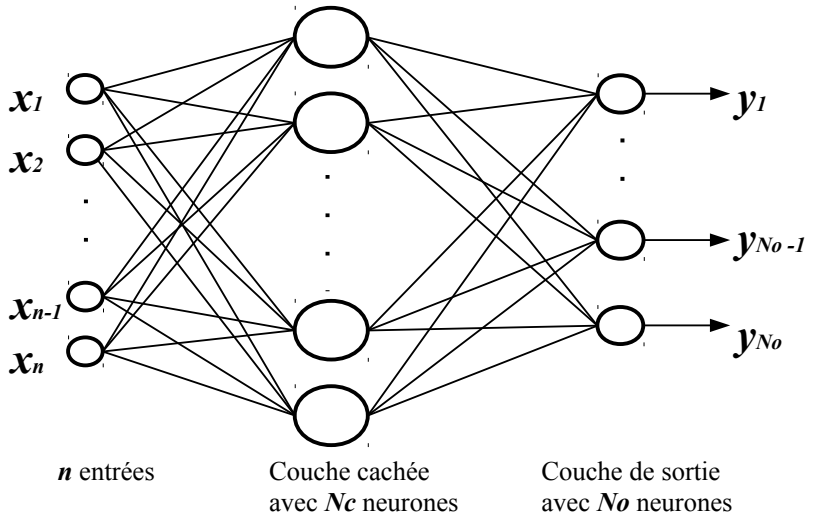


Fig. 1-15. Réseau de neurones multicouches avec *n* entrées, couche cachée avec *Nc* neurones, et *No* neurones de sortie

La linéarisation du capteur basée sur le réseau neuronal artificiel est réalisée par l'entraînement du réseau de neurones à mettre en œuvre la caractéristique inverse du capteur . Différentes méthodes pour l'apprentissage de réseaux de neurones ont été développées allant de recherche aléatoire à les méthodes basées sur le gradient . La méthode la plus connue est la méthode de rétro-propagation d'erreur (Error Back Propagation).

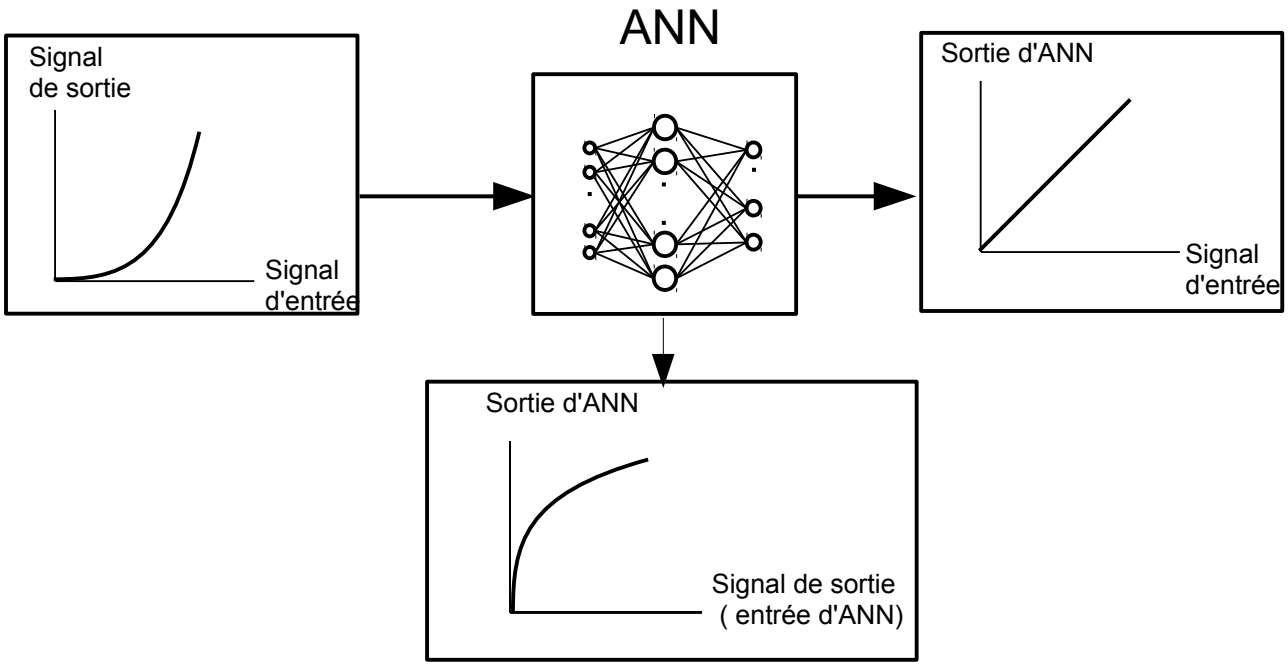


Fig.1-16.Linéarisation du capteur par réseaux de neurones (ANN).

Plusieurs articles ont été publiés pour la linéarisation des capteurs en utilisant des réseaux neuronaux artificiels. Par exemple, M. Attari et.al [ATT95] ont présenté un réseau de neurones artificiels pour linéariser la caractéristique du thermocouple G (rhénium de tungstène vs 26% de tungstène), dans la plage de température de 0 à 2000 ^{0}C. Le réseau proposé est un MLP à trois couches avec une structure 1-12-12-1 qui est d'abord entraîné par l'algorithme de rétro-propagation (EBP) pour apprendre l'ensemble d'apprentissage jusqu'à ce qu'il atteigne une erreur stagnante. L' ensemble de poids est ensuite utilisé pour initialiser un algorithme d'optimisation aléatoire (méthode des perturbations) qui atteint une nouvelle erreur finale plus petite que celle obtenue par EBP. Une comparaison entre les techniques d'interpolation et la méthode de neurones a été présentée montrant l'efficacité du réseau de neurones sur les autres méthodes.

JC Patra et al [PAT00] ont proposé un schéma d'un capteur de pression capacitif intelligent (CPS) en utilisant un réseau de neurones artificiels (ANN) pour modéliser la caractéristique du capteur directe et inverse. Le réseau proposé est un MLP à deux couches avec la structure 3-5-1 entraîné par l'algorithme EBP. La modélisation directe peut être utilisée pour la détection de défaillance du capteur et le contrôle de la qualité au cours de son processus de fabrication. Le modèle inverse proposé a le but de l'estimation de la pression appliquée.

Les ANNs sont également appropriés et avec des performances supérieures en compensation pour les dépendances de l'environnement comme la température et autres paramètres indésirables. Des résultats ont été rapportés par M. Attari et.al dans [ATT96] où un 2D linéariseur artificiel basé sur les réseaux de neurones est utilisé pour linéariser la mesure des ions fluor [F-] en utilisant un électrode sélectif d'ions (électrode du fluor) et en prenant en compte le problème de minimisation de l'effet de la perturbation (les ions OH-). Un autre résultat est rapporté par JC.Patra et.al dans [PAT08], où ils ont montré que le modèle basé sur un ANN pour un capteur CPS peut permettre une lecture de pression avec un maximum d'erreur à pleine échelle de seulement ± 1,5% sur une plage de température de -50 à 200 ^{0}C.

Chapitre 2

Les Réseaux de neurones pour les systèmes embarqués

INTRODUCTION:

Les réseaux de neurones artificiels (ANN) sont devenus un domaine de recherche attractif au cours de ces dernières décennies et ont touché de nombreuses branches de l'industrie. Aujourd'hui, ils sont appliqués à un grand nombre d'applications en monde réel: allant de la prédiction à la modélisation fonctionnelle du système (où les processus physiques ne sont pas bien compris ou sont très complexes), aux systèmes de reconnaissance de formes et de classificateurs robustes, avec la possibilité de généraliser tout en prend des décisions sur des données d'entrée imprécises. La capacité de réseaux de neurones d'apprendre et d' approximer des relations entre l'entrée et la sortie d'un système est découplée de la taille et la complexité du problème [HAY99]. En fait, comme les relations basées sur les entrées et sorties sont enrichisses, la capacité d'approximation s'améliore ,les réseaux de neurones artificiels offrent des solutions idéales pour ce type de problèmes. Il y a beaucoup de différents types de réseaux de neurones, certains des plus populaires comprennent les réseaux multi-couches (MLP) (qui sont généralement entraînés par la rétro-propagation), quantification vectorielle d'apprentissage (SVM), les fonctions de base radiale (RBF), Hopfield et Kohonen, pour n'en nommer que quelques-uns [MEI03]. Dans ce qui suit, on va discuter quelques aspects de réseaux de neurones et leurs implémentations au niveau des systèmes embarqués.

2.1.Les composants critiques d'un réseau de neurones :

Les réseaux de neurones sont constitués de plusieurs éléments essentiels. Le principal composant est le neurone lui-même qui est le composant cercle dans la figure 2.1. Un réseau de neurones est constitué d'un ou plusieurs neurones connectés dans n'importe quelle configuration. Les lignes reliant représentent des poids. La sélection de ces des poids détermine la façon dont laquelle le réseau de neurones répond à des modèles d'entrée particuliers. L'apprentissage de réseaux de neurones est la méthode de sélection des ces poids pour donner le résultat souhaité avec un ensemble donné d'entrée. Les réseaux de neurones gagnent leur propriété de non-linéarité de leur fonction d'activation qui est représentée par le signal passant par le neurone. Les réseaux de neurones peuvent prendre de nombreuses formes et tailles et être disposés dans un nombre infini de manières.

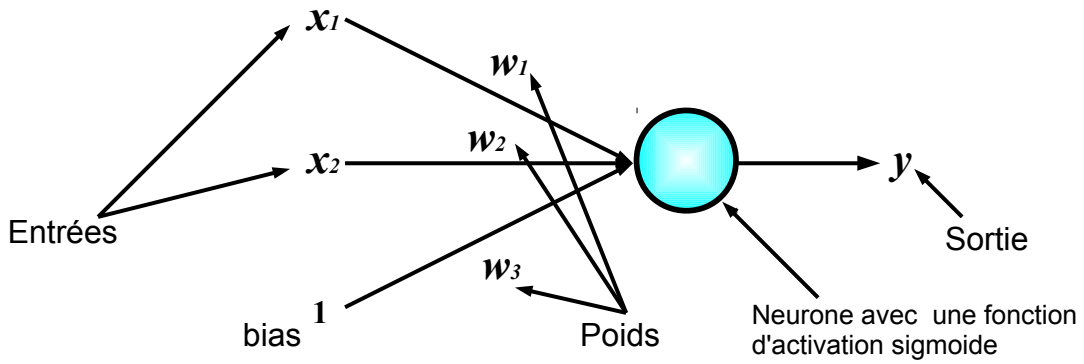


Figure 2.1: Un réseau de neurones constitué d'un seul neurone, deux entrées et une sortie .

2.2.Les réseaux multicouches (MLP) par rapport aux réseaux arbitrairement connectés

Une grande partie de la recherche de réseaux de neurones a impliqué l'utilisation de réseaux multicouches (MLP). Peu de recherches, cependant, implique des réseaux de neurones connectés arbitrairement au niveau du système embarqué. Cependant, d'après B.Wilamowski [WIL09], les réseaux de neurones entièrement connectés en cascade (FCN) sont supérieurs aux réseaux MLP traditionnels de plusieurs façons. Ces réseaux sont plus rapides, plus fiables pour l'apprentissage, plus efficace car moins neurones sont nécessaires pour résoudre des problèmes similaires, et ils peuvent résoudre des problèmes plus difficiles qui sont a peu prés impossible pour les réseaux MLP a résoudre [WIL 09 ,NIC 10].Un exemple qui révèle la supériorité de réseaux de neurones entièrement connectés en cascade devant les réseaux MLP est le problème de parité-N. Le problème de parité-N est considéré comme l'ensemble le plus difficile des modèles pour l'apprentissage de réseaux de neurones. Le

problème de parité-2 est également connu comme le problème de OU exclusif (XOR). Plus grand est le N, le plus il est difficile de le résoudre. Les nombres minimums de neurones nécessaires pour des problèmes de parité-N sont donnés par les équations (2.1) – (2.3)[WIL2009]. Dans ces équations, n_n est le nombre minimum de neurones, et n_w est le nombre de poids.

Pour les architectures traditionnelle MLP:

$$\begin{aligned} nn &= N + 1 \\ nw &= nn^2 = (N+1)^2 \end{aligned} \tag{2.1}$$

Pour les architectures BMLP:

$$\begin{aligned} nn &= [\frac{N}{2}] + 1 \\ nw &= nn(N+2) - 1 \end{aligned} \tag{2.2}$$

Pour les architectures FCC :

$$\begin{aligned} nn &= \log_2(N+1) \\ nw &= nn(N + 0.5 + \frac{nn}{2}) \end{aligned} \tag{2.3}$$

Le tableau 1 montre le nombre minimal de **neurones/poids** nécessaire pour des problèmes de parité différents en utilisant diverses architectures de réseaux de neurones. On peut facilement conclure que l'architecture FCC peut résoudre les problèmes de parité-N avec le plus petit nombres de neurones et de poids.

ARCHITECTURE	PARITY-3	PARITY-7	PARITY-15	PARITY-31	PARITY-63
MLP	4/16	8/64	16/256	32/1024	64/4096
BMLP	3/14	5/44	9/152	17/560	33/2144
FCN	2/9	3/27	4/70	5/170	6/399

Table2-1–nombre de neurones/poids nécessaire pour différent problèmes de parité pour différents architectures (tiré de [WIL09]).

Il y a un autre avantage pour les architectures avec des connexions entre les couches non adjacentes. Grâce à ces connexions supplémentaires, les réseaux de neurones sont plus transparents pour la propagation du signal, et ils sont plus faciles pour l'apprentissage. Dans les architectures typiques MLP, les signaux propagés en avant et en arrière doivent passer plus d'éléments de non-linéarité (les neurones) que dans les autres topologies [WIL09].

Les MLP, les réseaux entièrement connectés en cascade(FCN) et les BMLP peuvent être vus dans les Figure 2.2 et Figure 2.3, respectivement.

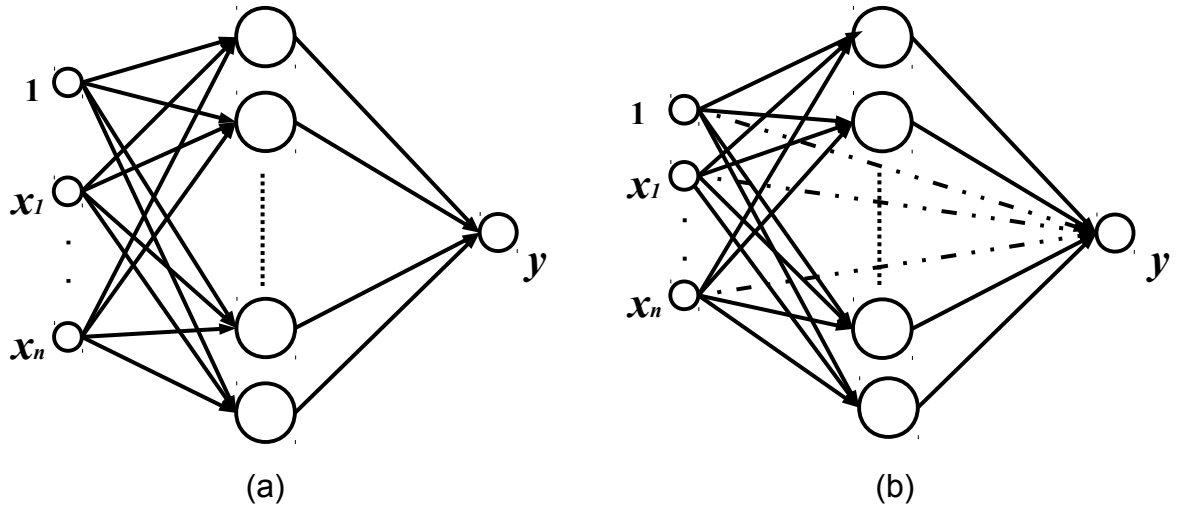


Figure2.2 (a) Réseau de neurones multi-couches Feedforward (MLP) .
(b) Réseau de neurones multi-couches Feedforward avec connections entrée-sortie directe (BMLP).

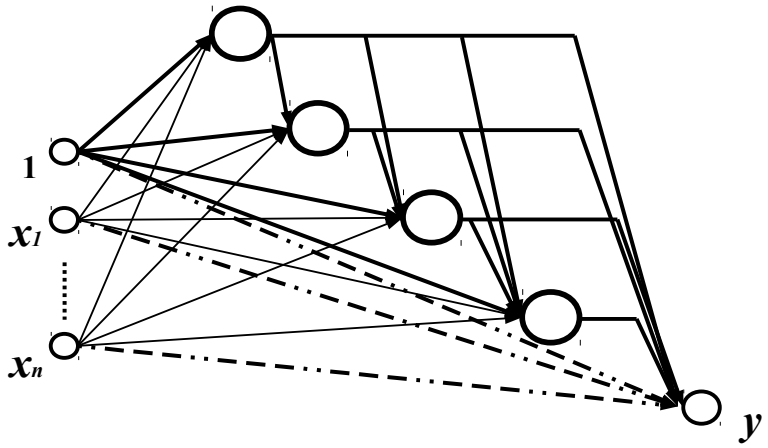


Figure 2.3 Un réseau de neurones entièrement connectés en cascade (FCN)

2.3. La fonction d'activation :

La plus commune des fonctions d'activation de réseaux de neurones est la tangente hyperbolique (tanh) qui est définie par l'équation (2.4) et représentée dans la figure 2.4. Beaucoup de recherches sur les réseaux de neurones embarqués utilisent une approximation de tanh, et la plupart des logiciels d'apprentissage forment les réseaux de neurones sur la base des fonction d'activation tanh.

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \qquad (2.4)$$

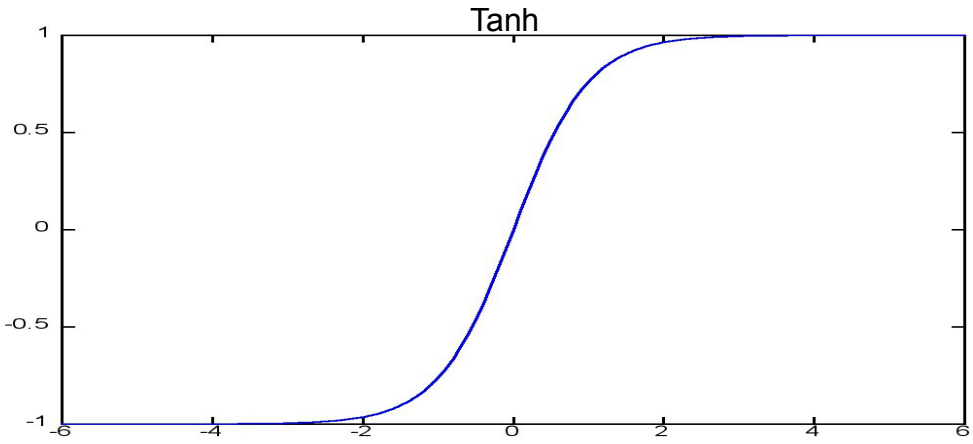


Figure 2.4: La fonction Tangent Hyperbolique.

Une méthode classique pour avoir une fonction d'activation dans un système embarqué est de la stocker dans une table de mémoire (look up table). La méthode de table peut avoir une variété de différentes couches de précision, mais cette précision est exponentiellement proportionnelle au nombre de points de données qui doivent être stockées. Cette approche a été utilisée dans [DEP97] où il a stockés 256 valeurs, qui donne une résolution non suffisante. Le problème avec cette méthode est qu'une vaste quantité de mémoire doit être dédiée au stockage des valeurs.

Une des plus simples approximations de tanh est l'approximation linéaire avec des points de saturation comme le montre la figure 2.5(a). Petchjatuporn et.al [PET05] ont utilisé cette approximation dans un petit réseau de neurones pour contrôler le point de puissance sur un chargeur solaire. Cependant, cette approche limite fortement les capacités de réseau de neurones qui obtient son propriété de non-linéarité de sa fonction d'activation, mais dans ce cas, cette non-linéarité a été supprimée.

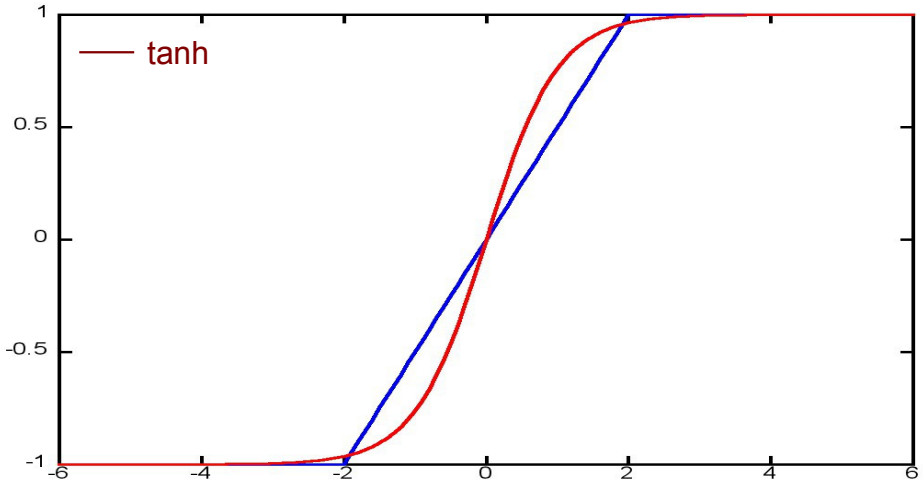


Figure 2.5: La function linéaire avec des points de saturation.

Une meilleure approximation de la fonction tanh est la fonction d'activation linéaire par morceaux comme le montre la figure 2.6. Cette approche a été utilisée par S.O Saleh et.al [SAL11] dans la mise en œuvre d'un contrôleur basé sur un réseau de neurones pour un robot mobile de navigation dans un microcontrôleur à faible coût. Le but de l'utilisation d'une approximation linéaire par morceaux est de réduire la complexité des calculs pour les microcontrôleurs à faible coût. Toutes ces approximations linéaires permettent des calculs rapides et faciles , mais au détriment de la diminution de précision .

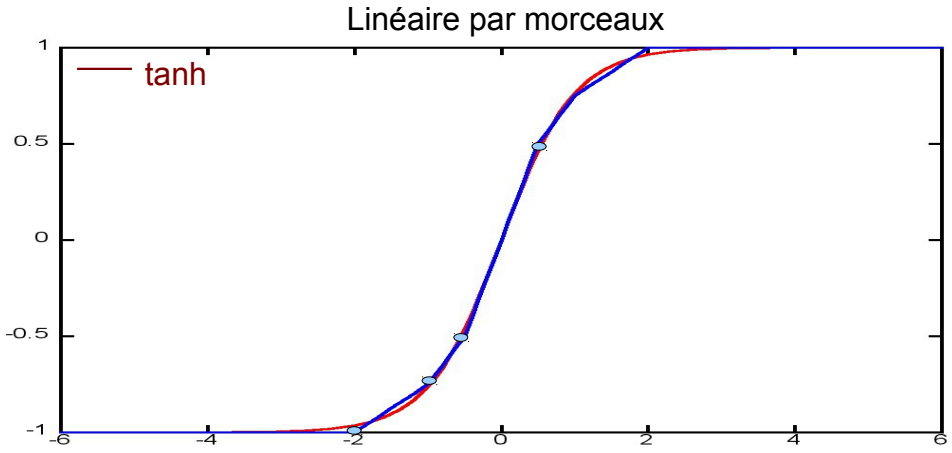


Figure 2.6. La fonction linéaire par morceaux

Si nous voulons une plus grande précision à la fonction tanh en utilisant la méthode d'interpolation par morceaux, nous pouvons soit:

1 - accroître le nombre d'intervalles utilisés pour l'interpololation.

2 - ou augmenter le degré du polynôme utilisé pour interpoler dans chaque intervalle.

En augmentant le nombre d'intervalles cela va augmenter les points de données que doivent être stockées dans la mémoire. De même, si on augmente le degré du polynôme, la rapidité de calcul va diminuer.

À titre d'exemple, une interpolation linéaire par morceaux à la fonction tanh avec 16 segments pour la gamme d'entrée entre 0 et 4 cédera à une erreur maximale inférieure à 1%. En utilisant une interpolation par morceaux quadratiques avec les mêmes conditions que précédemment, l'erreur maximale a été réduite à moins de 0,03%. Si 32 segments sont utilisés, alors l'erreur maximale est réduite à 0,003% pour l'interpolation par morceaux quadratiques, tandis que si l'interpolation linéaire par morceaux est utilisée, l'erreur maximale est d'environ 0,2% [COT11].

Une fonction d'activation très commun est la fonction sigmoïde. La fonction sigmoïde est montrée dans l'équation 2.5 et la figure 2.7. S.Bashyal et.al [BAS2008] ont utilisé cette fonction pour entraîner

un réseau de neurones pour la classification d'incendie en utilisant un réseau de capteurs de gaz et ensuite l'appliquer au niveau d'un système embarqué en utilisant un microcontrôleur 89c55. Cependant, S.Bashyal et.al ont uniquement utilisé les nombres entiers et pas des mathématiques fractionnaire qui est un facteur limitant du travail et ils n'ont pas mentionné comment ils ont mis en œuvre la fonction d'activation dans le microcontrôleur.

$$f(x)=\frac{1}{1+e^{-x}} \qquad (2.5)$$

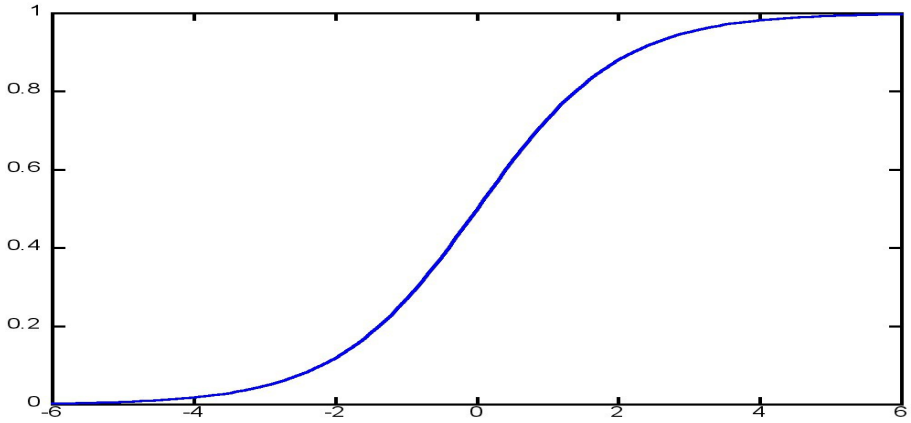


Figure 2.7: La function sigmoide .

La fonction présentée par D.L.Elliott est une autre fonction d'activation non-linéaire [ELL93]. Cette fonction se situe entre moins un et plus un , mais sa forme n'est pas aussi raide que tanh. On peut voir la fonction d'Elliott dans l'équation 2.6 et la figure 2.8.

$$f(x)=\frac{x}{1+|x|} \qquad (2.6)$$

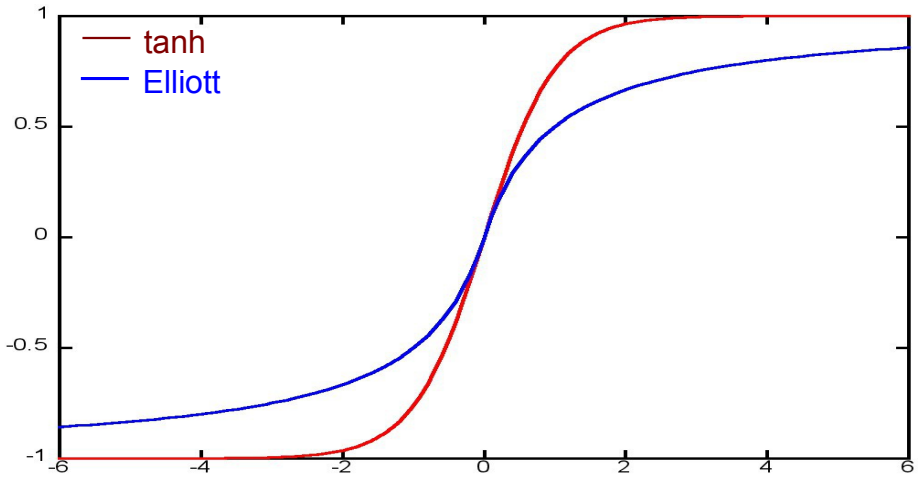


Figure 2.8: La fonction d'Elliott .

Si on multiplions la variable x dans la fonction d'Elliott par un facteur de 2 ; on aura une meilleure approximation pour la fonction Tanh. L'équation 2.7 et la figure 2.9 montrent la fonction d'Elliott multipliée par un facteur de 2.

$$f(x)=\frac{2*x}{1+2*|x|} \qquad (2.7)$$

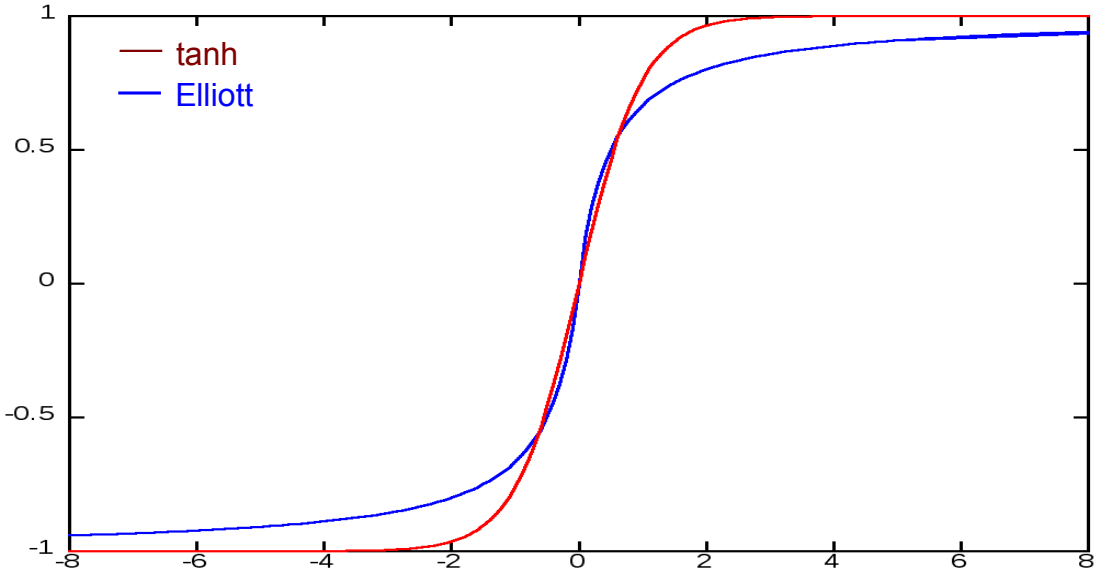


Figure 2.9 La fonction d'Elliott multipliée par un facteur de 2.

P.V. Vehviläinen et H.A.TIhalainen ont proposé une autre fonction d'activation afin de l'implémenter dans un microcontrôleur à faible coût qui ne supporte pas le matériel pour effectuer des calculs arithmétiques à virgule flottante [VEH2000]. Ils ont suggéré deux fonctions pour remplacer à la fois la fonction logistique sigmoïde (Eq 2.5) et de la fonction tangente hyperbolique (équation 2.4) qui sont asymptotiques et même continûment différentiables. La fonction suggérée pour remplacer la fonction logistique sigmoïde est représentée dans l'équation (2.8) et la fonction qui est censée remplacer la fonction tangente hyperbolique est donnée dans l'équation (2.9). Ces fonctions sont représentées dans les figures (2.10) et (2.11) respectivement.

$$f(x)=\begin{cases} 1-\dfrac{a}{1+(0.5*x+b)^3} & if\ x\geqslant 0 \\ \dfrac{a}{1-(0.5*x-b)^3} & if\ x<0 \end{cases} \quad \text{où} \quad a=\frac{3}{4}, b=\frac{1}{\sqrt[3]{2}} \qquad (2.8)$$

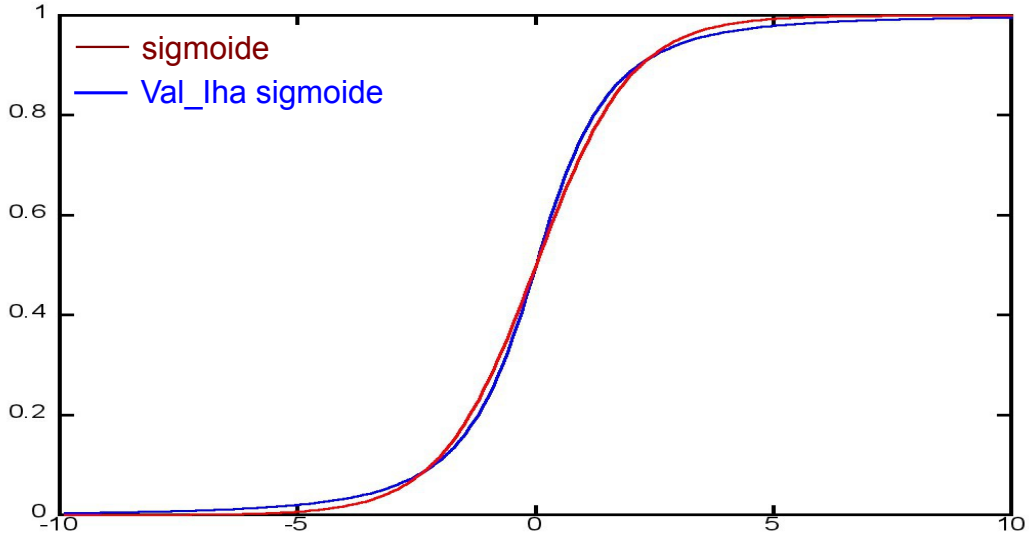


Figure 2.10 La fonctiond'Ihalainen & Vehviläinen Log-Sigmoide

$$f(x)=\begin{cases} 1-\dfrac{a}{1+(x+b)^3} & if\ x\geqslant 0 \\ \dfrac{a}{1-(x-b)^3}-1 & if\ x<0 \end{cases} \qquad \text{où} \quad a=\frac{3}{2}, b=\frac{1}{\sqrt[3]{2}} \qquad (2.9)$$

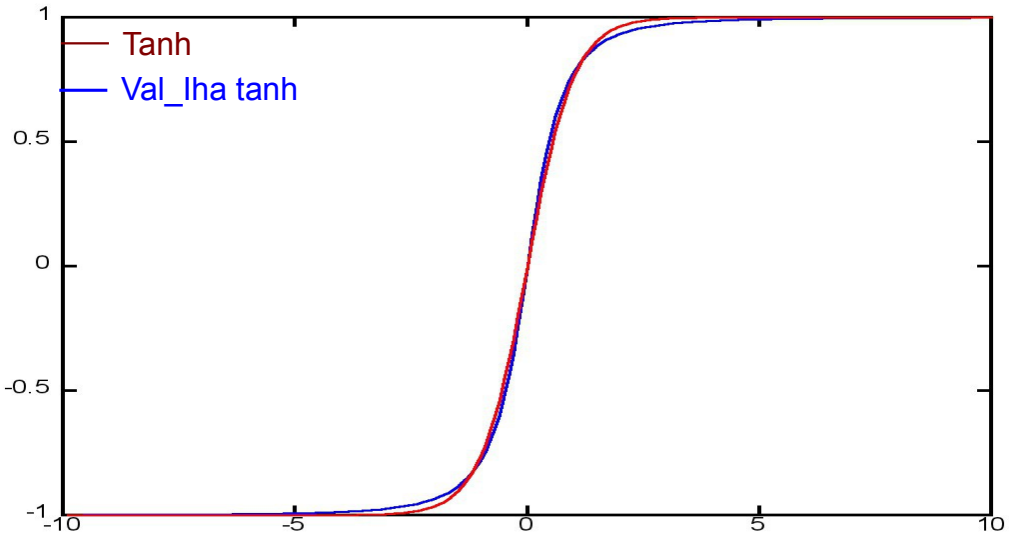


Figure 2.11 La fonction d'halainen & Vehviläinen Tangent-Hyperbolique

Nous pouvons voir clairement que ces fonctions sont les plus ressemblantes à les fonctions log-sigmoïde et tanh parmi les autres citées auparavant.

O. Postolache et.al ont utilisé cette fonction pour mettre en œuvre un ANN dans un PIC micro-

contrôleur 16F877 combiné avec un ADC non linéaire afin de linéariser un capteur de gaz [POS01]. L'ANN a été utilisé comme un sélecteur pour générer l'entrée à l'ADC en prenant comme entrées les effets de la température et l'humidité qui agissent comme des perturbations sur le capteur de gaz.

P.V.Vehviläinen et H.A.TIhalainen ont montré dans leur article par l' entraînement d'un réseau MLP par ces fonctions d'activation qu'elles peuvent faire aussi bien que les fonctions log-sigmoïde et tanh sans aucune perte significative dans les performances du réseau.

2.4. Algorithmes mis en œuvre

Il y a plusieurs algorithmes développés pour l'apprentissage des réseaux de neurones. Ici, nous discutons le plus utilisé d'entre eux.

2.4.1.Rétro-propagation d'erreur (EBP)

Cet algorithme est le EBP traditionnelle avec la capacité de gérer les réseaux de neurones arbitrairement connectés. L'EBP est l'algorithme le plus utilisé pour l'apprentissage des réseaux de neurones. Cette section va montrer comment l'EBP réalise la descente du gradient pour traîner un réseau entièrement connecté en cascade (FCN) illustré à la figure 2.12. Le réseau a une entrée, une sortie et quatre neurones.

Le calcul de l'algorithme EBP se fait en trois étapes:

1. calcul Forward.
2. calcul Backward.
3. calcul de gradient par élément.

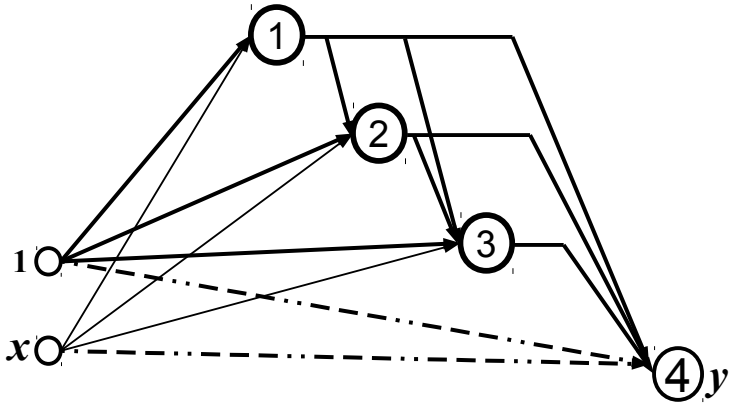


Figure 2.12 Un réseau de neurones complètement connecté en cascade avec une entrée et une sortie.

Calcul Forward:

Dans le calcul Forward, les neurones connectés aux entrées du réseau de neurones sont d'abord traités de telle sorte que leurs résultats peuvent être utilisés comme entrées aux neurones suivants. Les neurones suivants sont alors traitées dès que leurs valeurs d'entrée sont disponibles. En d'autres termes, la séquence de calcul choisie doit suivre le concept de propagation du signal Feedforward. Si un signal atteint les entrées de plusieurs neurones dans le même temps, ces neurones peuvent être traités dans n'importe quel ordre. Dans l'exemple de la figure 2.12, il n'y a qu'une seule façon possible dans lequel les neurones peuvent être traités dans le sens direct: $n_1n_2n_3n_4$. Lorsque le Calcul Forward est conclu, les deux vecteurs suivants temporaires sont stockés: le vecteur Y d'abord avec les valeurs des signaux de sorties des neurones, et le second vecteur S avec les valeurs des pentes des fonctions d'activation des neurones, qui dépendent sur les signaux de Y [NIC08].

Calcul Bacward:

La séquence du calcul Backward est opposée à la séquence Forward. Le processus commence par le dernier neurone et se poursuit vers l'entrée. Dans le cas du réseau de la figure 2.12, que pour le calcul Forward, il n'y a qu'une seule façon possible dans lequelle les neurones peuvent être traités dans le sens Backward: (chemin de rétro-propagation): $\mathbf{n_4n_3n_2n_1}$. Le vecteur $\boldsymbol{\delta}$ représente la propagation du signal de sortie du réseau vers les entrées de tous les autres neurones. La taille de ce vecteur est égal au nombre de neurones. Pour le neurone de sortie n_4, sa sensibilité est initiée à l'aide de sa pente : $\delta 4 = s4$. Pour le neurone n_3, le delta à n_4 sera propagée par w_{34} : le poids entre $\mathbf{n_3}$ et n_4, puis par la pente de neurone n_3. Ainsi, le paramètre delta de n_3 est présenté comme $\delta 3 = \delta_4 w_{34} s_3$. Pour le neurone n_2, les paramètres delta de n_3 et n_4 sont propagés à la sortie du neurone n_2 à travers les poids w_{24} et w_{23} respectivement et additionnés, puis multipliés par la pente du neurone n_2, comme $\delta 2 = (\delta_4 w_{24} + \delta_3 w_{23}) s_2$. Avec le même procédure, il pourrait être obtenu que $\delta 1 = (\delta_2 w_{12} + \delta_3 w_{13} + \delta_2 w_{12}) s_1$. Après le processus de rétro-propagation est fait à neurone n_1, tous les éléments du vecteur δ sont obtenus [NIC08].

Calcul du gradient :

Après les calculs Forward et Backward, les vecteurs de sorties de neurones Y et de pentes δ sont calculés. En appliquant tous les modèles d'apprentissage, puis en utilisant l'équation 2.10, les éléments de gradient pour un poids donné peuvent être obtenus comme suit :

$$g(n,k)=\sum_{p=1}^{P}\sum_{o=1}^{O}\frac{\partial e_{po}}{\partial w_{nk}}e_{po} \qquad (2.10)$$

où:

P: nombre de modèles.

O: nombre de sorties.

n: est l'indice du nombre de neurones.

k: est l'indice des entrées de neurones.

$$\frac{\partial e_{po}}{\partial w_{nk}}=y_{nk}*\delta_k \qquad (2.11)$$

La matrice du gradient peut être alors calculée et stockée.

la règle de mise du poids est donnée par:

$$\Delta w=\alpha*g \qquad (2.12)$$

où α est une constante définie par l'utilisateur connue comme le taux d'apprentissage utilisé pour ajuster la taille du pas. Le nouveau poids est alors représenté par:

$$w_{new}=w+\Delta w \qquad (2.13)$$

Le paramètre alpha est un multiplicateur qui agit comme la valeur numérique de la taille du pas dans la direction du gradient. Si alpha est trop grand, l'algorithme peut osciller au lieu de réduire l'erreur. Cependant, si alpha est trop petit, l'algorithme peut se déplacer vers la solution trop lentement et stabilise. Ce paramètre doit être ajusté par l'utilisateur jusqu'une valeur optimale est trouvée, qui a une certaines oscillations qui diminue avec le temps alors que l'erreur continue à diminuer [LEC98].

Ce processus est ensuite répété des milliers de fois dans l'espoir d'atteindre l'erreur souhaitée. L'EBP donne des bons résultats dans nombreux cas, mais il présente plusieurs inconvénients. Le premier étant l'algorithme est très lent à converger si il converge. L'algorithme approche généralement le voisinage du minimum d'erreur très rapidement, mais puis ralentit considérablement, en prenant beaucoup plus de temps pour se rapprocher à la solution finale que quand il a commencé. Ce processus continue de ralentir comme il se rapproche, rendant ainsi très difficile d'obtenir une réponse précise [NIC08].

Le pseudo code de l'algorithme de EBP est représenté ci-dessous:

Pour chaque modèle ***p***
pour chaque neurone ***n*** (commençant de 1 à N)

```
        net=0;
        pour chaque entrée i au neurone n
            net =net + w(i,n)*entrée(p, (i,n))
        fin entrée
      node(p,n)=actfun(net)        //  calculer la sortie d'un neurone
      slop(p,n)=(1-node(p,n)^2)    // calculer la pente de sortie pour  une fonction d'activation Tanh
   fin neurone

   Pour chaque neurone de sortie ( commençant de N-O à N)
      err(p,o) =desired(p,o) – node(p,o)
   fin neurone de sortie
fin modèle

errnode(n)= err(p,o)

Pour chaque modèle p
    Pour chaque neurone de sortie o
        pour chaque neurone n (commençant de N à 1)
            delta(n)=slop(p,n)* errnode(n)
             pour chaque entrée i au neurone n
                errnode(ind(i,n))= errnode(ind(i,n))+   delta(n)*w(i)
            fin entrée
         fin neurone
     fin neurone de sortie
fin modèle
```

Plusieurs améliorations pour l'EBP ont été développées afin d'améliorer la rapidité et la convergence des réseaux neuronaux telles que l'algorithme quickprop, EBP résilient (Rprop), la percolation de retour, et le delta-bar-delta. Nous décrivons ici certains d'entre eux brièvement.

2.4.2.Rétro-propagation avec un terme d'inertie (momentum):

Lorsque le minimum de la fonction d'erreur pour une tâche d'apprentissage donnée se situe dans une étroite «vallée», en suivant la direction du gradient , cela peut conduire à des oscillations larges au processus de recherche. La figure 2.1 montre un exemple pour un réseau avec juste deux poids W1 et W2. La meilleure stratégie dans ce cas est d'orienter la recherche vers le centre de la vallée, mais la forme de la fonction d'erreur est telle que le gradient ne pointe pas dans cette direction. Une solution simple consiste à introduire un terme d'inertie (momentum). Le gradient de la fonction d'erreur est calculée pour chaque nouvelle combinaison de poids, mais au lieu de simplement suivre la direction du gradient négatif, une moyenne pondérée de la pente actuelle et la direction de correction précédente est calculée à chaque étape. Théoriquement, cette approche devrait fournir au processus de recherche une

sorte d'inertie et pourrait aider à éviter les oscillations excessives dans les vallées étroites de la fonction d'erreur [ROJ96].

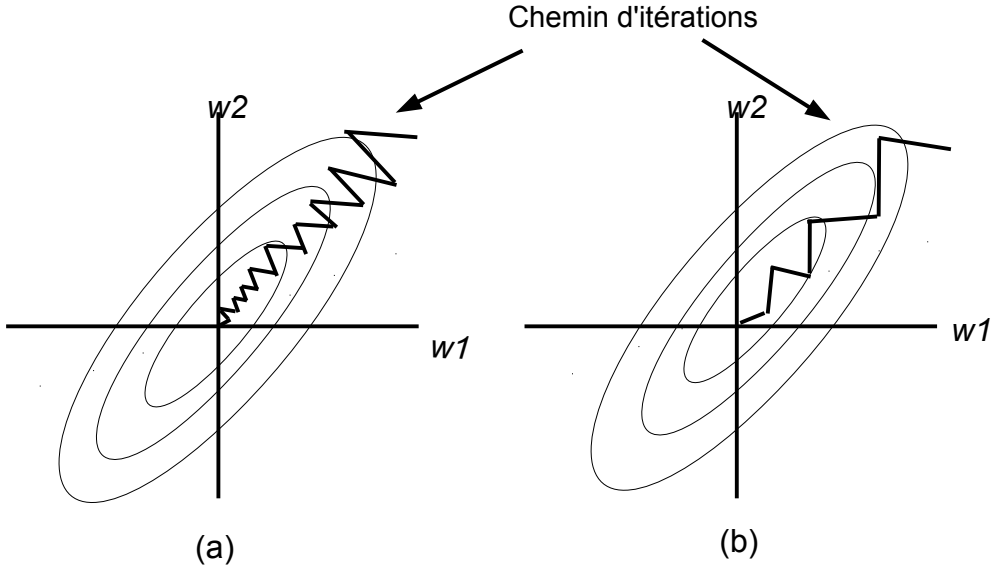


Fig. 2.13 Rétropropagation (a) sans momemt (b) avec moment

L'équation de la mise des poids est donnée par :

$$\Delta w(t+1) = -\eta \frac{\partial E_{t+1}}{\partial w} + \alpha \Delta w(t) \qquad (2.14)$$

où η et α sont les constantes d'apprentissage et le terme d'élan respectivement.

2.4.3.Rprop:

Cet algorithme est d'abord proposé par Ried-Miller et Braun [RIE93]. L'idée principale de l'algorithme est de mettre à jour les poids du réseau neuronale en utilisant simplement le taux d'apprentissage et le signe de la dérivée partielle de la fonction d'erreur à l'égard de chaque poids. Cela accélère l'apprentissage principalement dans les régions plates de la fonction d'erreur ainsi que lors qu'elle est arrivée près d'un minimum local [ROJ96].

Les taux d'apprentissage sont mis à jour dans l'itération t-ème par:

$$\Delta w(t)_{ij} = \begin{cases} -\Delta(t)_{ij} & if \ \frac{\partial E(t)}{\partial w_{ij}} > 0 \\ \Delta(t)_{ij} & if \ \frac{\partial E(t)}{\partial w_{ij}} < 0 \\ 0 & else \end{cases} \qquad (2.15)$$

où $\frac{\partial E(t)}{\partial w_{ij}}$ désigne l'information du gradient additionnée sur tous les modèles (lot d'apprentissage).

La deuxième étape de l'apprentissage est de déterminer la nouvelle mise des valeurs ***Δ(t)ij***. Ceci est basé sur un processus d'adaptation dépendant de signe du gradient à l'état actuel et l'état passé.

$$\Delta(t)_{ij}=\begin{cases}\eta^{+}*\Delta(t-1)_{ij} & if \ \frac{\partial E(t-1)}{\partial w_{ij}}*\frac{\partial E(t)}{\partial w_{ij}}>0\\ \eta^{-}*\Delta(t)_{ij} & if \ \frac{\partial E(t-1)}{\partial w_{ij}}*\frac{\partial E(t)}{\partial w_{ij}}<0\\ \Delta(t-1)_{ij} & else\end{cases} \qquad (2.16)$$

Autrement dit, la règle de l'adaptation fonctionne comme suit: Chaque fois que la dérivée partielle du poids ***wij*** correspondant change de signe, qui indique que la dernière mise à jour était trop grand et que l'algorithme a sauté par-dessus un minimum local, la mise de ***Δ(t)ij*** est diminuée par le facteur η^{-}. Si la dérivée conserve sa signe, la mise de valeur est légèrement augmentée afin d'accélérer la convergence dans les régions peu profondes [ROJ96].

2.4.4.Quickprop:

Une méthode pour accélérer l'apprentissage est d'utiliser l'information sur la courbure de la surface d'erreur. Cela nécessite le calcul de la dérivée seconde de la fonction d'erreur. Quickprop assume que la surface d'erreur soit localement quadratique et tente de sauter en une seule étape à partir de la position actuelle directement vers le minimum de la parabole.

Quickprop calcule les dérivées dans le sens de chaque poids. Après que le calcul du gradient soit fait avec rétro-propagation régulière dans une première étape, une étape directe à l'erreur minimale est tentée par :

$$\Delta w_{ij}(t+1)=\Delta w_{ij}(t)\left(\frac{\nabla E_{t+1}}{\nabla E_{t}-\nabla E_{t+1}}\right) \qquad (2.17)$$

où :

∇E_t, ∇E_{t+1} :les derivées partielles de la fonction d'erreur aux étapes t et t+1 respectivement.

$\Delta w_{ij}(t)$, $\Delta w_{ij}(t+1)$: les changements de poids aux étapes ***t*** et ***t+1*** respectivement.

Notez que si nous écrivons l'équation (2.17) comme suit:

$$\Delta w_{ij}(t+1)=\frac{-\nabla E_{t+1}}{(\nabla E_{t+1}-\nabla E_{t})/\Delta w_{ij}(t)} \qquad (2.18)$$

Le dénominateur est juste une approximation discrète de dérivée de second ordre, donc l'algorithme de

Quickprop est une approximation discrète de l'algorithme de Newton.

L'apprentissage stochastique (en ligne) par rapport à l'apprentissage batch:

L'apprentissage stochastique est plus souvent beaucoup plus vite que l'apprentissage batch en particulier dans les ensembles de données redondantes. Il en résulte également de meilleures solutions à cause du bruit dans les mises de poids. L'apprentissage stochastique est également utile lorsque la fonction en cours de modélisation évolue avec le temps, un scénario assez fréquent dans les applications industrielles où les données changent progressivement au fil du temps (par exemple due à la déchirure et l'usure des machines).

Les algorithmes stochastiques d'apprentissage pour les réseaux de neurones exigent moins de stockage et moins de calculs à chaque itération, qui est particulièrement important lorsque l'apprentissage des ensembles qui sont grandes et redondantes. Le bruit inhérent au processus d'apprentissage diminue aussi les chances de devenir coincé dans des minima locaux. Dans le même temps, la convergence (par exemple en moyenne quadratique) est garantie si le taux d'apprentissage est recuit (diminué lentement vers zéro) à des moments tard pour supprimer le bruit. Cependant, les techniques d'accélération basées sur l'estimation de la courbure locale de la surface de coûts telles que les méthodes du second ordre ne peuvent pas être mises en œuvre stochastiquement parce que les estimations des effets de second ordre sont beaucoup trop bruyantes [LEC98].

2.4.5.Algorithme de Levenberg-Marquardt:

L'algorithme populaire de Levenberg-Marquardt (LM) (Levenberg, 1944; Marquardt, 1963) est une technique amortie des moindres carrés pour les modèles non-linéaires, où une constante positive (amortissement) est ajoutée à la diagonale de la matrice jacobienne afin de contrôler le comportement du système et éviter la singularité. Cette méthode combine les avantages de la méthode de Gauss-Newton et la méthode du gradient descente. Si l'amortissement utilisé à une itération réduit l'erreur, l'amortissement est divisé par une constante de réduction avant la prochaine itération et la convergence vers la solution est accélérée. Si l'erreur augmente, alors que l'amortissement est multipliée par une constante d'amplification, ce qui rend la convergence lente mais veiller à ce qu'une solution puisse être trouvée. De cette façon, la méthode passe d'une technique à l'autre en douceur. L'algorithme de LM utilise la méthode du gradient descente lorsque les résultats sont loin d'être le minimum, mais que la solution se rapproche du minimum, l'algorithme passe à la méthode de Gauss-Newton, qui aura

tendance d'avoir un pas de taille zéro lorsque on approche le meilleur ajustement.

L'équation de mise de poids dans l'algorithme de LM est donnée par:

$$w_{k+1} = w_k - \left(J_k^T J_k + \mu I \right)^{-1} J_k^T e_k \qquad (2.19)$$

où:

w_{k+1}, w_k sont les vecteurs de poids à l'itération k +1 et k.

I :est la matrice d'identité.

J_K, e_k : sont la matrice jacobienne et le vecteur d'erreur à l'itération k, qui sont définis comme suit:

$$J_k = \begin{bmatrix} \frac{\partial e_{11}}{\partial w_1} \frac{\partial e_{11}}{\partial w_2} \cdots \frac{\partial e_{11}}{\partial w_N} \\ \frac{\partial e_{12}}{\partial w_1} \frac{\partial e_{12}}{\partial w_2} \cdots \frac{\partial e_{12}}{\partial w_N} \\ \cdots \\ \cdots \\ \frac{\partial e_{1M}}{\partial w_1} \frac{\partial e_{1M}}{\partial w_2} \cdots \frac{\partial e_{1M}}{\partial w_N} \\ \frac{\partial e_{P1}}{\partial w_1} \frac{\partial e_{P1}}{\partial w_2} \cdots \frac{\partial e_{P1}}{\partial w_N} \\ \frac{\partial e_{P2}}{\partial w_1} \frac{\partial e_{P2}}{\partial w_2} \cdots \frac{\partial e_{P2}}{\partial w_N} \\ \cdots \\ \cdots \\ \frac{\partial e_{PM}}{\partial w_1} \frac{\partial e_{PM}}{\partial w_2} \cdots \frac{\partial e_{PM}}{\partial w_N} \end{bmatrix} \qquad (2.20)$$

$$e_k = \begin{bmatrix} e_{11} \\ e_{12} \\ \cdots \\ e_{1M} \\ \cdots \\ e_{P1} \\ e_{P2} \\ \cdots \\ e_{PM} \end{bmatrix} \qquad (2.21)$$

où P est le nombre de modèles d'apprentissage , M est le nombre de sorties, N est le nombre de poids.

Les éléments de vecteur d'erreur est calculés par:

$$e_{pm} = d_{pm} - o_{pm} \qquad (2.22)$$

où les $\mathbf{d_{pm}}$ et $\mathbf{o_{pm}}$ sont les valeurs de sortie désirée et sortie réelle respectivement à la sortie m du réseau pendant l'apprentissage du modèle p.

Le processus standard d'apprentissage ***LM*** peut être illustré dans le pseudo-codes suivant :

1. Initialiser les poids et le paramètre μ_0.
2. Calculer la somme des erreurs carrées pour toutes les entrées $F(w)$.
3. Résoudre l'équation (2.19) pour obtenir l'incrément des poids Δw .
4. Recalculer la somme des erreurs carrées $F_{essai}(w)$ en utilisant $w + \Delta w$ comme essai , puis juger

Si $F_{essai}(w) < F(w)$ en étape 2 alors

$$w = w + \Delta w$$
$$\mu = \mu * \beta_1$$

Aller au étape 2

Sinon

$$\mu = \mu / \beta_2$$

Aller au étape 4

Fin Si

Malheureusement, l'algorithme LM présente quelques difficultés liées au calcul du taux de changement d'amortissement, qui est contrôlé par quatre paramètres principaux: l'amortissement initial, les constantes d'amplification et de réduction , et l'amortissement minimal. Un choix inapproprié de ces paramètres causera l'algorithme de diverger ou converger trop fortement ou trop lentement [LAM97]. En outre, en raison de la présence de l'amortissement, ce qui augmente les valeurs de valeurs propres, empêchant de cette manière la singularité du système, la présence d'un amortissement minimal est fondamentale. Cette valeur minimale affecte surtout la rapidité et la stabilité de la méthode, et une quantité adéquate est essentielle pour l'efficacité de l'algorithme de LM.

2.4.6.Neurone par neurone (Neuron By Neuron (NBN))

NBN est une version modifiée de l'algorithme de Levenberg-Marquardt pour les réseaux de neurones entièrement connectés en cascade ou arbitrairement connectés développée par BM Wilamowski et H. Yu au sein de l'université d'Auburn. En comparaison avec le célèbre algorithme de Levenberg-Marquardt, l'algorithme NBN a plusieurs avantages:

1. la capacité de manipuler des réseaux de neurones arbitrairement connectés.
2. possibilité d'avoir uniquement le calcul Forward (sans processus de rétro-propagation) et
3. calcul direct de la quasi-matrice hessienne (pas besoin de calculer et stocker la matrice jacobienne).

L'algorithme de NBN est brièvement décrit dans [WIL08]. L'équation de mise à jour de poids est la même que pour l'algorithme de Levenberg-Marquardt:

$$w_{k+1} = w_k - \left(J_k^T J_k + \mu I \right)^{-1} J_k^T e \qquad (2.23)$$

Si μ = 0, alors l'algorithme devient la méthode de Gauss-Newton. Pour de très grandes valeurs μ, l'algorithme devient la méthode du rétro-propagation d'erreur ou EBP.

Comme l'algorithme EBP, le calcul de l'algorithme de NBN se fait en trois étapes:

1. Calcul Forward.
2. Calcul Backward.
3. Calcul des éléments jacobiens.

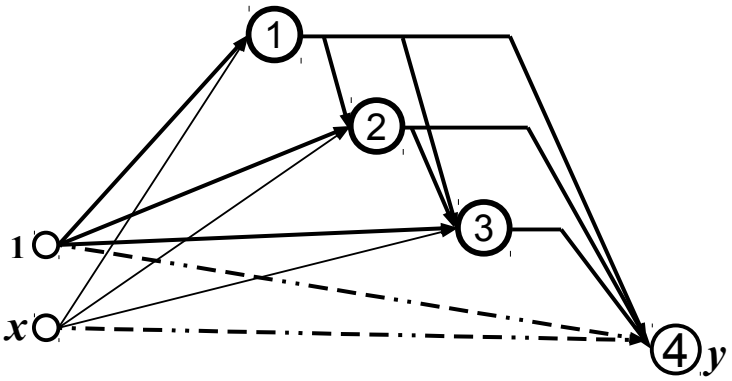


Figure 2.14 Un réseau de neurones complètement connecté en cascade avec une entrée et une sortie.

Calcul Forward:

Dans le calcul Forward, les neurones connectés aux entrées du réseau de neurones sont d'abord traités de telle sorte que leurs résultats peuvent être utilisés comme entrées aux neurones suivants. Les neurones suivants sont alors traités dès que leurs valeurs d'entrée sont disponibles. En d'autres termes, la séquence de calcul choisie doit suivre le concept de propagation du signal Feedforward. Si le signal atteint les entrées de plusieurs neurones dans le même temps, ces neurones peuvent être traités dans n'importe quel ordre. Dans l'exemple de la figure 2.14, il n'y a qu'une seule façon possible dans lequel les neurones peuvent être traitées dans le sens direct: $\mathbf{n_1 n_2 n_3 n_4}$. Lorsque le Calcul Forward est conclu, les deux vecteurs suivants temporaires sont stockés: le vecteur $\boldsymbol{Y}$ d'abord avec les valeurs des signaux de sorties des neurones, et le second vecteur $\boldsymbol{S}$ avec les valeurs des pentes des fonctions d'activation des neurones, qui dépendent sur les signaux de $\boldsymbol{Y}$.

Calcul Bacward:

La séquence du calcul Backward est opposée à la séquence Forward. Le processus commence par le dernier neurone et se poursuit vers l'entrée. Dans le cas du réseau de la figure 2.12, que pour le calcul Forward, il n'y a qu'une seule façon possible dans lequelle les neurones peuvent être traités dans le sens Backward: (chemin de rétro-propagation): $\mathbf{n_4 n_3 n_2 n_1}$. Le vecteur $\boldsymbol{\delta}$ représente la propagation du signal de

sortie du réseau vers les entrées de tous les autres neurones. La taille de ce vecteur est égal au nombre de neurones. Pour le neurone de sortie n_4, sa sensibilité est initiée à l'aide de sa pente : $\delta 4 = s4$. Pour le neurone n_3, le delta à n_4 sera propagée par w_{34} : le poids entre n_3 et n_4, puis par la pente de neurone n_3. Ainsi, le paramètre delta de n_3 est présenté comme $\delta 3 = \delta_4 w_{34} s_3$. Pour le neurone n_2, les paramètres delta de n_3 et n_4 sont propagés à la sortie du neurone n_2 à travers les poids w_{24} et w_{23} respectivement et additionnés, puis multipliés par la pente du neurone n_2, comme $\delta 2 = (\delta_4 w_{24} + \delta_3 w_{23}) s_2$. Avec le même procédure, il pourrait être obtenu que $\delta 1 = (\delta_2 w_{12} + \delta_3 w_{13} + \delta_2 w_{12}) s_1$. Après le processus de rétro-propagation est fait à neurone n_1, tous les éléments du vecteur δ sont obtenus [NIC08].

Calcul des éléments jacobiens:

Après les calculs Forward et Backward, les vecteurs de sorties de neurones Y et de pentes δ sont calculés. La ligne jacobienne pour un modèle de donnée peut être obtenue. En appliquant tous les modèles d'apprentissage, toute la matrice Jacobienne peut être calculée et stockée.

Pour les réseaux de neurones connectés arbitrairement, l'algorithme NBN peut être organisé comme le montre la figure 13.5.

```
Pour touts les modèles  (np)

   % Calcul Forward
      pour touts les neurones (nn)
          pour touts les poids d'un neurone  (nx)
              calculer  l'entrée du neurone nn;
          fin;
         calculer la sortie du neurone;
          calculer la pente de sortie du neurone;
      fin;

      pour toutes les sorties (no)
           calculer l'erreur;

   % Calcul Backward

       initialiser  delta  = pente ;
          pour touts les neurones ( commençant par les neurones des sorties (nn))
              pour les poids connectés aux autres neurones (ny)
                  multiplier delta par les poids
                  sommer   delta aux nœuds appropriés
              fin;
              multiplier delta par la pente (pour les neurones cachés);
           fin;
```

```
        calcul de la matrice jacobienne;
        fin;
fin;
```

2.4.7.Gradient évolutif (Evolutionary Gradient):

Gradient évolutive est un algorithme nouvellement développé, qui évalue les gradients à partir d'ensemble de poids générés aléatoirement et utilise les informations du gradient pour générer une nouvelle population de poids. Il s'agit d'un algorithme hybride qui combine l'utilisation des populations aléatoires avec une approche du gradient approximé. Comme les méthodes standard de calcul évolutif, l'algorithme est mieux adapté pour éviter les minima locaux par rapport aux méthodes de gradient communs tels que EBP. En générale, l'algorithme est capable de converger beaucoup plus rapidement que d'autres formes de calcul évolutif. Cette combinaison de méthodes de gradient et évolutives offre essentiellement le meilleur des deux méthodes. Cet algorithme a été écrit par Joel Hewlett et les détails concernant ces paramètres peut être trouvée dans [HEWL07].

Chapitre 3

Simulations et Tests Expérimentaux

Dans ce chapitre, nous allons essayer de linéariser un capteur de thermocouple type K en appliquant quelques méthodes parmi les discutées avant dans le chapitre 1 et 2. Ces méthodes sont:

1- Interpolation polynomiale de Lagrange.

2- Méthode d'étalonnage polynomiale Progressive.

3- Réseaux de neurones.

puis nous discuterons l'efficacité de chaque méthode en termes de précision, rapidité et flexibilité.

3.1. Thermocouple type K:

La caractéristique d'une thermocouple type K et sa approximation BLSF (Best Fit Linear Approximation) sont présentées dans la figure (3.1)

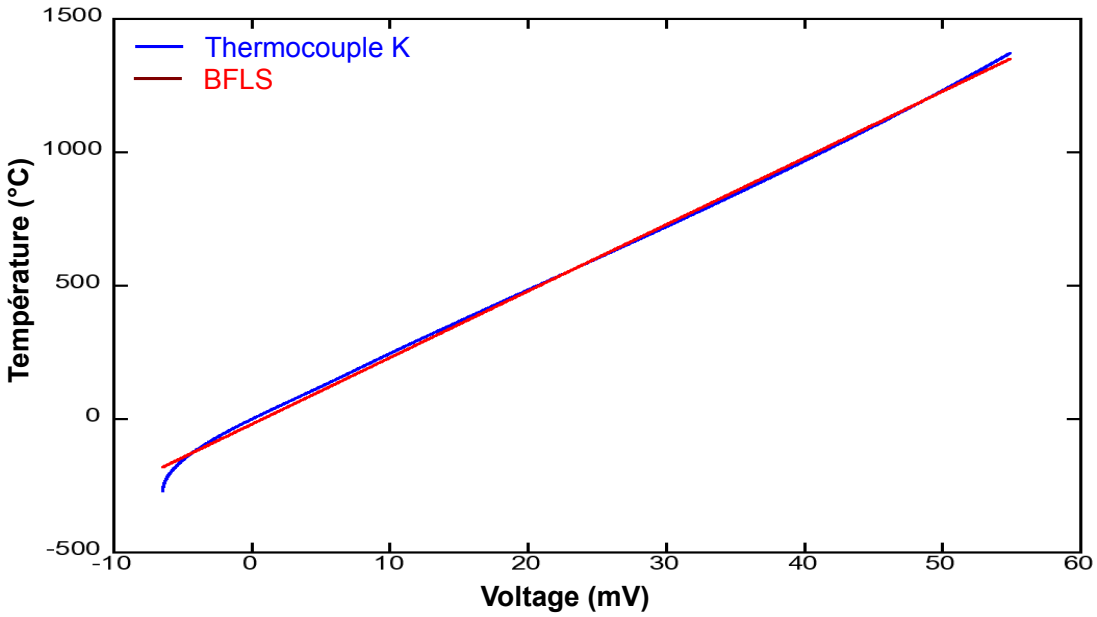


Figure 3.1. Caractéristique de thermocouple K et sa appriximation BFLS.

L'erreur de non-linéarité est montrée dans la figure (3.2):

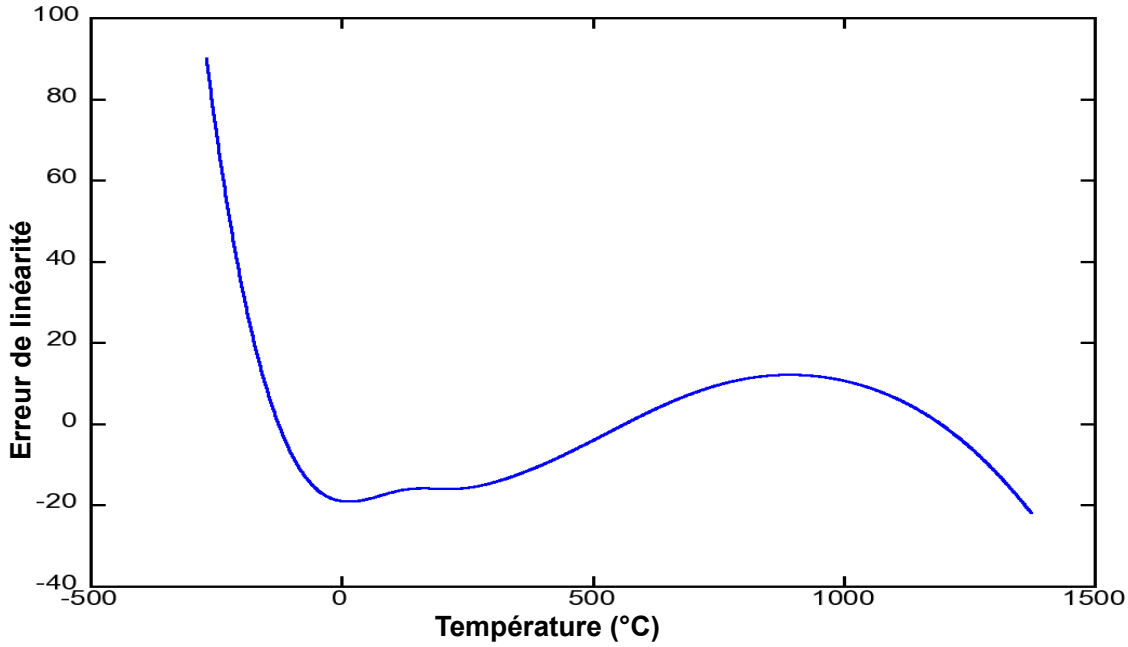


Figure 3.2. Erreur de non-linéarité de thermocouple K

Les données de la thermocouple K ont été fournis par le National Institute of Standards and Technology (NIST). Ils ont également fourni des coefficients pour les approximations de fonctions inverses pour les sous-gammes de température et de tension ci-dessous:

Température (^{0}C)	-200 à 0	0 à 500	500 à 1372
Voltage (mV)	-5.891 à 0.000	0.000 à 20.644	20.644 à 54.886
Coefficients (^{0}C/mV)	c_0 = 0.0000000E+00 c_1 = 2.5173462E+01 c_2 = -1.1662878E+00 c_3 = -1.0833638E+00 c_4 = -8.9773540E-01 c_5 = -3.7342377E-01 c_6 = -8.6632643E-02 c_7 = -1.0450598E-02 c_8 = -5.1920577E-04 c_9 = 0.0000000E+00	c_0 = 0.000000E+00 c_1 = 2.508355E+01 c_2 = 7.860106E-02 c_3 = -2.503131E-01 c_4 = 8.315270E-02 c_5 = -1.228034E-02 c_6 = 9.804036E-04 c_7 = -4.413030E-05 c_8 = 1.057734E-06 c_9 = -1.052755E-08	c_0 = -1.318058E+02 c_1 = 4.830222E+01 c_2 = -1.646031E+00 c_3 = 5.464731E-02 c_4 = -9.650715E-04 c_5 = 8.802193E-06 c_6 = -3.110810E-08 c_7 = 0.00000000E+00 c_8 = 0.00000000E+00 c_9 = 0.0000000E+00
Erreur	-0.02 à 0.04	-0.05 à 0.04	-0.05 à 0.06
RMS	0.014167	0.018898	0.020997

La fonction d'approximation inverse pour chaque sous-gamme est donnée par:

$$T(^0C) = c_0 + c_1 * V + c_2 * V^2 + c_3 * V^3 + c_4 * V^4 + c_5 * V^5 + c_6 * V^6 + c_7 * V^7 + c_8 * V^8 + c_9 * V^9$$

où V est la tension de sortie de thermocouple donnée (mV).

Ainsi, dans chaque sous-gamme, ils ont utilisé 10 points de données pour calculer la fonction d'approximation inverse.

L'erreur des fonctions d'approximation inverses est présentée dans la figure (3.3):

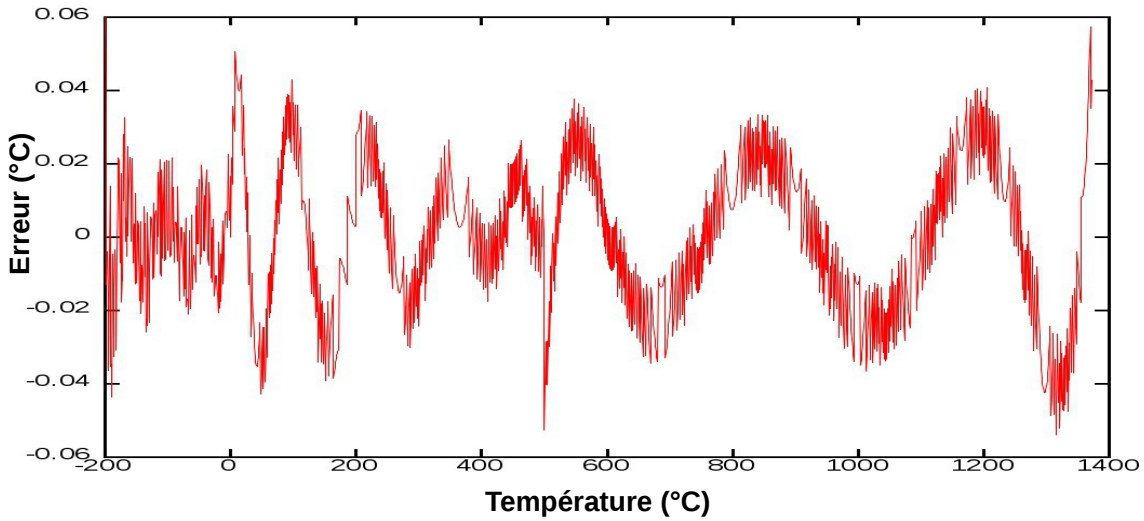


Figure 3.3. Erreur d'approximation inverse d'une thermocouple type K.

Dans le reste de ce chapitre, nous allons comparer les résultats obtenus par les méthodes appliquées avec ceux donnés par NIST.

3.2. Méthode d'étalonnage polynomiale Progressive :

La méthode d'étalonnage polynomiale progressive semble d'être une solution attrayante pour la linéarisation des capteurs intelligents, car elle nécessite un nombre réduit de points d'étalonnage, un petit nombre d'emplacements de mémoire pour stocker les coefficients d'étalonnage, et une simple évaluation algorithmique des coefficients d'étalonnage, sans itérations mathématiques. En outre, cette méthode permet la mise en œuvre d'un procédure d'étalonnage étape par étape en utilisant un algorithme répétitif qui est particulièrement bien adapté à mettre en œuvre pour une calibrage dynamique et adaptatif des données mesurées. Dans ce cas, l'inclusion d'un point de calibrage supplémentaire dans les données initiales de calibrage conserve les coefficients de correction qui sont déjà évalués, en minimisant la charge du calcul associée à des méthodes classiques d'étalonnage polynomiales.

L'ensemble des points de données fourni par le NIST comprend 1643 point entre [-270 1372]°C.
Le procédé de choisir les points pour calibrer le capteur est comme suit:
1 - Les premier et second points sont les premiers et les derniers points de l'ensemble de données pour corriger offset et de gain d'erreur.
2 - le troisième point est dans la proximité du milieu de l'ensemble de données.
3 - Pour les autres points, nous allons les disperser dans toute la gamme de données afin de compenser les non-linéarités dans toute la gamme.

Cas 1: intervalle [500 ° C, 1773 ° C]

Après plusieurs tentatives, nous avons obtenu une petite erreur que celle donnée par NIST avec seulement 8 points.
Les points utilisés pour l'étalonnage sont les suivants: [500 1371 988 728 1228 608 1128 858] ° C.
Figure (3.4) montre l'erreur après le calibrage par rapport à l'erreur du NIST.
Nous pouvons extrapoler le polynôme obtenu à l'intervalle [400 500] (° C) comme nous pouvons voir dans la figure 3.4, sans perdre en précision.

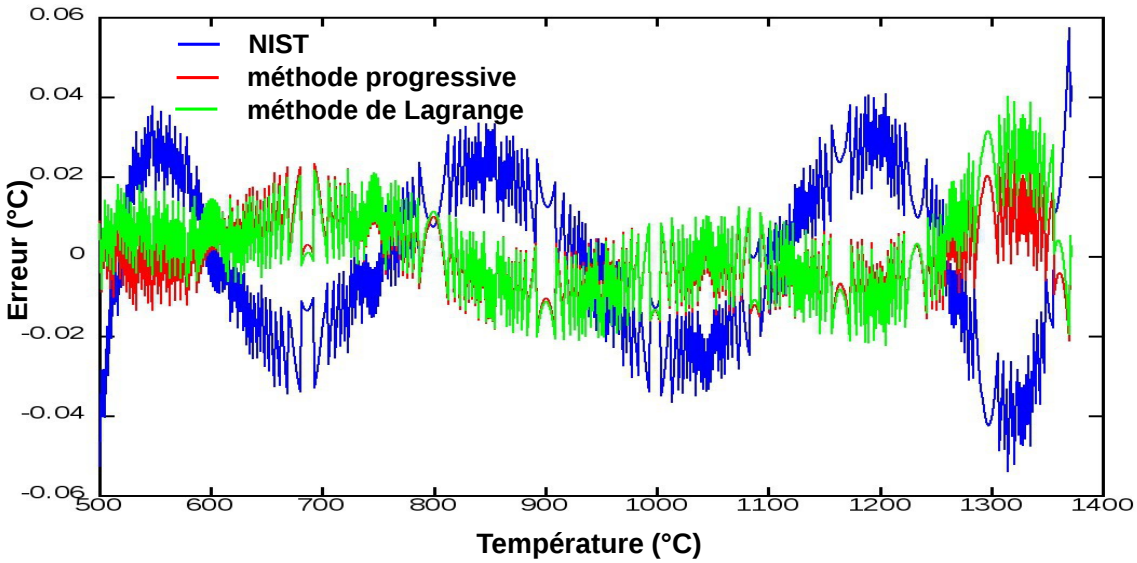


Figure 3.4.Erreurs de calibration de polynômes de lagrange et progressive Dans la plage de température [500 , 1373]°C

L'erreur RMS du polynôme progressive dans l'intervalle [400 1373] (° C) est égale à : 0.0101 ce qui représente environ la moitié des erreurs RMS du NIST dans la gamme [500 1373] (° C).

L'erreur RMS du polynôme d'interpolation de Lagrange dans l'intervalle [400 1373] (° C) est égale à : 0.0120.

$$RMS_{prog} = 0.0101$$
$$RMS_{lagr} = 0.0120$$

De la figure, nous pouvons voir que la méthode d'étalonnage polynomial progressive a une précision peu mieux que la méthode de Lagrange et elle extrapole mieux dans la gamme [400 500] (° C) que le polynôme de Lagrange.

Cas 2: intervalle [0 ° C, 400 ° C]

Après plusieurs tentatives, nous avons trouvé une petite erreur que celle donnée par NIST avec 9 points.

Les points utilisés pour l'étalonnage sont les suivants: [0 400 249 26 349 119 189 73 329] ° C.

La figure 3.6 représente les erreurs qui en résultent pour les méthodes d'interpolation de Lagrange et progressive.

Les deux méthodes ont approximativement les mêmes résultats.

Les erreurs RMS sont les suivantes: $RMS_{prog} = 0.0161$
$RMS_{lagr} = 0.0166$

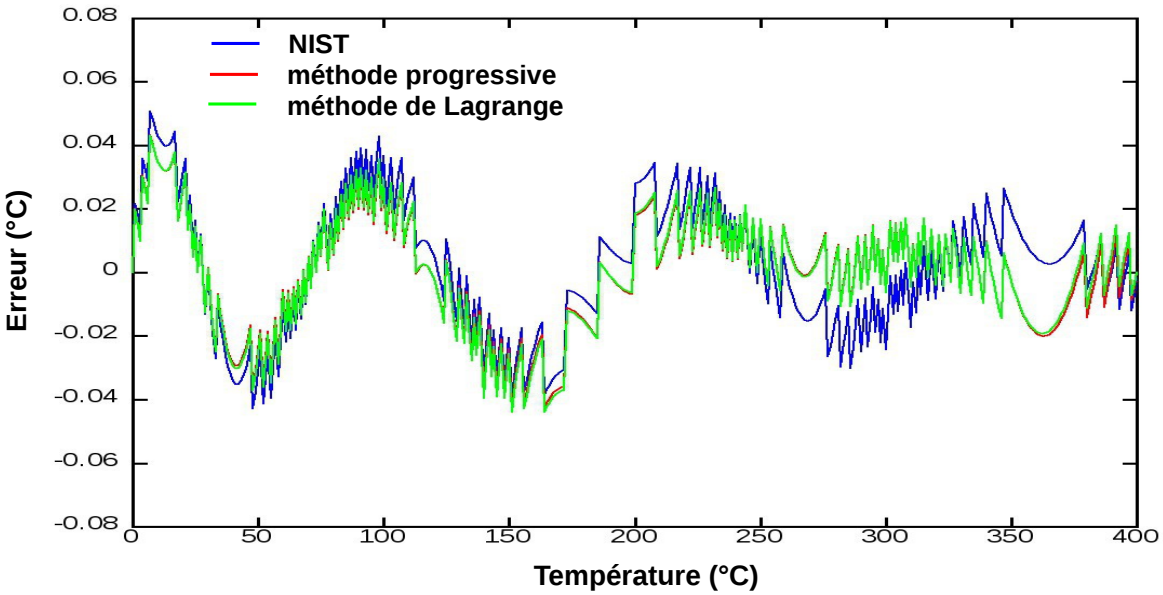


Figure 3.5.Erreurs de calibration de polynômes de lagrange et progressive Dans la plage de température [0 ,400]°C

Cas 3: intervalle [-200 ° C, 0 ° C]

Avec 8 points, L'étalonnage polynomiale progressive a atteint une plus grande précision tandis que la méthode des polynômes de Lagrange a eu un manque de précision avec les mêmes points.

Les points utilisés pour l'étalonnage sont les suivants: [70 270 170 120 220 90 145 250] ° C.

La figure 3.7 présente les erreurs qui en résultent pour les méthodes d'interpolation de Lagrange et progressive.

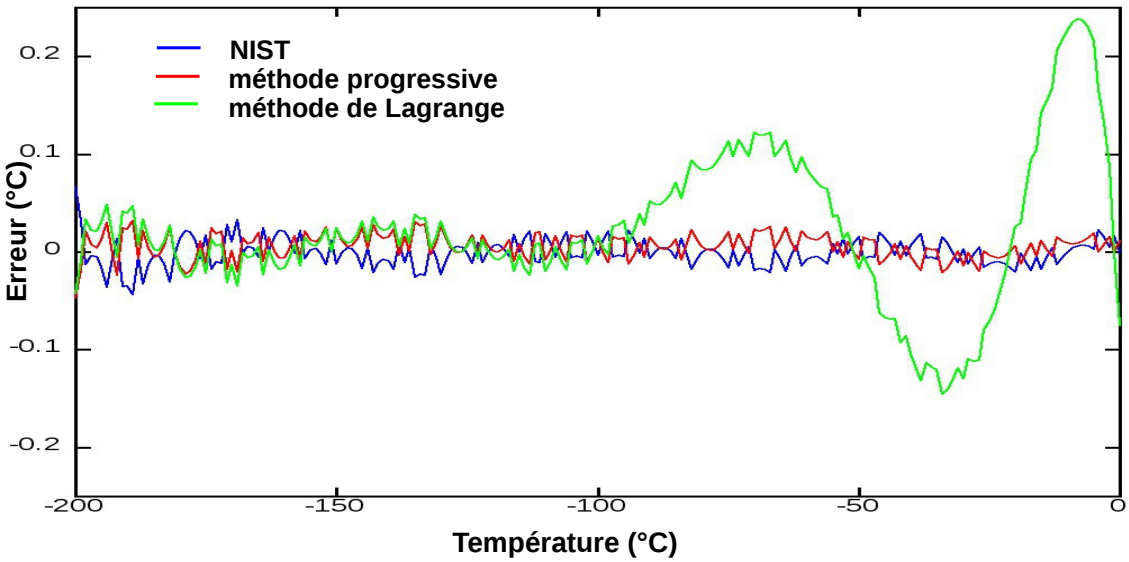


Figure 3.6.Erreurs de calibration de polynômes de lagrange et progressive Dans la plage de température [-200 ,0]°C

Les erreurs RMS sont les suivantes:

$$RMS_{prog} = 0.0139$$
$$RMS_{lagr} = 0.0757$$

3.3. Méthode d'étalonnage par réseaux de neurones artificiels:

Dans cette section, nous allons essayer de compenser la caractéristique du thermocouple K sur la gamme [-200 1373]°C.

Les réseaux de neurones utilisés sont :

1 - Un réseau de neurones entièrement connectés en cascade (FCN).

2 - Un réseau multi-couches (MLP).

3 - Un réseau multi-couches avec des connections entre les entrées et les sorties (BMLP).

Les fonctions d'activation sigmoïde que nous allons utiliser sont les suivantes:

1 - Tanh (Tangente Hyperbolique)

2 - la fonction d'Elliott.

3 - Ihalainen & Vehviläinen comme-tanh fonction sigmoïde.

Toutes ces fonctions sont bipolaires (entre -1 et 1).

Les algorithmes d'apprentissage que vont être utilisés sont :

1 - Rétro-propagation avec élan (momentum).

2 - Rprop.

3 - Neurone par neurone (NBN).

L'ensemble de données contient 51 points dispersés dans la gamme [-200 1373]°C. Les données ont été normalisées et l'apprentissage est limité par 30000 itérations.

3.3.1 Rétropropagation avec élan (momentum)

Pour cet algorithme, plusieurs configurations de réseaux de neurones parmi les discutées auparavant ont été utilisées afin de calibrer le thermocouple K mais les résultats sont loin d'être satisfaisants. Après plusieurs tentatives, les meilleurs résultats sont obtenus avec un réseau MLP avec une configuration 8-3-1 (12 neurones, 2 couches cachées) traîné avec la fonction d'Elliott. Pour les deux autres fonctions d'activation (Tanh et Ihalainen & Vehviläinen comme-tanh fonction sigmoïde), les erreurs d'apprentissage et de généralisation sont plus mauvaises. Les erreurs moyennes quadratiques pendant les deux phases d'apprentissage et de généralisation sont données dans le tableau suivant :

Fonction d'activation	RMS pour l'algorithme de Rétropropagation avec élan (momentum)	
	MLP (8-3-1)	
	Apprentissage	Généralisation
Tanh	24.212	23.869
Elliott	7.789	7.189
Iva et Iha	11.362	11.642

Table 3.1 Les erreurs RMS pendant les deux phases d'apprentissage et de généralisation. pour l'algorithme de Rétropropagation avec élan (momentum).

Les figures 3.7 et 3.8 montrent le réseau MLP utilisé et le graphe d'erreur de généralisation.

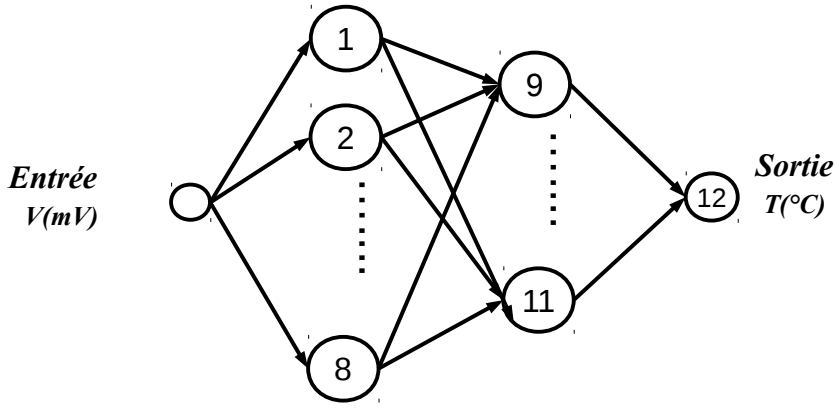


Figure 3.7. réseau de neurones MLP avec 12 neurones

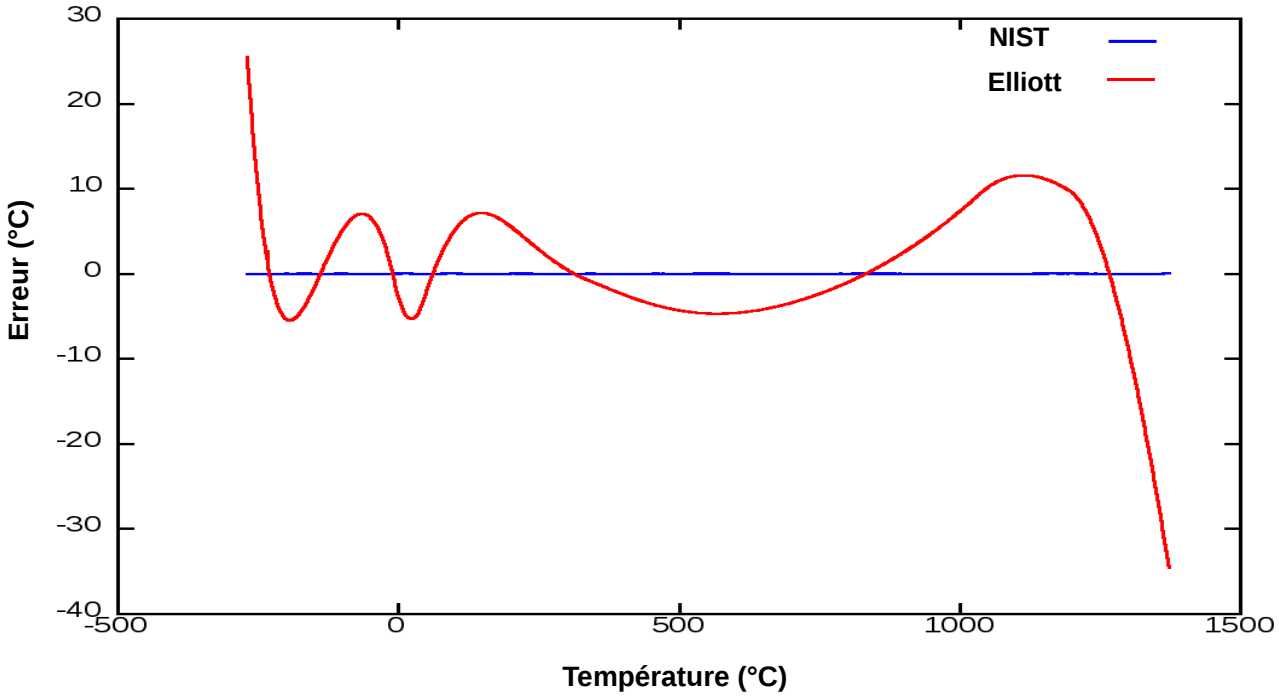


Figure 3.8 Erreur de linéarisation de réseau MLP avec 12 neurones Pour l'algorithme rétropropagation avec élan (momentum)

3.3.2 Rprop

Pour l'algorithme Rprop, les résultats obtenus sont mieux que celles de l'algorithme Rétro-propagation avec élan (momentum). En ce qui suit, on va citer les résultats pour les trois configurations de réseaux de neurones pour les trois fonctions d'activation.

3.2.2.1 Le réseau FCN

Après plusieurs essais, les meilleurs résultats sont obtenus pour un réseau de neurones entièrement connecté en cascade avec 6 neurones. Les RMS résultantes des deux phases d'apprentissage et de généralisation sont :

Fontions d'activations	RMS	
	Apprentissage	Généralisation
Tanh	0.644	0.659
Elliott	1.615	1.543
Iva et Iha	1.030	0.930

Table 3.2 Les erreurs RMS pendant les deux phases d'apprentissage et de généralisation. pour l'algorithme de Rprop pour le réseau FCN.

Les figures 3.9 et 3.10 montrent le réseau FCN utilisé et le graphe d'erreur de généralisation.

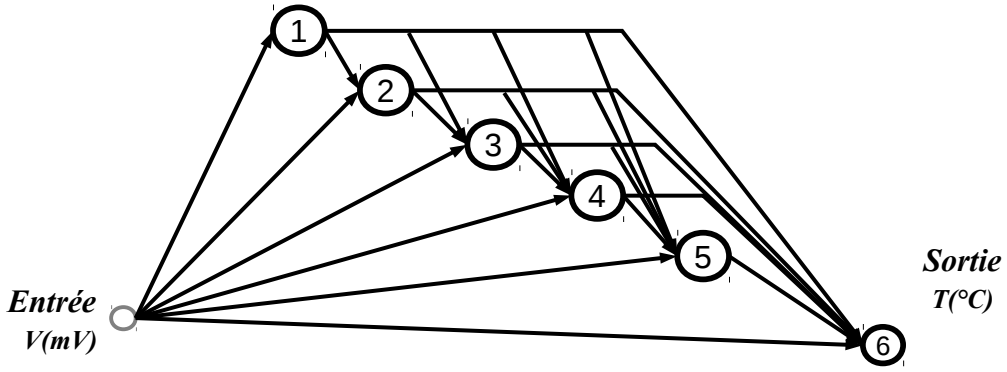


Figure 3.9. réseau de neurones FCN avec 6 neurones

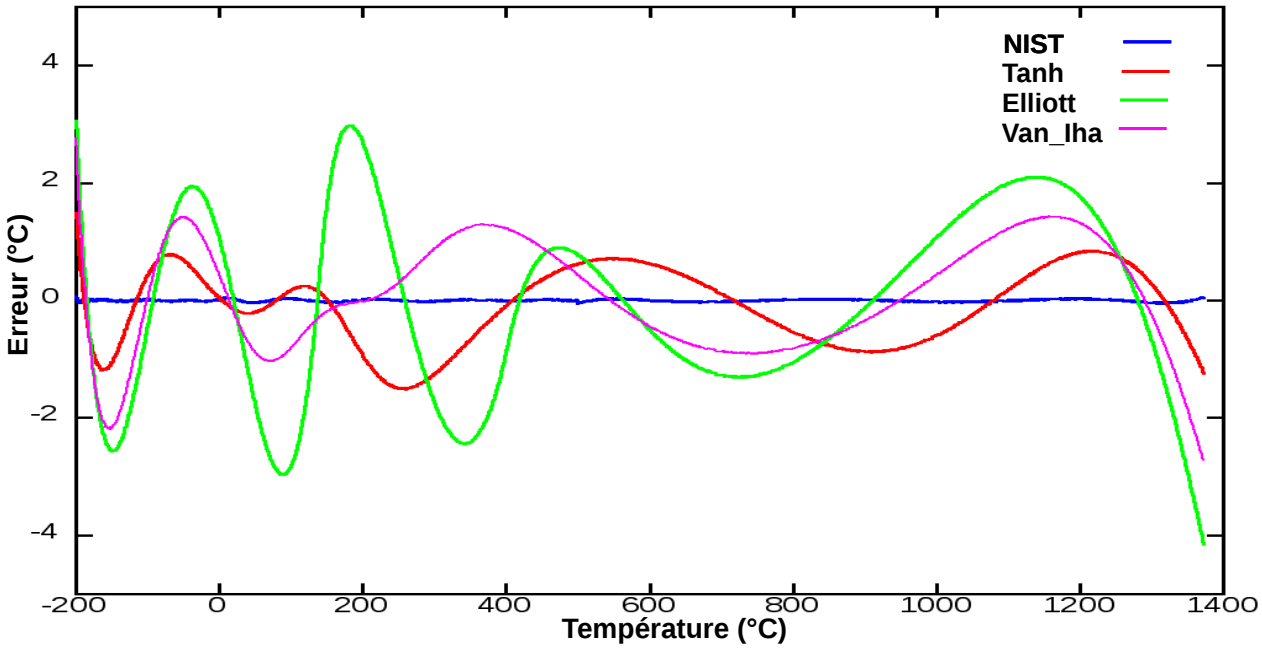


Figure 3.10 Erreur de linéarisation de réseau FCN avec 6 neurones Pour l'algorithme Rprop

D'après le tableau et le graphe, l'algorithme Rprop a des meilleurs résultats que la méthode de rétro-propagation avec élan. Dans ce cas, la fonction Tanh a atteint les erreurs les plus bases pendant les deux phases d'apprentissage et de généralisation. Les résultats sont beaucoup mieux et ils sont acceptables.

3.2.2.2 Réseau MLP

Pour les réseaux multi-couches, on a testé deux réseaux avec les configurations suivantes :

1- Un réseau MLP avec la configuration 4-3-1 (8 neurones, 2 couches cachées)

2- Un réseau MLP avec la configuration 8-3-1 (12 neurones, 2 couches cachées)

Les RMS résultantes des deux phases d'apprentissage et de généralisation sont décrits dans le tableau suivant :

Fonctions d'activation	RMS			
	MLP (4-3-1)		MLP (8-3-1)	
	Apprentissage	Généralisation	Apprentissage	Généralisation
Tanh	5.855	4.910	1.521	1.413
Elliott	0.737	0.919	1.220	1.639
Iva et Iha	2.459	2.252	1.232	1.375

Table 3.3 Les erreurs RMS pendant les deux phases d'apprentissage et de généralisation. pour l'algorithme de Rprop pour le réseau MLP .

Les figures 3.11, 3.12 et 3.13 montrent les réseaux MLP utilisés et les graphes d'erreur de généralisation respectivement.

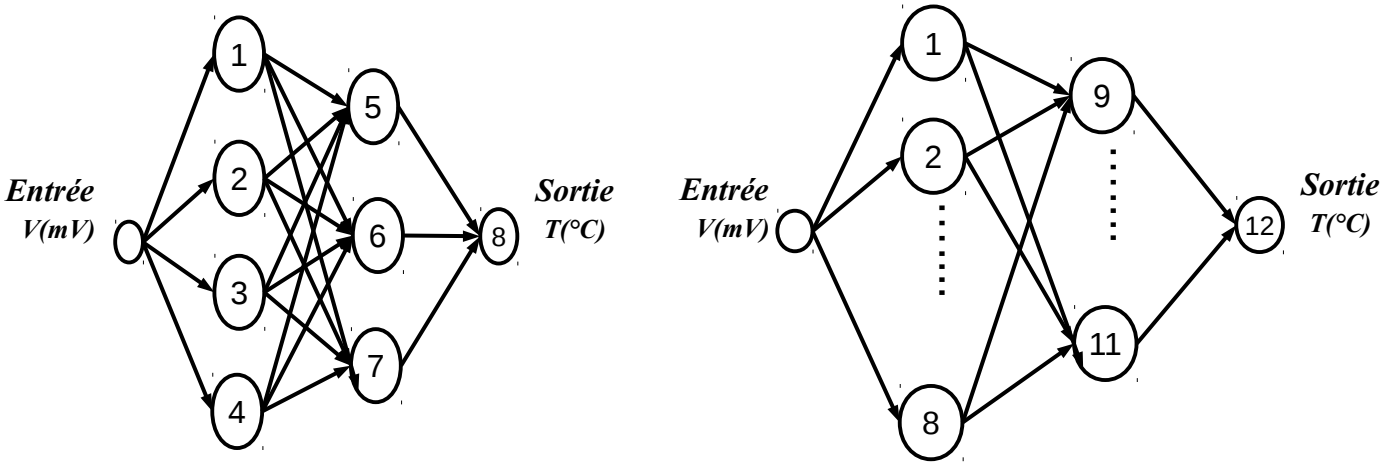


Figure 3.11. (a) réseau de neurones MLP avec 8 neurones
(b) réseau de neurones MLP avec 12neurones

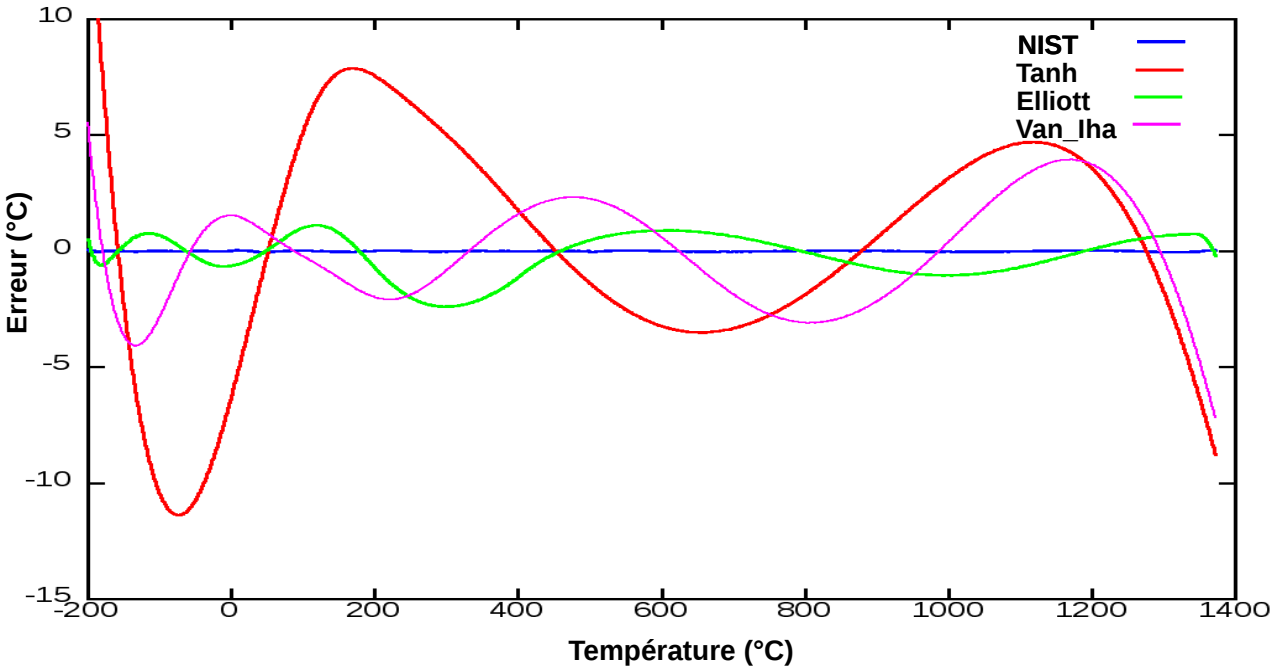


Figure 3.12 Erreur de linéarisation de réseau MLP avec 8 neurones Pour l'algorithme Rprop

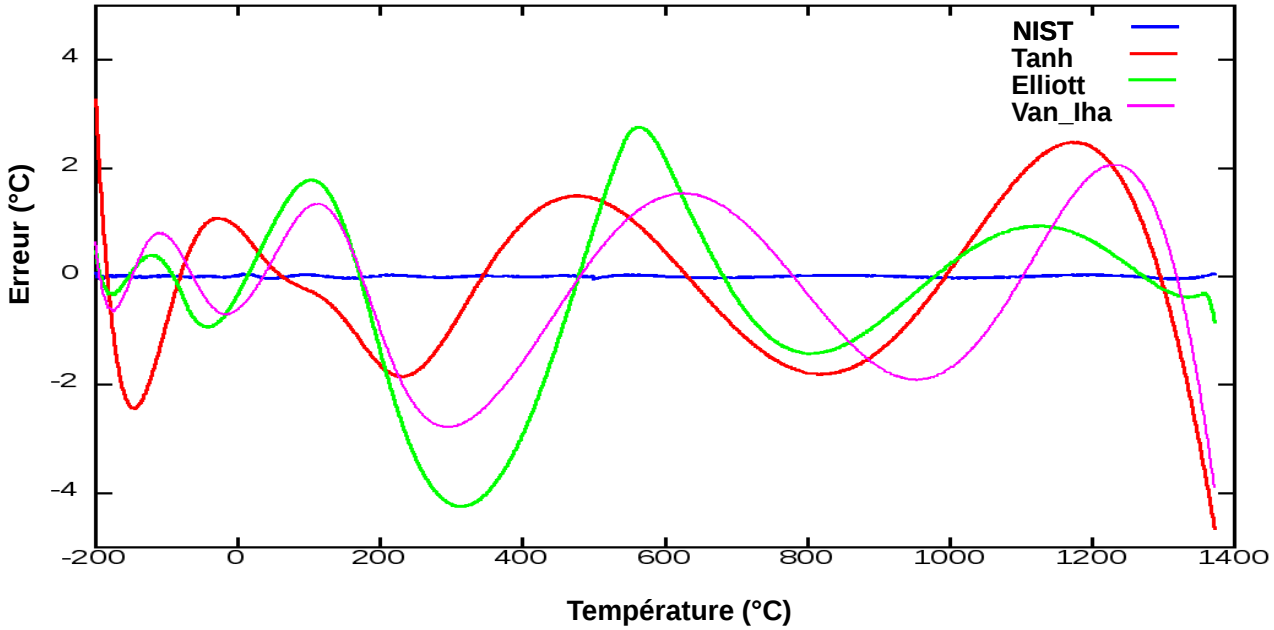


Figure 3.13 Erreur de linéarisation de réseau MLP avec 12 neurones Pour l'algorithme Rprop

D'après la table 3.3 et les graphes 3.12 et 3.13, le réseau MLP (4-3-1) a atteint les erreurs les plus bases pendant les deux phases d'apprentissage et de généralisation pour la fonction d'activation d'Elliott. On voit clairement que la fonction d'Elliott a des capacités aussi bien que la fonction Tanh pour les deux phases d'apprentissage et de généralisation

3.2.2.3 Réseau BMLP

Pour le réseau BMLP, après plusieurs essais, les meilleurs résultats obtenus sont avec un réseau avec la configuration 6-3-1 (10 neurones, 2 couches cachées). Les RMS résultantes des deux phases d'apprentissage et de généralisation sont données dans le tableau suivant :

Fontions d'activations	RMS	
	Apprentissage	Généralisation
Tanh	2.843	3.042
Elliott	3.544	3.356
Iva et Iha	1.156	1.056

Table 3.4 Les erreurs RMS pendant les deux phases d'apprentissage et de généralisation. pour l'algorithme de Rprop pour le réseau BMLP.

Les figures 3.14 et 3.15 montrent le réseau BMLP utilisé et le graphe d'erreur de généralisation respectivement.

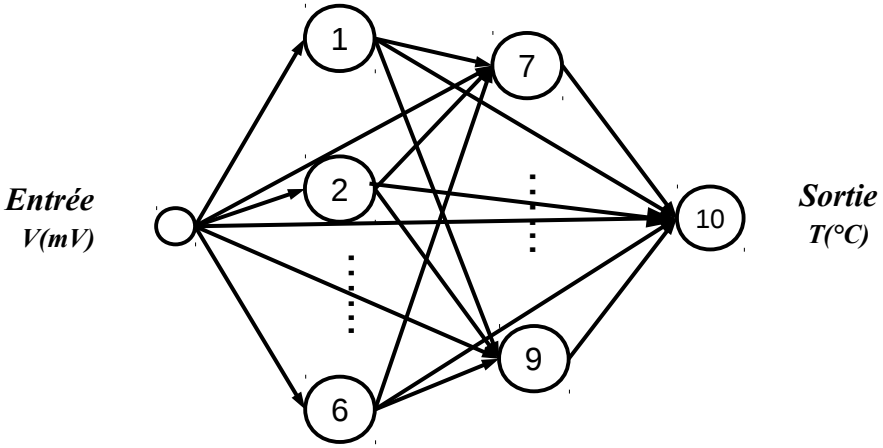


Figure 3.14 réseau de neurones BMLP avec 10 neurones

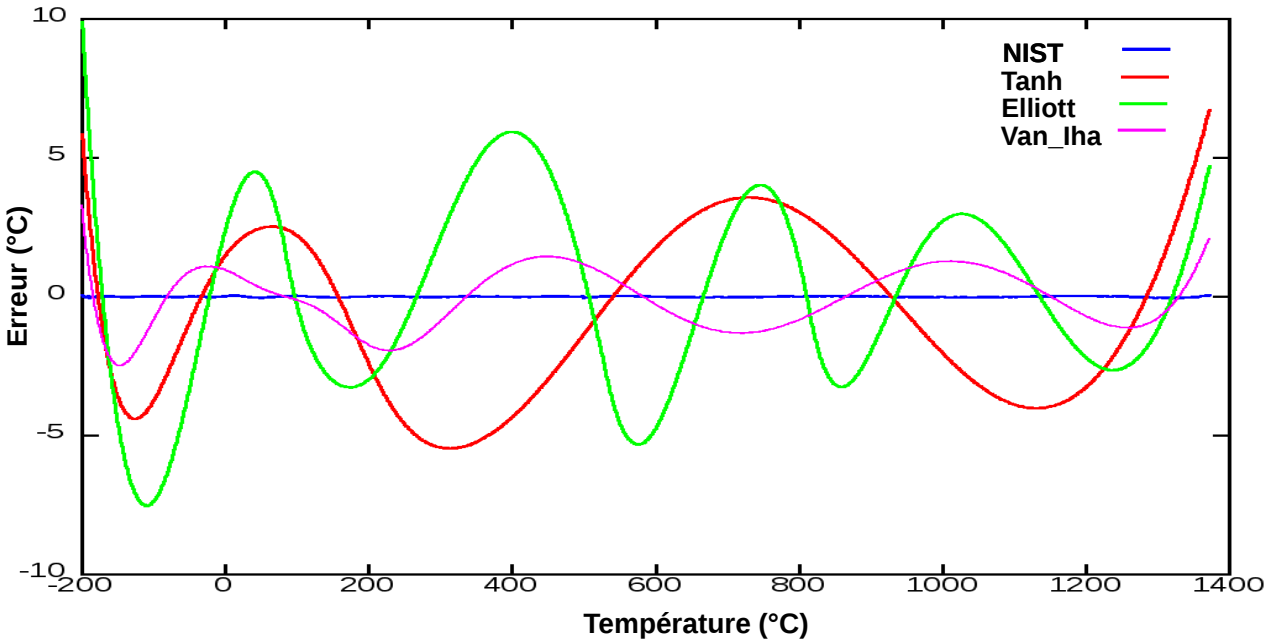


Figure 3.15 Erreur de linéarisation de réseau BMLP avec 10 neurones Pour l'algorithme Rprop

Dans ce cas, les meilleurs résultats sont obtenus pour la fonction d'activation proposée par Ihalainen & Vehviläinen pendant les deux phases d'apprentissage et de généralisation.

D'après les essais précédents, on voit que l'algorithme Rprop donne des résultats beaucoup mieux que ceux de l'algorithme de rétropropagation avec élan (momentum), mais malgré ça, on n'a pas pu atteindre une erreur au voisinage de celle de NIST. Les fonctions d'activations d'Elliott et celle proposée par Ihalainen & Vehviläinen sont aussi bien que la fonction Tanh. Parmi les trois réseaux de neurones utilisés avec l'algorithme Rprop, le réseau FCN avec 6 neurones a les meilleures performances.

Dans ce qui suit, on va entamé l'apprentissage avec l'algorithme NBN afin d'obtenir des meilleurs résultats que ceux obtenus jusqu'à maintenant. Il faut noter que les algorithmes utilisés précédemment sont des algorithmes de premier ordre, ils n'utilisent que l'information du gradient pour la mise à jour des poids, alors que l'algorithme NBN est un algorithme de deuxième ordre qui utilise l'information du gradient ainsi que la matrice hessienne pour l'adaptation des poids.

3.3.3 NBN

Dans ce cas, les réseaux de neurones vont être traînés par l'algorithme NBN afin d'achever une erreur inférieure au celle donnée par NIST.

3.2.3.1. réseau FCN

Avec un réseau FCN avec seulement 5 neurones, on a pu achever une erreur inférieure à celle de NIST. Le réseau a été traîné pour 15000 itérations qui est la moitié pour les méthodes précédentes. Les RMS résultantes des deux phases d'apprentissage et de généralisation sont données dans le tableau suivant :

Fontions d'activations	RMS	
	Apprentissage	Généralisation
Tanh	0.016	0.016
Elliott	0.071	0.091
Iva et Iha	0.104	0.132

Table 3.5 Les erreurs RMS pendant les deux phases d'apprentissage et de généralisation. pour l'algorithme NBN pour le réseau FCN.

Les figures 3.16 et 3.17 montrent le réseau FCN utilisé et le graphe d'erreur de généralisation respectivement.

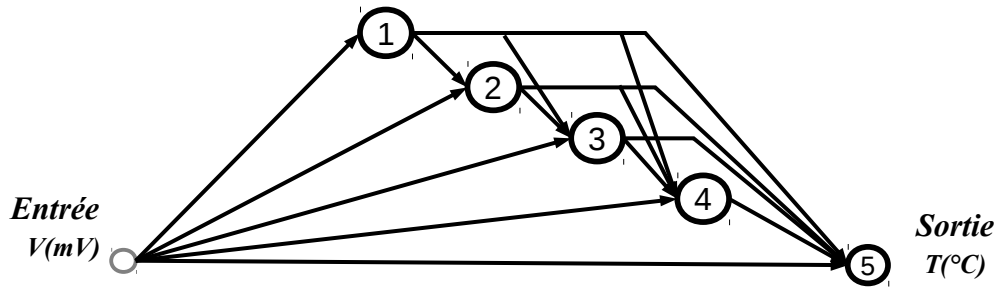


Figure 3.16 réseau de neurones FCN avec 5 neurones

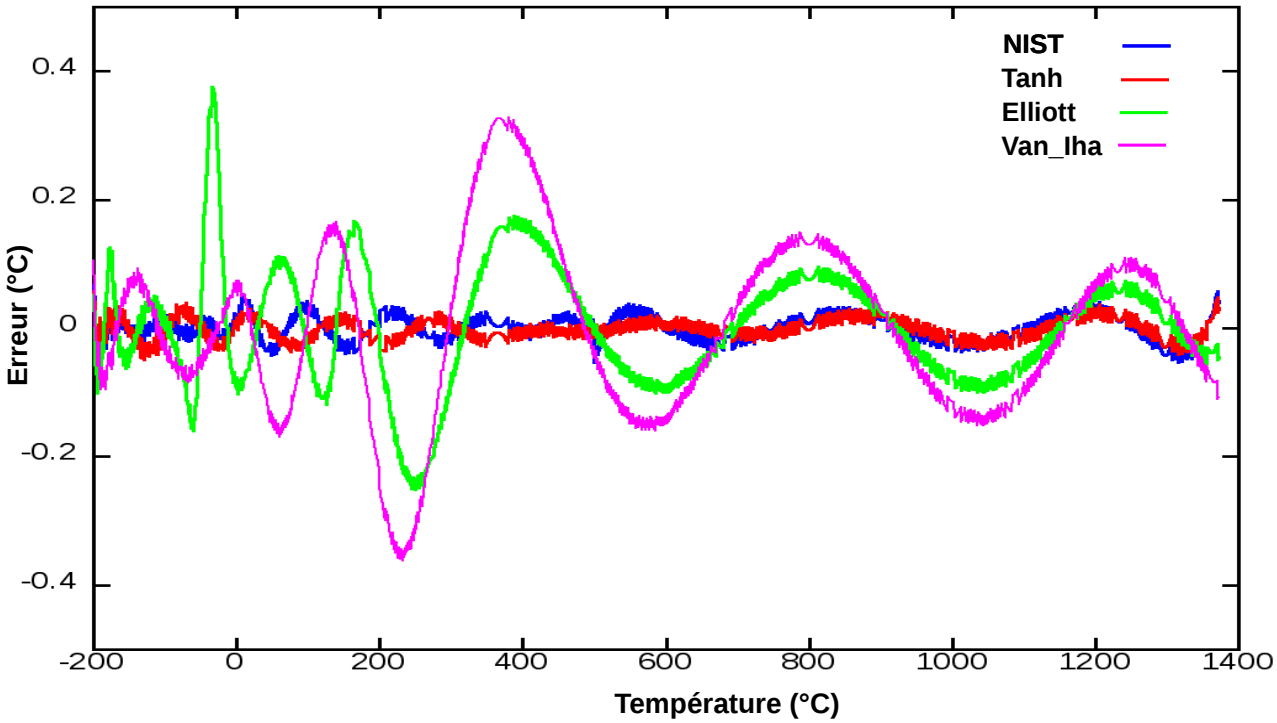


Figure 3.17. Erreur de linéarisation de réseau ACN avec 5 neurones. Pour l'algorithme NBN

Parmi ces trois erreurs moyennes quadratiques, seule la fonction Tanh a atteint une erreur inférieure que celle de NIST qui est : 0.019559.

3.2.3.2. réseau MLP (Multilayer Perceptrons):

Après plusieurs tentatives, des résultats satisfaisantes sont obtenues avec un réseau ayant la configuration 5-4-1 (10 neurones, 2 couches cachées). Le réseau a été traîné pour 15000 itérations qui est la moitié pour les méthodes précédentes. Les RMS résultantes des deux phases d'apprentissage et de généralisation sont données dans le tableau suivant :

Fontions d'activations	RMS	
	Apprentissage	Généralisation
Tanh	0.006	0.009
Elliott	0.017	0.064
Iva et Iha	0.006	0.011

Table 3.6 Les erreurs RMS pendant les deux phases d'apprentissage et de généralisation. pour l'algorithme NBN pour le réseau MLP.

Les figures 3.18 et 3.19 montrent le réseau MLP utilisé et le graphe d'erreur de généralisation respectivement.

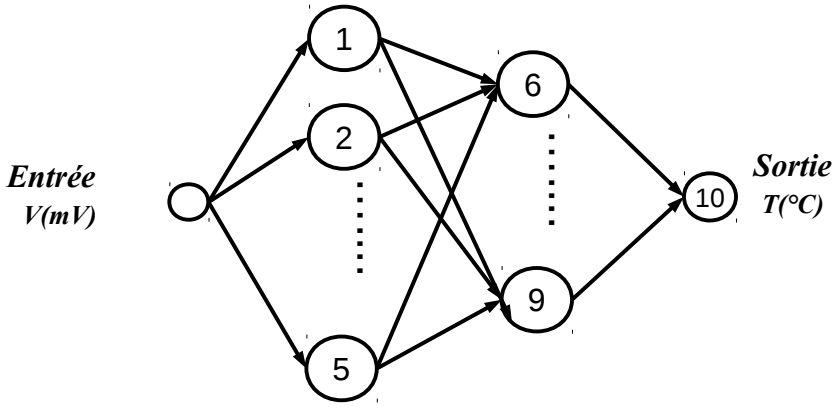


Figure 3.18. réseau de neurones MLP avec 10 neurones

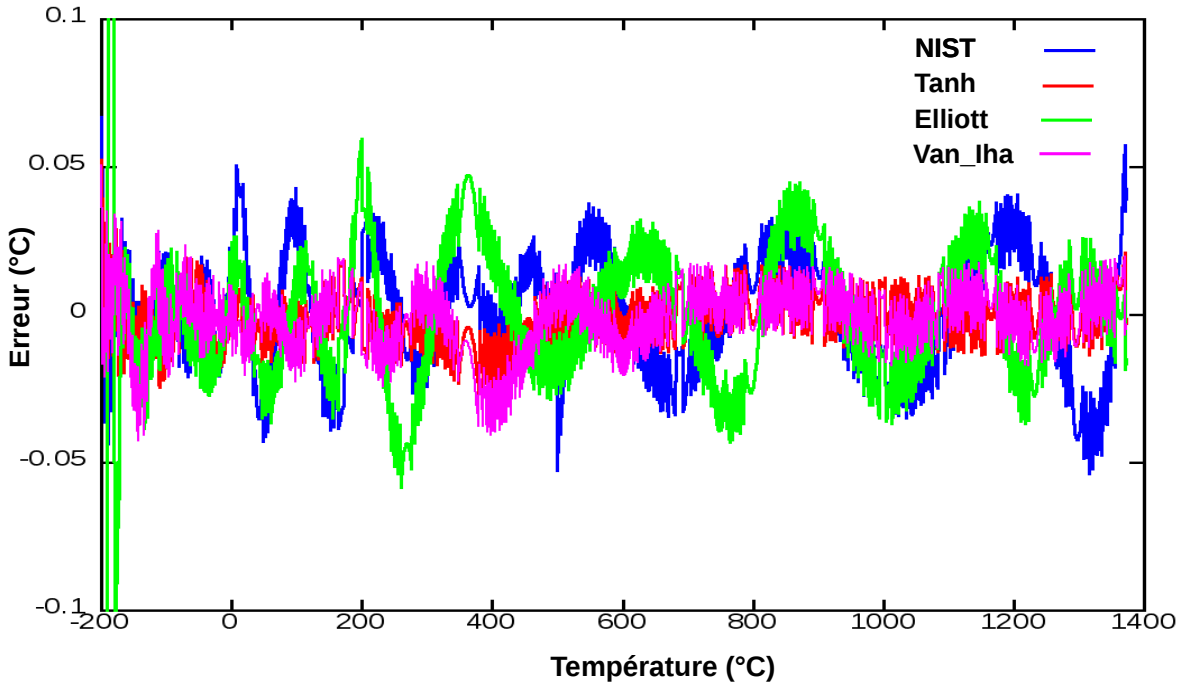


Figure 3.19 Erreur de linéarisation de réseau MLP avec 10 neurones Pour l'algorithme NBN

3.2.2.3. réseau BMLP (Bridge Multilayer Perceptrons):

Après plusieurs tentatives, des résultats satisfaisantes sont obtenues avec un réseau ayant la configuration 3-2-1 (6 neurones, 2 couches cachées). Le réseau a été traîné pour 15000 itérations. Les RMS résultantes des deux phases d'apprentissage et de généralisation sont données dans le tableau suivant :

Fontions d'activations	RMS	
	Apprentissage	Généralisation
Tanh	0.023	0.028
Elliott	0.022	0.035
Iva et Iha	0.014	0.016

Table 3.7 Les erreurs RMS pendant les deux phases d'apprentissage et de généralisation. pour l'algorithme NBN pour le réseau BMLP.

Les figures 3.20 et 3.21 montrent le réseau BMLP utilisé et le graphe d'erreur de généralisation respectivement.

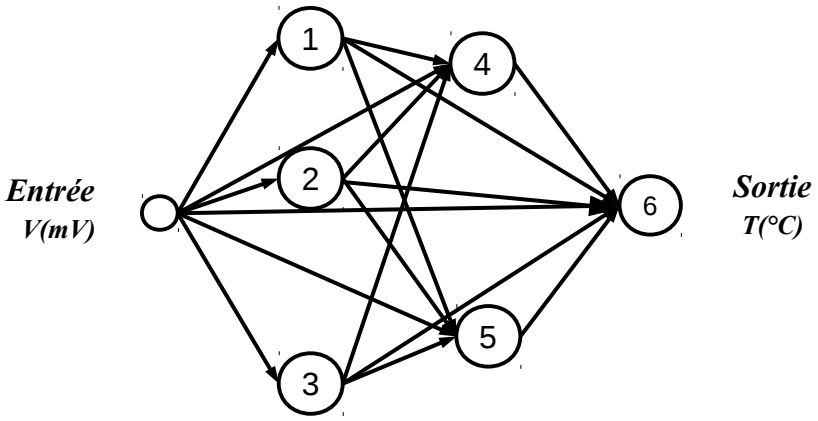


Figure 3.20 réseau de neurones BMLP avec 6 neurones

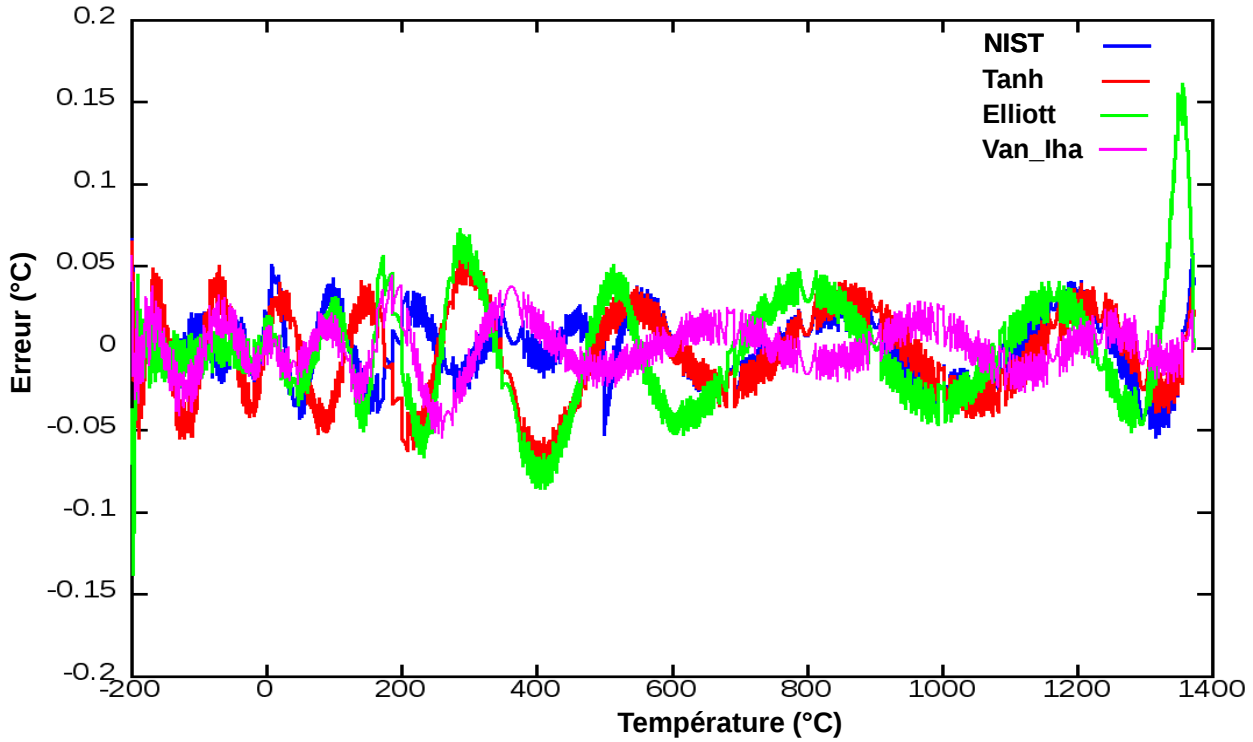


Figure 3.21. Erreur de linéarisation de réseau BMLP avec 6 neurones. Pour l'algorithme NBN

On peut voir que la fonction d'Ihalainen & Vehviläinen aille le meilleur résultat que ceux de fonctions Tanh et d'Elliott. Ainsi pour la phase de généralisation, les résultats obtenus par la fonction d'Ihalainen & Vehviläinen sont mieux que ceux de la fonction Tanh, et les résultats obtenus par cette dernière fonction sont mieux que ceux de la fonction d'Elliott malgré que l'erreur d'apprentissage de la fonction d'Elliott est inférieur que celui de la fonction Tanh.

Les résultats précédents sont résumés dans le tableau suivant :

Réseau de neurones	Erreur de généralisation		
	RMS		
	Tanh	**Elliott**	**Vah et Iha**
ACN	0.0164	0.0949	0.0354
MLP	0.2175	0.1422	0.2356
BMLP	0.0277	0.0354	0.0161

Résumé :

D'après les résultats obtenus, on peut voir que le réseau FCN requise un nombre minimale de neurones pour avoir des résultats similaires aux celles des réseaux multi-couches (MLP) et BMLP. Le réseau BMLP est mieux que le réseau MLP. Ceci peut être expliqué par l'effet que les réseaux FCN et BMLP aillent plus de nombres de connexions (poids) que les réseau MLP pour le même nombre de neurones. Ainsi les connexions entre des neurones non adjacents que les réseaux FCN et BMLP aillent peut affecter les performances d'apprentissage et de généralisation de réseau.

Pour les algorithmes d'apprentissage, les résultats de l'algorithme de rétro-propagation avec élan étaient loin d'être acceptables alors qu'on a pu obtenir des résultats acceptables pour l'algorithme Rprop. L'algorithme Rpop semble d'être parmi les meilleurs d'algorithme du premier ordre. Les résultats obtenues par cet algorithme peut être améliorer par augmenter le nombre de neurones de réseau d'apprentissage, mais ceci va augmenter le temps d'exécution pendant l'implémentation du calcul en temps réel. Les résultats d'algorithme NBN étaient très satisfaisantes. Cet algorithme requit le nombre minimal de neurones et du temps d'apprentissage. Cependant, cet algorithme nécessite le calcul de la matrice hessienne (calcul de l'inverse d'une matrice), ce qui rendre cet algorithme inadéquat afin d'implémenter l'apprentissage avec cet algorithme en temps réel sur un microcontrôlleur à faible coût. L'algorithme Rprop semble d'être le meilleur choix pour cette tâche.

Les fonctions d'activations présentées par D.L.Elliott et P.V. Vehviläinen et H.A.TIhalainen sont aussi efficaces que la fonction tangent hyperbolique (Tanh) pour l'apprentissage de réseaux de neurones. Cependant, ces fonctions sont avantageuses que la fonction Tanh, elles sont simples à calculer car elles ne nécessitent pas la puissance de calculs en virgule flottante que la fonction Tanh la demande, ainsi ces fonctions sont efficaces en terme du temps d'exécution.

3.4 Tests expérimentaux:

3.4.1 Conditionnement du capteur :

Un capteur thermocouple nécessite un circuit de conditionnement avant d'être calibrer. La figure 3.22 montre un circuit de conditionnement pour une thermocouple type K.

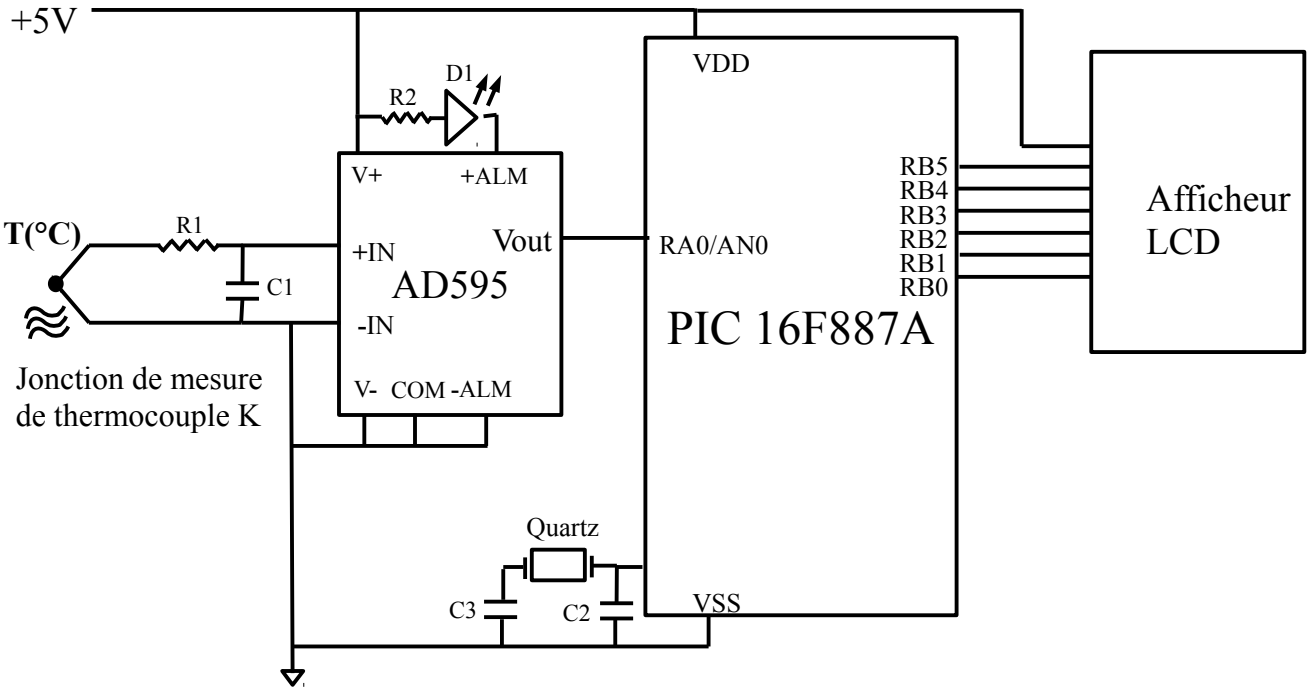


Figure 3.22 Circuit de conditionnement de thermocouple type K

Une thermocouple produit à sa sortie une tension proportionnelle à la température de jonction de mesure à des niveaux de mV pendant que la température de soudure froide est constante. Afin de faire des mesures précises, la température de soudure froide (Cold junction) doit être connue et compensée ainsi que la tension de sortie doit être amplifiée. Pour cela, on a utilisé l'AD595 qui est un amplificateur monolithique de thermocouple avec compensation de soudure froide.

L'ADC595 délivre une tension proportionnelle à la température mesurée donnée par l'équation suivante [AN-396]:

$$ADC595\,(mV)_{sortie} = (Voltage\,Type\,K + 11\,\mu V) \times 247.3$$

La diode D1 est utilisée comme une alarme. Si un ou les deux des fils de thermocouple sont interrompus, la pin +ALM sera à l'état bas et la diode s'allume.

3.4.2 Implémentation sur le microcontrôleur:

Dans cette section, nous allons implémenter les résultats obtenus à partir de simulations précédentes dans un microcontrôleur PIC 16F877A, et comparer les résultats d'implémentation avec celles de simulation.

Les contraintes imposées par le microcontrôleur sont :

1- La résolution : Le microcontrôleur PIC16F877A a un convertisseur ADC de 10 bits avec une gamme d'entrée de 0-5V, ce qui impose une résolution de 4,887 mV.

2- Représentation de données : les nombres à virgule flottante sont représentés dans le microcontrôleur sur 32 bits, tandis que sur le PC, ils sont représentés sur 64 bits.

Ces facteurs vont générer des erreurs supplémentaires aux erreurs de calibration, ce que peut rendre l'erreur totale inacceptable.

Pour minimiser l'erreur de résolution, on a réduit la plage de température entre [0 100]°C. Le signal de sortie de thermocouple est amplifiée de telle sorte que :

- à 0 °C, le signal d'entrée du PIC est 0V.
- À 100 0C, le signal d'entrée est 5V.

3.4.2.1 Méthode polynomiale Progressive :

Les coefficients obtenus de simulation sont implémentés sur le PIC16F877A. La Figure 3.23 montre l'erreur de calibration sur PC et l'erreur de l'implémentation sur le PIC.

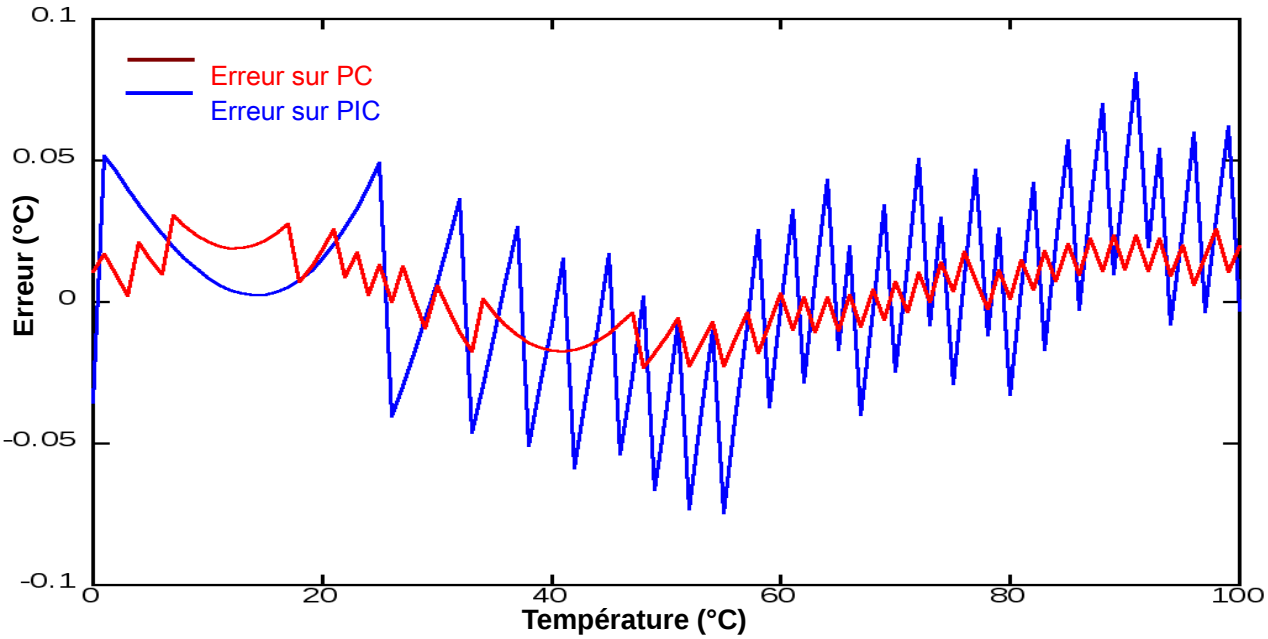


Figure 3.23 Les erreurs de calibration de la Méthode polynomiale Progressive sur PC et sur PIC.

On peut voir que l'erreur a clairement grandit de trois fois à peu prés. Cette erreur est dû considérablement aux limitations du convertisseur ADC c-à-d de l'effet de quantification que peut être identifier à partir de la forme de la courbe d'erreur.

Le temps d'exécution pour cette méthode pour une fréquence d'horloge de 20Mhz est de : 2.89 ms.

3.4.2.2 Réseau de neurones artificielles :

Pour les réseaux de neurones, on va implémenter des réseaux avec les trois fonctions d'activations utilisées pour la simulations, et on va comparer leurs temps d'exécution.

3.4.2.2.1 Réseau ACN :

La figure 3.24 montre l'erreur de calibration sur PC et l'erreur de l'implémentation sur le PIC pour un réseau ACN avec 5 neurones. Pour ce réseau on a utilisé la fonction Tanh comme une fonction d'activation.

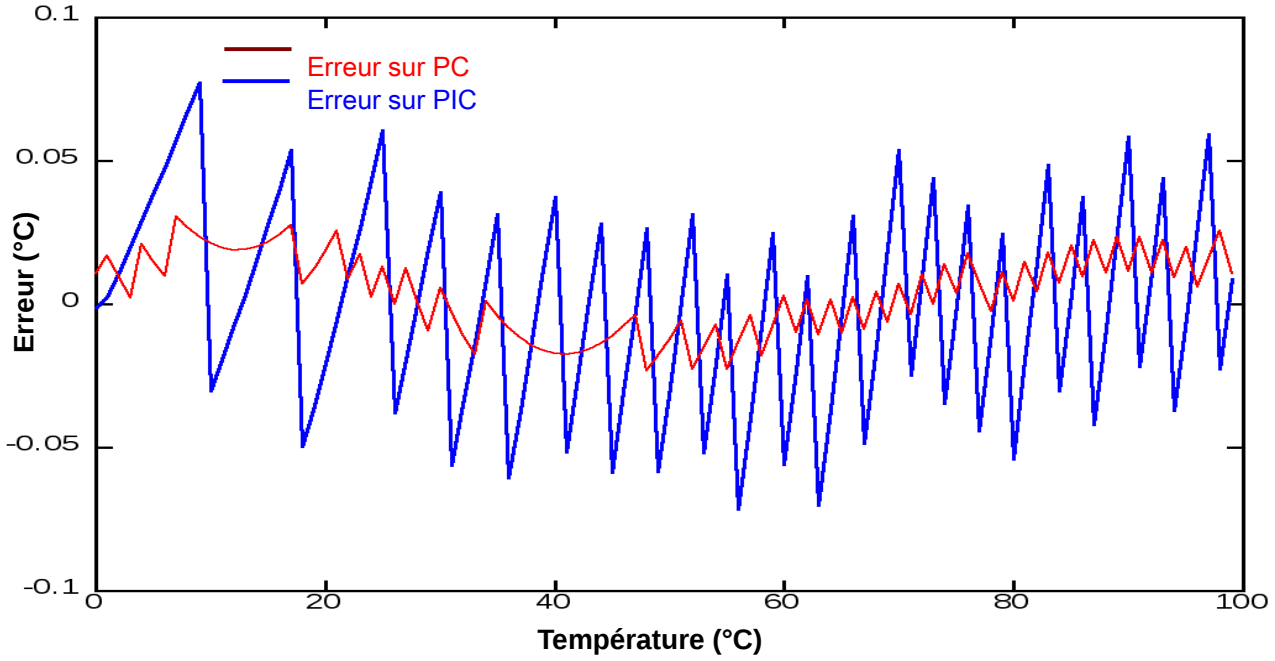


Figure 3.24 Les erreurs de calibration de réseau ACN sur PC et sur PIC.

Comme le cas de la méthode polynomiale progressive, l'erreur a grandit de trois fois approximativement.

Le temps d'exécution pour cette méthode pour une fréquence d'horloge de 20Mhz est de : 9 ms.

L'augmentation du temps d'exécution est dû considérablement aux calcules de la fonction Tanh. Le calcul d'une seule fonction Tanh nécessite approximativement 1.8 ms.

3.4.2.2.1 Réseau BMLP :

La figure 3.25 montre l'erreur de calibration sur PC et l'erreur de l'implémentation sur le PIC pour un réseau BMLP avec 6 neurones. Pour ce réseau on a utilisé la fonction proposée par P.V. Vehviläinen et H.A.TIhalainen comme une fonction d'activation.

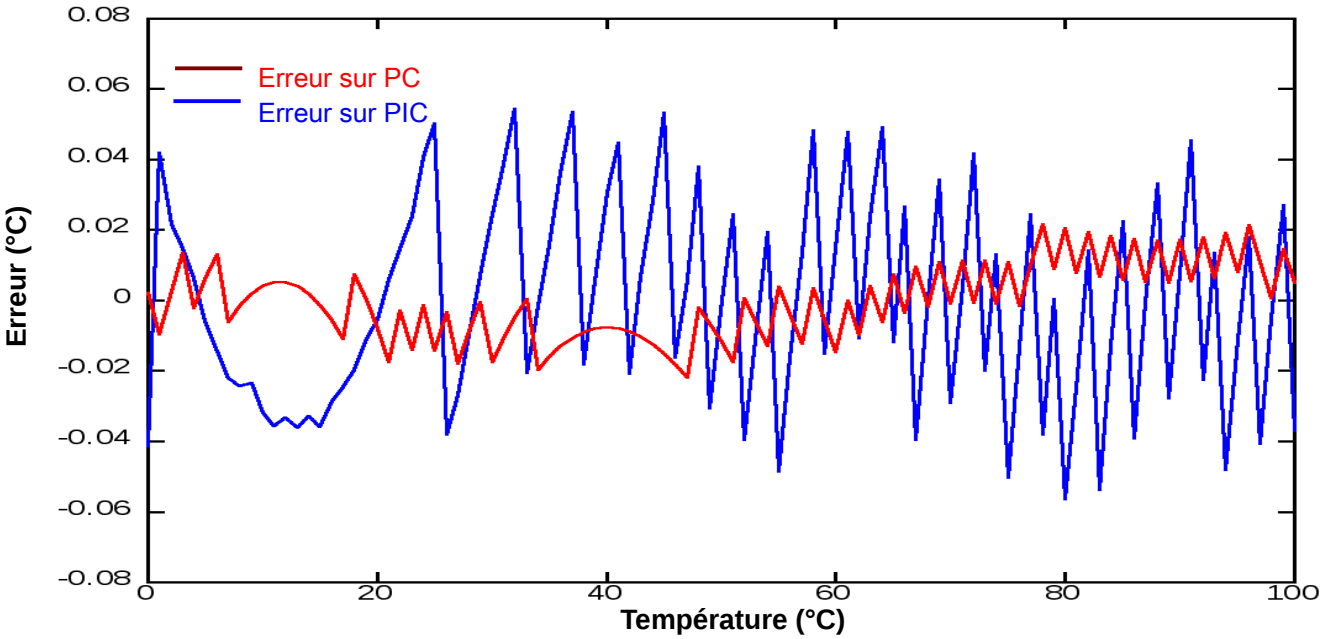


Figure 3.25 Les erreurs de calibration de réseau BMLP sur PC et sur PIC.

Le temps d'exécution pour cette méthode pour une fréquence d'horloge de 20Mhz est de : 5.1ms.
Le calcul d'une seule fonction d'activation nécessite approximativement 0.6 ms qui est le tiers du temps nécessaire pour calculer la fonction Tanh.

La figure 3.26 montre l'erreur de calibration sur PC et l'erreur de l'implémentation sur le PIC pour le réseau BMLP avec 6 neurones. Pour ce réseau on a utilisé la fonction proposée par D.L. Elliott comme une fonction d'activation.
Le temps d'exécution pour cette méthode pour une fréquence d'horloge de 20Mhz est de : 3.8ms.
Le calcul d'une seule fonction d'activation nécessité approximativement 0.35 ms qui est le cinquième du temps nécessaire pour calculer la fonction Tanh.

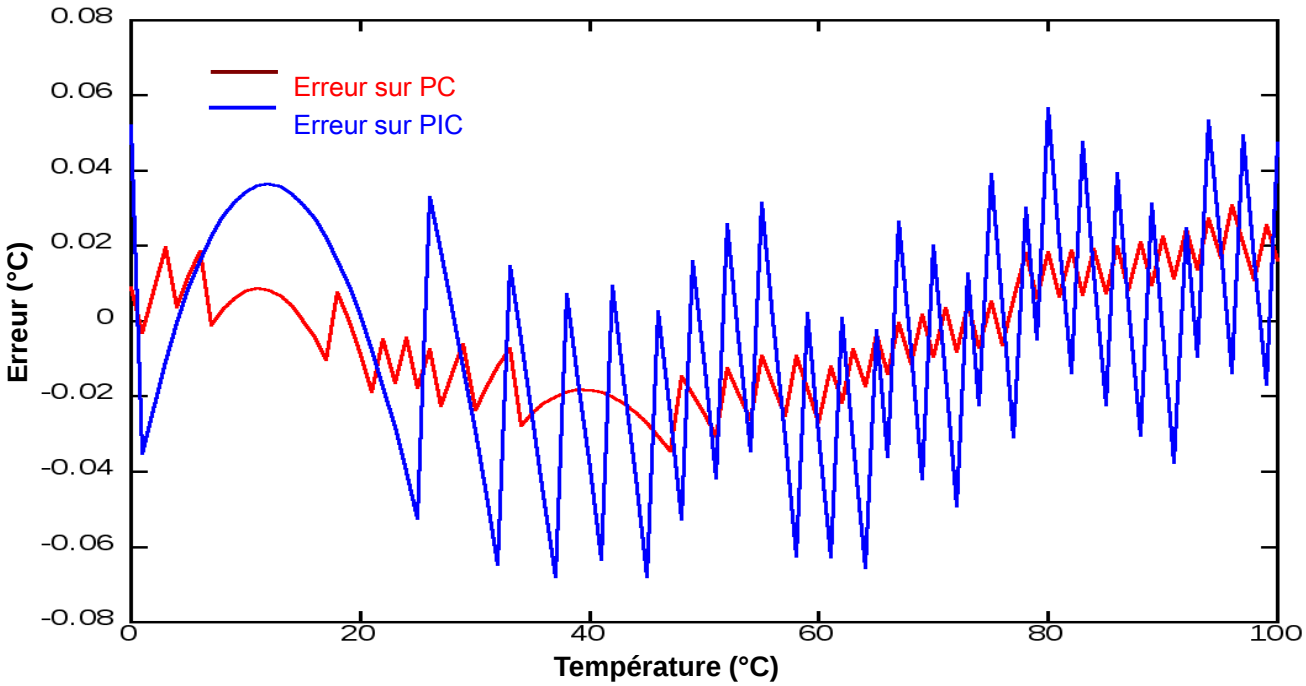


Figure 3.26 Les erreurs de calibration de réseau BMLP sur PC et sur PIC.

La fonction proposée par D.L. Elliott nécessite le minimum temps d'exécution parmi les trois fonctions d'activation utilisées. Cependant, on a vu pendant les simulations qu'elle est moins efficace que la fonction Tanh ou celle proposée par P.V. Vehviläinen et H.A.TIhalainen. Cette dernière semble d'être le meilleur choix parmi ces trois fonctions car elle est simple a calculer et assez semblable de la fonction Tanh et elle a donnée des résultats aussi efficace que la fonction Tanh pendant les simulations.

Conclusion :

La calibration du capteur est une tâche essentielle afin d'avoir des mesures précises. Des différentes techniques ont été proposées dans la littérature pour la calibration des capteurs comprenant les méthodes analogiques et numériques. Les méthodes analogiques, malgré leurs rapidités d'exécution, il n'est pas toujours possible de trouver un circuit analogique qui intègre pleinement la fonction inverse du capteur que nous souhaitons corriger, de sorte que la compensation, lorsqu'elle est réalisée, n'atteint pas la résolution désirée. En revanche, les méthodes numériques d'étalonnage, en dépit de consommer une plus grande surface lorsqu'elles sont appliquées, soutiennent une grande variation dans le type de fonction de transfert. En outre, les circuits numériques sont constamment développés et peuvent offrir un plus large éventail d'avantages dans le contexte de capteurs intelligents. Des tâches telles que l'auto-calibrage et auto-test sont en général plus facilement mises en œuvre, en réduisant les coûts de production et de tests, intégration et automatisation des installations, d'autre part, réduisant le coût de conception, les méthodes numériques font une excellente solution. Ainsi, le concept de capteur intelligent intégrant un microprocesseur favorise l'utilisation des méthodes numériques qui sont plus efficaces et plus flexibles.

Dans ce travail, on a abordé deux méthodes numériques : la première est la méthode polynomiale progressive, une méthode proposée essentiellement pour le calibrage des capteurs intelligents. La calibration d'une thermocouple type K par cette méthode et la méthode polynomiale de Lagrange montre l'efficacité de la méthode progressive par rapport à la méthode polynomiale de Lagrange en terme de nécessité de points de calibrage et d'erreur de calibration. La deuxième méthode est la calibration par réseau de neurones artificiels, un outil très puissant pour la modélisation des systèmes non linéaires. Les résultats obtenus sont satisfaisants et ils sont implémentés dans un microcontrôlleur PIC 16F877A.

Réferences:

[AN-396] Joe Marcin . " Thermocouple Signal Conditioning Using the AD594/AD595 ". Analog Device. AN-396 Application note .

[AN-709] G. King and T. Fukushima . "RTD Interfacing and Linearization Using an ADuC8xx MicroConverter® ". Analog Device. AN-709 Application note .

[AN-942] J. Day and S. Bible ."Piecewise Linear Interpolation on PIC12/14/16 Series Microcontrollers " . Microchip Technology Inc.

[ANT03] ANTONIO J. LOPEZ-MARTIN ."A CMOS A/D Converter with Piecewise Linear Characteristic and Its Application to Sensor Linearization ".Kluwer Academic Publishers. Analog Integrated Circuits and Signal Processing, Vol. 36. pp. 39–46, 2003.

[ATT94] M.Attari . "Methods for linearization of non-linear sensors". In: Proceedings of CMMNI-4 vol. 1. 1993. pp. 344–50.

[ATT95] M. Attari, F. Boudjema, and M. Heniche, "An artificial neural network to linearize a G (tungsten vs. tungsten 26%Rhenium) thermocouple characteristic in the range of Zero to 2000 ^{o}C ", ISIE'95 Proceedings of the IEEE International Symposium on Industrial Electronics, 1995, pp. 176-180.

[ATT96] M.Attari, M.Heniche and F. Boudjema, "A two dimensional intelligent calibration of an ion sensor". IMTC 1996. Vol 2. pp. 788-791.

[BAS08] S. Bashyal, G. K. Venayagamoorthy, and B. Paudel, "Embedded neural network for fire classification using an array of gas sensors," in Sensors Applications Symposium, 2008. SAS 2008. IEEE, 2008, pp. 146-148.

[BOL85] W. T. Bolk. " A general digital linearising method for transducers". J. Phys. E: Sci. Instrument, Vol. 18, 1985.

[BRI96] B. Brignell e N. White. "Intelligent Sensor Systems". IOP Publishing Ltd. Revised Edition. 1996.

[DEM97] G.L.Dempsey, J. S. Alig, N. L. Alt, B. A. Olson and d. E. Redfield ."Control Sensor Linearization Using Artificial Neural Networks ". Kluwer Academic Publishers .Analog Integrated Circuits and Signal Processing. Vol. 13. pp. 321-332. Boston. 1997.

[DEP97] G. L. Dempsey, N. L. Alt, B. A. Olson, and J. S. Alig, "Control sensor linearization using a microcontroller-based neural network," in Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'., 1997 IEEE International Conference on, 1997, pp. 3078-3083 vol.4.

[DOE90] E. O. Doebelin. “Measurement Systems: application and design”. McGraw-Hill Book Co. c1990.

[ELL93] D.L. Elliott, " A Better Activation Function for Artificial Neural Networks". ISR Technical Report TR 93-8. January 29, 1993.

[HAY99] S. Haykin, "Neural Networks: A Comprehensive Foundation", 2nd Edition, Prentice-Hall, 1999.tion, 2nd Edition, Prentice-Hall, 1999.

[LAM97] Lampton, M. " Damping–undamping strategies for the Levenberg–Marquardt nonlinear least-squares method " Computers in Physics, Vol. 11, No. 1, Jan/Feb 1997. pp110 -115.

[LEC98] Y. LeCun, L. Bottou, G. Orr and K. Muller: "Efficient BackProp", in: Orr, G. and Muller K. (Eds), Neural Networks: Tricks of the trade, Springer, 1998.

[FAV87] J. M. Favennec, “Smart sensors in industry”. J.Phys. E: Sci. Instrum. 20. 1987.

[HAU09] John R. Hauser . “ Numerical Methods for Nonlinear Engineering Models ”, Published by Springer 2009. “chapter 7 , Curve Fitting and Data Plotting ”

[HEW07] J.Hewlett, B.Wilamowski, and G. Dundar, "Merge of Evolutionary Computation with Gradient Based Method for Optimization Problems," in Proc. IEEE Int. Symp. Industrial Electronics ISIE 2007, 2007, pp. 3304-3309.

[HOR97] G.V. D. Horn and J. H. Huijsing. “Integrated Smart Sensor Calibration”. Analog Integrated Circuits and Signal Processing. Vol. 14. pp. 207-222. Boston. 1997.

[HOS97] B. J. Hostica, W. Brockherde, and D. Hammerschmidt. “Silicon Sensor Systems”. Kluwer Academic Publishers. Analog Integrated Circuits and Signal Processing. Vol. 14. pp. 261-273. Boston. 1997.

[HUI94] J. H. Huijsing, F.R. Riedijk e G. v. d. Horn. “Developments in Integrated Smart Sensors”. Ed. Elsevier. Sensor and Actuators A, vol. 43, pp. 276-288, 1994 .

[IVA01] M.V.Ivanov. “ Bridge sensor linearization circuit and method”. US Patent, no. 6,198,296 B1, Mar. 6 2001.

[JOS08] J. Rivera, G. Herrera, M. Chacón, P. Acosta and M. Carrillo. ” Improved Progressive Polynomial Algorithm for Self- Adjustment and Optimal Response in Intelligent Sensors ”. Sensors 2008, 8 , pp. 7410-7427(www.mdpi.com/journal/sensors).

[KHA97] K.F. Lyahou, G.V.der Horn and J.H.Huijsing . “A Noniterative Polynomial 2-D Calibration Method Implemented in a Microcontroller”. IEEE Transactions on instrumentation and measurement, Vol. 46, No. 4, August 1997 .

[KOU05] M. KOUIDER, M. NADI, J. PRADO and D. KOURTICHE. “Embedded system design and implementation of standard auto-calibrated measurement chain ”. 1st International Conference on Sensing Technology November 21-23, 2005 Palmerston North, New Zealand , pp. 377-382.

[LLO97] J.A. Lloyd. “An integrated circuit Pressure sensing system with Adaptive Linearity Calibration”. MIT, Juine 1997.

[MAL94] P. Malcovati, C. Azeredo Leme, P. O`Leary, F. Maloberti, e H. Baltes. “Smart Sensor Interface with A/D Conversion and Programmable Calibration” .Special Brief Papers. IEEE Journal of Solid State Circuits. Vol. 29, no 8, August de 1994.

[MAL07] F. Maloberti . “Data Converters ”.Published by Springer 2007. chapter 6 ,Oversampling and low order ΣΔ modulators.

[MEI03] Magali R. G. Meireles, Paulo E. M. Almeida, and Marcelo Godoy Simões. " A Comprehensive Review for Industrial Applicability of Artificial Neural Networks ". IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 50, NO. 3, JUNE 2003

[MID87] S. Middelhoek e S. A. Audet. “Silicon sensors: full of promises ad pitfalls”. J.Phys. E: Sci. Instrum. 20. 1987.

[MON09] N. Mondal, A. Abudhahir, S.K. Jana, SMmunshI and D. P. Bhattacharya . “A Log Amplifier Based Linearization Scheme for Thermocouples ”. Sensors & Transducers Journal, Vol. 100, Issue 1, January 2009, pp. 1-10 .

[NIC08] N. Cotton, "Training Arbitrarily Connected Neural Networks with Second Order Algorithms," Masters of Science, Electrical and Computer Engineering, Auburn University, Auburn, 2008.

[NIC10] N. J. Cotton, “A neural network implementation on embedded systems,” Ph.D. dissertation, Dept. Elect. Eng., Auburn Univ., Auburn, AL, 2010.

[NIC11] Nicholas J. Cotton and Bogdan M. Wilamowski, “Compensation of Nonlinearities Using Neural Networks Implemented on Inexpensive Microcontrollers ”. IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 58, NO. 3, MARCH 2011

[PAŠ05] A. Pašić and J. Dowling . “Linearising Calibration Methods for a Generic Embedded Sensor Interface (GESI) ”. 1st International Conference on Sensing Technology November 21-23, 2005 Palmerston North, New Zealand , pp. 185-190.

[PAT00] J.C. Patra, A.C. Kot and G. Panda. “An Intelligent Pressure Sensor Using Neural Networks ”. IEEE Transactions on instrumentation and measurement, vol. 49, no. 4, pp. 829-834, august 2000 .

[PAT08] J.C. Patra, G. Chakraborty and P.K. Meher. “Neural Network-based Robust Linearization and Compensation Technique for Sensors under Nonlinear Environmental Influences ”. IEEE Transactions on Circuits and Systems I,Vol. 55, pp. 1316–1327.2008.

[PER09] J.M.Dias Pereira, O. Postolache and P. M. B. Silva Girão. “ PDF-Based Progressive Polynomial Calibration Method for Smart Sensors Linearization ”. IEEE Transactions on instrumentation and measurement, vol. 58, no. 9, september 2009 .

[PET05] P. Petchjatuporn, W. Ngamkham, N. Khaehintung, P. Sirisuk, W. Kiranon, and A. Kunakorn, "A Solar-powered Battery Charger with Neural Network Maximum Power Point Tracking Implemented on a Low-Cost PIC-microcontroller," in TENCON 2005 IEEE Region 10, 2005, pp. 1-4.

[POS01] O. Postolache, P. Silva Girão, J.M. Dias Pereira, C. Fosalau, "Microcontroller – Based Data Processing For Non-Linear Measuring Sensors".11 IMEKO TC-4 Symp. - Trends in Electrical Measurement and Instrumentation - September 13-14, 2001 - Lisbon, Portugal, pp 428-432.

[RIE93] Riedmiller, M., and H. Braun (1993), “A Direct Adaptive Method for Faster Backpropagation Learning: the Rprop Algorithm”, in: IEEE International Conference on Neural Networks, San Francisco, CA, pp. 586-591.

[ROJ96] R.Rojas , "Neural Networks :A Systematic Introduction". Springer-Verlag, Berlin, 1996 .

[SAL11] S.O Saleh, M.U Asad, M.A Javed and H.Moazzam, “ Design and Implementation of Neural Network Based Controller for Mobile Robot Navigation ”.26th IEEEP Students’ Seminar 2011 Pakistan Navy Engineering College National University of Sciences & Technology

[ŠAP01] D. Šaponjic and A. Zigic . “ Correction of a Piezoresistive Pressure Sensor Using a Microcontroller ”. Instruments and Experimental Techniques, Vol. 44, No. 1, 2001, pp. 38–44.

[VEH00] P. Vehvilainen, H. Ihalainen, '"Estimating the Activation Functions of an MLP-Network", Proc. IMEKO World Congress Wien, Vol. IX, pp. 359-364, Sept. 2000.

[WIL08] B. M. Wilamowski, N. J. Cotton, O. Kaynak, and G. Dundar, ‘‘Computing gradient vector and Jacobian matrix in arbitrarily connected neural networks,’’ IEEE Trans. Industrial Electronic., vol. 55, no. 10, pp. 3784–3790, Oct. 2008.

[WIL09] B. M. Wilamowski, “Neural network architectures and learning algorithms,” IEEE Ind. Electron. Mag., vol. 3, no. 4, pp. 56–63, Dec. 2009.