الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

المدرسة الوطنية المتعددة التقنيات

ECOLE NATIONALE POLYTECHNIQUE

Département de Génie Electrique

Projet de Fin d'Etudes

En vue de l'obtention du Diplôme d'Ingénieur d'Etat

Filière: *Automatique*

THÈME

Nouvelle Approche Pour la Synthèse d'un ADC Flash

Etude, Conception et Réalisation

Proposé et dirigé par,
Dr. M. ATTARI, IE/USTHB
Dr. F. BOUDJEMA, ENP

Etudié par,
Mr. Sofiane BOUALLAG
Mr. Mounir BOUHEDDA

Promotion
JUIN 1996

Projet Préparé au sein du Laboratoire d'Automatique de l'ENP et au Laboratoire d'Instrumentation et Capteurs de l'IE/USTHB

الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

المدرسة الوطنية المتعددة التقنيات

ECOLE NATIONALE POLYTECHNIQUE

Département de Génie Electrique

Projet de Fin d'Etudes

En vue de l'obtention du Diplôme d'Ingénieur d'Etat

Filière: *Automatique*

THÈME

Nouvelle Approche Pour la Synthèse d'un ADC Flash

Etude, Conception et Réalisation

Proposé et dirigé par,
Dr. M. ATTARI, IE/USTHB
Dr. F. BOUDJEMA, ENP

Etudié par,
Mr. Sofiane BOUALLAG
Mr. Mounir BOUHEDDA

Promotion
JUIN 1996

Projet Préparé au sein du Laboratoire d'Automatique de l'ENP et au Laboratoire d'Instrumentation et Capteurs de l'IE/USTHB

بسم الله الرحمن الرحيم

إلـى الخوارزمي

A la mémoire du cher et regretté " Mouloud "... que Dieu ait son âme.

A la mémoire de ceux que j'aimais et qui me manquent énormément... mes Grand-Parents.

Au deux êtres qui me sont les plus chers: mes Parents.

A toute mon adorable famille... et spécialement ma petite perle " Safa ".

Sofiane.

Je dédie ce modeste travail...

A la mémoire de mes Grand-Parents...
A mes Grand -Parents...
A mes Parents...
A ma soeur et mes frères...
A toute la famille...
A tous mes amis...
Et à ceux qui me sont très chers et qui se reconnaîtront.

Mounir.

AVANT-PROPOS

Ce projet a été réalisé au Laboratoire d'Automatique de l'Ecole Nationale Polytechnique, ainsi qu'au Laboratoire d'Instrumentation et Capteurs de l'Institut d'Electronique de l'Université des Sciences et de Technologie HOUARI BOUMEDIENE .

Au bout du chemin, nous ne manquerons pas d'exprimer, à Mrs. M. ATTARI et F. BOUDJEMA, notre parfaite considération et profonde reconnaissance, pour la bienveillance avec laquelle ils ont dirigé ce travail, pour leurs précieux conseils ainsi que pour leurs inestimables aides.

Nous remercions les responsables du Laboratoire d'Electroacoustique de l'Institut d'Electronique de l'USTHB, pour avoir eu l'amabilité de mettre à notre disposition les moyens de calcul du laboratoire, sans lesquels ce travail n'aurait pu aboutir.

Nos remerciements vont aussi à tous nos enseignants, du primaire au supérieur. Ainsi qu'à ceux qui ont participé, de loin ou de près, à l'aboutissement de ce travail; spécialement H. Boumediène et L. Salim.

Pour terminer; un GRAND SALUT à la Promotion d'AUTOMATIQUE de 1996.

ملخـــص

الغاية من هذا العمل هو وضـع طريقـة جديـدة للتحويـل المسـتمر/الرقمـي الفـا ئـق السرعة. لهذا الغرض أبدت الشبكات العصبية الإصطناعيـة نجاعتهـا، فقد جربـت عـدة صيـغ (مخـارج حسـب النظام الثنـائي، مخـارج حسـب نظـام قـراي، شـبكة ديناميكية) معطية نتائج مرضية نسبيا. في النهاية، اقترحت هندسة جديدة للمحـولات العصبية المستمرة/الرقمية ذات ن مخرج بدون تعلم. برهنت إستعماليةهذه الهندسة الجديدة بإنشاء محول مستمر /رقمي عصبي ذو أربع مخـارج الـذي أعطـى النتـائج المنشودة.

كلمـات مفتاحيـة: محـول مسـتمر/رقمـي، تحويـل، تمثيـل، تعلـم، تعميـم، الشـبكات العصبية الإصطناعية،مضخم عملي، قراي، البت الأكبر قيمة، البت الأصغرقيمة.

ABSTRACT

The aim of this work is to set up a new high-speed Analog to digital converter. The artificial neural networks offer good abilities for this application, then several approaches(binary outputs, Gray outputs, dynamic network,...) bordering to relatively satisfying results were used . At the end, a new architecture allowing neural n bits ADC's synthesis without learning was elaborated. The feasability of this new approach was proved by the realization of a four bits neural ADC, which gives the desired results.

Key words: ADC, Conversion, Modelisation, Learning, Generalisation, Artificial Neural Networks, Operationnal Amplifier, Gray, MSB, LSB.

RESUME

La finalité de ce travail était d'établir une nouvelle structure pour la conversion A/D ultra-rapide. A cet effet les réseaux de neurones artificiels se sont bien prêtés; plusieurs approches (sortie binaire, sortie Gray, réseau dynamique...) ont été utilisées aboutissant à des résultats relativement satisfaisants. A la fin a été élaboré une nouvelle architecture permettant la synthèse d'un ADC neuronal à n bits sans aucun apprentissage. La faisabilité de cette nouvelle architecture a été prouvée par la réalisation d'un ADC neuronal à quatre bits qui donna les résultats désirés.

Mots clefs: ADC, Conversion, Apprentissage, Généralisation, Réseaux de neurones artificiels, Amplificateur opérationnel, Gray, MSB, LSB.

SOMMAIRE

SECTION 3
ALGORITHME D'APPRENTISSAGE

2ème PARTIE
MODELISATION, CONCEPTION ET REALISATION

SECTION 1
MODELISATION D'ADCs PAR ENTRAINEMENT DE RESEAUX DE NEURONES

SECTION 2
METHODE DE LA SYNTHESE D'UN ADC NEURONAL SANS APPRENTISSAGE

SECTION 3
ETUDE THEORIQUE SUR LES AMPLIFICATEURS OPERATIONNEL: APPLICATION A LA CONCEPTION ET REALISATION D'UN NEURONE.

Voilà certainement la révolution la plus importante que le monde de l'informatique a subi depuis l'introduction des ordinateurs numériques à la fin de la seconde Guerre Mondiale. Des architectures parallèles sont désormais capables d'apprendre à prononcer correctement un texte en dépit des pièges de la phonétique, de reconnaître des chiffres mal dessinés sur une enveloppe ou de commander les déplacements d'un robot pour saisir un objet sans aucune autre programmation qu'un simple apprentissage. Comment? Tout simplement en s'inspirant avec l'aide de la neurobiologie, du meilleur modèle de la machine polyvalente, incroyablement rapide et surtout douée d'une incomparable capacité d'auto-adaptation; le cerveau humain.

Cette révolution s'appelle ***"les réseaux de neurones"***.

INTRODUCTION GENERALE

Cela fait quelques années (pour ne pas dire décennies) déjà, que l'ordinateur est devenu un véritable partenaire dont ne peut se passer l'être humain. On le trouve désormais partout, dans l'administration, dans le laboratoire, dans l'usine, dans la maison... et la liste est loin d'être terminée.

Mais cette machine qui semble penser et réfléchir, qui semble intelligente, ne fait - si bien que mal- que des calculs... et qui dit calcul dit nombres. Ceci est vrai même pour elle (cette machine), seulement ici, les nombres ont un format un peu spécial: Ils sont binaires[1] et pulsés.

On a donc besoin d'un interprète pour communiquer avec elle (rappelons-le: Nos nombres sont décimaux et les signaux dans ce monde sont, en général, analogiques c.à.d. continus dans le temps et non pas pulsés).

En langage scientifique cet interprète est appelé *interface*, et entre nous et cette machine, il y a à la base de toute interface, deux circuits primordiaux: Le Convertisseur Analogique/Digital qu'on note ADC (Analog to Digital Converter) et le Convertisseur Digital/Analogique noté DAC (Digital to Analog Converter). C'est le premier circuit qui fait l'objet de cette étude.

Rapidité et Précision (haute résolution) des ADCs sont les deux caractéristiques les plus souhaitées et les plus demandées. Surtout avec l'élan considérable que connaissent le développement, l'utilisation et la diffusion des systèmes d'acquisition et de traitement numérique des données.

Plusieurs architectures, dédiées à la satisfaction de ces deux prérequis, ont été proposées. On peut citer entre-autres la technique Flash[2], très rapide mais de résolution limitée (jusqu'à 10 bits). Ainsi que l'application des réseaux de neurones artificiels à la conversion A/D.

Cette dernière technique a attiré, depuis des années déjà, l'attentiondes chercheurs et a fait l'objet de plusieurs travaux de recherche aboutissant à diverses publications.

---

[1] Voir 1ère Partie, Section 1, § 1.2.
[2] Voir 1ère Partie, Section 1, § 1.3.2.

A l'origine de cette orientation, le travail de Tank et Hopfield [Tan 86] qui consistait en l'application des réseaux de Hopfield[3] à la conversion A/D. Mais le réseau qui en découla présentait des anomalies dues au problème des minimums locaux. D'autres publications ont suivi et avaient pour but la résolution de ce problème [Avi 91] [Mar 91].

Parmi les publications les plus récentes dans ce domaine, citons celles publiées dans I^{3}C (Integrating Intelligent Instrumentation & Control) dans le cadre du IMTC 95 (Instrumentation Measurement Technology Conference) qui a eu lieu à Waltham, Massachusetts (Boston, USA) en Avril 95: [Ber 95][Dap 95] [Bor 95].

Le présent travail ayant pour dessein, l'élaboration de structures neuronales parallèles hardware en vue de la conversion Analogique/Digital, peut être considéré comme une tentative de nouvel apport à la synthèse d'ADCs ultra-rapide. Il est organisé en deux parties :

La première qui regroupe l'essentiel de la théorie nécessaire pour contourner le problème posé, elle consiste en trois sections:

- Etude théorique sur les ADCs.
- Etude théorique sur les réseaux de neurones artificiels.
- Algorithmes d'apprentissage.

Et la deuxième qui concerne la modélisation d'ADCs neuronaux par différentes approches, ainsi que la réalisation pratique. Elle est scindée en quatre sections:

- Modélisation d'ADCs par entraînement des réseaux de neurones.
- Méthode pour la synthèse d'un ADC neuronal sans apprentissage.
- Etude théorique sur les amplificateurs opérationnels: Application à la conception et à la réalisation d'un neurone.
- Réalisation pratique.

Enfin nous terminerons par une conclusion générale et des perspectives, autrement dit, par la moralité du présent projet.

[3] Voir 1ère Partie, Section 4

1ère Partie

ETUDES THEORIQUES

SECTION

1

ETUDE THEORIQUE SUR LES ADCs

1.1 INTRODUCTION

La plupart des systèmes physiques étudiés sont de nature analogique (leurs comportements sont continus dans le temps). Aussi les instruments de mesures (électroniques ou autres) délivrent des signaux analogiques. Toutefois les calculateurs analogiques ne présentent pas de bonnes capacités de traitement et de stockage. En contrepartie les systèmes de traitement numériques (à base de microprocesseurs ou microcontrôleurs) possèdent beaucoup d'avantages dans ce domaine: coût réduit, grande capacité de traitement et de stockage et de restitution - avec précision - des données... Il est donc parfois nécessaire de convertir les signaux analogiques en signaux digitaux: C'est le rôle assumé par les convertisseurs analogiques/digitaux communément désignés par: ADCs (Analog to Digital Converters).

Dans cette section, on se propose une étude ~~détaillée~~ des différentes techniques utilisées actuellement pour la conversion analogique/numérique, en citant avantages et inconvénients de chacune.

1.2 CODES NUMERIQUES

Avant de parler des techniques de conversion, il est, pensons-nous, nécessaire et instructif de faire un bref rappel sur différents codes numériques.

1.2.1 CODE BINAIRE

- Dans une transmission série de données binaires le premier bit transmis s'appelle le bit le plus faible qu'on notera LSB de sa spécification anglaise, Least Significant Bit et le bit reçu en dernier s'appelle le bit le plus fort: MSB de Most Significant Bit.

- On appelle un mot (ou une donnée) binaire une série de n bits.
- A partir de n bits on peut constituer 2^n mots.
- La conversion binaire / décimale se fait de la manière suivante:

Mot binaire	b_{n-1}(MSB)	. . .	b_1	b_0(LSB)
Poids	2^{n-1}	. . .	2^1	2^0
Valeur décimale	$b_{n-1}\times2^{n-1}$ +	. . .	$+b_1\times2^1$	$+ b_0\times2^0$

Exemple:

$$(1011)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 = (11)_{10}$$

1.2.2 CODE BCD

BCD veut dire: Binary Coded Decimal.

C'est une manière de représenter un nombre décimal: Chaque chiffre du nombre est remplacé par son équivalent binaire (toujours sur quatre bits).

Exemple:

$$(459)_{10} = (0100\ 0101\ 1001)_{BCD}$$

Ce code est recommandé pour l'affichage numérique (7 segments).

1.2.3 CODE GRAY

Le code Gray est caractérisé par le changement d'un seul bit lors du passage de l'état actuel à l'état suivant, cette caractéristique permet d'éliminer l'effet des "Glitchs" [Cla 80] qui apparaît dans la conversion digitale / analogique (voir figure 1.2.1).

- Exemple illustratif:

Décimal	Binaire	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

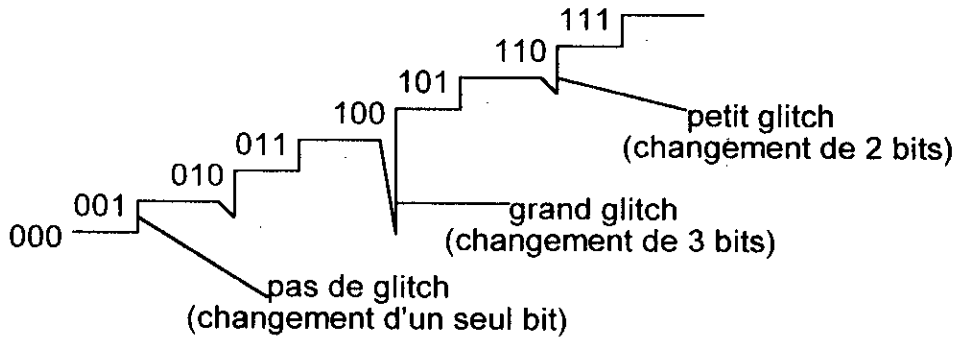


Figure 1.2.1: Effet des " glitchs " lors de la conversion digitale /analogique

1.3 TECHNIQUES DE CONVERSION

1.3.1 FEEDBACK ADCs

Le schéma synoptique de base de ce type de convertisseur est le suivant:

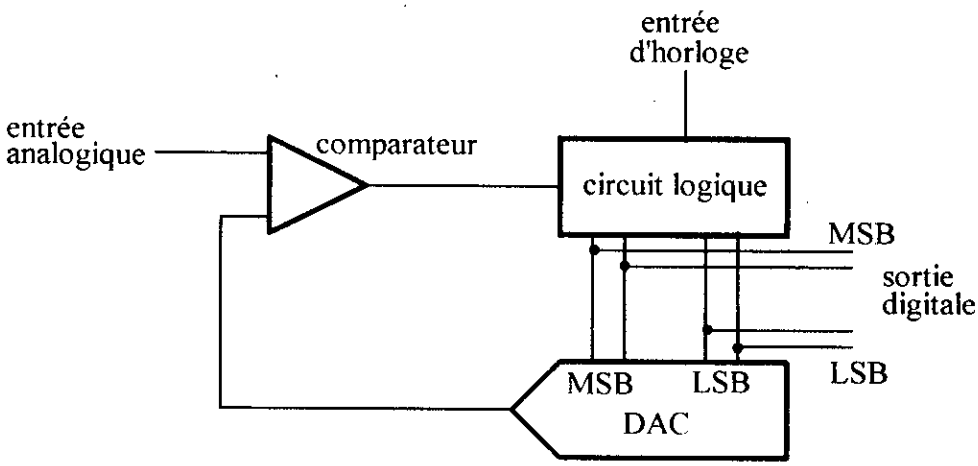


Figure 1.3.1: Schéma synoptique de base de l'ADC à feedback

Ces ADCs sont constitués d'un convertisseur digitale / analogique (DAC: de Digital to Analog Converter), d'un circuit logique et d'un comparateur. Le circuit logique incrémente (ou décrémente) le nombre binaire appliqué au DAC (qui est aussi la sortie de l'ADC), la tension de sortie du DAC est comparée à la tension d'entrée de l'ADC, ensuite le comparateur délivre un signal commandant le circuit logique.

1.3.1.1 L'ADC rampe - type

C'est le plus simple des FEEDBACK ADCs; le circuit logique (voir figure 1.3.1) consiste essentiellement en un compteur.Ce dernier doit être mis à zéro au début de la conversion.

Quand $(V_{in} - V_{DAC}) \leq 0$ le signal délivré par le comparateur inhibe le compteur; la sortie de l'ADC est la dernière sortie du compteur.

Son temps de conversion est donné par la relation [Cla 80]

$$t_c = \frac{V_{in}}{R_{in} I_{ref}} . 2^n T_c$$

où:

n: nombre de bits de l'ADC

T_c: période d'horloge

I_{ref}: courant de référence du DAC

R_{in}: résistance d'entrée

V_{in}: tension d'entrée du DAC

1.3.1.2 L'ADC à poursuite (Tracking ADC)

La différence entre ce type d'ADCs et le précédent réside essentiellement en :

• Le circuit logique n'est plus un compteur mais plutôt un compteur / décompteur, de ce fait on a pas besoin d'initialiser le circuit logique à chaque conversion.

• Le principe de fonctionnement se base toujours sur la comparaison entre V_{in} et V_{DAC}, mais de la façon suivante:

Si $V_{DAC} - V_{in} > 0$ alors le circuit logique travaille en décompteur sinon en compteur. A la fin de la conversion la sortie de l'ADC oscille entre deux valeurs adjacentes.

Limitation du taux de variation de l'entrée (Slew rate)

Si une tension analogique variable est appliquée au tracking ADC, le retour force le signal digital généré à suivre les variations du signal analogique. Toutefois il existe un taux maximum de variation de l'entrée analogique pour lequel cette poursuite est maintenue; il est dicté par la relation suivante [Cla 80]:

$$t_v = \frac{V_{fs}}{2^n . T_c} \quad (V/\mu s)$$

où V_{fs} est la longueur maximale de l'interval de variation de V_{in} (*full scale* analog input).

1.3.1.3 L'ADC à approximations successives

C'est l'une des techniques de conversion analogique/digitale les plus populaires, son fonctionnement est comme suit:

- Initialement tous les bits sont mis à 1.
- Commencer par mettre - provisoirement - à 0 le MSB.
- Si la sortie du DAC excède l'entrée, le MSB est laissé à 0 sinon il est mis à 1.
- Refaire les mêmes opérations pour les (n-1) bits restants selon l'ordre décroissant des poids.

Il faut donc (n+1) cycles d'horloge (pour un ADC à n bits) pour avoir le résultat final (n cycles pour déterminer le mot binaire plus un cycle pour le valider).

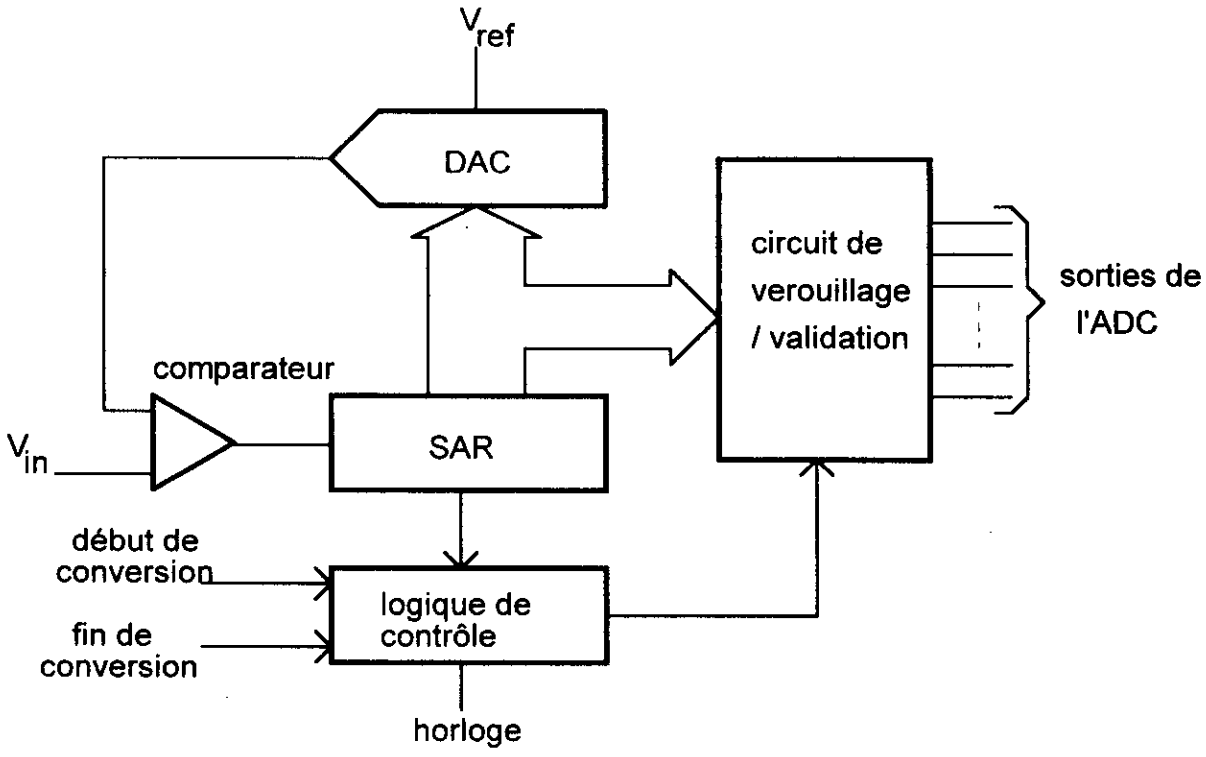


Figure 1.3.2: Schéma synoptique d'un ADC à approximations successives

Le registre d'approximations successives:
(SAR: Successive Approximation Register)

Le SAR est le circuit logique explicité dans le schéma de la figure 1.3.2. C'est un registre SIPO (Serial In / Parallel Out), l'entrée du SAR est la sortie du comparateur pendant la séquence d'essais nécessaire pour la conversion (Voir étapes plus haut).

Cycle	MSB b_1	b_2	b_3	b_4	b_5	b_6	b_7	LSB b_8	Sortie du comparateur
1	0	1	1	1	1	1	1	1	1
2	1	0	1	1	1	1	1	1	0
3	1	0	0	1	1	1	1	1	0
4	1	0	0	0	1	1	1	1	1
5	1	0	0	1	0	1	1	1	1
6	1	0	0	1	1	0	1	1	0
7	1	0	0	1	1	0	0	1	1
8	1	0	0	1	1	0	1	0	0
9(conversion terminée)	1	0	0	1	1	0	1	0	–

L'ADC à approximations successives est relativement précis et rapide. En pratique son temps de conversion varie entre 1µs et 50µs avec une précision de 8 à 20 bits (Voir tableau illustratif à l'annexe A.1).

Ces convertisseurs fonctionnent sur un bref échantillon de la tension d'entrée, et si l'entrée change pendant la conversion une erreur plus ou moins importante est induite et, faut-il signaler que, les piques à l'entrée sont fatals.

Bien que ces convertisseurs se tiennent généralement précis, ils peuvent avoir d'étranges nonlinéarités et des "missing codes" [Hor 89].

1.3.2 ADCs A CONVERSION PARALLELE

1.3.2.1 ADCs Flash

Afin de mieux assimiler le principe de fonctionnement de ce type d'ADCs, il est préférable, pensons-nous, de dresser tout d'abord le schéma synoptique.

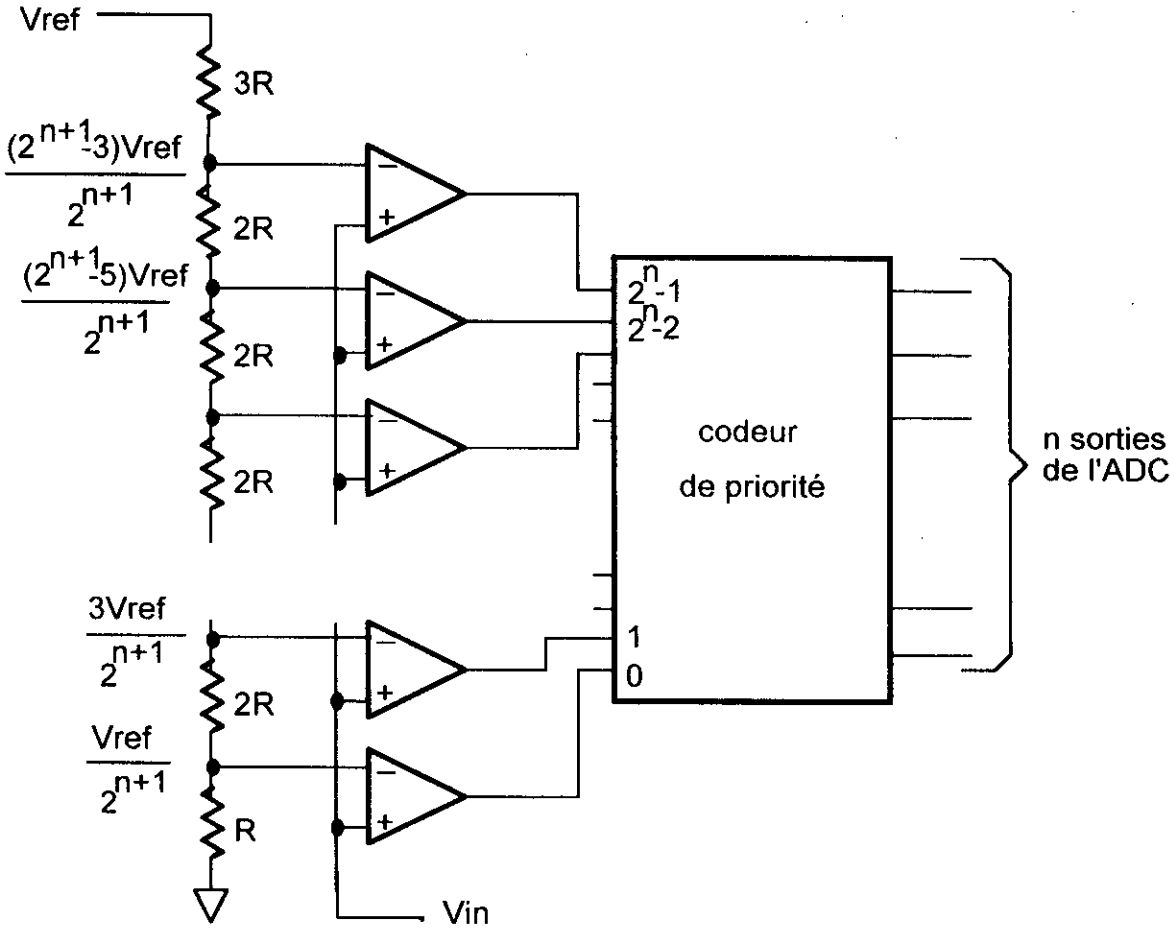


Figure 1.3.3: Schéma synoptique d'un ADC Flash

Comme on le voit sur le schéma, la tension d'entrée est comparée à des fractions de V_{ref} équidistantes (V_{ref} est la tension de référence de l'ADC). Les sorties des comparateurs sont les entrées d'un codeur de priorité[1] qui attribut à l'entrée - activée - qui jouit de la plus haute priorité, c.à.d. la sortie de l'Amp Op activé (sortie non nulle) qui compare V_{in} à la plus grande fraction de V_{ref}.

[1] Appelé aussi codeur (2^n - 1) à n lignes.

• c'est la plus rapide des méthodes de conversion analogique / digitale , le temps de propagation est égale à la somme de celui du comparateur et du codeur (15 à 300 conversions par seconde - cf. A.1-)

• Les codeurs de priorité commerciaux qui existent, jouissent de 4 à 10 sorties, les ADCs Flash se trouvent ainsi limités à 10 bits.

• En plus ils sont excessivement chers et encombrants.

1.3.2.2 ADCs demi Flash

Une variante de la technique Flash est appelée demi Flash dont le but est d'augmenter la résolution puisque l'erreur entre l'équivalent analogique du mot binaire - qui est la conversion de l'entrée par la technique Flash à n/2 bits - et l'entrée analogique est aussi converti par la même technique et injectée dans un ADC à n/2 bits dont la tension de référence est $V_{ref}/16$ [Hor 89] (Voir figure 1.3.4).

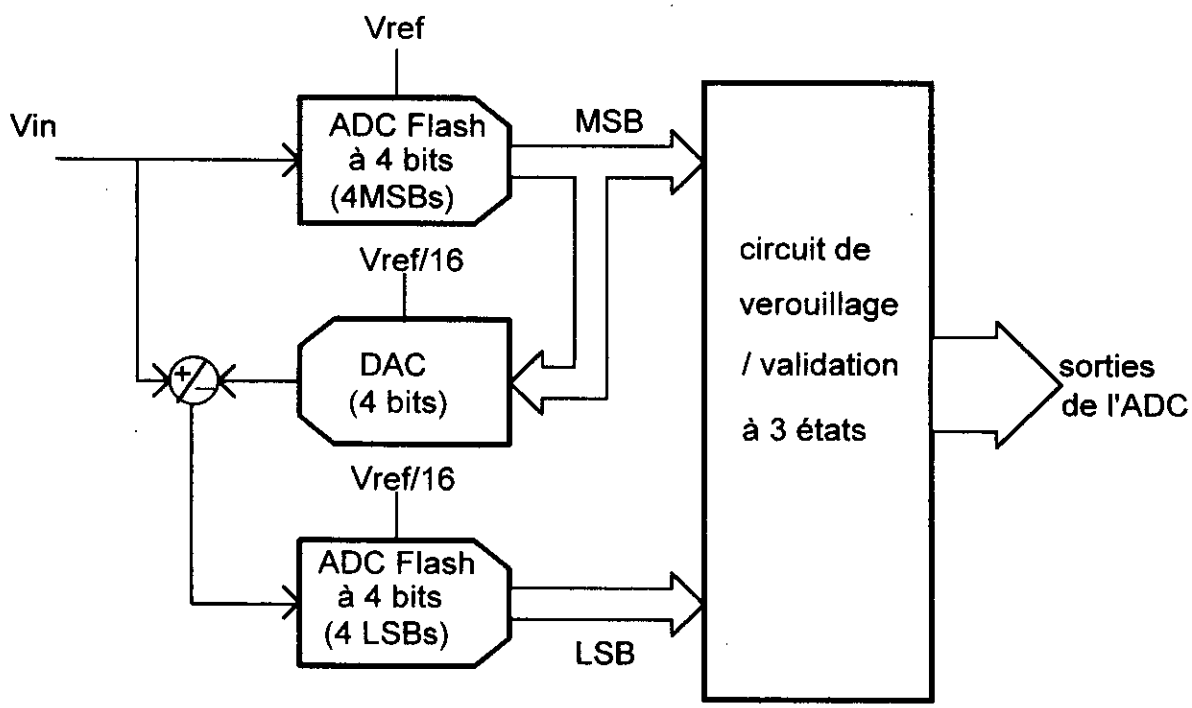


Figure 1.3.4: ADC demi Flash

1.3.3 TECHNIQUES D'INTEGRATION POUR LA CONVERSION A/D

1.3.3.1 Intégration Simple rampe

Dans cette technique un générateur de rampe est lancé au début de la conversion, en même temps un compteur commence à compter; quand la tension aux bornes de la capacité V_c atteint la tension V_{in} un comparateur arrête le compteur, le nombre d'impulsions d'horloge est proportionnel à la tension d'entrée: Ce nombre est la sortie de l'ADC.

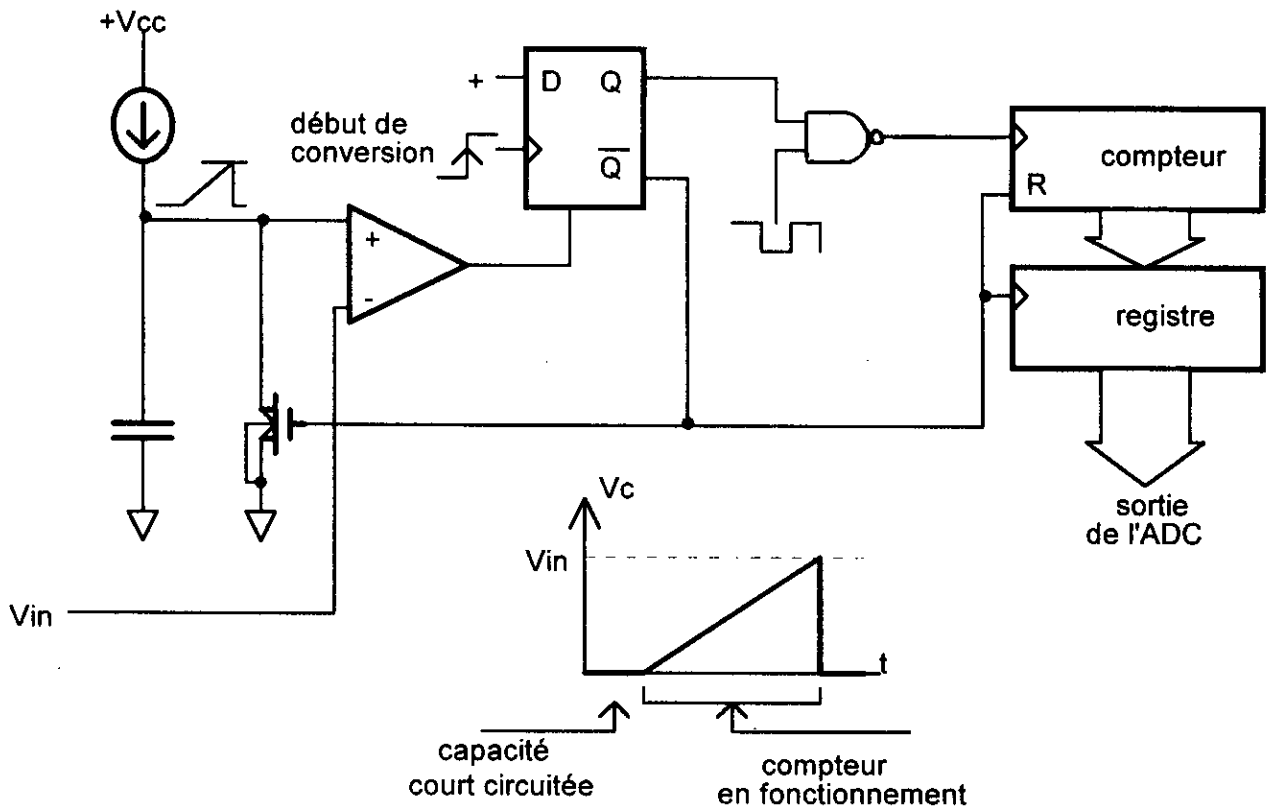


Figure 1.3.5: ADC à Simple rampe

A la fin de la conversion $\overline{Q}$=1, le compteur est mis à zéro et la capacité est court circuitée par le transistor; le convertisseur est prêt pour une nouvelle conversion.

• Cette technique est conseillée particulièrement pour des applications qui ne nécessitent pas une grande précision mais qui ont toujours besoin d'une bonne résolution et d'un espacement uniforme des niveaux adjacents [Hor 89].

• Mais cet ADC fait défaut quand une grande précision est demandée.

• En plus ça nécessite une grande précision et une grande stabilité de la capacité et du comparateur. Solution:

1.3.3.2 Intégration double rampe

Le principe est presque le même de la technique précédente, seulement ici, la capacité est chargée par un courant proportionnel à la tension d'entrée pendant un temps fixé, ensuite elle est déchargée par un courant constant jusqu'à $V_C = 0$. Le temps de décharge est proportionnel à la tension d'entrée.

Pendant la décharge de la capacité le compteur comptait les impulsions d'horloge, quand $V_C = 0$; il est inhibé. Le nombre d'impulsions est la sortie de l'ADC (puisqu'il est proportionnel à V_{in}).

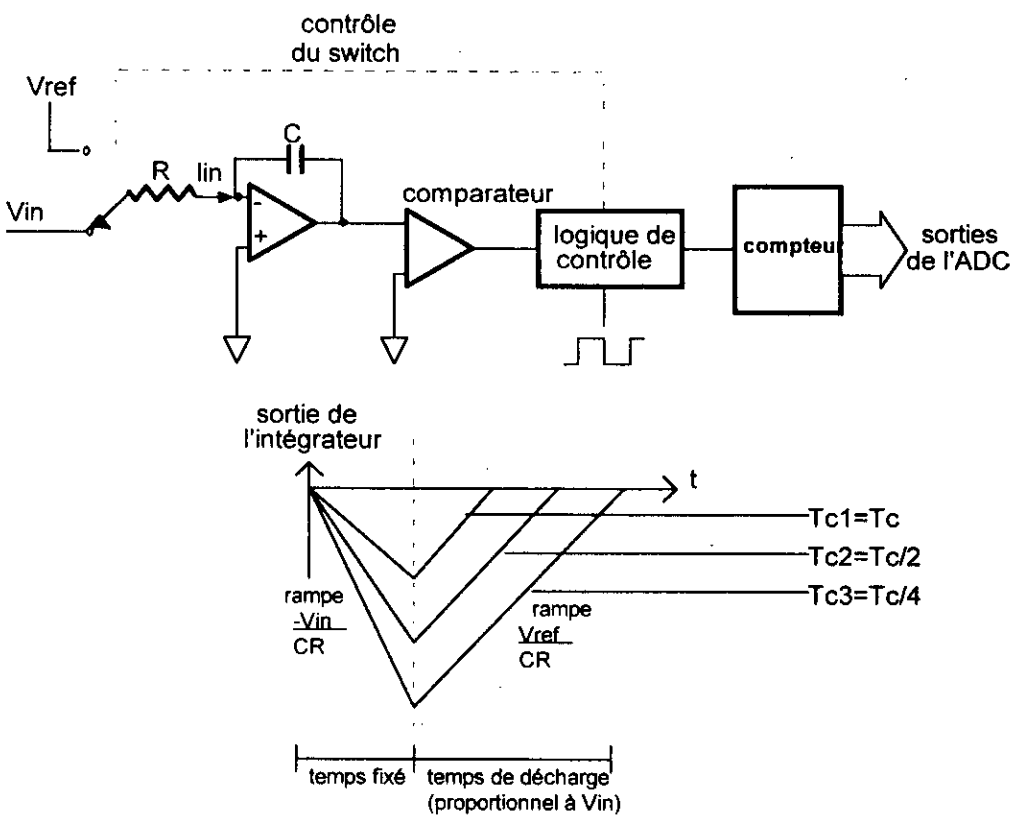


Figure 1.3.6: ADC double rampe (avec graphe illustratif)

• Cette technique est caractérisée par une bonne précision, sans trop d'exigences sur les performances des composants.

• La fréquence d'horloge n'a pas d'influence sur la conversion, ce qui augmente la fidélité de l'ADC [Hor 89] (Voir figure 1.3.6: graphe illustratif).

1.3.3.3 ADCs Delta-sigma

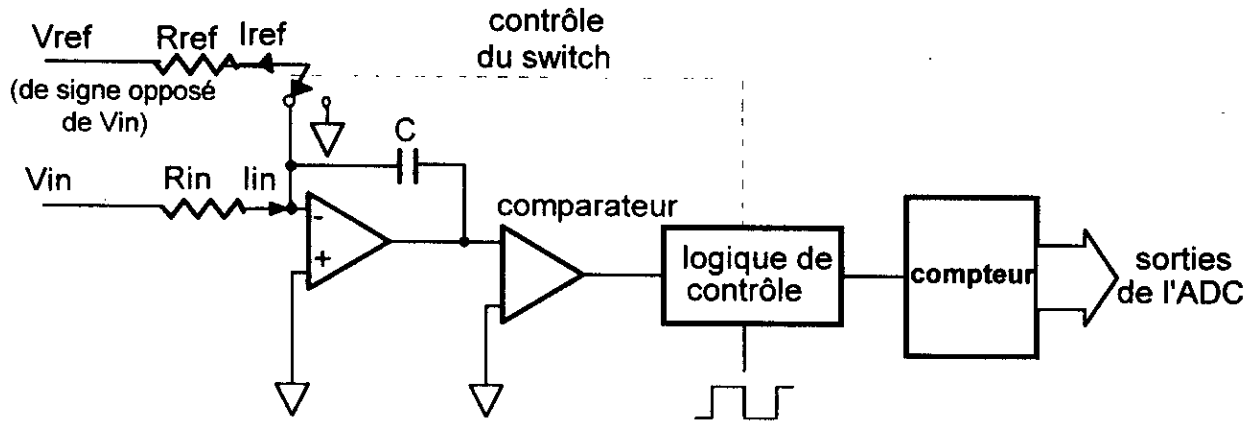


Figure 1.3.7: ADC Delta-sigma

Cette méthode requiert l'annulation du courant d'entrée à l'aide d'une commutation " switching " à une source de courant interne.

La tension d'entrée est appliquée à un intégrateur dont la sortie est comparée à une tension fixée (par exemple la masse). Selon la sortie du comparateur, des impulsions de courant de largeur fixe sont injectées alternativement -à l'aide d'un switch- à la jonction ou à la masse à chaque transition d'horloge, dans le but de maintenir proche de zéro le courant appliqué au comparateur. Un compteur compte ces balancements pendant un temps fixe de conversion (par exemple 2048 cycles d'horloge), le nombre d'impulsions est proportionnel à la tension d'entrée [Cla 80]:

L'augmentation de la charge pendant un temps T_c est:

$$q = \int_0^{T_i} I_{ref} dt = \frac{V_{ref}}{R_{ref}} T_c$$

La charge fournie par le courant d'entrée durant le temps fixe de conversion T_i est

$$Q = \int_0^{T_i} \frac{V_{in}}{R_{in}} dt$$

où $T_i = NT_c$ (pour l'exemple donné ci-dessus N = 2048), ce qui donne:

$$Q = \frac{NT_c}{R_{in}} \bar{V}_{in}$$

où $\bar{V}_{in}$: valeur moyenne du signal d'entrée appliqué durant T_i.

Soit N_X le nombre de balancements nécessaires pour garder la référence de courant appliquée au comparateur moyennement nulle, alors:

$$N_x q = Q \Rightarrow N_x = \frac{R_{ref}}{R_{in} V_{ref}} NV_{in}$$

N_X est aussi le nombre compté. Ceci établit la proportionnalité entre V_{in} et la sortie du compteur.

• En comparaison avec la technique de Double rampe, Delta-Sigma exige des circuits de switching précis, mais le comparateur en aval de l'intégrateur peut être de basse précision.

• Les techniques Double rampe et Delta-Sigma s'appellent aussi méthodes de balancement de charge.

• En général cette méthode n'est pas chère et est précise et fidèle.

• Mais elle reste lente comparée à la technique d'approximations successives (Voir annexe A.1).

1.4 CONCLUSION

Dans cette section nous avons présenté l'essentiel des techniques de conversion analogique/digitale en expliquant leurs principes de base, tout en montrant leurs avantages et inconvénients.

On a pu voir la technique des FEEDBACK ADCs, simple dans sa conception mais peu précise et plus ou moins lente. Toutefois il existe une variante offrant un bon

compromis rapidité-précision: C'est la méthode des approximations successives; élevée au rang de la technique la plus populaire.

Ensuite on a abordé les techniques parallèles (Flash et demi Flash) rapides, coûteuses et de précision limitée. A la fin ont été traité les techniques d'intégration (Simple rampe, Double rampe, Delta-Sigma), pas coûteuses, fidèles, précises mais lentes.

Quand il est question de choisir, tout dépend de l'application et des trois facteurs dominants: la vitesse, la précision et le coût.

Afin de voir plus clair en ce qui concerne le choix, nous conseillons au lecteur de s'aiguiller sur l'annexe A.2.

SECTION

2

ETUDE THEORIQUE SUR LES RESEAUX DE NEURONES ARTIFICIELS

2.1 INTRODUCTION

Bien que capables d'effectuer des millions d'instructions par seconde, les machines de Von Neuman sont et resteront toujours incapables (à titre d'exemples non exhaustifs) de distinguer instinctivement la prononciation de h dans "habile" et celle de h dans "chariot", de reconnaître une voiture vue d'un angle différent de celui sous lequel elle a été mémorisée et même d'effectuer la moindre opération arithmétique en absence du programme adéquat...

C'est là quelques tâches qu'un homme ordinaire qualifiera d' "évidentes"...! "Pourquoi?" diront certains "car l'homme est doué d'intelligence, alors que la machine ne l'ait pas! " répliqueront d'autres...
On est donc ramené à parler d'intelligence; et qui dit "intelligence" dira "cerveau"...

Partant de l'idée de simuler cette miraculeuse machine -"The brain is the last and greatest biological frontier"[1] -, et voulant assimiler son fonctionnement, des scientifiques ont commencé par modéliser son unité élémentaire "le neurone", et à partir de cette dernière ont construit des systèmes connectables; donnant ainsi naissance aux réseaux de neurones.

---

[1] James Watson: Codécouvreur de la structure de l'ADN avec F. Crich, prix Nobel de medecine en 1962.

Etant capable de simuler des fonctions humaines les applications des réseaux de neurones s'imaginent à volonté: reconnaissance de formes, vision artificielle [Yed 94], traitement de la parole [SVM 89], commande des système non linéaires [Ham 95], instrumentation [Hen 94].

Malgré leur nom, les réseaux de neurones ne sont pas des dispositifs biologiques mais bien des circuits électroniques dont chaque élément est censé simuler le fonctionnement de la cellule élémentaire du cerveau: *le neurone* [B&B 95].

2.2 NOTIONS DE NEUROPHYSIOLOGIE

2.2.1 ANATOMIE DU NEURONE

Hormis l'aspect macroscopique; la physiologie et l'anatomie ancienne ne permettaient pas de comprendre la structure réelle du système nerveux.

Ce n'est qu'en 1891 qu'une hypothèse fut faite par Waldayer [Cou 80]. Permettant ainsi une première analyse du système nerveux, elle est que tout système nerveux, malgré les apparences les plus inextricables, est constitué par des chaînes de cellules particulières dotées de prolongements conducteurs d'une impulsion nerveuse (stimulation). Ces cellules se relient entre elles par simple contiguïté (contact). Chacune d'elles est appelée "neurone" du mot grec "neuron = nerf".

La méthode de synthèse de Colgi-Cox [Besl 78]; permet de présenter le neurone de la manière suivante:

1- Une masse de cytoplasme dont le noyau est appelé: cytone.

2- Prolongements; de deux types:

a) Les dendrites (du grec dendron = arbre car elles sont arborescentes) souvent multiples (neurones multipolaires), l'information est céllulipète.

b) L'axone: l'influx nerveux (information) y circule dans un sens céllulifuge.

2.2.1.1 La synapse

Un neurone peut être relié à des dizaines de milliers d'autres neurones dont il reçoit des informations, qu'il émet après intégration vers plusieurs autres dizaines de milliers de neurones.

Les terminaisons axoniques s'articulent soit avec le cytone d'un ou plusieurs neurones, soit avec des dendrites, soit avec des actionneurs nerveux (cellules musculaires) pour former des synapses.

Il existent deux types de synapses [Bes1 78]:

- Synapses excitatrices: Les synapses sont dites excitatrices quand elles sont responsables d'engendrer un influx vers la cellule postsynaptique. Ce qui provoque la transmission de l'information.
- Synapses inhibitrices: Il existe des synapses de loin les plus nombreuses, comme les synapses neuronales, où l'action exercée par les terminaisons presynaptiques reste souvent latente, locale et non propagée.

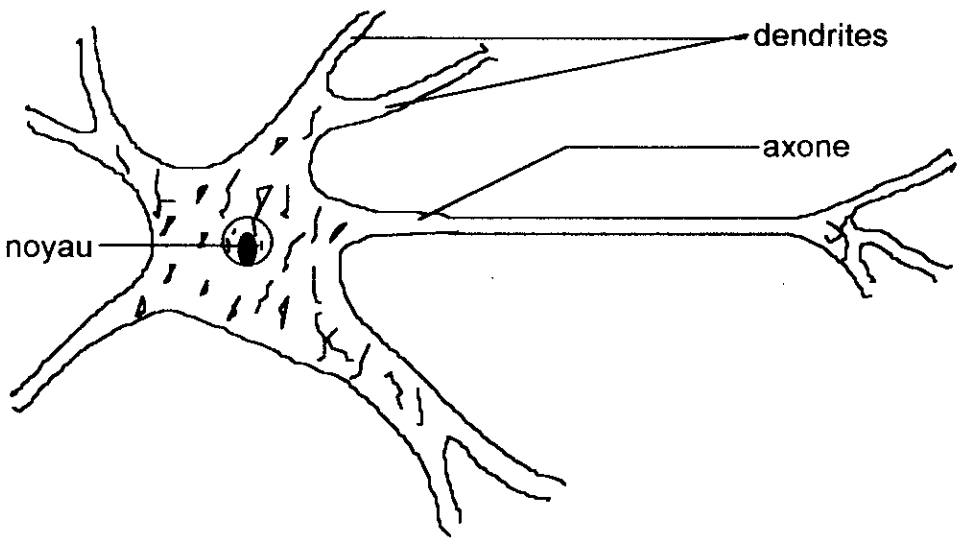


Figure 2.2.1: Schéma simplifié du neurone

2.2.2 PHYSIOLOGIE DU NEURONE

L'interconnexion de neurones forme le système nerveux, qui intègre des informations qu'il reçoit du monde extérieur et de l'organisme lui même, afin d'assurer toutes les fonctions vitales et nécessaires à la survie de l'homme. Donc pour bien comprendre, une étude détaillée de l'élément constitutionnel s'avère nécessaire.

2.2.2.1 Fonctions du neurone

Le neurone est chargé de [Cno 83]:

1- Transmettre les informations captées au niveau des entrées (récepteurs);

2- Traiter ces informations au niveau des centres (cerveau et moelle épinière);

3- Elaborer des réponses;
4- Transmettre les informations aux effecteurs.

2.2.2.2 Caractéristiques du neurone

- Seuil d'excitabilité: La réponse à une stimulation apparaît dans le cas où le courant de stimulation est suffisant [Besl 78];

L'intensité juste pour obtenir une réponse est appelée seuil d'excitation .

- Loi du tout ou rien: Un potentiel d'action engendré par un stimulus juste assez intense pour être efficace est exactement semblable à un potentiel produit par un stimulus 10 ou 100 fois plus important [Besl 78];

Le potentiel d'action est un phénomène tout ou rien.

- Sommation des potentiels: Deux stimulus qui séparément n'auraient pu exciter un neurone, peuvent sommer leurs effets et déclencher un influx nerveux nécessaire à l'excitation du neurone [Besl 78].
- Codage de l'information: Le neurone ne peut moduler sa réponse qu'en jouant sur la fréquence d'envoi de ce signal, donc en formulant des messages nerveux à une fréquence variable, autrement dit des messages transmis selon un code temporel (comme le Morse)[Cou 80].

2.2.2.3 Classification des chaînes synaptiques

Les synapses unissent des ensembles de neurones formant ainsi des réseaux plus ou moins complexes, il y a plusieurs modalités d'arrangement dans les réseaux qui constituent le système nerveux [Besl 78].

- Dispositif divergent: Un neurone distribue l'information à plusieurs cellules postsynaptiques.
- Dispositif convergent: Plusieurs cellules presynaptiques rentrent en contact avec une seule cellule postsynaptique.
- Dispositif linéaire: Les neurones se succèdent et chaque maillon reçoit l'information du maillon précédent.
- Dispositif récurent: Les collatérales d'un neurone peuvent s'articuler sur le même neurone d'une façon indirecte ou avec des neurones du même ordre hiérarchique.

2.2.2.4 Structure d'un système nerveux

La connexion entre plusieurs structures élémentaires (composées de cellules nerveuses spécifiques: Cellule sensorielle, protoneurone, neurone intercalaire, protoneurone moteur...) par le biais de leurs neurones centraux, avec mise en place d'autres neurones intercalaires, qui eux même unissent entre eux les neurones centraux en leur imposant une action hiérarchique, édifiera un système nerveux de plus en plus complexe et de plus en plus apte à assumer des tâches qui sont de plus en plus difficiles et complexes [Cou 80].

2.2.3 L'APPRENTISSAGE

On désigne sous le terme d'apprentissage l'action qui permet d'acquérir, de modifier une connaissance ou une aptitude motrice, grâce à l'expérience, en vue d'une meilleure adaptation de l'individu au milieu dans lequel il vit, il est l'expression de deux propriétés fondamentales du système nerveux:

- La capacité de conserver les empruntes du passé.
- La possibilité de discriminer ce qui est utile et ce qui est nuisible.

Les mécanismes de l'apprentissages sont liés à une faculté particulière de l'homme: La mémoire.

2.2.3.1 L'importance de la mémoire

La mémoire est la capacité (que possèdent les organismes vivants) d'acquérir, de retenir et d'utiliser un ensemble d'informations ou de connaissances.
Elle est au service d'un nombre considérable de fonctions, au point où on peut dire qu'il n'y a pas de connaissance sans mémoire.

La mémoire est une fenêtre sur les événements du monde environnant, grâce à elle le passé de l'homme guide son appréhension au présent et dispose d'une base pour exercer ses capacités d'anticipation et d'adaptation.

La mémoire n'est pas un simple enregistrement passif d'expérience, le souvenir d'un événement bâti autour de l'expérience perceptive et étroitement emprunt d'impressions et d'images qui reflètent nos interprétations de cet événement.

2.2.3.2 Mécanisme de la mémorisation

Les bases de la physiologie, jusqu'à présent, sont encore imprécises mais il y a des tentatives d'explication de ce phénomène.
Selon un article paru dans la revue Science & Vie [S&V 91] on admet que la représentation de chaque souvenir correspond à une activité unique au sein des

neurones; les neurones simultanément actifs forment un assemblage qu'on nomme *réseau.*

2.3 MODELISATION DU NEURONE ARTIFICIEL

2.3.1 DU NEURONE HUMAIN AU NEURONE ARTIFICIEL *(Approche de Mc Culloch & Pitts)*

Le premier modèle du neurone fut élaboré en 1943 par Mc Culloch & Pitts [Dav 90], se basant sur une analogie quasi directe avec le neurone biologique (cf 2.2). Ils ont décrit leur neurone formel par les critères suivants [Fre 92]:

1- Le neurone reçoit des signaux (qui peuvent être des potentiels) émis par d'autres neurones dont les sorties sont connectées à ses entrées.

2- Le neurone calcule la moyenne de ces signaux, pondérés par des coefficients appelés poids synaptiques (W_{ij}).

3- Si cette moyenne dépasse un certain seuil le neurone s'active et transmet un signal; sinon il ne transmet rien, on dit alors que le neurone a une activité tout ou rien.

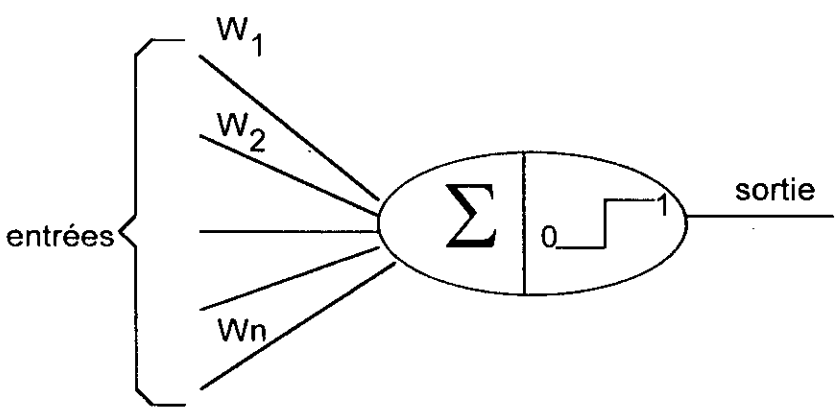


Figure 2.3.1: Neurone formel de Mc Culloch & Pitts

2.3.2 MODELE GENERAL DU NEURONE

Le neurone de Mc Culloch & Pitts a un comportement statique de ce fait les réseaux bâtis sur ce modèle (ou des variantes: on peut jouer sur la fonction d'activation, l'entrée, la sortie...) ne peuvent satisfaire les applications aux systèmes dynamiques. Le modèle général du neurone devrait donc être dynamique;

Il est caractérisé par:

1- Un sommateur pondéré.

2- Un système dynamique SISO.

3- Une fonction d'activation généralement non linéaire.

Explicitement on définit les éléments ci dessus de la manière suivante:

2.3.2.1 Sommateur pondéré

C'est un élément réalisant la somme pondérée -par des gains- des entrées du neurone. Il est régit par l'équation:

$$v_i(t) = \sum_{j=1}^{N} a_{ij} y_j(t) + \sum_{k=1}^{M} b_{ik} u_k(t) + w_i$$

où:

i: Indice du neurone en question.

$v_i(t)$: Sortie du sommateur pondéré du neurone i à l'instant t.

$y_j(t)$, j = 1..N: Sorties des N neurones connectées au neurone i (un neurone peut avoir un retour sur lui même).

a_{ij}: Poids synaptique reliant le neurone j au neurone i.

$u_k(t)$, k = 1..M: Entrées du réseau.

b_{ik}: Poids synaptique reliant l'entrée k au neurone i

w_i: Poids du terme de biais (son entrée est toujours égale à 1). Ce terme est ajouté pour des raisons de convergence [Fre 92].

2.3.2.2 Dynamique du neurone

Elle est généralement linéaire, et si on admet la notation suivante: $H(s) = \dfrac{X_i(s)}{V_i(s)}$

tels que:

$X_i(s) = L[x_i(t)]$, avec $x_i(t)$ sortie du sommateur pondéré du neurone i à l'instant t.

$V_i(s) = L[v_i(t)]$, avec $v_i(t)$ sortie du système dynamique du neurone i à l'instant t.

Alors cinq choix de H(s) existent [Nar 91]:

1) $H(s) = 1$

2) $H(s) = \frac{1}{s}$

3) $H(s) = \frac{1}{1 + Ts}$

4) $H(s) = e^{-sT}$

Les trois premiers choix sont des cas particuliers de

5) $H(s) = \frac{1}{\alpha_0 + \alpha_1 s}$

2.3.2.3 La fonction d'activation

Il existe plusieurs types de fonctions d'activation selon l'application à laquelle est dédié le réseau et aussi les contraintes relatives aux algorithmes d'apprentissage (toutefois c'est généralement une fonction monotone croissante et bornée).

Remarquons que même en biologie ces fonctions changent d'un niveau à autre, on peut citer comme exemple la fonction d'activation du système oculaire donnée par:

$$S(x) = \frac{1}{1 + e^{-\alpha x}}$$

qui est différente de celle des neurones du champ de vision qui, elle, est Gaussienne [Lee 88].

En ce qui concerne le choix de la fonction d'activation, tout dépend de l'application à laquelle est dédié le réseau. En effet le choix d'une fonction d'activation linéaire facilite l'apprentissage mais diminue la robustesse, alors que l'utilisation d'une fonction d'activation non linéaire augmente la robustesse et l'aptitude du réseau à apprendre des fonctions plus ou moins complexes; en contrepartie on perd en temps de calcul et on favorise l'instabilité du réseau [Kos2 92].

A titre d'exemples non exhaustifs on peut citer quelques types de fonction d'activation [Dav 90].

- ***Une fonction binaire à seuil:*** On s'arrange pour que sa forme soit telle qu'on puisse utiliser la fonction de Heaviside ou la fonction signe (figure 1.3.2).

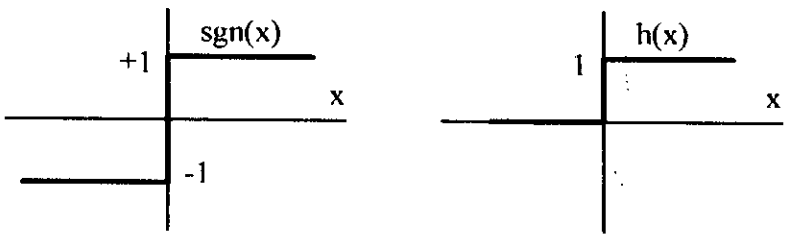


Figure 2.3.2: Fonction signe et de Heaviside

- ***Une fonction linéaire à seuil:***

$$S(x) = \begin{cases} x \text{ si } x \in [u\ \ v] \\ u \text{ si } x \geq u \\ v \text{ si } x \leq v \end{cases}$$

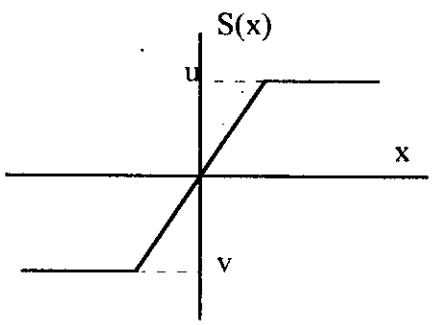


Figure 2.3.3: Fonction à seuil

- ***Une fonction sigmoïde:*** $f(x) = \alpha \dfrac{e^{\beta x} - 1}{e^{-\beta x} + 1} = \alpha . \tanh(\beta x)$

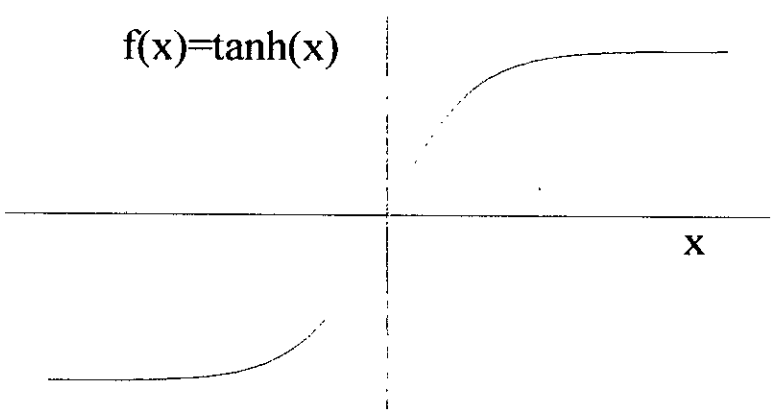


Figure 2.3.4: La fonction sigmoïde

Ou toute autre fonction généralement choisie croissante et bornée. Elle est bornée à cause de la nature tout ou rien du neurone (Voir caractéristiques du neurone biologique -cf. 2.2-).

En pratique la fonction sigmoïde est la plus utilisée car elle a des caractéristiques qui sont proches de la fonction signe (proche de la fonction d'activation du neurone humain), en plus de ses caractéristiques mathématiques, elle est dérivable ce qui nous permet d'utiliser certains algorithmes d'apprentissage.

2.4 RESEAUX DE NEURONES ARTIFICIELS

Les réseaux de neurones sont constitués à partir de plusieurs neurones en les connectant entre eux d'une certaine manière. C'est justement cette dernière associée au modèle du neurone et à l'apprentissage, qui définit la classe du neurone. En effet, il existe deux types de réseaux: statiques (H(s) = 1) et dynamiques (H(s) choisie parmis les quatre autres fonctions -cf. 2.3.2.2-)

2.4.1 CLASSIFICATION

On peut classer les réseaux neuronaux selon:

1- Le type de neurone utilisé: qui peut être statique ou dynamique.

2- L'architecture du réseau: il existe essentiellement deux manières pour connecter les neurones. Ce qui donne:

1- Les réseaux entièrement connectés[2]

Constitués de N neurones, dont chaque neurone est connecté aux (N-1) autres neurones et éventuellement à lui même. Les connections sont bidirectionnelles.

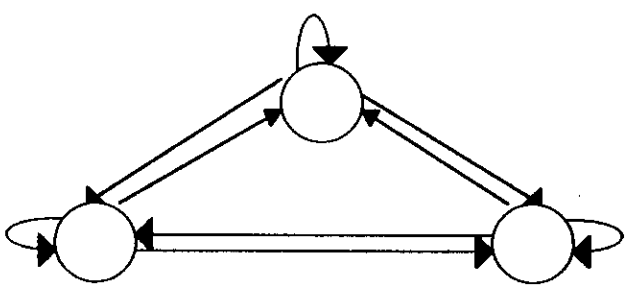


Figure 2.4.1: Réseau entièrement connecté (3 neurones)

[2] C'est l'architecture des réseaux dynamiques (récurrents): de tels réseaux donnent, à différents instants, différentes sorties pour une même entrée.

2- Les réseaux multicouches[3]

Ces réseaux disposent de deux couches principales une d'entrée et une autre de sortie reliées par le biais de couches de connexions modifiables dites cachées. Les neurones d'une même couche ne sont pas connectées entre eux.

Chaque neurone de la couche i reçoit des informations de la couche (i-1) et est connecté a tous les neurones de la couche (i+1).

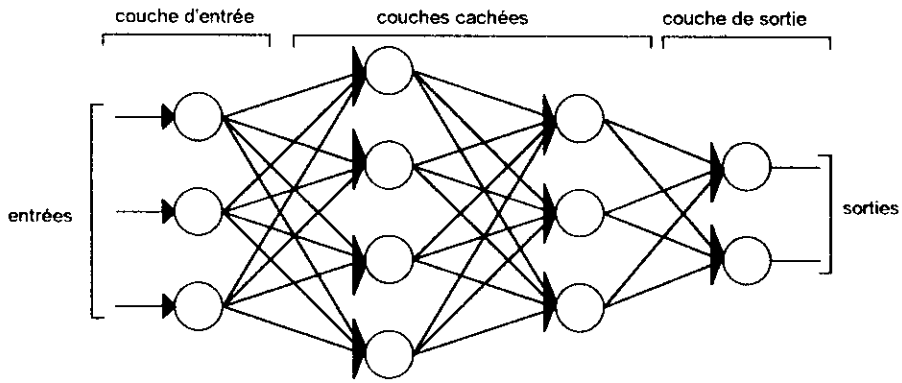


Figure 2.4.2: Réseau multicouche (2 couches cachées)

3- La méthode d'apprentissage: L'intérêt de l'apprentissage est d'organiser, classer, mémoriser et apprendre l'information, il se fait de deux manières différentes.

1- Apprentissage supervisé

Les poids synaptiques sont déterminés par le biais d'algorithmes qui consistent à minimiser l'erreur entre la sortie désirée et la sortie du réseau jusqu'à l'obtention d'une performance acceptable (erreur inférieure à un certain seuil).

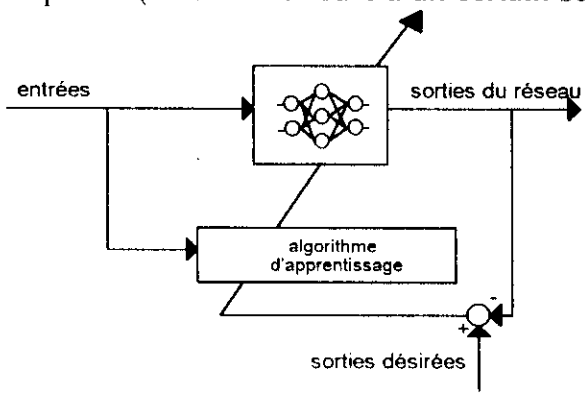


Figure 2.4.3: Apprentissage supervisé

[3] Les réseaux de neurones statiques adoptent cette architecture (neurone statque: H(s) = 1 et pas de bouclage synaptique fermé).

2- Apprentissage non supervisé

Dans ce cas la détermination des poids synaptiques n'est pas en fonction des erreurs, mais en présentant au réseau des informations groupés sous formes de classes à l'intérieur desquelles les données présentent des caractères communs.

Ainsi on peut classer les réseaux de neurones de la manière suivante:

- Réseau récurrent supervisé: Il présente une bonne capacité d'apprentissage mais il très peut utilisé, en effet pour ce modèle la récurrence n'affecte pas la stabilité ce qui a été montré par Almerda et Pineda [Widl 90].
- Réseau récurrent non supervisé: Hopfield.
- Réseau non récurrent supervisé: Réseau statique.
- Réseau non récurrent non supervisé: Kohonen [Dav 90].

2.5 CONCLUSION

Dans cette section nous avons montré le passage du neurone biologique au neurone artificiel. Celà en présentant quelques notions de neurobiologie, qui permettent la modélisation du neurone et également les réseaux de neurones artificiels. Tout en se penchant sur les caractéristiques du neurone et du réseau qui servent à la classification de ces derniers.

SECTION

3

ALGORITHMES D'APPRENTISSAGE

3.1 INTRODUCTION

L'homme n'est pas né avec toutes les connaissances nécessaires pour sa survie. Mais c'est son contact avec le milieu extérieur qui lui permet d'acquérir et de modifier ses connaissances, en vue d'une meilleure adaptation.

Il possède, donc, une préprogrammation naturelle qui lui assure l'apprentissage nécessaire à cette adaptation. Les réseaux de neurones étants "artificiels" n'ont pas cette caractéristique naturelle; il a fallu donc mettre au point des méthodes mathématiques (algorithmes) permettants de simuler la dite caractéristique naturelle qu'est l'apprentissage.

Dans cette section nous étudierons quelques algorithmes d'apprentissage parmi les plus notoires et les plus utilisés:

- BP: Backpropagation.
- FBP: Fast Backpropagation.
- ROM: Random Optimization Method.
- ELEANNE 7: Efficient Learning Algorithm for Neural NEtwork (Version: 7).

On a aussi présenté les notions de base de la théorie des réseaux de Hopfield.

3.2 BACKPROPAGATION

En 1969 Minsky et Papert [Min 69] ont démontré les limites du perceptron (voir Annexe A.2) et ont réussi par la même occasion à enterrer les recherches sur les réseaux monocouches, s'appuyant sur le fait que ceux ci ne possèdent pas des

techniques d'entraînement capables de donner de bons résultats face à des situations compliquées.

Il aura fallu dix ans au total pour que Werbos et plus tard Rumelhart [Dav 90] s'aperçoivent qu'il suffit d'ajouter des couches et d'appliquer ensuite, les règles de la rétropropagation. C'est ainsi que sont nés les réseaux de neurones multicouches, portant la Backpropagation au rang de la méthode d'apprentissage la plus utilisée.

3.2.1 DESCRIPTION GENERALE

On dit que l'algorithme converge si, après un certain nombre d'itérations, l'erreur sur les sorties est inférieure à un certain seuil qu'on s'est fixé. Le réseau doit être, alors, à même de donner des résultats satisfaisants et cela même pour des points non appris: C'est la généralisation.

Cet algorithme est appliqué au réseau de la manière suivante: on stimule le réseau par un vecteur d'entrée, l'information se propage à travers les couches cachées (voir figure 3.2.1), la sortie du réseau est recueillie et comparée à une réponse désirée. Ensuite on modifie les poids synaptiques par rétropropagation de l'erreur. L'apprentissage par Backpropagation est un apprentissage supervisé.

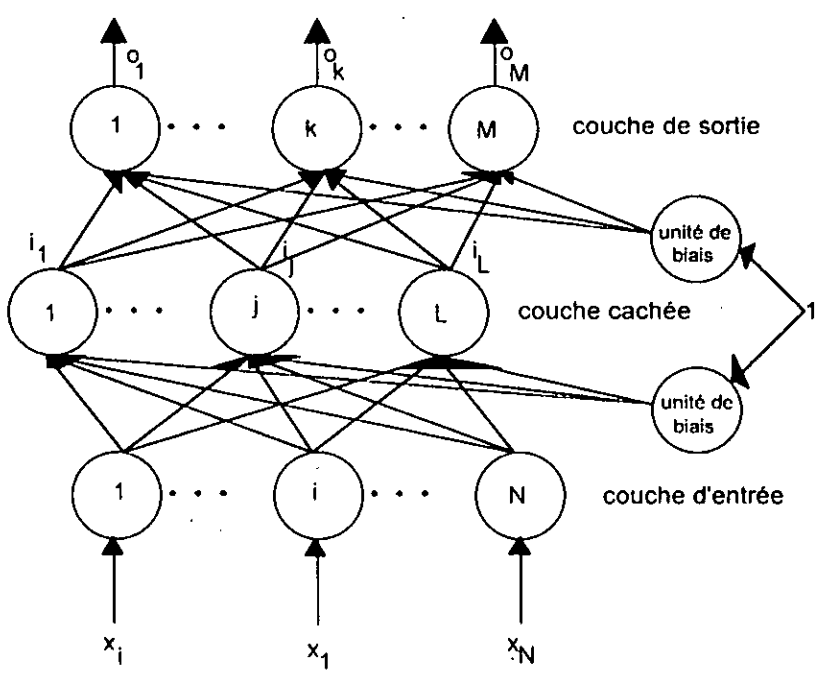


Figure 3.2.1: Structure d'un réseau à rétropropagation

3.2.2 AJUSTEMENT DES POIDS

Pour simplifier les notations et la présentation de l'algorithme on utilisera un réseau à 3 couches, la généralisation à un réseau de plus de 3 couches se fait aisément en considérant, pour une couche cachée n, la couche (n-1) comme couche d'entrée et la couche (n+1) comme couche de sortie.

3.2.2.1 Notations

$\mathbf{x}_l = (x_{l1}, x_{l2}, ..., x_{lN})$: vecteur d'entrée ou N est le nombre d'entrées et l l'indice de l'exemple (rappelons-le: un exemple est le couple (**x**, **y**) tel que **x** est le vecteur d'entrée et **y** le vecteur de sortie désirée).

$\mathbf{y}_l = (y_{l1}, y_{l2}, ..., y_{lM})$: vecteur de sortie désiré et M est le nombre de sorties.

$\mathbf{o}_l = (o_{l1}, o_{l2}, ..., o_{lM})$: vecteur de sortie du réseau en réponse à l'entrée $\mathbf{x}_l$.

$\mathbf{i}_l = (i_{l1}, i_{l2}, ..., i_{lL})$: vecteur des sorties des L neurones de la couche cachée.

w^o_{kj}: connexion synaptique entre le k^{eme} neurone de la couche de sortie et le j^{eme} neurone de la couche cachée.

w^h_{ji}: connexion synaptique entre le i^{eme} neurone de la couche d'entrée et le j^{eme} neurone de la couche cachée.

$w^o_{k\,L+1}$: poids d'entrée de biais du k^{eme} neurone de la couche de sortie (son entrée est toujours égale à 1).

$w^h_{j\,N+1}$: celui du j^{eme} neurone de la couche cachée.

net^o_{lk}: l'entrée totale du k^{eme} neurone de la couche de sortie relative au l^{eme} exemple.

net^h_{lj}: l'entrée totale du j^{eme} neurone de la couche cachée relative au l^{eme} exemple.

σ: fonction d'activation.

Ainsi on peut écrire les équations suivantes:

$$net^h_{lj} = \sum_{i=1}^{N+1} w^h_{ji} x_{li} \qquad (3.1)$$

$$i_{lj} = \sigma(net^h_{lj}) \qquad (3.2)$$

$$net^o_{lk} = \sum_{j=1}^{L+1} w^o_{kj} i_{lj} \qquad (3.3)$$

$$o_{lk} = \sigma(net^o_{lk}) \qquad (3.4)$$

3.2.2.2 Couche de sortie

A l'étape l (exemple l) on considère l'erreur totale:

$$E_l = \frac{1}{2}\sum_{k=1}^{M}(y_{lk} - o_{lk})^2 = \frac{1}{2}\sum_{k=1}^{M}\delta_{lk}^2$$

C'est la minimisation de cette erreur qui déterminera la manière de modifier les poids (soit en les augmentant soit en les diminuant). Pour cela on utilise le critère du gradient. Cependant pour simplifier les choses on considérera chaque composant du vecteur gradient ∇E_l séparément [Fre 92].

$$E_l = \frac{1}{2}(y_{lk} - o_{lk})^2 + \sum_{\substack{m=1 \\ m\neq k}}^{M}(y_{lm} - o_{lm})^2$$

$$\Rightarrow \quad \frac{\partial E_l}{\partial w_{kj}^o} = -\frac{\partial o_{lk}}{\partial w_{kj}^o}(y_{lk} - o_{lk}) + 0$$

d'après l'équation (3.4):

$$\frac{\partial E_l}{\partial w_{kj}^o} = -(y_{lk} - o_{lk})\frac{\partial \sigma}{\partial (net_{lk}^o)}\frac{\partial (net_{lk}^o)}{\partial w_{kj}^o} \tag{3.5}$$

$$\frac{\partial (net_{lk}^o)}{\partial w_{kj}^o} = \frac{\partial}{\partial w_{kj}^o}(w_{kj}^o i_{lj} + \sum_{\substack{m=1 \\ m\neq l}}^{l,+1} w_{km}^o) = i_{lj}$$

ce qui donne:

$$-\frac{\partial E_l}{\partial w_{kj}^o} = (y_{lk} - o_{lk})\sigma'(net_{lk}^o)i_{lj}$$

Les poids de la couche de sortie sont ajustés par une variation proportionnelle au gradient inversé de signe:

$$w_{kj}^o(t+1) = w_{kj}^o(t) - \eta\frac{\partial E_l}{\partial w_{kj}^o}$$

où η est le paramètre d'apprentissage rajouté pour des considérations pratiques [Fre 92].

Finalement $$w_{kj}^o(t+1) = w_{kj}^o(t) + \eta(y_{lk} - o_{lk})\sigma'(net_{lk}^o)i_{lj}$$

3.2.2.3 Couche cachée

L'ajustement des poids de la couche cachée se fait de la même façon que pour la couche de sortie. Le seul problème qui se pose est celui des sorties désirées de cette couche; on y remédie en admettant qu'elles sont les mêmes que celles de la couche de sortie.

Aussi pour cette couche la dérivation de E_l se fait par rapport à w_{ij}^h puisque justement c'est w_{ij}^h qu'on veut ajuster.

$$\begin{aligned} E_l &= \frac{1}{2}\sum_{k=1}^{M}(y_{lk} - o_{lk})^2 \\ &= \frac{1}{2}\sum_{k=1}^{M}(y_{lk} - \sigma(net_{lk}^o))^2 \\ &= \frac{1}{2}\sum_{k=1}^{M}(y_{lk} - \sigma(\sum_{j=1}^{L+1} w_{kj}^o i_{lj}))^2 \end{aligned}$$

d'où

$$\begin{aligned} \frac{\partial E_l}{\partial w_{ij}^h} &= \frac{1}{2}\sum_{k=1}^{M}\frac{\partial}{\partial w_{ij}^h}(y_{lk} - o_{lk})^2 \\ &= -\sum_{k=1}^{M}(y_{lk} - o_{lk})\frac{\partial o_{lk}}{\partial (net_{lk}^o)}\frac{\partial (net_{lk}^o)}{\partial i_{lj}}\frac{\partial i_{lj}}{\partial (net_{lj}^h)}\frac{\partial (net_{lj}^h)}{\partial w_{ij}^h} \end{aligned}$$

de (3.4) on a $\dfrac{\partial o_{lk}}{\partial (net_{lk}^o)} = \sigma'(net_{lk}^o)$.

de (3.3) $\dfrac{\partial (net_{lk}^o)}{\partial i_{lj}} = w_{kj}^o$.

de (3.2) $\dfrac{\partial i_{lj}}{\partial (net_{lj}^h)} = \sigma'(net_{lj}^o)$.

et de (3.1) $\dfrac{\partial (net_{lj}^h)}{\partial w_{ij}^h} = x_{li}$.

de la même façon que pour la couche de sortie:

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) - \eta \frac{\partial E_l}{\partial w_{ji}^{h}}$$

d'où

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \eta\sigma'(net_{lj}^{h})x_{li}\sum_{k=1}^{M}(y_{lk} - o_{lk})\sigma'(net_{lk}^{o})w_{kj}^{o}$$

3.2.3 ALGORITHME

Comme on peut le remarquer cet algorithme nécessite une fonction d'activation dérivable.La sigmoïde est choisie car elle représente une bonne approximation de la fonction seuil et est en même temps, dérivable.

Enfin on énonce l'algorithme de Backpropagation:

Début

Initialiser les poids à des valeurs aléatoires , donner η et E_0

1: $E = 0$, $l = 0$

2: $l = l + 1$

Donner $\mathbf{x}_l$ et $\mathbf{y}_l$

$$i_j = \sigma(\sum_{i=1}^{N+1} w_{ji}^{h} x_i)$$

$$o_k = \sigma(\sum_{j=1}^{L+1} w_{kj}^{o} i_j)$$

$$\varepsilon_k^{o} = (y_k - o_k)\sigma'(\sum_{j=1}^{L+1} w_{ji}^{o} i_j)$$

$$\varepsilon_j^{h} = \sigma'(\sum_{i=1}^{N+1} w_{ji}^{h} x_i)\sum_{k=1}^{M} \varepsilon_k^{o} w_{kj}^{o}$$

$$w_{kj}^{o} \leftarrow w_{kj}^{o} + \eta\varepsilon_k^{o} i_j$$

$$w_{ji}^{h} \leftarrow w_{ji}^{h} + \eta\varepsilon_j^{h} x_i$$

$$E \leftarrow E + \frac{1}{2}\sum_{k=1}^{M}(y_k - o_k)^2$$

Si $l < P$ aller à 2 (P: nombre d'exemples)

Si $E > E_0$ aller à 1

Fin

3.2.4 UNE APPLICATION AVEC BP

La Backpropagation est utilisée dans plusieurs domaines tels que la reconnaissance de formes, la compression d'images, l'identification et commande des systèmes etc...

L'une des plus célèbres est Nettalk ou l'ordinateur qui lit... et qui parle! [SVM 90].

Nettalk, ou le réseau qui sait lire

Pour faire lire un texte à un ordinateur il suffit généralement d'un scanner, d'un logiciel, d'un circuit de reconnaissance de formes et d'un circuit de synthèse, mais la prononciation reste quand même mal.

Le problème était soumis à un réseau de neurones. Au lieu de déchiffrer bêtement les lettres une à une, il sélectionne des mots pour évaluer le contexte et déterminer quel phonème il doit prononcer. Au début la lecture ressemble à un balbutiement, mais au fur et à mesure qu'elle progresse, le système mémorise les configurations et parvient, au bout de plusieurs présentations, à lire la page entière et même des pages contenant des mots inconnus (pour lui).

Terry Sejnowski, de l'université de Baltimore et Charles Rosenberg, de Princentron ont simulé un réseau de neurones sur Vax 780, pour lui apprendre à prononcer phonétiquement un texte sans qu'on lui ait programmé la moindre règle.

Ce réseau est constitué de 300 neurones et 18000 connexions réparties sur 3 couches:

1- Couche d'entrée: On trouve 7 groupes de 29 neurones, cette couche est connectée à un scanner qui balaie le texte à travers une grille. Grâce à lui le réseau identifie une suite de caractères pour pouvoir prononcer correctement la lettre centrale en fonction du contexte, c'est à dire des trois caractères précédents et des trois caractères suivants.

2- Couche cachée: On trouve 80 neurones et selon les deux scientifiques Sejnowski et Rosenberg. dans la couche cachée un groupe détecte les voyelles, un autre les consonnes et les classe selon les règles de la phonétique: hauteur d'une voyelle (haute, moyenne, basse), ponctuation (silence, pause, arrêt, accentuation).

3- Couche de sortie: Elle est composée de 26 neurones correspondants aux 26 phonèmes possibles en sortie. Après avoir calculer les poids par l'algorithme de Backpropagation, le réseau fournie aux neurones de sortie le phonème qui correspond à la lettre centrale.

Les expériences de Sejnowski et Rosenberg ont prouvé que le taux de reconnaissance atteignait 95% après la présentation de 50000 mots, avec un temps d'apprentissage de 12 heures sur Vax 780.

3.3 FAST BACKPROPAGATION

L'un des inconvénients de l'apprentissage des réseaux de neurones sur calculateurs numériques, est le temps énorme que met le réseau pour converger vers une performance acceptable, surtout si les fonctions à apprendre sont complexes (non- linéarité dur). Ceci a poussé les chercheurs ces dernières années à essayer de développer des algorithmes rapides et efficaces. Parmi ces algorithmes citons celui de la rétropropagation rapide (FBP: Fast Back Propagation) [Kar 93].

3.3.1 DESCRIPTION GENERALE

Le principe d'apprentissage dans cet algorithme est identique à celui de Backpropagation, la différence réside dans le changement de la fonction objective (fonction à minimiser) en vue d'accélérer la convergence après les 1ères itérations.

La déscription est la même que pour Backpropagation sauf que la fonction d'erreur a changé (pour la même architecture du réseau -figure 3.2.1-).

3.3.1.1 La fonction objective

Dans l'algorithme de Backpropagation la fonction objective était:

$$E_l = \frac{1}{2}\sum_{k=1}^{M}(y_k - o_k)^2$$

(Pour les notations, on prendra celles de l'algorithme de Backpropagation).

Lors de l'apprentissage avec l'algorithme de Backpropagation, on a remarqué que la diminution de l'erreur est rapide pendant les 1ères itérations mais devient de plus en plus lente par la suite, pour palier à ce problème il a été proposée une nouvelle fonction objective:

$$G(\lambda) = \lambda E + (1-\lambda)E'$$

$$G(\lambda) = \lambda \sum_{l=1}^{P}\sum_{k=1}^{M}\Phi_2(e_{kl}) + (1-\lambda)\sum_{l=1}^{P}\sum_{k=1}^{M}\Phi_1(e_{kl})$$

où:

$e_{kl} = (y_{kl} - o_{kl})$

$\Phi_2(x) = \frac{1}{2}x^2$

Φ_1 est une fonction définit positive, convexe, continue et dérivable

$\lambda \in [0, 1]$

De cette façon on définit une variété de fonctions objectives entre deux extrêmes correspondants à λ=0 et λ=1. Pour λ=1 on trouve la fonction objective de Backpropagation (diminuant rapidement aux 1ères itérations).

L'introduction de Φ_1 dans la fonction objective a l'intérêt de permettre une diminution plus ou moins rapide de l'erreur même au delà des 1ères itérations; $G(\lambda)$ étant l'approximation de l'erreur absolue (écart entre les poids recherchés et les poids actuels).

Il est à noter que dans le cas d'un réseau à sorties binaires (cas qu'on a à traiter), il est permis de prendre -1 et +1 comme sorties désirées (la fonction d'activation des neurones devenant la tangente hyperbolique).

On définit la distance de Hamming par l'expression:

$$d[\mathbf{y}_l, \mathbf{o}_l] = \sum_{k=1}^{M} (1 - y_{kl} o_{kl})$$

On pose

$$E' = \sum_{l=1}^{P} d[\mathbf{y}_l, \mathbf{o}_l] = \sum_{l=1}^{P} \sum_{k=1}^{M} (1 - y_{kl} o_{kl})$$

Comme y_{kl}^2=1 on peut écrire:

$$\Phi_1(e_{kl}) = y_{kl}(y_{kl} - o_{kl})$$

Φ_1 atteint son minimum si o_{kl} approche de y_{kl}. Au début l'apprentissage du réseau est basé sur la minimisation de l'erreur quadratique, quand λ approche le zéro (λ varie de 1 à 0 au cours de l'apprentissage) il se base sur la minimisation de la distance de Hamming.

Supposons que o_{kl} et y_{kl} prennent la valeur -1 ou 1 (fonction d'activation très raide) ceci nous donne:

$$\frac{1}{2}(y_{kl}-o_{kl})^2=\frac{1}{2}(y_{kl}^2-2y_{kl}o_{kl}+o_{kl}^2)$$
$$=1-y_{kl}o_{kl}$$
$$=\Phi_1(e_{kl})$$

d'où

$$E'=\sum_{l=1}^{P}\sum_{k=1}^{M}(1-y_{kl}o_{kl})=\frac{1}{2}\sum_{l=1}^{P}\sum_{k=1}^{M}(y_{kl}-o_{kl})^2$$

Φ_1 est alors l'erreur quadratique quand o_{kl} approche de y_{kl}

Finalement on peut écrire la fonction objective du réseau à sortie binaire comme suit:

$$G(\lambda)=\sum_{l=1}^{P}\sum_{k=1}^{M}y_{kl}(y_{kl}-o_{kl})+\frac{1}{2}\lambda\sum_{l=1}^{P}\sum_{k=1}^{M}(y_{kl}-o_{kl})^2$$

3.3.1.2 Comment varie λ?

Au début l'apprentissage est basée sur la minimisation de l'erreur quadratique (λ=1), mais après les 1ères itérations on doit changer la fonction d'activation ($\lambda \rightarrow 0$) afin d'éliminer l'effet de E et favoriser celui de E'.

Dans chaque itération λ doit avoir une valeur en fonction de l'erreur, i.e: λ=f(E), il existe plusieurs méthodes pour déterminer cette fonction, l'une d'elles est proposée dans [Kar 93]:

quand $E >> 1 \Rightarrow \lambda \approx 1$

d'où pour un nombre positif entier n (n>1)

$$\lambda=\exp(-\frac{1}{E^n})$$

d'autre part si $E<<1 \Rightarrow \lambda=\exp(-\frac{1}{E^n})\approx 0$

donc on peut prendre:

$$\lambda=\exp(-\frac{\mu}{E^n})$$

μ: nombre positif réel, s'il est grand on remarque des oscillations de l'erreur, dans ce cas il faut le diminuer [Kar 93].

3.3.2 ALGORITHME

Début

Initialiser les poids à des valeurs aléatoires . donner η, μ et E_0

$\lambda = 1$

1: $E = 0$, $l = 0$

2: $l = l + 1$

Donner $\mathbf{x}_l$ et $\mathbf{y}_l$

$$i_j = \sigma\left(\sum_{i=1}^{N+1} w_{ji}^{h} x_i\right)$$

$$o_k = \sigma\left(\sum_{j=1}^{L+1} w_{kj}^{o} i_j\right)$$

$$\varepsilon_k^{o} = (y_k - \lambda o_k)\sigma'\left(\sum_{j=1}^{L+1} w_{ji}^{o} i_j\right)$$

$$\varepsilon_j^{h} = \sigma'\left(\sum_{i=1}^{N+1} w_{ji}^{h} x_i\right)\sum_{k=1}^{M} \varepsilon_k^{o} w_{kj}^{o}$$

$$w_{kj}^{o} \leftarrow w_{kj}^{o} + \eta \varepsilon_k^{o} i_j$$

$$w_{ji}^{h} \leftarrow w_{ji}^{h} + \eta \varepsilon_j^{h} x_i$$

$$E \leftarrow E + \frac{1}{2}\sum_{k=1}^{M} (y_k - o_k)^2$$

Si $l < P$ aller à 2 (P: nombre d 'exemples)

$$\lambda = \exp\left(-\frac{\mu}{E^2}\right) \quad (n = 2)$$

Si $E > E_0$ aller à 1

Fin

3.3.3 PROBLEMES ET CONSIDERATIONS PRATIQUES

Dans cette partie on présentera les différents problèmes rencontrés lors de l'utilisation de l'algorithme de BP ou de FBP (qui sont du même genre pour ces deux algorithmes), ensuite on donnera des solutions possibles à ces problèmes.

3.3.3.1 Convergence

Il n'existe pas encore une preuve théorique pour la convergence des réseaux, mais des efforts dans ce sens ont donné quelques résultats tel que le théorème de Kolmogorov-Sprecher [Kol 73] (théorème resté sans démonstration recevable) qui annonce que "n'importe qu'elle fonction continue et réelle peut être approximée par un réseau de neurones à couches".

Cependant en pratique l'algorithme converge toujours. Mais parfois vers des solutions non satisfaisantes et cela à cause des minimums locaux.

3.3.3.2 Minimums locaux

C'est le problème le plus rencontré dans l'apprentissage, en effet la surface de la fonction d'erreur n'est pas toujours convexe; l'algorithme converge vers un fond qui n'est pas le plus bas: il en résulte que l'erreur peut être relativement importante. C'est le minimum local. solution:

1- Changer le pas de correction: on peut donner des pas différents à chaque couche du réseau.

2- Changer les poids initiaux: ne jamais donner la même valeur à tous les poids car le réseau ne pourra rien apprendre, on peut simplement le démontrer en résonnant sur la surface de l'erreur [Fre 92].

3- Changer les fonctions d'activation: on peut donner une fonction d'activation propre à chaque couche, ou changer le gain de la fonction d'activation. Pour un réseau à sorties binaires la fonction d'activation des neurones doit être proche de la fonction de Heaviside.

4- Augmenter le nombre de neurones dans les couches cachées.

5- Ajouter les termes de biais.

6- Augmenter le nombre de couches cachées.

Normalement avec toutes ces dispositions, on peut sortir du minimum local.

3.3.3.3 Pas de correction

Il présente un dilemme s'il est trop grand, il peut entraîner des oscillations dans le cas contraire le temps d'apprentissage peut être infini.

Solution: faire varier le pas en commençant par une grande valeur et la diminuant au fur et à mesure qu'on se rapproche de la solution.

3.3.3.4 Dimension du réseau

Le nombre de couches cachées et de neurones peut être déterminé par tâtonnement sinon par les trois règles suivantes:

Règle 1: plus les exemples sont complexes plus on augmente le nombre de neurones.

En pratique il n'existe pas de règles rigoureuses pour dimensionner un réseau, cependant certains chercheurs ont proposé donnent des règles à partir de leurs expériences pratiques tels que les deux règles de Baum et Hassler [Bau 89]:

Règle 2: elle est applicable pour les réseaux à une seule couche cachée, le nombre minimal de neurones H dans la couche cachée est égal à:

$$H = \frac{C}{10(M+N)}$$

où C est le nombre de lignes ou de colonnes de la matrice des exemples (voir pour M et N -figure 3.2.1-).

3- Le nombre de liaisons synoptiques W est donné par la relation:

$$\frac{MP}{1+\log_2(P)} \leq W \leq M\left(\frac{P}{N}+1\right)(N+M+1)+M$$

où P est le nombre d'exemple.

Ou bien utiliser l'algorithme d'ALADIN [Kar 93] (Algorithm for Learning and Architecture DetermINation): l'idée de cet algorithme est de commencer au début avec un grand nombre de neurones dans les couches cachées, ensuite calculer la sensibilité de chaque neurone, si elle est constante (le neurone n'a pas d'effet sur le réseau) on élimine le neurone (voir algorithme à l'annexe A.3).

3.3.3.5 Saturation

Elle présente un dilemme relatif au pas de correction:

petit pas implique temps d'apprentissage infini, grand pas implique saturation.

Solution: poids initiaux entre -0.5 et 0.5.

3.3.3.6 Le fichier d'exemples

Si la fonction d'activation de la couche de sortie est sigmoïdale, alors on doit mettre à l'échelle les sorties. Comme la sigmoïde est bornée entre -1 et 1 (ou 0 et 1), les sorties du réseau varieront entre ces deux bornes mais ne les atteindront jamais. Alors, il faut normaliser les sorties entre -0.9 et 0.9 (ou entre 0.1 et 0.9), car en dehors de cet intervalle, la derivé de la fonction d'activation étant presque nulle, il n'y aura pas de variation des poids de la couche de sortie et donc pas d'apprentissage: on dit que le réseau est saturé!.

3.4 ROM

ROM abréviation de: Random Optimization Method (Méthode d'Optimisation Aléatoire: MOA), c'est une méthode d'optimisation basée sur des techniques aléatoires utilisée pour trouver l'extremum d'une fonction et adaptée à l'apprentissage des réseaux de neurones artificiels.

L'un des plus grands problèmes qui se pose lors de l'utilisation de l'algorithme de BP ou de FBP est de tomber dans un minimum local. Dans cette partie on présentera une méthode pour la recherche du minimum global de la fonction d'erreur qui a été proposée par Matayas en 1969 et modifiée par Solis & Wets en 1981 [Nor 89], [Sol 84].

3.4.1 QUAND DOIT-ON UTILISER ROM ?

Cette méthode est utilisée quand:

1- Les caractéristiques de la fonction à apprendre au réseau sont difficiles à calculer.

2- On veut trouver un minimum global sachant que la fonction objective (fonction d'erreur) présente des minimums locaux.

3- La dimension du réseau devient grande.

4- Temps d'apprentissage trop grand dans le cas d'utilisation d'autres algorithmes.

3.4.2 DESCRIPTION GENERALE

Cette technique consiste à ajouter aux poids une valeur aléatoire, si la nouvelle erreur est inférieure à la précédente prendre les nouveaux poids sinon repartir avec les anciens et ajouter une autre valeur aléatoire.

3.4.3 ALGORITHME

Notons par

f: la fonction objective (fonction à minimiser)

x(k): le vecteur cherché qui minimise f à la k^{eme} itération

g(k): le vecteur aléatoire Gaussien

b(k): le vecteur des moyennes du bruit Gaussien

Ainsi on peut dresser l'algorithme de ROM modifié [Nor 89].

Choisir un vecteur initial **x**(0), poser **b**(0)= 0 pour k= 0

1: Générer un vecteur aléatoire Gaussien **g**(k)

Si f(**x**(k) + **g**(k))< f(**x**(k))

poser **x**(k+1) = **x**(k) + **g**(k)

et **b**(k+1) = 0.4**g**(k)+0.2**b**(k)

Si f(**x**(k) + **g**(k))≥f(**x**(k)) et

f(**x**(k) - **g**(k))<f(**x**(k))

poser **x**(k+1) = **x**(k) - **g**(k)

et **b**(k+1) = **b**(k) - 0.4**g**(k)

Sinon poser **x**(k+1) = **x**(k)

et **b**(k+1) = 0.5**b**(k)

Si l'extremum est satisfaisant: arrêter le calcul sinon:

k = k+1 et aller à 1

En appliquant cet algorithme pour l'apprentissage des réseaux de neurones, notre objectif était de trouver le vecteur de poids **w** qui minimise la fonction $E(\mathbf{w}) = \sum_l E_l(\mathbf{w})$ ce qui nous ramène à poser **w** →**x** et E(**w**) →f(**x**).

3.4.3.1 Génération d'un bruit Gaussien

Il existe plusieurs méthodes numériques pour la génération d'un bruit Gaussien, l'une d'elles est proposé dans [Kun 84]

1- Génération des signaux pseudo-aléatoires

$$x(kT) = [ax(k)] \bmod p \qquad (3.6)$$

où a et p sont des entiers avec a racine primitive de p.

Tout signal généré de la relation (3.6) est périodique et de période p-1.
Le signal

$$x'(k) = x(k)/p$$

a pour densité de probabilité la loi uniforme.

2- Génération des signaux possédants d'autres densité de probabilité

A partir d'un signal pseudo aléatoire ayant une loi de distribution on peut générer d'autres signaux qui ont d'autres loi de distribution:

Exemple

$$y(k) = \sqrt{2v^2 \ln(1/x(k))}$$

Si x(k) suit la loi uniforme alors y(k) suit la loi de Rayleigh, de plus

$$z(k) = y(k)\cos(2\pi x(k+1))$$

suit la loi de Gauss ou la loi normale.

3- Génération d'un bruit Gaussien

La suite $x1(t) = x(t)/(2^n - 1)$ suit la loi uniforme [0, 1]
avec $x(k+1) = [ax(k)] \bmod (2^n - 1)$ où
a = 131, x(0) = 12375 et n = 16.
posons

$$R(k) = \sqrt{2v^2 \ln(1/x1(k))}$$

Le signal

$$z(k)=R(k)\cos(2\pi x1(k+1))+b$$

est un bruit Gaussien de moyenne b et de variance v.
C'est " une " expression numérique du bruit Gaussien.

3.4.4 PROBLEMES ET CONSIDERATIONS PRATIQUES

1- La variance

v est généralement prise entre 0 et 1 [Nor 89]: et il faut la diminuer au cours d'apprentissage: on peut poser par exemple v=0.1/k. k: nombre d'itérations.

2- Temps d'apprentissage

L'un des inconvénients ce cette méthode est le temps d'apprentissage qu'elle prend, pour y remédier on peut faire apprendre le réseau par BP et ensuite prendre les poids ainsi obtenus comme poids initiaux pour démarrer la ROM: cela diminue largement le temps d'apprentissage [Hen 94].

3.5 ELEANNE 7

En voulant toujours améliorer la rapidité et rendre efficace (obtenir des résultats satisfaisants en fonction du temps d'apprentissage) les algorithmes d'apprentissage, d'autres méthodes ont été proposé dans [Kar 93], l'une d'elle est l'Efficient Learning Algorithm for Neural NEtwork: version 7 (ELEANNE 7)ý.

Dans cette partie on va expliciter l'algorithme pour un seul réseau à une seule couche cachée (on pourra par la suite généraliser à un réseau multicouche).

3.5.1 DESCRIPTION GENERALE

L'idée est basée sur la méthode de Newton-Raphson dont la loi d'adaptation pour la couche de sortie est (pour les indices on prend les mêmes que ceux de BP):

$$\mathbf{w}_k^o(t+1) = \mathbf{w}_k^o(t) - \alpha \mathbf{H}^{-1}(t)\Big|_{\mathbf{w}_k^o=\mathbf{w}_k^o(t)} \cdot \frac{\partial E_k(t)}{\partial \mathbf{w}_k^o}\Big|_{\mathbf{w}_k^o=\mathbf{w}_k^o(t)}$$

où:

H: Matrice Hessienne.

$E_k = \frac{1}{2}\sum_{l=1}^{P}(y_{lk} - o_{lk})^2$. P: nombre d'exemples.

$$\frac{\partial E_k}{\partial \mathbf{w}_k} = \left[\frac{\partial E_k}{\partial w_{k0}} \quad \frac{\partial E_k}{\partial w_{k1}} \cdots \frac{\partial E_k}{\partial w_{kl}}\right]$$

$$= \sum_{l=1}^{P} e_{lk}\mathbf{b}_{lk}$$

avec

$e_{lk} = y_{lk} - o_{lk}$

et $\mathbf{b}_{lk} = \left[\frac{\partial e_{lk}}{\partial w_{k0}^o} \quad \frac{\partial e_{lk}}{\partial w_{k1}^o} \cdots \frac{\partial e_{lk}}{\partial w_{kl}^o}\right]$

Ce qui nous permet d'écrire

$$\frac{\partial E_k(t+1)}{\partial \mathbf{w}_k^o}\Big|_{\mathbf{w}_k^o=\mathbf{w}_k^o(t)} = \frac{\partial E_k(t)}{\partial \mathbf{w}_k^o}\Big|_{\mathbf{w}_k^o=\mathbf{w}_k^o(t)} + e_{Pk}(t)\mathbf{b}_{Pk}(t)$$

$e_{kP}(t)$ et $\mathbf{b}_{kP}(t)$ sont évalués à $\mathbf{w}_k^o = \mathbf{w}_k^o(t)$.

En supposant que $E_k(t)$ est minimisée à $\mathbf{w}_k^o = \mathbf{w}_k^o(t)$:

$$\left.\frac{\partial E_k(t)}{\partial \mathbf{w}_k^o}\right|_{\mathbf{w}_k^o=\mathbf{w}_k^o(t)} = 0$$

$$\Rightarrow \quad \left.\frac{\partial E_k(t+1)}{\partial \mathbf{w}_k^o}\right|_{\mathbf{w}_k^o=\mathbf{w}_k^o(t)} = e_{Pk}(t)\mathbf{b}_{Pk}(t)$$

Donc

$$\mathbf{w}_k^o(t+1) = \mathbf{w}_k^o(t) - \alpha \mathbf{H}_k^{-1}(t+1)\Big|_{\mathbf{w}_k^o=\mathbf{w}_k^o(t)} \cdot e_{Pk}(t)\mathbf{b}_{Pk}(t)$$

L'expression de $H_k(t+1)$ est :

$$\mathbf{H}_k(t+1) = \sum_{l=1}^{P} \frac{\partial e_{lk}}{\partial \mathbf{w}_k^o} \cdot \frac{\partial e_{lk}}{\partial \mathbf{w}_k^{oT}} + \sum_{l=1}^{P} \frac{\partial^2 e_{lk}}{\partial \mathbf{w}_k^o \partial \mathbf{w}_k^{oT}} e_{lk}$$

Lorsqu'on est proche du minimum le 2eme terme peur être négligé et

$$\mathbf{H}_k(t+1) = \sum_{l=1}^{P} \mathbf{b}_{lk}\mathbf{b}_{lk}^T$$

$$= \sum_{l=1}^{P} c_{lk}\mathbf{i}_l\mathbf{i}_l^T$$

dont $c_{lk} = \sigma'(o_{lk})$.

On a

$$\mathbf{H}_k^{-1}(t+1) = (\mathbf{H}_k^{-1}(t) + \mathbf{i}_l^h(t)\mathbf{i}_l^{h\,T}(t))^{-1}$$

D'après le lemme d'inversion des matrices et en posant $\mathbf{P}_k(t) = \mathbf{H}_k^{-1}(t)$,

$$\mathbf{P}_k(t+1) = \mathbf{P}_k(t) - c_{lk}(t)(1 + c_{lk}(t+1)\mathbf{i}_l^T(t+1)\mathbf{P}_k(t)\mathbf{i}_l(t+1)) \cdot \mathbf{P}_k(t)\mathbf{i}_l(t+1)\mathbf{i}_l^T(t+1)\mathbf{P}_k(t)$$

Et finalement on aura:

$$\mathbf{w}_k^o(t+1) = \mathbf{w}_k^o(t) + \alpha\varepsilon_k^o(t+1)\mathbf{P}_k(t+1)\mathbf{i}_l(t+1)$$

avec $\varepsilon_k^o(t+1) = \sigma'(o_{lk}(t+1))e_{lk}(t+1)$

Pour l'adaptation des poids de la couche cachée, on utilise la règle de la descente la plus raide (LMS: Least Mean Square) voir [Fre 92]; la loi d'adaptation est:

$$\mathbf{w}_j^h(t+1) = \mathbf{w}_j^h(t) + \alpha\varepsilon_j^h\mathbf{x}$$

avec $\varepsilon_j^h = \sigma'(\mathbf{i}_j)\sum_{k=1}^{M} \varepsilon_k^o \mathbf{w}_{kj}^o$

Pour améliorer encore la rapidité de l'algorithme on calcul la moyenne

$$\overline{\mathbf{H}}(t) = \frac{1}{M}\sum_{k=1}^{M}\mathbf{H}_k(t) = \sum_{l=1}^{P}\overline{c}_l \mathbf{i}_l \mathbf{i}_l^T$$

où $\overline{c}_l = \frac{1}{M}\sum_{k=1}^{M}\sigma'(o_{lk})$.

Ce calcul permet de réduire le nombre de matrices $\mathbf{H}_k(t)$ calculées à une seule matrice moyenne.

Enfin on peut annoncer l'algorithme de ELEANNE 7.

3.5.2 ALGORITHME

Début

Initialisation des poids par des valeurs aléatoires

Donner α, η et E_0.

1: $E = 0$

$\mathbf{P} = (\frac{1}{\eta})\mathbf{I}$

$l = 0$

2: $l \leftarrow l + 1$

Donner $\mathbf{x}_l$ et $\mathbf{y}_l$

$i_j = \sigma(\sum_{i=1}^{N+1} w_{ji}^h x_i)$

$o_k = \sigma(\sum_{j=1}^{L+1} w_{kj}^o i_j)$

$\overline{c} = \frac{1}{M}\sum_{k=1}^{M}\sigma'(o_k)$

$\overline{\delta} = \overline{c}(1 + \overline{c}\mathbf{i}^T\overline{\mathbf{P}}\mathbf{i})^{-1}$

$\overline{\mathbf{P}} = \overline{\mathbf{P}} - \overline{\delta}\overline{\mathbf{P}}\mathbf{i}\mathbf{i}^T\overline{\mathbf{P}}$

$\varepsilon_k^o = \sigma'(o_k)(y_k - o_k)$

$\mathbf{w}_k^o \leftarrow \mathbf{w}_k^o + \alpha\varepsilon_k^o\overline{\mathbf{P}}\mathbf{i}$

$\varepsilon_i^h = \sigma'(i_j)\sum_{k=1}^{M}\varepsilon_k^o w_{ki}^o$

$\mathbf{w}_i^h \leftarrow \mathbf{w}_i^h + \alpha\varepsilon_i^h\mathbf{x}$

$E \leftarrow E + \frac{1}{2}\sum_{k=1}^{M}(y_k - o_k)^2$

Si $l < P$ aller à 2

Si $E > E_0$ aller à 1

Fin

3.6 RESEAUX DE HOPFIELD

Grâce à leur développeur (le physicien J.J. Hopfield du California Institute of Technology), ces réseaux ont joué un rôle important dans la résurrection des R.N.A[1] (rappelons-le les R.N.A. ont été enterrés avec le Perceptron en 1969 par Minsky et Papert [Min 69], et déterrés par un article de J.J. Hopfield paru en 1982 [Hop 82]). En plus, ils sont bien adaptés à l'implémentation électronique, car d'architecture simple et de convergence rapide [Hop 84].

Un réseau de Hopfield d'une seule couche entièrement connectée peut être utilisé, entre-autres, comme une mémoire associative[2], un classificateur de données et un circuit d'optimisation [Tan 86]. Notons au passage que la conversion A/D est, justement, un problème d'optimisation.

3.6.1 MODELE DU NEURONE

Pour le cas discret on retrouve le neurone de Mc Culloch & Pitts. Hopfield utilisait des sorties binaires (0,1), mais dorénavant on travaillera avec des sorties bipolaires (-1,1). Ceci est, d'ailleurs, équivalent.
Mais pour le cas continu, le modèle proposé suit le modèle général détaillé plus haut, avec une dynamique assurée par un réseau RC (on cherche une modélisation fidèle du neurone biologique). La fonction d'activation est la sigmoïde.

3.6.2 ARCHITECTURE DU RESEAU

C'est un réseau récurrent entièrement connecté. Les neurones n'ont cependant pas de retour sur eux-mêmes. La figure 3.6.1 illustre cette architecture:

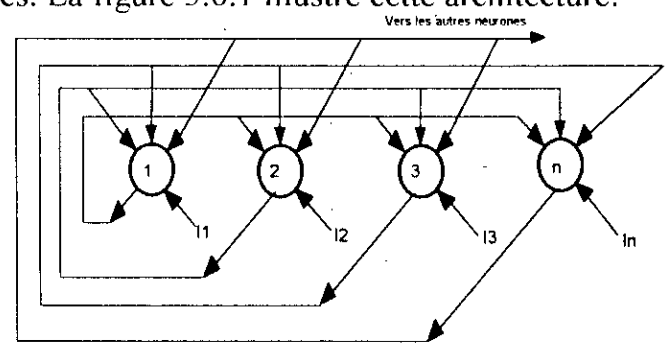


Figure 3.6.1: Architecture d'un réseau de Hopfield

[1]Réseaux de Neurones Artificiels.

[2]Une mémoire associative est une mémoire adressable par son contenu c.à.d. qu' à partir d'une partie ou d'une déformation du vecteur mémorisé, on retrouve celui-ci.

On remarque la présence des termes de biais.

3.6.3 DYNAMIQUE DES SORTIES

Notons la sortie du sommateur du i^{ème} neurone net_i alors:

$$net_i = \sum_{j=1}^{n} W_{ij} x_j + I_i$$

où

I_j: terme de biais

W_{ij}: poids synaptique d'entrée du i^{ème} neurone; j = 1,n j≠i

x_j: j = 1,n j≠i entrées du i^{ème} neurone

Supposons que la fonction d'activation du i^{ème} neurone a un seuil S_i ; alors la dynamique de sortie du i^{ème} neurone est:

$$x_i(t+1) = \begin{cases} +1 & \text{si } net_i > S_i \\ x_i(t) & \text{si } net_i = S_i \\ -1 & \text{si } net_i < S_i \end{cases}$$

Mais faut-il déterminer. d'abord. les poids c'est l'apprentissage.

3.6.4 APPRENTISSAGE

Dans la section précédente on a vu que l'apprentissage consistait en la recherche du vecteur des poids minimisant au maximum l'erreur entre la sortie du réseau et celle désirée, par des méthodes itératives.

Pour les réseaux de Hopfield la situation est. quelque peu, différente. Les poids sont calculés à l'avance. Il y a deux manières de le faire

3.6.4.1 Calcul des poids par autocorrélation des entrées

On a recours à cette méthode quand on ne connaît pas la fonction d'énergie (voir plus bas pour la définition) propre à notre application; ce qui est souvent le cas.

Soit un réseau de Hopfield appelé à stocker et rappeler un groupe de vecteurs $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L)$, la matrice des poids **W** est déterminée par autocorrélation des entrée:

$$\mathbf{W} = \sum_{i=1}^{L} \mathbf{X}_i \mathbf{X}_i^t$$

C'est une matrice symétrique avec une diagonale nulle.

3.6.4.2 Calcul des poids par identification de la fonction d'énergie

D'abord expliquons ce que est une fonction d'énergie pour un réseau de Hopfield:

Un exemple inconnu présenté au réseau de Hopfield, peut nécessiter que ce dernier fasse quelques itérations avant de se stabiliser sur le résultat final. Pendant ce temps le vecteur de sortie varie dans le temps et il forme, alors, un système dynamique.

pour un système dynamique en général, et pour les réseaux de Hopfield en particulier, l'existence d'une solution stable est un problème posé. Mais dans ce contexte, on admettra l'existence de la solution, on dira que le réseau converge toujours vers cette solution stable.

Toutefois dans cette théorie des systèmes dynamiques, l'existence de solution peut être prouvée par l'utilisation du concept de la fonction de Lyapunov, qu'on appelle aussi la fonction d'énergie: c'est une fonction des variables d'état, bornée et tel que tout changement d'état entraîne une diminution de sa valeur.

Pour un réseau de Hopfield:

$$E = -\frac{1}{2}\mathbf{X}^{T}\mathbf{W}\mathbf{X}$$

où

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}\mathbf{X}_i W_{ij}\mathbf{X}_j$$

La détermination des poids à partir de cette fonction, se fait quand on peut trouver une fonction d'énergie propre à notre application, en identifiant la forme générale de la fonction d'énergie par la fonction spécifique à l'application[3]

3.7 CONCLUSION

Quatre algorithmes d'apprentissage des réseaux de neurones statique (un ADC étant un dispositif à comportement statique) ont été traité dans cette section.

Au premier lieu l'algorithme de BP; le plus notoire parmi les quatre, mais caractérisé par certains inconvénients qui le rende parfois inutilisable. C'est dans le but de palier à ces problèmes qu'on présenté les trois autres algorithmes:

- ROM: pour éviter les problèmes des minimums locaux.

---

[3]Ceci est le cas pour la conversion A/D, voir partie 2.

- FBP: pour améliorer la rapidité d'apprentissage.
- ELEANNE 7: pour augmenter l'efficacité de l'apprentissage.

A la fin, un type de réseaux dynamiques (Hopfield) a été étudié car il se prête bien à notre application.

2ème Partie

MODELISATION, CONCEPTION ET REALISATION

SECTION

1

MODELISATION D'ADCs PAR ENTRAINEMENT DES RESEAUX DE NEURONES

1.1 INTRODUCTION

Si on veut modéliser un système par réseaux de neurones, la première étape consiste à l'identifier en apprenant son comportement au réseau. La deuxième est la validation du modèle neuronal ainsi obtenu par la comparaison entre ses sorties et celles du système. La troisième et dernière étape consiste quant à elle, en des tests appelés finaux déterminant les performances du modèle lorsqu'il sera en phase de production.
Un ADC est un systèm ene sort pas de cette règle.

Le travail effectué dans ce sens lors du déroulement du présent P.F.E., est le thème de cette section.

1.2 IDENTIFICATION DU MODELE DE L'ADC

L'ADC étant un composant statique (toujours la même sortie pour une entrée donnée), nous avons utilisé des réseaux statiques à couches. La stratégie d'identification -d'apprentissage- utilisée est décrite dans la figure suivante:

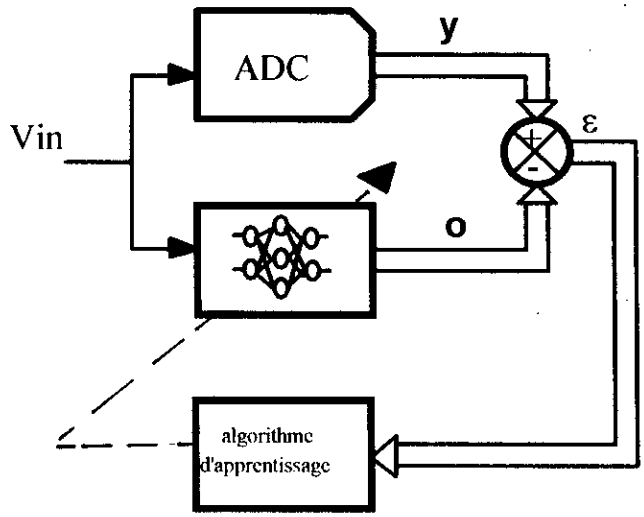


Figure 1.2.1: Modélisation neuronal de l'ADC

Durant ce travail, nous avons été amenés à essayer plusieurs approches, dans le but de trouver la solution optimale au problème posé; critère: minimiser le nombre de neurones en vue de simplifier la réalisation Hardware de l'ADC neuronal (notre finalité dans ce projet).

1.2.1 ADC A SORTIES BINAIRES -1ère APPROCHE-

En toute évidence, la première idée qui vient à l'esprit quand on veut résoudre ce problème par les réseaux de neurones est l'utilisation d'un réseau monocouche à n sorties pour un ADC à n bits; c'est à dire une même couche cachée pour le traitement de toutes les sorties. Ces sorties sont selon le code binaire[1] .

Donc pour commencer on a repris le travail de Bernieri et al. [Ber 95]:

1.2.1.1 ADC à 4 bits

D'après l'article, le réseau entraîné (par BP) a requis 80 neurones dans la couche cachée et 4 000 000 d'itération (resp. 5 000 000) pour un *full scale* de +1.6 V (resp. ±0.8 V) avant d'arriver à une solution satisfaisante.

Les figures suivantes illustrent la difficulté de l'apprentissage (on peut remarquer, pour le fs = 1.6V par exemple, que durant les 3 500 000 denières itérations la MSE n'a diminué que de 0.1):

[1] Qu'on désignera sans ambiguïté par: sorties binaires.

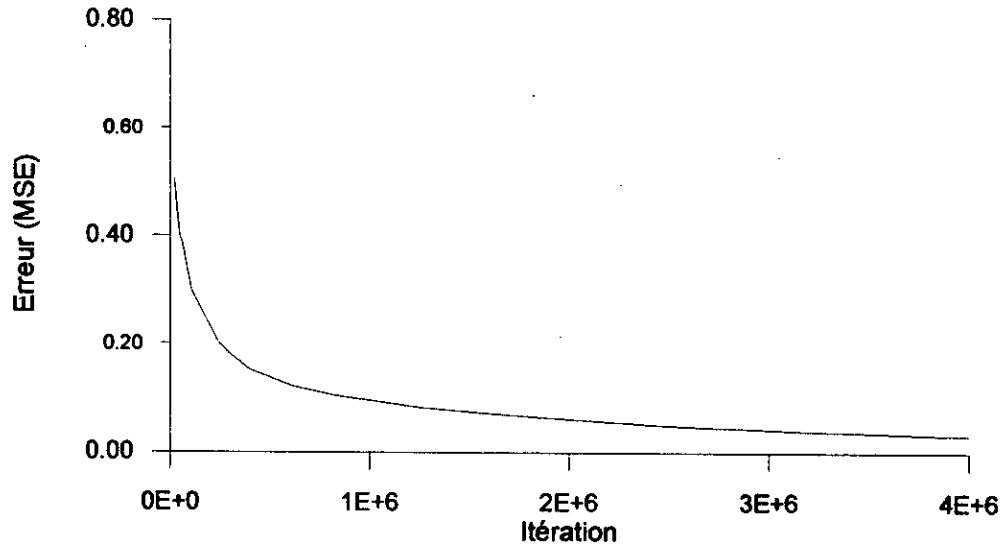


Figure 1.2.2 : Evolution de la MSE[2] lors de l'apprentissage d'un ADC à 4 bits (fs = +1.6 V)

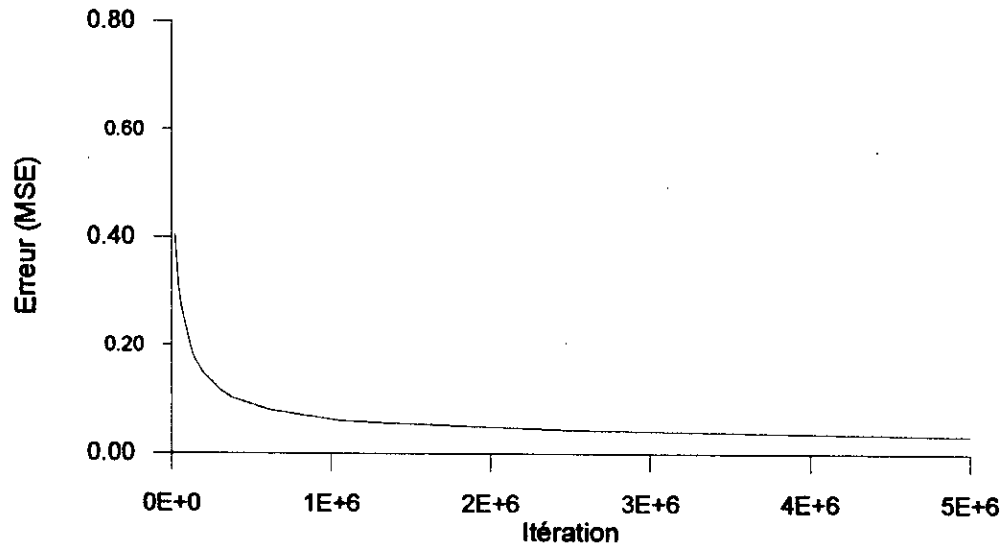


Figure 1.2.3 : Evolution de la MSE lors de l'apprentissage d'un ADC à 4 bits (fs = ±0.8 V)

1.2.1.2 ADC à 6 bits

Pour le 6 bits, Bernieri a utilisé un réseau de 160 neurones dans la couche cachée, qui a convergé après 20 000 000 d'itérations! (entraînement par BP):

[2] Mean Square Error (Erreur au sens des moindres carrés).

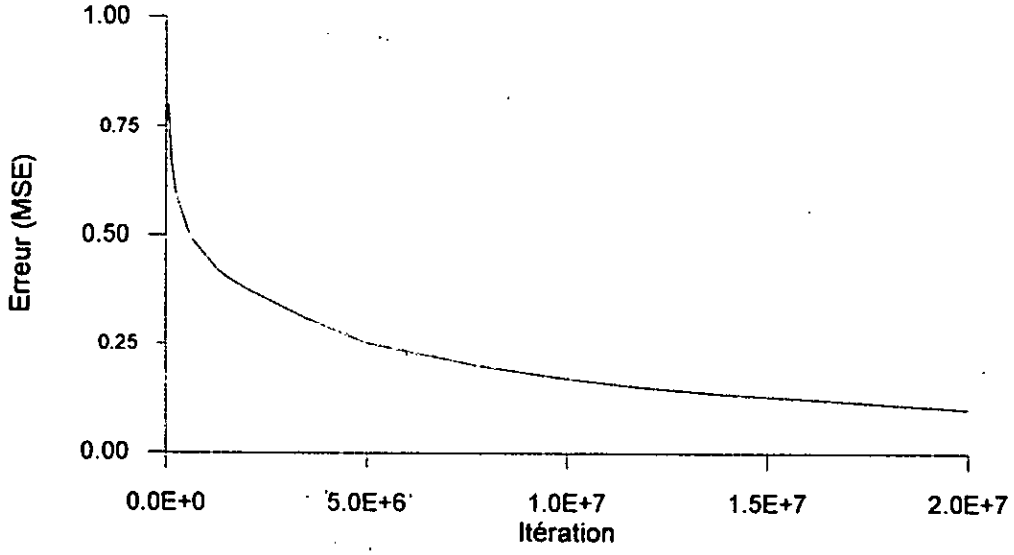


Figure 1.2.4 : Evolution de la MSE lors de l'apprentissage d'un ADC à 6 bits (fs = ±5.12 V)

1.2.1.3 Problèmes rencontrés et solutions

Quand on a repris ce travail on avait en tête une pensée et une idée qui en découlait. La pensée était que Bernieri n'a pas cherché à optimiser (selon notre critère) son réseau; l'idée était, alors, de réentraîner le réseau avec un nombre réduit de neurones.

Mais on s'est heurté, essentiellement, à un problème: celui de la convergence, ou plutôt la rapidité de convergence. Ce problème est lui même dû à plusieurs autres:

1- Les moyens de calculs insuffisants et peu puissants dont on disposait.

2- La lenteur de l'algorithme de rétropropagation (BP).

3- La non-linéarité très dure que le réseau a à apprendre; Exemple pour un ADC à 4 bits, lors du passage de l'état 0111 ($8^{ème}$ état) à l'état 1000 ($9^{ème}$ état), toute les sorties basculent en même temps pour une petite variation de l'entrée ($V_{ref}/16$). Cela fait perdre au réseau les informations déjà apprises (on a constaté une augmentation importante de la MSE).

Nous avons donc pensé à des solutions:

1- Pour la lenteur de convergence: utiliser des algorithmes plus performants que BP. On a opté pour FBP.

2- Pour la non-linéarité, deux solutions:

- La première est la synthèse d'un ADC Gray[3] .
- La deuxième: entraîner *un* réseau pour *chaque* sortie.

1.2.2 ADC GRAY

Ce modèle -doucement non-linéaire- a été identifié, pratiquement, sans problème.

Pour notre travail on a adopté la stratégie suivante:

- • Commencer par un ADC avec un petit nombre de bits (initialement = 2).
  • Apprendre au réseau un petit nombre d'exemple (égal au nombre d'états des sorties de l'ADC). Avec un faible gain[4] pour la fonction d'activation.

  • On refait cette dernière étape jusqu'à que le modèle soit validable(voir plus bas dans cette section).

- Une fois l'ADC neuronal identifié, incrémenter de 1 le nombre de bits et refaire les mêmes opérations décrites ci dessus.

- Refaire cette dernière jusqu'à l'obtention de la résolution désirée.

Pour ce qui est de notre cas on est allé jusqu'à 4 bits (les moyens de calculs ne permettant pas d'aller plus loin).

1.2.2.1 Résultats

On explicitera ici, sous forme de graphes l'évolution de l'erreur au cours de l'apprentissage. Avant et lors de la généralisation[5] .

---

[3] Sorties selon le code Gray.

[4] Soit F(x): La fonction d'activation et b son gain. Initialement on pose b = 1 et F(x) = f(x). A la seconde étape on augmente b et F(x) = f(bx).

[5] On dira sans ambiguïté: Généralisation ou généralisation par apprentissage, différente de la capacité de généraliser (interpoler ou extrapoler) *propre* au réseau.

❑ ***ADC Gray à 2 bits***

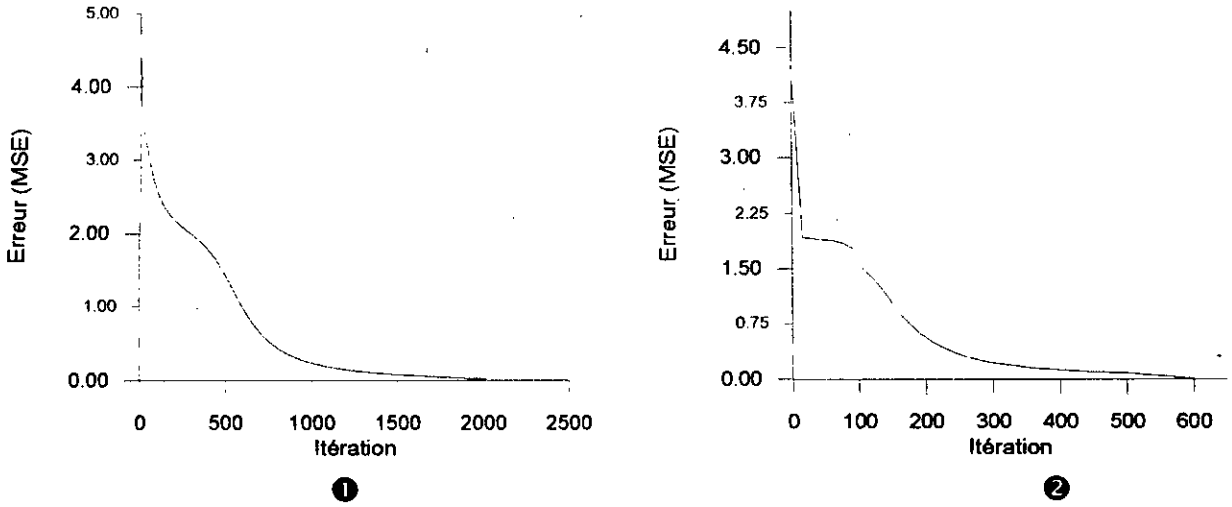


Figure 1.2.5: Evolution de l'erreur au cours de l'apprentissage pour un ADC Gray (2 bits)

❶ Avant la généralisation

❷ Lors de la généralisation

❑ ***ADC Gray à 3 bits***

Ici on a utilisé des fonctions d'activations différentes; de la couche cachée à la couche de sortie (gains différents).

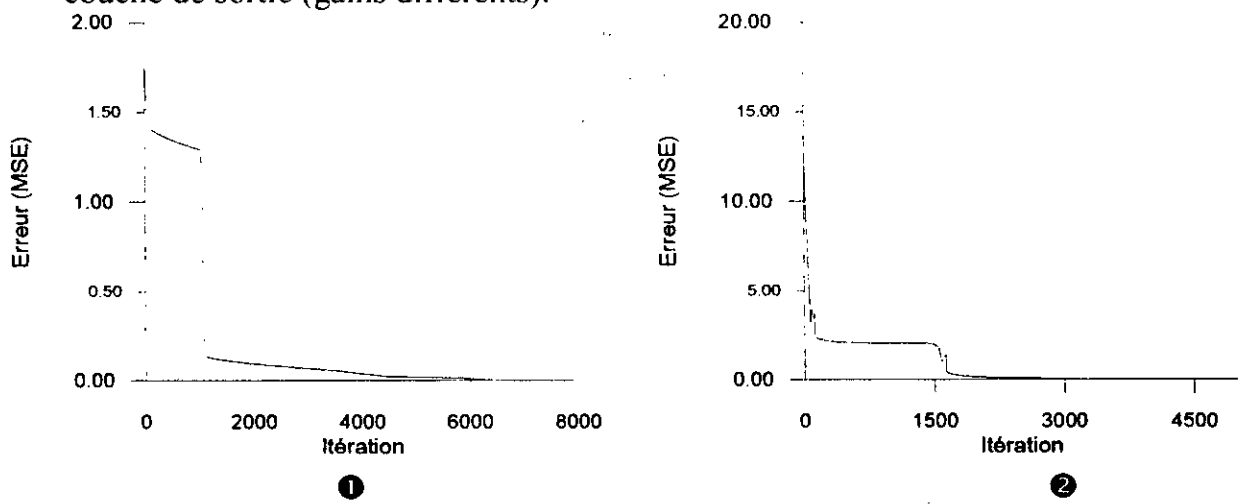


Figure 1.2.6: Evolution de l'erreur au cours de l'apprentissage pour un ADC Gray (3 bits)

❶ Avant la généralisation

❷ Lors de la généralisation

❑ ***ADC Gray à 4 bits***

Idem, ici, que pour l'ADC Gray à 3 bits (Gains différents pour les fonctions d'activation de la couche cachée et la couche de sortie).

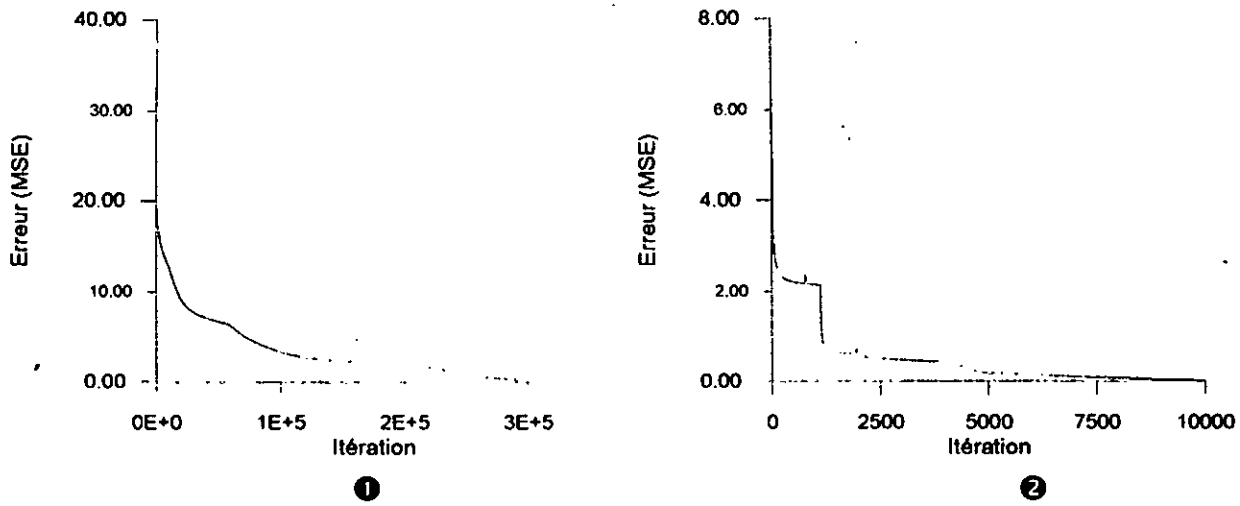


Figure 1.2.7: Evolution de l'erreur au cours de l'apprentissage pour un ADC Gray (4 bits)

❶ Avant la généralisation

❷ Lors de la généralisation

1.2.2.2 Interprétations

L'interprétation des graphes, indifférente au nombre de bits, est la suivante:

Avant la généralisation: Comme tout est nouveau au réseau, ce dernier met beaucoup de temps pour "assimiler" le comportement de l'ADC. Autrement dit le nombre d'itérations nécessaire pour que le réseau converge est plus ou mois important (selon le nombre de bits: exemple pour le 4 bits vers 290 000 itérations).

Après la généralisation: Les choses sont beaucoup plus évidentes pour le réseau, puisqu'il a déjà une idée sur le comportement de l'ADC, en plus -faut-il le rappeler- les réseaux de neurones tirent leurs intérêt de leur capacité de généralisation (interpolation). Le nombre d'itérations se trouve réduit.

1.2.3 ADC A SORTIES BINAIRES -2ème APPROCHE-

On entend par deuxième approche, l'entraînement d'*un* réseau *pour chaque* sortie. La non-linéarité se trouve ainsi adoucie. En effet chaque réseau n'aura à apprendre qu'un signal carré (le nombre de cycles est variable selon le poids du bit) plutôt que n signaux à la fois (pour un ADC à n bits) dans la 1ère approche.

La stratégie adoptée est pratiquement la même que celle décrite dans le §1.2.2; à la différence qu'il y a ici n réseaux et que chaque réseau identifie le signal d'un bit -une sortie- de l'ADC (et non pas l'ADC lui même). L'un des avantages de cette approche est qu'à partir d'un ADC neuronal à n bits on peut obtenir un ADC à (n+1) bits en rajoutant, tout simplement, un réseau ayant appris le nouveau LSB.

Pour cette approche on est allé jusqu'à 3 bits sans problèmes mais le réseau du 4ème n'a pas pu converger vers une solution convenable faute, pensons-nous, de moyens de calculs et de critères bien définis pour l'initialisation des poids et le choix de la fonction d'activation selon l'application.

1.2.3.1 Résultats

❑ *1er bit (MSB):*

Le réseau est -tout simplement- un neurone dont la fonction d'activation a été décalée de $V_{ref}/2$ par le biais du poids du terme de biais. Il n'y a donc pas d'apprentissage à faire.

❑ *2ème bit (MSB-1):*

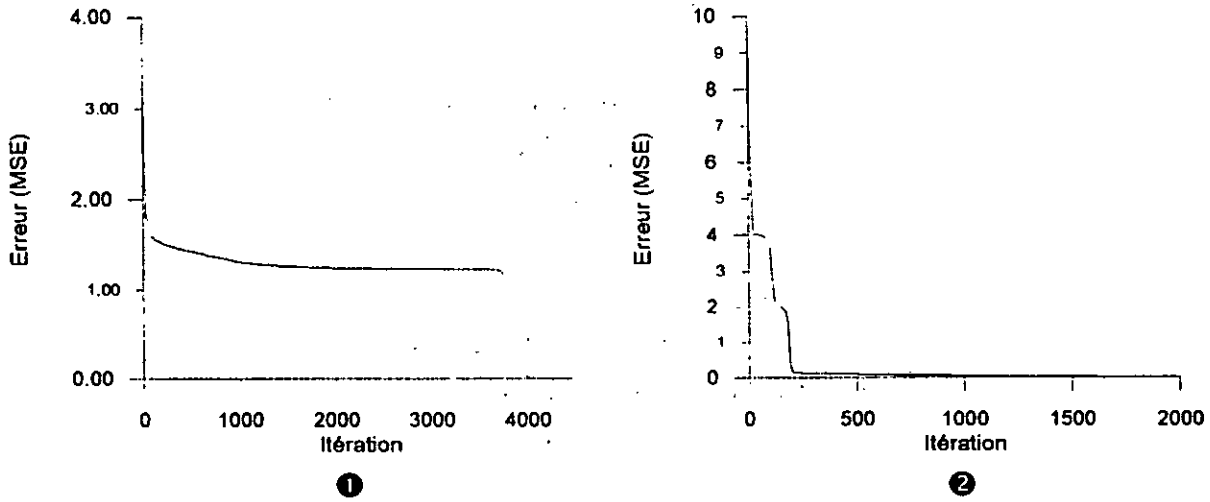


Figure 1.2.8: Evolution de l'erreur au cours de l'apprentissage du LSB (2 bits)

❶ Avant la généralisation

❷ Lors de la généralisation

❑ *3ème bit :*

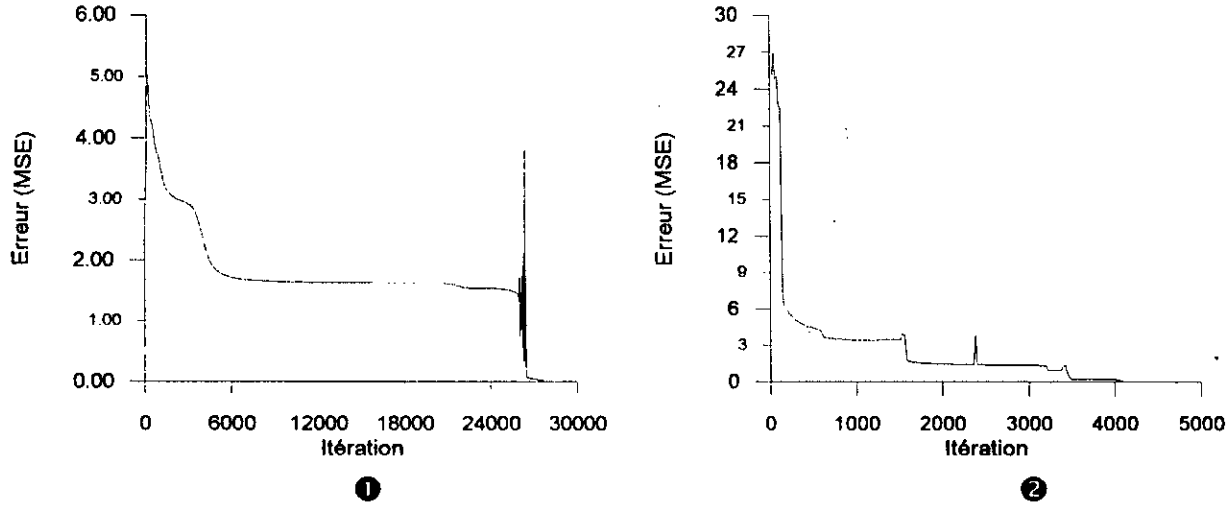


Figure 1.2.9: Evolution de l'erreur au cours de l'apprentissage du LSB (3 bits)

❶ Avant la généralisation

❷ Lors de la généralisation

❑ *4ème bit :*

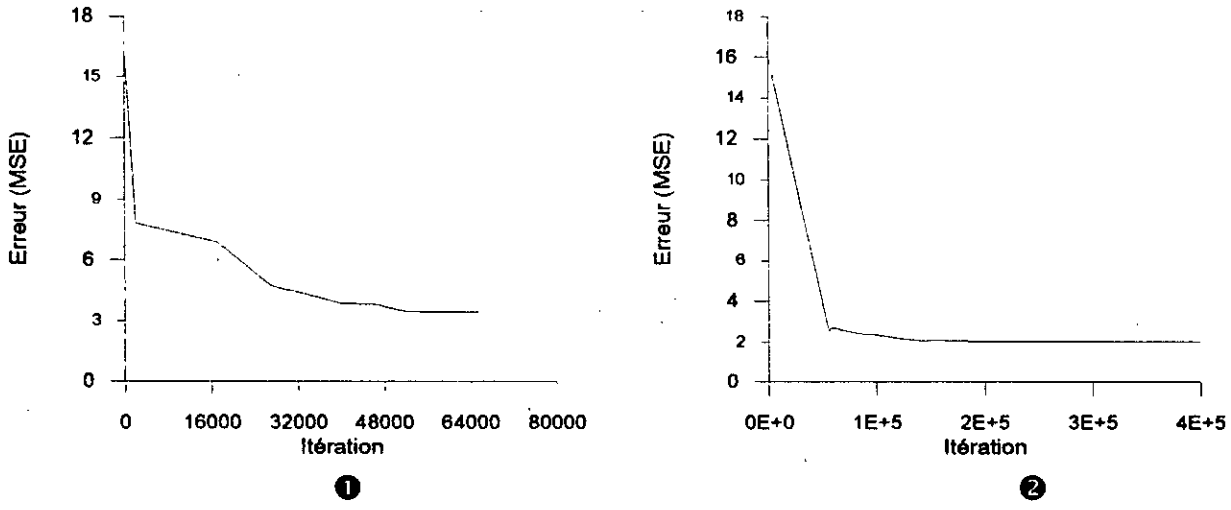


Figure 1.2.10: Evolution de l'erreur au cours de l'apprentissage du LSB (4 bits)

❶ Même fonctions d'activation pour les 2 couches (cachée et sortie)

❷ Fonctions d'activation différentes

1.2.3.2 Interprétations

Pour le nombre d'itérations nécessaire à l'apprentissage et à la généralisation, on peut faire deux remarques:

1- La première est la même que celle faite pour les ADCs Gray .

2- La deuxième est qu'on remarque que plus le poids du bits diminue plus la généralisation devient facile (voir §1.3.2 pour le «Pourquoi ?»).

Pour le 4ème bit, les deux graphes n'explicitent pas l'apprentissage avant et lors de la généralisation, mais plutôt l'apprentissage d'un réseau dont tous les neurones ont la même fonction d'activation -pour le 1er graphe-, et d'un autre dont le neurone de sortie diffère des neurones de la couche cachée par le gain de sa fonction d'activation -pour le 2ème graphe-.

On remarque que le deuxième réseau a atteint une erreur moindre que celle atteinte par le premier. Cela est du au fait que la fonction d'activation du neurone de sortie du 2ème réseau avait un gain très grand, est était donc proche de la fonction de Heaviside (qui correspond au type de la sortie désirée).

1.3 VALIDATION DES RESULTATS

La validation des modèles neuronaux obtenus, se fait en observant les sorties (chaque bit est pris séparément). On entend par observation, la comparaison entre les sorties désirées (sortie de l'ADC idéal -voir figure 1.2.1-) et les sorties du réseau respectivement correspondantes.

Cette comparaison se base sur les deux points suivants:

- Précision du passage d'un état à l'autre: le réseau doit faire *basculer* ses sorties aux multiples du quantum[6] .
- La différence entre la sortie désirée et la sortie du réseau (pour chaque bit)

Cette observation se fait à partir de graphes, représentants les sorties des ADCs neuronaux qu'on a synthétisés:

---

[6] Un quantum d'un ADC à n bits: $q=V_{ref}/2^n$.

1.3.1 PRESENTATION DES RESULTATS

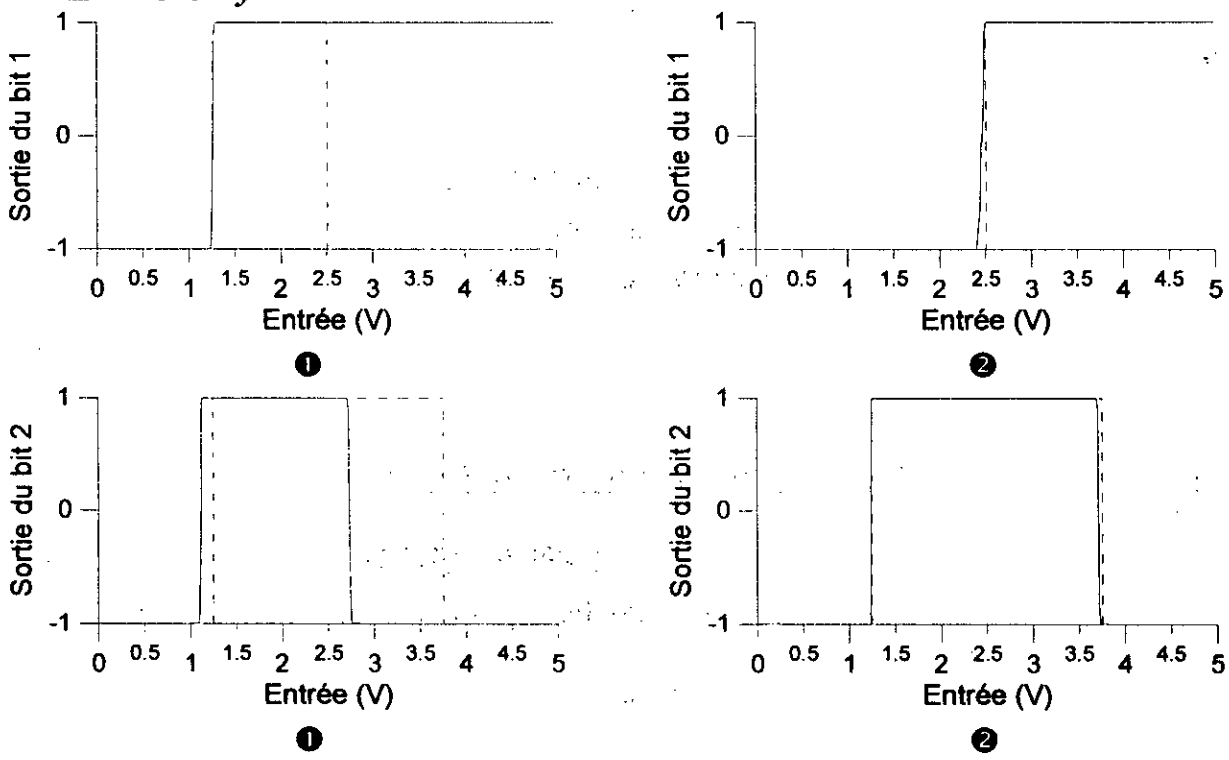
Pour la représentation des graphes on a adopté la légende suivante:

Sortie désirée

Sortie du réseau

1.3.1.1 ADC Gray

❑ ***ADC Gray à 2 bits***



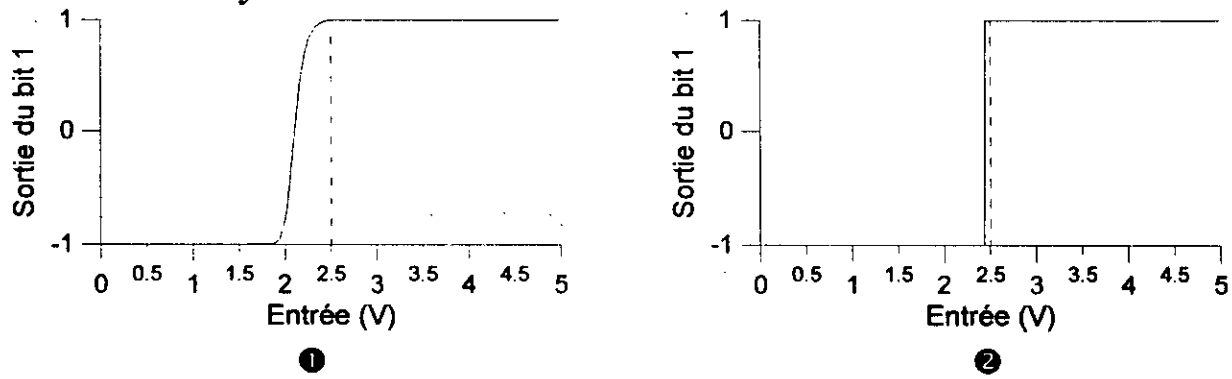
❶ ❷

Figure 1.3.1: Sorties de l'ADC Gray à 2 bits

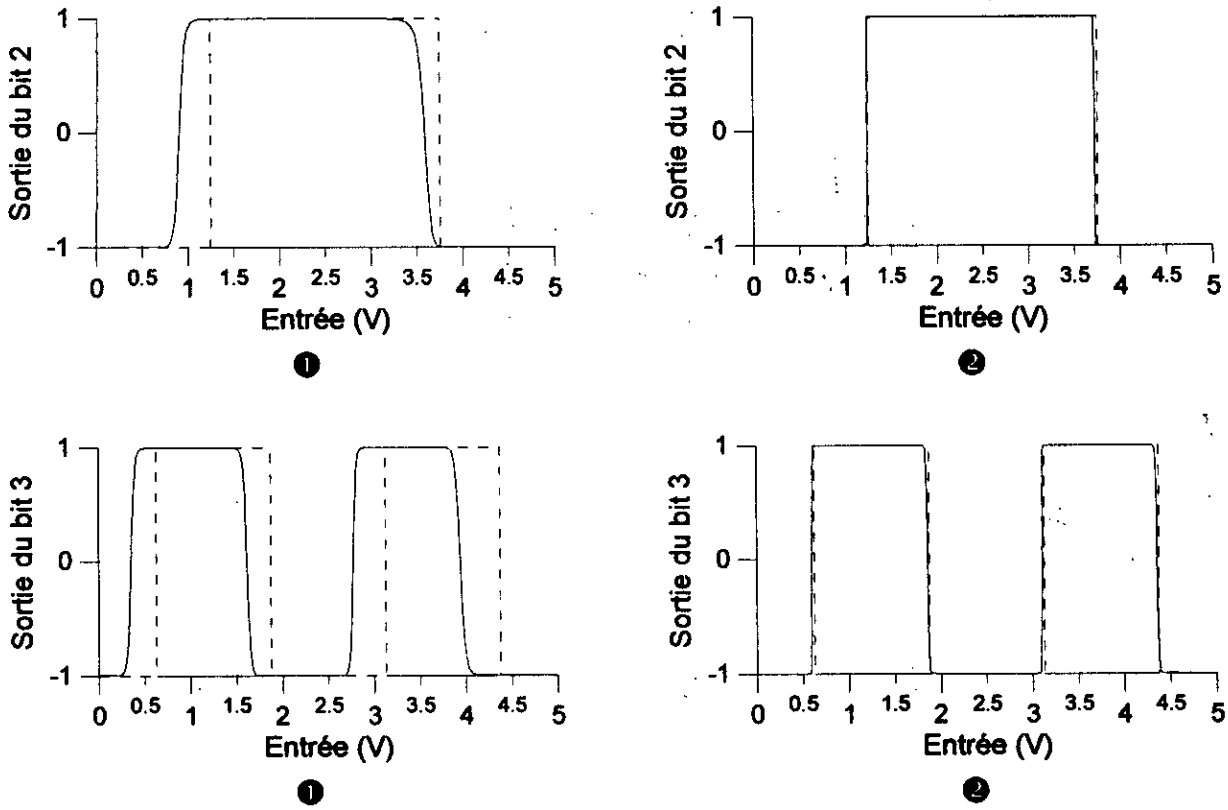
❶ Avant généralisation

❷ Après généralisation

❑ ***ADC Gray à 3 bits***



❶ ❷



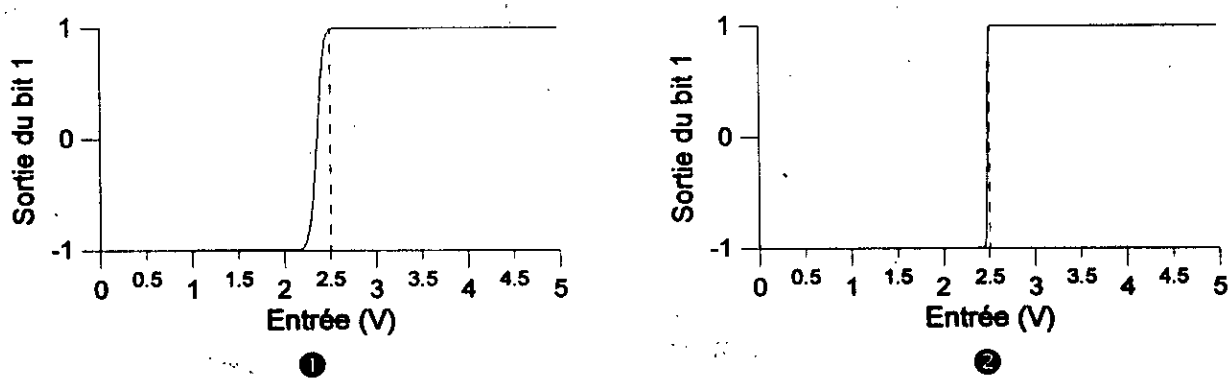
❶ ❷

Figure 1.3.2: Sorties de l'ADC Gray à 3 bits

❶ Avant généralisation

❷ Après généralisation

❑ ***ADC Gray à 4 bits***



❶ ❷

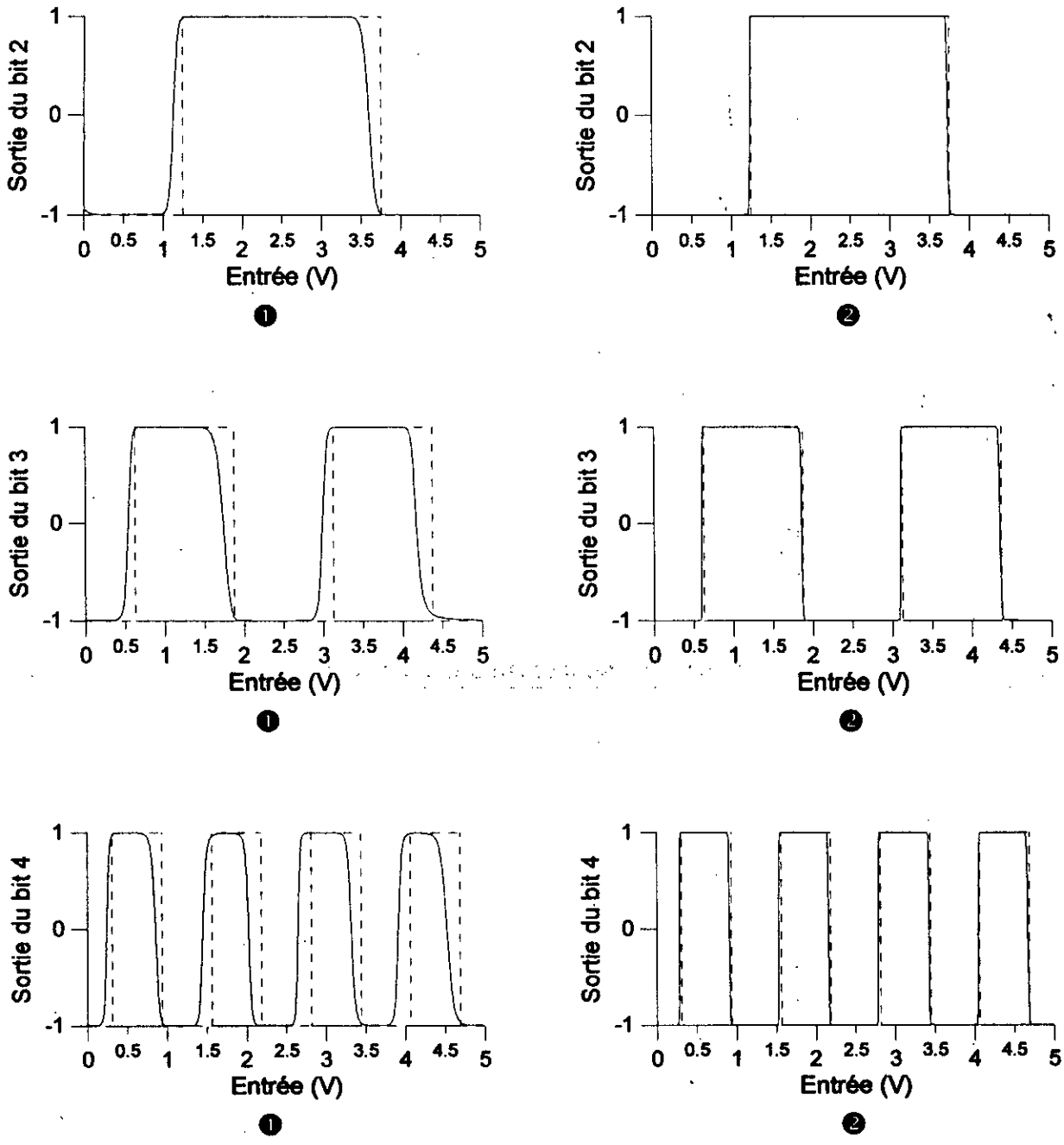


Figure 1.3.3: Sorties de l'ADC Gray à 4 bits

❶ Avant généralisation

❷ Après généralisation

1.3.1.2 ADC à sorties binaires -2ème approche-

❑ *1er bit (MSB)*

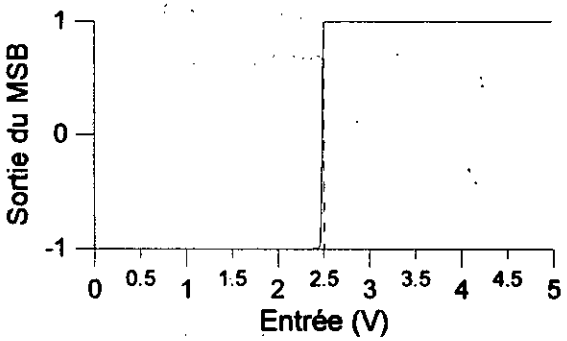


Figure 1.3.4: Sortie du MSB

❑ *2ème bit (MSB -1)*

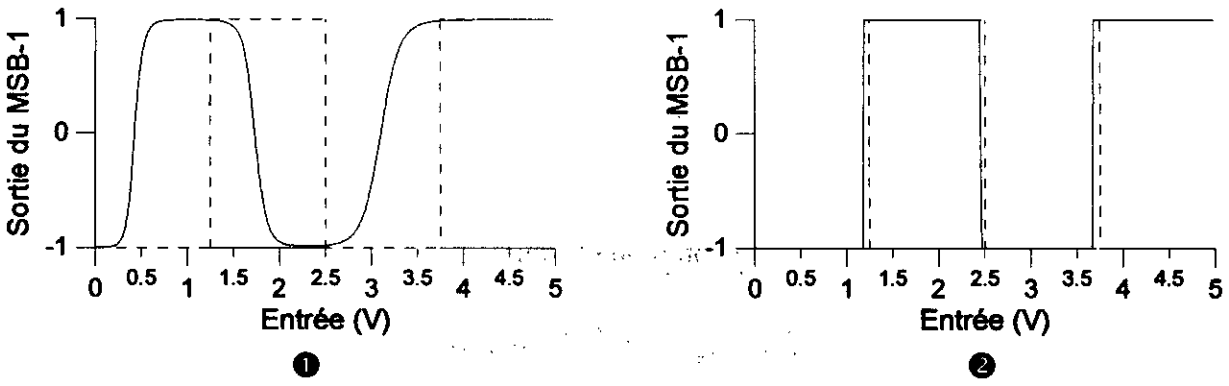


Figure 1.3.5: Sorties du MSB-1

❶ Avant généralisation

❷ Après généralisation

❑ *3ème bit (MSB -2)*

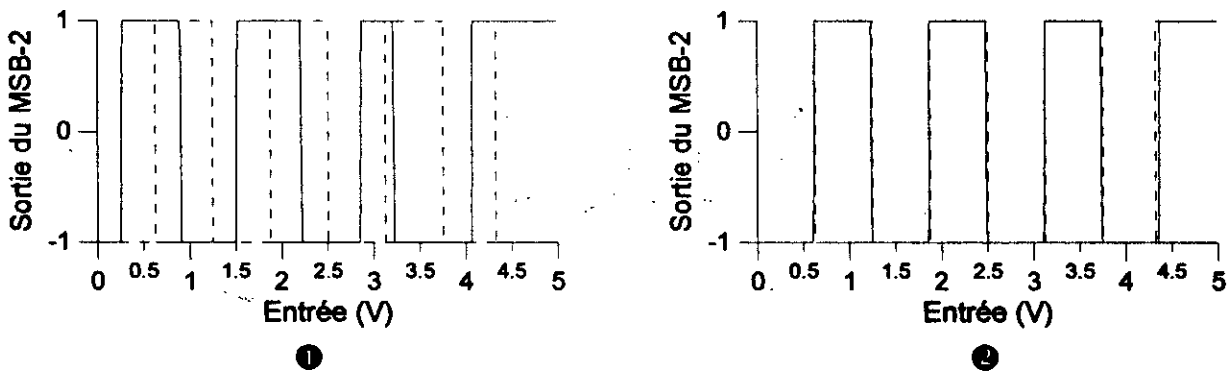


Figure 1.3.6: Sorties du MSB-2

❶ Avant généralisation

❷ Après généralisation

❑ *4ème bit (MSB -3)*

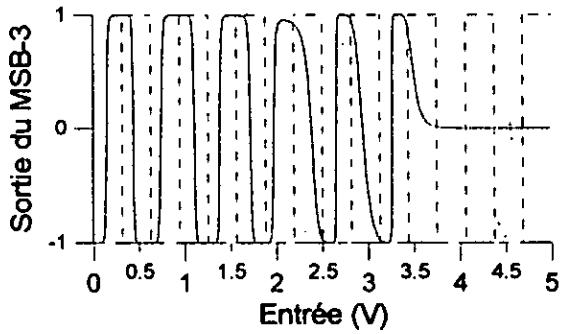


Figure 1.3.7: Sortie du MSB-3

1.3.2 INTERPRETATION ET VALIDATION DES RESULTATS

Les résultats (graphes) présentés ci-dessus établissent, en toute évidence, l'importance de la généralisation par apprentissage pour cette application. En effet on remarque que les sorties avant généralisation sont considérablement loin des sorties désirée; mais elles s'améliorent énormément après la généralisation et sont confondues avec les réponses désirées.

La précision sur les tensions d'entrées pour lesquelles il y a basculement d'état, devient de plus en plus grande au fur et à mesure qu'on augmente le nombre d'exemples à apprendre lors de la généralisation.

On trouve également ici, la réponse à la question qu'on s'est posé au § 1.2.3.2: «Pourquoi la généralisation, devient-elle facile au fur et à mesure que le poids du bit diminue ?». Cela est due à la diminution du quantum combinée à la capacité de généralisation propre au réseau (différente de la généralisation par apprentissage) qui devient de plus en plus efficace.

Pour la validation des ADCs neuronaux modélisés, tous sont validables (bonnes réponses, MSE satisfaisante) sauf L'ADC à 4 bits de la deuxième approche; le LSB ne donnant pas la réponse désirée. On établit ici que plus le nombre de bits augmente plus l'apprentissage devient difficile (le nombre de cycles[7] augmente: Par exemple, le réseau du 4ème bit a appris seulement 6 cycles sur 8 en 200 000 itérations et il était dans la même situation 300 000 itérations plus loin).

[7] On entend par cycle une alternance de deux états consécutifs.

1.4 TESTS FINAUX

Après avoir validé les résultats obtenus, on arrive à l'étape finale des tests, qui consiste à relever les réponses des ADCs neuronaux à différents signaux -de dynamique différentes-. Ces tests présentent l'intérêt de pouvoir donner une appréciation sur la fiabilité de l'ADC et sa capacité de poursuite des signaux. En bref, ils déterminent si l'ADC synthétisé peut produire dans un environnement réel.

1.4.1 APPLICATION DES TESTS ET PRESENTATION DES RESULTATS

On a choisi, ici, deux tests:

- Le 1er: appliquer un signal rampe et recueillir sa réponse.
- Le 2ème: appliquer un signal sinusoïdal puis relever la réponse.

Pour pouvoir comparer les signaux d'entrée et les sorties de l'ADC, nous avons converti par ces dernières par un DAC:

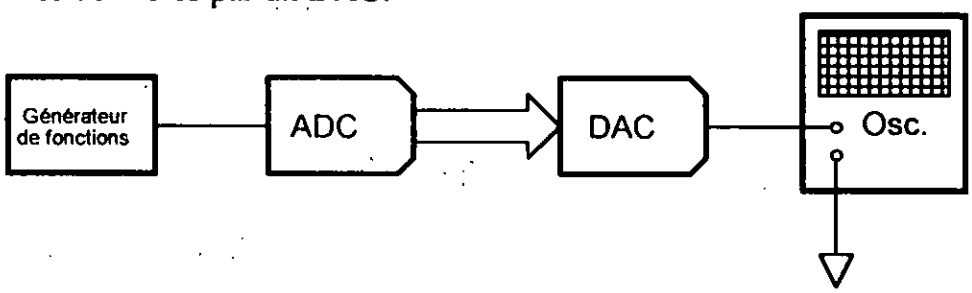


Figure 1.4.1: Application des tests et recueil des résultats

1.4.1.1 ADCs Gray

Pour que la sortie du DAC ait une signification, nous avons convertit les sorties Gray en sorties binaires à l'aide d'un décodeur Gray/Binaire (XOR):

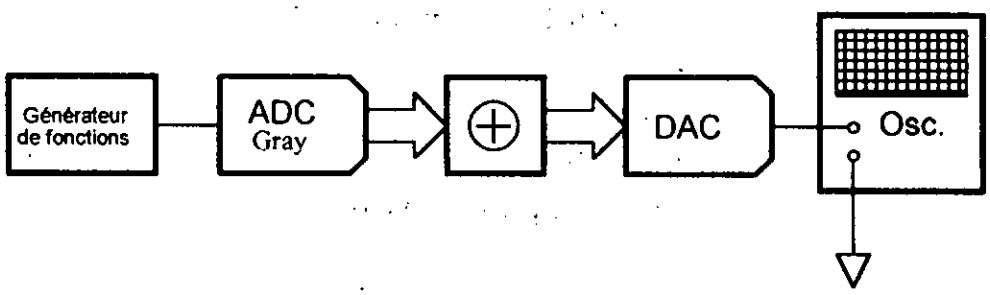


Figure 1.4.2: Application des tests et recueil des résultats pour un ADC Gray

Dans la partie 1.4, pour la présentation des graphes nous avons adopté la légende suivante:

- - - - - Signal d'entrée de l'ADC (test)

——— Signal de sortie du DAC

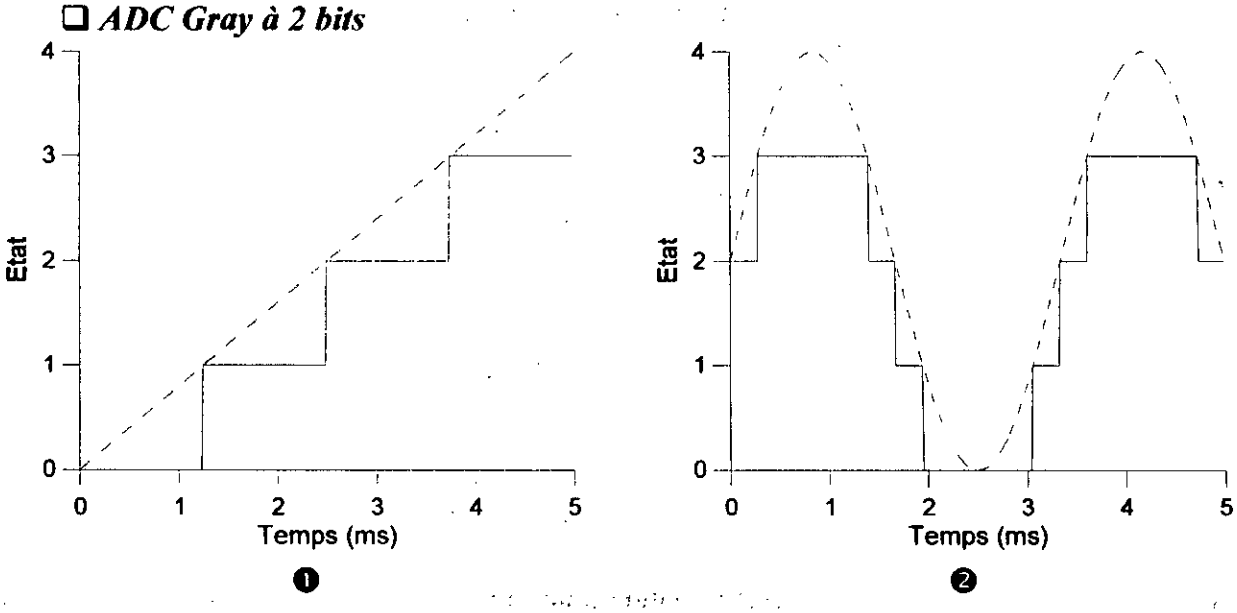


Figure 1.4.3: Réponse de l'ADC (après conversion D/A de ses sorties)

❶ à un signal rampe

❷ à un signal sinusoïdal

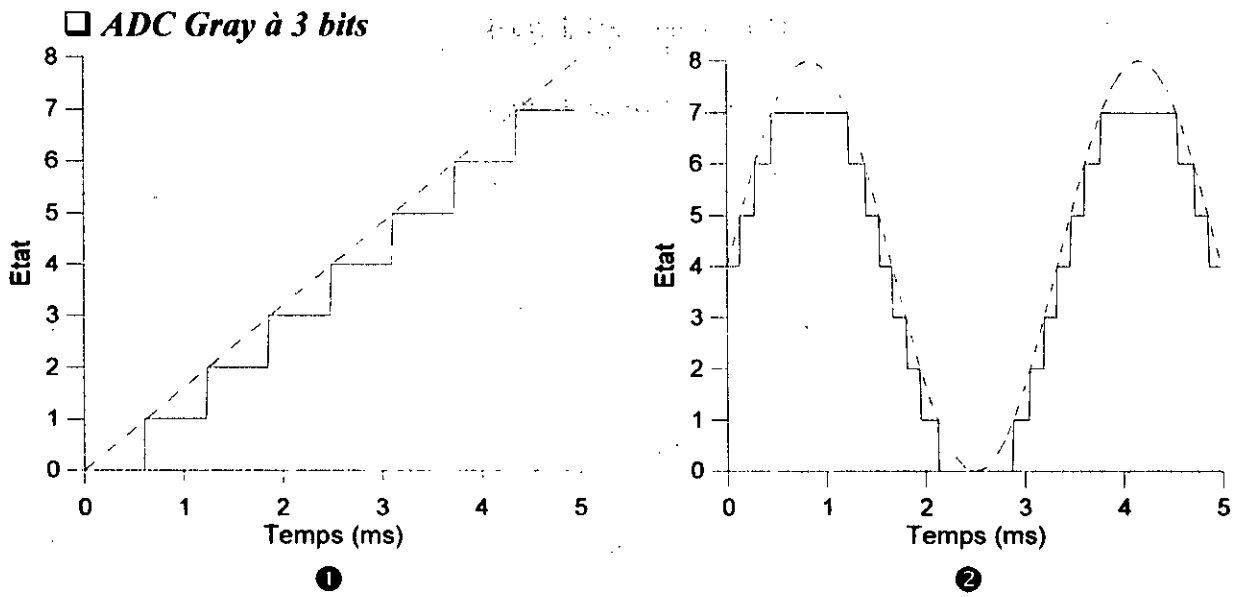
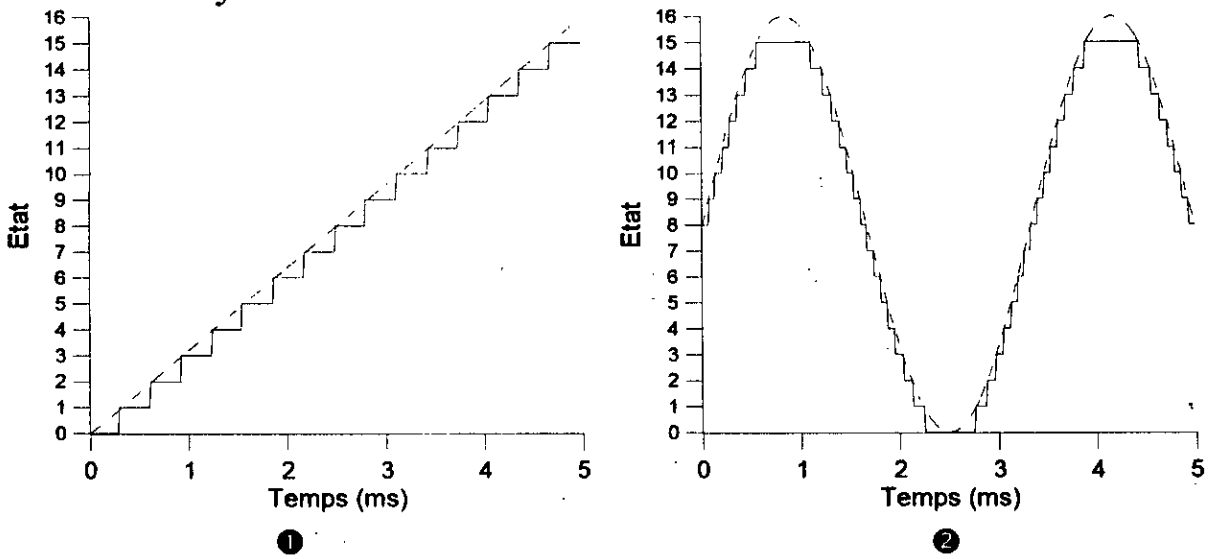


Figure 1.4.4: Réponse de l'ADC (après conversion D/A de ses sorties)

❶ à un signal rampe

❷ à un signal sinusoïdal

❑ ***ADC Gray à 4 bits***

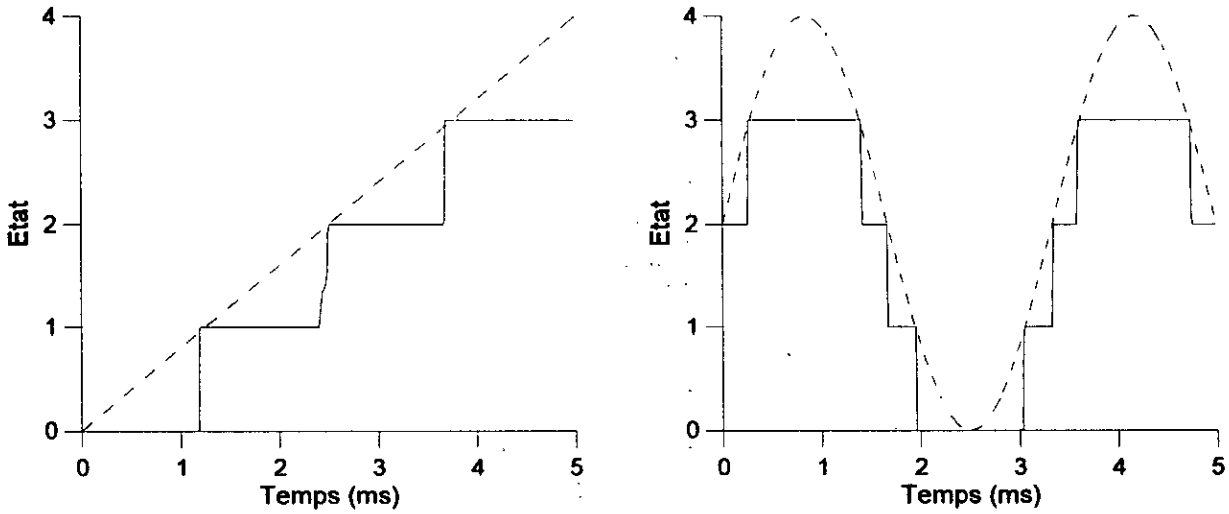


❶ ❷

Figure 1.4.5: Réponse de l'ADC (après conversion D/A de ses sorties)
❶ à un signal rampe
❷ à un signal sinusoïdal

1.4.1.2 ADC binaire -2ème approche-

❑ ***ADC Binaire à 2 bits***



❶ ❷

Figure 1.4.6: Réponse de l'ADC (après conversion D/A de ses sorties)
❶ à un signal rampe
❷ à un signal sinusoïdal

❑ ***ADC binaire à 3 bits***

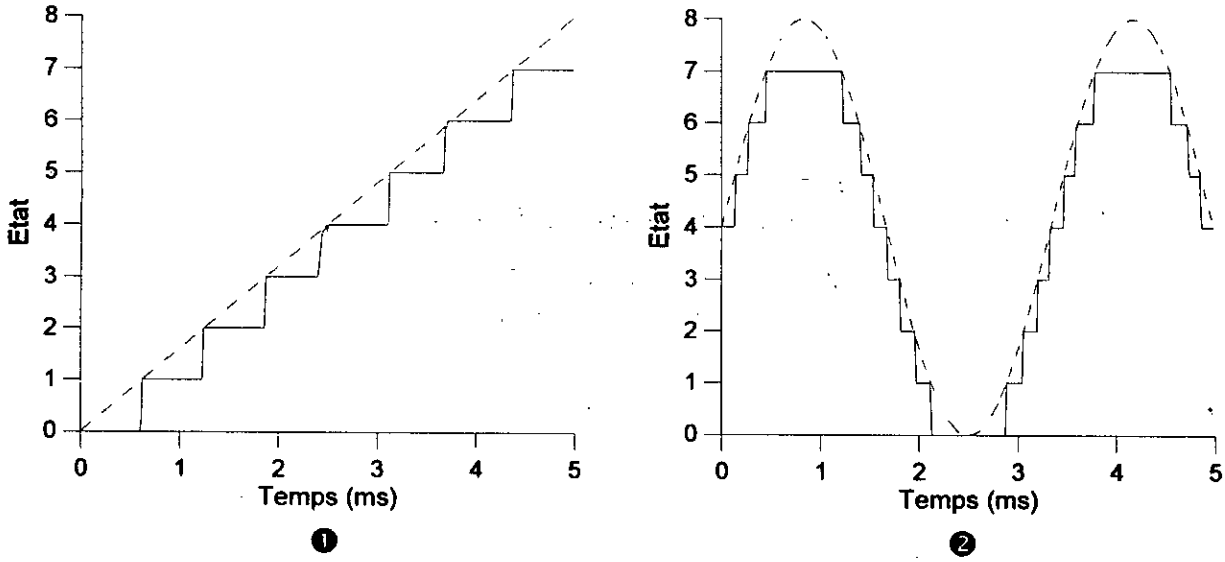


Figure 1.4.7: Réponse de l'ADC (après conversion D/A des ses sorties)
❶ à un signal rampe
❷ à un signal sinusoïdal

1.4.2 INTERPRETATIONS

Les ADCs neuronaux suivent bien leur entrées même pour des signaux à dynamique très variable (le signal sinusoïdal) ce qui établit leur rapidité (la réponse obtenue instantanément)

De plus, comme pour les ADCs classiques, l'augmentation du nombre de bits augmente la résolution -Evident-.

1.5 UN ADC PAR UN RESEAU DE HOPFIELD

La fonction d'un ADC étant de retrouver le mot binaire (b_1,b_2,…,b_n) qui représente le mieux l'entrée analogique (V_{in}). Donc la fonction à minimiser sera la suivante:

$$E_0 = (V_{in} - \sum_{i=1}^{n} b_i 2^{i-1})$$

En vue d'avoir une fonction d'énergie ressemblant à celle de Hopfield, quelques modifications sont nécessaires; on aura:

$$E = \frac{1}{2}E_0 - \frac{1}{2}\sum_{i=1}^{n}(2^{i-1})^2\left[b_i(b_i - 1)\right]$$

Le terme additionnel n'affecte en rien la solution finale. Il est utilisé pour éliminer la diagonale de la matrice des poids ($W_{ij} = 0$).

En développant la dernière équation nous aurons:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}(-2^{i+j-2})b_i b_j - \sum_{i=1}^{n}(-2^{2i-3} + 2^{i-1}V_{in})b_i$$

Ainsi on détermine les poids (W_{ij}) est les entrées de biais (I_i) par identification à la fonction objective d'un réseau de Hopfield qui s'énonce -en tenant compte des termes de biais- :

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}W_{ij}b_i b_j - \sum_{i=1}^{n}I_i b_i$$

d'où :

$$W_{ij} = -2^{i+j-2}$$

et $I_i = -2^{2i-3} + 2^{i-1}V_{in}$

Electriquement parlant les termes de biais sont des courants. Pour les implémenter on utilise une tension constante (V_{ref}) convertie par des conductances W_{iR} ainsi qu'une autre $W_{i\,in}$ chargée de convertir V_{in} (deuxième terme de I_i); ce qui donne:

$$W_{iR} = 2^{2i-3}$$

$$W_{i\,in} = 2^{i-1}$$

Les sorties seront négatives pour avoir des conductances W_{ij} positives:
d'où le circuit pour un ADC à 4 bits:

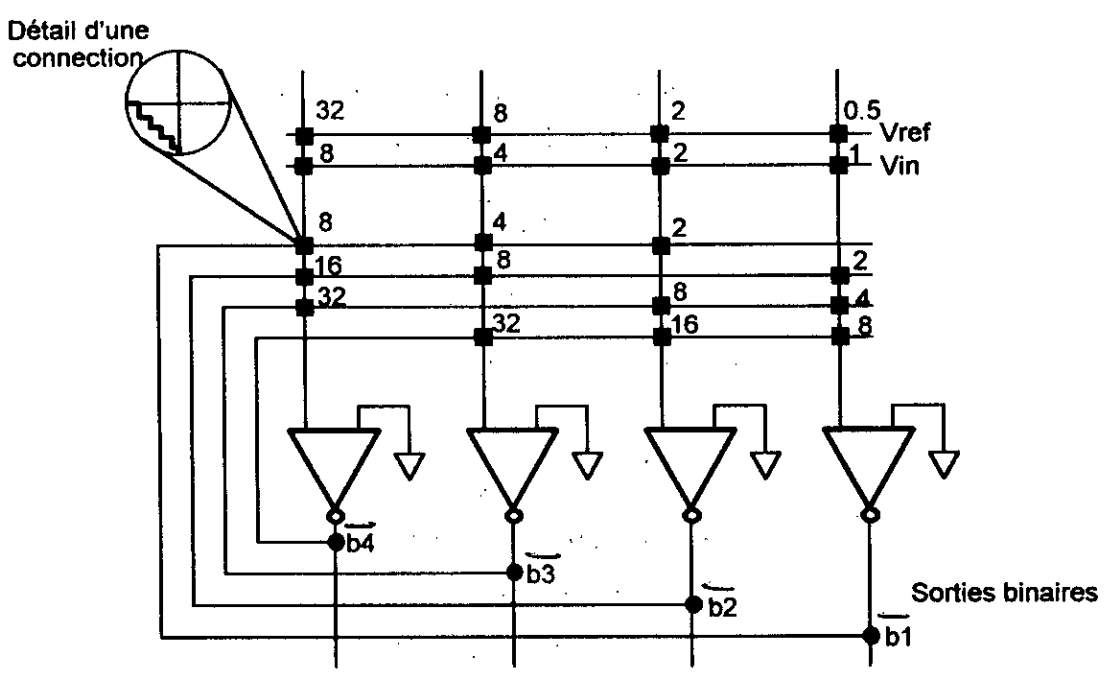


Figure 1.5.1: ADC de Hopfield à 4 bits

La simulation de ce circuit nous a donné la caractéristique entrée/sortie suivante

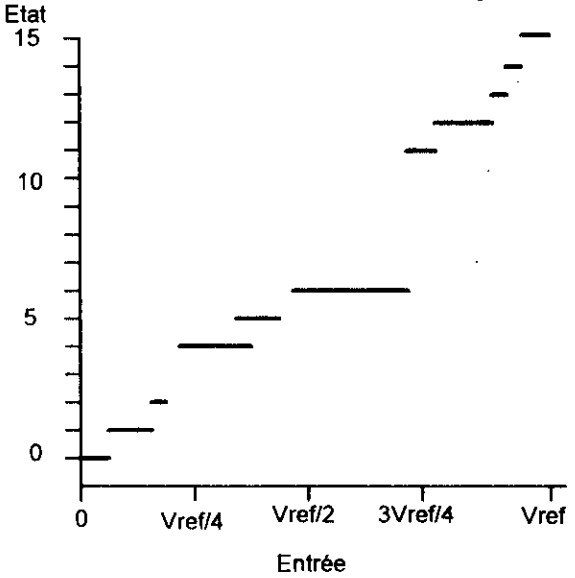


Figure 1.5.2: Caractéristique entrée/sortie de l'ADC de Hopfield (4 bits)

Les anomalies que présente cet ADC sont dues aux minimums locaux. Pour y remédier Lee & Sheu proposent le circuit suivant [Kos 92]:

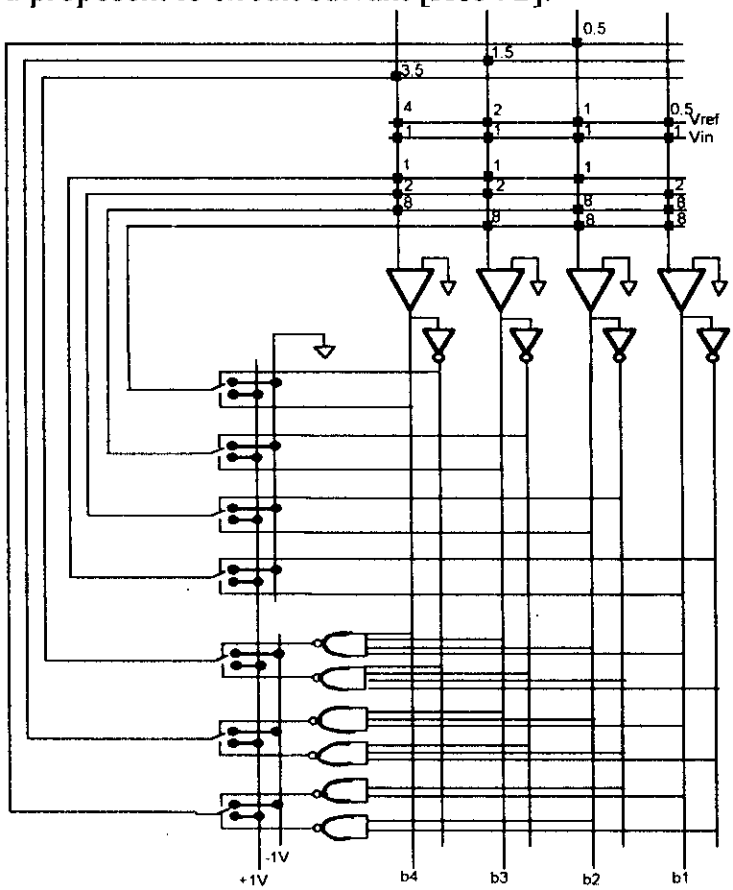


Figure 1.5.3: ADC de Hopfield modifié -avec autocorrection-

La simulation de ce circuit sur Spice a donné la caractéristique suivante:

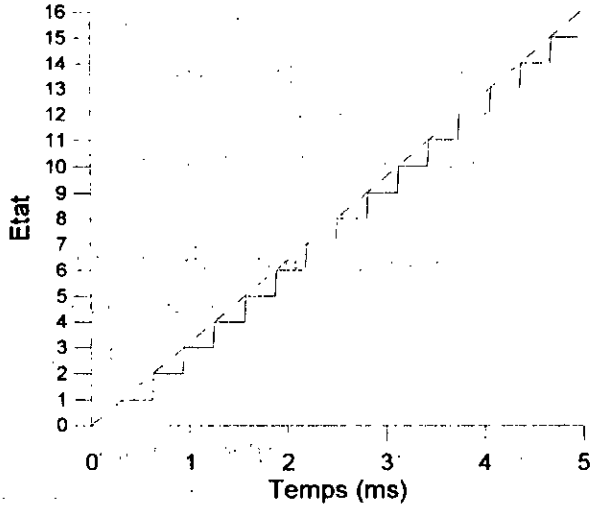


Figure 1.5.4: Caractéristique de l'ADC de Hopfield modifié

On remarque que les anomalies qui existaient dans le circuit précédent disparaissent.

1.6 CONCLUSIONS

Notre but était de modéliser un ADC neuronal avec le minimum de neurones. On a commencé par entraîner un réseau pour toutes les sorties de l'ADC à la fois. Le type de modélisation a été désigné par "1ère approche ". Cette technique "bouffait" beaucoup de neurones dans la couche cachée [Ber 95] et énormément de temps de calcul. on a donc essayé deux solutions: Le code Gray pour les sorties et l'entraînement d'un réseau pour chaque sortie de l'ADC binaire c'est la "2ème approche ".

Ces deux solutions ont permis de réduire le nombre de neurones et le temps de calcul par rapport à la première approche (Voir comparaison entre les différentes méthodes dans la section suivante).

Toutefois, pour la deuxième approche, on a pu terminer l'entraînement du 4ème bit faute de moyens de calcul.

Le étapes de validation et des tests finaux, ont montré la faisabilité et l'avantage énorme des ADCs neuronaux, en l'occurrence , la rapidité.

A la fin signalons que l'approche optimale pour la conversion analogique/digitale par réseau de neurones, qu'on a développé fera l'objet de la section suivante.

SECTION

2

METHODE POUR LA SYNTHESE D'UN ADC NEURONAL SANS APPRENTISSAGE

2.1 INTRODUCTION

Certains trouveront, nouveau, original, et pas normal de parler des réseaux de neurones sans évoquer l'apprentissage. En fait, ils ont, en quelque sorte, raison. Car la solution présentée dans cette section fait partie des autres, vers lesquelles le réseau pourrait converger par apprentissage.

Et il existe autant de solutions que de minimums[1] -dans la surface d'erreur-. Mais nous pensons que la solution proposée ici (qu'on nommera 3ème approche) correspond au minimum global. Du fait que l'erreur au sens des moindres carrés -MSE- est nulle; comme on le verra plus tard.

Pour ce qui est du plan de cette section: sera, d'abord, exposée l'idée en raisonnant sur les signaux. Puis la modélisation de l'idée par un réseau de neurones. La généralisation à un ADC à n bits se fera selon une récurrence. A la fin on illustra un tableau comparatif entre cette méthode et celles exposées dans la section précédente.

---

[1] Une solution vers laquelle converge le réseau peut être acceptée ou rejetée suivant le minimum correspondant.

2.2 L'IDEE

On va d'abord raisonner sur les signaux nécessaires à la génération du signal désiré (qui est un signal carré uniforme de 2^{n-1} cycles, avec n nombre de bits de l'ADC).

2.2.1 LE MSB

La fonction de Heaviside décalée de $V_{ref}/2$ est suffisante pour générer le signal correspondant au MSB:

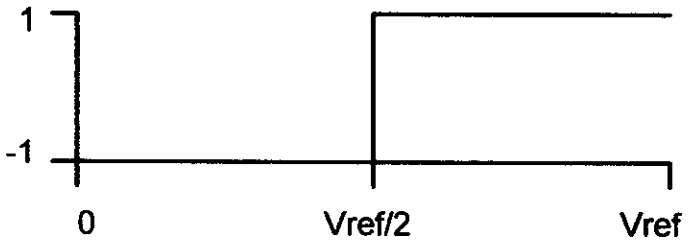
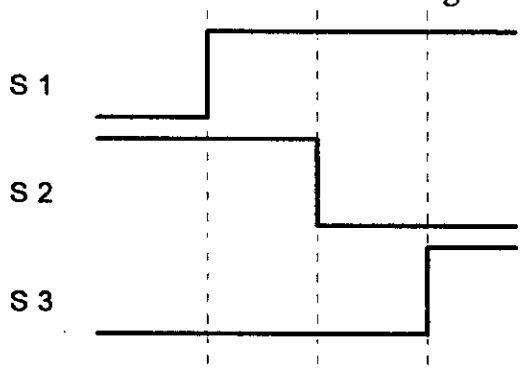


Figure 2.2.1: Génération du MSB

2.2.2 LSB DE L'ADC A 2 BITS

Il faut trois signaux de Heaviside décalés et inversé de signe de la manière suivante:



En les additionnant on aura la signal suivant:

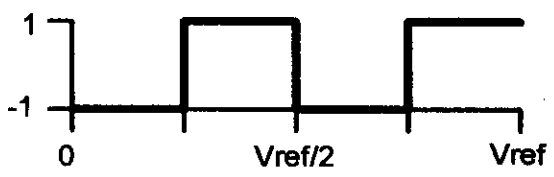


Figure 2.2.2: Génération du signal du LSB de l'ADC à 2 bits

Qui est, exactement, le signal désiré.

2.2.3 LSB DE L'ADC A 3 BITS

Dans ce cas on a besoin de sept signaux, en les additionnant on aura le signal carré du LSB de l'ADC à 3 bits comme le montre la figure suivante:

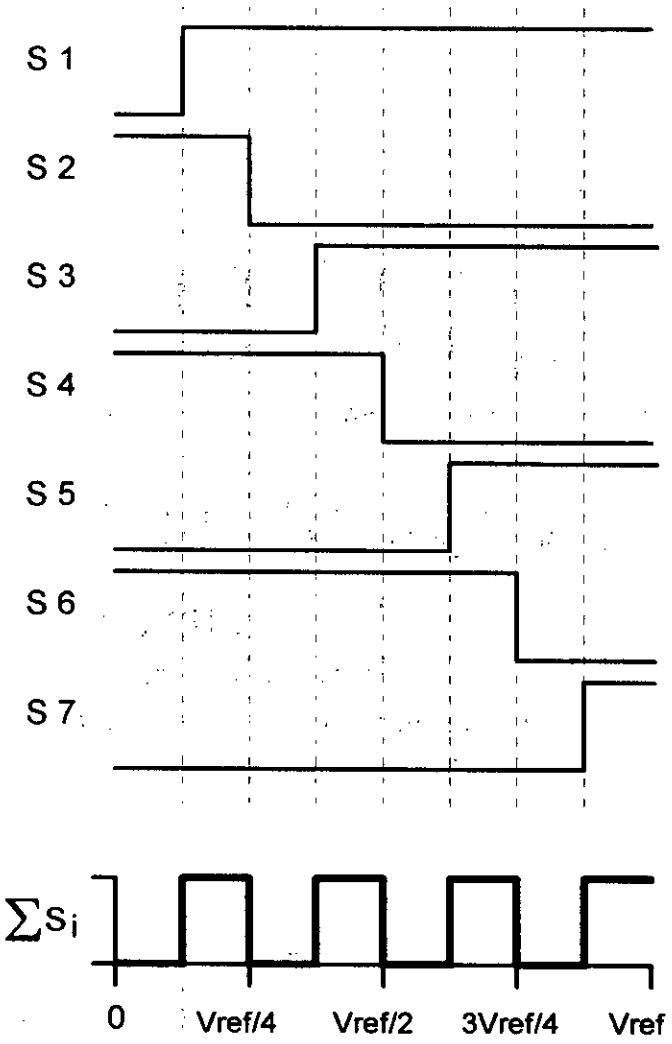


Figure 2.2.3: Génération du signal du LSB d'un ADC à 3 bits

2.2.4 LSB DE L'ADC A 4 BITS

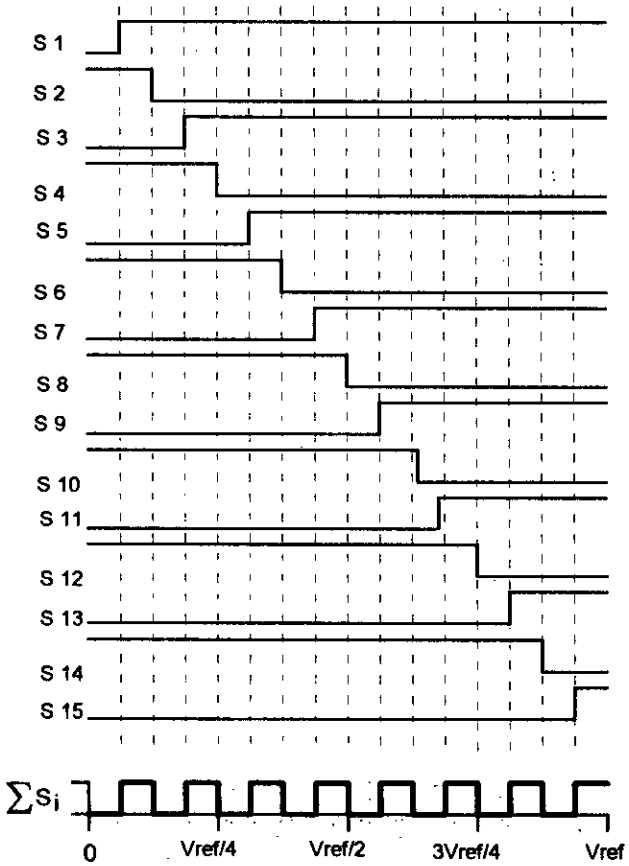


Figure 2.2.4: Génération du signal du LSB d'un ADC à 4 bits

2.2.5 LSB DE L'ADC A n BITS

En raisonnant par récurrence on peut générer le signal du LSB d'un ADC à n bits; il faut 2^n -1 signaux:

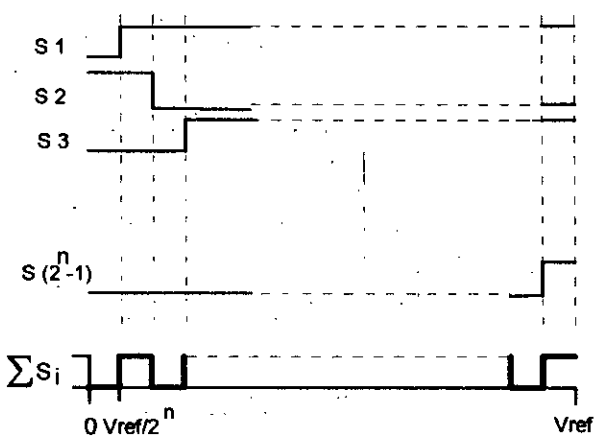
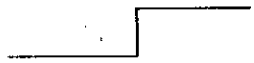


Figure 2.2.5: Génération du signal du LSB d'un ADC à n bits

2.3 MODELISATION PAR RESEAU DE NEURONES

Les signaux utilisés pour la génération du signal carré peuvent être délivrés par des neurones dont la fonction d'activation est de Heaviside.

- Le décalage est assuré par l'entrée de biais.
- Le poids de l'entrée est soit 1, soit -1 selon la forme du signal:

1 pour le signal de la forme 

-1 pour le signal de la forme 

La somme des signaux donne le signal carré: Elle est assurée par un neurone dont tous les poids sont égaux à l'unité et la fonction d'activation est linéaire.

2.3.1 DEDUCTION DES AUTRES BITS A PARTIR DU RESEAU DU LSB

On retrouve les signaux nécessaires à la génération du LSB de l'ADC à 3 bits (qui devient ici MSB -2) et ceux du LSB de l'ADC à 2 bits (ici MSB -1), ainsi que celui du MSB, dans les signaux du LSB de l'ADC à 4 bits, et cela à quelques signes près.

2.3.2 GENERALISATION A LA SYNTHESE D'UN ADC A n BITS

La procédure pour la conception d'un ADC à n bits est la suivante:

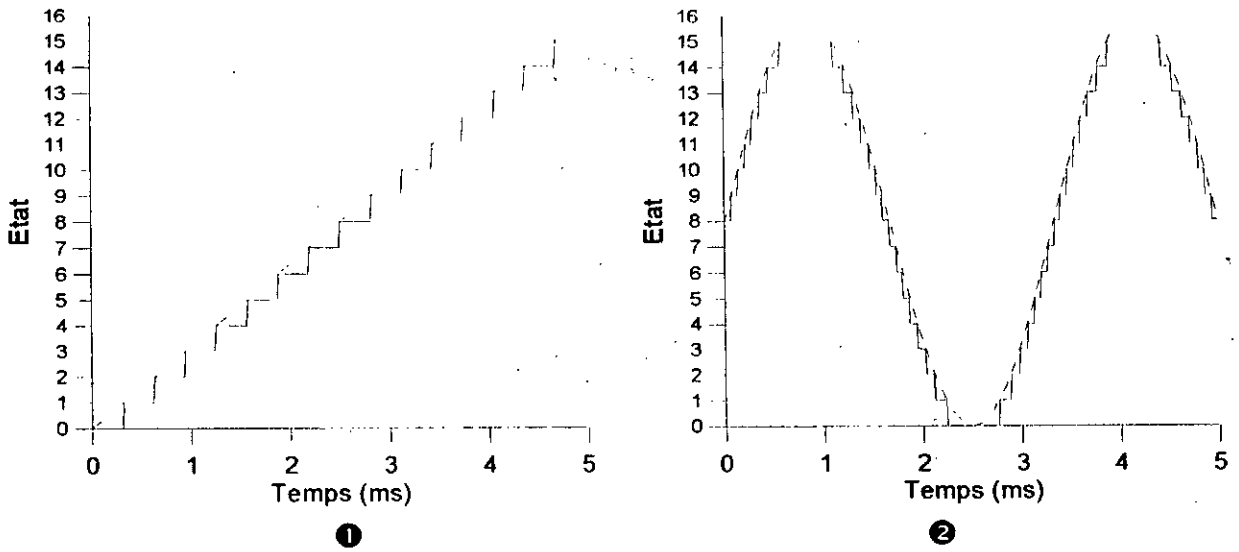
- Faire le réseau du LSB (cf. 2.3 et 2.2.4)
- Pour les bits restants suivre l'algorithme suivant:

Début
Initialisation: m =1, K = 1
1: Pour le LSB-m prendre les signaux dont le terme de biais = $KV_{ref}/2^{n-m}$
K = K+1: Tant que terme de biais inférieur à V_{ref} faire:
Pour K impair poids = -1 sinon poids = 1
Injecter les signaux pondérés dans le neurone de sortie
Si m<n aller à 1
Fin.

2.4 SIMULATIONS

On applique toujours les mêmes tests que la partie 1.4 de la section précédente (Nous utiliserons la même légende).

❑ ***ADC à 4 bits***



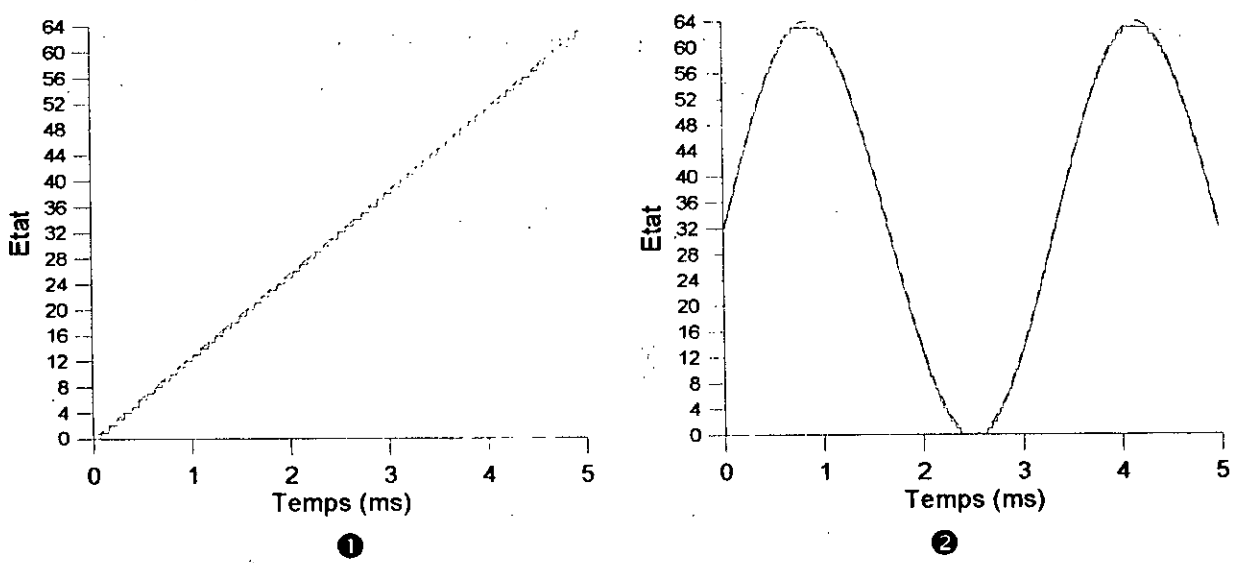
❶ ❷

Figure 2.4.1: Réponse de l'ADC à 4 bits (après conversion D/A de ses sorties)

❶ à un signal rampe

❷ à un signal sinusoïdal

❑ ***ADC à 6 bits***



❶ ❷

Figure 2.4.2: Réponse de l'ADC à 6 bits (après conversion D/A de ses sorties)

❶ à un signal rampe

❷ à un signal sinusoïdal

❑ ***ADC à 8 bits***

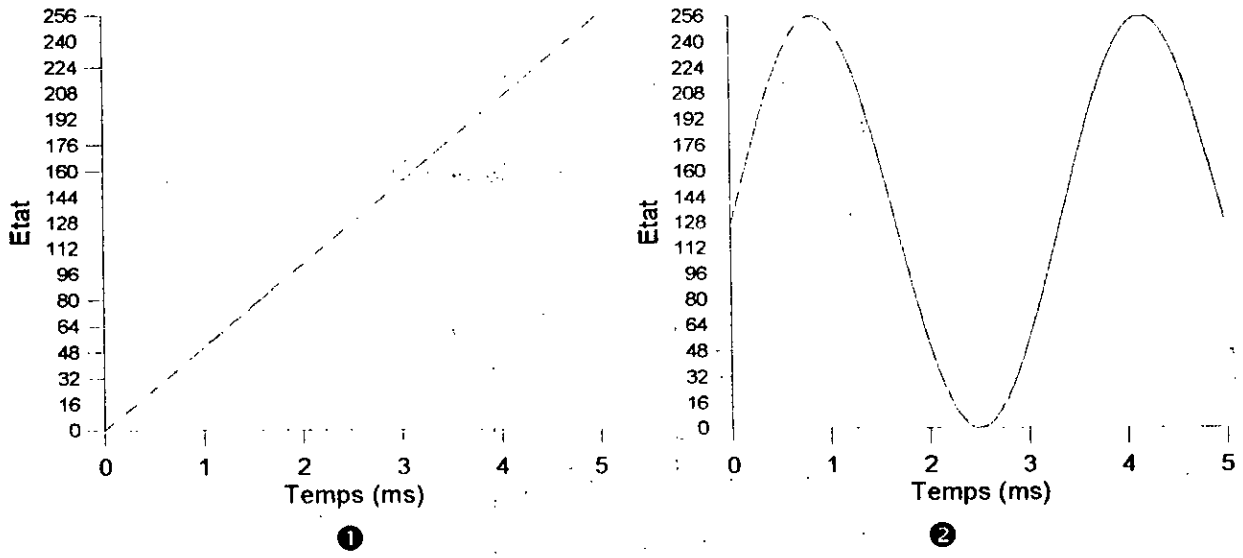


Figure 2.4.3: Réponse de l'ADC à 8 bits (après conversion D/A de ses sorties)

❶ à un signal rampe

❷ à un signal sinusoïdal

2.5 TABLEAU COMPARATIF

Pour un ADC à 4 bits.

Approche	Nbre de neurones dans la couche cachée	Nbre de connexions	Accessoire	Rapidité[2]	Temps de d'apprentissage
1ère approche	80	484	aucun	rapide	Très énorme
ADC Gray	16	100	décodeur Gray/Binaire	rapide	moyen
2ème approche[3]	28	87	aucun	_	énorme
3ème approche[4]	15	55	aucun	ultra-rapide	0

[2] cf section 4 § 4.5: conclusion.

[3] Pas de résultats faute de temps.

2.6 CONCLUSION

Cette nouvelle approche pour la conception d'ADCs neuronaux présente des avantages énormes par rapport aux autres méthodes: La conception est simple, il n'y a aucun apprentissage, on peut augmenter à volonté la résolution , rapide...

En plus la réalisation électronique s'avère simple: le nombre de connexions et le nombre de neurones sont minimes, les poids sont faciles à implémenter (voir partie 2 section 4).

[4] Sans apprentissage.

SECTION

3

ETUDE THEORIQUE SUR LES AMP OPS: APPLICATION A LA CONCEPTION ET REALISATION D'UN NEURONE

3.1 INTRODUCTION

Initialement destinés à la réalisation des calculateurs analogiques, les amplificateurs opérationnels (amp ops) ont vite élargi leur champ d'applications, on les trouve désormais un peu partout que ce soit dans le monde de l'électronique, d'instrumentation, de l'automatisme et bien d'autres domaines qu'on ne pourra citer exhaustivement. C'est donc un composant universel qui peut réaliser plusieurs opérations telles que l'amplification, la conversion courant-tension et tension-courant, le calcul opérationnel, la génération de fonctions etc...

On le trouve également dans les réseaux de neurones, où il constitue l'élément de base dans l'implémentation de ces derniers (comme on le verra au § 3.6 de cette Section).

3.2 DEFINITIONS

❑ ***Qu'est ce qu'un amplificateur opérationnel ?***

C'est essentiellement un amplificateur différentiel (lorsque il fonctionne en boucle ouverte) amplifiant les signaux qui lui sont appliqués, avec un gain théoriquement infini (en pratique: 10^5 ou 10^6).

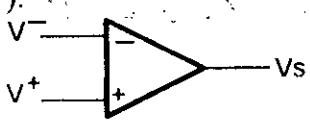


Figure 2.1.1: symbole universel de l'amp op

❑ ***Notations***

A^+: Gain positif en boucle ouverte.

A^-: Gain négatif en boucle ouverte.

Dans le cas idéal $A^+ = A^-$.

V^+: Tension à la borne positive.

V^-: Tension à la borne négative.

I_b^+: Courant de biais d'entrée positive.

I_b^-: Courant de biais d'entrée négative.

V_S: Tension de sortie: $V_S = A^+V^+ - A^-V^-$.

❑ ***Nomenclature***

- Tension en mode commun (V_{mc})

$$V_{mc} = \frac{V^+ - V^-}{2}$$

- Tension différentielle (V_d)

$$V_d = V^+ - V^-$$

- Gain de mode commun (A_{mc})

$$A_{mc} = A^+ - A^-$$

- Gain de mode différentielle

$$A_d = \frac{A^+ + A^-}{2}$$

- Taux de réjection en mode commun (CMMR)

$$CMMR = \frac{A_d}{A_{mc}} \quad \text{et en décibels} \quad CMMR = 20\log\frac{A_d}{A_{mc}}$$

On remarque que plus le CMMR est grand, plus la réjection de la tension en mode commun est bonne. Dans le cas idéal le CMMR est infini.

- Tension d'offset (V_{os})

$$V_{os} = A_d(V^+ - V^-)$$

- Courant de décalage (I_b)

$$I_b = \frac{I_b^+ + I_b^-}{2}$$

- Courant d'offset (I_{os})

$$I_{os} = I_b^+ - I_b^-$$

3.3 LES DEUX COMMANDEMENTS DES AMP OPS

En d'autres mots c'est les règles d'or nécessaires à la compréhension du fonctionnement des amp ops.

1- La sortie essaye tant qu'il est nécessaire de maintenir la différence de tensions entre les deux entrées nulle.

2- L'amp op fait osciller sa sortie à travers le réseau externe de retour afin de conduire si possible l'entrée différentielle à zéro (masse virtuelle).

3.4 CIRCUITS DE BASE

3.4.1 AMP OP INVERSEUR

Son schéma est le suivant:

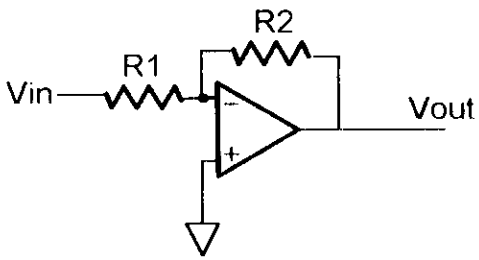


Figure 3.4.1: Amp op inverseur

On le réalise de la manière suivante:

1- La borne (+) est reliée à la masse ce qui implique selon la règle 1 que la borne (-) l'est tout autant.

2- Cela veut dire que le potentiel appliqué à R_2 est V_{out} et celui à R_1 est V_{in}.

3- En utilisant la règle 2 on aura

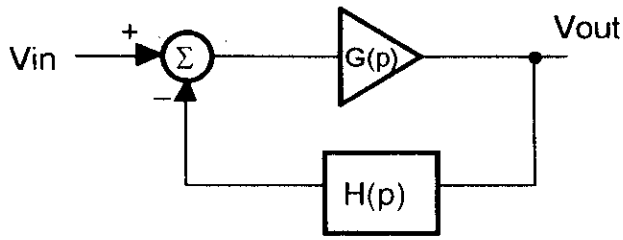
$$\frac{V_{out}}{R_2} = -\frac{V_{in}}{R_1}$$

d'où

$$V_{out} = -\frac{R_2}{R_1} V_{in}$$

3.4.2 AMP OP NON INVERSEUR

Considérons un amp op en boucle fermée suivant la représentation générale de la figure ci dessus. G est la réponse en boucle ouverte, et H une fonction de transfert de retour.



L'équation décrivant la réponse globale est donnée par:

$$\frac{V_{out}(p)}{V_{in}(p)} = \frac{G(p)}{1+G(p)H(p)}$$

Si le produit G(p)H(p)>>1, alors $V_{out}(p)/V_{in}(p) \approx 1/H(p)$. Cela revient à dire que la réponse globale peut être indépendante des variations de G(p) et dépend pratiquement que de la fonction de retour H(p).

Exemple:

Considérons la fonction de retour constituée par deux résistance montées en diviseur de tension. On aura le circuit suivant, appelé montage non-inverseur,

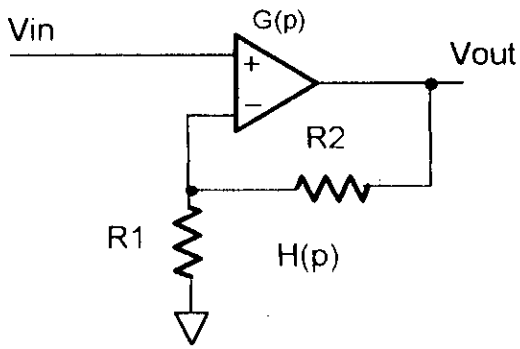


Figure 3.4.2: Amp op non-inverseur

La fonction H sera exprimée comme suit,

$$H(p) = \frac{R_1}{R_1 + R_2}$$

d'où on déduit la fonction de réponse globale,

$$A(p) = \frac{V_{out}(p)}{V_{in}(p)} = \frac{1}{H(p)} = 1 + \frac{R_2}{R_1}$$

3.4.3 LE SUIVEUR

C'est un amp op non-inverseur dont R_1 est infinie et R_2 nulle ce qui donne

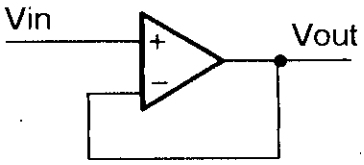


Figure 3.4.3: Le suiveur

L'amp op suiveur est parfois appelé buffer (tampon) à cause de ses propriétés isolatrices: grande impédance d'entrée et faible impédance de sortie.

3.4.4 LE SOMMATEUR

Le circuit de la figure 3.4.4 est un sommateur inverseur.

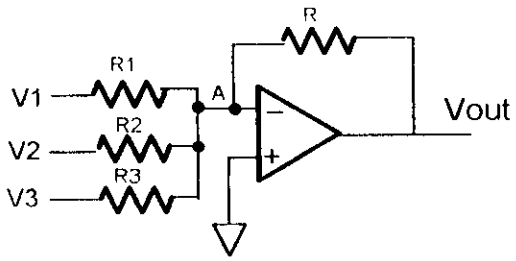


Figure 3.4.4: Amp op sommateur

Le point A étant la masse (la somme des courant à ce point est nulle) c. à. d.,

$$\frac{V_1}{R_1}+\frac{V_2}{R_2}+\frac{V_3}{R_3}=-\frac{V_{out}}{R}$$

d'où

$$V_{out}=-R\left(\frac{V_1}{R_1}+\frac{V_2}{R_2}+\frac{V_3}{R_3}\right)$$

3.4.5 AMP OP DIFFERENTIEL

Le circuit de la figure 3.4.5 représente un amp différentiel classique

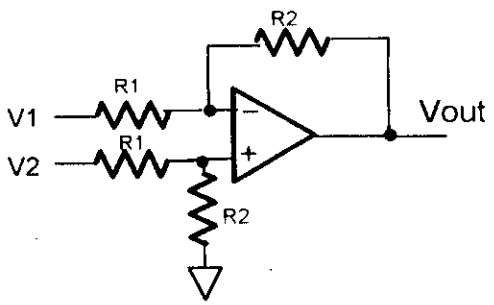


Figure 3.4.5: Amp op différentiel

En appliquant les règles d'or on aura:

$$V_{out} = \frac{R_2}{R_1}(V_2 - V_1)$$

3.4.6 AMP INTEGRATEUR

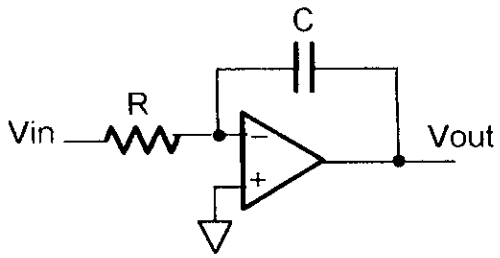


Figure 3.4.6: Amp intégrateur

d'après la règle 2

$$\frac{V_{in}}{R} = -C\left(\frac{dV_{out}}{dt}\right)$$

ce qui donne

$$V_{out} = -\frac{1}{RC}\int V_{in}dt + c^{ste}$$

Ce typr d'intégrateur pose le problème de conditions initiales, pour le résoudre voici quelques circuits pour la remise à zéro de l'intégrateur [Hor 89]:

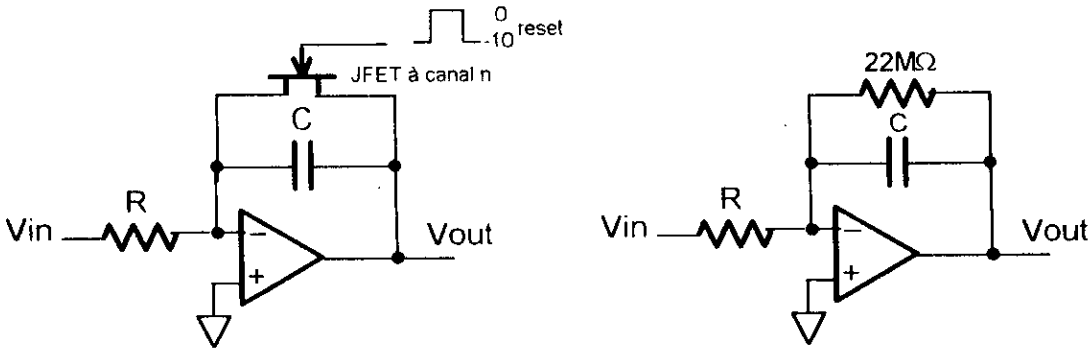


Figure 3.4.7: Circuits de remise à zéro de l'intégrateur

3.4.7 AMP OP DERIVATEUR

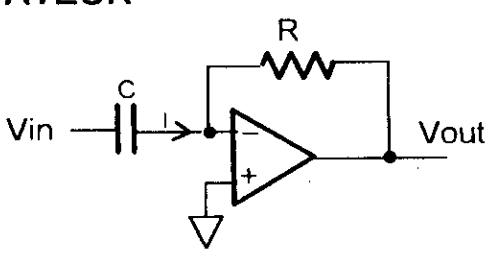


Figure 3.4.8: Amp dérivateur

En intervertant la résistance et la capacité de l'amp intégrateur on obtient l'amp dérivateur.

V_{in} produit un courant $i = C\frac{dV_{in}}{dt}$ qui est convertit en tension à travers la résistance R ce qui donne

$$V_{out} = -RC\frac{dV_{in}}{dt}$$

3.4.8 AMP OP LOGARITHMIQUE

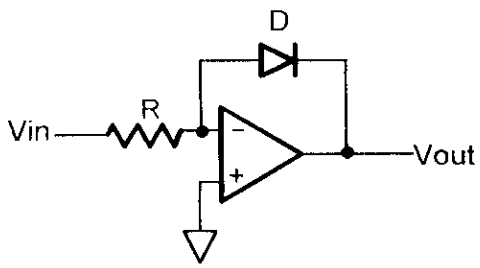


Figure 3.4.9. Amp op logarithmique

Un amplificateur opérationnel inverseur avec une diode en contre réaction donne un amplificateur logarithmique ou bien générateur de la fonction logarithmique, cela est du au fait que le courant passant par la diode a pour expression

$$I = I_s\left(\exp\frac{V_d}{V_T} - 1\right)$$

avec $V_T = kT/e$, où:
k: Constante de Boltzmann
e: Charge d'électron
T: température ambiante
I_S: Courant de saturation de la diode
V_d: Tension aux bornes de la diode

V_T étant généralement très faible l'expression du courant devient $I=I_S.\exp(V_{out}/V_T)$, l'amp op étant idéal,

$$V_{out} = -V_d = -V_T \ln\frac{I}{I_s}$$

$$\Rightarrow V_{out} = -V_T \ln\frac{V_{in}}{RI_s}$$

On remarque que ce circuit est très sensible à la température, pour palier à ce problème des circuits de compensation ont été proposé dans [Hor 89].

3.4.9 L'AMP OP EXPONENTIEL

En intervertant la diode D et la résistance R de l'amp log on obtient l'amplificateur exponentiel.

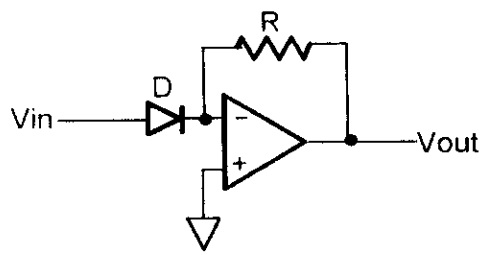


Figure 3.4.10: Amp exponentiel

D'après le schéma

$$V_{out} = -RI$$

$$= -RI_s \exp\frac{V_d}{V_T}$$

$$V_{out} = -RI_s \exp\frac{V_{in}}{V_T}$$

3.4.10 LE COMPARATEUR

En électronique, on est souvent appelé à comparer des tensions. La forme la plus simple réalisant cette fonction est un amp op en boucle ouverte.

$$V_{out} = A(V^+ - V^-)$$

Comme A est très grand (théoriquement infini) la sortie V_{out} bascule de la saturation la plus faible (-Vcc) à la saturation la plus forte (+Vcc)[1] selon la différence entre les deux tension appliquées aux entrées.

[1] -Vcc, +Vcc: Tensions d'alimentation de l'amp op.

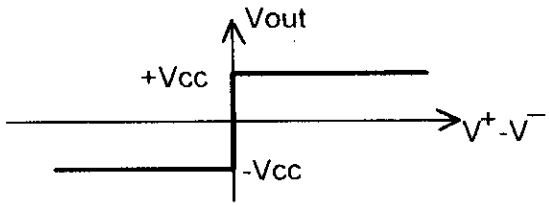


Figure 3.4.11: Caractéristique du comparateur

3.5 ETUDE DES IMPERFECTIONS DES AMP OPS

3.5.1 MODELE IDEAL D'UN AMP OP

Les caractéristique d'un amp op idéal sont les suivantes

1- Impédance d'entrée en mode différentiel ou commun infinie.
2- Impédance de sortie en boucle ouverte nulle.
3- Gain en mode différentiel infini.
4- Gain en mode commun nul.
5- Si $V^+ = V^-$ alors $V_{out} = 0$ ($V_{os} = 0$).
6- Changement instantané de la sortie (slew rate infini).

Toutes ces caractéristiques sont indépendantes de la température et la variation de la tension d'alimentation.

3.5.2 MODELE REEL

Mais en réalité nul n'est parfait! même pas les amp ops...

❑ ***Courant de biais (courant de polarisation moyen)***

D'abord on définit les courant de polarisation I_b^+ (d'entré positive) et I_b^- (d'entrée négative), comme les courant de polarisation de bases des deux transistors d'entrée de l'amp op , le courant de biais est la moyenne de ces deux derniers

$$I_b = \frac{I_b^+ + I_b^-}{2}$$

Ces courants sont de l'ordre du nA, mais ils produisent des erreurs de l'ordre du µV pou une source d'impédance inférieure à 1kΩ (Exemple: l'OP-77E -bipolaire- à 25°C, I_b =2µA. L'AD549 -JFET- à 25°C, I_b=0.06pA).

❑ ***Courant de décalage (courant d'offset)***

C'est la différence entre I_b^+ et I_b^-:

$$I_{os} = I_b^+ - I_b^-$$

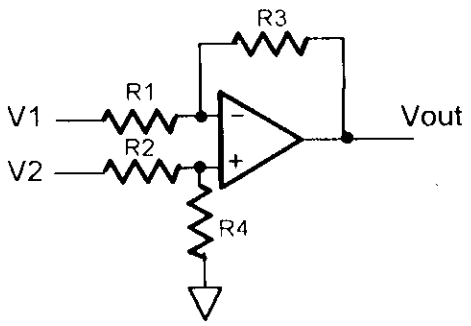
Ce courant caractérise la symétrie du circuit d'entrée de l'amp op. Sa signification est que quand l'amp op est alimenté par des sources d'impédances identiques, il observe une différence de tension entre ces entrées (même si elles n'existent pas) [Hor 89].

❑ ***Tension d'offset V_{os}***

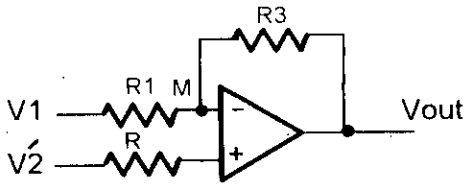
C'est une conséquence du déséquilibre de la paire différentielle (qui se compose de deux transistors bipolaires ou FETs). C'est une source d'erreur évidente pour la tension de sortie.

Expression de la tension de sortie avec prise en compte des imperfections:

En général la montage de l'amp op est le suivant.



Ce schéma est équivalent à



avec $V_2' = V_2 \dfrac{R_4}{R_4 + R_2}$ et $R = \dfrac{R_2 R_4}{R_2 + R_4}$ (théorème de Thevenin).

On a les équations suivantes:

l'impédance -source. Dans ce cas on peut résoudre ce problème par la mise en place, à l'entrée, soit d'un suiveur soit d'une impédance (résistance) très grande.

❑ ***Impédance de sortie***

En pratique elle de l'ordre de quelques Ohms (40 pour le LF411CN) en boucle ouverte, le retour (Feed-Back) la diminue énormément, voir même l'annule.

❑ ***Common Mode Rejection Ratio (CMRR)***

Les tensions en mode commun, présentées à l'entrée de l'amplificateur, perturbent la sortie de celui ci. C'est pour cette raison qu'il faut choisir un amp op à CMRR élevé. Une deuxième lecture est qu'un CMRR insuffisant dégrade la précision de l'amp op en introduisant une tension d'offset plus au moins importante.

A l'encontre de l'amplificateur non-inverseur, l'amplificateur inverseur est insensible à la valeur du CMRR.

❑ ***Slew rate (Taux de variation de la tension d'entrée)***

Il exprime en combien de temps la sortie varie en réponse à une grande variation de l'entrée (Exemple: pour le LF411CN S.R. = 15V/µs). Le slew rate limite le basculement de la sortie aux hautes fréquences.

Si l'amp op est utilisé en comparateur, il est recommandé qu'il ait un slew rate assez grand.

❑ ***Influence de la température***

Tous les paramètres étudiés ci dessus dépendent de la température.

3.6 APPLICATION: CONCEPTION ET REALISATION D'UN NEURONE

L'implémentation Software des réseaux de neurones offre tellement d'avantage au point où on peut se demander sur l'utilité des réseaux Hardware.

Mais en comparant avantages et inconvénients de l'une et de l'autre forme des réseaux, on pourra réaliser l'importance de l'implémentation Hardware. Car en effet c'est cette dernière qui présente la modélisation la plus proche du cerveau du fait qu'elle présente l'avantage primordial qu'est le parallélisme (les réseaux de neurones Software sont implémentés dans des ordinateurs... machines d'architecture séquentielle).

Dans cette partie on se propose de concevoir et de réaliser le neurone formel électronique.

3.6.1 MODELE DU NEURONE A REALISER

C'est le même explicité dans la 1ère partie, Section 2, §2.3.2 et qui peut être représenter par le schéma suivant:

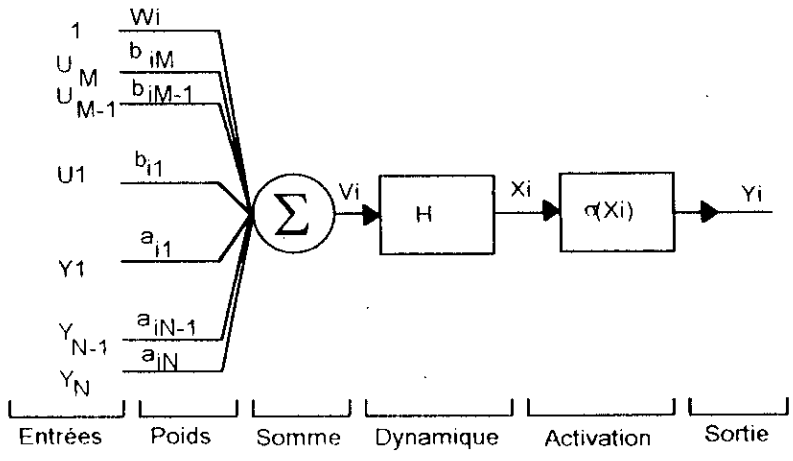


Figure 3.6.1: Modèle du neurone à réaliser

Le modèle étant défini, il nous reste à trouver la forme électronique adéquate.

3.6.2 CONCEPTION DES DIFFERENTS ELEMENTS DU NEURONE

3.6.2.1 Les entrées/sorties

Elles seront des tensions électriques.

3.6.2.2 Poids et sommateur pondéré

Pour les poids on utilisera des impédances (particulièrement des résistances), qui selon la loi d'Ohm, pondéreront les tensions d'entrées et les convertiront -par la même occasion- en courants qui seront des entrées du sommateur qui est un amp op, monté en amplificateur différentiel -voir §3.4.5 dans cette section-.

Les entrées de poids positifs sont injectées dans l'entrée positive et celles de poids négatifs sont injectées dans l'entrée négative.

Certaines dispositions (voir pour la résistance ρ plus bas) sont à prendre pour que ce circuit assure sa fonction (sommation pondérée).

Le schéma du circuit ainsi obtenu est le suivant:

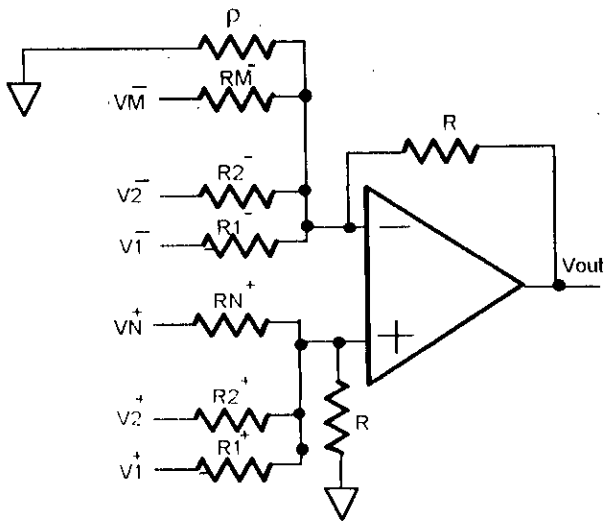


Figure 3.6.2: Sommateur pondéré

❑ ***Calcul de*** ρ

ρ peut être connectée soit à la borne (+) soit à la borne (-), selon une règle qu'on va établir. Pour commencer on la connectera à la borne (-):

D'après la règle 2:

Pour la borne (+) $$\sum_{i=1}^{N}\frac{V_i^+}{R_i^+} - V^+\sum_{i=1}^{N}\frac{1}{R_i^+} = \frac{V^+}{R} \qquad (3.6.1)$$

Pour la borne (-) $$\frac{V - V_{out}}{R} = \sum_{j=1}^{M}\frac{V_j}{R_i} - V\left[\frac{1}{\rho} + \sum_{i=1}^{M}\frac{1}{R_i}\right] \qquad (3.6.2)$$

(3.6.2) donne $$\frac{V_{out}}{R} = -\sum_{j=1}^{M}\frac{V_j}{R_j} + V\left[\frac{1}{R} + \frac{1}{\rho} + \sum_{j=1}^{M}\frac{1}{R_j}\right] \qquad (3.6.3)$$

et (3.6.1) donne $$\sum_{i=1}^{N}\frac{V_i}{R_i} = V\left[\frac{1}{R} + \sum_{i=1}^{N}\frac{1}{R_i}\right] \qquad (3.6.4)$$

Comme $V^+ = V^-$ et en combinant (3.6.3) et (3.6.4) nous aurons:

$$\frac{V_{out}}{R} = \frac{\sum_{i=1}^{N} \frac{V_i^+}{R_i^+}}{\frac{1}{R} + \frac{1}{\rho} + \sum_{j=1}^{M} \frac{1}{R_j^-}} - \frac{\sum_{j=1}^{M} \frac{V_j^-}{R_j^-}}{\frac{1}{R} + \sum_{i=1}^{N} \frac{1}{R_i^+}}$$

et nous voulons avoir

$$\begin{cases} V_{out} = \sum_{i=1}^{N} W_i V_i - \sum_{j=1}^{M} W_j V_j \\ W_i = \frac{R}{R_i} \quad \text{et} \quad W_j = \frac{R}{R_j} \end{cases}$$

donc il faut que

$$\frac{1}{R} + \frac{1}{\rho} + \sum_{j=1}^{M} \frac{1}{R_j^-} = \frac{1}{R} + \sum_{i=1}^{N} \frac{1}{R_i^+}$$

On voit que si:

- $\sum_{j=1}^{M} \frac{1}{R_j} = \sum_{i=1}^{N} \frac{1}{R_i}$ alors ρ est infinie (n'est pas utilisée).
- $\sum_{j=1}^{M} \frac{1}{R_j} < \sum_{i=1}^{N} \frac{1}{R_i}$ alors ρ doit être connecté à la borne (+).

Enfin sa valeur est donnée par:

$$\rho = \begin{cases} \infty & \text{si} \quad \sum_{j=1}^{M} \frac{1}{R_j} = \sum_{i=1}^{N} \frac{1}{R_i} \\ \frac{1}{\left| \sum_{j=1}^{M} \frac{1}{R_j} - \sum_{i=1}^{N} \frac{1}{R_i} \right|} & \text{sinon} \end{cases}$$

Remarque: Si tous les poids ont le même signe, on utilise, tout simplement un sommateur inverseur (conseillé surtout si tous les poids sont négatifs).

3.6.2.3 La dynamique

On utilise généralement un circuit RC (R//C)

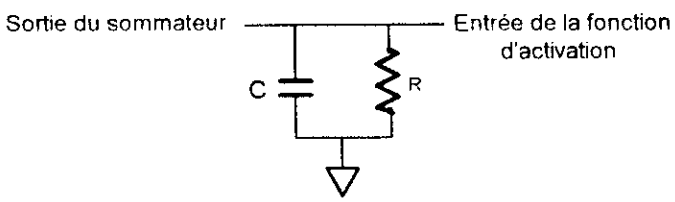


Figure 3.6.2: Dynamique du neurone

Pour les réseaux statique H(s) = 1, l'étage de la dynamique est supprimée.

3.6.2.4 La fonction d'activation

La fonction d'activation qu'on a choisie pour le neurone réalisé était la tangente hyperbolique. Du moins dans la pratique on ne peut générer parfaitement cette fonction à cause des imperfections des composants électroniques, mais faut-il savoir qu'il est suffisant de générer une fonction en s (sigmoïde) puisque même la tanh est une approche de la vraie fonction d'activation du neurone biologique.

Il existe plusieurs méthodes pour la synthèse de circuits de génération de fonctions:

❑ ***Génération par variation de la résistance de contre réaction***

L'entrée en conduction de chaque diode est commandée par V_{out} (voir figure 3.6.4)[Buh 79].

Exemple:

D1 conduit à partir de l'instant où $V_{out} = \frac{R_{1a}}{R_{1b}} V_0$

D'une manière générale chaque diode D_k entre en conduction à partir de:

$$V_{out} = \frac{R_{ka}}{R_{kb}} V_0$$

A chaque fois qu'une diode entre en conduction la courbe s'incline (variation de la pente), car la résistance de contre réaction équivalente change. La tension V_{out} en fonction de V_{in} est représenté à la figure 3.6.3, on remarque que pour ce circuit on ne peut avoir une saturation parfaite car la résistance de contre réaction globale ne peut être nulle.

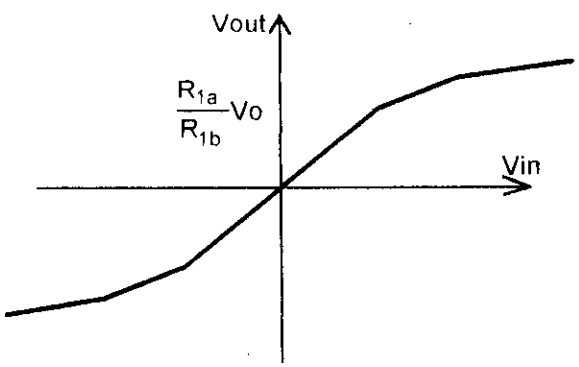


Figure 3.6.3: Caractéristique du circuit générateur la fonction tanh (par variation de la résistance de réaction)

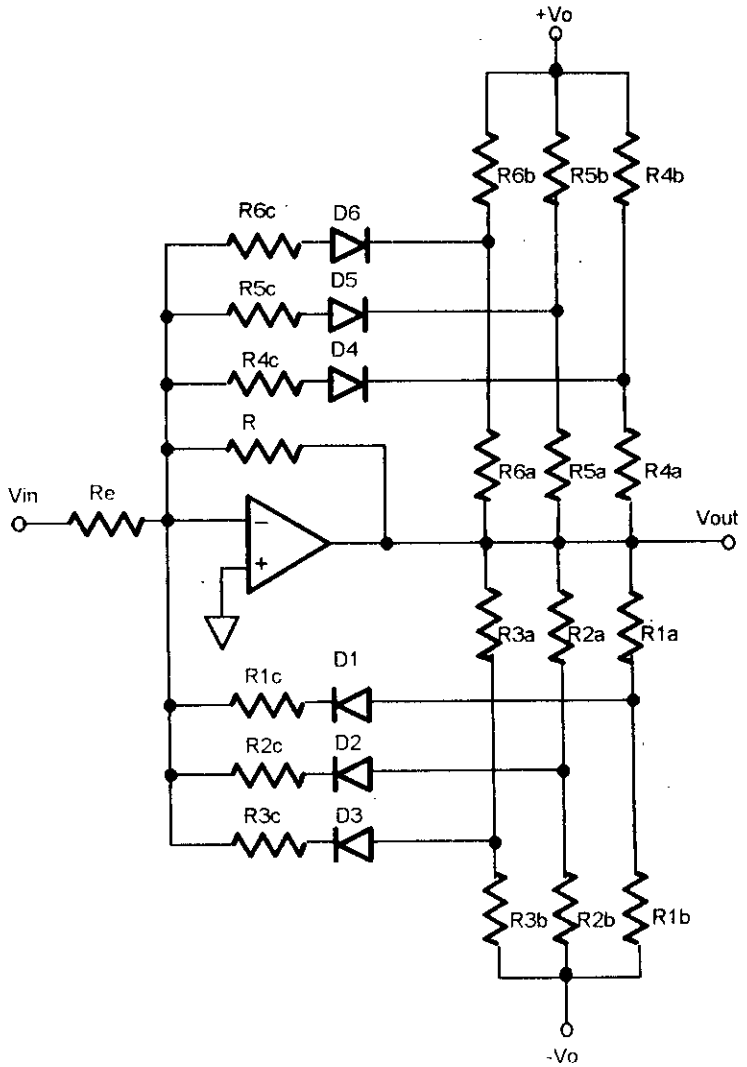


Figure 3.6.4: Générateur de la fonction tanh par variation de la résistance de contre réaction

❑ ***Génération à l'aide d'amplificateurs exponentiels et logarithmiques***

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

Le circuit obtenu (voir figure 3.6.5) consomme beaucoup d'amp ops (8 pour un seul neurone!), en plus il est très sensible à la température. En effet les valeurs de certaines résistances sont choisies en fonction soit de V_T soit de I_S; deux grandeurs très sensibles à la température:

$$V_T = \frac{kT}{e}$$

$$I_s = \beta T^3 \exp\left(\frac{-\delta w}{kT}\right)$$

avec $\delta w = 1.2eV$

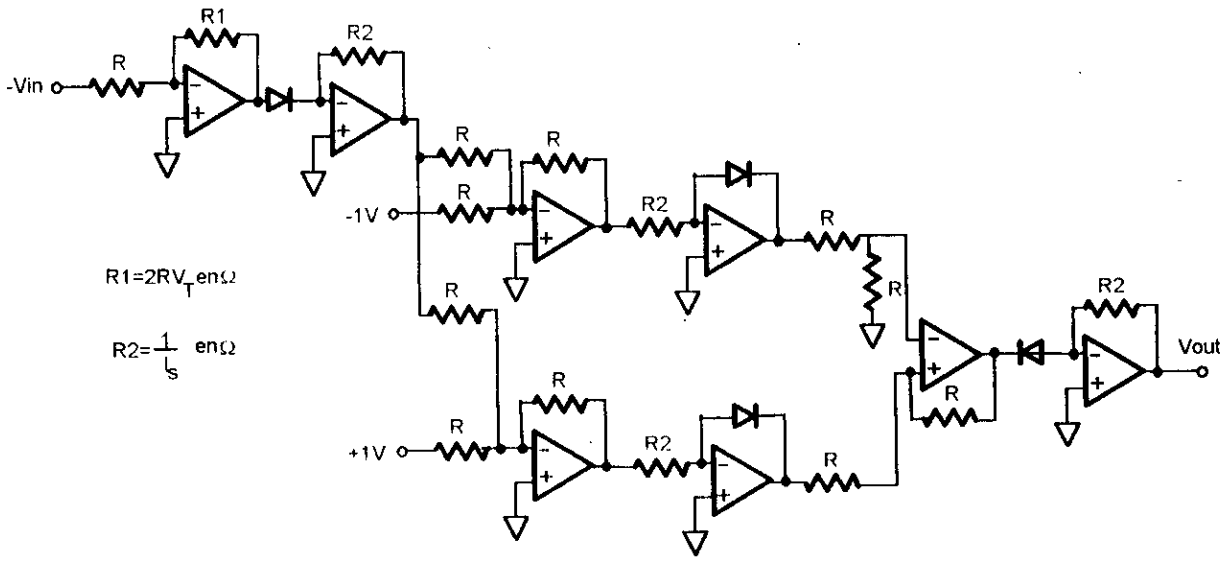


Figure 3.6.5: Générateur de la fonction tanh à l'aide d'amplificateur logarithmiques et exponentiels

❑ ***Génération par sommation des courant***

Principe: l'élément de base de ce montage est un circuit simple qui peut être considéré comme une diode fictive dont on peut varier à volonté la tension de seuil et la résistance dynamique. Plusieurs éléments sont regroupés en parallèle, le courant résultant sera converti en tension par un amp op inverseur. On agira sur les tensions de seuil et la résistance dynamique des diodes fictives de façon à ce que la tension de sortie soit une interpolation linéaire de la fonction tangente hyperbolique.

Circuits élémentaires[Cha 84]

On utilisera deux types de circuits:

-Circuit de type 1:

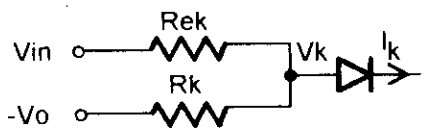


Figure 3.6.6: Circuit de type 1

$$V_k = \frac{R_k}{R_k + R_{ek}} V_{in} - \frac{R_{ek}}{R_k + R_{ek}} V_0$$

La diode conduit si $V_k \geq U_S$ (U_S tension de conduction de la diode)
La tension appliquée à ce circuit pour que $V_k = U_S$ est

$$V_{ek} = \frac{R_{ek}}{R_k} V_0 + \frac{R_{ek} + R_k}{R_k} U_s \qquad (3.6.5)$$

pour $V_{in} > V_{ek}$ $\qquad I_k = \frac{V_{in} - V_{ek}}{R_{ek}}$

de l'équation (3.6.5) on trouve

$$R_k = R_{ek}\left(\frac{U_s + V_0}{V_{ek} - U_s}\right)$$

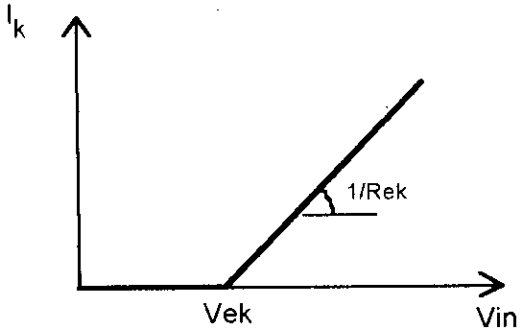


Figure 3.6.7: Caractéristique du circuit de type 1

Circuit de type 2:

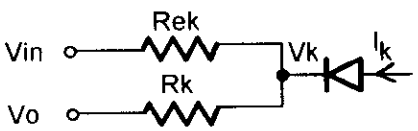


Figure 3.6.8: Circuit de type 2

De la même manière on a

$$V_k = \frac{R_k}{R_k + R_{ek}} V_{in} + \frac{R_{ek}}{R_k + R_{ek}} V_0$$

$$V_{ek} = -\frac{R_{ek}}{R_k} V_0 - \frac{R_{ek} + R_k}{R_k} U_s \qquad (3.6.6)$$

pour $V_{in} \leq V_{ek}$, $I_k = \dfrac{V_{in} - V_{ek}}{R_{ek}}$

De (3.6.6) on trouve:

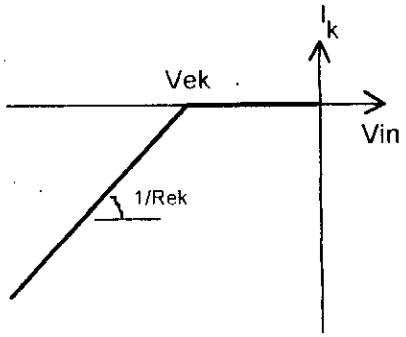


Figure 3.6.9: Caractéristique du circuit de type 2

On remarque bien que la caractéristique Entrée/Sortie de ces circuit est semblable à celle d'une diode dont on a imposé la résistance dynamique R_{ek} et la tension de seuil V_{ek}.

R_{ek} définit la pente de la caractéristique (imposée). V_{ek} est imposé en choisissant adéquatement R_k.

Le circuit final régissant la fonction est le suivant:

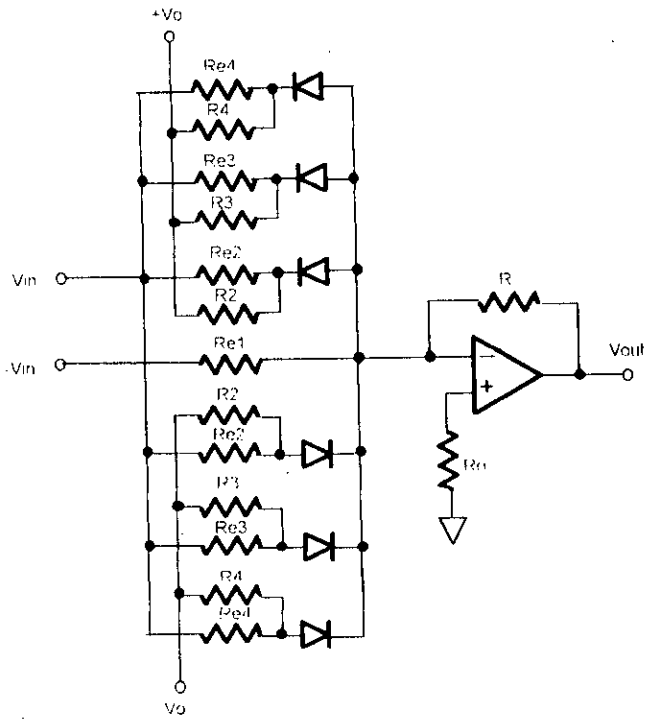


Figure 3.6.10: Générateur de la fonction tanh par la méthode de sommation des courants

Les différents paramètres sont définit par l'interpolation désirée qui est choisi linéaire comme suit (voir figure 3.6.11 à l'appui).

- $0 \le V_{in} \le V_{in2}$. $V_s = \frac{R}{R_e} V_{in}$: la pente est $tg\theta = \frac{R}{R_e}$
- $V_{in2} \le V_{in} \le V_{in3}$. la pente devient $tg\alpha = tg\theta - \frac{R}{R_{e2}}$
- $V_{in3} \le V_{in} \le V_{in4}$. la pente devient $tg\beta \quad tg\alpha - \frac{R}{R_{e3}}$
- $V_{in4} \le V_{in} \le \infty$. la pente sera $tg\varphi = tg\beta - \frac{R}{R_{e4}} = 0$

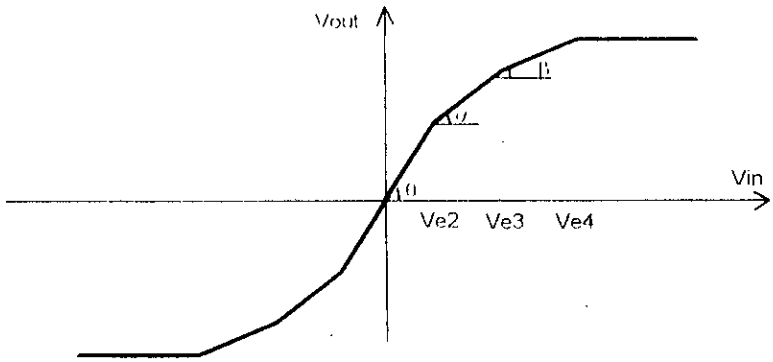


Figure 3.6.11: Points d'interpolation choisis pour la réalisation

3.6.2.5 Réalisation du circuit

Dans notre réalisation on a utilisé pour générer la fonction d'activation (tangente hyperbolique) la méthode de sommation des courants. Le choix des segments de l'interpolation se fait afin de reproduire avec une bonne fidélité la fonction tanh c'est à dire, que dans chaque intervalle tanh est pratiquement linéaire.

On a pris trois points d'interpolation $V_{e2} = 0.85$, $V_{e3} = 1.543$ et $V_{e4} = 2.63$. Tout calcul fait, on obtient le tableau suivant:

Résistance	Valeur Calculée(kΩ)	Valeur correspondante pour une précision de (kΩ)		
		1%	2%	5%
Re1	1	1	1	1*
Re2	1.47	1.47	1.47	1.5*
Re3	4.02	4.02	4.02	3.9*
Re4	14.09	14	14	15*
R2	87.14	86.6	86.6	91*
R3	49.13	48.7	47* ou 48.7	51
R4	76.37	76.8	75	75*

* Valeurs trouvées sur le marché.

Test pratique

Après avoir réaliser le neurone sur maquette nous avons relevé la caractéristique $V_{out} = f(V_{in})$ (figure 3.6.12).

Les résultats obtenus sont satisfaisants puisque la caractéristique a une forme sigmoïde très proche de la fonction tanh (les écarts sont dus à l'interpolation et aux imperfections des composants).

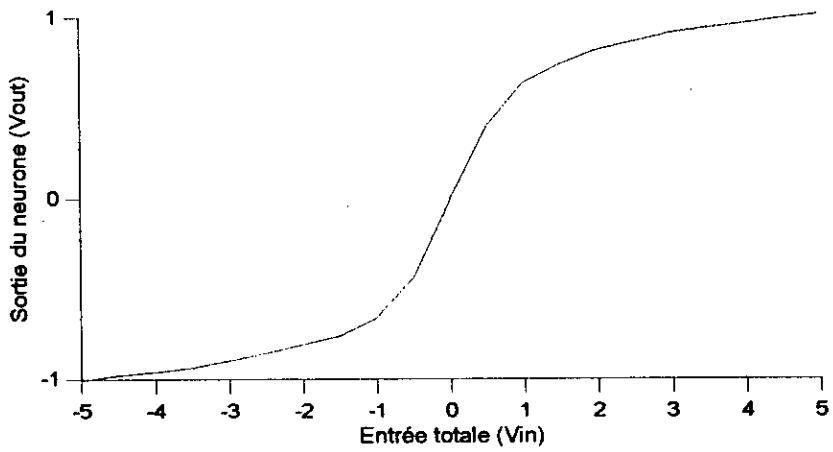


Figure 3.6.12: Caractéristique pratique du générateur de la fonction d'activation (méthode de sommation des courants)

3.7 CONCLUSIONS

les réseaux de neurones Hardware sont le modèle qui reflète le plus l'architecture du cerveau. Mais pour l'édifier, il faut d'abord réaliser son élément de base "le neurone".

La modélisation d'un neurone nécessite d'assez bonnes notions de neurophysiologie; mais une fois qu'on a le modèle, c'est à l'électronique non-linéaire et à ses composants qu'on fait appel, afin d'implémenter le neurone.

Cependant la réalisation, proprement dite, pose certains problèmes dont essentiellement celui de la précision des composants (résistances, tension de référence, amp ops...) et celui de biais entre les valeurs des résistances calculées et celles commercialisées qui suivent une certaine norme. Mais le neurone doit être très précis, pour donner de bons résultats en réseau. Il faut donc trouver un bon compromis précision-coût... la précision coûtant chère.

SECTION 4

REALISATION PRATIQUE

4.1 INTRODUCTION

D'après le tableau comparatif (cf. 2ème partie, Section 2, §2.5), le modèle de l'ADC neuronal choisi pour la réalisation est celui de l'approche "3", pour tous les avantages qu'il présente (rapidité, simplicité -est donc coût-, précision -MSE=0-...).

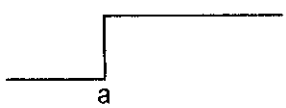
Dans cette section, on présentera les modèles de neurones utilisés et les circuits électroniques les réalisants, ~~ainsi que le schéma électrique de l'ADC~~. On relèvera après la caractéristique Entrée/Sortie de l'ADC neuronal à 4 bits réalisé.

4.2 MODELES DU NEURONE

4.2.1 NEURONES DE LA COUCHE CACHEE

La caractéristique d'un neurone de la couche cachée prend l'une des formes suivantes:

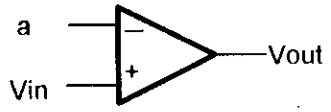
La forme 1:



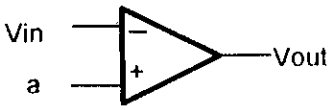
La forme 2:

La fonction d'activation est donc de Heaviside (équivalente à la tangente hyperbolique avec un gain très grand). Le circuit du neurone se trouve réduit à un comparateur, qui sera monté -suivant la forme de la caractéristique- comme suit:

La forme 1:



La forme 2:



4.2.2 NEURONES DE LA COUCHE DE SORTIE

Ici la fonction d'activation est linéaire, donc égale à la sortie du sommateur pondéré (c'est un fil).

Pour le sommateur pondéré, on utilisera pour le LSB un amp op sommateur inverseur, pour le MSB il n'y a pas de neurones de sortie (le neurone de la couche cachée donnant directement sa caractéristique) et pour les bits restants sera utilisé l'amplificateur différentiel avec ρ de même valeur que celles des résistances d'entrée (toutes identiques) -cf. 2ème partie, Section 2, §3.6.2.2-.

4.3 GENERATION DES ENTREES DE BIAIS

On utilise simplement un diviseur de tension où on utilisera 2^n résistances égales, avec n est le nombre de bits de l'ADC.

Ce diviseur sera alimenté entre V_O et V_r tel que (V_r - V_O) est le full scale de l'entrée de l'ADC.

4.4 CARACTERISTIQUE ENTREE/SORTIE

On l'a relevé à l'aide d'un multimètre mesurant la tension d'entrée. Des LEDs nous permettaient de relever la sortie (qui est sous forme d'états).La tension de référence est de 5V (V_O= 0V, V_r = 5V).

4.5 CONCLUSION

L'ADC neuronal -3ème approche- est facile à la réalisation, donne une bonne précision et est très rapide. Pour la rapidité, on a dit qu'il était plus rapide que les autres approches car dans sa couche de sortie, les neurones ne sont constitués que d'un sommateur, alors que pour les autres réseaux les neurones de sorties possèdent en plus du sommateur, l'étage de génération de la fonction d'activation.

ANNEXES

PERCEPTRON

Avez-vous entendu parler du *perceptron?* Non!..Et bien voilà:
Le perceptron (ancêtre des R.N.A.) est né vers les années 50, de l'idée d'expliquer le fonctionnement du système nerveux humain, par la modélisation de celui ci à l'aide de réseaux de neurones formels (neurones de Mc Culloch & Pitts). Ce modèle se composait de trois parties:

1- Rétine artificielle: qui convertit des signaux perçus du monde extérieur en des signaux binaires modulés suivant l'intensité du stimulus.
2- Couche d'association: formée de neurones responsables du décodage des signaux reçus de la rétine, les neurones sont connectés à la rétine, entre eux et aux neurones de la couche de sortie (appelée aussi de décision).
3- Couche de décision: formée de neurones recevants leurs entrées des neurones d'association et d'autres neurones de décision. C'est la sortie du perceptron.

Son apprentissage se fait par la règle de Hebb qui date de 1949, qui édicte que lorsque deux neurones sont activés en même temps la connexion synaptique entre eux doit être renforcée.

Le perceptron était supposé capable de reconnaître des formes avec un certain sens de nuance et même capable "d'écouter n'importe quelle conversation et prendre des notes restituées sur imprimante comme n'importe quelle secrétaire" selon Frank Rosenblatt.

Le perceptron étant à architecture trop simple pour donner de bons résultats quand les choses se compliquaient: il était incapable de généraliser ou de reconnaître un environnement à partir d'un ou plusieurs objets, de tirer une règle d'un ensemble d'éléments ou de réaliser la fonction de parité si l'un des neurones d'association n'est pas relié à toutes les cellules de la rétines (problème du XOR).
C'est Marvin Minsky et Seymour Papert en 1969 qui démontrèrent ces limites, enterrant par la même occasion Le Perceptron...[Dav90]

CARACTERISTIQUES DE QUELQUES ADCs

Type	Bits	Méthode	Temps de réponse(µs)	V_{ref} (V)	Etendu de V_{in} (V)	Prix[b] de la pièces ($)	Remarques
HS9582	6	F	0.07	ext: 1~5	0-V_{ref}	15	rapide
TDC1047	7	F	0.05	ext: -1	0-V_{ref}	40	sorties verouillées
HS9583	8	F	0.2	ext: +5	0-5	44	rapide
AD9002	8	F	0.007	ext	-2~0	90	rapide
CXA1176A	8	F	0.003	ext	-2~0	175	le plus rapide des ADCs
TDC1049	9	F	0.03	ext: -2	-2-0	_	sortie à 9 bits
HADC77600	10	F	0.02	ext: ±2	±0.5; ±0.2	_	haute résolution
ADC0820	8	DF	2.5	ext 1~5	0~V_{ref}	_	possibilité de saturation
AD7820	8	DF	1.6	ext 1~5	0~V_{ref}	10	rapide
ADC511	12	DF	1	int	0~10; ±5	9.9	pas de "mising codes"
AD9003	12	DF	1	int	0~5	250	_
CAV1220	12	DF	0.05	int	±1	2500	le plus rapide des DF
TLC548	8	AS	22	ext	0~5	_	facile à utiliser
AD7575	8	AS	5	ext 1.2	0-2V_{ref}	55	rapide
ADC1001	10	AS	200	ext 1~5	0~V_{ref}	14	rapide
AD573	10	AS	20	int	0~10	20	rapide
AD7578	12	AS	100	ext: +5	0~5	_	consomme faible puissance
AD7672	12	AS	3	ext: -5	0~5;0~10	46	_
TLC1205B	13	AS	10	ext	±5	30	_
ICL7115	14	AS	40	ext: +5	0~5	50	extension d'une ROM
ADC76	16	AS	15	int	±5;±10	100	_
ADAM-826-3	16	AS	1.5	int	0~10;±10	_	_
MN5420	20	AS	3	int	±5	_	_
CX20018	16	DR	15	int	0~5;0~10	18	_
ICL7109	12	DR	6[a]	0.2~2	0~0.2V_{ref}	_	_
CSZ5316	16	SD	20[a]	int	±2.75	_	_
AD1170	18	SD	1000[a]	int	±5	_	_

Remarques pour les notations:

F: Flash, DF: Demi Flash, AS: Approximations successives, DR: Double Rampe, SD: Sigma Delta.

a: l'unité est conversions par seconde.b: ces prix sont valables aux U.S.A en 1990.

ALGORITHME D'ALADIN
(Algorithm for Learning and Architecture DetermINation)

Dé but

Initialiser les poids à des valeurs alé atoires, donner α, γ, et E_{tot}^{0}

$n_j = 1 \ \forall j = 1,2,\ldots N$ ($n_i = 0$ le $j^{ème}$ neurone de la couche caché e inactive)

$v = 0$

1: $v \leftarrow v + 1$

$I = \{n_j = 1 : j \in [1, L]\}$ (L: nombre de neurones initial de la couche caché e)

$S_j = 0 \ \forall j \in I$

$E_{tot} = 0, \ l = 0$

2: $l = l + 1$

Donner $\mathbf{x}_l$ et $\mathbf{y}_l$

$$i_j = \sigma(\sum_{i=1}^{N+1} w_{ji}^{h} x_i) \quad \forall j \in I$$

$$o_k = \sigma(\sum_{j=1} w_{kj}^{o} i_j)$$

$$\varepsilon_k^{o} = (y_k - o_k)\sigma'(\sum_{j=1} w_{ji}^{o} i_j)$$

$$\varepsilon_j^{h} = \sigma'(\sum_{i=1}^{N+1} w_{ji}^{h} x_i)\sum_{k=1}^{M} \varepsilon_k^{o} w_{kj}^{o} \quad \forall j \in I$$

$$w_{kj}^{o} \leftarrow w_{kj}^{o} + \eta \varepsilon_k^{o} i_j \quad \forall j \in I$$

$$w_{ji}^{h} \leftarrow w_{ji}^{h} + \eta \varepsilon_j^{h} x_i \quad \forall j \in I$$

$$E = \frac{1}{2}\sum_{k=1}^{M} (y_k - o_k)^2$$

$$S_j(E) = \left| \frac{i_j}{E} \sum_{k=1}^{M} \varepsilon_k^{o} w_{kj}^{o} \right| \quad \forall j \in I$$

$$S_j \leftarrow S_j + S_j(E)$$

$$E_{tot} \leftarrow E_{tot} + E$$

Si $l < P$ aller à 2 (P: nombre d'exemples)

Evaluer la moyenne μ_s et la variance σ_s de S_j, $j \in I$

Si $S_j < \mu_s - \gamma\sigma_s$ ou $S_j > \mu_s + \gamma\sigma_s$ alors $n_j = 0 \ \forall j \in I$

Si $E_{tot} > E_{tot}^{0}$ aller à 1

Fin

BIBLIOGRAPHIE

Ouvrages

[Att 94] **M. Attari**, *Capteurs et instrumentation électronique*, Alger, Institut d'éléctronique USTHB, 1994.

[Bes1 78] **P. Bessou**, *Physiologie humaine: le système nerveux (tome 1)*, SIMEP Editions, 1978.

[Bes2 79] **P. Bessou**, *Physiologie humaine: le système nerveux (tome 2)*, SIMEP Editions, 1979

[Buh 79] **H. Bühler**, *Eléctronique de réglage et de commande*, Paris, Dunod, 1979.

[Cha 84] **J.D. Chatelain**, *Eléctronique*, Lausanne, Presse Polytechnique Romande, 1984.

[Cla 80] **G.B. Clayton**, *Data converters*, London, Butterworth, 1980.

[Cou 80] **R. Coujard**, *Précis d'histologie*, Paris, Masson, 1980.

[Cno 83] **J.C. Cnockaert et D. Saidi**, *Physiologie du neurone*, Alger; OPU, 1983.

[Dav 90] **E. Davalo et P. Naïm**, *Des réseaux de neurones*, Paris, Eyrolles, 1990.

[Fre 92] **J.A. Freeman & D.M. Skapura**, *Neural networks*, New York, Addison Wesley, Juillet 1992.

[Has 81] **M. Hasler et J. Neirynck**, *Filtres éléctriques*, Paris, Dunod, 1981.

[Hor 89] **P. Horowitz & W.Hill**, *The Art of Electronics (2nd edition)*, New York / Port Chester / Melbourne / sydney, Cambridge Press University, 1989.

[Kar 90] **N.B. Karayanis & A.N. Venetsanpoules**, *Artificial neural network: learning algorithms, performance, evaluation and applications*, Boston/ Dordiecht / London, Klumer Academic Publisher, 1990.

[Kos 92] **B. Kosko**, *Neural networks for signal processing*, New Jersey, Prentice Hall, 1992.

[Kun 81] **M. Kunt**, *Traitement numérique des signaux*, Paris, Dunod, 1981.

[Min 69] **M. Minsky and S. Papert**, Perceptrons, MIT press, 1969.

Articles

[Avi 91] **G. Avitable & al.**, " On a class of non symetrical neural networks with application of ADC ", IEEE Trans, Février 1991, Vol. Cas-38, N°2, pp 202-209.

[Bau 89] **E. B. Baum**, " What size net gives valid generalisation ? ", Neural computation N°1, 1989; pp151-160.

[Ber 95] **A. Bernieri & al.**, " ADC neural modeling ", IEEE instrumentation and measurement society, Avril 1995, ISBN: 0-7803-2615-6195, pp 789-794.

[Bet 93] **G. Bethian & al.**, " An association of simulated annealingand electrical simulator Spice-Pac for learning of analog neural betworks ", IEEE Proceedings 1993, ISBN: 1066-1409193, pp 254-259.

[Dap 95] **P. Daponte & al.**, " A full neural Gray coded based ADC ", IEEE instrumentation and measurement society, Avril 1995, ISBN: 0-7803-2615-6195, pp 834-839.

[Gir 92] **F. Girosi and T. Poggio,** " Representation properties of networks Kolmogorov's theorem is irrelevant ", Neural computation 1, pp 465-469.

[Hop 82] **J.J. Hopfield**, " Neural networks and physical systems with emergent collective computational activity ",Proceedings of National Academy of Sciences, U.S.A, 1982, Vol. 79, pp 2554-2558.

[Hop 84] **J.J. Hopfield**, " Neurons with graded response have collective computationnal properties like those of two state neurons ", Proc. Natl. Acad. Sci. U.S.A., May 1984, Vol. 81, pp 3088-3092.

[Mar 91] **G. Martinnelli and R. Perfetti**, " Synthetesis of feedforward neural analog-digital converters ", IEE Proceedings, Octobre 1991, Vol 138, N°5, pp 567-574.

[Min 69] **M. Minsky and S. Papert**, Perceptrons, MIT press, 1969.

[Nor 89] **Norio Baba**, " A new approach for finding the global minimum of error function of neural networks ", Neural Networks, Mars 1989, Vol N°2, pp 367-373.

[Ona 95] **B. M. Onat & al.**, " Implementation ao a charge based neural Euclidien classifier for a 3 bit Flash analog to digital converter ", Proc IEEE Instrumentation and measurement Technology conference , Avril 1995, ISBN: 0-7803-2615-6195, pp 834-839.

[Sol 81] **F. J. Solis and R. J-B Wets**, " Minimization by random search techniques ", Mathematics of operations research, Fevrier 1981, Vol:6 N°1, pp 19-29.

[Tan 86] **D. W. Tank and J. J. Hopfield**, " Simple neural optimization networks: An A/D converter, signal decision circuit and systems ", Mai 1986, Vol, CAS-33, N°5, pp 533-541.

[Wer 90] **P. J. Werbos**, " Backpropagation through time: What it does and how to do it ? ", Proc IEEE, Octobre 1990, Vol 18 N°10, pp 1550-1560.

[Wid1 90] **B. Widrow**, "30 years of adaptive neural networks: Perceptron, Madaline and Backpropagation ", Proc IEEE, Vol 78, N°9, pp 1415-1442.

Théses et Mini-projet

[B&B 95] **M. Bouhedda & S. Bouallag**, *" Etude et réalisation d'un neurone "*, Ecole Nationale Polytechnique/ Département Génie Eléctrique/ Option Automatique, Mini-Projet, Juin 1995.

[Ham 95] **B. Hamzi & S.Labiod**, *" Identification et commande des systèmes dynamiques par les réseaux de neurones "*, Alger, Ecole Nationale Polytechnique / Département Génie Eléctrique/ Option Automatique, Thése d'ingéniorat, Juin 1995.

[Hen 94] **M. Henniche**, *" Sur la linéarisation des capteurs par les réseaux de neurones artificiels "*, Alger, Ecole Nationale Polytechnique / Département Génie Eléctrique/ Option Automatique, Thése d'ingéniorat, Juin 1994.

[Yed 94] **Y.M. Yazid & Dj. Senouci-Briksi**, *" Etude comparative entre le traitement d'images par les méthodes classiques et par les réseaux de neurones "*, Alger, Ecole Nationale Polytechnique / Département Génie Eléctrique/ Option Automatique, Thése d'ingéniorat, Juin 1994.

Revues

[S&V 91] Science & Vie, Hors série N°117, Décembre 1991.

[SVM 90] Science & Vie Micro, N°65, Février 1990.

ملخص

الغاية من هذا العمل هو وضع طريقة جديدة للتحويل المستمر/الرقمي الفا نق السرعة. لهذا الغرض أبدت الشبكات العصبية الإصطناعية نجاعتها، فقد جربت عدة صيغ (مخارج حسب النظام الثنائي، مخارج حسب نظام قراي، شبكة ديناميكية) معطية نتائج مرضية نسبيا. في النهاية، اقترحت هندسة جديدة للمحولات العصبية المستمرة/الرقمية ذات ن مخرج بدون تعلم. برهنت إستعماليةهذه الهندسة الجديدة بإنشاء محول مستمر/رقمي عصبي ذو أربع مخارج الذي أعطى النتائج المنشودة.

كلمات مفتاحية: محول مستمر/رقمي، تحويل، تمثيل، تعلم، تعميم، الشبكات العصبية الإصطناعية،مضخم عملي، قراي، البت الأكبر قيمة، البت الأصغرقيمة.

ABSTRACT

The aim of this work is to set up a new high-speed Analog to digital converter. The artificial neural networks offer good abilities for this application, then several approaches(binary outputs, Gray outputs, dynamic network,...) bordering to relatively satisfying results were used . At the end, a new architecture allowing neural n bits ADC's synthesis without learning was elaborated. The feasability of this new approach was proved by the realization of a four bits neural ADC, which gives the desired results.

Key words: ADC, Conversion, Modelisation, Learning, Generalisation, Artificial Neural Networks, Operationnal Amplifier, Gray, MSB, LSB.

RESUME

La finalité de ce travail était d'établir une nouvelle structure pour la conversion A/D ultra-rapide. A cet effet les réseaux de neurones artificiels se sont bien prêtés; plusieurs approches (sortie binaire, sortie Gray, réseau dynamique...) ont été utilisées aboutissant à des résultats relativement satisfaisants. A la fin a été élaboré une nouvelle architecture permettant la synthèse d'un ADC neuronal à n bits sans aucun apprentissage. La faisabilité de cette nouvelle architecture a été prouvée par la réalisation d'un ADC neuronal à quatre bits qui donna les résultats désirés.

Mots clefs: ADC, Conversion, Apprentissage, Généralisation, Réseaux de neurones artificiels, Amplificateur opérationnel, Gray, MSB, LSB.