République Algérienne Démocratique et populaire.

Université Dr Yahia Fares de Médea

Mémoire

En vue de l'obtention du diplôme de : MASTER

Spécialité : Mécatronique

Thème :

COMMANDE NEUROMIMETIQUE D'UN ROBOT MOBILE

Réalisé par :

LECHAB YASSER

BELKAIS Mohamed

Encadré par :

Dr R.kara

Membres du Jury :

Mr N.TOUBAL	**Université de Médea**	**Président**
Dr M.BACHEN	**Université de Médea**	**Examinateur**
Mr B.BENGHERBIA	**Université de Médea**	**Examinateur**

- 2014 -

REMERCIEMENT

On tien à remarier tout particulièrement Monsieur Dr R.KARA, nous directeur de mémoire , pour nous avoir confie et dirigé ce projet, pour ces remarque et ces critique ,et qui n'a jamais manqué de nous conseiller et de nous orienter tout au longe de la réalisation de ce travaille, ainsi que Dr M. BOUHEDA. qu'ils trouve ici l'expression de nous respecte et de nous profonde reconnaissance.

A tout nos enseignants et plus particulièrement Mr S.TCHOUKECH, Dr BOUKHLEF , Mr B.BENGHARBIA, Mr N.TOUBEL,MLLE ABBADI.

Nous remercie sincèrement les paressent de laboratoire 03-AUTO :Mr SKANDER ,Mr YAHYA et Mr ZOHIR

Ma profonde gratitude s'adresse à mansion M.BACHEN de m'avoir accordé l'honneur d'intégrer la faculté et de nos avoir facilité l'intégration et la reprise de nous étude après 15années pour Y.LECHAB et après 2années pour M.BELKAIS

De Y.LACHEB : Je remercie mes parents.ma petit femme, et ma lumière –Ma fille SARAH

De M.BELKAIS : Je remercie mes parents, mes frères et tout qui m'encourager

Resume

Cette thèse est présentent dans le cadre d'obtenir un diplôme master mécatronique dans le but de commander neuronale d'un robot mobile omnidirectionnelle par un réseau de neurone CMAC .donc l'objectif de cette thèse et de faire un programme matlab pour RN CMAC et injecter ce programme ou robot.

Historiquement premier réseau de neurone artificielle est réalisé en 1943, aujourd'hui il existe plusieurs architecteur pour les réseaux de neurones artificielle, chaque de ces architecteur ayons leur caractéristiques ainsi leur règles d'apprentissages.

Dans cette thèse , une présentent générale sur les réseaux de neurones artificielle et leur calcification, ainsi l'explication des différents règles d'apprentissages des réseau de neurone artificiel, Une étude préciser sur le réseau CMAC, Le modèle de réseau neuronal CMAC est basé sur le fonctionnement biologique du cervelet. Le cervelet est en charge de tous les mouvements du corps et de leur précision. Il est capable de résoudre plusieurs problèmes qui se posent dans le contrôle de systèmes dynamiques, comme ceux de la robotique, donc l'étude est basé après d'une présentation de la philosophé, sur les déférant propriété et les règles d'apprentissages de CMAC.

L'implantation de RN CMAC ou robot pose un problème de taille et de temps de calcule telle que le microcontrôleur de robot ne supporter pas la taille de CMAC donc comme une solution une carte est réalisé pour stuquer et transformée les informations entre une PC et le robot ,les résultats obtenez pour des différents trajectoires de robot par deus commande traditionnelles avec programmation c++ et par commande neuronale avec model CMAC

executive Summary

This thesis is presented in the context of obtaining a master degree in mechatronics order to control neuronal an omnidirectional mobile robot by CMAC neural network. Therefore the objective of this thesis and make a matlab program for RN CMAC and inject this program or robot.

Historically first network of artificial neuron is achieved in 1943, today there are several architecteur for artificial neural networks, each of these architecteur have their characteristics and their learning rules.

In this thesis, a general present on neural networks and their calcification and the explanation of the different learning rules of artificial neuron network, specify a study on the CMAC network, the CMAC neural network model is based on the biological functioning of the cerebellum. The cerebellum is responsible for all body movements and precision. It is able to solve several problems in the control of dynamic systems, such as robotics, so the study is based after a presentation of philosophized on the property and deferring the rules of learning CMAC .

The implementation of RN CMAC robot or a problem size and computation time as the robot microcontroller does not support the size of CMAC thus as a solution card is made for stucco and transform information between a PC and the robot, get the results for different trajectories deus traditional robot control programming with c + + and neural control with CMAC model

ملخص

ملخص

تقدم هذه الأطروحة في سياق الحصول على درجة الماستر الميكاترونيك للسيطرة العصبية على الروبوت المتحرك ، وبالتالي فإن الهدف من هذه الأطروحة هو الشبكة العصبية اصطناعية على مبرمج ماتلاب وحقن هذا البرنامج في . الروبوت إنشاء

تم إنشاء أول شبكة من الخلايا العصبية الاصطناعية تاريخيا في عام 1943، واليوم هناك العديد من تصاميم للشبكات .العصبية الاصطناعية، ولكل منها خصائصها وقواعد تعلم جاصة بها

في هذه الأطروحة، نظرة عامة عن الشبكات العصبية الاصطناعية وشرح قواعد التعلم المختلفة في شبكة الخلايا العصبية الاصطناعية.

كما تقدم دراسة معمقة على نموذج الشبكة العصبية الاصطناعية CMAC ، يستندCMAC على مبدء العمل لشبكة المخيخ. المخيخ هو المسؤول عن جميع حركات الجسم والدقة. أنها قادرة على حل العديد من المشاكل في السيطرة على . الأنظمة الديناميكية، مثل الروبوتات، لذلك تقوم الدراسة بعد عرض فلسفة هدا النموذج قواعد تعلم CMAC

ان حقن البرنامج في الروبوت يطرح مشكلة تتمثل في حجم البرنامج و وقت الاستجابة من طرف متحكم الروبوت لدا .الطراز CMAC وجب انشاء بطاقة لاجل تجزين وتحويل المعلومات بين جهاز كمبيوتر و الروبوت

Sommaire

Liste des figures0 0

Introduction générale 04

CHAPITRE I : introduction aux réseaux de neurones

Introduction 06

Le neurone seul

I- Historique 07

II- Du neurone biologique au neurone artificiel 08

II-1 Le neurone biologique 08

II-2 Le neurone artificiel 08

III Neurone formel 09

III -1 Définition 09

III -2 Le neurone en équation 09

III -3 Les défiants types de fonction d'activation 10

Réseau de neurone

I- Réseau de neurone 12

II- Bio-ingénierie 12

III- Comparaisons avec l'intelligence artificielle classique 13

III -1 L'intelligence artificielle classique 13

III -2 Bref histoire de l'intelligence artificielle classique 13

III -3 Différences 14

Classification topologique

I- Les réseaux feed-forward (Non bouclés) 15

I-1 Les perceptrons 15

I-2 Les réseaux à fonction radiale (RBF) 15

II- Les réseaux feed-back (récurrents) 16

II-1Les cartes auto-organisatrices de Kohonen 16

II-2 Les réseaux de Hopfieid (RNF) 17

II-3 Les ART (Adaptive résonance Theory) 17

II-4 Les réseaux à mémoire associative 18

II-5 Les réseaux à compétition 20

III- Récapitulatif 20

Apprentissage

IV-Apprentissage par adaptation 21

IV-1 Paradigmes d'Apprentissage 21

a) Apprentissage supervisé 22

b) Apprentissage par renforcement 24

c) Apprentissage non supervisé 26

IV-2 Adaptation du réseau au cours du temps

27

IV-3 Règles d'Apprentissage 28

a) Règle de hebb 28

b) Règle de correction d'erreur 28

c) Apprentissage de boltzmann 29

d) Règle d'Apprentissage par compétition 29

V- Classification des déférant Apprentissage 30

Application

I. Les réseau feed-forward(non bouclés) 31

I.1. Le perceptron multicouche 31

I.2. Le réseau à fonction raiale (RBF) 31

II. Les réseau feed-back (récurrent) 32

II-1. La carte auto organisé de kohonen 32

II-2. Le réseau de hopfield (RNF) 32

III. Hybrid 32

IV. Autre application 32

V. Conclusion 34

CHPITRE II : programmation de réseau de neurones CMAC

I. Introduction 35

II. Qu'est ce qu'un Réseau Neuronal CMAC? 36

a)CMAC comme un perceptron classique 36

b) Différence entre un RN CMAC et perceptron multicouche? 37

c) L'algorithme du CMAC 37

III. L'apprentissage du CMAC 45

VI. Propriétés du RN CMAC 54

a) Capacité d'approximation des fonctions non-linéaires du RN CMAC 54

b) Discontinuité dans la mise en correspondance 54

c) Généralisation locale 54

d) Facilité pour l'implantation en matériel 56

V. CMAC comme approximateur de fonction 57

IV. Conclusion et Discussion 61

CHAPITRE *III :* implémentation de la commande neuronale sur la plate-forme de robot mobile

Introduction 62

I- Description du robot 63

I-1 Description mécanique 63

a) Type de plateforme 63

b) Matériaux de la plateforme 64

c) Les roues 64

d) *Le bloc moteur* 65

I-2 Modélisation du robot ED-7275 66

I-2 -1 Modélisation cinématique 67

I-2-1- MODELISATION DYNAMIQUE 69

I-3 Description électronique 74

I-3-1 Caractéristique du microcontrôleur ATmega128 75

I-3-2 : Caractéristiques des capteurs 83

a) Capteur CNY70(IR) 83

b) Capteur de distance PSD 84

II- Le robot ED-7275 suiveur de ligne 85

II-1 - Le principe de la détection de la ligne 85

III- Application 90

ANNAXE A orrigéne de CMAC

ANNAXE B déplacement des couche de CMAC

BIBIOGRAPHE

REFERENCES

figure 1-1 : Neurone biologique 7
figure.1-2 : Neurone artificiel 7
figure.1-3 : Neurone formel de MacCulloch et Pitts 8
figure.1-4 : Neurone à seuil 9
figure.1-5 : Fonction binaire 9
figure.1-6 : Fonction linéaire par morceaux 10
figure.1-7 :Fonction linéaire 10
figure.1-8 :Fonction sigmoïde 11
ifgure.1-9 : Fonction gaussienne 11
figure.1-10 : Neurone greffé à une puce 12
figure .1-11 : schéma I-R-R 13
figure.1-1 2 : Perceptron mono- couche 15
figure.1-13 :Perceptron multicouche 15
figure .1-14 : Modèle de Kohonen (SOM) 16
figure.1-15 :Réseau de Hop- field 17
figure.1-16 :Réseau ART 17
Figure 1-17 : Architecture d'un réseau à mémoire associative. 18
figure.1-18 :Les réseaux à compétition 19
figure.1-19 : Classification topologique 20
Figure 1.20 : l'apprentissage supervisé 22
Figure. 1-21: l'apprentissage par renforcement. 24
Figure. 1-22: L'apprentissage non supervisé 25
FIGURE. 20 – Demande de prêt Perceptron multicouche 32
figure 2-1 : Perceptron classique du cervelet. 36
FIgURE 2.2 : Principe de généralisation locale dans un CMAC à une dimension 39
Figure 2.3 : Structure d'un CMAC à 2 dimensions 4**0**
figure 2.4: Principe de généralisation dans un CMAC à 2 dimensions 41

figure2-5 :principe de la technique « Memory Hashing » représentant les deux catégories de collisions 44

figure2-6 :Illustration du phénoméne d 'interférence 47

figure. 2.7 - Diagramme général du fonctionnement du RN CMAC MIMO 48

figure. 2.8- Exemple d'un RN CM AC avec deux entré 49

figure 2.9 : cellule de detection activé par l entrée x1=0.42 5

figure 2.10 Mise en forme de la 1er Couche 51

figure (2.11) : Mise en forme de la 2ème Couches 52

figure (2.12) : Mise en forme des 5 Couches 52

Figure2-13 :La réponse du CMAC pour deux 55

Figure2-14 :L 'influence du paramétre de généralisation sur l'apprentissage du réseau 56

Figue 3-1 : le Robot 63

Figue 3-2 : forme de plateforme d'ED7275 63

Figue3-3 : Roue du robot 64

Figuer3-4: Caractéristiques des moteurs IG-32RGM 66

Figure 3-4 : représentation de modèle cinématique 67

Figure 3-5 – Modèle électrique du moteur à courant continu 71

Figure3-6 : Modèle en simulation du Moteur à Courant Continu 73

Figure 3-7 :l'architecteur générale de robot ED7275 75

Figure 3-8 Trame USART 78

Figure3-9 :atmega128 82

Figuer3-10 : plaque présenté les cinq IR de ED7275 83

Figue3-11: chemin présenter le principe de fonctionnement de PSD 84

Figure 3-12 : Système simplifié de suivi de ligne mis en place 86

Figure 3-13 : Organigramme du suivie de la ligne avec 3 capteurs IR 86

Figure 3-14 : Deuxième Algorithme 88

Figure 3-15 : Seul le capteur central détecte la ligne 88

Figure 3-16 : Seul le capteur 1er latéral droit détecte la ligne 88

Figure 3-17 : Seul le dernier capteur droit détecte la ligne 89

Figure 3-18: trajectoire triangulaire 90

Figure 3-19: Courbe d'erreur sur trajectoire triangulaire 91

Figure 3-20 : trajectoire demi cercle 91

Figure 3-21 schéma synoptique de la carte 94

Figure 3-22 : Restitution CMAC de la vitesse angulaire 1. 99

Figure 3-23 : Restitution CMAC de la vitesse angulaire 1(avec zoom). 100

Figure 3-24 : Restitution CMAC de la vitesse angulaire 2. 100

Figure 3-25 : Restitution CMAC de la vitesse angulaire 3. 101

Figure 3-26 : Courbe d'erreur pour les 3 vitesse angulaire CMAC . 101

Figure 3-27 : trajectoire triangulaire 103

Figure 3-28 : trajectoire triangulaire(zoom) 103

Figure 3-29 : trajectoire demi-cercle 104

Figure 3-30 : trajectoire demi-cercle avec zoom 104

Figure 3-31: courbe d'erreur pour trajectoire triangulaire 105

Figure 3-32: courbe d'erreur pour trajectoire Demi-Cercle 105

INTRODUCTION GENERALE

L'un des défis de l'homme aujourd'hui est de copier la nature et de reproduire des modes de raisonnement et de comportement qui lui sont propre . Cette envie a fait naitre le concepte d'intelligence artificielle. Celui-ci se traduit par l'émergences des techniques avancées telles que les réseaux de neurones artificiels. Ces approches dites « intelligentes » se sont déjà imposées dans un grand nombre de domaines tels que la reconnaissance de forme, le traitement du signal, la gestion financière, la robotique, le contrôle des processus, etc…

Depuis 1970 le monde scientifique et industriel porte des efforts sans cesse croissants sur l'étude et la réalisation d'engins mobiles. Ces derniers sont capables d'effectuer des déplacements sans liaison mécanique avec un point fixe de l'espace dans lequel ils évoluent, d'où le nom de robots mobiles.

Aujourd'huit, l'industrie mondial subit une véritable mutation qui se modifie profondément les données. La robotisation constitue une nouvelle avancée de l'automatisation grâce aux progrès de la micro-électronique permettant la mémorisation des informations de réglage et de contrôle pour un cout abordable et un volume réduit.

De ce fait , nous pouvons considérer que les robots mobiles sont synonymes de véhicules qui disposent de structures matérielle et mécanique qui leurs permettent la réalisation de taches complexes.

Dans ce contexte , les techniques basées sur l'emploi des réseaux de neurones artificiels suscitent au aujourd'hui un intérêt croissant dans les domaines de la commande robotique et plus particulièrement l'asservissement de robots mobiles. La rapidité de traitement, les capacités d'apprentissage et d'adaptation , mais aussi la robustesse de ces approches motivent en grande partie de nombreux travaux dans ce domaine.

Notre travail s'articule autour de deux objectifs majeurs. Le premier étant d'établir un état de l'art des différentes structures et architectures des neurones artificiels. Cette étude nous permettra de situer le réseau de neurones que nous implémenterons afin d'asservir note système robot mobile. Le deuxième but consiste à modifier la structure matérielle existante du robot pour qu'elle puisse accueillir des contrôleurs élaborés.

Ce document est organisé en trois chapitres qui portent sur les aspects suivants :

Le premier chapitre constitue une introduction aux réseaux de neurones artificiels. Il sera pour nous l'occasion d'introduire les concepts fondamentaux sur lesquels reposent ces techniques neuromimétiques. A travers cette étude, nous mettrons en exergue la nature, les principales architectures ainsi que les mécanismes d'apprentissage qui régissent leur fonctionnement. La connaissance et la compréhension des techniques d'apprentissage des réseaux de neurones constitueront pour nous une base pour étudier le réseau CMAC

(Cerebellar Model Articulation Controller) que nous avons choisi pour mettre en œuvre notre approche neuronale de la commande du robot mobile.

Le deuxième chapitre est consacré à l'étude que nous avons réalisée pour maitriser et implémenter le réseau de neurones CMAC. Nous présenterons de manière détaillée l'architecture et l'algorithme qui régit le fonctionnement de ce réseau. A travers cette étude, nous mettrons en évidence les propriétés attractives qui ont motive notre choix pour le réseau de neurones CMAC (rapidité, généralisation locale,….). par ailleurs , les résultats des simulation que nous réaliserons sur des fonction déterministes, nous aiderons à mettre en évidence ces performances.

Le troisième chapitre sera pour nous l'occasion d'exposer les résultats obtenus lors de l'implémentation du contrôleur CMAC sur la plate-forme robot mobile. Mais avant , nous présenterons une description de l'architecture mécanique et électronique du robot telle que nous l'avons trouvée. Au travers de cette analyse et des différents tests réalisés, nous mettrons en exergue les problèmes inhérents à cette architecture et solutions que nous avant mis en œuvre pour y remédier et accroitre de manière significative les performances du système. Des lors, nous réaliserons un ensemble d'expérimentations qui serviront à évaluer et à valider l'utilisation du réseau de neurones CMAC dans l'asservissement du robot mobile.

Enfin, la conclusion nous permettra de résumer l'ensemble des travaux réalisés.

CHAPITRE I :

Introduction aux réseaux de neurones

Introduction

Prédictions météorologiques, prévision de la consommation d'eau, reconnaissance du code postal, compression de données, filtrage, classification ... ont tous un point en commun : les réseaux de neurones.

Le but de ce dossier décomposé, en cinq grandes parties est de présenter de façon générale les réseaux de neurones.
La première partie comprend un historique succinct, balayant les soixante dernières années. Nous décrirons le neurone biologique qui a servi de modèle au neurone artificiel que nous nommerons neurone formel. Un neurone est soit excité soit inhibé, pour cela nous listerons et expliquerons les mécanismes lié au changement d'état.

Dans un deuxième temps, une introduction à la bio- ingénierie sera faite afin de donner une idée sur les évolutions futures. Nous clôturerons cette partie en comparant les réseaux de neurones à l'intelligence artificielle classique.

Qui dit réseaux sous-entend topologie. Dans la troisième partie nous expliciterons une classification des réseaux de neurones selon leur topologie. Nous nous baserons sur des schémas synoptiques pour illustrer l'architecture de chaque structure neuronale et pour mettre en exergue les points forts et les inconvénients de chacune d'elles.

Un réseau de neurones est comme le système nerveux d'un enfant. Il a besoin d'une phase ou période d'apprentissage. Nous verrons qu'il existe deux types principaux : guidé ou par tests.
Comme l'apprentissage de la lecture à l'école primaire nous avons besoin de choisir la meilleure méthodologie. Nous listerons ces dernières, que nous appellerons «règles d'apprentissage».
Nous finirons la quatrième partie, en dégageant une méthodologie résumée dans un tableau. Il sera ensuite plus aisé de choisir le bon type de réseau de neurones associé à la tache désirée. De même le type d'apprentissage et les règles nécessaires affineront notre choix.

La cinquième partie est consacrée à des exemples de mise en application Nous donnerons un exemple d'utilisation pour chaque type de réseau, démontrant ainsi l'utilité et l'importance de la diversité qu'elle soit topologique ou traitant de l'apprentissage.
Une liste moins détaillée donnera une idée sur la variété et l'intérêt des réseaux de neurones.

Le neurone seul

1 Historique

Les réseaux de neurones sont «nés »il y a plus de soixante ans. Les premiers à proposer un modèle sont deux bio-physiciens de Chicago, McCulloch et Pitts [1], qui inventèrent en 1943 le premier neurone formel. Qui est un dispositif à plusieurs entrées et une sortie qui modélise certaines propriétés du neurone biologique. Il peut être dans un état soit inhibé soit excité, et changer au cours du processus. Le neurone formel reste l'élément de base des réseaux de neurones actuels.

Un grand pas fut franchi en 1949 avec l'élaboration de la loi d'Hebb [5]. Il utilisa le conditionnement de type pavlovien afin d'écrire une règle d'apprentissage. Cette règle, appelée loi de Hebb, consiste à renforcer les connexions des neurones synchronisés.

Le modèle perceptron, fut inventé en 1957[6]. C'est un réseau de neurones inspiré du système visuel. Il possède deux couches de neurones : une couche de perception et une couche liée à la prise de décision. Mais ces limites furent mises à jour en 1969, en particulier le problème du XOR. Le perceptron étant une fonction linéaire il se trouve dans l'incapacité de résoudre des fonctions non linéaires, tel le XOR ou OU exclusif.
En 1983, la machine de Boltzmann arrivera à traiter de manière satisfaisante ces limites, grâce à l'ajout de couches cachées..

Nous pouvons mettre en relief quelques autres dates. Comme 1973 [8] où des études ont été conduites afin de faire de la reconnaissance de forme. Ou bien 1986, avec la règle de rétro-propagation du gradient, ou appelé aussi correction d'erreur.
Celui-ci est un algorithme où la différence entre la sortie désirée et la sortie propagée (le gradient) modifie le poids des neurones de sortie aux neurones d'entrée (retro-propagation).

Il est difficile de résumer en quelques lignes plus de 60 années de recherche sur les réseaux de neurones, dont les étapes décisives sont jalonnées par des publications clés.

2 Du neurone biologique au neurone artificiel

2.1 Le neurone biologique

Le neurone biologique est une cellule constituant l'élément fondamental du tissu nerveux. Son noyau est bloqué en «interphase », ce qui l'empêche de se diviser. Ce sont les neurones qui constituent l'unité fonctionnelle du système nerveux.

Le relais qui assure la transmission de l'influx nerveux est la synapse. Il existe deux sortes de synapse : les synapses électriques (minoritaires) et les synapses chimiques (majoritaires). La synapse est constituée d'un élément pré-synaptique, d'une fente synaptique et d'un élément post-synaptique.

Du point de vu fonctionnel, il faut considérer le neurone comme une entité polarisée, c'est-à-dire que l'information ne se transmet que dans un seul sens : des dendrites vers l'axone.

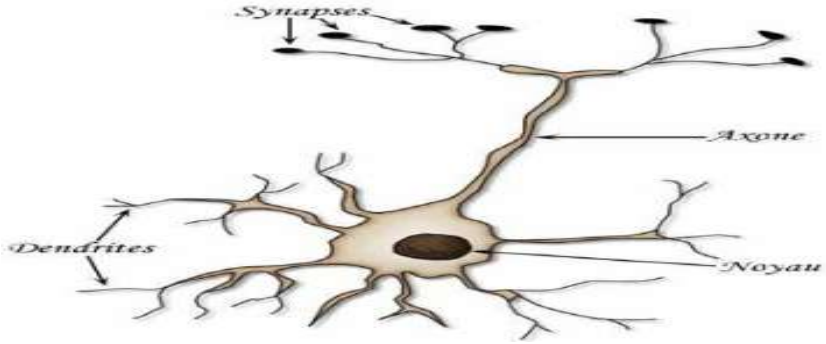


FIG. 1 – Neurone biologique

II.2 Le neurone artificiel

Le neurone artificiel est un processeur élémentaire, simulé sur ordinateur ou réalisé sur un circuit intégré. Il reçoit un nombre variable d'entrées en provenance de neurones appartenant à un niveau situé en amont (on parlera de neurones «amonts »). A chacune des entrées est associée un poids «**w** »représentatif de l'intensité de la connexion. Chaque processeur élémentaire est doté d'une sortie unique, qui se ramifie ensuite, pour alimenter un nombre variable de neurones appartenant à un niveau situé en aval (on parlera de neurones «avals»). A chaque connexion est associée un poids.

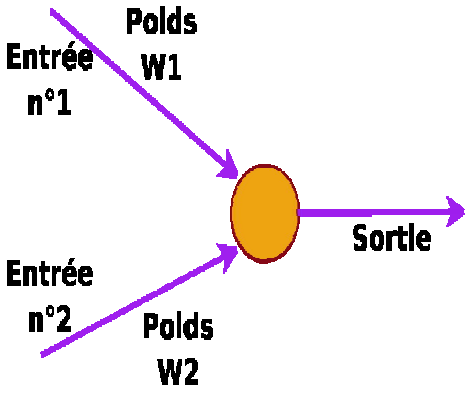


FIG. 2 – Neurone artifici

III- Neurone formel

III-1 Définition

Le neurone formel est une approximation très grossière de l'opération du neurone biologique Le neurone créé par McCulloch et Pitts [1] est un neurone utilisant une fonction binaire.

On peut le définir ainsi :

- Les noeud à plusieurs entrées.
- Les entrées proviennent d'autres neurones.
- Les entrées sont pondérées.
- Les poids sont soit positifs soit négatifs.
- Les entrées sont sommées au niveau du noeud pour produire une valeur d'activation.
- Le neurone s'active si l'activation est plus grande qu'un certain seuil.

III-2 Le neurone en équation

Le neurone réalise la fonction mathématique suivante :

$S = f_A(P)$

$P =$ Somme pondérée

$P = \sum_{i=1}^{n} X_i W_i$

$X_i = X_1 ... X_n =$ Entrées

$W_i = W_1 ... W_n =$ Poids synaptique

f_A est une fonction linéaire, bornée

On peut la réecrire $S = g(X_i)$

ou $g(x_i) = f_A(\sum_{i=1}^{n} X_i W_i)$

Le processus est paramétré par les W_i

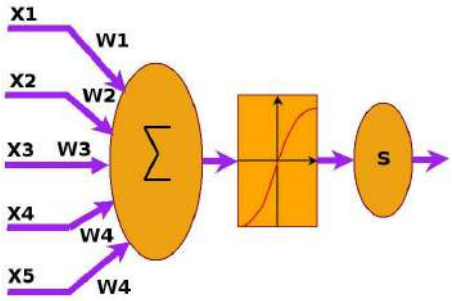


FIG. 3 – Neurone formel de Mac Culloch et Pitts

III-3 Les différents types de fonction d'activation

Nous allons définir le neurone à seuil et son utilité. Nous verrons ensuite les différents types de fonctions en décrivant leurs caractéristiques. $f(x)$ est la fonction d'activation.

Neurone à seuil

Dans un neurone formel il faut distinguer deux parties.

D'un coté la somme pondérée des entrées ($f_A(P)$), d'un autre la fonction d'activation ($f(x)$) qui permettra de définir la valeur de sortie.

Il existe plusieurs types de fonctions d'activations, dont les sorties sont soit linéaires soit non linéaires.

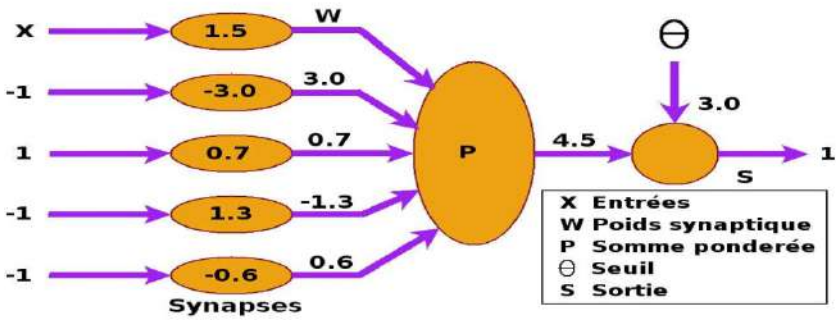


FIG. 4 – Neurone à seuil

Le poids d'entré P= 4,5 or le seuil θ = 3 donc P > 0 et donc S = 1

Fonction binaire

Les premiers modèles de neurones étaient caractérisés par une fonction d'activation à seuil simple (binaire : 0=inactif, 1=ac- tif). Le déclenchement de l'activité intervient si la somme des excitations dépasse un certain seuil propre au neurone.

Si $P > 0$ ==> Excitatrice $S = 1$

Si $P < 0$ ==> Inhibitrice $S = -1$

Si $P = 0$ ==> Non Connectée

La fonction binaire confère une certaine efficacité aux réseaux et correspond à un modèle proche de la réalité.

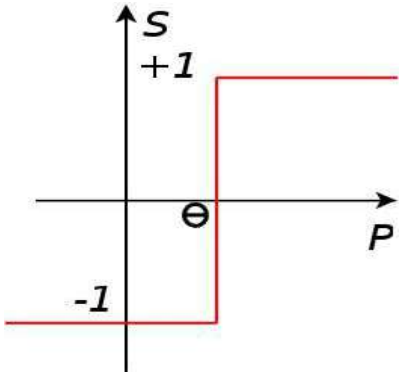


FIG. 5 – Fonction binaire

Fonction linéaire par morceaux

$f(x) = -1$ *ou* $+1$,au-delà des bornes.

$f(x) = A.x$, sinon.

Ce modèle est plus réaliste en faisant apparaître des phénomènes de saturation.

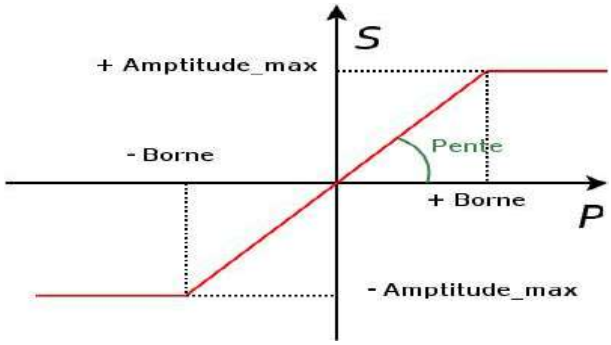


FIG. 6 – Fonction linéaire par morceaux

Fonction linéaire

$f(x) = A.x$

Lorsque la fonction d'activation est dérivable, il est possible d'utiliser des algorithmes d'apprentissage performants.

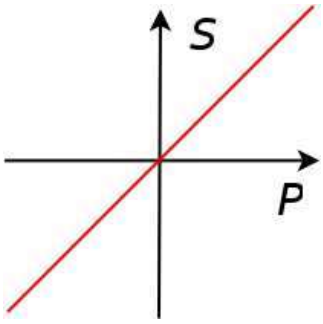


FIG. 7 – Fonction linéaire

Fonction sigmoïde

Ce type de fonction combine l'avantage de l'effet de seuil et de la dérivabilité. Les fonctions sigmoïdes sont donc très utilisées.

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

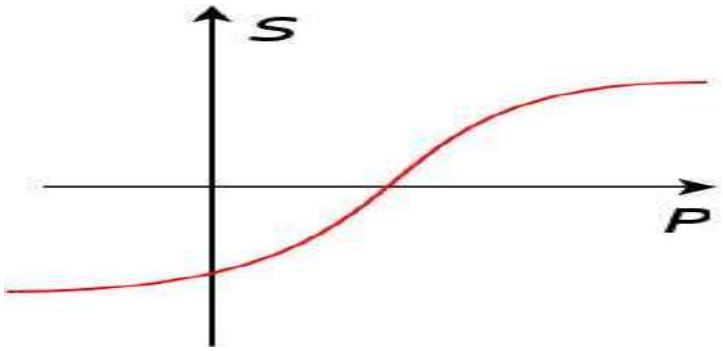


FIG. 8 – Fonction sig moïde

Fonction gaussienne

Fonction non monotone : effet inhibiteur d'une trop grande sollicitation Utilisée dans les réseaux de type «RBF»(Radial Basis Functions)

$$R(x) = exp(\frac{x^2}{2*SIGMA^2})$$

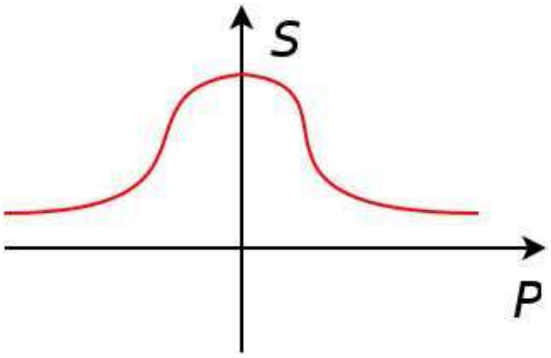


FIG. 9 – Fonction gaussienne

Réseau de neurones

I-Réseau de neurones

Les réseaux de neurones formels ou réseaux neuromimétiques sont des programmes informatiques qui en simulant le fonctionnement des neurones du cerveau humain, permettent aux calculateurs d'apprendre à effectuer certaines tâches simples.
Ils ne fonctionnent pas de façon programmée mais de façon adaptative. Cela leur permet de fonctionner même avec des informations partielles ou brouillées.
Les réseaux neuronaux sont composés souvent de plusieurs couches de cellules reliées entre elles et formant une vaste toile (réseaux multicouches). Ils sont capable d'ajuster l 'intensité ou la force de leurs propres interconnexions, exactement comme cela se passe pour celles du cerveau humain, lors de la phase d'apprentissage. Les réseaux de neurones ont par exemple la faculté de reconnaître les visages, les voix, les règles du jeu de dames ou d'échecs, et même de stabiliser le vol d'un hélicoptère.

II-Bio-ingénierie

La bio-ingénierie est l'application des connaissances de l'in- génierie dans les sciences biologiques.
[D.05] Dans le cadre des réseaux de neurones on peu distinguer deux domaines.
[dmE05] Dans le premier cas on étudie la possibilité de construire des réseaux de neurones hybrides vivant-artificiel, où s'instaure un dialogue entre un système de neurones électroniques et des neurones biologiques. Ces expériences permettent de mener des tra- vaux de neurobiologie fondamentale et comprendre le fonctionne- ment de certains réseaux neuronaux. Les premières manipulations ont permis notamment d'analyser le filtrage du signal visuel vers le cortex, selon les niveaux d'éveil ou de sommeil.
Actuellement les recherches se dirigent vers le domaine du traite- ment de la douleur.

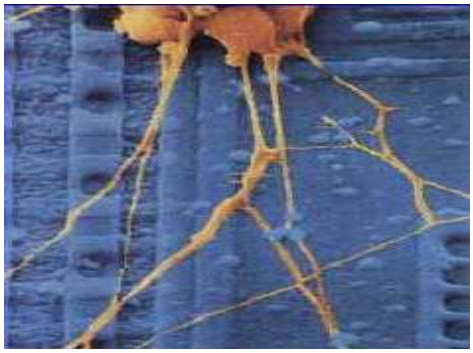


FIG. 10 – Neurone greffé à une puce

Dans un second cas on étudie des applications basées sur des cultures de réseaux de neurones, maintenues en vie pendant plusieurs semaines. Grâce à un réseau de neurones artificiels analogiques, les chercheurs peuvent connecter ces neurones vivants avec un robot muni de capteurs. Le robot devra se situer et se déplacer dans un milieu changeant. Le but du programme est de comprendre le processus d'apprentissage dans le cerveau.

III-Comparaisons avec l'intelligence artificielle classique

III.1 L'intélligence artificielle classique

L'intelligence artificielle (**IA**) classique ou symbo- lique qui s'ancre dans les principes de base de la psycho- logie d'information. En intelligence artificielle classique, la plupart des modélisations caractérisent un comporte- ment intelligent par des opérations formelles effectuées sur des structures symboliques D'où le slogan

cognition = computation + représentation

L'agent intelligent de l'IA classique est essentiellement un résolveur de problèmes tel qu'il est représenté dans le schéma I-R-R



FIG. 11 – schéma I-R-R

III.2 Bref histoire de l'intelligence artificielle classique

L'IA est utilisée dès ses débuts au développement de programmes capables de jouer aux échecs. Les premiers programmes apparaissent au début des années 60, et au fil des décennies, arrivent à battre des joueurs de niveaux de plus en plus élevés.

Les années 70, furent aussi celles des premières expérimentations avec des robots mobiles (comme par exemple le robot Shakey du SRI à Menlo Park en Californie).

Prolog, un des principaux langages de programmation logique créé en 1972 sera reconnu comme langage de développement en intelligence artificielle.

Le système ELIZA en repérant des expressions clés dans des phrases et en reconstruisant à partir d'elles des phrases toutes faites. Il était capable dès 1965, de dialoguer en langage naturel, pouvant tromper des interlocuteurs humains pendant un moment.

Aujourd'hui ALICE [2] représente son meilleur successeur. Il est possible de converser avec elle, sur des salons de discution (irc ou internet relay chat), ou sur des sites de démonstration. [2]

III.3 Différences

Les réseaux de neurones son essentiellement numériques, par opposition à l'IA symbolique qui est basée sur des règles, raisonnement par cas, etc...

On peut partir d'algorithmes médiocres et les laisser améliorer leurs capacités grâce à l'expérience. Plutôt que d'indiquer pas à pas comment parvenir au résultat recherché, on fournit simplement des données à ces réseaux qui, suivant un processus d'apprentissage, comparent leur résultat à celui recherché pour tenter ensuite de s'en rapprocher au mieux.

La solution actuelle est de developper des architectures mixtes intégrées à la manière du cerveau gauche et du cerveau droit. Les réseaux neuronaux traitent les aspects globaux de la perception et de catégorisation, pendant que l'I.A. se sert du résultats de ce processus et le traite de manière logique.

> Gérard Dreyfus, dirigeant le laboratoire d'électronique de l'Ecole supérieure de physique et de chimie industrielles de la Ville de Paris (ESPCI) et auteur d'un livre sur le sujet [3] a écrit dans une interview [4] :
>
> «*On pourrait dire que l'Intelligence Artificielle est la branche de l'informatique qui traite des données symboliques, et qui permet notamment de formaliser les raisonnements. Les produits les plus connus de l'Intelligence Artificielle sont les systèmes experts, qui permettent d'implanter dans des ordinateurs les connaissances d'experts sous forme de règles du genre «Si la température dépasse 250, il faut fermer telle vanne». En ce sens, l'apprentissage numérique, donc les réseaux de neurones, ne fait pas partie de l'intelligence artificielle.*»

Classification topologique

I- Les réseaux non bouclés (feed-forward)

Appelés aussi «réseaux de type Perceptron», ce sont des réseaux dans lesquels l'information se propage de couche en couche sans retour en arrière possible.

I-1 Les Perceptrons

Le perceptron monocouche

le perceptron monocouche qui contient une couche d'entrée et une couche de sortie (c'est la seule couche active d'où le terme de monocouche).
Il est calqué, à la base, sur le système visuel.

Utilisation : reconnaissance des formes, classification, opérations logiques simples («ET»ou «OU»)...

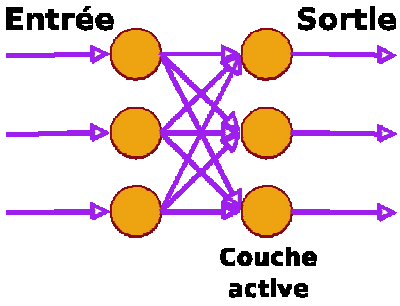


FIG. 12 – Perceptron mono-couche

Le perceptron multicouches (PCM)

Le perceptron multicouche, quant à lui, comprend en plus des deux couches de base une ou plusieurs couches intermédiaires appelées couches cachées.

Il peut résoudre des problèmes non-linéairement sé- parables et des problèmes logiques plus compliqués, et notamment le fameux problème du XOR.

Utilisation : Reconnaissance de chiffres.

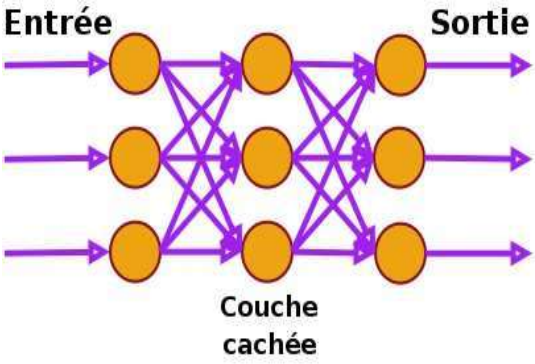


FIG. 13 – Perceptron multicouche

I-2 Les réseaux à fonction radiale (RBF)

Les réseaux à fonction radiale (RBF) qui possèdent deux couches forment une classe particulière de réseaux multi-couches. Chaque cellule de la couche cachée utilise une fonction d'activation Gaus- sienne.

Utilisation : Classification, approximation de fonctions...

II- Les réseaux Récurrents (feed-back)

Appelés aussi «réseaux récurrents», ce sont des réseaux dans lesquels il y a retour de l'information. Ils sont caractérisés par :

- La présence d'au moins une boucle de rétroaction, au niveau des neurones et/ou entre les couches.

L'aspect temporel du phénomène (la mémoire).

II-1 Les cartes auto-organisatrices de Kohonen

La carte auto-organisée de Kohonen est en général à deux dimensions. Chaque neurone de la couche d'entrée est relié à chaque neurone de la carte de Kohonen.
Chaque neurone de la carte de Kohonen est relié à tous les neurones de la carte.
Il est utilisé au sein d'une méthode d'analyse de données et de compréhension

Utilisation : Aide à la décision, Data mining ...

Il existe un cas particulier des SOM, les LVQ (pour «Learning Vector Quantization»), où seul le noeud ayant la meilleure corrélation est adapté. On parle, pour ce type de réseau où seul le «vainqueur» est sélectionné, de réseau à compétition.Utilisation : Catégorisation, compression de données

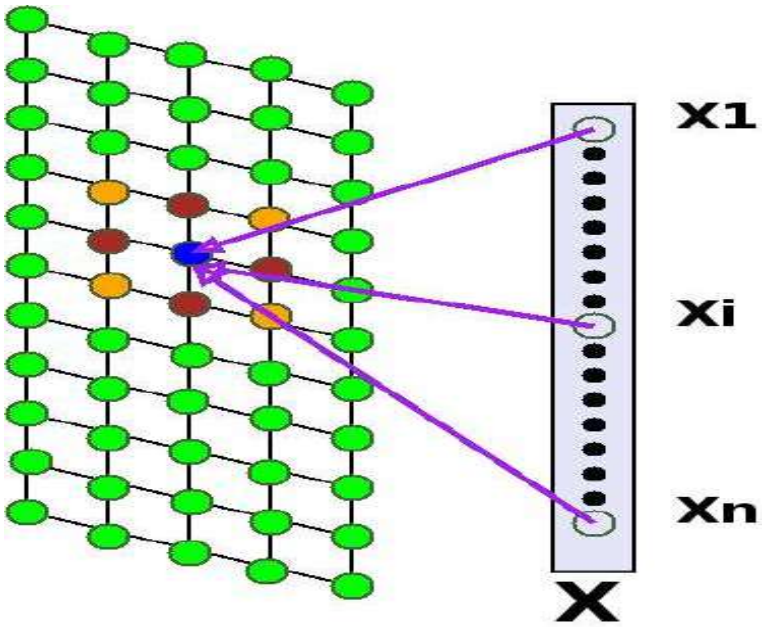


FIG. 14 – Modèle de Kohonen (SOM)

II-2 Les réseaux de Hopfield (RNF)

Les réseaux de Hopfield sont des réseaux récurrents et entièrement connectés. Dans ce type de réseau, chaque neurone est connecté à chaque autre neurone et il n'y a aucune différenciation entre les neurones d'entrée et de sortie.

Ils fonctionnent comme une mémoire associative non-linéaire et sont capables de trouver un objet stocké en fonction de représentations partielles ou bruitées. L'application principale des réseaux de Hop- field est l'entrepôt de connaissances mais aussi la résolution de problèmes d'optimisation.

Utilisation : La reconnaissance de forme, l'entrepôt de connaissances, la résolution de problèmes d'optimisation...

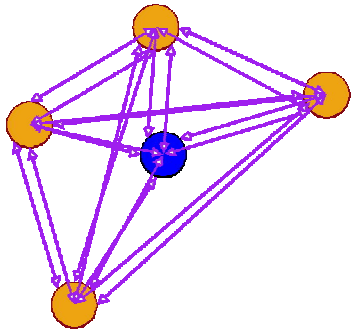


FIG. 15 – Réseau de Hop- field

II-3 Les ART (Adaptive Resonance Theory)

Les réseaux ART («Adaptative Resonnance Theorie») sont des réseaux à apprentissage par compétition.

Il existe deux principaux types de réseaux ART : les ART-1 pour des entrées binaires et les ART-2 pour des entrées continues.

Utilisation : Catégorisation

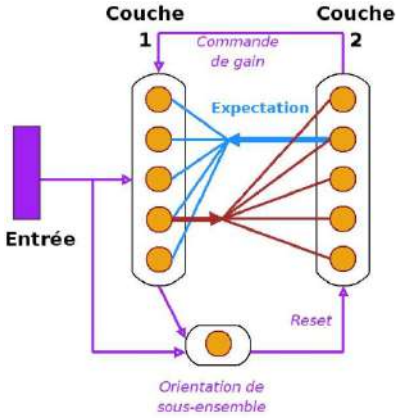


FIG. 16 – Réseau ART

II-4 Les réseaux à mémoire associative

Dans les réseaux à mémoire associative "*Associative Memory Networks*", le terme "mémoire" fait référence à la fonction de stockage de l'information et le terme "associative" au mode d'adressage. L'expression "mémoire adressable par son contenu" est aussi souvent employée. L'information mémorisée ne peut être obtenue avec une adresse précise. Le seul moyen d'accès est de fournir une information.

La structure des réseaux à mémoire associative présente de nombreuses similitudes avec celle du Perceptron multicouche. La figure 1-17 représente l'architecture d'un réseau à mémoire associative. Celui-ci est constitué de trois couches et d'un jeu de paramètres qui peuvent être ajustés itérativement, dans le but d'atteindre un comportement souhaité.

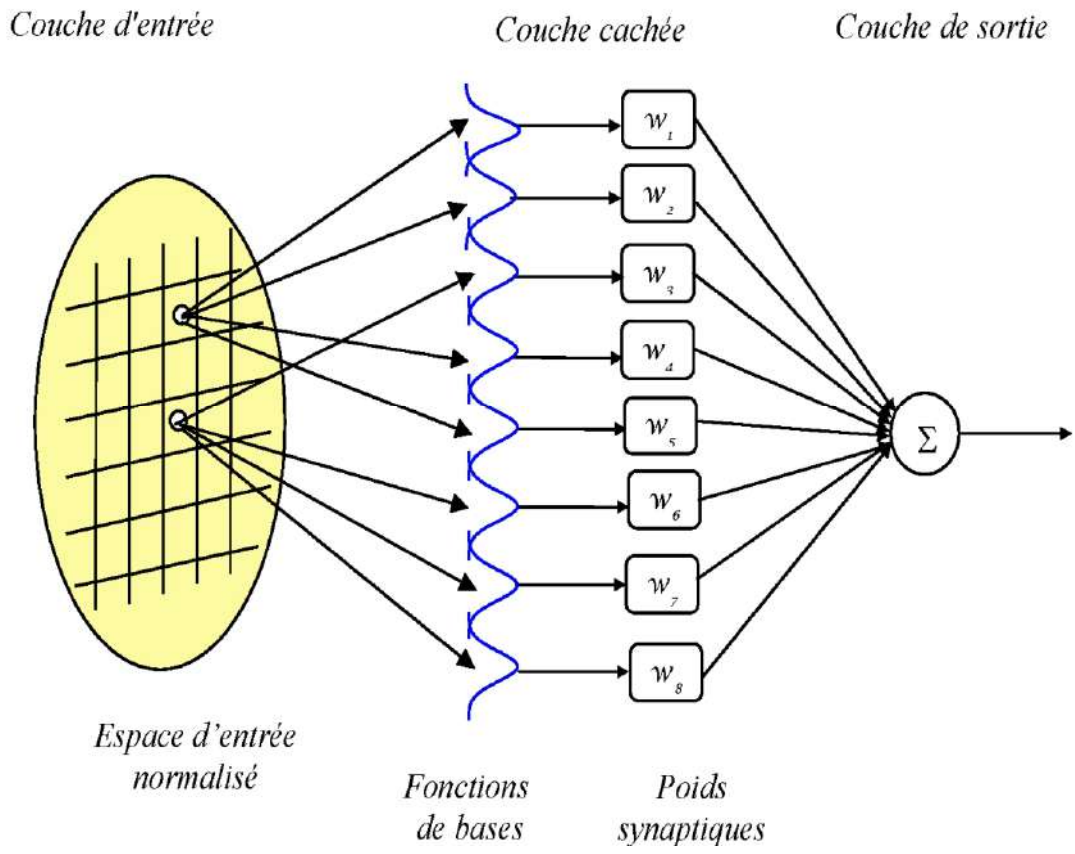


Figure 1-17 : Architecture d'un réseau à mémoire associative.

Cependant deux différences distinguent les mémoire associatives des réseau de neurone unidirectionnelle multicouches.la premier repose sur le remplacement de la premier couche par une couche qui normalisé l'espace d'entré, cette fonction de normalisation peut prendre deverses forme :

- Elle peut être utilisé pour transformé les entres réale en un vecteur dans les composant sont bennières ce qui a pour conséquences d'émuler les cellule sensorielles du perceptron multicouche
- Elle peut définie un grille à N dimensions qui sera superposé à l'espace d'entré
- Ou simplement faire transiter l'entré ver la couche suivant e sans aucune modification.

La seconde déférence réside dans la fait d'utilisé des fonction de base « basis function » à la place des cellules associatives du perceptron, leur taille, leur forme et leur nombre déterminent le degré de complexité du réseau ,ainsi que sa capacité sa généralisé l'information acquise au cours de la phase d'apprentissage ,la sortie de ces fonction de base est comprise en intervalle[0,1].

Quand cette sortie est égale à zéro, le poids synaptique associe a cette fonction de base ne sera pas mis à contribution lors du calcul de la réponse du réseau.

En résumé ,cette classe de réseau de emmagasiné l'information de façon locale en créant une représentation interne d'entrée. C'est cette caractéristique qui la distingue des autres classes de réseaux de neurones décrites jusqu'ici . En fait, la représentation interne éparse procure aux réseaux à mémoire associative une capacité d'adaptation quasi-instantanée, ainsi qu'une souplesse plus accrue que les autres types de réseaux de neurones. Ces deux points forts s'avèrent très importants lorsque ce réseau est utilisé dans la commande robotique.

2.4 Les réseaux à compétition

Un réseau à compétition est un réseau monocouche, la couche de sortie étant entièrement connectée, c'est-à-dire qu'une cellule de cette couche est connectée à toutes les autres, un lien excitateur revient sur lui-même, tandis que des liens inhibiteurs la connectent à ses voisins. Lorsque l'on présente un patron en entrée, les cellules de la couche de sortie reçoivent une activation. à ce moment, les cellules de la couche de sortie entrent en compétition pour déterminer quelle sera la cellule qui sera finalement activée.

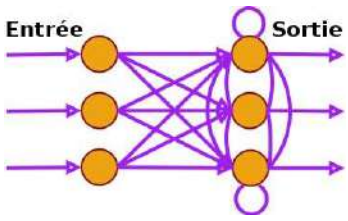
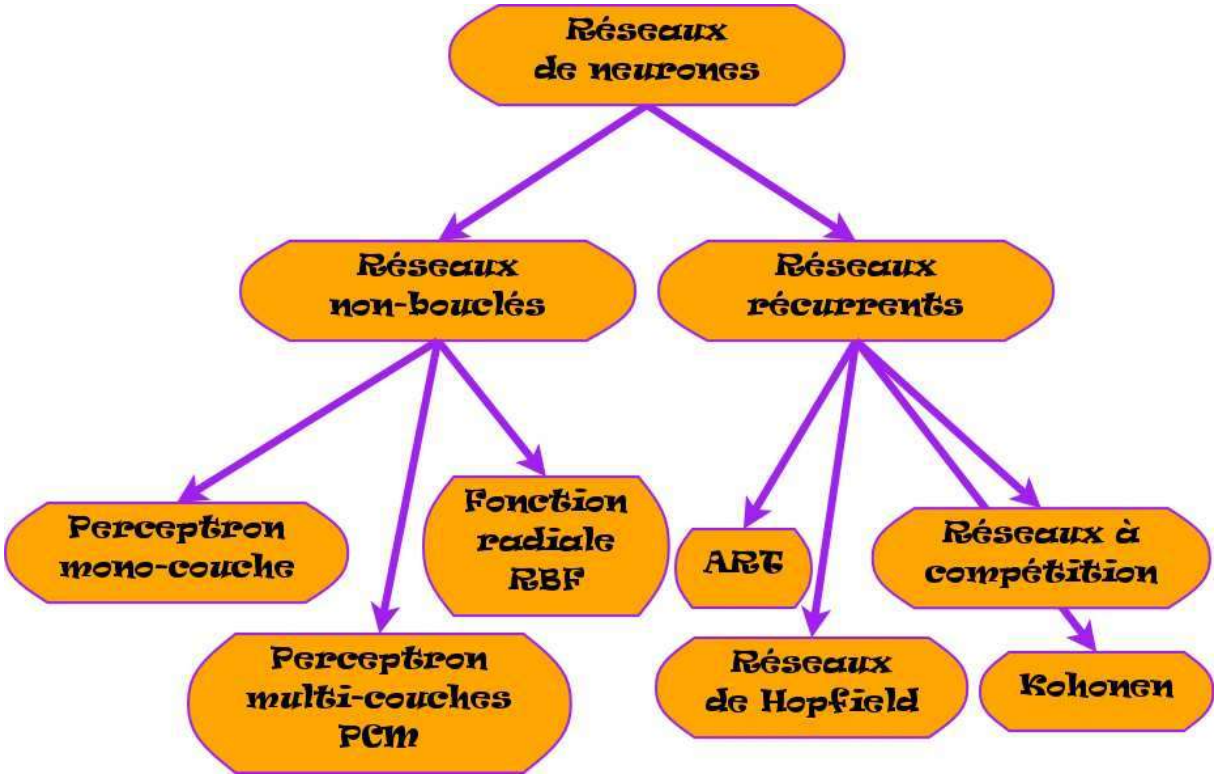


FIG. 17 – Les réseaux à compétition

Tu n'as pas décrit les réseaux de type mémoire associative pour classer ton CMAC

3. Récapitulatif



Apprentissage

L'apprentissage c'est d'abord calculer les valeurs des coefients synaptique fonction des exemples disponibles.

C'est un processus d'acquisition de connaissance de l'environnement. Il va de pair avec la mémoire qui est un mécanisme de rétention de la connaissance.

Dans les réseaux neuroniques seul le résultat compte, le système va distinguer les spécificités de telle ou telle solution pour ne plus les confondre.

La seule contrainte est de montrer inlassablement le résultat attendu au système.

Lorsque le système donne lui-même le résultat attendu dans une grand majorité des cas, l'apprentissage est terminé.

IV. Apprentissage et adaptation

IV-1 Paradigmes d'apprentissage

La notion d'apprentissage, inhérente aux réseaux de neurones artificiels, est induite par une procédure itérative d'ajustement des paramètres internes (les connexions). Cette procédure désigne, en clair, un processus d'adaptation des poids synaptique du réseau avec comme finalité l'obtention d'un état stable. Cet état stable corresponde alors à un modèle de connaissance dans un environnement dont une base d'exemples fournit une description.

D'un point de vue plus forme, l'apprentissage d'un réseau consiste à modifie ses poids de manière à optimiser un critère de performance (par exemple l'erreur quadratique sur la base d'apprentissage) il peut être veut comme la recherche d'une solution optimale dans un espace de dimension N ,cette recherche ne peut obtenir à une solution satisfaisante que si la complexité(le nombre des paramètres),est suffisant pour apprendre une tâche donné. Si cette complexité est trop faible, le réseau n'a pas assez de cette flexibilité pour modéliser correctement les donné de la base d'exemple. Un réseau trop complexe a tendance à entrainer un phénomène de sur-apprentissage. Dons les deux cas, les performance de généralisation du Réseau sont amoindrie Idéalement le réseau doit acquérir d'avantage de connaissances à chaque itération à laide des trois étapes suivantes du mécanisme d'apprentissage :

1) le réseau est stimulé par l'environnement ;
2) en réponse à cette stimulation, le réseau adapte son comportement ;
3) le réseau réagit alors différemment à l'environnement en fonction de la nouvelle expérience acquise consécutivement à la stimulation.

Concrètement, les règles d'apprentissage sont destinées à chercher une solution au problème de l'apprentissage que constitue l'assimilation de la connaissance, à partir d'une base d'exemple. Ces règles sont liées à l'architecture du réseau, mais sont plus ou moins efficaces selon l'application qui est

considérée. « error-correction learning », l'apprentissage basé sur la mémorisation « memory-based learning ».

L'apprentissage de Hebb « Hebbian learning » , l'apprentissage compétitif « competitive learning » et l'apprentissage de Boltzmann « Boltzmann learning ». pour plus de détail sur ces méthode, nous invitons le lecteur à se référer à ouvrages (cichocki and unbehauen1993) et (Haykin 1994)

En se basant sur l'information relative à l'environnement disponible, noues pouvons définir trois grande classe d'algorithmes d'apprentissage : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage par renforcement.

a) l'apprentissage supervisé

L'apprentissage supervisé « supervised learning » doit déposé un comportement de référence précis par pouvoir l'inculquer au réseau neuronal .le réseau doit mesurer la différence entre sont propre comportement et le comportement de référence qui ce lui d'un expert. Le réseau doit alors corriger sons poids de façon de réduire cette erreur.

D'un point de vue conceptuel, cet expert ou éducateur possède la connaissance de l'environnement connaissance qui se représente par des couples d'entrées/sorties .cet environnement est supposé inconnu de réseau. Le rol de l'expert, dans ce cas-ci, est de fournir les informations relative à ces environnement.ces informations nécessaires à l'apprentissage du réseau, se présentent sous forme d'un exemple évoluent pour ajuster en fonction des influence combinées du vecteur d'apprentissage et du signal d'erreur. Ce signal étant défini comme la déférence entre la réponse courante du réseau et de la réponse désirée fournir par l'expert.

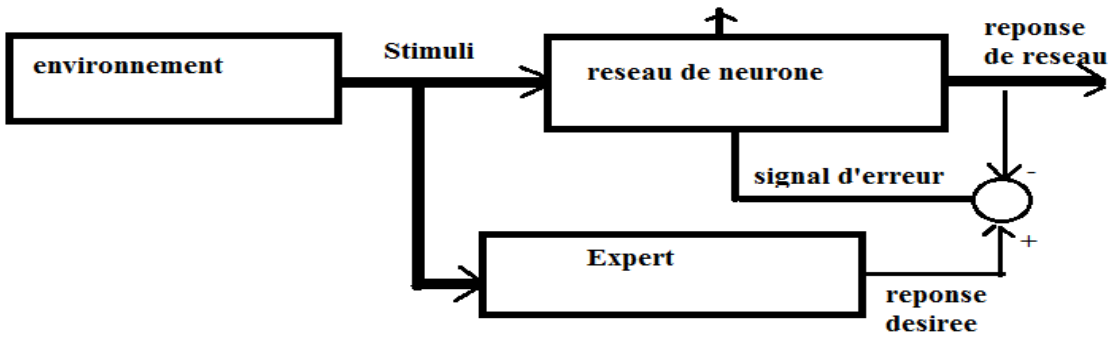


Figure 1.20 : **l'apprentissage supervisé**

Le réseau doit à terme, après un certain nombre de pas d'apprentissage, émuler la connaissance de l'expert de façon optimale (au sens statique). Lorsque ce transfert de connaissance de l'expert est achevé, on se passe alors de lui, et c'est le réseau qui « travaille » directement avec son environnement.

L'apprentissage supervisé que nous venons de décrire est fondé sur un processus de correction de l'erreur d'apprentissage. Il réalise un système bouclé dont l'environnement est toutefois exclu. La mesure de la performance du système est évaluée d'après l'erreur définie au sens des moindres carrés

(la moyenne de la somme des erreurs au carré). Cette fonction d'erreur se visualise sous la forme d'une surface d'erreur dont les coordonnées sont les paramètres internes du réseau de neurones. Toute opération du système sous la supervision de l'expert, peut etre représentée comme un point de la surface d'erreur. Pour que le réseau améliore ses performances au fil du temps en apprenant la connaissance de l'expert, ce point de la surface d'erreur doit descendre vers un point minimum.

Dans l'apprentissage supervisé, la présence de l'expert est nécessaire. Ce qui peut etre une contrainte dans la mesure où, selon l'application, aucun comportement de référence n'est disponible. Un apprentissage qui ne nécessite pas de comportement de référence explicite mais seulement d'information grossières, tel qu'un encouragement ou une pénalisation, est appelé apprentissage par renforcement.

b) L'apprentissage par renforcement

Comme nous venons de le voir, l'apprentissage supervisé nécessite la définition d'une base d'exemples d'apprentissage représentative. Cependant, la difficulté inhérente à la construction d'une tellebaselimitesonutilisation.Desrecherchesontdoncétéconduitesdanscesens.Leurobjectif est de construire, de manière automatique, cette base d'apprentissage .L'apprentissage par renforcement est donc une alternative possible à la définition par l'opérateur de la base d'apprentissage.Ladifférenceessentielleavecl'apprentissagesupervisé réside dans la forme des exemples d'apprentissage. Il s'agit de triplets (entrée, sortie ,utilité),où la dernière composante encode l'utilité de produire telle "sortie" pour telle "entrée" (figure 1-21). Les exemples d'apprentissage sontic générés automatiquement au cours d'une phase dite "d'exploration". Ils'agit le plus souvent d'une exploration aléatoire de l'espace d'entrée .Bien entendu ,la grande taille de l'espace de recherche ne permet pas une couverture complète(exhaustive) etil faut donc réunir, durant cette étape ,une base d'exemples représentative de la fonction de mise en correspondance souhaitée pour le réseau de neurones. Du fait du caractère automatique de la construction de la base d'apprentissage, l'utilité d'une paire (entrée, sortie) doit être décrite par une équation, une règle de calculouneprocédure.C'estàceniveauquesesituel'interventionprincipale de l'opérateur .Le succès de l'application dépendra de la qualité de la fonction spécifiant l'utilité d'une paire .Cette fonction d'utilité ,habituellement appelée fonction de renforcement, mesure la performance du réseau.

La performance est définie comme l'utilité des sorties proposées relativement à la tâche à accomplir .L'utilitées ticiun conceptqualitatif, généralement codé de la façon suivante:+1 équivaut à bon,-1àmauvaiset0 lorsque l'onestin capable des eprononcer . Généralement, la génération de la base d'apprentissages fait en parallèle de la phase d'exploitation ,l'apprentissage estincrémentaletc'estpourquoi,lorsqu'unebased'apprentissagereprésentativeafinalementété construite, l'apprentissage est terminé.

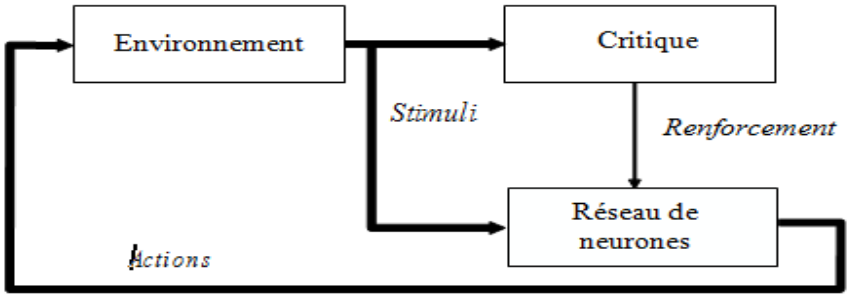


Figure. 1-21: l'apprentissage par renforcement.

Echangerlaconstruction"manuelle"delabased'apprentissagecontreunesimplemesure delaperformanceestcertainementéconomiquepourl'opérateur,maisleprix à payer est une convergence de l'apprentissage plus lente. En effet, l'utilité d'une sortie particulière pour une entrée donnéeestuneinformationmoinsrichequecelledesavoirexactementquelleest la sortie de plus grande utilité (c'est-à-dire la sortie désirée).L'objectif de l'apprentissage par renforcement est donc de trouver la sortie de plus grandeutilitéenutilisantpourcelalesutilités"binaires"desexemples disponibles. En complément à l'apprentissage par renforcement, on trouve toujours un apprentissage spécifique du réseau de neurones employé, dont la tâche est de réaliser la modification des poids du réseau .Il est important de noter qu'en fait l'apprentissage supervisée l'apprentissage par renforcement ne se concurrencent pas .Chacun correspond à un champ d'applications particulier, selon que l'ondispose d'une base d'exemples représentative ou d'une mesure de la performance du comportement recherché.

c) L'apprentissage non supervisé

Contrairement aux apprentissages décrits précédemment qui s'effectuent sous contrôle d'un expert, l'apprentissage non supervisé "*unsupervisedlearning*" est autodidacte. L'ensemble des exemples d'apprentissage ne comprend que des stimuli. Aucune réponse désirée n'est prise en considération.

De par sa nature , cet apprentissage est destiné à l'élaboration d'une représentation interne de la connaissance de l'environnement, identifiant la structure statistique sous-jacente des stimuli sous une forme plus simple ou plus explicite. L'algorithme d'apprentissage exploite pour se faire une mesure prédéterminée de la qualité de représentation de la connaissance afin d'ajuster les paramètres du réseau. Ce type d'apprentissage est représenté schématiquement par la figure suivante :

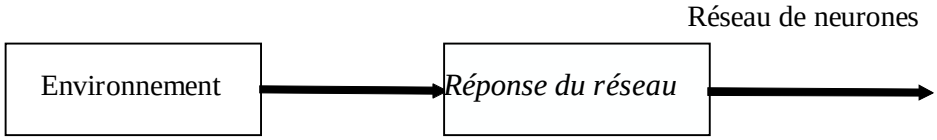


Figure. 1-22: L'apprentissage non supervisé.

L'apprentissage non supervisé s'applique aux réseaux à base de prototypes.Dans ce cas,le réseau fonctionne comme un système de catégorisation "*clustering*" dans lequel les différents prototypes s'adaptent au furet à mesure que sont présentées les données. Cette adaptation des prototypes peut être réalisée de trois façons différentes: par assimilation, par accommodation ou par différenciation.

Dans tous les cas,le but est d'effectuer le meilleur pavage possible du domaine d'entrée à l'aide des prototypes.On parle d'apprentissage non supervisé par ce que la réponse attendue n'est pas donnée au réseau.Le réseau apprend d'une façon qui est prédéterminée par les lois d'évolution fixées par le concepteur, en particulier à travers le choix de la distance d'un exemple avec les différents prototypes.

VI.3Règles d'apprentissage

a) Règle de Hebb

La plus ancienne règle d'apprentissage repose sur le postulat de Hebb [5] établi à partir d'observations d'expériences de neurobiologie : si des neurones, de part et d'autre d'une synapse, sont activés de manière synchrone et répétée, la force de la connexion synaptique va aller croissant. Cette règle peut s'exprimer de la façeon suivante :

$$w_{ij}(t+1) = w_{ij}(t) + \eta y_j(t) x_i(t)$$

où :

$xi(t)$ et $yj(t)$ sont les sorties, au temps t des neurones
i et j dont le poids de connexion (entre i et j) vaut $w_{ij}(t)$
η est le coefficient d'apprentissage, où x_i est l'entrée de la synapse.

L'une des propriétés remarquables de cette règle est qu'elle exprime que l'apprentissage se fait localement c'est-à-dire que la modification de w_{ij} ne dépend que de l'activité des cellules *i* et *j*. Nous verrons l'application de l'algorithme d'apprentissage de Hebb dans l'annexe B .

b) **Règle de correction d'erreurs**

Cette règle s'inscrit généralement dans le paradigme d'apprentissage supervisé, c'est-à-dire dans le cas où l'on fournit au réseau une entrée et la sortie correspondante.
Si on considère «y »comme étant la sortie calculée par le réseau, et «d »la sortie désirée, le principe de cette règle est d'utiliser l'erreur («d - y »), afin de modifier les connexions et de diminuer ainsi l'erreur globale du système.
Le réseau va donc s'adapter jusqu'à ce que «y »soit égal à «d ».
Notez que l'apprentissage n'intervient seulement que lorsqu'une erreur est faite, dans le cas contraire, les poids demeurent inchangés.
Cette règle est donc une forme modifiée de la règle d'apprentissage de Hebb.

a) **Apprentissage de Boltzmann**

Les réseaux de Boltzmann sont des réseaux symétriques récurrents. Ils possèdent deux sous-groupes de cellules, le premier étant relié à l'environnement (cellules dites visibles) et le second ne l'étant pas (cellules dites cachées).
Les machines de Boltzmann opèrent en deux modes distincts :
le mode figé («clamped»en anglais), dans ce cas les cellules visibles sont affectées à une valeur déterminée par l'environnement ;
le mode libre évolution («free-running») dans lequel l'ensemble des cellules, qu'elles soient visibles ou cachées, peuvent changer d'état librement.
La règle d'apprentissage est de type stochastique, elle est dérivée de la théorie de l'information et des principes de la thermodynamique. L'objectif de cet apprentissage est d'ajuster les poids des connexions, de sorte que l'état des cellules visibles satisfasse une distribution probabiliste souhaitée. En accord avec la règle d'apprentissage de Boltzmann, les modifications se font par :

$$\Delta w_{ij} = \eta(\overline{\rho_{ij}} - \rho_{ij})$$

où :
η est le coefficient d'apprentissage et
$\overline{\rho}_{ij}$ (resp. ρ_{ij}) sont les corrélations entre les états *i* et *j* lorsque le système est en mode figé (resp. libre-évolution).
Les valeurs $\overline{\rho}_{ij}$ et ρ_{ij} sont classiquement estimées à l'aide de la méthode de Monte-Carlo qui est

b)Règle d'apprentissage par compétitions

A la différence de la règle de Hebb (dans laquelle plusieurs neurones peuvent être activés en sortie), cet apprentissage n'active qu'un seul neurone.

Le principe de cet apprentissage est de regrouper les données en catégories. Les patrons similaires vont donc être rangés dans une même classe, en se basant sur les corrélations des données, et seront représentés par un seul neurone, on parle de «winner-take-all ».

Dans un réseau à compétition simple, chaque neurone de sortie est connecté aux neurones de la couche d'entrée, aux autres cellules de la couche de sortie (connexions inhibitrices) et à elle-même (connexion excitatrice).

La sortie va donc dépendre de la compétition entre les connexions inhibitrices et excitatrices.

Le résultat de la compétition est de choisir la cellule ayant la plus grande (ou la plus petite) entrée. A noter que seules les connexions du vainqueur sont mises à jou

c)Apprentissage de Boltzmann

Les réseaux de Boltzmann sont des réseaux symétriques récurrents. Ils possèdent deux sous-groupes de cellules, le premier étant relié à l'environnement (cellules dites visibles) et le second ne l'étant pas (cellules dites cachées).

Les machines de Boltzmann opèrent en deux modes distincts :

le mode figé («clamped»en anglais), dans ce cas les cellules visibles sont affectées à une valeur déterminée par l'environnement ;

le mode libre évolution («free-running») dans lequel l'ensemble des cellules, qu'elles soient visibles ou cachées, peuvent changer d'état librement.

La règle d'apprentissage est de type stochastique, elle est dérivée de la théorie de l'information et des principes de la thermodynamique. L'objectif de cet apprentissage est d'ajuster les poids des connexions, de sorte que l'état des cellules visibles satisfasse une distribution probabiliste souhaitée. En accord avec la règle d'apprentissage de Boltzmann, les modifications se font par :

$$\Delta w_{ij} = \eta(\overline{\rho_{ij}} - \rho_{ij})$$

où :

η est le coefficient d'apprentissage et

$\overline{\rho}_{ij}$ (resp. ρ_{ij}) sont les corrélations entre les états i et j lorsque le système est en mode figé (resp. libre-évolution).

Les valeurs $\overline{\rho}_{ij}$ et ρ_{ij} sont classiquement estimées à l'aide de la méthode de Monte-Carlo qui est extrêmement lente.

d)Règle d'apprentissage par compétitions

A la différence de la règle de Hebb (dans laquelle plusieurs neurones peuvent être activés en sortie), cet apprentissage n'active qu'un seul neurone.
Le principe de cet apprentissage est de regrouper les données en catégories. Les patrons similaires vont donc être rangés dans une même classe, en se basant sur les corrélations des données, et seront représentés par un seul neurone, on parle de «winner-take-all ».
Dans un réseau à compétition simple, chaque neurone de sortie est connecté aux neurones de la couche d'entrée, aux autres cellules de la couche de sortie (connexions inhibitrices) et à elle-même (connexion excitatrice).
La sortie va donc dépendre de la compétition entre les connexions inhibitrices et excitatrices.
Le résultat de la compétition est de choisir la cellule ayant la plus grande (ou la plus petite) entrée.
A noter que seules les connexions du vainqueur sont mises à jour.

3 Classification des différents apprentissages

Paradigme	Règle d'apprentissage	Architecture	Tâches
Supervisé	Correction d'erreur	Perceptron Simple ou Multi-Couches	Classification, Approximation de fonctions, Prédiction, Contrôle
	Bolzmann	Récurrente	Classification
	Hebb	Multi-Couches non bouclés	Analyse de données, Classification
	Par compétition	A compétition	Catégorisation au sein d'une classe, Compression de données
		ART	Classification, Catégorisation au sein d'une classe
Non supervisé	Correction d'erreur	Multi-couches non bouclés	Analyse de données
	Hebb	Non bouclé ou à compétition	Analyse de données, compression de données
	Par compétition	A compétition	Catégorisation, Compression de données
		Cartes de Kohonen	Catégorisation, Analyse de données
		ART	Catégorisation
Mixte	Correction d'erreur et par compétition	RBF	Classification, Approximation de fonctions, Prédiction, Contrôle

Tableau 1-1– Synthèse de la classification

Applications

De par leur souplesse et leur capacité d'apprentissage, les réseaux de neurones sont utilisés dans des domaines aussi variés que les jeux, la reconnaissance de caractères, la synthèse vocale, l'analyse boursière, le traitement des images, bref, dans toute activité où l'environnement est par définition en constant changement.
Afin de ce faire une idée plus precise, nous avons choisi d'illustrer les structures de réseaux avec des exemples.

1 Les réseaux feed-forward (Non bouclés)

1.1 Le perceptron multicouches (PCM)

- Chez michelin, est utilisé un réseau de neurones dans un mode de contrôle
  Le facteur influant est passé en entrée, en sortie se trouve les mesures probables, il ne reste plus qu'à vérifier si les mesures sont dans le non intervalle.
  Le réseau utilisé est un perceptron multicouche avec une couche cachée.
  Le type d'apprentissage est un mode supervisé.
  La règle retenue est la correction d'erreur.
- Détection de nuage pour obtenir des images SPOT VGT (capteur européen d'observation de la terre).
  On utilise fonction sigmoïde avec un algorithme par correction de l'erreur.

1.2 Les réseaux à fonction radiale (RBF)

Surveillance d'équipements industriels. L'utilisation d'une fonction radiale permet d'exploiter les propriétés dynamiques des architectures localement récurrentes pour la prise en compte de l'aspect temporel des données d'entrée.
Le réseau de neurones chargé dans l'automate est entièrement configurable à distance par le protocole de communication TCP/IP. Une connexion Internet permet alors à un expert distant de suivre l'évolution de son équipement et également de valider l'apprentissage du réseau de neurones artificiels.

2 Les réseaux feed-back (Récurrents)

2.1 Les cartes auto-organisatrices de kohonen

Etude d'entreprise à travers 2 ratio : le retour d'investissement et la dette total passif.
On utilise alors un réseau de Kohonen à une dimension nommée ficelle.
L'apprentissage compétitif est retenu.

2.2 Les réseaux de Hopfield (RNF)

Utilisation dans de nombreux jeux, comme «fiels of battle (1995)».
L'entraînement à été fait hors ligne

3 Hybride

Aide au diagnostic médical. On utilise un réseau à fonction radiale (RBF) et d'un LVQ.
Le LVQ prendra en entrée la sortie du RBF.
Le RBF fait la mesure de ressemblance alors que le LVQ est utilisé pour la partie décision et classification.

4 D'autres applications

- Reconnaissance de codes postaux
- Contrôle de paramètres du processus de production industriel de pâte à papier (siemens)
- Prévision consommation d'eaux (Lyonnaise des eaux, générale des eaux)
- Finance : prévision et modélisation du marché (cours de monnaies...), sélection d'investissements, attribution de crédits...
- Diagnostic (allumage voiture, Renault ; photocopieur, Canon ; circuits VLSI, IBM...)
- Compression de données (JVC)
- Prévision de consommation d'eau (Lyonnaise, CGE), d'électricité (EDF), du trafic routier (Cofiroute), des cours boursiers...
- Avec Sollac, sur la modélisation du soudage par points, qui est le procédé de soudage le plus utilisé dans l'industrie automobile.
- Avec Michelin, pour concevoir des méthodes d'aide à la conception de pneumatiques.
- Pour 3M, nous avons effectué une modélisation semiphysique du séchage du ruban Scotch
- Exosme est un procédé de filtrage d'informations développé à la Direction des Techniques Avancées de la Caisse des Dépôts à partir des années 1990. Ce procédé a donné lieu à une

application, ExoWeb, qui filtre les dépêches de l'Agence France-Presse en temps réel selon des thèmes prédéfinis.

– Secteur de transport : Régulation de trafic, péage automatique
– ...

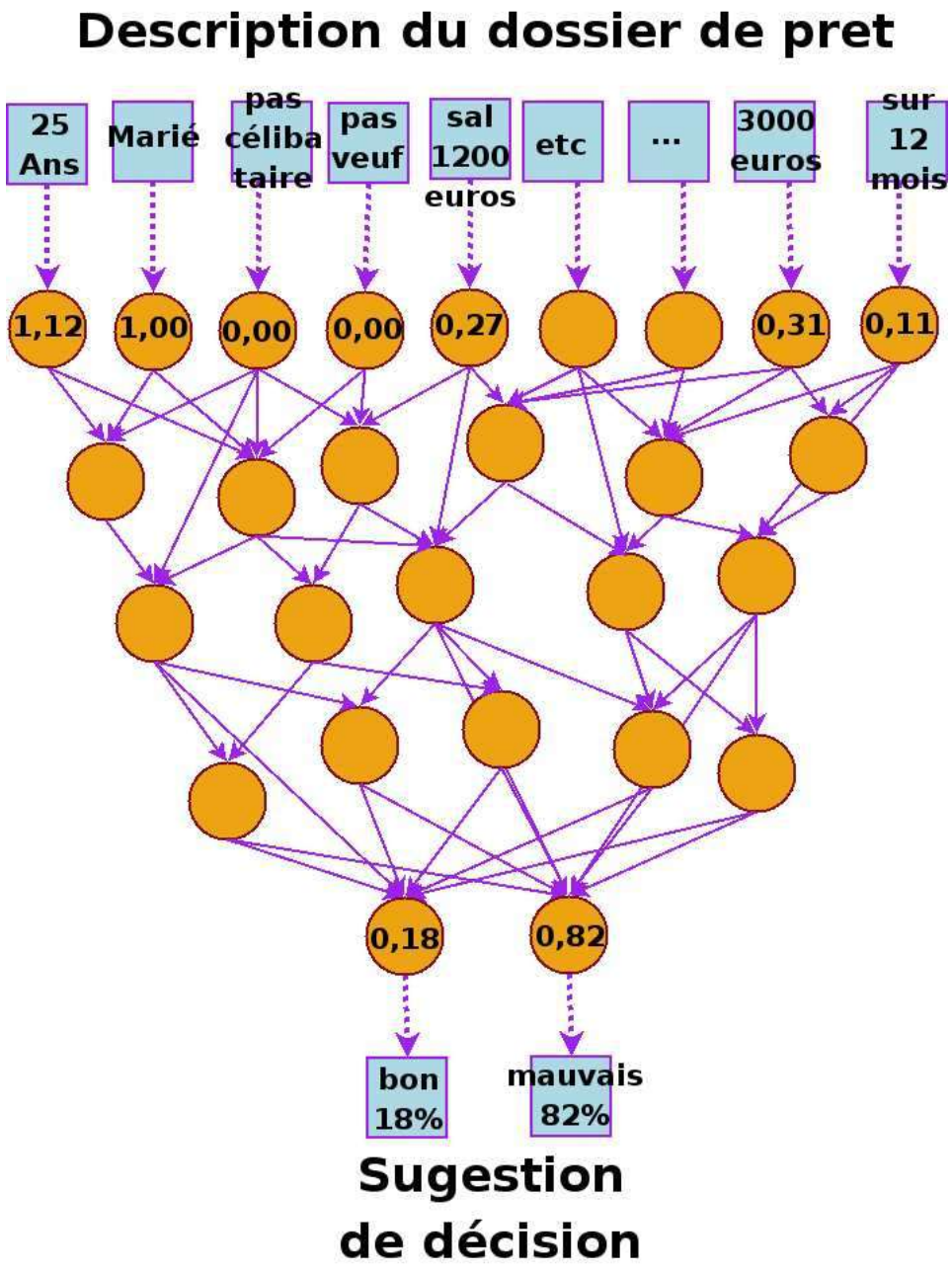


FIG.1-23 – Demande de prêt
Perceptron multicouche

Conclusion

La Présentation faite dans la première partie nous a permis de définir le neurone formel qui est l'élément de base. Malgré son apparence simpliste, nous avons pu voir combien les fonctions d'activation sont importantes et permettent aux neurones d'avoir un comportement différent.

Dans la deuxième partie, nous avons présenté le regroupement des neurones en réseau. Nous avons également découvert, au travers de deux applications, la bio-ingénierie qui permet d'accéder à d'autres utilisations des réseaux.
La première application présentée est un réseau de neurones hybride vivant-artificiel. Elle nous a permis de mieux comprendre les réseaux de neurones biologiques dont découle la compréhension du fonctionnement du cerveau, affinant nos connaissances dans des domaines diverses tel que le filtrage du signal visuel vers le cortex, ou le traitement de la douleur.
L'autre application consiste en une culture de réseaux de neurones biologiques interfacés à des capteurs électroniques. Nous avons introduit la notion de réseaux de neurones analogiques, pou- vant prendre des valeurs continues et ainsi affiner les résultats.
L'une des premières utilisation a été faite sur un robot et permettra de comprendre le processus d'apprentissage du cerveau.

Les parties trois et quatre sont fondamentales, elles nous ont permis d'établir une classification par topologie ainsi que par les différentes méthodes d'apprentissage. Le tableau 19 est une synthèse pertinente dégageant une méthodologie dans le choix d'un réseau pour une tache donnée.

De nombreuses applications sont gérées par des réseaux de neurones souvent insoupçonnés car intégrés dans nos outils informatique. Nous avons donné une liste non exhaustive des applications couvrant des domaines variés tel que la détection, la surveillance, le jeu, la reconnaissance de caractères, la classification, la prévision, les finances ...

A la base, les réseaux de neurones avaient été créés pour copier et modéliser le cerveau humain mais aujourd'hui l'utilisation de la bio-ingénierie nous permet d'améliorer la compréhension de cette organe qui reste le plus complexe et le moins maîtrisé.

CHPITRE II :

programmation de réseau de neurones CMAC

... l'intelligence n'est pas l'instrument
le plus subtil, le plus puissant, le plus
approprié pour saisir le Vrai...
... c'est la vie qui peu à peu, cas par cas,
nous permet de remarquer que ce qui est le
plus important pour notre coeur, ou pour
notre esprit, ne nous est pas appris par le
raisonnement mais par des puissances autres.
(Marcel Proust)

Mot Clef « « Application en temps réel » »

I. Introduction

Comme nous l'avons vu au cours du précédent Chapitre, la notion d'intelligence dans un réseau de neurones artificiels réside dans sa capacité à *apprendre* l'environnement avec lequel il interagit et à *généraliser* ces connaissances.

En s'inspirant des deux caractéristiques qui définissent l'intelligence :

Potentiel à apprendre grâce à son interaction avec l'environnement dans lequel il est immergé, et **capacité** à mémoriser localement de nouvelles informations,

Le modèle de réseau neuronal CMAC est basé sur le fonctionnement biologique du cervelet. Le cervelet est en charge de tous les mouvements du corps et de leur précision. Il est capable de résoudre plusieurs problèmes qui se posent dans le contrôle de systèmes dynamiques, comme ceux de la robotique [Collins et al.99].

Plusieurs théoriciens ont modélisé le comportement biologique du cervelet. L'un d'entre eux, James Albus, a proposé un modèle mathématique simplifié du comportement du cervelet appelé « ***CMAC*** ». « ***CMAC*** » est l'acronyme de l'anglais « ***C***erebellar Model Articulation Controller ».

Les réseaux neuronaux les plus utilisés jusqu'à présent, ont été les réseaux Multicouches avec apprentissage par rétro-propagation du gradient. Cependant, il a été démontré dans les travaux de recherche de Miller [Miller et al.90] que ce type de modèle est mal adapté dans le cadre d'applications qui exigent la réalisation de l'approximation du comportement (apprentissage) en temps réel. En revanche, le modèle du CMAC [Albus75b], permet un apprentissage en ligne grâce à la simplicité de son fonctionnement interne.

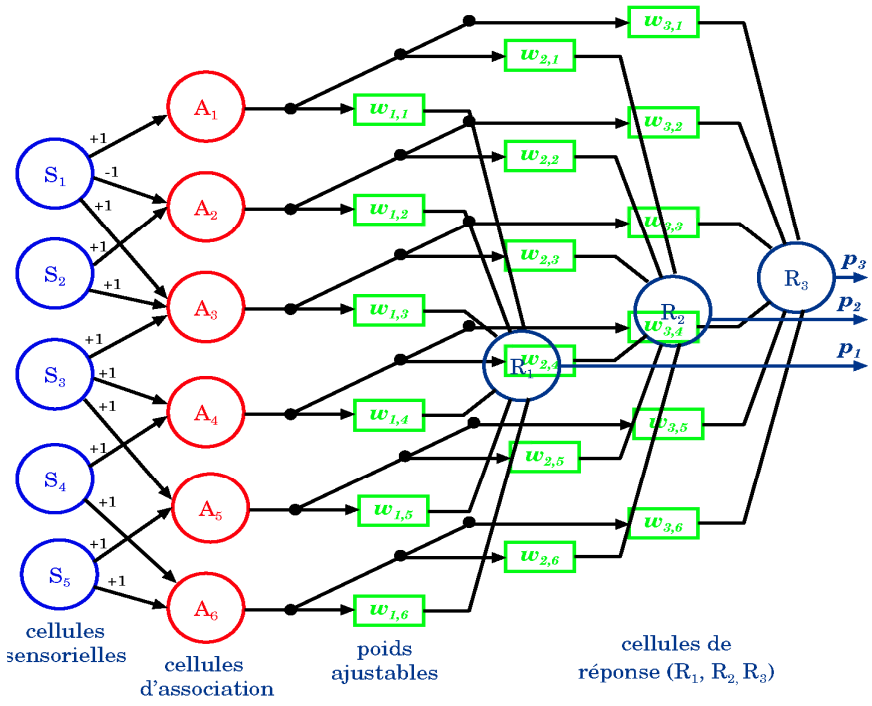
Dans le présent chapitre, nous décrirons le fonctionnement du réseau de neurones CMAC et nous ferons une analogie avec le fonctionnement, d'un point de vu biologique, du cervelet humain. Ensuite, nous présenterons le modèle mathématique proposé par James Albus. A travers des exemples concrets, nous détaillerons la structure, le fonctionnement et les propriétés du réseau de neurones CMAC.

Qu'est-ce qu'un Réseau Neuronal CMAC?

Les avancées dans le domaine de la biologie ont permis d'accroître les connaissances de la structure et du fonctionnement du cervelet humain. Ces nouvelles compétences ont amené plusieurs théoriciens à proposer des modèles mathématiques pour expliquer les caractéristiques du traitement de l'information dans cet organe. au milieu des années 1970, James Albus, a développé une architecture neuronale dotée de propriétés similaire a celle du cervelet et l'a nommée CMAC (***C***erebellar ***M***odel ***A***rticulation ***C***ontroller) [10]. Cependant, ce n'est qu'à partir du début des années 1980 [11] que les travaux sur le Réseau de Neurones CMAC et son application dans beaucoup de domaines et surtout en robotique ont repris de manière plus intense.[9]

En effet, le principal avantage du CMAC est la rapidité de son exécution et de son apprentissage. Cette vitesse de calcul lui permet d'être utilisé dans des applications en temps réel qui exigent un apprentissage en ligne et incrémentiel. C'est le cas par exemple du routage de paquets dans un réseau de télécommunication ou du contrôle des mouvements d'un robot Bipède .

II-a-CMAC comme un perceptron classique



FlG. 2-1 *Perceptron classique que J. Albus a utilisé pour décrire le comportement du cervelet.* La figure2.1 montre un modèle équivalent à celui du perceptron que James Albus [10] a utilisé pour décrire le fonctionnement du cervelet. Ce modèle est basé sur la ressemblance au niveau biologique entre le cervelet et le perceptron. Dans cette figure, les vecteurs d'entrées Sj représentent les « cellules sensorielles ». Ces vecteurs d'entrée ne contiennent que des valeurs binaires. Lorsqu'un vecteur d'entrée est présenté, un vecteur d'association Ak est généré de manière arbitraire. Pour chaque cellule d'association, la somme des valeurs des cellules sensorielles qui lui sont reliées pondérées par le poids de la liaison entre les deux cellules est effectuée. Si la somme obtenue est positive alors la cellule d'association sera activée avec une valeur de 1 sinon elle sera inactivée avec une valeur de 0.

II-b-Différence entre un Réseau Neuronal CMAC et perceptron multicouche?

La sortie du réseau CMAC est linéairement dépendante d'un ensemble de poids, ce qui n'est pas le cas du Perceptron qui utilise tous les poids des neurones pour calculer la sortie. De plus, cette sortie est limitée par une fonction, la rendant ainsi binaire ;

Les connexions entre les neurones dans le CMAC sont déterminées grâce à un ensemble de paramètres prédéfinis par le concepteur du réseau. A l'inverse, les connexions des neurones du Perceptron sont formées de manière aléatoire et éparse ;

La propriété de généralisation dans le Perceptron est dite globale car pour une entrée donnée, l'ensemble des poids qui le constituent est modifié par le processus d'apprentissage, ce qui engendre dans certains cas une instabilité du réseau. En revanche, on dit du CMAC qu'il réalise une généralisation locale du fait que seuls quelques poids sont modifiés à chaque sollicitation, ce qui contribue à la convergence du réseau vers un état stable plus rapidement (pas de problèmes des minimums locaux).

II-c-L'algorithme du CMAC

Découpage de l'espace d'entrée

En plus du paramètre de généralisation, le concepteur du réseau, doit définir la grille de dimension n qui normalise l'espace d'entrée. Cette opération de treillage, que nous avons décrit précédemment (*fig2-1)*, implique la détermination des valeurs minimales et maximales qui bornent l'espace d'entrée sur lequel le

CMAC sera défini. De plus, la position de l'ensemble des r_i nœuds répartis sur chaque axe doit être fixée. Ceci se traduit concrètement, par la spécification du *paramètre de quantification* q qui subdivise chaque axe en un ensemble de

plusieurs intervalles. Dans cet espace normalisé, l'index qui détermine la position des nœuds est donné par l'équation suivante :

$$ad(j) = \left[\frac{q_i + d_i^j}{\rho}\right] \qquad (2\text{-}1)$$

avec $i = 1,...,\rho$, $j = 1,...,n$, et où $\boldsymbol{d}_i^j$ représente un vecteur de dimension n dont les éléments sont entiers et définissent la stratégie de déplacements des couches. L'utilité ainsi que le calcul des éléments qui le constituent seront décrits en détails plus loin dans ce chapitre.

No

us mettons l'accent sur la description de la fonction non linéaire ***f*** car elle joue un rôle prépondérant dans la conception de la structure. Elle permet la mesure des distances entre les vecteurs d'entrées pour définir le degré de similitude entre eux. Mesurer ces distances dans l'espace réel des entrées n'étant ni fiable ni facile, la fonction ***f*** crée un espace normalisé dans lequel ces distances sont calculées plus aisément et avec une plus grande précision.

La généralisation locale

La notion de généralisation locale que l'on retrouve dans le CMAC est due au fait que l'on utilise la fonction non linéaire ***f*** pour associer, à chaque fonction de base un support (champ réceptif) dont la forme est un hypercube de volume $\boldsymbol{\rho}^n$. Par conséquent, nous pouvons dire que le paramètre de généralisation ne définit pas uniquement le nombre de poids qui contribuent au calcul de la réponse, mais détermine aussi, la forme et la taille des champs réceptifs qui leur sont associés. L'augmentation du paramètre de généralisation $\boldsymbol{\rho}$ faisant croître la taille des champs réceptifs, sa capacité à généraliser localement l'information emmagasinée au cours de l'apprentissage se trouve diminuée. En effet, plus ces hypercubes sont volumineux, plus l'efficacité du CMAC à modéliser la dynamique de la fonction à apprendre diminue.

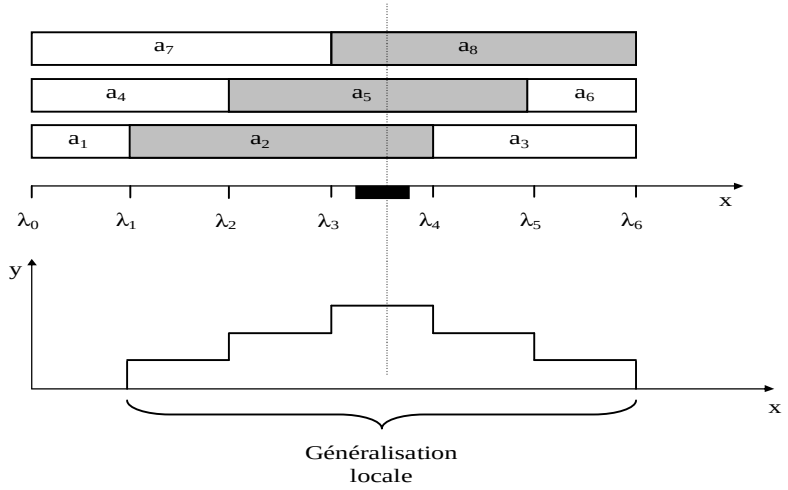


Fig 2.2 : Principe de généralisation locale dans un CMAC à une dimension.

La sortie des fonctions de base est non nulle si et seulement si le champ réceptif, de forme hypercubique, auquel elle est associée, a été activé par un vecteur d'entrée $\boldsymbol{x}$. Les hypercubes sont répartis de telle façon que les supports des $\boldsymbol{\rho}$ fonctions de base couvrent partiellement chaque case de la grille qui normalise l'espace d'entrée. Par conséquent, pour mettre en œuvre cette structure particulière, un ensemble de $\boldsymbol{\rho}$ couches est formé ou,

chacune d'elle est constituée d'un certain nombre de champs réceptifs adjacents. La fig2.2 illustre la structure d'un CMAC monodimensionnel avec un paramètre de généralisation $\boldsymbol{\rho}$ = 3 et où les trois couches qui constituent cette structure sont décalées successivement d'une case à la fois.

La méthode de répartition des $\boldsymbol{\rho}$ couches les unes par rapport aux autres permet de réaliser une projection uniformément des entrées sur chacun des axes de l'espace d'entrée. De cette façon, lorsque le CMAC est sollicité par un vecteur d'entrée qui se déplace le long d'un axe donné par pas d'une case à la fois, le jeu de poids qui lui est associé aura ($\boldsymbol{\rho}$ - 1) éléments en commun avec l'ensemble précédent. Par conséquent, si deux vecteurs sont distants de plus de ($\boldsymbol{\rho}$ - 1) cases dans l'espace normalisé, les ensembles de poids qui leur sont assignés n'auront aucun élément en commun. C'est grâce à cette définition du voisinage dans l'espace d'entrée normalisé, que le CMAC arrive à généraliser localement les informations qu'il emmagasine.

Efin, la structure du CMAC ne sera totalement définie que si le paramètre de généralisation $\boldsymbol{\rho}$ satisfait la relation suivante : $1 \le \rho \le \max_{i=1}^{n}(r_i + 1)$ (2.2)

Ce n'est qu'en respectant cette condition qu'on peut éviter que plusieurs supports (champs réceptifs) ne

recouvrent complètement une case de la grille (espace d'entrée normalisé), engendrant ainsi une confusion entre les fonctions de bases qui leurs sont associées.

Cas multidimensional

Jusque ici, nous avons décrit les étapes nécessaires à l'élaboration de la structure du réseau de neurone CMAC à une dimension, mais nous avons, surtout, montrer la manière avec laquelle il arrive à généraliser localement les informations qu'il acquière. Pour généraliser cette structure sur des problèmes à plusieurs dimensions, nous allons décrire et illustrer la structure d'un CMAC à deux dimensions en entrée.

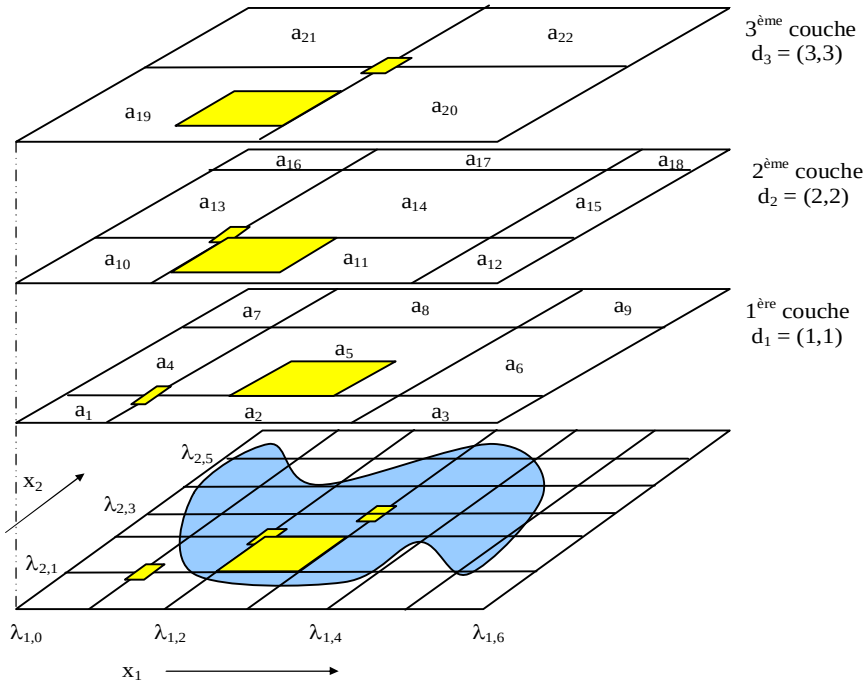


Figure 2.3 : Structure d'un CMAC à 2 dimensions avec ρ = 3 et 22 supports de fonctions de bases.

La figure 2.4 représente la structure d'un réseau de neurones CMAC à deux dimensions avec un paramètre de généralisation ρ égal à 3. Chaque dimension de l'espace d'entrée est discrétisée en un ensemble de 6 cases (5 nœuds intérieurs $\lambda_{i,j}$ avec $i = 1,2$ et $j = 1,...,6$) formant ainsi une grille composée de 36 cases. Pour recouvrir cet espace d'entrée normalisé, 22 fonctions de base sont utilisées et dont les champs réceptifs sont réparties sur trois couches obtenues par un déplacement de l'une par rapport à l'autre, d'une case le long de la diagonale de la grille.

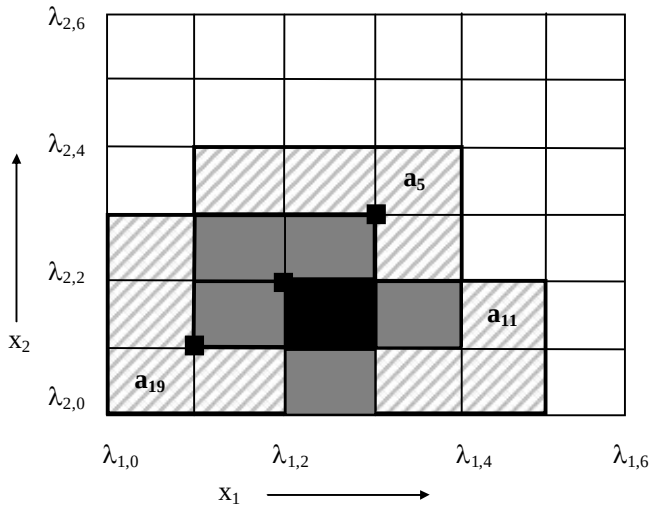


Fig2.4: Principe de généralisation dans un CMAC à 2 dimensions.

La propriété de généralisation locale du CMAC est définie par la relation de voisinage qui lie les ρ fonctions de base activées par un vecteur d'entrée x. Pour illustrer cette propriété dans un espace multidimensionnel, considérant la fig2.4 ou la case noire représente la projection d'un vecteur d'entrée (référence) x sur la grille 2D qui normalise l'espace d'entrée. Quant aux cases qui sont colorées en grille et en grille claire hachurées, elles partagent, respectivement, deux et une fonction de base avec l'entrée de référence. Par conséquent en utilisant une règle d'apprentissage classique LMS ("***Least Mean Square***") par exemple, seuls les poids qui seront associés à ces fonctions de base seront ajustés. Par conséquent, la réponse du réseau sera modifiée que dans cette petite région. On dit alors que le CMAC sauvegarde et apprend localement l'information. Cette propriété dépend donc de la stratégie de distribution des fonctions de bases le long de l'espace d'entrée normalisé et donc du déplacement des ρ couches qui composent la structure du CMAC.

Stratégies de répartition des couches

De nombreux travaux ont été menés, au cours de ces dix dernières années, pour définir des stratégies efficaces de répartition des couches sur la grille qui normalise l'espace d'entrées. Parmi ces travaux qui

ont abouti à des méthodes performantes, nous citons celle de (An 1991) et (Parks and Militzer 1991). Dans ces publications,

la méthode qui est décrite consiste à positionner, d'abord, un coin de la première couche sur cette grille, pour déterminer, ensuite, un ensemble de $\boldsymbol{\rho}$ vecteurs (un par couche) qui doivent satisfaire la définition suivante :

> « Si deux points x_1 et x_2, appartenant à l'espace d'entrée normalisé, alors la distan
> de Hamming qui sépare les doux vecteurs $\boldsymbol{a_1}$ et $\boldsymbol{a_2}$ qui leur est associée e
> proportionnelle à la distance Euclidienne entre ces deux points x_1, x_2 sur la grille. »

Satisfaire cette condition pour le calcul des vecteurs de déplacement, revient en somme à appliquer le principe de projection uniforme. Dans la méthode de Parks et Militezer, tous les éléments du vecteur de déplacement $(d_1, d_2, ..., d_n)$ sont premiers avec $\boldsymbol{\rho}$ et sont compris entre [0, $\boldsymbol{\rho}$]. Ils sont définis par les relations suivantes :

$$\boldsymbol{d_1} = (d_1, d_2, ..., d_n)$$
$$\boldsymbol{d}_2 = (2d_1, 2d_2, ..., 2d_n)$$
$$\vdots$$
$$\boldsymbol{d}_{\rho-1} = ((\rho-1)d_1, (\rho-1)d_2, ..., (\rho-1)d_n)$$
$$\boldsymbol{d}_{\rho} = (\rho, \rho_2, ..., \rho)$$

où le calcul des éléments d_i se fait en utilisant l'opérateur arithmétique *modulo* de $\boldsymbol{\rho}$. Pour éviter de refaire le calcul à chaque fois, Parks et Militezer publient une table relativement complète englobant les valeurs des vecteurs de déplacement pour des CMAC dont le paramètre de généralisation varie de 1 à 100 et pour des dimensions d'entrées pouvant aller jusqu'à 15. Ne pouvant la donner complètement dans ce rapport, une version réduite de cette table est présentée en annexe (B). La valeur de ces vecteurs n'est donnée que pour $1 < n \leq 6$ et $1 < \rho \leq 20$ car lors de nos manipulations et nos application la dimension des CMAC, ainsi que le paramètre de généralisation ne dépassent jamais ces valeurs.

La taille du CMAC

L'un des principaux inconvénients du CMAC est son besoin important en mémoire informatique. Pour un CMAC à n dimension d'entrée et à une dimension de sortie, le nombre de poids est équivalent au nombre

p de fonctions de bases qu'il utilise. Si en revanche, on augmente la dimension de sortie à m, le nombre de poids qui devront être stockés en mémoire sera tout simplement$m * p$. Ceci étant, en comparaison avec une table associative simple, une réduction en ressources informatiques (mémoire) est obtenue grâce à l'introduction du paramètre de généralisation.

Quoiqu'il en soit l'un des facteurs qui pénalise le plus ce réseau de neurones restera le nombre relativement important de champs réceptifs qu'il faut utiliser pour apprendre un problème complexe. Théoriquement, du fait que la taille du CMAC croit avec la dimension n de l'espace d'entrées, ce réseau de neurones ne conviendrait qu'à des applications de taille moyenne. Pour pallier, d'une certaine façon, à cet handicape Albus (Albus 1975b) utilise une technique de compression appelé "*Memory Hashing*". Cette technique, très utilisée en informatique, consiste à sauvegarder le contenue d'une mémoire de capacité élevée, désignée dans le cas du CMAC, par mémoire virtuelle, dans une mémoire de capacité moindre, désignée par mémoire physique. Pour ce faire, on définit une transformation pseudo aléatoire de telle façon à ce que la sortie de la mémoire

virtuelle soit une adresse de la mémoire physique est c'est cette dernière qui fournit l'information utile. La figure (2.5) qui illustre ce principe dans le CMAC, permet, à première vue, de soulever un problème majeur; une information peut être écrasée par une autre quand il y a *Collision*. Le terme *Collision* est défini par Albus comme étant la transformation de deux adresses mémoire totalement distinctes en une seule et même adresse de la mémoire physique. L'utilisation de cette technique pour réduire la taille du CMAC engendre deux catégories de *Collision* (fig 2.5) :

- celles qui se produisent quand, suite à une sollicitation du réseau par un vecteur d'entrée, une même adresse de la mémoire physique est activée par deux ou plusieurs adresses de la mémoire virtuelle,
- ou des *Collisions* quand une même adresse de la mémoire physique est activée par deux ou plusieurs adresses de la mémoire virtuelle correspondant à différents vecteurs d'entrée.

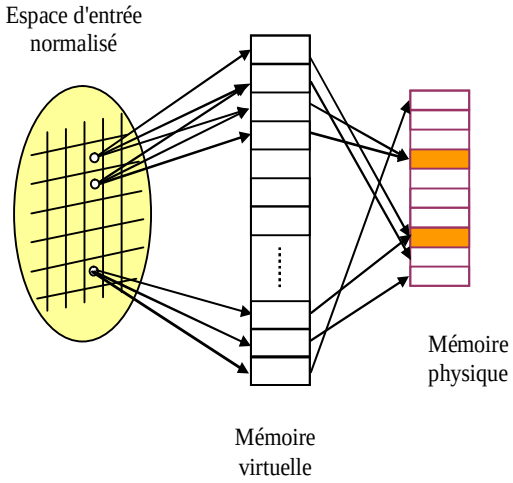


Figure2.5 : Principe de la technique de " *Memory Hashing* " représentant les deux catégories de *collisions*.

En théorie, un choix judicieux de la taille de la mémoire physique permet de réduire la probabilité de *Collision*. En effet, plus la taille de la mémoire physique est petite, plus il y aura de *Collisions*, ce qui entraînera une dégradation des performances du CMAC.

Il est aussi possible d'intégrer, à l'algorithme du CMAC, une routine qui détecte les *Collisions* et agit sur l'algorithme pour qu'il réitère le processus d'adressage jusqu'à ce qu'il trouve une adresse libre. Par contre, cette astuce algorithmique implique d'une part, que la transformation soit réellement pseudo aléatoire et d'autre part, que la taille de la mémoire physique soit suffisante (9).

Finalement, l'utilisation de la technique de "*Memory Hashing*" peut être intéressante dans la mesure ou la fonction que l'on désire approximer recouvre qu'une partie de l'espace d'entrée. En revanche, cette technique est inappropriée pour de applications qui nécessitent d'apprendre tout l'espace d'entrées. C'est le cas des applications robotiques qui nous intéresse, où le CMAC doit apprendre, en tout point de l'espace de travail du robot, la fonction robotique qui lie l'espace articulaire du manipulateur à celui des images.

L'apprentissage du CMAC

Le processus d'apprentissage du RN CMAC consiste a ajuster les poids pour minimiser une erreur entre la sortie réelle du réseau « y_i » et une sortie désirée « y_d ». Cet ajustement est fait a l'aide de la « règle delta » (nommée également règle d'apprentissage de Widrow-Hoff) basée sur la règle des moindres carrés ("Least means square", ou LMS).

Lorsqu'une entrée x_i est présentée au CMAC et la sortie y_i calculée, l'apprentissage consiste a modifier chaque élément wj_k du vecteur colonne j de la matrice de poids W de manière a ce que la sortie y_j devienne la plus proche possible d'une sortie désirée donnée y_d.

D'après l'équation 6, la sortie yj du RN CMAC est égale a:

$$y_j = \sum_{k=1}^{k} a_k * w_{jk} \qquad (2.3)$$

Où K est le nombre total de poids équivalent au nombre de neurones. Si chaque neurone du RN CMAC contribue dans la même proportion pour la sortie, alors, la sortie pour chaque neurone devient:

$$\text{Sortie pour un neurone} = \frac{y_j}{k} \qquad (2.4)$$

Comme dans le réseau neuronal Adaline ('Adaptive linear element"), l'équation 8 est un modèle linéaire exact avec K paramétres. Ainsi, l'utilisation des méthodes des "LMS" est possible. Cependant, la plupart des méthodes de "LMS" exigent un calcul extensif qui est impossible pour les systèmes usuels relativement limite en puissance de calcul.

Pour surmonter ce problème, Widrow and Hoff ont introduit la « regle delta » qui permet de régler les poids de manière simplifiée. Pour une sortiey_j, l'erreur pour un neurone peut être mesurée par l'équation suivante:

$$E_{kj} = \frac{1}{2k}(y_d - y_j)^2 \qquad (2.5)$$

En tenant compte de l'équation(2.6) :

$$E_{ij=}\frac{1}{2k}(y_d - a_1 w_{1j} - a_2 w_{2j-} a_3 \ldots . a_n w_{nj})^2 \qquad (2.6)$$

Le gradient de E_{ij} par rapport a chaqué poids w_{kj} est alors égal a:

$$\frac{\partial E_{kj}}{\partial w_{kj}} = \frac{1}{2k}(-2a_k(y_d - y_j)) \quad (2.7)$$

Ainsi, on diminue E_{ij} par la méthode du gradient, pour chercher son minimum. Cela représente la modification des poids en sens inverse du gradient. La formule de mise á jour pour la modification des poids Δ_{wk} est égale a:

$$\Delta w_{kj} = -\alpha \nabla E_{ij} = \beta \frac{a_k}{k}(y_d - y_j) \quad (2.8)$$

ou $\beta \in [0,1]$ est le taux d'apprentissage et K le nombre de poids qui ont contribué pour obtenir la sortiey_j.

. Si β est égal a 1, la sortie y_j sera égale á la sortie désirée y_d. Si β est égal a 0, la sortie ne changera pas.

La formule 12 montre intuitivement que si $y_d > y_j$, nous voulons augmenter $y_j$par l'élévation de $a_k w_k$. Si $y_d < y_j$, nous voulons diminuer y_j par la diminution de $a_k w_{kj}$.

Nous pouvons constater que la fonction d'erreur de l'équation (2.5) est fonction du carré de w_{kj}. Cette fonction donne une surface d'erreur quadratique, qui ne comporte qu'un seul minimum [12]. L'absence de minima locaux permet une convergence rapide de l'algorithme d'apprentissage.

Les paires des entrées et des sorties désirées (x_i, y_d), normalement appelées points d'apprentissage, peuvent étre présentées au CMAC de deux manieres différentes:

- Dans un ordre aléatoire : si le CMAC est entrainé à partir d'une base d'exemples non ordonnée.
- Dans un ordre pre-établi: si le CMAC est utilisé dans un contrôleur en ligne. Les points d'apprentissage vont varier graduellement. Dans ce cas, chaque point d'apprentissage sera près d'un autre dans l'espace d'entrée.

Les principales caractéristiques de la règle d'apprentissage du RN CMAC sont:

- La simplicité: mise en évidence dans l'équation (2.8)

- L'apprentissage local: l'apprentissage ne repose pas sur un contrôle global du réseau. II peut être accompli au niveau local de chaque nœud.

- L'apprentissage en ligne: la mise à jour des poids peut être faite après la présentation de chaque point d'apprentissage.

Le phénomène d'interférence

En plus de la capacité du réseau de neurones CMAC à apprendre et à généraliser les informations acquises au cours de l'apprentissage, ce réseau se distingue par la convergence rapide de son algorithme. En partant de cette propriété, on peut s'imaginer que l'augmentation de la valeur du paramètre de généralisation ρ implique qu'il y a, à chaque sollicitation, plus de poids pour calculer la sortie, ce qui aura pour finalité une erreur *MSE* plus petite. Cependant, cette approche peut, dans certains cas, susciter quelques problèmes. En effet, plus on augmente la généralisation du réseau, plus le risque d'altérer le contenu de la mémoire du réseau est élevé. On parle alors de phénomène d'interférence ou "*learning interference*".

Pour illustrer ce problème, considérons l'apprentissage de deux points x_a et x_b relativement loin l'un de l'autre, dans l'espace d'entrée (figure 2.6).

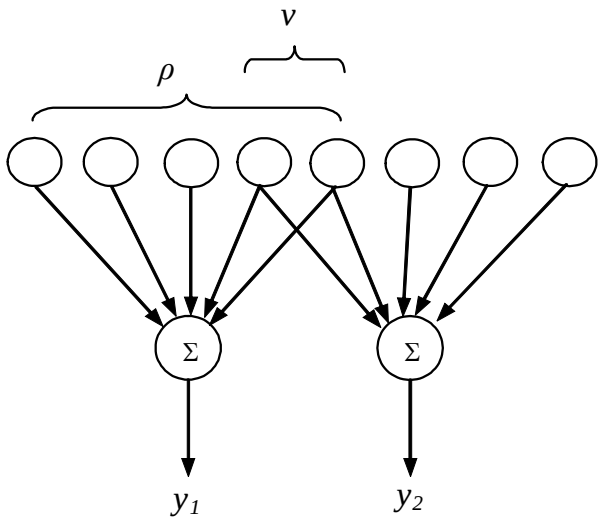


FIG 2.6illustration du phénomaine d'interférence

Plus on augmente le paramètre de généralisation plus le nombre v de poids en commun sera important. Par conséquent, l'apprentissage du point x_b affecte la sortie du réseau y_a, quand celui-ci sera sollicité par l'entrée x_a car les v poids qui contribuent au calcul de y_a ont été modifiés. Ce phénomène s'avère problématique pour les applications de suivi de trajectoire, étant donné que les points qui la constituent et que doit apprendre le CMAC, se trouvent très proches les uns des autres. Comme en pratique il est très difficile de prédire ce phénomène d'interférence sur un jeu d'apprentissage réel, une solution pour réduire ces effets sur les performances du réseau consiste à faire décroître le coefficient d'apprentissage, et surtout à ne pas trop augmenter la valeur du paramètre de généralisation.

- Diagramme du RN CMAC Multi-entrées Multi-sorties

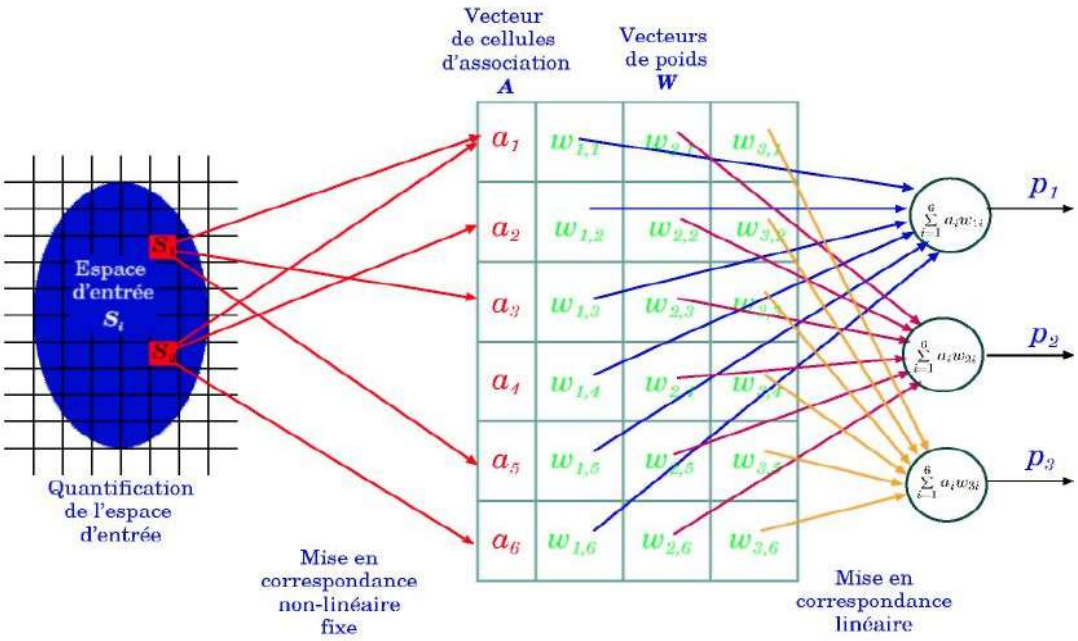


FIG. 2.7 - *Diagramme général du fonctionnement du RN CMAC MIMO*

Dans un premier temps les données vont être discrétisées (ou quantifiées) en « champs réceptifs ». Le nombre de « champs réceptifs » que la valeur d'une variable d'entrée pourra activer dépendra du niveau de généralisation souhaité. Ensuite on associe à chaque « champ réceptif » un ensemble unique de taille fixe de |A| « cellules d'association » de manière arbitraire (voir la table 5) ou en utilisant un algorithme de mise en correspondance. Lorsqu'une entrée est discrétisée en un « champ réceptif », les « cellules d'associations » liées à ce champ vaudront 1 alors que les autres vaudront 0.

Les cellules dans la matrice de poids contiennent des valeurs réelles parmi lesquelles le vecteur A permet de sélectionner celles qui seront additionnées pour obtenir la sortie du CMAC

L'apprentissage du RN CMAC sera accompli en ajustant les valeurs réelles des cellules de la matrice de poids. La jme sortie du RN CMAC est comme précédemment obtenue en effectuant:

$$[a_1, a_2 \quad \cdots \quad a_m] \begin{bmatrix} w_{j,1} \\ w_{j,2} \\ \vdots \\ w_{j,m} \end{bmatrix} = p_j \qquad (2.8)$$

L'amplitude de variation des composantes des vecteurs d'entrée S dépendra du système pour lequel CMAC est utilise [MossoOO]. Par exemple, dans le cas du contrôle d'un bras

manipulateur de robot, nous avons un vecteur d'entrée qui contient un élément. Cet élément comprend une variable angulaire [0,360°].

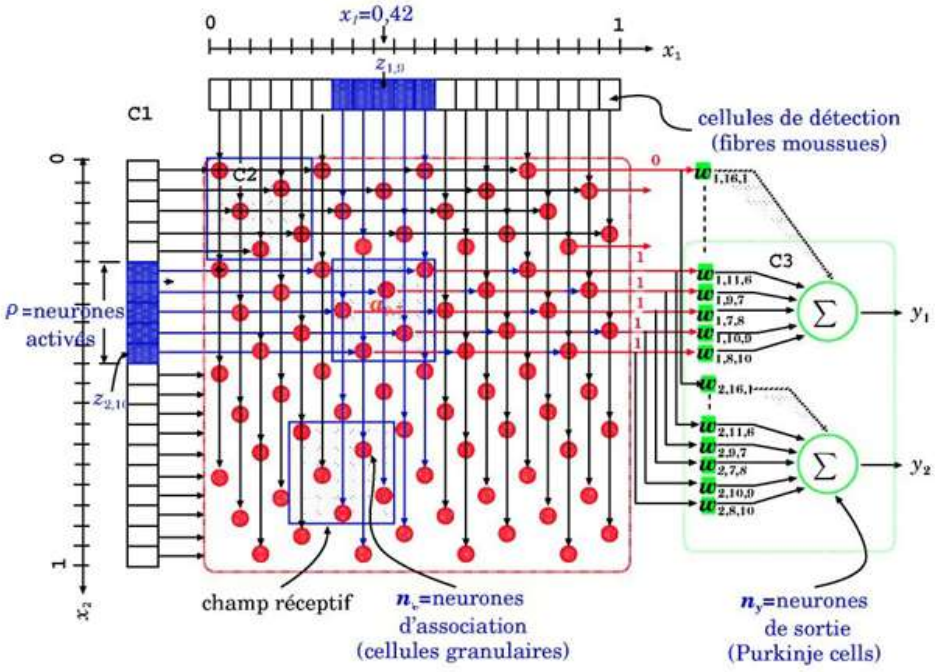
Une autre représentation de la structure du CMAC

Nous avons abordé la structuredu réseau de neurones CMAC sous l'aspect géométrique. Même si cette approche reste la plus utilisée dans les ouvrages consacrés à la description du CMAC, il existe une autre façon de décrire la structure de ce réseau. Cette derniere fait intervenir , implicitement, la notion de neurones. Nous allons faire, dans cette partie, l'analogie entre cette approche plus neuronale et la description précédente et cela en nou basant sur un exemple 2D concret.

Dans la figure (2.8) nous présentons un exemple d'un Réseau Neuronal CMAC (adapté de [Smith98]) avec deux entrées et deux sorties. Cette représentation contient trois couches appelées Cl, C2 et C3. Les variables du vecteur d'entrée X sont x_1 *et* x_2, avec $x_1 \in$ [0,1] et $x_2 \in$ [0,1] .

Dans l'exemple, x_1 est égal a 0,42 et x_2 est égal á 0,35. La couche 1 (Cl) contient un tableau de cellules appelées « cellules de detection Z_i» pour chaqué entrée *xi*. La sortie d'une cellule de detection $Z_{i,k}$

dépendra de la valeur d'entrée x_i. Cette sortie aura une valeur de 1 pour certaines cellules de detection dans un intervalle limité, et 0 pour toutes les autres.



-FIG. 2.8- *Exemple d'un RN CM AC avec deux entrées*

Pour commencé il faut bien faire attention a la mise en correspondance entre l'espace d'entrées discrétisé et les cellules de détection pour cela il faut bien respecté la condition :

Que seul **ρ**cellules sont active par une entréedonnée.

Dans notre exemple notre espace d'entrée = [0,1] est discrétisé en 20 zones, ces zones-là active (pour notre exemple ρ = 5) 5 cellules de détection à la fois voici le diagramme :

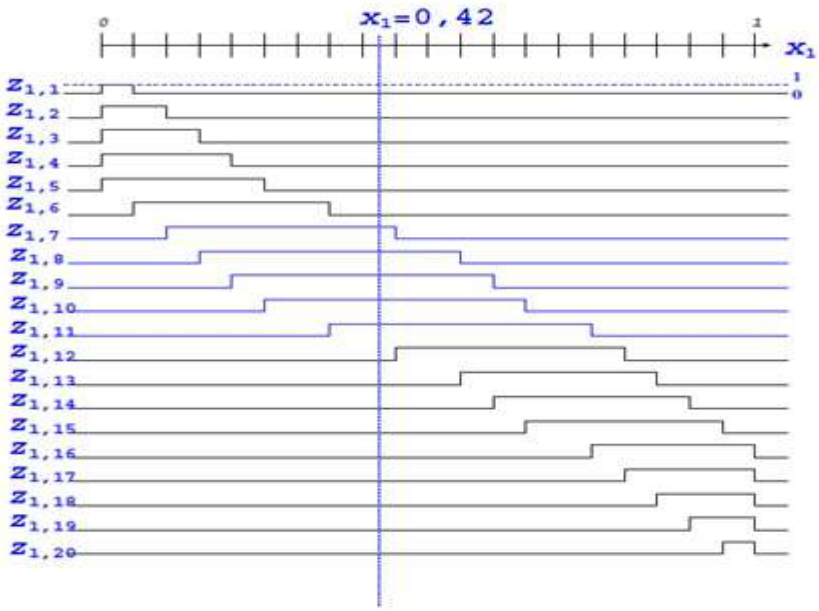


FIG 2.9 : cellule de detection activé par l entrée x1=0.42

En constate que l'entrée x1=0.42 correspond aux paramètre de quantification q1=8 de ce fait il en result cinq cellules de détection active (ρ 5)$z_{1,7}$, $z_{1,8}$, $z_{1,9}$, $z_{1,10}$, $z_{1,11}$

Les cellules de détection sont semblables à un filtre passe-bande qui ne laisse passer qu'un certain nombre de fréquences. La fonction d'activation pour l'entrée*x*\ est montrée dans la figure (2.8). Pour chaque

valeur d'entrée *Xi* un nombre ρ déterminé de « cellules de détection » *z* seront activées dans la couche 1. Pour cet exemple, ρ = 5.

Nous pouvons constater que les cellules de detection de la couche 1 (CI) servent a quantifier les entrées*Xi*.

Apres cette étape en commence a construire la grille (cellule d'association) , pour cela on commence par construire la premiére couche figure (2.10)

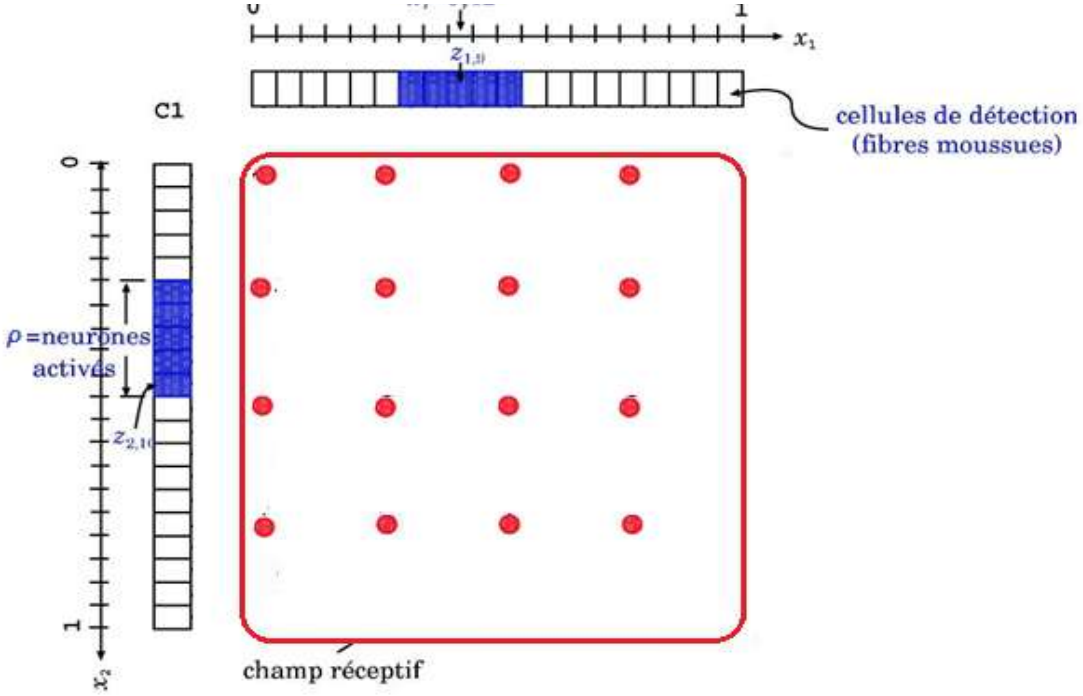
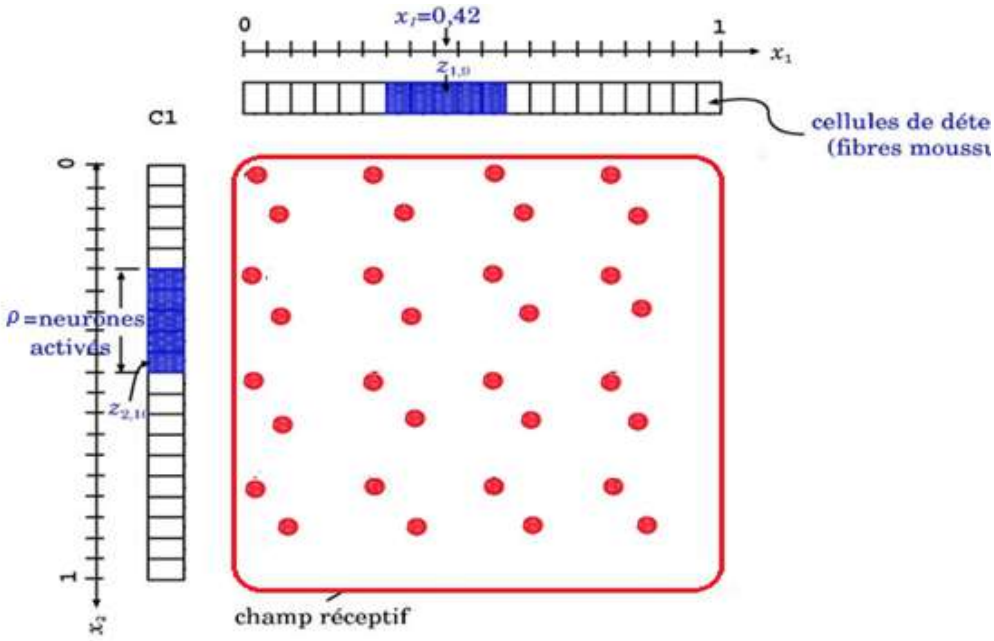


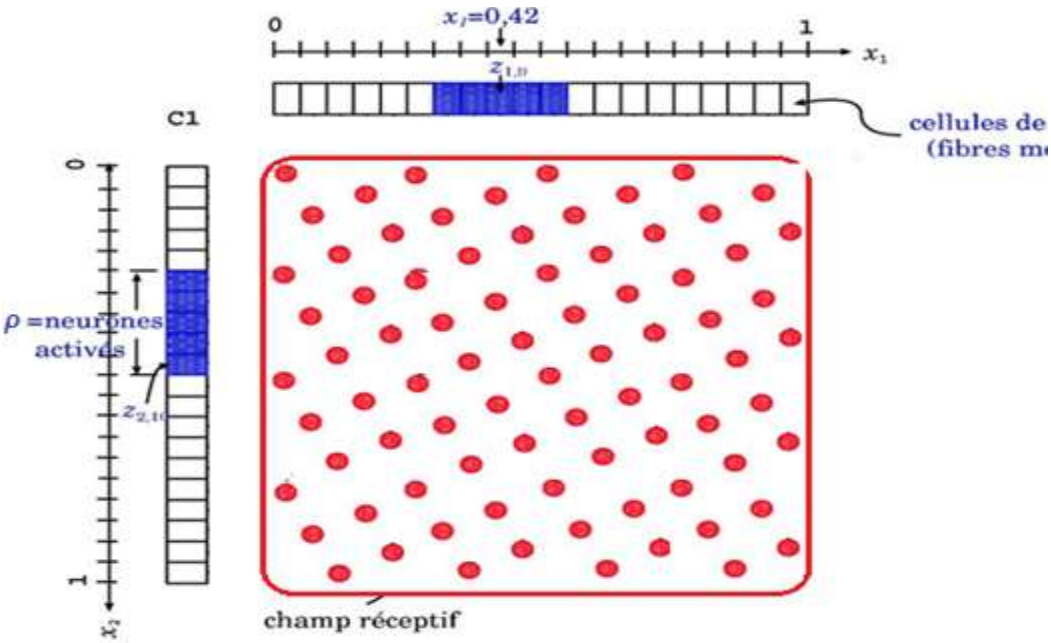
FIG 2.10 Mise en forme de la 1er Couche

On continue avec la 2ème couche en respectant le vecteur de deplacement d=[1 2] qui correspond a un deplacement d'une cellule sur la 1erentrée et deux cellules sur la 2em entrées on obtienderas une grille comme la figure(2.11)



FIG(2.11) : Mise en forme de la 2ème Couches

De la même manière en continue jusqu'à la 5ème couche pour obtenir la grille finale.



FIG(2.12) : Mise en forme des 5 Couches

La couche 2 (C2 la Grille finale) contient $n_v$neurones d'association a_{Zl_iZ2}. Chaque neurone d'association

est connecté à une cellule de détection de chaque tableau d'entrée. La sortie des neurones d'association de la C2 est égal a 1 lorsque toutes les entrées sont différentes de zéro, c'est-á-dire que ces neurones calculent un « AND » logique á partir de leurs entrées (voir figure 35). Les neurones d'association sont ranges de manière arbitraire de telle sorte que seulement $n_a$sont actives par une entrée donnée.

La couche 3 (C3) contient n_y neurones de sortie. La fonction de ces neurones est de calculer l'addition pondérée de tous les poids $Wj,_{Zl,Z2}$pour une sortie *j* de la couche 2.

Admettons (vois figure 33) que pour l'entrée x_1= 0.42 les cellules de détection $z_{1,7}$, $z_{1,8}$, $z_{1,9}$, $z_{1,10}$, $z_{1,11}$sont respectivement activées, et pour l'entrée x_2 = 0.35 les cellules de détection $z_{2,6}z_{2,7}$,$z_{2,8}$,$z_{2,9}$,$z_{2,10}$ sont respectivement activées. La sortie *yj* sera alors égale:

$$y_{j=}a_{7,8}w_{j,7,8} + a_{8,10}w_{j,8,10} + a_{9,7}w_{j,9,7} + a_{10,9}w_{j,10,9} + a_{11,6}w_{j,11,6} \text{.....(2.9)}$$

Le poids $Wj,7,8$connecte la cellule α7,8 au neurone de sortie *nj*. La sortie *yj* sera égale à l'addition de tous les poids connectés au neurone *nj*. .

VI-Propriétés du RN CMAC

VI-1-Capacité d'approximation des fonctions non-linéaires du RN CMAC

La mise en correspondance non-linéaire fixe de CMAC est un des facteurs les plus importants qu'il nous faut considérer. Cette mise en correspondance non-linéaire permet à CMAC de faire l'approximation de fonctions non-linéaires. Cette mise en correspondance étant fixe, le calcul de la sortie de CMAC est donc plus rapide que pour les RN Multicouches avec rétro-propagation du gradient pour lesquels, à chaque sortie, cette mise en correspondance doit être recalculée.

Il faut remarquer que la non-linéarité qui caractérise les réseaux neuronaux se trouve dans la mise en correspondance associative de l'entrée, et non dans la fonction sigmoïde/seuil qui se trouve à la sortie de chaque neurone [9].

Nous supposons que cette mise en correspondance de CMAC pourrait difficilement être trouvée dans un RN Multicouches en utilisant l'algorithme de rétro-propagation du gradient. Cette mise en correspondance est une partie fondamentale dans CMAC.

VI-2-Discontinuité dans la mise en correspondance

La fonction approximée par CMAC peut contenir plusieurs discontinuités dues à la quantification de l'espace d'entrée. Ce fait pourrait montrer que CMAC n'approxime pas de façon correcte les fonctions. Cependant, cette discontinuité ne représente pas de problème pour plusieurs applications. Comme nous l'expliquerons dans le chapitre suivant, si CMAC est utilisé dans l'Apprentissage par Renforcement Connexionniste, cette discontinuité est quelquefois sans importance.

De toute façon, ce problème peut être évité si le concepteur de CMAC choisit une résolution suffisamment grande des variables d'entrée, de manière à réduire ces discontinuités. C'est un dilemme, parce qu'une élévation dans la résolution des variables d'entrée augmentera également le nombre d'adresses virtuelles des poids.

VI-3-Généralisation locale

La capacité de généralisation des réseaux neuronaux consiste à prendre en compte un nouveau cas en utilisant un cas voisin connu. Au contraire des RN Multicouches avec rétro-propagation du gradient, CMAC possède la capacité de généralisation locale. Ceci signifie que

lorsqu'un nouveau vecteur d'entrée est présenté au CMAC, seule une partie de la mise en correspondance est ajustée. Ceci est dû au fait que dans CMAC seulement les poids qui interviennent pour le calcul de la sortie sont modifiés.

Pour illustrer l'influence du paramètre de généralisation ρ sur la capacité du CMAC à approximer des fonctions, prenons $f(x)$ définie comme suit :

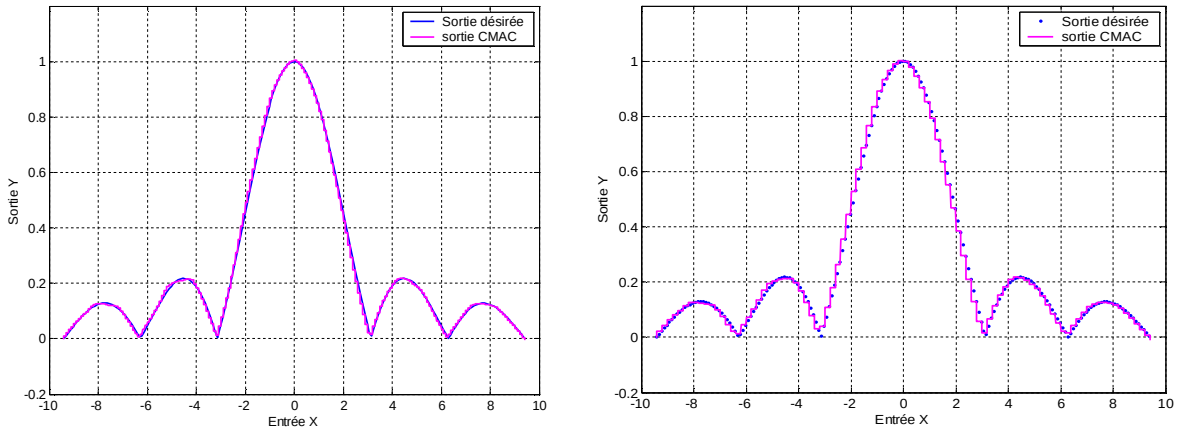
$$f(x) = \left|\frac{sin(x)}{x}\right| \qquad (2.11)$$

où $x \in [-3\pi, 3\pi]$ et $|.|$ désigne la valeur absolue.

Pour approximer cette fonction, nous avons utilisé un CMAC à deux dimensions dont le nombre de nœuds $r_i = 512$ ce qui nous permet de définir une grille de normalisation de l'espace d'entrées avec des intervalles réguliers. Nous fixons aussi le coefficient d'apprentissage $\beta = 0.7$.

Pour visualiser l'influence de ρ sur la généralisation locale du CMAC, nous avons pris deux valeurs qui nous permettent d'obtenir des différences significatives pouvant être visualisées sans ambiguïté.

A partir de la figure 3.13, qui représente l'allure de la fonction $f(x)$ ainsi que la réponse du CMAC pour les deux valeurs du paramètre de généralisation (ρ = 2, et ρ = 4), nous pouvons déduire que l'augmentation de ρ à pour effet de réduire la capacité du CMAC à modéliser et surtout à généraliser convenablement les informations qu'il a acquises au fur est à mesure qu'on lui présente des exemples d'apprentissage.

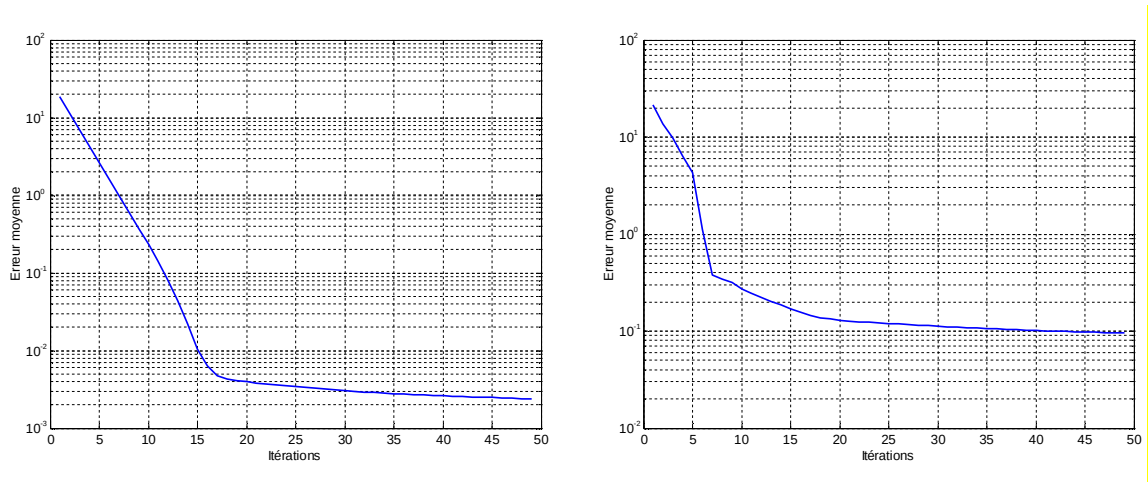


ρ = 2 ρ = 4

Figure 2.13: La réponse du CMAC pour deux valeurs du paramètre de généralisation.

La diminution des performances du CMAC s'explique du fait qu'en augmentant le paramètre de généralisation les risque d'interférences au cours de l'apprentissage augmente également. De plus, cette altération

des capacités du réseau à approximer s'accentue encore plus dès qu'il s'agit d'approximer des fonctions multivariable.



$\rho = 2$ $\rho = 4$

Figure 2.14 : L'influence du paramètre de généralisation sur l'apprentissage du réseau.

La différence de rapidité de l'évolution de l'erreur moyenne, au cours des premiers cycles d'apprentissage réside dans le fait qu'en augmentant le paramètre de généralisation ρ, l'algorithme d'apprentissage doit adapter moins de poids pour calculer la sortie du CMAC. Cependant, au bout de 30 itérations, ce nombre relativement faible de poids et la probabilité plus importante qu'il y ait interférence au cours de l'apprentissage ne permettent pas au CMAC d'approximer $f(x)$ avec la précision que l'on désire.

En résumé, le choix du paramètre de généralisation est très important. Il permet, non seulement de définir la forme, la largeur et le nombre de champs réceptifs qui rentrent en jeu dans le calcul de la réponse du réseau (ce qui définira la capacité du réseau à modéliser et à généraliser les informations acquises, au cours de l'apprentissage), mais aussi sa taille et donc l'espace mémoire (ressources informatique) qu'il va utiliser. De ce fait, le concepteur du CMAC doit être attentif au fait qu'il est impératif de trouver le bon compromis entre le souhait d'obtenir de bonnes performances et les ressources informatiques dont il dispose.

VI-4-Facilité pour l'implantation en matériel ("hardware conventionnel")

L'implantation de CMAC en matériel ("hardware conventionnel") est plus facile que pour d'autres approximateurs de fonction grâce à son fonctionnement sous forme de table de correspondance. La considération de la facilité d'implantation en matériel ("hardware") est très importante si nous voulons utiliser CMAC dans des problèmes du monde réel, par exemple, l'implantation dans un routeur d'un réseau de télécommunication. D'autre part, la rapidité du calcul de la sortie de CMAC est aussi un avantage que nous pouvons obtenir dans l'implantation en matériel. On peut trouver un exemple de l'implantation en matériel de CMAC dans [15].

I. CMAC comme approximateur de fonction

Ci-dessous, nous montrons les résultats en utilisant l'implantation de CMAC proposé dans ce chapitre dans le cadre de l'Apprentissage Supervisé. La liste des « fonctions à apprendre » que nous avons implantées sont :

- Fonction : F(x)=sin(x) .
- Fonction : F(x)= sinc(x)
- Fonction gaussienne 2d

Voici les résultats obtenus pour les fonctions *fsinc*(voir la figure 2.15), *ffonctiongausienne 2D* voir la figure 2.16), *fsin*(voir la figure 2.17), Nous présentons : à gauche la courbe obtenue en rouge avec la courbe références en bleu.

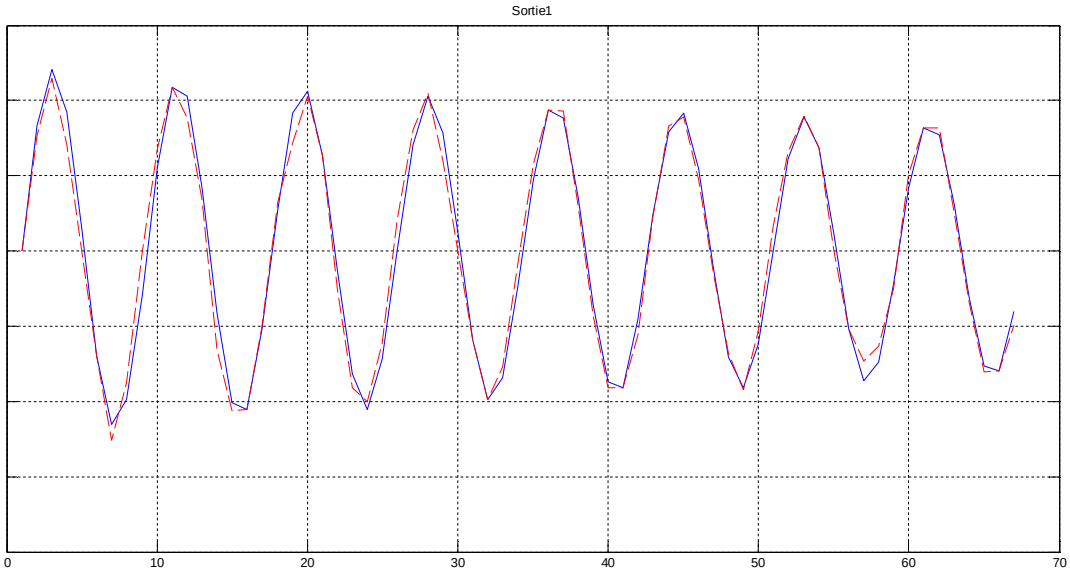


FIGURE 2.18 Fonction SINC

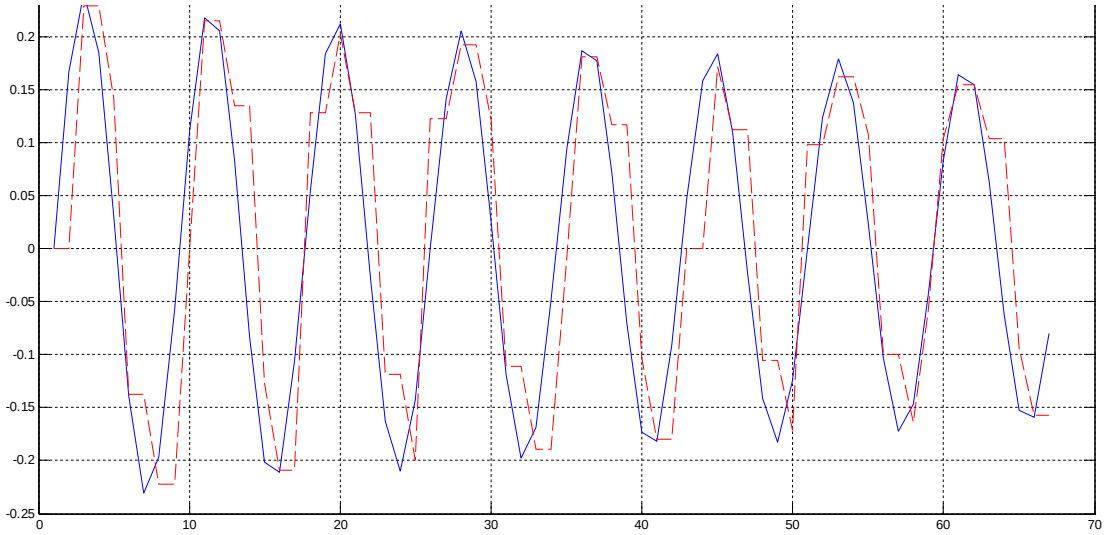


FIGURE 2.19 Fonction SINC

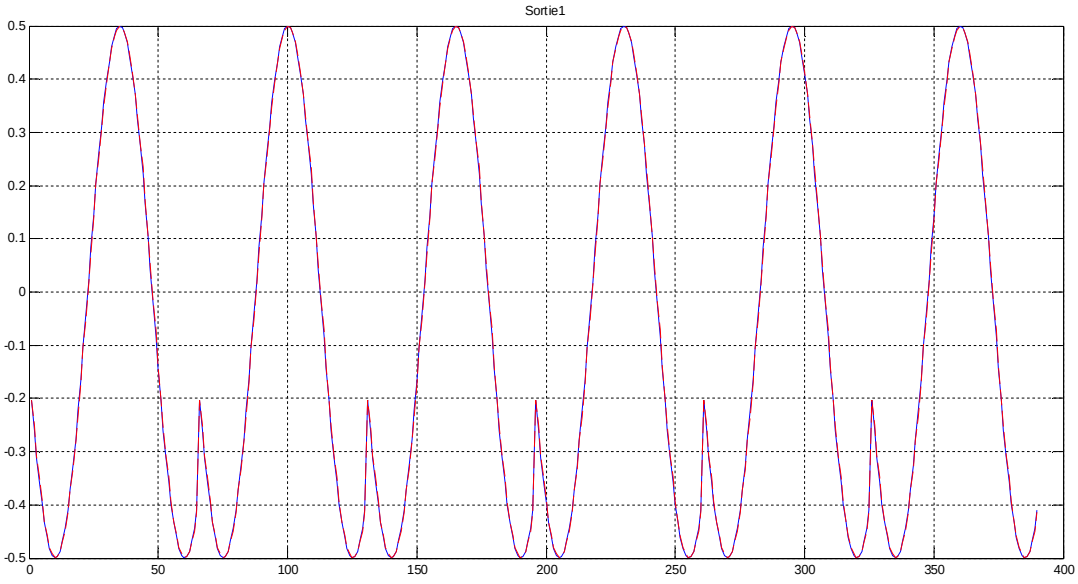


FIGURE 2.19 Fonction gaussienne 2d

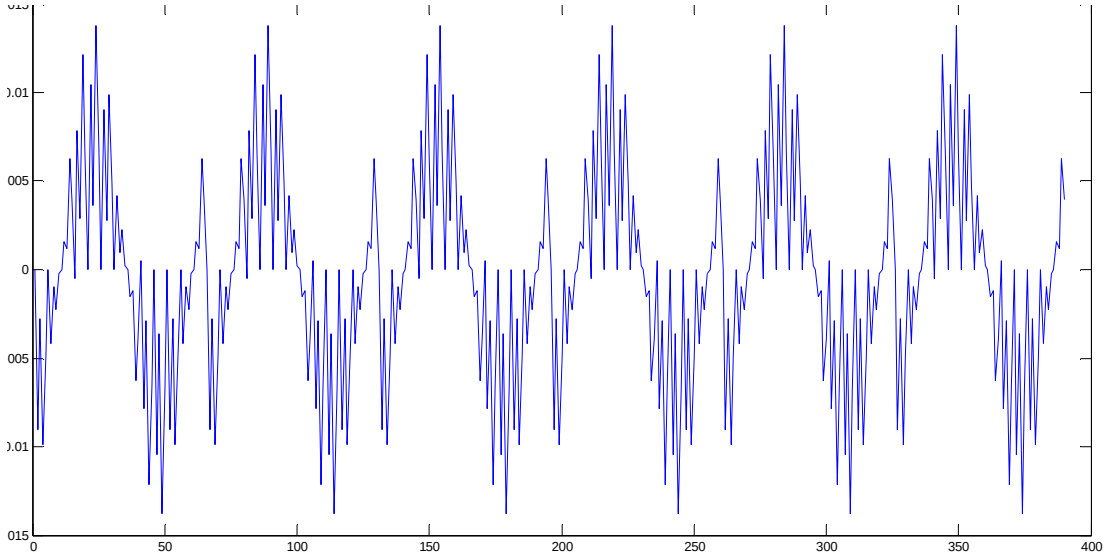


FIGURE 2.20distribution de l'erreur sur la fonction gaussienne

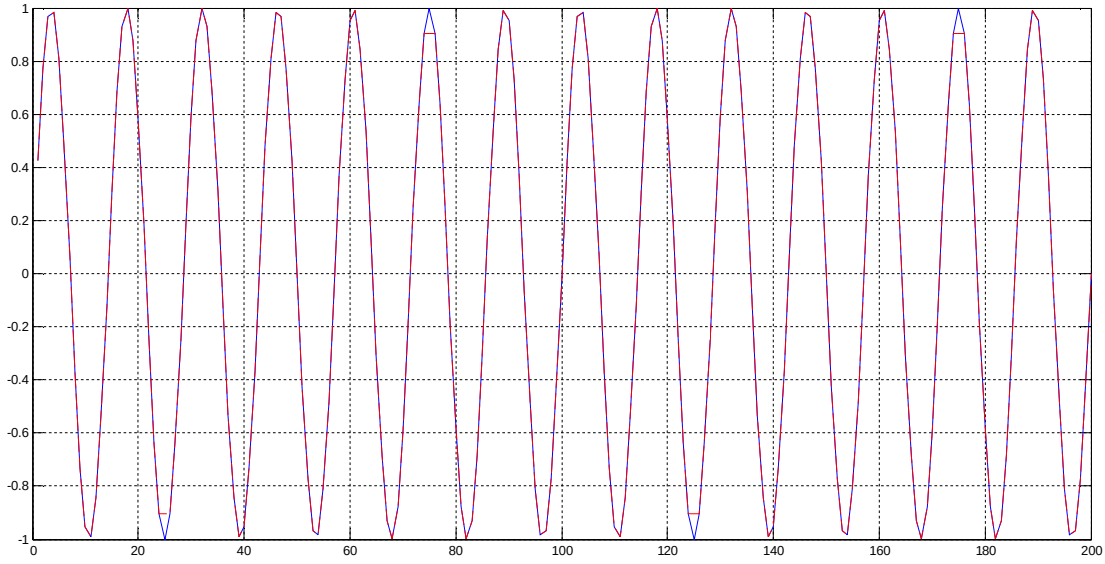


FIGURE 2.21Restitition de la fonction fsin

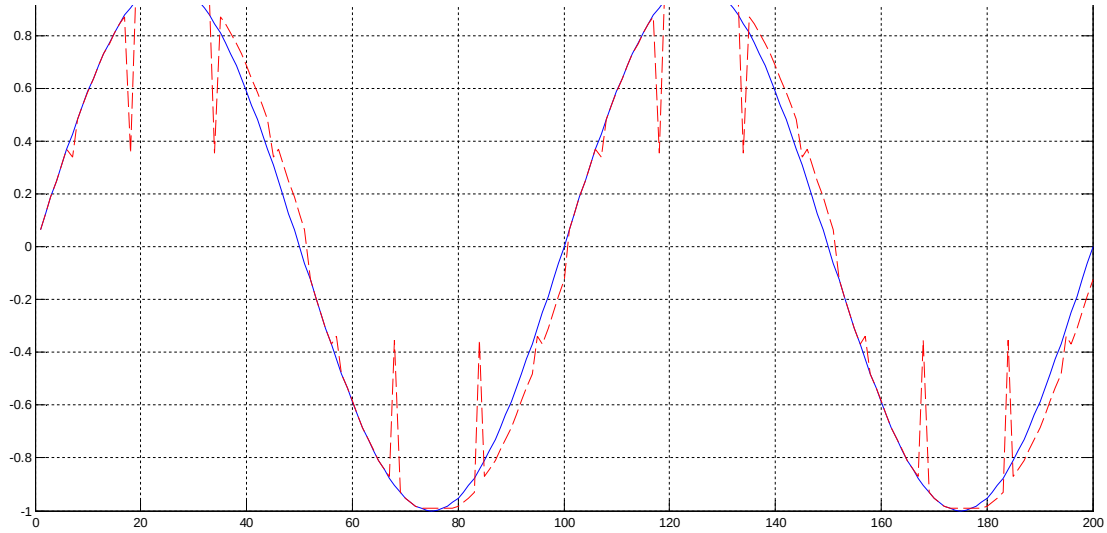
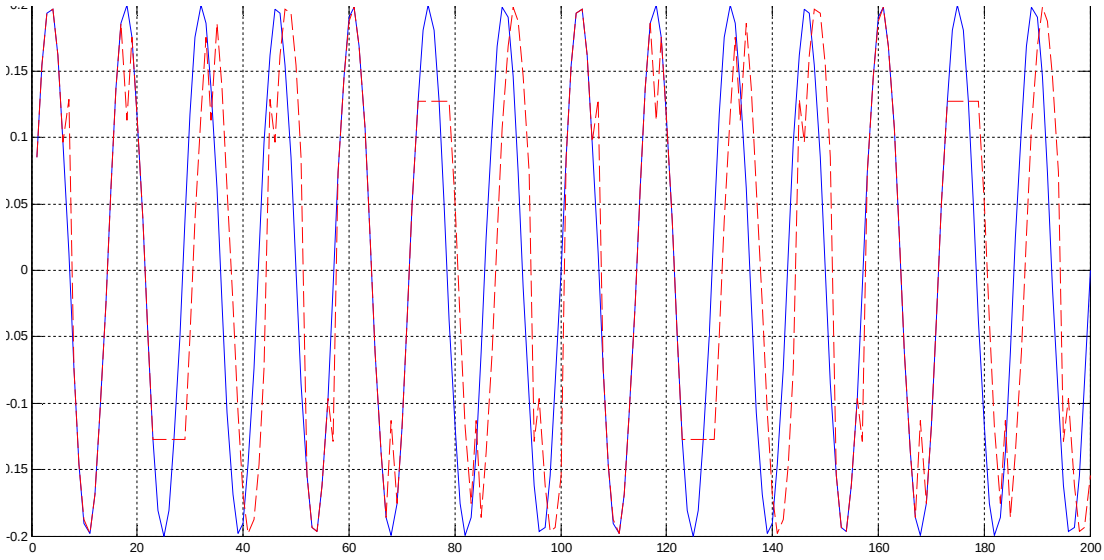
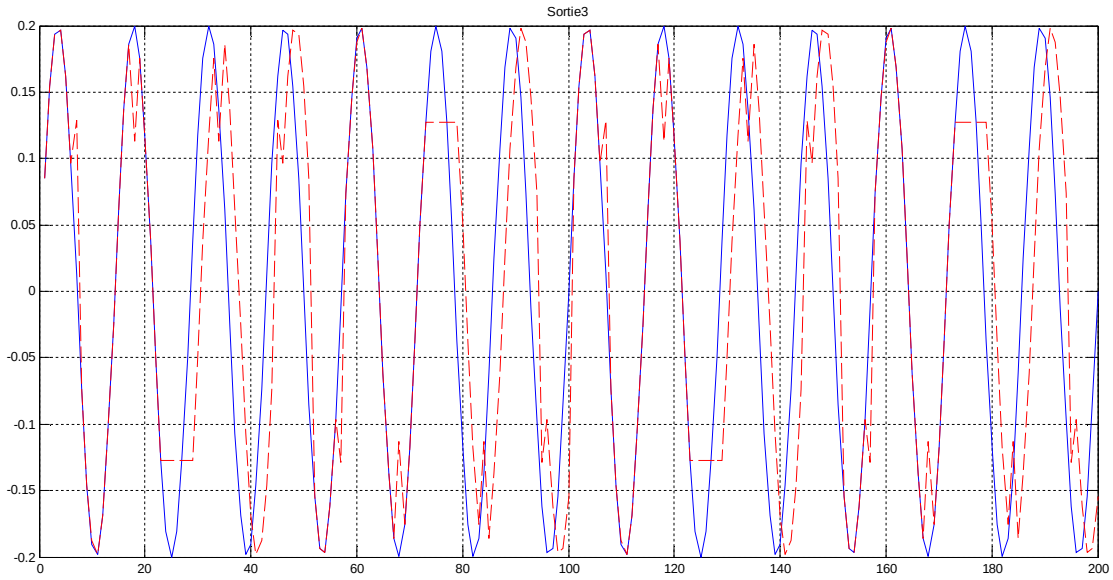
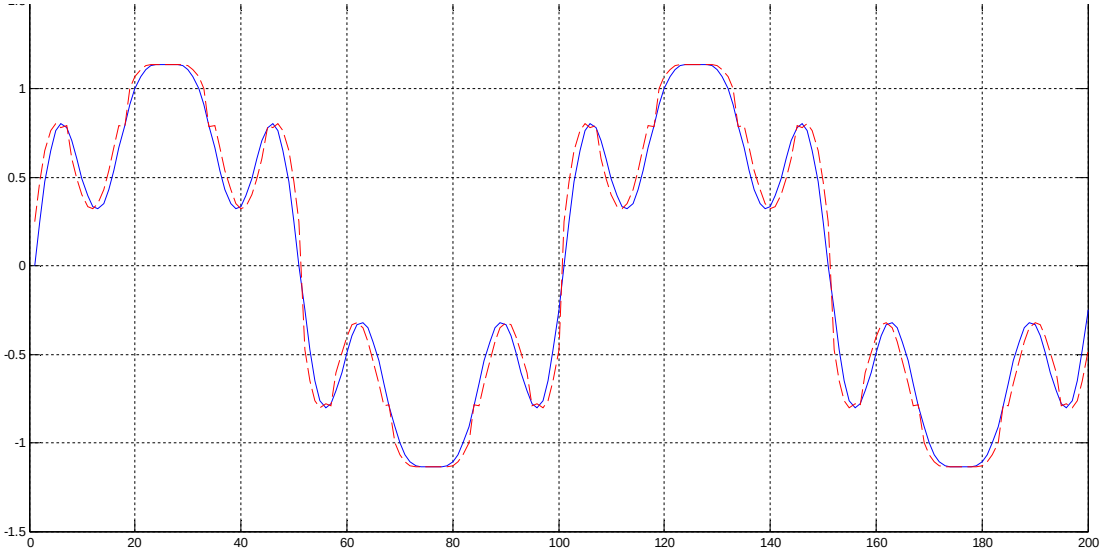


FIGURE 2.22 décomposition en série de Fourier







Le comportement du CMAC semble globalement bon au centre de l'espace des entrées, par contre, des problèmes se posent sur les bords. Ceci provient du manque de points d'apprentissages au niveau des bords et au principe des « décalages ». Nous pouvons aussi constater que CMAC est plus performant sur des fonctions continues (comme fsin) que sur des fonctions présentant des discontinuités. Ce dernier point est directement dû à la généralisation.

I. Conclusion et Discussion

Nous avons présenté le fonctionnement du cervelet du point de vue biologique. Nous avons fait le lien entre ce modèle biologique et le modèle mathématique développé par James Albus, nommé CMAC. Nous avons également présenté l'algorithme utilise pour l'implantation du RN CMAC. Cette implantation a été faite sous Matlab.

Les principaux avantages de CMAC sont: d'une part, la rapidité du calcul des sorties, et d'autre part, la rapidité de l'algorithme d'apprentissage. En outre, la surface d'erreur de CMAC est du type quadratique avec un seul minimum global. Ces avantages font de CMAC un RN qui peut être utilise pour l'apprentissage en ligne et incrémentiel. Cependant, l'utilisation de CMAC a besoin du réglage de plusieurs paramètres et d'une utilisation de mémoire supérieure á celle utilisée dans les RN Multicouches. .

Dans le chapitre suivant, nous allons tester la possibilité d'implanté le CMAC dans un robot mobile et de voir si le CMAC sera capable de reproduire cette capacité d'approximation pou une fonction réelle tiré de ce robot.

CHAPITRE III :

implémentation de la commande neuronale sur la plate-forme de robot mobile

Introduction

« L'être humain croira toujours que plus le robot paraît humain, plus il est avancé, complexe et intelligent. »

Paroles du personnage de Elijah Baley dans « *Les robots de l'aube* »
Isaac Asimov, J'ai lu, 1988 (tome 2, page 245)

Le robot est une machine qui est programmée pour effectuer des tâches. Cette approche présente plusieurs problèmes difficiles à surmonter. Premièrement, les capacités du robot sont limitées par l'agilité des programmateurs à lui inculquer la façon de réaliser les tâches désirées .Deuxièmement, un tel robot ne peut faire preuve d'"imagination", sauf si elle lui a été programmée (elle est donc limitée) .En donnant la liberté au robot de se créer sa propre représentation du monde, on lui laisse la possibilité de développer à la fois des capacités inattendues et des réactions non programmées

L'une des alternatives est l'utilisation des réseaux de neurones .Mais cette approche une puissance de calcul importante, d'autant que la tâche est complexe. De ce fait, l'utilisation des réseaux de neurones dans la commande robotique est de plus en plus répandue puisqu' on dispose, maintenant, de calculateurs très performants en termes de puissance et de mémoire.

Afin de mettre en œuvre notre réseau de neurones CMAC décrit lors du précédent chapitre, nous nous sommes heurtés à des problèmes de taille :

1. Puissance de calcul : vu le nombre important de calcul a effectué il est donc impossible de commandé le robot en temps réelles car le processeur qui equipe notre robot(atmega 128) n'est pas capable d'assurer la gestion de ce flux important de données.

2. Taille mémoire insuffisante pour stocker les poids du réseau.

De plus ,l'imprécision des capteurs infra-rouge du eaux phénomènes d'éblouissement nous a amené à développer une carte électronique. Celle-ci sera dotée de nouveaux capteurs, de deux autres microcontrôleurs et de circuits de communications. Avec cette nouvelle configuration, l'électronique embarquée sur le robot mobile pour aisément gérer le nombre important de données et de calculs nécessaires à la réalisation d'applications en temps réel.

I- Description du robot :



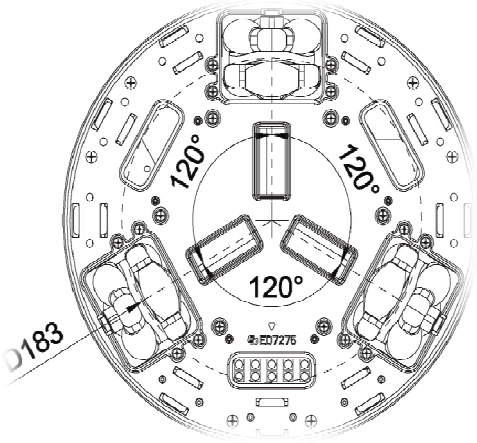
Figue 3-1 : le Robot

Notre robot, est un robot mobile à roues omnidirectionnel à trois degrés de liberté réalisé en 2008 ayant pour simple fonction d'éviter les différents obstacles qui peuvent se présenter sur son chemin. Pour ce faire, le robot est doté de trois moteurs lui permettant de se mouvoir, d'une structure mécanique qui lui sert de support, de dispositifs de perception et bien évidement d'une carte de contrôle réalisée à base d'un microcontrôleur Atmega128.

I-1 Description mécanique :

a) Type de plateforme :

La plateforme utiliser est une plateforme omnidirectionnel circulaire de rayon d'environs 255mm soutenue par trois ensembles de roues, chacune et formant deux à deux un angle de 120º. De plus la stabilité du système est assurée par cette disposition symétrique.la distance entre le centre de chaque des trois roue et le centre de plateforme est d'environs 91.5mm



Figue 3-2 : forme de plateforme d'ED7275

b) Matériaux de la plateforme:

La configuration électromécanique des matériaux qui compose la base, dépend de la stabilité de cette dernière et sa résistance aux chocs et aux sollicitations diverses

L'utilisation d'un alliage à base d'aluminium constitue un choix évident étant donné les propriétés mécaniques et les principales caractéristiques appréciables de ce type d'alliage :

- légèreté: grâce à la faible masse volumique de l'aluminium (2700kg/m3, soit près de 3 fois inférieure à celle des alliages d'acier), nous pouvons espérer un poids total faible pour la structure.
- ductilité: cette caractéristique facilite l'usinage des alliages d'aluminium à l'état solide. Il devient donc relativement facile de mettre en forme et de fabriquer les différentes pièces de la structure mécanique.
- possibilité de traitements (écrouissage, addition d'éléments d'alliage et traitement thermique): ces traitements permettent de modifier de façon appréciable les propriétés mécaniques initiales de l'aluminium. Il est donc possible, selon l'application donnée, d'améliorer certaines propriétés cruciales.
- résistance à la corrosion: évidemment, il est peu souhaitable que notre structure soit sensible à la corrosion et que son état se détériore avec le temps.
- conductibilité électrique: cette propriété est particulièrement intéressante au niveau de la conception du système électrique, puisque la structure peut être utilisée comme la mise à la terre du système.

c) Les roues :

Etant le seul contact avec le sol, Les roues (figure 3-3) sont des roues suédoises. Elles sont en fait composées d'un assemblage de deux roues omnidirectionnelles, dont l'une est déphasée d'un angle de 60° par rapport à l'autre pour leurs permettre des déplacements en deux directions :

- Une dans la direction perpendiculaire à l'axe du moteur (forcée).
- La seconde dans la même direction que l'axe du moteur (libre).



Figue3-3 : Roue du robot

Le matériau utilisé est un polyuréthane moins dur que celui utilisé pour les rollers, il permet d'avoir une traction améliorée sur les surfaces lisses. Le diamètre de la roue est de 6cm. De plus, la charge maximale qu'elle peut supporter est de 222.5kgf. Elles offrent une performance qui donne:

- Une bonne adhérence au sol pour éviter les glissements.
- Un contact permanent au sol quel que soit la position de la roue pour avoir le meilleur équilibre possible.
- Un contact réduit avec le sol pour réduire autant que faire se peut la force de friction avec le sol.
- Une masse réduite pour pouvoir négliger l'inertie de la roue.

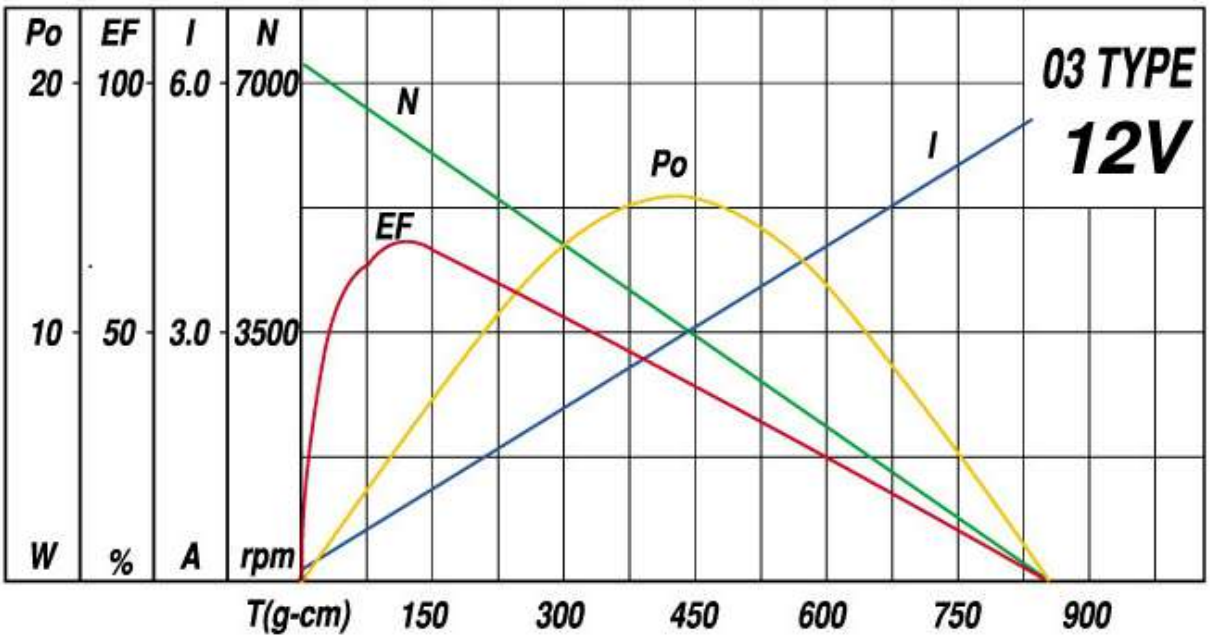
d) Le bloc moteur:

Les moteurs utilisé DC modèle IG-32RGM. Les caractéristiques attrayantes de ce moteur, sont d'abord son couple de démarrage de 1.07Nm (nous pouvons espérer une excellente accélération avec un réducteur), couple continu de 110mN.cm ainsi qu'une vitesse à vide qui s'élève à 7300rpm.

Ce moteur est couplé à un encodeur quadratique et un réducteur.

Paramètre	Symbole	Valeur
Vitesse à vide	n0	7300rpm
Tension nominale	V	12V
Couple	M H	110 mN.cm
Courant à vide	I0	153mA
Vitesse limite	Nmax	7300rpm
Humidité	H	20-80
Température	T	-10 à +60deg
Constante de couple	Cm	0.0140Nm
Constante de vitesse	Cr	0.0156Nm
Résolution encodeur	R	26puls/Toure
Rapport de réduction	I	1/27

Tableau 3-1 : Caractéristiques des moteurs IG-32RGM



Figuer3-4: Caractéristiques des moteurs IG-32RGM

I-2 Modélisation du robot ED-7275:

Afin de simplifier la modélisation du robot on confdérera les hypothèses suivantes :

- glissement latéral négligé: nous considérons que le glissement latéral est impossible. Cela permet de Simplifier le modèle puisque nous n'avons pas à considérer la force Centrifuge et l'adhérence du robot face à des forces latérales.
- Centre de masse au milieu de l'axe des roues motrices: cette hypothèse qui est d'ailleurs valide vue la configuration du robot, permet de modéliser le robot Comme un disque, dont le centre est le centre de l'axe et la masse est Uniformément répartie sur la surface du disque. Cela permet de Simplifier le modèle de l'inertie de rotation du Robot puisqu'elle devient simplement l'inertie d'un disque
- Régime transitoire des moteurs négligé: le circuit équivalent des moteurs DC possède une constante de temps très faible comparativement à la constante de temps mécanique. Nous pouvons donc négliger le régime transitoire électrique.

-2 -1 Modélisation cinématique :

soit $R_0\{o_0,x_0,y_0\}$ le repère fixe et R{o,x,y} le repère locale liéau robot.

Le vecteur d'état du robot est défini par $[x \quad y \quad \theta]^{T,}$ avec :

$[x \quad y]^T$: la position de centre de gravité du robot G et θ L'orientation du repère R, tous deux par rapport au repère R_0.

Dans cette Modélisation on va calculer la vitesse du centre de gravité $G/R_{0,}$ telle que : La vitesse de G = la somme des vitesses des différentes roues motrices.

Le but est de déterminer la vitesse de chaque roue motrice, La figure (3-5) montre les déférentes vitesses des roue avec :

δ : l'angle d'orientation de moteur

Ø : angle de rotation de la plateforme du robot par rapport à repère absolu $R_{0.}$

L : le rayon de plateforme

R : le rayon d'une roue

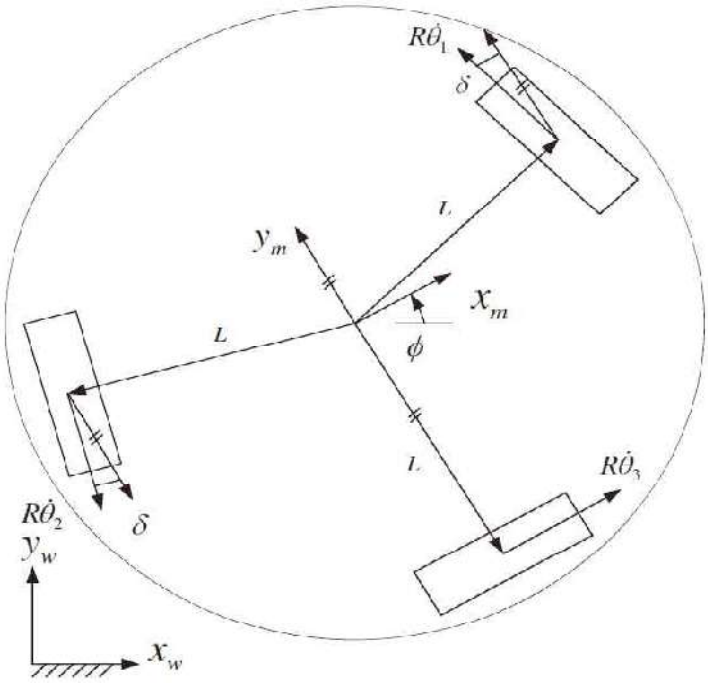


Figure 3-4 : représentation de modèle cinématique

Afin de déterminer le modèle cinématique de notre plateforme nous calculons les vitesses des roue par rapporte au repère absolu (R_0).

$\vec{v}$ (roue/R_0) = $\overrightarrow{vt}$ (roue/R_0) + $\overrightarrow{vr}$ (roue/R_0) (1)

Avec :

$\vec{v}$: vitesse total de la roue

$\overrightarrow{vt}$: vitesse de translation de la roue

$\overrightarrow{vr}$: vitesse de rotation de la roue

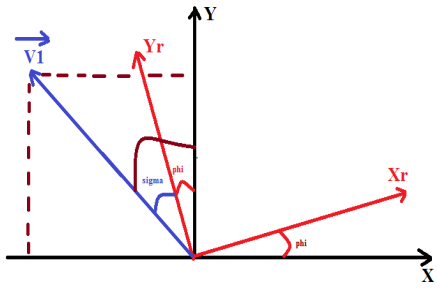
Si on note v_i (i varie de 1 à 3) la vitesse linéaire des roues,$\dot{Ø}$ vitesse angulaire, la relation (1) devient :

$$vi = vx(i) + vy(i) + L(i)\dot{Ø}(i) \qquad (2)$$

les deux figures suivant montre les relation entre les vecteurs vitesse(en bleu) des moteur et les vecteurs de repère fixe(en noire)

Ø: est angle d'orientations de repère liée ou robot (en rouge) et le repère fixe,

δ et l'angle entre Y_r et le vitesse



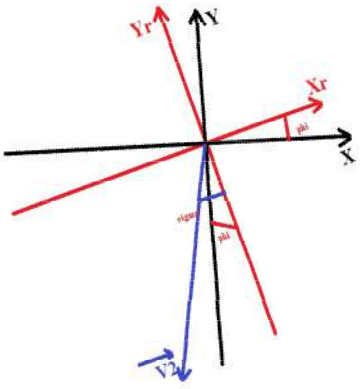
Pour la vitesse de translation de premier moteur $v1t$ on a :

$$v1t = -\dot{x}\sin(\delta + Ø) + \dot{y}\cos(\delta + Ø)$$

de même en déduire $v2t$:

$$v2t = -\dot{x}\sin(\delta - Ø) - \dot{y}\cos(\delta - Ø)$$

Pour la troisième vitesse on a $\delta = 0$ voire figure(3-4)



Donc $v3 = \dot{x}\cos(Ø) + \dot{y}\sin(Ø)$

la relation (2) peut s'écrire pour chacune des trois roues :

$$v1 = -\dot{x}\sin(\delta + Ø) + \dot{y}\cos(\delta + Ø) + l\dot{Ø} \quad (3)$$

$$v2 = -\dot{x}\sin(\delta - Ø) - \dot{y}\cos(\delta - Ø) + l\dot{Ø} \quad (4)$$

$$v3 = \dot{x}\cos(Ø) + \dot{y}\sin(Ø) + l\dot{Ø} \quad (5)$$

D'autre part on a

$$vi = R * \dot{\theta}(i)$$

Le modèle cinématique de notre ROBOT est représentée par les trois équations suivantes :

$$\begin{bmatrix}\dot{\theta}1\\ \dot{\theta}2\\ \dot{\theta}3\end{bmatrix} = 1/R * \begin{bmatrix}-\sin(\delta + Ø) & \cos(\delta + Ø) & l\\ -\sin(\delta - Ø) & -\cos(\delta - Ø) & l\\ \cos(Ø) & \sin(Ø) & l\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{Ø}\end{bmatrix} \quad (7)$$

I-2-1- MODELISATION DYNAMIQUE :

Dans la modélisation dynamique on va déduire la relation entre la masse de robot et son accélération.

a) Principes fondamentaux de la dynamique :

- ***Premier principe*** (1ére lois de NEWTON) :

$$\sum_i^n fext = m * a \quad (8)$$

- **Deuxième principe** (2ére lois de NEWTON) :

$$\sum_i^n M/fext = j * \Omega \quad (9)$$

b)*Application du formalisme au robot :*

Soit : -

m : La masse du robot.

J : Le moment d'inertie par rapport à l'axe vertical.

fi : les force d'entraînement des moteurs (i = 1 :3)

a : L'accélération linéaire du robot

Ω : L'accélération angulaire du robot

En appliquant les lois fondamentales de la dynamique on aura le système suivant :

$$\sum_i^3 fext = m * a$$

$$\sum_i^3 M/fext = j * \Omega$$

Donc on a les deux relations suivantes :

$$m * \vec{a} = \overrightarrow{f1} + \overrightarrow{f2} + \overrightarrow{f3} \quad (10)$$

$$j * \ddot{\vec{\theta}} = \overrightarrow{Mf1} + \overrightarrow{Mf2} + \overrightarrow{Mf3} \quad (11)$$

La projection des forces sur le repère fixe [o, x, y] nous donne :

$$f1x = -f1 * \sin(\delta + Ø)$$

$$f1y = f1 * \cos(\delta + Ø)$$

$$f2x = -f2 * \sin(\delta - Ø)$$

$$f2y = -f2 * \cos(\delta - Ø)$$

$$f3x = f3 * \cos(Ø)$$

$$f3y = f3 * \sin(Ø)$$

L'accélération suivant les deux axe et l'accélération angulaire s'écrits comme suit :

$$m * \ddot{x} = -f1 * \sin(\delta + \emptyset) - f2 * \sin(\delta - \emptyset) + f3 * \cos(\emptyset) \quad (12)$$

$$m * \ddot{y} = f1 * \cos(\delta + \emptyset) - f2 * \cos(\delta - \emptyset) + f3 * \sin(\emptyset) \quad (13)$$

$$j * \ddot{\Theta} = l * (f1 + f2 + f3) \quad (14)$$

Alors on obtient le modèle dynamique suivant :

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & j \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\Theta} \end{bmatrix} = \begin{bmatrix} -\sin(\delta + \emptyset) & -\sin(\delta - \emptyset) & \cos(\emptyset) \\ \cos(\delta + \emptyset) & -\cos(\delta - \emptyset) & \sin(\emptyset) \\ l & l & l \end{bmatrix} \begin{bmatrix} f1 \\ f2 \\ f3 \end{bmatrix} \quad (15)$$

Il nous reste à déterminer les forces d'entrainement (fi) des actionneurs :

b)Modélisation des moteurs électriques DC :

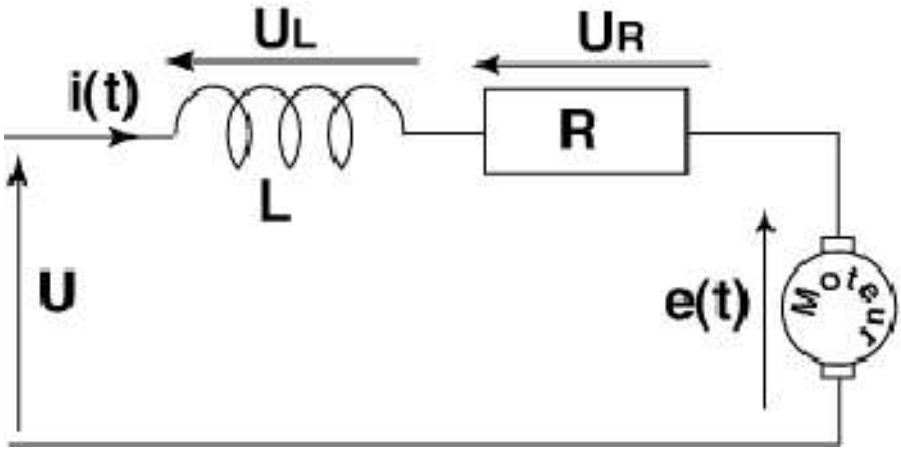


Figure 3-5 – Modèle électrique du moteur à courant continu

Le modèle des moteurs à courant continu est donné par les équations suivantes :

$$U(t) = L\frac{di(t)}{dt} + Ri(t) + E(t) \quad (16)$$

$$Cm(t) = Ki(t) \quad (17)$$

$$E(t) = Kc\omega(t) \quad (18)$$

$$Jeq\frac{d\omega(t)}{dt} = Cm(t) + Cr(t) \quad (19)$$

L'équation (16) correspond au modèle électrique du moteur à courant continu comportant, en série, une résistance R, une bobine d'inductance L et la force électromotrice e(t), qui comme son nom ne l'indique pas est une tension. $U(t)$ est la tension d'alimentation du moteur. Le terme $E(t)$ est la tension aux bornes de la bobine. Le terme $Ri(t)$ correspond à la tension aux bornes de la résistance (loi d'Ohm).

L'équation (17) est une équation de couplage électromécanique. Le couple $Cm(t)$ (action mécanique ayant tendance à créer une rotation, un couple s'exprime en Newton mètre -Nm) fournit par le moteur électrique est proportionnel au courant (en ampère) qui traverse le moteur. La constante de proportionnalité est la constante de couple (en Nm/A).

L'équation (18) est une seconde équation de couplage électromécanique. La force électromotrice (en Volt) est proportionnelle à la vitesse de rotation du moteur $\omega(t)$ (qui

s'exprime en radian/seconde ou en tour/minute). La constante de proportionnalité est la constante de vitesse (en tours/min par V ou en rad/s par V).

L'équation (19) est une équation mécanique issue du principe fondamental de la dynamique (au programme de spé). Jeq est l'inertie équivalente ramenée à l'arbre moteur et représentela difficulté à mettre en rotation le moteur.

$\frac{d\omega(t)}{dt}$ Représente l'accélération angulaire. Elle dépend du couple moteur $Cm(t)$ et du couple résistant $Cr(t)$. Le couple résistant représente une action mécanique extérieure au moteur qui rend difficile sa mise en rotation. Cette action peut être due au frottement, la pesanteur, etc…

Des quatre équations précédentes on obtient la relation entre la tension $U(t)$ et le couple moteur $\omega(p)$:

L'équation (16) nous donne :

$$U(p) = Lpi(p) + Ri(p) + E(p)$$

Donc $$i(p) = \frac{1}{Lp+R}[U(p) - E(p)] \quad (20)$$

L'équation (19) nous donne :

$$\omega(p) = \frac{1}{Jp}[Cm(p) + Cr(p)] \quad (21)$$

Avec : $Cm(p) = Ki(p)$ (22) et $E(p) = Kc\omega(p)$ (23)

Des équations (20), (22) et (23) on obtient la fonction de transfère de $\frac{U(p)}{\omega(p)}$

Le schéma bloc suivant représente la fonction de transfère $\frac{U(p)}{\omega(p)}$

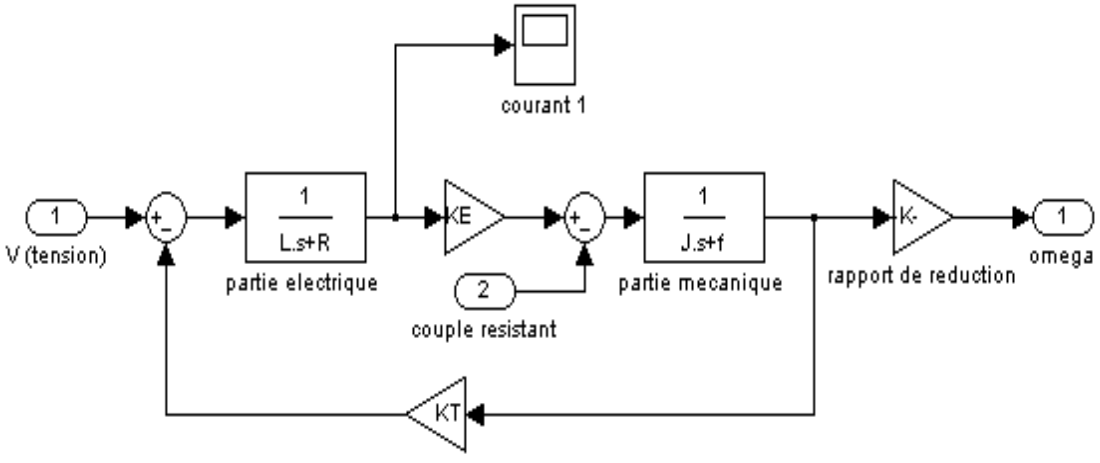


Figure3-6 : Modèle en simulation du Moteur à Courant Continu

b) Transformations dynamiques :

Soit $cm1$,$cm2$ et $cm3$ les couples développés par les trois moteurs

La relation entre le couple moteur et la force d'entraînement est :

$$fi = 1/R * (cmi) \quad (24)$$

Pour les trois moteurs :

$$F = 1/R * (cm1 + cm2 + cm3) \quad (25)$$

L'équation (15) devient comme suit et represente le modèle dynamique du robot:

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & j \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = 1/R \begin{bmatrix} -\sin(\delta + \emptyset) & -\sin(\delta - \emptyset) & \cos(\emptyset) \\ \cos(\delta + \emptyset) & -\cos(\delta - \emptyset) & \sin(\emptyset) \\ l & l & l \end{bmatrix} \begin{bmatrix} cm1 \\ cm2 \\ cm3 \end{bmatrix} \quad (26)$$

I-3 Description électronique :

L'architecture du robot ED-7275est basée sur le microcontrôleur atmega128.

Son autonomie : 80min grâce à une batterie de lithium polymère.

Le ED-7275 possède plusieurs interfaces de communication avec le monde extérieur :

- un afficheur GLCD bleu de 2 lignes et 20 caractères ;
- 4 butons et 4 LED ;
- une porte de 8pin pour des entrer analogique et une autres porte entrée/sortie digitale de 8pin

Le ED-7275, possèdedes capteurs proprioceptifs :

- 5capteurs infrarouges logés en dessous du robot ;

- 3 capteurs de distances à l'avant.

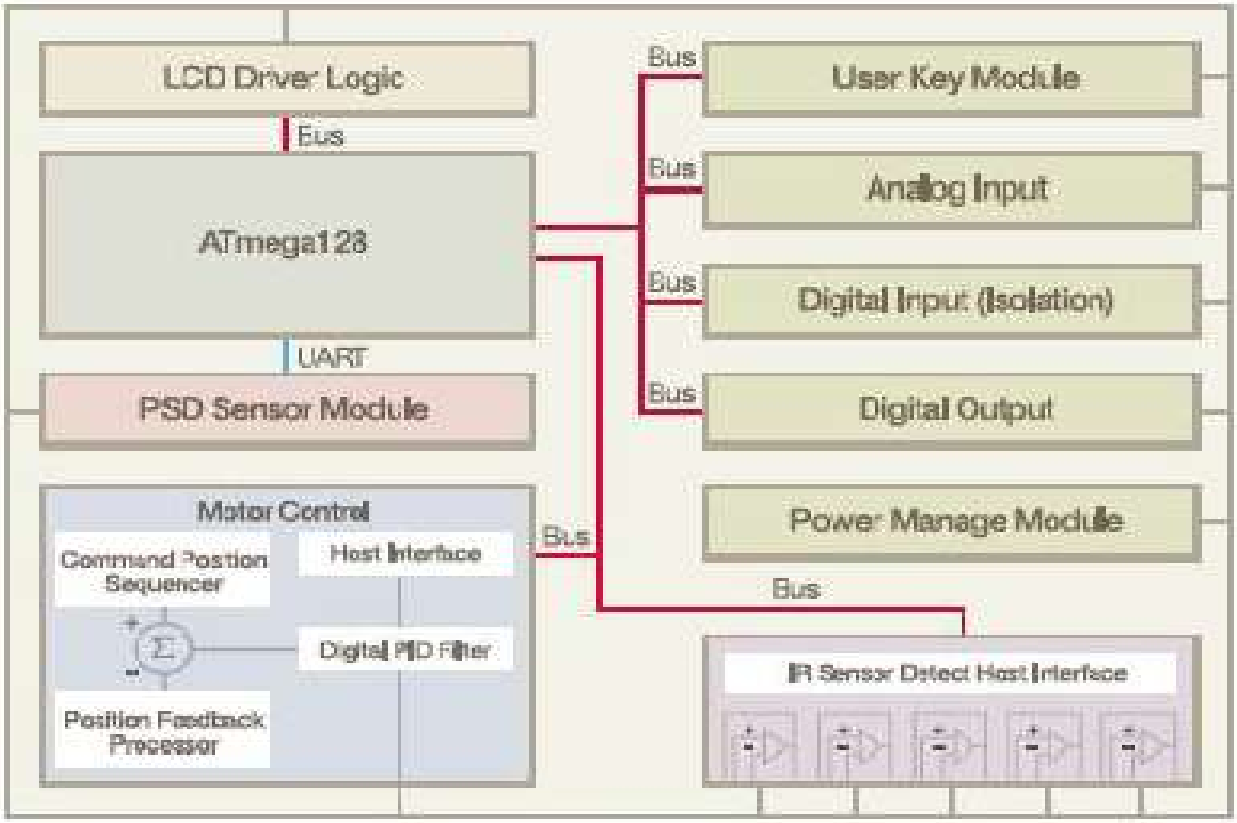


Figure 3-7 :l'architecteur générale de robot ED7275

I-3-1 Caractéristique du microcontrôleur ATmega128

Qu'est-ce qu'un microcontrôleur?

Un microcontrôleur est composé d'un microprocesseur, de mémoire ROM pour stocker le programme et d'une mémoire RAM volatile pour conserver les résultats intermédiaires lors de l'exécution du programme.

Mais les microcontrôleurspossèdent des avantages incomparables pour l'informatique embarquée par rapport aux microprocesseurs. Ce sont d'abord les broches qui peuvent servirent d'entrées/sorties mais surtout toutes les fonctions périphériques. En effet, selon le microcontrôleurconsidéré, on peut trouver :

- plusieurs Timers/Counters

- Génération de signaux PWM

- Nombreux convertisseurs analogiques/numériques
- Communication série USART
- Comparateur analogique
- Watchdog Time
- Oscillateur RC interne
- Modes d'économies d'énergie
- Interruptions internes et externes

Les entrées/sorties d'un microcontrôleur

La plupart des pattes d'un microcontrôleur peuvent servir soit d'entrées, soit de sorties, et ce choix peut changer à tout instant comme le veut le développeur.

Les entrées/sorties sont regroupées par huit et le groupement de huit pattes ainsi formé est appelé *port* et se voit attribué une lettre. Par exemple, les broches 44 a 51 forment le port*A*.

Chaque patte est ensuite numérotée selon sa position au sein de ce port. Par exemple, la broche 46 est appelée *PAÚ*[2]. Trois registres servent à gérer les entrées/sorties :

- DRx : indique quelles sont les pattes considérées comme des entrées et celles considérées comme des sorties. Si le n^{ieme} bit de DDRx (noté DDxn) est à 1, alors la n^{ieme}patte du port x (notée Pxn) est considéré comme une sortie. Si DDxn est à 0, alors Pxn est considérée comme une entrée.
- PORTx : la valeur du n^{ieme}bit de PORTx (noté PORTxn) définit la valeur de sortie de Pxn. Si PORTxn=l, alors Pxn=+Vcc, si PORTxn=0, alors Pxn=Gnd.

- PINx : la valeur du nieme bit de PINx (noté PINxn) est la valeur qui est appliquée à l'entrée du microcontrôleur. Par exemple, si on applique +Vcc a Pxn, alors PINxn=l.

La plupart des pattes possèdent des fonctions périphériques qui inhibent l'utilisation en tant qu'entrées/sorties. Si l'on utilise les fonctions périphériques, il ne faut en aucun cas modifier les bits DDRxn, PORTxn et PINxn correspondants.

Timers/Counters et génération de PWM

Les Timers/Counters, notes TCNTn oú *n* est le numéro Timer/Counter, sont des compteurs cadencés selon une certaine fréquence définie. Lorsqu'il atteint son maximum (MAX[5]), il passe à 0 et recommence a compter. Ce passage de MAX a 0 peut s'accompagner d'une interruption interne (cf. *Interruptions internes et externes*).

A ces Timers/Counters se rajoutent un comparateur dont on peut définir le seuil à l'aide du registre OCRn. Lorsque le TCNTn atteint la valeur de OCRn, alors la patte OCn change de valeur, passant de 0 à 1 ou de 1 à 0 selon la configuration puis retournant a la valeur initiale lorsque TCNTn passe de MAX a 0.

Ces Timers/Counters peuvent donc avoir plusieurs utilisations. D'abord, le programmeur peut savoir le temps écoulé depuis qu'il a lancé le Timer/Counter. Ou encoré, l'interruption de passage de MAX a 0 peut être utilisée pour temporisation, et commencer ou arrêter un programme au bout d'un certain temps. La dernière utilisation, très utile en robotique pour la commande de servomoteurs, est la génération de signaux PWM. En effet, si la patte OCn change de valeur lorsque le TCNTn atteint la valeur du comparateur OCRn, on obtient sur OCn un signal alternativement à 1 puis à 0 avec des durées a 0 et a 1 réglables. On a bien OCn qui est un signal PWM dont on peut définir le rapport cyclique en mettant la bonne valeur dans OCRn.

Plusieurs registres sont utilisés pour configurer les Timers/Counters : TCCRn, OCRn, ASSR, TIMSK et TIFR. Leurs configurations sont compliquées, il convient de se reporter aux chapitres correspondants de la datasheet. La valeur du Timer/Counter, accessible via le

registre TCNTn, peut être modifiée à tout instant afin de réinitialiser la séquence à partir d'une certaine valeur, après avoir atteint MAX, le compteur recommencera à 0. A noter que la fréquence de comptage peut être un sous-multiple de la fréquence du microcontrôleur ou encoreêtre imposée par une horloge externe connectée à une patte du microcontrôleur.

Certains compteurs fonctionnent sur 16 bits. Ils présentent plusieurs avantages : ils peuvent compter plus loin donc plus longtemps; lors d'une utilisation pour générer du PWM, la résolution est plus élevée; et surtout ils possèdent deux comparateurs dont les seuils peuvent êtreréglésdifféremment, ce qui permet d'obtenir sur les pattes OCnA et OCnB deux

signaux PWM de même fréquence mais de rapport cyclique différent a l'aide d'un seul Timer/Counter. Dans lecas de Timer/Counter sur 16 bits, le registre TCNTn se décompose en deux registres TCNTnL et TCNTnH tandis que OCRn se décompose en OCRnAL et OCRnAH pour le premier comparateur et OCRnBL, OCRnBH pour le deuxième comparateur.

Convertisseurs analogique/numérique

Le monde réel étant continu, c'est-á-dire analogique, tandis que les microcontrôleursfonctionnent en numérique, il faut bien avoir une interface entre les deux : le convertisseur analogique/numérique. Son fonctionnement est simple à comprendre : à intervalles de temps réguliers, la tension analogique est mesurée puis arrondie afin de pouvoir être convertie en numérique. La conversion est donc d'autant plus fidèle que le nombre de bits du codage est enlevé et est inversement proportionnelle à la plage de tension analogique en entrée. Les microcontrôleurs de la famille AVR possèdent des convertisseurs embarques donc il n'est pas nécessaire d'en concevoir ou d'en acheter.

Par exemple, l'ATmegal28 possède un CAN fonctionnant sur 10 bits. Ce CAN peut convertir jusqu'á 8 signaux analogiques par intermittence. Le temps maximum de conversion est de 260µs mais peut être diminué en réduisant la résolution (nombre de bits du codage). L'échelle de tension en entrée est par défaut [0, *Vcc]* mais peut être passée à [0, V_{ref}*]* oú V_{ref} < *Vcc* est une tension appliquée à la patte A_{ref}. A noter que les CAN peuvent être configures pour ne faire qu'une conversion ou pour convertir en continu la tension analogique, la conversion étant alors régulièrement mise à jour selon la durée de conversion.

Communication serie USART

USART est l'abréviation pour *Universal Synchronous and Asynchronous serial Receiver and Transmitter,* c'est un systéme de communication point-á-point tres flexible.

Les données sont transmises par trames de 7 à 13 bits. La trame a le format suivant :

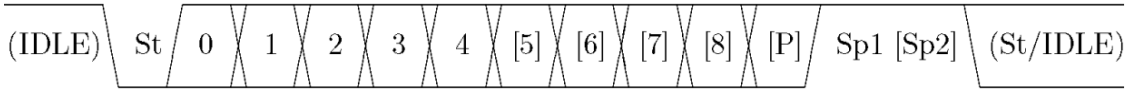


Figure 3-8 Trame USART

St : bit de départ, toujours à 0
n : bits de données
P : bit de parité
Sp : bit(s) de fin, toujours à 1

IDLE : pas de transfert, la ligne doit êtreà 1

Comparateur analogique

Outre les convertisseurs analogiques/numériques, les microcontrôleurs AVR possèdent un comparateur analogique. Celui-ci compare les deux tensions appliquées aux pattes AINO et AIN1. Si *AI NO <AI NI* alors le bit ACÓ (5^{eme} bit du registre ACSR) passe à 1. Sinon, ACÓ reste a 0. De plus, une interruption peut être déclenchée lors du passage à 1 de ACÓ.

Watchdog Timer

Le Watchdog Timer se charge de faire des RESET a intervalle de temps régulier. II utilise un oscillateur RC interne en tant qu'horloge cadencée à 1MHz. Ce système permet de recommencer le même programme à intervalles de temps réguliers ou encore de se réveiller d'un mode d'économie d'énergie après un certain lapse de temps (voir paragraphe suivant). A noter que la mémoire RAM n'est pas perdu au cours de ce RESET, ce qui permet d'augmenter

le temps entre deux réexécution du programme en tenant à jour un compteur qui est décrémenté a chaque Watchdog Reset jusqu'á ce qu'il atteigne 0 et alors on commence réellement le programme.

Modes d'économie d'énergie

Certaines applications ne nécessitent pas que le microcontrôleur tourne continuellement. II serait donc souhaitable de le désactiver, de "l'endormir", pour qu'il consommé aussi peu d'énergie que possible. Ceci est possible grâce aux Sleep Mode. Plusieurs modes sont disponibles, afin de pouvoir "endormir" ce dont on n'a pas besoin et de conserver ce qui doit l'être.

Une fois configuré, il suffit pour entrer dans le mode d'énergie d'exécuter l'instruction SLEEP. La sortie du mode se fait via une interruption externe ou interne, selon le mode utilisé.

Interruptions internes et externes

Certains événements peuvent intervenir á n'importe quel instant et doivent être gérés sans attendre. II est évident que de faire constamment une batterie de tests pour voir si tel ou tel événement s'est produit n'est pas performant du tout. Les interruptions sont LA solution à ce problème. En effet, dès qu'un événement se produit, si l'interruption est correctement configurée, le microcontrôleur arrête immédiatement son exécution pour exécuter une autre portion de code gérant cet événement. Des que cette portion de code a fini d'être exécutée, le microcontrôleur retourne à son programme principal, à l'endroit où il l'avait quitté.Les interruptions possibles sont de deux types :

Externe : lorsqu'une des pattes passe dans un certain état ou tant que la patte est dans un état défini
Interne : par exemple lorsque qu'un compteur dépasse sa valeur maximale[8], lorsqu'un transfert via USART est terminé...

Caractéristiques techniques de AVR Atmega 128:

- High-performance, Low-power Atmel AVR 8-bit Microcontroller
- Advanced RISC Architecture
  - 133 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers + Peripheral Control Registers
  - Up to 16MIPS Throughput at 16MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 128Kbytes of In-System Self-programmable Flash program memory
  - 4Kbytes EEPROM
  - 4Kbytes Internal SRAM
  - Write/Erase cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85□C/100 years at 25□C
  - Optional Boot Code Section with Independent Lock Bits

In-System Programming by On-chip Boot Program

True Read-While-Write Operation

  - Up to 64Kbytes Optional External Memory Space
  - Programming Lock for Software Security
  - SPI Interface for In-System Programming
- QTouch library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition

- Up to 64 sense channels
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - Two Expanded 16-bit Timer/Counters with Separate Prescaler, Compare Mode and Capture

Mode

  - Real Time Counter with Separate Oscillator
  - Two 8-bit PWM Channels
  - 6 PWM Channels with Programmable Resolution from 2 to 16 Bits
  - Output Compare Modulator
  - 8-channel, 10-bit ADC

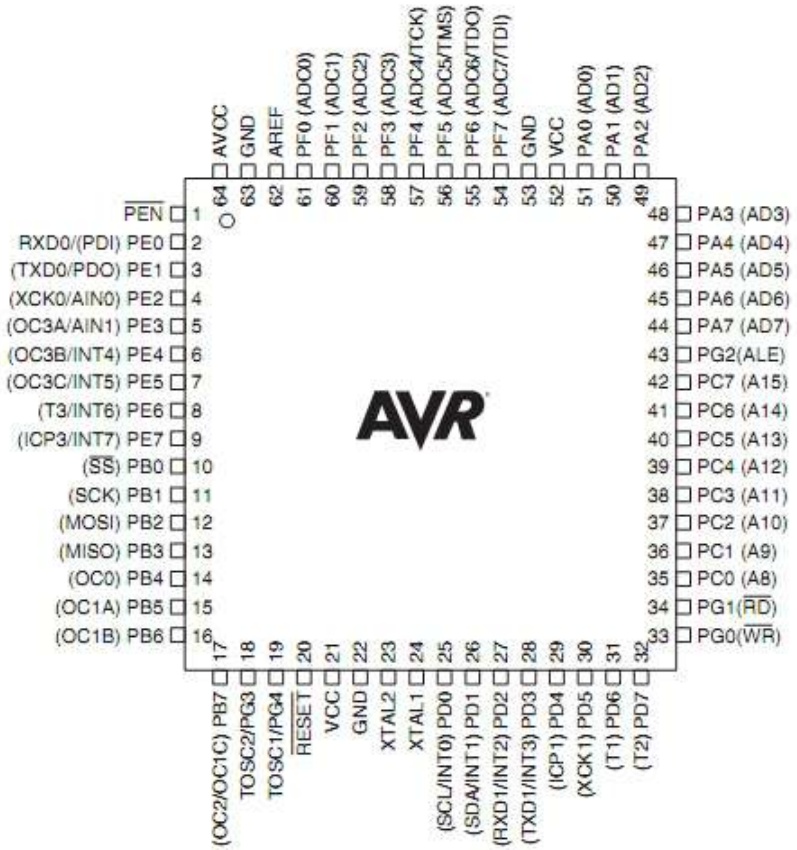
    8 Single-ended Channels

    7 Differential Channels

    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Dual Programmable Serial USARTs
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and

Extended Standby

  - Software Selectable Clock Frequency
  - ATmega103 Compatibility Mode Selected by a Fuse
  - Global Pull-up Disable
- I/O and Packages
  - 53 Programmable I/O Lines
  - 64-lead TQFP and 64-pad QFN/MLF
- Operating Voltages
  - 4.5 - 5.5V ATmega128
- Speed Grades
  - 0 - 16MHz ATmega128



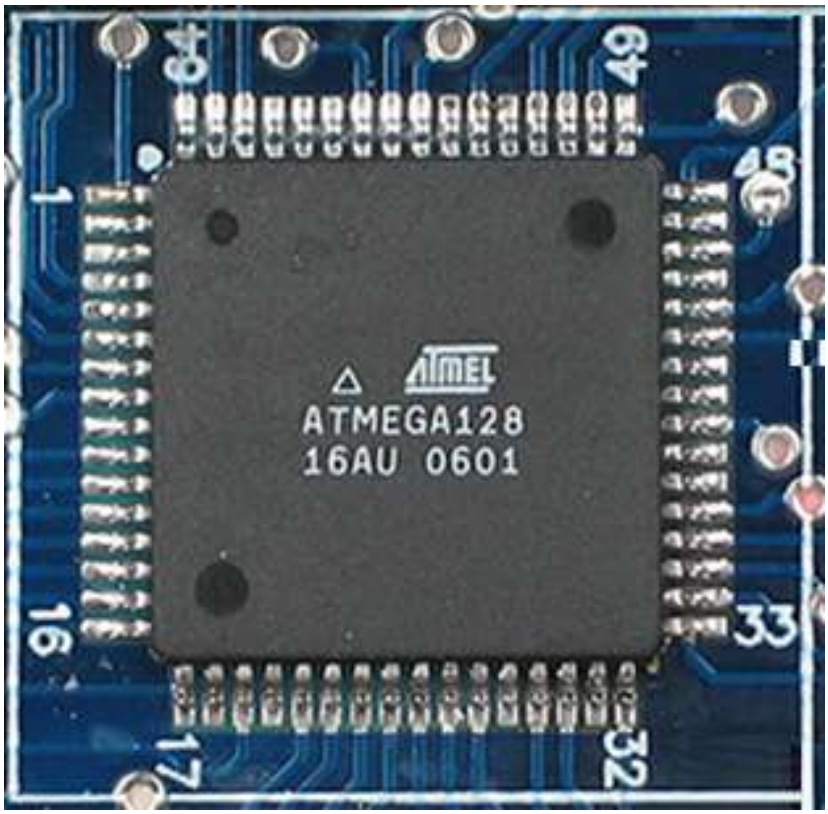


Figure3-9:atmega128

I-3-2 : Caractéristiques des capteurs

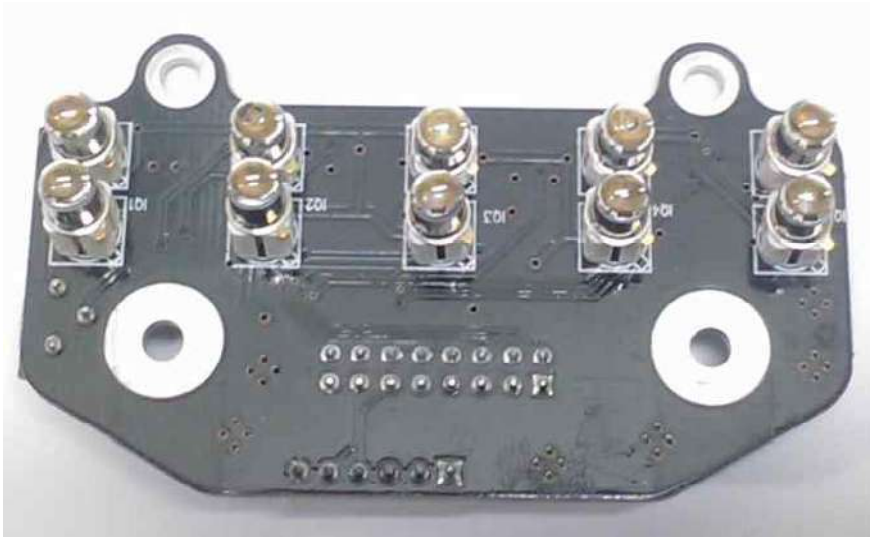
a) Capteur CNY70(IR) : On a cinq capteur infrarouge sensible a la couleur en dessous de plateforme de robot de distance entre eux 15mm, en effet il est au niveau logique haut (5V) lorsqu'il perçoit une couleur blanche et au niveau bas pour du noir. Le niveau de tension varie selon la surface et la couleur utilisée.. Le capteur doit être place très proche du sol pour capter de manière optimal. Le principe de fonctionnement est une émission et réception infrarouge.

Émetteur

- Tension d'alimentation nominale : 1.25 V
- Tension d'alimentation maximale : 1,6 V
- Courant d'alimentation : 50 mA

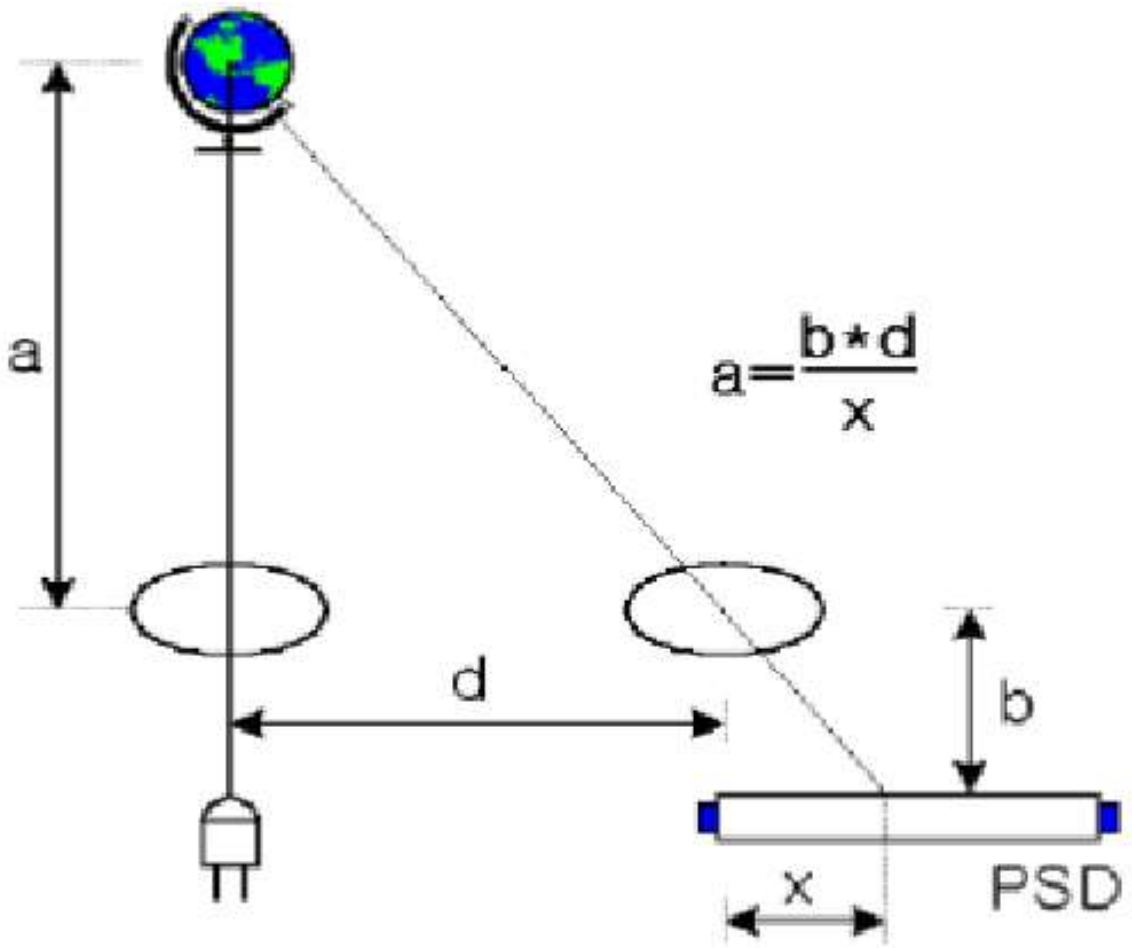
Détecteur

- Tension de sortie nominale : 5 V
- Tension de sortie maximale : 5 V
- Courant de sortie nominal : 100µA
- Courant de sortie maximum (théorique) : 1mA



Figuer3-10 : plaque présenté les cinq IR de ED7275

b) Capteur de distance PSD : Sont des capteurs de détection de distance, placer à l'avant du robot, chaque deux PSD adjacent font un angle de 30°, avec une plage de détection de distance allant de 10 à 80cm.



Figue3-11: chemin présenter le principe de fonctionnement de PSD

I- Le robot ED-7275 suiveur de ligne

Grâce à des capteurs infrarouges qui se trouver au-dessous de la plateforme du robot on peut détecter une ligne noir et commandé notre robot pour suivre cette ligne au cours de son déplacement.

II-1 - Le principe de la détection de la ligne :

Nous savons qu'une radiation de longueur d'onde sera absorbée par un matériau, si celui-ci diffuse toutes les radiations sauf celle ayant cette longueur d'onde, qui sera diffusée par un autre matériau.

Pour notre cas, nous avons utilisé deux "couleurs" : le noir qui absorbe toutes les longueurs d'ondes dont les infrarouges, et le blanc qui, lui, diffuse toutes les longueurs d'ondes dont les infrarouges Quant aux yeux de notre robot, ils ne devront qu'être capables de détecter l'absence ou la présence de la ligne. Par conséquent des capteurs isolés devraient suffire

Le robot se déplacera sur un terrain blanc et devra suivre une ligne noire d'unelargeur de 20 mm. Il pourra ensuite effectuer plusieurs figures. Cette piste sera étendue sur environ 10m.

Algorithme d'un système avec3 capteurs alignés

Dans cet algorithme on a utilisé seulement 3 capteurs de détection de ligne,aligné sur une ligne perpendiculaire (et décalée par rapport au centre du robot) au sens de déplacement du robot.

Ce système consiste à détecter les deux sorties de lignes possibles, en supposant qu'initialement le robot est correctement placé sur la ligne. Si on détecte que le robot sort de celle-ci, il tournera à l'instant même dans le sens favorisant le retour sur la ligne. Ici, nous utilisons le capteur central pour certifier si le robot est bien sur la ligne (sachant que son épaisseur est suffisamment grande pour ne pas être "cachée" entre deux capteurs).

Pour résumer le système de suivi qui découle de ce système à trois capteurs alignés, on distingue trois cas (conditions de l'algorithme) :

1. Si le robot détecte la ligne à droite, il tourne à droite
2. Si le robot détecte la ligne à gauche, il tourne à gauche

3. Si le robot ne détecte rien, il s'arrête

On note les trois capteurs comme suite :

Rc : capteurs de détection de ligne centre

Rd1 : capteurs de détection de ligne droite

Rg1 : capteurs de détection de ligne gauche

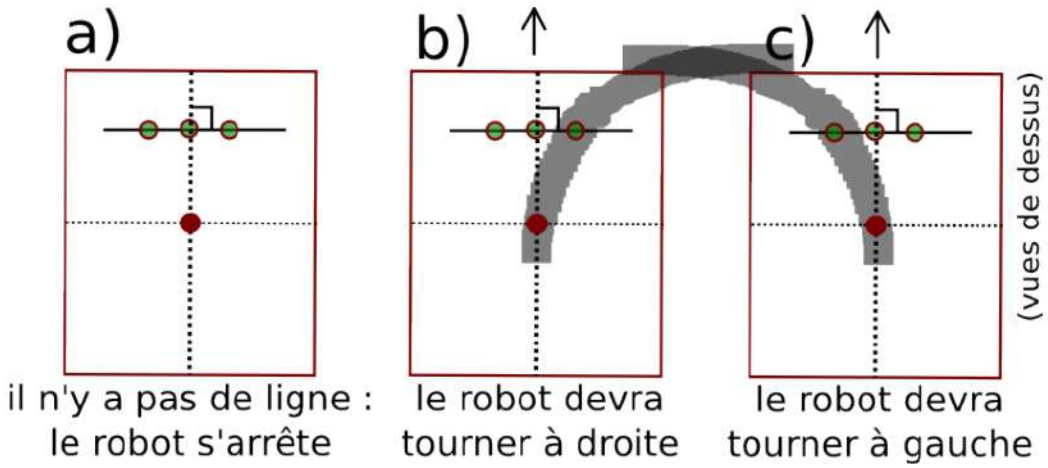


Figure 3-13 : Système simplifié de suivi de ligne mis en place

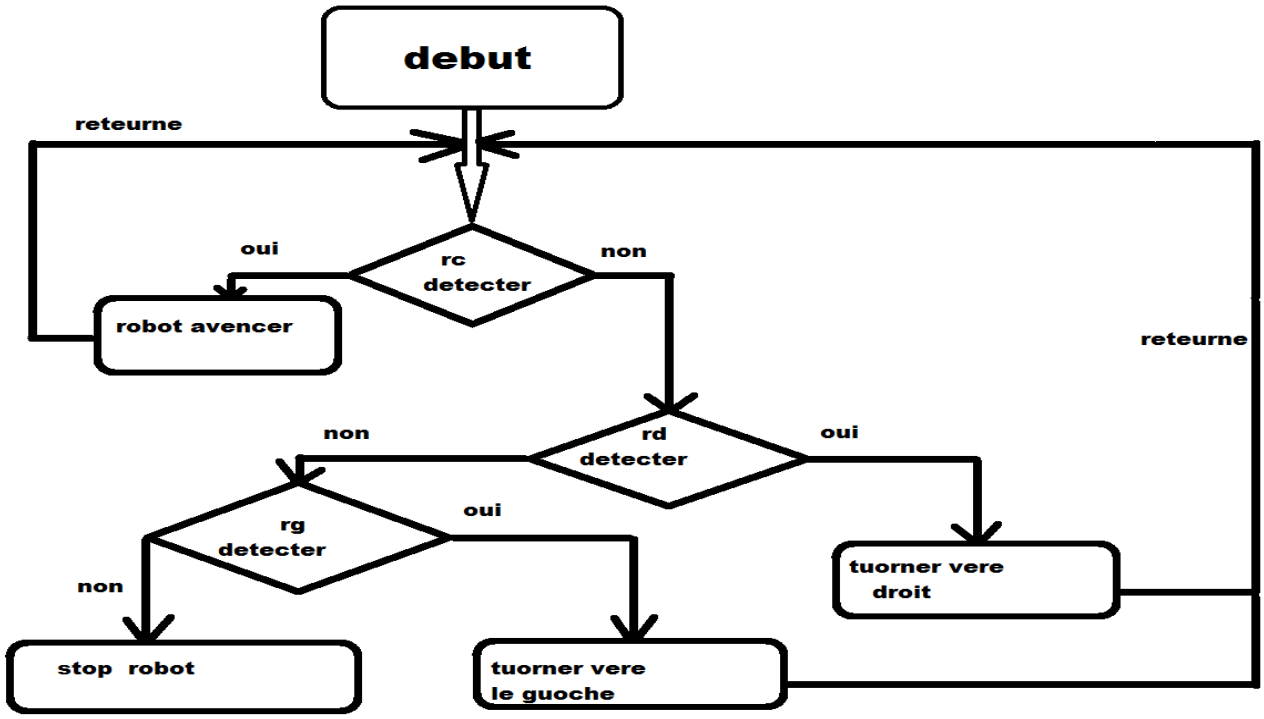


Figure 3-14 : Organigramme du suivie de la ligne avec 3 capteurs IR

En se limitant à cette conception théorique, le systèmerobotique semble totalement fonctionnel. Cependant, on a constaté qu'en utilisant des vitesses importantesle robot tourne soit trop à gauche, soit trop à droite. Cela engendre des ralentissements et même quelques fois des décrochages.

On explique cette défaillance technique par le fait que l'analyse de la trajectoire se fait en continue et donc :

Il y a un décalage entre l'analyse des données et l'action mécanique des moteurs

Algorithme d'un système avec 5 capteurs alignés

Maintenant on va utilisertous les cinq capteurs de détection de ligne, On note les cinq capteurs comme suite :

Rc : capteurs de détection de ligne centre

Rd1 : capteurs de détection de ligne droite1

Rd2 : capteurs de détection de ligne droite2

Rg1 : capteurs de détection de ligne gauche1

Rg2 : capteurs de détection de ligne gauche2

Pour résumer le système de suivi qui découle de ce système à cinq capteurs alignés, on distinguecinq cas (conditions de l'algorithme) :

• Si Rc détecte la ligne, robot avenacé avec vitesse maximum

• Si Rd1 détecte la ligne, robot diminué la vitesse et avenacé

• Si Rd2 détecte la ligne, le robot tourné à droite

•Si Rg1 détecte la ligne, robot diminué la vitesse et avenacé

•Si Rg2 détecte la ligne, le robot tourné à gauche

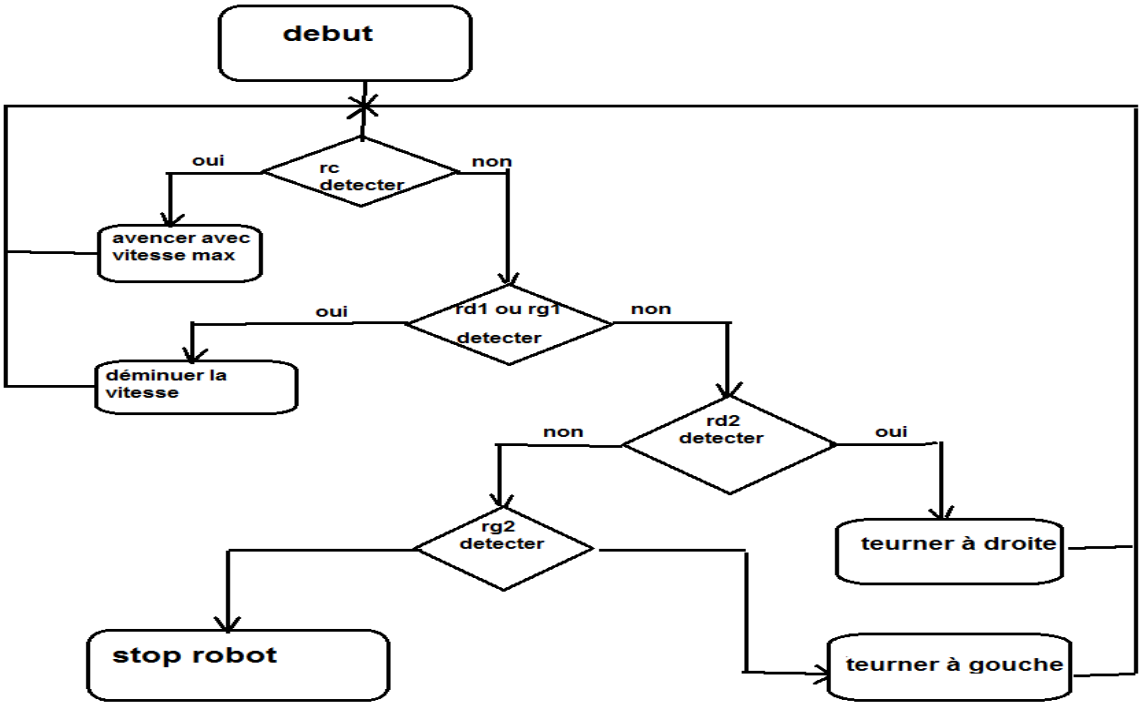


Figure 3-14 : Deuxième Algorithme

Pour ce système la vitesse du robot décroit (5cm/s), dès qu'il capte la ligne sur les capteurs rd1 ou rg1 il ne tourne que si le capteur rd2 ou rg2 détecte la ligne de ce fait le robot arrive a suivre correctement avec des vitesse honorable.

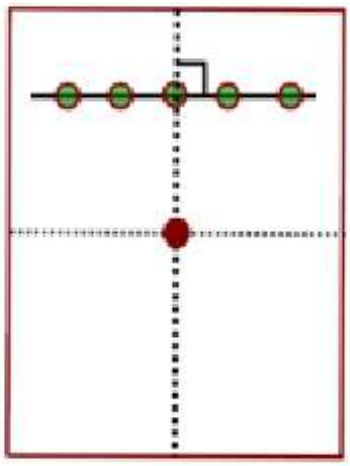


Figure 3-15 : Seul le capteur central détecte la ligne

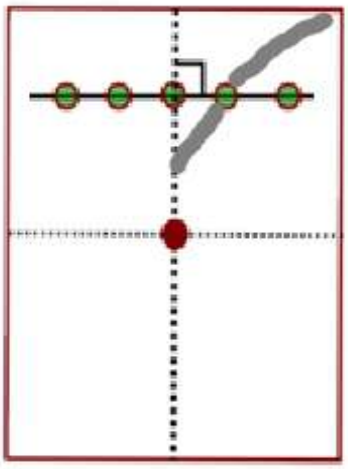


Figure 3-16: Seul le capteur 1er latéral droit détecte la ligne

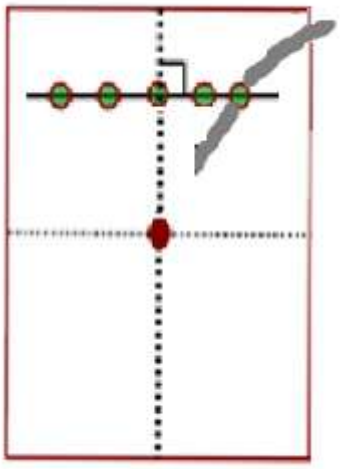


Figure 3-17 : Seul le capteur 1er latéral droit détecte la ligne

La ligne est détecté par le capteur central **(**Fig 3-15) donc vitesse max. La ligne est détecté par le 2em capteurs**(**Fig 3-16) donc vitesse =5cm/s ,En diminuons la vitesse on peut suivre l'évolution de la courbure du trajectoire .

La ligne est détecté par le 3em capteur**(**Fig 3-17) le robot arrête sa translation et tourne sur sa droite jusqu'a ace que le capteur central détecte la ligne ainsi le robot reprend son déplacement

I- MANIPULATIONS :

Le modèle cinématique de notre ROBOT est représentée par trois équationsdémontréesprécédemment:

$$\begin{bmatrix} \dot{\theta}1 \\ \dot{\theta}2 \\ \dot{\theta}3 \end{bmatrix} = 1/R * \begin{bmatrix} -\sin(\delta + \emptyset) & \cos(\delta + \emptyset) & l \\ -\sin(\delta - \emptyset) & -\cos(\delta - \emptyset) & l \\ \cos(\emptyset) & \sin(\emptyset) & l \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\emptyset} \end{bmatrix}$$

Dans le but de vérifier et de valider ces modèles mathématique, nous avons procédé à plusieurs expérimentations. Celles-ci consistent à générer différentes trajectoires pour vérifier les capacités du robot à les suivre correctement.

Nous avons réalisé des tests sur un grand nombre de trajectoires toutes différentes de par leurs formes et leurs complexités. Nous vous présentons dans ce mémoire deux exemples de trajectoires (arc et triangulaire) qui permettent de montrer les performances du robot mobile.

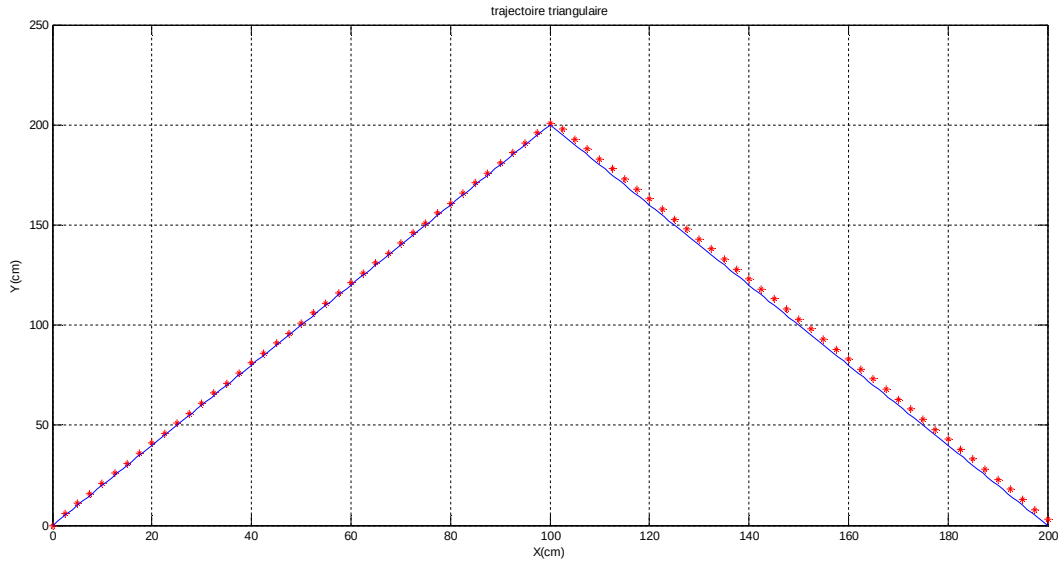


Figure 3-18: trajectoire triangulaire

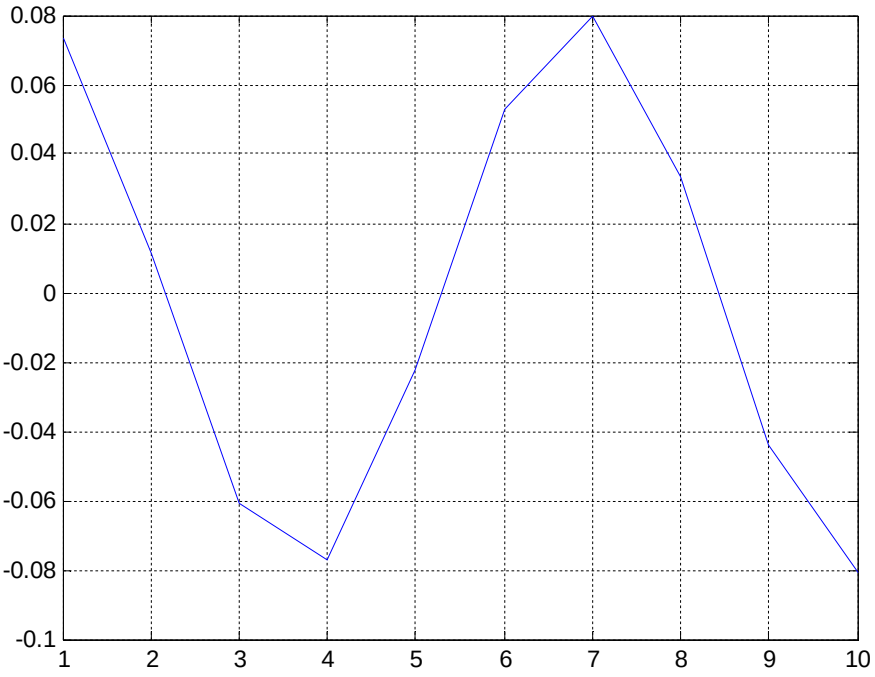


Figure 3-19 : Courbe d'erreur sur trajectoire triangulaire

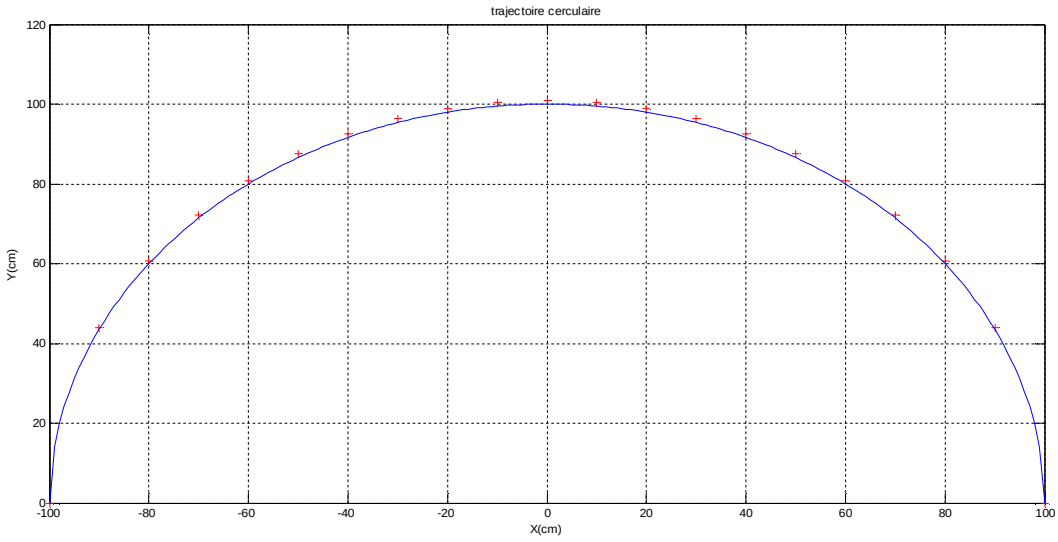


Figure 3-20 : trajectoire demi -cercle

Au travers les figure ci-dessus, nous constatons que le robot suit parfaitement les trajectoires imposées. Même dans le cas où la trajectoire change brusquement de direction, le robot arrive à la suivre avec une bonne précision. En effet, les erreurs de positionnements enregistrées n'excèdent pas en moyenne $\mathbf{E_{Moy} = 0.8\ cm}$, avec un maximum de $\mathbf{E_{Max} = 0.79\ cm}$ sur des portions très courbées de la trajectoire et une erreur minimum de $\mathbf{E_{min} = -0.08\ cm}$sur les portions droite. Ces résultats confortent d'une part, la réussite de la mise en route expérimentale du système robot mobile et sa configuration pour le suivi de trajectoire et d'autre part, la validité de nos calculs des différents modèles mathématiques du robot mobile. Cela nous amène logiquement à la deuxième partie de notre travail expérimental, à savoir, l'asservissement de ce robot mobile utilisant notre réseau de neurones CMAC.

En effet, nous remplacerons le model cinématique par un RN CMAC qui devra **apprendre** ce modèle à partir d'un jeu d'apprentissage que nous avons recueillis du système robotique. In fine, le CMAC devra être capable de reproduire les mêmes vitesses calculées précédemment par le modèle cinématique. Lesentréesdu CMAC serontles vitesses linéaires et la vitesse de rotation du robot et sa réponse représentera les 3 vitesses angulaires à fournir aux3 moteurs du robot.

Pour ce faire nous avons utilisé un CMAC avec les paramètres suivant :

Vecteur de déplacement d=[1 2 1]

Paramètre de généralisation $\rho = 5$

Problème :

On optant pour cette configuration du réseau de neurones CMAC, nous obtenons 900 poids qui représentent donc 900 réels. Notre robot mobile étant équipé d'un processeur atmega 128 est un µC de 8 bits, les réels signés (FLOAT) sont stockés sur 4 octets, ce qui nous donne un total de 3600 octets. Ce nombre important est difficilement gérable par ce dispositif. Pour clarifier cette surcharge du processeur atmega 128 prenons l'exemple avec un CMAC avec un seul poids w1=0.5. Ce dernier va êtretransforméselon la norme IEEE745 en 4 octets

1er octet : 0x00

2ème octet 0x00

3ème octet 0x00

4ème octet 0x3F

Pour le multiplier par 2 il faut que le µC multiplie le 1eroctet et garde la retenue puis le 2ème en rajoutant la retenue de la 1èreopération, ainsi de suite jusqu'au 4 octets sachant que chaque opération va prendre 1 cycle donc le calcule devient trop important pour une application en temps réel sur µC de 8 bits.

Plus simplement,Comme il ne fait que 8 bits "d'un coup", il doit faire un grand nombre d'opérations sur 8bits pour trouver le résultat d'une opération sur 32 bits.
C'est pour palier à ce manque de puissance que nous avons développé une carte électronique dotée de microcontrôleurs plus performants pour aider le µc et nous l'avons doté par d autre capteurs afin de mieux utilisé et sécurisé le robot dans ces déplacements :

Description de la carte :

Nous avons mis en place un système basé sur le PIC 18F2550 et le bus de communication I2C.

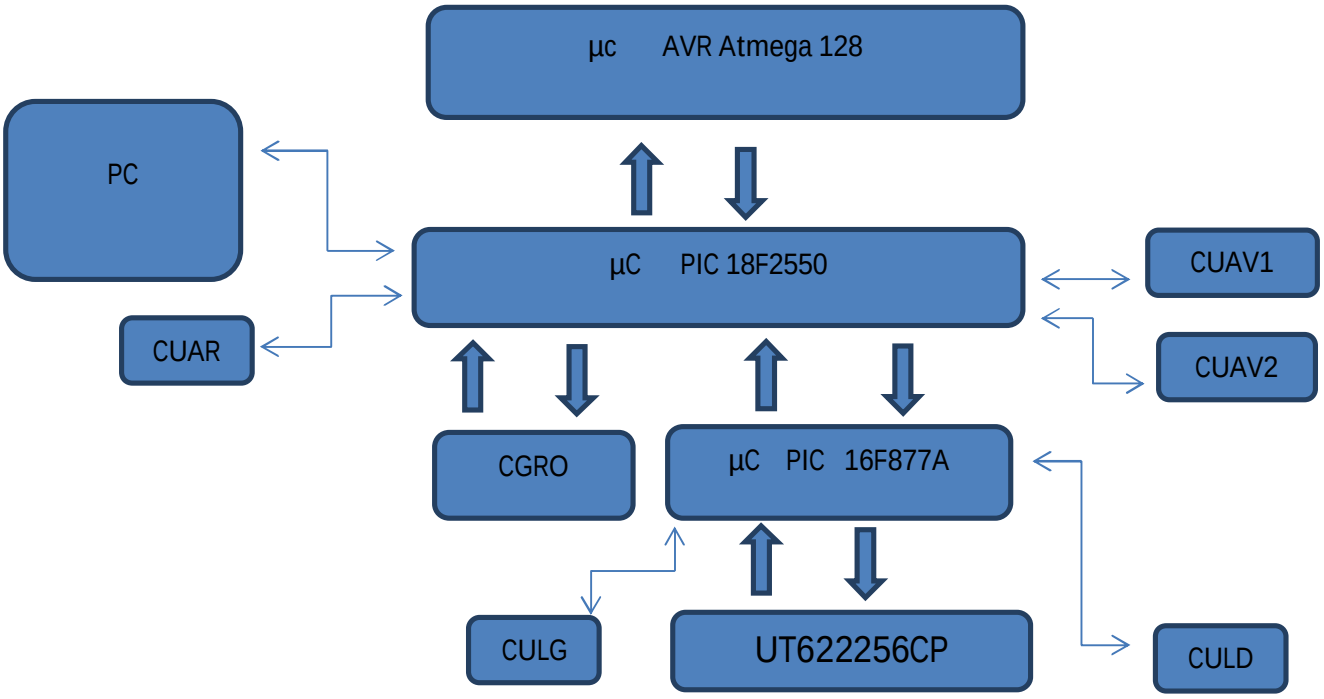
Le choix du PIC 18f2550 a été dicté par sa gestion du port USB (la librairie HID de MICROCHIP) ainsi la communication avec le PC se voit amélioré (débit plus élevé), facilité (puisque aucun driver n'est nécessaire) et surtout nous disposerons d'une trame de 64 octets.

Nous avons connecté à ce PICune mémoire UT 26566 70-pll. Celle-ci servira à mémoriser les informations par les 5 capteurs ultrason et du capteur gyroscope. L'utilisation des capteurs ultrason nous permettra de palier les problèmes d'éblouissement des capteurs IR. Quant au capteur

gyroscopique, il permet d'indiquer l'accélération, ainsi que l'orientation du robot par rapport à son centre.

Afin de rendre le système plus performant, nous avons rajouté un deuxièmePIC16 (16F877A) en mode esclavepourgérerla mémoire UT 26566 ainsi que les deux modules ultrasons latéraux. Ceci diminuera la charge sur le premier PIC18 (18F2550).

Schéma synoptique de la carte développé :



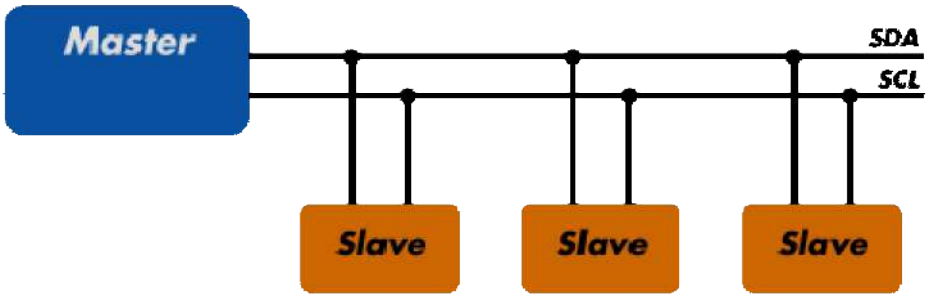
CUAV : capteur ultrason avant
CUAR : capteur ultrason arrière
CGRO : module gyroscope
CULG capteur ultrason latéral gauche
CULD capteur ultrason latéral Droite

Figure 3-21Schéma synoptique de la carte

Pour gérer l'ensemble, nous avons développé une application sous DELPHI. Elle permet la gestion, la commande et l'échange d'informations entre le robot et le PC via le port USB.

Le bus I2C facilite la communication entre le maitre et l'esclave (PIC18 et PIC16). De plus le PIC esclave ne répond que lorsque le PIC maître le sollicite et cela est possible grâce aux caractéristiques du protocole I2C (voir plus bas)

Présentation du bus I2C :



Le bus I2C (Inter Integrated Circuit) a été développé au début des années 80 par Philips Semi-conducteurs pour permettre de relier facilement à un microprocesseur les différents circuits d'un téléviseur moderne.

Caractéristiques

Le bus I2C permet de faire communiquer entre eux des composants électroniques très divers grâce à seulement trois fils : un signal de données (SDA), un signal d'horloge (SCL) et un signal de référence électrique (masse).

Ceci permet de réaliser des équipements ayant des fonctionnalités très puissantes (en apportant toute la puissance des systèmes micro-programées) tout en conservant un circuit imprimé très simple par rapport un schéma classique (8 bits de données, 16 bits d'adresse + les bits de contrôle).

Les données sont transmises en série à 100 Kbits/s en mode standard et jusqu'à 400 Kbits/s en mode rapide. Ce qui ouvre la porte de cette technologie à toutes les applications où la vitesse n'est pas primordiale.

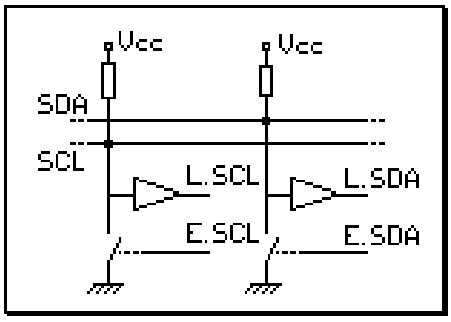
De nombreux fabricants ayant adopté le système, la variété des circuits disponibles disposant d'un port I2C est énorme : ports d'E/S bidirectionnels, convertisseurs A/N et N/A, mémoires (RAM, EPROM, EEPROM,...), circuits audio (égaliseur, contrôle de volume,...) et autre drivers (LED, LCD,...).

Le nombre de composants qu'il est ainsi possible de relier est essentiellement limité par la charge capacitive des lignes SDA et SCL : 400 pF.

Principes

Afin de d'éviter les conflits électriques, les Entrées/Sorties SDA et SCL sont de type "collecteur ouvert".

Voici un schéma de principe :



Structure d'E/S d'un module I2C

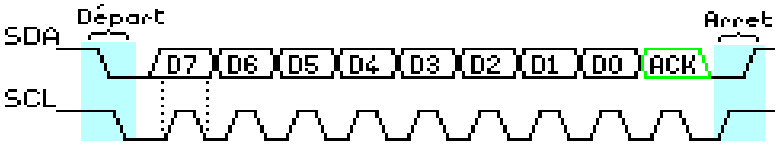
Le protocole I2C

Plusieurs circuits pouvant être branchés en même temps sur le même bus, il a été nécessaire d'instaurer un protocole entre eux, afin d'éviter les problèmes dus à une prise de parole simultanée de différents modules. C'est le protocole I2C. Celui-ci définit la succession des états logiques possibles sur SDA et SCL, et la façon dont doivent réagir les circuits en cas de conflits.

Prise de contrôle du bus

Pour prendre le contrôle du bus, il faut que celui-ci soit au repos (SDA et SCL à '1'). Pour transmettre des données sur le bus, il faut donc surveiller deux conditions particulières : la condition de départ (SDA passe à '0' alors que SCL reste à '1') et la condition d'arrêt (SDA passe à '1' alors que SCL reste à '1').

Lorsqu'un circuit prend le contrôle du bus après avoir vérifié qu'il était pris, il en devient le maître. C'est lui qui génère alors le signal d'horloge.

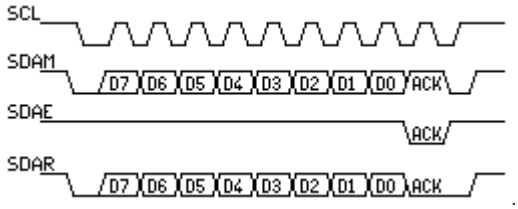


Exemple de condition de départ et d'arrêt

Transmission d'un octet

Après avoir imposé la condition de départ, le maître applique sur SDA le bit de poids fort D7. Il valide ensuite la donnée en appliquant pendant un instant un niveau '1' sur la ligne

SCL. Lorsque SCL revient à '0', il recommence l'opération jusqu'à ce que l'octet complet soit transmis. Il envoie alors un bit ACK à '1' tout en scrutant l'état réel de SDA. L'esclave doit alors imposer un niveau '0' pour signaler au maître que la transmission s'est effectuée correctement. Les sorties de chacun étant à "collecteur ouvert", le maître voit le '0' et peut alors passer à la suite.



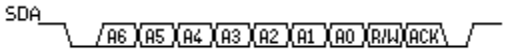
Exemple de transmission réussie

Dans cet exemple : SCL (horloge imposée par le maître), SDAM (niveaux de SDA imposés par le maître), SDAE (niveaux de SDA imposés par l'esclave), SDAR (niveaux de SDA réels résultants).

Transmission d'une adresse

Le nombre de composants qu'il est possible de connecter sur un bus I2C étant largement supérieur à deux, il est nécessaire de définir pour chacun une adresse unique.

L'adresse d'un circuit, codée sur sept bits, est définie d'une part par son type et d'autre part par l'état appliqué à un certain nombre de ses broches. Cette adresse est transmise sous la forme d'un octet au format particulier.



Exemple d'octet d'adresse

On remarque ici que les bits D7 à D1 représentent les adresses A6 à A0, et que le bit D0 est remplacé par le bit de R/W qui permet au maître de signaler s'il veut lire ou écrire une donnée. Le bit d'acquittement ACK fonctionne comme pour une donnée, ceci permet au maître de vérifier si l'esclave est disponible.

Cas particulier des mémoires

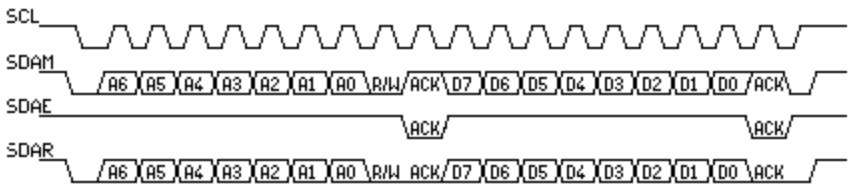
L'espace adressable d'un circuit de mémoire étant sensiblement plus grand que la plupart des autres types decircuits, l'adresse d'une information y est codée sur deux octets ou plus. Le premier représente toujours l'adresse du circuit, et les suivants l'adresse interne de la mémoire.

Les adresses réservées

Les adresses 00000XXX et 111111XX sont réservés à des modes de fonctionnement particuliers.

Ecriture d'une donnée

L'écriture d'une donnée par le maître ne pose pas de problème particulier :



Exemple d'écriture d'une donnée

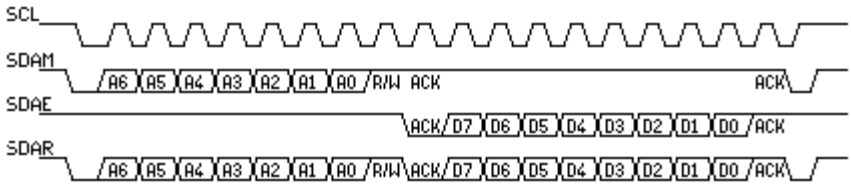
Cas particulier d'utilisation d'ACK

L'écriture d'un octet dans certains composants (mémoires, microcontrôleur,...) peut prendre un certain temps. Il est donc possible que le maître soit obligé d'attendre l'acquittement ACK avant de passer à la suite.

Lecture d'une donnée

La lecture d'une donnée par le maître se caractérise par l'utilisation spéciale qui est faite du bit ACK. Après la lecture d'un octet, le maître positionne ACK à '0' s'il veut lire la

donnée suivante (cas d'une mémoire par exemple) ou à '1' le cas échéant. Il envoie alors la condition d'arrêt.



Exemple de lecture d'une donnée

Implémentation et évaluation du CMAC sur la nouvelle plate-forme robot mobile.

Apres avoir résolu les contraintes liées à l'architecture du système robot mobile, nous avons implémenté et évalué le neuro-contrôleur CMAC. Afin de suivre la trajectoire désirée, le CMAC devra convertir les vitesses linéaires et angulaires du robot pour calculer les consignes (vitesses angulaires)à fournir aux 3 moteurs du robot. Grace à notre experiences aquise lors de la programmation de ce reseau de neurones et apres quelque tests, nous avons choisi de paramétrer le CMAC comme suit :

Vecteur de déplacement d=[1 2 1]

Paramètre de généralisation $\rho = 5$

Les resultats obtenue sont illustrés par les figures suivant :

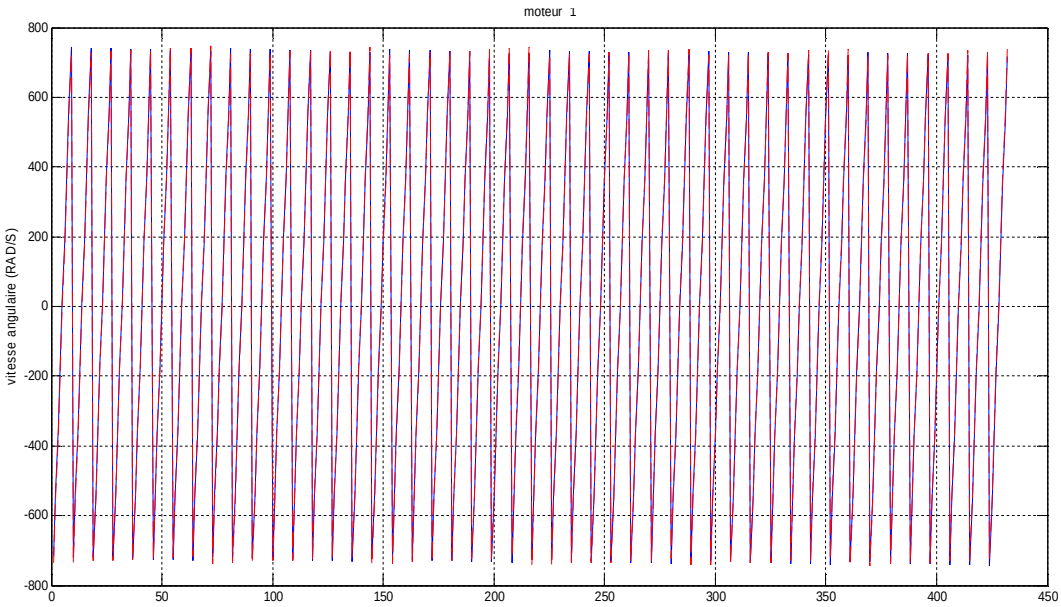


Figure 3-22 : Restitution CMAC de la vitesse angulaire 1.

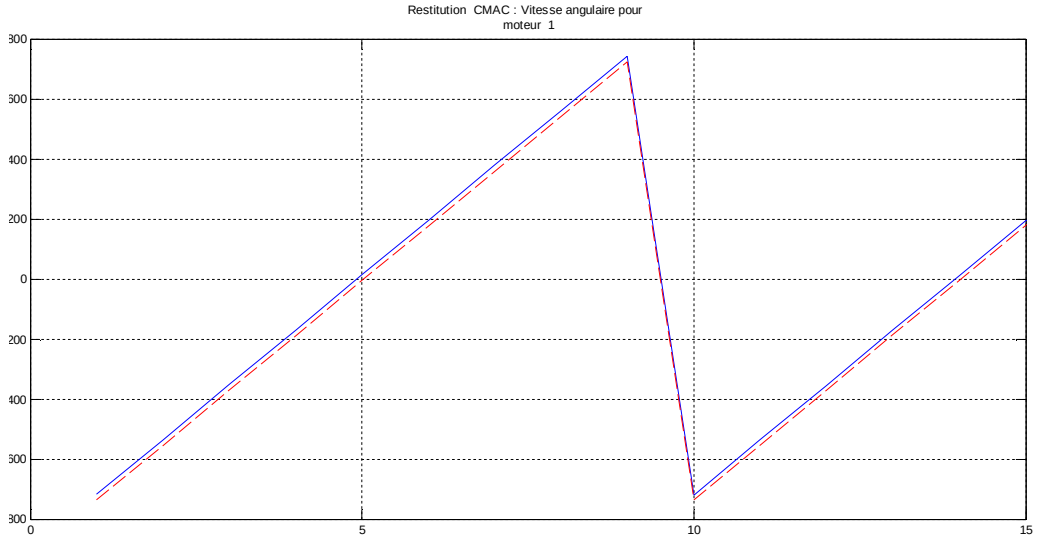


Figure 3-23 : Restitution CMAC de la vitesse angulaire 1(avec zoom).

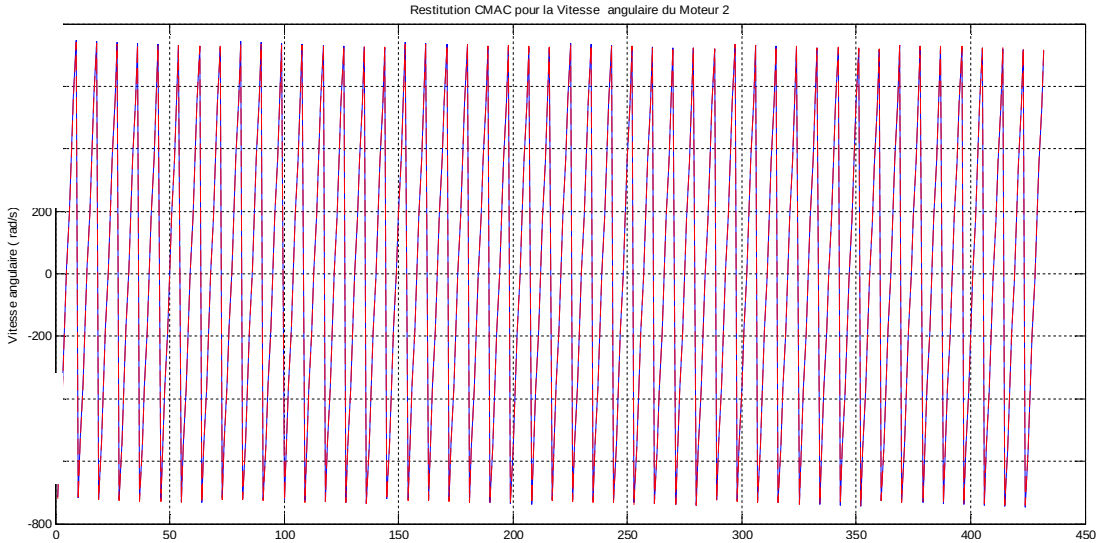


Figure 3-24 : Restitution CMAC de la vitesse angulaire 2.

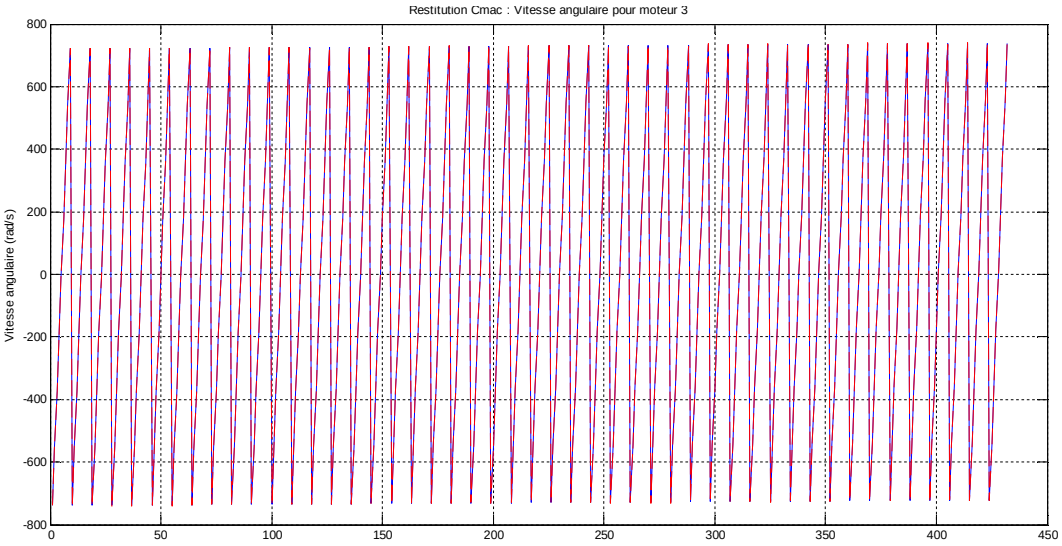


Figure 3-25 : Restitution CMAC de la vitesse angulaire 3.

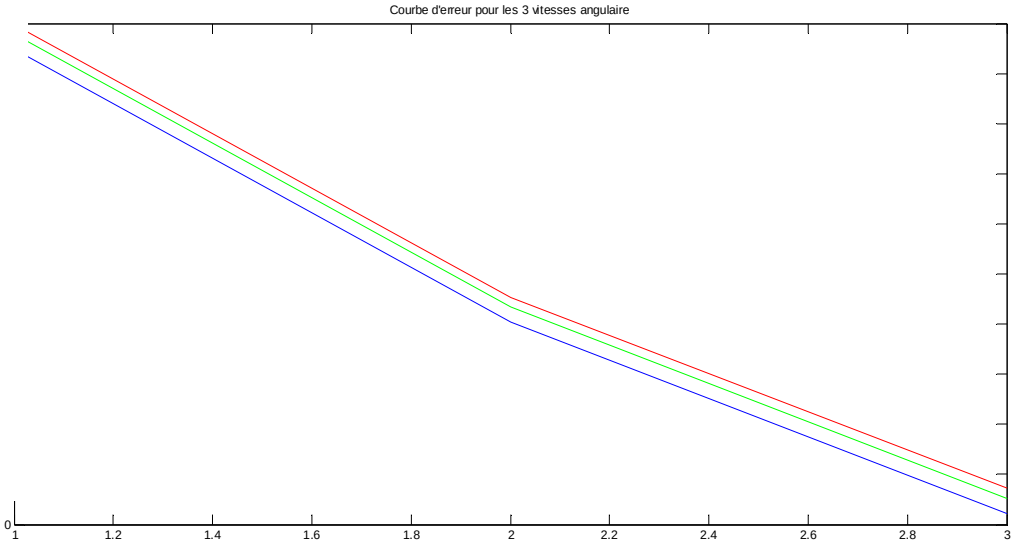


Figure 3-26 : Courbe d'erreur pour les 3 vitesse angulaire CMAC .

Taille du CMAC 900 poids .

L'Erreurobtenue :

	Emin	**Emax**	**Emoy**
Moteur 1	-0.8376 rad/s	0.8376 rad/s	0.8376 rad/s
Moteur 2	-0.6287 rad/s	0.6287 rad/s	0.6287 rad/s
Moteur 3	-1.1510 rad/s	1.1510 rad/s	1.1510 rad/s

On peut constater que le CMAC arrive à restituer la matrice cinématique avec une erreur moyenne 0.8376 rad/s cette donne l'impression d'être significatif et fatal pour le robot mais réellement cette erreur et négligeable prenons l'exemple suivant :

Demandant a notre CMAC de nous donner les vitesses angulaire correspondantes a une vitesse linéaire de 15cm/s.

Le résultats du CMAC comparé a celui de la matrice nous donne les graphes suivant :

Pour la matrice cinématique le résultat :

Vitesse angulaire Moteur 1	Vitesse angulaire Moteur 1	Vitesse angulaire Moteur 1
-4.33 (rad/s)	-4.33 (rad/s)	5 (rad/s)

Vitesse linéaire 15cm/s

Pour le CMAC :

Vitesse angulaire Moteur 1	Vitesse angulaire Moteur 1	Vitesse angulaire Moteur 1
-4.6 (rad/s)	-4.6 (rad/s)	4.97 (rad/s)

Vitesse linéaire 15.13 cm/s

On constate une erreur de 0.13cm/s.

L'effet sur les trajectoires du robot :

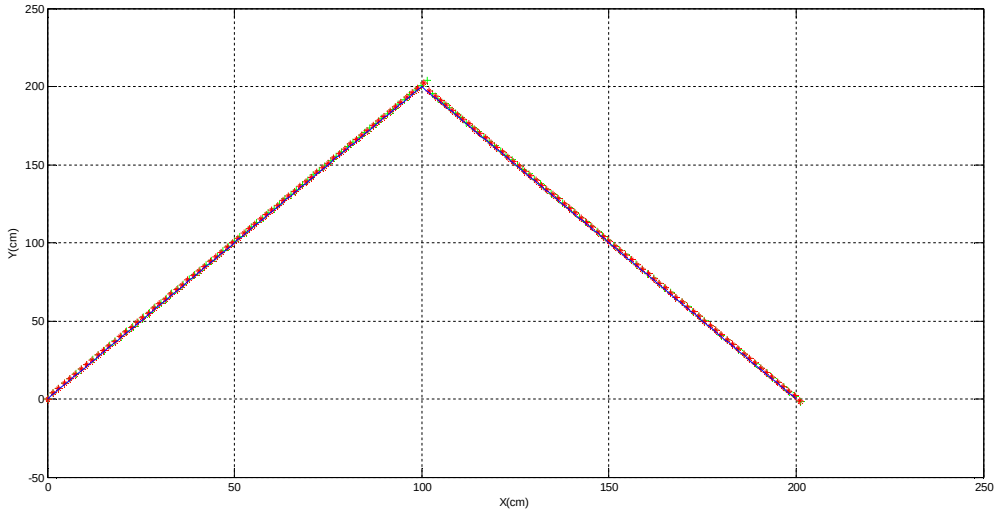


Figure 3-27 : trajectoire triangulaire (comparaison entre CMAC en vert et matrice cinématique en rouge) .

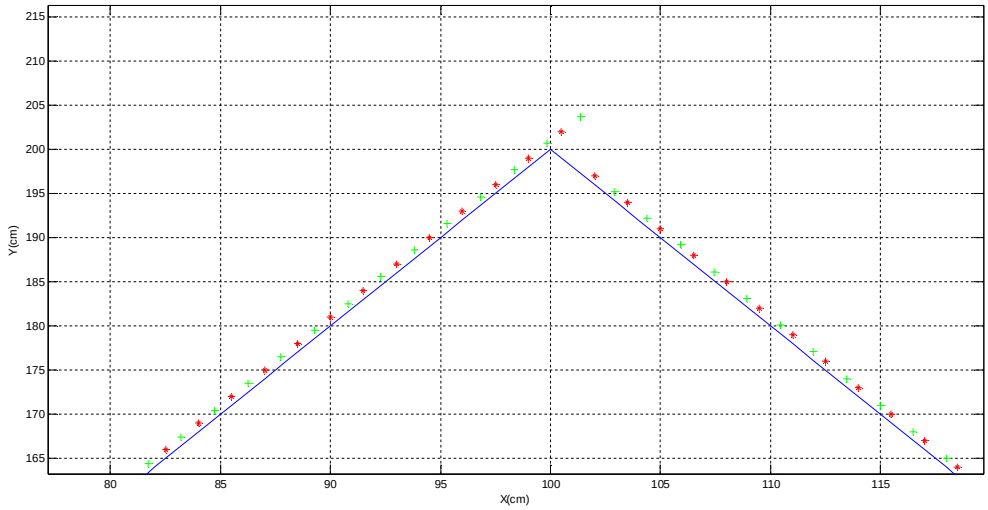


Figure 3-28 : trajectoire triangulaire(zoom) (comparaison entre CMAC en vert et matrice cinématique en rouge) .

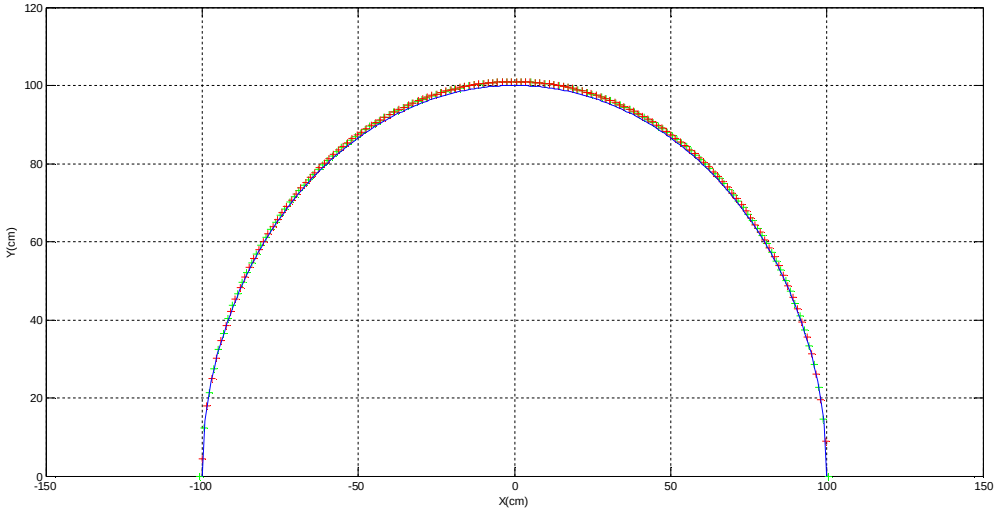


Figure 3-29 : trajectoire demi-cercle(comparaison entre CMAC en vert et matrice cinématique en rouge) .

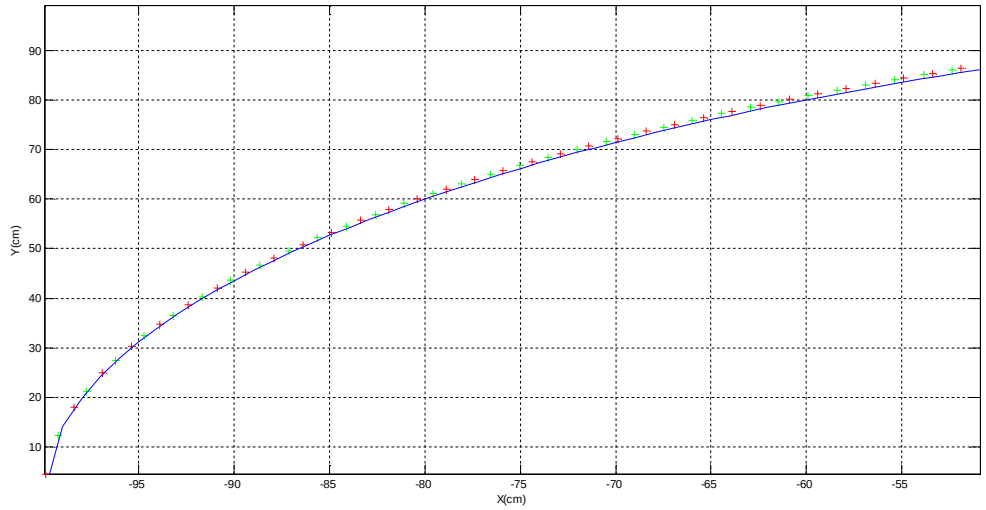


Figure 3-30 : trajectoire demi-cercle avec zoom(comparaison entre CMAC en vert et matrice cinématique en rouge) .

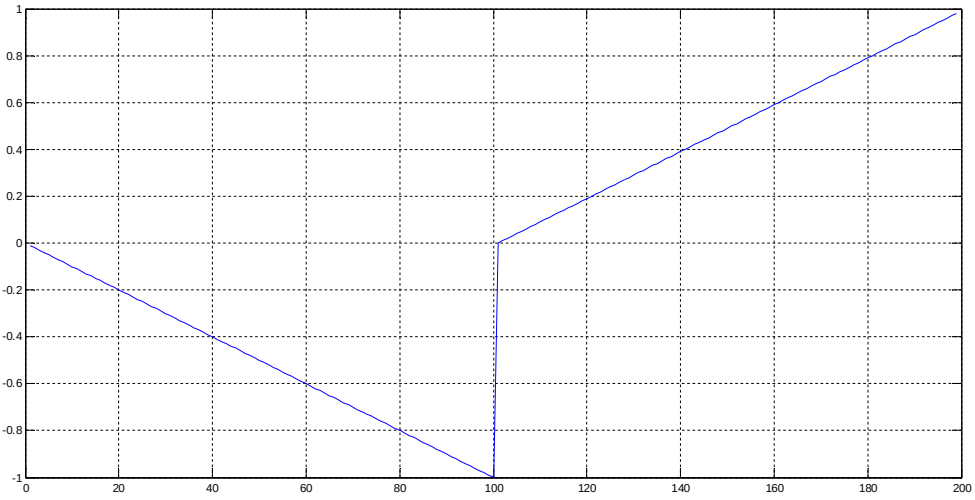


Figure 3-31: courbe d'erreur pour trajectoire triangulaire

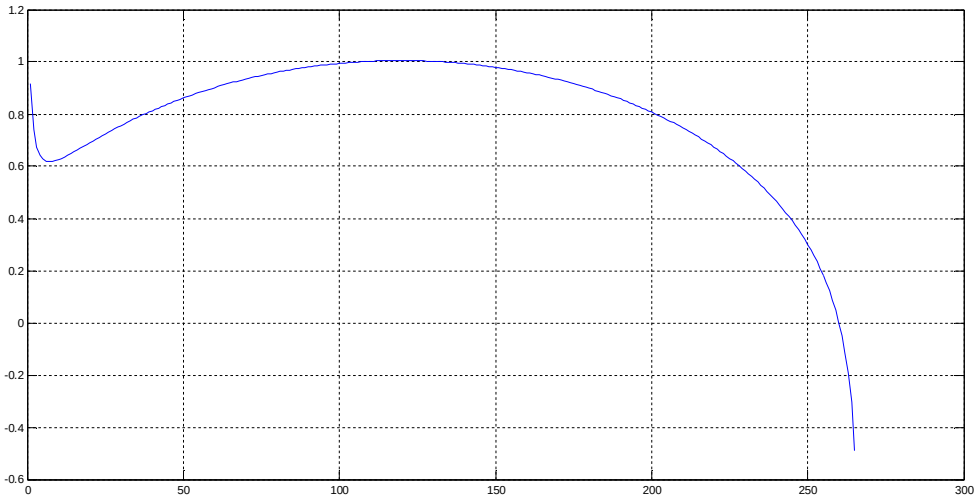


Figure 3-32: courbe d'erreur pour trajectoire Demi-Cercle.

On constate que l'erreur de CMAC na aucun effet sur les trajectoire du robot et cela est due au correcteur intégré au robot .

Donc le CMAC a réussi a reproduire le vitesses angulaire désiré afin de reproduire les trajectoire prédéfinit , certes il est possible de gommé cette erreur en jouant encore sur les paramètres du CMAC notamment la résolution sur l'axe des entrées en diminuant ce paramètre encore peut arrivéa une erreur presque nulle mais avec un nombre de poids très important nous avons testé cette solution avec un paramètre de 0.01 nous nous somme retrouvé avec 3400 poids et une erreur de 0.1 (rads/s) la charge que représente le stockage et le traitement de ces poids rends cette opération inutile parce que, en voit bien que l'incidence réelle de cette erreur est vraiment négligeable .

.

ANNEXE A

1- Le cervelet biologique

Le cervelet, situé en arrière du tronc cérébral, est de nos jours l'une des parties du système nerveux dont nous connaissons le mieux le fonctionnement. C'est la partie du cerveau chargée du *contrôle* de l'activité musculaire. L'intervention du cervelet est très importante pour l'apprentissage, l'équilibre et la coordination des mouvements du corps [Adelman87] tels que ceux utilisés pour jouer du piano ou tenir un objet.

Le cervelet permet de maintenir la posture et l'équilibre grâce au contrôle du tonus des muscles et de la position des membres du corps dans l'espace. Une tumeur du cervelet peut produire des mouvements saccadés des bras et des jambes (comme un ivrogne). Le cervelet est également important en ce qui concerne la capacité à effectuer des gestes rapides et répétitifs, comme taper sur le clavier d'un ordinateur. Contrairement à ce qui se passe dans le reste du cerveau, une lésion du côté droit du cervelet peut provoquer des symptômes du même côté du corps. Le poids du cervelet représente seulement 10% du poids du cerveau. Cependant, le cervelet contient plus de la moitié de tous les neurones du cerveau [Kandel et al.91].

1.1 Le cortex cérébelleux

Le *cortex cérébelleux* est constitué de *quatre couches de neurones* qui contiennent *cinq types de cellules* différentes et *trois types de fibres.* La figure 29 montre une représentation du tissu cortical cérébelleux où nous pouvons remarquer les quatre couches :

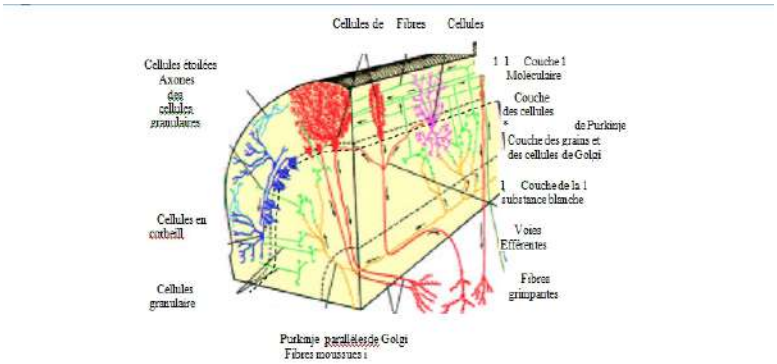
a. *La couche de la substance blanche.*
b. *La couche des grains et des cellules de Golgi.*
c. *La couche des cellules de Purkinje.*
d. *La couche moléculaire.*

Ainsi que les cinq types de cellules différentes :

a. *Les cellules de Purkinje.*
b. *Les cellules granulaires.*
c. *Les cellules de Golgi.*
d. *Les cellules étoilées.*
e. *Les cellules en corbeille.*

Et les trois types de fibres existantes :

a. *Les fibres moussues.*
b. *Les fibres grimpantes.*
c. *Les fibres parallèles.*



Le cervelet biologique

.1.2 Les couches du cortex cérébelleux

La couche de la substance blanche

Cette couche ne contient que des axones. Les entrées arrivent au cervelet via les *fibres moussues* [Spoelstra99] à partir de plusieurs sources du corps. Les *fibres moussues* sont les axones excitateurs des neurones de la moelle épinière et des neurones du pont (protubérance). Ce sont les principales entrées du cervelet qui contiennent des informations sensorielles et de rétroaction. Les fibres moussues peuvent être classifiées en deux types selon leur point d'origine : celles qui portent des ordres ("commands") de haut niveau dans le système moteur et celles qui portent des informations de rétroaction ("feedback") à propos des résultats des sorties motrices. Chaque fibre moussue forme une synapse excitatrice avec plusieurs centaines de cellules granulaires.

D'autres axones présents dans cette couche sont les *fibres grimpantes*. Ces fibres sont les axones des neurones du bulbe rachidien qui véhiculent des informations vestibulaires (l'équilibre). Elles font des synapses excitatrices avec les *cellules de Purkinje* et contiennent des informations importantes pour le processus d'apprentissage dans le cervelet.

La couche des grains et des cellules de Golgi

Cette couche, la plus profonde, contient *les cellules de Golgi*. Les *cellules de Golgi* sont des neurones inhibiteurs des *cellules granulaires*. Les *cellules de Golgi* reçoivent les informations venant de la moelle. Ces cellules reçoivent des contacts synaptiques des *fibres excitatrices moussues*. Chaque axone de sortie des cellules de Golgi forme une synapse inhibitrice avec plusieurs cellules granulaires. Les *cellules granulaires*, également présentes dans cette couche, projettent leurs axones appelés *fibres parallèles* jusqu'à la *couche moléculaire* du cortex cérébelleux, mais aussi sur *les cellules de Golgi. Les cellules granulaires* sont très fines et nombreuses (il y en a 10^{11}

environ). Chacune prend en moyenne 4 fibres moussues comme entrée [Smith98]. Elles forment synapse avec les dendrites des *cellules de Purkinje.*

La couche des cellules de Purkinje

Cette couche contient les corps cellulaires des *cellules de Purkinje*. Les *cellules de Purkinje* sont des neurones dont le corps cellulaire, en forme de poire, établit des synapses avec les axones des *cellules en corbeille*. Les cellules de Purkinje sont les cellules prédominantes du cervelet car elles sont à l'origine du message cérébelleux efférent. Elles sont beaucoup plus grandes que les *cellules granulaires*. Il en existe environ 15 millions dans le cervelet. Les dendrites de ces cellules se trouvent dans la couche moléculaire. Les corps cellulaires des *cellules de Purkinje* sont entourés par les terminaisons synaptiques des axones des *cellules étoilées* et des *fibres grimpantes.*

La couche moléculaire

Dans cette couche, la plus externe, nous trouvons les *cellules étoilées* et les *cellules en corbeille*. Les *cellules étoilées* ou *cellules en corbeille* (selon la morphologie) sont des neurones inhibiteurs des *cellules de Purkinje* qui établissent des synapses avec les *fibres parallèles.*

Les *fibres parallèles* sont les axones parallèles des cellules granulaires. Ces fibres forment des synapses avec les dendrites des *cellules de Golgi,* qui se projettent aussi dans la *couche moléculaire.* Il y a donc un microcircuit entre les *cellules granulaires* et les *cellules de Golgi.*

Ces cellules sont inhibitrices des *cellules de Purkinje.* Les messages reçus au niveau de ces

synapses sont transmis par les *cellules étoilées ou en corbeille* aux *cellules de Purkinje.* Les *fibres parallèles* forment des synapses avec les dendrites des *cellules de Golgi* qui se projettent aussi dans la *couche moléculaire.* Les *cellules de Golgi* reçoivent dans cette couche des contacts synaptiques des *fibres excitatrices parallèles et grimpantes.*

L'arborisation dendritique des *cellules de Purkinje* est en contact synaptique avec les axones des *cellules étoilées* et les *fibres grimpantes.* Les axones des *cellules de Purkinje* envoient les messages cérébelleux efférents vers les *noyaux intra-cérébelleux* qui les transmettent ensuite vers la moelle (tonus), le bulbe (équilibre) et le cortex via, le thalamus (motricité). Les *cellules de Purkinje* constituent la sortie du cortex cérébelleux.

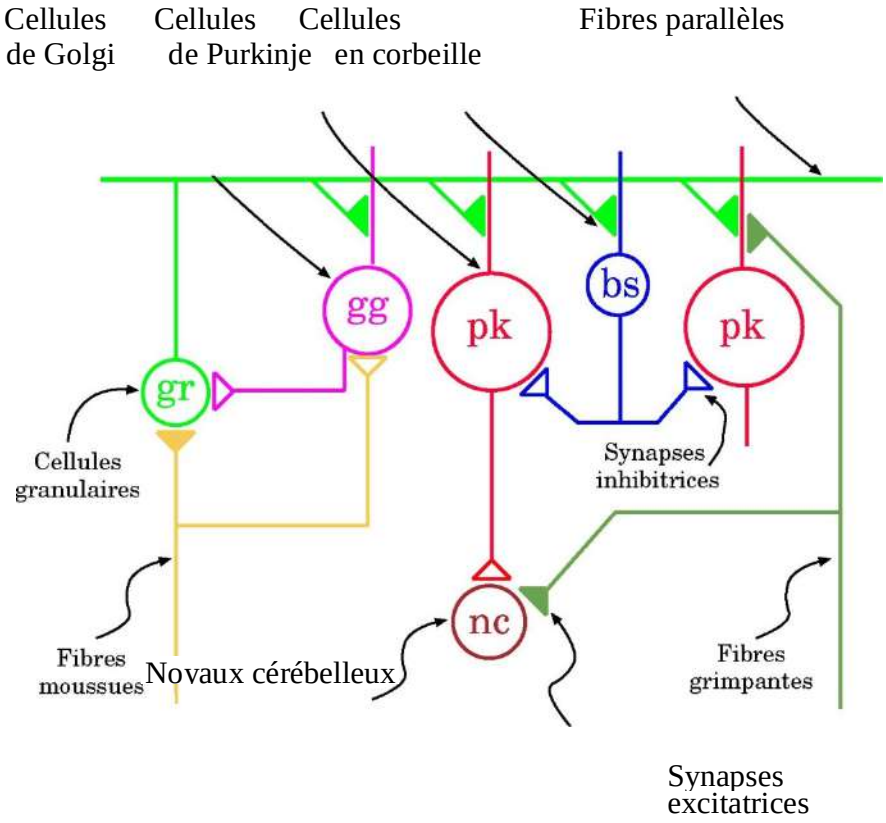
1.3 Câblage neuronal dans le cervelet

Dans le diagramme de la figure 30, nous présentons une représentation simplifiée du câblage neuronal dans le cervelet. Les *fibres moussues* et les fibres grimpantes sont les principales entrées du cervelet. La fonction des fibres grimpantes est le réglage de l'intensité des connections synaptiques avec les fibres parallèles de manière à modifier la sortie des cellules de Purkinje. Les fibres grimpantes contiennent l'information requise pour accomplir l'apprentissage. Nous pouvons remarquer que les fibres moussues finissent dans les *cellules granulaires (gr)* et inhibent les cellules de Golgi.

Un patron d'activation particulier provenant des fibres moussues excitera les cellules granulaires. Les cellules de Golgi inhiberont toutes les cellules qui n'ont pas été excitées de manière maximale. Le résultat consistera à transformer, pour chaque patron d'entrée (ou vecteur), la couche granulaire en un petit pourcentage, ou sous-ensemble, de fibres parallèles actives.

Les *axones des cellules de Purkinje (pk)* forment la sortie du cortex cérébelleux. Ils inhibent des neurones dans les *noyaux cérébelleux (ne)* avant de quitter le cervelet pour des destinations

ANNEXE A



FlG. 2 - *Représentation schématique du câblage neuronal dans le cervelet*

spécifiques dans le corps. Les *cellules de Golgi (gg)* inhibent les *cellules granulaires*, et *les cellules en corbeille (bs)* inhibent les *cellules de Purkinje*.

.2 Le cervelet artificiel CMAC

C.2.1 CMAC comme une alternative aux RN Multi-couches

Les RN CMAC sont une alternative aux Réseaux Neuronaux Multi-couches qui utilisent l'algorithme de *rétro-propagation du gradient de l'erreur* pour l'approximation de fonctions [Miller et al.90].

Les RN Multi-couches (avec rétro-propagation du gradient) présentent plusieurs inconvénients [Miller et al.90] :

- Ils ont besoin de plusieurs itérations pour converger.
- Ils sont inappropriés pour l'apprentissage en ligne (besoin de petits réseaux).
- Le nombre de calculs par itération est assez grand [Tham94], de sorte que l'algorithme est lent (sauf avec un matériel ("hardware") sophistiqué).
- Il existe des minima locaux dans la surface d'erreur.
- Il existe une interférence entre connaissances nouvelles et anciennes.

Par contre, les avantages du modèle de RN CMAC sont les suivants [Miller et al.90] :

- L'entraînement du réseau neuronal CMAC est très rapide (calcul assez simple), celui-ci est basé sur la règle d'apprentissage de Widrow-Hoff).
- La capacité d'apprendre de manière incrémentielle.
- La surface d'erreur est quadratique, il existe donc un seul minimum.
- La structure de CMAC sous la forme de table de recherche facilite sa réalisation matérielle.
- La capacité d'une généralisation locale.

Annexe B :

Déplacements

descouches du CMAC

La stratégie de déplacement des couches est une phase importante pour l'algorithme du réseau de neurones CMAC. Cette étape, présentée dans le chapitre2,consiste à répartir les couches(et donc toutes les fonctions de base) sur l'ensemble de l'espace d'entrée normalisé.De ce fait,cette stratégie qui influe sur l'organisation de la structure du réseau permet de définir la relation de voisinage entre les fonctions de base activé esparun stimuli.

Dans nos implémentations,nous avons utilisé la méthode de (ParksandMilitzer1991) pour positionner et déplacer les couches du réseau CMAC.Cette technique,qui est de loin la plus utilisée, est résumée dans cette annexe. Le tableau ci-dessous présente les valeurs optimales des vecteurs de déplacement **d** pour un espace d'entrée de dimensions n=2,…,6 et pour un paramètre de généralisation □ =2,…,20.C'est une version réduite de la table donnée par Parks et Militzer car, dans nos

applications, le paramètre de généralisation et la dimension des CMAC n'excède jamais ces valeurs.

□ *n*	2	3	4	5	6
2	[1,1]	[1,1,1	[1,1,1,1	[1,1,1,1,1	[1,1,1,1,1,1
3	[1,1]	]	]	]	]
4	[1,1]	[1,1,1	[1,1,1,1	[1,1,1,1,1	[1,1,1,1,1,1
5	[1,2]	]	]	]	]
6	[1,1]	[1,1,1	[1,1,1,1	[1,1,1,1,1	[1,1,1,1,1,1
7	[1,2	]	]	]	]

8	[1,3]	[1,1,3	[1,1,3,3	[1,1,1,3,3	[1,1,1,3,3,3
9	[1,2]	]	]	]	]
10	[1,3]	[1,2,4	[1,1,2,4	[1,1,2,2,4	[1,1,2,2,4,4
11	[1,3]	]	]	]	]
12	[1,5]	[1,1,3	[1,1,3,3	[1,1,1,3,3	[1,1,1,3,3,3
13	[1,5]	]	]	]	]
14	[1,3]	[1,2,4	[1,1,3,5	[1,2,3,4,5	[1,1,2,3,4,5
15	[1,4]	]	]	]	]
16	[1,3]	[1,1,5	[1,1,5,5	[1,1,1,5,5	[1,1,1,5,5,5
17	[1,4]	]	]	]	]
18	[1,5]	[1,2,5	[1,2,3,6	[1,2,3,4,5	[1,2,3,4,5,6
19	[1,4]	]	]	]	]
20					

Table au annexe: Vecteurs de déplacements des couches du CMAC

Annexe B :

Déplacements

descouches du CMAC

La stratégie de déplacement des couches est une phase importante pour l'algorithme du réseau de neurones CMAC. Cette étape, présentée dans le chapitre2,consiste à répartir les couches(et donc toutes les fonctions de base) sur l'ensemble de l'espace d'entrée normalisé.De ce fait,cette stratégie qui influe sur l'organisation de la structure du réseau permet de définir la relation de voisinage entre les fonctions de base activé esparun stimuli.

Dans nos implémentations,nous avons utilisé la méthode de (ParksandMilitzer1991) pour positionner et déplacer les couches du réseau CMAC.Cette technique,qui est de loin la plus utilisée, est résumée dans cette annexe. Le tableau ci-dessous présente les valeurs optimales des vecteurs de déplacement **d** pour un espace d'entrée de dimensions n=2,…,6 et pour un paramètre de généralisation □ =2,…,20.C'est une version réduite de la table donnée par Parks et Militzer car, dans nos

applications, le paramètre de généralisation et la dimension des CMAC n'excède jamais ces valeurs.

□ *n*	2	3	4	5	6
2	[1,1]	[1,1,1	[1,1,1,1	[1,1,1,1,1	[1,1,1,1,1,1
3	[1,1]	]	]	]	]
4	[1,1]	[1,1,1	[1,1,1,1	[1,1,1,1,1	[1,1,1,1,1,1
5	[1,2]	]	]	]	]
6	[1,1]	[1,1,1	[1,1,1,1	[1,1,1,1,1	[1,1,1,1,1,1
7	[1,2	]	]	]	]

8	[1,3]	[1,1,3	[1,1,3,3	[1,1,1,3,3	[1,1,1,3,3,3
9	[1,2]	]	]	]	]
10	[1,3]	[1,2,4	[1,1,2,4	[1,1,2,2,4	[1,1,2,2,4,4
11	[1,3]	]	]	]	]
12	[1,5]	[1,1,3	[1,1,3,3	[1,1,1,3,3	[1,1,1,3,3,3
13	[1,5]	]	]	]	]
14	[1,3]	[1,2,4	[1,1,3,5	[1,2,3,4,5	[1,1,2,3,4,5
15	[1,4]	]	]	]	]
16	[1,3]	[1,1,5	[1,1,5,5	[1,1,1,5,5	[1,1,1,5,5,5
17	[1,4]	]	]	]	]
18	[1,5]	[1,2,5	[1,2,3,6	[1,2,3,4,5	[1,2,3,4,5,6
19	[1,4]	]	]	]	]
20					

Table au annexe: Vecteurs de déplacements des couches du CMAC