الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Mohamed Khider – Biskra

Faculté des Sciences et de la technologie

Département : **Génie Electrique**

Ref :

جامعة محمد خيضر بسكرة
كلية العلوم و التكنولوجيا
قسم:الهندسة الكهربائية.
المرجع: /2011.

Thèse présentée en vue de l'obtention
Du diplôme de

Doctorat d'Etat

Spécialité : Electronique

Identification et commande de robot manipulateur rigide et flexible en utilisant les réseaux de neurones et la logique floue

Présentée par : **Guesbaya Tahar**
Soutenue publiquement le 07/03/2012

Devant le jury composé de :

Dr. Moussi Ammar	**Professeur**	**Président du jury**	**université de Biskra**
Dr. Benmahammed Khier	**Professeur**	**Directeur de thèse**	**université de Setif**
Dr Kazar Okba	**Professeur**	**Examinateur**	**université de Biskra**
Dr Ziet Lahcene	**M.C**	**Examinateur**	**université de setif**
Dr Saigâa Djamel	**M.C**	**Examinateur**	**université de M'Sila**

Résumé

Cette thèse consiste en l'étude de l'utilisation des réseaux de neurones et de la logique floue en robotique. Pour cela les thèmes, robot manipulateur, réseaux de neurones, robotique mobile et logique floue sont exposés.
La présentation du robot manipulateur inclus sa modélisation géométrique, cinématique et dynamique. La simulation de la commande de robot manipulateur, par des méthodes classiques telles que PID, couple-calculé, montre leurs incapacités à cause de leurs caractères locaux. Ce qui nécessitera de trouver d'autres techniques de contrôle et de génération de trajectoires plus adaptées aux exigences : grandes vitesses, fortes précisions et modèles à paramètres variants.

Comme les réseaux de neurones peuvent êtres une alternative au commandes classiques, on a étudié plusieurs types. On a exposé leurs domaines d'applications, et les méthodes d'apprentissage adéquates. Un ensemble de schémas de commandes par réseaux de neurones tel que contrôle par modèle inverse sont discutés. Certains schémas sont dédiés au contrôle de bras manipulateurs.

Un intérêt est donné à la catégorie robot mobile. Les architectures et modèles cinématiques sont présentés. Une méthode réactive locale pour la génération de trajectoire avec évitement d'obstacle est simulée et donne de bons résultats lorsque on lui associé des solutions heuristiques en cas de minima locaux.

Le contrôle flou comme une deuxième alternative est mieux adaptée en robotique mobile, sur tout le contrôle par comportements flous cascadés. La stratégie, tiendra compte du critère multicomportements des tâches robotiques, donc considère la concurrence et l'antagonisme. Les résultats de simulation prouvent clairement l'intérêt de se genre de contrôleurs.

Abstract

The studies in this thesis are about the use of neural networks and fuzzy logic in robotics. The themes exposed are robot manipulator, neural networks, mobile robotics and fuzzy logic.

The presentation of robot manipulator includes the cinematic, the geometric and the dynamic modulation. The control of robot manipulator by classical methods (PID, computed torque) is not efficient when fast speeds and several precisions are needed. Searching new methods to get round the defaults of the classical ones is necessary.

When looking for the neural networks like alternative to robot control, we have studied some kinds of them. We have exposed their application domains, the adequate learning methods. We have discussed some neural control schemes like control by rivers model. Some schemes are dedicated to robot manipulator.

Some interest is given to the mobile robot by presenting the architectures and cinematic model. Constrained method, like reactive local method for generating trajectory with avoidance of obstacles, is simulated and gives good results. The local minima causes blocking situation for the mobile robot, but it's surpassed using heuristic rules.
The Fuzzy control as second alternative is better for the mobile robot and principally the cascade behavior fuzzy controllers. The fusion and the grinding of behaviors decision take in consideration the concurrence and the antagonism between them. The results of simulation are interesting that justified the use of this kind of controllers.

ملخص

هذا البحث يعنى بالخلايا العصبونية والمنطق الغامض وتطبيقها في مجال الربوت. ولهذا تمت دراسة مواضيع، الربوت المعالج ،الخلايا العصبونية، الربوت المتحرك والمنطق الغامض.
تقديم الربوت المعالج يشمل النموذج الهندسي، النموذج الحركي والنموذج الديناميكي. إن محاكاة التحكم بالربوت بالطرق الكلاسيكية مثل' pid ' او 'العزم المحسوب' أثبت عجزا في الحالات المتقدمة بسبب الموضعية المنسوبة لها. ولهذا أستوجب إيجاد طرق جديدة أكثر ملائمة للتحكم ورسم المسارات تناسب استعمال الربوت لسرعات عالية ومهام بالغة الدقة وتتحمل تغيرات أو إضطربات في النموذج.
لأن الخلايا العصبونية يمكن أن تكون بديلا للطرق الكلاسيكية للتحكم، قمنا بدراسة عدة أنواع ،حيث تطرقنا إلى مجالات تطبيقها، أشكالها الهندسية وطرق تعلمها المناسبة. لقد تمت كذلك مناقشة عدة رسوم تحكم بواسطة الخلايا العصبونية ،ومنها التحكم بالنموذج المقلوب. وقدمنا عدة رسوم لأجل الربوت المعالج.
أهمية معتبرة وجهت في هذه الرسالة للربوت المتحرك، حيث عرضنا أشكاله الهندسية ونماذجه الحركية. كما تم محاكاة طريقة تحكم تفاعلية موضعية من أجل إنجازمسارحركي للربوت مع تفادي الإصطدام بالموانع ، وقد أبان ذلك على نتائج جد حسنة إلا في حالات الجمود الحركي حيث تم اللجوء إلى حلول افتراضية.
التحكم باستعمال المنطق الغامض يعتبر بديل أفضل ملائمة للربوت المتحرك؛ وخاصة "التصرفات الغامضة المتتالية " مع مراعاة تنافسها وإمكانية تنافيها. ولقد أظهرت المحاكاة نتائج جد مقنعة بتفادي أنواع الموانع والوصول إلى الهدف دون حصول أي جمود حركي.

DEDICASSE

A la Mémoire de mon Père

A ma mère

A ma petite famille

REMERCIEMENTS

Je tiens à remercier mon directeur de thèse Pr. Khier Benmahammed pour ses directives et conseils consistants pour mener à bien ce travail de recherche. Je remercier aussi mon codirecteur à l'étranger Pr. Benali Abderraouf du laboratoire LVR pour son accueil et aide.

Je remercier le Pr. Moussi Ammar d'avoir accepté de présider le jury de soutenance, comme je remercier le Pr. Kazar Okba, le Pr. Saigâa Djamel et le Pr. Ziet Lahcene d'avoir accepté d'examiner ce travail de recherche.

Je remercier aussi le Pr. Mimoun Souri mohammed de même le Pr. Allag Abdelkarim.

Je remercier tout personne m'ayant aidé de prés ou de loin.

SOMMAIRE

CHAPITRE 3 : Les Réseaux de Neurones pour le Contrôle

CHAPITRE 4 : Les Réseaux de Neurones pour le Contrôle de Robot manipulateurs

CHAPITRE 5 : Robotique Mobile

CHAPITRE 6 : Planification de trajectoire pour robot mobile avec évitement d'obstacles en utilisant la méthode des contraintes

CHAPITRE 7 : La logique floue

CHAPITRE 8 : Planificateur de trajectoire basé sur les comportements flous pour robot mobile

LISTE DES FIGURES

CHAPITRE 1

CHAPITRE 2

CHAPITRE 3

CHAPITRE 4

CHAPITRE 5

CHAPITRE 6

CHAPITRE 7

CHAPITRE 8

HAPITRE 9

LISTE DES TABLEAUX

Introduction générale

L'apparition des robots manipulateurs est pulsée par le fait d'une ère où la fabrication est devenue en chaîne, ce qui a exigé de longues heures de travail répétitif et pénible. Quand à ces dernières décennies des tâches plus complexes nécessitant des déplacements dans des milieux non permets à l'être humain (nucléaire, mine, militaire, espace, …etc.), ont favorisé le robot mobile de s'installer.

Le développement rapide de l'industrie a invoqué le perfectionnement des robots manipulateurs. Alors le robot doit manipuler avec des vitesses importantes et des précisions de plus en plus accrues. Cela exige des structures mécaniques plus adaptées mais aussi de nouvelles techniques de commande mieux adaptées.

Les chercheurs non pas cesser ces deux dernières décennies d'investiguer les différents axes qui traitent la robotique. Ces recherches se distinguent par les différents angles avec les quels sont abordées les préoccupations envers la robotique qu'on peut classer comme suit :

1. Modélisation : La plus part du temps le modèle du robot (manipulateur ou mobile) est nécessaire pour réaliser une commande par exemple. Mais parfois le modèle peut être non linéaire et couplet, même à paramètres variables, ce qui exige de s'orienter vers les méthodes de modélisation et d'identification non linéaires aux lieu de se contenter des formalismes de Lagrange et d'Euler pour le modèle dynamique [1] [2].
2. Planification des tâches : il s'agit des stratégies qui permettent de gérer un ensemble d'opérations constituants une tâches. On parle de la coordination entre ces opérations et leur réalisation. La planification est globale si on possède d'avance toutes les informations sur l'environnement et souvent statiques. Parmi ces architectures celle de Paytron [9], de Brooks [10], et d'Arkin [11]. La planification est locale si le mouvement ne peut être prévu d'avance ce qui exige des réflexes réactifs [3].
3. Génération de trajectoire : la génération de trajectoire que doit parcourir un robot et soit réalisée hors ligne ou en ligne.

* Méthodes hors ligne :

_ Optimisation, utilisé par W.F. Corriker [12] et R.La [7] on se basant sur le recuit simulé, Alors que Zhao [13], Chen et Zolzala [14] ont utilisé les algorithmes génétiques.

_Bande élastique : Technique utilisé S.Quinlan et O.khatib [16] et aussi par Brock et Khatib [17], qui consiste à élaborer une trajectoire approchée puis adaptée suivant les situations comme une bande élastique.

* Méthodes en ligne : Parce que l'environnement n'est pas connu d'avance les méthodes sont réactives. Parmi ces méthodes on cite le champ de potentiel proposée par O.Khatib.

4. La commande : C'est l'étape qui génère le signal de commande à envoyer aux actionnaires du robot pour assurer la trajectoire à suivre. Parmi ces commande en cite la commande PID, la commande adaptative [4], la commande par retour linéarisant [15],[5], la commande robuste [6].
5. L'identification : Des méthodes classiques tel que les moindres carrés sont utilisées si on suppose que le système est linéaire ; ce que n'est pas le cas, donc recours aux méthodes d'approximation non linéaires.

Alors comme alternative aux méthodes classiques pour la commande, l'identification et la planification de trajectoires avec évitement d'obstacle on s'intéressera particulièrement à la logique floue et les réseaux de neurones, ayant des particularités d'approximation des non linéarités et comme des éléments de l'intelligence artificielle.

L'objectif de cette thèse qui se situe à l'intersection des domaines de la robotique et de la commande nous incite à organiser ce travaille comme ci-dessous.

Le robot manipulateur est présenté au chapitre 1, en donnant un ensemble de définitions et de méthodes de modélisation cinématique, géométrique et dynamique. Par des exemples de simulation on prouvera les limitations des commandes classiques basées partiellement (PID) ou totalement (couple calculé) sur le modèle dynamique qui n'est jamais exactement estimé.

Au chapitre 2 une étude théorique sur les réseaux de neurones a permis de revenir sur leur historique, leurs différents modèles, leurs règles d'apprentissages et leurs domaines d'applications.
Vu que les capacités des réseaux de neurones en apprentissage, en adaptation et en robustesse sont importantes, et ayants aussi la possibilité de modéliser des systèmes fortement non

linéaires ; nous avons consacré le chapitre3 à l'étude des opportunités variées qu'offrent les réseaux de neurones en commande.

Au chapitre 4 et à cause de l'insuffisance des commandes classiques, une investigation sur un ensemble de schémas de commande utilisant les réseaux de neurones est réalisée vis-à-vis du robot manipulateur ; Cela pour montrer l'existence d'autres alternatives plus efficaces et plus robustes.

Un autre volet à étudier dans cette thèse est la robotique mobile. C'est au chapitre5 qu'on donne par une étude générale l'historique de la robotique mobile, les différents types de robots (suivant leur moyen de locomotion) et la modélisation des différentes structures de robots mobiles roulants. Entre autre une étude sur les stratégies globales et locales de planification de trajectoires est présentée. Cette étude permettra de bien se comporter avec les robots mobiles.

C'est au chapitre6 qu'une méthode de planification de trajectoires avec évitement d'obstacle appelée méthode des contraintes qu'été simulée. La simulation porte sur les capteurs à ultrason, la détection d'obstacles de formes convexes et l'évitement d'obstacles pour atteindre une cible fixe ou mobile. La méthode prouva son efficacité comme méthode réactive. Mais comme handicaps, des situations de blocages se présentent à cause des minima locaux ; et des solutions heuristiques sont apportées.

L'incapacité, de la plus parts des méthodes locales (champs de potentiel, méthodes des contraintes,…etc.), envers les tâches complexes en robotique mobile, exigera de tourner vers les nouvelles techniques plus intelligentes et plus robustes. Pour cela au chapitre7 des rappelles sur la logique floue et le contrôle flou sont exposés.

Par le chapitre8 on montrera que malgré la complexité des tâches de la robotique mobile, il est possible de les subdiviser en sous tâches plus simples appelées comportements. Nous prouverons aussi que le contrôle flou peut supporter de telle complexité, en suggérant des contrôleurs à comportements flous cascadés réalisants une fusion pondérée des différentes décisions, toute en respectant une stratégie globale de planification de trajectoires. Un contrôleur flou en cascade sera utilisé en simulation pour planifier la trajectoire d'un robot mobile tricycle dont le but d'éviter une collision avec des obstacles de différentes natures et d'atteindre la cible.

On achèvera ce travail par une conclusion générale qui résumera les déductions faites envers les profils des nouvelles techniques pour la robotique de nos jours ; de même on estimera les perspectifs pour les chercheurs.

Bibliographie

[1] M.Egerstedt, X. Hu and A. Stotsky, 'Control of Car-Like Robot Using a Dynamique Model', Royal Institute of Technologie SE-100 44 Stockholm, Sweden, 2004.

[2] Corrado Guarino, Lo Bianco, Aurelio Piazzi, and Massimo Romano,' Smooth Motion Génération for Unicycle Mobile Robots Via Dynamique Path Inversion', IEEE,Transaction on robotics Vol.20 No.5 2004.

[3] T.H.Lee, H.K.Lam, F.H.F.Leung, P.K.S.Tam, 'A Fast Path Planning-and-Tracking Control for Wheeled Mobile Robots', ICRA, IEEE, 2001.

[4] W.E. Dixon, M.S. de Queiroz, D.M.Dawson, and T.J. Flynn, 'Adaptive Tracking and regulation of Wheeled Mobile Robot With Controller/Update Law Modularity', IEEE, TCST, Vom.12. No.1 2004.

[5] D. Benmerzouk, N. Gouali, 'Analyse de la commande dynamique des robots a axes rigides', Journal des Techologies Avancées No.12, janvier 2000.

[6] H.G. Sage, M. F. De Mathelin and E. Ostertag, 'Robust control of robot manipulators: a survey', Int. J. Control , Vol.72 ,No. 16,1999.

[7] R.La, F.Guély, and P.Siarry ,'Apprentissage d'une base de régles floues par la méthode du recuit simulé', In Troisièmes journées Nationales: Les applications des ensembles flous, Nimes, France, 1993.

[9] D .W.Payton, 'An architecture for reflexive autonomous vehicle control', In Proc of the IEEE Int. Conf. on Robotics and Automation, volume 3, pages 1838-1845, San Francisco, CA(US), 1986.

[10] R.A. Brooks, 'A robust layred control system for mobile robot'. IEEE Transactions Robotique and Automation, Mars 1986.

[11] R.C. Arkin. , 'Motor schema-based mobile robot navigation', The International Journal of Robotics Research, 1987.

[12] W.F Carriker, P.K Kholsa et B. H. Krogh., 'Path planning for mobile manipulator for multiple task execution', IEEE, 1995.

[13] M.Zhao, N. Ansari et E.S.H. Hou, 'Mobile manipulator path planning by a genetic algorithm', Journal of Robotic System, 1994.

[14] M. Chen et A. M. S. Zalzala, 'A genetic approach to motion planning of redundant mobile manipulator systems considering safety and configuration. JRS, 1997.

[15] Costase S. Tzafestas, Spyros G. Tzafestas, 'Full-State modelling, Motion Planning and Control of Mobile Manipulators', National technical University of Athens 2001.

[16] S. Quinlan, O. Khatib, ‘Elastic Bands: connecting path and control’, international conference on robotics and automation, Atlanta, GA, USA, IEEE, 1993.

CHAPITRE 1

Robot manipulateur

1. Robot manipulateur

1.1 Introduction

Le robot manipulateur est devenu une nécessité du fait que l'industrie manipule des objets lourds de façon répétitive et dans des milieux hasardeux. L'ensemble des recherches ont mené à des bras de toute tailles et poids, de toutes vitesses et précisions, et adaptés aux tâches confiées.

1.2 Anatomie d'un robot

Les interactions entre un système mécanique articulé et un ensemble d'organes associés forment un robot manipulateur. Le schéma synoptique de la fig(1.1) montre les différentes parties d'un robot.

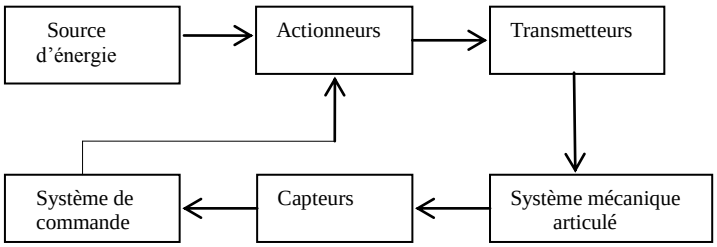


Figure 1.1 Schéma synoptique d'un robot manipulateur

1.3 Capteurs en robotique

Les capteurs en robotique recueillent, des informations sur la situation du robot tel que, la position et la vitesse des articulations, ou des informations sur son environnement par des caméras et des capteurs ultrasons.

1.4 Degrés de liberté d'un solide indéformable

Le positionnement et l'orientation d'un solide indéformable dépendent de ses degrés de liberté (d.d.l). Pour décrire la situation d'un solide, on lui associe un repère cartésien oxyz. La translation du solide le long de ox,oy et oz assure son positionnement à n'importe quel point de l'espace. De même sa rotation autour de ox, oy et oz permet de l'orienter dans n'importe quelle direction.

1.5 Les transformations homogènes

Les transformations homogènes sont utilisées en robotique pour faciliter la représentation d'un quelconque point du robot dans un repère et le passage systématique d'un point à un autre [2].

Le passage d'un repère R_n à un autre repère R_m se fait par translation et/ou rotation. Une matrice ${}^{n}T_{m}$ dite de transformation permet d'assurer les opérations précédentes. La matrice ${}^{n}T_{m}$ est définie comme suit :

$$ {}^{n}T_{m} = [S\ N\ A\ P] \tag{1.1}$$

avec

$$S = [s_x\ s_y\ s_z\ 0]^T, N = [n_x\ n_y\ n_z\ 0]^T\ ,\ A = [a_x\ a_y\ a_z\ 0]^T,\ P = [p_x\ p_y\ p_z\ 1]^T$$

Où P représente les coordonnées de l'origine du repère R_m exprimé dans R_n et S, N, A représentent les vecteurs unitaires suivant X_m,Y_m, Z_m du repère R_m exprimé dans R_n.

La matrice ${}^{n}T_{m}$ peut être représenté comme suit :

$$ {}^{n}T_{m} = \begin{bmatrix} {}^{n}Rot_{m} & {}^{n}P_{m} \\ 0 & 1 \end{bmatrix} \tag{1.2}$$

Où ${}^{n}Rot_{m}$ est la matrice d'orientation et ${}^{n}P_{m}$ vecteur de position. Cette représentation matricielle nous permet d'écrire la relation suivante : ${}^{0}T_{j} = {}^{0}T_{1}\,{}^{1}T_{2} \ldots {}^{j-1}T_{j}$.

1.6 Modélisation géométrique d'un bras manipulateur

Le modèle géométrique d'un robot manipulateur permet de représenter les coordonnées (position et orientation) de son effecteur en fonction des coordonnées articulaires (angles des articulations) [2],[5],[6].

Pour calculer le modèle géométrique d'un bras manipulateur un ensemble de règles doit être considéré :

- Les corps du robot sont supposés rigides.
- Les articulations sont sans jeu mécanique et sans élasticité.
- L'axe Z_j du repère R_j est porté par l'axe de l'articulation j.
- La variable de l'articulation j est notée q_j.

Le robot manipulateur est considéré comme un système mécanique articulé composé de n+1 corps C_0, C_1, C_2,…, C_n, donc de n articulations. Alors la détermination de la matrice de

transformation permettant le passage d'un corps C_i au corps C_{i+1} nécessite la connaissance des valeurs des variables α, q, d et r définie ci-dessous.

Le choix du repère R_j fixé au corps C_j est défini de la façon suivante:

∗ L'axe Z_j est porté par l'axe de l'articulation j.

∗ L'axe X_j est porté par la perpendiculaire commune aux axes Z_j et Z_{j+1}.

Le passage de R_{j-1} à R_j s'exprime, comme montré en figure 1.2, en fonction de quatre paramètres:

α_j : Angle entre les axes Z_{j-1} et Z_j correspondant à une rotation autour de X_{j-1}.

d_j : Distance entre Z_{j-1} et Z_j le long de X_{j-1}.

q_j : Angle entre les axes X_{j-1} et X_j correspondant à une rotation autour de Z_j.

r_j : Distance entre X_{j-1} et X_j le long de Z_j.

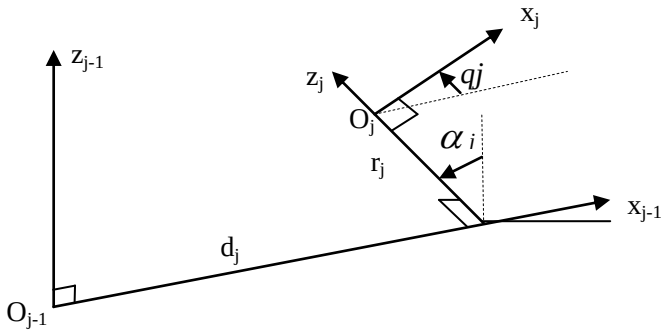


Figure 1.2 Paramètres géométriques de passage

Donc la matrice de passage du repère R_{j-1} à R_j aura pour expression :

$$ {}^{j-1}T_j = \begin{bmatrix} Cq_j & -Sq_j & 0 & d_j \\ C\alpha_j Sq_j & C\alpha_j Cq_j & -S\alpha_j & -r_j S\alpha_j \\ S\alpha_j Sq_j & S\alpha_j Cq_j & C\alpha_j & r_j C\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3} $$

Obtenir le modèle géométrique c'est de déterminer les coordonnées opérationnelles de l'effecteur en fonction des coordonnées généralisées. Par l'exemple suivant on peut voir comment est calculé le modèle géométrique en utilisant la matrice de transformation ${}^{j-1}T_j$.

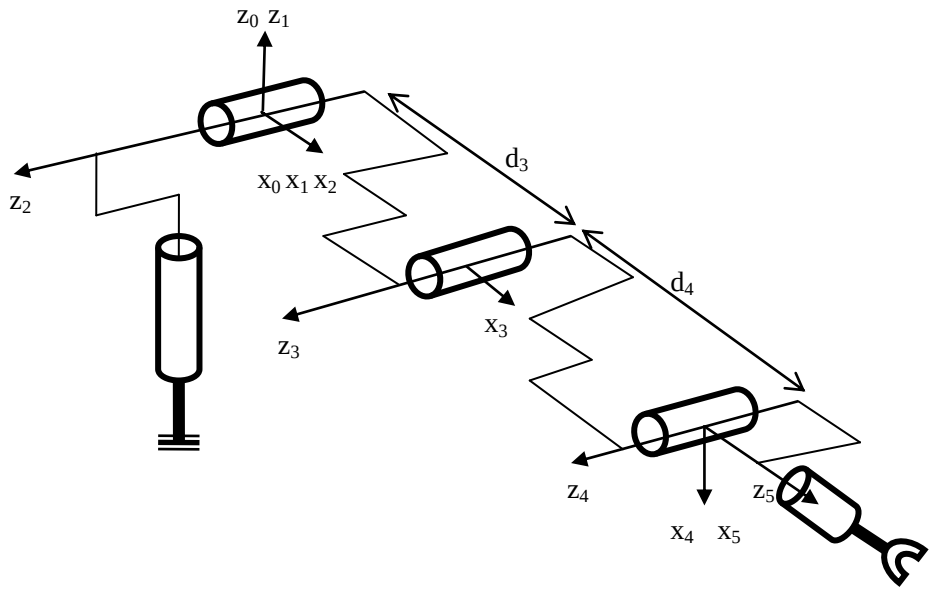


Figure 1.3 Association des repères aux articulations

D'après les définitions précédentes, les paramètres de chaque articulation et on regardant la figure 1.3, sont donnés par le tab 1.1 suivant :

j	α_j	d_j	q_j	r_j
1	0	0	q_1	0
2	90	0	q_2	0
3	0	d3	q_3	0
4	0	d4	q_4	0
5	-90	0	q_5	0

Tableau 1.1 Paramètres des articulations

Le modèle géométrique qui détermine l'orientation et la position du repère R_5 dans le repère de base R_0 est donné par la matrice de passage ${}^0T_5$. A savoir que ${}^0T_5 = {}^0T_1\,{}^1T_2\,{}^2T_3\,{}^3T_4\,{}^4T_5$.

Le calcul nous donne les résultats suivants :

$$ {}^0T_5 = \begin{bmatrix} C_1C_5C_{234} - S_1S_5 & -C_1S_5C_{234} - S_1C_5 & -C_1C_{234} & C_1(C_{234}d_4 + C_2d_3) \\ S_1C_5C_{234} + C_1S_5 & -S_1S_5C_{234} + C_1C_5 & -S_1S_{234} & S_1(C_{234}d_4 + C_2d_3) \\ C_5S_{234} & -S_5S_{234} & C_{234} & S_{23}d_4 + S_2d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (1.4) $$

Avec $C_n = Cos(q_n)$, $S_n = Sin(q_n)$, $S_{nml} = Sin(q_n + q_m + q_l)$ et $C_{nml} = Cos(q_n + q_m + q_l)$

L'identification de l'équation (1.4) avec (1.2) donne la position et l'orientation de l'effecteur par rapport au repère de référence R_0.

Inversement au modèle géométrique direct, le modèle géométrique inverse donne les coordonnées articulaires en fonction des coordonnées opérationnelles.
Le besoin au modèle géométrique inverse vient du fait qu'on doit réaliser des commandes dans le domaine opérationnel. Vu qu'il n'existe pas de solution analytique générale, certaines méthodes sont utilisées, tel que la méthode de PIEPER et celle de PAUL.

1.7 Modélisation cinématique d'un bras manipulateur

Le modèle cinématique se base sur le calcul variationnel, il permet de décrire les variations élémentaires des coordonnées opérationnelles en fonction des variations élémentaires des coordonnées articulaires.
Le modèle cinématique direct est décrit par l'équation : $dX = J(q)dq$ où $J(q)$ est appelé Jacobien. Des méthodes existent pour le calcul de $J(q)$, parmi les quelles on cite la méthode de propagation de la vitesse et la méthode du Jacobien de base [1], [2], [9].

Le modèle cinématique inverse est utile chaque fois qu'on veut calculer à partir d'une configuration donnée, la différentielle dq pour satisfaire une différentielle des coordonnées opérationnelles dX. L'intérêt du modèle cinématique inverse réside dans sa linéarité. Le calcul du modèle cinématique inverse est obtenu soit par la dérivation du modèle géométrique inverse soit par l'inversion du modèle cinématique direct [3].

1.8 Modèle dynamique d'un bras manipulateur

Le besoin aux performances dynamiques du robot manipulateur exige la connaissance de son modèle dynamique. L'étude dynamique consiste à déterminer les forces ou les couples à appliquer aux articulations en considérant les effets centrifuge, coriolis, de gravité et de frottement [10], [12], [13]. Le modèle dynamique est décrit par l'équation suivante :

$$\tau = D(q)\ddot{q} + H(q,\dot{q}) + G(q) + \tau_f \qquad (1.5)$$

Avec :

D : Matrice d'inertie du robot, symétrique définie positive.

H : Matrice des termes centrifuges et coriolis.

G : Vecteur des forces de gravité.

τ_f : Vecteur des forces de frottement.

τ : Vecteur des couples à appliquer aux articulations.

q : Vecteur des coordonnées articulaires.

L'équation (1.8.1) est déterminée en utilisant soit le formalisme mécanique de Lagrange-Euler soit de Newton-Euler.

1.9 Association de moteurs à courants continus

Le robot manipulateur qu'on veut commander est constitué de plusieurs sous-systèmes à savoir : la structure mécanique articulée, les actionneurs, les organes de transmissions et les capteurs. Quelle sera donc l'équation du modèle dynamique du robot si le modèle des moteurs est incorporé.

La figure 1.4 ci-dessous nous montre le schéma bloc du moteur :

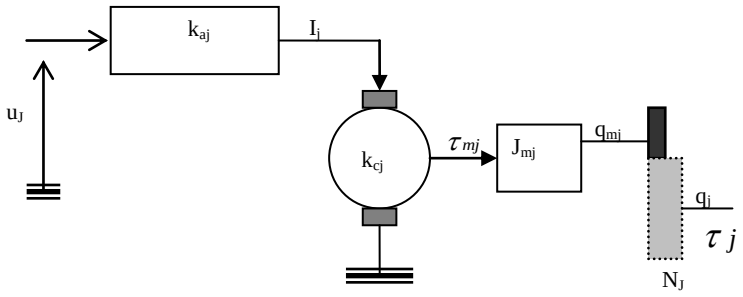


Figure 1.4 Schéma bloc du moteur

Avec :

u_j : Signal de commande à l'entrée de l'amplificateur J.

ka_j : Gain de l'amplificateur j.

I_j : Courant du moteur j.

kc_j : Gain en couple du moteur j.

J_{mj} : Inertie propre du moteur j ramenée à l'arbre moteur et augmenter de l'inertie des organes de transmission.

q_{mj} : Position angulaire de l'axe du moteur j.

q_j : Position angulaire de l'articulation j.

N_j : Rapport de réduction de transmission du moteur j.

τ_j : Couple transmit à la j^{eme} articulation.

τ_{mj} : Couple moteur du moteur j.

L'équation fondamentale de la dynamique pour un mouvement de rotation s'écrit pour le cas des moteurs du robot:

$$\tau_m = J_m.\ddot{q}_m + \frac{1}{N}\tau \tag{1.6}$$

Avec :

$J_m = diag[J_{m1}, J_{m2}, ...J_{mn}]$: Représente la matrice des inerties propres aux moteurs ramenées côté arbre moteur, augmentées de l'inertie des organes de transmission.

$q_m = [q_{m1}, q_{m2}, ...q_{mn}]^T$, Est le vecteur des positions angulaires du rotor par rapport au stator.

$\tau_m = [\tau_{m1}, \tau_{m2}, ...\tau_{mn}]^T$, Est le vecteur des couples moteurs de chaque articulation.

En considérant l'équation électrique du moteur à courant continu à excitation séparée on peut écrire pour une articulation *j* les équations suivantes:

$$I_j = ka_j.u_j \tag{1.7}$$

$$q_{mj} = N_j.q_j \ ; \ \dot{q}_{mj} = N_j.\dot{q}_j \ ; \tag{1.8}$$

$$\tau_{mj} = kc_j.I_j \tag{1.9}$$

$$\tau_{mj} = kc_j.ka_j.u_j = J_{mj}.\ddot{q}_{mj} + \tau_j / N_j \tag{1.10}$$

Vu que :

$$\ddot{q}_{mj} = N_j.\ddot{q}_j \tag{1.11}$$

Et en posant : $M_j = N_j^2 J_{mj}$

On peut écrire :

$$N_j.kc_j.ka_j.u_j = N_j^2.J_{mj}.\ddot{q}_j + \tau_j = M_j.\ddot{q}_j + \tau_j \tag{1.12}$$

Pour l'ensemble des articulations et sous forme matriciel :

$$N.kc.ka.u = M.\ddot{q} + \tau \tag{1.13}$$

L'équation du modèle dynamique (1.5) intégrée dans l'équation (1.13) donne l'équation du modèle dynamique en considérant les paramètres mécaniques et électriques des moteurs:

$$N.kc.ka.u = (D(q) + M)\ddot{q} + H(q, \dot{q}) + G(q) + \tau_f \tag{1.14}$$

On remarque bien l'effet de l'inertie des moteurs et du rapport de réduction des transmissions sur la dynamique du robot. Plus le rapport de réduction est grand et l'inertie du moteur est importante plus leurs effet devient considérable, sur tout si les vitesses des articulations sont faibles.

1.10 Commande classique

Nous entendons dire par la commande classique les lois linéaires de type PID aux gains constants. Pour élaborer une commande PID, il faut considérer chaque articulation du robot comme un mécanisme indépendant dont le modèle peut être linéarisé dans une zone de fonctionnement [7]. L'équation (1.15) illustre la commande PID.

La commande classique est souvent utilisée dans le domaine industriel à cause de sa facilité d'implémentation et de son faible coût. Cette commande n'est pas acceptable dans le cas des vitesses élevées des articulations du fait de la non linéarité du modèle et de son couplage et parce que les valeurs des coefficients du régulateur sont constantes.

La loi de commande PID associe aux articulations est comme suit :

$$\tau = K_p(q_d - q) + K_d(\dot{q}_d - \dot{q}) + K_i\int_0^t (q_d - q)dt \tag{1.15}$$

Avec :

τ : Couples de commandes appliqués aux articulations.

q_d : Position angulaire désirée.

q : Position angulaire courante.

$\dot{q}_d$: Vitesse angulaire désirée.

$\dot{q}$: Vitesse angulaire courante.

K_p, K_d et K_i : matrice diagonale, contenant les gains k_{pj}, k_{dj} et k_{ij} de chaque articulation.

Le bras manipulateur doit être traité comme un système linéaire découplé, ce qui exige de négliger les forces centrifuges et les forces de coriolis. Cela ne peut être vrai que lorsque les vitesses sont faibles et les rapports de réduction des transmissions sont grandes. Ces considérations nous permettent d'écrire le modèle simplifie du robot manipulateur :

$$\tau_j = D_{jj}\ddot{q}_j \tag{1.16}$$

Avec :

D_{jj} : Partie fixe de l'élément *jj* de la matrice d'inertie D du robot manipulateur.

En considérant les équations (1.15) et (1.16) la fonction de transfère est obtenue :

$$\frac{q_j}{q_{d\,j}} = \frac{k_{dj}s^2 + k_{pj}s + k_{ij}}{D_{jj}s^3 + k_{dj}s^2 + k_{pj}s + k_{ij}} \tag{1.17}$$

On remarque bien que $\frac{q_j}{q_{d\,j}} \neq 1$, donc, la présence d'une erreur.

Alors l'équation caractéristique est écrite comme suit :

$$\Delta(s) = s^3 + \frac{k_{dj}}{D_{jj}} s^2 + \frac{k_{pj}}{D_{jj}} s + \frac{k_{ij}}{D_{jj}} \tag{1.18}$$

En robotique, la pratique la plus courante consiste à choisir les gains de manière à obtenir comme pôles dominants un pôle double réel négatif, dans le but d'obtenir une réponse sans oscillations et rapide ; l'autre pôle est choisi réel négatif loin des deux autres. Alors l'équation caractéristique est écrite comme suit :

$$\Delta(s) = (s+\beta)^2 (s+n\beta) \tag{1.19}$$

Avec les gains sont donnés par :

$$k_{pj} = D_{jj}(2n+1)\beta^2 \ , \ k_{dj} = D_{jj}(2n+1)\beta \quad \text{et} \quad k_{ij} = D_{jj} n \beta^3 \tag{1.20}$$

Tout en sachant que $\beta > 0,$ n >0.

1.10.1 Résultats de simulation de la commande classique d'un bras manipulateur

1.10.1.1 Bras à deux ddl

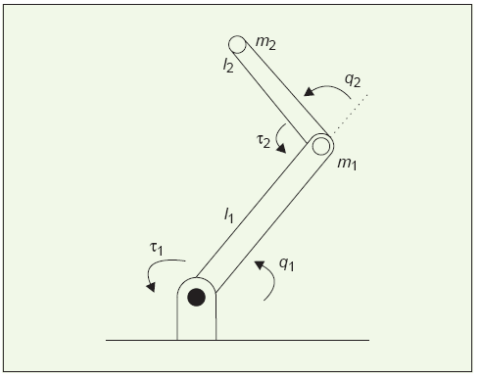


Figure 1.5 Robot manipulateur 2ddl

m_1=1kg ; m_2=1kg ; l=l_1=l_2=0.4m

Le modèle dynamique du bras manipulateur de deux d.d.l de la figure 1.5 est comme suit :

$$\begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix}=\begin{bmatrix}\frac{1}{3}m_1l^2+\frac{4}{3}m_2l^2+m_2\cos(q_2)l^2 & \frac{1}{3}m_2l^2+\frac{1}{2}m_2l^2\cos(q_2)\\ \frac{1}{3}m_2l^2+\frac{1}{2}m_2l^2\cos(q_2) & \frac{1}{3}m_2l^2\end{bmatrix}\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\end{bmatrix}+\begin{bmatrix}-\frac{1}{2}m_2l^2s\,\mathrm{in}(q_2)\dot{q}_2{}^2-m_2l^2s\,\mathrm{in}(q_2)\dot{q}_1\dot{q}_2\\ \frac{1}{2}m_2l^2s\,\mathrm{in}(q_2)\dot{q}_1{}^2\end{bmatrix}$$
$$+\begin{bmatrix}\frac{1}{2}m_1\mathrm{g}\,l\cos(q_1)+\frac{1}{2}m_2\mathrm{g}\,l\cos(q_1+q_2)+m_2\mathrm{g}\,l\cos(q_1)\\ \frac{1}{2}m_2\mathrm{g}\,l\cos(q_1+q_2)\end{bmatrix}$$

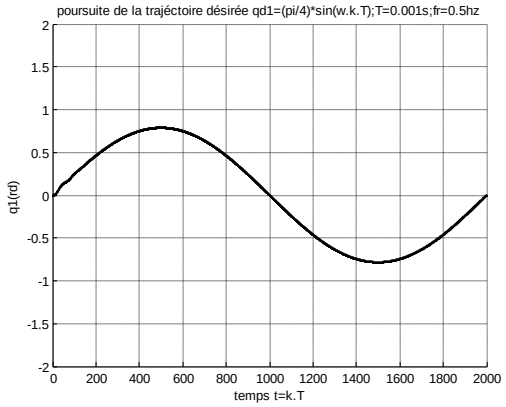
Les tests consistent à voir les possibilités de la commande classique PID à faire suivre aux articulations des trajectoires désirées ayants des vitesses et des amplitudes différentes. On teste aussi la robustesse de la commande vis-à-vis des brusques variations de la charge emportée par l'effecteur du robot.

Test1 : Poursuite de trajectoires :

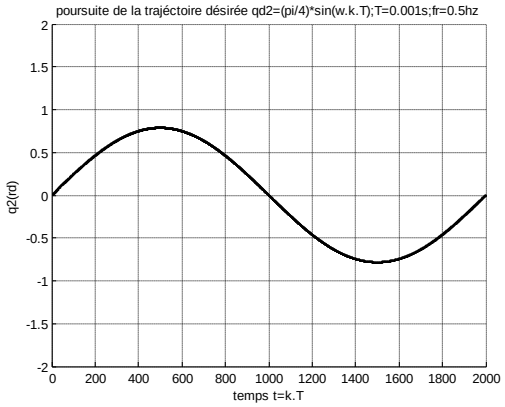
$qd1=(\pi/4)\sin(\omega kT)$; $qd2=(\pi/4)\sin(\omega kT)$; T=0.001s; fr=0.5hz

Après plusieurs essais les valeurs des gains de régulation obtenus sont tel que :

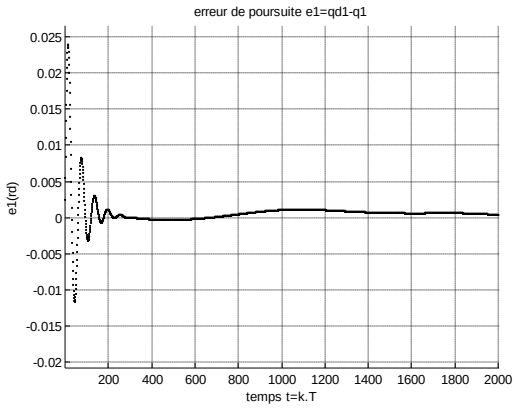
$k_{p1}=5000$ $k_{d1}=20$ $k_{i1}=5000$ $k_{p2}=5000$ $k_{d2}=20$ $k_{i2}=5000$



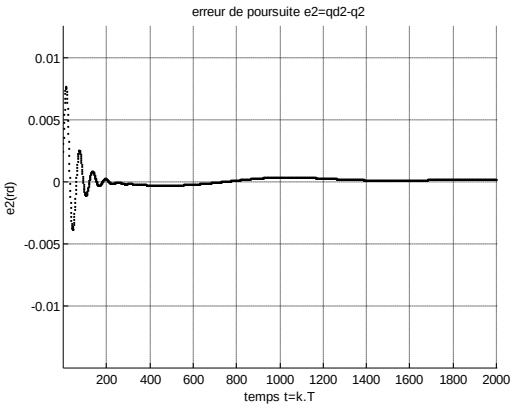
(a)



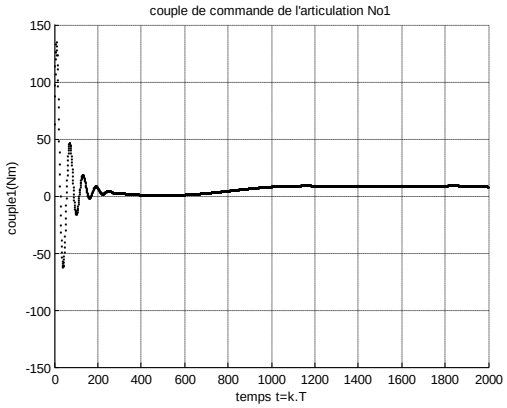
(b)



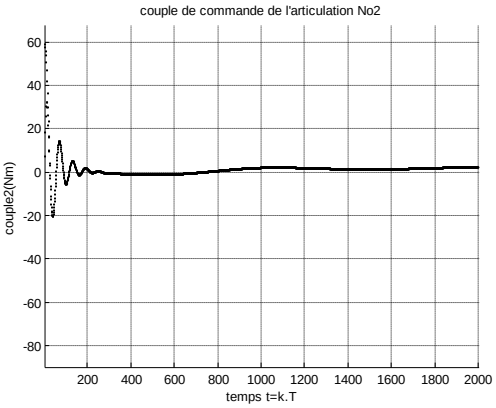
(c)



(d)



(e)

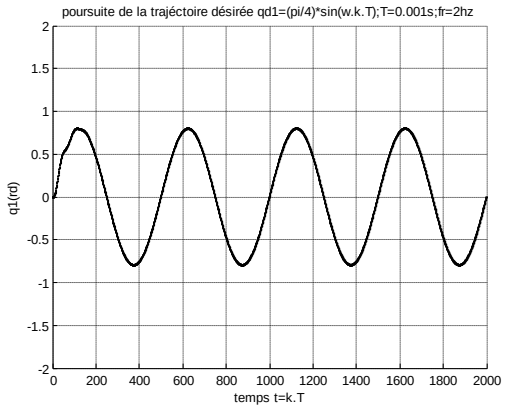


(f)

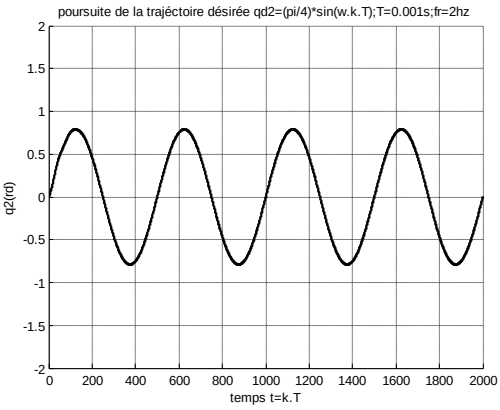
Figure 1.6.a-f Différents résultats de simulation
de la poursuite de trajectoires pour test1

Résultat1 : Les résultats montrés par la figure ci-dessus indiquent la possibilité de commander un bras manipulateur sous conditions de faible vitesse.

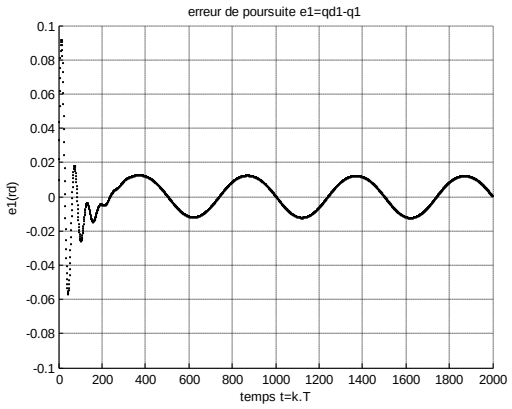
Test2: $qd1 = (\pi/4)\sin(\omega kT)$; $qd2 = (\pi/4)\sin(\omega kT)$; T=0.001s; fr =2hz



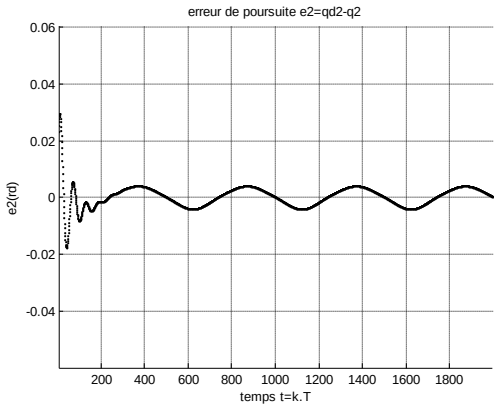
(a)



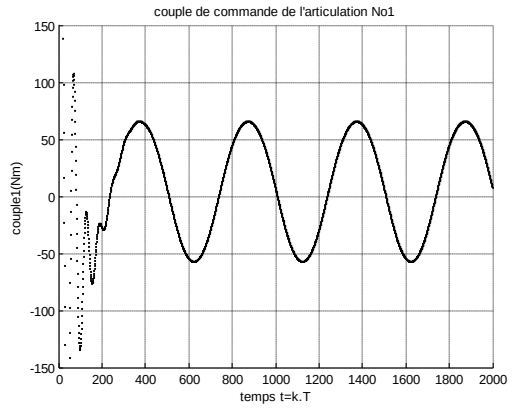
(b)



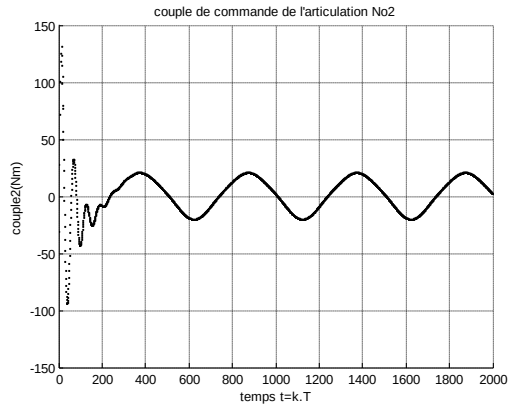
(c)



(d)



(e)



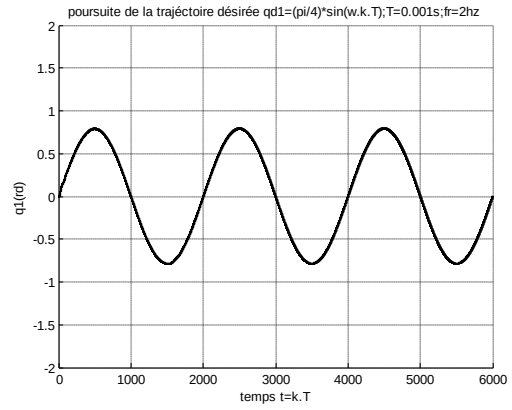
(f)

Figure 1.7.a-f Différents résultats de simulation
de la poursuite de trajectoires pour test2

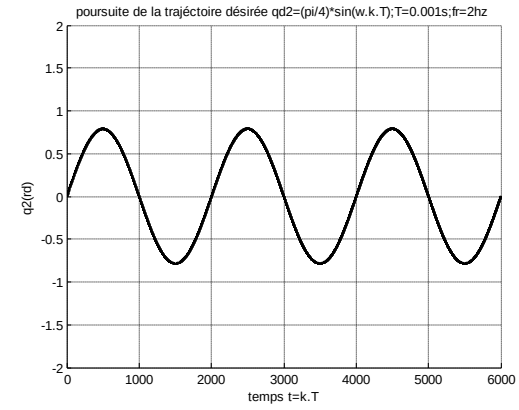
Resultat2 : On remarque bien que une augmentation de la vitesse du mouvement des articulations fait augmenter le taux d'erreur de la poursuite de même la valeur des couples.

Test3: $qd1 = (\pi/4)\sin(\omega kT)$; $qd2 = (\pi/4)\sin(\omega kT)$; T=0.001s; fr=0.5hz

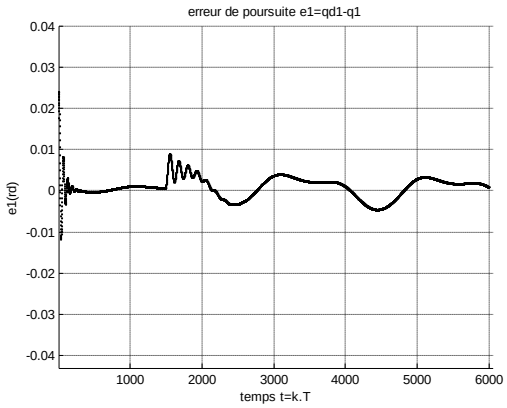
Une variation de la valeur de la masse de la charge (simulée par l'augmentation de m_2 de **1kg** à **5kg** à **t=1500T**).



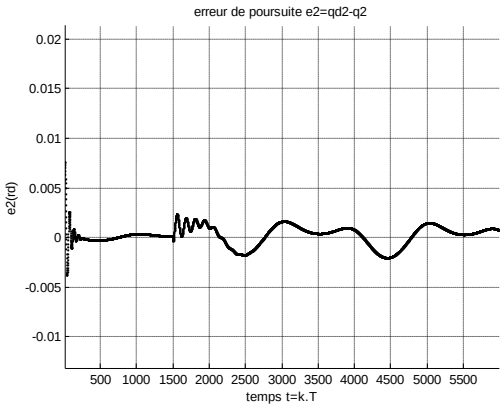
(a)



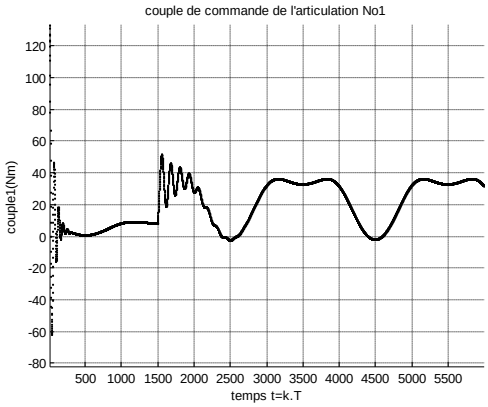
(b)



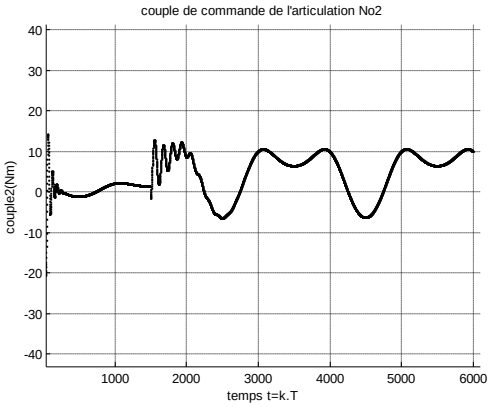
(c)



(d)



(e)



(f)

Figure 1.8.a-f Différents résultats de simulation
de la poursuite de trajectoires pour test3

Resultat3 : La variation de charge à l'instant t=1500T, de 1kg à 5kg a provoqué une augmentation considérable des erreurs de poursuites e1 et e2.

On conclusion pour les trois cas de test on peut dire que la commande PID est une commande locale qui exige le réajustement des gains k_p, k_d *et* k_i pour chaque situation.

1.10.1.2 Bras à trois DDL

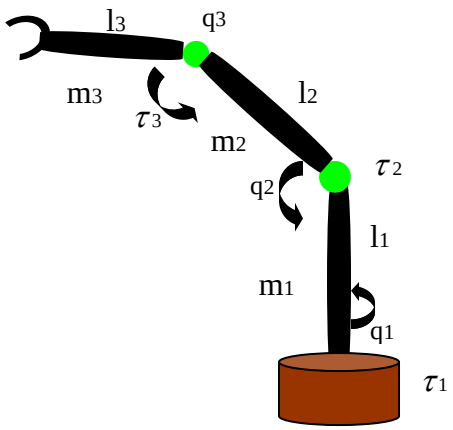
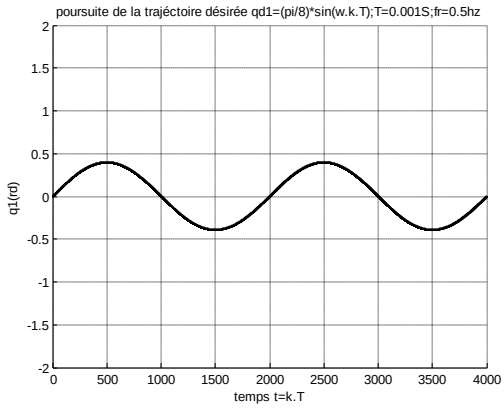


Figure 1.9 Robot à 3ddl

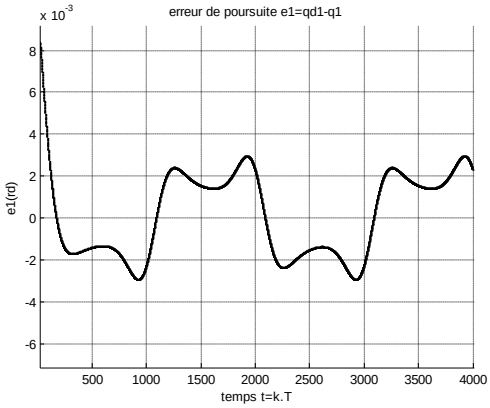
Voir Annexe D pour le modèle dynamique.

Test1: Poursuite de trajectoires :

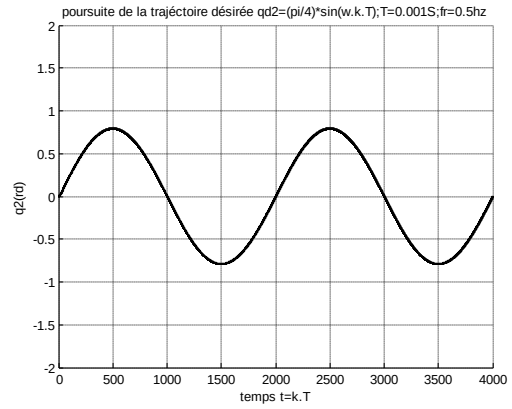
$qd1 = (\pi/8)\sin(\omega.k.T);\ qd2 = (\pi/4)\sin(\omega.k.T);\ qd3 = (\pi/2)\sin(\omega.k.T);$ T=0.001;fr=0.5 hz



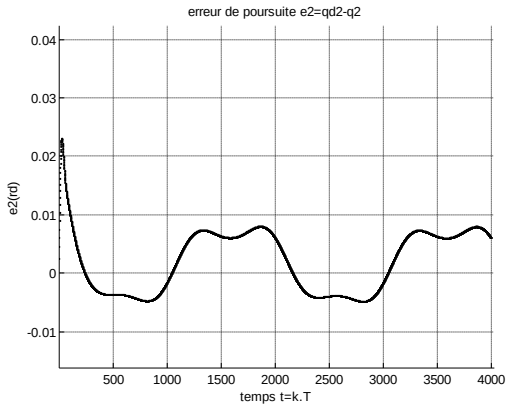
(a)



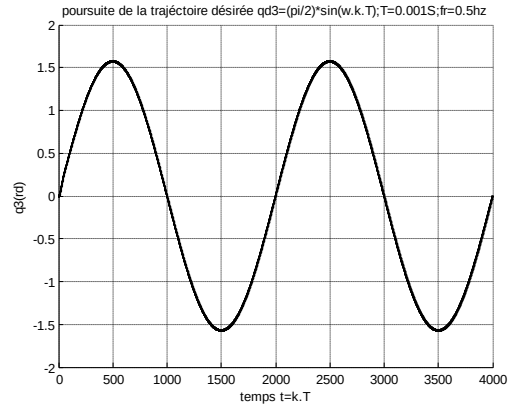
(b)



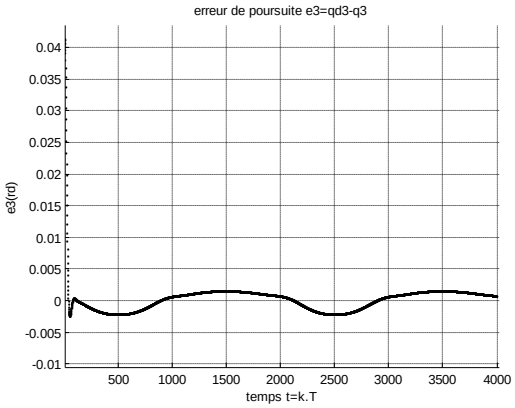
(c)



(d)



(e)

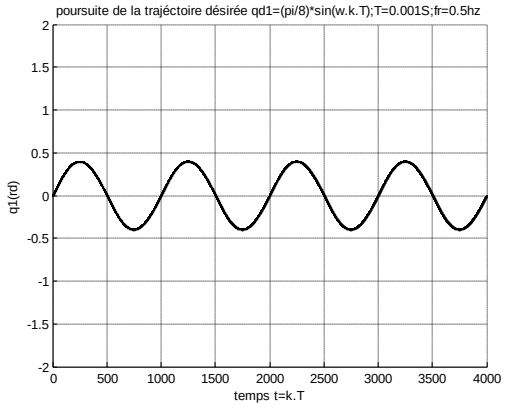


(f)

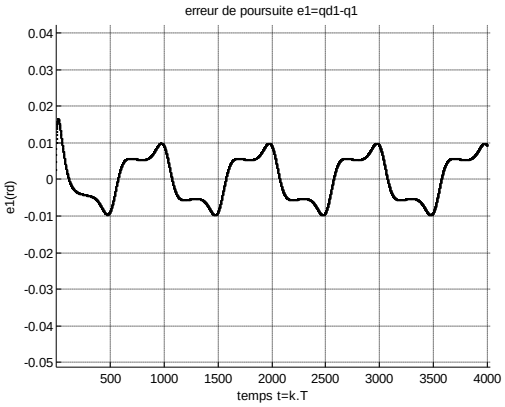
Figure 1.10.a-f Différents résultats de simulation
de la poursuite de trajectoires pour test1

Résultats du test1 : Après plusieurs essais les valeurs des gains de régulation obtenus sont tel que : $k_{p1}=100000$, $k_{d1}=12200$, $k_{i1}=1500$, $k_{p2}=100000$, $k_{d2}=12150$, $k_{i2}=1500$ $k_{p3}=100000$, $k_{d3}=1000$ et $k_{i3}=1500$ .Les erreurs sont acceptables pour ce cas d'étude.

Tes2 : Augmentation de la vitesse des articulations.
$qd1=(\pi/8)\sin(\omega.k.T)$; $qd2=(\pi/4)\sin(\omega.k.T)$; $qd3=(\pi/2)\sin(\omega.k.T)$; T=0.001;**fr=1hz** ;



(a)



(b)

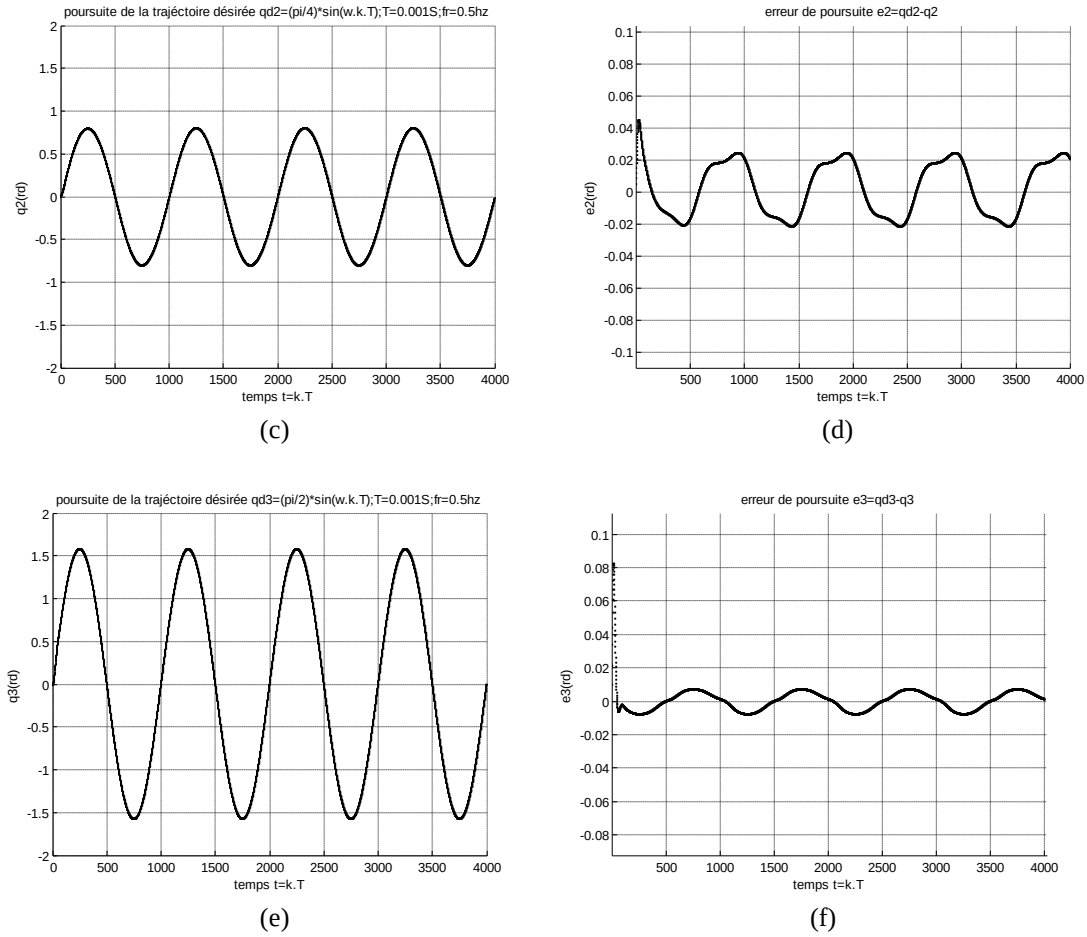


Figure 1.11.a-f Différents résultats de simulation
de la poursuite de trajectoires pour test2

Résultat du test2 : On remarque une augmentation des erreurs de poursuite par augmentation des vitesses des articulations.

En conclusion on peut bien dire que la commande classique est une commande locale. Ce qui exige d'adapter les gains k_p, k_d *et* k_i à chaque situation de fonctionnement.

1.11 La commande dynamique d'un bras manipulateur

1.11.1 Introduction

Lorsqu'un robot manipulateur doit être commandé de manière rapide et précise, les contraintes dynamiques du robot doivent êtres prises en compte, contrairement à la commande classique (pid). Alors le modèle dynamique doit être le plus complet possible et les paramètres qui le caractérisent bien identifiés [14].

1.11.2 Commande par découplage non linéaire

Cette commande basée sur le modèle dynamique consiste à transformer par retour d'état le problème d'un système non linéaire en un problème de commande de système linéaire.

Le schéma de cette commande dynamique, souvent appelé 'couple calculé' ou 'computed torque', est illustré sur la figure ci-dessous.

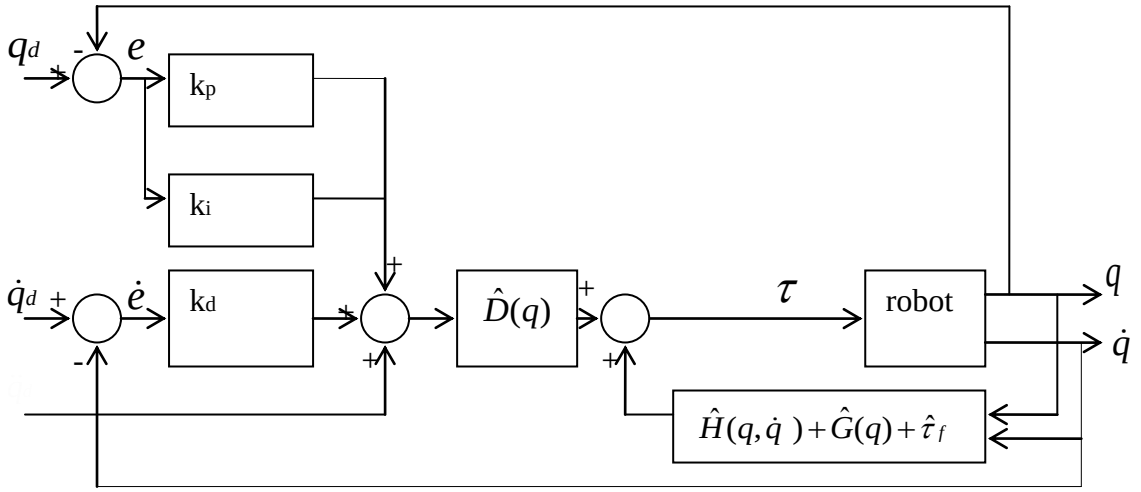


Figure 1.12 Schéma bloc de la commande 'couple calculé'

1.11.3 Résultats de simulation de la commande dynamique (couple calculé)

Si l'équation du modèle dynamique du robot est :

$$\tau = D(q)\ddot{q} + H(q,\dot{q}) + G(q) + \tau_f$$

La loi de commande sera:

$$\tau = \hat{D}(q)u + \hat{H}(q,\dot{q}) + \hat{G}(q) + \hat{\tau}_f(\dot{q}) \qquad (1.21)$$

Et

$$u = \ddot{q}_d + k_p(q_d - q) + k_d(\dot{q}_d - \dot{q}) + k_i\int_0^t (q_d - q)dt \qquad (1.22)$$

Avec $\hat{D}$, $\hat{H}$, $\hat{G}$ et $\hat{\tau}_f$ les termes estimés de H, D, G et τ_f.

Si les paramètres estimés du robot sont exacts l'équation de l'erreur devient comme suit :

$$\ddot{e} + k_d\dot{e} + k_p e + k_i\int_0^t e dt = 0 \qquad (1.23)$$

Alors cette équation est, bien sur, linéaire et de même découplée.

Lorsque les erreurs de modélisation sont importantes, que ce soit à cause d'incertitudes sur les paramètres inertiels, soit à cause des charges inconnues, soit à cause des frottements non modélisables, l'équation de l'erreur sera donnée par la relation suivante :

$$\hat{D}(q)u + \hat{H}(q,\dot{q}) + \hat{G}(q) + \hat{\tau}_f(\dot{q}) = D(q)\ddot{q} + H(q,\dot{q}) + G(q) + \tau_f$$

Soit :

$$\ddot{e} + k_d\dot{e} + k_p e + k_i\int_0^t e dt = \hat{D}^{-1}[(D-\hat{D})\ddot{q} + (G-\hat{G}) + (H-\hat{H}) + (\tau_f - \hat{\tau}_f)]$$

Ou aussi :

$$\ddot{e} + k_d\dot{e} + k_p e + k_i\int_0^t e dt = \hat{D}^{-1}\tau_{prt} \qquad (1.24)$$

$$\tau_{prt} = (D-\hat{D})\ddot{q} + (H-\hat{H}) + (G-\hat{G}) + (\tau_f - \hat{\tau}_f)$$

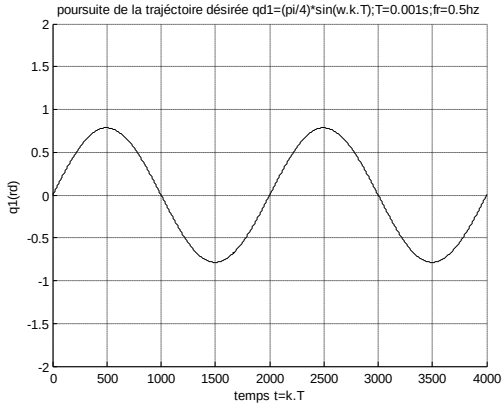
(1.25)

τ_{prt} : Couple de perturbation.

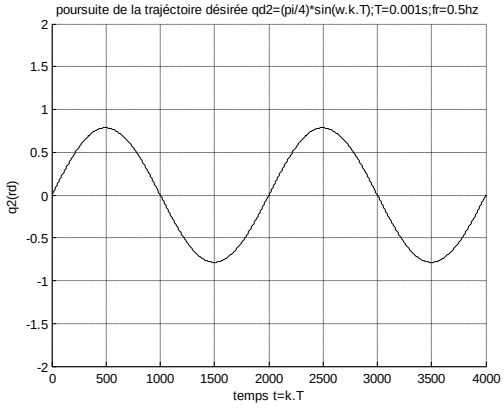
Nous déduisons que l'erreur de modélisation constitue une excitation pour l'équation de l'erreur 'e'. Remédie à ce problème c'est augmenter les gains k_p , k_d et k_i .

Test1: Par ce test en cherche a trouver les bonnes valeurs des gains k_p , k_d et k_i permettant le bon suivi :

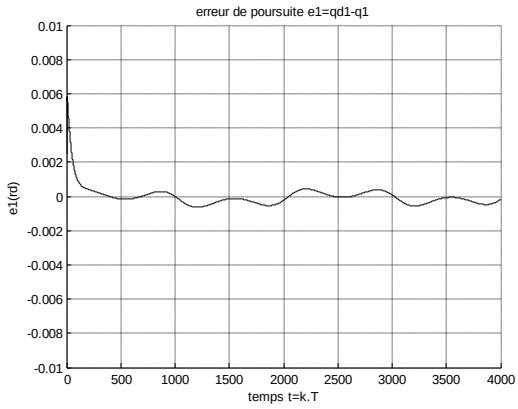
$qd1 = (\pi/4)\sin(\omega kT)$; $qd2 = (\pi/4)\sin(\omega kT)$; T=0.001s; fr=0.5hz



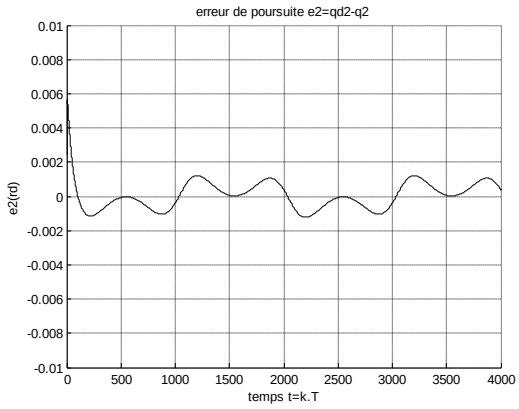
(a)



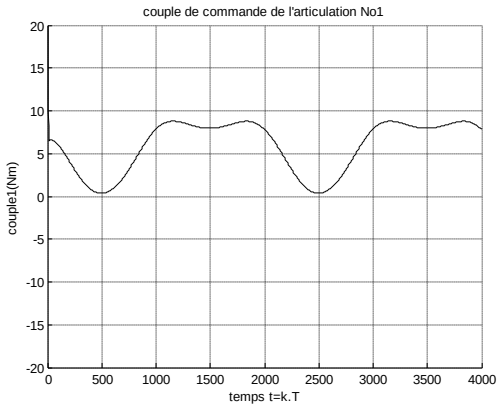
(b)



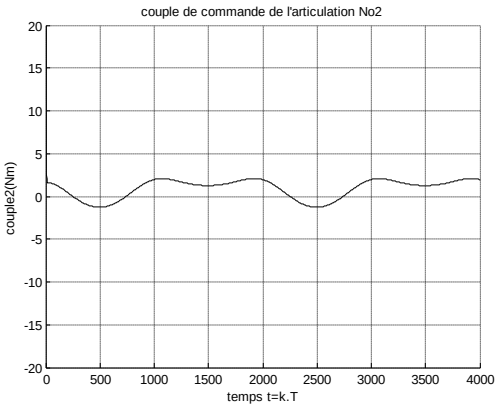
(c)



(d)



(e)



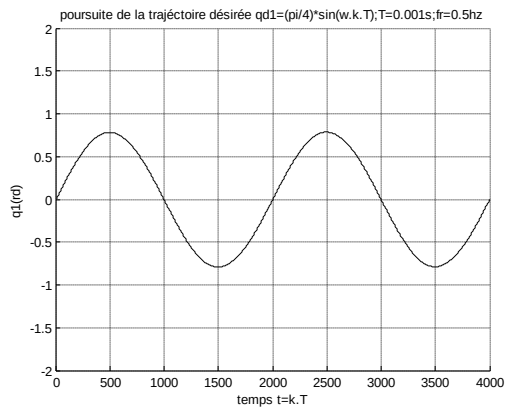
(f)

Figure 1.13.a-f Différents résultats de simulation de la poursuite de trajectoires pour test1

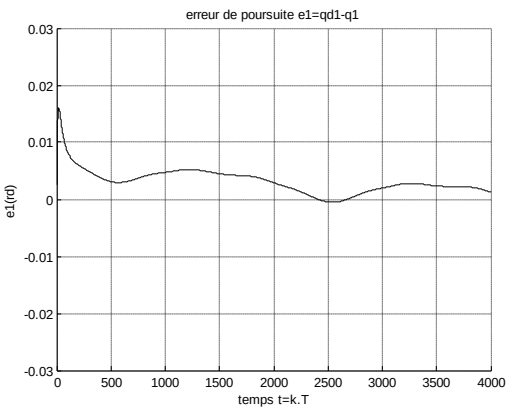
Résultat du teste1 : Un bon suivit de trajectoire désirées est assuré par la commande 'couple calculé'. Les erreurs sont très faibles. Les valeurs des gains sont comme suit :

$k_{p1} = 10000$, $k_{d1} = 500$, $k_{i1} = 5000$, $k_{p2} = 10000$, $k_{d2} = 500$, $k_{i2} = 5000$

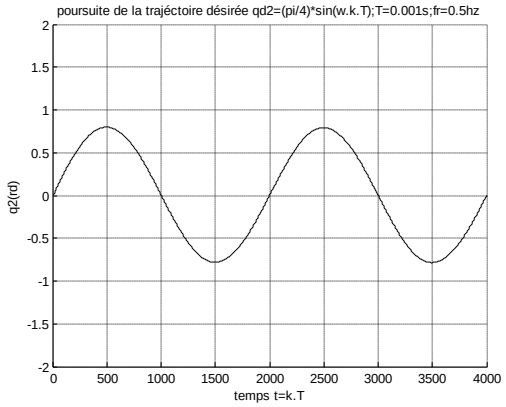
Test2 : Pour ce cas on va voir l'effet d'une erreur de modélisation concernant les masses des liens où $\Delta m_2 = \Delta m_1 = 0.7kg$.



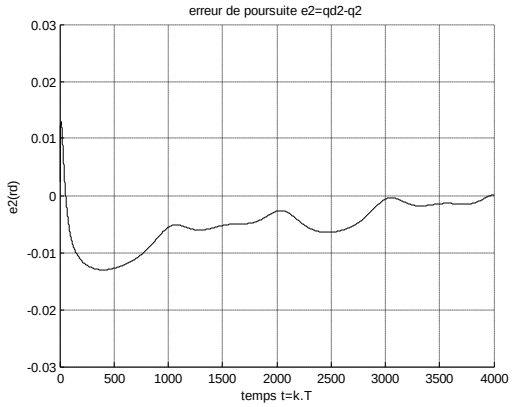
(a)



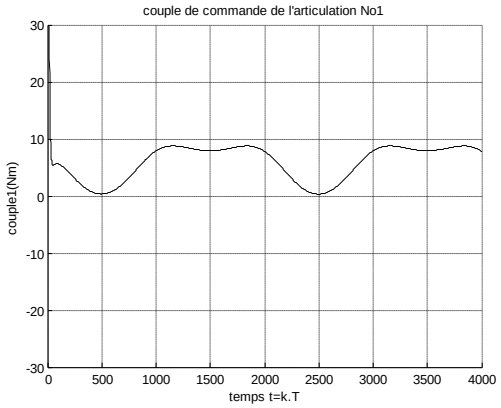
(b)



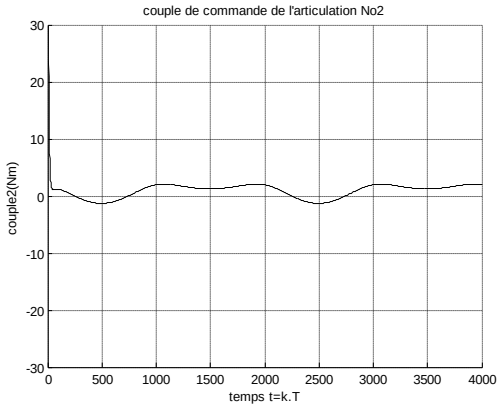
(c)



(d)



(e)



(f)

Figure 1.14.a-f Différents résultats de simulation de la poursuite de trajectoires pour test2

Résultat du test2 : Une erreur de modélisation sur les masses des liens du robot, se manifeste par une erreur de suivi, mais qui n'est pas très importante.

1.12 Conclusion : La structure et la commande d'un robot manipulateur sont fonction de la tâche associée. Une tâche lente peut être contrôlée par un contrôleur classique ; mais pour les tâches rapides une commande basée sur le modèle dynamique devient nécessaire. Parfois lorsque les paramètres du modèle dynamique sont variants ou male identifies on doit faire recours à la commande adaptative ou à d'autres commandes.

BIBLIOGRAPHIE

LIVRES :

[1] John J.Craig, 'Introduction To Robotics Mechanics and Control', Addison Wesley Publishing company, 1986.

[2] Etienne Dombre et Wisam Khalil, 'Modélisation et Commande des robots', Hermès, 1988.

[3] Philip Coiffet 'la Robotique : Principes et applications', Hermès ,1986.

[4] William A. Wolovich 'robotics : Basic Analyse and Design', Holt. Rinehart and Winston, 1986.

[5]J.P. Lallemand et S.Zeghloul,'Robotique ,Aspects fondamentaux: modélisation mécanique, CAO robotique, commande ', masson, 1994.

[6] John J.Craig, 'Introduction to Robotics Mechanics and Control: second edition', Addison Wesley Publishing company, 1989.

ARTICLES:

[7] Bestaoui.Y , ' Decentralised PD and PID robotic regulator', IEE proceedings, vol 136, July 1989.

[8] Hirohiko Arai and Susumu Tachi,'Position control system of two degree of freedom manipulator with a passive joint', IEEe 1991.

[9] Krzysztof Tchon and Robert Muszynski, 'Singular inverse kinematic problem for robotic manipulators ', IEEE 1998.

[10] Milos R. Popovie and Andrew A. Goldenberg,' Modeling of friction using spectral analysis', IEEE1998.

[11] Guangjun Lieu, Karl iagnemme, Steven Dubowsky, Guillaum Morel'A base force/torque sensor approach to robot manipulator inertial parameter estimation',IEEE ,1998.

[12] Patrica conti,'contribution à la commande dynamique adaptative des robots manipulateurs ', thèse de doctorat, université Paul Sabatier, Toulouse, 1987.

[13] Alberto Isaguirre, Richard P.Paul, 'Computation of the inertial and gravitational coefficients of the dynamics equation for robot manipulator with load', proceeding of the American control conference vol2, 1985.

THESES :

[14] Agnès Aubin épouse Pracht ,'Modélisation, identification, et commande du bras TAM', thèse à l'Institut nationale polytechnique de Grenoble,1991.

[15] M. Onder Efe, Okyay Kaynak, Xinghuo Yu, 'Sliding mode control of a three degrees of freedom anthropoid robot by driving the controller parameters to equivalent regime',ASME,2000.

[16] Zvi Shiller, Hai Change, 'Trajectory preshaping for high-speed articulated systems', ASME, journal of dynamic systems, measurement, and control, vol.117, 1995.

[17] H. G. SAGE, M. F. DE Mathelin and E. Ostertag,' Robust control of robot manipulators: a survey', Int. Nat. J. Control, vol. 72, No. 16, 1999.

[18] T. C. Steve Hsia, Ty A. Lasky, and Zhengyu Guo,' Robust Independent Joint controller design for industrial robot manipulators', IEEE Transactions on industrial electronics, vol. 38, No. 1, 1991.

[19] D. Benmerzouk, N. Ghouali, 'Analyse de la commande dynamique des robot a axes rigides' Technologies avancées No.12, janvier 2000.

[20] H. Seraji 'An approach to multivariable control of manipulators', ASME, journal of dynamic systems, measurement, and control, vol.109, June 1987.

CHAPITRE 2

Structure des réseaux de neurones

2. Structure des réseaux de neurones

2.1 Introduction

Les recherches en réseaux de neurone remontent à la fin du 19eme siècle ; et consistent en premier lieu aux travaux multidisciplinaires en physique, en psychologie et en neurophysiologie, cela par des scientifiques tels Hermann von holmholtz, Ernst Mach, Ivan Pavlov et bien d'autres. A cette époque, il s'agissait de théorie plutôt générale sans modèle mathématique précis d'un neurone. C'est qu'en 1943 que Warren McCulloch et Walter Pitts proposent une structure d'un neurone formel inspiré du neurone biologique du système nerveux. Ce neurone formel doté d'une représentation mathématique peut réaliser des fonctions logiques et arithmétiques.

En 1949 Donald Hebb vient renforcer le neurone en présentant dans son ouvrage (The Organization of Behavior) une règle d'apprentissage des paramètres du neurone.

Frank Rosenblatt développe en 1957 le Perceptron inspiré du système visuel possédant deux couches, une de perception et une de décision. On peut dire que c'est le premier système artificiel capable à apprendre par expérience. Environ en même moment Bernad Widrow et Ted Hoff ont proposé leur modèle Adaline avec une règle d'apprentissage qui servira par la suite aux réseaux multicouches [4].

Les auteurs Marvin Minsky et Seymour Papert écrivent en 1969 une critique sur les limitations du Perceptron, et la généralise sur les réseaux de neurones artificiels. Cette critique est venue jeter beaucoup d'ombre sur le domaine des réseaux de neurones. Malgré ces critiques certains chercheurs n'ont pas hésités a continué de travailler sur les réseaux de neurones. Parmi eux Teuvo Kohonen et James Anderson, qui ont développé, en 1972 et en même temps et indépendamment, des réseaux qui peuvent servirent de mémoire associative, de même Stephen Grossberg a investigué ce qu'on appelle les réseaux auto-organisés.

En 1982 J.J.Hopfield, physicien, propose un réseau complètement rebouclé, mais lui fixe un objectif de même une règle d'apprentissage ce qui a donné aux chercheurs un nouveau souffle.

L'année 1985 a fait tout basculer au profit des réseaux de neurones suite aux travaux de David Rumelhart et James Mclelland concernant l'élaboration de la rétropropagation des erreurs comme méthode d'apprentissage des paramètres des réseaux de neurones multicouches.

Les réseaux de neurones servent aujourd'hui à toutes sortes d'applications dans divers domaines tel qu'autopilotage d'avion, guidage d'automobile, lecture de document bancaire et postale, traitement de signal, traitement de la parole, traitement de l'image, prévision sur les marchés monétaires, génération de trajectoire pour les robots manipulateurs et mobiles, et il y a fort à parier que leur importance ira en grandissant dans le futur.

2.2 Le neurone définition et propriétés

Le neurone artificiel inspiré du neurone biologique est illustré en figure 2.1. Le neurone réalise une somme pondérée de ses entrées x_i obtenant ce qui est appelé potentiel *v*. Ce potentiel est transformé par une fonction de transfert *f* qui produit la sortie *y* [3].

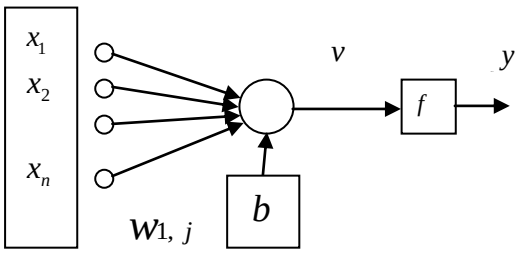


Figure 2.1 Modèle d'un neurone artificiel

Le potentiel *v* est égal :

$$v = \sum_j (w_{1,j}\, x_j) - b \tag{2.1}$$

$$v = W^T X - b \tag{2.2}$$

$w_{1,j}$: Pondérations appelées poids.

La sortie *y* est égale :

$$y = f(W^T X - b) \tag{2.3}$$

La fonction de transfert f peut prendre plusieurs formes comme indiquées au tableau 2.1.

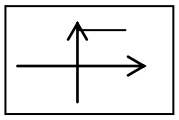
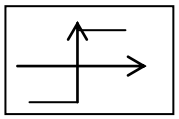
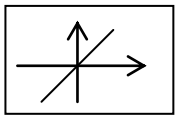
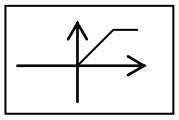
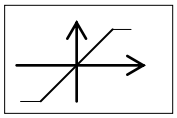
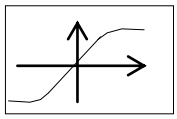
Nom de la fonction	Relation entré/sortie	Icône
Seuil	y=0 si v <0 y=1 si $v \geq 0$	
Seuil symétrique	y=-1 si v <0 y=1 si v ≥ 0	
Linéaire	$y=v$	
Linéaire positive saturée	y =0 si v <0 y=v si $0 \leq v \leq 1$ y =1 si v >1	
Linéaire saturée symétrique	y =-1 si v <-1 $y = v$ si-$1 \leq v \leq 1$ y =1 si v >1	
Tangente hyperbolique	$y = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}}$	
Compétitive	y=1 si v max y=0 autrement	

Tableau 2.1 Différentes fonctions de transfères y=f(v)

2.3 Les réseaux de neurones

Comme nous venons de le voir, un neurone réalise simplement une fonction non linéaire, paramétrée, de ses variables d'entrée. L'intérêt des neurones réside dans les propriétés qui résultent de leur association en réseaux, c'est-à-dire la composition des fonctions non linéaires réalisées par chacun des neurones.

Malgré les différentes structures, généralement, un réseau de neurone possède une structure en couches ; soit une couche d'entrée, des couches cachées et une couche de sortie comme montré en figure 2.2.

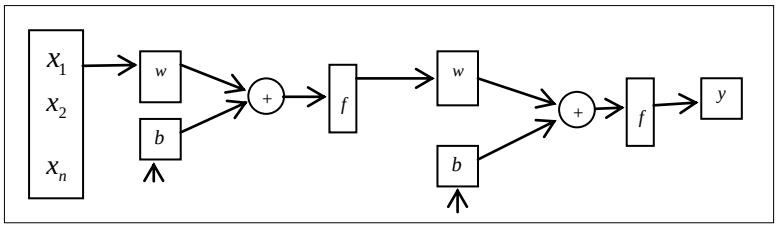


Figure 2.2 Réseaux multicouches

2.4 Réseaux statiques et réseaux dynamiques

Les réseaux de neurones se partagent en deux groupes, les réseaux statiques et les réseaux dynamiques.

Pour les réseaux statiques ou non bouclés l'information circule des entrées vers les sorties sans retour en arrière et la notion de temps n'intervient pas ; donc si l'entrée est stable la sortie reste stable.

Pour les réseaux dynamiques ou bouclés un retour de l'avant vers l'arrière est possible. La notion de temps intervient par les retards associés aux bouclages. On peut donc représenter un système d'équations aux différences non linéaire.

2.5 L'apprentissage des réseaux de neurones

On appelle apprentissage des réseaux de neurones la procédure qui consiste à estimer les paramètres (poids, biais,…) des neurones du réseau, afin que celui-ci remplisse au mieux la tâche qui lui est affectée. A savoir qu'il existe plusieurs paradigmes d'apprentissage.

2.5.1 Apprentissage supervisé

Un réseau de neurone comme il est paramétré (poids, biais…) doit faire l'ajustement de ses paramètres pour mieux remplir sa tâche.

Le réseau s'il est guidé lors de l'adaptation des paramètres par un professeur, qui à chaque entrée, lui offre la sortie désirée, peut s'ajuster en se référenciant à l'erreur vis-à-vis de sa sortie. On dit qu'il est supervisé par un professeur [3].

2.5.2 Apprentissage non supervisé (auto-organisationnel)

Une fois le professeur est absent le réseau ne peut bénéficier de la présence de la sortie désirée de chaque entrée. Alors le réseau doit s'auto-organiser pour regrouper des ensembles de

données selon des critères de ressemblances qui sont inconnus à priori. Le réseau lui-même doit découvrir les ressemblances et créer des catégories dans un espace de dimensions plus faible [2].

2.5.3 Apprentissage hybride

C'est le cas où le type supervisé et non supervisé sont tout deux utilisés.

2.5.4 Apprentissage par renforcement

C'est un type d'apprentissage qui ressemble bien à l'apprentissage supervisé, sauf que la sortie est une situation qu'il faut renforcer soit par une récompense ou une pénalisation. Le réseau et suivant la récompense ajuste ses paramètres [3].

2.5.5 Apprentissage compétitif

L'apprentissage compétitif, comme son nom l'indique, consiste à faire compétitionner les neurones d'un réseau pour déterminer lequel sera actif à un instant donné. Contrairement aux autres types d'apprentissage, où généralement tous les neurones peuvent apprendre simultanément et de la même manière, l'apprentissage compétitif produit un « vainqueur » ainsi que parfois, un ensembles de neurones « voisins » vainqueurs, et seuls se vainqueur et son voisinage bénéficient d'une adaptation de leurs poids, alors que le neurone qui ne gagne pas la compétition ne modifiera aucunement ses poids. Ainsi, les neurones individuels peuvent apprendre à se spécialiser sur des sous-ensembles de données pour devenir des détecteurs de caractéristiques [7].

2.6 Règles d'apprentissages

L'adaptation des poids d'un réseau de neurones nécessite un algorithme ou une règle. Pour chaque paradigme d'apprentissage et type de réseau on peut trouver une règle ou plus [1], [2].

2.6.1 Règle basée sur correction de l'erreur

Cette règle est fondée sur la minimisation de l'erreur, observée en sortie, entre la valeur de la sortie désirée y_d et la sortie du réseau y ; cette erreur est pour chaque neurone :

$$e_i(t) = y_{d_i}(t) - y_i(t) \tag{2.4}$$

Soit pour tout le réseau : $e(t) = y_d(t) - y(t)$ (2.5)

Alors on doit chercher un moyen de réduire autant que possible cette erreur. L'apprentissage par correction des erreurs consiste à minimiser un indice de performance F basé sur l'erreur e, dans le but de faire converger les sorties du réseau avec ce qu'on voudra qu'elles soient. Un critère très populaire est la somme des erreurs quadratiques :

$$F(e(t)) = \sum_{i=1}^{m} e^2{}_i(t) = e(t)^T e(t) \tag{2.6}$$

Sachant que les paramètres libres du réseau sont les poids $W(t)$ donc l'indice peut être écrit :

$$F(e(t)) = F(W(t)) = F(t) \tag{2.7}$$

Les poids du réseau évoluent de la façon suivante :

$$W(t+1) = W(t) + \eta L(t) \tag{2.8}$$

Où L(t) est la direction dans la quelle nous allons chercher le minimum et η est une constante positive déterminant l'amplitude du pas dans cette direction (la vitesse d'apprentissage). L'objectif est de faire en sorte que $F(t+1) < F(t)$.

Le développement en séries de Taylor de $F(t+1)$ autour de $W(t)$ donne :

$$F(t+1) = F(t) + \nabla F(t)^T \Delta W(t) \tag{2.9}$$

Où $\nabla F(t)$ est le gradient de $F(t)$.

Si notre objectif est que $F(t+1) < F(t)$ on peut écrire :

$F(t+1) - F(t) = \nabla F(t)^T \Delta W(t) < 0$ Soit $\nabla F(t)^T \Delta W(t) < 0$ Ou aussi $\nabla F(t)^T \eta L(t) < 0$

Cela est toujours vrais si : $L(t) = -\nabla F(t)$ (2.10)

Donc $W(t+1) = W(t) - \eta \nabla F(t)$ (2.11)

2.6.2 Règle de Hebb

Cette règle est au nom du neurobiologiste Donald Hebb que vient d'énoncer en 1949 de la façon suivante :

« Si deux neurones de part et d'autre d'un synapse (connexion) sont activés simultanément, d'une manière synchrone, alors la force de ce synapse doit être augmentée. Si les mêmes deux

neurones sont activés d'une manière asynchrone, alors le synapse correspondant doit être affaibli ». Une telle synapse (connexion) est dite « synapse hebbien ». Il utilise un mécanisme interactif, dépendant du temps et de l'espace, pour augmenter l'efficacité synaptique d'une manière proportionnelle à la correction des activités prés et post synaptiques.

Sous sa forme mathématique la plus simple la règle d'apprentissage est écrite comme suit :

$$\Delta w_j(t-1) = \eta x_j(t) y(t) \tag{2.12}$$

η : Constante qui détermine la vitesse d'apprentissage.

$x_j(t)$ activité pré-synaptique et $y(t)$ activité post-synaptique à ce même temps .

L'équation (2.12) montre que les changements de poids $\Delta w_j(t)$ peuvent croître de façon exponentielle. Pour pallier à cette croissance exponentielle qui provoquerait invariablement une saturation du poids, on ajoute parfois un facteur d'oubli qui retranche de la variation de poids, une fraction α du poids actuel. On obtient ainsi :

$$\Delta w_j(t-1) = \eta x_j(t) y(t) - \alpha w_j(t-1) \;; 0 \le \alpha \le 1 \tag{2.13}$$

2.6.3 Règle d'apprentissage compétitif

L'apprentissage compétitif s'applique généralement pour les réseaux à apprentissage non supervisé. Dans leur forme la plus simple, les réseaux qui utilisent cet apprentissage sont souvent constitués d'une seule couche de neurone de sortie totalement connectée sur les entrées. Un neurone vainqueur modifiera ses poids en les rapprochant d'un stimulus d'entrée X pour lequel il a battu tous les autres neurones lors de la compétition [2].

Si le neurone vainqueur admis des voisins la règle d'adaptation est comme suit :

$$\Delta w = \begin{cases} \eta_1(x - w) & \text{si le neurone est vainqueur} \\ & \quad avec \quad \eta_1 > \eta_2 \\ \eta_2(x - w) & \text{si le neurone est voisin du vainqueur} \\ 0 & \text{autrement} \end{cases} \tag{2.14}$$

Le neurone est vainqueur s'il est le plus proche du stimulus d'entrée.

2.6.4 Règle de Boltzmann

Cette règle s'applique au réseau de Boltzmann ; qui est un réseau symétrique ($w_{ij} = w_{ji}$) et récurrent, ayant deux cellules de neurones une visible relie à l'environnement et une cachée pour le calcul. Ce réseau vient comme amélioration du réseau de Hopfield et possède une fonction d'activation stochastique.

La règle d'apprentissage consiste en l'ajustement des poids des connexions de telle sorte que les états des neurones de la cellule visible satisfassent une distribution probabiliste souhaitée. Durant la période d'apprentissage le réseau passe par deux phases, phase contrainte et phase libre. Alors l'adaptation des poids se fait comme suit :

$$\Delta w_{ij} = \eta(P_{ij}^{+} - P_{ij}^{-}) \tag{2.15}$$

Avec P_{ij}^{+} est la probabilité que les neurones i et j soient coactives durant la phase contrainte et P_{ij}^{-} la probabilité qu'ils soient coactives durant la phase libre.

2.7 Les différents modèles de réseaux de neurones, leurs règles d'apprentissage et leurs applications

Depuis l'apparition du premier neurone de McCulloch, plusieurs modèles sont apparus au fil du temps; où chacun possède une structure propre, de même sa règle d'apprentissage [1].

2.7.1 Le Perceptron

Le Perceptron simple est monocouche ayant n entrées et m neurones à fonction d'activation à seuil et nxm connexions. La figure 2.3 illustre ce modèle.

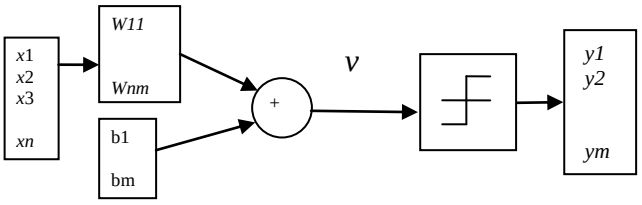


Figure 2.3 Le Perceptron à une seule couche avec fonction seuil

xi: entrée *wij*: poids bi :biais *y*: sortie

La fonction de transfère est comme suit :

$$y_i = \begin{cases} +1 \; si \; v_i \geq 0 \\ -1 \; si \; autrement \end{cases} \tag{2.16}$$

$$v_i = \sum_{j=1}^{n} (x_j w_j^i + b_i) \tag{2.17}$$

La règle d'apprentissage (sous forme matricielle) a une forme simple mais puissante est donnée par :

Soit $e = y_d - y$ Alors $\Delta w = eX^T$ et $\Delta b = e$ (2.18)

Le Perceptron assure généralement une séparation linéaire de ses entrées (s'ils sont linéairement séparables).

2.7.2 L'Adaline

Est un modèle que proposent Widrow et Hoff, cela en 1960. En figure 2.4 ce modèle est illustré. Développé dans un contexte du traitement de signal, trouve immédiatement application en télécommunication.

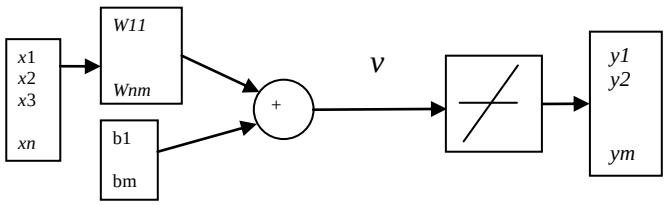


Figure 2.4 L'Adaline à une seule couche avec fonction linéaire

xi: entrée *wij*: poids *bi*: biais *y*: sortie

La fonction de transfère est comme suit :

$$y_i = f(v_i) \quad f : \text{Fonction linéaire} \tag{2.19}$$

$$v_i = \sum_{j=1}^{n} (x_j w_j^i + b_i)$$

La méthode d'apprentissage basée sur la règle LMS (erreur quadratique moyenne) tente de minimiser un critère $F(W)$; et comme indiqué en équations (2.7) et (2.11) on a :

$$F(e(t)) = \sum_{i=1}^{m} e^2{}_i(t) = e(t)^T e(t)$$

$$\Delta W(t) = -\eta \nabla F(t) \tag{2.20}$$

On abouti à la formule d'apprentissage suivante :

$$\begin{aligned} \Delta W(t) &= 2\eta e(t) X^T(t) \\ \Delta b(t) &= -\eta e(t) \end{aligned} \tag{2.21}$$

Avec un choix adéquat du paramètre d'apprentissage η la méthode du descente du gradient garantie la convergence de l'erreur vers un minimum globale donc de trouver les paramètres du réseau (poids et biais).

2.7.3 Le Perceptron multicouches

Le Perceptron simple monocouche ne réalise que des classifications linéairement séparables, et les réseaux multicouches permettent de lever cette limitation. On peut donc construire des frontières de décision de complexité quelconque, ouvertes ou fermées, concaves ou convexes; à condition d'employer une fonction de transfert non linéaire et avoir suffisamment de neurones sur les couches cachées [5].

Le réseau multicouche, à une couche d'entrée et une ou plus couches cachées et une couche de sortie, est appelé aussi Perceptron Multicouches (PMC).

La fonction d'activation des neurones de la couche cachée doit être non linéaire, généralement une sigmoïde. La figure 2.5 illustre le PMC.

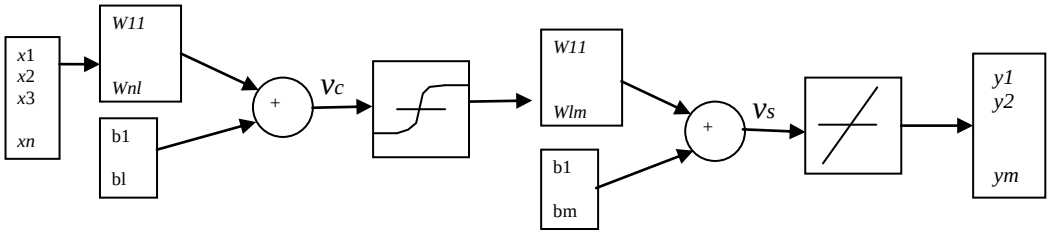


Figure 2.5 Le Perceptron multicouche avec fonction sigmoïde en couche cachée

xi: entrée *wij*: poids bi :biais *y*: sortie

La fonction de transfert des neurones de la couche cachée est une sigmoïde ayant l'une des formes suivante :

Sigmoïde : $y = \dfrac{1}{1+e^{-v}}$ Tansig : $y = \dfrac{e^{+v} - e^{-v}}{e^{+v} + e^{-v}}$, (2.22)

L'apprentissage du PMC est basé sur l'erreur on utilisant un critère d'optimisation $F(W)$ pour minimiser l'erreur quadratique moyenne :

$$F(W) = F(t) = e(t)^T e(t) \quad ; \quad e(t) = y_d(t) - y(t) \tag{2.23}$$

La méthode de la descente du gradient est utilisée pour l'adaptation des paramètres du réseau (poids et biais) soit :

$$\Delta w(t) = -\eta \frac{\partial F(t)}{\partial w} \quad ; \; \Delta b(t) = -\eta \frac{\partial F(t)}{\partial b} \tag{2.24}$$

Comme connu l'algorithme de la **rétropropagation** proposé par David Rumelhart et James Mclelland en 1985 est le plus utilisé pour calculer les paramètres du réseau. L'astuce dans la méthode est de **rétropropager** l'erreur **e** de la sortie vers l'entrée lors du calcul des sensibilités S^k avec :

$$S^k = \frac{\partial F(t)}{\partial v^k} \quad , v^k = (v_1^k, v_2^k, ..., v_l^k) \text{ potentiel de la couche cachée.} \tag{2.25}$$

L'Algorithme d'adaptation des paramètres se résume comme suit :

1. Initialiser tous les poids et biais à de petites valeurs.
2. Pour chaque couple d'entrée désirée (x_d, y_d) de la base d'apprentissage Faire :
   a- Propager les entrées x_d vers l'avant à travers les couches du réseau.
   $y^0 = x_d$, $y^{k+1} = f^k(w^k y^k - b^k)$ pour k=1,…, m ; m : nombre de couches (m=2).
   b- Rétropropager les sensibilités vers l'arrière à travers les couches du réseau.
   $S^m = -2\dot{f}^m(v^m)(y_d - y)$ f^m : Fonction d'activation de la couche de sortie.
   $S^k = -2\dot{f}^k(v^k)(w^{k+1})S^{k+1}$ Pour les couches de k=m-1,..,1 .
   c- Mettre à jours les poids et biais :
   $\Delta w^k = -\eta S^k (y^{k-1})^T$ Pour k=1,…, m.
   $\Delta b^k = -\eta S^k$ Pour k=1,…, m.
3. Si le critère d'arrêt est atteint, alors Stop.
4. Sinon, permuter l'ordre de présentation des associations de la base d'apprentissage.
5. Recommencer de l'étape 2.

2.7.3.1 Des considérations pour la méthode de rétropropagation

La méthode de rétropropagation sous sa version originale souffre de quelque limitations tel que, les minima locaux de la surface d'erreur, la saturation des poids, le ralentissement de l'apprentissage, le surapprentissage. Pour cela plusieurs solutions d'amélioration sont apportées à la rétropropagation.

A. Eviter le surapprentissage :

Le surapprentissage signifie que le réseau apprend par cœur tous les exemples qu'on lui présente de façon à perdre la notion de généralisation. Pour éviter cela [7], on doit procéder selon une manière parmi celles-ci :

A.1 Arrêt prématuré (Early stopping):

Revient à arrêter l'apprentissage avant qu'il arrive à sa fin.

A.2 Validation croisée (cross validation)

Qui consiste à partager l'ensemble des données en trois groupes, un premier groupe pour l'apprentissage, un deuxième pour la validation et un troisième pour le test.

Si par le groupe d'apprentissage on modifie les poids, alors par le groupe de validation on arrête la procédure d'apprentissage une fois le critère de validation se détériore. Le groupe de teste vient en dernière phase pour s'assurer de la généralisation du réseau.

A.3 Régularisation par modération des poids (weight decay)

Sa consiste en l'évitement que les paramètres du réseau prennent des valeurs exagérées pour s'assurer que les fonctions d'activations, 'sigmoïde', agissent en leurs zones linéaires [7].

B. Améliorer la vitesse de convergence

Une solution pour augmenter la vitesse de convergence de l'algorithme est de modifier le taux d'apprentissage dynamiquement tout au long de l'entraînement [3].

Par exemple, si l'erreur quadratique totale, calculée pour toutes les associations de la base d'apprentissage, augmente d'une période à l'autre par un certain pourcentage (1 à 5%) à la suite d'une mise à jour des poids, alors cette mise à jour doit être abandonnée et le taux d'apprentissage augmenté.

C. Améliorer la descente du gradient

Pour filtrer les oscillations possibles sur la trajectoire de la descente du gradient on ajoute un terme d'inertie appelé « momentum » à la formule d'ajustement des poids :

$$\Delta w^k(t) = \alpha \Delta w^k(t-1) - (1-\alpha)\eta S^k (y^{k-1})^T$$
$$\Delta b^k(t) = \alpha \Delta b^k(t-1) + (1-\alpha)\eta S^k \qquad \text{k=1,…, m} \qquad (2.26)$$
$$0 \le \alpha \le 1$$

D. Une autre procédure d'optimisation

Lors de l'évaluation de l'indice de performance $F(W)$ qui minimise l'erreur quadratique ; cet indice $F(W)$ est décomposé en séries de Taylors. Pour l'optimisation avec le gradient simple, seule le terme du premier ordre du développement en séries de Taylors est considéré. Mais pour la méthode de Neuwtone on considère le terme du second ordre.

$$F(W+\Delta W) \square F(W) + \nabla F(W)^T \Delta W + \frac{1}{2}\Delta W^T \nabla F^2(W)$$

Qui donne : $\Delta W = -(\nabla F^2(W))^{-1} \nabla F(W)^T$ soit $\Delta W = -H^{-1}(W)\nabla F(W)^T$

H : La matrice hessienne ; calculée par l'algorithme BFGS ou l'algorithme de Levenberg-Marquard.

2.7.3.2 Les applications du PMC

Les domaines d'applications sont nombreux tel que : Approximation universel parcimonieuse, reconnaissance des formes, classification, identification, commande, prédiction, filtrage.

2.7.4 Les réseaux RBF

Les réseaux à fonctions de bases radiales, ayant deux couches de neurones, sont une classe spéciale des réseaux statiques multicouches. La couche cachée utilise comme fonction d'activation une gaussienne et la couche de sortie utilise une fonction linéaire. La figure 2.6 illustre le schéma d'un réseau RBF [2],[1].

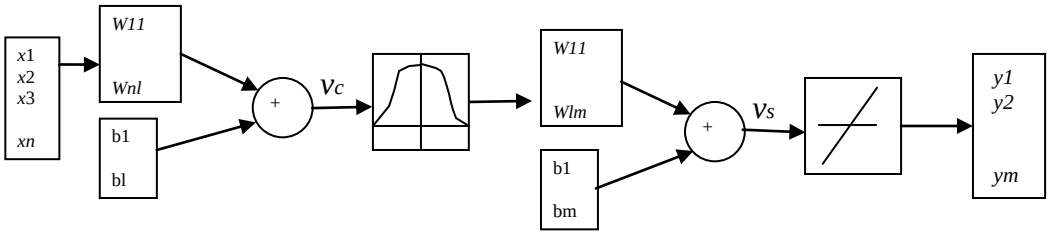


Figure 2.6 Illustration d'un réseau RBF

La fonction d'activation de la couche cachée est une gaussienne :

$$f_i(v_i) = e^{-d_i(vi)} = e^{-\|vi-c_i\|^2/\sigma^2} \quad (2.27)$$

Où c_i : Position du centre. σ_i : La taille du champ récepteur.

A la différence du PMC les neurones de la couche cachée du RBF ayants la fonction de transfert gaussienne agissent d'une façon locale et cela suivant son champs d'action, lui-même fonction de c_i et σ_i. Donc chaque neurone agit juste sur un ensemble de données et non pas sur toutes les données.

Pour entraîner le réseau RBF, on peut utiliser plusieurs stratégies, la première consiste à optimiser simultanément tous les paramètres du réseau, par exemple en utilisant la rétropropagation des erreurs. Il s'agit de la position des centres des fonctions radiales, de leurs variances, et finalement des poids de la couche linéaire de sortie. Cette approche comporte certaines difficultés liées à la nature très différente de ces deux couches et leurs dynamiques de convergence. La première couche constituée de neurones non linéaires agissant localement dans l'espace des entrées, a plutôt tendance à converger lentement, alors que la seconde, avec ses neurones linéaires, converge généralement très rapidement. Ces dynamiques très différentes provoquent souvent une stagnation de l'apprentissage autour d'un minimum local parfois très éloigné de l'optimum global.

On peut remarquer que les deux couches du réseau RBF réalisent des fonctions distinctes, en ce sens, on peut procéder à leur apprentissage en deux étapes. La première consiste à estimer les positions des centres des neurones radiaux puis à estimer leurs variances et en deuxième étape estimer les poids de la couche linéaire.

Les réseaux RBF réclament plus de neurones en couche cachée que les réseaux PMC qui fait qu'ils sont plus lents en utilisation, mais ayant la même puissance de calcul, donc presque les mêmes utilisations.

2.7.5 Réseau de hopfield

C'est un réseau récurent proposé par Hopfield en 1982, inspiré de la physique statistique des milieux désordonnés, à partir de la représentation des atomes et de leurs interactions (modèle des spins de Ising) [4].

Le réseau de Hopfield est totalement connecté et symétrique $(w_{ij} = w_{ji}, w_{ii} = 0)$; comme illustré en figure 2.7 ses sorties sont aussi ses entrées. Hopfield insiste sur le caractère dynamique des réseaux de neurones naturels provoqué par la récurrence des connections. Le réseau possède des états d'équilibres qui sont des attracteurs.

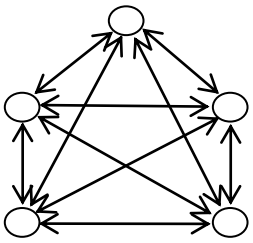
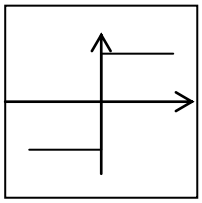


Figure 2.7 Réseau de Hopfield

Dans le cas où le réseau de Hopfield est binaire la fonction de transfère des neurones est la fonction « signe » :

$$y_i = sign(\sum_j w_{ij} y_i - b_i) \qquad \text{Signe :} \qquad (2.28)$$



Pour le cas ou le réseau de Hopfield est analogique la fonction de transfert des neurones est une sigmoïde :

$$y_i = \tanh(\sum_j w_{ij} y_i - b_i) \qquad (2.29)$$

L'apprentissage consiste à calculer les paramètres du réseau de telle manière que les codes des informations que l'on souhaite mémoriser soient des états stables du réseau.

Hopfield utilise la règle d'apprentissage de Hebb adaptée aux contraintes de connectivité de son réseau.

$$w_{ij} = \frac{1}{N} \sum_{p \in P} y_i^p y_j^p \quad , \quad w_{ii} = 0 \qquad (2.30)$$

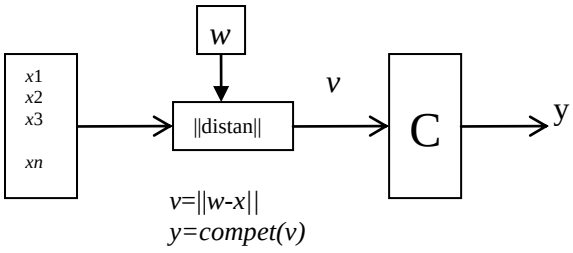
N : Nombre d'exemples d'informations qu'on veut coder.

P : Est l'ensemble des patrons à apprendre.

Le réseau de Hopfield est utilisé en optimisation et en reconnaissance de formes.

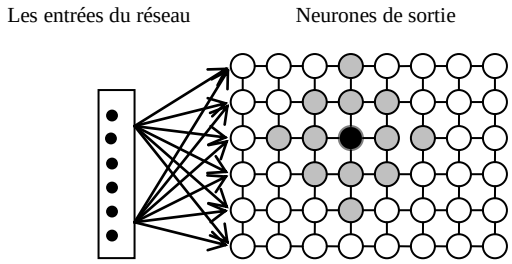
2.7.6 Réseau de Kohonen

Le réseau de Kohonen introduit en 1972 dit aussi carte auto-organisatrice, s'agit d'un réseau supervisé avec un apprentissage compétitif. L'ensemble des neurones constituants la carte auto-organisatrice permet de structurer en catégories l'ensemble des données en entrée, qui sont généralement de grande dimension [4]. Comme illustré en figure 2.8.a-b les données d'entrée sont reliés à tout les neurones constituant la carte qui est une couche unique de sortie. La carte auto-organisatrice peut être de dimension 1,2 ou 3, où chaque neurone possède un voisinage mais sans liaison synaptique.



(a)

Figure 2.8.a Réseau de kohonen



(b)

Figure 2.8.b Carte auto-organisatrice un neurone et son voisinage

La fonction d'activation des neurones peut être comme suit :

$$f(v) = \frac{1}{(1+e^{-v})} \tag{2.31}$$

L'algorithme d'apprentissage du réseau de Kohonen est de type compétitif, se base sur le principe, le gagnant prend tout « the winger take all ». La mise à jour des poids du neurone i au temps t s'exprime de la façon suivante :

$$\Delta w_i(t) = \begin{cases} \eta(t)(\, x(t) - w_i(t)\,) & \text{si } i \in \Lambda_g(t) \\ 0 & \text{autrement} \end{cases} \tag{2.32}$$

$x(t)$ le stimulus d'apprentissage au temps t et $\Lambda_g(t)$ représente le voisinage du neurone gagnant « g » à ce même temps t.

Le neurone gagnant est déterminé en cherchant le neurone i qui vérifie la condition :

$$g(x) = \arg\min \left\| x_i - w_i \right\| \qquad i = 1, 2, ..., n \tag{2.33}$$

C-à-d celui qui est le plus proche du stimulus ou celui qui répond le mieux.
L'idée de Kohonen pour l'apprentissage étant d'employer au départ un grand taux d'apprentissage et un grand voisinage, et de réduire ceux-ci au fur et à mesure que le temps (donc l'apprentissage) progresse.

On utilise souvent une décroissance linéaire pour le taux d'apprentissage :

$$\eta(t) = \begin{cases} \eta_0 - \dfrac{(\eta_0 - \eta_\tau)}{\tau} t & si\ t < \tau \\ \eta_\tau & \text{autrement} \end{cases} \qquad (2.34)$$

η_0 : Taux d'apprentissage initial.

η_τ : Taux d'apprentissage final.

τ : Délimite la frontière entre deux phases d'apprentissage (phase d'organisation, phase de convergence).

Pour le voisinage Λ_g la décroissance des neurones voisins est décrite par :

$$\Lambda_g(t) = \begin{cases} \Lambda_0 (1 - \dfrac{t}{\tau}) & \text{si } t < \tau \\ 0 & \text{autrement} \end{cases} \qquad (2.35)$$

La carte auto-organisatrice de Kohonen assure que deux régions adjacentes dans l'espace des entrées seront aussi adjacentes dans l'espace des sorties.
Le réseau de Kohonen est utilisé pour la catégorisation, et les transformations de bases.

2.7.7 Autres Réseaux de neurone [2][4]

2.7.7.1 Réseau LVQ (learning vector quatisation)

C'est un réseau supervisé hybride du fait qu'il est constitué d'une première couche compétitive permettant de catégoriser les entrées en classes et d'une deuxième couche linéaire de décision, qui offre une décision suivant la classe active. Le type de décision est fixé avant même l'étape d'apprentissage et une fois pour tout.

2.7.7.2 Réseau GNG(Growing Neural Gas)

C'est un réseau constructif, introduit par Bernd Fritzke, ne pose à priori aucune hypothèse sur la topologie des entrées. Un réseau minimal initialement crée lui seront ajoutés de nouveaux neurones ainsi que de nouvelles connexion au fil de l'apprentissage non supervisé.

Les neurones peuvent apparaître et disparaître du graphe tout au long du processus d'apprentissage. L'apprentissage est compétitif et semblable à celui de Kohonen avec certaines particularités.

2.7.7.3 Réseau de Jordan

Jordan considère le fait que, on ne peut donner simultanément au système tout ce dont il a besoin pour traiter un problème, il est nécessaire de le lui donner par morceau. Intervient alors le notion du temps dans le traitement, et une mémoire contextuelle devient nécessaire pour retenir les résultats intermédiaires.

Alors Jordan propose en 1986 un réseau récurent à couches. Le réseau à pour but d'effectuer une séquence d'actions en rapport à un 'plan' donné par l'utilisateur et constant durant l'exécution de la séquence.

Cet état des choses est réalisé dans le réseau de Jordan par une série de couches, comme illustré en figure 2.9. La couche du plan (« plan layer ») reçoit une entré qui reste constante durant l'exécution du réseau. La couche cachée et la couche de sortie sont responsables de l'expression des actions à effectuer. Enfin la couche de contexte (« state layer ») sert de mémoire au réseau, elle retient l'action du réseau au temps précédent et maintient par une boucle récurrente locale sa propre activation à chaque itération.

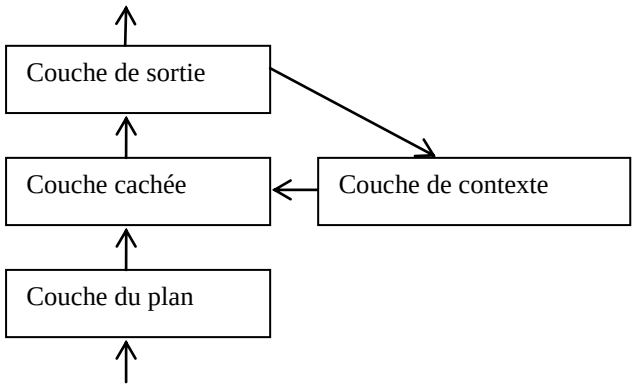


Figure 2.9 Illustration des couches du réseau de Jordan

L'aspect récurent du réseau est ignoré par la fonction d'apprentissage, qui est une variante de la rétro-propagation mais un peut plus complexe. Le réseau de Jordan est appliqué en commande de processus comme en robotique par exemple.

2.7.7.4 Réseau d'Elman

C'est une inspiration directe du réseau de Jordan qui a donné le réseau d'Elman. Elman propose de boucler la couche cachée sur elle-même au lieu de la couche de sortie. L'effet de cette boucle est de recopier sans autre traitement la valeur de la couche cachée dans la couche de contexte ; donc l'état de la couche de contexte à l'instant 't'est égale à celui de la couche cachée à l'instant 't-1' ; c-à-d on sauvegarde l'état de la couche cachée pour la séquence suivante comme on peut le voir en figure 2.10.

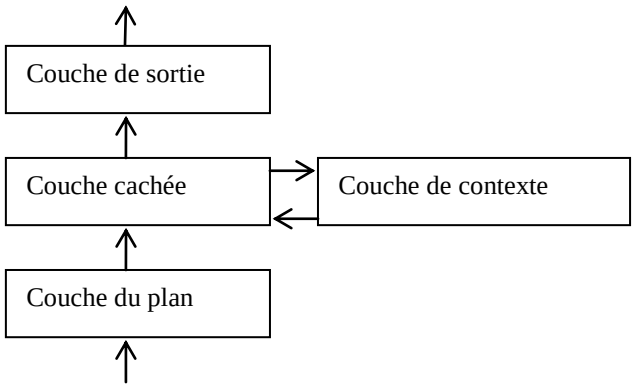


Figure 2.10 Illustration des couches du réseau d'Elman

La couche cachée et du fait qu'elle est plus libre que les couches d'entrée et de sortie, prenne en considération l'historique du réseau donc le temps ; ce qui met en évidence la récurrence du réseau. L'apprentissage comme le réseau de Jordan se base sur la rétro-propagation classique.

2.7.7.5 Réseau ARTs

C'est Grossberg qui a introduit les réseaux ART pour, théorie de la résonance adaptative « Adaptive Resonance Theory » [3].

ART1 est un réseau binaire dont la fonction est d'associer à un patron d'entrée un seul neurone de sortie. Le principe des réseaux ARTs d'inspiration biologique qui est un peut complexe est décrit par Grossberg comme indiqué en Annexe B. Parmi les applications des ARTs la classification et l'approximation des fonctions.

2.8 Conclusion : Les réseaux de neurones par leur caractéristique d'approximation et de généralisation peuvent prendre la caractéristique 'universel'. Les différentes architectures des réseaux de neurones évoquent leur utilisation dans plusieurs domaines sur tout en robotique qui est multi disciplines. Sont utilisés en commande (réseaux PMC), comme en reconnaissance d'objets fixes ou mobiles (réseaux de Kohonen).

BIBLIOGRAPHIE

[1] Anil.k Jain, jianchang Mao and K.M. Mohiudin, 'Artificial Neural Networks: A Tutorial', IEEE, 1996.

[2] Jean-François Jodouin 'les réseaux neuromimétriques', Hermes, 1994.

[3] Marc Parizeau 'Réseaux de neurones', université Laval, 2004.

[4] Claude Touzet, 'cours : Les réseaux de neurones artificiels', juillet 1992.

[5] I.Rivals, L.Personnaz, G.Dreyfus et J.L. Ploix, 'Modélisation, classification et commande par réseaux de neurones : Principes fondamentaux, méthodologie de conception et illustrations industrielles', Lavoisier Technique et documentation, Paris, 1996.

[6] Marc Lucea 'Modélisation dynamique par réseaux de neurones et machines vecteurs supports : contribution à la maîtrise des émissions polluantes de véhicules automobiles', thèse de doctorat, de l'université Paris 6, France, 2006.

[7] G.Dreyfus, J.M.Martinez, M.Samuelides, M.B.Gordon, F.Badran, S.Thinria, L.Hérault, 'Réseaux de neurones : méthodologie et applications', Eyrolles, 2002.

CHAPITRE 3

Les réseaux de neurones pour le contrôle

3. Les réseaux de neurones pour le contrôle

3.1 Introduction

La commande des systèmes linéaires ou des systèmes approchés par des systèmes linéaires est un domaine bien maîtrisé qui permet d'avoir des contrôleurs possédant des propriétés de stabilité et d'optimalité ; cependant le monde réel nous contredit par le fait que la plus part des systèmes sont non linéaires et dynamiques de même à paramètres variants.

Alors il est bon de chercher à trouver un outil qui permet de surpasser la non linéarité des systèmes réels et de même s'adapter par un apprentissage hors ligne ou en ligne aux variations de leurs paramètres. Les réseaux de neurones s'avèrent une solution du fait qu'ils peuvent modéliser l'importe qu'elle fonction non linéaire, et possèdent aussi la faculté d'apprentissage à partir d'exemples (entrées /sorties) du système à commander ; cette faculté d'apprentissage est plus intéressante que la commande adaptative traditionnelle, qui ne possède pas les caractéristiques de la mémorisation par la généralisation. Par contre il y'a un manque d'étude complète sur la stabilité des systèmes contrôlés par les réseaux de neurones.

D'autres parts les réseaux de neurones possèdent plusieurs caractéristiques intéressantes pour la réalisation de systèmes de commandes, telles que :

- Le parallélisme du traitement assure la rapidité donc l'application temps réel.
- Le caractère distribué du traitement de la donnée par les réseaux de neurones assure une résistance aux pannes internes du système réalisé.
- La capacité de généralisation leurs confèrent une bonne marge aux bruits.

3.2 Méthodes de contrôles basées sur les réseaux de neurones

3.2.1 Reproduction d'un contrôleur existant

Parmi les utilisations des réseaux de neurones pour la réalisation d'un système de commande neuronal c'est la reproduction du fonctionnement d'un contrôleur existant [1]. Cette reproduction est utile lorsque le contrôleur est trop complexe ou trop lent pour être réalisé en temps réel ou trop sensible aux bruits. Parfois le système de contrôle reproduit est un opérateur humain.

Le schéma de la figure 3.1 illustre la manière de reproduction du contrôleur.

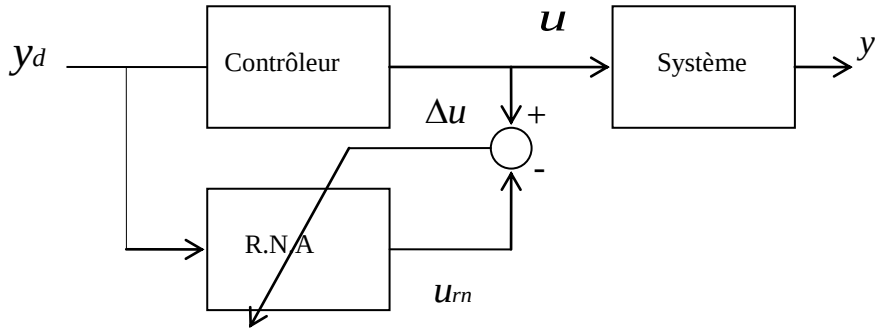


Figure 3.1 Reproduction par réseaux de neurones d'un contrôleur existant

3.2.2 Amélioration d'un système de commande linéaire

Cette approche consiste à utiliser conjointement un contrôleur linéaire classique et un contrôleur neuronal [1]. L'idée est de réaliser une somme des commandes issues des deux contrôleurs, en augmentant progressivement l'importance donnée à la commande préconisée, au fur et à mesure de l'apprentissage de ce dernier. Ce principe est illustré en figure 3.2.

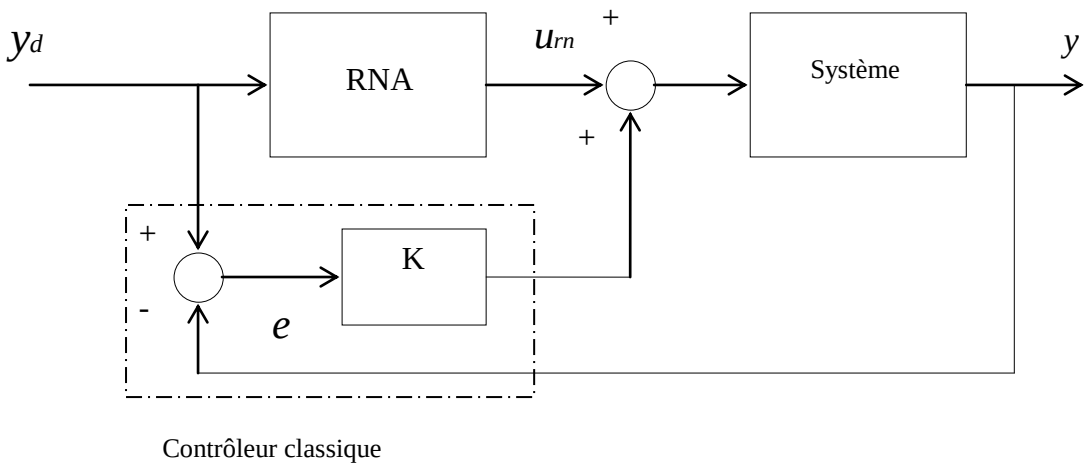
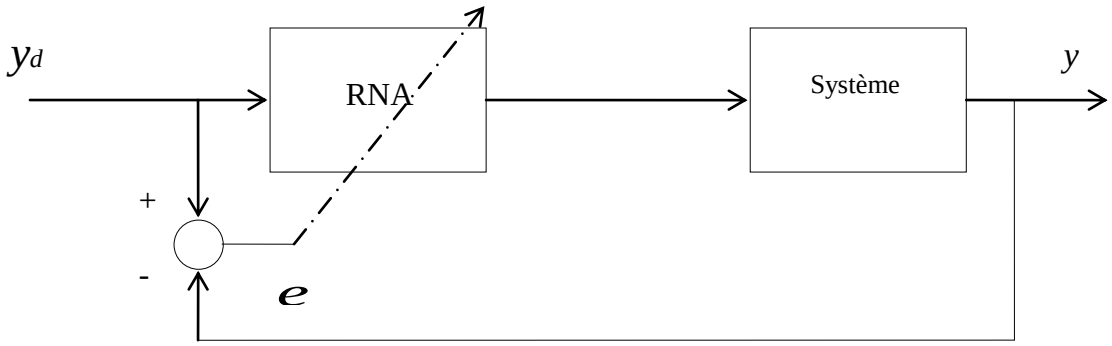


Figure 3.2 Amélioration d'un contrôleur linéaire existant

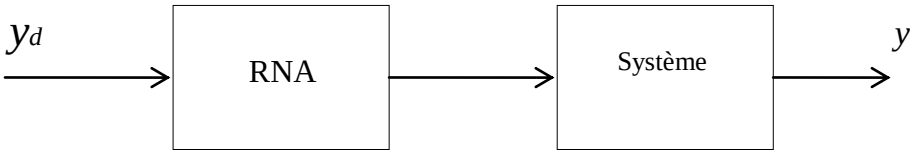
3.3 Control neuronal basé sur le modèle du processus [1], [2], [3]

3.3.1 Contrôle neuronal direct par le modèle inverse du système en boucle ouverte

Dans la plus part des applications de commande en boucle ouverte des systèmes, il est désirable de placer le réseau de neurone en série avec le système pour assurer un contrôle direct ; cela comme indiqué en figure 3.3.



(a): Phase d'apprentissage



(b) : Phase de commande

Figure 3.3 Control par modèle inverse

Le réseau de neurone dans ce cas représente simplement l'inverse du modèle du processus. Si ce modèle est non linéaire son inverse l'est généralement.

La méthode directe nécessite lors de l'apprentissage de savoir le Jacobien du système à commander, parfois difficile à calculer et qui parfois affaiblie l'erreur à rétropropager vers le réseau de neurone.

Dans le cas de certains systèmes (robot manipulateur), des incertitudes ou des variations sur les paramètres du modèle produisent une instabilité du système globale ce qui exige d'assurer

la stabilité du système par un contrôleur classique (PD,...) puis réaliser le modèle inverse de l'ensemble comme illustré en figure 3.4.

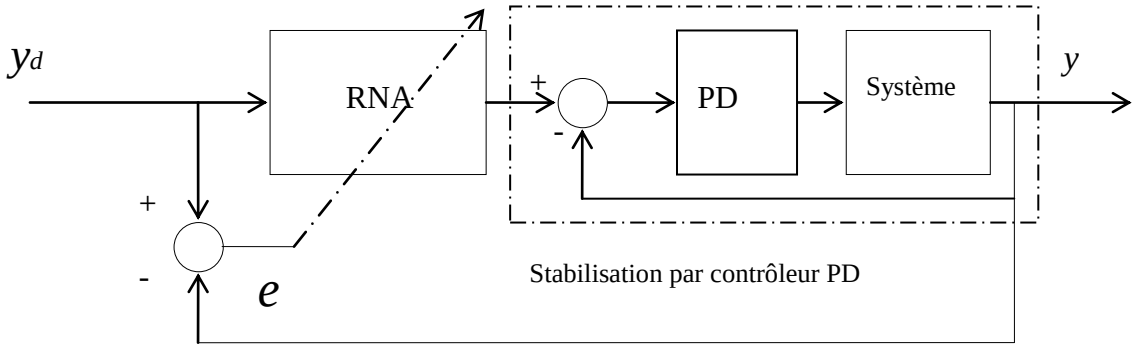


Figure 3.4 La stabilité du système est assuré par un contrôleur classique PD

3.3.2 Contrôle neuronal par modèle interne

Pour surmonter le problème des erreurs de modélisation et les variations des paramètres du processus, on introduit le modèle du processus dans le système de control comme illustré en figure 3.5. L'erreur entre la sortie du processus et son modèle dit « modèle interne » et réintroduit à l'entrée du contrôleur ce qui assure une meilleur robustesse.

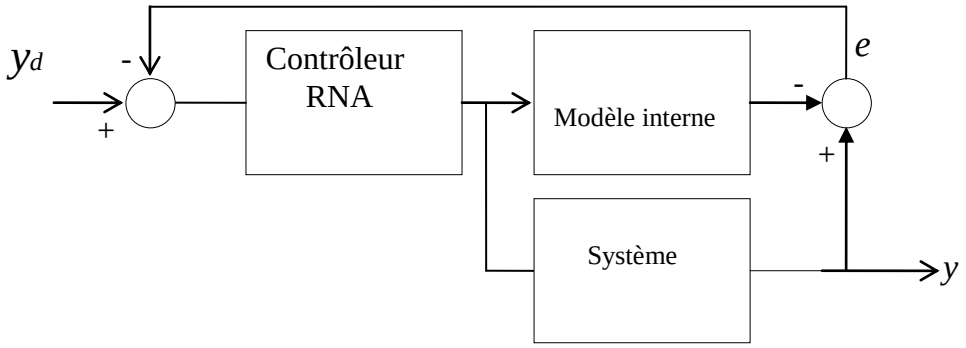


Figure 3.5 Contrôle neuronal par modèle interne

Si le système à commander possède une certaine dynamique il est préférable d'utiliser des réseaux de neurones récurrents en utilisant pour l'apprentissage la méthode de rétropropagation dans le temps.

3.3.3 Contrôle neuronal indirect par modèle inverse (en boucle ouverte)

Parce que le Jacobien du système à commander est obtenu par le biais de son modèle neuronal, obtenu par identification, qu'on dit que le control est indirect.

Le modèle neuronal du système est obtenu en premier lieu par une identification hors ligne puis ajusté on-line pour compenser tout changement des paramètres donc du Jacobient du système. La figure 3.6 expose le schéma du contrôleur.

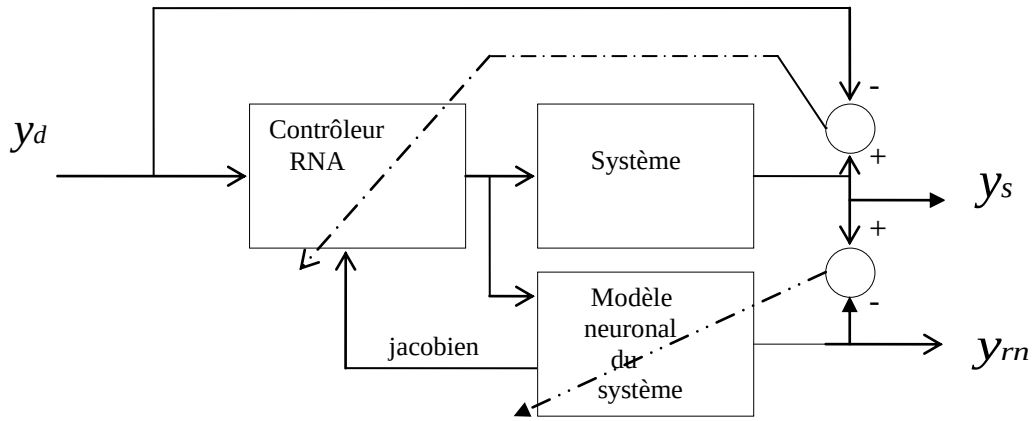


Figure 3.6 Contrôleur neuronal indirect

3.3.4 Contrôle neuronal à apprentissage par l'erreur de contre réaction

Ce schéma de contrôle utilise la commande générée par la contre réaction comme signal d'erreur qu'on doit minimiser lors de l'application de la rétropropagation sur la modèle dynamique inverse du système. Alors la rétropropagation de l'erreur à travers le système à commander n'est pas nécessaire ; c-à-d on évite le calcul du Jocobien.

Le signal d'erreur u_e sert juste pour l'adaptation des poids du contrôleur neuronal, qui sera remplacé par la suite par la commande générée par le modèle inverse neuronal.

Le contrôleur PD assure la stabilité du système au début de l'opération control-stabilité et adaptation des poids ; la figure 3.7 schématise ce principe de control.

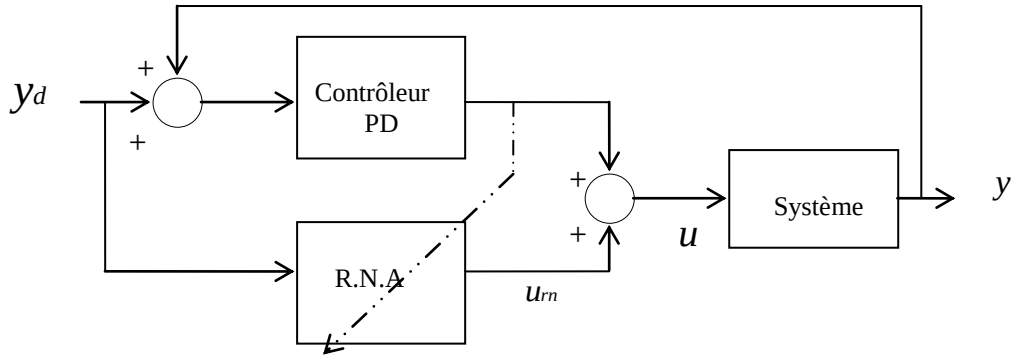


Figure 3.7 Apprentissage par l'erreur de contre réaction

3.3.5 Contrôle neuronal par la technique de compensation de l'entrée de référence

C'est la structure de contrôle la plus simple du fait que l'intervention du contrôleur peut se faire en d'hors de la boucle de retour ce qui évite la modification de la structure du contrôleur initial.

On remarque et d'après le schéma de la figure 3.8 que la sortie du contrôleur neuronal se retrouve à l'entrée du contrôleur générant le signal qui commande le système. Alors on peut dire que la sortie du réseau de neurone est petite en amplitude; ce qui assure une plus grande robustesse au système du fait que la perturbation du système est compensé par une faible valeur à générer par le réseau de neurone contrôleur.

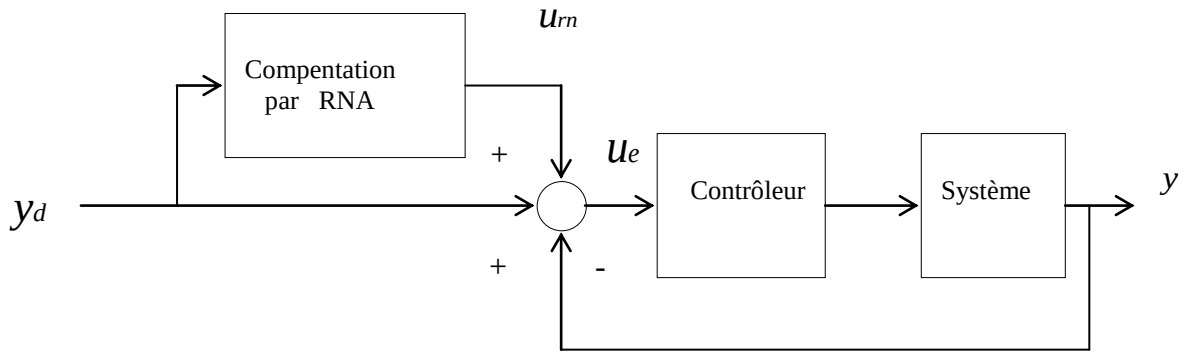


Figure 3.8 Contrôle neuronal par la technique de compensation de l'entrée de référence

3.4 Conclusion: les réseaux de neurones sont des systèmes universels et pour cela on les utilise sur tout pour contrôler des processus complexes non linéaires difficilement modélisable, et à paramètres variants. Un contrôleur neuronal peut intervenir soit juste pour améliorer un régulateur existant ou représenter le modèle inverse d'un système complexe. Une défaillance d'une quelconque cellule du réseau n'entraîne pas la défaillance de tout le système.

BIBLIOGRAPHIE

[1] G.Dreyfus, J.M.Martinez , M.Samuelides, M.B.Gordon, F.Badran, S.Thinria, L.Hérault, 'Réseaux de neurones : méthodologie et applications',Eyrolles 2002.

[2] Fabien moutarde,'cours : Introduction aux réseaux de neurones', Ecole des mines de paris, avril 2007.

[3] Claude Touzert, Cesar-Ornl,'L'apprentissage par renforcement', Masson 1999.

CHAPITRE 4

Les réseaux de neurones pour le contrôle de robot manipulateurs

4. Les Réseaux de neurones pour le contrôle de robot manipulateurs

4.1 Introduction

Au file des années l'industrie, la médicine, l'armé, et le nucléaire ont exigé plus de performances envers le robot manipulateur ; cela à suscité d'améliorer les techniques de commande. La commande classique n'est plus suffisante du faite qu'elle néglige presque le modèle dynamique. L'exploitation totale du modèle dynamique n'est pas possible du fait de la difficulté d'identifier exactement tous les paramètres du modèle qui sont parfois variables. Quoique la commande adaptative a apporté des améliorations elle reste insuffisante du fait qu'elle ne mémorise pas les traces de ses paramètres. Les réseaux de neurones comme jugés par les chercheurs présentent une grande aptitude de généralisation et de mémorisation vis à vis des systèmes non linéaires à paramètres instables. Pour la commande de robots manipulateurs plusieurs solutions sont apportées.

4.2 Contrôle neuronal basé sur le modèle du robot

4.2.1 Contrôle neuronal auxiliaire

Le contrôle auxiliaire par réseaux de neurones vient améliorer les performances du contrôle par 'couple calculé ' (computed torque) des robots manipulateurs. Le contrôle prend la forme directe (feedforward) ou à contre-réaction (feedback) [1], [3], [4], [6].

4.2.1.1 Schémas de control neuronal direct et à contre-réaction

Le contrôleur neuronal par sa sortie vient compenser les incertitudes sur le modèle du robot pour la commande 'couple calculé'.

La différence entre un schéma feedforward et un schéma feedback est que les entrées du premier sont les valeurs désirées $q_d(t)$ et ceux du deuxième sont les valeurs actuelles $q(t)$.

La figure 4.1 et la figure 4.2 montrent les schémas de contrôle.

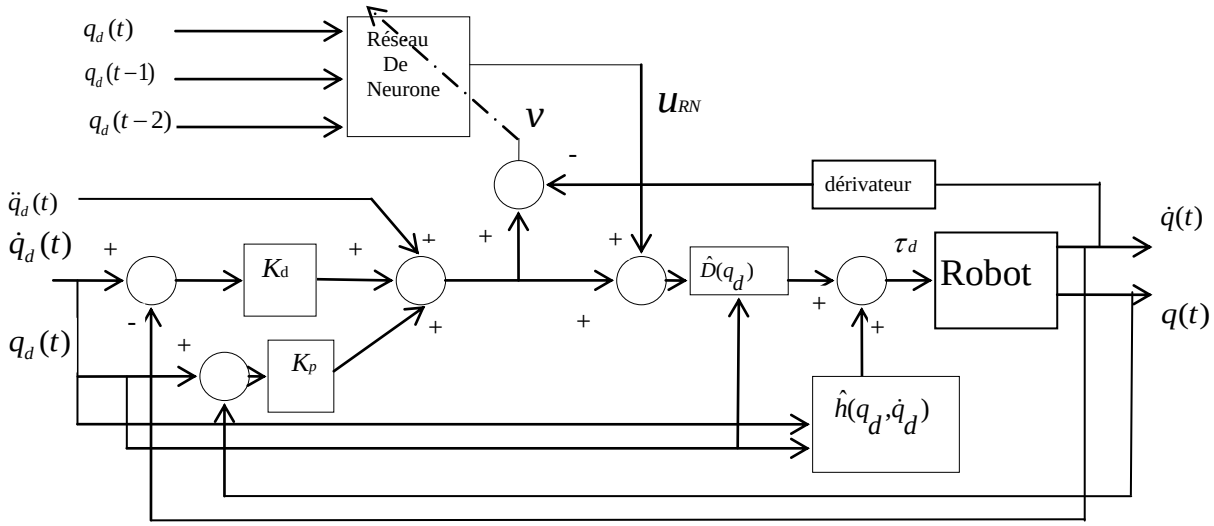


Figure 4.1 Structure du compensateur neuronal direct 'feedforward'

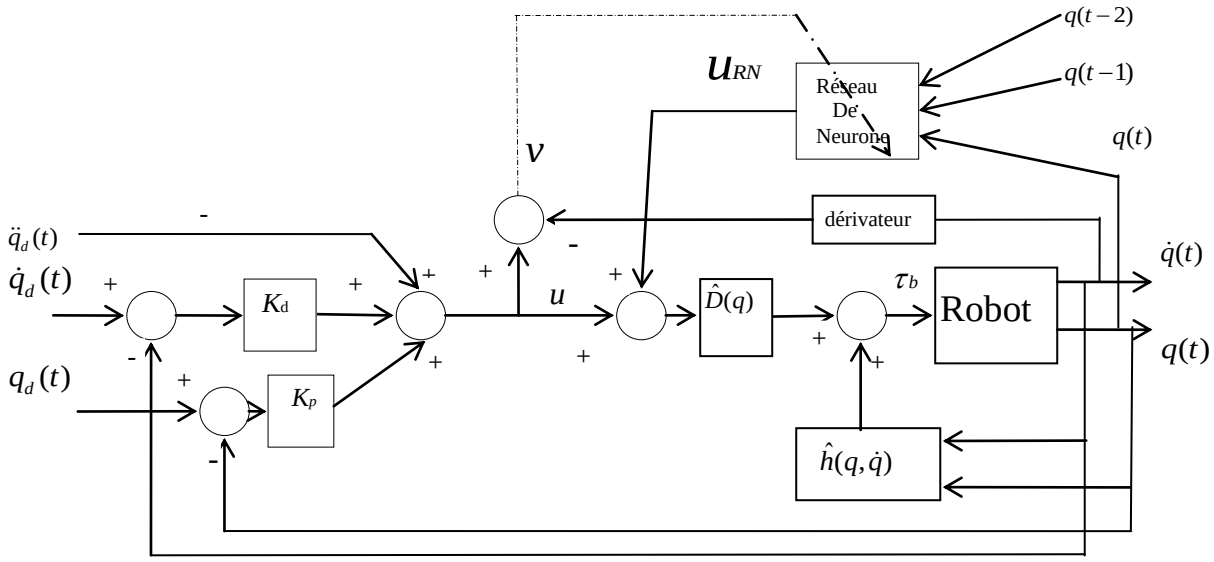


Figure 4.2 Structure du compensateur neuronal à contre-réaction 'feedback '

Le couple de compensation τ_d calculé pour le cas direct (feedforward) est donné par :

$$\tau_d = \hat{D}(q_d)[u + u_{RN}] + \hat{h}(q_d, \dot{q}_d) \tag{4.1}$$

Le couple de compensation τ_b calculé pour le cas à contre-réaction (feedback) est donné par :

$$\tau_b = \hat{D}(q)[u + u_{RN}] + \hat{h}(q, \dot{q}) \tag{4.2}$$

$\hat{D}$ et $\hat{h}$ estimés de D et h.

Les équations précédentes peuvent être représentées par :

$$\tau = \hat{D}[u + u_{RN}] + \hat{h} \tag{4.3}$$

Soit v l'erreur de la boucle fermé (avec $q_d \cong q$ pour le cas direct) donnée par :

$$v = \ddot{E} + K_d\dot{E} + K_pE = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f) - u_{RN} \quad (4.4)$$

Avec, $\Delta D = D - \hat{D}$, $\Delta h = h - \hat{h}$, K_d et K_p matrice des gains.

L'objectif bien sur est de générer la compensation u_{RN} pour réduire v à zéro.

v est considérée comme le signal sur lequel on se base pour assurer l'apprentissage du réseaux de neurone. La valeur de u_{RN} , correspondant a v=0, est :

$$u_{RN} = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f) \quad (4.5)$$

4.2.1.2 Conception du Réseaux de neurone de compensation

Un réseau de neurone multicouche permet de réaliser la compensation par sa sortie u_{RN} assurant la tendance de v vers zéro. L'apprentissage des poids du Réseaux de neurones par la méthode de rétropropagation nécessite un critère J à minimiser qui est choisi comme suit :

$$\mathrm{J} = \frac{1}{2} v^{\mathrm{T}} v \quad (4.6)$$

Donc $$\frac{\partial J}{\partial w} = \frac{\partial v^T}{\partial w} v = -\frac{\partial u_{RN}{}^T}{\partial w} v \quad (4.7)$$

$$\Delta w(t) = \eta \frac{\partial u_{RN}{}^T}{\partial w} v + \alpha \Delta w(t-1) \quad (4.8)$$

η : taux d'apprentissage et α le momentum

4.2.2 Contrôle Neuronal par Modèle inverse

Deux schémas de contrôle sont proposés, où l'un se base sur le Jacobien du système (robot et contrôleur) et l'autre se base sur l'erreur de contre réaction et ne nécessite pas le Jacobien du système.

4.2.2.1-Approche Jacobien : Le principe d'utilisation du Réseau de neurone pour compenser les incertitudes sur le modèle du robot lors d'un contrôle par 'couple calculé' se base sur la

modification de la trajectoire désirée q_d . Le Réseau de neurone joue le rôle d'un filtre non linéaire. Ce principe est illustré en figure 4.3.

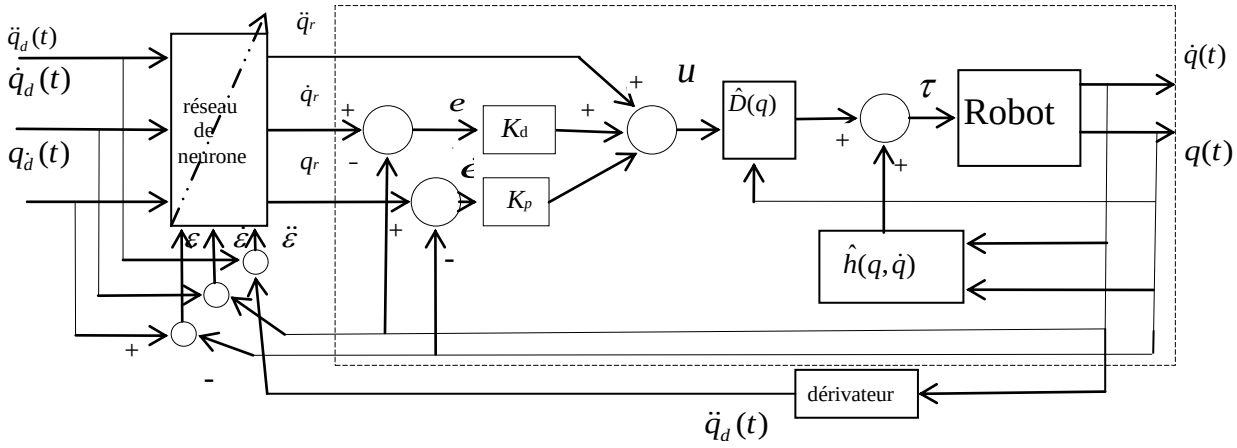


Figure 4.3 Contrôle neuronal inverse pour couple calculé

La trajectoire ($q_d, \dot{q}_d, \ddot{q}_d$) après filtrage par réseau de neurone devient $q_r, \dot{q}_r, \ddot{q}_r$ et sera la trajectoire de référence désirée pour la commande 'couple calculé'.

Le signal d'erreur $\varepsilon = q_d - q$ est le signal d'apprentissage du réseau de neurone. Le réseau de neurone agit comme le modèle inverse du système (robot et contrôleur).

Du fait que le signal d'apprentissage dépend de la sortie du système, alors pour appliquer la méthode de rétropropagation, le Jacobien du système est nécessaire.

D'après la figure 4.3 la commande u du contrôleur est :

$$u = \ddot{q}_r + K_d(\dot{q}_r - \dot{q}) + K_p(q_r - q) \tag{4.9}$$

On utilisant la formule (1.21) concernant la commande 'couple calculé' :

$$(\ddot{q}_r - \ddot{q}) + K_d(\dot{q}_r - \dot{q}) + K_p(q_r - q) = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f) \tag{4.10}$$

Soit aussi:

$$\ddot{\varepsilon} + K_d\dot{\varepsilon} + K_p\varepsilon = \Delta M + (\ddot{q}_d + K_d\dot{q}_d + K_pq_d) - (\ddot{q}_r + K_d\dot{q}_r + K_pq_r) \tag{4.11}$$

Où : $\Delta M = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f)$

Après apprentissage du réseau de neurone par $(\varepsilon, \dot{\varepsilon}, \ddot{\varepsilon})$ comme indiqué en figure ci-dessus et après convergence alors $\varepsilon \to 0$, cela parce que :

$$(\ddot{q}_r + K_d\dot{q}_r + K_pq_r) = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f) + (\ddot{q}_d + K_d\dot{q}_d + K_pq_d) \tag{4.12}$$

On remarque bien que les sorties du réseau de neurone viennent pour assurer une relation qui dépend de la trajectoire désirée et aussi des incertitudes sur le modèle.

Pour améliorer le schéma précédent de façon que les sorties du réseau de neurone assurent une relation qui ne dépend que des incertitudes sur le modèle du robot ; le schéma de la figure4.4 est adopté.

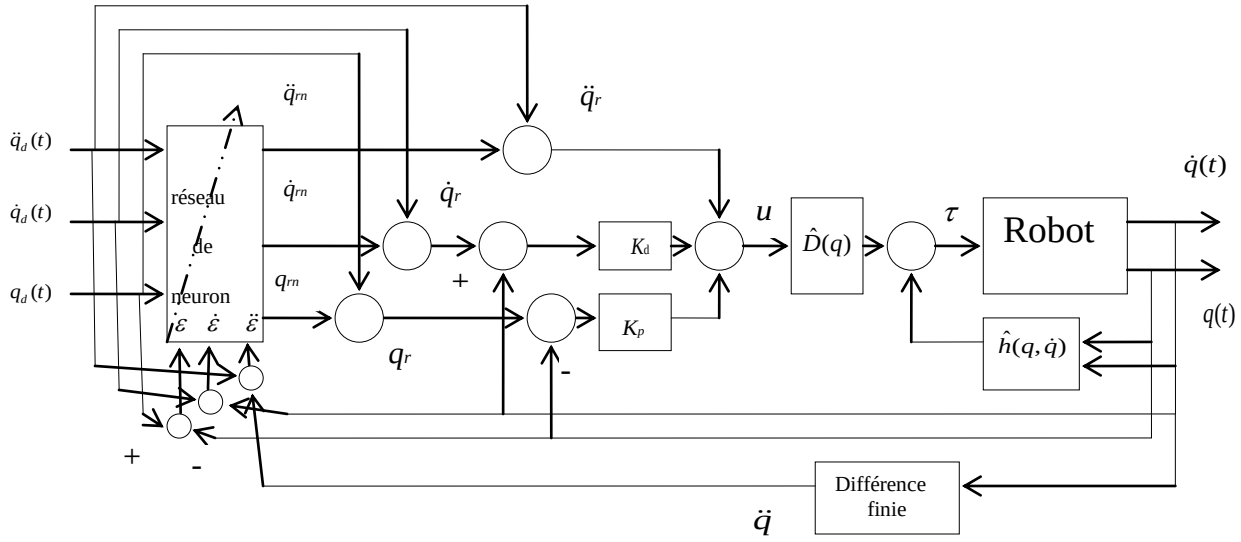


Figure 4.4 Contrôle neuronal inverse modifié pour couple calculé

Dans ce schéma on a :

$$q_r = q_d + q_{rn} \qquad \dot{q}_r = \dot{q}_d + \dot{q}_{rn} \qquad \ddot{q}_r = \ddot{q}_d + \ddot{q}_{rn} \tag{4.13}$$

On remplaçant dans l'équation $q_r, \dot{q}_r, \ddot{q}_r$ on obtient :

$$\ddot{\varepsilon} + K_d\dot{\varepsilon} + K_p\varepsilon = \hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f) - (q_{rn} + K_d\dot{q}_{rn} + K_p\ddot{q}_{rn}) \tag{4.14}$$

Lorsque $\varepsilon \to 0$ Alors : $\hat{D}^{-1}(\Delta D\ddot{q} + \Delta h + \tau_f) = (q_{rn} + K_d\dot{q}_{rn} + K_p\ddot{q}_{rn})$ (4.15)

Cela montre que pour ce schéma le réseau de neurone vient pour compenser les incertitudes du modèle dynamique du robot manipulateur pour améliorer le contrôleur 'couple calculé'.

4.2.2.2 Conception du réseau de neurone de compensation

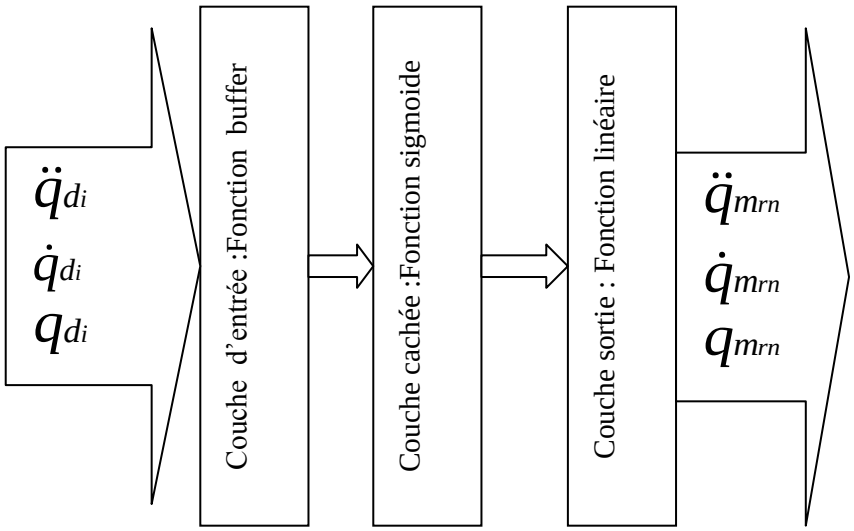


Figure 4.5 Structure en couche du réseau de neurone associe

La loi d'adaptation des poids du réseau de neurones doit minimiser le critère J décrit par :

$$\mathrm{J}(\varepsilon) = \frac{1}{2}\varepsilon^{\mathrm{T}}\varepsilon \tag{4.16}$$

Où $\varepsilon = [\varepsilon, \dot{\varepsilon}, \ddot{\varepsilon}]$; $\varepsilon = q_d - q$

Le gradient de $\mathrm{J}(\varepsilon)$ est donné par :

$$\frac{\partial \mathrm{J}(\varepsilon)}{\partial w} = \frac{\partial \varepsilon^T}{\partial w}\varepsilon = -[\frac{\partial q^T}{\partial w}\frac{\partial \dot{q}^T}{\partial w}\frac{\partial \ddot{q}^T}{\partial w}]\varepsilon \; ; \; \frac{\partial {q_d}^T}{\partial w} = 0 \tag{4.17}$$

Sachant que : $\frac{\partial q^T}{\partial w} = \frac{\partial q^T}{\partial {q_r}^T}\frac{\partial {q_r}^T}{\partial w}$; le calcule de $\frac{\partial q^T}{\partial {q_r}^T}$ nécessite la connaissance du modèle du robot manipulateur donc du Jacobien du système (robot plus contrôleur).

Le gradient de $\mathrm{J}(\varepsilon)$ est donné par :

$$\frac{\partial \mathrm{J}(\varepsilon)}{\partial w} = -[\frac{\partial {q_r}^T}{\partial w}\frac{\partial {\dot{q}_r}^T}{\partial w}\frac{\partial {\ddot{q}_r}^T}{\partial w}].Jacob.\begin{bmatrix}\varepsilon \\ \dot{\varepsilon} \\ \ddot{\varepsilon}\end{bmatrix} \tag{4.18}$$

Alors la loi d'adaptation des poids est comme suit :

$$\Delta w(t) = -\eta \frac{\partial J^T}{\partial w} + \alpha \Delta w(t-1) \tag{4.19}$$

4.3 Contrôle neuronal non basé sur le modèle du robot

4.3.1 Introduction : dans ce cas on utilise l'erreur à minimiser de contre réaction d'un contrôleur classique PD pour ajuster par rétropropagation les poids du réseau de neurone ; cela sans avoir besoin du Jacobien du système robot- contrôleur.

Le contrôleur PD assure au démarrage le contrôle et la stabilité du robot, et au fur et à mesure le réseau de neurone vient remplacer le contrôleur PD par minimisation de l'erreur de contre réaction. Le Réseau de neurone représentera donc le modèle inverse du robot manipulateur.

4.3.2 Contrôleur PD pour robot manipulateur

Le contrôle classique PD pour robot manipulateur est souvent utilisé parce que il est simple. Le contrôle PD est valable sous conditions, tel que, fonctionnement en faible vitesse, ou fonctionnement autour d'un point, alors le control PD est local. Le schéma de la figure 4.6 illustre le contrôleur PD dédie au robot manipulateur.

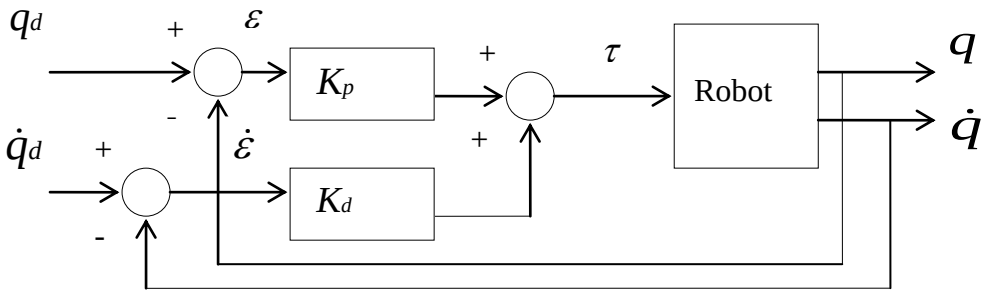


Figure 4.6 Contrôleur PD pour robot manipulateur

Le couple de commande est égale à : $\tau = K_d\dot{\varepsilon} + K_p\varepsilon$; $\varepsilon = q_d - q$ (4.20)

L'équation de l'erreur en boucle fermée est donnée par :

$$\tau = D\ddot{q} + h + \tau_f = K_d\dot{\varepsilon} + K_p\varepsilon \tag{4.21}$$

Peut être écrite sous la forme :

$$\ddot{\varepsilon} + D^{-1}K_d\dot{\varepsilon} + D^{-1}K_p\varepsilon = D^{-1}(D\ddot{q}_d + h + \tau_f) \tag{4.22}$$

D'après cette équation le contrôle PD dépend fortement de la dynamique du robot (D^{-1}) de même du point de fonctionnement $(q_d, \dot{q}_d, \ddot{q}_d)$.

Pour minimiser l'effet de la dynamique du robot on choisit des gains suffisamment grands sans altérer à la stabilité du système robot-controleur.

Si $K_d \gg 1$ *et* $K_p \gg 1$ Alors on à :

$$\dot{\varepsilon} + K_d^{-1} K_p \varepsilon = K_d^{-1} D\ddot{\varepsilon} + K_d^{-1}(D\ddot{q}_d + h + \tau_f) \cong 0 \qquad (4.23)$$

Soit : $\dot{\varepsilon} + K_d^{-1} K_p \varepsilon \cong 0$ (4.24)

4.3.3 Contrôle neuronal avec apprentissage par contre réaction

Le schéma de la figure 4.7 présente la possibilité de faire l'identification et le contrôle du robot manipulateur en-ligne avec l'aide d'un contrôleur PD qui assure la stabilité et la commande du robot [1],[2],[5].

Cette phase transitoire, où les contrôleurs PD et neuronal (le réseau de neurone représente le modèle inverse du robot) fonctionnent ensemble, aboutie à l'identification des poids du réseau de neurones par la méthode de rétropropagation en se basant sur l'erreur de contre réaction et sans avoir besoin du Jacobien du système.

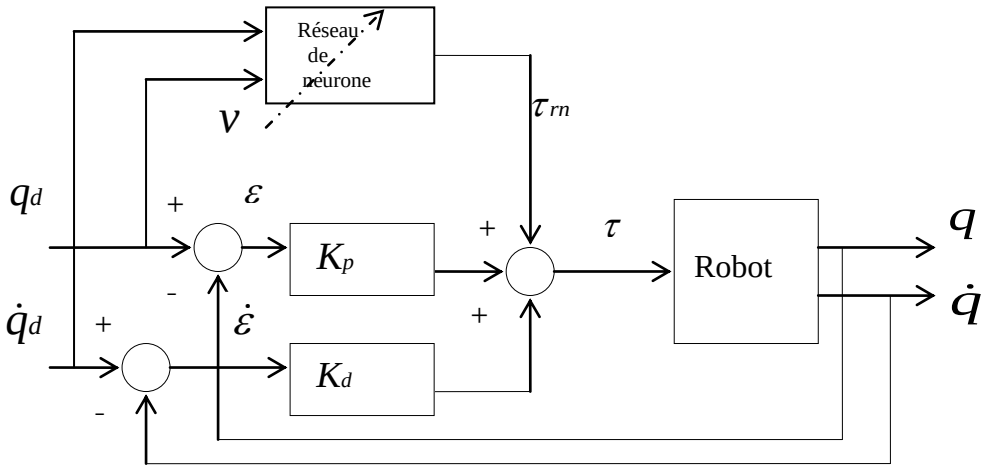


Figure 4.7 Apprentissage et contrôle par contre-réaction

L'équation de commande est : $\tau = K_d \dot{\varepsilon} + K_p \varepsilon + \tau_{rn}$ (4.25)

L'équation dynamique de l'erreur peut être écrite par :

$$\tau = K_d \dot{\varepsilon} + K_p \varepsilon + \tau_{rn} = D\ddot{q} + h + \tau_f$$

Soit aussi : $K_d \dot{\varepsilon} + K_p \varepsilon = (D\ddot{q} + h + \tau_f) - \tau_{rn}$ (4.26)

Considérant le signal d'apprentissage $v = K_d\dot{\varepsilon} + K_p\varepsilon$. Minimiser v permet $\varepsilon \to 0$

Ce qui découle :

$$(D\ddot{q} + h + \tau_f) - \tau_{rn} = 0 \text{ Ou } \tau_{rn} = (D\ddot{q} + h + \tau_f) \quad (4.27)$$

Alors le réseau de neurone représentera après convergence de l'apprentissage de ses paramètres, le modèle inverse du robot. De même la participation du contrôleur PD devient minime.

4.3.3.1 Conception du réseau de neurone

Un réseau de neurones multicouches est choisi, et la méthode de rétropropagation est utilisée pour l'apprentissage des ses poids.

Le critère à minimiser J est donné par :

$$\mathrm{J}(w) = \frac{1}{2} v^{\mathrm{T}} v \; ; \qquad v = K_d\dot{\varepsilon} + K_p\varepsilon \quad (4.28)$$

Et

$$\frac{\partial J}{\partial w} = \frac{\partial v^T}{\partial w} v = -\frac{\partial \tau_{rn}{}^T}{\partial w} v$$

L'adaptation des poids est assurée par la relation suivante :

$$\Delta w(t) = \eta \frac{\partial \tau_{rn}{}^T}{\partial w} v + \alpha \Delta w(t-1) \quad (4.29)$$

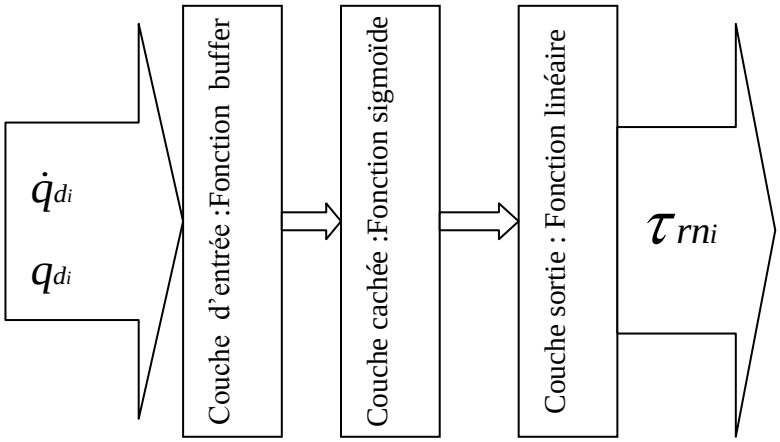


Figure 4.8 Structure en couche du réseau de neurone associe

4.4 Conclusion: Le contrôle de robot manipulateur par des réseaux de neurones manifeste plusieurs solutions. Si nous possédons un contrôleur à couple calculé imprécis on ajoute un compensateur neuronale direct ou en contre réaction pour affiner la commande et annuler l'erreur possible. Si la solution précédente est insuffisante on considère le modèle inverse de l'ensemble contrôleur dynamique et robot. Parfois on veut réaliser un contrôleur par modèle inverse du robot mais sans être exposé à des instabilités et ce la exige de faire une identification en ligne des poids du réseau parallèlement à la présence d'un contrôleur classique (PD) réduisant au fur et à mesure son rôle.

BIBLIOGRAPHIE

[1]Seul Jung and T.C. Hsia,'A neural neutwork control technique for robot manipulators', NITTA corporation of japan, 1996.

[2] H. Daniel Patino, Ricardo Carelli and Bejamin R. kuchen,' neural networks for advanced control of robot manipulators', IEEE transaction of neural networks, vol. 13, no. 2, march 2002.

[3] Young h; kim and Frank l. Lewis,' optimal design of CMAC neural-network controller for robot manipulators',IEEE ,Transaction on systems, Man, and cybernitics, vol. 30, No.1 February 2000.

[4] Tomochica Ozaki, Tatsuya Suzuki, Takeshi Furuhachi, Shigeru Okuma, and Yoshiki Uchikawa,' Trajectory control of robotic manipulators using neural networks', IEEE, transaction on industrial electronics, vol;38, No.3, June 1991.

[5] Yutaka Maeda, and Rui J. P. Figueiredo,' Learing rules for neuro-controller via silmultaneous perturbation',IEEE, transaction on neural networks, vol. 8, No.5, septembre 1997.

[6] David A. Handelman, Stephen H. Lan, and Jack J. Gelfand,'Integrating neural networks and knowledge-based systems for intelligent robotic control', IEEE, 1990.

[7] Charles W. Anderson ,'Learning to control an inverted pendulum using neural networks',American control conference ,Atlanta,Georgia, April 1989.

[8] Vicente Ruiz de angulo and Carme Torras, ' Self-Calibration of space robot', IEEE Transaction on neural networks , vol..8. No.4, Julie 1997.

CHAPITRE 5

Robotique mobile

5. Robotique mobile

5.1 Introduction

Les robots manipulateurs se trouvent plongés dans beaucoup de domaines industriels et la recherche scientifique nous a permis avec le temps d'augmenter leur vitesse, leur précision et leur capacité en charges. Ces robots manipulateurs se trouvent au fur et à mesure adaptés aux tâches confiées.

Toute fois les robots manipulateurs se sont confrontés à la contrainte de la mobilité, ce qui limite leur espace atteignable. Néanmoins, les progrès technologiques ont conduit à la création d'un nouveau type de robot complètement différent des robots manipulateurs, aussi bien dans sa conception comme dans la tâche à accomplir c'est le robot mobile.

La caractéristique principale des robots mobiles est l'absence d'un lien mécanique qui fixe le robot à l'environnement. Ces robots sont donc capables d'évoluer librement dans leurs espaces de travail. Cette caractéristique de 'mobilité' ouvre grandement les portes d'applications autres que le milieu industriel. Les robots mobiles sont de plus en plus utilisés dans plusieurs domaines tels que les milieux hostiles (milieux nucléaires), la lutte contre les incendies, l'assistance aux personnes handicapées, et les milieux sous-marins.

La recherche en robotique mobile englobe la conception mécanique, les lois de contrôle, la perception de l'environnement, la planification de trajectoires sans collision, la modélisation dynamique et cinématique, et coopération entre robots mobiles.

5.2 Les classes de robots mobiles

Le moyen de locomotion est la caractéristique la plus importante pour un robot mobile et cela dépend du terrain et aussi de l'application. On peut donc identifie des robots mobiles à roues, à chenilles et à pattes.

5.2.1 Robots mobiles à roues : C'est le type le plus répondu, et opère dans des sites aménagés d'intérieure ou parfois d'extérieure. Il est dit de type voiture s'il possède deux roues motrices et deux roues de direction. Ce type de robot présenté en figure 5.1 ne peut pas se déplacer latéralement ce qui complique la planification des trajectoires.

(a) : Dala (b) : Lama (c) : ExoMars Rover

Figure 5.1 Robot mobile à roues (Dala , Lama et ExoMArs)

5.2.2 Robots mobiles à chenilles : Lorsque le robot mobile à roues se trouve incapable de franchir des terrains accidentés il est bon de l'équiper par des chenilles pour avoir une bonne adhérence au sol comme présenté en figure 5.2.

Figure 5.2 Robot mobile à chenille

5.2.3 Robots mobiles à pattes : C'est la solution parfaite lorsque le terrain possède des différences d'altitude considérables. La modélisation et le contrôle d'un robot à pattes sont plus complexes, comme illustré en figure 5.3.

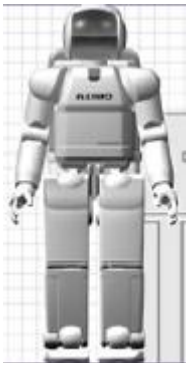


Figure 5.3 Robot mobile à pattes (Asimo)

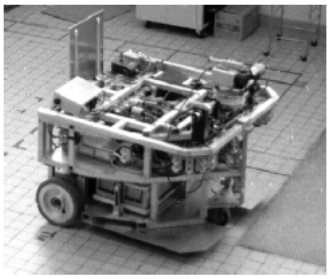
5.3 Quelque exemples de robots mobiles

5.3.1 Robot SHAKEY : c'est l'un des premiers robots mobiles autonomes doté des capacités de perception, d'analyse et de planification et équipé de deux roues motrices. Ce robot est

développé au Standford Research institute (USA ,en 1967) et se déplace à la vitesse de 2m/heure.

5.3.2 Robot AGROS : c'est à l'université de Paul Sabatier de Toulouse (France, en 1978) que se robot est développé et il est équipé d'un système de vision.

5.3.3 Robot HILARE : développé au laboratoire LAAS de Toulouse (France, en 1979) pour la planification de trajectoires d'un robot mobile ponctuel.



Hilare, LAAS-CNRS, Toulouse, 1977 [Giralt 84]
Entraînement : batteries au plomb 24V, 2 moteurs DC avec codeurs incrémentaux
Calculateur : 4 processeurs Intel 80286, pas d'OS, multibus, modem série radio à 9600 bauds
Capteurs : odométrie, 16 capteurs US, un télémètre laser
Dimensions $(L \times l \times h)$: $80\ cm \times 80\ cm \times 60\ cm$
Poids : $400\ kg$

Figure 5.4 Robot mobile Hilare

5.3.4 Robot JASON : Développé à l'université Berkeley (USA, en 1981), équipé de caméras, capteurs à ultrasons et de diodes infrarouges pour réaliser la détection d'obstacles. L'environnement est modélisé par une grille qui est remplie à partir des informations données par l'ensemble de perception.

5.3.5 Robot Pionner : Conçu au 'Standford Research Institute' (USA, en1993) à un prix abordable possédant un ensemble de capteurs intéroceptifs et extéroceptifs qui servent à la planification de trajectoires, figure 5.5.

Pioneer P3-DX, ActiveMedia Robotics, 2004 [ActivMedia 04]
Entraînement : batteries $252\ Wh$, 2 moteurs DC avec codeurs incrémentaux
Calculateur : micro-contrôleur Hitachi HS-8, I/O Bus, 2 ports série
Capteurs : odométrie, 8 capteurs US en façade + options (bumpers, télémètre laser, gyroscope)
Autonomie : $24 - 30\ h$
Vitesse : maximum $1,6\ m/s$
Dimensions $(L \times l \times h)$: $44\ cm \times 38 \times 22\ cm$
Poids : $9\ kg$ (charge admissible : $23\ kg$)

Figure 5.5 Robot mobile Pioneer P3-DX

5.4 Architectures des planificateurs de trajectoires

5.4.1 Introduction : Parce que le robot mobile se déplace dans des environnements qui sont généralement encombrés d'obstacles statiques et dynamiques, alors cela nécessite la planification de sa trajectoire pour le mener à son but. Lorsque on peut avoir une information globale sur la scène sujette du mouvement du robot mobile, une planification globale est possible ; dans le cas contraire une planification locale (réactive) doit être réalisée.

5.4.2 Planification Globale : on vise par cette architecture à reproduire le mode de raisonnement humain. La planification est donc décomposée en série d'opérations successives décrites par la figure 5.6.

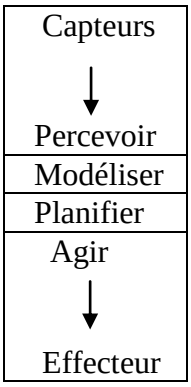


Figure 5.6 Planification globale

La première opération consiste à traiter les données sensorielles qui fournissent au robot des informations sur son environnement. La deuxième étape consiste en la construction (ou la mise à jours) à partir des informations capteurs, du modèle de l'environnement tel que la position et le type des obstacles.

L'étape suivante est la planification du chemin à suivre par le robot mobile pour mener à bien sa tâche.

L'étape finale à pour but de calculer puis d'exécuter les actions permettant au robot mobile de suivre le plan généré par l'étape précédente.

Les étapes modélisation de l'environnement et planification de la trajectoire sont gourmandes en calcul et ceci notamment en milieu peu structuré. Alors le temps restant pour l'action devient plus petit.

On peut dire sur les architectures globales que la vitesse de réaction est très faible, donc on ne peut pas prendre en considération les obstacles dynamiques. Un autre inconvénient fait que

l'opération de planification devient plus complexe lorsqu'on veut de la robustesse vis-à-vis des incertitudes de perception.

L'utilisation de cette approche globale est limitée aux robots mobiles évoluant dans des environnements statiques et dont la structure est fortement contrainte et connue à priori. Les architectures délibératives possèdent un point fort important, qui est leur capacité à prendre en compte des raisonnements de haut niveau lors de la phase de planification. Il leur est possible en effet de gérer des missions complexes enchaînant plusieurs tâches et buts successifs.

5.4.3 Planification locale (réactive) : Dans le cas où on ne possède pas une vue globale de l'environnement ; parce que les obstacles et les buts à atteindre sont dynamiques; alors on passe à une planification locale réactive où le robot réagit continûment à son environnement. En partant de cette constatation on peut définir une architecture de commande basée sur la composition de plusieurs modules implémentant des comportements simples. Le comportement global du robot est alors le résultat de la composition des divers comportements élémentaires. Chacun des comportements, qui peuvent être par exemple l'évitement d'obstacle ou se diriger vers le but, réalise un couplage direct entre les capteurs et les effecteurs du robot ; ce qui permette une rapidité d'exécution. La trajectoire que le robot doit suivre ne peut être connue d'avance. Parmi les architectures comportementales celle proposée par Brooks.

5.4.3.1 Architecture de Brooks : C'est l'architecture la plus connue qui a donné naissance à plusieurs travaux. La méthode de Brooks [1], donnée en figure 5.7, s'appuie sur une décomposition verticale en niveaux de compétences. Chacun de ces niveaux correspond à un comportement indépendant recevant des données capteurs et agissant sur les effecteurs. Chaque module est capable d'inhiber ceux de niveau inférieur au sien ou de les utiliser pour son propre traitement. Cette architecture ne possède aucun arbitre central permettant de choisir l'un ou l'autre des comportements. Ce choix se fait par la réalisation des différents niveaux et le précâblage des communications possible entre eux.

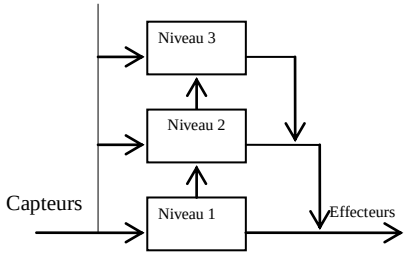


Figure 5.7 Hiérarchie des niveaux dans l'architecture de Brooks

Cette architecture possède les points forts suivants :

- La rapidité des réponses lui permet d'évoluer dans des environnements fortement dynamiques.
- Sa robustesse qui découle du parallélisme et de l'indépendance relative des comportements, permet au robot de continuer à fonctionner en cas de panne de l'un d'entre eux.
- Possibilité d'ajouter aisément des niveaux supérieurs.

L'architecture de Brooks possède bien l'inconvénient des architectures locales, qui est le manque du raisonnement haut niveau.

5.4.4 Les approches hybrides : Du fait que les approches purement réactives ou purement délibératives sont des extrêmes, les chercheurs tentent de combiner les deux approches pour mieux profiter de leurs avantages. Cette approche dite hybride peut favoriser une approche ou l'autre.

5.4.4.1 Approche à composante globale dominante

5.4.4.1.1 Architecture de Payton : Cette architecture proposée par D .W.Payton [9] montrée en figure 5.8, se base sur une décomposition verticale et de quatre modules à caractères hiérarchique:

- Un planificateur de missions chargé de définir une série de buts géographiques et de spécifier les contraintes sur le déplacement.
- Un planificateur de chemins chargé de relier les buts géographiques par des chemins à partir d'un modèle global de l'environnement.
- Un planificateur local dont le rôle est de sélectionner les manœuvres qui doivent permettre au véhicule de suivre les chemins issus du niveau supérieur.
- Un planificateur réflexe chargé de la commande en temps réel du véhicule.

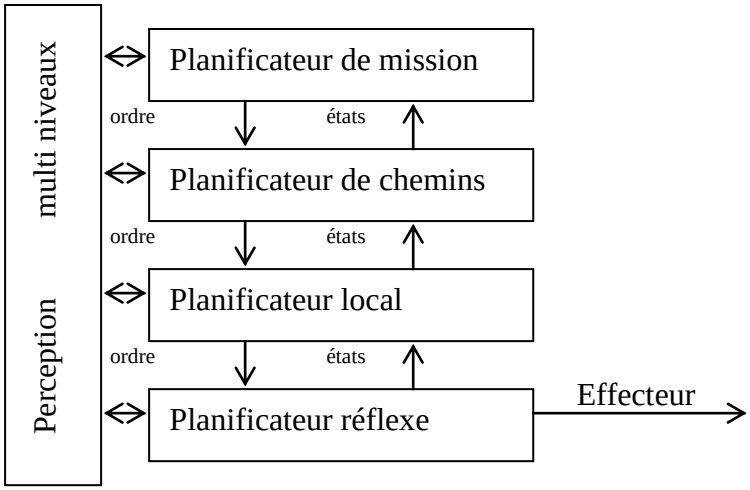


Figure 5.8 Architecture de Paytron

Le niveau supérieur envoi des ordres aux niveaux inférieur et reçoit des informations sur leurs états.

5.4.4.1.2 Architecture TCA de Simmons : L'architecture TCA (Task Control Architecture) présentée par R.G.Simmons [10] rompt avec l'aspect hiérarchique et choisi un aspect centralisé. Elle se compose d'un nombre arbitraire de modules, suivant les besoins, spécialisé chacun dans une tâche dialoguant avec un module de gestion central. Le module central gère les priorités entre les tâches suite aux messages reçus. Des mécanismes réflexes d'urgence sont présents pour répondre aux circonstances.

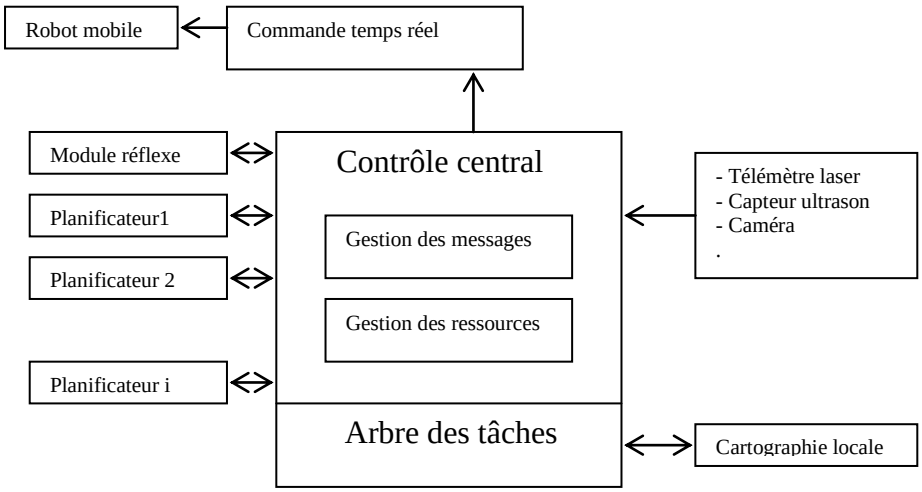


Figure 5.9 Architecture TCA de Simmons

5.4.4.2 Architecture à forte composante réactive

5.4.4.2.1 Architecture AuRA d'Arkin

L'approche proposée par R.C.Arkin [2],[3] s'appuie sur le principe de la perception et de l'action, nommé dans son schéma par 'gestionnaire de schémas moteurs', il s'agit par exemple d'un déplacement en ligne droite, d'un suivi de mur ou d'un évitement d'obstacle.
L'architecture AuRA (Autonomous Robot Architecture) se compose de sous système : perception, cartographie, planification et moteur, comme montré en figure 5.10.

- Le sous système de perception traite les données capteurs (ultrason, vision, capteurs internes) et fournit les résultats au cartographe et au gestionnaire de schémas moteur.
- Le cartographe est de deux types : le premier est statique, maintien une information sur l'environnement connu à priori. le deuxième est dynamique reproduit tout changement au niveau de l'environnement indiqué par les capteurs.

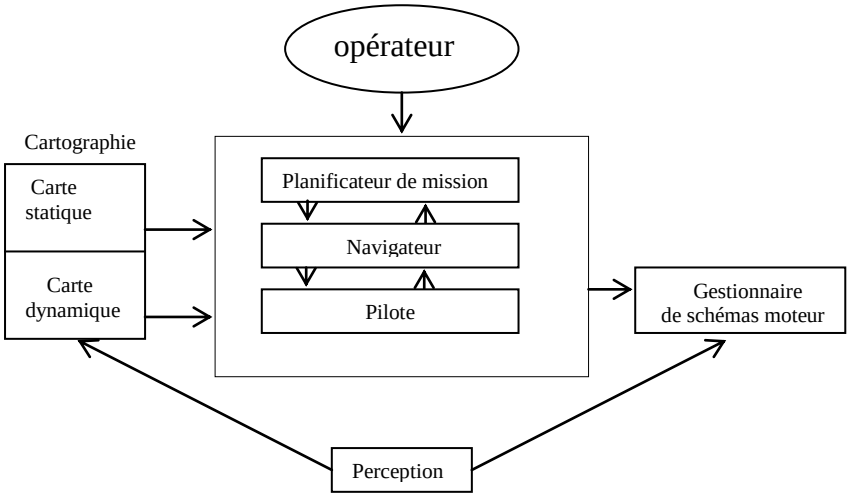


Figure 5.10 Architecture Au.R.A

- Le système de planification et avec ses trois modules, réalise les opérations suivantes :
  - ✓ Le planificateur de mission fournit un certain nombre de sous buts à partir des spécifications de la mission.
  - ✓ Le navigateur délivre un chemin permettant de les relier.
  - ✓ Le pilote détermine les éléments moteurs qui doivent être exécuté.
- Le système moteur exécute les éléments désignés par le pilote en faisant le lien avec le niveau physique du moteur.

5.4.5 Architecture à 3 niveaux

C'est parmi les architectures les plus récentes où les auteurs voient la nécessité d'un niveau intermédiaire entre globale et locale (réactive).

5.4.5.1 Architecture 3T : cette architecture proposée par R.P.Bonasso [4], est de trois parties : un planificateur pour la partie délibérative, des mécanismes de commande réactifs et modules intermédiaires chargés de gérer les séquences d'actions. Ces trois niveaux sont appelés par Bonasso 'Controller, sequencer, and déliberator'.

- La couche 'Controller' regroupe les algorithmes implémentant des comportements réactifs, tel que suivi de mur et évitement d'obstacle. Ce niveau est la bibliothèque utilisée sous demande de niveau supérieure.
- Le niveau intermédiaire 'sequencer' sélectionne quel groupe de comportements doit être active à un moment donné.
- La couche supérieure fourni les différents séquences nécessaires au séquenceur pour mener à bien l'exécution du plan.

5.4.5.2 Architecture LAAS : L'architecture proposée par le laboratoire LAAS de Toulouse pour la plate-forme du robot mobile HILAR, exposée en figure 5.11, se compose de 3 niveaux appelés niveau fonctionnel, niveau d'exécution et niveau décisionnel [5].

- Le niveau décisionnel prend en charge la planification et la décision. Divisé en deux parties : un planificateur et un superviseur. Le planificateur produit les séquences d'actions nécessaires pour réaliser le but recherché. Le superviseur utilise le planificateur comme ressource et interagisse avec le niveau d'exécution et contrôle le déroulement du plant.
- Le niveau d'exécution est un système purement réactif sans possibilité de planification. Il reçoit du niveau décisionnel les séquences d'actions à exécuter, puis il sélectionne et paramètre dynamiquement les modules de niveau fonctionnel.
- Le niveau d'exécution est un système purement réactif sans possibilité de planification. Il reçoit du niveau décisionnel les séquences d'actions à exécuter, puis il sélectionne et paramètre dynamiquement les modules de niveau fonctionnel.

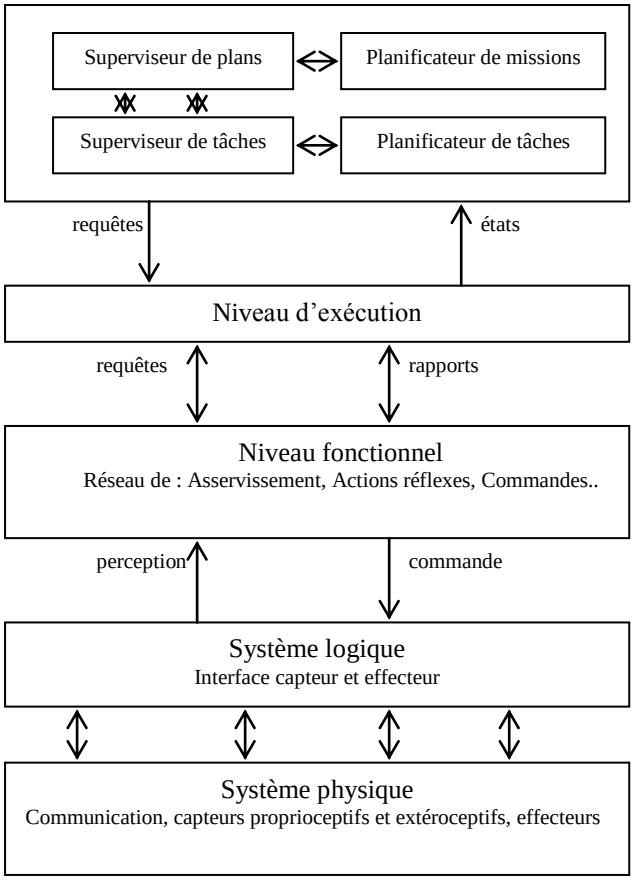


Figure 5.11 Architecture développée au laboratoire LAAS de Toulouse

- Le niveau fonctionnel inclut toutes les fonctionnalités nécessaires pour la perception et l'action. Ces fonctionnalités peuvent être des fonctions de traitement des données, des boucles de commande et de régulation ou des opérations réflexes. Chaque opération dialogue avec les autres par des messages. Alors cette couche est un réseau de modules interagissant les uns avec les autres.

5.5 Etat de l'art des méthodes réactives

5.5.1 Types de méthodes réactives : Les méthodes réactives sont nombreuses, proposées par plusieurs hauteurs aux files des années. Ces méthodes sont intégrées en architecture locale ou hybride.

5.5.2 Méthode champs de potentiel : La méthode est proposée par O.khatib en 1986 pour assurer la navigation d'un bras manipulateur avec évitement d'obstacles. Ça consiste à simuler le robot à une particule contrainte à se déplacer dans un champ de potentiel fictif obtenu par la composition d'un premier champ attractif (atteindre la cible) et d'un ensembles

de champs répulsifs modélisant la présence d'obstacles dans l'espace d'évolution du robot et qu'il faut éviter. A chaque position du robot, une force résultante de l'action conjuguée des obstacles et du but est calculée et correspond à la direction à suivre. Cette méthode est adaptée par la suite aux robots mobiles. Cependant il faut noter que ces méthodes purement réactives sont sujettes à des minima locaux et provoquent le blocage du robot en sa position. Elles peuvent entraîner un mouvement oscillatoire du robot dans certaines situations (couloires étroits).Ces problèmes sont résolus par fois par des solutions heuristiques comme indiqué par Patrick Reignier [6].

5.5.3 Méthode de la fenêtre dynamique : C'est une technique proposée par D.Fox [7] qui travaille dans l'espace des commandes du robot. La taille du domaine de recherche des vitesses permises, n'entraînant pas de collisions, est réduite par la prise en compte explicite du modèle cinématique du système. Les commandes envoyées au robot sont le résultat de la maximisation sur ce domaine de recherche d'une fonction coût liée à la configuration finale.

5.5.4 Méthode de la Bande élastique : Ce concept de bande élastique a été proposé par Quinlan [8], utilise une trajectoire initialement planifiée qui est représentée par une série de boules adjacentes dans l'espace des configurations. Le rayon d'une boule centrée en une configuration est la distance de cette configuration à l'obstacle le plus proche. Ainsi une trajectoire est sans collision dés que les boules qui la composent se recouvrent. Développée pour des systèmes sans contraintes cinématiques, cette technique considère la trajectoire comme une bande élastique se modifiant sous l'action de forces internes de contraction ou d'élasticité.
Du fait de la représentation en boules, la mise à jour de la trajectoire sous l'action des forces est très rapide.

5.5.5 Méthodes Roadmaps : Elles sont basées sur la construction d'un graphe connectant la configuration initiale à la configuration finale. Parmi ces méthodes on nomme le diagramme de Voronoï, Les graphes de visibilité.

5.5.5.1 Diagramme de Voronoï : Les premières 'roadmaps' apparues étaient basées sur le diagramme de Voronoï, qui donne, dans un environnement polygonal plan, les lignes d'égale distance aux obstacles, cela est montré en figure 5.12. Ces lignes permettent théoriquement de naviguer de manière sûre au plus loin des obstacles, en reliant la configuration initiale au

point le plus proche du graphe et en naviguant sur le graphe jusqu'au point le plus proche de la configuration finale, avant de rejoindre celle-ci.

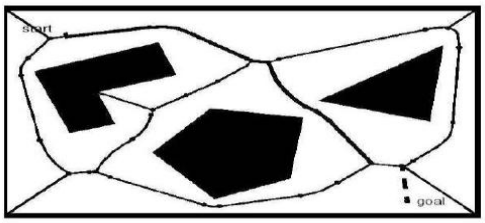


Figure 5.12 Planification de trajectoire
à l'aide d'un diagramme de voronoï

5.5.5.2 Graphes de visibilité : C'est une technique utilisée sur le premier robot nommé Shakey. Comme montré en figure 5.13, elle permet à partir des sommets d'obstacles polygonaux, de capturer l'ensemble de la topologie de l'environnement à l'aide d'un graphe reliant une configuration initiale à une configuration finale. Le graphe est construit comme suit : on relie le point de départ aux sommets des obstacles qu'il peut voir. On fait de même avec le point final que l'on souhaite atteindre ; on obtient un premier graphe. Ensuite, on reproduit l'algorithme en considérant tour à tour chacun des nouveaux nœuds du graphe ainsi crée. Une fois le graphe final construit, on cherche un chemin dans celui-ci. Le chemin passera au plus proche des obstacles.

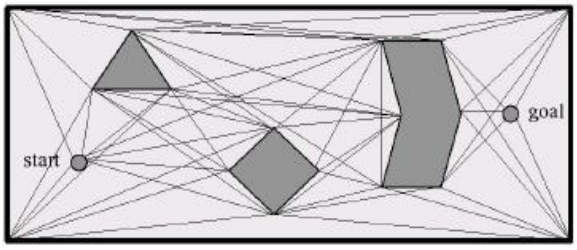


Figure 5.13 Construction d'un graphe de visibilité

5.6 Modélisation cinématique des robots mobiles à roues

5.6.1 Introduction : Les robots mobiles à roues sont de différents types, et suivant la disposition des roues sur la plateforme on a le robot type unicycle, type tricycle, type voiture et type omnidirectionnel.

5.6.2 Repérage d'un robot mobile : On note $R(O,\vec{x},\vec{y},\vec{z})$ un repère fixe dont l'axe $\vec{z}$ est vertical et $R'(O',\vec{x}',\vec{y}',\vec{z}')$ un repère mobile lié au robot. On choisi généralement O' le centre de l'axe des roues motrices. La situation d'un robot mobile peut être représentée par le vecteur (x,y,θ) où x et y sont respectivement l'abscisse et l'ordonnée du point O' dans R et θ l'angle $(ox,o'x')$. La figure 5.14 illustre ce repérage. Le robot mobile est repéré par ces coordonnées: $q=(x,\mathrm{y},\theta)$.

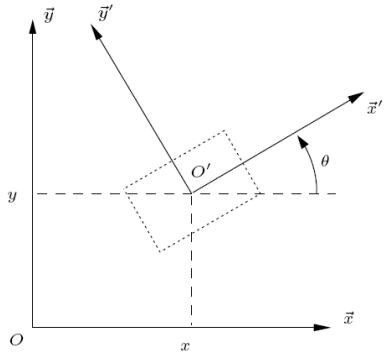


Figure 5.14 Repérage du robot mobile

5.6.3 Modélisation du roulement sans glissement

La locomotion et la modélisation du robot mobile à l'aide de roues exploitent la friction au contact entre roue et sol. Alors la nature du contact, du point de vue régularité du sol et nature du matériau, influence les hypothèses de modélisation et dans le meilleur cas on à un roulement sans glissement qui assure une vitesse nulle au point de contact. Les conditions qui assurent un roulement sans glissement sont : le contacte roue sol est ponctuel et les roues sont indéformables.

Pour une roue du robot mobile, et d'après la figure 5.15, on peut obtenir le modèle cinématique en considérant le roulement sans glissement avec les représentations suivantes : Soit P le centre de la roue, Q le point de contact de la roue avec le sol, φ l'angle de rotation propre de la roue et θ l'angle entre le plan de la roue et le plan $(O,\vec{x},\vec{z})$ comme indiqué à la figure ci-dessous.

La nullité de la vitesse relative $\vec{v}_Q$ roue/sol au point de contact permet d'obtenir une relation vectorielle entre la vitesse $\vec{v}_P$ du centre P de la roue et le vecteur vitesse de rotation $\vec{\omega}$ de la roue :

$$\vec{v}_Q=\vec{v}_P+\vec{\omega}\wedge\overrightarrow{PQ}=\vec{0} \qquad (5.1)$$

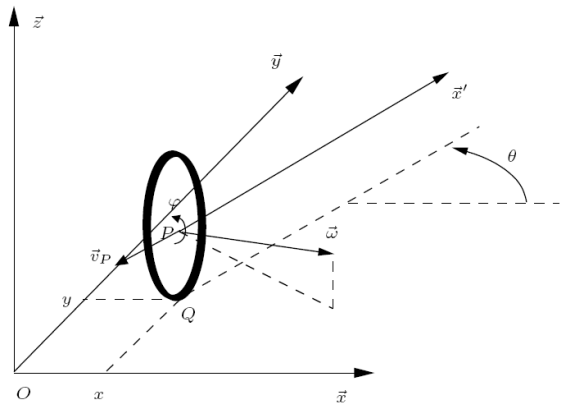


Figure 5.15 Caractérisation du roulement sans glissement

Les points P et Q ont pour coordonnées respectives $(x\ y\ r)^T$ et $(x\ y\ 0)^T$. Il vient alors :

$$\dot{x}\vec{x} + \dot{y}\vec{y} + (\dot{\theta}\vec{z} + \dot{\varphi}(\sin\theta\vec{x} - \cos\theta\vec{y})) \wedge (-r\vec{z}) = \vec{0} \tag{5.2}$$

$$(\dot{x} + r\dot{\varphi}\cos\theta)\vec{x} + (\dot{y} + r\dot{\varphi}\sin\theta)\vec{y} = \vec{0} \tag{5.3}$$

Ceci nous donne le système de contraintes scalaires suivant :

$$\dot{x} + r\dot{\varphi}\cos\theta = 0 \tag{5.4}$$

$$\dot{y} + r\dot{\varphi}\sin\theta = 0 \tag{5.5}$$

Que l'on peut réécrire pour faire apparaître les composantes de vitesse dans le plan de la roue d'une part et perpendiculairement à la roue d'autre part :

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{5.6}$$

$$\dot{x}\cos\theta + \dot{y}\sin\theta = -r\dot{\varphi} \tag{5.7}$$

Ces deux contraintes montrent que le vecteur $\vec{v}_P$ est dans le plan de la roue et ait pour module $r\dot{\varphi}$. Les deux équations précédentes sont non holonomes.

5.6.4 Classes de robots mobiles et leurs modèles cinématiques

Sur les robots mobiles, on rencontre principalement trois types de roues :

- Les roues fixes dont l'axe de rotation, de direction constante, passe par le centre de la roue.
- Les roues centrées orientables, dont l'axe d'orientation passe par le centre de la roue.
- Les roues décentrées orientables, souvent appelées roues folles, pour lesquelles l'axe d'orientation ne passe pas par le centre de la roue.

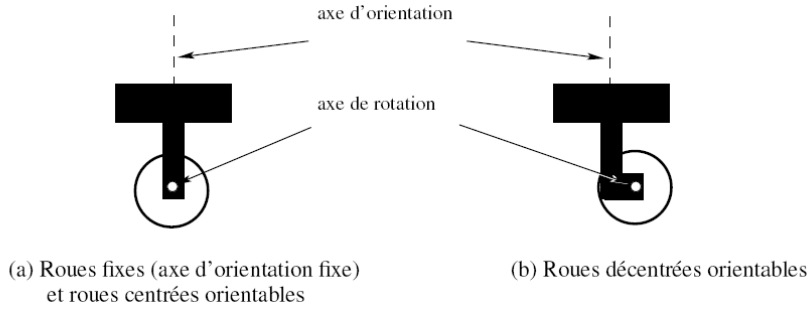


Figure 5.16 Types de roues des robots mobiles

Pour qu'une disposition de roues soit viable et n'entraîne pas de glissement des roues sur le sol, il faut qu'il existe pour toutes ces roues un unique point de vitesse nulle autour duquel tourne le robot de façon instantanée. Ce point lorsque il existe, est appelé centre instantané de rotation (CIR).

5.6.4.1 Modèle du robot type unicycle : On désigne par unicycle un robot actionné par deux roues indépendantes et possédant éventuellement un certain nombre de roues folles assurant sa stabilité. Voir donc la figure 5.17.

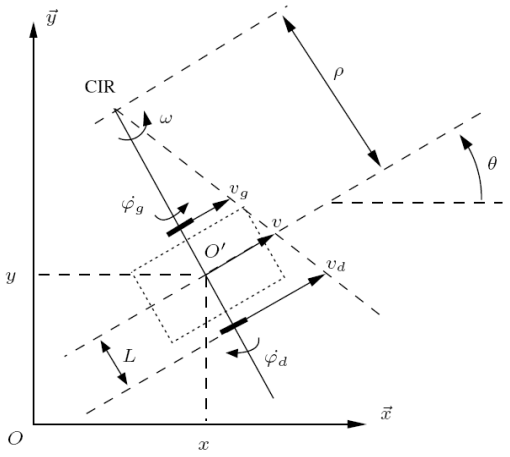


Figure 5.17 Paramétrage cinématique d'un robot unicycle

Les roues motrices ayant même axe de rotation, le CIR du robot est un point de cet axe. Soit ρ le rayon de courbure de la trajectoire du robot, c'est-à-dire le distance du CIR au point O' origine du repère et soit L l'entre-axe des roues motrices et ω la vitesse de rotation du robot autour du CIR, Alors on peut écrire pour les vitesses des roues gauche et droite les relations suivantes :

$$v_d = -r\dot{\varphi}_d = (\rho + L)\omega \tag{5.8}$$

$$v_g = r\dot{\varphi}_g = (\rho - L)\omega \tag{5.9}$$

On peut conclure aussi ρ et ω en fonction des vitesses des roues :

$$\rho = L\frac{\dot{\varphi}_d - \dot{\varphi}_g}{\dot{\varphi}_d + \dot{\varphi}_g} \tag{5.10}$$

$$\omega = -\frac{r(\dot{\varphi}_d + \dot{\varphi}_g)}{2L} \tag{5.11}$$

En ce qui concerne la commande cinématique on préfère généralement le couplet de commande $u = (v, \omega)$ en relation avec $\dot{\varphi}_g$ et $\dot{\varphi}_d$ obtenu comme suit :

$$v = \frac{v_d + v_g}{2} = \frac{r(\dot{\varphi}_g - \dot{\varphi}_d)}{2} \tag{5.12}$$

$$\omega = \dot{\theta} = -\frac{r(\dot{\varphi}_d + \dot{\varphi}_g)}{2L} \tag{5.13}$$

5.6.4.1.a Modèle de commande cinématique

Une considération géométrique donne la relation entre le vecteur de commande et le vecteur des vitesses du robot mobile:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \omega \end{aligned} \quad \text{Soit sous forme matricielle} \quad \dot{q} = C(q)u \;;\; \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{5.14}$$

5.6.4.1.b Modèle des contraintes cinématiques

L'application de la condition du glissement sans roulement aux deux roues du robot mobile comme représenté en Figure 5.17 ci-dessus donne les relations suivantes :

$\dot{x} + L\dot{\theta}\cos\theta + r\dot{\varphi}_d\cos\theta = 0 \quad \dot{y} + L\dot{\theta}\sin\theta + r\dot{\varphi}_d\sin\theta = 0$ roue droite

$\dot{x} - L\dot{\theta}\cos\theta - r\dot{\varphi}_g\cos\theta = 0 \quad \dot{y} - L\dot{\theta}\sin\theta - r\dot{\varphi}_g\sin\theta = 0$ roue gauche

De simples transformations donnent la représentation suivante :

$$\begin{aligned} &\dot{x}\sin\theta - \dot{y}\cos\theta = 0 \\ &\dot{x}\cos\theta + \dot{y}\sin\theta + L\dot{\theta} + r\dot{\varphi}_d = 0 \\ &\dot{x}\cos\theta + \dot{y}\sin\theta - L\dot{\theta} - r\dot{\varphi}_g = 0 \end{aligned} \qquad (5.15)$$

Si on considère le vecteur de configuration suivant $[x\ y\ \theta\ \varphi_d\ \varphi_g]$ on peut écrire l'ensemble des contraintes sous la forme (pfaffienne) $A(q)^T \dot{q} = 0$:

$$\text{Avec : } A(q) = \begin{bmatrix} \sin\theta & -\cos\theta & 0 & 0 & 0 \\ \cos\theta & \sin\theta & L & r & 0 \\ \cos\theta & \sin\theta & -L & 0 & -r \end{bmatrix} \qquad (5.16)$$

5.6.4.2 Modèle du robot type tricycle

Le robot est constitué de deux roues fixes de même axe et d'une roue centrée orientable placée sur l'axe longitudinal du robot, comme en figure 5.18. Le mouvement est conféré au robot par deux actions : la vitesse longitudinale et l'orientation de la roue orientable.

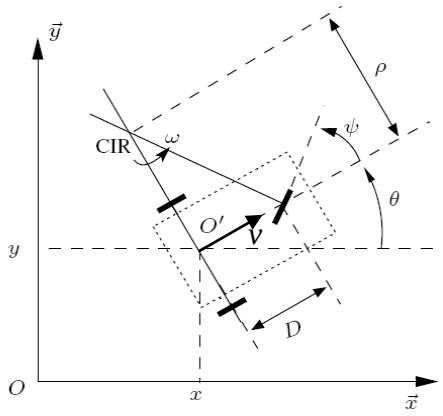


Figure 5.18 Robot mobile tricycle et son CIR

Le CIR du robot se situe à la rencontre des axes des roues fixes et de la roue orientable. On détermine ρ de manière géométrique à partir de l'angle d'orientation de la roue avant et on détermine ω à partir de v et de ρ :

$$\rho = \frac{D}{\tan\psi} \qquad \omega = \frac{v}{D}\tan\psi \qquad (5.17)$$

5.6.4.2.a Modèle de commande cinématique

Un raisonnement géométrique permet d'établir le modèle de commande, sachant que le vecteur de commande choisie est $u = (v, \eta)^T$:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{D}\tan\psi \\ \dot{\psi} &= \eta \end{aligned} \tag{5.18}$$

5.6.4.3 Modèle du robot omnidirectionnel

Un robot est dit omnidirectionnel s'il peut se déplacer dans toutes les directions et il peut être réalisé par trois roues décentrées orientables ou de trois roues suédoises disposées aux sommets d'un triangle équilatéral ; voir figure 5.19.

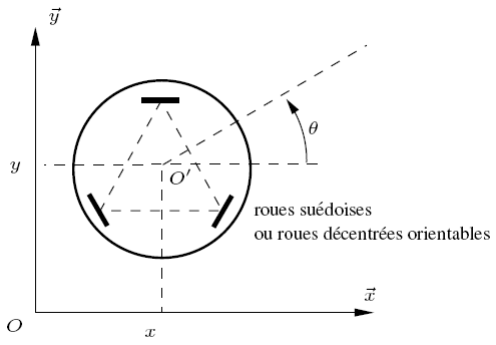


Figure 5.19 Représentation d'un robot mobile omnidirectionnel

Pour le robot omnidirectionnel on peut commander directement et indépendamment les trois variables du modèle géométrique :

$$\begin{aligned} \dot{x} &= u_1 \\ \dot{y} &= u_2 \\ \dot{\theta} &= u_3 \end{aligned} \tag{5.19}$$

Le robot omnidirectionnel ne possède pas de contraintes de non holonomie.

5.7 Conclusion : Le robot mobile est apparu suite à la nécessité d'augmenter le champ atteignable du robot manipulateur. Suivant les besoins, des robots de différents moyens de locomotion ont été fabriqués. On trouve des robots à roues, à chenilles, à pattes et volants. Plusieurs architectures de planification de trajectoires sont étudiées, certaines sont globales (AuRA, ..) et d'autres locales de nature réactives (champs de potentiel,…). La modélisation des robots mobile permet de réaliser de commandes basées modèle.

BIBLIOGRAPHIE

[1] R. A. Brooks, ‘A robust layered control system for a mobile robot’, IEEE journal of robotics and automation 1986.

[2] Arkin R. C. , ’Motor schema based navigation for a mobile robot’, In Proc. Of the IEEE INT. Conf on Robotics and automation, Volume 1, page 264 271, San Fransisco, CA(US), 1987.

[3] Arkin R. C.,’ Dynamic replanning for mobile robot based on internal sensing’, In Proc. Of the IEEE int. Conf. On Robotics and Automation, volume3, page1416-1421, Scottsdale, AZ(US),1989.

[4] R. P. Bonasso, D. Kortenkamp, D. P. Miller, and M. Slack, ‘Experiences with an architecture for intelligent, reactive agents’, Lecture Notes in Computer Science, 1037:187 202,1996.

[5] R. Alami, R. chatila, S. fleury, M. Ghallab, and F. Ingrand,’ An architecture for autonomy’, Int. Journal of Robotics Research,17(4):315 337, 1998.

[6] Patrick Reignier, ‘pilotage réactif d’un robot mobile : Etude de lien entre la perception et la reaction’, thèse de Doctorat, Institut National polytechnique de Grenoble, 1994.

[7] D.Fox W.Burgard and S.thrun ‘the Dynamic Window Approach to collision Avoidance’. IEEE Robotics,vol,18,no 10 pages 1001-1024, 2004.

[8] S.Quinlan, O.Khatib ‘Elastic Bands : connecting path planning and control’, international conference on Robotics and Automation, Atlanta, GA,USA,mai,1993.IEEE

[9] D .W.Payton, ‘An architecture for reflexive autonomous vehicle control’, In Proc of the IEEE Int. Conf. on Robotics and Automation, volume 3, pages 1838-1845, San Francisco, CA(US), 1986.

[10] R.G.Simmons, ‘Structured control for autonomous robots. IEEE Trans. Robotics and Automation, 10(1):34-43, 1994.

CHAPITRE 6

Planification de trajectoire pour robot mobile avec évitement d'obstacles en utilisant la méthode des contraintes

6. Planification de trajectoire pour robot mobile avec évitement d'obstacles en utilisant la méthode des contraintes

6.1 Introduction : Lorsque le robot mobile se déplace dans un environnement pas trop encombré il est possible de considérer l'opération d'évitement des obstacles d'une façon indépendante de la tâche globale. Alors sa revient à utiliser une méthode locale qui interagisse à son environnement suivant les informations que produit les différents capteurs. La méthode à étudie est appelée Méthode des contraintes.

6.2 Méthode des contraintes : La méthode des contraintes a été proposée par B.Faverjon et P.Tournasound [5] et appliquée aux bras manipulateurs pour éviter une collision possible avec des obstacles. Cette méthode vient comme une alternative à la méthode des champs de potentiels proposée par O.khalil.

La méthode se base sur des contraintes géométriques dans le plan des vitesses de commande. Ces contraintes en vitesses apparaissent lorsqu'un obstacle se trouve proche du robot mobile. Les contraintes limitent l'ensemble des vitesses de commande permises de façon à éviter au robot mobile la possibilité d'entrer en collision avec l'obstacle.

Pour affirmer cette possibilité, il revient à dire et d'après la figure ci-dessous, que la vitesse du point P (position du capteur qui a détecté l'obstacle) doit être nulle lorsque il sera proche à une distance d_s (distance de sécurité) de l'obstacle. La restriction doit être considérée une fois le robot mobile se trouve à une distance d_i (distance d'influence) quelque soit sa vitesse à cette distance.

D'après le schéma ci-dessous, plus la vitesse $\dot{d}$ du point P est positive plus il s'éloigne de l'obstacle et plus la vitesse $\dot{d}$ est négative plus se rapproche de l'obstacle.

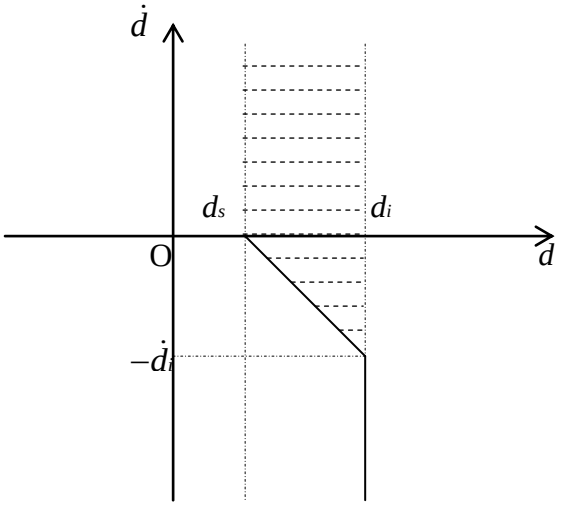


Figure 6.1 Illustration de la condition de la non collision par la relation entre $\dot{d}$ et d

D'après la figure 6.1 on a les conditions suivantes : à la position $d = d_s$ il faut que la vitesse soit positive ou nulle $(\dot{d}(d_s) \geq 0)$ c'est-à-dire s'éloigne de l'obstacle, et à la position d_i la vitesse peut avoir une valeur maximale négative $(-\dot{d}_i)$ c'est-à-dire se dirige vers l'obstacle.
L'ensemble des points dans la partie hachurée sont autorisés et sont limités inférieurement entre d_s et d_i par une droite qui peut avoir pour équation :

$$\dot{d} = \begin{cases} 0 & \text{si } d = d_s \\ -\dot{d}_i & \text{si } d = d_i \end{cases} \quad \text{et} \quad \begin{cases} \dot{d} = ld + m \quad si\ d_s < d < d_i \\ l = \dfrac{-\dot{d}_i}{d_i - d_s} \\ m = -\dfrac{-\dot{d}_i d_s}{d_i - d_s} \end{cases} \quad \Rightarrow \dot{d} = -\dot{d}_i \frac{d - d_s}{d_i - d_s} \tag{6.1}$$

Pour éviter la collision il faut que : $\dot{d} \geq -\dot{d}_i \dfrac{d - d_s}{d_i - d_s}$ (6.2)

L'intégration de l'équation (6.2.2) donne :

$$d(t) \geq d_s + (d_0 - d_s) EXP(\frac{-\lambda(t - t_0)}{d_i - d_s}) \ ; \ \lambda = -\dot{d}_i \ ; \ d = d_0 \text{ à } t = t_0 \ (d_0 > d_s) \tag{6.3}$$

Il est bien évident qu'il n'aura pas de collision entre le robot mobile et l'obstacle du fait que $d(t) \geq d_s \quad \forall t > t_0$.

6.3 Modèle cinématique de commande du robot différentiel (unicycle)

Comme indiqué en figure 5.17 le modèle de commande qui va être utilisé pour la simulation est le suivant :

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \qquad (6.4)$$

6.4 Modélisation des capteurs

Le robot mobile est équipé de huit capteurs à ultrason installés comme indiqués en figure 6.2.

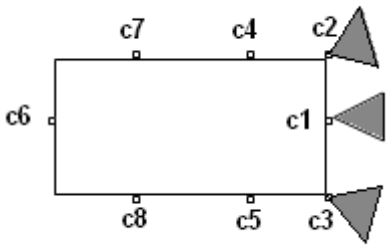


Figure 6.2 Position des capteurs sur le robot mobile

Le capteur est défini comme montré en figure 6.3 par l'ensemble des paramètres suivant:

P : Représente les coordonnées de la position du capteur sur le repère lie au robot.

α : C'est l'angle d'émission du capteur par rapport à l'axe O'x'.

γ : C'est l'angle d'ouverture du capteur.

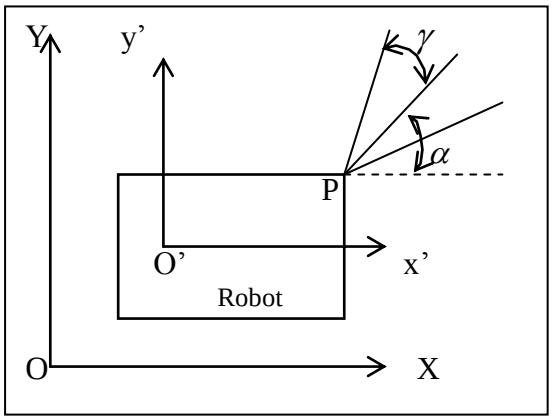


Figure 6.3 Paramètres du capteur

En simulation chaque capteur est modélisé d'une manière géométrique par deux contraintes linéaires du type $(y \le a_i x + b_i)$ représentant l'ensemble des points entre les deux droites c'est à dire appartenant au cône d'émission.

Donc :

Pour i=1,2 ; $(y \le a_i x + b_i)$ ou $(-a_i x + y \le b_i)$; sous forme matriciel : $A_{sens} X \le B_{sens}$ (6.5)

Avec : $X = [x\ y]^T$ $A_{sens} = \begin{bmatrix} -a_{1_{sens}} & 1 \\ -a_{2_{sens}} & 1 \end{bmatrix}$, $B_{sens} = \begin{bmatrix} b_{1_{sens}} \\ b_{2_{sens}} \end{bmatrix}$ (6.6)

6.5 Modélisation des obstacles

Chaque obstacle est considéré ayant une forme géométrique polygonale convexe. L'obstacle est donc l'ensemble des points appartenant au polygone modélisé par un ensemble de contraintes linéaires de type $(y \le a_i x + b_i)$, cela comme présenté en figure 6.4.

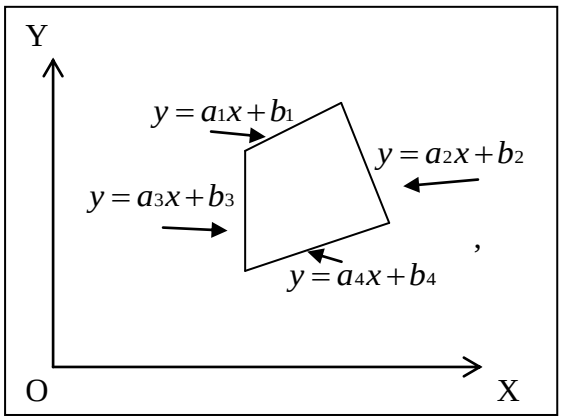


Figure 6.4 Modélisation de l'obstacle par des contraintes géométriques $(y < a_i x + b_i)$

Alors un obstacle dans notre cas est représenté par quatre contraintes et à chaque obstacle de quatre arrêtes on a :

Pour i=1,4 : $(y \le a_i x + b_i)$ ou $(-a_i x + y \le b_i)$ où se forme matricielle : $A_{ob} X \le B_{ob}$ (6.7)

Avec : $X = [x\ y]^T$ $A_{ob} = \begin{bmatrix} -a_{1_{ob}} & 1 \\ -a_{2_{ob}} & 1 \\ -a_{3_{ob}} & 1 \\ -a_{4_{ob}} & 1 \end{bmatrix}$, $B_{ob} = \begin{bmatrix} b_{1_{ob}} \\ b_{2_{ob}} \\ b_{3_{ob}} \\ b_{4_{ob}} \end{bmatrix}$ (6.8)

6.6 Calcule de distance minimale

Le capteur détecte la présence d'un obstacle revient à dire qu'il existe au moins un point appartenant à son cône d'émission de même appartient au polygone représentant l'obstacle.

Le point le plus proche du capteur dire du robot se trouve logiquement sur le contour de l'obstacle.

La distance entre le point d'émission du capteur P(x_p,y_p) et un quelconque point Obs(x, y) de l'obstacle est :

$$d = [(x - x_p)^2 + (y - y_p)^2]^{1/2} \tag{6.9}$$

Le point le plus proche Obs($x_{\min}, y_{\min}$) qui donne la distance minimale $d_{\min}$ est obtenue par la minimisation de *d* sous les contraintes de l'obstacle et du capteur :

$$AX \leq B \tag{6.10}$$

Avec : $A = \begin{bmatrix} A_{ob} \\ A_{sens} \end{bmatrix}$, $B = \begin{bmatrix} B_{ob} \\ B_{sens} \end{bmatrix}$, $X = [x\ y]^T$

Par suite : $d_{\min} = [(x_{\min} - x_p)^2 + (y_{\min} - y_p)^2]^{1/2}$ (6.11)

Donc le calcule de distance minimale devient un problème d'optimisation d'un critère quadratique avec les contraintes géométriques de l'obstacle et du capteur.

L'optimisation du critère quadratique déduit de *d* (formule (6.9)) est :

$$(1/2 X^T G X + F^T X) \tag{6.12}$$

Avec $X = [x\ y]^T$, $G = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, $F = [-2x_p\ -2y_p]$

L'instruction '*quadprog*' de Matlab permet la résolution du problème de minimisation sous contraintes du type $AX \leq B$. Ce qui permet de trouver le point minimal ' x_{dmin} ,y_{dmin}' .

6.7 Loi de contrôle pour parcours libre

Si le parcourt devant le robot mobile est considéré libre, il doit donc optimiser son chemin pour aller en ligne droite vers la cible, et d'une façon stable.

Soit $q_f = [\ x_f\ , y_f\ , 0]^T$ la position de la cible à atteindre et $q = [\ x\ , y\ , \theta]^T$ la position courante du robot mobile.

Soit $x_e = x_f - x$ et $y_e = y_f - y$; et parce que les variables de contrôle sont au nombre de deux (v, ω), on doit définir pour le robot les coordonnées polaires a et $\alpha 1$.

$$\text{Avec : } a = [x_e^2 + y_e^2]^{1/2},\ \alpha 1 = \text{atan2}(y_e, x_e) - \theta,\ -\pi < \alpha 1 < \pi \qquad (6.13)$$

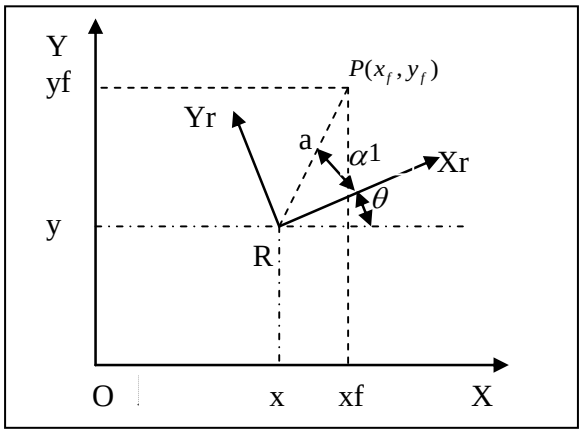


Figure 6.5 Représentation cinématique
en coordonnées polaires(a, $\alpha 1$)

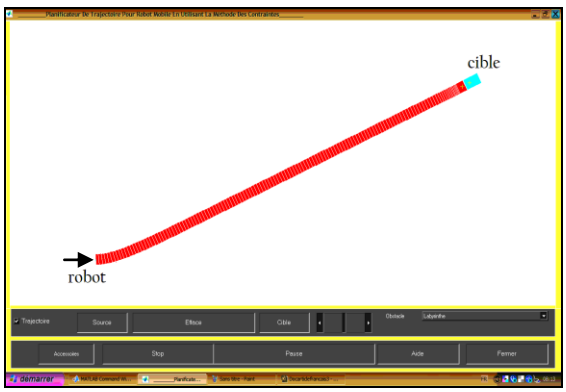
Les variables de contrôle (v, ω) doivent assurer la réorientation du robot vers la cible par ω et son rapprochement de la cible par v.

La loi de contrôle suivante [7],[11] est proposée :

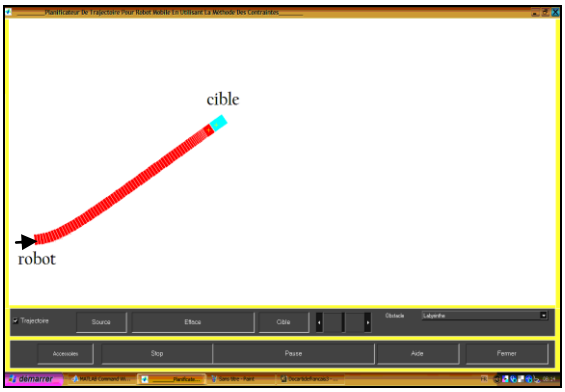
$$v = k_1.a.\cos(\alpha 1)\ ;\quad \omega = k_2.\alpha 1 + k_1.\sin(\alpha 1)\cos(\alpha 1) \qquad (6.14)$$

Avec k_1, k_2 sont des constantes positives .

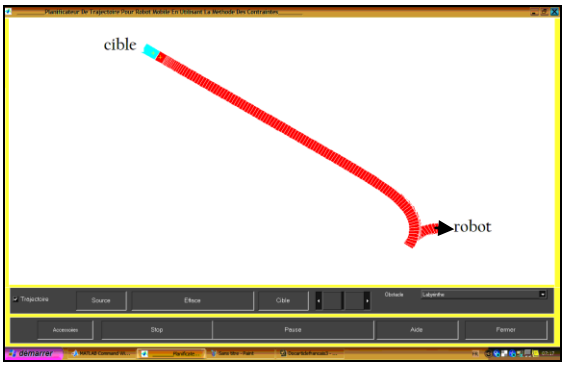
Cette loi de contrôle est appliquée tant qu'on ne détecte pas d'obstacle, et mène le robot en ligne droite vers la cible. Les résultats de la poursuite sont montrés en figure 6.6.



(a)



(b)



(c)

Figure 6.6.a,b,c Robot mobile le long de parcourt libre d'obstacle

6.8 Méthode des contraintes

La méthode comme a été cité plus haut est une alternative à la méthode de champs de potentiel pour la navigation sans collision de robots manipulateurs dans des environnements encombrés d'obstacles [5], [9], [4].

La méthode est adaptée aux robots mobiles de sorte que les obstacles exercent, dans le plan des vitesses, des contraintes cinématiques sur le robot. Ces contraintes cinématiques obligent le robot à rester éloigné des obstacles pour éviter leur collision. Les contraintes entrent en action lorsque le robot s'approche suffisamment des obstacles.

Et comme mentionner en figure 6.1 on a :

d : Distance minimale entre obstacle et robot mobile.

di : Distance d'influence à partir de laquelle la contrainte devient active.

ds : Distance de sécurité représentant la distance minimale permise entre le robot et l'obstacle.

λ : Coefficient d'ajustement de la vitesse de convergence de d vers ds.

Lorsque le robot détecte par ses capteurs un ensemble d'obstacles, alors en associe à chaque obstacle une contrainte cinématique d'anticollision:

Contrainte k:

$$\dot{d}_k \geq -\lambda_k (d_k - ds_k) / (di_k - ds_k) \tag{6.15}$$

En plus on à des contraintes concernant les limitations physiques sur les vitesses :

$$(|v| < v_{\max}, |\omega| < \omega_{\max}) \tag{6.16}$$

Cet ensemble de contraintes construit un polygone convexe des vitesses permises (PVP) assurant la non collision. Le (PVP) est évalué à chaque cycle de contrôle et est présenté en figure 6.7.

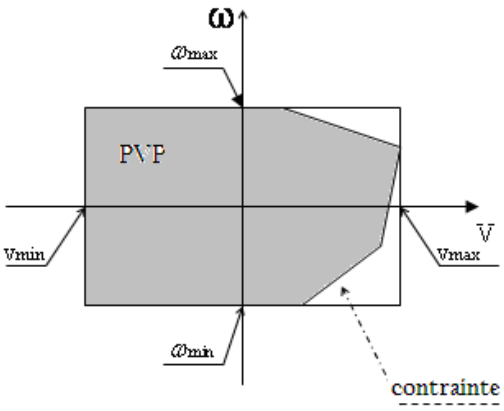


Figure 6.7 Polygone des vitesses permises (PVP)

6.9 Représentation des contraintes dans le plan des vitesses

La contrainte porte sur les vitesses de contrôle linéaire v et de rotation ω du robot mobile.

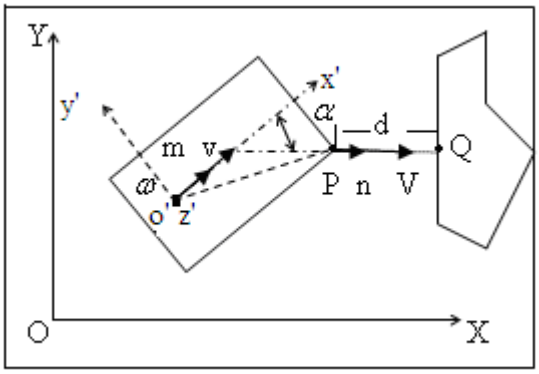


Figure 6.8 Représentation des contraintes dans le plan v, ω

D'après le schéma indiqué sur la figure 6.8 on peut écrire :

$$\dot{d} \geq -\lambda(d-ds)/(di-ds)$$

Ou

$$V_p.n < +\lambda(d-ds)/(di-ds) \qquad (6.17)$$

V_p : Vecteur vitesse du point P.

m : Vecteur unitaire le long de $o'x'$

n : Vecteur unitaire le long de PQ.

α : Angle que fait PQ avec $o'x'$.

z : Vecteur unitaire le long de l'axe o'z'.

Avec

$$V_p = v.m + \omega.z \wedge \overline{O'P} \qquad (6.18)$$

Soit :

$$(m.n).v + (\overline{O'P} \wedge n).z)\omega < \lambda(d - ds)/(di - ds) \tag{6.19}$$

Les coordonnées des vecteurs n,m, $O'P$ par rapport au repère (o', x',y',z') lie au robot sont :

$n = [\cos(\alpha)\ \sin(\alpha)\ 0]^T$ $\quad m = [1\ 0\ 0]^T$ $\quad O'P = [x_p\ y_p\ 0]^T$

Alors (6.19) s'écrit:

$$(\cos(\alpha).v) + (x_p \sin(\alpha) - y_p \cos(\alpha))\omega < \lambda(d - ds)/(di - ds) \tag{6.20}$$

Soient: $A = \cos(\alpha)$; $B = x_p \sin(\alpha) - y_p \cos(\alpha)$; $C = \lambda(d - ds)/(di - ds)$

Finalement la contrainte (6.20) peut être écrite comme suit:

$$Av + B\omega < C \quad \text{ou aussi :} \quad [A\ B]\begin{bmatrix} v \\ \omega \end{bmatrix} < C$$

Si on considère *n* contraintes actives pour *n* obstacles on aura l'équation suivante:

$$A_{ppv} V < C_{ppv} \tag{6.21}$$

Avec: $V = \begin{bmatrix} v \\ \omega \end{bmatrix}$; $A_{ppv} = \begin{bmatrix} A_1 & B_1 \\ \vdots & \vdots \\ A_n & B_n \end{bmatrix}$ et $C_{ppv} = \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix}$

n : nombre des contraintes actives.

6.10 Evitement d'obstacles

Dans le cas où le robot mobile détecte par ses capteurs la présence d'obstacles, l'algorithme de contrôle calcule le couple de valeurs (v*, ω*) qui permet d'éviter ces obstacles comme montré en figure 6.9.

Ce couple de commande $u^* = (v^*, \omega^*)$ est le point le plus proche de la commande $u = (v, \omega)$ (cas de parcourt libre, calculé par la formule (6.14)) et appartenant aussi au polygone des vitesses permises PVP définie à ce moment par les contraintes en vitesse imposées par les obstacles (formule 6.21). Le point le plus proche (v^*, ω^*) donnant la distance minimale est calculé en utilisant 'les procédures de Matlab' associées à la contrainte générale $A_{ppv} V < C_{ppv}$.

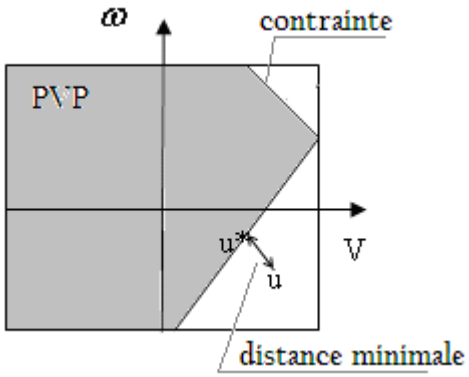
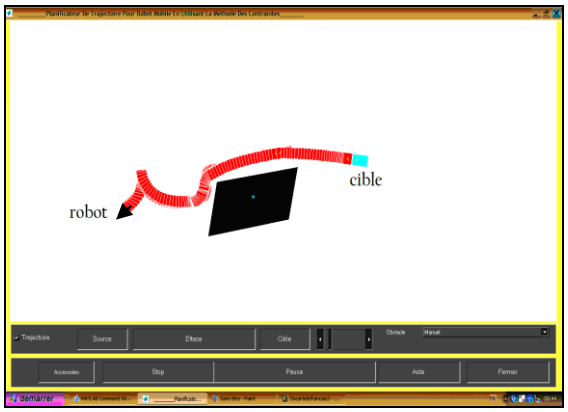
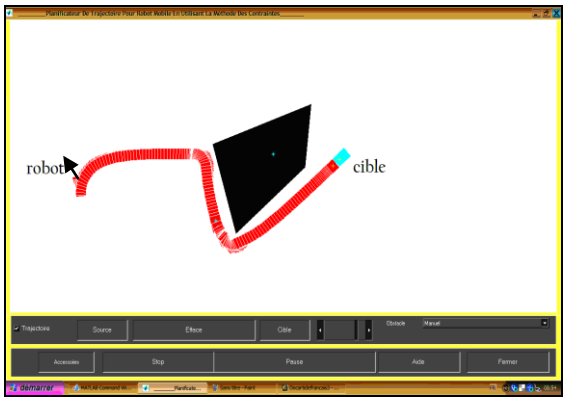


Figure 6.9 Calcul de u* =dis-min(u,PVP)



(a)



(b)

Figure 6.10.a,b Evitement d'un obstacle

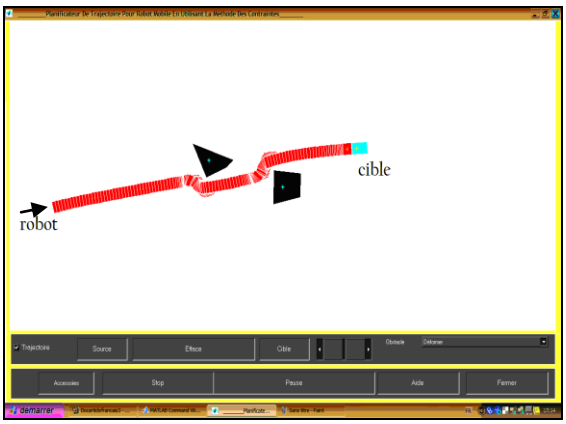


Figure 6.11 Evitement de deux obstacles

On montre le long du parcours indiqué sur la figure 6.10 et la figure 6.11 que l'évitement est réalisé par le calcul, à chaque itération de commande, de u* permettant de s'éloigner des obstacles et se dirigeons vers la cible en ligne droite.

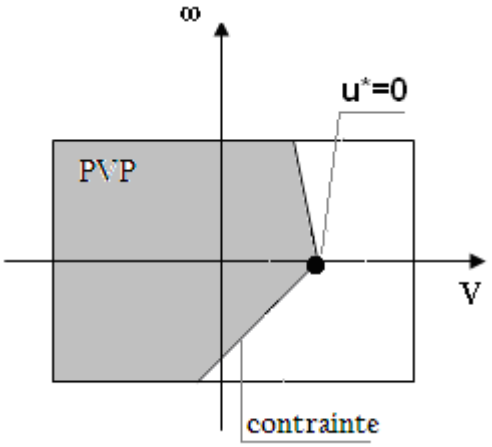


Figure 6.12 Cas où u*=0

L'application de la commande u* n'est pas valable si elle est nulle cela provoque le blocage du robot à sa position.

6.11 Situation de blocage et règles heuristiques pour le contournement de l'obstacle

Le fait qu'il arrive que le vecteur de contrôle $u*$ s'annule dans certains situations comme illustré en figure 6.13, sans que cela signifie que le robot est en position finale ; ce la veut dire que le robot est en situation de blocage, ce qui exige de résoudre ce problème par un autre algorithme plus approprié que celui appliqué précédemment.

Cet algorithme adopté, se base sur le polygone des vitesses permises (PVP) et sur des règles heuristiques. L'algorithme doit assurer le contournement de l'obstacle, à droite ou à gauche, et décider de la situation de fin de contournement. Il doit aussi identifier l'obstacle à contourner par le biais de sa contrainte (passe par l'origine) parmi d'autres qui constituent le PVP. A chaque itération de calcul, la contrainte doit être retrouvée pour continuer le contournement de se même obstacle.

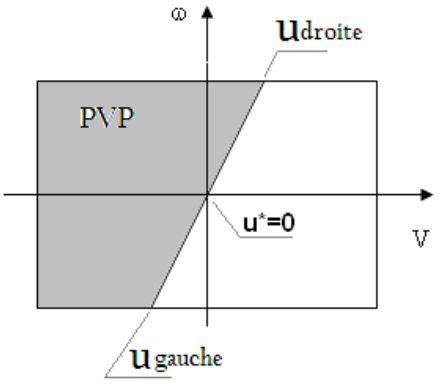


Figure 6.13 Blocage du robot (à une contrainte) à cause de u*=0

Le vecteur u recherché à appliquer pour le contournement est l'un des sommets u_{droite} ou u_{gauche} de la contrainte à suivre ; cela permet de suivre le contour de l'obstacle par la gauche ou par la droite [14].

Si l'obstacle bloquant se trouve à gauche le contournement se fait à droite et s'il se trouve à droite le contournement se fait à gauche. Le sens de contournement est gardé jusqu'à la décision de fin de contournement, figure 6.14.

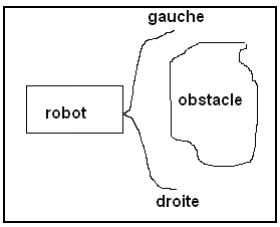
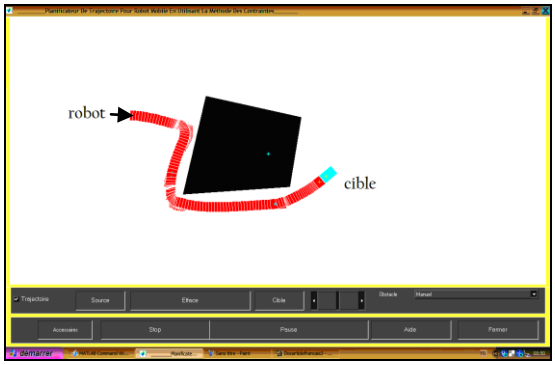
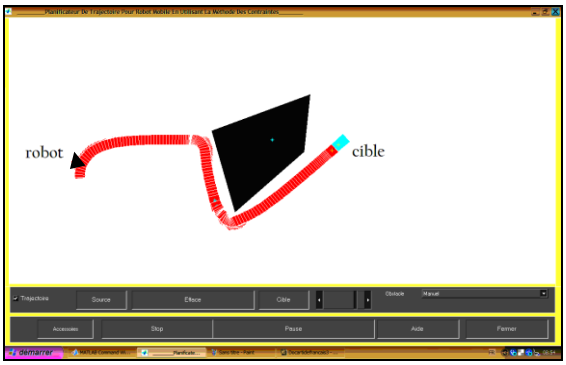


Figure 6.14 Illustration du contournement par la gauche et par la droite

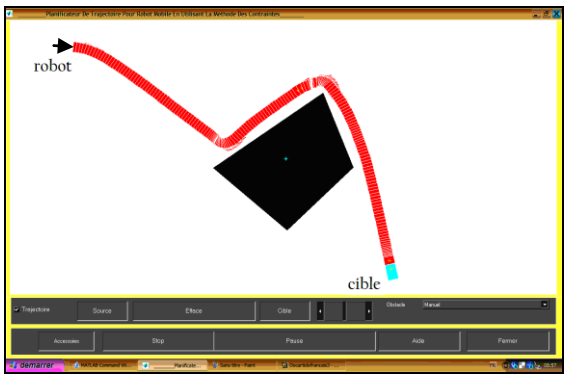


(a)

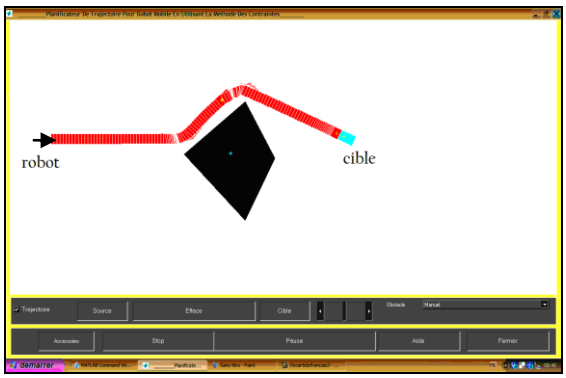


(b)

Figure 6.15.a,b Contournement d'un obstacle par la droite



(a)



(b)

Figure 6.16.a,b Contournement d'un obstacle par la gauche

Si au cours du contournement d'un obstacle par la gauche ou par la droite on se mène à avoir une commande $(u^* = 0 : v^* = 0, \omega^* = 0)$ cela indique la présence d'un autre obstacle, donc on bascule vers sa contrainte en gardant toujours le même sens de contournement à droite ou à gauche pour avoir un suivi de contour.

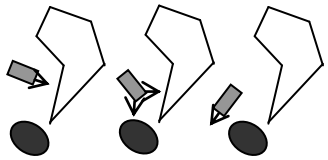
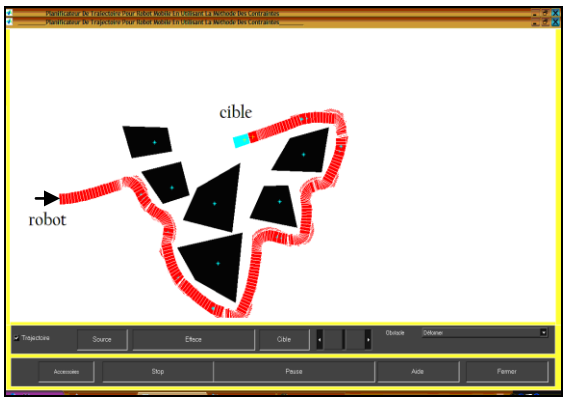
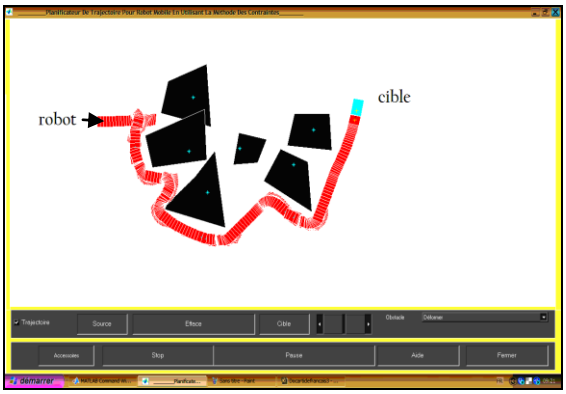


Figure 6.17 Illustration du changement d'obstacle à contourner

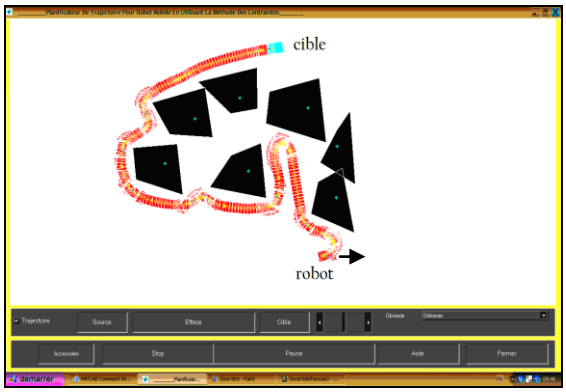


(a)

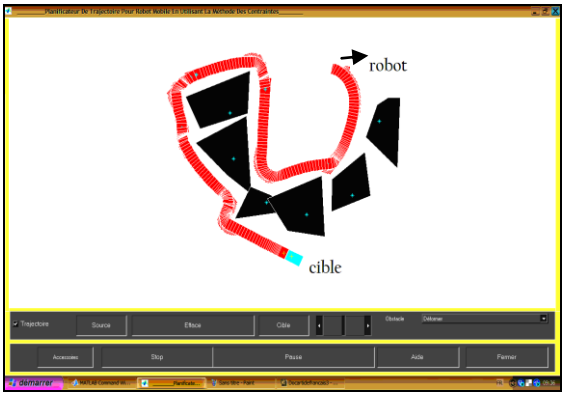


(b)

Figure 6.18.a,b Contournement de plusieurs obstacles par la droite



(a)



(b)

Figure 6.19.a,b Contournement de plusieurs obstacles par la gauche

6.12 Décision de fin de contournement

La fin de contournement d'un obstacle est détectée lorsque le robot mobile est dirigé vers la cible (à ± 0.3rd) et pas d'obstacles proches ni en face (détecter par capteur $n^0 1$) ni au coté gauche (détecter par capteur $n^0 2$) ni au coté droite (détecter par capteur $n^0 3$).

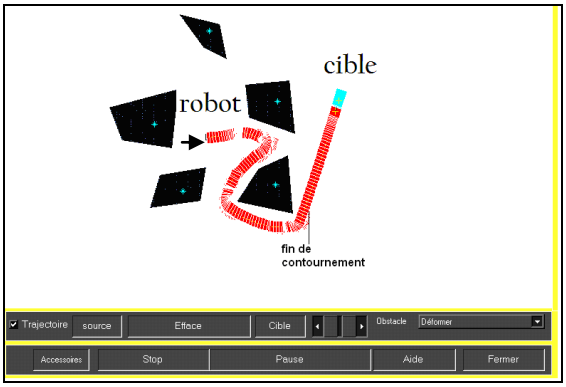


Figure 6.20 Décision de fin de contournement

6.13 Le robot en labyrinthe

Le type des obstacles comme vu précédemment sont des polygones convexes de quatre arrêtes générés d'une façon volontaire ou aléatoire. Un labyrinthe est une succession d'obstacles polygonaux convexes juxtaposés.

Le labyrinthe donne un bon exemple pour confirmer l'efficacité du planificateur.

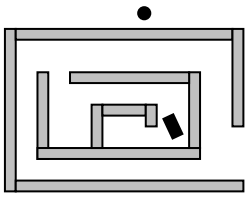


Figure 6.21 Illustration de la navigation
du robot mobile en labyrinthe

La figure(6.22) et La figure(6.23) montrent les résultats de simulation concernant la recherche d'une cible via un labyrinthe.

Figure 6.22 Le robot à l'extérieur joint la cible à l intérieur du labyrinthe

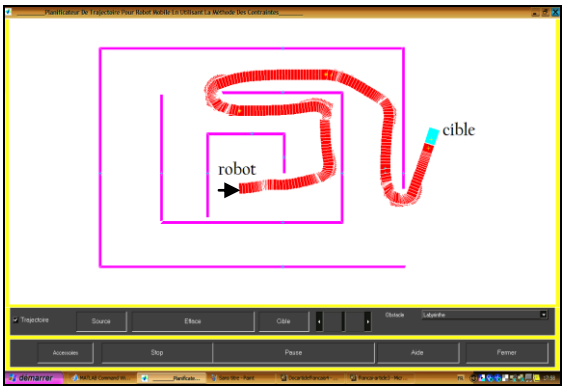


Figure 6.23 Le robot à l'intérieur joint la cible à l'extérieur du labyrinthe

6.14 Situation de perte d'obstacle par les capteurs

Un obstacle, quoique qu'il existe, peut ne pas être perçu par le robot s'il n'est pas dans le champ de vision d'au moins un capteur. La solution consiste et suivant le capteur actif de réaliser une manœuvre approprie (v=v+, ω= ω+) pour retrouver l'obstacle. Si l'obstacle n'est pas perçu par se même capteur voire donc avec les capteurs juxtaposés, cette opération est répétée jusqu'à ce que un capteur détecte l'obstacle. La figure 6.24 présente ce cas.

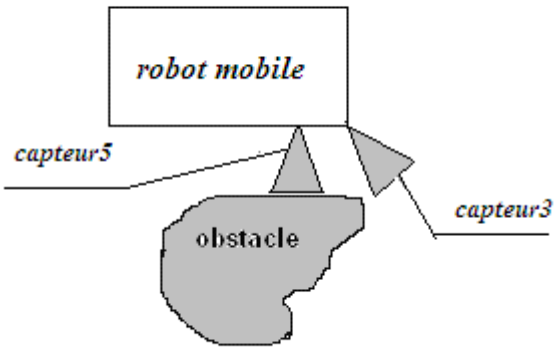


Figure 6.24 Obstacle perdu par capteur n^{0}3 retrouver par capteur n^{0}5

6.15 Situation de pénétration dans un tunnel

Si le robot mobile se retrouve au fond d'un tunnel de façon que la possibilité de sortir devienne non évidente par les règles précédentes, les règles 'sortir d'un tunnel' sont activées.

L'identification d'un blocage dans un tunnel est réalisée lorsque les capteurs c1, c2, c3, c4 et c5 détectent des obstacles très proches, ce qui ne permet ni d'avancer ni de manœuvrer pour faire demi tours.

La solution retenue, est de faire marche arrière jusqu'à ce que les capteurs précédents soient loin des obstacles, ensuite faire une rotation d'un angle β déterminer expérimentalement pour ne pas retourner au tunnel.

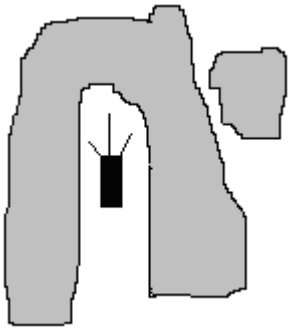


Figure 6.25 Illustration de la présence du robot mobile au fond du tunnel

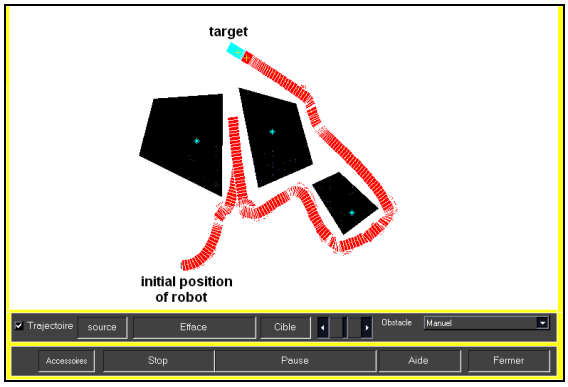


Figure 6.26 Echappement du robot mobile d'un tunnel

6.16 Cas de cible mobile

Le mouvement de la cible est identifié par le changement de ses coordonnées. L'exigence dans ce cas est que la vitesse de la cible soit plus faible que celle du robot mobile. Les figures 6.27 et 6.28 montrent les résultats de poursuite.

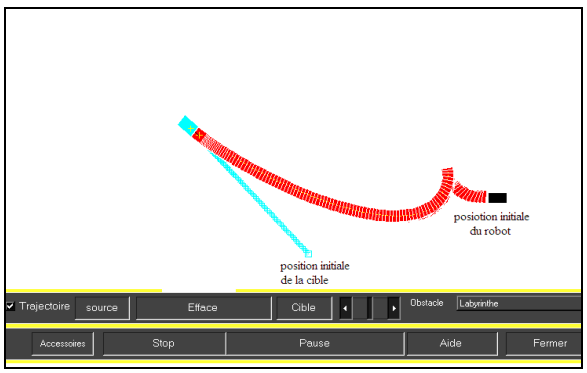


Figure 6.27 Poursuite de cible mobile en environnement libre d'obstacles

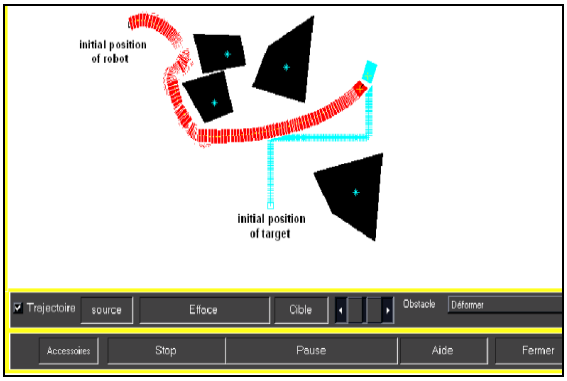


Figure 6.28 Poursuite de cible mobile en environnement encombré d'obstacles

6.17 Conclusion : Dans ce travail le planificateur intervient sous trois formes, cas où il n'y a pas des obstacles on utilise une commande qui optimise le parcours pour aller en ligne droite; dans le cas où il y'a des obstacles on utilise la méthode des contraintes pour construire le PVP puis on calcul la commande comme le point qui appartient au PVP et assure que la distance, au point de commande au cas de parcours libre, soit minimale.

Dans le cas où se passe un blocage du robot derrière un obstacle à cause de la valeur nulle de la commande on utilise le PVP et un ensemble de règles heuristiques qui permettent de contourner l'obstacle en suivi de mur.

Le planificateur décide de la fin du contournement de l'obstacle si une condition relative à l'orientation du robot envers la cible et la non présence d'obstacles proches est vérifiée.

Le robot mobile peut s'échapper d'un tunnel, et peut rattraper une cible mobile si sa vitesse est relativement faible.

L'ensemble des résultats de simulation montrent que se planificateur local permet de franchir des scènes libres d'obstacles, des scènes contenant des obstacles et des labyrinthes.

BIBLIOGRAPHIE

[1] Ching-Long.S and Jane-Yu.L,'Computing the Minimum Directed Distances Between Convex Polyhedra', Journal of information on science and engineering vol15, pp353-373, 1999.

[2] Ellepola.R and Kovesi.P, ' mobile robot navigation using recursive motion control', IEEE ,pp168-174, 1997.

[3] Elmer.G.G, W.Johnson.D and Keerthi.S.S ,'A fast procedure for computing the distance between complexes objects in three-dimensional space', IEEE, journal of robotics and automation ,vol.4, no.02,pp193-203, April,1988.

[4] Faverjon.B, 'Hierchical object models for efficientanti-collision algorithms', IEEE, pp333- 340 , 1989.

[5] Faverjon.B and Tournassoud.P, 1987, 'A local based approache for path planning of manipulators with a high number of degrees of freedom ',IEEE , 1987.

[6] Paulo.C and Urbano.N, April,'Path-following control of mobile robot in presence of uncertainties', IEEE, Transactions on robotics, vol21 .no2,pp252-261,2005.

[7] Ramirez.G and Zeghloul.S, ' A New local path planner for nonholonomic mobile robot

navigation in cluttered environments', IEEE, ICRA, san Francisco pp2058-2063, April 2000.

[8] Shirong.L, Simon.X.Y and Huidi.Z, 'Adaptive neurons Based control system design for mobile robot', proceedings of 2004 IEEE/RSJ, ICIRS, sendal Japan, pp2636-2641, September 28-Octobre02, 2004.

[9] Tournassoud.P and Jehl.O, ' Motion planning for a mobile robot with a kinematics constraint', IEEE, pp1785-1790, 1988.

[10] Patrick G. X, 'Fast Swept-Volume distance for Robust Collision Detection', Proceeding of the 1997 IEEE, ICRA, Albuquerque, New Mexico, April 1997.

[11] Zhihua Qu, Jing Wang and Clinton.E 'A New Analytical Solution to Mobile Robot trajectory Generation in the Presence of Moving Obstacles' IEEE Transaction on Robotics Vol.20, No.6. Decembr 2004.

[12] Yasutaka Umeda and Takahiro Yakoh ,'Configuration and Readhesion Control for Mobile Robot With External Sensors ',IEEE Transaction on Idustrial Electronics, Vol.49 No.1, February 2002.

[13] Ji Yeong Lee and Howie Choset 'Sensor-Based Exploration for Convex Bodies: A New Raodamp for a convex-Shaped Robot ',IEEE Transactions on Robotics Vol.21 .No2 April.2005.

[14] Maaref.H and Barret.C 'Sensor-Based Navigation of a Mobile Robot in an Indoor Environment' Elsevier, Robotics and Autonomous Systems, 1-8, (2002).

CHAPITRE 7

La logique floue

7. La logique floue

7.1 Introduction : La logique floue a pris sa place au détriment de la logique du certain. Elle se base sur les ensembles flous et utilise le langage naturel. La logique floue se caractérise par sa généralisation ; elle se présente comme une alternative par le bai de l'expérience d'un expert lorsque le modèle déterministe est difficile à obtenir. Utilisée dans plusieurs domaines tels que le contrôle et la régulation, le traitement de signal, et la commande de robot.

7.2 Ensembles flous : Soit X un espace de points avec comme élément générateur x, donc $\mathrm{X}=\{x\}$. Un ensemble flou A dans X est un ensemble défini par sa fonction $\mu_A(x)$ qui associe à chaque point dans X un nombre réel appartenant à l'intervalle $[0\ \ 1]$,[2].

$$\mu_A(x):\mathrm{X}\rightarrow[0\ \ 1] \tag{7.1}$$

Ce nombre réel représente le degré d'appartenance de x dans A :

$$\mu_A(x)\begin{cases}=1 & x \text{ est un élément complet de } A\\ \in\ ]0\ \ 1[& x \text{ est un élément partiel de } A\\ =0 & x \text{ n'est pas un élément de } A\end{cases} \tag{7.2}$$

7.3 Les fonctions d'appartenance : L'ensemble flou A peut être défini par un ensemble de pairs 'degré d'appartenance-élément' [3] :

$$A=\int_X \mu_A(x)/x \tag{7.3}$$

On peut trouver plusieurs fonctions d'appartenance.

7.3.1 Fonction d'appartenance trapézoïdale et triangulaire

$$\mu(x,a,b,c,d)=\max\left(0,\min\left(\frac{x-a}{b-a},1,\frac{d-x}{d-c}\right)\right) \tag{7.4}$$

Où a,b,c *et* d sont les coordonnées des apex du trapèze. Si $b=c$ nous obtenons une fonction triangulaire, voir figure 7.1.

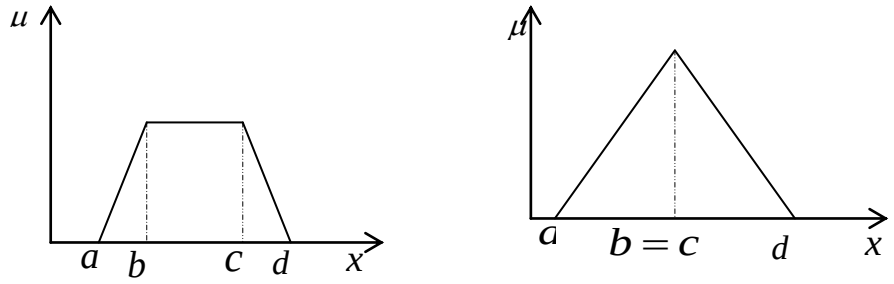


Figure 7.1 Fonctions d'appartenance trapézoïdale et triangulaire

7.3.2 Fonction d'appartenance exponentielle

$$\mu(x, c_g, c_d, w_g, w_d) = \begin{cases} \exp(-(x - c_g / 2w_g)^2) & si\ x < c_g \\ \exp(-(x - c_d / 2w_d)^2) & si\ x < c_d \\ 1 & autrement \end{cases} \quad (7.5)$$

Où c_g et c_d sont les limites gauche et droite, respectivement ; et w_g et w_d sont les largeurs gauche et droite.

7.3.3 Fonction d'appartenance gaussienne

$$\mu(x, c, \sigma) = \exp(-(x - c / 2\sigma)^2) \quad (7.6)$$

7.3.4 Fonction d'appartenance singleton

$$\mu(x) = \begin{cases} 1 & si\ x = x_0 \\ 0 & autrement \end{cases} \quad (7.7)$$

7.4 Variables linguistiques : Le concept de variable linguistique est toute la nouveauté qu'apporte la logique floue. Donc on utilise des mots ou des phrases au lieu de nombres. 'température' est une variable linguistique dont les valeurs peuvent être : très chaud, chaud, tiède, froid, très froid. Cet ensemble des valeurs que peut prendre la variable linguistique est appelé ensemble des termes. Chaque terme (chaud….) est défini par une fonction d'appartenance $\mu_{chaud}(x)$ qui associe à chaque valeur x de 'température' un degré de vérité d'être tel ou tel terme (chaud) avec bien sur la condition $\mu_{chaud}(x)$ [7].

7.5 Sous ensemble flou

7.5.1 $\alpha - coupe$ **:** $\alpha - coupe$ d'un ensemble floue A_α est un sous ensemble ordinaire de l'univers de discours X dont ses éléments possèdent un degré d'appartenance supérieur ou égale à α :

$$A_\alpha = \{x / \mu_A(x) \geq \alpha \quad \alpha \in [0\ 1]\} \tag{7.8}$$

7.5.2 Support : Le support d'un ensemble floue A est un sous ensemble ordinaire de l'univers de discours X dont tout ses éléments possèdent un degré d'appartenance non nul :

$$\text{supp}(A) = \{x / \mu_A(x) > 0\} \tag{7.9}$$

7.5.3 Noyau : Le noyau d'un ensemble flou A est un sous ensemble ordinaire de l'univers de discours X dont tous ses éléments possèdent un degré d'appartenance unitaire :

$$core(A) = \{x / \mu_A(x) = 1\} \tag{7.10}$$

7.5.4 La hauteur : La hauteur d'un ensemble flou A notée $h(A)$, représente la valeur maximale de la fonction d'appartenance sur tout l'univers X :

$$h(A) = Sup_X \mu_A(x) \qquad \forall\ x \in \text{X} \tag{7.11}$$

7.5.5 Point de croisement : C'est l'élément x_c de l'univers X dont $\mu_A(x_c) = 0.5$.

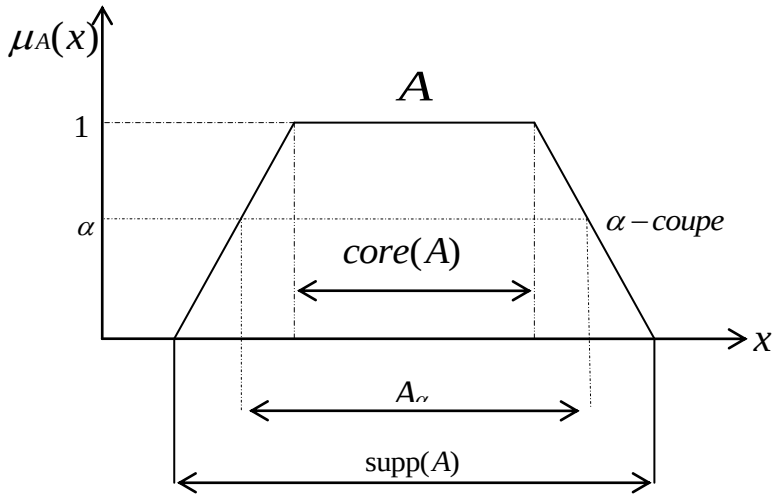


Figure 7.2 Sous ensembles flous

7.6 Opérations sur les ensembles flous [4]

- Un ensemble flou est dit vide si sa fonction d'appartenance est nulle sur tout X.
- Le complément d'un ensemble flou A nommé A' est caractérisé par sa fonction d'appartenance : $\mu_{A'} = 1 - \mu_A$
- Si $\mu_A(x) = \mu_B(x)$ *pour tout* $x \in \mathrm{X}$ on dit que les deux ensembles flous sont égaux.
- On dit que $A \subset B$ si $\mu_A(x) \leq \mu_B(x)$ (7.12)
- L'union de deux ensembles flous A et B est un ensemble flou C :

$$C = A \cup B \qquad \text{et} \qquad \mu_C(x) = \max[\mu_A(x), \mu_B(x)] \tag{7.13}$$

- L'intersection de deux ensembles flous A et B est un ensemble flou C :

$$C = A \cap B \qquad \text{et} \qquad \mu_C(x) = \min[\mu_A(x), \mu_B(x)] \tag{7.14}$$

7.7 Règles floues en logique floue

L'une des composantes fondamentales d'un système flou, c'est les règles floues [8]. Ces règles sont matérialisées par des implications logiques de type IF-THEN :

$$R_i : IF\ x\ est\ A_i\ \ THEN\ \text{y}\ est\ B_i \qquad i = 1, 2, ..., m \tag{7.15}$$

x est la variable linguistique d'entrée (antécédent), A_i est le terme linguistique d'antécédent (Prémisse), y est la variable linguistique de sortie (conséquence) et B_i terme de la conséquence. Les variables x et y, et les termes A_i et B_i sont des ensembles flous définies dans leurs domaines respectifs. Les fonctions d'appartenances des ensembles flous antécédents et conséquents sont des applications $\mu_A(x) : X \longrightarrow [0\ 1]$ et $\mu_B(x) : X \longrightarrow [0\ 1]$. Les termes linguistiques A_i et B_i sont sélectionnés d'un ensemble de termes prédéfinis, comme très froid, froid, tiède, chaud, très chaud. L'ensemble des termes dont appartient A_i et B_i sont notés $\mathscr{A}$ et $\mathscr{B}$ soit $A_i \in \mathscr{A}$ et $B_i \in \mathscr{B}$.

7.7.1 Base de règles : Une base de règle R est l'ensemble des règles qui régisse le système flou.

$$\mathrm{R} = \{\mathrm{R}_i / i = 1, 2, .., m\} \quad m : \text{Nombre des règles} \tag{7.16}$$

7.7.2 Base de connaissances: La base des règles R et les ensembles $\mathscr{A}$ et $\mathscr{B}$ constituent la base de connaissance d'un système flou.

7.8 Modèle flou

Le modèle flou est défini selon la fonction du terme B_i de la conséquence d'une règle. Deux types de modèle flou peuvent être cités :

- Modèle Flou de Mamdani : l'antécédent et la conséquence sont des termes flous.
- Modèle de Takagi-Sugeno : La conséquence est une fonction ordinaire des antécédents.

Les points traités concernent juste le modèle flou de madani.

7.8.1 Mécanisme d'inférence : La formulation des inférences dépend inévitablement du comportement statique et dynamique du système à contrôler et du but envisagé.
Le mécanisme d'inférence et en fonction des antécédents détermine les règles actives puis déduit la valeur floue de la conséquence. Bien sur numériquement le mécanisme d'inférence raisonne sur le degré d'appartenance de l'antécédent au terme flou, définie par une fonction d'appartenance [11], [12].

Les opérateurs ET, OU et ALORS de la base des règles sont remplacés lors de l'opération d'inférence par des opérateurs mathématiques qui agissent sur les fonctions d'appartenances.
Parmi les méthodes d'inférence les plus utilisées on cite :

- Méthode d'inférence de Mamdani (max-min) : où 'ET' et 'ALORS' sont remplacés par 'min' et 'OU' par 'max'.
- Méthode d'inférence max-prod : dans cette méthode 'ET' et 'ALORS' sont remplacés par 'produit' et 'OU' par 'max'.
- Méthode d'inférence somme-prod : dans cette méthode 'ET' et 'ALORS' sont remplacés par 'produit' et 'OU' par 'moyen'.

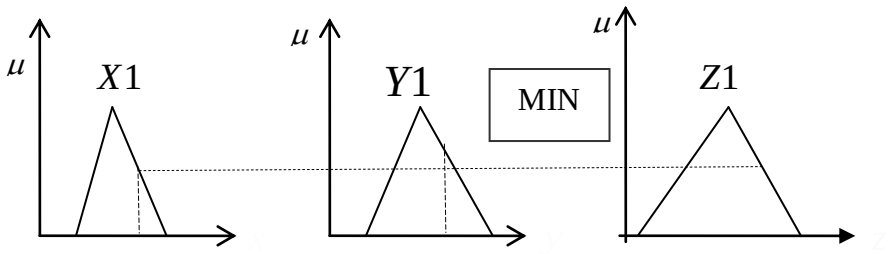


Figure 7.3 Inférence de Mamdani par 'Min' pour la règle
if (x est X1 et y est Y1) Alors z est Z1

7.8.2 Agrégation des règles: Le degré d'activation d'une règle et fonction des propositions qui la constitue. Dans le cas ou plusieurs règles sont actives, comme en cas de la figure 7.4, l'agrégation de ces règles donne le résultat de la conséquence en considérant le degré d'activation de chaqu'une [2].

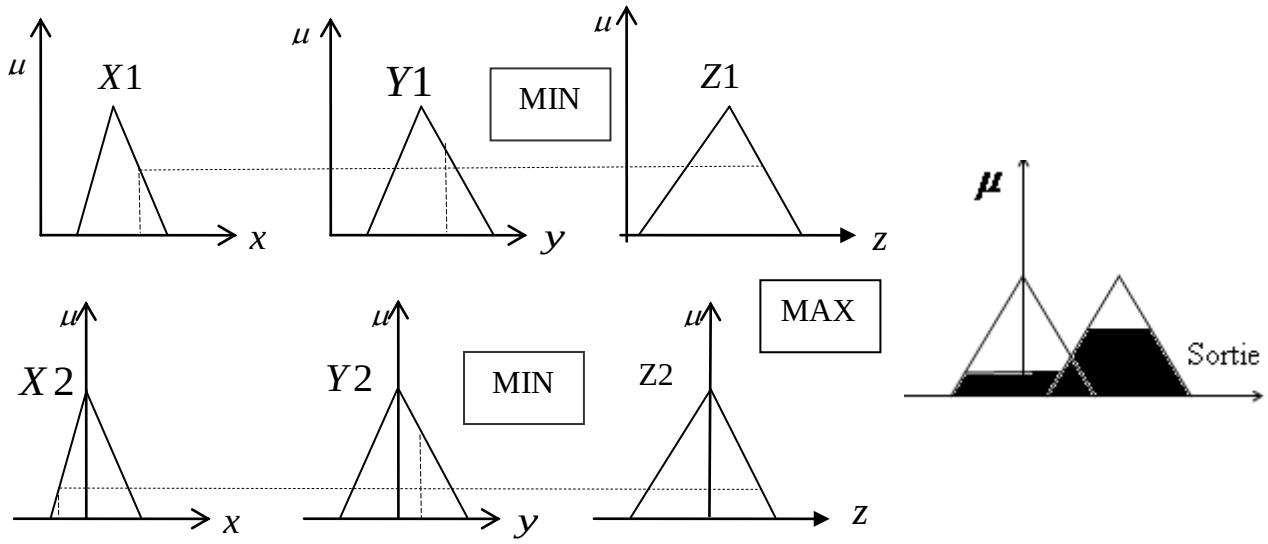


Figure 7.4 Agrégation des règles

7.8.3 La fuzzification : Cette phase permet de transformer l'entrée numérique en son équivalent flou comme illustré en figure 7.5 :

$$\textit{antécédent numérique}(x) \longrightarrow \textit{antécédent flou}(\textit{ensemble flou}, \mu(x))$$

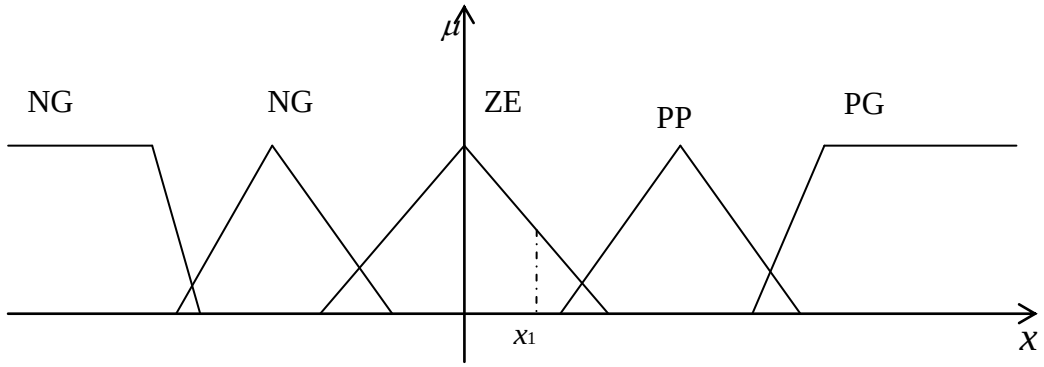


Figure 7.5 Fuzzification d'un antécédent x_1 ($x_1 \in ZE, \mu(x_1) = 0.55$)

7.8.4 La défuzzification : Cette phase vient après la phase d'inférence et permet de transformer une conséquence floue en une valeur numérique qui peut être utiliser pour réaliser l'objectif de commande. Plusieurs méthodes de défuzzification sont utilisées [7].

7.8.4.1 Défuzzification par le centre de gravité : C'est la méthode la plus utilisée en contrôle flou. Elle calcule le centre de gravité de l'ensemble flou résultant de l'inférence de l'ensemble des règles actives qui est le cas de la figure 7.6.

Soit v la variable linguistique de sortie, V l'ensemble de discours et v^* la sortie numérique après défuzzification alors :

$$v^* = \frac{\int_V v\mu(v)dv}{\int_V \mu(v)dv}$$

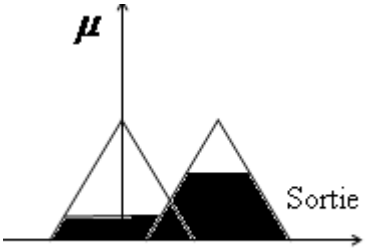


Figure 7.6 Défuzzification par le centre de gravité

La méthode du centre de gravité est gourmande en temps de calcul.

7.8.4.2 Défuzzification par le maximum: Pour cette méthode la valeur de sortie est l'abscisse de la valeur maximale de la fonction d'appartenance résultante après inférence, voir figure 7.7.

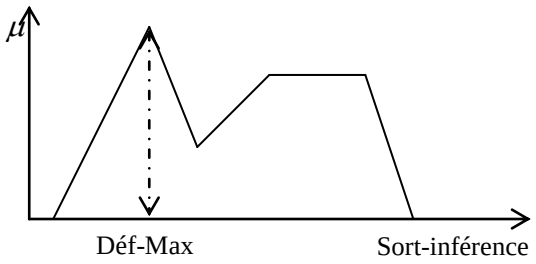


Figure 7.7 Défuzzification par le maximum

7.8.4.3 Défuzzification par la moyenne des maxima : Pour cette méthode on détermine les maximums de la même façon que la méthode précédente puis on calcule la valeur moyenne de ces maxima.

7.8.4.4 Défuzzification par le centre de la plus grande surface : Dans cette méthode la résultat de la défuzzification est calculé comme le centre de gravité de la plus grande surface, parmi les fonctions d'appartenance caractérisants la fonction d'appartenance résultante.

7.8.4.5 Premier et dernier maximum : Lorsque la surface de sortie possède des maximums égaux (palier) mais ayant des abscisses différentes alors on a le choix entre l'abscisse du premier ou le dernier maximums.

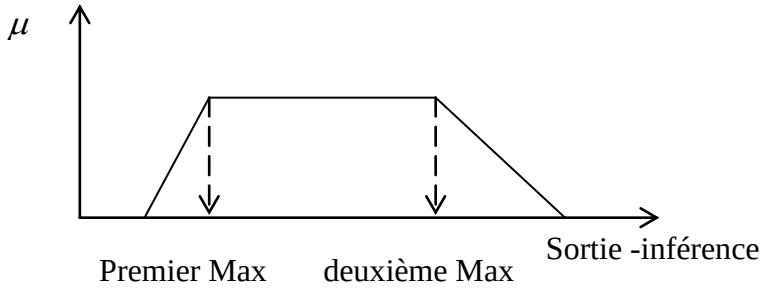


Figure 7.8 Défuzzification par premier ou dernier maximum

7.8.4.6 Milieu entre premier et dernier maxima : Comme son nom l'indique la méthode donne comme résultat l'abscisse qui est au milieu des abscisses du premier et dernier maximum. Regarder la figure 7.9 exposant cette méthode.

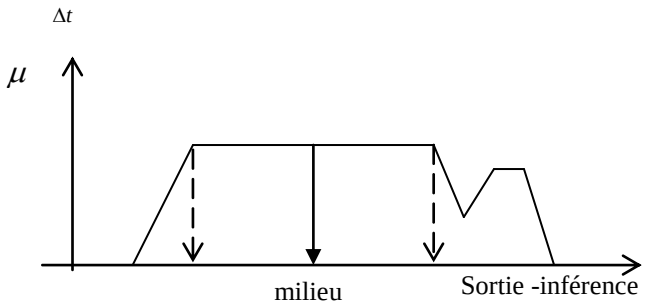


Figure 7.9 Défuzzification par le milieu

7.9 Le contrôleur flou

La structure d'un contrôleur flou d'une manière générale est composée de trois parties comme montrées en figure 7.10 ci-dessous.

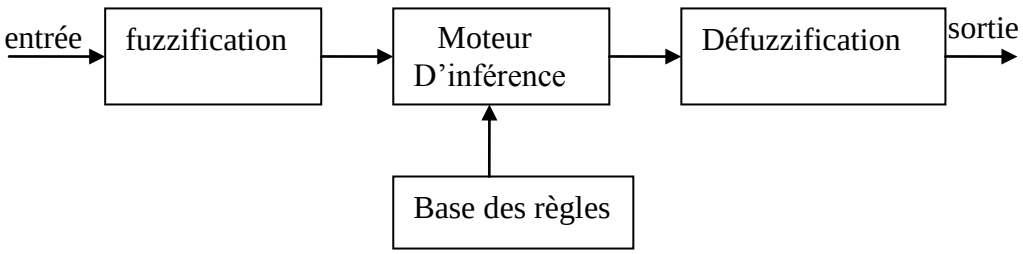


Figure 7.10 Structure générale d'un contrôleur flou

7.9.1 Le module de fuzzification : Traite les entrées du système et permet d'associer à chacune des entrées réelles par le biais des fonctions d'appartenances, un sous ensemble flous définis sur l'univers de discours [3],[5].

Les fonctions d'appartenances sont choisies en fonction de la nature des entrées, tel que leurs distributions sur l'univers de discours, leur excitation de la dynamique du système ou aussi la précision de la commande. Généralement pour faciliter les réglages du contrôleur flou nous choisissons des formes triangulaires. Les fonctions d'appartenances sont placées de telle manière qu'à tout moment il n'y ait que deux fonctions d'appartenances activées pour chaque entrée, donc juste deux règles seront activées permettant aussi de limiter le temps de calcule.

Ayant choisi le type des fonctions d'appartenances, il est bon de déterminer leur nombre, c'est-à-dire la couverture de l'univers de discours. Plus se nombre sera important, plus le nombre de sous ensemble flou sera considérable, et la sensibilité de la commande floue augmentera. Mais se nombre important d'ensemble flous engendre un nombre important de règles donc un temps de calcule plus grand pour déterminer l'ensemble des règles actives.

Généralement le nombre d'ensembles flous est impaire (trois, cinq, sept, neuf). Un ensemble se situe au milieu de l'univers et les autres se réparties équitablement de part et d'autre. La plus part du temps on fixe les fonctions d'appartenances extrêmes et on ajuste les autres, illustré en figure 7.11.

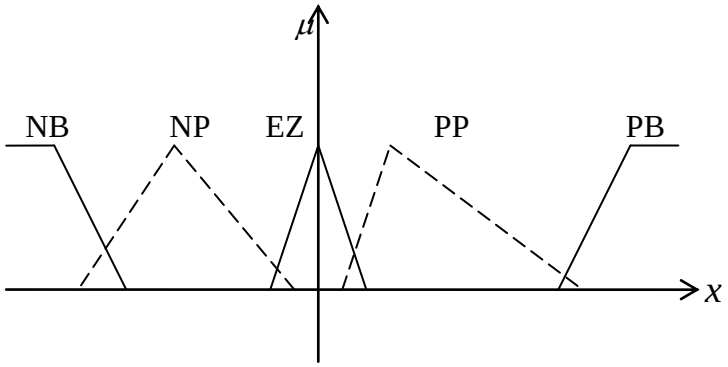


Figure 7.11 Illustration du choix et de l'ajustement des ensembles flous

7.9.2 Le module base de règles : Si le modèle du système à commander est partiellement connu il sera facile d'élaborer une base de règles du faite qu'on peut simuler le système et conclure sa dynamique. Dans le cas où on ne possède pas d'information sur le modèle alors on se base juste sur les connaissances de l'expert et

après plusieurs tentatives et avec précaution on peut établir l'ensemble des règles nécessaires pour contrôler le système.

La base de règles, et pour assurer une commande parfaitement correcte, doit assurer certaines caractéristiques à savoir :

- Continuité : l'ensemble de règles floues de prémisses adjacentes ont des conclusions adjacentes.
- Consistance : pas de contradiction entre les règles pour les prémisses identiques.
- Complétude : la base de règle est complète lorsque pour toute valeur d'entrée lui correspond une conséquence.

Parfois on utilise les algorithmes génétiques pour réaliser l'ajustement optimal des règles [15].

7.9.3 Le module défuzzification : la méthode du centre de gravité est la plus utilisée, parce qu'elle considère toute la surface de sortie générée par le moteur d'inférence. Du fait que cette méthode consomme assez de temps on fait parfois recours aux méthodes moins gourmandes telle que la moyenne des maxima. La puissance du calculateur nous offre plus de liberté dans le choix de la méthode de défuzzification [16].

Le choix de la méthode de défuzzification est parfois réalisé vis-à-vis de certains critères qui sont :

- Continuité : pour éviter des changements brusque de la commande.
- Désambiguïté : pour éviter le risque de ne pas pouvoir prendre décision entre deux choix identiques par exemple.
- Complexité des calcules : dans le cas pratique le temps de calcule influe le contrôle en temps réel.

7.9.4 La commande du système : Après inférence et défuzzification une valeur numérique de commande est obtenue au lieu d'une valeur linguistique, et sera utilisé pour agir sur le système. Cette valeur doit être adaptée au système par des procédés hardwares électroniques.

7.9.5 Contrôleur flou comme un PID : Construire un contrôleur PID flou revient à choisir ses entrées et ses sorties et les règles floues adéquates [3]. Généralement on choisie pour les variables d'entrées l'erreur 'e' et la variation de l'erreur 'Δe' et la

somme des erreurs. On choisi pour la conséquence ou la sortie du contrôleur soit la commande $u(t)$ soit $\Delta u(t)$.

Il faut toujours mettre dans la tête le caractère non linéaire du contrôleur flou. Ce caractère se manifeste via les fonctions d'appartenance (gaussienne, trapézoïdale,...etc.) les fonctions d'inférence (MAX, MIN) et les méthodes de défuzzification (max , moyen des max).

7.9.5.1 Contrôleur PD flou

L'équation classique d'un contrôleur PD est comme suit :

$$u(k) = k_p e(t) + k_d \Delta e(t) \tag{7.17}$$

Où k_p et k_d sont les gains de proportionnalité et de dérivation. Le schéma bloc d'un contrôleur PD Flou présenté en figure 7.12 peut avoir la forme suivante [3]:

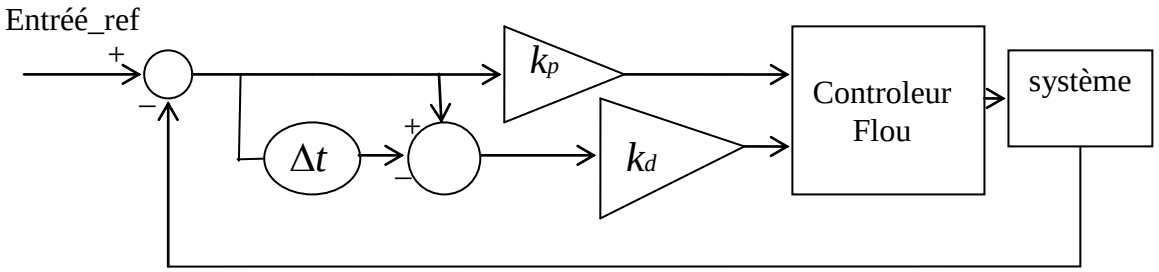


Figure 7.12 Contrôleur PD Flou

La base des règles du contrôleur flou proposée et comme suit :

Δe / e	PG	PM	PP	Z	NP	NM	NG
PG	NG	NG	NG	NG	NM	NP	Z
PM	NG	NG	NG	NM	NP	Z	PP
PP	NG	NG	NM	NP	Z	PP	PM
Z	NG	NM	NP	Z	PP	PM	PG
NP	NM	NP	Z	PP	PM	PG	PG
NM	NP	Z	PP	PM	PG	PG	PG
NG	Z	PP	PM	PG	PG	PG	PG

Figure 7.13 La base des règles

Voir en annexe A l'interprétation de cette base de règles.

7.9.5.2 Contrôleur PI flou: L'équation conventionnel du contrôleur PI est donnée par :

$$u(t) = k_p e(t) + k_i \int e(t)dt \tag{7.18}$$

Dans ce cas les entrées du contrôleur PI flou sont $e(t)$ et $\int e(t)$, comme présentées en figure 7.14; mais du fait que l'intégrateur peut aboutir à des entrées importantes qui dépassent l'univers de discours on peut modifier les choses de la façon suivante :

$$\Delta u(t) = k_p \Delta e(t) + k_i e(t) \tag{7.19}$$

Donc le contrôleur flou aura pour entrée Δe et e, et pour sortie Δu, et après une intégration on aura $u(t)$; ce qui est équivalent à dire :

$$\int (k_p \Delta e(t) + k_i e(t))d(t) = k_p \int \Delta e(t)d(t) + \int k_i e(t)d(t) = k_p e(t) + k_i \int e(t)dt \tag{7.20}$$

Sa revient à une représentation type régulateur PI ce qui permet d'utiliser la même base de règles, figure7.15.

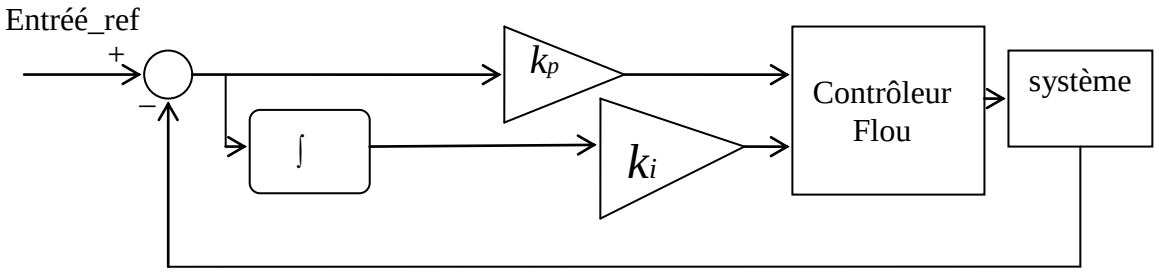
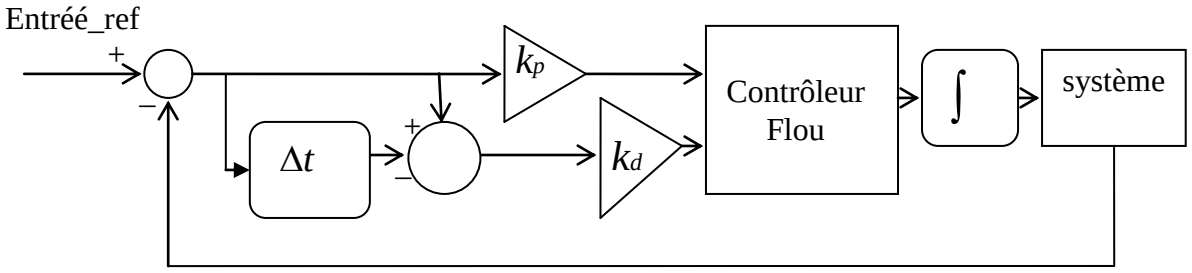


Figure 7.14 Contrôleur PI flou (version1)



(b)

Figure 7.15 Contrôleur PI flou (version2)

7.9.5.3 Contrôleur PID flou: L'équation du contrôleur conventionnel est comme suit :

$$u(t) = k_p e(t) + k_d \Delta e(t) + k_i \int e(t)dt \tag{7.21}$$

Les entrées du contrôleur sont donc au nombres de trois e , Δe et $\int e$, donc un contrôleur flou à trois entrées nécessitant une base de règles de 343(7x7x7) règles. Il n'est pas logique de travailler avec ce grand nombre de règles. La solution est donc de travailler avec deux contrôleurs flous parallèles ayant le même effet qu'un PID flou, soit un PD flou et un PI flou :

$$u(t) = (k_p / 2e(t) + k_d \Delta e(t)) + (k_p / 2e(t) + k_i \int edt) \tag{7.22}$$

Le schéma bloc équivalent est :

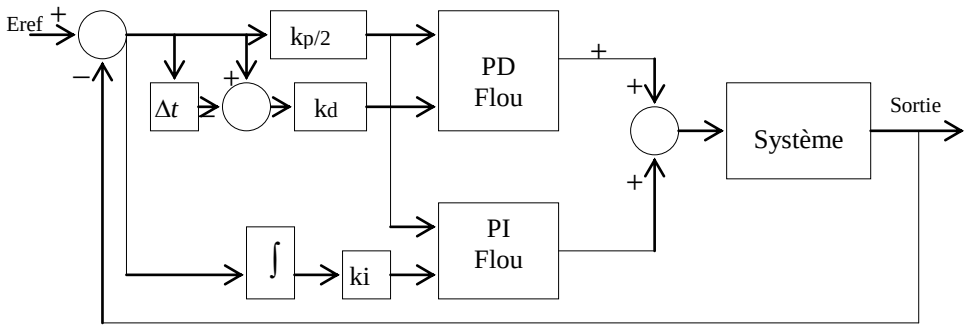
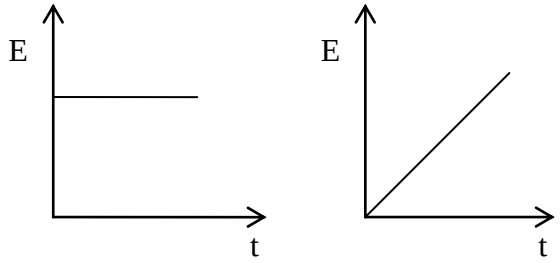


Figure 7.16 Structure parallèle d'un contrôleur PID flou

7.9.6 Stabilité et performances de contrôleurs Flous

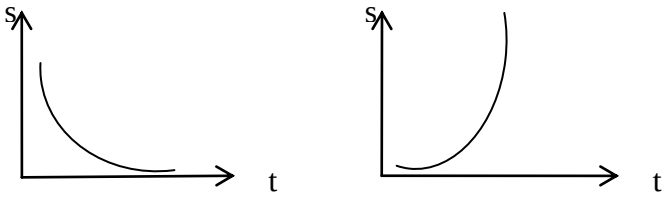
7.9.6.1 Evaluation de la stabilité et la performance par observation de la sortie du système de contrôle : Parfois on juge la performance et la stabilité du système de contrôle par l'étude de la réponse du système. La courbe de sortie reflète la dynamique de système. Comme entrées d'excitation on peut utiliser l'échelon ou la droite [18].



(a) échelon (b) droite

Figure 7.17 Exemple de fonctions d'excitation

L'interprétation de sortie du système peut nous confirmer sa stabilité cela comme indiqué sur la figure ci-dessous :



(a) sortie stable (b) sortie instable

Figure 7.18 Sorties de systèmes

Les méthodes classiques d'étude de stabilité tel que critère de Nyquist et critère de Routh ne s'appliquent pas directement en contrôle flou du fait qu'ils nécessitent le modèle mathématique du système, ce qui n'est pas disponible.

7.9.6.2 Indicateurs de performance et de stabilité : La réponse à un échelon donne quelques indices de performance, tel que le temps de montée, la valeur du Pic et le temps de stabilisation.

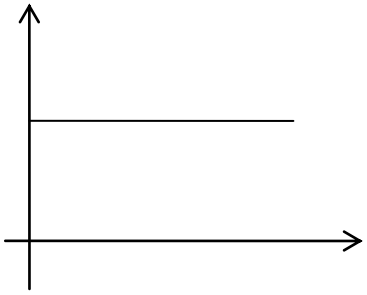


Figure7.19 Indicateurs de performance

Le système manifeste une instabilité si le temps de montée est trop petit ou le rapport de dépassement est important. Pour remédier à cela il faut ajuster la base de règles pour assurer le passage souple d'une règle à une autre.

Il est possible de percevoir la stabilité du système par l'analyse de la tendance de l'évolution des règles actives lors de la régulation. Des méthodes systématiques permettent à partir d'un ensemble de données de déterminer les ensembles flous leurs fonctions d'appartenances leurs optimisations et même l'étude de la stabilité [20].

Δe / e	PG	PM	PP	Z	NP	NM	NG
PG	NG	NG	NG	NG	NM	NP	Z
PM	NG	NG	NG	NM	NP	Z	PP
PP	NG	NG	NM	NP	Z	PP	PM
Z	NG	NM	NP	Z	PP	PM	PG
NP	NM	NP	Z	PP	PM	PG	PG
NM	NP	Z	PP	PM	PG	PG	PG
NG	Z	PP	PM	PG	PG	PG	PG

(a) stable

Δe / e	PG	PM	PP	Z	NP	NM	NG
PG	NG	NG	NG	NG	NM	NP	Z
PM	NG	NG	NG	NM	NP	Z	PP
PP	NG	NG	NM	NP	Z	PP	PM
Z	NG	NM	NP	Z	PP	PM	PG
NP	NM	NP	Z	PP	PM	PG	PG
NM	NP	Z	PP	PM	PG	PG	PG
NG	Z	PP	PM	PG	PG	PG	PG

(b) instable

Figure 7.20 indice de stabilité

7.10 Régulateur flou pour la génération de trajectoire d'un robot manipulateur

7.10.1 Introduction : comme réalisé au chapitre1 le contrôle d'un robot manipulateur consiste à faire suivre à l'effecteur une trajectoire que les articulations doivent assurer. Alors la régulation consiste à générer les couples adéquats aux articulations.

Si en régulation PID et couple- calculé le modèle dynamique du robot est considéré partiellement au totalement, la régulation par la logique floue omis le modèle dynamique et se base plus sur l'expert.

7.10.2 structure du contrôleur : Comme vu le modèle dynamique du bras manipulateur est non linéaire et couplé c à d l'effet des forces centrifuges et corriolises de chaque articulation sur l'autre est considérable. Mais nous avons associe un contrôleur flou pour chaque articulations et indépendamment l'un de l'autre comme illustré en figure 7.21.

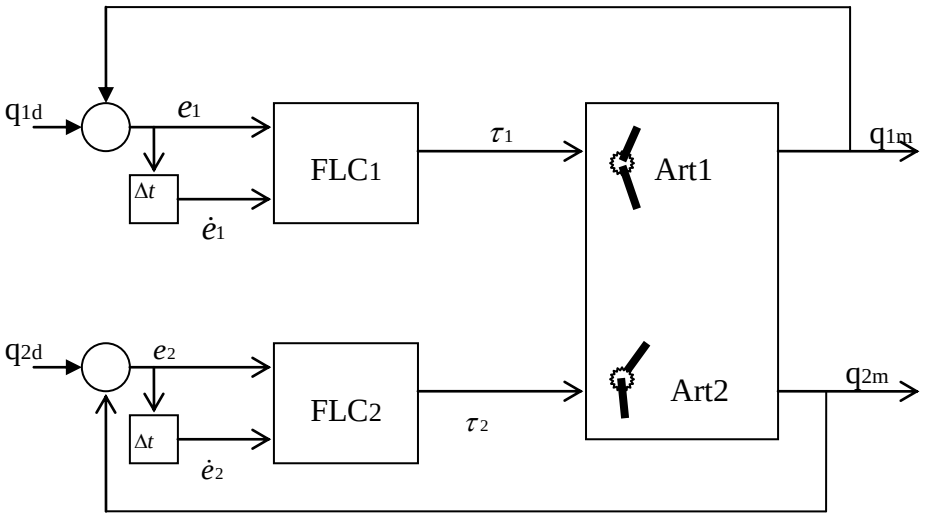


Figure 7.21 Structure du contrôleur flou pour articulations d'un robot manipulateur

7.10.3 Fonctions d'appartenance des entrées e_1 , $\dot{e}_1$, e_2 et $\dot{e}_2$:

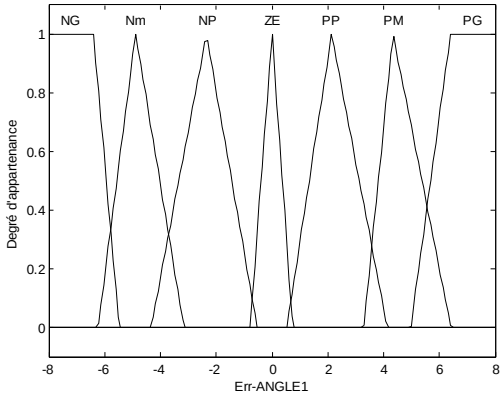


Figure 7.22 fonction d'appartenance de e_1

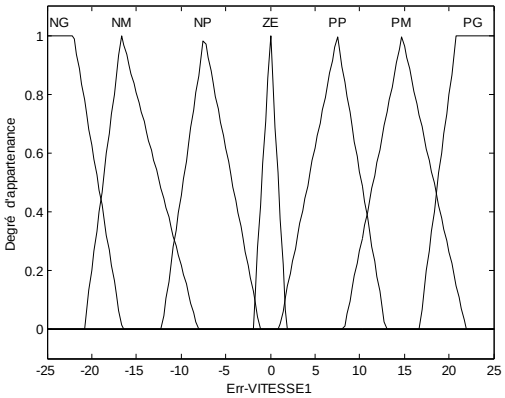


Figure 7.23 fonction d'appartenance de $\dot{e}_1$

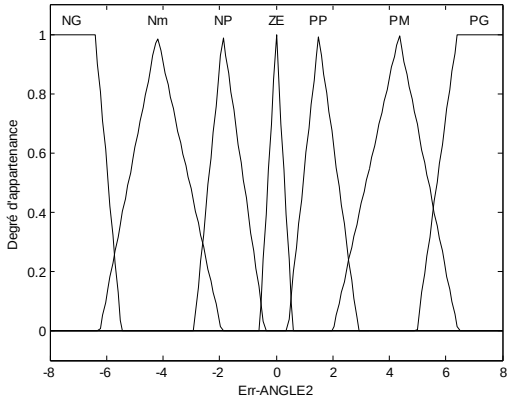


Figure 7.24 fonction d'appartenance de e_2

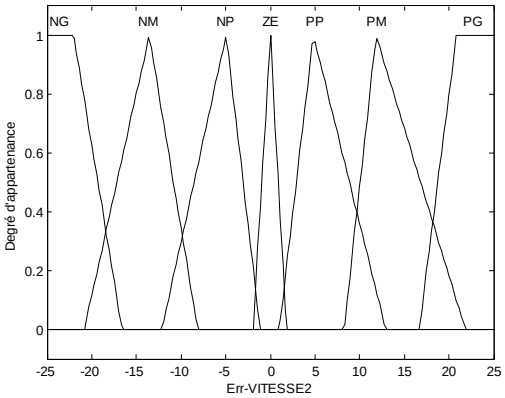


Figure 7.25 fonction d'appartenance de $\dot{e}_2$

7.10.4 Fonctions d'appartenance des commandes τ_1 et τ_2 :

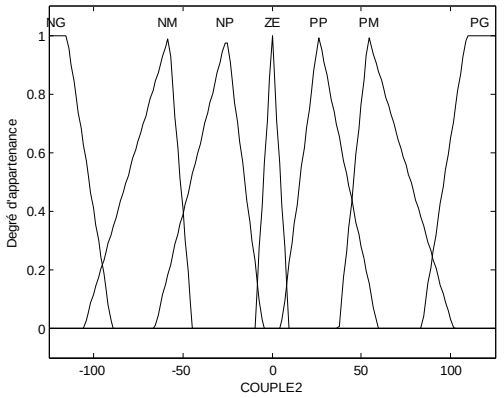


Figure 7.26 fonction d'appartenance de τ_2

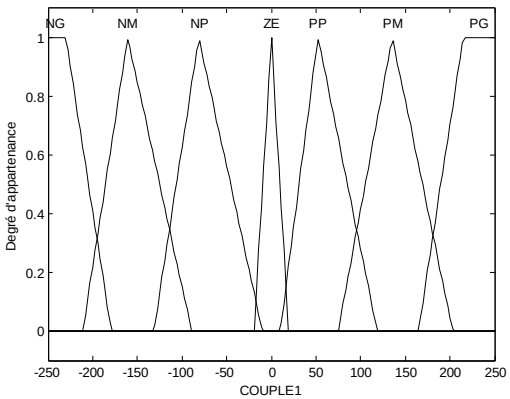


Figure 7.27 fonction d'appartenance de τ_1

7.10.5 Base des règles :

Δe / e	PG	PM	PP	Z	NP	NM	NG
PG	NG	NG	NG	NG	NM	NP	Z
PM	NG	NG	NG	NM	NP	Z	PP
PP	NG	NG	NM	NP	Z	PP	PM
Z	NG	NM	NP	Z	PP	PM	PG
NP	NM	NP	Z	PP	PM	PG	PG
NM	NP	Z	PP	PM	PG	PG	PG
NG	Z	PP	PM	PG	PG	PG	PG

Figure 7.28 base des règles du contrôleur flou

7.10.6 Exemples de simulation :

Exemple 1 :

$Ampli = pi/4$; $fr = 0.5hz$; $T = 0.001s$

Consigne à suivre pour articulation 1: $q_{1d} = Ampli * \sin(wt)$

Consigne à suivre pour articulation 2: $q_{2d} = Ampli * \sin(wt)$

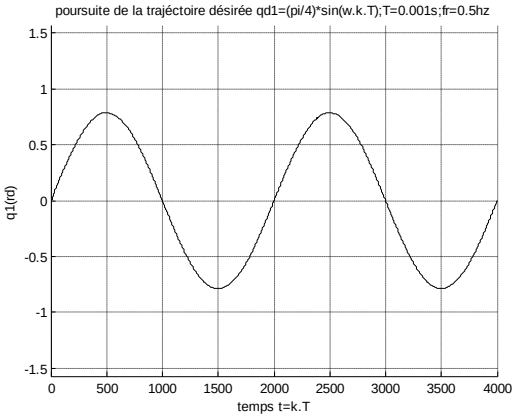


Figure 7.29 Angle q_1 mesuré

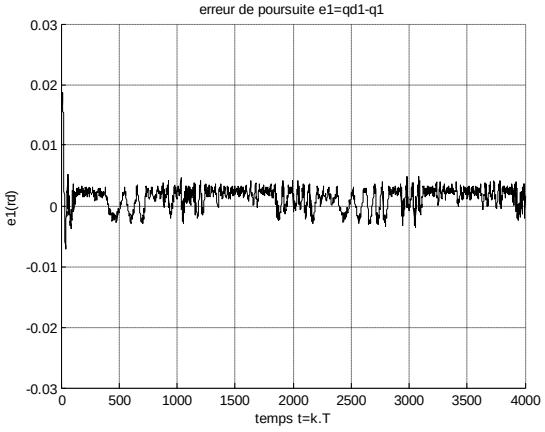


Figure 7.30 Erreur de poursuite e_1 mesurée

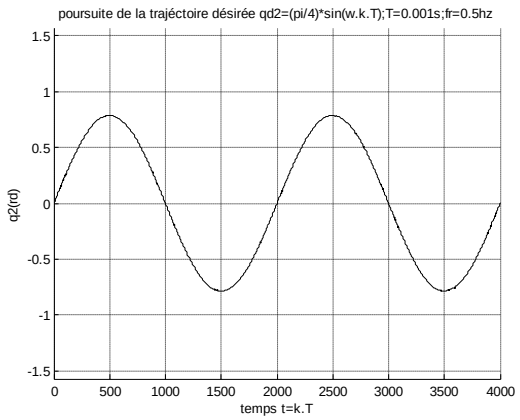


Figure 7.31 Angle q_2 mesuré

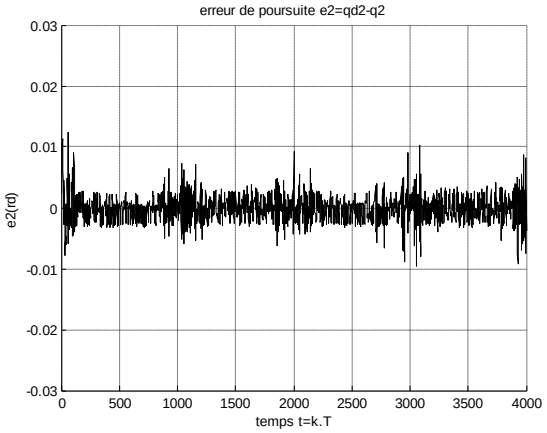


Figure 7.32 Erreur de poursuite e_2

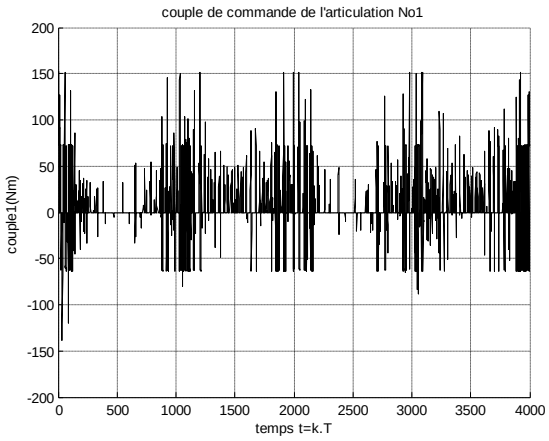


Figure 7.33 Couple de commande τ_1

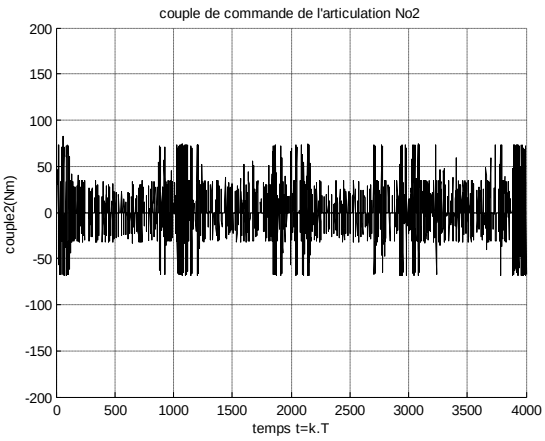


Figure 7.34 Couple de commande τ_2

Déduction1: le contrôleur flou nous à permet de faire suivre aux articulations des trajectoires sinusoïdales, figure 7.29 et 7.31, et avec faible erreur comme montré par les figures 7.30 à 7.32.

Exemple 2:

$Ampli = pi/2$ (Augmentation de l'amplitude) ; $fr = 0.5hz$; $T = 0.001s$

Consigne à suivre pour articulation 1: $q_{1d} = Ampli * \sin(wt)$

Consigne à suivre pour articulation 2: $q_{2d} = Ampli * \sin(wt)$

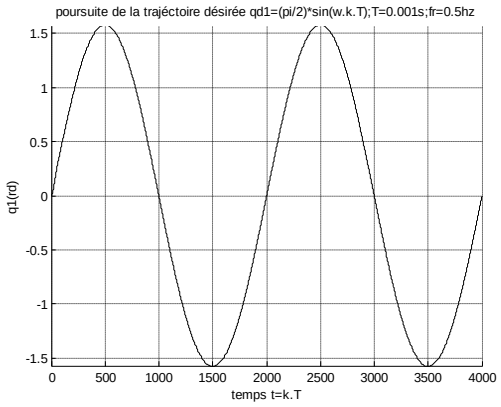


Figure 7.35 Angle q_1 mesuré

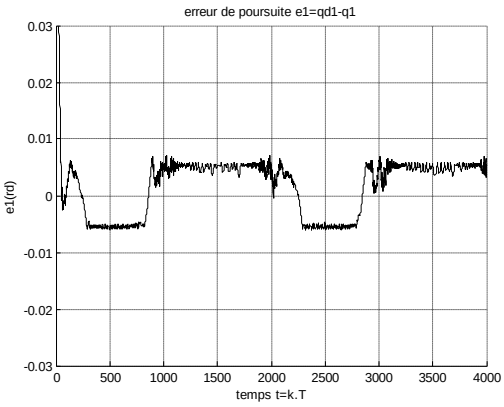


Figure 7.36 Erreur de poursuite e_1

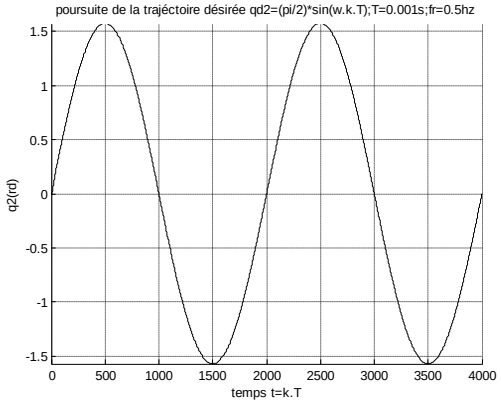


Figure 7.37 Angle q_2 mesuré

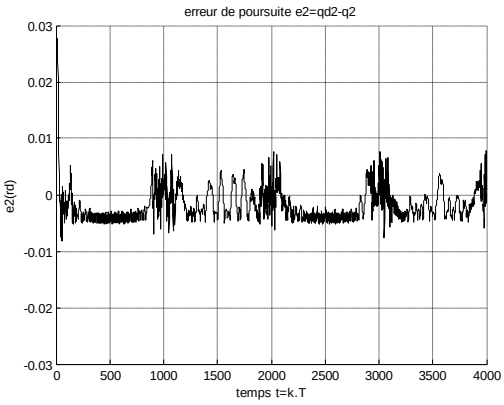


Figure 7.38 Erreur de poursuite e_2

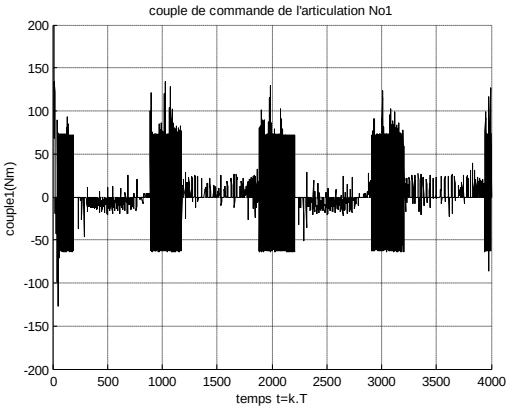


Figure 7.39 Couple de commande τ_1

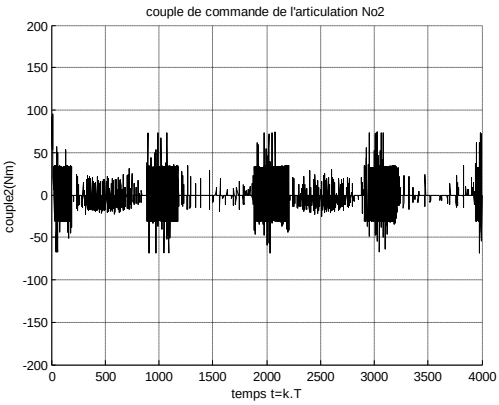


Figure 7.40 Couple de commande τ_2

Déduction 2 : On remarque que le contrôleur fonctionne d'une façon normale même en cas d'augmentation de l'amplitude de la consigne sauf une légère augmentation des erreurs, figures 7.36 et 7.38, quand peut réduire en cas de nécessité par les facteurs

d'amplifications kp et kd ou par le choix des ensembles flous et des fonctions d'appartenances.

Exemple 3:

Augmentation de la fréquence de la consigne $fr = 1hz$.

$Ampli = pi/4$; $T = 0.001s$

Consigne à suivre pour articulation 1: $q_{1d} = Ampli * \sin(wt)$

Consigne à suivre pour articulation 2: $q_{2d} = Ampli * \sin(wt)$

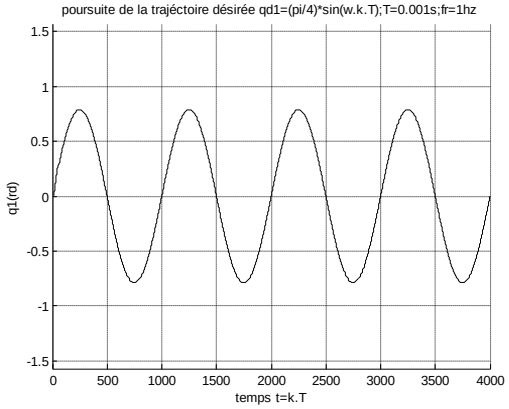


Figure 7.41 Angle q_1 mesuré

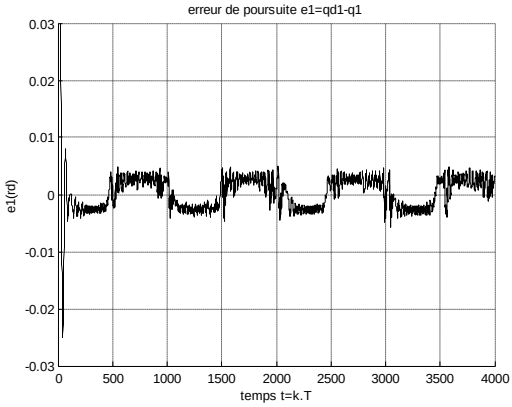


Figure 7.42 Erreur de poursuite e_1

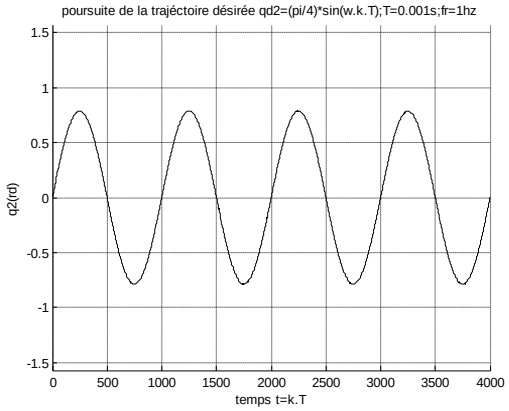


Figure 7.43 Angle q_2 mesuré

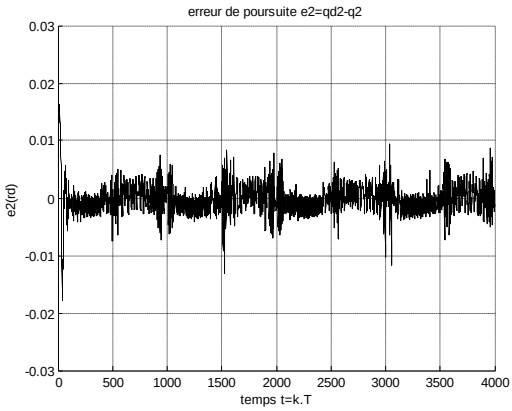


Figure 7.44 Erreur de poursuite e_2

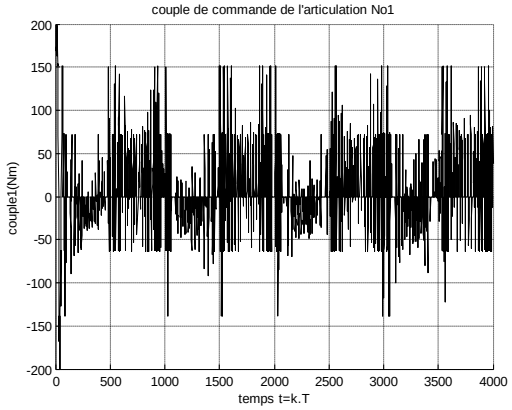


Figure 7.45 Couple de commande τ_1

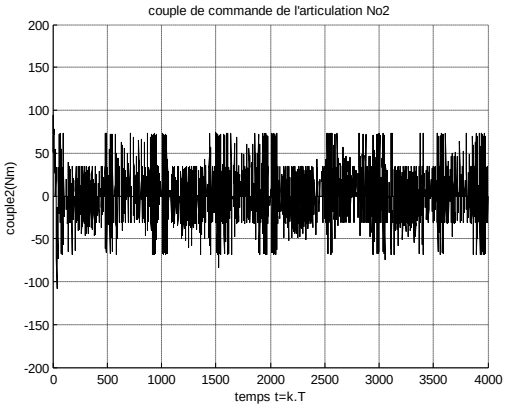


Figure 7.46 Couple de commande τ_2

Déduction 3 : Quoique la fréquence de la consigne sinusoïdale a doublée, figure 7.41 le résultat de la simulation ne diffère pas de celui de l'exemple1 et sans apporter aucune modification sur le contrôleur flou.

Exemple 4:

Par cet exemple on traite la régulation autour de la consigne pi/2. Avec la possibilité d'avoir un temps de montée petit et un faible dépassement assurant la convergence rapide.

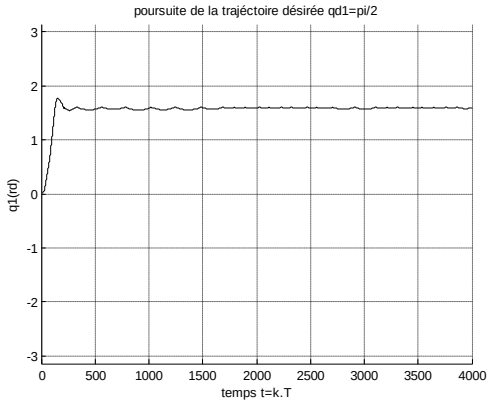


Figure 7.47 Angle q1 mesuré

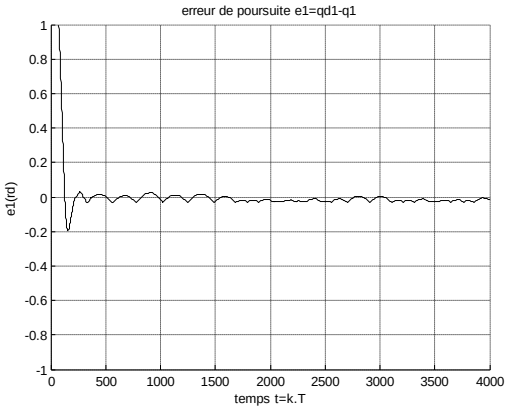


Figure 7.48 erreur de régulation e1

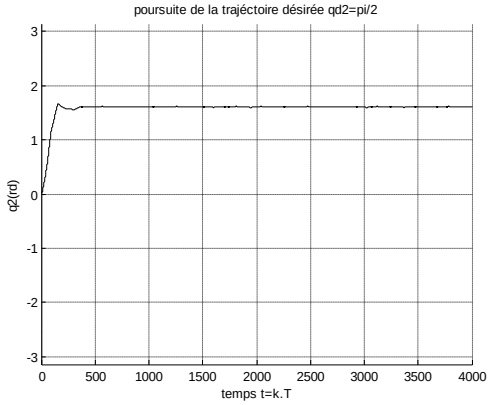


Figure 7.49 Angle q2

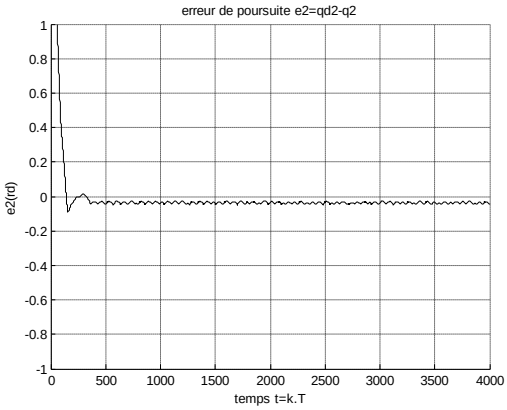


Figure 7.50 erreur de régulation e_2

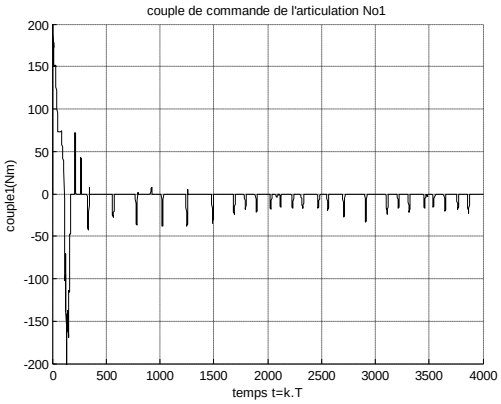


Figure 7.51 Couple de commande τ_1

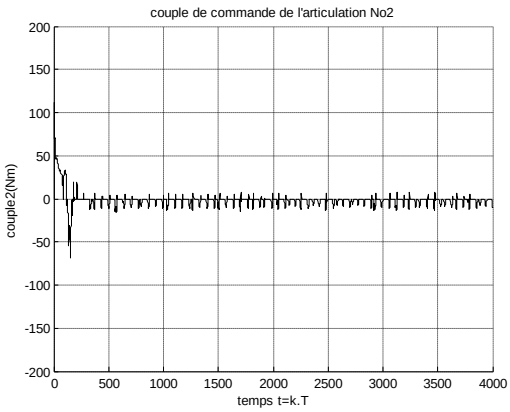


Figure 7.52 Couple de commande τ_2

Déduction 4 : le contrôleur vient d'assurer une bonne régulation autour de la consigne pi/2, figure 7.47 à 7.52, avec un bon temps de monté et un faible dépassement. Mais ce résultat est au détriment d'une modification des valeurs des gains ; qui prennent les nouvelles valeurs suivantes :

Kp1= 20 ; kp2= 10 ; kd1= 1; kd2=1 ;

Exemple 5 : Le but de cette simulation c'est de tester la poursuite d'une trajectoire sous forme de droite pour chaque articulation.

q_{d1}= (pi/8).k.T

q_{d2}= (pi/8).k.T

T=0.001s

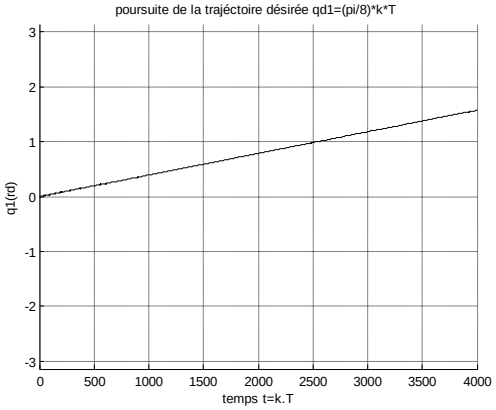


Figure 7.53 Angle q_1 mesuré

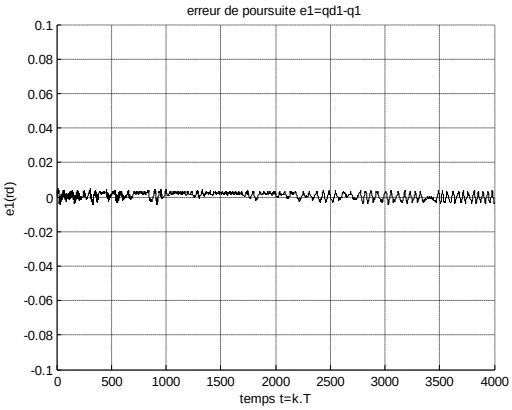


Figure 7.54 erreur de régulation e_1

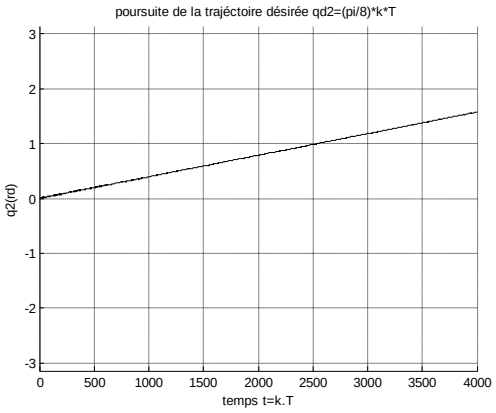


Figure 7.55 Angle q_2 mesuré

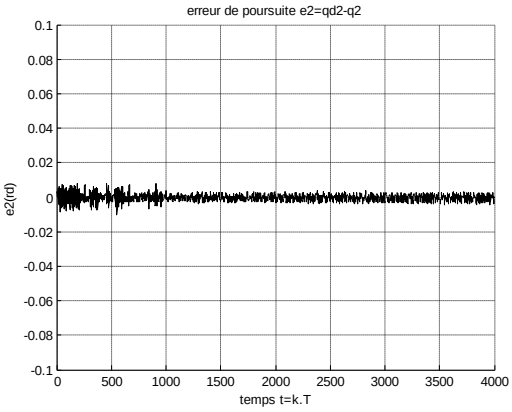


Figure 7.56 erreur de régulation e_2

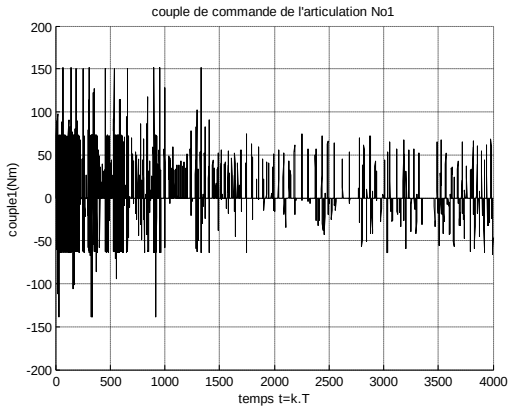


Figure 7.57 Couple de commande τ_1

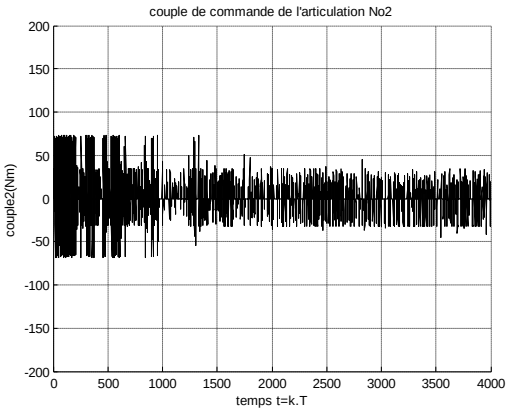


Figure 7.58 Couple de commande τ_2

Déduction 5: Le changement de l'allure de la consigne en une droite ne sujette pas des modifications sur le contrôleur. Les résultats de la poursuite sont toujours très bons comme présentés sur les figures 7.53 à 7.58.

Exemple 6 : Pour ce cas on étudie l'effet d'une opération de manutention au cours de la poursuite d'une trajectoire. Ce problème est simulé par le changement de la charge de l'effecteur soit du deuxième lien du robot.

Alors :

Si t>2s m2 est augmentée de 4kg soit m2=5kg.

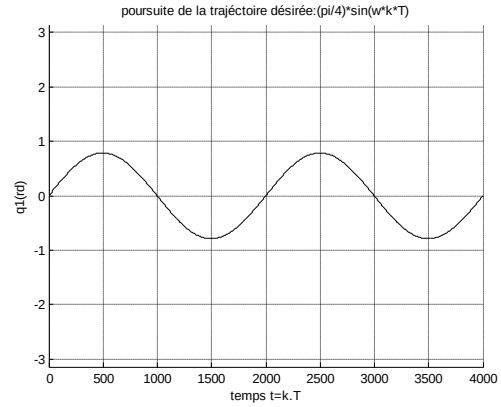


Figure 7.59 Angle q_1 mesuré

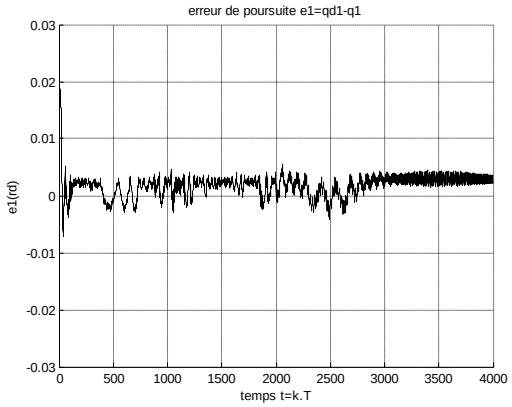


Figure 7.60 erreur de régulation e_1

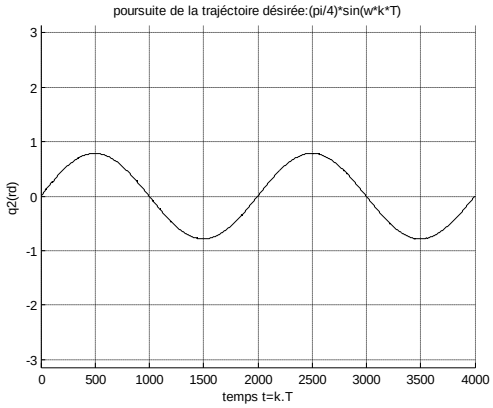


Figure 7.61 Angle q_2 mesuré

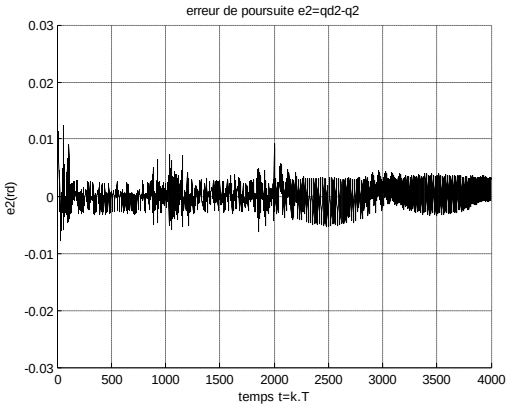


Figure 7.62 erreur de régulation e_2

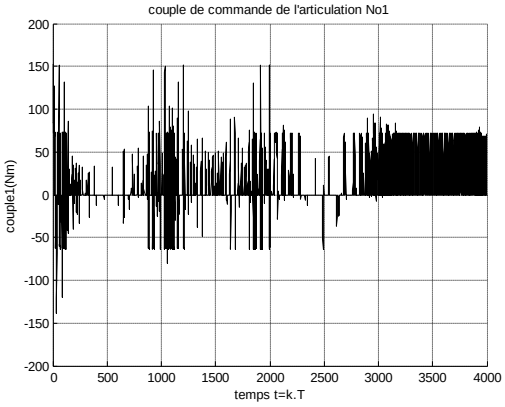


Figure 7.63 Couple de commande τ_1

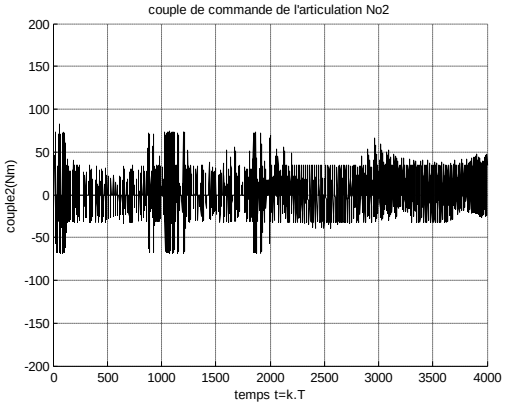


Figure 7.64 Couple de commande τ_2

Déduction 6 : Le changement brusque de la masse du deuxième lient du robot est bien supporté par le contrôleur flou comme montré sur les figure 7.59 à 7.64, sans nécessité de modifications sur le contrôleur flou.

Exemple 7: Dans cet exemple on veut tester le contrôleur s'il peut supporter une trajectoire en escalier.

Si t=0s Ampli =0.

Si t>0s Ampli = pi/4.

Si T>2s Ampli = pi/2.

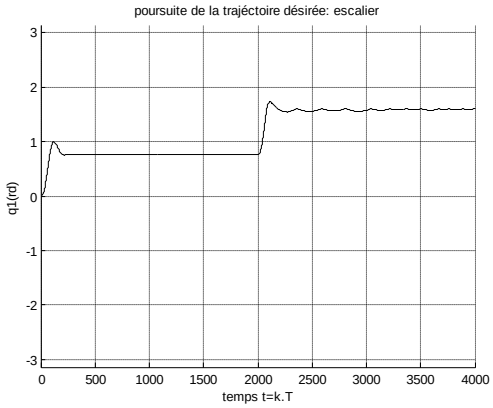


Figure 7.65 Angle q_1 mesuré

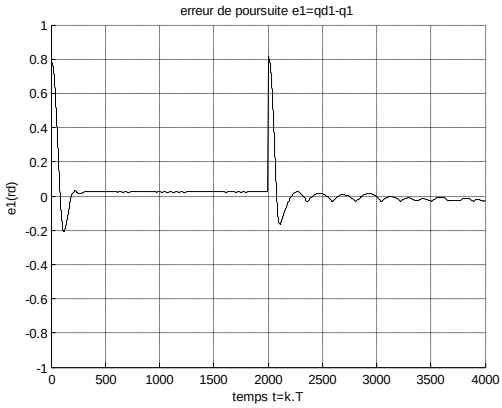


Figure 7.66 erreur de régulation e_1

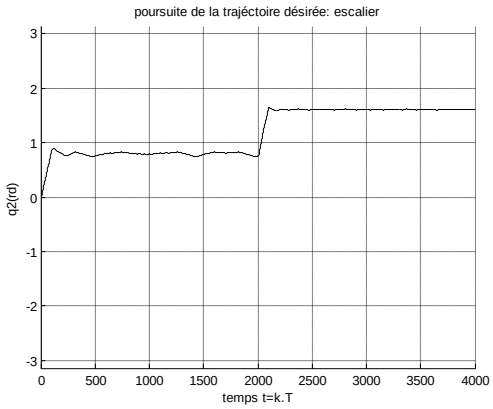


Figure 7.67 Angle q_2 mesuré

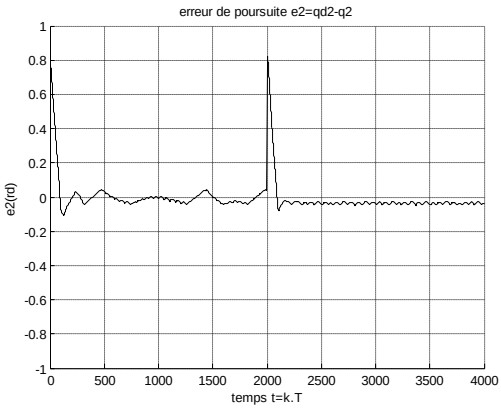


Figure 7.68 erreur de régulation e_2

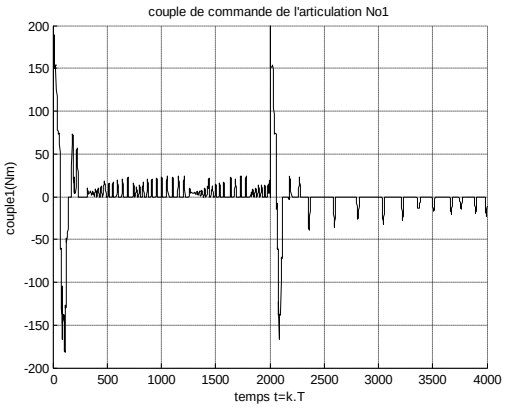


Figure 7.69 Couple de commande τ_1

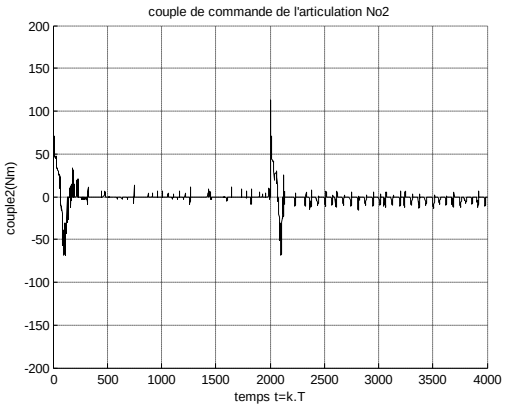


Figure 7.70 Couple de commande τ_2

Déduction 7: Il est évident d'après les figures 7.65 à 7.70 que le contrôleur supporte des sauts brusques de la consigne toute en garantissant un temps de réponse cours et un dépassement faible.

7.11 Conclusion: La logique floue nous a permet de contourner la difficulté du modèle mathématique en passant au modèle linguistique basé sur une expertise. Le contrôleur flou nous a permet de remplacer les contrôleurs classiques PD et PID, il permet généralement de compenser le bruit d'entrée et les variations de certains paramètres du système. Le contrôle flou est valable pour un système linéaire comme pour un système non linéaire. La logique floue peut être utilisée, en plus de la poursuite et la régulation, à gérer des architectures de contrôle intelligentes complexes.

BIBLIOGRAPHIE

[1] Kazuo Tanaka, Hua O. Wang, 'Fuzzy Control Systems Design and Analysis: A Linear matrix inequality approach', Wiley, John & Sons, 2001.

[2] Kwang H. Lee,' First Course on Fuzzy Theory and Applications', Springer, 2005.

[3] Jan Jantzan,'Foundations of Fuzzy control', 2007,John Wiley

[4] Toshinori Munakata',Fundamentals of the New artificial Intelligence: Neural, Evolutionary, Fuzzy and More', Springer,2008.

[5] Leonid Reznik, 'Fuzzy controllers', Newnes , 1997.

[6] William Siler, James J. Buckley,' Fuzzy Expert Systems and Fuzzy Reasoning', John Wiley & Sons, 2005.

[7] Didier Dubois, Henri Prade,' Fuzzy Sets and Systems: Theory and Applications', Academic Press, 1978.

[8] F. Martin McNeill, Ellen Thro ,'Fuzzy Logic:A Practicl Approach',Academic Press, 1994.

[9] Jyh-Shing Roger Jang, Chuen-Tsai Sun , Eiji Mizutani, 'Neuro-fuzzy and Soft Computing: A Computational Approach to learning and Machine Intelligence' Prentice Hall, 1997.

[10] Runtong Zhang, Yannis A. Phillis and Vassilis S. Kouikoglou, 'Fuzzy Control of Queuing Systems', Springer, 2005.

[11] J. Harris,'Fuzzy Logic Applications in Engineering Science', Springer, 2006.

[12] Ahmad M. Ibrahim, 'Fuzzy Logic for Embedded Systems Applications', ElsevierScience (USA), 2004.

[13] Hungxing li C.l. Philip Chen Han-Pang Huang ,'Fuzzy Neural Intelligent Systems Mathematical Foundation and the Applications in Engineering', CRC Press LLC,2001.

[14] José Galindo, Angélica Urrutia, Mario Piattini, 'Fuzzy databases :Modelling Design and Implementation',IGP,2006.

[15] Eric A. Wan,'Control Systems: Classical, Neural, and Fuzzy', Oregon Graduate ,Institute Lecture Notes – 1998.

[16] Kevin M. Passino, Stephen Yurkivich, 'Fuzzy Control', Addison-Wesly,1998.

[17] James J. Buckley, Leonard J. Jowers, 'Simulating Continuous Fuzzy Systems', Springer, 2006.

[18] Petr Pivonka, 'Comparative Analysis of Fuzzy PI/PD/PID Controller Based on classical PID Controller Approach',

[19] Jun Zhao,'System Modelling, Identification and Control Using Fuzzy Logic, these de Doctorat, Université Catholique de Louvain, 1995.

[20] José Valente de Oliveira, Wotod Pedrucs, 'Advance in Fuzzy Clustering and Applications', John Wiley& Sons, Ltd, 2007.

CHAPITRE 8

Planificateur de trajectoire basé sur les comportements flous pour robot mobile

8. Planificateur de trajectoire basé sur les comportements flous pour robot mobile

8.1 Introduction : Le principe du raisonnement en comportements est introduit en premier lieu par Brooks [19], pour une architecture réactive. Le comportement est donc défini comme étant la réaction à tout événement. Si l'objectif que doit exécuter le robot mobile est unique on dit que le comportement est simple; mais si l'objectif est multitâche on dit que le comportement est complexe. Un comportement complexe est donc divisé en plusieurs comportements simples [4], [5].

Chaque comportement reçoit des informations sur l'environnement du robot mobile par l'intermédiaire des différents capteurs et par la suite produit un effet conclu par la base des règles floues.

Du faite que en robotique mobile on se retrouve généralement confronté à des comportements de plus en plus complexes et simultanés, tel qu'aller en ligne droite, éviter obstacle et suivre la cible ; alors une coordination et un arbitrage flou entre ces comportements est indispensable. Pour résoudre ces situations, plusieurs solutions sont apportées [2],[27].

Coordination compétitive : dans ce cas c'est le comportement le plus prioritaire qui prend la main et peut donc inhiber les autres. Le comportement le plus prioritaire est défini soit de façon statique ou dynamique et suivant la stratégie et l'environnement. Les limitations de cette solution c'est dans l'impossibilité de considérer plusieurs objectifs en même temps.

Coordination coopérative : dans ce cas l'ensemble des comportements intervient et chaque un offre une contribution. L'inconvénient dans ce cas est la possibilité d'une concurrence entre comportements ce qui produit des décisions contradictoires.

Coordination contextuelle : c'est une solution proposée par Saffioti [2],[1],[26] où un ensemble de méta-règles floues représentent le contexte d'activation des comportements et chaque un ayant sont degré d'applicabilité, avec la prise en compte des concurrences entre eux. La fuzzification réalise la fusion des résultats pour obtenir un résultat unique. Cette solution peut s'étendre à des stratégies de plus en plus complexes.

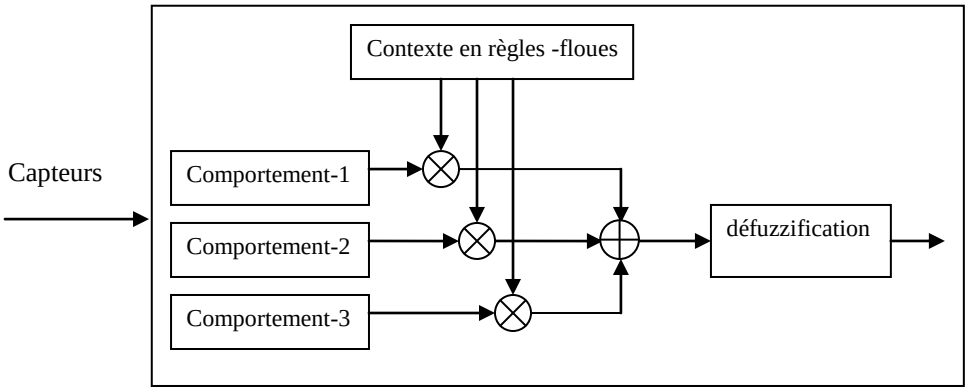


Figure 8.1 Coordination hiérarchique basée sur
le contexte en règles–floues

8.2 Planification d'une trajectoire avec évitement d'obstacles pour un robot mobile basée sur comportements flous

8.2.1 Cinématique du robot mobile

Le robot mobile considéré est type tricycle (voir pour cela 5.18 : *Modèle du robot type tricycle*) dont le modèle de commande est :

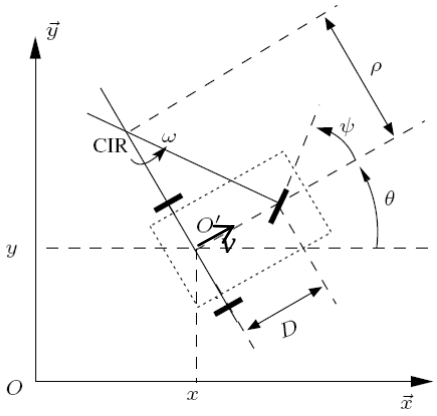


Figure 8.2 Cinématique du robot mobile tricycle

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{D}\tan\psi \\ \dot{\psi} &= \eta \end{aligned} \tag{8.1}$$

Sachant que le vecteur de commande choisie est $u = (v, \eta)^T$.

Alors pour chaque période de commande il faut déterminer les valeurs de la vitesse linéaire v et la variation de l'angle de braquage η pour mener à bien le robot mobile afin d'atteindre la cible tout en évitant les obstacles.

Le robot mobile est équipé de huit capteurs ultrason comme indiqué en figure 6.2.

8.2.2 Architecture du contrôleur à comportements flous

Pour contrôler le mouvement du robot mobile un contrôleur à comportements flous à deux couches type Mamdani est implémenté [12],[3].

Pour la première couche il y'a quatre contrôleurs pour quatre comportements flous responsables de la détection des obstacles et le calcule de la 'possibilité de collision ', cela pour les quatre principales directions Avant, Arrière, Gauche et Droite. Les contrôleurs reçoivent en leurs entrées les distances minimales entre robot mobile et obstacles, mesurées par les différents capteurs ultrason, et retournent en sorties les 'possibilités de collision'.

Les 'possibilités de collision' calculées par la première couche du contrôleur, l'angle entre le robot mobile et la cible et la distance entre le robot mobile et la cible et la variable présence obstacle sont les entrées de la deuxième couche du contrôleur qui donne en sa sortie la vitesse linéaire et l'angle de braquage du robot mobile nécessaires à chaque instant.

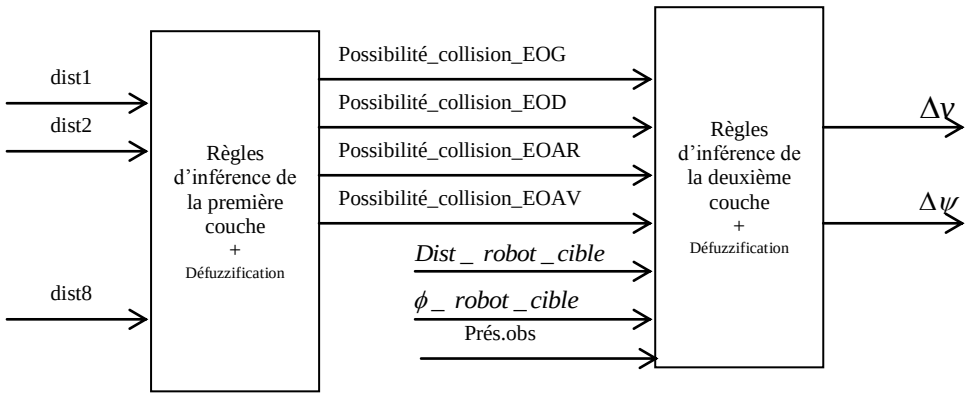


Figure 8.3 Contrôleur flou à deux couches

8.2.3 Comportements flous

Pour réaliser la tâche ' mener le robot vers la cible en évitant les différents obstacles que peut rencontrer dans son parcours', on la divise en plusieurs comportements :

- Eviter obstacle en avant du robot (EOAV) : consiste à éviter les obstacles se trouvant en avant du robot mobile.
- Eviter obstacle à droite du robot (EOD) : consiste à éviter les obstacles se trouvant à droite du robot mobile.
- Eviter obstacle à gauche du robot (EOG) : consiste à éviter les obstacles se trouvant à gauche du robot mobile.

- Eviter obstacle en arrière du robot (EOAR) : consiste à éviter les obstacles se trouvant en arrière du robot mobile.

8.2.4 Fonctions d'appartenances

8.2.4.1 Fonctions d'appartenances des distances (entrées de la première couche)

La fonction d'appartenance concernant la distance entre obstacle et robot est comme suit :

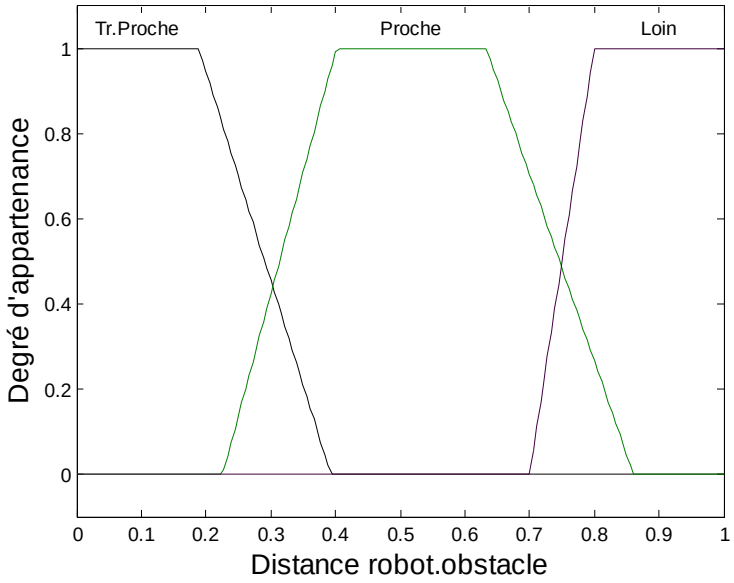


Figure 8.4 Ensembles flous pour la variable d'entrée
Distance robot.obstacle (Dist)

L'ensemble de discours de la variable 'Distance robot.obstacle' est partagé en trois ensembles flous qui sont : Tr.Proche (très proche), Proche et Loin.

8.2.4.2 Fonctions d'appartenances des sorties de la première couche

La sortie de chaque comportement de la première couche comme indiqué ci-dessus est 'la possibilité de collision' dont les ensembles flous et la fonction d'appartenance est comme indiqué en figure ci-dessous.

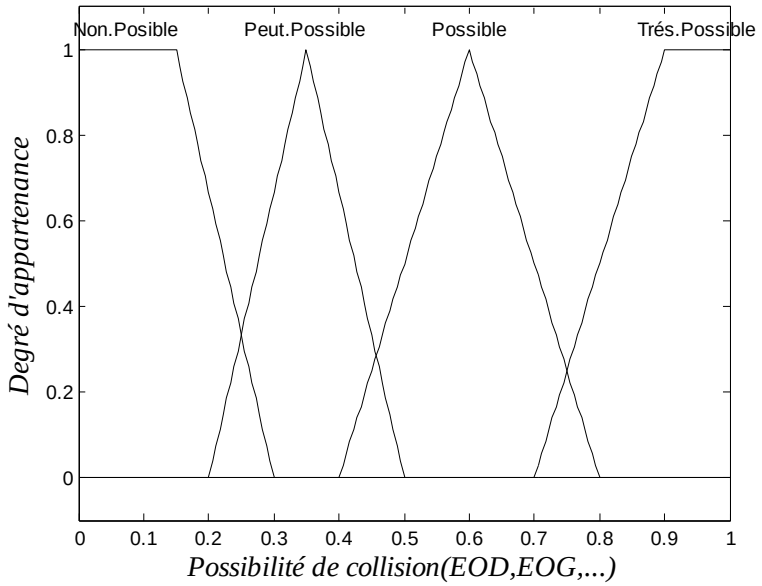


Figure 8.5 Ensembles flous pour la variable de sortie 'Possibilité de collision' pour les différents comportements

8.2.5 Règles d'inférence de la première couche

Chaque comportement est influencé par un ensemble de capteurs qui donnent la distance minimale robot_Obstacle à savoir :

- Comportement EOAV : capteur1 (dist1), capteur2 (dist2) et capteur3 (dist3).
- Comportement EOD : capteur2 (dist2), capteur4 (dist4) et capteur6 (dist6).
- Comportement EOG : capteur3 (dist3), capteur5 (dist5) et capteur7 (dist7).
- Comportement EOAR : capteur8 (dist8).

Base de règles pour le comportement éviter obstacle se trouvant à droite du robot mobile 'EOD ' est le suivant :

dist2	*Tr.proche*			*proche*			*Loin*		
dist4 / dist6	*Tr.proche*	*proche*	*Loin*	*Tr.proche*	*proche*	*Loin*	*Tr.proche*	*proche*	*Loin*
Tr.proche	Très.Possible	Très.Possible	Possible	Possible	Possible	Possible	Possible	Peut.possible	Peut.possible
proche	Très.Possible	Possible	Possible	Possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible	non.possible
Loin	Possible	Possible	Possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible	Non.Possible

Figure 8.6 Possibilité_collision_EOD

Base de règles pour le comportement éviter obstacle se trouvant à gauche du robot mobile 'EOG ' est le suivant :

Dist3	*Tr.proche*			*proche*			*Loin*		
Dist5 / Dist7	*Tr.proche*	*proche*	*Loin*	*Tr.proche*	*proche*	*Loin*	*Tr.proche*	*proche*	*Loin*
Tr.proche	Très.Possible	Très.Possible	Possible	Possible	Possible	Possible	Possible	Peut.possible	Peut.possible
proche	Très.Possible	Possible	Possible	Possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible	non.possible
Loin	Possible	Possible	Possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible	Non.Possible

Figure 8.7 Possibilité_collision_EOG

Base de règles pour le comportement éviter obstacle se trouvant en avant du robot mobile 'EOAV' est le suivant :

Dist1	*Tr.proche*			*proche*			*Loin*		
Dist2 / Dist3	*Tr.proche*	*proche*	*Loin*	*Tr.proche*	*proche*	*Loin*	*Tr.proche*	*proche*	*Loin*
Tr.proche	Très.Possible	Très.Possible	Très.Possible	Possible	Possible	Possible	Possible	Peut.possible	Peut.possible
proche	Très.Possible	Très.Possible	Très.Possible	Possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible
Loin	Très. Possible	Très.Possible	Très.Possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible	Peut.possible	Non.Possible

Figure 8.8 Possibilité_collision_EOAV

Base de règles pour le comportement éviter obstacle se trouvant en arrière du robot mobile 'EOAR ' est le suivant :

Dist8	*Tr.proche*	*proche*	*Loin*
Possibilité de collision	Très.Possible	Peut.Possible	Non.Possible

Figure 8.9 Possibilité_collision_EOAR

8.2.6 Fonctions d'appartenances des entrées de la deuxième couche

Les entrées floues de la deuxième couche sont bien sur les sorties de la première couche et aussi l'angle ϕ et la distance D entre le robot mobile et la cible soit :

- Possibilité_collision_EOAV
- Possibilité_collision_EOAR
- Possibilité_collision_EOG
- Possibilité_collision_EOD
- $\phi\_\ robot\_cible$
- $Dist\_\ robot\_cible$
- Pré.obs

8.2.6.1 Fonction d'appartenance de ϕ_robot_cible

L'angle ϕ_robot_cible est l'angle que fait le robot avec la cible. Cet angle est formé entre l'axe O'x' et l'axe qui joint la cible au centre O', figure 8.2.

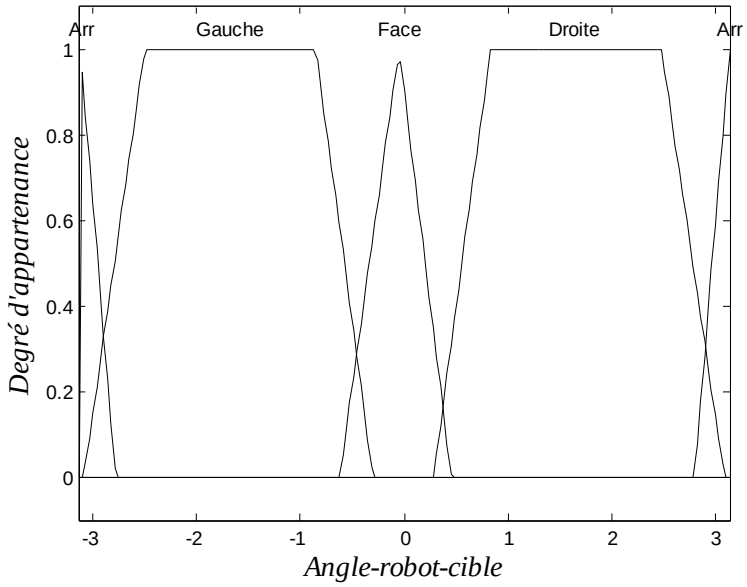


Figure 8.10 Angle entre robot et cible ϕ_robot_cible

8.2.6.2 Fonction d'appartenance de $Dist\_robot\_cible$

$Dist\_robot\_cible$ est la distance qui relie la cible au centre O' du repère porté sur le robot mobile.

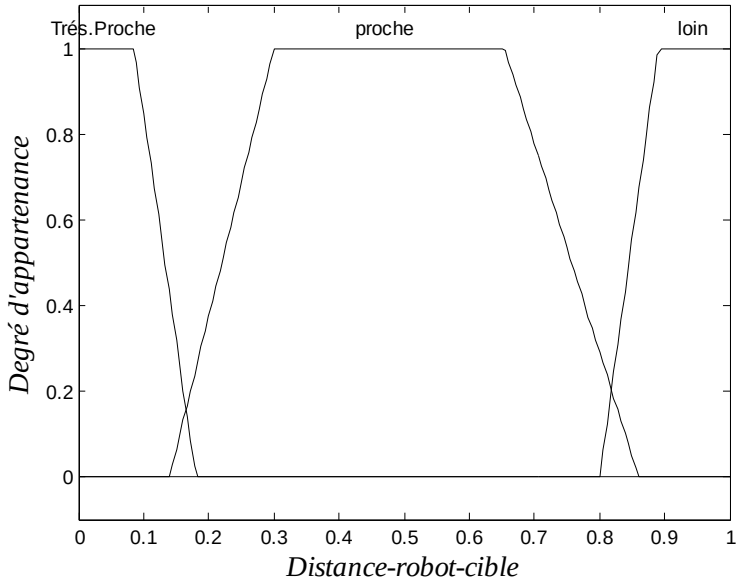


Figure 8.11 Distance entre le robot et la cible

8.2.6.3 Fonction d'appartenance de Pré.obs

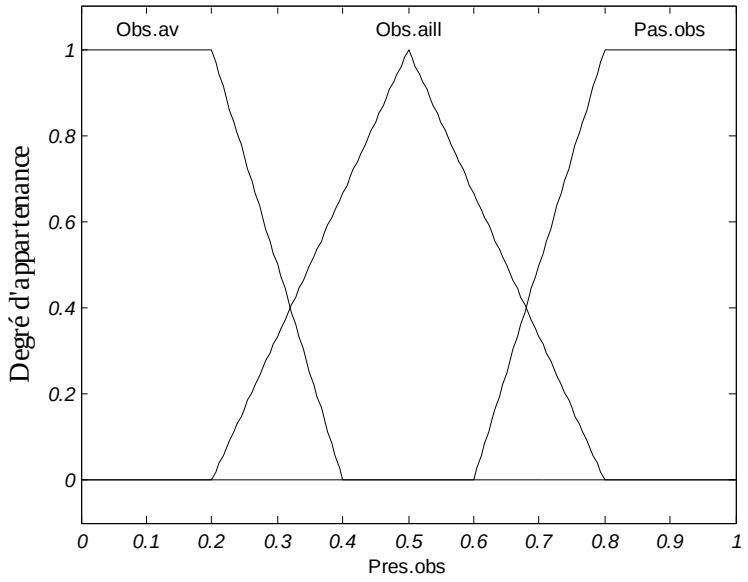


Figure 8.12 Variable Prés_obs

8.2.7 Fonctions d'appartenances des sorties de la deuxième couche

Les sorties de la deuxième couche sont bien sur les sorties de notre contrôleur à comportements flous. Ces sorties sont la variation de l'angle de braquage $\Delta\psi$ et la variation de la vitesse linéaire Δv sur une période de contrôle.

8.2.7.1 Fonctions d'appartenances de Δv

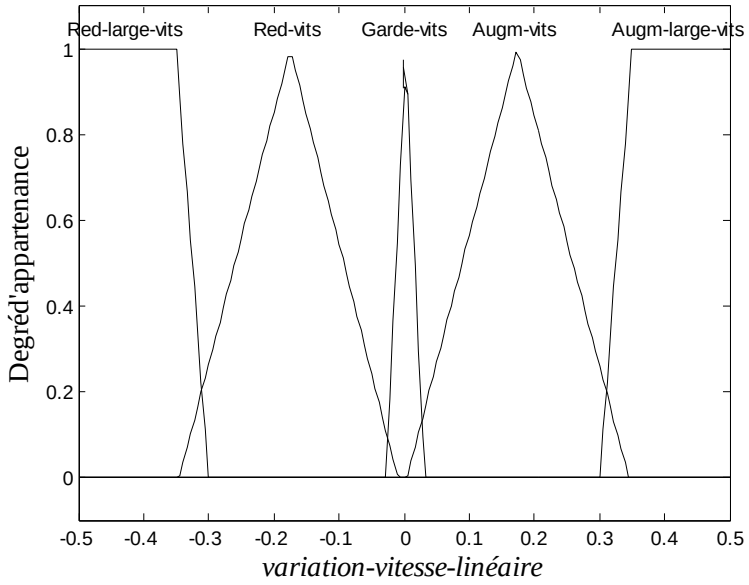


Figure 8.13 Ensembles flous concernant la variation

de la vitesse linéaire Δv

8.2.7.2 Fonctions d'appartenances de $\Delta\psi$

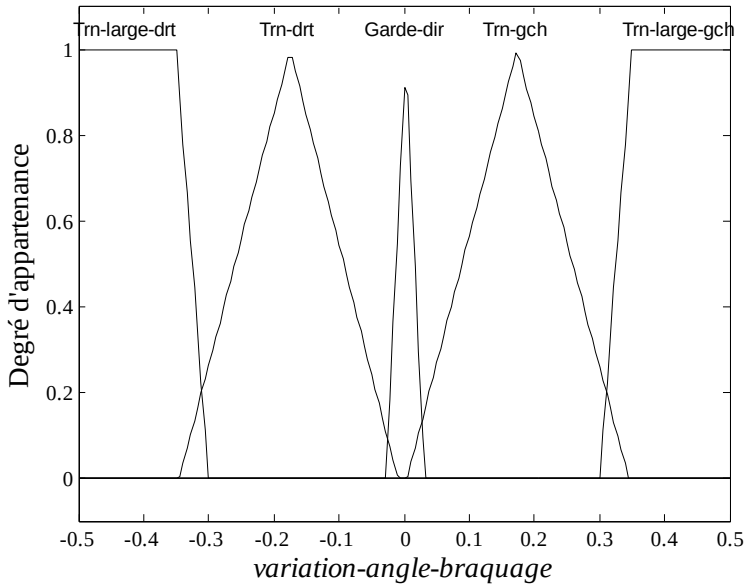


Figure 8.14 Ensembles flous concernant la variation de l'angle de braquage $\Delta\psi$

8.2.8 Règles d'inférence de la deuxième couche

Les sorties du contrôleur comme mentionnées sont Δv et $\Delta\psi$ où :

$\Delta\psi = f(EOAV,\ EOAR,\ EOG,\ EOD\ ,\ \phi\_\ robot\_cible\ )$

$\Delta v = f(Prés\_obs,\ \phi\_\ robot\_cible, Dist\_\ robot\_cible\ )$

Alors on voit que le nombre de règles donnant $\Delta\psi$ est important.

L'entrée ' $Prés\_obs$ ' pour le calcul de Δv résume la présence d'obstacles proches à qui on associe les trois possibilités : 'pas d'obstacle :Pas.obs', 'obstacle proche en avant: Obs.av' et 'obstacle proche d'autres cotées :Obs.aill'.

Règles d'inférences pour la sortie $\Delta\psi$:

$\Delta\psi = f(EOAV,\ EOAR,\ EOG,\ EOD\ ,\ \phi\_\ robot\_cible\ )$

Avec :

T-l-d(trn-large-drt): Tourne largement à droite.

T-l-g(trn-large-gch) : Tourne largement à gauche.

G-d(garde-dir): Garder la même direction.

T-d (Trn-drt): Tourne droite.

T-g(Trn-gch): Tourne gauche.

EOAV \ EOD		N.p				P.p				P				T.p			
		N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p
N.p	N.p	T-l-d	T-l-d	T-l-d	T-l-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	P.p	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	P	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	T.p	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	T-g	T-g	T-g	G-d
P ;p	N.P	T-l-d	T-l-d	T-l-d	T-l-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	P ;p	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	P	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	T.p	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	T-g	T-g	T-g	G-d
P	N.p	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	P.p	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	P	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	T.p	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	T-g	T-g	T-g	G-d
T.p	N.p	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	P.p	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-g	T-g	T-g	G-d
	P	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	G-d	G-d	G-d	G-d	T-g	T-g	G-d	G-d
	T.p	T-d	T-d	T-d	T-d	T-d	T-d	T-d	T-d	G-d	G-d	G-d	G-d	T-g	T-g	G-d	G-d

Figure 8.15 Règles d'inférences de $\Delta\psi$ dans le cas de ϕ_robot_cible = *Droite*

EOAV \ EOG		N.p				P.p				P				T.p			
		N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p
N.p	N.p	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	T.p	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	T-d	T-d	T-d	G-d
P ;p	N.P	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P ;p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	T.p	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	T-d	T-d	T-d	G-d
P	N.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	T.p	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	T-d	T-d	T-d	G-d
T.p	N.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	G-d	G-d	G-d	G-d	T-d	T-d	G-d	G-d
	T.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	G-d	G-d	G-d	G-d	T-d	T-d	G-d	G-d

Figure 8.16 Règles d'inférences de $\Delta\psi$ dans le cas de ϕ_robot_cible = *Gauche*

EOG / EOAV / EOD / EOAR		N.p				P.p				P				T.p			
		N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p
N.p	N.p	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d
	P.p	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d
	P	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d
	T.p	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d	G-d
P ;p	N.P	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P ;p	G-d	G-d	T-g	T-g	G-d	G-d	T-g	T-g	T-d	T-d	G-d	G-d	T-d	T-d	G-d	G-d
	P	G-d	G-d	T-g	T-g	G-d	G-d	T-g	T-g	T-d	T-d	G-d	G-d	T-d	T-d	G-d	G-d
	T.p	G-d	G-d	T-g	T-g	G-d	G-d	G-g	G-g	T-d	T-d	G-d	G-d	T-d	T-d	G-d	G-d
P	N.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	T.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
T.p	N.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	T-d	G-d
	P	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d	T-d	T-d	G-d	G-d
	T.p	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d	T-d	T-d	G-d	G-d

Figure 8.17 Règles d'inférences de $\Delta\psi$ dans le cas de $\phi\_\ robot\_cible$ = *Avant*

EOG / EOAV / EOD / EOAR		N.p				P.p				P				T.p			
		N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p
N.p	N.p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P.p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	T.p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
P ;p	N.P	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P ;p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	T.p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
P	N.p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P.p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	T.p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
T.p	N.p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P.p	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	P	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d
	T.p	T-l-g	T-l-g	T-g	T-g	T-g	T-g	T-g	T-g	T-d	T-d	G-d	G-d	T-d	T-d	G-d	G-d

Figure 8.18 Règles d'inférences de $\Delta\psi$ dans le cas de $\phi\_\ robot\_cible$ = Arrière

Règles d'inférences de Prés_obs :

EOG / EOAV / EOD / EOAR		N.p				P.p				P				T.p			
		N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p	N.p	P.p	P	T.p
N.p	N.p	Pas. obs	Pas. obs	Obs. aill	Obs. aill	Pas. obs	Pas. obs	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill
	P.p	Pas. Obs	Pas. obs	Obs. aill	Obs. aill	Pas. obs	Pas. obs	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill
	P	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill
	T.p	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill
P ;p	N.P	Pas. obs	Pas. obs	Obs. aill	Obs. aill	Pas. obs	Pas. obs	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill
	P ;p	Pas. obs	Pas. obs	Obs. aill	Obs. aill	Pas. obs	Pas. obs	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill
	P	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill
	T.p	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill	Obs. aill
P	N.p	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av
	P.p	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av
	P	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av
	T.p	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av
T.p	N.p	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av
	P.p	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av
	P	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av
	T.p	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av	Obs. av

Figure 8.19 Règles d'inférences de Prés_obs

Règles d'inférences pour la sortie Δv :

D / ϕ / Prés_ob	*Trés.Proche*				*Proche*				*Loin*			
	Arrière	*Gauche*	*Face*	*Droite*	*Arrière*	*Gauche*	*Face*	*Droite*	*Arrière*	*Gauche*	*Face*	*Droite*
Pas.obs	Aug.la.vit	Aug.vit	Red.vit	Aug.vit	Aug.vit	Aug.vit	Red.vit	Aug.vit	Gard.vit	Aug.la.vit	Aug.la.vit	Aug.la.vit
Obs.av	Red.vit	Red.vit	Red.vit	Red.vit	Red.vit	Red.vit	Red.vit	Red.vit	Red.vit	Red.vit	Red.vit	Red.vit
Obs.aill	Red.vit	Red.vit	Red.vit	Red.vit	Gard.vit	Gard.vit	Gard.vit	Gard.vit	Gard.vit	Gard.vit	Aug.la.vit	Gard.vit

Figure 8.20 Règles d'inférences pour la sortie Δv

Avec :

Prés_obs : Présence ou non d'obstacles.

ϕ : $\phi\_\ robot\_cible$

D : $Dist\_\ robot\_cible$

Aug.vit(Augm-vits) :Augmenter vitesse ;

Red.vit(Red-vits) :Réduire vitesse ;

Gard.vit(gard-vits) :Garder la même vitesse ;

Aug.la.vit(augm-larg-vits) :Augmenter largement la vitesse.

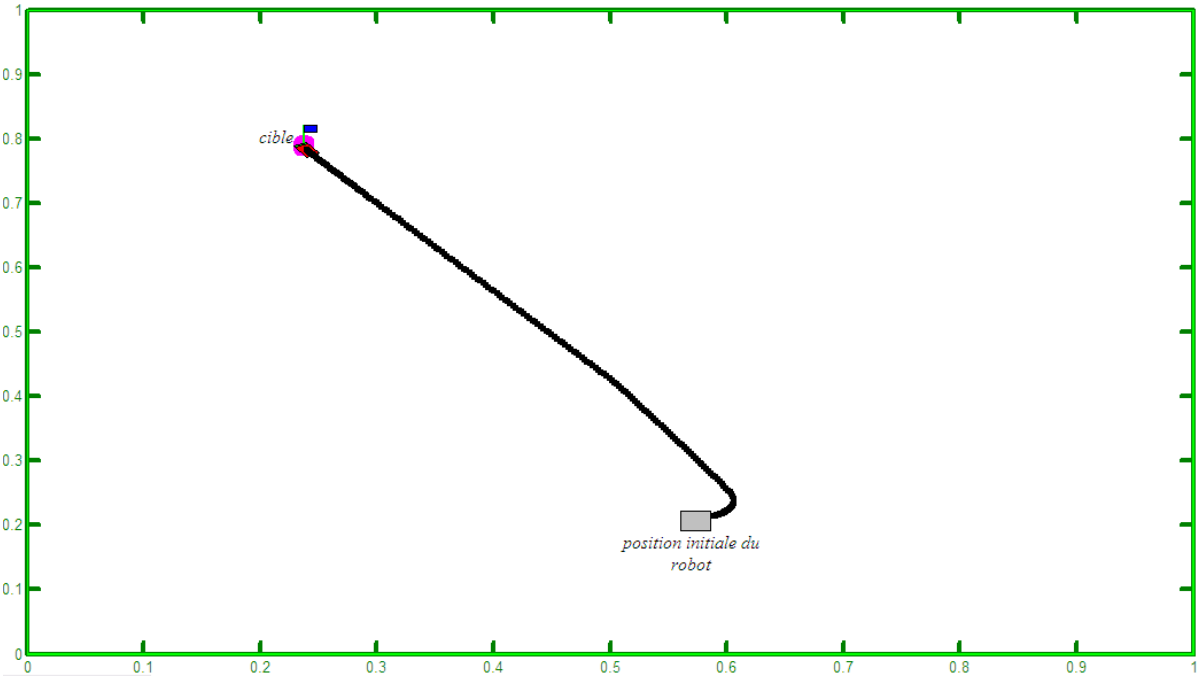
Red.la.vits(Red-la-vits) : Réduire largement la vitesse.

8.2.9 Défuzzification : Toutes les défuzzifications sont réalisées par la méthode du centre de gravité.

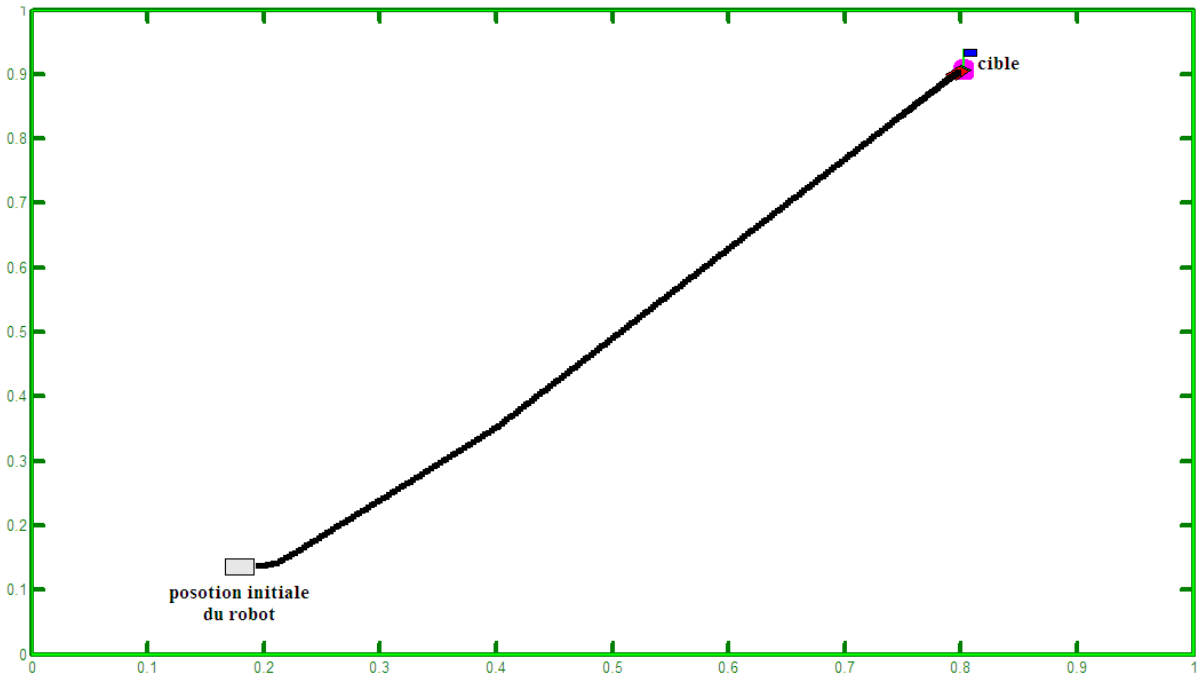
8.3 Simulation du planificateur de trajectoire avec évitement d'obstacle

La simulation porte sur différents cas où le robot mobile tente de joindre la cible toute en évitant des obstacles de différentes formes.

8.3.1 Planification de trajectoires sans obstacles



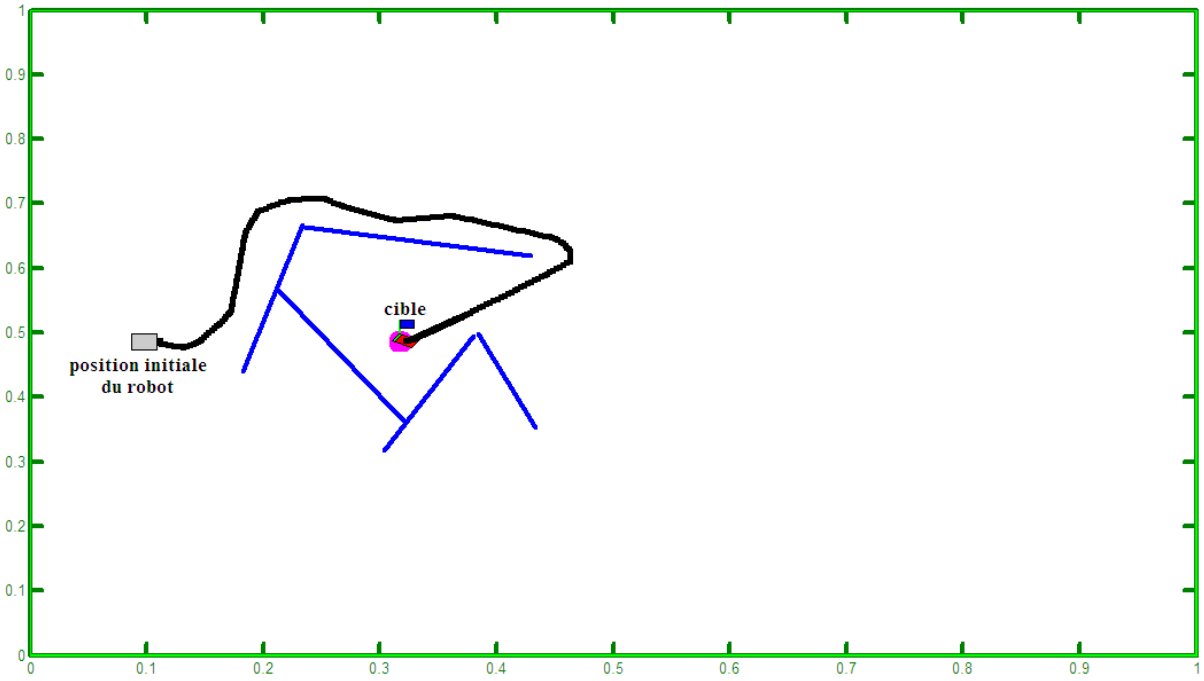
(a)



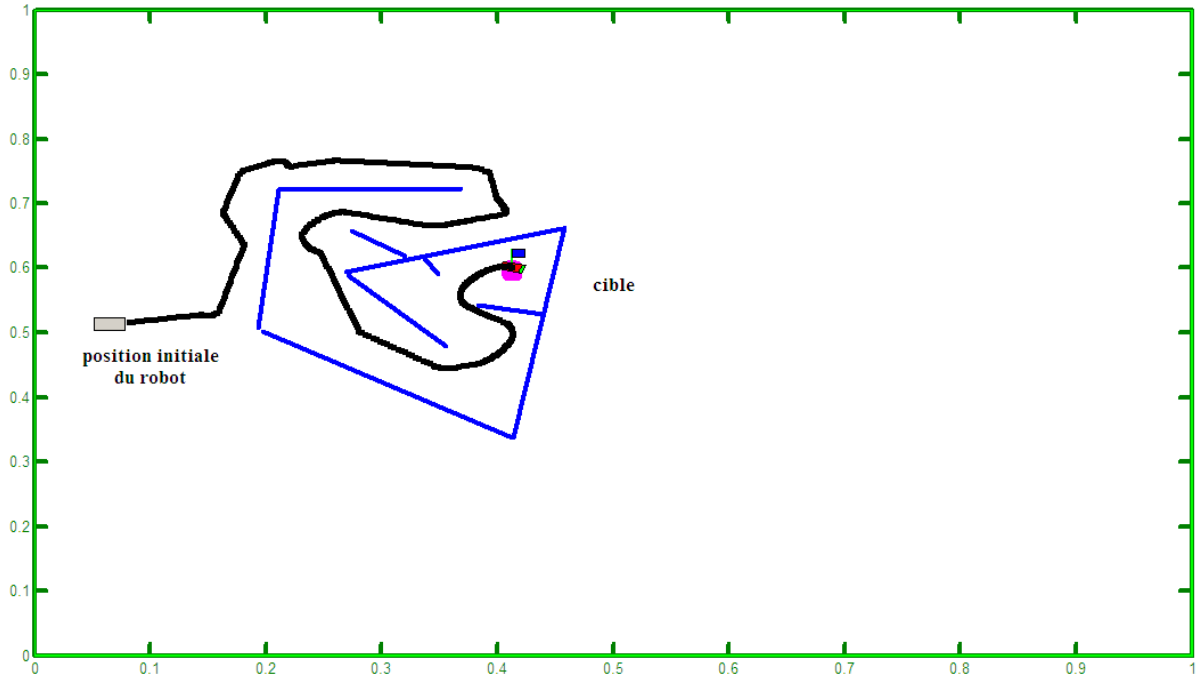
(b)

Figure 8.21.a et b Trajectoires pour environnement sans obstacles

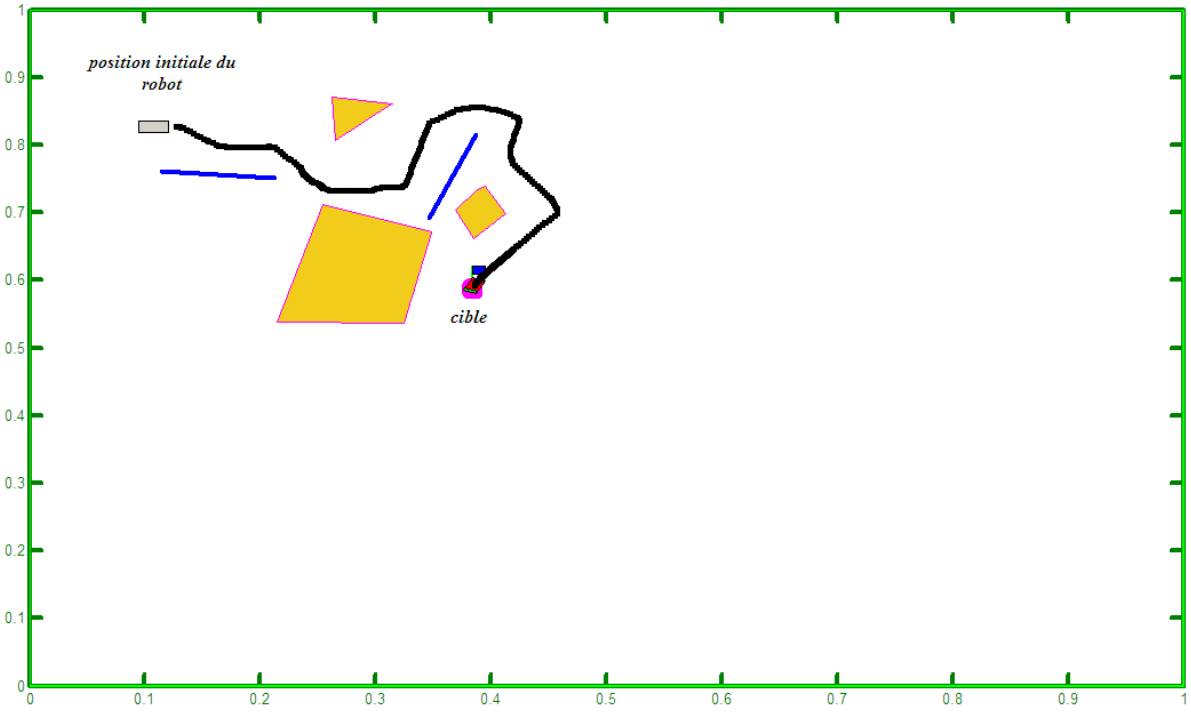
8.3.2 Planification de trajectoires avec évitement d'obstacles



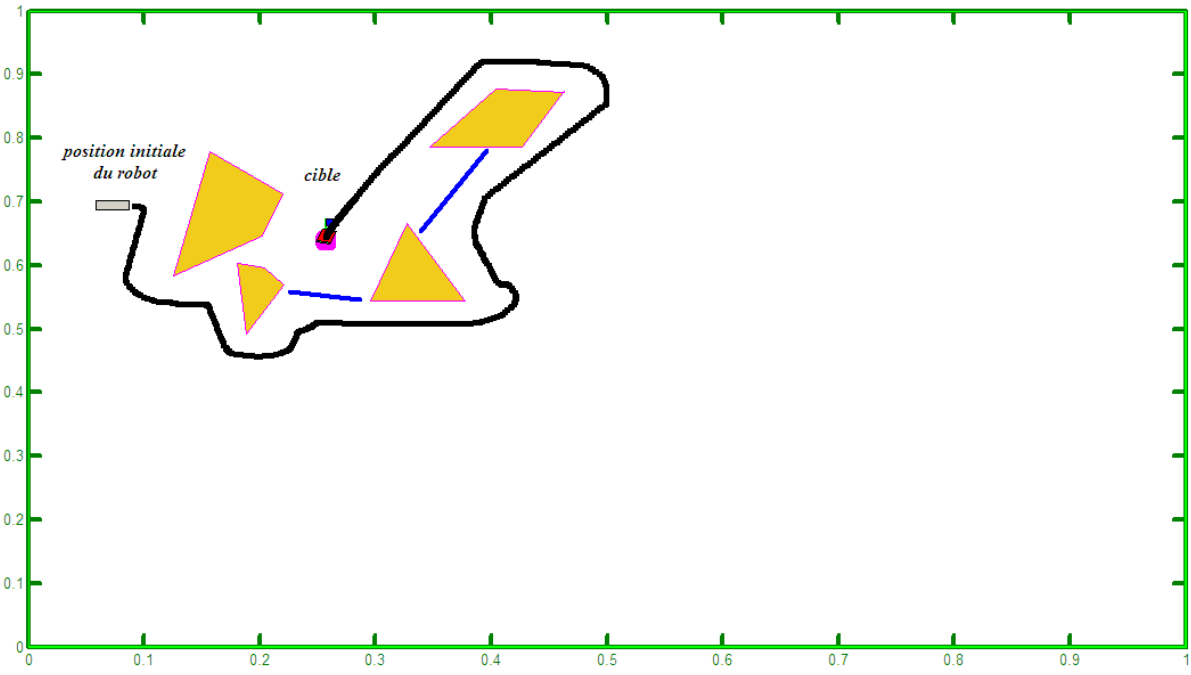
(a)



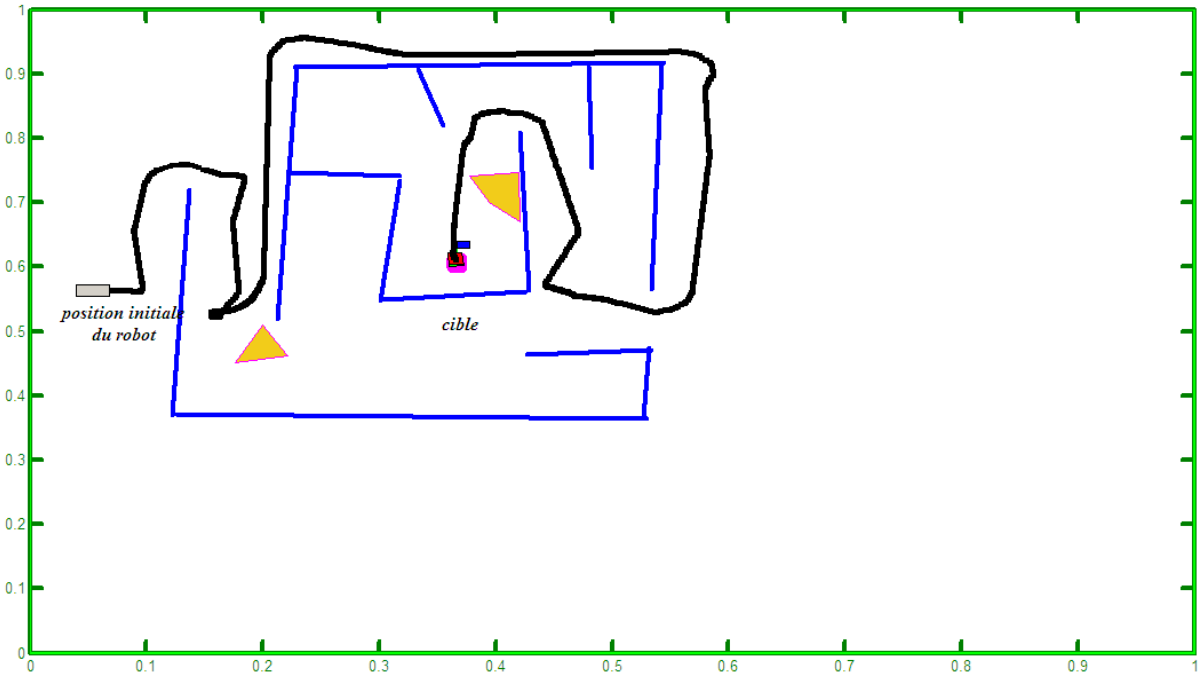
(b)


position initiale du robot
cible

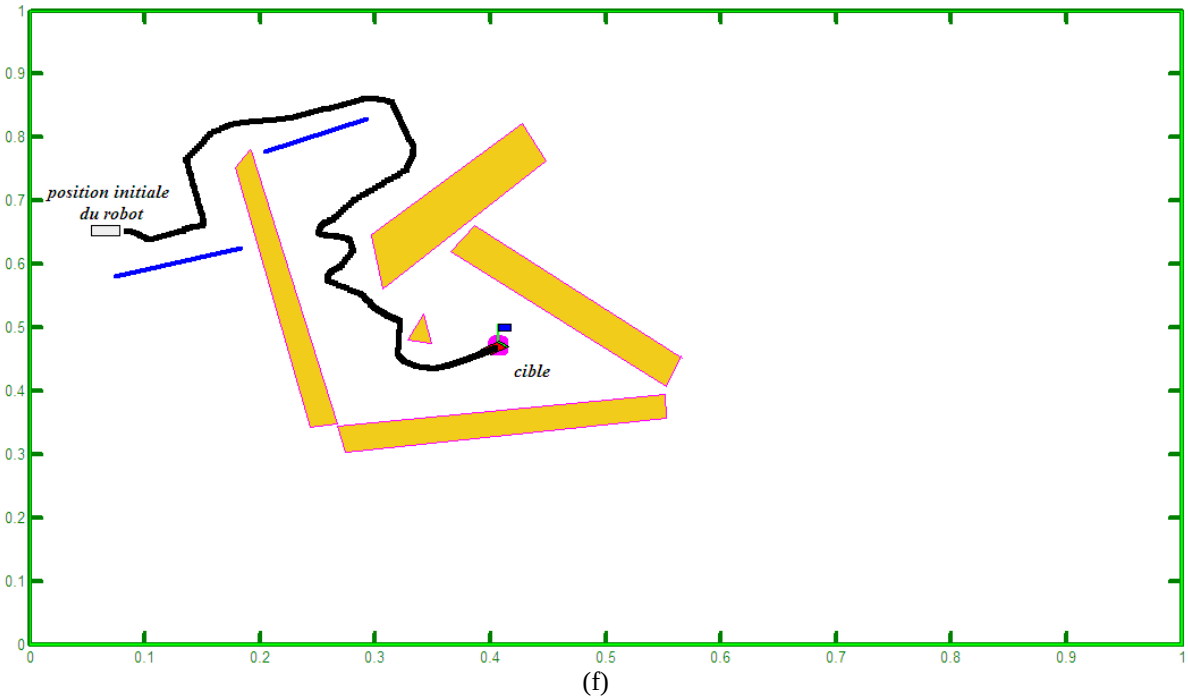
(c)


position initiale du robot
cible

(d)



(e)



(f)

Figure 8.22.a-f Trajectoires pour environnement avec différents obstacles

8.4 Conclusion : il est bien clair qu'un planificateur à comportements flous de deux couches à donner de bons résultats du fait que le robot mobile a pu joindre la cible qu'il soit dans le cas d'environnement libre d'obstacles ou avec obstacles. Cette architecture permet de bien se porter avec des tâches robotiques complexes à plusieurs comportements.

BIBLIOGRAPHIE

[1] Alessandro Saffioti, 'Fuzzy Logic in autonomous Navigation',Springer-physica Verlag, DE,2001.

[2] Alessandro Saffioti,' Autonomous Robot Navigation : a Fuzzy logic approach', thèse de doctorat, 1997/1998 IRIDIA, ,Université libre de Bruxelles.

[3]L. Doitsidis, K. P. Valavanis, N.C. Tsourveloudis, 'Fuzzy Logic Based Autonomous Skid Steering Vehicle Navigation', Proceeding of the 2002 IEEE.

[4] Edward Tunstel and Mo Jamshidi, ' Fuzzy Logic and Behavior Control Strategy for Autonomous Mobile Robot Mapping', IEEE1994.

[5]Wei Li, 'Fuzzy Logic-Based Perception-Action Behavior Control of a Mobile Robot in Uncertain Environnements, IEEE1994.

[6]W.-S. Lin, C.-L Huang and M.-K. Chuang , 'Hierarchical fuzzy control for autonomous navigation of wheeled robots', IEE 2005.

[7]H. Maaref, C. Barret, 'Sensor-based navigation of mobile robot in an indoor environment', Elsevier Science 2002.

[8]Toshio Fukuda, Naoyuki Kubota, 'An Intelligent Robotic System Based on a Fuzzy Approach', Proceeding of the IEEE, vol. 87, No.9, September 1999.

[9]Nikhil D. Kelkar,' A Fuzzy Controller for Three Autonomous Mobile Robot', Master University of Cincinnati, 1997.

[10]Pangiotis G. Zavlangas, Spyros G. Tzafestas, K. Althoefer, 'Fuzzy Obstacle Avoidance for omnidirectional Mobile Robots', ESIT 2000, 14-15 Septembre 2000, Aachen, Germany.

[11]Kim C. Ng, Mohan M. Trivedi,' A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multirobot Convoying', IEEE 1998.

[12]Alessandro Saffiotti Enrique H. Ruspini, Kurt Konolige 'Using Fuzzy logic For Mobile Robot Control', chapter 5, Kluwer Academic 1999.

[13]Rui Araujo, Gonçalo Gouveia, and Nuno santos 'Mobile Robot Localization Using a Fuzzy ART World Model' IEEE 2000.

[14] Jianwei Zhang, Alois Knoll, 'Designing fuzzy controllers by rapid learning' Elsevier 1999.

[15]Ivan N. da Silva, Fernando A C. Gomid, Wagner C. do Amaral, 'Navigation of Mobile Robots Using Fuzzy Logic Controller',IEEE 1998.

[16] Young D. Kwon, Jim M. Wonn and Jin S. Lee,'Control of Mobile Robot by using Evolutionary Fuzzy Controller',IEEE, 1998.

[17] Xianhua Jiang, Yuichi Motai, Xingquan Zhu,'Predictive Fuzzy Logic Controller for Trajectory Tracking', IEEE 2005.

[18] Tzuu-Hseng S. Li,Shih-Jie Chang and Wei tong,'Fuzzy Target Tracking Control Of autonomous Mobile Robot by Using Infrared Sensors', IEEE 2004.

[19] R. A. Brooks, 'A robust layered control system for a mobile robot', IEEE journal of robotics and automation 1986.

[20] H.Eskandar, Poya Salehi and M.H.Sabour, 'Fuzzy Logic Tracking Control for a Three wheel Circular Robot in Unkown Environnement', World Applied science journal IDOSI Publication, 2010.

[21] Olumide Obe, Ioan Dumitrache, 'fuzzy control of autonomous mobile robot', U.P.B. Sci.Bull, Vol.72, Issue.3, 2010.

[22] Khaldoun K.Tahboub, Munafs.N.AL.Din, 'A neuro-fuzzy reasoning system for mobile robot navigation', JJMIE, Vol.3, No1, 2009.

[23] T.C.Kuo,'trajectory control of a robotic manipulator utilizing and adaptive fuzzy slinding mode', World academy of science engineering and technology, 2010.

[24]Zafer Bingül, Oguzhan Karahan,'A fuzzy logic controller tuned with PSO for 2 DOF robot trajectory control', Expert system with applications, 2011.

[25] Joshi.M.M, Zaveri.M.A, 'Neuro-fuzzy based autonomous mobile robot navigation system', Control automation robotics & vision (ICARCV), IEEE, 2010.

[26] Kundu.S, 'A fuzzy approach towards behavioural strategie for navigation of mobile agent', Emerging trends in robotics and communication technologies, IEEE, 2010.

[27] Saroj K.P, Dyal.R.P, Anup.K.P, 'Fuzzy logic techniques for navigation of several mobile robots', Journal of applied soft computing, Vol.9, Issue.1, 2009.

CHAPITRE 9

Planificateur de trajectoire basé sur les réseaux de neurones pour robot mobile

9. Planificateur de trajectoire basé sur les réseaux de neurones pour robot mobile

9.1 Introduction : Le but est de faire suivre au robot mobile tricycle une trajectoire pour aboutir à joindre une cible. Dans le cas où l'identification du modèle cinématique inverse du robot mobile qui est non linéaires paraît difficile, alors on doit faire recours à des approximateurs de fonctions non linéaires. Les réseaux de neurones qui sont de bons approximateurs parcimonieux sont suggérés à représenter ce modèle inverse, [1][2],[6], [7],[8].

9.2 Contexte et méthodologie : Le modèle cinématique du robot tricycle décret par l'équation (8.1) utilisé en simulation est discrétisé de la façon suivante.

$$\begin{aligned} x(k) &= x(k-1) + T.V.\cos(\theta(k-1)) & \Delta x &= x(k) - x(k-1) \\ y(k) &= y(k-1) + T.V.\cos(\theta(k-1)) & \Delta y &= y(k) - y(k-1) \\ \theta(k) &= \theta(k-1) + T.V / D.tg(\alpha(k-1)) & \Delta\theta &= \theta(k) - \theta(k-1) \end{aligned} \qquad (9.1)$$

Alors pour un angle de braquage α et une vitesse linéaire V du robot, il en découle les variations $\Delta\theta$, Δx et Δy comme montrées en figure ci-dessous.

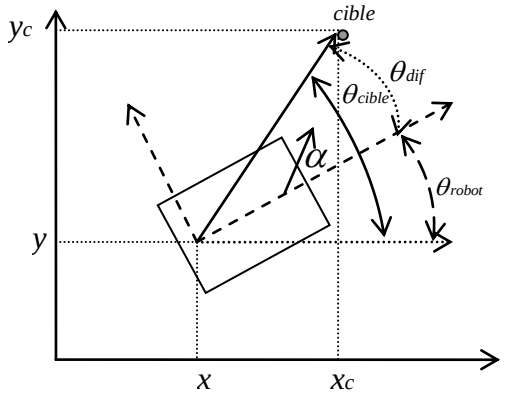


Figure 9.1 Illustration des paramètres de mouvement

Pour commander le robot à atteindre une cible, il est facile d'agir d'une manière variationnelle c-à-d par de petites variations $\Delta\theta$ pour s'orienter vers la cible et aller en ligne droite, et par des petites variations Δr sur la distance r qui sépare le robot de la cible.

$\Delta r = \sqrt{\Delta x^2 + \Delta y^2}$ $\quad r = \sum_i \Delta r_i$

$\theta_{dif} = \theta_{cible} - \theta_{robot}$ $\quad \theta_d = F(\theta_{dif})$ $\quad \theta_d$: Angle désiré

Alors la question c'est : Quel est l'angle de braquage α et la vitesse V permettant d'atteindre notre but c-à-d avoir un $\Delta\theta$ et un Δr sur une période d'asservissement T. Un choix été fait on fixant la vitesse V et cherchant l'angle de braquage α. Le modèle cinématique inverse nous permet de trouver cet angle braquage.

Le schéma de commande pour la génération de trajectoire à utiliser en simulation est le suivant :

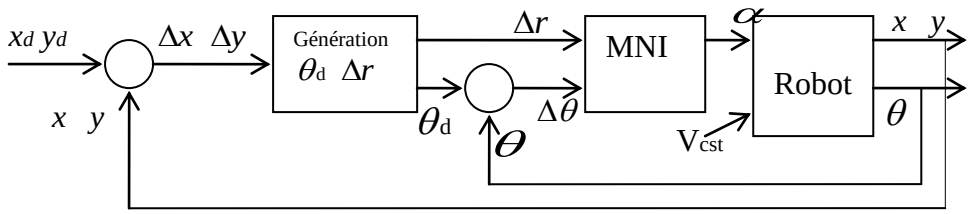


Figure 9.2 Contrôleur neuronal pour la poursuite de trajectoire

9.3 Modèle neuronal du modèle cinématique inverse (MNI) :

Le modèle cinématique inverse du robot mobile est approximé par un réseau de neurones type PMC (MLP) [6]. C'est un réseau multicouche statique, avec une couche d'entrée, des couches cachées et une couche de sortie. Dire trouver le modèle neuronale c'est dire déterminer les valeurs des poids des liaisons entres neurones. Alors un ensemble d'entrées et de sorties est nécessaire pour faire l'apprentissage du réseau par la méthode de rétropropagation de l'erreur de la sortie vers l'entrée.

Les hypothèses de simulations sont :

- La vitesse linéaire du robot est fixée à 0.5m/s
- On utilise le modèle cinématique de (9.1) ci-dessus pour réaliser le couple d'E/S, ayant comme entrée $\alpha = \sin(w.k.T)$ et comme sortie x, y et θ.
- Comme le modèle désiré est un réseau de neurone équivalent au modèle cinématique inverse, alors le couple E/S nécessaire pour l'apprentissage est :

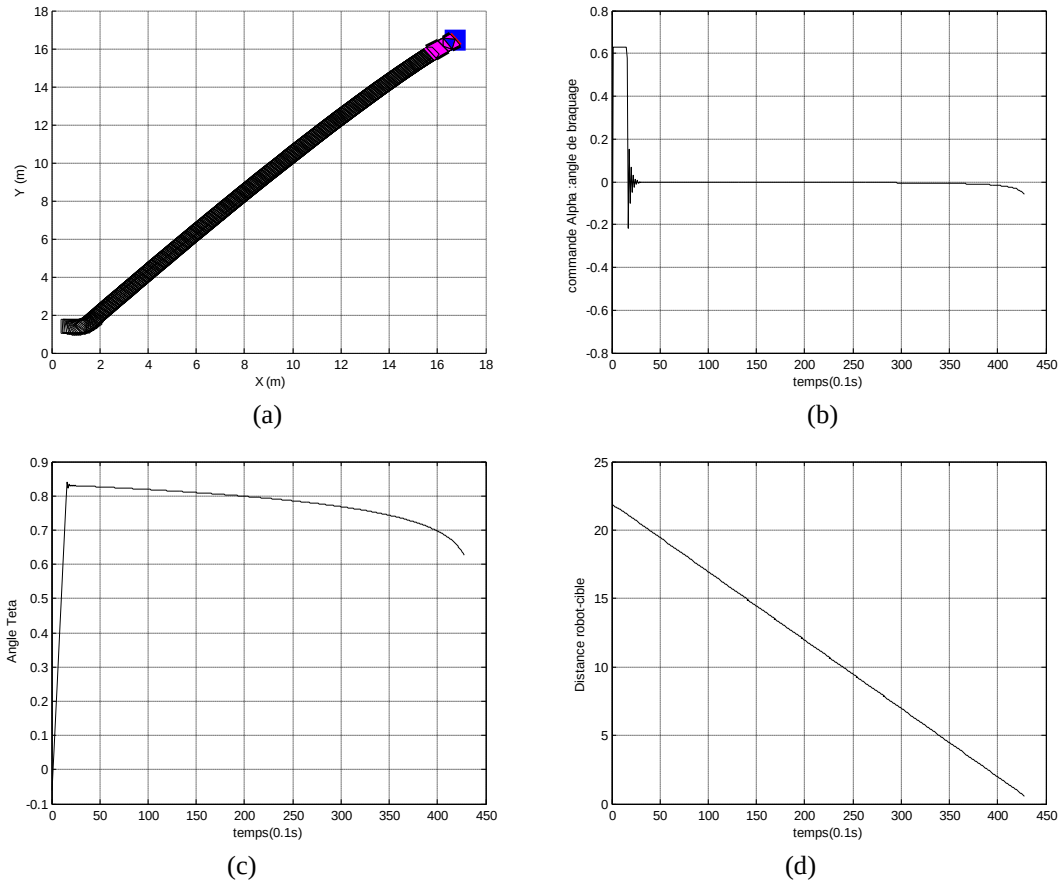
  Entré : (1) $\Delta r = \sqrt{\Delta x^2 + \Delta y^2}$ (2) $\Delta\theta$

  Sortie :(1) Angle de braquage α (2) V=Cst=0.5m/s

9.4 Résultat de simulation : Après apprentissage du réseau de neurones, il est utilisé pour générer la commande nécessaire pour faire suivre au robot une trajectoire dont le but d'atteindre une cible.

9.4.1 Atteindre une cible en ligne droite :

EXEMPLE 1 :



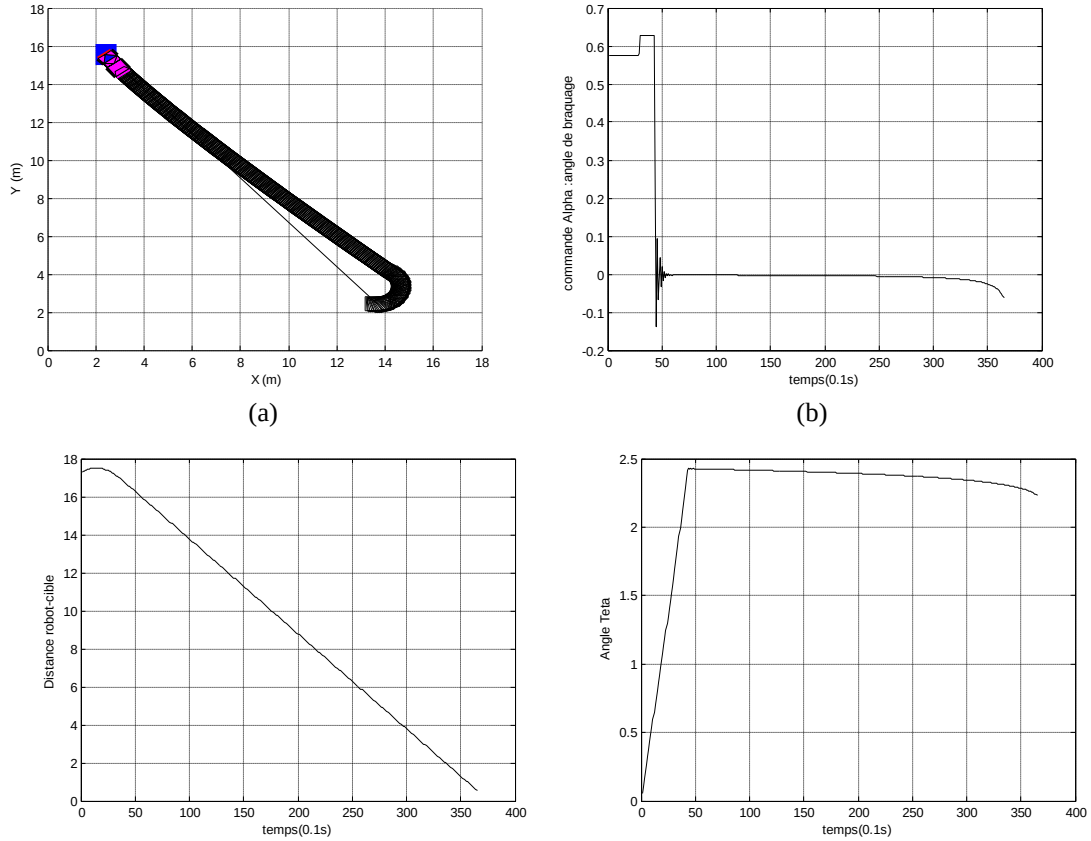
(a) (b)

(c) (d)

Figure 9.3.a-d Atteinte d'une cible en avant du robot en ligne droite

La figure 9.3.a présente le mouvement réalisé par le robot mobile pour atteindre la cible. La trajectoire suivie permet d'optimiser le parcours en ligne droite. La commande α ('Alpha') générée par le contrôleur de la figure 9.2 exprime nettement que le robot doit être redirigé vers la cible, puis garder cette direction jusqu'à ce que la cible soit atteinte. La figure 9.3.d montre la distance robot cible qui diminue au fur et à mesure.

EXEMPLE 2 :



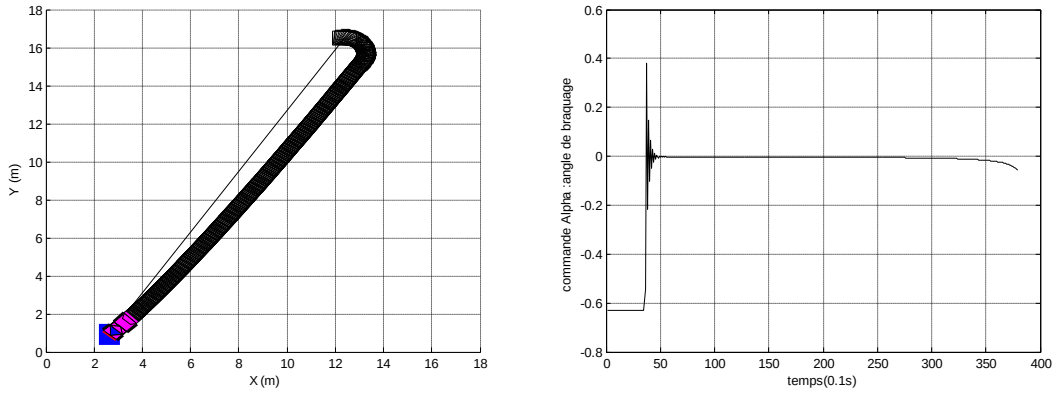
(a) (b)

(c) (d)

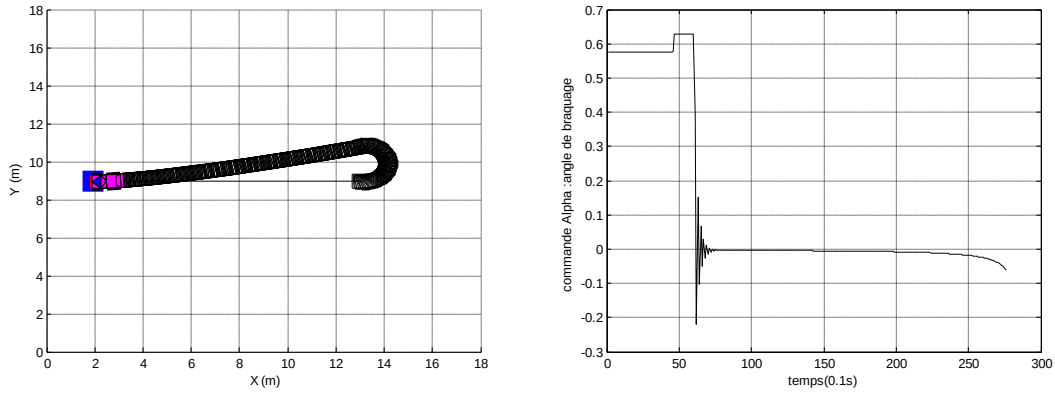
Figure 9.4.a-d Braquage vers la cible se trouvant en arrière du robot

Les figures 9.4.a-d ci-dessus, montrent que le contrôleur neuronal a permet de générer la commande α nécessaire au braquage du robot vers la cible malgré que la cible se trouve derrière le robot.

EXEMPLE 3 :



(a) (b)



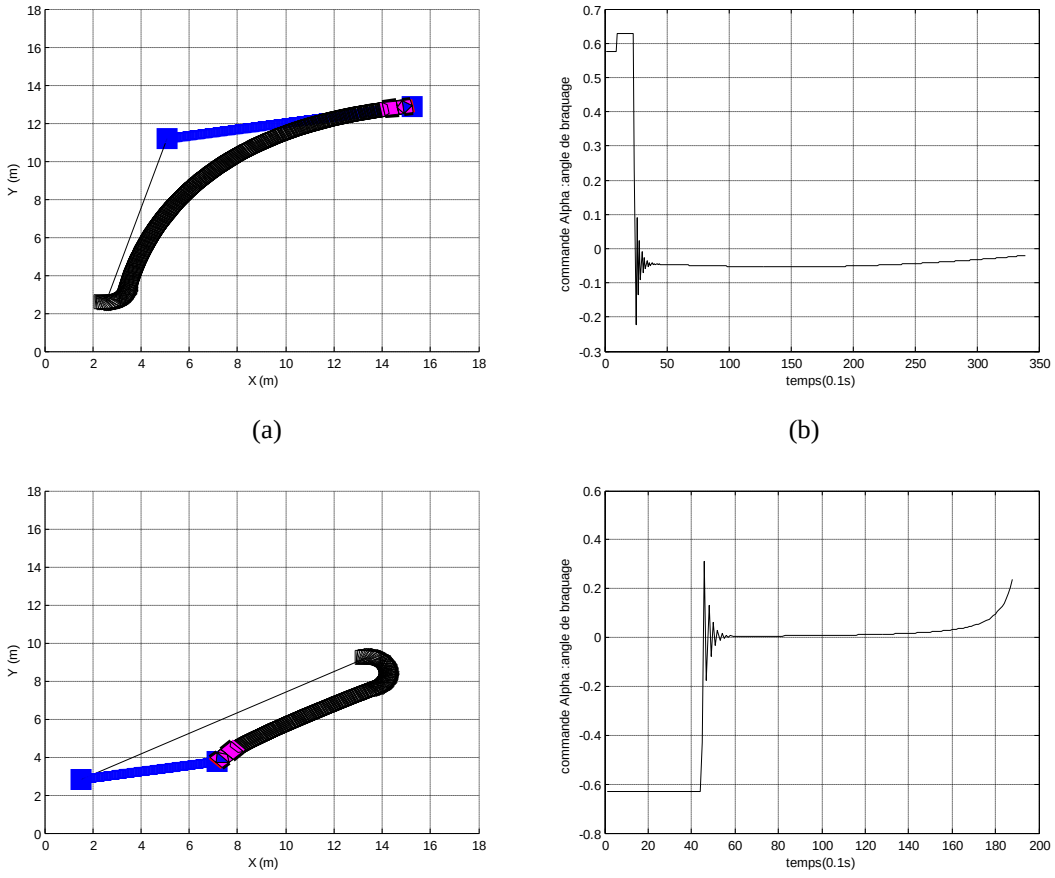
(c) (d)

Figure 9.5.a-d Génération de la commande suivant le cas

Les figures 9.5.a-d prouvent que le contrôleur basé sur le modèle neuronal inverse (MNI) génère la commande adéquate pour toute situation de la cible par rapport au robot.

9.4.2 Poursuite d'une cible en mouvement :

EXEMPLE 1 :



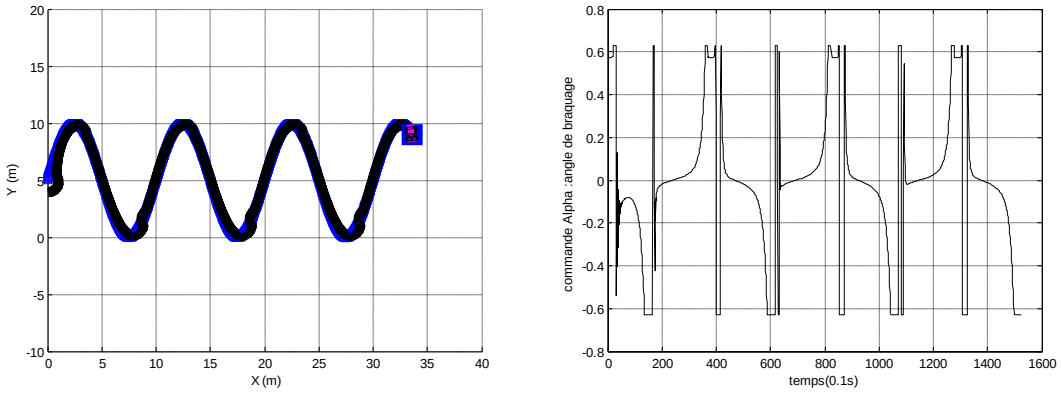
(a) (b)

(c) (d)

Figure 9.6.a-d Poursuite d'une cible en mouvement

Comme montré par la figure 9.6, le contrôleur neuronal permet de suivre une cible mouvante lorsque la vitesse de la cible est plus faible que celle du robot.

EXEMPLE 2 :



(a) (b)

Figure 9.7.a-b Poursuite d'une cible qui se déplace selon une trajectoire sinusoïdale

Le contrôleur neuronal génère la commande nécessaire pour assurer la poursuite de la cible qui se déplace selon une trajectoire sinusoïdale.

9.5. Conclusion : Le contrôleur basé sur un réseau de neurones comme modèle cinématique inverse du robot mobile a montré des capacités intéressantes pour la poursuite de trajectoires de différentes formes. Il est évident que le réseau de neurone MNI à bien approximer le modèle du robot.

BIBLIOGRAPHIE

[1]Catarina Silva Æ Bernardete Ribeiro 'Navigating mobile robots with a modular neural architecture',Neural Comput & Applic (2003) 12: 200–211.

[2]Janglová. D,' Neural Networks in Mobile Robot Motion'., Inernational Journal of Advanced Robotic Systems,Volume 1 Number 1 pp. 15-22 (2004), ISSN 1729-8806

[3]Guilherme A. Barreto, Aluizio F. R. Araújo, Christof Dücker, and Helge Ritter , 'A Distributed Robotic Control System Based on aTemporal Self-Organizing Neural Network', IEEE vol. 32, no. 4, Novembre 2002.

[4]Young H. Kim and Frank L. Lewis ,'Optimal Design of CMAC Neural-Network Controllerfor Robot Manipulators', IEEE no.1, February 2000.

[5]H. Daniel Patiño, Ricardo Carelli and Benjamín R. Kuchen, 'Neural Networks for Advanced Control of Robot Manipulators', IEEEvol. 13, no. 2, March 2002.

[6] Abderrazek Chatti, Imen Ayari, Piere Born, Mohamed Benrejeb ‘On the use of neural techniques for path Following control of car-like mobile robot’, Studies informatics and control, Vol.14,No.4 2005.

[7] Low Kian Hsion, ‘Intégrated Robot Planing and Control with Extended Kohonen Maps’, Phd thesis, National University of Singapore, 2002.

[8] Dierks.T, Jagamathan.S, ‘Neural network output feedback control of robot formation’, IEEE systemes, Man and cybernics society, 2009.

[9] Vesnac, Ilija Nikolic,’ Control of industriel robot using neural network compensator’, Theoret. APPL.Mech, Vol.32, No.02, Belgrad, 2005.

[10] Slike Stegories;Marc Timme, Florention Wörgöther & Poramate Manoopong, ‘ self-Organized adaptation of simple neural circuit enable complexes robot behaviour’, nature physics, 2010.

[11] Toshiyuki Kondo, ’Evolutionary design and behavior analysis of neuromodulatore neural networks for mobile robots control’, Science direct, Applied soft computing, Vol.7, issue.1, 2007.

[12] Masonari Sato, Atushi Kanda, Karua ishii, ‘Performance evaluation of neural network controller system for a wheel type mobile robot’, ScienceDirect, Elsevier, 2007.

Conclusion Générale

L'intersection des recherches entre la commande et la robotique a produit un domaine vraiment multidisciplinaire. Ce travail nous a conduit à faire le tour de plusieurs axes de recherches. En robotique nous avons exposé la robotique de manipulation et la robotique mobile. En commande nous avons discuté et simulé une variété de commandes classiques et d'autres intelligentes.

Au cours de cette thèse nous avons débuté par une récapitulation sur le robot manipulateur, ses différentes architectures, sa modélisation cinématique, géométrique et dynamique. Nous avons aussi montré par simulation les limites des commandes classiques (PID) lorsqu'on exige des vitesses importantes et des précisions sévères, parce que la dynamique du robot entre fortement en action. La simulation de la commande dynamique (couple-calculé) a montré une amélioration mais qui reste insuffisante si les paramètres du modèle sont variants ou male identifiés.

A cause d'insuffisance des commandes classiques on a voulu voir avec les réseaux de neurones. Une étude suffisamment détaillée a montré les performances des réseaux de neurones. Ils peuvent modéliser des systèmes non linéaires, sont robustes et possèdent un critère d'apprentissage et d'approximation parcimonieuse de systèmes non linéaires. Les variétés des réseaux de neurones permettent de répondre au caractère pluridisciplinaire de la robotique.

Pour le robot manipulateur un ensemble de schémas à base de réseaux de neurones est étudié pour apporter des solutions. Parmi ces schémas, on trouve un réseau de neurones comme compensateur axillaire du contrôleur 'couple-calculé', et un réseau de neurones comme modèle inverse du système globale composé du contrôleur et du robot. Un autre schéma consiste au contrôle par modèle inverse du robot, où l'apprentissage est en ligne, et avec l'aide d'un contrôleur classique PD assurant la stabilité au cours de l'apprentissage.

Un autre type de robot est étudié, c'est le robot mobile. Une étude théorique générale est réalisée, ce qui a permis de mieux le connaître. Une investigation est faite sur les différents moyens de locomotions (roues, chenilles,…etc.), sur la modélisation cinématique, et sur les stratégies globales et locales de planification de trajectoires.

La simulation d'une méthode locale, dite méthode de contraintes, pour la génération de trajectoire avec évitement d'obstacles a présenté de bons résultats, où le robot mobile se déplace en évitant des obstacles et en traversant des labyrinthes pour joindre une cible qui peut être mobile. Mais cette méthode locale souffre de minima locaux qui figent le robot, ce qui a exigé de trouver des solutions heuristiques pour faire face.
Un schéma de contrôle pour la génération de trajectoire pour robot mobile basé réseaux de neurones est proposé. C'est la commande par le modèle neuronal inverse (MNI) qui est investie dont le but d'atteindre une cible en choisissant une trajectoire prédéfinie. Les résultas de simulations le prouvent pour différents type de parcoures : droite, sinusoïde,…, etc.
Alors trouver une méthode plus intelligente donnant des résultats meilleurs et sur tous pour des mouvements plus complexes nous a encouragé de voir avec la logique floue.

Une étude théorique sur la logique floue nous a fait preuve de son efficacité lorsque le système est non linéaire et bruité. Un tour sur le contrôle flou nous confirme la possibilité d'utiliser un contrôleur flou pour le control des articulations d'un bras manipulateurs et la planification de trajectoire avec évitements d'obstacles pour robot mobile.
Un contrôleur flou est conçu pour la planification de trajectoire d'un robot mobile tricycle. Le robot à huit capteurs ultrason, navigue dans un environnement contraint non connu d'avance. La tâche que le robot mobile doit réaliser, consiste en l'évitement d'obstacles de différentes formes, détectés par ses huit capteurs, toute en se dirigeant vers la cible. On ayant jugé de la complexité de la tâche pour un contrôleur flou simple on a fait recours aux multicomportements. Vu le nombre de comportements donc de règles on a opté pour un contrôleur flou multicomportements et multicouches cascadées. Pour éviter toute possibilité de concurrence et antagonisme entre comportements, la fusion des décisions est faite selon une stratégie réactive globale. Les résultats de simulation montrent le robot mobile rattrape une cible même mobile tout en évitant différentes forme d'obstacles.

On peut dire que le développement apporté à la vie contemporaine a permis au robot manipulateur et au robot mobile de justifier leurs places. Les succès ont incité les chercheurs à poursuivre leurs travaux sur tout en robotique mobile vu les horizons qu'elle ouvre. D'autres travaux peuvent être entamés combinant la logique floue et les réseaux de neurones et les algorithmes génétiques.

BIBLIOGRAPHIE

CHAPITRE 1

LIVRES :

[1] John J.Craig, 'Introduction to Robotics Mechanics and Control', Addison Wesley Publishing company, 1986.

[2] Etienne Dombre et Wisam Khalil, 'Modélisation et Commande des robots', Hermès, 1988.

[3] Philip Coiffet 'la Robotique : Principes et applications', Hermès ,1986.

[4] William A. Wolovich 'robotics : Basic Analyse and Design', Holt. Rinehart and Winston, 1986.

[5]J.P. Lallemand et S.Zeghloul,'Robotique ,Aspects fondamentaux: modélisation mécanique, CAO robotique, commande ', masson, 1994.

[6] John J.Craig, 'Introduction to Robotics Mechanics and Control: second edition', Addison Wesley Publishing company, 1989.

ARTICLES:

[7] Bestaoui.Y , ' Decentralised PD and PID robotic regulator', IEE proceedings, vol 136, July 1989.

[8] Hirohiko Arai and Susumu Tachi,'Position control system of two degree of freedom manipulator with a passive joint', IEEe 1991.

[9] Krzysztof Tchon and Robert Muszynski, 'Singular inverse kinematic problem for robotic manipulators ', IEEE 1998.

[10] Milos R. Popovie and Andrew A. Goldenberg,' Modeling of friction using spectral analysis', IEEE1998.

[11] Guangjun Lieu, Karl iagnemme, Steven Dubowsky, Guillaum Morel'A base force/torque sensor approach to robot manipulator inertial parameter estimation',IEEE ,1998.

[12] Patrica conti,'contribution à la commande dynamique adaptative des robots manipulateurs ', thèse de doctorat, université Paul Sabatier, Toulouse, 1987.

[13] Alberto Isaguirre, Richard P.Paul, 'Computation of the inertial and gravitational coefficients of the dynamics equation for robot manipulator with load', proceeding of the American control conference vol2, 1985.

THESES :

[14] Agnès Aubin épouse Pracht ,'Modélisation, identification, et commande du bras TAM', thèse à l'Institut nationale polytechnique de Grenoble,1991.

[15] M. Onder Efe, Okyay Kaynak, Xinghuo Yu, 'Sliding mode control of a three degrees of freedom anthropoid robot by driving the controller parameters to equivalent regime',ASME,2000.

[16] Zvi Shiller, Hai Change, 'Trajectory preshaping for high-speed articulated systems', ASME, journal of dynamic systems, measurement, and control, vol.117, 1995.

[17] H. G. SAGE, M. F. DE Mathelin and E. Ostertag,' Robust control of robot manipulators: a survey', Int. Nat. J. Control, vol. 72, No. 16, 1999.

[18] T. C. Steve Hsia, Ty A. Lasky, and Zhengyu Guo,' Robust Independent Joint controller design for industrial robot manipulators', IEEE Transactions on industrial electronics, vol. 38, No. 1, 1991.

[19] D. Benmerzouk, N. Ghouali, 'Analyse de la commande dynamique des robot a axes rigides' Technologies avancées No.12, janvier 2000.

[20] H. Seraji 'An approach to multivariable control of manipulators', ASME, journal of dynamic systems, measurement, and control, vol.109, June 1987.

CHAPITRE 2

[1] Anil.k Jain, jianchang Mao and K.M. Mohiudin, 'Artificial Neural Networks: A Tutorial', IEEE, 1996.

[2] Jean-François Jodouin 'les réseaux neuromimétriques', Hermes, 1994.

[3] Marc Parizeau 'Réseaux de neurones', université Laval, 2004.

[4] Claude Touzet, 'cours : Les réseaux de neurones artificiels', juillet 1992.

[5] I.Rivals, L.Personnaz, G.Dreyfus et J.L. Ploix, 'Modélisation, classification et commande par réseaux de neurones :Principes fondamentaux, méthodologie de conception et illustrations industrielles', Lavoisier Technique et documentation, Paris, 1996.

[6] Marc Lucea 'Modélisation dynamique par réseaux de neurones et machines vecteurs supports : contribution à la maîtrise des émissions polluantes de véhicules automobiles', thèse de doctorat, de l'université Paris 6, France, 2006.

[7] G.Dreyfus, J.M.Martinez, M.Samuelides, M.B.Gordon, F.Badran, S.Thinria, L.Hérault, 'Réseaux de neurones : méthodologie et applications', Eyrolles, 2002.

CHAPITRE 3

[1] G.Dreyfus, J.M.Martinez , M.Samuelides, M.B.Gordon, F.Badran, S.Thinria, L.Hérault, 'Réseaux de neurones : méthodologie et applications',Eyrolles 2002.

[2] Fabien moutarde,'cours : Introduction aux réseaux de neurones', Ecole des mines de paris, avril 2007.

[3] Claude Touzert, Cesar-Ornl,'L'apprentissage par renforcement', Masson 1999.

CHAPITRE 4

[1] Seul Jung and T.C. Hsia,'A neural neutwork control technique for robot manipulators', NITTA corporation of japan, 1996.

[2] H. Daniel Patino, Ricardo Carelli and Bejamin R. kuchen,' neural networks for advanced control of robot manipulators', IEEE transaction of neural networks, vol. 13, no. 2, march 2002.

[3] Young h; kim and Frank l. Lewis,' optimal design of CMAC neural-network controller for robot manipulators',IEEE ,Transaction on systems, Man, and cybernitics, vol. 30, No.1 February 2000.

[4] Tomochica Ozaki, Tatsuya Suzuki, Takeshi Furuhachi, Shigeru Okuma, and Yoshiki Uchikawa,' Trajectory control of robotic manipulators using neural networks', IEEE, transaction on industrial electronics, vol;38, No.3, June 1991.

[5] Yutaka Maeda, and Rui J. P. Figueiredo,' Learing rules for neuro-controller via silmultaneous perturbation',IEEE, transaction on neural networks, vol. 8, No.5, septembre 1997.

[6] David A. Handelman, Stephen H. Lan, and Jack J. Gelfand,'Integrating neural networks and knowledge-based systems for intelligent robotic control', IEEE, 1990.

[7] Charles W. Anderson ,'Learning to control an inverted pendulum using neural networks',American control conference ,Atlanta,Georgia, April 1989.

[8] Vicente Ruiz de angulo and Carme Torras, ' Self-Calibration of space robot', IEEE Transaction on neural networks , vol..8. No.4, Julie 1997.

CHAPITRE 5

[1] R. A. Brooks, 'A robust layered control system for a mobile robot', IEEE journal of robotics and automation 1986.

[2] Arkin R. C. , 'Motor schema based navigation for a mobile robot', In Proc. Of the IEEE INT. Conf on Robotics and automation, Volume 1, page 264 271, San Fransisco, CA(US), 1987.

[3] Arkin R. C.,' Dynamic replanning for mobile robot based on internal sensing', In Proc. Of the IEEE int. Conf. On Robotics and Automation, volume3, page1416-1421, Scottsdale, AZ(US),1989.

[4] R. P. Bonasso, D. Kortenkamp, D. P. Miller, and M. Slack, 'Experiences with an architecture for intelligent, reactive agents', Lecture Notes in Computer Science, 1037:187 202,1996.

[5] R. Alami, R. chatila, S. fleury, M. Ghallab, and F. Ingrand,' An architecture for autonomy', Int. Journal of Robotics Research,17(4):315 337, 1998.

[6] Patrick Reignier, 'pilotage réactif d'un robot mobile : Etude de lien entre la perception et la reaction', thèse de Doctorat, Institut National polytechnique de grenoble,1994.

[7] D.Fox W.Burgard and S.thrun 'the Dynamic Window Approach to collision Avoidance'. IEEE Robotics,vol,18,no 10 pages 1001-1024,2004.

[8] S.Quinlan, O.Khatib 'Elastic Bands : connecting path planning and control', international conference on Robotics and Automation, Atlanta, GA,USA,mai,1993,IEEE.

CHAPITRE 6

[1] Ching-Long.S and Jane-Yu.L,'Computing the Minimum Directed Distances Between Convex Polyhedra', Journal of information on science and engineering vol15, pp353-373, 1999.

[2] Ellepola.R and Kovesi.P, ' mobile robot navigation using recursive motion control', IEEE ,pp168-174, 1997.

[3] Elmer.G.G, W.Johnson.D and Keerthi.S.S ,'A fast procedure for computing the distance between complexes objects in three-dimensional space', IEEE, journal of robotics and automation ,vol.4, no.02,pp193-203, April,1988.

[4] Faverjon.B, 'Hierchical object models for efficientanti-collision algorithms', IEEE, pp333- 340 , 1989.

[5] Faverjon.B and Tournassoud.P, 1987, 'A local based approache for path planning of

manipulators with a high number of degrees of freedom ',IEEE , 1987.
[6] Paulo.C and Urbano.N, April,'Path-following control of mobile robot in presence of uncertainties', IEEE, Transactions on robotics, vol21 .no2,pp252-261,2005.
[7] Ramirez.G and Zeghloul.S, ' A New local path planner for nonholonomic mobile robot navigation in cluttered environments', IEEE, ICRA, san Francisco pp2058-2063, April 2000.
[8] Shirong.L, Simon.X.Y and Huidi.Z, 'Adaptive neurons Based control system design for mobile robot', proceedings of 2004 IEEE/RSJ, ICIRS, sendal Japan, pp2636-2641, September 28-Octobre02, 2004.
[9] Tournassoud.P and Jehl.O, ' Motion planning for a mobile robot with a kinematics constraint', IEEE, pp1785-1790, 1988.
[10] Patrick G. X, 'Fast Swept-Volume distance for Robust Collision Detection', Proceeding of the 1997 IEEE, ICRA, Albuquerque, New Mexico, April 1997.
[11] Zhihua Qu, Jing Wang and Clinton.E 'A New Analytical Solution to Mobile Robot trajectory Generation in the Presence of Moving Obstacles' IEEE Transaction on Robotics Vol.20, No.6. Decembr 2004.
[12] Yasutaka Umeda and Takahiro Yakoh ,'Configuration and Readhesion Control for Mobile Robot With External Sensors ',IEEE Transaction on Idustrial Electronics, Vol.49 .No.1, February 2002.
[13] Ji Yeong Lee and Howie Choset 'Sensor-Based Exploration for Convex Bodies: A New Raodamp for a convex-Shaped Robot ', IEEE Transactions on Robotics Vol.21 .No2 April.2005.
[14] Maaref.H and Barret.C 'Sensor-Based Navigation of a Mobile Robot in an Indoor Environment' Elsevier, Robotics and Autonomous Systems 38 (2002) 1-8.

CHAPITRE 7

[1] Kazuo Tanaka, Hua O. Wang, 'Fuzzy Control Systems Design and Analysis: A Linear matrix inequality approach', Wiley, John & Sons, 2001.
[2] Kwang H. Lee,' First Course on Fuzzy Theory and Applications', Springer, 2005.
[3] Jan Jantzan,'Foundations of Fuzzy control', 2007, John Wiley.

[4] Toshinori Munakata',Fundamentals of the New artificial Intelligence: Neural, Evolutionary, Fuzzy and More', Springer,2008.

[5] Leonid Reznik, 'Fuzzy controllers', Newnes , 1997.

[6] William Siler, James J. Buckley,' Fuzzy Expert Systems and Fuzzy Reasoning', John Wiley & Sons, 2005.

[7] Didier Dubois, Henri Prade,' Fuzzy Sets and Systems: Theory and Applications', Academic Press, 1978.

[8] F. Martin McNeill, Ellen Thro ,'Fuzzy Logic:A Practicl Approach',Academic Press, 1994.

[9] Jyh-Shing Roger Jang, Chuen-Tsai Sun , Eiji Mizutani, 'Neuro-fuzzy and Soft Computing: A Computational Approach to learning and Machine Intelligence' Prentice Hall, 1997.

[10] Runtong Zhang, Yannis A. Phillis and Vassilis S. Kouikoglou, 'Fuzzy Control of Queuing Systems', Springer, 2005.

[11] J. Harris,'Fuzzy Logic Applications in Engineering Science', Springer, 2006.

[12] Ahmad M.Ibrahim, 'Fuzzy Logic for Embedded Systems Applications', ElsevierScience (USA), 2004.

[13] Hungxing li C.l. Philip Chen Han-Pang Huang ,'Fuzzy Neural Intelligent Systems Mathematical Foundation and the Applications in Engineering', CRC Press LLC,2001. .

[14] José Galindo, Angélica Urrutia, Mario Piattini, 'Fuzzy databases : Modelling Design and Implementation',IGP,2006.

[15] Eric A. Wan,'Control Systems: Classical, Neural, and Fuzzy', Oregon Graduate ,Institute Lecture Notes – 1998.

[16] Kevin M. Passino, Stephen Yurkivich, 'Fuzzy Control', Addison-Wesly,1998.

[17] James J. Buckley, Leonard J. Jowers, 'Simulating Continuous Fuzzy Systems', Springer, 2006.

[18] Petr Pivonka, 'Comparative Analysis of Fuzzy PI/PD/PID Controller Based on classical PID Controller Approach',

[19] Jun Zhao,'System Modelling, Identification and Control Using Fuzzy Logic, these de Doctorat, Université Catholique de Louvain, 1995.

[20] José Valente de Oliveira, Wotod Pedrucs, 'Advance in Fuzzy Clustering and Applications', John Wiley& Sons, Ltd, 2007.

CHAPITRE 8

[1] Alessandro Saffioti, 'Fuzzy Logic in autonomous Navigation',Springer-physica Verlag, DE,2001.

[2] Alessandro Saffioti,' Autonomous Robot Navigation : a Fuzzy logic approach', thèse de doctorat, 1997/1998 IRIDIA, Université libre de Bruxelles.

[3] L. Doitsidis, K. P. Valavanis, N.C. Tsourveloudis, 'Fuzzy Logic Based Autonomous Skid Steering Vehicle Navigation', Proceeding of the 2002 IEEE.

[4] Edward Tunstel and Mo Jamshidi, ' Fuzzy Logic and Behavior Control Strategy for Autonomous Mobile Robot Mapping', IEEE1994.

[5] Wei Li, 'Fuzzy Logic-Based Perception-Action Behavior Control of a Mobile Robot in Uncertain Environnements, IEEE1994.

[6] W.-S. Lin, C.-L Huang and M.-K. Chuang , 'Hierarchical fuzzy control for autonomous navigation of wheeled robots', IEE 2005.

[7] H. Maaref, C. Barret, 'Sensor-based navigation of mobile robot in an indoor environment', Elsevier Science 2002.

[8] Toshio Fukuda, Naoyuki Kubota, 'An Intelligent Robotic System Based on a Fuzzy Approach', Proceeding of the IEEE, vol. 87, No.9, September 1999.

[9] Nikhil D. Kelkar,' A Fuzzy Controller for Three Autonomous Mobile Robot', Master University of Cincinnati, 1997.

[10] Pangiotis G. Zavlangas, Spyros G. Tzafestas, K. Althoefer, 'Fuzzy Obstacle Avoidance for omnidirectional Mobile Robots', ESIT 2000, 14-15 Septembre 2000, Aachen, Germany.

[11] Kim C. Ng, Mohan M. Trivedi,' A Neuro-Fuzzy Controller for Mobile Robot Navigation and Multirobot Convoying', IEEE 1998.

[12] Alessandro Saffiotti Enrique H. Ruspini, Kurt Konolige 'Using Fuzzy logic For Mobile Robot Control', chapter 5, Kluwer Academic 1999.

[13] Rui Araujo, Gonçalo Gouveia, and Nuno santos 'Mobile Robot Localization Using a Fuzzy ART World Model' IEEE 2000.

[14] Jianwei Zhang, Alois Knoll, 'Designing fuzzy controllers by rapid learning' Elsevier 1999.

[15] Ivan N. da Silva, Fernando A C. Gomid, Wagner C. do Amaral, 'Navigation of Mobile Robots Using Fuzzy Logic Controller',IEEE 1998.

[16] Young D. Kwon, Jim M. Wonn and Jin S. Lee,'Control of Mobile Robot by using Evolutionary Fuzzy Controller',IEEE, 1998.

[17] Xianhua Jiang, Yuichi Motai, Xingquan Zhu,'Predictive Fuzzy Logic Controller for Trajectory Tracking', IEEE 2005.

[18] Tzuu-Hseng S. Li,Shih-Jie Chang and Wei tong,'Fuzzy Target Tracking Control Of autonomous Mobile Robot by Using Infrared Sensors', IEEE 2004.

[19] R. A. Brooks, 'A robust layered control system for a mobile robot', IEEE journal of robotics and automation 1986.

[20] H.Eskandar, Poya Salehi and M.H.Sabour, 'Fuzzy Logic Tracking Control for a Three wheel Circular Robot in Unkown Environnement', World Applied science journal IDOSI Publication, 2010.

[21] Olumide Obe, Ioan Dumitrache, 'fuzzy control of autonomous mobile robot', U.P.B. Sci.Bull, Vol.72, Issue.3, 2010.

[22] Khaldoun K.Tahboub, Munafs.N.AL.Din, 'A neuro-fuzzy reasoning system for mobile robot navigation', JJMIE, Vol.3, No1, 2009.

[23] T.C.Kuo,'trajectory control of a robotic manipulator utilizing and adaptive fuzzy slinding mode', World academy of science engineering and technology, 2010.

[24]Zafer Bingül, Oguzhan Karahan,'A fuzzy logic controller tuned with PSO for 2 DOF robot trajectory control', Expert system with applications, 2011.

[25] Joshi.M.M, Zaveri.M.A, 'Neuro-fuzzy based autonomous mobile robot navigation system', Control automation robotics & vision (ICARCV), IEEE, 2010.

[26] Kundu.S, 'A fuzzy approach towards behavioural strategie for navigation of mobile agent', Emerging trends in robotics and communication technologies, IEEE, 2010.

[27] Saroj K.P, Dyal.R.P, Anup.K.P, 'Fuzzy logic techniques for navigation of several mobile robots', Journal of applied soft computing, Vol.9, Issue.1, 2009.

CHAPITRE 9

[1]Catarina Silva Æ Bernardete Ribeiro 'Navigating mobile robots with a modular neural architecture',Neural Comput & Applic (2003) 12: 200–211.

[2]Janglová. D,' Neural Networks in Mobile Robot Motion'., Inernational Journal of Advanced Robotic Systems,Volume 1 Number 1 pp. 15-22 (2004), ISSN 1729-8806.

[3]Guilherme A. Barreto, Aluizio F. R. Araújo, Christof Dücker, and Helge Ritter , 'A Distributed Robotic Control System Based on aTemporal Self-Organizing Neural Network', IEEE vol. 32, no. 4, Novembre 2002.

[4]Young H. Kim and Frank L. Lewis ,'Optimal Design of CMAC Neural-Network Controller for Robot Manipulators', IEEE no.1, February 2000.

[5]H. Daniel Patiño, Ricardo Carelli and Benjamín R. Kuchen, 'Neural Networks for Advanced Control of Robot Manipulators', IEEEvol. 13, no. 2, March 2002.

[6] Abderrazek Chatti, Imen Ayari, Piere Born, Mohamed Benrejeb 'On the use of neural techniques for path Following control of car-like mobile robot', Studies informatics and control, Vol.14,No.4 2005.

[7] Low Kian Hsion, 'Intégrated Robot Planing and Control with Extended Kohonen Maps', Phd thesis, National University of Singapore, 2002.

[8] Dierks.T, Jagamathan.S, 'Neural network output feedback control of robot formation', IEEE systemes, Man and cybernics society, 2009.

[9] Vesnac, Ilija Nikolic,' Control of industriel robot using neural network compensator', Theoret. APPL.Mech, Vol.32, No.02, Belgrad, 2005.

[10] Slike Stegories;Marc Timme, Florention Wörgöther & Poramate Manoopong, ' self-Organized adaptation of simple neural circuit enable complexes robot behaviour', nature physics, 2010.

[11] Toshiyuki Kondo, 'Evolutionary design and behavior analysis of neuromodulatore neural networks for mobile robots control', Science direct, Applied soft computing, Vol.7, issue.1, 2007.

[12] Masonari Sato, Atushi Kanda, Karua ishii, 'Performance evaluation of neural network controller system for a wheel type mobile robot', ScienceDirect, Elsevier, 2007.

ANNEXE

ANNEXE A

Interprétation de la base des règles du contrôleur flou comme un PD.

Δe / e	PG	PM	PP	Z	NP	NM	NG
PG	NG	NG	NG	NG	NM	NP	Z
PM	NG	NG	NG	NM	NP	Z	PP
PP	NG	NG	NM	NP	Z	PP	PM
Z	NG	NM	NP	Z	PP	PM	PG
NP	NM	NP	Z	PP	PM	PG	PG
NM	NP	Z	PP	PM	PG	PG	PG
NG	Z	PP	PM	PG	PG	PG	PG

Pour mieux comprendre cette base de règles du contrôleur flou comme PD en partage cette table en cinq (05) zones. Ayant le regard d'un expert qui peut interpréter l'action d'un contrôleur PD on peut donc expliquer le choix de ces règles et par groupe.

Zone 01: dans ce groupe de règles, e et Δe sont petit ou zéro (positive ou négative). Cela veut dire que la sortie du système ne suit pas exactement la consigne mais reste très proche. Alors l'amplitude de la commande doit être petite ou zéro pour compenser la faible erreur. Lorsque e et Δe sont tout deux négatives ou tout deux positives indique que la sortie s'éloigne de la consigne ce qui exige une commande plus importante (NM ou PM) pour prédire par la suite faire réduire ce gap. L'hypothèse e et Δe de même signe indique une divergence entre consigne et mesure, e et Δe de signe contraire indique une convergence de la mesure vers la consigne.

Zone02 : dans ce groupe e et Δe sont de même signe alors la mesure à tendance de ce rapprocher de la consigne mais à quelle vitesse. Si le gap est grand en peut accélérer la convergence par une commande type NM, si le gap est petit et pour éviter une divergence dans l'autre sens on conserve une faible commande comme NP ou Zéro et se mettre et l'inertie d'un effet précédent.

Zone03 : dans ce cas où e est soit petite (PP, Z, NP) donc la mesure est suffisamment collée à la consigne, ou soit relativement grande (NM, NG) la mesure est éloignée de la consigne. Mais pour ce groupe on remarque que Δe est négative (NM, NB) ce qui à tendance d'élargir

le gap. Pour remédier à cette situation une commande consistante (PG, PM) peut rétablir l'ordre.

Zone04 : un raisonnement identique à celui de la zone02 ne mène au résultat.

Zone05 : un raisonnement identique à celui de la zone 03 ne mène au résultat.

Cette base de règle peut être ajuster par l'expert suite aux circonstances du système à contrôler et de l'univers de discoure.

ANNEXE B

Réseau de neurones ARTs

C'est Grossberg qui a introduit les réseaux ART pour, théorie de la résonance adaptative « Adaptive Resonance Theory ».

ART1 est un réseau binaire dont la fonction est d'associer à un patron d'entrée un seul neurone de sortie. Le principe des réseaux ARTs d'inspiration biologique qui est un peut complexe est décrit par Grossberg comme ci-dessous.

Une première couche proposée par Grossberg est appelée couche de filtrage dans le but de réduire l'effet des bruits sur le signal utile. Les fonctions d'activations utilisées sont des fonctions multiplicatives, à savoir que les neurones sont sans interconnections. La sortie de la couche de filtrage est appelée patron de réflectance « reflectance pattern » et représente l'entrée aux couches suivantes, figure B.1.

Du fait que la couche de filtrage ne peut faire, en plus, une reconnaissance; alors Grossberg introduit une couche de codage où des liens ascendants de la couche de filtrage vers la couche de codage mémorisent les associations entre les entrées et les codes. La couche de codage est compétitive et associer à une entrée un seul neurone active.

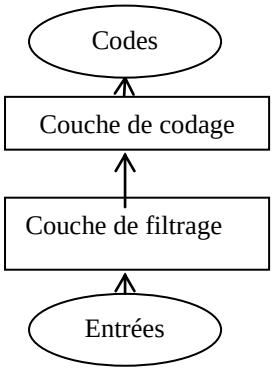


Figure B.1 Illustration des couches du réseau ART1

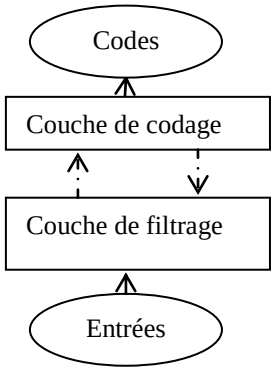


Figure B.2 Ascendance et descendance des informations entre les deux couches jusqu'à la résonance

Mais Grossberg se pose la question de la vérification du code généré, s'il correspond bien à l'entrée, et propose de permettre à la couche de codage de rétroagir sur la couche de filtrage par des liens descendants ayants des poids assurant le recodage et la comparaison.
Cette boucle de vérification, comme montrée en figure B.2, forme un système couplé, et l'activation se propage de façon cyclique entre les deux couches et très vite une résonance s'établie et le réseau se stabilise sur un point fixe.
S'il arrive que la couche de codage soit liée à une autre couche supérieure et qu'elle reçoit une activation descendante, il se peut que l'activation se répercute sur la couche de filtrage et provoque une nouvelle résonance sans présence d'entrée extérieure. Alors Grossberg conditionne l'activation de la couche de filtrage par la présence de deux signaux parmi trois, et de la figure B.3 on voit que le cycle de résonance ne démarre que par la présence des entrées.

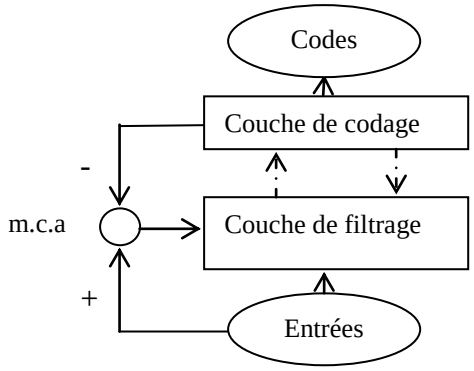


Figure B.3 Conditionnement du démarrage du cycle de résonance
par le mécanisme de contrôle attentionnel 'm.c.a'

Le modèle présenté précédemment (couche de filtrage, couche de codage, mécanisme de contrôle attentionnel) constitue le système attentionnel ("attentionnel subsystem") de ART1. C'est à ce système qui est attribué la responsabilité de trouver le code associé à une entrée donnée, et de valider ce choix par résonance.

Mais que se passe-t-il quand un objet est mal reconnu par ART1 ? C'est que le code choisi par compétition dans la couche de codage n'est pas suffisamment semblable à l'entrée. Il est alors nécessaire d'invalider ce choix, puis de passer à une seconde compétition pour choisir un second code. Ce comportement de recherche séquentiel permettant de passer d'un code à un

autre est géré par une composante supplémentaire de ART1, le mécanisme d'orientation ("orientating subsystem"). Pour ce faire, le mécanisme d'orientation dispose d'un seuil nommé paramètre de vigilance"ρ" ("vigilance parameter"). Ce paramètre mesure si le code déterminé par résonance correspond bien si non on passe vers un autre code et inhibition de l'ancien, et de nouveau le cycle de compétition validation recommence, voir figure B.4.

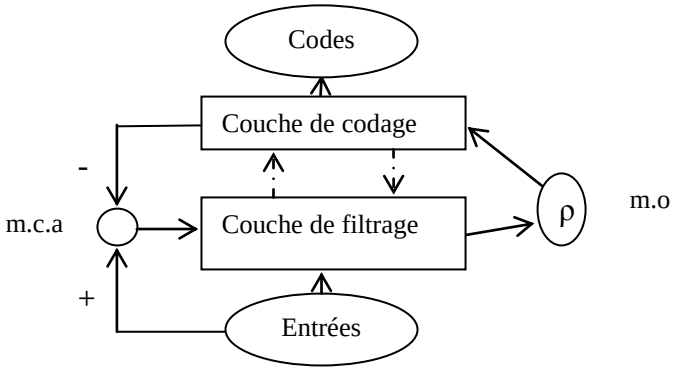


Figure B.4 Le schéma complet du réseau ART1

D'autres réseaux ARTs sont développés par Grossberg tel que ART2, ART3, FuzzyART, FuzzyARTmap où chaque réseau est utilisé pour une application différente. Parmi les applications des ARTs la classification et l'approximation des fonctions.

ANNEXE C

Modèle dynamique du robot mobile différentiel

Le formalisme utilisé est celui de Lagrange qui est décrit par l'équation :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right)-\frac{\partial L}{\partial q}=\tau$$

Où τ représente le vecteur des couples. La variable lagrangienne L est égale à la différence entre l'énergie cinétique et l'énergie potentielle. L'équation dynamique du robot mobile peut être écrite comme suit :

$$M(q)\ddot{q}+V_m(q,\dot{q})\dot{q}+G(q)=B(q)\tau-A^T(q)\lambda$$

Avec
$M(q)$: Représente la matrice d'inertie symétrique 3x3.

$Vm(q,\dot{q})\dot{q}$: Vecteur 3x1 des forces centrifuges et centripètes.
$G(q)$: Force de gravité.
$A(q)$: Matrice des contraintes.
λ : Facteur multiplicatif de Lagrange associe au contraintes.
$B(q)$: Mtrice de 3x2.
τ : Couples extérieure appliqués au robot mobile.

Où :

$$M(q)=\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} ; V_m(q,\dot{q})\dot{q}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} ; \; G(q)=0 ; \; A^T(q)=\begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix} ;$$

$$B(q)=\frac{1}{r}\begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ R & R \end{bmatrix} \; ; \qquad \lambda=-m(\dot{x}_c\cos\theta+\dot{y}_c\sin\theta)$$

m : Masse du robot mobile.
I : Moment d'inertie.
R : Distance entre roues.
r : rayon des roues.

L'équation du modèle dynamique peut être écrite comme suit :

$$\begin{bmatrix} \ddot{x}_c \\ \ddot{y}_c \\ \ddot{\theta} \end{bmatrix}=\begin{bmatrix} -\dot{\theta}\sin\theta\cos\theta & -\dot{\theta}\sin^2\theta & 0 \\ \dot{\theta}\cos^2\theta & \dot{\theta}\sin\theta\cos\theta & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\theta} \end{bmatrix}+\begin{bmatrix} \frac{\cos\theta}{mr} & \frac{\cos\theta}{mr} \\ \frac{\sin\theta}{mr} & \frac{\sin\theta}{mr} \\ \frac{R}{Ir} & \frac{-R}{Ir} \end{bmatrix}\begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}$$

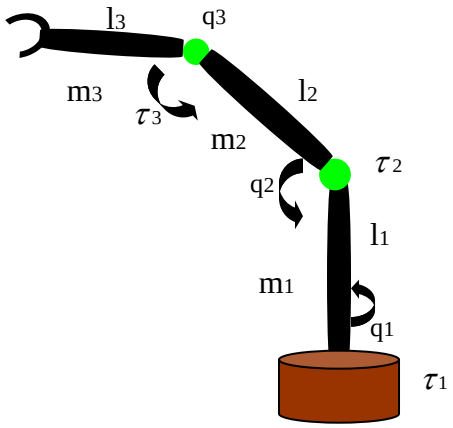
Voir pour ce modèle dynamique l'article de T.H.Lee suivant :

T.H.Lee, H.K.Lam, F.H.F.Leung, P.K.S.Tam,' A path Planning-and-Traking Control for Wheeled Mobile Robots',ICRA, IEEE, 2001.

ANNEXE D

Modèle dynamique d'un bras manipulateur de ddl

l_1=2m ; l_2=2m ; l_3=2m ; m_1=20kg ; m_2=12kg ; m_3=6kg



Robot 3 ddl

$$\tau_1 = (\frac{1}{3}m_2l_2{}^2C_2{}^2 + \frac{1}{3}m_3l_3{}^2C_{23}{}^2 + m_3l_2{}^2C_2{}^2 + m_3l_2l_3C_2C_{23})\ddot{q}_1 + (-\frac{2}{3}m_2l_2{}^2C_2S_2 - \frac{2}{3}m_3l_3{}^2C_{23}S_{23})\dot{q}_1\dot{q}_2$$
$$-(m_3l_2l_3(C_2S_{23} + S_2C_{23}) - 2m_3l_2{}^2C_2S_2)\dot{q}_1\dot{q}_2 + (-\frac{2}{3}m_3l_3{}^2C_{23}S_{23} - m_3l_2l_3C_2S_{23})\dot{q}_1\dot{q}_2$$
$$+(-\frac{2}{3}m_3l_3{}^2C_{23}S_{23} - m_3l_2l_3C_2S_{23})\dot{q}_1\dot{q}_3$$

$$\tau_2 = (\frac{1}{3}m_2l_2{}^2 + \frac{1}{3}m_3l_3^2 + m_3l_2l_3C_3 + m_3l_2{}^2)\ddot{q}_2 + (\frac{1}{3}m_3l_3{}^2 + \frac{1}{2}m_3l_2l_3C_3)\ddot{q}_3 + (\frac{1}{3}m_2l_2{}^2C_2S_2 + \frac{1}{3}m_3l_2l_3C_{23}S_{23}$$
$$+\frac{1}{2}m_3l_2l_3(C_{23}S_2 + C_2S_{23}) + m_3l_2{}^2C_2S_2)\ddot{q}_1^2 - m_3l_2l_3S_3\dot{q}_2\dot{q}_3 - \frac{1}{2}m_3l_2l_3S_3\ddot{q}_3^2 + \frac{1}{2}m_3l_2gC_2$$
$$+m_3g(\frac{1}{2}l_3C_{23} + l_2C_2)$$

$$\tau_3 = (\frac{1}{3}m_3l_3{}^2 + \frac{1}{2}m_3l_2l_3C_3)\ddot{q}_2 + \frac{1}{3}m_3l_2l_3{}^2\ddot{q}_3 + (\frac{1}{3}m_3l_3^2C_{23}S_{23} + \frac{1}{2}m_3l_2l_3C_2S_{23})\ddot{q}_1^2$$
$$+\frac{1}{2}m_3l_2l_3S_3\dot{q}_2^2 - \frac{1}{2}m_3l_3gC_{23}$$