République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Batna
Faculté des Sciences de l'Ingénieur
Département d'Electronique

Thèse
Préparée au
Laboratoire d'Electronique Avancée (LEA) Batna

Par :
Benmakhlouf Abdeslam
Ingénieur d'état en électronique

En vue de l'obtention du diplôme de :
Magister En Robotique

Thème

Contrôleur Flou Pour La Navigation D'un Robot Mobile D'intérieur

Soutenu le : 11/12/2006

Membres du Jury

Benoudjit Nabil	*Président de Jury*	*Maître de Conférences Université de Batna*
Louchene Ahmed	*Rapporteur*	*Maître de Conférences Université de Batna*
Betka Achour	*Examinateur*	*Maître de Conférences Université de Biskra*
Ameddah Djamel eddine	*Examinateur*	*Maître de Conférences Université de Batna*

Contrôleur Flou Pour La Navigation D'un Robot Mobile D'intérieur

Remerciements

Je tiens à remercier vivement monsieur ***LOUCHENE Ahmed*** pour son esprit scientifique et compréhensif qui m'a beaucoup aidé avec ses idées, ses conseils et surtout ses critiques.
Je tiens également à remercier les membres de notre groupe pour leur coopération.
Je remercie monsieur ***N. BENOUDJIT*** d'avoir accepté la présidence du jury, monsieur ***A. BETKA*** et monsieur ***D. AMMEDAH*** d'avoir pris de leurs temps pour examiner ma thèse.

A toute ma famille.
A tous mes amis.

Sommaire

Introduction générale 01

Chapitre1 : La robotique mobile.

1.1. Introduction 05

1.2. Définitions 05

1.3. Les robots mobiles à roues 06

1.3.1. L'unicycle 08

1.3.2. Le tricycle 08

1.3.3. Les véhicules 08

1.3.4. Robot à traction synchrone 09

1.4. La perception 10

1.5. La localisation 11

1.5.1. Localisation relative ou à l'estime 12

1.5.1.1. Odométrie directe 13

1.5.1.2. Odométrie indirecte 13

1.5.2. Localisation absolue 14

1.6. La navigation 14

1.7. Conclusion 15

Chapitre2 : La logique floue et la commande.

2.1. Introduction 17

2.2. Notions De Bases De La Logique Floue 17

2.2.1. La théorie des ensembles flous 17

2.2.2. Différence entre ensemble flou et ensemble booléen 18

2.2.2.1. Variables linguistiques 18

2.2.2.2. Définitions 19

2.2.2.3. Caractéristiques d'un ensemble flou 19

2.2.2.4. Opérations sur les ensembles flous 20

2.2.2.5. Le principe d'extension 22
2.2.3. Relations floues 22
2.2.3.1. Produit cartésien 22
2.2.3.2. Projection 23
2.2.3.3. Extension cylindrique 23
2.2.3.4. Composition de relations floues 23
2.3. La Commande En Logique Floue 24
2.3.1. Acquisition de la connaissance et écriture de la base de règles 24
2.3.2. Propriétés des bases de règles 26
2.3.2.1. Continuité 26
2.3.2.2. Consistance 27
2.3.2.3. Complétude 28
2.3.3. Les différentes étapes de la commande floue 29
2.3.3.1. Mise en forme des entrées, normalisation 31
2.3.3.2. Fuzzification 31
2.3.3.3. Traitement des prémisses composées 33
2.3.3.4. Inférence floue 33
2.3.3.5. Agrégation des règles 35
2.3.3.6. Défuzzification 36
2.3.3.7. Dénormalisation 39
2.3.4. Structure de base d'un contrôleur flou : analogie structurelle avec les P.I.D 39
2.3.5. Réglage, stabilité et robustesse d'un contrôleur flou 40
2.3.5.1. Réglage 40
2.3.5.2. Stabilité 41
2.3.5.3. Robustesse 43
2.4. Conclusion 43

Chapitre 3 : La conception d'un contrôleur flou

3.1. Introduction 46
3.2. Conception D'un Contrôleur Flou 47
3.2.1. Choix de la structure du contrôleur 48
3.2.2. Choix de la stratégie de fuzzification 48
3.2.3. Etablissement des règles d'inférence 49
3.2.4. Choix de la méthode d'inférence 50
3.2.5. Choix de la stratégie de défuzzification 50
3.3. Notre contribution : La Logique Classique Modifiée 51
3.3.1. Les modifications proposées 51
3.4. Exemple d'application 56
3.5. Conclusion 60

Chapitre 4 : Application A La Poursuite D'une Trajectoire Par Un Robot Mobile

4.1. Introduction 62
4.2. Le robot mobile 62
4.3. La stratégie de la poursuite 65
4.4. Le contrôleur conçu 68
4.5. Les résultats de simulation 69
4.6. Conclusion 76
Conclusion générale 77
Bibliographie 79

Introduction Générale

La robotique mobile cherche depuis des années à rendre une machine mobile autonome face à son environnement pour qu'elle puisse sans intervention humaine accomplir les missions qui lui sont confiées. Le spectre des missions que les roboticiens veulent voir accomplir par leurs machines est immense : exploration en terrain inconnu, manipulation d'objets, assistance aux personnes handicapées, transport automatisé, etc. De grand progrès ont été accomplis dans tous les domaines de la robotique : perception et modélisation de l'environnement, commande automatique des actionneurs, planification de mouvements, ordonnancement de tâches, gestion de l'énergie,...

Bien que la machine autonome parfaite, capable de s'adapter à de nombreuses situations et capable d'estimer ses possibilités d'actions, n'existe pas encore, des applications concrètes des techniques de robotique mobile ont été réalisées. Ces applications très remarquables ne sont toutefois que des cas particuliers pour lesquels des solutions plus ou moins spécifiques ont été développées. En effet, les problèmes posés par la navigation autonome d'un véhicule sont très nombreux et font l'objet de plusieurs recherches. En simplifiant, une machine autonome peut être définie comme étant l'association d'une intelligence artificielle avec des capacités de perception et de modélisation de son environnement et de son propre état ; elle doit être aussi dotée de capacités d'actions sur son propre état et sur son environnement.

Toutes ces capacités sont nécessaires mais chacune amène ses difficultés. Alors, pour progresser, les chercheurs de chaque spécialité isolent les problèmes,

proposent des solutions et les comparent. Les robots d'expérimentations des laboratoires deviennent alors les cobayes des sciences naturelles. On implante sur leurs calculateurs un planificateur de mouvement à réseau probabiliste, on connecte sur leurs bus d'acquisition une paire de caméras ou un capteur télémétrique laser ou autre système de perception et on étudie les possibilités d'actions obtenues.

Dans ce travail, on s'intéresse à la navigation d'un robot mobile à roues. Plus précisément à la poursuite d'une trajectoire par un robot mobile à commande différentielle. Dans le cas idéal d'un environnement exactement modélisé et où la cinématique du robot est parfaitement maîtrisée les étapes de la navigation se résument simplement comme étant la planification d'une trajectoire puis son exécution. Mais alors pourquoi nos voitures ne sont-elles pas entièrement automatiques ?

Dans la réalité nous faisons, sans nous en rendre compte, des actions extrêmement complexes lorsque nous nous déplaçons : nous estimons notre position, nous analysons en permanence les objets qui nous entourent et nous nous mouvons ; c'est dans cette optique que les robots mobiles sont développés.

Pour imiter les compétences d'un bon conducteur d'automobile, la logique floue est la technique la plus convenable pour la représentation des connaissances imprécises d'un expert. Notre travail concerne la conception d'un contrôleur flou pour la poursuite de trajectoire par un robot mobile d'intérieur. Notre intérêt est concentré sur l'optimisation du temps de traitement pour conserver l'aspect temps réel du contrôleur conçu. Cela nous a mené a proposé une méthode basé sur la logique classique sur laquelle on a introduit deux modifications pour rendre le raisonnement en logique traditionnelle plus souple et imiter le comportement d'un contrôleur flou.

Ce mémoire est organisé comme suit : dans le chapitre 1, nous allons faire un tour rapide dans le monde de la robotique mobile pour examiner en bref les

différentes parties constitutives d'un robot mobile. Le chapitre 2 est consacré à la logique floue et à la commande floue, où, nous exposerons les principales théories de la logique floue nécessaires à la conception d'un contrôleur. Une méthode générale de conception de contrôleur flou est présentée dans le chapitre 3, suivie de la présentation du contrôleur proposé. Des exemples d'application y sont aussi présentés pour comparer les performances du contrôleur proposé avec le contrôleur flou conçu. Dans le chapitre 4, nous présenterons les résultats de simulation des deux algorithmes appliqués à la matérialisation d'une stratégie de poursuite de trajectoire par un robot mobile d'intérieur. Nous terminerons notre travail par une conclusion générale récapitulant ce qui a été fait et exposant les perspectives de notre travail.

Chapitre 1
La Robotique Mobile

1.1. Introduction

De manière générale, on regroupe sous l'appellation *robots mobiles* l'ensemble des robots à base mobile, par opposition notamment aux robots manipulateurs. L'usage veut néanmoins que l'on désigne le plus souvent par ce terme les robots mobiles à roues. Les autres robots mobiles sont en effet le plus souvent désignés par leur type de locomotion, qu'ils soient marcheurs, sous-marins ou aériens.

On peut estimer que les robots mobiles à roues constituent la grande partie des robots mobiles. Historiquement, leur étude est venue assez tôt, suivant celle des robots manipulateurs. Leur faible complexité en a fait de bons premiers sujets d'étude pour les roboticiens intéressés par les systèmes autonomes. Cependant, malgré leur simplicité apparente, ces systèmes ont soulevé un grand nombre de problèmes difficiles. De ce fait, les applications industrielles utilisant des robots mobiles sont rares. Cela est dû au fait que, contrairement aux robots manipulateurs qui travaillent exclusivement dans des espaces connus et de manière répétitive, les robots mobiles sont destinés à évoluer de manière autonome dans des environnements dynamiques qui peuvent ne pas être connus.

Néanmoins, l'intérêt indéniable de la robotique mobile est d'avoir permis d'augmenter considérablement les connaissances sur la localisation et la navigation des robots mobiles autonomes.

1.2. Définitions

En général, on peut définir un robot mobile [1] comme étant une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a.

En particulier, *un robot mobile autonome* est un système mécanique, électronique et informatique complexe mettant en oeuvre :

- Un ensemble de capteurs (extéroceptifs et/ou proprioceptifs) :

Les capteurs extéroceptifs ont pour objectif d'acquérir des informations sur l'environnement proche du véhicule. Les capteurs proprioceptifs fournissent des données sur l'état interne du robot (telles que sa vitesse ou sa position).

- Un ensemble d'effecteurs :

L'objectif du robot est d'atteindre un objectif dans son environnement en évitant les obstacles. Le problème que l'on doit résoudre est de déterminer en fonction des données capteurs quelles commandes doivent être envoyées à chaque instant au robot pour atteindre cet objectif. Ces effecteurs ont comme but de permettre au robot d'évoluer dans un monde prévu à l'origine pour l'homme. Les plus courants sont les systèmes à roues, mais il existe aussi des robots à chenilles, à pattes ou se déplaçant par reptation.

Le type de locomotion définit deux types de contraintes :

- les contraintes cinématiques, qui portent sur la géométrie des déplacements possibles du robot.
- Les contraintes dynamiques, liées aux effets du mouvement (accélérations bornées, vitesses bornées, présence de forces d'inertie ou de friction)

Selon sa cinématique, un robot est dit :

- **holonome**, s'il peut se déplacer instantanément dans toutes les directions.
- **non- holonome**, si ses déplacements autorisés sont des courbes dont la courbure est bornée.

1.3. Les robots mobiles à roues

Les roues sont le moyen de locomotion le plus répandu en matière de robotique mobile [2]. En fait, les robots mobiles à roues sont faciles à réaliser et présentent de grandes possibilités de déplacement et de manoeuvrabilité avec

une vitesse et une accélération importantes. Bien évidemment, pour un ensemble donné de roues, toute disposition ne conduit pas à une solution viable. Un mauvais choix peut limiter la mobilité du robot ou occasionner d'éventuels blocages. Par exemple, un robot équipé de deux roues fixes non parallèles ne pourrait pas aller en ligne droite ! 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, lorsqu'il existe, est appelé *centre instantané de rotation (CIR)*. Les points de vitesse nulle liés aux roues se trouvent sur leurs axes de rotation, il est donc nécessaire que le point d'intersection des axes de rotation des différentes roues soit unique. Pour cette raison, il existe en pratique trois principales catégories de robots mobiles à roues, que l'on va présenter par la suite. En général, les robots mobiles à roues ont des structures à locomotion différentielle. La figure 1.1 montre une structure type de robot mobile à commande différentielle où le déplacement est assuré par le biais de deux moteurs d'entraînement indépendants, chacun d'eux est lié à une roue motrice.

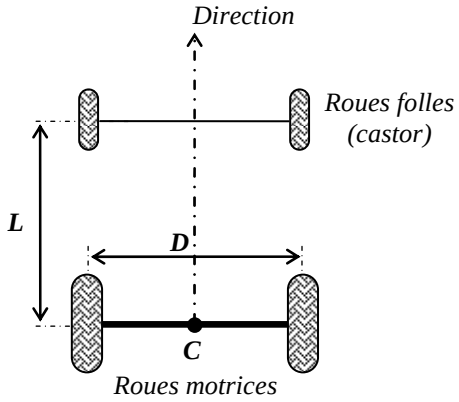

Figure 1.1. *Robot mobile à commande différentielle*

En ce qui concerne la commande des moteurs, cette indépendance permet au robot mobile de changer de direction en jouant uniquement sur les vitesses des deux roues motrices. Les roues folles sont libres, leur rôle essentiel est d'assurer la stabilité de la plate-forme du robot.

1.3.1. L'unicycle

On désigne par unicycle un robot actionné par deux roues indépendantes et possédant éventuellement un certain nombre de roues folles assurant sa stabilité. Le schéma d'un robot de type unicycle est donné à la figure 1.2. On y a omis les roues folles, qui n'interviennent pas dans la cinématique, dans la mesure où elles ont été judicieusement placées. Ce type de robot est très répandu en raison de sa simplicité de construction et de ses propriétés cinématiques intéressantes.

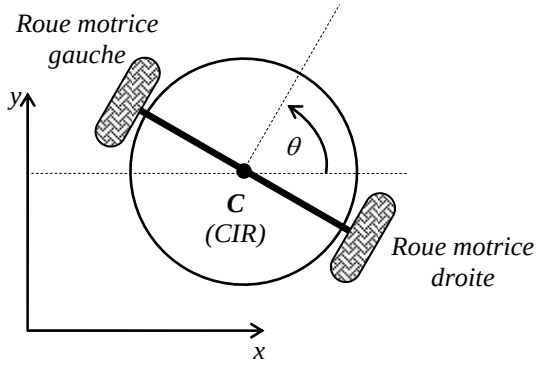

Figure 1.2. *Robot mobile de type unicycle*

1.3.2. Le tricycle

L'architecture d'un robot mobile de type tricycle est donnée sur la figure 1.3, il utilise une roue motrice avant et deus roues passives arrières (ou vice versa). Le mouvement est conféré au robot par deux actions : la vitesse longitudinale et l'orientation de la roue orientable. Ce type de robots est très connu dans les applications des AGVs pour leur simplicité inhérente. Pour la commande et la localisation ce type de robot : un capteur d'orientation est associé à la roue orientable, et deux encodeurs sont associés aux roues motrices.

1.3.3. Les véhicules (structure d'ackerman)

Le cas du robot de type voiture est très similaire à celui du tricycle. La différence se situe au niveau du train avant, qui comporte deux roues au lieu d'une. Cela va de soit, on rencontre beaucoup plus souvent ce type de systèmes. On parle de robot dès lors que la voiture considérée est autonome [3], donc sans

chauffeur ni télépilotage. Il s'agit là d'un des grands défis issus de la robotique mobile. La figure 1.4 représente ce type d'architecture.

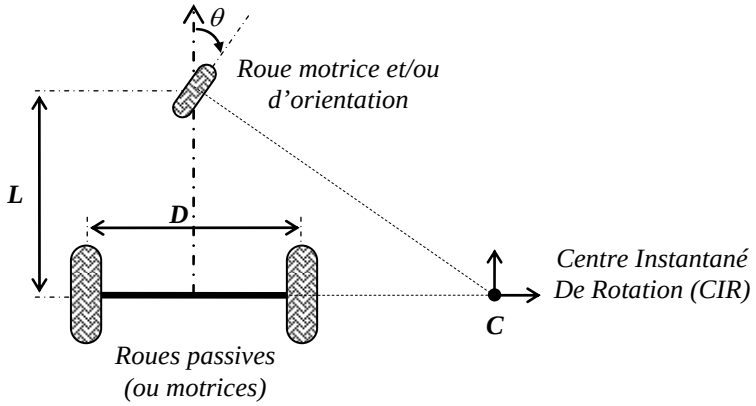

Figure 1.3. *Robot mobile de type tricycle*

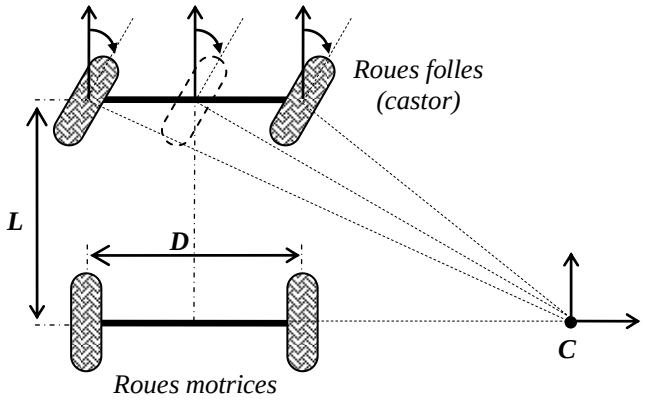

Figure 1.4. *Robot mobile de type voiture*

1.3.4. Robot à traction synchrone

La traction synchrone est une technique utilisée pour minimiser l'effet de glissement et augmenter la force de traction.

On rencontre ce type de robot dans l'industrie automobile, et dans les robot tout terrains. La configuration du robot à traction synchrone est similaire à un robot à trois roues couplées de façon qu'elles soient actionnées en même temps. La figure 1.5 montre un robot à quatre roues couplées avec des chaînes.

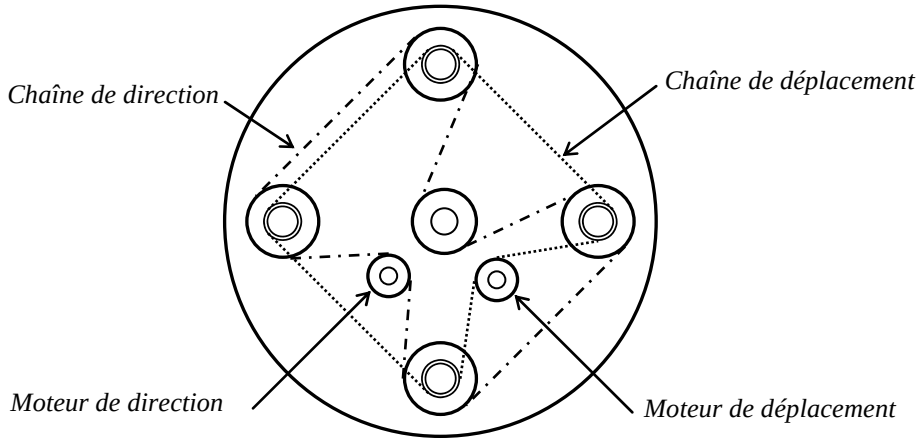

Figure 1.5. *Robot mobile à traction synchrone.*

1.4. La perception

La notion de perception en robotique mobile est relative à la capacité du robot à recueillir, traiter et mettre en forme des informations qui lui sont utiles pour agir et réagir dans l'environnement qui l'entoure [4]. Elle est donc la faculté de détecter et/ou appréhender l'environnement proche ou éloigné du robot. Alors que pour des tâches de manipulation on peut considérer que l'environnement du robot est relativement structuré, ce n'est plus le cas lorsqu'il s'agit de naviguer de manière autonome dans des lieux très partiellement connus. Aussi, pour extraire les informations utiles à l'accomplissement de sa tâche, il est nécessaire que le robot dispose de nombreux capteurs mesurant aussi bien son état interne que l'état de l'environnement dans lequel il évolue. Le choix des capteurs dépend bien évidemment de l'application envisagée.

La perception est nécessaire pour la sécurité du robot, la modélisation de l'environnement et l'évitement et le contournement d'obstacles.

Les moyens utilisés pour la perception de l'environnement sont nombreux et variés, parmi ceux-ci nous pouvons citer :

- Les systèmes de vision,
- Les télémètres laser et ultrasonores,

- Les capteurs optiques et infrarouges,
- Les capteurs tactiles.

1.5. La localisation

La localisation d'un robot mobile s'effectue par des capteurs proprioceptifs et/ou extéroceptifs. Cette localisation est d'autant plus nécessaire que le lieu d'évolution est encombré et complexe. Le comportement de l'être vivant illustre bien ces propos, en effet il doit toujours connaître sa situation pour se déplacer d'un point à un autre, soit en identifiant des repères artificiels, on parle de localisation absolue, soit tout simplement en mesurant les distances parcourues et les directions empruntées depuis sa position initiales. Un robot mobile doit connaître ses coordonnées de position pour être autonomes vis-à-vis de l'espace et de l'intervention humaine.

Les premières applications de robotique mobiles consistaient essentiellement en des tâches répétitives. Ces robots nécessitaient soit l'intervention humaine à distance pour beaucoup d'opérations, un environnement structuré avec des systèmes de guidage passifs ou actifs. Le filoguidage, ou guidage inductif, est une des techniques les plus utilisées pour les robots mobiles d'atelier. Le principe consiste à encastrer, dans le sol, un fil parcouru par un courant alternatif. Une force électromotrice est induite dans deux bobines disposées sous le robot de part et d'autre du fil inductif. Le guidage consiste à minimiser la différence entre les deux forces électromotrices induites afin de maintenir par asservissement le robot mobile sur la trajectoire matérialisée par le fil. Ce fil sert également à transmettre des informations au robot. La modification de la trajectoire avec ce système de guidage nécessite des travaux importants qui peuvent entraîner l'arrêt de toute ou d'une partie de la production.

Dans le cas du guidage optique, la position relative du robot mobile est obtenue par la réflexion d'une onde lumineuse sur une piste matérialisée par des

peintures ou des bandes adhésives réflectrices disposées sur le sol. Le signal détecté est maximal si le faisceau lumineux se réfléchit sur la bande, et il est minimal quand le faisceau se réfléchit en dehors de la bande. Des marquages particuliers sont disposés en certains endroits stratégiques pour informer le robot mobile sur l'évolution de la trajectoire, sur sa position ou sur le point d'arrêt. Ce système ne fonctionne correctement que si le contraste entre le sol et le marquage au sol est suffisant, ainsi, il ne supporte pas les poussières et convient mal au milieu industriel.

La lourdeur de ces systèmes de guidage : statisme de l'installation, difficultés pour le faire évoluer … etc, ont conduit les chercheurs à étudier d'autres systèmes plus souples d'utilisation et plus performants [5].

C'est ainsi que les méthodes de localisation se regroupent en deux catégories, soit :

- La localisation à l'estime ou relative qui est obtenue par des informations issues de capteurs proprioceptifs.
- La localisation absolue qui est obtenue par des informations issues de capteurs extéroceptifs.

1.5.1. Localisation relative ou à l'estime

Cette méthode est basée sur l'intégration des déplacements élémentaires du robot mobile, on l'appelle aussi localisation relative car les coordonnées du robot sont calculées en fonction de la position précédente et du déplacement en cours.

Les erreurs dûes à cette méthode peuvent être importantes car cumulatives avec la distance et fonction du type de trajectoires. On distingue deux méthodes principales de localisation relative :

- La méthode odométrique (directe ou indirecte)
- La méthode inertielle.

1.5.1.1. Odométrie directe

La technique de l'odométrie est très utilisée pour localiser les robots mobiles, elle présente l'avantage d'être simple d'emploi et d'un coût faible. Son principe repose sur la mesure de la vitesse angulaire de rotation des roues motrices. Pour un robot mobile à roues non orientables, on déduit à partir de la somme et de la différence des vitesses de rotation de chaque roue, la vitesse linéaire du robot ainsi que sa vitesse angulaire. Le déplacement du robot est obtenu par l'intégration des déplacements élémentaires entre deux instants et ainsi de proche en proche on obtient une estimation de la position à partir d'une référence initiale. Ceci suppose de parfaites conditions d'évolution.

La mesure des angles est généralement faite par des codeurs optiques incrémentaux couplés à des roues qui peuvent être distinctes des roues motrices.

Les erreurs de cette technique peuvent être de différentes natures. Elles peuvent être systématiques et constantes ou non selon qu'elles sont liées à des paramètres du robot mobile (longueur de l'entraxe, diamètre des roues…) ou à l'environnement. Ainsi, l'état du terrain introduit des erreurs de même que le glissement des roues. Des erreurs peuvent provenir de la résolution des capteurs numériques de mesure et de la discrétisation des calculs.

1.5.1.2. Odométrie indirecte

La localisation relative par odométrie indirecte s'effectue par des moyens inertiels qui mesurent soit la vitesse soit l'accélération puis par intégration, on en déduit les déplacements élémentaires. Ces moyens permettent une localisation précise pour certains engins parcourant de longues distances.

Il existe d'autres méthodes de localisation relative, mais toutes présentent des dérives assez importantes qui ne peuvent être compensées que par des informations supplémentaires issues d'un autre mode de localisation.

1.5.2. Localisation absolue

Les inconvénients cités ci-dessous ont conduit à développer des systèmes de localisation absolue qui fournissent la position et l'orientation du robot mobile par rapport à des points fixes du repère de travail donc en coordonnées absolues.

Pour se faire le robot doit effectuer des mesures par rapport à l'environnement. Ces mesures sont fournies par des capteurs dits extéroceptifs ou externes car ils ne s'intéressent pas aux mouvements internes du robot, comme c'est le cas pour l'odométrie [5].

1.6. La navigation

Les sections précédentes ont permis de mettre en place les outils nécessaires pour faire naviguer un robot mobile dans un environnement d'intérieur : la compréhension du mode de locomotion et de localisation de ce robot dans son environnement. Il s'agit maintenant d'utiliser au mieux la motricité du robot et sa localisation pour accomplir la tâche de navigation de manière autonome.

Un mouvement est une application définie en fonction du temps *t*, reliant un point initial à l'instant t_0 à un point final à l'instant t_f. Une trajectoire est le support d'un mouvement. Il s'agit donc d'une courbe paramétrée par une variable *s* quelconque, par exemple l'abscisse curviligne normalisée ($s \in [0,1]$) de la courbe sur laquelle se déplace le robot. L'évolution du paramètre *s* en fonction du temps *t* est appelée mouvement sur la trajectoire.

Le problème de navigation d'un robot mobile consiste de la manière la plus générale à trouver un mouvement dans l'espace des configurations sans collisions, traditionnellement noté C_{free}. Ce mouvement amène le robot d'une configuration initiale $q_0 = q(t_0)$ à une configuration finale $q_f = q(t_f)$. On peut néanmoins donner des définitions différentes de la tâche de navigation à accomplir, selon le but recherché par exemple on peut souhaiter seulement placer le robot dans une zone donnée et relâcher la contrainte d'orientation, etc.

La tâche de navigation ainsi définie est donc limitée à un seul mouvement. Il existe néanmoins une très grande variété de travaux et de méthodes permettant d'aborder ce problème difficile. Pour différencier les techniques de navigation, on peut de manière générale distinguer deux approches :

– la première consiste à planifier le mouvement dans l'espace des configurations et à l'exécuter par asservissement du robot sur le mouvement de consigne (schéma planification exécution) ;

– la seconde consiste à offrir un ensemble de primitives plus réactives. Elles correspondent alors à des sous tâches (suivre un mur, éviter un obstacle) dont on estime que l'enchaînement est du ressort d'un planificateur de tâches ayant décomposé la tâche globale.

1.7. Conclusion

Les robots mobiles à roues sont les robots mobiles les plus répandus, vu leurs structures mécaniques simples et leur commande relativement plus facile que les autres robots mobiles qui diffèrent par leurs moyens de locomotion.

La commande d'un robot mobile se divise en trois étapes principales : perception, décision et action. La dernière étape concerne l'exécution des mouvements planifiés, c'est une étape que doit maîtriser un robot mobile pour accomplir ses missions avec succès.

Chapitre 2
La Logique Floue Et La Commande

2.1. Introduction

Parmi les récents développements des techniques de commande, l'introduction de nouvelles techniques telles que la logique floue, a suscité un intérêt sans cesse croissant depuis les quelques dernières décennies. Il suffit de voir les nombreuses applications industrielles qui en découlent et de consulter l'abondante littérature sur le sujet pour s'en convaincre.

L'intérêt de la logique floue réside dans sa capacité à traiter, l'imprécis, l'incertitude et le vague. Elle est issue de la capacité de l'homme à décider et agir de façon pertinente malgré le flou des connaissances disponibles et a été introduite dans le but d'approcher le raisonnement humain à l'aide d'une représentation adéquate des connaissances. Aussi, le succès de la commande floue trouve en grande partie son origine dans sa capacité à traduire une stratégie de contrôle d'un opérateur qualifié en un ensemble de règles linguistiques « *si … alors* » facilement interprétables.

L'utilisation de la commande floue est particulièrement intéressante lorsqu'on ne dispose pas de modèle mathématique précis du processus à commander ou lorsque ce dernier présente de trop fortes non linéarités ou imprécisions.

Dans ce chapitre, nous présenterons quelques aspects théoriques de la logique floue, ainsi que les bases de son application pour la commande de processus.

2.2. Notions De Bases De La Logique Floue

2.2.1. La théorie des ensembles flous

Ce n'est qu'à partir de 1965 que L. A. Zadeh, professeur à l'université de Berkeley, jeta les bases de ce qu'il dénomma « *fuzzy set* » (ensemble flou), prenant ainsi en considération le problème posé par les connaissances imprécises ou vagues. La notion d'ensemble flou permet alors des graduations dans l'appartenance d'un élément à une classe, c'est-à-dire autorise un élément à appartenir plus ou moins fortement à cette classe.

2.2.2. Différence entre ensemble flou et ensemble booléen

Alors qu'un ensemble booléen est défini par sa fonction caractéristique *f* à valeurs 0 ou 1, un ensemble flou est défini par sa fonction d'appartenance μ à valeurs *dans l'intervalle [0, 1].*

Considérons l'exemple suivant :

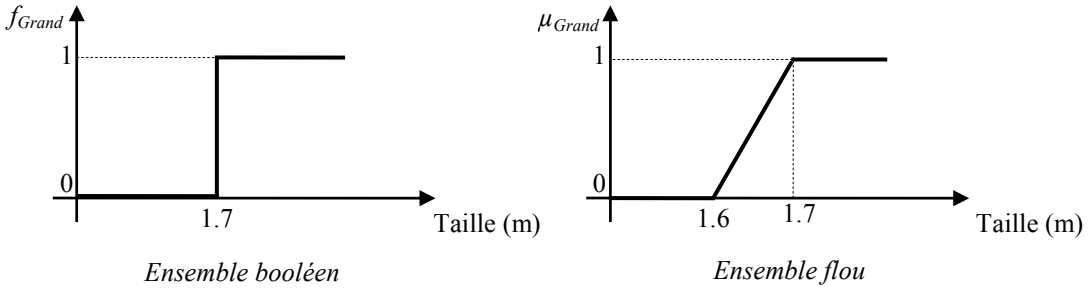

Figure 2.1. *Ensemble booléen et ensemble flou*

L'ensemble des tailles possibles d'un individu représente *l'univers de discours* de la variable « *taille* ». « *Grand* » est une *valeur linguistique* de cette variable.

Soit 1.65 m la taille de *X.* dans le cas de l'ensemble booléen *X* n'appartient pas à la classe des « *Grands* ». Dans le cas de l'ensemble flou *X* appartient à la classe des « *Grands* » avec un certain degré d'appartenance.

La notion d'ensemble flou évite l'utilisation arbitraire des limites rigides d'appartenance à des classes, il serait aberrant de considérer qu'un individu de 1.70 m est grand, mais qu'un individu de 1.695 m ne l'est pas.

2.2.1.1. Variables linguistiques

Une variable linguistique est représentée par un triplet (V, U, T_V) où *V* est la variable linguistique elle-même, *U* est l'univers de discours et T_V l'ensemble des caractérisations floues de la variable.

Considérons par exemple la variable *taille* définie sur l'ensemble des entiers positifs et caractérisée par les ensembles flous *petit, moyen, grand.* La variable *taille* est alors représentée par le triplet suivant : *{taille, R^+, (petit, moyen, grand)}.*

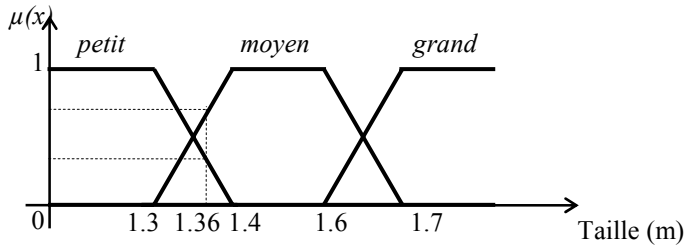

Figure 2.2. *Une personne mesurant 1.36 m est petite avec un degré d'appartenance 0.4 et moyenne avec un degré de 0.6*

2.2.1.2. Définitions

Un ensemble flou A est défini sur l'univers de discours X est noté :

$A = \int_u \mu_A(x) \Big/ x$, dans le cas continu,

$A = \sum_{i=1}^{n} \mu_A(x_i) \Big/ x_i$, dans le cas discret.

2.2.1.3. Caractéristiques d'un ensemble flou

Ce sont essentiellement celles qui montrent dans quelle mesure l'ensemble flou diffère de l'ensemble booléen.

- **Support :** $\text{supp}(A) = \{x \in X / f_A(x) \neq 0\}$, c'est l'ensemble booléen des éléments de X qui appartiennent au moins un peu à l'ensemble flou.
- **Hauteur :** $\text{h}(A) = \sup_{x \in X} f_A(x)$, c'est le plus fort degré d'appartenance avec lequel un élément de X appartient à l'ensemble flou, ce dernier est dit normalisé si sa hauteur est de 1.
- **Noyau :** $\text{noy}(A) = \{x \in X / f_A(x) = 1\}$, c'est l'ensemble booléen de tous les éléments appartenant de façon absolue à l'ensemble flou.

Coupe de niveau α ou α-coupe : $\alpha - coupe(F) = F_0 = \{x \in X | \mu_A(x) \geq \alpha\}$, c'est l'ensemble booléen des éléments de X qui appartiennent à A avec un degré d'appartenance au moins égal à α.

Singleton flou de X : $\mu_{\{x\}}(x) = 1 \; et \; \mu_{\{x\}}(y) = 0, \forall y \neq x$

Ensemble flou convexe : un ensemble flou A est convexe si :

$$\mu_A\left(\lambda x_1 + (1-\lambda)x_2\right) \geq \min\left(\mu_A(x_1), \mu_A(x_2)\right), \textit{ tel que } x_1, x_2 \in X, \lambda \in [0,1]$$

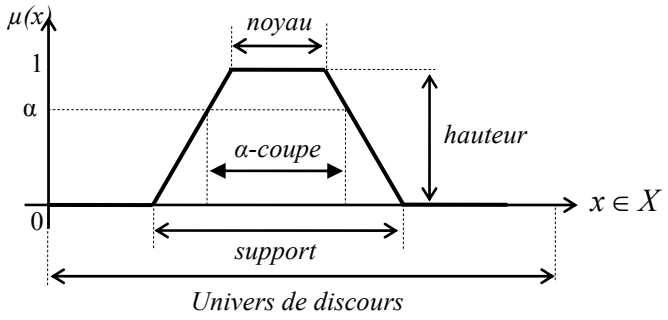

Figure 2.3. *Support, noyau, hauteur et α-coupe d'un ensemble flou.*

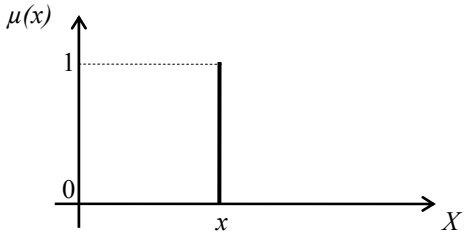

Figure 2.4. *Singleton flou.*

Partition floue : N ensembles flous $(A_1, A_2, \ldots, A_N)$ définis sur l'univers de discours X forment une partition floue si :

$$\forall x \in X, \sum_{i=1}^{N} \mu_{A_i}(x) = 1$$

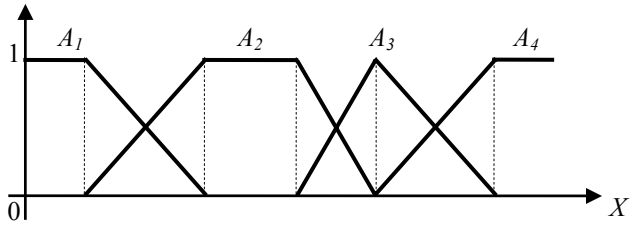

Figure 2.5. *Exemple de partition floue.*

Une partition floue composée d'ensembles flous convexes normaux implique que pas plus de deux fonctions d'appartenance se recouvrent.

2.2.1.4. Opérations sur les ensembles flous

Nous définissons, ci-dessous, les notions d'intersection, d'union et de complémentation d'ensembles flous.

Soient A et B deux ensembles flous décrits par leurs fonctions d'appartenance $\mu_A(x)$ et $\mu_B(x)$. Une définition de l'union flou mène à la fonction d'appartenance donnée par :

$$\mu_{A\cup B}(x) = \max\left[\mu_A(x), \mu_B(x)\right]$$

Et une définition de l'intersection floue mène à la fonction d'appartenance donnée par :

$$\mu_{A\cap B}(x) = \min\left[\mu_A(x), \mu_B(x)\right]$$

En ajoutant la définition de la fonction d'appartenance du complément flou :

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$

Il existe d'autres opérateurs qui sont les normes triangulaires (t-normes) pour l'intersection et les conormes triangulaires (t-conormes) pour l'union.

	t-normes	**t-conormes**
Zadeh (1973)	$\min(x,y)$	$\max(x,y)$
Bendler et Kahout (1980)	$x.y$	$x+y-x.y$
Lukasiewicz, Giles (1976)	$\max(x+y-1,0)$	$\max(x+y,1)$
Weber (1983)	$\begin{cases} x, si\ y=1 \\ y, si\ x=1 \\ 0, ailleurs \end{cases}$	$\begin{cases} x, si\ y=0 \\ y, si\ x=0 \\ 0, ailleurs \end{cases}$
Hamacher (1978) $\gamma>0$	$\dfrac{x.y}{\gamma+(1-\gamma)(x+y-x.y)}$	$\dfrac{x+y-(2-\gamma)x.y}{1-(1-\gamma)x.y}$
Dubois et Prade (1986) $\alpha\in[0,1]$	$\dfrac{x.y}{\max(x,y,\alpha)}$	$\dfrac{x+y+x.y-\min(x,y,1-\alpha)}{\max(1-\alpha,1-y,\alpha)}$

Tableau 2.1. Principales t-normes et t-conormes

Complémentation : $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$

Avec : $A\cup\bar{A}\neq U$
$A\cap\bar{A}\neq\phi$

2.2.1.5. Le principe d'extension

Le principe d'extension a été introduit par L.A. Zadeh en 1975 et constitue l'un des concepts les plus importants de la théorie des ensembles flous. Il permet l'extension de référence *Y*, éventuellement identique au premier. Il est alors possible de définir un ensemble flou *B* de *Y* de la donnée d'un ensemble flou *A* de *X* :

$$\forall y \in Y, \mu_B(y) = \begin{cases} \sup_{\{x \in X | y = f(x)\}} \mu_A(x) & si \left\{ x \in X \mid y = f(x) \right\} \neq 0 \\ 0 & si\ non \end{cases}$$

2.2.3. Relations floues

Jusqu'à présent nous avons considéré des ensembles flous monodimensionnels. Lorsque ceux-ci sont multidimensionnels, leur fonction d'appartenance est aussi communément appelée relation floue.

Une relation floue *R* définie sur le produit cartésien $X_1 \times \ldots \times X_n$ est un ensemble flou (n-dimensionnel) et est noté :

$R = \int_{X_1 \times \cdots \times X_n} \frac{\mu_R(x_1, \ldots, x_n)}{(x_1, \ldots, x_n)}$ dans le cas continu.

$R = \sum_{X_1 \times \cdots \times X_n} \frac{\mu_R(x_1, \ldots, x_n)}{(x_1, \ldots, x_n)}$ dans le cas discret,

2.2.3.1. Produit cartésien

Considérons différents ensembles flous $A_1, A_2, \ldots, A_n$, respectivement sur $X_1, X_2, \ldots, X_n$. On peut définir, à partir de ces ensembles flous, un ensemble flou global multidimensionnel, $A = A_1 \times A_2 \times \ldots \times A_n$, considéré comme leur *produit cartésien*, de fonction d'appartenance :

$$\forall x = (x_1, \ldots, x_n) \in X, \mu_A(x) = \min\left(\mu_{A_1}(x_1), \cdots, \mu_{A_n}(x_n)\right)$$

2.2.3.2. Projection

De façon inverse, à partir d'un ensemble flou multidimensionnel défini sur un univers complexe, il est possible, à l'aide d'une projection mathématique, d'obtenir des informations sur chacune des différentes composantes de cet univers.

Soit un ensemble flou A défini sur un univers $X = X_1 \times X_2$. La projection de A sur X_1 est définie par :

$$\forall x_1 \in X_1, proj\left(A; X_1\right)\left(x_1\right) = \sup_{x_2 \in X_2}\left[\mu_A\left(x_1, x_2\right)\right]$$

On définit de façon analogue la projection de A sur X_2.

En particulier, si A est le produit cartésien des ensembles flous A_1 de X_1 et A_2 de X_2, ses projections $proj\left(A; X_1\right)$ et $proj\left(A; X_2\right)$ sont respectivement A_1 et A_2.

2.2.3.3. Extension cylindrique

L'extension cylindrique de l'ensemble flou B de $X_a, X_b, \ldots, X_k$, avec $1 \le a < b < \cdots < k \le n$, est l'ensemble flou $B^c = extc\left(B; X\right)$ de $X_1, X_2, \ldots, X_n$ de fonction d'appartenance :

$$\forall x = \left(x_1, \cdots, x_n\right) \in X, \mu_{B^c}\left(x\right) = \mu_B\left(x_a, \cdots, x_k\right)$$

2.2.3.4. Composition de relations floues

Soit R une relation floue définie sur $X \times Y$ et A un ensemble flou de X; l'ensemble flou B de Y est déduit de la composition de A par R :

$$\begin{aligned} B &= A \circ R \\ &= proj\left(R \cap extc\left(A, X \times Y\right); Y\right) \end{aligned}$$

En considérant l'extension cylindrique comme implicite, la composition de relations floues comporte deux phases : combinaison et projection. L.A. Zadeh propose l'opérateur de composition *sup-min* :

$$\forall y \in Y, \mu_B(y) = \sup_x \min\left(\mu_A(x), \mu_R(x,y)\right)$$

2.3. La Commande En Logique Floue

Bien que la logique floue possède un champ d'application extrêmement vaste (commande, classification, aide à la décision, base de données imprécises,…) nous nous intéressons à son utilisation dans le domaine de la commande. Cette partie a comme objectif la présentation des principes de base de la commande floue.

L'objectif d'une commande floue est de traiter des problèmes de commande de processus, le plus souvent à partir des connaissances des experts ou d'opérateurs qualifiés travaillant sur le processus. L'acquisition et l'écriture des règles font l'objet du paragraphe 2.2.1. Dans le paragraphe 2.2.3 sont citées les différentes propriétés d'une base de règles.

Les différentes étapes dans le traitement de ces règles sont exposées dans le paragraphe 2.2.3. La fuzzification, le traitement des prémisses composées, l'inférence floue, l'agrégation des règles et la défuzzification en sont les phases principales. La structure de base d'un contrôleur flou est donnée dans le paragraphe 2.2.4.

Enfin, dans le paragraphe 2.2.5, nous abordons le problème du réglage, de la stabilité et de la robustesse d'un contrôleur flou.

2.3.1. Acquisition de la connaissance et écriture de la base de règles

L'algorithme de commande consiste en une collection de règles floues appelée base de règles. Plusieurs solutions peuvent être envisagées pour obtenir ces règles :

- L'extraction des connaissances d'opérateurs humains est certainement la méthode la plus utilisée pour la commande des systèmes complexes. Elle peut être obtenue directement sous forme de règles énoncées par des

experts dans la commande du processus ou bien à partir d'un jeu de données entrées sorties représentatifs du comportement de l'opérateur humain. Dans le second cas, il s'agit de construire un modèle flou des actions de commande prises par l'opérateur [6].

- A l'image de la démarche usuelle en automatique, le contrôleur flou peut être obtenu à partir d'un modèle flou du processus (règles à conclusion polynomiale [6], modèle d'état flou [7][8]). *L'inversion* de ce modèle conduit directement à un contrôleur flou. Cependant, l'utilisation de ce modèle inverse en tant que contrôleur n'est envisageable que lorsque le système à commander est minimum de phase. Dans le cas contraire, cela conduirait à une instabilité du système en boucle fermée. L'utilisation à la fois directe et indirecte (après inversion) du modèle flou du processus conduit à des lois de commande floue par modèle interne ou prédictif [9][10].
- Les connaissances des automaticiens sur le comportement des processus vis-à-vis de certaines classes d'entrées (par exemple, les réponses temporelles) peuvent aussi être utilisées pour la réalisation de contrôleurs flous [11][12]. Il existe également des règles d'équivalence entre contrôleurs classiques de type P.I.D et les contrôleurs flous [13]. Une table de règles très connue en régulation est celle de Mac Vicar-Whelan [15]. Ces contrôleurs sont des versions floues des contrôleurs usuels.

En commande, les règles floues utilisées sont généralement de la forme :

$$\textbf{\textit{Si}}\ \underbrace{\left(X_1\ est\ A_1\right)et\left(X_2\ est\ A_2\right)}_{prémisse}\ \textbf{\textit{ALORS}}\ \underbrace{\left(Y\ est\ B\right)}_{conclusion}$$

Lorsque les règles floues sont de sémantiques plus complexes, elles peuvent facilement s'écrire sous une forme plus simple du même type que précédemment.

Ainsi, en pratique, des règles de la forme :

- *« Si X_1 est A_1 Alors (Y est B_1 Sinon Y est B_2) »* se décomposent en :

$$\begin{cases} "Si\ X_1\ est\ A_1\ Aolrs\ Y\ B_1" \\ \qquad\qquad ou \\ "Si\ X_1\ est\ \overline{A}_1\ Aolrs\ Y\ B_2" \end{cases}$$

- *« Si X_1 est A_1 Alors (Y est B_1 à moins que X_1 soit A_2) »* se décomposent en :

$$\begin{cases} "Si\ X_1\ est\ A_1\ Aolrs\ Y\ B_1" \\ \qquad\qquad ou \\ "Si\ X_1\ est\ A_2\ Aolrs\ Y\ B_1" \end{cases}$$

- *« Si X_1 est A_1 Alors (Y est B_1 Sinon (Si X_1 est A_2 Alors Y est B_2)) »* se décomposent en :

$$\begin{cases} "Si\ X_1\ est\ A_1\ Aolrs\ Y\ B_1" \\ \qquad\qquad ou \\ "Si\ X_1\ est\ \overline{A}_1\ et\ X_1\ est\ A_2\ Aolrs\ Y\ B_2" \end{cases}$$

- *« Si X_1 est A_1 Alors (Si X_1 est A_2 Alors Y est B_1) »* s'écrivent aussi :

  « Si X_1 est A_1 et X_1 est A_2 Alors Y est B_1 »

2.3.2. Propriétés des bases de règles

Une base de règles floues doit posséder certaines caractéristiques importantes afin d'assurer une commande correcte du système. La continuité, la consistance, et la complétude d'un ensemble de règles floues sont successivement définies [15].

2.3.2.1. Continuité

Définition 2.1 *Un ensemble de règles floues « Si-Alors » est continu si toutes les règles de prémisses « adjacentes » ont des conclusions « adjacentes ».*

La notion d'ensembles flous *adjacents* consiste à ordonner les ensembles flous sur leur univers de discours :

$$A_1 < A_1 < \cdots < A_{i-1} < A_i < A_{i+1} < \cdots$$

Où A_{i-1} et A_i sont adjacents au même titre que A_i et A_{i+1}. Dans le cas d'une partition floue, les ensembles flous adjacents sont ceux qui se recouvrent (leur intersection est non nulle).

Les parties prémisses des règles sont dites *adjacentes* si elles contiennent les mêmes ensembles flous pour chacune des variables en entrée sauf pour l'une d'entre elles où les ensembles flous sont adjacents. Considérons, par exemple, l'ensemble suivant :

$$R_k : Si\ x_1\ est\ A_1^k\ et\ x_2\ est\ A_2^k\ Alors\ y\ est\ B \quad (k = 1, \cdots, r)$$

Les prémisses des règles R_k et $R_{k'}, k \neq k'$, adjacentes si l'une des deux conditions suivantes est vérifiée :

1. $A_1^k = A_1^{k'}$ *et* A_2^k *et* $A_2^{k'}$ *sont adjacents*.
2. $A_2^k = A_2^{k'}$ *et* A_1^k *et* $A_1^{k'}$ *sont adjacents*.

La base de règles est dite continue si, pour chacune des règles de prémisses adjacentes, les ensembles flous en sortie B_k et $B_{k'}$ le sont aussi.

2.3.2.2. Consistance

Un ensemble de règles floues « *Si-Alors* » est consistant s'il ne contient pas de contradictions.

Définition 2.2 *un ensemble de règles floues « Si-Alors »est inconsistant s'il existe au moins deux règles de prémisses identiques mais conclusions différentes.*

Un exemple très célèbre de base de règles inconsistante est celle utilisée pour la commande d'un robot devant éviter un obstacle, dans laquelle deux règles contradictoires coexistent :

$$\begin{cases} Si\ obstacle\ en\ face\ Alors\ aller\ à\ gauche, \\ Si\ obstacle\ en\ face\ Alors\ aller\ à\ droite. \end{cases}$$

En commande, si la partie prémisse des règles floues contient le connecteur « ***ET*** », il est très rare de trouver deux règles de prémisses identiques. Cependant, ce problème de consistance de la base de règles peut apparaître lorsque les connecteurs utilisés en partie prémisse sont des « ***OU*** » ou des opérateurs de complémentation [15].

2.3.2.3. Complétude

Définition 2.3 *un ensemble de règles floues « Si-Alors » est complet si, quelle que soit la combinaison dans l'espace d'entrée, il existe une valeur de commande.*

Une base de règles est incomplète s'il existe une situation de l'espace d'entrée pour laquelle aucune règle n'est activable. L'incomplétude de la base de règles peut conduire à une discontinuité indésirable dans la loi de commande.

Soit la base de règles suivante :

$$R_k : Si\ x_1\ est\ A_1^k\ et\ x_2\ est\ A_2^k\ Alors\ y\ est\ B \quad (k = 1, \cdots, r)$$

Une mesure de complétude *(C)* de la base de règles est donnée par [15] :

$$C(\underline{x}) = \sum_{k=1}^{r} \left\{ \prod_{i=1}^{n} \mu_{A^k}(x_i) \right\}$$

Où $\underline{x} = (x_1, x_2, \cdots, x_n)$ est le vecteur d'entrée.

- Si $C(\underline{x})=0$, la base de règles est *incomplète.*
- Si $0<C(\underline{x})<1$, la base de règles est *sous complète.*
- Si $C(\underline{x})=1$, la base de règles est *strictement complète.*
- Si $C(\underline{x})>1$, la base de règles est *sur complète.*

La sous complétude, la complétude stricte et la sur complétude sont divers degré de complétude de la base de règles.

La complétude d'une base de règles est stricte si :

1. pour chacune des variables d'entrée en partie prémisse, les ensembles flous correspondants forment une partition floue,
2. la base de règles énumère toutes les combinaisons possibles en partie prémisse.

2.3.3. Les différentes étapes de la commande floue

Un contrôleur flou est un système à base de connaissances particulier, utilisant un raisonnement en profondeur limité, dans une procédure de chaînage avant règles (activation des règles par les prémisses). Toutes les règles activables (prémisse caractérisée par un degré d'appartenance non nul) sont activées. On considère ensuite une « moyenne » sur le résultat de ces règles pour engendrer une décision finale.

On peut distinguer plusieurs étapes dans le traitement des règles. Un schéma représentatif peut être le suivant :

Les variables caractéristiques du système à commander et les consignes définissent les variables d'entrée du contrôleur flou. Les variables caractéristiques sont en général, les grandeurs de sortie du processus et, le cas échéant, d'autres mesures déterminantes pour saisir l'évolution dynamique du processus. Les variables de sortie du contrôleur flou sont les commandes à appliquer au processus.

La base de connaissances est composée d'une base de données et d'une base de règles.

La base de données regroupe :

- les ensembles flous associés aux variables d'entrée et de sortie du contrôleur flou,

- les facteurs d'échelle (gains) en entrée (normalisation) et en sortie (dénormalisation).

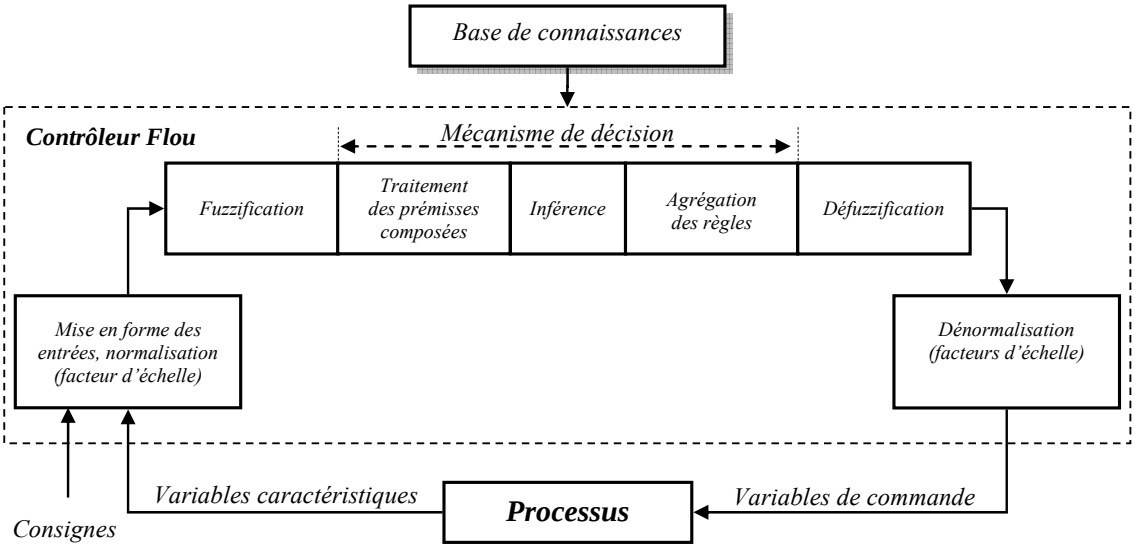

Figure 2.6. *Structure de base d'un contrôleur flou.*

La base de règles contient des règles de la forme :

$$\text{« } SI\left(X_1 \text{ est } A_1 \text{ et } X_2 \text{ est } A_2\right) \text{ ALORS } \left(Y \text{ est } B\right) \text{»}$$

X_1, X_2, et Y sont des grandeurs physiques caractéristiques du système et du problème de commande. A_1, et A_2 sont des labels linguistiques.

Suivant la nature de B on parlera de :

- règles à conclusion symbolique (contrôleur de type Mamdani) : B est une valeur linguistique. Exemple : *Si l'erreur est « Négatif Moyen » et la variation de l'erreur » est « Positif Petit » Alors la commande est « Négatif Petit ».*
- règles à conclusion algébrique (contrôleur de Sugeno) : B est une valeur numérique (singleton) ou une équation mathématique bien précise (non floue). Exemple : *Si l'erreur est « Négatif Moyen » et la variation de l'erreur » est « Positif Petit » Alors la commande est -0.3.* Lorsque B est une valeur numérique on parle de règles de Takagi-Sugeno « d'ordre zéro », sinon, de règles à conclusion polynomiale.

Bien que les sorties des contrôleurs flous de type Sugeno soient généralement des fonctions non linéaires statiques de leurs entrées, il ne faut pas oublier de mentionner les contrôleurs dits « flous dynamiques » de Sugeno, où B est un modèle dynamique, certain ou incertain, à temps continu ou discret. L'utilisation de tels contrôleurs permet d'étendre certains résultats de l'automatique classique à la commande floue.

On distingue classiquement trois parties dans la structure d'un contrôleur flou : la fuzzification, le mécanisme de décision et la défuzzification.

- La fuzzification est l'étape qui permet de transformer une grandeur mesurée sur le processus en un ensemble flou.
- Le mécanisme de décision permet de calculer l'ensemble flou associé à la commande.
- La défuzzification est l'étape qui permet de transformer l'ensemble flou, obtenu par le calcul précédent, en une grandeur de commande à appliquer au processus.

Les opérations de normalisation et de dénormalisation sont des étapes optionnelles.

Nous allons maintenant revenir en détail sur les différentes étapes dans le traitement des règles.

2.3.3.1. Mise en forme des entrées, normalisation

Cette première étape permet le traitement des variables d'entrée du contrôleur flou, par exemple, calcul d'erreurs et de variations d'erreurs. L'utilisation de domaines normalisés (univers de discours compris entre -1 et 1) nécessite une transformation d'échelle transformant les grandeurs physiques des entrées en des valeurs normalisées appartenant à l'intervalle [-1, 1].

2.3.3.2. Fuzzification

C'est l'opération de *projection* de variables physiques réelles sur des ensembles flous caractérisant les valeurs linguistiques prises par ces variables. Deux cas peuvent se présenter selon que la mesure d'une variable physique réelle est précise (valeur numérique) ou pas (issue par exemple d'un capteur flou) :

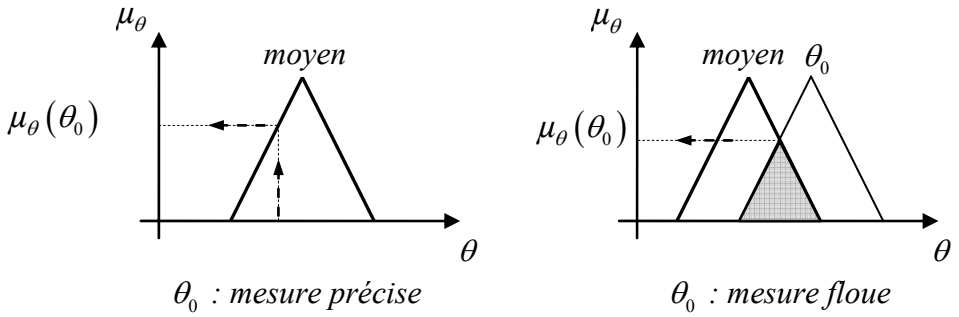

Figure 2.7. *Conversion numérique symbolique.*

Le choix de la forme des fonctions d'appartenance (triangulaires, trapézoïdales, exponentielles, gaussiennes,…) est arbitraire. Des études comparatives ont montré, selon différentes formes de fonctions d'appartenance, des résultats pratiquement similaires en boucle fermée mais les formes triangulaires facilitent la programmation ce qui explique qu'elles soient le plus fréquemment utilisées. Quant au nombre de fonctions d'appartenance, il est généralement impair car elles se répartissent autour de zéro. Un exemple de fonctions d'appartenance triangulaires est donné dans la figure 2.8.

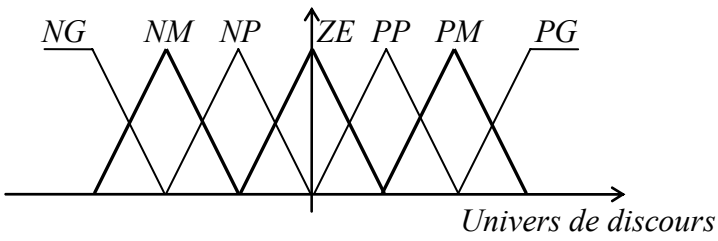

Figure 2.8. *Exemple de fonctions d'appartenance.*

NG, NM,…, PG sont des valeurs linguistiques, avec :

- *NG : « Négatif Grand »*

- *NM : « Négatif Moyen »*
- *NP : « Négatif Petit »*
- *ZE : « Zéro »*
- *PP : « Positif Petit »*
- *PM : « Positif Moyen »*
- *PG : « Positif Grand »*

2.3.3.3. Traitement des prémisses composées

En général, les prémisses des règles vont comporter plusieurs clauses liées par des connecteurs *« ET »*, *« OU »* et *« NON »*. Dans la pratique, pour les opérations de conjonction et de disjonction, on a souvent recours, parmi les normes et conormes triangulaires, aux opérateurs *min* et *max*. quant à la négation A^C d'un ensemble flou *A*, elle est caractérisée par $\mu_{A^C}(x) = 1 - \mu_A(x)$.

Des modificateurs linguistiques peuvent aussi être utilisés dans l'écriture des règles, par exemples :

$$\mu_{Peu\ A} = \sqrt{\mu_A}$$
$$\mu_{Très\ A} = \mu_A^2$$

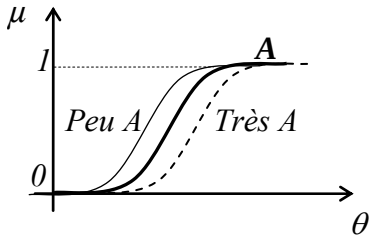

Figure 2.9. *Exemple de modificateurs linguistiques.*

2.3.3.4. Inférence floue

Elle repose sur l'utilisation d'un opérateur d'implication, lequel permet d'évaluer un degré de vérité d'une règle *R* de la forme *« Si X est A alors Y est B »*. En d'autres termes, cet opérateur quantifie la force de la liaison entre la prémisse et la conclusion de la règle.

Il existe de nombreux et différents opérateurs d'implication selon l'interprétation logique que l'on donne à l'implication *« A implique B »* ($A \rightarrow B$). On distingue ainsi l'implication classique où *« A implique B »* est définie par *« Non A ou B »* $\left(\overline{A} \text{ ou } B\right)$ de l'implication dite conjonctive où *« A implique B »* est définie par *« A et B »* $\left(A \cap B\right)$.

Les opérateurs les plus courants en commande sont de type conjonctif :

$$\begin{cases} l'implication\ de\ Mamdani\ (1974) : \mu_R(x,y) = \min\left(\mu_A(x), \mu_B(y)\right) \\ l'implication\ de\ Larsen\ (1980) : \mu_R(x,y) = \mu_A(x).\mu_B(y) \end{cases}$$

Soit la règle : *« Si la cylindrée du véhicule est importante, alors sa consommation en carburant est élevée »*. Cette règle doit pouvoir être utilisée pour un véhicule donné dont on connaît précisément la cylindrée, qui n'est pas forcément typique de la caractérisation *« importante »*, et doit également fournir une conclusion relative à sa consommation en carburant si la cylindrée du moteur n'est que relativement importante, par exemple. Dans la logique classique, le modus ponens ne permettrait pas d'obtenir une conclusion que si l'on savait exactement que la cylindrée du véhicule considérée est importante. Il faut donc modifier le modus ponens pour atteindre la souplesse de raisonnement souhaitée.

Le modus ponens trouve son équivalent flou dans le cadre du raisonnement approximatif sous la forme du *modus ponens généralisé* que nous présentons formellement ci-après.

	Logique classique	Logique floue
Fait	*X est A*	X est A′
Règle	*Si X est A alors Y est B*	*Si X est A alors Y est B*
Déduction	*Y est B*	*Y est B′*

La logique floue permet ainsi de déduire un nouveau fait *B'* caractérisé par :

$$\mu_{B'}(y) = \sup_{x \in X} T\left(\mu_{A'}(y), \mu_R(x,y)\right)$$

Pour une t-norme *T* appelé *opérateur de modus ponens généralisé.*

En commande, l'opérateur de Zadeh est couramment utilisé :

$$\mu_{B'}(y) = \sup_{x \in X} \min\left(\mu_{A'}(x), \mu_R(x,y)\right)$$

Si *A'* est un singleton x_0, alors : $\mu_{B'}(y) = \sup_{x \in X} \min\left(\mu_{A'}(x), \mu_R(x,y)\right) = \mu_R(x_0, y)$

Pour l'implication de Mamdani, on obtient : $\mu_{B'}(y) = \min\left(\mu_A(x_0), \mu_B(y)\right)$.

2.3.3.5. Agrégation des règles

Selon le type de l'implication, classique ou conjonctive, l'opérateur utilisé pour agréger les règles est, respectivement, de type conjonctif ou disjonctif. Ainsi, en commande, l'implication étant généralement de type conjonctif, cela revient à considérer que les règles sont liées par un opérateur *OU*. En pratique, on utilise l'opérateur *max* :

$$\mu_{B'}(y) = \max_{i=1 \text{ à } N}\left(\mu_{B'_i}(y)\right)$$

Avant de décrire la prochaine étape consistant à transformer le résultat de l'agrégation des règles en une valeur précise de commande, considérons d'abord un exemple classique illustrant le raisonnement de Mamdani.

L'exemple illustré par la figure 2.10 considère les deux règles suivantes :

R_1 : Si (x_1 est ZE et x_2 est ZE) alors y est ZE

R_2 : Si (x_1 est PP et x_2 est PP) alors y est PP

La méthode de Mamdani repose sur l'utilisation de l'opérateur *min* pour la combinaison des prémisses et pour l'implication. Chaque règle est activée séparément et les conclusions sont agrégées pour définir l'ensemble flou associé à la variable de sortie *y*. l'agrégation des règles est réalisée par l'opérateur *max*.

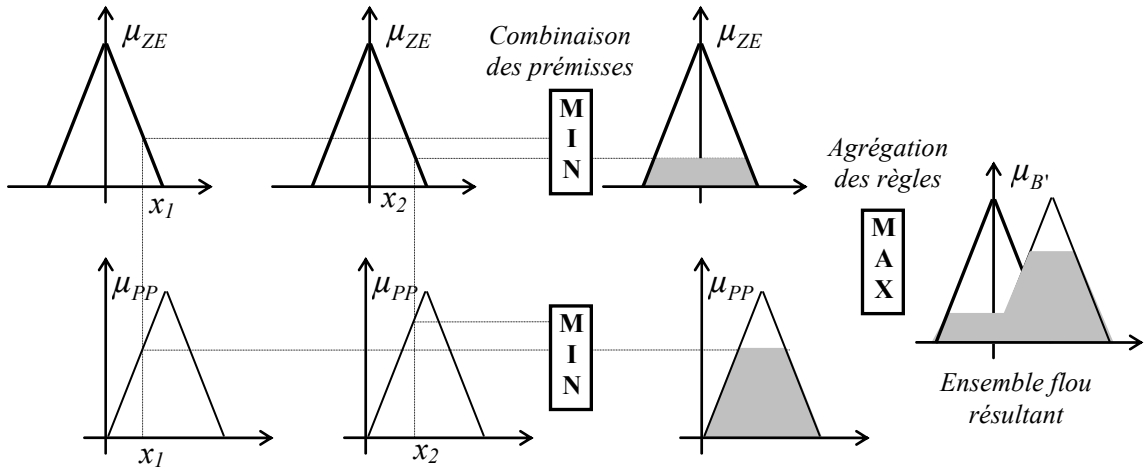

Figure 2.10. *Illustration de la méthode de Mamdani.*

2.3.3.6. Défuzzification

La défuzzification consiste à transformer l'ensemble flou résultant de l'agrégation des règles en une grandeur de commande précise. Là aussi il existe plusieurs méthodes [16], parmi lesquelles :

- La méthode de la hauteur,
- Le premier des maxima,
- Le dernier des maxima,
- La moyenne des maxima,
- Le centre de gravité,
- Le centre des aires,
- Le centre de la plus grande surface,
- Le centre des maxima.

Les méthodes de défuzzification les plus utilisées en commande floue sont le centre de gravité, le centre des aires et le centre des maxima.

La méthode de la hauteur et ses variantes

La méthode de la hauteur consiste à choisir comme grandeur de commande la valeur du maximum.

$$u = \arg\left\{\max \mu_{B'}(y)\right\}$$

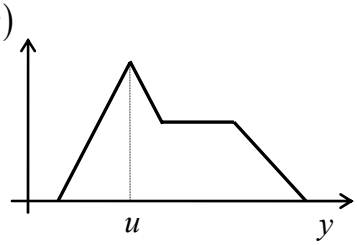

Figure 2.11. *Défuzzification par la méthode des hauteurs.*

Premier des maxima : $\overline{y}_1$

Dernier des maxima : $\overline{y}_2$

Moyenne des maxima : $u = \dfrac{\overline{y}_1 + \overline{y}_2}{2}$

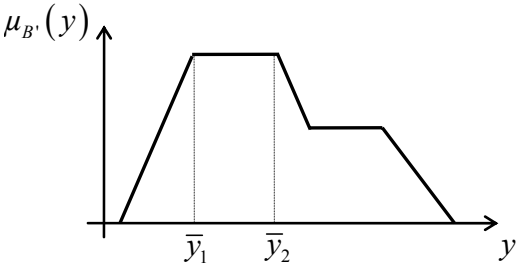

Figure 2.12. *Variantes de la méthode des hauteurs.*

Dans le cas où la fonction d'appartenance $\mu_{B'}(y)$ a plus d'un maximum, on a le choix entre le premier, et le dernier ou la moyenne des maxima. La moyenne des maxima est similaire à la méthode du premier des maxima et consiste tout simplement à en faire la moyenne.

Ces méthodes nécessitent peu de calculs mais peuvent introduire des discontinuités dans la loi de commande, ce qui explique pourquoi elles sont si peu utilisées en commande floue.

Le centre de gravité

$$u = \frac{\int_y \mu_{B'}(y).y.dy}{\int_y \mu_{B'}(y).dy}$$

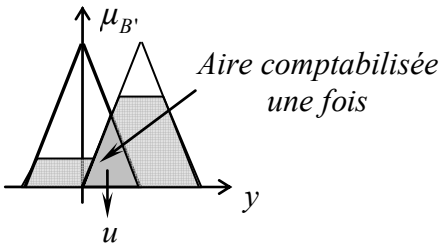

Figure 2.13 *Défuzzification par un centre de gravité.*

C'est la méthode de défuzzification la plus connue en commande floue. Cette méthode fournit intuitivement la valeur la plus représentative de l'ensemble flou issu de l'agrégation des règles. C'est aussi la méthode la plus coûteuse en temps de calcul.

Le centre des aires (centre des sommes)

Cette méthode est similaire à la précédente mais ne nécessite pas le calcul de $\mu_{B'}(y)$. L'idée est de considérer la contribution de chaque aire individuellement. L'ensemble *B'* est alors construit à partir de la somme de chaque aire. Ainsi, les aires qui se recouvrent, si elles existent, sont comptabilisées plus d'une fois.

$$u = \frac{\int_y y.\sum_{i=1}^{n} \mu_{B'}(y).dy}{\int_y \sum_{i=1}^{n} \mu_{B'}(y).dy}$$

Avec n : nombre d'aires

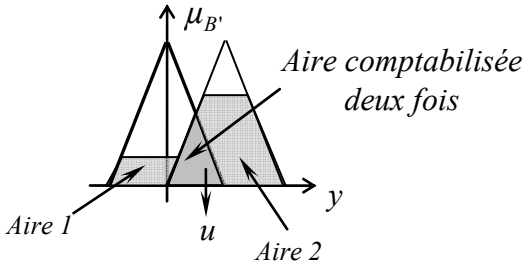

Figure 2.14. *Défuzzification par un centre des aires.*

Le centre de la plus grande surface

Cette méthode consiste à trouver le centre de gravité de la plus grande surface.

Le centre des maxima

Cette méthode considère le maximum de chacune des contributions et en fait la moyenne pondérée.

$$u = \frac{\sum_{i=1}^{n} \mu_{B'_i}(\overline{y}_i).\overline{y}_i}{\sum_{i=1}^{n} \mu_{B'_i}(\overline{y}_i)}$$

Avec n : nombre d'aires

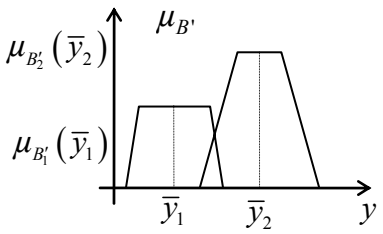

Figure 2.15. *Défuzzification par un centre des maxima.*

Un cas particulier : méthode de Sugeno

Dans le cas de règles à conclusions polynomiales du type :

$$R_i : Si\left(x_1 \; est \; A_1^i \; et \; x_2 \; est \; A_2^i \; et \cdots x_n \; est \; A_n^i\right) Alors\left(y \; est \; f\left(x_1, x_2, \cdots, x_n\right)\right)$$

La commande *u* est obtenue par une simple moyenne pondérée selon les niveaux d'activation α_i de chacune des règles $R_i\left(i=1,\cdots,r\right)$:

$$u=\frac{\sum_{i}^{r}\alpha_i.f\left(x_1,x_2,\cdots,x_n\right)}{\sum_{i}^{r}\alpha_i}$$

Avec : $\alpha_i=T\left(\mu_{A_1'}\left(x_1\right),\mu_{A_2'}\left(x_2\right),,\mu_{A_n'}\left(x_n\right)\right)$

Pour la t-norme *T*, on choisit très souvent l'opérateur *min* ou le *produit*.

2.3.3.7. Dénormalisation

Cette dernière étape transforme les valeurs normalisées des variables de commande en des valeurs appartenant à leur domaine physique respectif.

2.3.4. Structure de base d'un contrôleur flou : analogie structurelle avec les P.I.D

Pour des problèmes de régulation monovariable simples, les entrées du contrôleur flou sont généralement l'erreur *e* et la variation de l'erreur *Δe*.

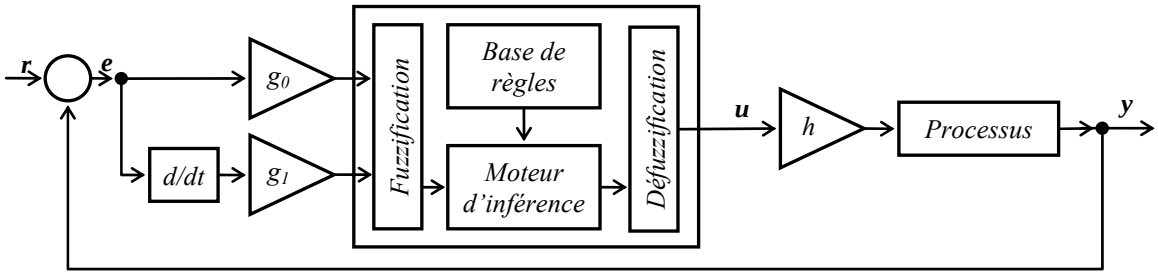

Figure.2.16. *Structure d'un contrôleur flou.*

g_0, g_1, et *h* sont des gains ou facteurs d'échelle.

Selon que la sortie du contrôleur flou concerne la commande ou sa variation. On obtiendra des équivalents structuraux non linéaires des contrôleurs classiques P.D, P.I, ou même des P.I.D, en augmentant le nombre d'entrées des contrôleurs.

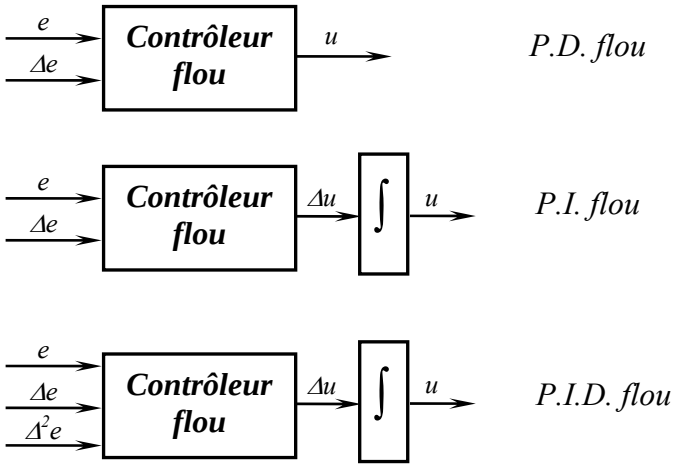

Figure 2.17. *Equivalents flous des contrôleurs usuels.*

2.3.5. Réglage, stabilité et robustesse d'un contrôleur flou

2.3.5.1. Réglage

Un contrôleur flou possède de nombreux paramètres de réglage, ce qui peut, à priori, effrayer ses utilisateurs potentiels. En effet, contrairement aux contrôleurs classiques, le contrôleur flou possède un nombre plus conséquent de paramètres, et offre, par la même, davantage de degrés de liberté.

On peut distinguer parmi les choix et les réglages à faire :

- L'expression des règles,
- La définition des variables et des valeurs linguistiques, avec leurs fonctions d'appartenance associées,
- La méthode d'implication,
- La méthode d'inférence,
- La méthode de défuzzification,
- Les facteurs d'échelle sur les entrées et les sorties du contrôleur.

Cependant, on constate une certaine insensibilité du résultat au choix des méthodes d'implication, d'inférence et de défuzzification. Pour les autres paramètres, un réglage séquentiel est possible :

- Facteurs d'échelle,
- Fonctions d'appartenance,
- Règles.

Le réglage par essais successifs de ces nombreux paramètres étant assez long et fastidieux, diverses techniques d'autoréglage, d'optimisation et d'apprentissage ont été développées ces dernières années. On peut citer, à titre d'exemple, les techniques de la programmation mathématique, les réseaux de neurones, les algorithmes génétiques.

2.3.5.2. Stabilité

L'analyse de la stabilité est à priori difficile dans une approche de type *« système à base de connaissances »* dont l'objectif est d'éviter l'utilisation de modèle mathématique du procédé. Une modélisation floue du système bouclé est alors nécessaire pour l'analyse de la stabilité. Toutefois, si une caractérisation mathématique du processus à commander existe, le contrôleur flou étant, en général, un contrôleur non linéaire, certains résultats de la théorie des systèmes non linéaires peuvent être utilisés. On distingue alors deux grandes familles d'approches :

- *La méthode directe de Lyapunov :* cette méthode permet d'affirmer la stabilité asymptotique d'un état d'équilibre s'il existe une fonction de Lyapunov $V(x,t)$ définie positive telle que sa dérivée $\frac{dV(x,t)}{dt}$ soit définie négative.
- *L'approche entrée sortie :* cette approche permet d'affirmer la stabilité d'un système si la sortie de celui-ci, en réponse à une entrée bornée, est elle-même bornée.

Nous choisissons ici de distinguer les différentes méthodes d'analyse de la stabilité selon que l'on dispose du modèle mathématique du processus ou pas. La

liste des méthodes ici citées est non exhaustive, on peut se référer à [17, 18] pour en avoir un aperçu plus complet.

Il n'existe pas de modèle mathématique du processus

Dans ce cas, le système en boucle fermée est modélisé par un système flou. Les méthodes d'analyse de la stabilité dépendent alors du type du modèle.

- Si le modèle du système en boucle fermée est un modèle symbolique (de type Mamdani ou modèle aux relations floues) du type temps discret, la stabilité n'est pas étudiée au sens classique du terme et de nouvelles définitions de la stabilité sont introduites. On parle alors de la stabilité d'un symbole flou [19].
- Dans le cas où le système en boucle fermée est modélisé par un système flou de type Sugeno, des propriétés de stabilité asymptotique peuvent être démontrées à partir de l'utilisation de fonctions de Lyapunov quadratiques [20].
- Dans le cas ou le modèle du système en boucle fermée est un modèle hybride [21] on utilise également des fonctions de Lyapunov quadratiques pour étudier la stabilité. Les modèles linguistiques et modèles de Sugeno sont deux cas particuliers de modèles hybrides.

Il existe un modèle mathématique du processus à commander

Dans ce cas, on peut faire appel à diverses méthodes d'analyse de la stabilité issues de la théorie des systèmes non linéaires. Ces méthodes diffèrent selon que le contrôleur flou a une expression mathématique simple ou pas.

- Si le contrôleur flou peut facilement se mettre sous forme analytique [22] l'on est ramené à l'étude de la stabilité asymptotique d'un système en boucle fermée non linéaire et on peut alors utiliser la méthode directe de Lyapunov.

- Pour des contrôleurs flous plus complexes, l'expression mathématique étant inexploitable, on doit alors se contenter d'une propriété de secteur géométrique [23] : le contrôleur flou est considéré comme une fonction mathématique non linéaire évoluant dans un secteur géométrique bien défini. On cherchera alors les conditions de stabilité absolue correspondantes à ce secteur, lesquelles peuvent s'exprimer soit dans un formalisme d'état soit dans un formalisme entrée sortie [24].

Les conditions les plus générales de stabilité peuvent être obtenues sans aucune hypothèse sur la nature du système (linéaire, non linéaire, variant, invariant, de dimension finie ou infinie). Cependant, bien que générales, ces conditions sont uniquement suffisantes et difficilement exploitables. C'est pourquoi le système est généralement supposé linéaire invariant de dimension finie bouclé par une non linéarité (ici, contrôleur flou). Selon les propriétés géométriques de cette non linéarité, différents critères de stabilité sont obtenus : critère du cercle, critère de conicité, lemme borné réel, lemme positif réel.

2.3.5.3. Robustesse

La robustesse, bien que maintes fois constatée, ne possède aucune démonstration rigoureuse. Son explication est à chercher dans l'aspect non linéaire de cette commande qui peut apparaître comme un contrôleur de type P.I.D à gains variables ou préprogrammés, selon le point de fonctionnement. Toutefois, si le modèle du système en boucle fermée est un modèle « *flou dynamique* » de Sugeno, on peut se reporter aux travaux de l'automatique classique.

2.4. Conclusion

Dans ce chapitre, nous nous sommes intéressés à la logique floue et à l'utilisation de la logique floue en commande. L'accent a particulièrement été mis sur les différentes étapes dans le traitement des règles d'un contrôleur flou.

Retenons, que l'intérêt majeur de la logique floue en commande réside dans sa capacité à traduire une stratégie de contrôle d'un opérateur qualifié en un ensemble de règles linguistique facilement interprétables.

Nous disposons maintenant des notions élémentaires relatives à la commande en logique floue. Dans le chapitre suivant, on abordera la conception d'un contrôleur flou de type Mamdani de telle façon à limiter le temps de calcul, et on le comparera avec un contrôleur conçu selon des modifications proposées.

Chapitre 3
La Conception D'un Contrôleur Flou

3.1. Introduction

Comme l'on a vu dans le deuxième chapitre, la logique floue [26] qui est la logique sur laquelle est basée la commande floue, est plus proche en son esprit au raisonnement humain et au langage naturel que les systèmes logiques traditionnels. Elle fournit principalement des moyens efficaces pour la capture de la nature approximative et inexacte du monde réel. La partie la plus importante d'un contrôleur flou est un ensemble de règles de commande reliées par les concepts d'implication et de composition floues, et des règles d'inférence floue. De ce fait, un contrôleur flou représente un algorithme qui peut convertir une stratégie formelle (linguistique) de commande basée sur les connaissances d'un expert en une stratégie automatique de commande. L'expérience a montré que la commande floue donne de meilleurs résultats que les algorithmes de commande conventionnelle.

Cependant, il n'existe pas de procédure systématique pour la conception de contrôleurs flous, cela est dû à la grande diversité des choix des paramètres et d'opérateurs dans l'algorithme de commande floue. Le concepteur doit choisir :

- Une stratégie de fuzzification,
- Une méthode pour l'élaboration de la base de règles,
- Une logique de prise de décision, et
- Une stratégie de défuzzification.

Dans ce chapitre, nous présenterons dans la section 3.1 une méthode de conception d'un contrôleur flou de type Mamdani. Nous avons choisi de le rendre le plus simple et le plus rapide possible pour des raisons d'optimisation du temps de traitement. Cela est motivé par la conservation de l'aspect temps réel de notre contrôleur. Dans la section 3.2, nous présenterons notre contribution dans ce domaine que nous avons appelé la logique classique modifiée, qui vise la minimisation du temps de traitement dans les algorithmes de commande basés sur les connaissances. Nous expliquerons la procédure de conception de notre

contrôleur, qui est basé sur un système à logique classique utilisant des ensembles classiques de fonctions d'appartenance rectangulaires. La base de règles est la même utilisée pour le contrôleur flou. Nous avons introduit deux modifications sur le raisonnement classique pour le rendre plus souple. Dans la section 3.3, nous comparerons les résultats d'applications des deux algorithmes pour la commande d'un système du deuxième ordre pour comparer les performances des contrôleurs conçus.

3.2. Conception D'un Contrôleur Flou

La procédure générale de conception de contrôleurs flous [21] comporte les étapes montrées sur la figure 3.1. Comme on le remarque les retours à des étapes précédentes interviennent à plusieurs niveaux. En particulier, on a mis en évidence l'étape de modification qui doit être primordiale à plusieurs reprises.

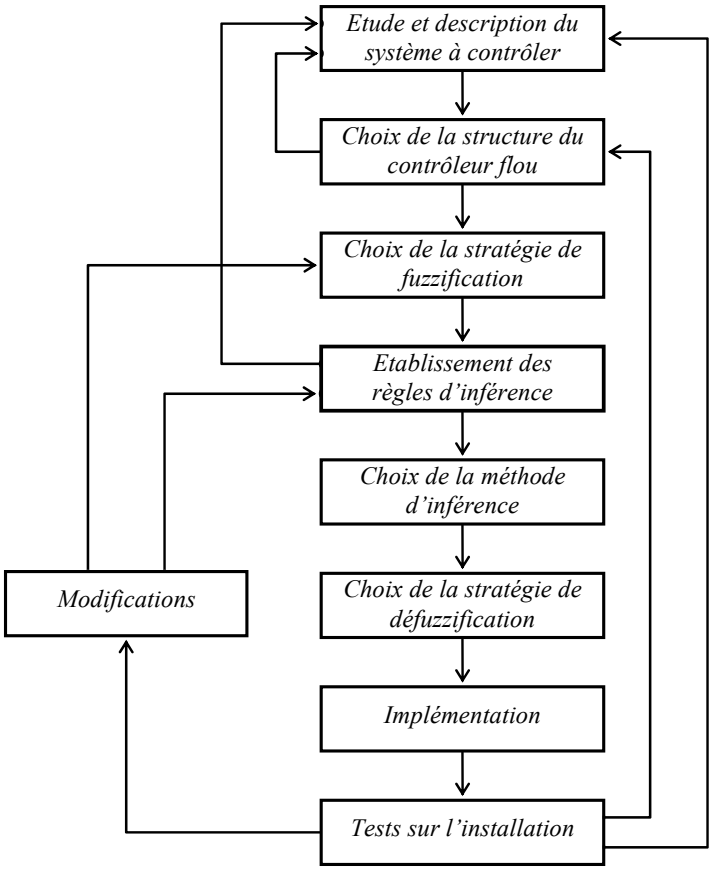

Figure.3.1. *Procédure de conception de contrôleur flou.*

Cette procédure de conception de régulateurs flous présente l'avantage qu'il ne faut pas établir un modèle pour le système à régler. Par contre, l'inconvénient est qu'il faut faire beaucoup de tests sur l'installation réelle, ce qui nécessite du temps et beaucoup de précautions. Malgré tout, le temps de développement total peut être inférieur à celui nécessaire à la conception avec établissement de modèle.

3.2.1. Choix de la structure du contrôleur

Comme nous l'avons déjà mentionné, nous avons choisi la structure du contrôleur proposée par Mamdani, qui est un contrôleur à deux entrées : l'erreur et la variation de l'erreur. La sortie du contrôleur est la commande au processus (figure 2.16).

3.2.2. Choix de la stratégie de fuzzification

Il s'agit du choix des formes des fonctions d'appartenance, et leur répartition sur l'univers de discours. Dans notre cas, nous avons choisi de travailler sur des univers de discours normalisé [-1, 1], partitionnés en cinq classes (sous ensembles flous) pour les entrées et pour la sortie. La forme la plus utilisée en commande floue est la forme triangulaire, pour cela nous utiliserons des fonctions d'appartenance triangulaires. La figure 3.2 montre un exemple de partition flou de l'univers de discours.

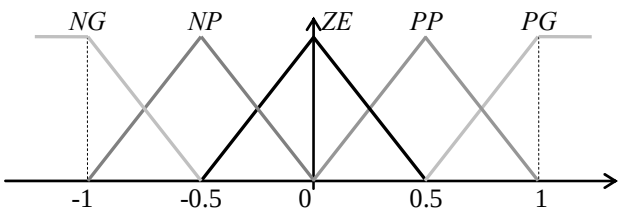

Figure.3.2. *Partition des entrées et de la sortie du contrôleur flou.*

Les fonctions d'appartenance sont définies comme suit :

La fonction NG :

$$\mu_{NG}(x)=\begin{cases}-2t-1 & si\ -1\le x\le -0.5\\ 0 & ailleurs\end{cases}$$

La fonction NP :

$$\mu_{NP}(x)=\begin{cases}2t+2 & si\ -1\le x\le -0.5\\ -2t & si\ -0.5\le x\le 0\\ 0 & ailleurs\end{cases}$$

La fonction ZE :

$$\mu_{ZE}(x)=\begin{cases}2t+1 & si\ -0.5\le x\le 0\\ -2t+1 & si\ 0\le x\le 0.5\\ 0 & ailleurs\end{cases}$$

La fonction PP :

$$\mu_{PP}(x)=\begin{cases}2t & si\ 0\le x\le 0.5\\ -2t+2 & si\ 0.5\le x\le 1\\ 0 & ailleurs\end{cases}$$

La fonction PG :

$$\mu_{PG}(x)=\begin{cases}2t-1 & si\ 0.5\le x\le 1\\ 0 & ailleurs\end{cases}$$

3.2.3. Etablissement des règles d'inférence

Sur la base de la description du système à régler avec des variables linguistiques et de la définition des fonctions d'appartenance pour les variables d'entrée et de sortie, on peut établir les règles d'inférence.

Nous avons utilisé une base de règle que l'on trouve dans la plupart des applications où la dynamique du système étudié présente une certaine symétrie autour d'un certain état stable. La table 3.1 représente la table des règles retenue dans notre étude.

		La variation de l'erreur				
		NG	*NP*	*ZE*	*PP*	*PG*
L'erreur	*NG*	*NG*	*NG*	*NG*	*NP*	*ZE*
	NP	*NG*	*NG*	*NP*	*ZE*	*PP*
	ZE	*NG*	*NP*	*ZE*	*PP*	*PG*
	PP	*NP*	*ZE*	*PP*	*PG*	*PG*
	PG	*ZE*	*PP*	*PG*	*PG*	*PG*

Table 3.1. *La table des règles.*

3.2.4. Choix de la méthode d'inférence

Les opérateurs *min* et *max* tels qu'ils sont définis par Zadeh sont utilisés pour inférer les règles, et l'opérateur *max* pour l'agrégation des règles.

3.2.5. Choix de la stratégie de défuzzification

La méthode la plus utilisée en commande flou est celle du centre de gravité, mais comme elle est très coûteuse en terme de temps de calcul, et le but de notre travail était de minimiser le temps de calcul, nous avons opté pour la méthode des centre des aires, qui est dérivée de la méthode du centre de gravité mais qui nécessitent peu de calculs.

La surface de décision (contrôle) du contrôleur conçu est donnée sur la figure 3.3. Elle représente la non linéarité et la souplesse de la commande floue.

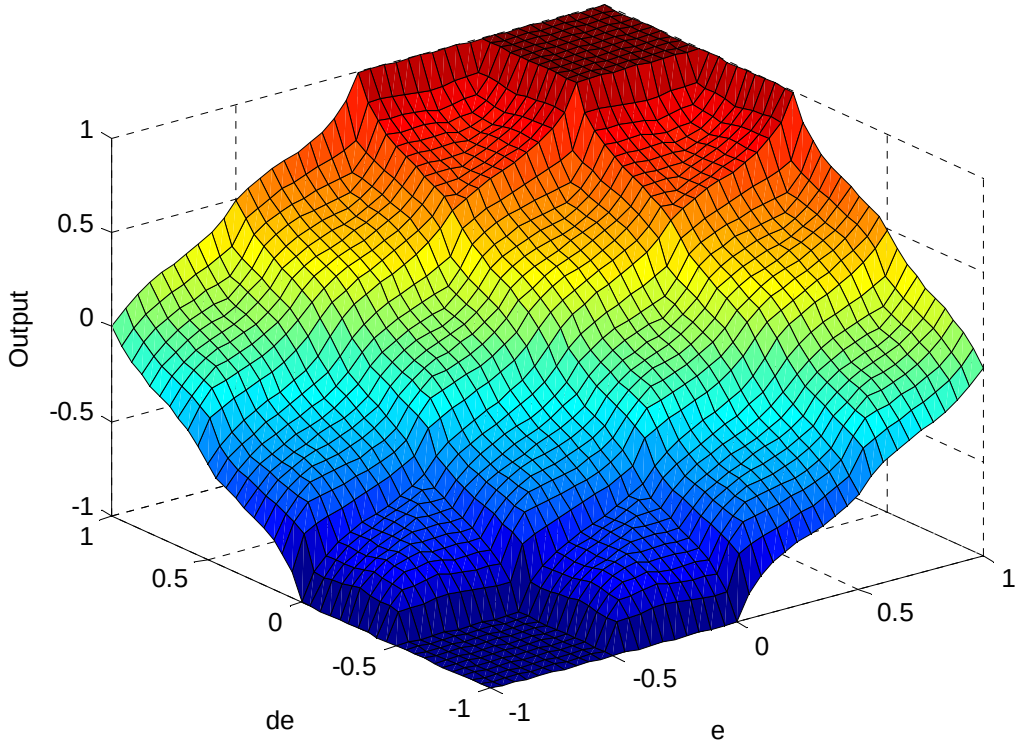

Figure.3.3. *La surface de décision du contrôleur flou.*

3.3. Notre contribution : La Logique Classique Modifiée.

Un des problèmes importants qui se posent dans les applications de la logique floue pour la commande de processus est le temps nécessaire à la prise de décision. Or, pour les applications temps réel le facteur temps est crucial, nous avons pensé à une méthode pour essayer d'optimiser le rapport temps de traitement et performances du contrôleur flou. Dans la présente section, nous proposons d'apporter deux modifications au raisonnement flou pour minimiser le temps de prise de décision. Une pour le choix des fonctions d'appartenance, et l'autre pour le calcul de la valeur de sortie.

3.3.1. Les modifications proposées

Nous proposons ici deux modifications pour la simplification des traitements dans un contrôleur flou. La première concernant le choix des fonctions d'appartenance d'entrées et de sortie. Nous utilisons pour les entrées des fonctions d'appartenance rectangulaires adjacentes (figure 3.4) uniformément

réparties sur l'univers de discours. Cela va simplifier les étapes de fuzzification et d'inférence, nous n'aurons plus de calculs à faire pour déterminer le degré d'appartenance des variables d'entrée aux différentes classes nettes. Par conséquent, nous n'aurons plus à inférer plusieurs règles, la connaissance des classes auxquelles appartiennent les variables d'entrées va directement tirer conclusion sur la sortie. Et nous utiliserons des singletons (impulsions) pour la partition de la variable de sortie. Ce qui va simplifier la défuzzification.

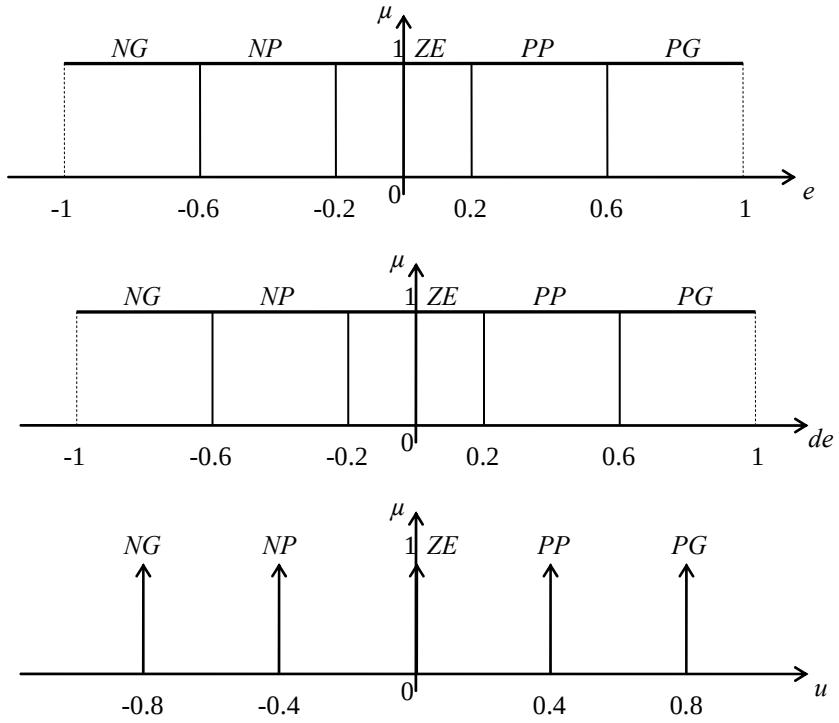

Figure.3.4. *Exemple de partitions nettes des univers de discours.*

La deuxième modification que nous proposons, est au niveau du calcul de la sortie, elle vise l'adoucissement des transitions brusques dûes au choix des fonctions d'appartenance. Elle consiste à calculer les différences entre les valeurs des variables d'entrées et les centres des classes auxquelles elles appartiennent. La valeur de sortie sera calculée selon la formule suivante :

$$u = règle(i, j) - \frac{(\Delta e + \Delta de)}{2}$$

Où :

i : est le numéro de la classe à laquelle appartient *e*.

j : est le numéro de la classe à laquelle appartient *de*.

Δe : est la différence entre la valeur de l'erreur *e* et le centre de la classe à laquelle elle appartient : $\Delta e = c(i) - e$.

C(i) : est le centre de la classe numéro *i* sur la partition de la variable erreur *e.* On attribue un code (numéro) à chaque classe.

Δde : est la différence entre la valeur de la variation de l'erreur de et le centre de la classe à laquelle elle appartient : $\Delta de = c(j) - de$.

C(j) : est le centre de la classe numéro *j* sur la partition de la variable variation de l'erreur *de.*

règle (i, j) : est le singleton de sortie contenu dans la cellule *(i, j*) dans la table des règles.

Le contrôleur aura par conséquent la structure montrée sur la figure 3.5. Nous remarquons que le contrôleur contient un bloc pour la détermination des l'intervalles (classes) auxquelles appartiennent les variables d'entrées, son rôle est de donner les numéros *i, j* des ensembles d'appartenance pour l'erreur et la variation de l'erreur, respectivement. A noter que le contrôleur ne contient pas d'interface de défuzzification, et le moteur d'inférence se réduit à un simple bloc pour le calcul de la valeur réelle de sortie.

Le principe de calcul est très simple. Il consiste à :

- Déterminer *i, j* les numéros des classes d'appartenance de chaque variable d'entrée.
- Se brancher sur la cellule de coordonnées *(i, j)* dans la table des règles et tirer la valeur du singleton de sortie.

- Ajuster la valeur du singleton en soustrayant la moyenne des distances Δe et Δde qui représentent les différences entre les valeurs de l'erreur et la variation de l'erreur et les centres des classes nettes auxquelles elles appartiennent. La valeur ainsi calculée est réelle et n'a pas besoin d'être convertie.

Cette méthode simplifie toutes les étapes du raisonnement flou, et minimise le temps calculs. Par conséquent le temps de la prise de décision sera très réduit, ce qui va faciliter l'implantation de tels contrôleurs pour des applications temps réel.

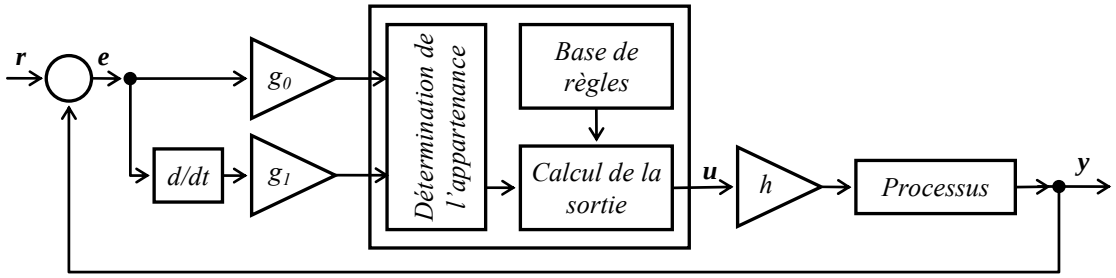

Figure.3.5. *Structure du contrôleur proposé.*

La surface de décision du contrôleur proposé est donnée sur la figure 3.6. Elle présente une certaine non linéarité. Dans notre méthode, le fait d'augmenter du nombre de classes pour la partition des variables n'affecte pas le temps de traitement, on propose de concevoir un contrôleur avec 9 classes (ensembles) pour chaque variable, ce qui va améliorer la résolution de notre contrôleur. La surface de décision de ce contrôleur avec 9 classes pour la partition de chaque variable est donnée sur la figure 3.7.

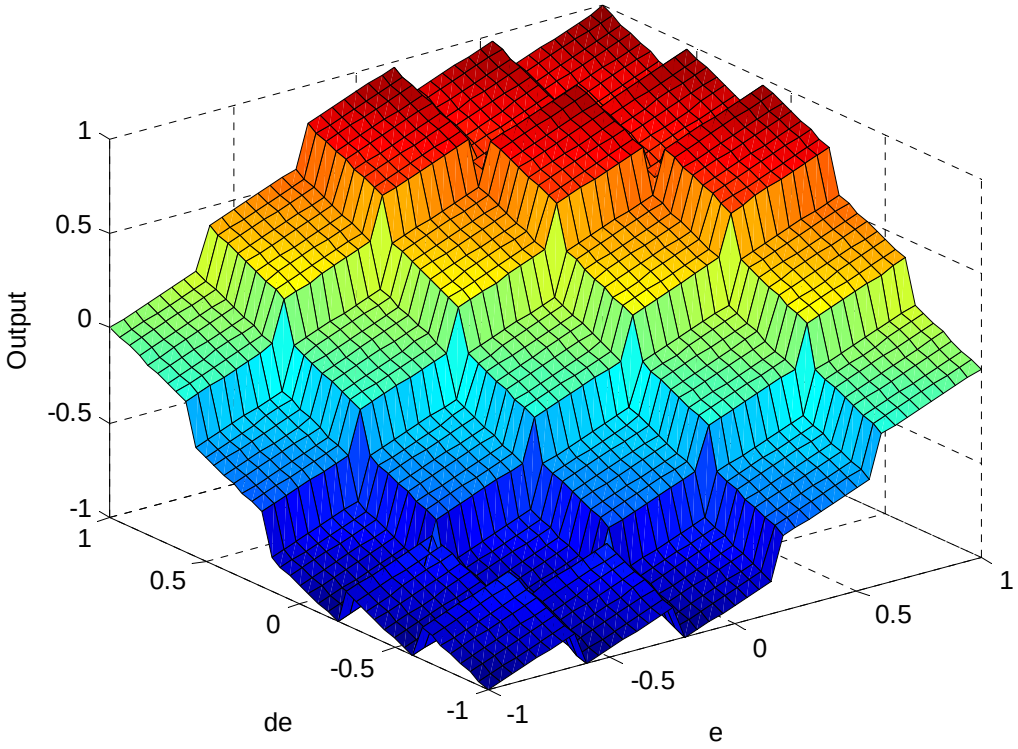

Figure.3.6. *Surface de décision du contrôleur proposé.*

(5 classes pour la partition de chaque variable)

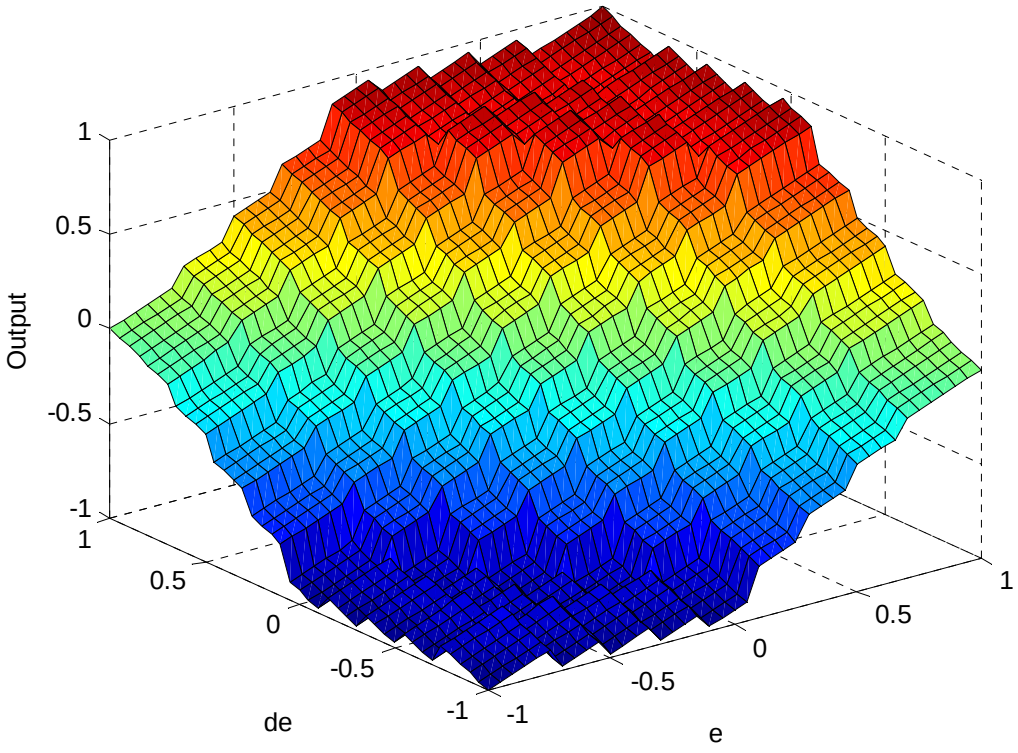

Figure.3.7. *Surface de décision du contrôleur proposé.*

(9 classes pour la partition de chaque variable)

3.4. Exemple d'application

Dans cette section, nous allons tester les performances des deux contrôleurs conçus sur un simple exemple de commande d'un système linéaire du deuxième ordre. Pour rendre la comparaison plus judicieuse on conserve les mêmes facteurs d'échelle *(g_0 =1, g_1 =1, h=100)*.

Nous allons étudier la poursuite d'une trajectoire sinusoïdale par un système caractérisé par une pulsation propre $\omega_n = 3\ rad/\sec$, et d'un coefficient d'amortissement $\xi = 0.75$.

La fonction de transfert du système est donnée par :

$$Y(s) = \frac{9}{s^2 + 4.5s + 9}$$

Le système est échantillonné à une période d'échantillonnage *Te= 0.01 s*.

La figure 3.8 représente les signaux de sortie du système commandé par les deux régulateurs, ce résultat montre la similitude des résultats et la qualité du signal de sortie du contrôleur proposé. Sauf que l'erreur commise par le contrôleur flou est plus petite que celle de notre contrôleur comme le montre l'agrandissement des régions marquées (figure 3.9 et 3.10).

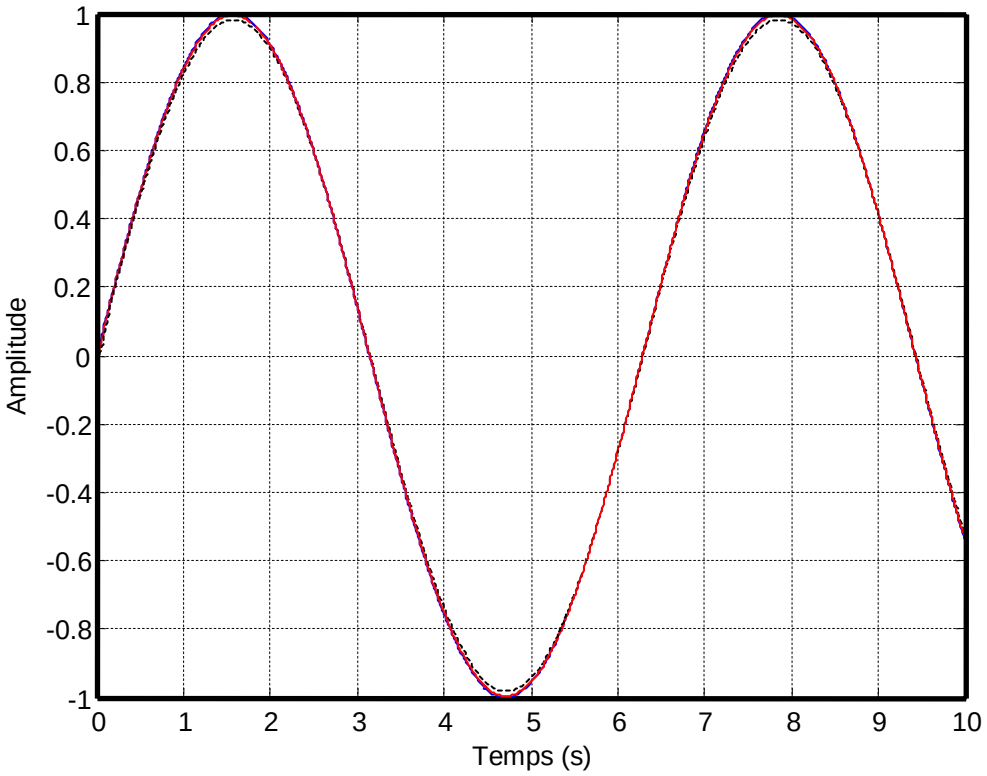

Figure.3.8. *Résultats de l'exemple.*

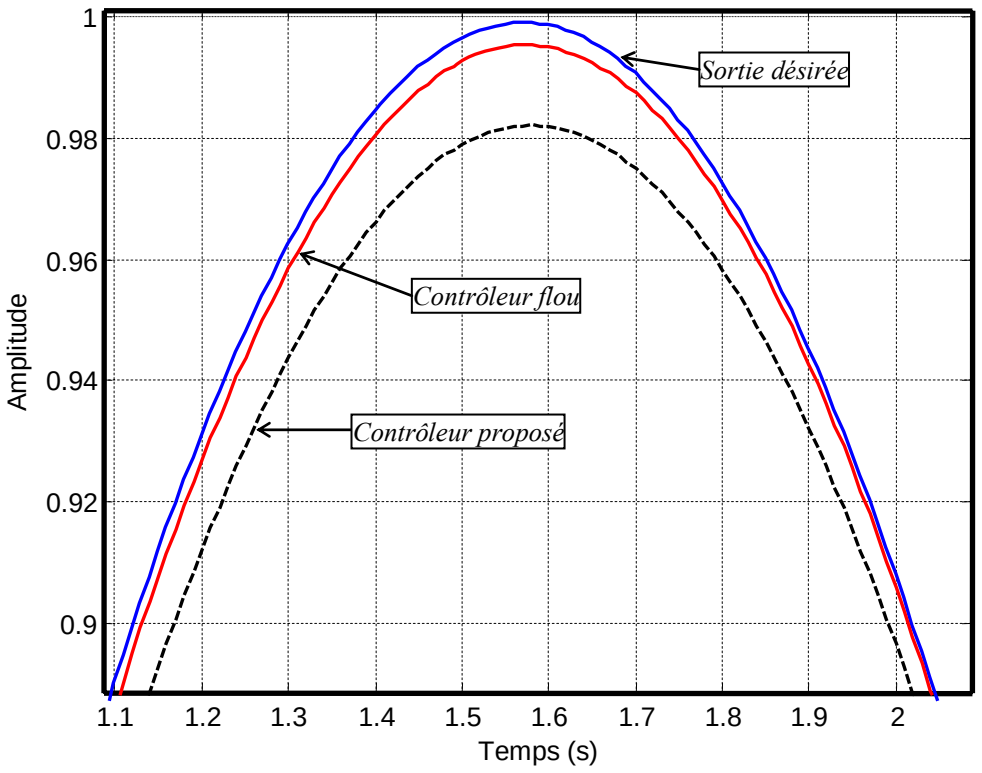

Figure.3.9. *Résultats de l'exemple.*

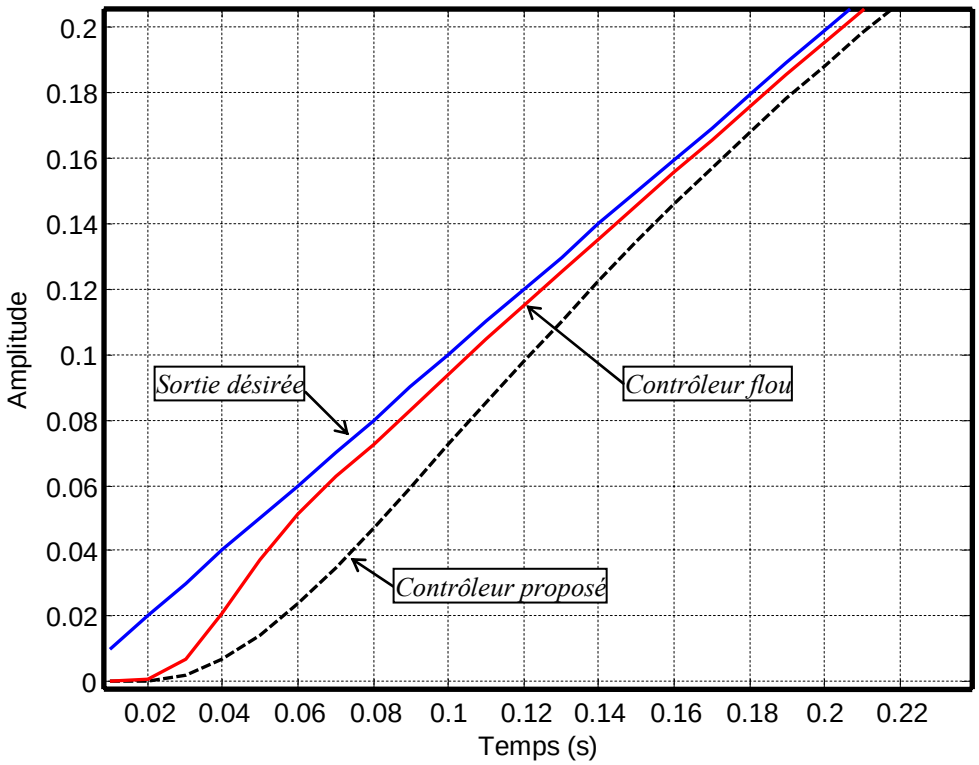

Figure.3.10. *Résultats de l'exemple.*

Sur les deux figures suivantes, sont représentés les signaux de commande issus des deux contrôleurs. Là, aussi, on remarque la similitude des résultats et la qualité du signal produit par le contrôleur proposé.

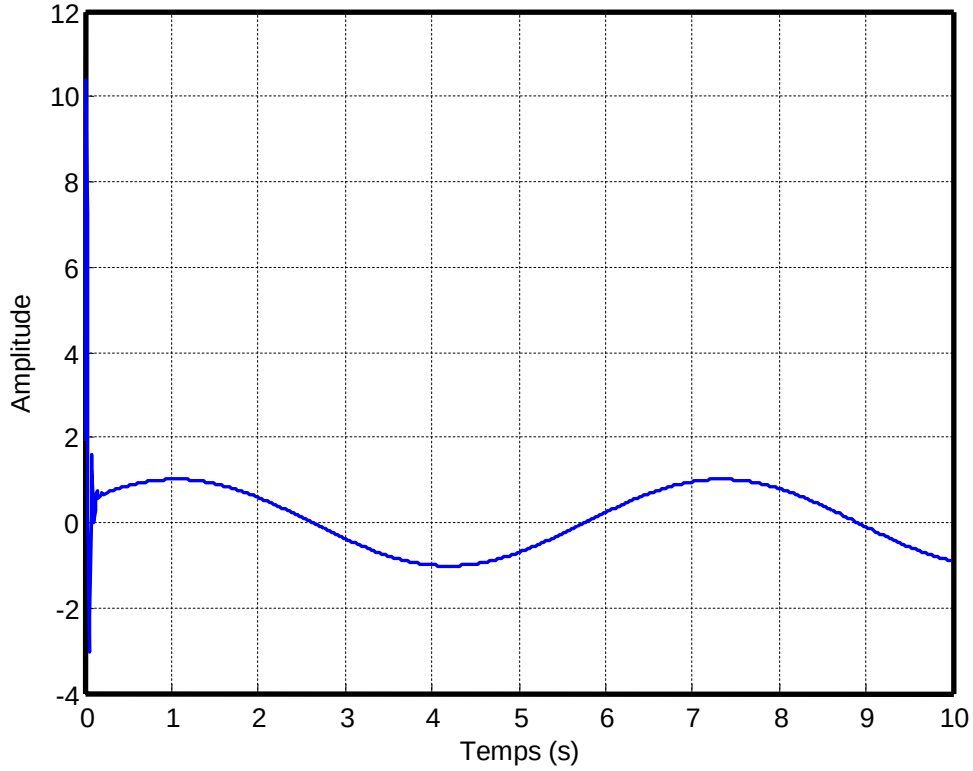

Figure.3.11. *Signal de commande (contrôleur flou).*

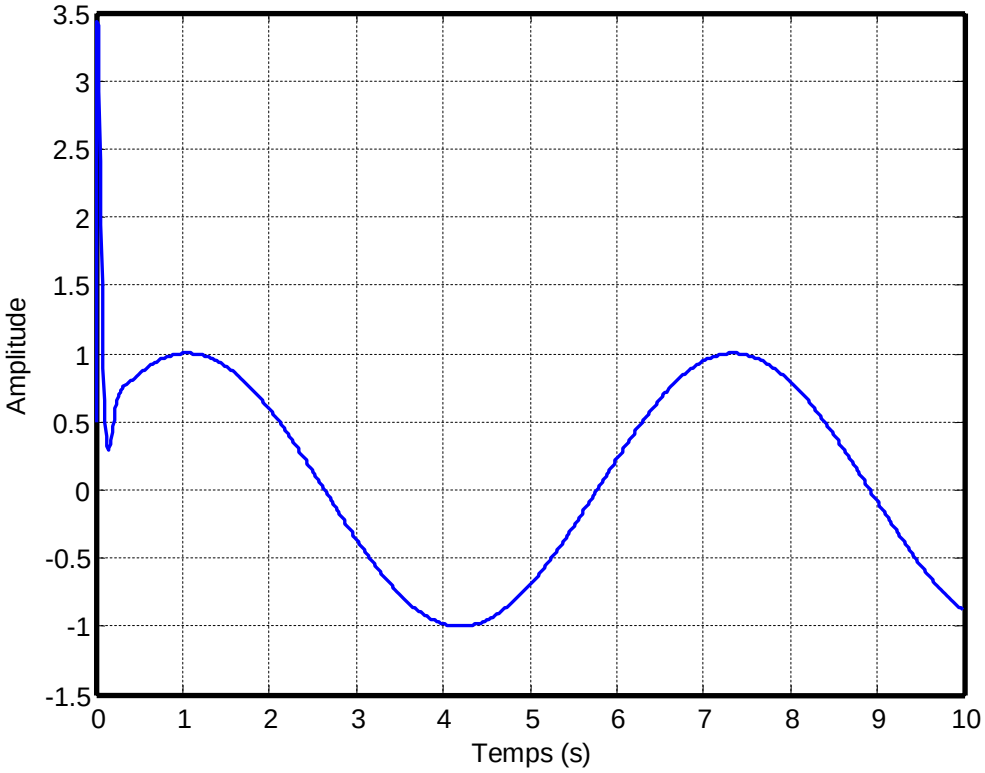

Figure.3.12. *Signal de commande (contrôleur proposé).*

Les signaux d'erreur représentés sur la figure 3.13, montrent qu'avec les mêmes facteurs d'échelle, l'erreur du contrôleur proposé est plus importante que celle du contrôleur flou.

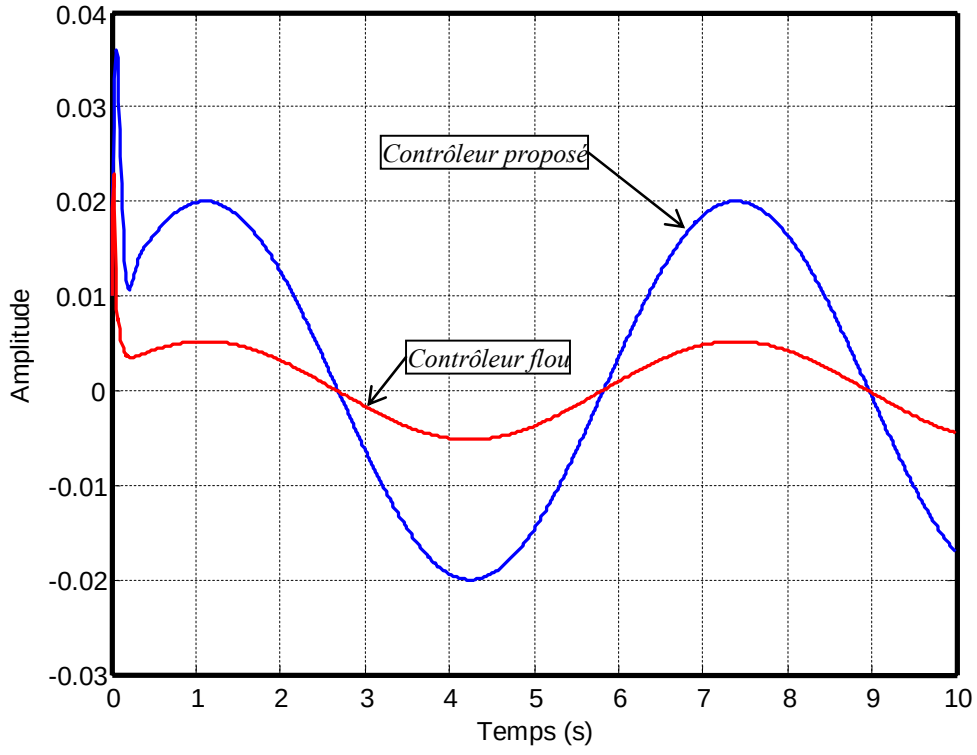

Figure.3.13. *Signaux d'erreur*

Sur la figure 3.14, on montre qu'il est possible de minimiser l'erreur du contrôleur proposé par la variation du facteur d'échelle *h*, on remarque qu'avec une valeur de $h = 400$, les signaux d'erreur se superposent, et les deux contrôleur donnent les mêmes résultats.

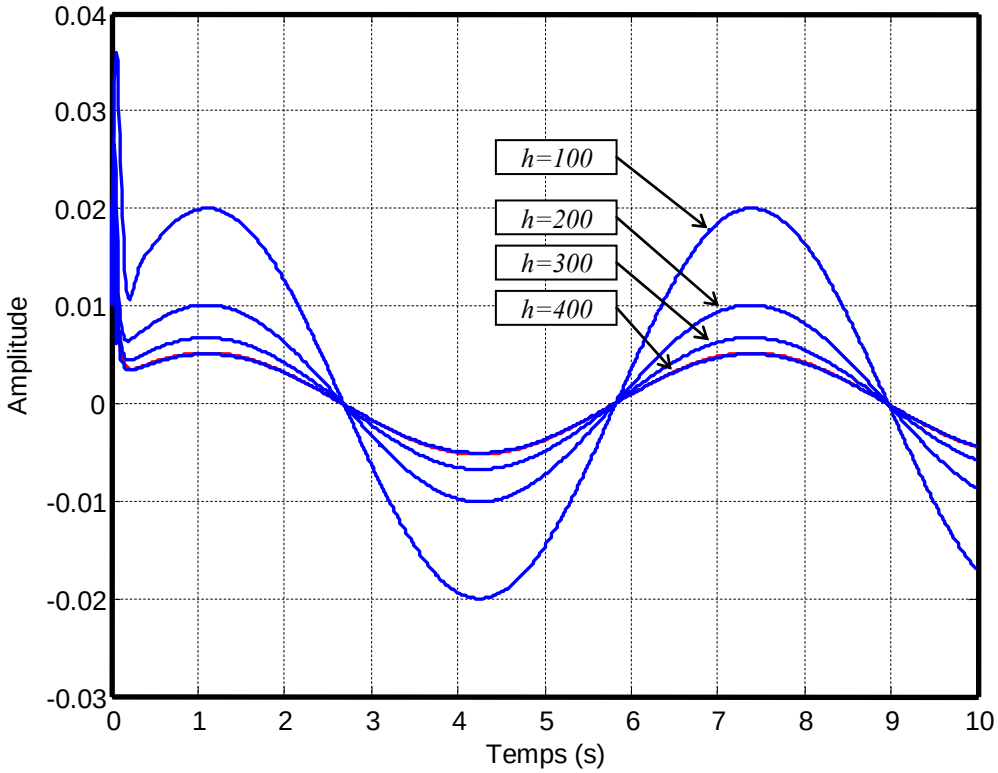

Figure.3.14. *La variation de l'erreur du contrôleur proposé en fonction de la variation du facteur d'échelle h.*

3.5. Conclusion

La procédure de conception de contrôleurs flous comporte plusieurs étapes communes, mais les choix des stratégies et des techniques pour la réalisation de chaque étape restent à déterminer par le concepteur. Les performances d'un contrôleur flou sont liées à l'expérience et le savoir faire du concepteur.

Nous avons montré qu'il est possible de réduire le temps de traitement dans les algorithmes de commande logique, en introduisant des modifications sur la logique classique qui souffre de plusieurs inconvénients parmi lesquels : les transitions raides entre les états.

Chapitre 4

Application A La Poursuite D'une Trajectoire Par Un Robot Mobile

4.1 Introduction

Dans le vaste domaine de la robotique mobile, l'étude du déplacement a une grande importance. De nombreuses approches du déplacement existent en utilisant différents types de capteurs.

Les robots mobiles, connaissant leurs positions, se déplacent d'un point à un autre avec diverses contraintes. Dans certains cas, seule l'atteinte du point final est importante, sans se préoccuper du trajet. Dans d'autres cas, le trajet parcouru est important tout comme sa dépendance du temps ; la trajectoire à suivre par le robot doit être définie.

Généralement, seul le chemin à suivre est défini. Le robot doit atteindre le point final selon une route indépendante du temps. C'est la tâche qui va nous intéresser dans ce chapitre.

Nous commencerons ce chapitre par la présentation du robot mobile utilisé pour la simulation, ainsi que son modèle cinématique, dans la section 4.1. Dans la section 4.2, nous expliquerons la stratégie de la poursuite de trajectoire utilisée. Le contrôleur conçu pour ce fait est présenté dans la section 4.3 et les résultats de simulation de la poursuite de quelques exemples de trajectoires sont donnés dans la section 4.4.

4.2. Le robot mobile

La structure du robot mobile utilisé pour la simulation est donnée sur la figure 4.1. Il possède deux roues motrices arrières, dotées chacune d'un moteur indépendant muni d'un encodeur odométrique pour la mesure de la position et la vitesse angulaires. La roue avant est une roue libre, son rôle est d'assurer la stabilité du robot pendant son mouvement, elle est dotée d'un capteur d'orientation qui donne la mesure de l'angle de braquage du robot.

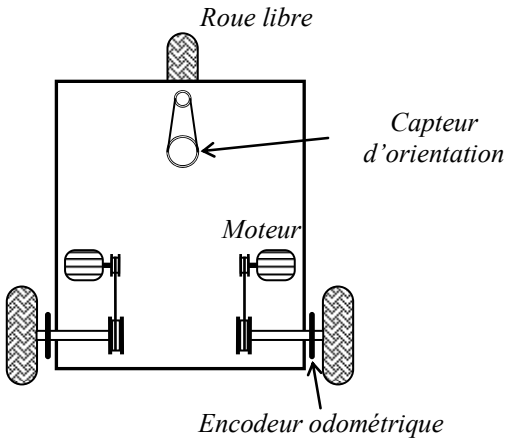

Figure 4.1. *Structure du robot mobile*

Pour la modélisation cinématique du robot mobile, nous avons besoin de déterminer les équations de quelques paramètres du robot. La figure 4.2 représente le paramétrage nécessaire pour la modélisation de notre robot mobile.

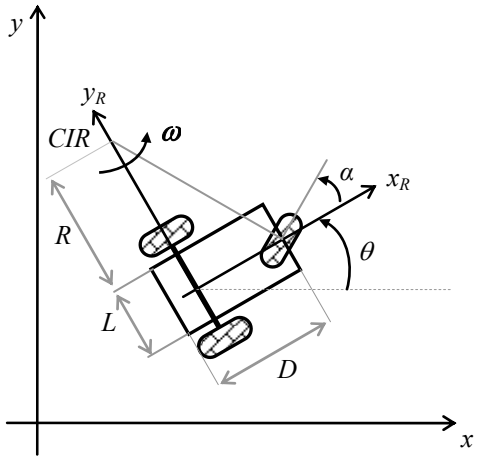

Figure 4.2. *Paramétrage du robot mobile*

L'angle de braquage est donné par :

$$\alpha = arctg\left(\frac{D}{R}\right)$$

$$R = \frac{L\left(V_d + V_g\right)}{2\left(V_d - V_g\right)}$$

$$\alpha = arctg\left(\frac{2D\left(V_d - V_g\right)}{L\left(V_d + V_g\right)}\right)$$

Où :

α : L'angle de braquage,

θ : L'orientation du robot,

ω : La vitesse de rotation du robot autour de son centre instantané de rotation *(CIR)*,

R : Le rayon de courbure,

L : L'entre axe,

D : La longueur du châssis,

V_d : La vitesse linéaire de la roue droite,

V_g : La vitesse linéaire de la roue gauche,

V : La vitesse linéaire du robot,

Le modèle cinématique d'un robot mobile peut est donné par [27] :

$$\begin{aligned}\dot{x} &= V\cos\theta \\ \dot{y} &= V\sin\theta \\ \dot{\theta} &= \frac{V}{D}tg\alpha\end{aligned}$$

L'intégration des équations cinématiques pour des déplacements élémentaires entre deux instants t et $(t + \Delta t)$ donne :

- Si $\dot{\alpha} \neq 0$, $\left(V_g \neq V_d\right)$ l'orientation du robot par rapport à l'axe horizontal à l'instant $(t + \Delta t)$ est donnée par :

$$\begin{aligned}\theta\left(t+\Delta t\right) &= \theta\left(t\right) + \frac{V}{D}\int_t^{t+\Delta t} tg\alpha\, dt \\ &= \theta\left(t\right) + \frac{V}{D\dot{\alpha}}\left(-\ln\left|\cos\alpha\left(t+\Delta t\right)\right| + \ln\left|\cos\alpha\left(t\right)\right|\right)\end{aligned}$$

- Si $\dot{\alpha} = 0$, $\left(V_g = V_d\right)$, l'orientation du robot à l'instant $(t + \Delta t)$ est donné par : $\theta\left(t+\Delta t\right) = \theta\left(t\right) + \frac{V}{D}tg\alpha\Delta t$

La position du robot à l'instant *(t + Δt)* peut être calculée comme suit.

– Si $\dot{\theta} \neq 0$, $\begin{cases} x(t+\Delta t) = x(t) + \dfrac{V}{\dot{\theta}}\left(\sin\theta(t+\Delta t) - \sin\theta(t)\right), \\ y(t+\Delta t) = y(t) + \dfrac{V}{\dot{\theta}}\left(\cos\theta(t+\Delta t) - \cos\theta(t)\right) \end{cases}$

– Si $\dot{\theta} = 0$, $\begin{cases} x(t+\Delta t) = x(t) + V\cos\theta\Delta t, \\ y(t+\Delta t) = y(t) + V\sin\theta\Delta t \end{cases}$

4.3. La stratégie de poursuite

Dans l'approche que nous avons utilisée, une trajectoire est représentée par un ensemble de points de passage reliant le point de départ au point d'arrivée. Le chemin à suivre peut être une trajectoire d'évitement d'obstacle issue d'un planificateur, ou un nombre de points de passage qui peuvent être des postes de travail. Dans notre cas, nous avons considéré le premier cas ou la trajectoire est un chemin d'évitement stockée en mémoire sous la forme de deux vecteurs d'abscisses et d'ordonnées des points formant la trajectoire. Dans ce cas le robot doit aboutir à son but sans en avoir besoin de passer sur les points intermédiaires avec précision.

Pour conserver l'aspect simpliste du système de contrôle, et utiliser le contrôleur flou conçu dans le chapitre précédent, notre robot doit calculer l'angle d'orientation θ_d qui va le mener directement de sa position courante vers le prochain point sur la trajectoire. La figure 4.3 illustre ce principe.

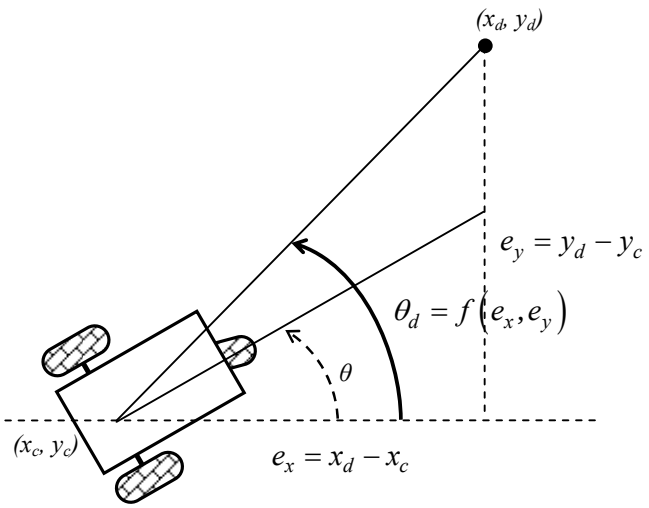

Figure 4.3. *Principe de la poursuite*

Le robot dans sa position courante de coordonnées (x_c, y_c), calculée par des moyens odométriques, va calculer les composantes de l'erreur en position, c'est-à-dire les distances (e_x, e_y), qui vont servir par la suite au calcul de l'orientation désirée.

Le calcul de l'orientation désirée se fait selon la position du robot par rapport au point de destination. Les différentes formules utilisées pour le calcul de θ_d sont données sur la figure 4.4, avec les conditions suivantes :

$$\begin{aligned}
&Si\ \theta_d < 0 \quad Alors \quad \theta_d = \theta_d + 2\pi \\
&Si\ \theta_d > \pi \quad Alors \quad \theta_d = \theta_d - 2\pi \\
&Si\ \Delta\theta_d > \pi \quad Alors \quad \theta_d = \theta_d - 2\pi \\
&Si\ \Delta\theta_d < -\pi \quad Alors \quad \theta_d = \theta_d + 2\pi
\end{aligned}$$

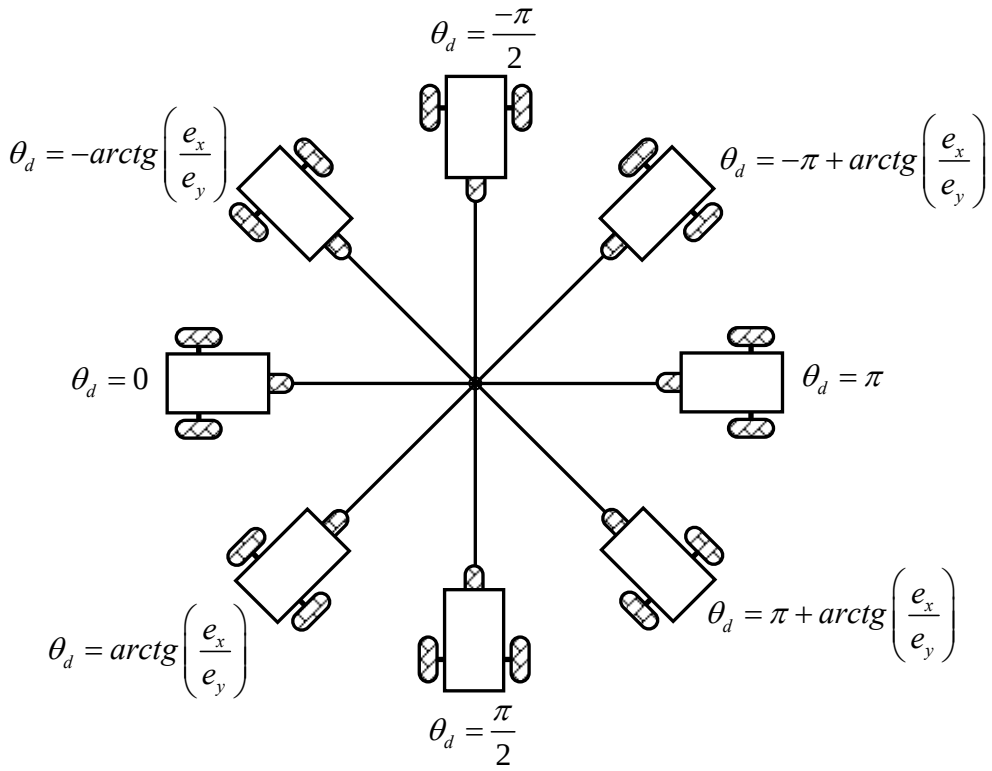

Figure 4.4. *Principe de calcul de l'orientation désirée*

Sur le chemin à parcourir par le robot, on va supposer des zones de tolérance autour des points de passage. Pendant son déplacement, si le robot entre dans la zone de tolérance du point auquel il se dirige, il commence à s'orienter vers le prochain point. Ce fait va donner un aspect souple au déplacement du robot, et va éviter les arrêts et les changements d'orientation sur site, ces rotations sur places sont coûteuses en terme de consommation énergétique. Le robot doit arriver au point final avec une tolérance très réduite. Sur la figure 4.3 est illustré un exemple de trajectoire à suivre, les cercles autour des points représentent les zones de tolérance. Les flèches décrivent le déplacement du robot le long de la trajectoire.

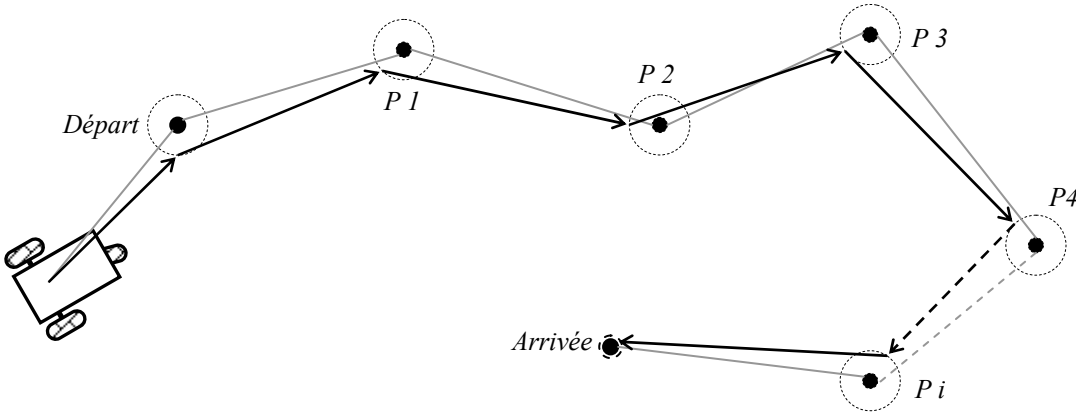

Figure 4.5. *Poursuite d'une trajectoire.*

4.4. Le contrôleur conçu

Pour matérialiser la stratégie décrite dans la section précédente, nous proposons un contrôleur dont la structure est la suivante :

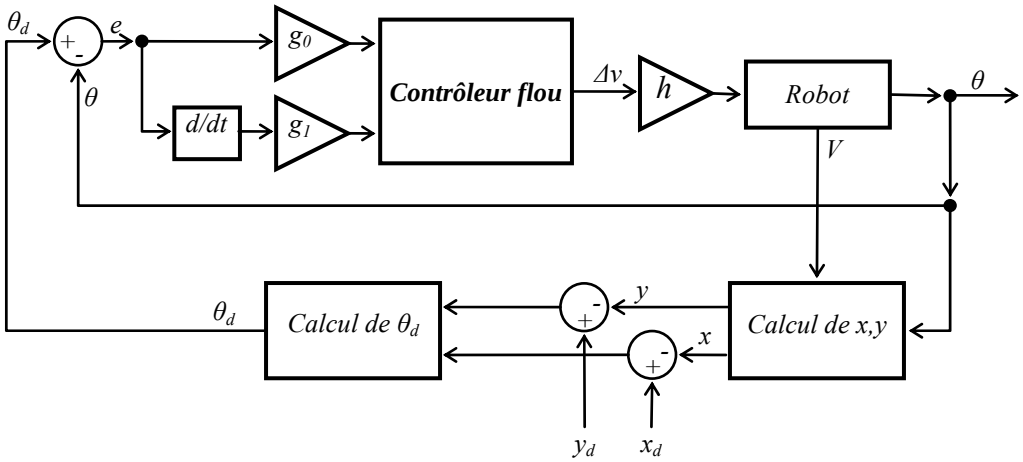

Figure 4.6. *Structure du contrôleur conçu.*

En odométrie les capteurs fournissent au robot des mesures qui vont servir à estimer sa position courante en fonction d'une position initiale connue et des déplacements effectués. Les entrées effectives du contrôleur sont les coordonnées du point désiré, à partir des quelles on calcule les composantes de l'erreur en position que va utiliser le module de calcul de l'orientation désirée pour donner la valeur de θ_d, cette valeur comparée avec l'orientation courante du robot fournit l'erreur qui va à son tour servir avec sa dérivée d'entrées au contrôleur flou.

4.5. Les résultats de simulation

Dans ce qui suit, en suppose nuls tous les états initiaux du robot. Les figures suivantes montrent les résultats de l'application des deux algorithmes de commande, à savoir la logique floue et la logique classique modifiée pour quelques exemples de trajectoires. Nous avons utilisé un contrôleur à logique classique modifiée à neuf classes pour chaque entrée et neufs singletons pour la sortie. Les facteurs d'échelle sont les même pour les deux méthodes.

Les figures ci dessous montrent la grande similitude des résultats obtenus, ainsi que l'efficacité et la simplicité de la stratégie de poursuite utilisée.

Trajectoires rectilignes de différentes pentes :

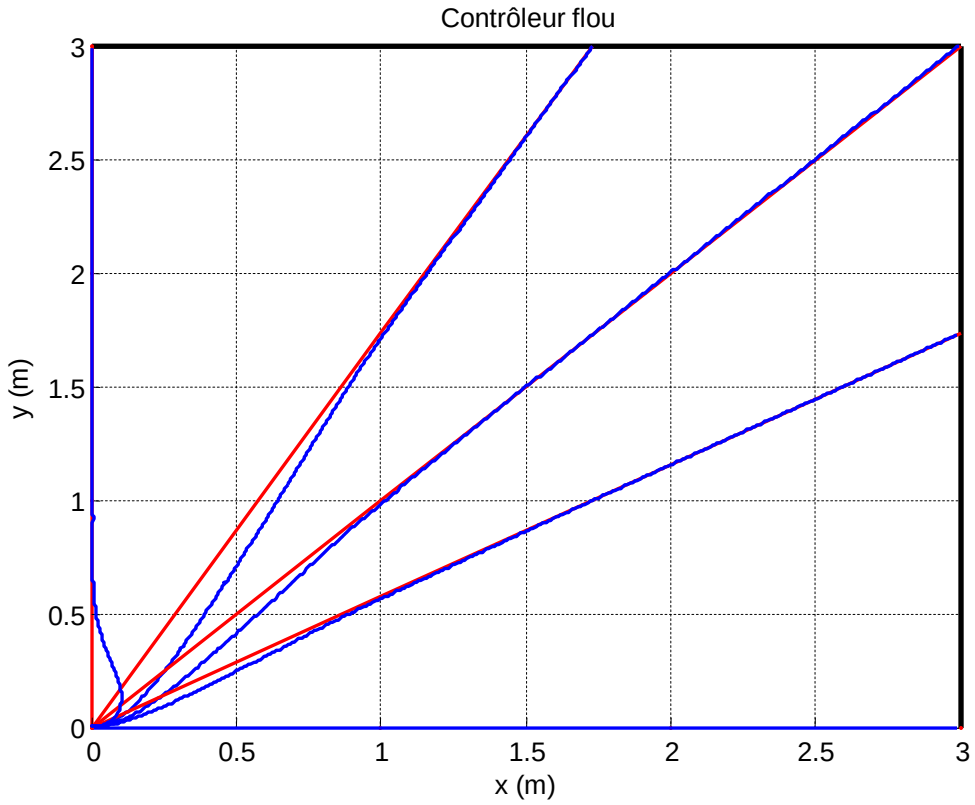

Figure 4.7. *Poursuite de trajectoires rectilignes.*

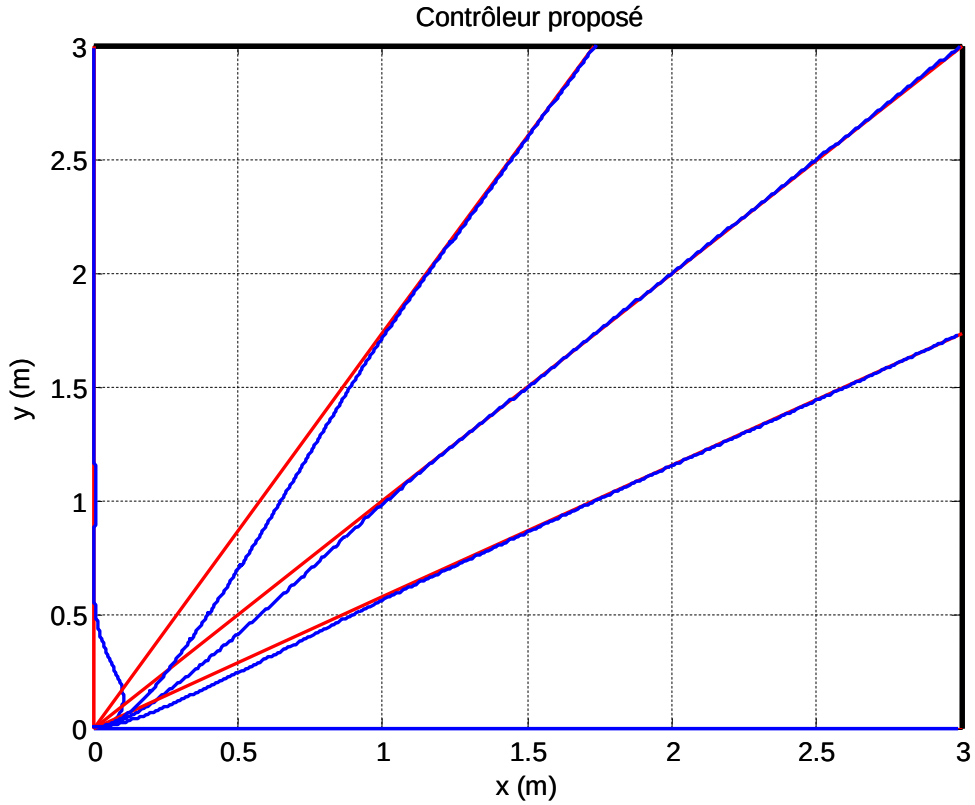

Figure 4.8. *Poursuite de trajectoires rectilignes.*

Trajectoire rectiligne parallèle à l'axe horizontal :

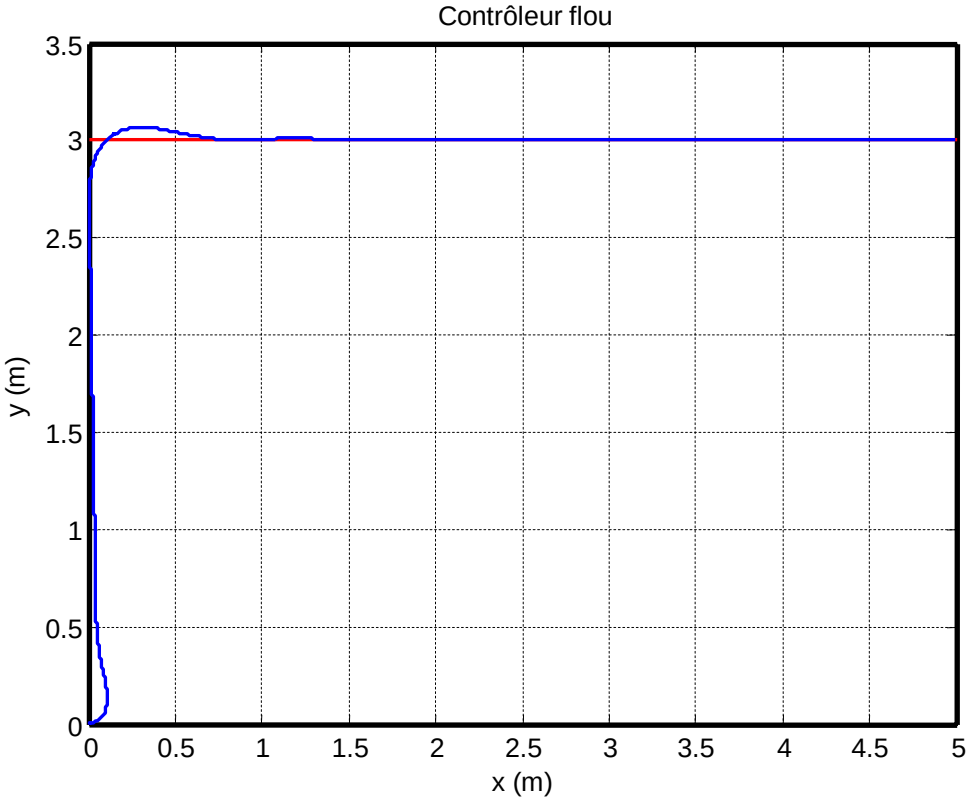

Figure 4.9. *Poursuite de trajectoire rectiligne.*

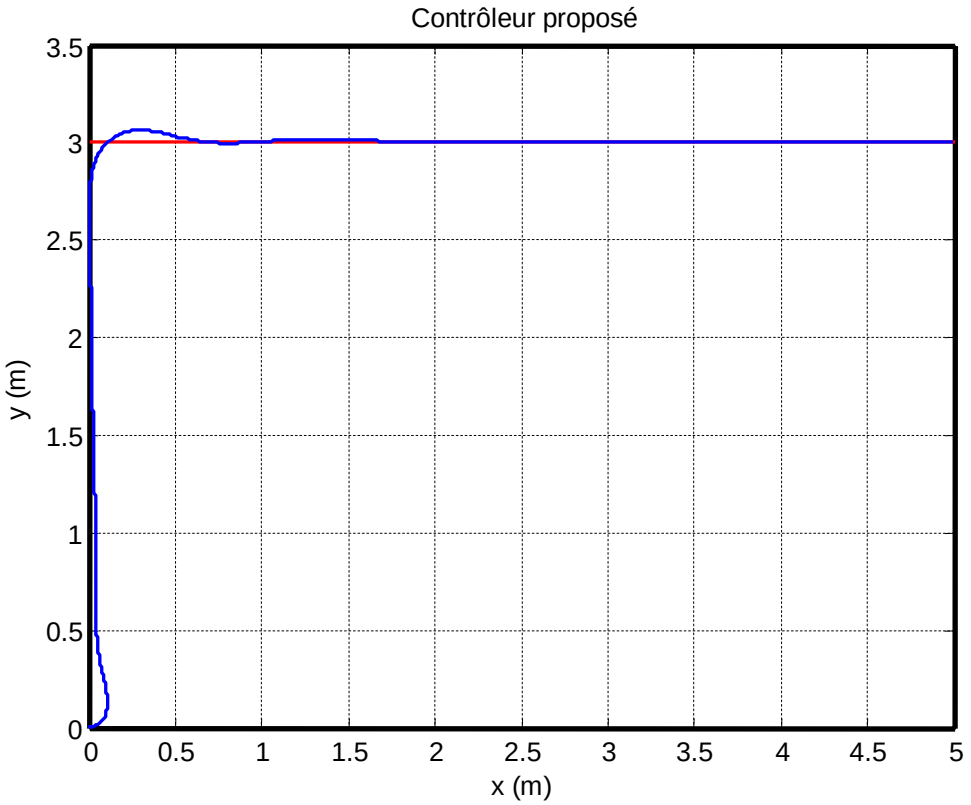

Figure 4.10. *Poursuite de trajectoire rectiligne.*

Trajectoire rectiligne parallèle à l'axe vertical :

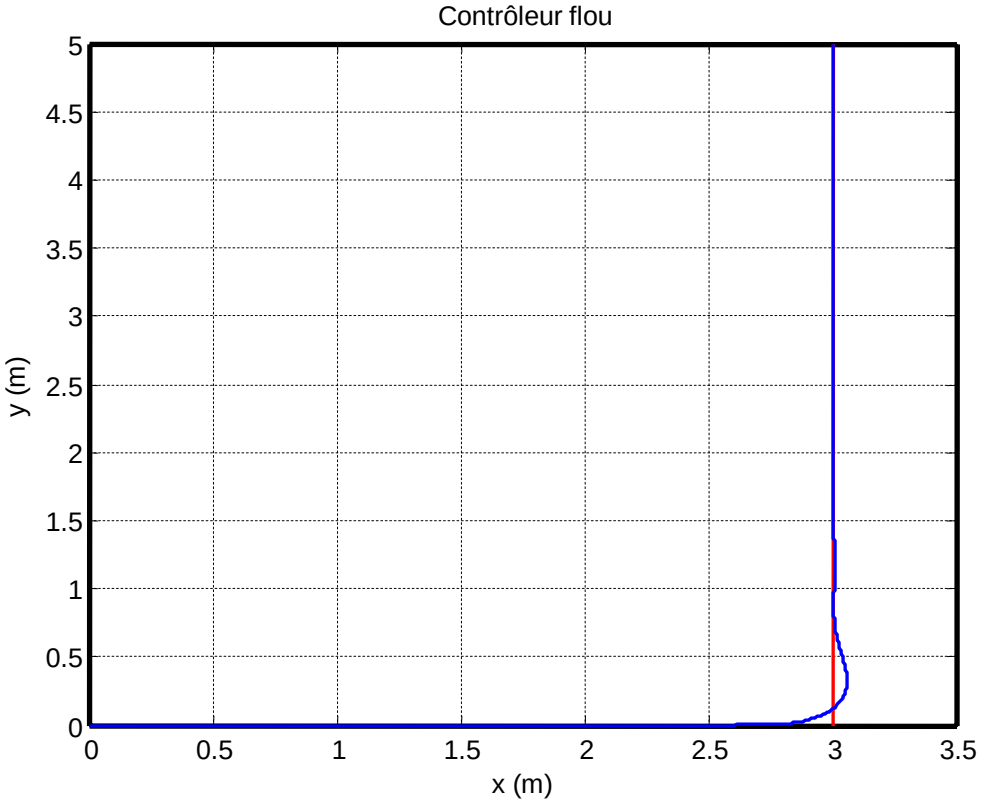

Figure 4.11. *Poursuite de trajectoire rectiligne.*

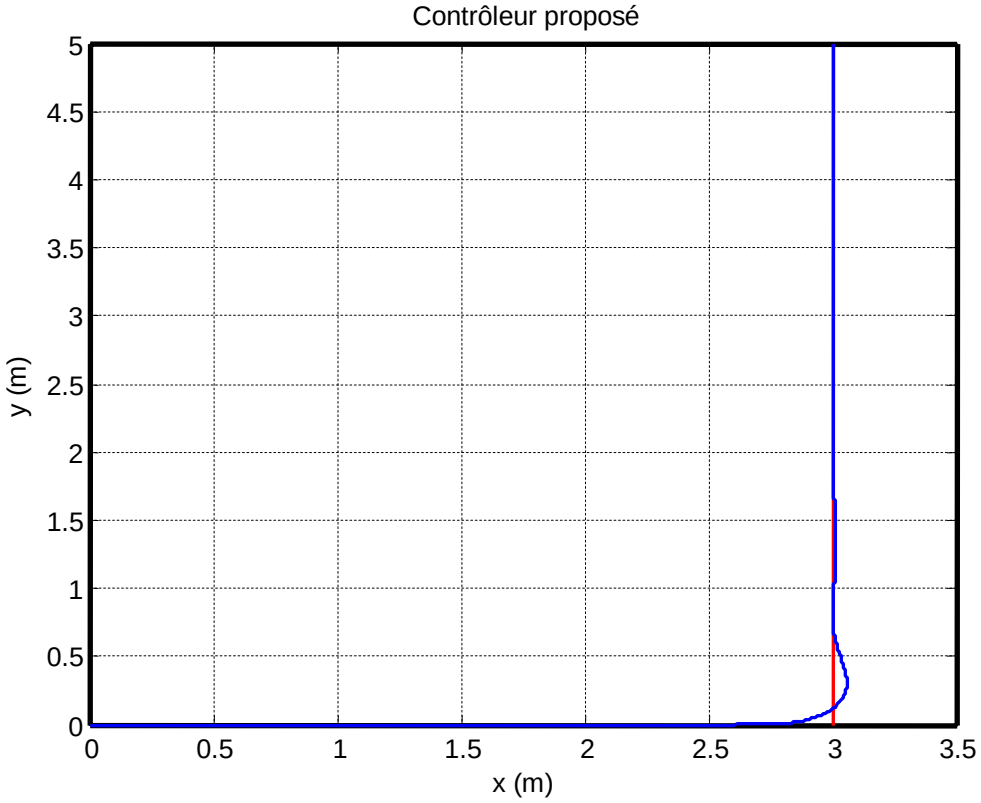

Figure 4.12. *Poursuite de trajectoire rectiligne.*

Trajectoire sinusoïdale :

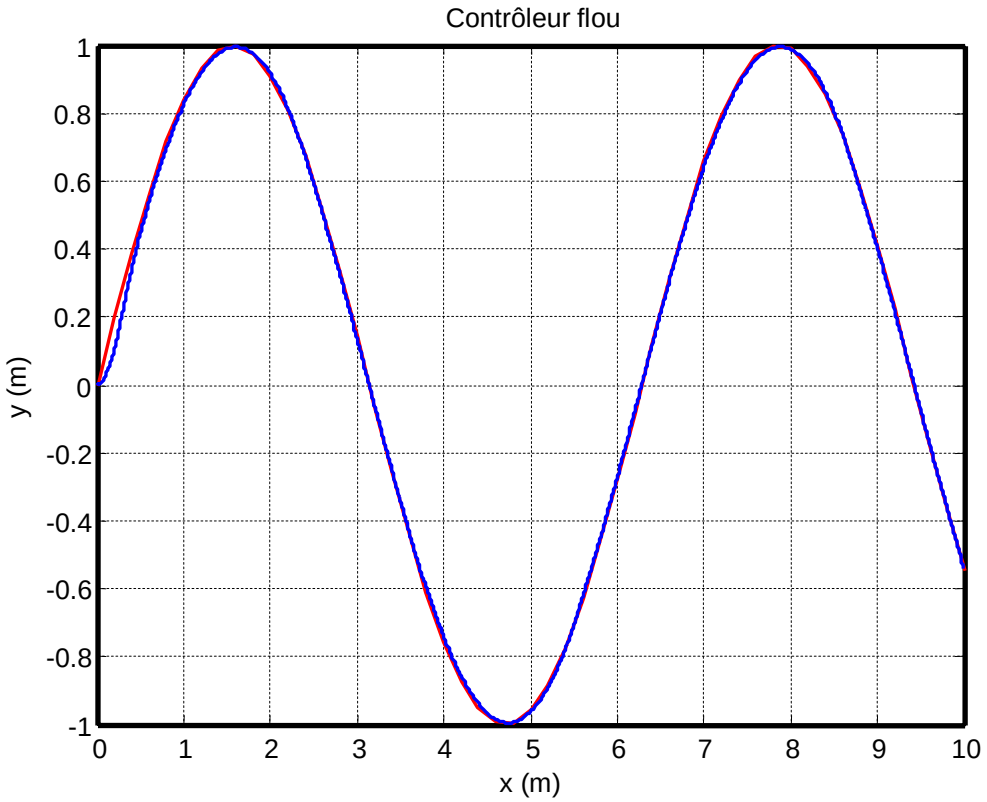

Figure 4.13. *Poursuite de trajectoire sinusoïdale.*

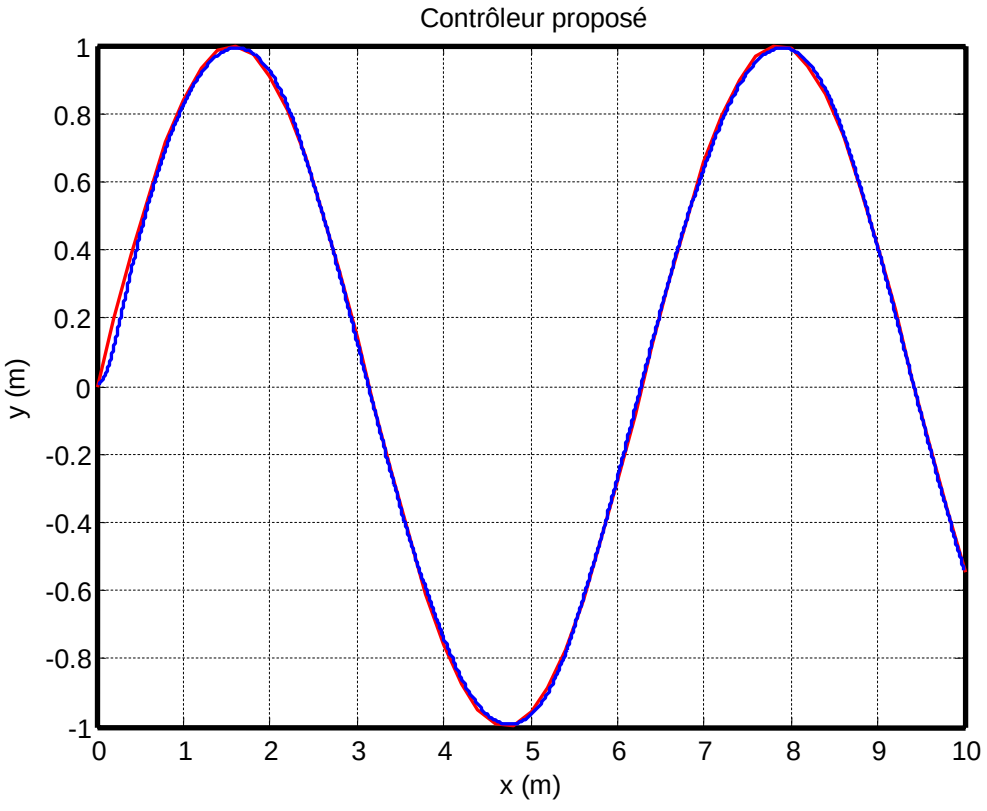

Figure 4.14. *Poursuite de trajectoire sinusoïdale.*

Trajectoire triangulaire :

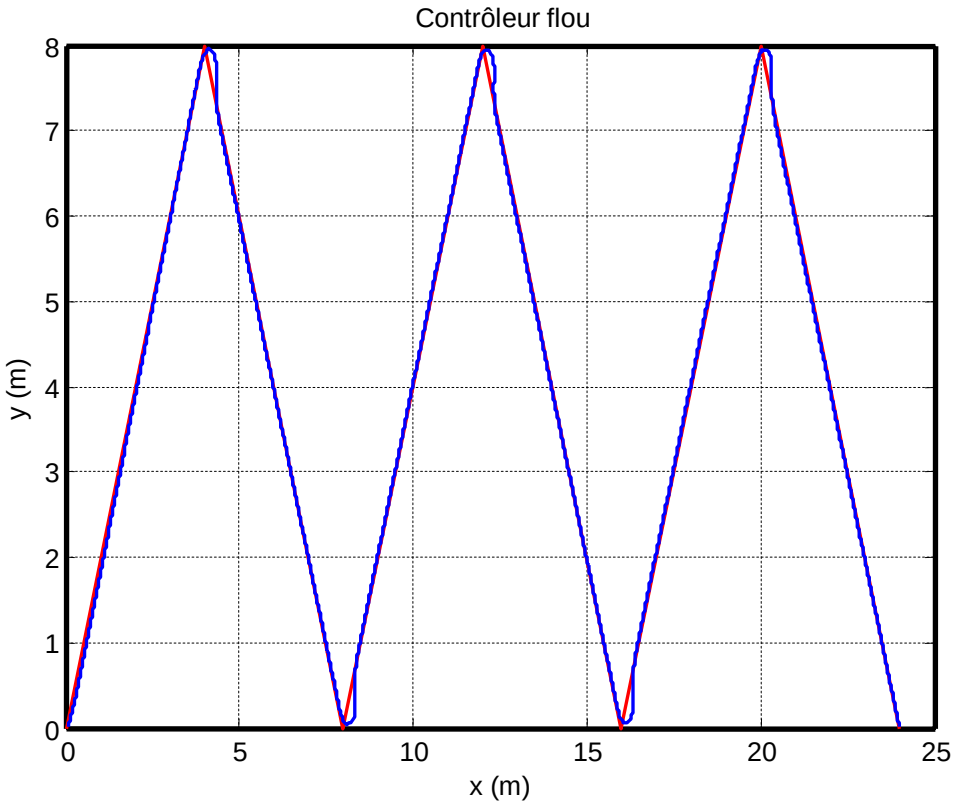

Figure 4.15. *Poursuite de trajectoire triangulaire.*

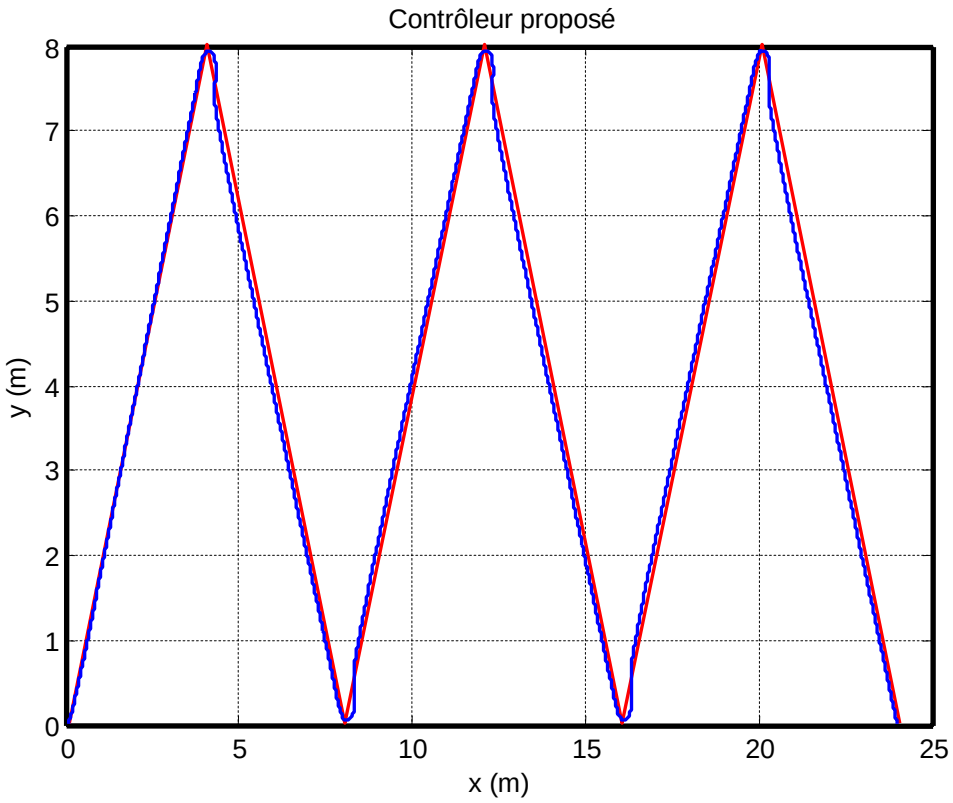

Figure 4.16. *Poursuite de trajectoire triangulaire.*

Trajectoire quelconque

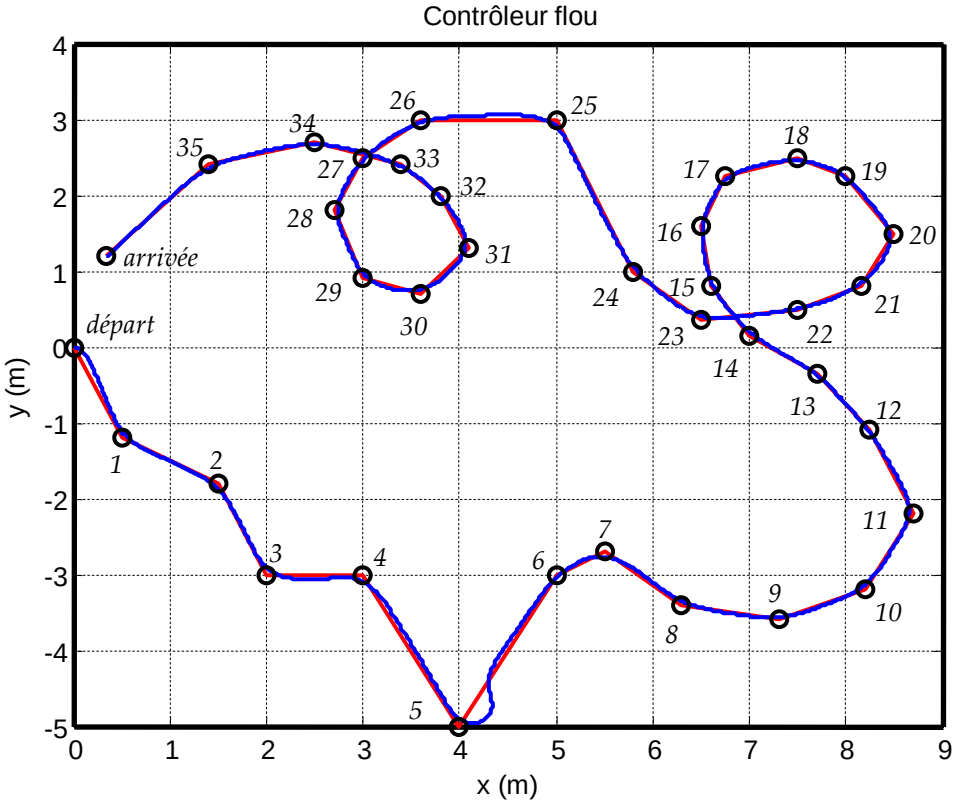

Figure 4.17. *Poursuite d'une trajectoire quelconque.*

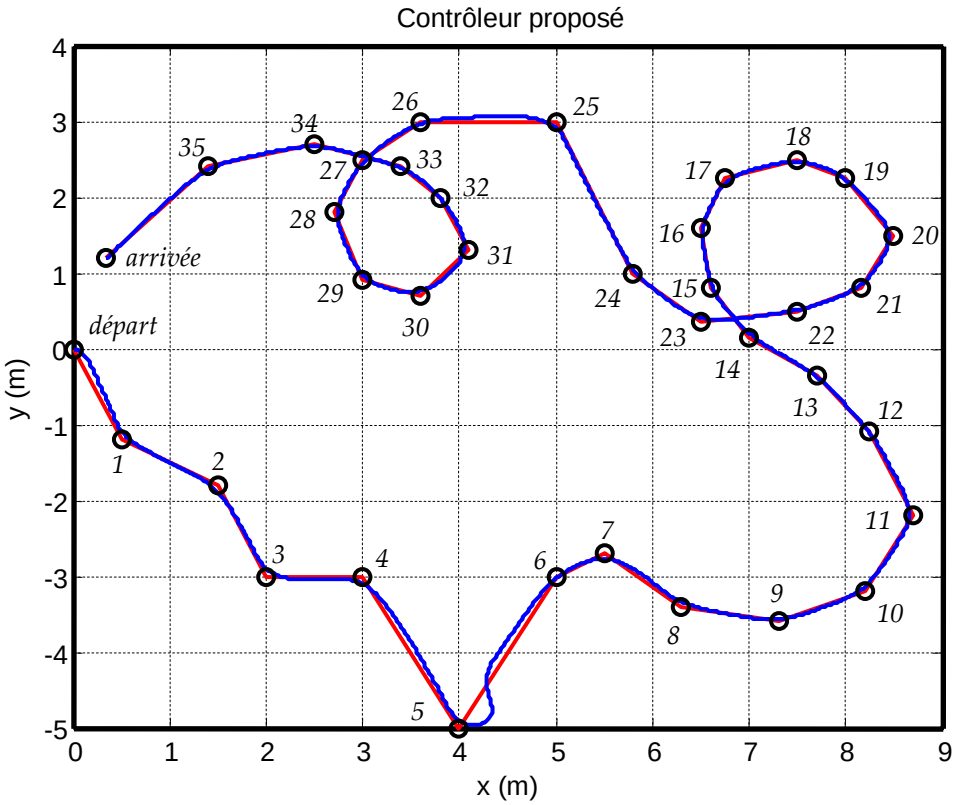

Figure 4.18. *Poursuite d'une trajectoire quelconque.*

4.6. Conclusion

La poursuite d'une trajectoire est une tâche importante que doit exécuter un robot mobile avec le minimum d'erreurs. Elle constitue la base de toute mission du robot. Pour les robots à commande différentielle, le maintien du robot sur son chemin revient à régler les vitesses des roues motrices de façon à donner au robot l'orientation désirée.

Une trajectoire est un ensemble de points à parcourir par le robot pour atteindre son but. La stratégie adoptée assure la souplesse des mouvements et simplifie la tâche de suivre une trajectoire.

Les résultats obtenus par la commande floue et par la méthode proposée montrent une grande ressemblance, ce qui encourage l'utilisation de notre méthode pour des processus plus compliqués et aussi pour d'autres applications de la logique floue.

Conclusion Générale

La logique floue est une technique qui permet une modélisation graduelle et nuancée des connaissances d'un expert ce qui mène à un mode de raisonnement très proche de celui d'un opérateur humain. Par conséquent, cette technique permet de prendre en comptes toutes nature de connaissances qualitatives d'un expert dans toute application. Cette souplesse et simplicité de la logique floue ont fait de la logique floue l'un des domaines de recherche les plus actifs durant les dernières décennies. Les chercheurs travaillant sur l'application de la logique floue dans le contrôle de processus se sont concentrés sur l'élaboration de règles générales pour la conception de contrôleurs flous, et sur les critères d'analyse de stabilité, et les algorithmes d'optimisation des systèmes flous. L'inconvénient majeur de la logique floue réside dans la quantité importante de calculs qu'effectue un système flou pour aboutir à une décision. Ces calculs nécessitent un temps de traitement assez important. Ce qui rend difficile l'implantation des contrôleurs flous pour des applications temps réel.

Lors de la conception du contrôleur flou, le temps de traitement était le facteur le plus important. Pour cela, nous avons opté pour des fonctions d'appartenance triangulaires uniformément réparties sur des univers de discours normalisés. Les stratégies d'inférence et de défuzzification choisies ne nécessitent pas beaucoup de calculs.

La logique booléenne classique souffre de son incapacité de traiter l'imprécis et le vague. Après l'étude et l'observation du fonctionnement de l'algorithme

flou. On a vu qu'il est possible de rendre souple la logique classique en prenant en compte les distances entre les variables d'entrée et les centres des classes auxquelles elles appartiennent. Cette approche, appliquée à des cas pratiques a montré une grande souplesse tout en minimisant le temps de traitement sans pour autant nécessiter de ressources matérielles spéciales. Comme l'on a montré dans le chapitre trois, la méthode que nous avons proposée simplifie toutes les étapes du traitement flou, et donne des résultats très proches de ceux du contrôleur flou. Il reste à démontrer la validité de cette méthode sur d'autres exemples de processus non linéaires plus compliqués, et dans d'autres applications où la logique floue a connu un grand succès. L'étude et l'analyse de la stabilité de tel contrôleur seront d'une importance primordiale.

En robotique mobile la poursuite de trajectoire est une tâche élémentaire, qui représente la base de toute autre tâche du robot. Dans la stratégie que nous avons utilisée la trajectoire à suivre est représentée par un certain nombre de points de passage entre le point de départ et le point d'arrivée. Le robot exécute sa trajectoire point par point, en corrigeant son orientation en fonction de sa position par rapport au point désiré. Le rôle du contrôleur conçu est de régler l'orientation du robot en jouant sur les vitesses des roues motrices.

Dans nos futurs travaux, nous essayerons d'améliorer davantage notre contrôleur par l'optimisation de la formule utilisée pour le calcul de sortie afin de rendre notre algorithme plus souple et s'approcher au maximum du contrôleur flou. Un autre sujet très important : les systèmes à plusieurs entrées plusieurs sorties, qui vont servir de bons tests pour l'algorithme proposé.

Bibliographie

[1] Roland Siegwart and Illah R. Nourbakhsh, *Introduction to Autonomous Mobile Robots,* the MIT press. ISBN: 0-262-19502-X.

[2] A. Louchene, *système de navigation pour robot mobile,* thèse de doctorat, de l'université de Batna 2004.

[3] T. Kanade, C. Thorpe et W. *Whittaker. Autonomous Land Vehicle Project at CMU*. In ACM Annual Computer Science Conference, pages 71–80. ACM Press, 1986.

[4] Bachir Lakehal, *commande d'un robot mobile destiné à des applications domotiques,* thèse de doctorat, de l'université Paris XII 1995.

[5] C. Durieu, *algorithmes de localisation d'un robot mobiles dans un milieu balisé par mesure de distance ou d'angle de gisement e tenant compte des mesures aberrantes. Algorithmes de calibration et de recalage du champ de balise,* thèse de doctorat de l'université de Paris sud 1989.

[6] T. Takagi et M. Sugeno, *fuzzy identification and its application to modeling and control,* IEEE transactions on systems. Man and Cybernetics. Pages: 116 – 132, février 1985.

[7] R. M. Tong, *some properties of fuzzy feedback systems.* IEEE transactions on systems. Man and Cybernetics. Pages: 327 – 330, 1980.

[8] M. Togai et P. P. Wang, *analysis of fuzzy dynamic system and synthesis off its controller.* International journal off Man Machine studies. Pages: 355 – 366, 1985.

[9] I. M. Bavelaar, *fuzzy model inversion.* Rapport LAAS 96475, décembre 1996.

[10] Chuen Chien Lee, *fuzzy logic in control systems: fuzzy logic controller- part I and II*, IEEE Transactions on Systems, Man, and Cybernetics, Vol 20, No 2. March/April 1990. Pages: 404 -418.

[11] M. Sugeno, *an introduction survey to fuzzy control. Information sciences*. Pages : 59 - 83, 1985.

[12] K. L. Tang et R. L. Mulholland, *comparing fuzzy logic with classical controller design*. IEEE transactions on systems. Man and Cybernetics. Pages: 1085–1087, 1987.

[13] Véronique Lacrose, *réduction de la complexité des contrôleurs flous : application à la commande multivariables*. thèse de doctorat, de l'institut national des sciences appliquées de Toulouse 1997.

[14] Mac Vecar Whelan, fuzzy sets for man machine interaction. International journal off Man Machine studies. Pages : 687 – 697, 1976.

[15] Bernadette et Bouchon Meunier, *la logique floue et ses applications*, Adison Wesley, ISBN 2-87908-073-8, 1995.

[16] Jerry M. Mendel, *Tutorial: fuzzy logic systems for engineering*, Proceeding of the IEEE, Vol. 83, N°3, pages: 345-377, March 1995.

[17] Ali Zilouchian and Mo Jamshidi, *Intelligent Control Systems Using Soft Computing Methodologies*, CRC Press LLC, ISBN 0-8493-1875-0, 2001.

[18] Kevin M. Passino and Stephen Yurkovich, *Fuzzy Control*, Addison Wesley Longman, ISBN 0-201-18074-X, 1998.

[19] R. M Tong, *analysis and control of fuzzy systems using finite state relations*. International journal of control. Pages: 131 – 140, 1978.

[20] K. Tanaka et M. Sugeno, *stability analysis and design of fuzzy control systems*. Fuzzy sets and systems. Pages: 135 – 156, 1992.

[21] Hansruedi Bühler, "réglage par logique floue", *Presses Polytechniques et Université Romandes*, ISBN 2-88074-271-4, 1994.

[22] C. Melin et B. Vidolf, *passive two rule based fuzzy logic controller*. Third IEEE conference on fuzzy systems, pages: 947 – 951, Orlando, Florida, juin 1994.

[23] K.S. Ray et D. D. Majumder, *application of the circle criteria for stability analysis of linear SISO and MIMO systems associated with fuzzy logic controller*. IEEE transactions on systems. Man and Cybernetics. SMC (14)-2. Pages: 1085– 1087, 1987.

[24] M. Vidyasagar, *non linear systems analysis*. Prentice Hall, 1993.

[25] G. Zames, *on the input-output stability of time varying nonlinear feedback systems. Part I: Conditions derived using concepts of loop gain, conicity and positivity*. IEEE transactions on automatic control. 11(2), pages: 282 – 238, 1966.

[26] Chuen Chien Lee, *fuzzy logic in control systems: fuzzy logic controller- part II*, IEEE Transactions on Systems, Man, and Cybernetics, Vol 20, No 2. March/April 1990. Pages: 419-435.

[27] Simon X. Yang et Hao li, *An Embedded Fuzzy Controller for a Behavior-Based Mobile Robot With Guaranteed Performance*. IEEE transactions on fuzzy systems. Vol 12, No 4. August 2004. Pages: 436-446.

Contrôleur Flou Pour La Navigation D'un Robot Mobile d'intérieur

Résumé

Dans ce travail, la poursuite d'une trajectoire par un robot mobile d'intérieur à commande différentielle est étudiée. Pour ce faire, une stratégie de poursuite relativement simple est adoptée pour assurer la souplesse des mouvements du robot. Et pour imiter les connaissances humaines dans le domaine de conduite des véhicules, un contrôleur flou est conçu. Lors de la conception de ce contrôleur l'effort était concentré sur la simplicité et l'efficacité de l'algorithme de commande. Le temps de traitement et de réponse sont des facteurs très essentiels dans la commande de processus, c'est ce qui a motivé les choix de la structure et les paramètres des différentes parties du contrôleur flou. Le but est de concevoir un contrôleur qui consomme moins de temps et qui présente des performances satisfaisantes.

Dans le cadre de ce travail et de thèse, un nouvel algorithme de commande est proposé. Il est basé sur la logique classique, sur laquelle des modifications sont introduites pour contourner les handicapes et les inconvénients de cette logique. La partition de l'univers de discours en ensembles nets est utilisée en entrée pour simplifiée la fuzzification. Cela va permettre à un élément d'appartenir à un seul ensemble. Par conséquent, on aura une seule règle active dans la table des règles au lieu de quatre pour la logique floue. Pour la sortie du contrôleur, des singletons sont utilisés, ils représentent les centres des ensembles nets de sortie. Le calcul de la valeur de sortie est très simple : on retranche une certaine quantité de la valeur du singleton impliqué. Cette quantité représente la moyenne des distances entre les valeurs des variables d'entrée aux centres de leurs ensembles d'appartenances.

Mots clés : robot mobile, poursuite de trajectoire, logique floue, logique classique modifiée.

Fuzzy Controller For An Indoor Mobile Robot Navigation

Abstract

In this work, the path following by a differential drive indoor mobile robot is studied. To do so, a relatively simple following strategy is adopted to assure the smoothness of the robot movements. And to imitate the human skills and knowledge in vehicle driving, a fuzzy controller is designed. During the design of the controller the efforts were emphasized on the simplicity and efficiency of the control algorithm. The processing and the response time are essential factors in process control; this was the motivation of the choice of the fuzzy controller structure and the parameters of its different parts. The main goal is to design a controller which consumes less time and having satisfactory performances.

Within the context of this thesis, a novel control algorithm is proposed. It is based on crisp logic on which modifications are introduced to avoid its handicaps and disadvantages. The discourse universe partition in crisp sets is used for the inputs in order to simplify the fuzzification stage. This allows an element to belong to only one set. Subsequently, there will be only one active rule in the rule base instead of four for the fuzzy logic. For the controller output, singletons are used; they represent centers of the output crisp sets. The computation of the output value is very simple: a certain quantity is subtracted from the value of the implied singleton. This quantity represents the mean of the distances of the input variable to the centers of their membership sets.

Key words: mobile robot, path following, fuzzy logic, modified crisp logic.