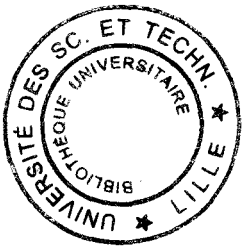
Université de Lille 1 — Sciences et Technologies de Lille

U.F.R. d'Informatique, Electronique, Electrotechnique et Automatique

Laboratoire d'Automatique I^3D

Numéro d'ordre : 2448



THÈSE

pour obtenir le grade de

Docteur de l'Université de Lille 1

Discipline : Automatique et Informatique Industrielle

Présentée et soutenue publiquement par

Wei WU

le 18 Décembre 1998

Synthèse d'un contrôleur flou par Algorithme Génétique : Application au réglage dynamique des paramètres d'un système

Christian VASSEUR	Professeur à l'USTL, Président de Jury
Mickael RUDKO	Professeur à Union College, Rapporteur
Denis HAMAD	Professeur à l'UPJV, Rapporteur
Hoang N'GUYEN	Chef de Projet DR-RENAULT, Examinateur
Daniel JOLLY	Maître de Conférences à l'USTL, Examinateur
François CABESTAING	Maître de Conférences à l'USTL, Co-directeur de recherche
Jack-Gérard POSTAIRE	Professeur à l'USTL, Co-Directeur de recherche

Remerciements

Le travail présenté dans cette thèse a été mené au Laboratoire d'Automatique I3D de l'Université des Sciences et Technologies de Lille.

J'exprime ma plus profonde gratitude à Monsieur le Professeur Christian VASSEUR pour m'avoir accueilli au sein de son laboratoire. Je le remercie d'avoir bien voulu accepter la présidence de ce jury.

Je remercie également Monsieur le Professeur Michael RUDKO pour l'intérêt porté à mes travaux et pour ses conseils. Je lui suis très reconnaissant d'avoir bien voulu juger ce travail et d'avoir pris part à mon jury.

Que soit également remercié Monsieur Denis HAMAD, Professeur à l'Université de Picardie Jules Verne, d'avoir bien voulu accepter de juger mon travail. Nos nombreuses discussions m'ont permis de formaliser des points importants de mon travail de recherche.

Je tiens également à remercier Monsieur Daniel JOLLY, Maître de Conférences à l'Université des Sciences et Technologies de Lille pour la part prise dans mon jury de thèse ainsi que pour les nombreux conseils qu'il m'a donnés sur les aspects de mon travail relevant de la logique floue.

Je tiens à remercier Monsieur Hoang N'GUYEN, Chef de Projet DR-RENAULT, d'avoir proposé le sujet à la base de ce travail et d'avoir bien voulu faire partie de ce jury.

Sans l'aide de Monsieur le Professeur Jack-Gérard POSTAIRE, cette thèse n'aurait pas abouti. Ses conseils, ses constants encouragements et sa disponibilité m'ont permis de mener à bien ce travail. Ces quelques mots sont le reflet de mes sincères remerciements.

Je souhaite remercier vivement Monsieur François CABESTAING, Maître de Conférences à l'Université des Sciences et Technologies de Lille, pour tous les encouragements, conseils apportés, pour toutes les heures consacrées à me guider durant mes travaux et

lors de la rédaction de ce travail.

Je remercie également toute l'équipe de chercheurs du Laboratoire d'Automatique I3D pour leur présence amicale et le soutien apporté.

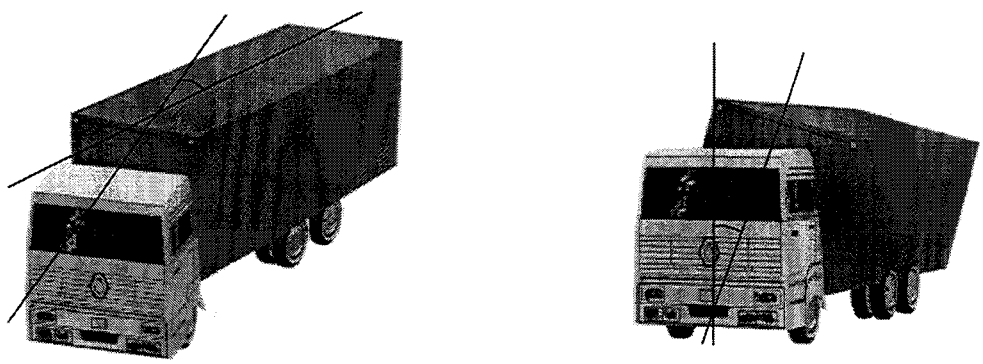
Je remercie mes amis Français et Chinois pour leur aide et le soutien constant tout au long des étapes de ce travail et en particulier Nathalie VIGIER, pour avoir relu et corrigé les fautes dans ce manuscrit.

Les derniers remerciements vont à ma famille pour tout ce qu'elle a fait pour moi et sans laquelle rien de ce qui est entre vos mains aujourd'hui n'aurait été réalisé.

Table des matières

Préambule

Durant ces dernières années, différentes études menées par les chercheurs du thème «Scènes Dynamiques» du Laboratoire d'Automatique I3D ont eu comme objectif l'amélioration de la sécurité dans les transports. En 1995, la Division de la Recherche de RENAULT nous a confié une étude portant sur la détection de la mise en portefeuille d'un semi-remorque lors d'un freinage d'urgence. Plus précisément, il s'agissait de mesurer les angles définissant la position de la remorque par rapport au tracteur, en considérant que la liaison entre ces deux éléments est de type rotule (trois degrés de liberté angulaires). Sur la figure P.1, nous avons représenté deux des angles à mesurer : l'angle de lacet (a) et l'angle de roulis (b).



(a) Angle de lacet (b) Angle de roulis

FIG. P.1 : Angles définissant la position de la remorque

Nous avons évalué une approche basée sur l'analyse d'images vidéo de la remorque acquises par deux caméras fixées sur le tracteur et dirigées vers l'arrière du camion (cf. figure P.2). Les traitements étaient basés sur la localisation de marques repérées sur l'avant de la remorque. Les positions relatives dans les deux images des marques détectées étaient utilisées pour définir la position de la remorque par rapport au tracteur.

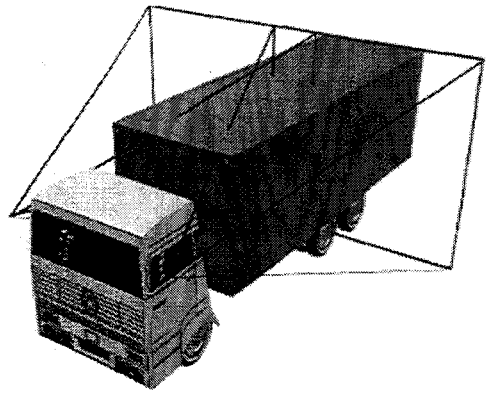


FIG. P.2 : Configuration des caméras

On constate facilement que ce problème d'estimation est surdimensionné. En effet, pour calculer les trois angles, on dispose d'un nombre assez important d'informations, qui sont les positions des différentes marques dans les deux images. La méthode d'estimation retenue doit faire intervenir toutes les données disponibles afin de diminuer l'incertitude sur le résultat. Pour satisfaire cette contrainte nous avons utilisé une méthode d'estimation basée sur une optimisation globale par un Algorithme Génétique. Cette approche sera décrite en détail dans le dernier chapitre de cette thèse.

L'ajustement des paramètres réglant la convergence d'un Algorithme Génétique est un problème complexe, sur lequel assez peu de chercheurs se sont focalisés. En général, ce choix est basé sur une heuristique qui dépend fortement du problème traité. Pourtant, l'efficacité de la méthode d'estimation, en termes de temps de calcul, est régie principalement par les paramètres utilisés. Nous avons ainsi orienté notre travail de recherche sur l'ajustement dynamique des paramètres de l'Algorithme Génétique.

Dans un Algorithme Génétique, l'évolution de la population est contrôlée principalement par les probabilités de croisement et de mutation. Durant les premières itérations, ces deux probabilités doivent prendre des valeurs élevées, pour que l'espace de recherche soit exploré au mieux. Par contre, lorsqu'on approche de la solution finale, il faut diminuer ces deux probabilités pour stabiliser la population. Un problème se pose alors : qu'entend on par «début» et «fin» du processus d'optimisation ?

Pour répondre à cette question, nous proposons d'utiliser le formalisme de la logique floue. Ainsi, l'évolution de la population durant le processus itératif peut être régie par des termes linguistiques comme «début» ou «fin». Ces qualifications sont exploitées par

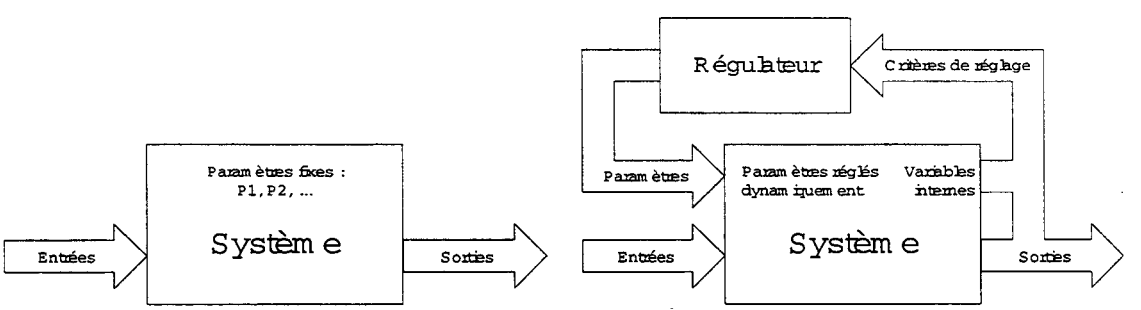
un contrôleur flou qui fournit les valeurs des probabilités de croisement et de mutation. Un problème subsiste malgré tout : comment peut on définir au mieux la structure de ce contrôleur flou ?

Dans ce préambule, nous avons décrit brièvement la démarche qui nous a permis d'aboutir aux résultats présentés dans cette thèse. Le plan du mémoire ne respecte pas l'ordre chronologique dans lequel les différents travaux ont été menés, mais plutôt un ordre logique considérant que l'objectif final est l'ajustement dynamique des paramètres d'un système complexe.

Introduction

De nos jours, des systèmes de régulation automatique du fonctionnement des processus sont intégrés dans de nombreuses applications, tant dans le domaine scientifique que technologique. Les systèmes devenant de plus en plus complexes, les performances des régulateurs utilisés ne cessent de s'améliorer. Les méthodes de réglage conventionnelles comme la commande optimale, la commande adaptative ou la commande robuste, se basent sur une connaissance plus ou moins précise du modèle mathématique du système à réguler. Lorsque le système est fortement non-linéaire, imprécis ou très complexe, il est parfois impossible de définir un modèle mathématique de son fonctionnement. Dans ce cas, les régulateurs conventionnels sont difficilement utilisables.

Dans cette thèse, nous abordons le problème du réglage dynamique des paramètres de fonctionnement d'un système. Nous considérons ainsi que les paramètres du système sont fixés par des entrées supplémentaires sur lesquelles va agir un régulateur. Pour ajuster dynamiquement les valeurs placées sur ces entrées particulières, le régulateur analyse l'évolution de la sortie du système et éventuellement celle de certaines de ses variables internes. Ce principe est schématisé sur la figure I.1.



(a) Paramètres fixes (b) Paramètres ajustés dynamiquement

FIG. I.1 : Principe du réglage dynamique des paramètres

La structure obtenue est similaire à celle des régulateurs en cascade, si on considère

que les paramètres du système sont effectivement des entrées, et que les variables internes deviennent des sorties. Malheureusement, cette constatation ne simplifie pas le problème puisque le système modifié est encore plus complexe que le système initial. Comment peut-on, dans ces conditions, définir la structure du régulateur qui assure l'ajustement dynamique des paramètres, de telle sorte que le fonctionnement global se trouve amélioré ?

Le «soft-computing» [Zad94b, Zad94a] qui regroupe principalement la logique floue [Zad65], les réseaux de neurones [Che96] et les algorithmes génétiques [Hol75] constitue une voie prometteuse pour aborder ce problème particulier de régulation. Ces méthodes font intervenir des mécanismes qui ont été observés et étudiés dans des domaines de recherche initialement très éloignés de l'informatique : linguistique et raisonnement humain pour la logique floue, physiologie humaine et animale pour les réseaux de neurones ou encore génétique et sélection naturelle pour les algorithmes évolutionnistes.

Afin de régler dynamiquement les paramètres d'un système, nous utilisons un contrôleur flou. Ce principe de régulation, basé sur des concepts relativement simples, permet de faire intervenir dans le contrôleur des connaissances acquises par un expert humain. L'intérêt principal de cette méthode est qu'elle ne nécessite pas de connaître le *fonctionnement* du système, mais simplement la façon de le *commander*. Les connaissances sont exploitées sous une forme linguistique par l'intermédiaire de règles comme « si *condition* alors *action* ». Une méthode de raisonnement (inférence) utilise ces règles pour définir les commandes qui sont envoyées au système.

Lorsque les règles de commande ne sont pas connues précisément, la conception du contrôleur flou peut être assimilée à un problème d'optimisation : comment choisir les paramètres du contrôleur afin d'assurer un fonctionnement optimal selon certains critères ? De nombreuses techniques d'optimisation ont été décrites dans la littérature (méthodes linéaires, programmation dynamique, recuit simulé ...), qui ont trouvé des applications dans la plupart des domaines scientifiques. Pour rester dans le domaine du soft-computing, nous avons choisi d'utiliser les algorithmes génétiques afin d'optimiser la structure du contrôleur flou.

Un algorithme génétique est une méthode d'exploration de l'espace des solutions possibles d'un problème exploitant des mécanismes similaires à ceux de la sélection naturelle. A chaque itération, l'algorithme génétique évalue un ensemble limité de solutions du

problème, et retient les meilleures d'entre elles (phase de sélection). Ces solutions sont combinées (phase de croisement) et éventuellement légèrement modifiées (phase de mutation) avant d'être à nouveau évaluées. A l'issue de ce processus itératif, on aboutit sous certaines conditions à la solution optimale du problème.

Dans les deux premiers chapitres de cette thèse, nous présentons de façon succincte les deux formalismes utilisés dans la suite du mémoire. Nous pourrons ainsi décrire plus facilement notre travail de recherche qui utilise des concepts relevant de la commande floue et des algorithmes génétiques.

Dans le premier chapitre, consacré à la commande floue, nous commençons par énoncer les fondements de la logique floue. Nous voyons comment elle permet d'exprimer selon un formalisme unique des informations très diverses (données incertaines ou imprécises, connaissances exprimées sous forme linguistique, ...). Ensuite, nous présentons en détails les méthodes de raisonnement flou (propositions, implications et inférence) qui constituent la base de la commande floue. Enfin, nous décrivons la structure générale d'un contrôleur flou et les différents sous-ensembles qui le constituent.

Dans le deuxième chapitre, nous décrivons le principe des algorithmes génétiques. Sur la base d'un exemple simple, nous analysons l'intérêt des différentes phases de traitement : sélection des individus, croisement et mutation. Nous exposons ensuite le théorème des schémas, qui explique de façon théorique le bien-fondé de cette méthode d'optimisation. Nous décrivons enfin plusieurs méthodes de représentation des informations traitées par un algorithme génétique sous la forme d'un codage binaire, ou par des nombres réels ou en base n.

Le troisième chapitre est consacré à la description de la méthode de synthèse d'un contrôleur flou par un algorithme génétique. Nous débutons ce chapitre par une analyse bibliographique présentant les méthodes de synthèse d'un contrôleur flou décrites dans la littérature. Nous abordons ensuite le problème du codage des paramètres sur lequel repose en grande partie l'efficacité de la phase d'optimisation. Nous proposons une méthode de codage mixte qui permet de décrire tous les paramètres régissant le fonctionnement d'un contrôleur flou. Ainsi, en utilisant un algorithme génétique, nous pouvons optimiser simultanément les différentes parties du contrôleur flou. Nous terminons la description de notre méthode par une présentation des opérateurs génétiques adaptés au mode de

codage mixte.

Dans les deux derniers chapitres de ce mémoire, nous décrivons deux exemples d'application de la méthode de synthèse décrite précédemment. Dans les deux cas, un algorithme génétique est utilisé pour définir un contrôleur flou assurant le réglage dynamique des paramètres d'un système complexe.

Dans le quatrième chapitre, nous décrivons une méthode permettant de régler dynamiquement les trois paramètres d'un régulateur standard de type PID grâce à un algorithme génétique. Cela consiste en fait à synthétiser un régulateur non linéaire en se basant sur une structure initialement linéaire. Ce principe a déjà été décrit dans la littérature par Zhao [HTXW93, ZTI93], qui utilise un contrôleur flou dont la structure est figée. Nous comparons les résultats obtenus grâce à notre méthode de synthèse à ceux initialement présentés par cet auteur.

Dans le cinquième et dernier chapitre, nous présentons une application dans laquelle le contrôleur flou assure le réglage dynamique des paramètres d'un algorithme génétique. Nous décrivons le principe de la mesure de la position angulaire d'un objet par un système de vision associé à une procédure d'optimisation. Nous mettons ensuite en évidence l'intérêt du réglage dynamique des probabilités de croisement et de mutation en comparant les résultats obtenus par quatre algorithmes différents : AG standard, AG dynamique proposé par Srinivas, AG piloté par un contrôleur flou de structure fixe et AG piloté par un contrôleur flou optimisé. Cette comparaison est réalisée en prenant en compte deux critères, la qualité de la solution et le nombre d'itérations de l'algorithme permettant d'aboutir à cette solution.

Chapitre 1

Commande en logique floue

1.1 Introduction

De nos jours, la logique floue (en anglais «fuzzy logic») est un axe de recherche important sur lequel se focalisent de nombreux scientifiques. Des retombées technologiques sont d'ores et déjà disponibles, tant dans le domaine grand public (appareils photos, machines à laver, fours à micro-onde), que dans le domaine industriel (réglage et commande de processus complexes liés à l'énergie, aux transports, à la transformation de la matière, à la robotique, aux machines-outils).

Les bases théoriques de la logique floue ont été formulées en 1965 par le professeur Lotfi A. Zadeh, de l'Université de Berkeley en Californie [Zad65, YOTN87]. Il a introduit la notion de sous-ensemble flou pour fournir un moyen de représentation et de manipulation des connaissances imparfaitement décrites, vagues ou imprécises. A cette époque, la théorie de la logique floue n'a pas été prise au sérieux excepté par quelques experts.

Dès 1975, Mamdani et Assilian publient les premiers résultats permettant une exploitation de cette théorie dans des systèmes de réglage [MA75]. En utilisant une structure de contrôleur relativement simple, ils ont obtenu de meilleurs résultats lors de la commande de certains processus que ceux fournis par un régulateur standard de type PID.

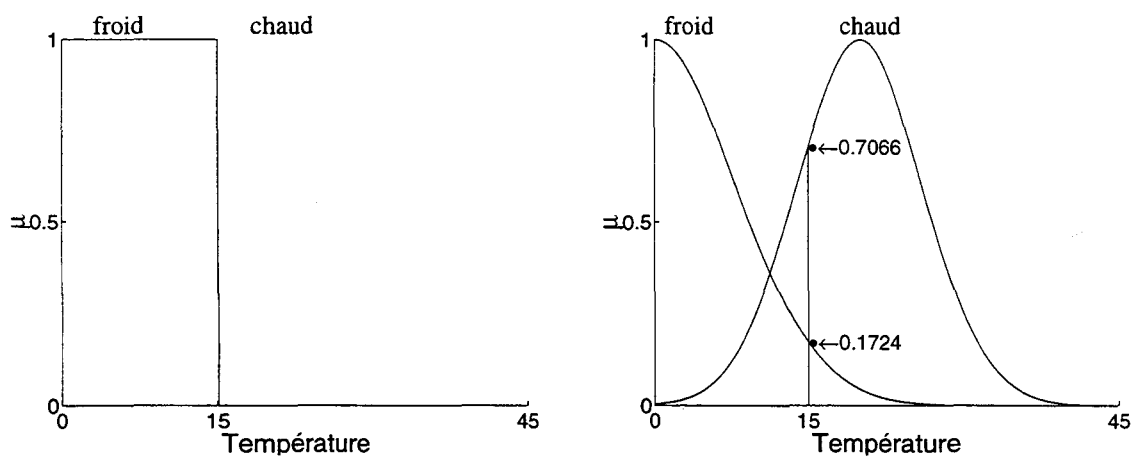
Peu de temps après, en 1977, le danois Ostergaard [Ost77] a appliqué la logique floue à la commande de tubes broyeurs pour la fabrication de ciment. A cette époque, la plupart des études concernant les systèmes de régulation exploitant la logique floue ont été réalisées en Europe [WMH77, WMM97, WM78]. A partir de 1985 environ, ce sont les

Japonais [SM84, KGN85, TW86, Tam86] qui commencent à utiliser largement la logique floue dans des produits industriels et de consommation pour résoudre des problèmes de réglage et de commande.

1.1.1 Logique classique et logique floue

Dans le cadre de la logique classique, une proposition est soit vraie, soit fausse (1 ou 0). Par exemple, la logique classique peut facilement partitionner la température d'une pièce en deux sous-ensembles, «moins de 15 degrés» et «15 degrés ou plus». La figure 1.1a montre le résultat de cette partition. Toutes les températures de moins de 15 degrés sont alors considérées comme appartenant à l'ensemble «moins de 15 degrés». On leur affecte une valeur de 1. Toutes les températures atteignant 15 degrés ou plus ne sont pas considérées comme appartenant à l'ensemble «moins de 15 degrés». On leur attribue une valeur de 0.

Cependant, le raisonnement humain s'appuie fréquemment sur des connaissances ou des données inexactes, incertaines ou imprécises. Une personne placée dans une pièce dont la température est soit de 14.95 degrés soit de 15.05 degrés, ne fera certainement pas de distinction entre ces deux valeurs. Cette personne sera pourtant capable de dire si la pièce est «froide» ou «chaude», sans pour cela utiliser de température limite ni de mesure précise.



(a) Deux ensembles selon la logique classique (b) Deux ensembles selon la logique floue

FIG. 1.1 : Classification des températures d'une pièce en deux ensembles

La logique floue permet de définir des sous-ensembles, comme «froid» ou «chaud», en

introduisant la possibilité pour une valeur d'appartenir plus ou moins à chacun de ces sous-ensembles.

1.1.2 Valeurs analogiques et logique floue

Lorsqu'on mesure une grandeur physique, on obtient une valeur qui peut ensuite être utilisée dans une série de calculs. Les grandeurs physiques sont en général continues (sauf par exemple en physique quantique) et le résultat de la mesure est un nombre réel. Dans bon nombre de systèmes de régulation ou de commande, on utilise directement la valeur de la mesure en tant qu'entrée du contrôleur.

Pourtant, réaliser une mesure sans tenir compte de sa précision est indigne d'un bon physicien. Non seulement la mesure est imprécise (le plus souvent à cause de l'appareil de mesure), mais elle peut également être incertaine puisqu'aucun appareil de mesure n'est parfaitement fiable : un capteur défectueux peut continuer à fournir une mesure erronée sans que le système de régulation en soit informé.

La logique floue permet de faire intervenir les notions d'imprécision et d'incertitude dans un système. Cela permet par exemple de faire intervenir une température «d'environ 15 degrés» dans un contrôleur flou. L'incertitude et l'imprécision peuvent également être prises en compte dans le cadre de la logique floue quand on utilise une connaissance issue d'un expert humain. Comment pourrait-on utiliser avec des outils standard une connaissance humaine du genre : «il pleut souvent en hiver» ?

1.2 Sous-ensembles flous

Dans cette section, nous décrivons rapidement les fondements mathématiques de la théorie des sous-ensemble flous [BM93, DP94, BM95, GY95]. Dans la théorie ensembliste classique, l'appartenance d'un élément à un sous-ensemble est définie par une valeur logique standard : 1 si l'élément appartient au sous-ensemble, 0 sinon. Dans la théorie floue, un élément peut appartenir *en partie* à un sous-ensemble : son degré d'appartenance est décrit par une valeur comprise entre 0 et 1.

1.2.1 Définitions

Etant donné un ensemble de référence X qui peut être fini ou infini, dénoté par ses éléments $\{x\}$, on peut indiquer les éléments $\{x\}$ qui appartiennent à une certaine classe de X (on leur donne une valeur 1) et ceux qui n'y appartiennent pas (on leur donne une valeur 0). Cette classe est alors un sous-ensemble classique de X caractérisé par une *fonction caractéristique* $\mathcal{X}_A$ prenant simplement deux valeurs 0 ou 1 :

$$\mathcal{X}_A : X \rightarrow \{0, 1\} \tag{1.1}$$

Si l'appartenance de certains éléments de X à une classe n'est pas absolue (l'élément appartient *un peu* au sous-ensemble), on peut remplacer la fonction caractéristique par une *fonction d'appartenance* qui prend ses valeurs dans l'intervalle $[0, 1]$. Cette classe est appelée sous-ensemble flou de X. L'ensemble X sera également appelé *univers du discours* tout au long de ce travail.

Définition 1.1 (Sous-ensemble flou) *Un sous-ensemble flou A dans un univers du discours X est caractérisé par sa* fonction d'appartenance $\mu_A(x)$ *qui associe à chaque élément x de X une valeur dans l'intervalle des nombres réels* $[0, 1]$.

$$\mu_A : X \rightarrow [0, 1]. \tag{1.2}$$

Ainsi un sous-ensemble flou A dans X peut être représenté par un ensemble de couples ordonnés

$$A = \{(x, \mu_A(x)) | x \in X\}. \tag{1.3}$$

Nous utiliserons parfois la terminologie *ensemble flou* pour citer un sous-ensemble flou dans la suite de ce mémoire.

Le sous-ensemble classique n'est en fait qu'un cas particulier de sous-ensemble flou dont la fonction d'appartenance ne prend que les valeurs 0 ou 1. Un sous-ensemble flou A de X est aussi souvent représenté par la notation suivante qui indique pour tout élément x de X son degré $\mu_A(x)$ d'appartenance à A :

$$A = \int_X \mu_A(x)/x \qquad \text{si X est continu} \tag{1.4}$$

et

$$A = \sum_{x_i \in X} \mu_A(x_i)/x_i \qquad \text{si X est discret} \tag{1.5}$$

Comme les valeurs $\mu_A(x_i)$ représentent les degrés d'appartenance avec lesquels les x_i appartiennent à A, si $\mu_A(x_i)$ prend la valeur 1 pour tous les éléments de X, cela signifie que A est identique à X. Au contraire, A est vide si $\mu_A(x_i)$ prend la valeur 0 sur tout X.

Les gabarits de fonctions d'appartenance les plus utilisés sont représentés sur la figure 1.2. En commande floue, les fonctions d'appartenance utilisées peuvent théoriquement être quelconques. Pourtant on choisit souvent des fonctions triangulaires ou trapézoïdales afin de simplifier les calculs.

Définition 1.2 (Support) *Le support d'un sous-ensemble flou A dans un univers du discours X est le sous-ensemble (au sens classique du terme) des éléments de X pour lesquels la fonction d'appartenance prend une valeur strictement positive. C'est l'ensemble des éléments de X qui appartiennent* au moins un peu *à A :*

$$S(A) = \{x | \mu_A(x) > 0\}. \tag{1.6}$$

Définition 1.3 (Point de croisement) *Le point de croisement d'un sous-ensemble flou A dans un univers du discours X est le sous-ensemble des éléments de X pour lesquels la fonction d'appartenance prend une valeur égale à 0.5. C'est l'ensemble des éléments de X qui appartiennent* autant *à A qu'à son complémentaire :*

$$C(A) = \{x | \mu_A(x) = 0.5\}. \tag{1.7}$$

Définition 1.4 (Noyau) *Le noyau d'un sous-ensemble flou A dans un univers du discours X est le sous-ensemble des éléments de X pour lesquels la fonction d'appartenance vaut 1. C'est l'ensemble des points qui appartiennent* intégralement *à A :*

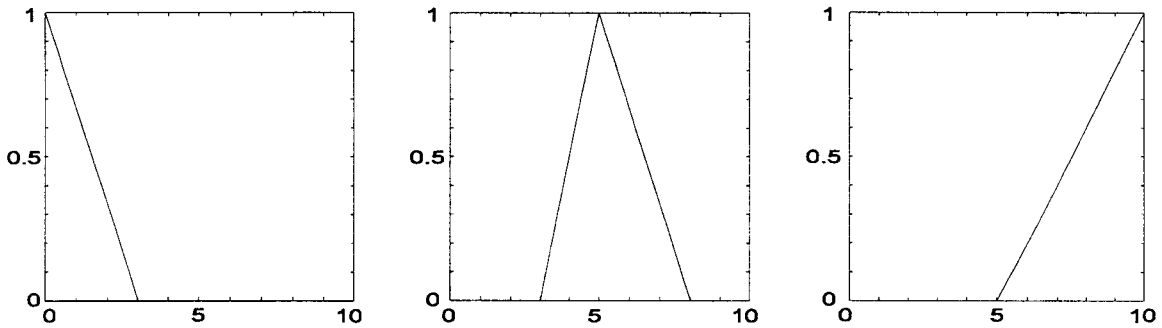
$$N(A) = \{x | \mu_A(x) = 1\}. \tag{1.8}$$

Définition 1.5 (Hauteur) *La hauteur d'un sous-ensemble flou A dans un univers du discours X est la valeur maximale prise par la fonction d'appartenance μ_A sur l'ensemble X. C'est le plus fort degré avec lequel un élément de X appartient à A :*

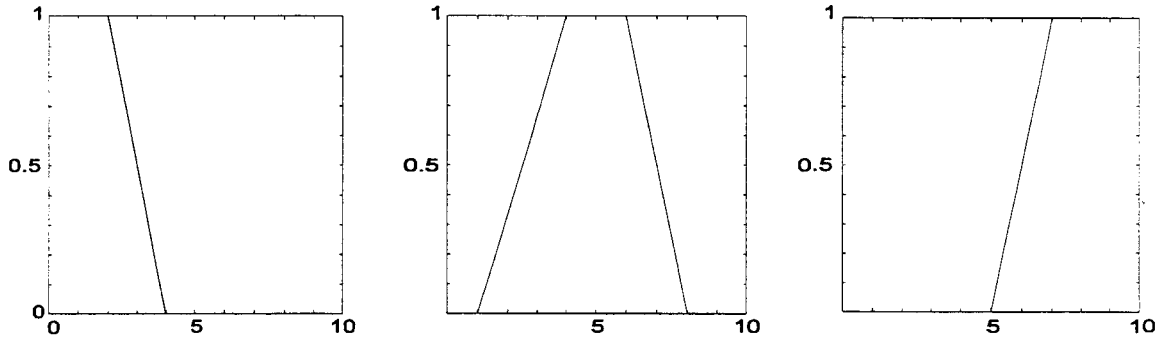
$$H(A) = sup_{x \in X} \mu_A(x). \tag{1.9}$$

Définition 1.6 (α−coupe) *Pour toute valeur α de l'intervalle $[0, 1]$, on appele $\alpha-$coupe d'un sous-ensemble flou A de X, le sous-ensemble noté A_α des éléments de X pour lesquels la fonction d'appartenance est supérieure ou égale à α :*

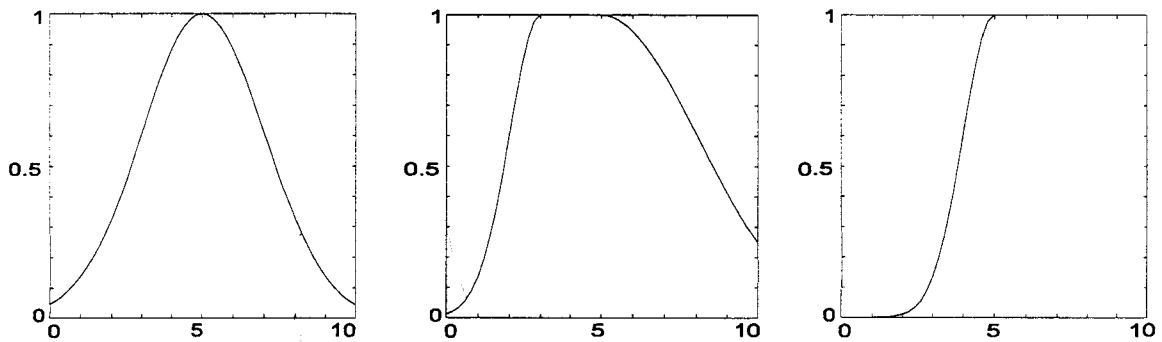
$$A_\alpha = \{x \in X | \mu_A(x) \geq \alpha\} \tag{1.10}$$



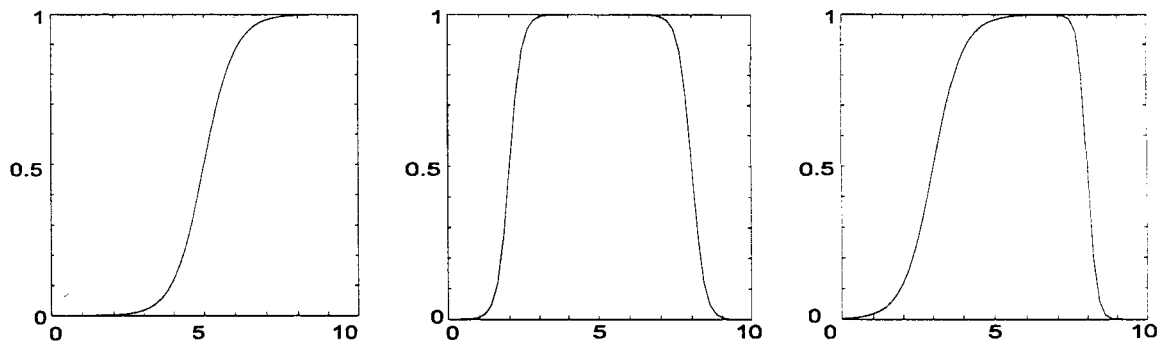
(a) Fonctions d'appartenance triangulaires



(b) Fonctions d'appartenance trapézoïdales



(c) Fonctions d'appartenance Gaussiennes



(d) Fonctions d'appartenance sigmoïdes

FIG. 1.2 : Les fonctions d'appartenance les plus utilisées

Ce sous-ensemble est défini par la fonction caractéristique suivante :

$$\mathcal{X}_{A_\alpha} = 1 \qquad si\ et\ seulement\ si\ \mu_A(x) \geq \alpha. \tag{1.11}$$

Si nous choisissons $\alpha = 0$, alors A_α est l'univers du discours X. Si nous choisissons $\alpha = 1$, alors A_α est le noyau de A, $N(A)$. Sur la figure 1.3 nous illustrons les définitions précédentes par un exemple. Le sous-ensemble flou A est celui des températures *tièdes* dans l'univers du discours X des températures.

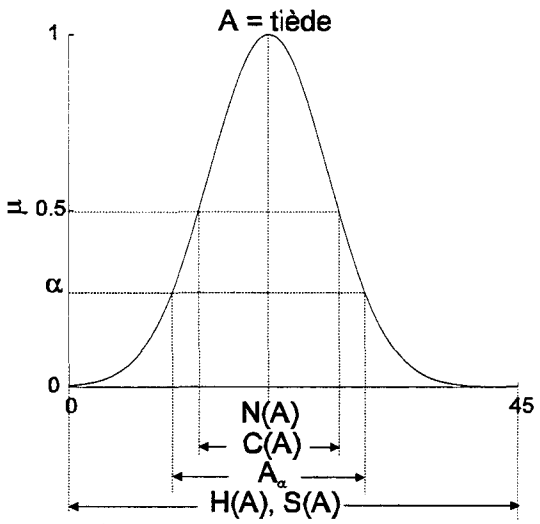


FIG. 1.3 : Les concepts flous décrivant une température *tiède*

Théorème 1.1 (Théorème de décomposition) *Le théorème de décomposition montre qu'il est possible de reconstituer un sous-ensemble flou A à partir de ses $\alpha-$coupes. En fait, le théorème permet de définir la fonction d'appartenance μ_A à partir des fonctions caractéristiques des $\alpha-$coupes :*

$$\forall x \in X \qquad \mu_A(x) = sup_{\alpha \in (0,1]} \alpha \cdot \mathcal{X}_{A_\alpha}(x) \tag{1.12}$$

1.2.2 Opérations sur les sous-ensembles flous

Supposons que A et B sont deux sous-ensembles flous définis dans un univers du discours X par les fonctions d'appartenance μ_A et μ_B. On peut définir des opérations ensemblistes telles que l'égalité, l'inclusion, l'intersection, l'union et le complément grâce à des opérations sur les fonctions d'appartenance.

Définition 1.7 (Egalité) *A et B sont dits égaux, propriété que l'on note $A = B$, si leurs fonctions d'appartenance prennent la même valeur en tout point de X :*

$$\forall x \in X \qquad \mu_A(x) = \mu_B(x). \tag{1.13}$$

Définition 1.8 (Inclusion) *A est dit inclus dans B, propriété que l'on note $A \subseteq B$, si tout élément x de X qui appartient à A appartient aussi à B avec un degré au moins aussi grand :*

$$\forall x \in X \qquad \mu_A(x) \leq \mu_B(x). \tag{1.14}$$

Les définitions d'intersection, d'union et de complément de sous-ensembles flous définies ci-dessous font intervenir les opérateurs de minimum, maximum et de complémentation à 1. Cela correspond à une extension triviale des opérateurs ensemblistes standard. D'autres définitions sont également possibles lorsque l'on fait intervenir les concepts de normes triangulaires et de conormes triangulaires, qui seront présentés dans la section suivante.

Définition 1.9 (Intersection) *L'intersection de A et B, que l'on note $A \cap B$, est le sous-ensemble flou constitué des éléments de X affectés du plus petit des deux degrés d'appartenance μ_A et μ_B :*

$$\forall x \in X \qquad \mu_{A \cap B} = min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x). \tag{1.15}$$

Dans cette définition, *min* et $\wedge$ désignent l'opérateur de calcul du minimum des deux valeurs.

Définition 1.10 (Union) *L'union de A et B, que l'on note $A \cup B$, est le sous-ensemble flou constitué des éléments de X affectés du plus grand des deux degrés d'appartenance μ_A et μ_B :*

$$\forall x \in X \qquad \mu_{A \cup B} = max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x). \tag{1.16}$$

Dans cette définition, *max* et $\vee$ désignent l'opérateur de calcul du maximum de deux valeurs.

Définition 1.11 (Complément) *Le complément de A, que l'on note A^c, est le sous-ensemble flou de X constitué des éléments x lui appartenant* d'autant plus *qu'ils appartiennent* peu *à A :*

$$\forall x \in X \qquad \mu_{A^c}(x) = 1 - \mu_A(x). \tag{1.17}$$

Définition 1.12 (Produit cartésien) *Le produit cartésien est une méthode de combinaison de sous-ensemble flous définis sur des univers du discours différents. Par exemple, cela permet de définir simplement ce que signifie* chaud et humide *sur un univers du discours* température et hygrométrie.

Soient A_1, A_2, ... ,A_n *des sous-ensembles flous définis respectivement dans les univers du discours* X_1, X_2, ... ,X_n. *Le produit cartésien de* A_1, A_2, ... , A_n, *que l'on note* $A = A_1 \times A_2 \times \cdots \times A_n$, *est le sous-ensemble flou défini dans l'univers du discours produit* $X = X_1 \times X_2 \times \cdots \times X_n$ *par la fonction d'appartenance :*

$$\forall x = (x_1, x_2, \ldots, x_n) \in X \qquad \mu_A(x) = min(\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_n}(x_n)) = \bigwedge_{i=1}^{n} \mu_{A_i}(x_i) \tag{1.18}$$

1.2.3 Normes et conormes triangulaires

Comme nous l'avons vu dans le paragraphe précédent, la définition d'une opération entre ensembles flous est basée sur une combinaison des fonctions d'appartenance. Les définitions les plus simples utilisent les opérations de minimum, maximum et de complément à 1. Les normes et conormes triangulaires constituent une généralisation des opérations de combinaison de type minimum ou maximum.

Définition 1.13 (Norme triangulaire, t-norme) *Une norme triangulaire* $\top$ *est une fonction définie sur l'ensemble* $[0,1] \times [0,1]$ *et prenant ses valeurs dans l'intervalle* $[0,1]$, *qui satisfait les conditions suivantes :*

$\forall x, y \in [0,1]$: $\top(x,y) = \top(y,x)$ *(commutativité),*

$\forall x, y, z \in [0,1]$: $\top(x, \top(y,z)) = \top(\top(y,z), z)$ *(associativité),*

$\forall x, y, z, t \in [0,1]$: $\top(x,y) \leq \top(z,t)$ *si* $x \leq z$ *et* $y \leq t$ *(monotonie),*

$\forall x \in [0,1]$: $\top(0,0) = 0, \top(x,1) = \top(1,x) = x$ *(élément neutre 1).*

Définition 1.14 (Intersection définie par une t-norme) *On vérifie simplement que l'opérateur minimum est une t-norme. La définition d'une opération d'intersection entre deux sous-ensembles flous se réfère à une t-norme, qui remplace l'opérateur* minimum *:*

$$\forall x \in X \qquad \mu_{A \cap B}(x) = \top(\mu_A(x), \mu_B(x)). \tag{1.19}$$

Définition 1.15 (Conorme triangulaire, t-conorme) *Une conorme triangulaire* $\perp$ *est une fonction définie sur l'ensemble* $[0,1] \times [0,1]$ *et prenant ses valeurs dans l'intervalle*

$[0,1]$, *qui satisfait les conditions suivantes :*

$\forall x, y \in [0,1]$: $\perp(x,y) = \perp(y,x)$ *(commutativité),*

$\forall x, y, z \in [0,1]$: $\perp(x, \perp(y,z)) = \perp(\perp(y,z), z)$ *(associativité),*

$\forall x, y, z, t \in [0,1]$: $\perp(x,y) \leq \perp(z,t)$ *si* $x \leq z$ *et* $y \leq t$ *(monotonie),*

$\forall x \in [0,1]$: $\perp(1,1) = 1, \perp(x,0) = \perp(0,x) = x$ *(élément neutre 0).*

Définition 1.16 (Union définie par une t-conorme) *On vérifie simplement que l'opérateur maximum est une t-conorme. La définition d'une opération d'union entre deux sous-ensembles flous se réfère à une t-conorme, qui remplace l'opérateur* maximum *:*

$$\forall x \in X \qquad \mu_{A \cup B}(x) = \perp(\mu_A(x), \mu_B(x)). \tag{1.20}$$

Les t-normes et t-conormes les plus utilisées sont représentées dans le tableau 1.1 :

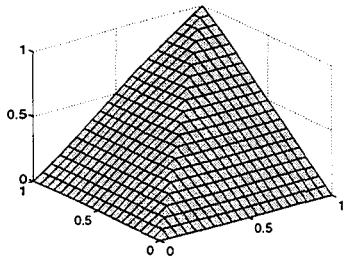
t-norme	t-conorme	négation	nom
$min(x,y)$	$max(x,y)$	$1-x$	Zadeh
$x \cdot y$	$x + y - x \cdot y$	$1-x$	probabiliste
$max(x+y-1, 0)$	$min(x+y, 1)$	$1-x$	Lukasiewicz
$\frac{xy}{\gamma+(1-\gamma)(x+y-xy)}$	$\frac{x+y-xy-(1-\gamma)xy}{1-(1-\gamma)xy}$	$1-x$	Hamacher($\gamma > 0$)
$\begin{cases} x & \text{si } y = 1 \\ y & \text{si } x = 1 \\ 0 & \text{sinon} \end{cases}$	$\begin{cases} x & \text{si } y = 0 \\ y & \text{si } x = 0 \\ 1 & \text{sinon} \end{cases}$	$1-x$	Weber

TAB. 1.1 : Définitions des t-normes et t-conormes les plus utilisées

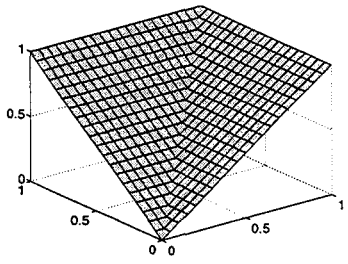
La figure 1.4 montre les représentations graphiques des t-normes et t-conormes décrites dans le tableau 1.1. Les opérateurs de Hamacher sont des opérateurs plus généraux dont les propriétés dépendent d'un paramètre γ. Si γ tend vers l'infini, les opérateurs de Hamacher deviennent identiques à ceux de Weber. Si $\gamma = 1$, on retrouve les opérateurs probabilistes.

Théorème 1.2 (Propriétés des t-normes et des t-conormes) *On peut démontrer les propriétés suivantes, vérifiées par toute t-norme et t-conorme :*

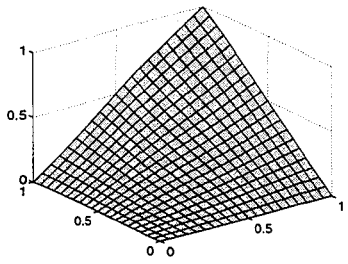
$$\begin{aligned} &\forall x, y \in [0,1] \\ &\top_{Weber}(x,y) \leq \top(x,y) \leq \top_{Zadeh}(x,y), \\ &\perp_{Zadeh}(x,y) \leq \perp(x,y) \leq \perp_{Weber}(x,y). \end{aligned} \tag{1.21}$$



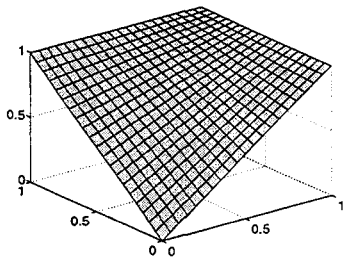
t-norme de Zadeh



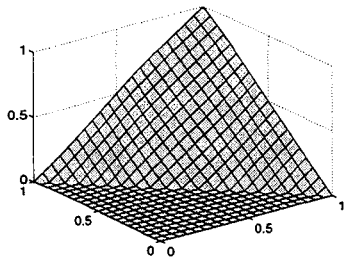
t-conorme de Zadeh



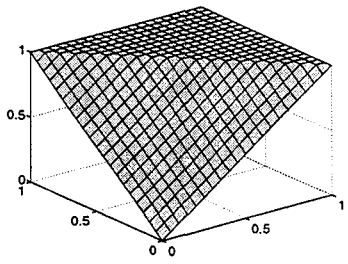
t-norme probabiliste



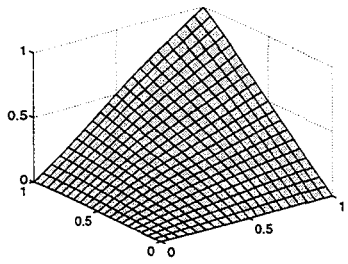
t-conorme probabiliste



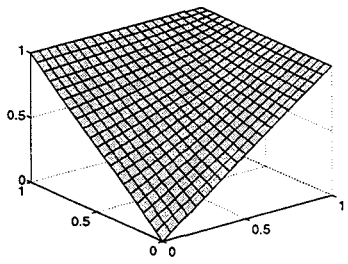
t-norme de Lukasiewicz



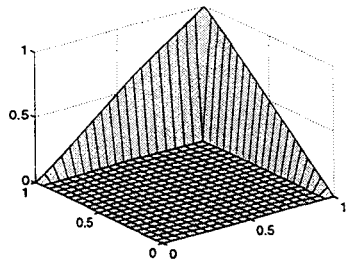
t-conorme de Lukasiewicz



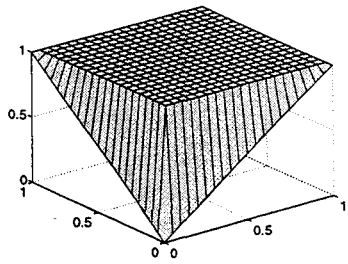
t-norme de Hamacher ($\gamma = 0.5$)



t-conorme de Hamacher ($\gamma = 0.5$)



t-norme de Weber



t-conorme de Weber

FIG. 1.4 : t-normes et t-conormes les plus utilisées

Exemple 1.1. *Si les fonctions d'appartenance de la température d'une pièce aux catégories «froid» et «tiède» sont celles présentées sur la figure 1.1b, les sous-ensembles flous «froid et tiède» et «froid ou tiède» pour les différentes t-normes et t-conormes sont représentés sur la figure 1.5.*

1.3 Relations floues

Dans la section précédente, nous avons présenté le produit cartésien de deux sous-ensembles flous. Cette méthode permet de construire un sous-ensemble flou sur un univers du discours qui est lui même un produit cartésien. Dans l'exemple présenté précédemment sur un univers du discours *température et hygrométrie*, le produit cartésien n'est assurément pas une bonne méthode pour définir le sous-ensemble flou *chaud et humide* : en effet, les grandeurs initiales ne sont pas indépendantes.

1.3.1 Concept de relation floue

Une relation floue est un concept qui permet de définir un sous-ensemble flou sur un univers du discours qui est un produit cartésien, tout en tenant compte des relations qui relient les univers du discours initiaux. La fonction d'appartenance définissant ce sous-ensemble flou ne se résume pas forcément à une simple combinaison de fonctions d'appartenance définissant des sous-ensembles flous dans les univers du discours initiaux.

Définition 1.17 (Relations floues) *Une relation floue du n-ième ordre est un sous-ensemble flou dans l'univers du discours produit* $X = X_1 \times X_2 \times \cdots \times X_n$, *qui s'exprime par :*

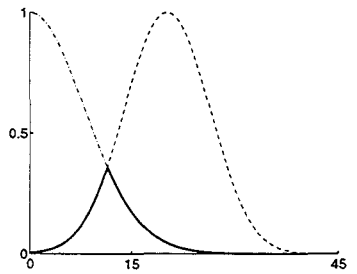
$$R_X = \{((x_1, x_2, \ldots, x_n), \mu_R(x_1, x_2, \ldots, x_n)) | (x_1, x_2, \ldots, x_n) \in X\}. \tag{1.22}$$

où $\mu_R(x_1, x_2, \ldots, x_n)$ *est une fonction de mappage :*

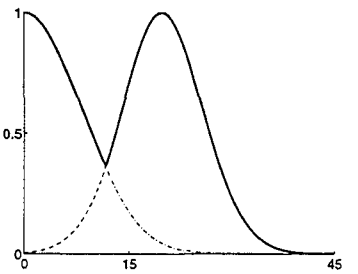
$$\mu_R(x_1, x_2, \ldots, x_n) : X_1 \times X_2 \times \cdots \times X_n \rightarrow [0, 1] \tag{1.23}$$

Une relation floue du deuxième ordre R, définie entre deux univers du discours de cardinaux infinis X et Y, peut s'exprimer de la façon suivante :

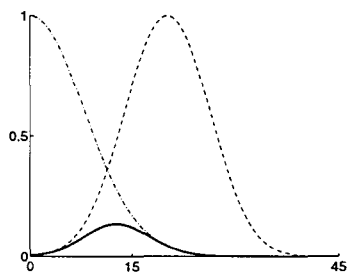
$$R_{X \times Y} = \int_{X \times Y} \mu_R(x, y) | (x, y); \quad x \in X, y \in Y \tag{1.24}$$



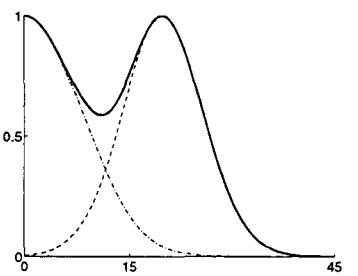
t-norme de Zadeh



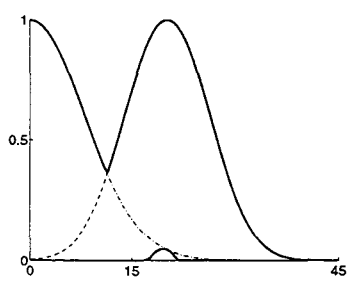
t-conorme de Zadeh



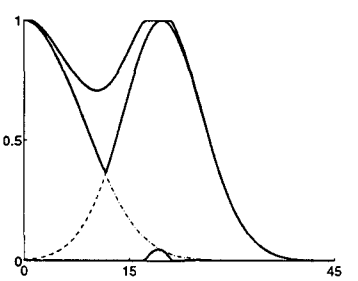
t-norme probabiliste



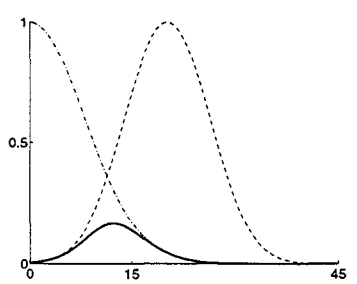
t-conorme probabiliste



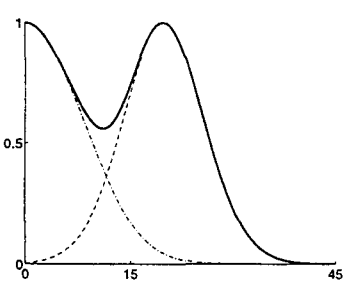
t-norme de Lukasiewicz



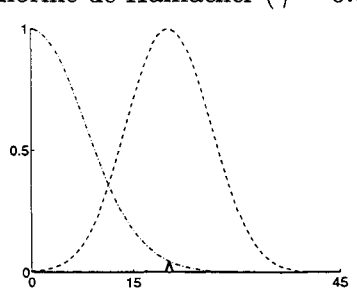
t-conorme de Lukasiewicz



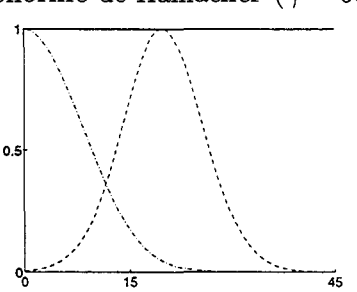
t-norme de Hamacher ($\gamma = 0.5$)



t-conorme de Hamacher ($\gamma = 0.5$)



t-norme de Weber



t-conorme de Weber

FIG. 1.5 : Exemples de définition de «froid et tiède» et de «froid ou tiède»

Si les univers du discours X et Y sont des ensembles de cardinaux finis, la relation du deuxième ordre R peut s'exprimer sous la forme d'une matrice constituée des valeurs de la fonction d'appartenance :

$$R = \begin{bmatrix} \mu_R(x_1, y_1) & \mu_R(x_1, y_2) & \cdots & \mu_R(x_1, y_n) \\ \mu_R(x_2, y_1) & \mu_R(x_2, y_2) & \cdots & \mu_R(x_2, y_n) \\ \vdots & \vdots & \ddots & \vdots \\ \mu_R(x_m, y_1) & \mu_R(x_m, y_2) & \cdots & \mu_R(x_m, y_n) \end{bmatrix} \tag{1.25}$$

1.3.2 Opérations sur les relations floues

Puisqu'une relation floue entre X et Y est un sous-ensemble flou défini sur l'univers du discours produit $X \times Y$, les opérations définies sur les sous-ensembles flous permettent de définir des opérations de combinaison des relations floues. Si R et S sont deux relations floues définies entre X et Y, l'union $R \cap S$, l'intersection $R \cup S$, le complémentaire R^C, l'égalité $R = S$ et l'inclusion $R \sqsubseteq S$, sont définies par les relations suivantes appliquées sur les fonctions d'appartenance :

$$\mu_{R \cap S}(x, y) = \mu_R(x, y) \top \mu_S(x, y)$$
$$\mu_{R \cup S}(x, y) = \mu_R(x, y) \perp \mu_S(x, y)$$
$$\mu_{R^C}(x, y) = 1 - \mu_R(x, y)$$
$$\forall x \in X, \forall y \in Y, R = S \Longleftrightarrow \mu_R(x, y) = \mu_S(x, y)$$
$$\forall x \in X, \forall y \in Y, R \sqsubseteq S \Longleftrightarrow \mu_R(x, y) \leq \mu_S(x, y)$$

On peut également définir des opérations de combinaison de relations floues lorsque les univers du discours sont distincts. Cela permet de déduire une relation entre les univers du discours X et Z lorsqu'on connait des relations existant entre X et Y d'une part et Y et Z d'autre part.

Définition 1.18 (Composition de relations floues) *Si R et S sont deux relations floues définies entre X et Y d'une part et Y et Z d'autre part, la composition de R et S est une relation floue notée par $R \circ S$ définie par :*

$$R \circ S = \{[(x, z), sup(\mu_R(x, y) \top \mu_S(y, z))], x \in X, y \in Y, z \in Z\}. \tag{1.26}$$

En commande floue, on choisit quelquefois l'opérateur *min* en tant que t-norme $\top$. Puisque *sup* est une opération qui consiste à rechercher le maximum d'un ensemble de valeurs, la composition définie en 1.26 est alors appelée «composition max-min».

1.4 Raisonnement en logique floue

Un des apports principaux de la logique standard a été la formalisation des méthodes de déduction, qui sont en quelque sorte un outil de *raisonnement.* Les méthodes de déduction utilisées en logique standard permettent de définir une nouvelle certitude à partir d'autres connaissances certaines. Dans le cadre de la logique floue, il est possible de généraliser les méthodes de raisonnement lorsqu'on dispose de connaissances incertaines ou imprécises.

1.4.1 Variables linguistiques

Pour qu'il soit possible de raisonner simplement sur un problème, il faut tout d'abord spécifier clairement les connaissances disponibles. Les variables linguistiques permettent de décrire dans un cadre très général la connaissance acquise sur une variable, même lorsqu'elle est vague ou imprécise.

Définition 1.19 (Nombre flou) *Un nombre flou Q est un sous-ensemble flou défini sur l'ensemble des nombres réels R, qui possède deux propriétés. Il est normalisé :*

$$max(\mu_F(x)) = 1, \tag{1.27}$$

et convexe :

$$\mu_F(\lambda \cdot x_1 + (1-\lambda) \cdot x_2) \geq min(\mu_F(x_1), \mu_F(x_2)) \tag{1.28}$$
$$x1, x2 \in R, \lambda \in [0,1].$$

On appelle intervalle flou un nombre flou qui correspond à un intervalle dont les bornes sont connues de façon imprécise.

Définition 1.20 (Partition floue) *Un ensemble de sous-ensembles flous $A_1, A_2, \cdots, A_n$ définis sur un univers du discours X par les fonctions d'appartenance $\mu_{A_1}(x), \mu_{A_2}(x), \cdots, \mu_{A_n}(x)$ est une* partition floue *si et seulement si :*

$$\forall x \in X \qquad \sum_{i=1}^{n} \mu_{A_i}(x) = 1. \tag{1.29}$$

Définition 1.21 (Variables linguistiques) *Une variable linguistique est caractérisée par un quintuplet $(V, T(V), X, G, M)$, dans lequel :*

- *V est le nom de la variable définie sur l'univers du discours X.*
- *$T(V) = A_1, A_2, \ldots, A_n$ est un ensemble des termes linguistiques qui sont des nombres flous, définissant des restrictions sur les valeurs que prend V dans X.*
- *G est un ensemble de règles syntaxiques qui permettent de former d'autres termes linguistiques à partir de $T(V)$. On les appelle modificateurs linguistiques. Par exemple, pour définir la fonction d'appartenance du terme linguistique «pas A» on utilise l'expression $\mu_{pas\,A} = 1 - \mu_A$. On peut aussi définir des nouveaux termes linguistiques comme «très très A», «très A», «assez A», «comparable à A», «un peu A», «un petit peu A» en utilisant par exemple les fonctions d'appartenance : ${\mu_A}^4$, ${\mu_A}^2$, ${\mu_A}^{1.25}$, ${\mu_A}^{0.75}$, ${\mu_A}^{0.5}$, ${\mu_A}^{0.25}$.*
- *M est l'ensemble des règles sémantiques qui permettent de définir les termes linguistiques.*

Exemple 1.2. *Afin de décrire la température d'une pièce par une variable linguistique, on peut utiliser l'ensemble des termes suivants :* T(V) = {froid, tiède, chaud}. *D'autres termes linguistiques peuvent être formés en utilisant le modificateur linguistique «très», comme «très froid» ou «très très chaud». En considérant que l'univers du discours est l'intervalle $[0, 45]$, on peut utiliser les règles sémantiques suivantes pour définir les termes linguistiques : «froid» est «une température environ inférieure à 10 degrés», «tiède» est «une température d'environ 17 degrés» et «chaud» est «une température environ supérieure à 24 degrés». Ces termes peuvent être caractérisés par les fonctions d'appartenance représentées sur la figure 1.6. N'importe quelle fonction d'appartenance, par exemple la fonction «froid», définit un nombre flou.*

1.4.2 Propositions floues

Une proposition floue est définie à partir d'un ensemble de variables linguistiques afin de représenter une connaissance. Par exemple, «la température de la pièce est froide».

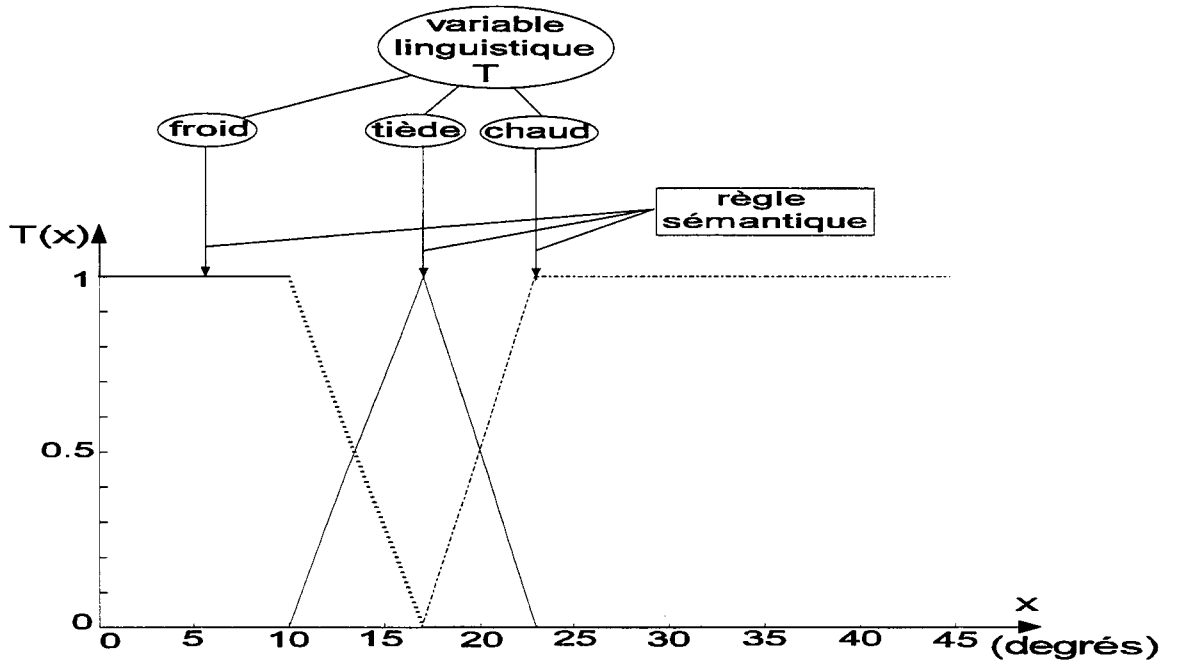


FIG. 1.6 : Variable linguistique $(V, T(V), X, G, M)$ pour décrire la température

Définition 1.22 (Propositions floues élémentaires) *Une forme élémentaire de proposition floue est définie à partir d'une seule variable linguistique* $(V, T(V), X, G, M)$ *et exprimée simplement par la phrase :*

$$p : \quad \mathcal{V} \text{ est } A$$

où $\mathcal{V}$ *est une variable qui prend sa valeur dans l'univers du discours* X*, et* A *est l'un des termes linguistiques de* $T(V)$*. Une valeur particulière* $\mathcal{V} = v$ *appartient à* A *avec le degré d'appartenance* $\mu_A(v)$*. Cela permet de définir la* valeur de vérité $V(p)$ *de la proposition lorsque* $\mathcal{V}$ *vaut* v *:*

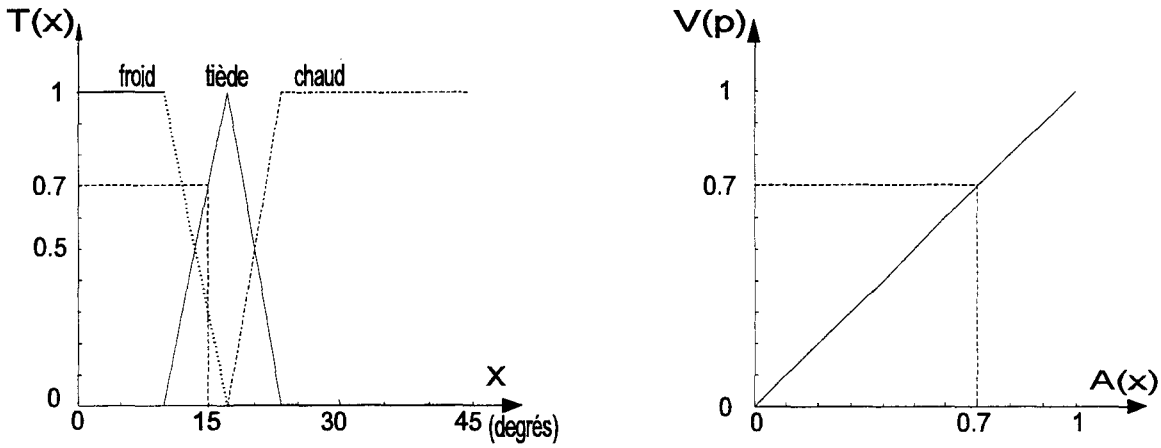
$$V(p) = \mu_A(v). \tag{1.30}$$

Exemple 1.3. *Supposons que l'on veuille définir la température* $\mathcal{V}$ *d'une pièce en utilisant la variable linguistique définie dans l'exemple 1.2. La proposition floue correspondante est exprimée par la phrase :*

$$p : \quad \mathcal{V} \text{ est tiède.}$$

A partir de la fonction d'appartenance définissant «tiède», nous trouvons que le degré d'appartenance d'une température de 15 degrés au sous-ensemble flou «tiède» est de 0.7,

comme on le voit sur la figure 1.7a. La valeur de vérité $V(p)$ de la proposition p lorsque la température vaut 15 degrés est donc de 0.7.



(a) (b)

FIG. 1.7 : Valeur de vérité de la proposition floue p : la température est tiède

Nous voyons clairement que le rôle de la fonction $V(p)$ est de définir un lien entre les sous-ensembles flous et les propositions floues. Comme la valeur de vérité $V(p)$ est exprimée à partir de la fonction d'appartenance $\mu_A(x)$ dans l'équation 1.30 et qu'elle est introduite pour décrire clairement une connaissance, nous remplaçons directement la valeur de vérité $V(p)$ par la fonction d'appartenance $\mu_A(x)$. Une explication plus détaillée est disponible dans [GY95].

Définition 1.23 (Propositions floues générales) *Une proposition floue générale est obtenue par la composition de propositions élémentaires «x est A», «y est B»,... pour des variables x, y, ... supposées non indépendantes.*

Habituellement les propositions floues générales sont classées en quatre types :

- La conjonction de propositions floues élémentaires :

$$p : (\mathcal{X}_1 \text{ est } A_1) \text{ et } \cdots \text{ et } (\mathcal{X}_n \text{ est } A_n).$$

Dans ce cas, la conjonction est associée au produit cartésien $A_1 \times A_2 \times \cdots \times A_n$ caractérisant la variable conjointe $(\mathcal{X}_1,\cdots,\mathcal{X}_n)$ sur les univers de discours $X_1 \times X_2 \times \cdots \times X_n$. Sa valeur de vérité est alors définie par :

$$V(p) = min\{\mu_{A_1}(\mathcal{X}_1), \cdots, \mu_{A_n}(\mathcal{X}_n)\} \tag{1.31}$$

- La disjonction de propositions floues élémentaires :

$$p : (\mathcal{X}_1 \text{ est } A_1) \text{ ou } \ldots \text{ ou } (\mathcal{X}_n \text{ est } A_n).$$

La valeur de vérité de la disjonction sur les univers du discours $X_1 \times X_2 \times \cdots \times X_n$ est définie par :

$$V(p) = max\{\mu_{A_1}(\mathcal{X}_1), \cdots, \mu_{A_n}(\mathcal{X}_n)\} \tag{1.32}$$

- Les implications entre propositions floues :

Règle 1 : Si $(\mathcal{X}$ est $A_1)$; alors $(\mathcal{Y}$ est $B_1)$,
Règle 2 : Si $(\mathcal{X}$ est $A_2)$; alors $(\mathcal{Y}$ est $B_2)$,
...
Règle n : Si $(\mathcal{X}$ est $A_n)$; alors $(\mathcal{Y}$ est $B_n)$.

Les implications seront présentées plus en détail dans le paragraphe suivant.

- Les combinaisons de conjonction, disjonction et implication de propositions floues élémentaires. Par exemple, «si $(\mathcal{X}_1$ est $A_{11})$ et $(\mathcal{X}_2$ est $A_{12})$; alors $(\mathcal{Y}$ est $B_1)$», etc.

1.4.3 Implications floues

Dans la proposition floue p : «Si $(\mathcal{X}$ est $A)$; alors $(\mathcal{Y}$ est $B)$», les propositions «$\mathcal{X}$ est A» et «$\mathcal{Y}$ est B» sont construites à partir des deux variables linguistiques $(x, T(x), X, G, M)$ et $(y, T(y), Y, G, M)$ qui sont a priori indépendantes. L'implication floue permet de définir une liaison entre la prémisse «$\mathcal{X}$ est A» et la conclusion «$\mathcal{Y}$ est B» de cette règle.

Définition 1.24 (Implications floues) *Considérons p : «$\mathcal{X}$ est A», q : «$\mathcal{Y}$ est B» deux propositions floues construites à partir de deux variables linguistiques $(x, T(x), X, G, M)$ et $(y, T(y), Y, G, M)$. Notons a, b les valeurs de vérité possibles de p et q respectivement. Une implication floue, que l'on note $I(a, b)$, est une fonction :*

$$I(a, b) : [0, 1] \times [0, 1] \rightarrow [0, 1], \tag{1.33}$$

qui définit la valeur de vérité de la proposition floue «si p alors q». L'implication floue est également notée :

$$A \Rightarrow B. \tag{1.34}$$

D'après la définition 1.22, les valeurs de vérité sont exprimées par :

$$a = \mu_A(x),$$
$$b = \mu_B(y).$$

La fonction $I(a, b)$ est donc équivalente à la fonction d'appartenance, que l'on note $\mu_R(x, y)$, d'une relation floue définie entre X et Y. Cette fonction d'appartenance s'exprime, pour tout (x, y) de $X \times Y$, en utilisant la fonction d'appartenance $\mu_A(x)$ intervenant dans la prémisse et celle $\mu_B(x)$ intervenant dans la conclusion de la règle :

$$\mu_R(x, y) = I(\mu_A(x), \mu_B(x)). \tag{1.35}$$

Ainsi, la définition d'une implication floue peut se ramener à celle d'une fonction d'appartenance. Les implications floues le plus souvent employées sont précisées dans le tableau 1.2.

Valeur de vérité	$I(\mu_A(x), \mu_B(x))$	nom
I_m	$min(\mu_A(x), \mu_B(x))$	Mamdani
I_l	$\mu_A(x) \times \mu_B(x)$	Larsen
I_r	$1 - \mu_A(x) + \mu_A(x) \times \mu_B(x)$	Reichenbach
I_w	$max(1 - \mu_A(x), min(\mu_A(x), \mu_B(x)))$	Willmott
I_{rg}	$\begin{cases} 1 & \text{si } \mu_A(x) \leq \mu_B(x) \\ 0 & \text{sinon} \end{cases}$	Rescher-Gaines
I_{kd}	$max(1 - \mu_A(x), \mu_B(x))$	Kleene-Dienes
I_{bg}	$\begin{cases} 1 & \text{si } \mu_A(x) \leq \mu_B(x) \\ \mu_B(x) & \text{sinon} \end{cases}$	Brouwer-Gödel
I_g	$\begin{cases} min(\mu_B(x)/\mu_A(x), 1) & \text{si } \mu_A(x) \neq 0 \\ 0 & \text{sinon} \end{cases}$	Goguen
I_l	$min(1 - \mu_A(x) + \mu_B(x), 1)$	Lukasiewicz

TAB. 1.2 : Implications floues les plus utilisées

Parmi les implications définies dans le tableau 1.2, les implications de Mamdani et de Larsen sont les plus connues en raison de leurs applications dans la commande floue.

1.4.4 Inférence floue

Les méthodes d'inférence, utilisées habituellement en logique standard, peuvent être généralisées dans le cadre de la logique floue pour permettre de raisonner lorsque les règles ou les faits sont connus de façon imparfaite. La méthode d'inférence la plus connue est le *modus ponens*, qui permet de déduire une nouvelle connaissance en se basant sur la connaissance d'un seul fait et d'une seule règle.

Définition 1.25 (Modus ponens généralisé) *Considérons connue une règle floue, définie par l'implication floue $A \Rightarrow B$ entre les deux propositions p : «$\mathcal{X}$ est A», q : «$\mathcal{Y}$ est B». Cette implication est caractérisée par une fonction d'appartenance $\mu_R(x, y)$ définie par l'équation 1.35. Supposons connue la proposition p' : «$\mathcal{X}$ est A'», de fonction d'appartenance $\mu_{A'}(x)$. Le modus ponens généralisé exprimé selon :*

Règle :	*Si* ($\mathcal{X}$ *est* A) ; *alors* ($\mathcal{Y}$ *est* B)
Fait :	($\mathcal{X}$ *est* A')
Conclusion :	($\mathcal{Y}$ *est* B')

permet de déduire une nouvelle proposition q' : «$\mathcal{Y}$ est B'» dont la fonction d'appartenance est définie par :

$$\forall y \in Y \qquad \mu_{B'}(y) = sup_{x \in X} \top(\mu_A(x), \mu_R(x, y)) \tag{1.36}$$

En fait, la relation précédente correspond à la définition d'un sous-ensemble flou B' à partir des sous-ensembles A, B et A', en utilisant :

$$B' = (A \Rightarrow B) \circ A'. \tag{1.37}$$

Le choix de la t-norme $\top$ utilisée dans l'équation 1.36 doit permettre de satisfaire la condition $B' = B$ lorsqu'on a initialement $A' = A$. Dans ce cas, le modus ponens généralisé est équivalent au modus ponens simple.

Définition 1.26 (Inférence avec plusieurs règles) *Considérons connues n règles floues, définies par les n implications «$A_i \Rightarrow B_i$» entre les propositions p_i : «$\mathcal{X}$ est A_i» et q_i : «$\mathcal{Y}$ est B_i». Supposons également connue la proposition p' : «$\mathcal{X}$ est A'». La forme générale d'inférence avec plusieurs règles est donnée par :*

Règle 1 :	*Si* ($\mathcal{X}$ *est* A_1)	; *alors* ($\mathcal{Y}$ *est* B_1)
Règle 2 :	*Si* ($\mathcal{X}$ *est* A_2)	; *alors* ($\mathcal{Y}$ *est* B_2)
...		
Règle n :	*Si* ($\mathcal{X}$ *est* A_n)	; *alors* ($\mathcal{Y}$ *est* B_n)
Fait :	($\mathcal{X}$ *est* A')	
Conclusion :	($\mathcal{Y}$ *est* B')	

Si on exprime la règle R_i par l'implication floue $A_i \Rightarrow B_i$, la composition de toutes les règles est donnée par :

$$R' = R_1 \bigcup R_2 \bigcup \cdots \bigcup R_n \ . \tag{1.38}$$

Dans ce cas, l'inférence à plusieurs règles consiste à définir un sous-ensemble flou B' à partir du sous-ensemble A' et de la règle floue R' selon :

$$B' = R' \circ A' \ . \tag{1.39}$$

1.5 Commande floue

De façon générale, un système de commande a pour objectif de piloter l'entrée d'un processus afin d'obtenir un fonctionnement correct de ce dernier. Lorsqu'on dispose d'un modèle plus ou moins précis du système à commander, on peut utiliser un contrôleur de structure standard, fixe ou adaptatif, dont les paramètres seront évalués à partir du modèle. Malheureusement, lorsque le système est difficilement modélisable, la conception du contrôleur peut s'avérer très complexe, sinon impossible.

Lorsqu'un opérateur humain commande manuellement un système, les actions qu'il réalise sont dictées par une connaissance *subjective* du fonctionnement de ce système. Par exemple, s'il fait «froid» dans une pièce, on «augmente» le chauffage ; s'il fait «très froid», on «chauffe plus». Cette commande du système peut être envisagée de façon différente selon la personne qui la réalise : la sensation de «froid» n'est pas directement liée à une mesure de la température.

Ce principe est à la base de la commande floue. La mesure réalisée sur le système («température») est prise en compte par l'intermédiaire d'une variable linguistique («froid», «tiède», «chaud»), qui est issue d'une analyse par un expert humain. Ensuite, l'action

à réaliser est déduite à la fois d'un ensemble de règles de commande («s'il fait froid, on chauffe plus» ...) et de l'état du système, qualifié par la variable linguistique. Enfin, la commande finale du système est créée en utilisant les conclusions de la déduction.

En résumé, un contrôleur flou comporte les différents éléments suivants :

- Un sous-système d'interface avec le flou, composé en général d'un ensemble de variables linguistiques.
- Une base de connaissances : «base de données» et «base de règles linguistiques de commande».
- Un sous-système réalisant un raisonnement en utilisant des méthodes issues de la logique floue.
- Un sous-système d'interface avec le non flou, qui fournit la ou les commandes envoyées au système.

La figure 1.8 montre la structure générale d'un contrôleur flou. Le fonctionnement précis de chacun des sous-ensembles est décrit dans la suite de ce chapitre.

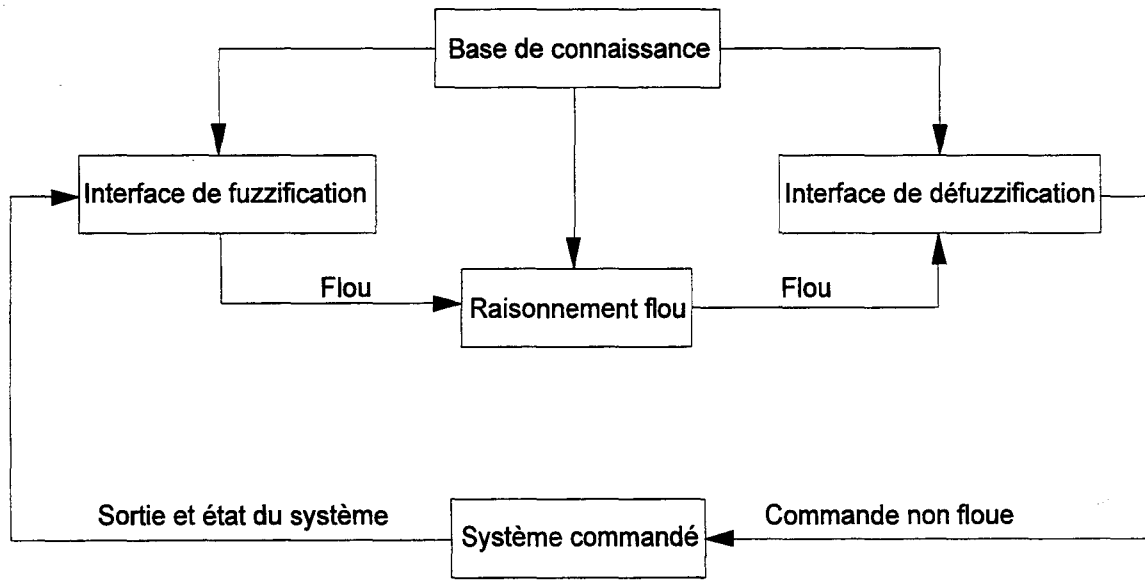


FIG. 1.8 : Configuration générale d'un contrôleur flou

1.5.1 Fuzzification

La fuzzification est réalisée dans l'interface d'entrée du contrôleur flou. Durant cette phase, les informations issues du système sont tout d'abord normalisées. Ensuite, les

données normalisées sont transformées en qualifications linguistiques, en utilisant des règles sémantiques définies par un expert.

Durant la phase de normalisation, chaque mesure issue du système est modifiée pour fournir une valeur appartenant à un univers du discours relativement simple. On peut choisir comme univers du discours un intervalle centré sur zéro : $[-c, +c]$. Si la mesure initiale x est comprise dans un autre intervalle $[a, b]$, la normalisation est souvent réalisée par transformation linéaire, selon :

$$y = \frac{2c}{b-a}[x - \frac{a+b}{2}]. \tag{1.40}$$

L'univers du discours est ensuite représenté par une variable linguistique, qui comporte un nombre assez restreint de termes (en général trois, cinq ou sept) de façon à limiter le nombre de règles.

Enfin, les valeurs normalisées déduites de chacune des entrées sont transformées en qualifications linguistiques, en utilisant les variables linguistiques correspondantes.

Exemple 1.4. *Une variable linguistique sur l'univers de discours* $[-6, +6]$, *peut être définie de la façon précisée dans le tableau 1.3 :*

Terme lingisutique	Signification	Règle sémantique
NG	négatif grand	environ -6
NM	négatif moyen	environ -4
NP	négatif petit	environ -2
EZ	environ zéro	environ 0
PP	positif petit	environ $+2$
PM	positif moyen	environ $+4$
PG	positif grand	environ $+6$

TAB. 1.3 : Exemple de variable linguistique

La figure 1.9 montre les fonctions d'appartenance associées aux différents termes linguistiques.

Dans cet exemple, les termes linguistiques utilisés sont ceux initialement proposés par Mamdani lorsque la variable linguistique en comporte sept. Les fonctions d'appartenance

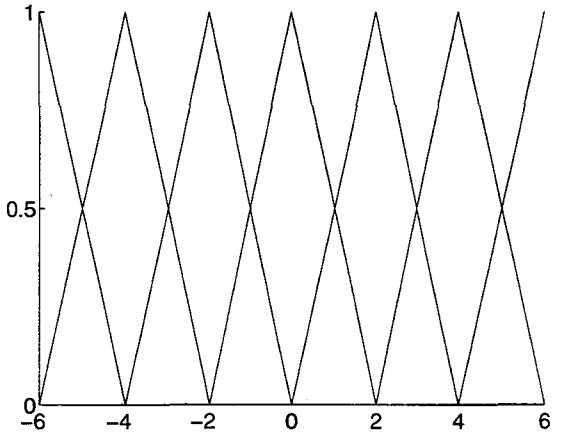
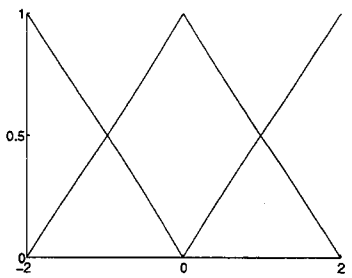
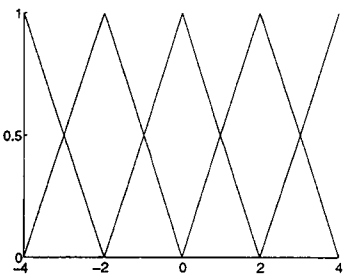


FIG. 1.9 : Univers du discours partitionné par les termes linguistique définis dans le tableau 1.3



(a) Désignations standard pour trois termes



(b) Désignations standard pour cinq termes

FIG. 1.10 : Désignations standard et fonctions d'appartenance

correspondant à des variables comportant trois et cinq termes sont représentées sur la figure 1.10.

La fuzzification est une étape clé dans tout contrôleur flou. La variable linguistique, qui est définie par une expertise, doit respecter un certain nombre de critères afin d'être efficace :

- Chaque terme linguistique est un nombre flou, de noyau non nul et dont la fonction d'appartenance est convexe.
- Les fonctions d'appartenance doivent respecter l'ordre linguistique. Dans l'exemple 1.2, on ne peut pas modifier l'ordre des termes {froid, tiède, chaud} en {chaud, tiède, froid}.
- Les fonctions d'appartenance ne doivent pas trop se superposer. On tolère en général un chevauchement qui ne dépasse pas la mi-hauteur des termes linguistiques

consécutifs.

1.5.2 Règles floues

Les règles floues permettent de déduire des connaissances concernant l'état du système en fonction des qualifications linguistiques fournies par l'étape de fuzzification. Ces connaissances sont également des qualifications linguistiques.

Habituellement, les règles floues sont déduites des expériences acquises par les opérateurs ou les experts. Ces connaissances sont traduites en règles simples pouvant être utilisées dans un processus d'inférence floue. Par exemple, si un expert exprime la règle «si la température de l'eau est chaude, il faut ajouter de l'eau froide», le système utilisera une règle du genre «si p alors q».

Aujourd'hui, il est cependant possible de constituer une base de règles floues grâce à des méthodes d'apprentissage, sans avoir nécessairement besoin d'un expert humain. Cette stratégie sera décrite plus en détail dans la suite de cette thèse.

1.5.3 Inférences floues

Dans la section 1.4.4, nous avons constaté qu'une inférence floue n'est ni plus ni moins qu'une relation floue définie entre deux sous-ensembles. La définition de la relation peut théoriquement faire intervenir n'importe quel opérateur de combinaison (cf. équation 1.39). Pourtant, on utilise souvent les inférences floues définies par Mamdani et Sugeno.

1.5.3.1 Inférence floue de Mamdani

Supposons que la base de connaissances est consituée de n règles d'inférence contenant chacune m prémisses et une conclusion. Le fait est également constitué de m propositions floues. Le processus d'inférence peut être décrit par le schéma suivant :

Règle 1 :	Si ($\mathcal{X}_1$ est A_{11}) et $\cdots$ et ($\mathcal{X}_m$ est A_{1m})	; alors ($\mathcal{Y}$ est B_1)
Règle 2 :	Si ($\mathcal{X}_1$ est A_{21}) et $\cdots$ et ($\mathcal{X}_m$ est A_{2m})	; alors ($\mathcal{Y}$ est B_2)
$\cdots$		
Règle n :	Si ($\mathcal{X}_1$ est A_{n1}) et $\cdots$ et ($\mathcal{X}_m$ est A_{nm})	; alors ($\mathcal{Y}$ est B_n)
Fait :	($\mathcal{X}_1$ est A'_1) et $\cdots$ et ($\mathcal{X}_m$ est A'_m).	
Conclusion :	($\mathcal{Y}$ est B')	

dans lequel $\mathcal{X}_1, \ldots, \mathcal{X}_m$ sont des éléments des univers du discours $X_1, \ldots, X_m$ et A_{ji}, $(j = 1, \cdots, m)$, A'_i sont des quantités floues sur l'univers du discours X_i, et B_j, $(j = 1, \cdots, m)$, B' sont également des quantités floues sur l'univers du discours Y.

Afin de définir une seule prémisse pour une règle i, les propositions «$\mathcal{X}_j$ est A_{ij}», $(j = 1, \cdots, m)$, sont combinées par l'opérateur minimum. La fonction d'appartenance de cette prémisse unique est donc donnée par :

$$\mu_{R_{A_i}}(x_1, \cdots, x_m) = \mu_{A_{i1}}(x_1) \wedge \cdots \wedge \mu_{A_{im}}(x_m) = \bigwedge_{j=1}^{m} \mu_{A_{ij}}(x_j) \tag{1.41}$$

En ce qui concerne le fait, les propositions «$\mathcal{X}_1$ est A'_1» sont aussi combinées par l'opérateur minimum, ce qui permet de déduire la fonction d'appartenance :

$$\mu_{R_{A'}}(x_1, \cdots, x_m) = \bigwedge_{j=1}^{m} \mu_{A'_j}(x_j) \tag{1.42}$$

D'après Mamdani, l'implication $A \Rightarrow B$ est également construite par un opérateur de minimum, selon :

$$(A \Rightarrow B) = (A \wedge B). \tag{1.43}$$

On peut ainsi déterminer simplement la fonction d'appartenance de la règle R_i :

$$\mu_{R_i}(x_1, \cdots, x_m, y) = \mu_{R_{A_i}}(x_1, \cdots, x_m) \wedge \mu_{B_i}(y) \tag{1.44}$$

Les différentes règles sont ensuite combinées, comme nous l'avons vu dans la définition 1.26, par un opérateur de minimum. La fonction d'appartenance de l'équivalence de toutes les règles est alors donnée par :

$$\begin{aligned} \mu_R(x_1, \cdots, x_m, y) &= \mu_{R_1}(x_1, \cdots, x_m, y) \vee \cdots \vee \mu_{R_n}(x_1, \cdots, x_m, y) \\ &= \bigvee_{i=1}^{n} \mu_{R_i}(x_1, \cdots, x_m, y) \\ &= \bigvee_{i=1}^{n} \{\mu_{R_{A_i}}(x_1, \cdots, x_m) \wedge \mu_{B_i}(y)\} \end{aligned} \tag{1.45}$$

En utilisant la règle de combinaison des relations floues (définition 1.18), on peut obtenir l'expression de la fonction d'appartenance de B', selon :

$$\begin{aligned}
\mu_{B'}(y) &= \bigvee_{x_1,\cdots,x_m} \{\mu_{R_{A'}}(x_1,\cdots,x_m) \wedge \mu_R(x_1,\cdots,x_m,y)\} \\
&= \bigvee_{x_1,\cdots,x_m} \{\mu_{R_{A'}}(x_1,\cdots,x_m) \wedge [\bigvee_{i=1}^{n} (\mu_{R_{A_i}}(x_1,\cdots,x_m) \wedge \mu_{B_i}(y))]\} \\
&= \bigvee_{x_1,\cdots,x_m} \{\bigvee_{i=1}^{n} [\mu_{R_{A'}}(x_1,\cdots,x_m) \wedge \mu_{R_{A_i}}(x_1,\cdots,x_m) \wedge \mu_{B_i}(y)]\} \\
&= \bigvee_{x_1,\cdots,x_m} \{\bigvee_{i=1}^{n} [(\mu_{R_{A'}}(x_1,\cdots,x_m) \wedge \mu_{R_{A_i}}(x_1,\cdots,x_m)) \wedge \mu_{B_i}(y)]\} \\
&= \bigvee_{i=1}^{n} \{[\bigvee_{x_1,\cdots,x_m} (\mu_{R_{A'}}(x_1,\cdots,x_m) \wedge \mu_{R_{A_i}}(x_1,\cdots,x_m))] \wedge \mu_{B_i}(y)\} \qquad (1.46)
\end{aligned}$$

où $\bigvee_{x_1,\cdots,x_m}$ est l'opérateur *sup* qui consiste à rechercher la valeur maximale de tous les $x_1,\cdots,x_m$. En notant :

$$\alpha_i(x_1,\cdots,x_m) = \bigvee_{x_1,\cdots,x_m} \{\mu_{R_{A'}}(x_1,\cdots,x_m) \wedge \mu_{R_{A_i}}(x_1,\cdots,x_m)\}, \qquad (1.47)$$

on obtient finalement l'expression suivante pour la fonction d'appartenance de B', qui définit l'inférence selon Mamdani :

$$\mu_{B'}(y) = \bigvee_{i=1}^{n} \{\alpha_i(x_1,\cdots,x_m) \wedge \mu_{B_i}(y)\} \ . \qquad (1.48)$$

En interprétant l'équation 1.47, on constate que $\alpha_i(x_1,\cdots,x_m)$ est l'intersection entre le fait A' et la prémisse A_i de la règle numéro i. Cette expression est donc donnée par la hauteur de l'intersection de A' et A_i :

$$\alpha_i(x_1,\cdots,x_m) = \mu_{R_{A'}}(x_1,\cdots,x_m) \wedge \mu_{R_{A_i}}(x_1,\cdots,x_m) = H(R_{A'} \bigcap R_{A_i}) \qquad (1.49)$$

$\alpha_i(x_1,\cdots,x_m)$ peut également être considéré comme le degré de compatibilité de A' avec A_i.

Si on se limite au cas de deux règles contenant deux prémisses ($m = n = 2$), l'inférence s'exprime par :

Règle 1 :	Si ($\mathcal{X}_1$ est A_{11}) et ($\mathcal{X}_2$ est A_{12}) ; alors ($\mathcal{Y}$ est B_1)
Règle 2 :	Si ($\mathcal{X}_1$ est A_{21}) et ($\mathcal{X}_2$ est A_{22}) ; alors ($\mathcal{Y}$ est B_2)
Fait :	($\mathcal{X}_1$ est A'_1) et ($\mathcal{X}_2$ est A'_2)
Conclusion :	($\mathcal{Y}$ est B')

La fonction d'appartenance du sous-ensemble flou B' est alors donnée par :

$$\mu_{B'}(y) = \bigvee_{i=1}^{2} \{\alpha_i(x_1, x_2) \wedge \mu_{B_i}(y)]\} \tag{1.50}$$

La figure 1.11 montre un synoptique de ce processus d'inférence. Dans cet exemple, au niveau des conditions, les opérateurs logiques standard «et» et «ou» sont remplacés respectivement par les opérateurs de minimum et de maximum. La conclusion de chaque règle, introduite par «alors» est également calculée par l'opérateur minimum. C'est pourquoi on appelle également l'inférence de Mamdani inférence max-min.

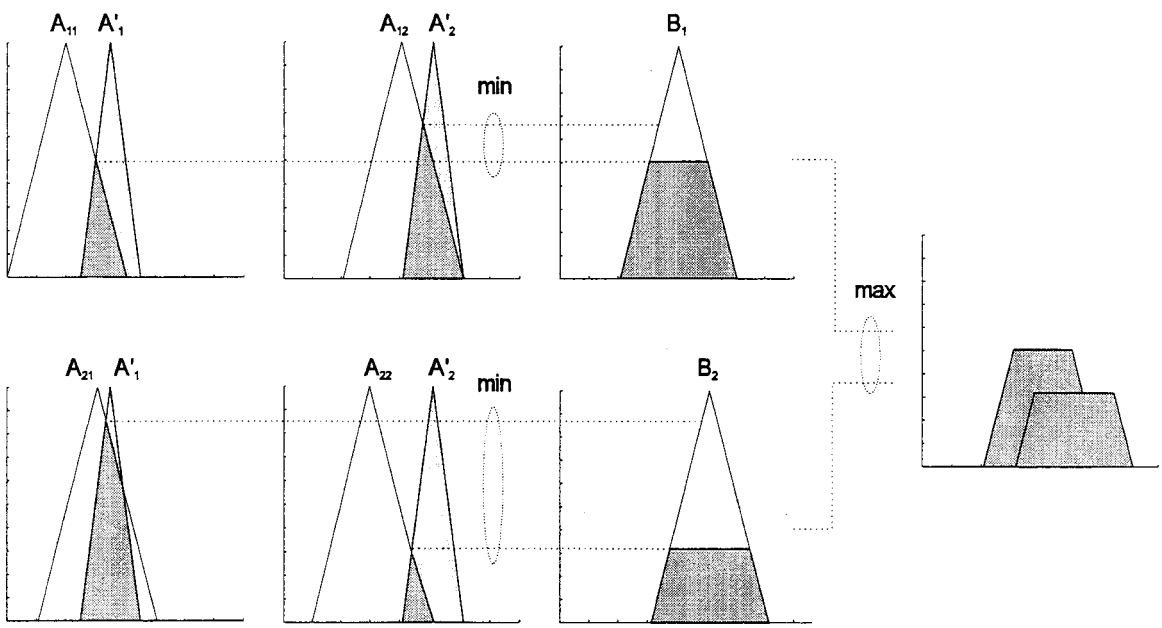


FIG. 1.11 : Exemple d'inférence max-min

L'inférence max-min est une méthode relativement simple largement utilisée dans les applications industrielles. Elle est parfois remplaçée par l'inférence *max-prod*, ou inférence de Larsen, qui permet d'améliorer la qualité du résultat. Pour cette inférence, plutôt que d'utiliser l'opérateur minimum pour définir la conclusion d'une règle, on utilise le produit. L'équation 1.44 devient alors :

$$\mu_{R_i}(x_1, \cdots, x_m, y) = \mu_{R_{A_i}}(x_1, \cdots, x_m) \cdot \mu_{B_i}(y) \tag{1.51}$$

dans laquelle "$\cdot$" est le produit arithmétique. L'équation 1.50 devient ainsi :

$$\mu_{B'}(y) = \bigvee_{i=1}^{2} \{\alpha_i(x_1, x_2) \cdot \mu_{B_i}(y)]\} \tag{1.52}$$

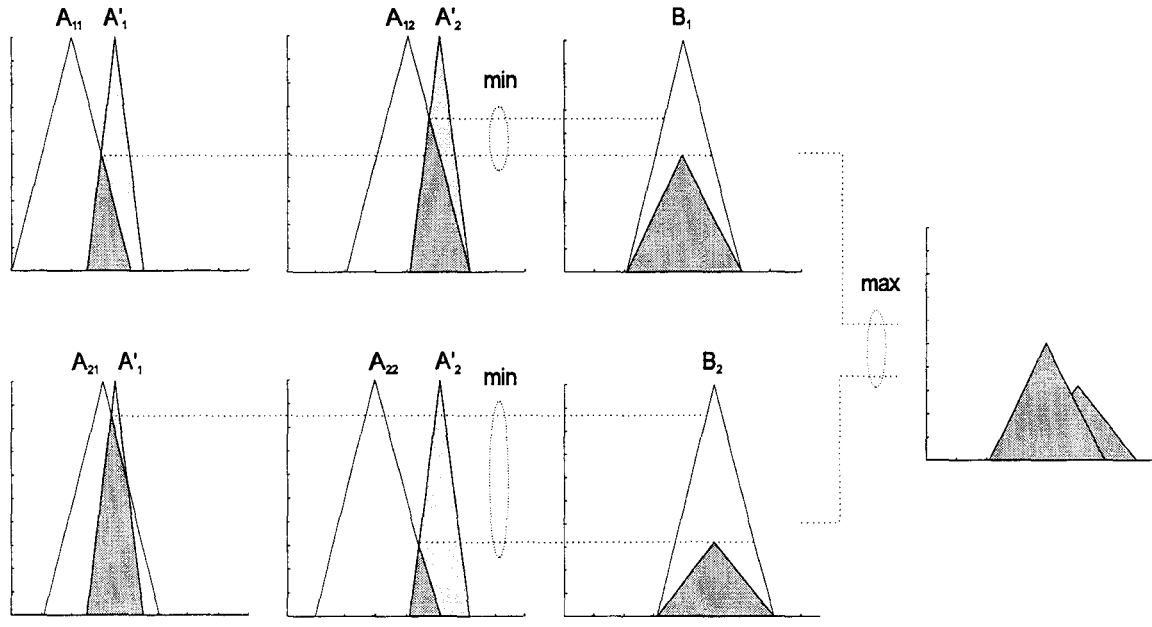


FIG. 1.12 : Exemple d'inférence max-prod

La figure 1.12 présente un synoptique du traitement de l'exemple de la figure 1.11 par l'inférence max-prod.

Exemple 1.5. *Soit un système à commander, dans lequel on analyse deux valeurs : l'erreur E et le taux d'erreur ΔE. Le contrôleur flou doit fournir la commande S. Les valeurs d'entrée du contrôleur sont transformées en termes linguistiques dont les fonctions d'appartenance sont représentées sur la figure 1.13. On utilise les règles floues suivantes :*

Règle 1 : Si (E est EZ) et (ΔE est PP) ; alors (S est EZ)

Règle 2 : Si (E est EZ) et (ΔE est EZ) ; alors (S est EZ)

Règle 3 : Si (E est NP) et (ΔE est NP) ; alors (S est PP)

Règle 4 : Si (E est NP) et (ΔE est EZ) ; alors (S est PG)

En utilisant l'inférence de Mamdani, on définit un contrôleur flou dont le mécanisme d'interprétation des règles est schématisé sur la figure 1.13.

1.5.3.2 Inférence floue de Sugeno

Sugeno a proposé une méthode d'inférence floue qui garantit la continuité de la sortie [TS83, Sug85]. Cette méthode d'inférence s'avère très efficace dans des applications faisant intervenir à la fois des techniques linéaires, d'optimisation et adaptatives. Dans l'inférence de Sugeno, les règles floues sont exprimées de la façon suivante :

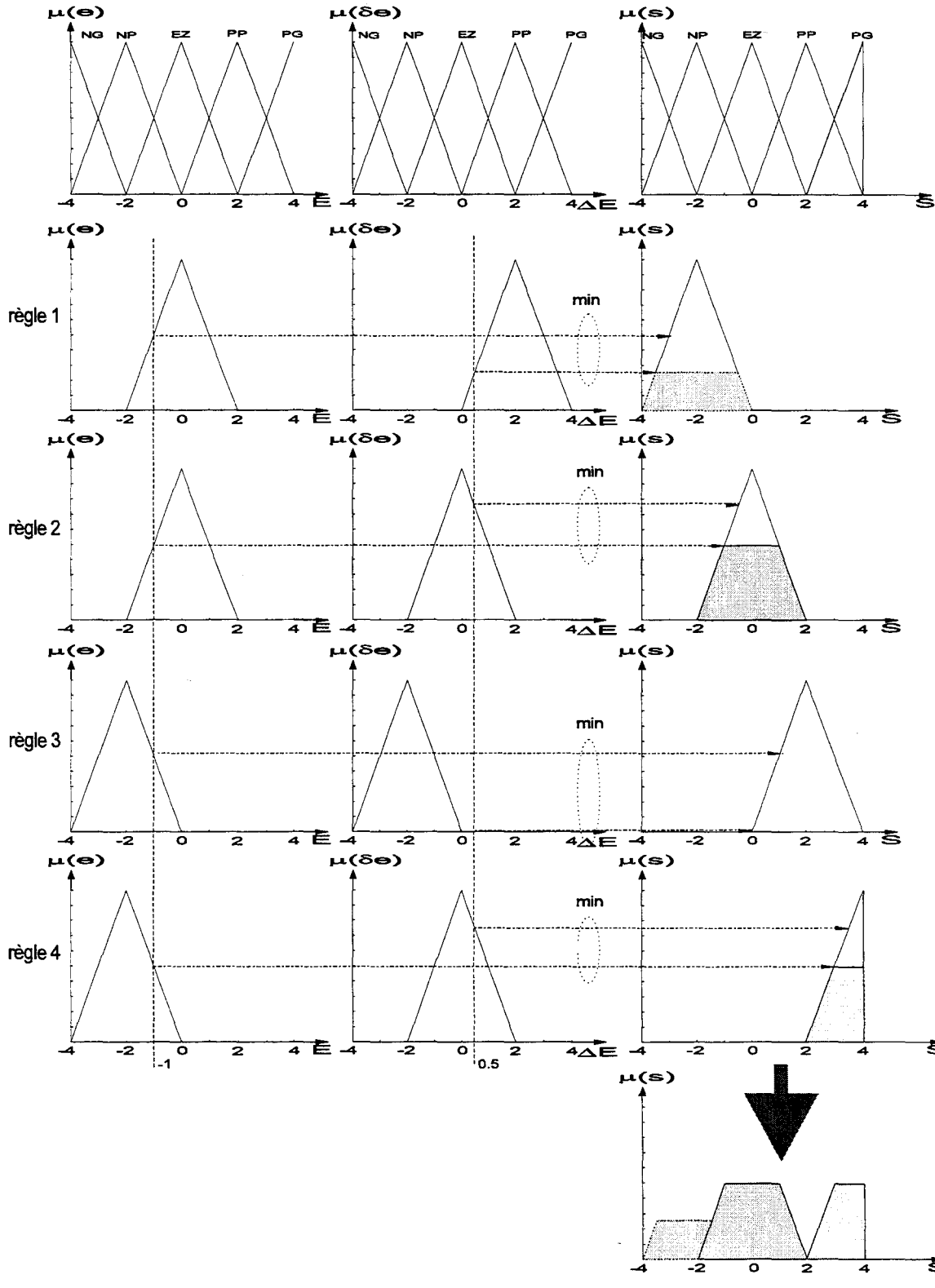


FIG. 1.13 : Exemple d'inférence de Mamdani

Règle i : Si (x_1 est A_{i1}) et $\cdots$ et (x_m est A_{im}) ; alors $y = f_i(x_1, \cdots, x_m)$

dans laquelle x_1,$\cdots$,x_m et y sont des éléments des univers du discours X_1,$\cdots$,X_m et A_{i1},$\cdots$,A_{im} sont des termes linguistiques sur ces mêmes univers du discours. y est une fonction de x_1,$\cdots$,x_m. Le problème consiste à déterminer les paramètres de la fonction, ce qui est possible en utilisant une méthode d'optimisation [TS85, SK88]. Une méthode utilisant un réseau de neurones comme système d'optimisation a été décrite dans [Jan93].

Par rapport à l'inférence de Sugeno, celle de Mamdani est plus intuitive, plus générale et elle s'adapte particulièrement bien à l'utilisation de connaissances issues d'une expertise humaine.

1.5.4 Défuzzification

Comme nous avons vu dans la section précédente, les méthodes d'inférence fournissent un résultat qui est une fonction d'appartenance. Or, la sortie du contrôleur est en général une grandeur continue, prenant sa valeur dans un intervalle. La *défuzzification* est le traitement qui permet de définir une correspondance entre le résultat de l'inférence et la grandeur continue fournie en sortie.

1.5.4.1 Défuzzification par centre de gravité

La défuzzification par centre de gravité consiste à calculer l'abscisse du centre de gravité de la fonction d'appartenance selon :

$$y_{cg} = \frac{\int_y y \cdot \mu_{B_{res}}(y) dy}{\int_y \mu_{B_{res}}(y) dy} \tag{1.53}$$

En pratique, on estime le centre de gravité en calculant la moyenne d'un certain nombre de points échantillonnés sur la fonction :

$$y_{cg} = \frac{\sum y_i \cdot \mu_{B_{res}}(y_i)}{\sum \mu_{B_{res}}(y_i)} \tag{1.54}$$

Le temps nécessaire au traitement est directement proportionnel au nombre de points retenus pour le calcul de la moyenne. Selon les contraintes fixées par l'application, il y a un compromis à réaliser entre la précision souhaitée et le temps de calcul disponible.

1.5.4.2 Défuzzification par centre maximum

Dans cette méthode, la valeur de sortie est estimée par l'abscisse du point correspondant au centre de l'intervalle pour lequel la fonction d'appartenance est maximale. Cette valeur est fournie par l'expression :

$$y_{cm} = \frac{\inf M + \sup M}{2} \tag{1.55}$$

dans laquelle M est l'ensemble des points pour lesquels la fonction d'appartenance est maximale :

$$M = \{y \in [-c, c] | \mu_{B_{res}}(y) = H(B_{res})\} \ . \tag{1.56}$$

Dans le cas discret, on explore en fait la liste de tous les points pour lesquels la fonction d'appartenance est maximale afin de trouver le plus petit et le plus grand.

1.5.4.3 Défuzzification par valeur maximum

Cette méthode ne s'utilise que dans le cas discret. On choisit comme sortie y_m l'abscisse de la valeur maximale de la fonction d'appartenance résultante $\mu_{B_{res}}(y)$. Lorsque $\mu_{B_{res}}(y)$ est écrêtée, on prend la moyenne des abscisses du maximum :

$$y_m = \frac{\sum_{y_i \in M} y_i}{|M|} \tag{1.57}$$

où M est défini dans l'équation 1.56.

1.6 Conclusion

Nous avons vu dans ce chapitre l'intérêt de la logique floue dans le domaine du contrôle de processus. Cette approche permet de tenir compte à la fois des connaissances d'un expert humain et de l'incertitude et de l'imprécision des données traitées par le contrôleur. Les variables linguistiques permettent de traiter ces deux informations initialement très différentes à l'aide d'un formalisme unique.

Pourtant, la conception d'un contrôleur flou n'est pas toujours chose aisée. Lorsqu'on utilise un contrôleur de type standard (par exemple un PID), on dispose de nombreux outils de synthèse permettant de choisir au mieux les paramètres du régulateur en fonction

de la structure ou du modèle du système à commander. Malheureusement, la panoplie d'outils disponibles est beaucoup plus limitée dans le cas des contrôleurs flous.

En fait, le fonctionnement d'un contrôleur flou dépend d'un nombre très important de paramètres (fonctions d'appartenance, règles floues, règles d'inférence, défuzzification) qu'il faut régler lors de la conception. Comme ces paramètres s'influencent mutuellement, il est peu probable qu'une méthode de synthèse traitant indépendamment chaque sous-système du contrôleur flou puisse fournir un résultat «optimal».

Dans certains cas, une approche globale de la conception est toutefois possible. Elle est basée sur une méthode *d'apprentissage*, dans laquelle un système extérieur au contrôleur analyse les performances de ce dernier lorsqu'il utilise un ensemble donné de paramètres. Par essais successifs, le système extérieur peut sélectionner le jeu de paramètres qui assurera le «meilleur» fonctionnement du contrôleur.

Dans le chapitre 2, nous décrivons le principe des *algorithmes génétiques*, qui permettent d'explorer de façon très efficace l'espace des solutions possibles d'un problème. Ensuite, dans le chapitre 3, nous verrons comment les algorithmes génétiques peuvent être utilisés pour choisir les paramètres d'un contrôleur flou.

Chapitre 2

Algorithmes génétiques

2.1 Introduction

Les Algorithmes Génétiques (en abrégé AG) sont des méthodes d'exploration de l'ensemble des solutions d'un problème utilisant les mêmes mécanismes que ceux intervenant dans la sélection naturelle. Ils sont utilisés principalement dans les domaines de l'optimisation et de l'apprentissage. Le parallélisme implicite et l'exploration globale de l'espace des solutions sont les deux principaux avantages des algorithmes génétiques.

Tout d'abord, les algorithmes génétiques peuvent traiter des problèmes pour lesquels les solutions potentielles sont situées dans un espace de grande dimension, ce qui ne permet pas d'utiliser des méthodes standard reposant sur une exploration systématique. Ensuite, la recherche d'une solution optimale à un problème peut être réalisée par un algorithme génétique sans nécessiter de connaissance a priori sur la *répartition* des solutions dans l'espace.

Bien que les mécanismes exploités dans les algorithmes génétiques aient été découverts par Darwin et Mendel, le livre de John Holland [Hol75] publié en 1975 est considéré comme étant à l'origine des algorithmes génétiques. Dans ce livre, il a démontré le théorème des schémas qui constitue la justification théorique de cette approche. La même année, De Jong [Jon75] a soutenu une thèse qui sert encore de référence dans ce domaine, dans laquelle il a proposé une série de fonctions de test et réalisé de nombreuses expérimentations. En 1989, Goldberg [Gol89] a publié un livre sur les AG qui est devenu une référence. D'autres ouvrages de référence ont été publiés depuis, comme [Dav91, Mic92].

De nos jours, les AG ont trouvé des applications dans des domaines très variés, allant de la biologie jusqu'aux applications liées plus directement à l'informatique (traitement d'images, optimisation, placement de formes) [TM92, Tam92, AT94, BZP94, TH95, LS95, DUC96, MC96, Sch97, KK97].

2.1.1 Evolution naturelle et algorithmes génétiques

Depuis que Charles Darwin a formulé l'idée simple d'une évolution naturelle des espèces basée sur un mécanisme de sélection et de reproduction, la théorie de l'évolution biologique est devenue un centre d'intérêt pour les scientifiques. L'étude des fossiles montre que la structure complexe de la vie peut évoluer sur des périodes de temps relativement courtes.

A l'heure actuelle, on sait que l'évolution d'une espèce à lieu lorsque les chromosomes, qui contiennent le code génétique de sa structure biologique, se trouvent modifiés. Bien que toutes les caractéristiques des évolutions naturelles ne soient pas connues, les spécialistes considèrent qu'il faut admettre au moins les règles suivantes :

- L'évolution n'agit pas directement sur les êtres vivants ; elle opère en réalité sur les chromosomes contenus dans leurs cellules.
- L'évolution est régie par deux composantes : la sélection et la reproduction. La sélection naturelle construit un lien entre les chromosomes et les individus. Elle permet d'aboutir à un meilleur taux de reproduction des individus qui ont de «bons» chromosomes.
- La reproduction est le moteur de l'évolution. Les modifications apparaissant lors de la reproduction peuvent produire de nouvelles générations dont les caractéristiques sont différentes.
- L'ensemble des chromosomes et le codage de ces chromosomes comprennent toutes les informations caractéristiques d'un individu. Ces individus peuvent cependant s'adapter à d'autres conditions de vie.

La plupart des espèces biologiques se reproduisent par sélection naturelle et reproduction sexuée. La sélection décide quel individu peut vivre et se reproduire alors que la reproduction sexuée assure le «mélange» du matériel génétique. Les espèces qui uti-

lisent un mode de reproduction non sexué évoluent en général beaucoup moins vite, car la modification du matériel génétique n'intervient que lors de mutations.

Ces principales caractéristiques de l'évolution naturelle ont attiré John Holland et ses collègues de l'université du Michigan lors de recherches sur l'apprentissage des machines dans les années 60. Holland a trouvé que l'apprentissage se réalise non seulement par l'adaptation d'un *individu* mais également par l'évolution d'un *groupe* lors de générations successives. En 1975, il a publié son ouvrage «Adaptation in Natural and Artificial Systems» dans lequel il a proposé les premières versions d'un *algorithme génétique*, qui permet un apprentissage basé sur les règles de la sélection naturelle.

2.1.2 Terminologie

Avant de décrire le principe des algorithmes génétiques, il est nécessaire de présenter le vocabulaire que nous allons utiliser tout au long de ce travail.

chromosome $\longrightarrow$ chaîne, chromosome : Dans les systèmes naturels, les chromosomes sont les porteurs de l'information génétique nécessaire à la construction et au fonctionnement d'un organisme. Dans les algorithmes génétiques, les *chaînes*, ou *chromosomes* sont analogues aux chromosomes des systèmes biologiques. Ils sont les éléments à partir desquels sont élaborées les solutions.

génotype $\longrightarrow$ structure : Dans les systèmes naturels, l'ensemble du matériel génétique est appelé le *génotype*. Dans les algorithmes génétiques, l'ensemble des chaînes est appelé *structure*.

phénotype $\longrightarrow$ ensemble de paramètres, solution, point : Dans les systèmes naturels, l'organisme formé par l'intéraction de l'ensemble du matériel génétique avec son environnement est appelé le *phénotype*. Dans les algorithmes génétiques, les structures décodées forment un *ensemble de paramètres* donné, ou une *solution* ou un *point* dans l'espace des solutions.

gène $\longrightarrow$ trait, détecteur : Dans les systèmes naturels, les chromosomes sont constitués par les *gènes*. Dans les algorithmes génétiques, on dit que les chaînes se composent de *traits* ou *détecteurs*.

allèle $\longrightarrow$ valeur de caractéristique : Dans les systèmes naturels, l'*allèle* est une composante du gène. Les allèles sont les différentes valeurs que peuvent prendre les gènes.

Dans les algorithmes génétiques, l'allèle est également appelé *valeur caractéristique.*

locus $\longrightarrow$ position dans la chaîne : Le *locus* est la position du gène dans le chromosome. Ce terme est appelé également *position dans la chaîne* dans les algorithmes génétiques.

individu, organisme $\longrightarrow$ individu, chromosome : Un *organisme* ou un *individu* biologique est une forme qui est le produit de l'activité des gènes. Dans le cadre d'un AG traditionnel, l'individu est réduit à un chromosome, on l'appelle indifféremment *individu* ou *chromosome.*

population $\longrightarrow$ population, génération : Dans les systèmes naturels, la *population* est un groupe d'individus. Dans les algorithmes génétiques, la *population* est l'ensemble des individus ou des chromosomes. Les populations sont également appelées des *générations.*

2.2 Un exemple

Dans cette section nous décrivons les caractéristiques fondamentales d'un algorithme génétique au travers d'un exemple. Supposons que nous recherchions le maximum de la fonction :

$$f(x) = \frac{1}{(x-0.3)^2+0.1} + \frac{1}{(x-1.5)^2+0.2} \qquad x \in [0,2], \tag{2.1}$$

par un algorithme génétique standard. La figure 2.1 montre la représentation graphique de cette fonction.

2.2.1 Représentation binaire

Holland propose de représenter les chromosomes par des chaînes de bits, appelées *chaînes binaires.* Le parallélisme structurel des représentations binaires rend plus facile l'analyse des caractéristiques des algorithmes génétiques standard, comme le parallélisme implicite, le théorème des schémas et même l'influence des opérateurs de croisement et de mutation.

Durant cette étape de codage, on transforme une valeur de la variable x prise dans l'intervalle $[0,2]$ en une chaîne binaire ou chromosome. Cette transformation est guidée par un critère très important : la précision de la solution à laquelle on souhaite aboutir.

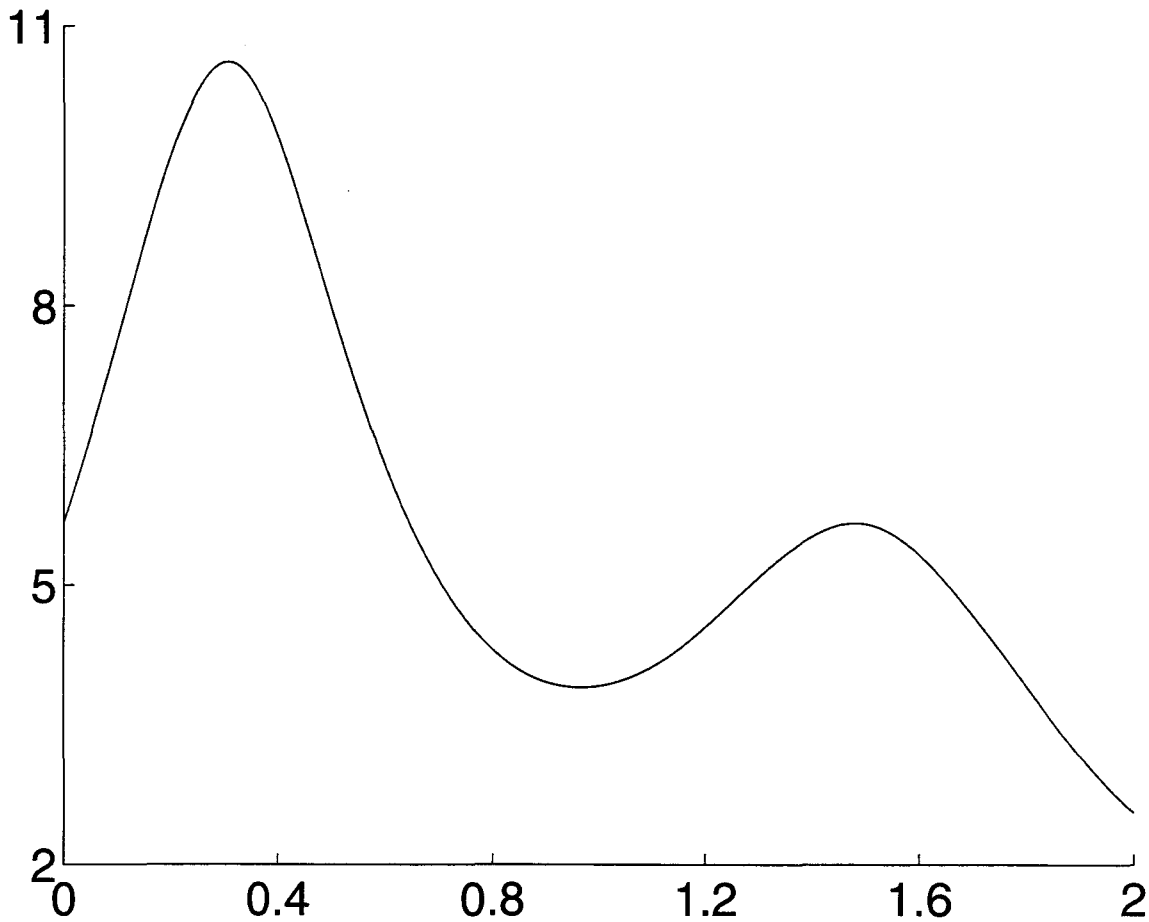


FIG. 2.1 : Fonction dont on cherche le maximum

Dans notre exemple, si la précision souhaitée est de 0.01, l'intervalle $[0, 2]$ doit être divisé en au moins 200 parties égales. Puisqu'on utilise un codage binaire des solutions, on utilise au minimum 8 bits pour distinguer les 200 valeurs :

$$128 = 2^7 < 200 < 2^8 = 256.$$

Un nombre x sera représenté par un individu décrit par la chaîne binaire $(b_7 b_6 \cdots b_0)$, issue d'une transformation simple appliquée à x :

$$\hat{x} = \sum_{i=0}^{7} b_i \cdot 2^i \cdot \left(\frac{2}{2^8 - 1}\right) \tag{2.2}$$

dans laquelle $\hat{x}$ est le nombre le plus proche de x pouvant être décomposé de cette façon. Par exemple, (01100000) représente le nombre décimal 0.7529, puisque :

$$\begin{aligned} (01100000)_2 &= 2^5 + 2^6 = 96, \text{ et} \\ 0.7529 &= 96 \cdot \frac{2}{2^8 - 1} \end{aligned}$$

2.2.2 Population initiale

Le processus d'initialisation est assez simple. On définit une population initiale d'individus qui correspond à la première génération. Le nombre d'individus, ou *taille de la population* N_p est un des paramètres de l'algorithme génétique. La chaîne binaire représentant chaque individu est constituée de bits initialisés de façon aléatoire. Dans notre exemple, on choisit une taille de population de 10 chromosomes ($N_p = 10$), initialisée au hasard à :

c_1^0=(01100000) c_2^0=(01011001) c_3^0=(10111100) c_4^0=(10010111) c_5^0=(10100001)
c_6^0=(00101111) c_7^0=(11110100) c_8^0=(11010100) c_9^0=(01010011) c_{10}^0=(01110111)

c_i^k désignant le $i^{\text{ème}}$ individu de la $k^{\text{ème}}$ génération ($k = 0$ pour la population initiale).

2.2.3 Fonction d'évaluation

La fonction d'évaluation joue le rôle de l'environnement où évolue la population. Elle permet d'avantager ou de désavantager un individu vis-à-vis des règles de sélection et de reproduction. Cette fonction, dont la seule variable est la chaîne binaire caractérisant un individu, prend une valeur élevée lorsque ce dernier est performant. Dans notre exemple, la fonction d'évaluation $f_e(c)$, pour une chaîne binaire c est identique à la fonction $f(x)$:

$$f_e(c) = f(\hat{x}) \ , \tag{2.3}$$

où c est le chromosome qui est représenté par la chaîne binaire et $\hat{x}$ est la valeur qu'on obtient par l'équation 2.2. En appliquant la fonction d'évaluation sur la population initiale obtenue précédemment, nous obtenons les valeurs suivantes :

$$\begin{aligned}
f_e(c_1^0 = (01100000)) &= f(\hat{x}_1 = 0,7529) = 4,5961 \\
f_e(c_2^0 = (01011001)) &= f(\hat{x}_2 = 0,6980) = 5,0555 \\
f_e(c_3^0 = (10111100)) &= f(\hat{x}_3 = 1,4745) = 5,6597 \\
f_e(c_4^0 = (10010111)) &= f(\hat{x}_4 = 1,1843) = 4,4709 \\
f_e(c_5^0 = (10100001)) &= f(\hat{x}_5 = 1,2627) = 4,8757 \\
f_e(c_6^0 = (00101111)) &= f(\hat{x}_6 = 0,3686) = 10,2259 \\
f_e(c_7^0 = (11110100)) &= f(\hat{x}_7 = 1,9137) = 3,0640
\end{aligned}$$

$$f_e(c_8^0 = (11010100)) = f(\hat{x}_8 = 1,6627) = 4,9263$$
$$f_e(c_9^0 = (01010011)) = f(\hat{x}_9 = 0,6510) = 5,5665$$
$$f_e(c_{10}^0 = (01110111)) = f(\hat{x}_{10} = 0,9333) = 3,9145$$

Dans la population initiale, la fonction d'évaluation est maximale pour l'individu numéro 6. On dit que l'individu caractérisé par le chromosome c_6^0 est le meilleur.

2.2.4 Sélection

Le rôle de la sélection est de choisir parmi tous les individus d'une population les parents qui assureront la reproduction. Ce choix est réalisé par tirage au sort parmi les individus, en tenant compte d'une probabilité de sélection affectée à chacun d'eux. Un individu a d'autant plus de chances d'être sélectionné que sa fonction d'évaluation prend une valeur importante. Pratiquement, la probabilité p_i d'évolution d'un individu c_i est définie par :

$$p_i = \frac{f_e(c_i)}{\sum_{j=1}^{N_p} f_e(c_j)} \tag{2.4}$$

La méthode souvent utilisée pour sélectionner les individus assurant la reproduction est la *roulette de casino*. Chaque individu occupe un secteur de la roulette dont l'angle est proportionnel à sa probabilité de sélection, définie par l'équation 2.4. On actionne N_p fois la roulette, afin de définir les parents qui assureront la reproduction.

La figure 2.2 représente la roulette de sélection obtenue en calculant les probabilités de sélection pour la population initiale définie pour notre exemple. On constate aisément que les individus les plus performants ont plus de chances d'être sélectionnés puisqu'ils occupent une surface plus importante de la roulette. Lors du tirage au sort, certains individus peuvent être retenus plusieurs fois, alors que d'autres sont tenus à l'écart. En cela, cette méthode respecte bien les règles de la sélection naturelle.

Dans l'exemple présenté, la sélection d'individus parmi la population initiale fournit les parents $c_i'^0$ caractérisés par les chromosomes suivants :

$$c_1'^0 = (c_2^0) = (01011001) \qquad c_2'^0 = (c_2^0) = (01011001)$$
$$c_3'^0 = (c_3^0) = (10111100) \qquad c_4'^0 = (c_3^0) = (10111100)$$
$$c_5'^0 = (c_5^0) = (10100001) \qquad c_6'^0 = (c_5^0) = (10100001)$$

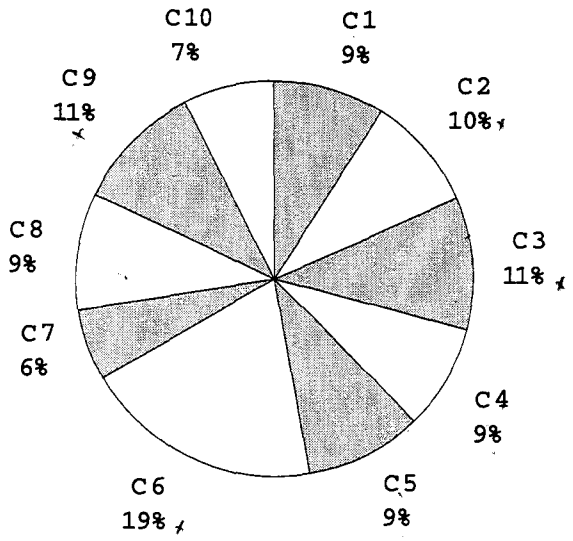


FIG. 2.2 : Sélection par la méthode de la roulette de casino

$$c'^0_7 = (c^0_5) = (10100001) \qquad c'^0_8 = (c^0_6) = (00101111)$$
$$c'^0_9 = (c^0_9) = (01010011) \qquad c'^0_{10} = (c^0_9) = (01010011)$$

On constate dans cet exemple que les individus de la population initiale pour lesquels la fonction d'évaluation est la plus élevée (c^0_3, c^0_5, c^0_6 et c^0_9) ont été sélectionnés au moins une fois comme parents. Par contre, les individus c^0_7 et c^0_{10} ont été éliminés.

2.2.5 Reproduction et croisement

L'étape de sélection a permis de choisir parmi une population les individus les plus aptes à se reproduire. Ensuite, ces individus sont regroupés par paires qui constitueront les parents de la génération suivante. Durant l'étape de reproduction, les chaînes binaires qui caractérisent deux parents sont utilisées pour créer les chaînes binaires qui caractérisent deux enfants.

Deux modes de reproduction sont envisageables : soit les enfants sont génétiquement identiques à leurs parents (conservation du matériel génétique), soit les enfants sont créés par un croisement des chromosomes des parents (modification du matériel génétique). L'opération de croisement est essentielle, car elle permet d'obtenir de nouveaux individus, distincts de ceux déjà existants, et donc d'explorer tout un espace de recherche. Dans un algorithme génétique standard, on définit ainsi un deuxième paramètre de fonctionnement : la *probabilité de croisement* p_c, qui permet de choisir entre une conservation ou

une modification des chromosomes.

Dans un algorithme génétique standard, le croisement de deux chromosomes est réalisé de la façon suivante : on choisit aléatoirement un *point de croisement* situé à la position p comprise entre 1 et $l_c - 1$, où l_c est la longueur de la chaîne binaire. Afin de créer les deux nouveaux chromosomes, on conserve intactes les sous-chaînes contenant les bits compris entre 1 et p, et on échange les sous-chaînes contenant les bits compris entre $p+1$ et l_c. Ce procédé est expliqué sur la figure 2.3.

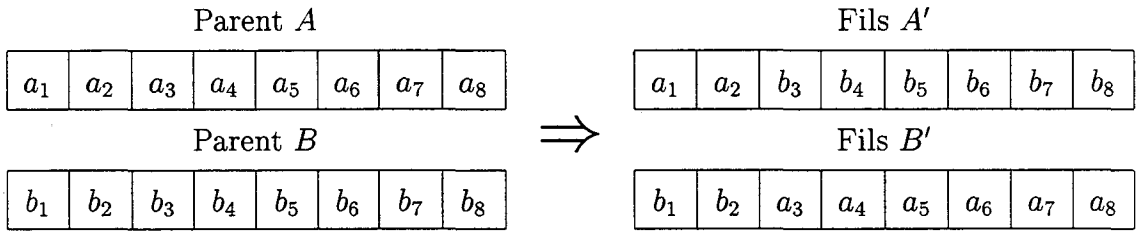


FIG. 2.3 : Croisement en position $p = 2$

Dans notre exemple, la phase de reproduction est menée de la façon suivante sur la population issue de la phase de sélection :

1. En fixant la probabilité de croisement à $0,6$, un premier tirage au sort désigne les individus c'^0_1, c'^0_3, c'^0_4 et c'^0_5 comme reproducteurs. Les autres individus seront conservés sans modification.
2. Un deuxième tirage au sort détermine les couples sur lesquels va intervenir le croisement : (c'^0_1 et c'^0_4) ainsi que (c'^0_3 et c'^0_5) sont regroupés.
3. Pour chaque paire de parents, un tirage au sort détermine le point de croisement. Par exemple, pour la paire (c'^0_1 et c'^0_4), le croisement s'effectue en position 6. La figure 2.4 montre le résultat du croisement pour cette paire de chromosomes.

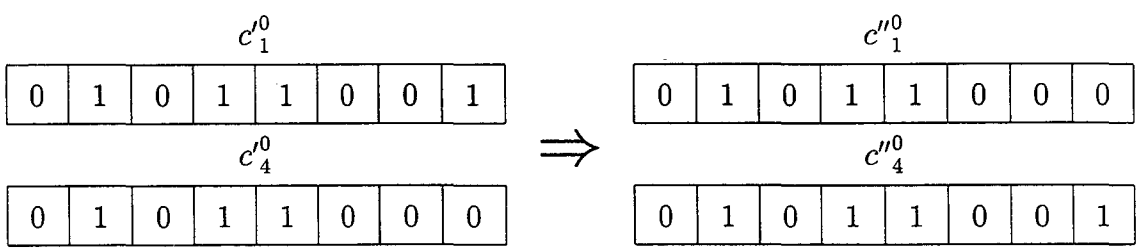


FIG. 2.4 : Croisement des parents c'^0_1 et c'^0_4

Après l'opération de reproduction, la population totale, constituée des individus c''^0_i issus d'un éventuel croisement, devient dans notre exemple :

$$c''^0_1 = (01011000) \qquad c''^0_2 = c'^0_2 = (01011001)$$
$$c''^0_3 = (10100001) \qquad c''^0_4 = (10111101)$$
$$c''^0_5 = (10111100) \qquad c''^0_6 = c'^0_6 = (10100001)$$
$$c''^0_7 = c'^0_7 = (10100001) \qquad c''^0_8 = c'^0_8 = (00101111)$$
$$c''^0_9 = c'^0_9 = (01010011) \qquad c''^0_{10} = c'^0_{10} = (01010011)$$

2.2.6 Mutation

En biologie, une mutation est une modification spontanée d'un chromosome qui n'est pas issue d'une opération normale de reproduction ou de croisement. Il est assez difficile de cerner l'effet d'une mutation dans le cas des organismes vivants. Cependant, pour certains biologistes, c'est un point clé de l'évolution, qui évite à une population de stagner lorsque tous ses individus sont devenus génétiquement identiques.

Dans les algorithmes génétiques, le processus de sélection peut parfois faire disparaître complètement certains gènes. Par exemple, suite à une phase de sélection, il peut arriver que tous les chromosomes contiennent la même sous-chaine binaire dans une position donnée. On constate facilement que cette évolution est irréversible, puisque cette sous-chaîne ne peut plus être modifiée par les opérations de sélection et de croisement.

C'est à ce niveau qu'intervient l'opération de mutation. Elle permet de modifier de façon tout à fait aléatoire le chromosome d'un individu. Toutefois, pour ne pas trop perturber l'évolution globale de la population, la probabilité d'apparition d'une mutation doit rester très faible.

Dans les algorithmes génétiques standard, une mutation correspond à une inversion de la valeur d'un bit de la chaîne binaire. Dans notre exemple, si on utilise une probabilité de mutation p_m égale à $0,01$, on doit s'attendre à obtenir une inversion d'environ $0,8$ bits à chaque génération, puisque la population comporte 10 individus caractérisés par 8 bits. Pour chaque bit, on tire au sort un nombre compris entre 0 et 1, et on inverse la valeur du bit si ce nombre est inférieur à p_m.

Après la phase de mutation, seul le sixième bit du chromosome c''_1 s'est trouvé inversé,

ce qui nous donne la population suivante pour la première génération :

$$c_1^1 = (01011100) \qquad c_2^1 = (01011001)$$
$$c_3^1 = (10100001) \qquad c_4^1 = (10111101)$$
$$c_5^1 = (10111100) \qquad c_6^1 = (10100001)$$
$$c_7^1 = (10100001) \qquad c_8^1 = (00101111)$$
$$c_9^1 = (01010011) \qquad c_{10}^1 = (01010011)$$

2.2.7 Evolution de la population

Les phases de sélection, reproduction et de mutation ont permis de créer une nouvelle population en partant de la population initiale. La fonction d'évaluation prend les valeurs suivantes pour la nouvelle population :

$$f_e(c_1^1) = f(0,7216) = 4,8415 \qquad f_e(c_2^1) = f(0,6980) = 5,0555$$
$$f_e(c_3^1) = f(1,2627) = 4,8757 \qquad f_e(c_4^1) = f(1,4824) = 5,6598$$
$$f_e(c_5^1) = f(1,4745) = 5,6597 \qquad f_e(c_6^1) = f(1,2627) = 4,8757$$
$$f_e(c_7^1) = f(1,2627) = 4,8757 \qquad f_e(c_8^1) = f(0,3686) = 10,2259$$
$$f_e(c_9^1) = f(0,6510) = 5,5665 \qquad f_e(c_{10}^1) = f(0,6510) = 5,5665$$

On constate que la nouvelle population est globalement plus performante que la population initiale, puisque la somme des valeurs de la fonction d'évaluation est passée de $52,3551$ à $57,2024$.

Le processus d'évolution se poursuit en réitérant les opérations de sélection, reproduction et mutation, de façon à créer une succession de populations. Dans notre exemple, on obtient la population suivante au bout de 50 générations :

$$c_1^{50} = (00100111) \qquad c_2^{50} = (00100111)$$
$$c_3^{50} = (10100111) \qquad c_4^{50} = (00100111)$$
$$c_5^{50} = (00100111) \qquad c_6^{50} = (00100111)$$
$$c_7^{50} = (00100111) \qquad c_8^{50} = (00100111)$$
$$c_9^{50} = (00100111) \qquad c_{10}^{50} = (00100111)$$

Les individus sont caractérisés par les valeurs suivantes de la fonction d'évaluation :

$$f_e(c_1^{50}) = f(0,3059) = 10,6116 \qquad f_e(c_2^{50}) = f(0,3059) = 10,6116$$
$$f_e(c_3^{50}) = f(1,3098) = 5,1272 \qquad f_e(c_4^{50}) = f(0,3059) = 10,6116$$
$$f_e(c_5^{50}) = f(0,3059) = 10,6116 \qquad f_e(c_6^{50}) = f(0,3059) = 10,6116$$
$$f_e(c_7^{50}) = f(0,3059) = 10,6116 \qquad f_e(c_8^{50}) = f(0,3059) = 10,6116$$
$$f_e(c_9^{50}) = f(0,3059) = 10,6116 \qquad f_e(c_{10}^{50}) = f(0,3059) = 10,6116$$

dont la somme vaut $100,6315$.

On constate que sur 10 individus initialement différents, 9 sont devenus identiques. Ils correspondent à une valeur de x égale à $0,3059$, qui est l'estimation la plus précise de l'abscisse du maximum de la fonction que l'on peut obtenir en utilisant en codage des chromosomes sur 8 bits.

2.3 La base des algorithmes génétiques

Comme nous l'avons vu dans la section précédente sur la base d'un exemple, le fonctionnement d'un algorithme génétique est basé sur des traitements extrêmement simples : sélection, croisement et mutation. Il est étonnant de constater qu'une méthode d'optimisation exploitant des principes aussi simples soit efficace. Dès 1975, Holland a cherché à expliquer le pourquoi de cette efficacité. Il a démontré le théorème des schémas, qui explique comment l'évolution d'une population d'individus affectés par les opérateurs génétiques a tendance à faire apparaître des individus plus performants.

2.3.1 Le théorème des schémas

Définition 2.1 (Schéma) *Un schéma, noté H, est un gabarit qui permet de mettre en évidence les ressemblances entre chromosomes.*

Le codage binaire utilisé dans les algorithmes génétiques standard consiste à représenter l'information par des chaînes dont les symboles appartiennent à l'alphabet $\{0,1\}$. Dans ce cas, un schéma H, qui permet de décrire simplement les évolutions possibles d'un individu, est une chaîne constituée de symboles appartenant à l'alphabet $\{0,1,*\}$, dans lequel $*$ est un méta-symbole représentant indifféremment 0 ou 1. De façon plus générale,

si l'alphabet utilisé dans le codage comporte n symboles, celui utilisé pour représenter les schémas en comporte $n+1$.

Par exemple, le schéma $H = (*11*)$ décrit les quatre chromosomes :

$$\{(0110), (0111), (1110), (1111)\} \ , \tag{2.5}$$

alors que le schéma $H = (00110011)$ ne représente que le chromosome $\{(00110011)\}$. On démontre facilement qu'un schéma contenant r symboles $*$ permet de représenter 2^r chromosomes différents.

D'autre part, un chromosome constitué de l symboles binaires est décrit par 2^l schémas. Par exemple, le chromosome (0110) est décrit par les 16 schémas suivants :

$$\begin{array}{cccc} (0110) & (*110) & (0*10) & (01*0) \\ (011*) & (**10) & (*1*0) & (*11*) \\ (0**0) & (0*1*) & (01**) & (***0) \\ (0***) & (*1**) & (**1*) & (****) \end{array}$$

Définition 2.2 (Ordre d'un schéma) *L'ordre d'un schéma H, noté $o(H)$, est le nombre de positions dont la valeur est déterminée. Lorsqu'on utilise un codage binaire, l'ordre d'un schéma est tout simplement le nombre de* 1 *et de* 0 *contenus dans la chaîne de description.*

Par exemple, l'ordre $o(H)$ du schéma $H = (*11*)$ est égal à deux, celui du schéma $H = (*0*10*1*)$ est égal à 4.

Définition 2.3 (Longueur d'un schéma) *La longueur d'un schéma H, notée $\delta(H)$, est la distance entre la première et la dernière position dont les valeurs sont déterminées.*

Par exemple, la longueur $\delta(H)$ du schéma $H = (*11*)$ vaut 1, celle du schéma $H = (*0*10*1*)$ vaut 5 et celle du schéma $H = (******1*)$ est nulle.

Pour comprendre pourquoi un algorithme génétique tend à faire évoluer la population d'individus en augmentant globalement ses performances, Holland étudie l'évolution des schémas représentant les différents individus durant une itération de l'algorithme. Pour ce faire, il suppose connu le nombre $n(H,t)$ de chromosomes représentés par le schéma H dans la population de la génération numéro t. Il évalue ensuite le nombre $n(H,t+1)$ d'individus caractérisés par ce même schéma dans la population de la génération suivante.

Notons $c_i(H,t)$ les $n(H,t)$ chromosomes de la génération t pouvant être représentés par le schéma H. Durant la phase de sélection, un chromosome c appartenant à la population

de N_p individus, peut être sélectionné selon une probabilité $p_s(c)$ définie par l'expression :

$$p_s(c) = \frac{f_e(c)}{\sum_{j=1}^{N_p} f_e(c_j)} \quad , \tag{2.6}$$

où f_e désigne la fonction d'évaluation permettant de noter un individu.

En additionnant les probabilités de sélection de tous les chromosomes $c_i(H,t)$ représentant le schéma H, on obtient la probabilité $p_s(H,t)$ de représentation du schéma H dans la nouvelle population issue de l'opération de sélection. Cette probabilité est donnée par l'expression :

$$p_s(H,t) = \frac{\sum_{i=1}^{n(H,t)} f_e(c_i(H,t))}{\sum_{j=1}^{N_p} f_e(c_j)} \quad . \tag{2.7}$$

En supposant dans un premier temps que l'évolution de la population ne dépend que de la phase de sélection, on peut estimer le nombre $n(H,t+1)$ d'individus représentant le schéma H dans la population de la génération $t+1$. Cette estimation est donnée par l'expression suivante :

$$\begin{aligned} n(H,t+1) &= N_p \cdot p_s(c_1(H,t)) + \cdots + N_p \cdot p_s(c_{n(H,t)}(H,t) \\ &= N_p \cdot \frac{f_e(c_1(H,t))}{\sum_{j=1}^{N_p} f_e(c_j)} + \cdots + N_p \cdot \frac{f_e(c_{n(H,t)}(H,t))}{\sum_{j=1}^{N_p} f_e(c_j)} \quad , \end{aligned} \tag{2.8}$$

dans laquelle chaque terme est une estimation du nombre d'individus de la nouvelle génération sélectionnés parmi les représentants du schéma H.

Notons $f_e(H)$ la valeur moyenne de la fonction d'évaluation sur l'ensemble des individus représentant le schéma H, donnée par :

$$f_e(H) = \frac{\sum_{i=1}^{n(H,t)} f_e(c_i(H,t))}{n(H,t)} \quad , \tag{2.9}$$

et $\bar{f}_e$ la valeur moyenne de cette même fonction sur l'ensemble de la population :

$$\bar{f}_e = \frac{\sum_{j=1}^{N_p} f_e(c_j)}{N_p} \quad . \tag{2.10}$$

En utilisant ces notations pour simplifier l'équation 2.8, on constate que le nombre d'individus caractérisant le schéma H dans la nouvelle population est donné par :

$$\begin{aligned} n(H,t+1) &= N_p \cdot \frac{\sum_{i=1}^{n(H,t)} f_e(c_i(H,t))}{\sum_{j=1}^{N_p} f_e(c_j)} \\ &= n(H,t) \cdot \left(\frac{\sum_{i=1}^{n(H,t)} f_e(c_i(H,t))}{n(H,t)} \right) \cdot \left(\frac{N_p}{\sum_{j=1}^{N_p} f_e(c_j)} \right) \\ &= n(H,t) \cdot \frac{f_e(H)}{\bar{f}_e} \quad . \end{aligned} \tag{2.11}$$

Cette équation permet d'évaluer l'évolution des schémas durant les générations successives. Le nombre d'individus représentant un schéma donné dans la nouvelle population est augmenté dans le rapport $f_e(H)/\bar{f}_e$. Si on suppose que ce rapport n'évolue pas trop durant les générations successives, le nombre d'individus caractérisant un schéma H suit une progression géométrique de raison $f_e(H)/\bar{f}_e$. La raison de cette progression est le rapport entre l'adaptation moyenne du schéma et l'adaptation moyenne de la population. En termes simples, les schémas ayant une valeur d'adaptation supérieure à la moyenne de la population voient le nombre d'individus qui les caractérisent augmenter durant l'évolution de la population.

Reste à évaluer l'influence des opérations de croisement et de mutation sur l'évolution des schémas. Lors de l'opération de croisement, certains chromosomes représentant un schéma donné H sont modifiés, et ils peuvent ne plus représenter ce schéma dans la génération suivante. Un chromosome de longueur l peut être coupé en $l-1$ positions différentes. Lorsque ce chromosome représente un schéma H de longueur $\delta(H)$, la probabilité p_1 pour que le schéma soit détruit lors de l'opération de croisement est égale à :

$$p_1 = \frac{\delta(H)}{l-1} \quad . \tag{2.12}$$

La probabilité p_2 pour que le schéma soit conservé à l'issue de l'opération de croisement et donc égale à :

$$p_2 = 1 - p_1 = 1 - \frac{\delta(H)}{l-1} \quad . \tag{2.13}$$

Si on suppose que l'opération de croisement est réalisée sur une paire de chromosomes selon une probabilité fixée p_c, en utilisant l'équation 2.11, on peut obtenir une minoration du nombre d'individus représentant le schéma H dans la génération $t+1$:

$$n(H, t+1) \geq n(H, t)\frac{f(H)}{\bar{f}} \cdot (1 - p_c \cdot \frac{\delta(H)}{l-1}) \tag{2.14}$$

L'interprétation de cette dernière équation se résume à la constatation suivante : les schémas sont propagés en fonction de leur adaptation moyenne et de leur longueur. Les schémas de faible longueur et dont l'adaptation est supérieure à la moyenne sont plus enclins à être reproduits.

Finalement, il faut prendre en compte l'opérateur de mutation. Une mutation est appliquée à un chaque élément d'un chromosome avec une probabilité p_m. Un schéma

survit donc à l'opération de mutation avec une probabilité :

$$(1 - p_m)^{o(H)} \quad , \tag{2.15}$$

dans laquelle $o(H)$ désigne l'ordre du schéma, c'est à dire le nombre d'éléments du chromosome dont la valeur est spécifiée. Lorsque la probabilité de mutation est faible, le développement limité au premier ordre de cette probabilité de survie du schéma donne :

$$(1 - p_m)^{o(H)} \approx 1 - o(H) \cdot p_m \quad . \tag{2.16}$$

Finalement, on peut minorer le nombre $n(H, t+1)$ de chromosomes représentant le schéma H dans la génération $t+1$, à l'issue des opérations de sélection, croisement et mutation, par l'expression suivante :

$$\begin{aligned} n(H,t+1) &\geq n(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot (1 - p_c \cdot \frac{\delta(H)}{l-1} - o(H) \cdot p_m) \\ &\geq n(H,t) \cdot \alpha(H,t) \quad , \end{aligned} \tag{2.17}$$

dans laquelle $\alpha(H,t)$ représente le taux d'accroissement du nombre d'individus représentant le schéma H entre la génération t et la génération $t+1$. Si ce taux est supérieur à un, le schéma H est représenté par un nombre d'individus plus important dans la nouvelle génération.

Cette constatation permet d'énoncer le théorème des schémas [Hol75].

Théorème 2.1 (Théorème des schémas [Hol75]) *Les schémas courts, d'ordre faible et d'adaptation supérieure à la moyenne sont propagés exponentiellement dans les générations suivantes.*

2.3.2 Conséquences du théorème des schémas

Dans un algorithme génétique, un chromosome correspond à une solution possible du problème : c'est un élément de l'espace de recherche. Si on considère qu'un schéma est un sous-ensemble de l'espace de recherche, on constate que le théorème des schémas permet d'expliquer simplement certaines propriétés des algorithmes génétiques. En effet, lorsqu'un schéma bien adapté se reproduit dans les générations successives, il correspond à un sous-ensemble qui a de fortes chances de contenir une *bonne* solution du problème.

2.3.2.1 Le parallélisme implicite

Une population contenant N_p chromosomes de longueur l peut représenter un nombre de schémas compris entre 2^l (si tous les chromosomes sont identiques) et $N_p \cdot 2^l$ (si tous les chromosomes sont différents). Le théorème des schémas permet d'étudier l'évolution de cette représentation dans les générations successives. Durant le passage d'une génération à la suivante, l'algorithme génétique va modifier la représentation des schémas en modifiant la population.

Si on considère qu'un chromosome représente un nombre important de schémas, on peut s'attendre à ce que le nombre de schémas évalués à chaque génération soit bien supérieur au nombre d'individus constituant la population. En se basant sur le théorème des schémas, on peut calculer le nombre de schémas traités par l'algorithme génétique lors d'un changement de population. Holland a montré que ce nombre est proportionnel à la taille de la population N_p élévé au cube [Hol75, Gol89].

C'est ce que Holland appelle le *parallélisme implicite* : quand un algorithme traite les N_p chromosomes constituant la population, il manipule en fait N_p^3 schémas. Cette constatation très importante explique en partie pourquoi les algorithmes génétiques permettent d'explorer efficacement l'espace des solutions d'un problème.

2.3.2.2 La convergence prématurée

La convergence prématurée est l'un des problèmes majeurs rencontrés dans tous les algorithmes itératifs. Dans le cas des algorithmes génétiques, on appelle convergence prématurée le fait d'aboutir à une population stable d'individus qui ont convergé vers un maximum local de la fonction d'évaluation plutôt que vers le maximum global. Cela se produit lorsque l'évolution de la population a été trop rapide, ce qui n'a pas permis d'explorer sufisamment l'espace des solutions.

Ce problème est fortement lié au choix des paramètres régissant le fonctionnement de l'algorithme génétique, c'est à dire la taille de la population N_p, et les probabilités de croisement p_c et de mutation p_m. En analysant l'équation 2.17, on constate que le taux d'accroissement $\alpha(H,t)$ de la représentation d'un schéma comporte deux termes. Le premier terme, supérieur à un pour les *bons* schémas, estime l'influence de l'opération de sélection. Il implique une augmentation de la représentation des *bons* schémas dans les

générations successives. Par contre, le second terme, qui estime l'influence des opérations de croisement et de mutation, est inférieur à un. Il entraîne donc une diminution du nombre de chromosomes représentant un schéma donné dans les générations successives.

On vérifie ainsi l'utilité des opérateurs génétiques : l'opération de sélection assure la convergence vers une bonne solution alors que les opérations de croisement et de mutation permettent d'explorer l'espace des solutions. Si on favorise l'opération de sélection, on peut aboutir à une convergence prématurée. Par contre, si on favorise les opérations de croisement et de mutation, l'algorithme peut ne pas converger. Dans le chapitre 5, nous discuterons en détails du problème du choix des paramètres d'un algorithme génétique.

2.4 Autres méthodes de codage

Dans les sections précédentes, nous avons décrit le principe de fonctionnement d'un algorihme génétique standard, qui repose sur un codage des informations dans une chaîne binaire. Holland a montré que cette méthode de codage est en principe la plus efficace, car elle fait intervenir un alphabet comportant uniquement deux symboles. Les schémas associés à cet alphabet permettent une exploration optimale de l'espace des solutions [Hol75, Gol89].

Cependant, le codage binaire présente un inconvénient majeur. Lorsque le nombre de paramètres à coder est important et que la précision souhaitée sur la solution est élevée, la chaîne binaire devient très longue. Dans [Mic92], Michalewicz présente un exemple de codage binaire pour un problème faisant intervenir 100 variables prenant leurs valeurs dans l'intervalle $[-500, 500]$, avec une précision souhaitée sur chaque valeur égale à 10^{-6}. La chaîne binaire correspondante contient 3000 bits ! Avec ce codage, si on utilise une probabilité de mutation de 10^{-3}, chaque chromosome est en moyenne affecté par trois opérations de mutation à chaque itération. Dans ces conditions, il est difficile d'aboutir à une population stable d'individus.

Pour résoudre ce problème, divers auteurs ont proposé d'utiliser une autre méthode de codage, basée sur une représentation des variables par des nombres réels [LK89, Dav89, Wri91, Dav91, Mic92, ES93, HLV95d]. En termes d'occupation mémoire, ce codage n'est pas forcément plus intéressant, puisqu'il utilise en général autant des bits en mémoire pour

représenter les variables. Cependant, les opérateurs génétiques associés à cette méthode de codage permettent un traitement plus efficace des problèmes nécessitant une précision importante sur les résultats.

Dans cette section, nous présentons également une méthode de codage faisant intervenir un alphabet contenant un nombre limité de symboles, appelée codage en base n. Cette méthode est très bien adaptée à la représentation de variables prenant un nombre limité de valeurs.

2.4.1 Codage en nombres réels

Dans un algorithme génétique utilisant un codage en nombres réels, chaque chromosome est en fait un vecteur dont les coordonnées sont les variables du processus d'optimisation. Par exemple, si on recherche le maximum de la fonction de deux variables $f(x_1, x_2)$, définie par :

$$f(x_1, x_2) = 21.5 + x_1 \cdot sin(4\pi x_1) + x_2 \cdot sin(20\pi x_2) \ , \tag{2.18}$$

on peut utiliser un chromosome contenant simplement ces deux variables réelles :

x_1	x_2

Si le problème fait intervenir n variables codées en nombres réels, le codage permet de représenter tous les élements de R^n. En pratique, dans la majorité des problèmes, les solutions sont cependant recherchées dans un sous-espace borné de R^n, qui correspond au produit cartésien des n intervalles dans lesquels les variables prennent leurs valeurs.

Lors de l'évolution de la population, les opérateurs génétiques assurant la modification des chromosomes doivent respecter cette contrainte imposée aux valeurs prises par les variables. Plusieurs opérateurs génétiques associés au codage par nombres réels ont été décrits dans la littérature.

2.4.1.1 Opérateurs de croisement

Les deux opérateurs de croisement les plus utilisés dans le cas d'un codage par nombres réels sont le croisement simple et le croisement arithmétique [Mic92]. Nous supposons dans les deux cas que l'opération de croisement est réalisée sur deux chromosomes A et B contenant chacun l nombres réels :

a_1	a_2	$\cdots$	a_{l-1}	a_l
b_1	b_2	$\cdots$	b_{l-1}	b_l

Le croisement simple est similaire à l'opérateur de croisement utilisé dans les algorithmes génétiques standards. Pour combiner deux chromosomes grâce à cet opérateur, on tire au sort un point de croisement entre les positions 1 et $l-1$, puis on échange les sous-chaînes des deux chromosomes situées de part et d'autre de ce point de croisement. Par exemple, les deux chromosomes A et B sont croisés de la façon suivante par rapport à la position p :

a_1	a_2	$\cdots$	a_p	b_{p+1}	$\cdots$	b_{l-1}	b_l
b_1	b_2	$\cdots$	b_p	a_{p+1}	$\cdots$	a_{l-1}	a_l

Le croisement arithmétique repose sur un principe différent. Dans cette opération, on considère que les deux chromosomes à combiner A et B sont des vecteurs $\vec{A}$ et $\vec{B}$ de R^l, que l'on va remplacer par deux autres vecteurs $\vec{A'}$ et $\vec{B'}$, chacun étant une combinaison linéaire des deux vecteurs initiaux. Ces combinaisons linéaires dépendent d'un paramètre unique β, intervenant de la façon suivante :

$$\begin{aligned} \vec{A'} &= \beta \cdot \vec{A} + (1-\beta) \cdot \vec{B} \\ \vec{B'} &= \beta \cdot \vec{B} + (1-\beta) \cdot \vec{A} \ . \end{aligned} \tag{2.19}$$

Les deux nouveaux chromosomes A' et B' sont donc constitués des nombres suivants :

$\beta a_1 + (1-\beta) b_1$	$\cdots$	$\beta a_{l-1} + (1-\beta) b_{l-1}$
$\beta b_1 + (1-\beta) a_1$	$\cdots$	$\beta b_{l-1} + (1-\beta) a_{l-1}$

Lorsque le coefficient β est constant, le croisement arithmétique est qualifié d'uniforme. Dans le cas d'un croisement non uniforme, ce paramètre est tiré au sort dans l'intervalle $[0, 1]$ pour chaque opération de croisement réalisée [Mic92].

2.4.1.2 Opérateurs de mutation

Dans les algorithmes génétiques basés sur un codage par nombres réels, on utilise principalement deux opérateurs de mutation : la mutation uniforme et la mutation non-uniforme [Mic92]. En supposant fixée la probabilité de mutation p_m, un tirage au sort

pour chaque élément a_k d'un chromosome A permet de décider si cet élément doit être ou non modifié. Nous supposons que l'élément a_k prend ses valeurs dans un intervalle $[c_k, d_k]$.

Pour la mutation uniforme, qui est une simple extension de la mutation binaire, on remplace l'élement a_k sélectionné par une valeur quelconque a'_k tirée au sort dans l'intervalle $[c_k, d_k]$. Ainsi, le chromosome initial A est remplaçé par le chromosome A' défini par :

a_1	a_2	$\cdots$	a'_k	$\cdots$	a_{l-1}	a_l

Pour la mutation non uniforme, le calcul de la nouvelle valeur d'un élément est un peu plus complexe. Le principe consiste à remplacer la valeur initiale a_k par une valeur a'_k d'autant plus proche de cette valeur initiale qu'on a progressé dans le processus d'optimisation. Cela permet de diminuer l'influence de l'opération de mutation lorsqu'on est en phase de convergence.

En pratique, on tire au sort une valeur binaire b qui décidera si le nouvel élément a'_k doit être supérieur ou inférieur à l'élément initial a_k. La nouvelle valeur est alors définie par :

$$a'_k = \begin{cases} a_k + \Delta(t, d_k - a_k) & \text{si } b \text{ est égal à 1} \\ a_k - \Delta(t, a_k - c_k) & \text{si } b \text{ est nul} \end{cases} \tag{2.20}$$

où t désigne le numéro de la génération, et $\Delta(t, x)$ est une fonction qui définit l'écart entre la nouvelle valeur et la valeur initiale.

Dans [Mic92], Michalewicz propose d'utiliser une fonction $\Delta(t, x)$ correspondant à une décroissance exponentielle de l'écart lors des générations successives. Cette fonction est définie par :

$$\Delta(t, x) = x \cdot (1 - r^{(1-\frac{t}{T})^{e_m}}) \ , \tag{2.21}$$

où T est le nombre maximum d'itérations de l'algorithme génétique, e_m est un paramètre de l'opérateur de mutation qui permet de régler la dispersion des écarts et r est un nombre tiré au hasard dans l'intervalle $[0, 1]$.

2.4.2 Codage en base n

Dans l'exemple traité en début de chapitre, nous avons décrit le principe du codage d'une valeur par une sous-chaîne binaire constituée d'un nombre m de bits fixé en fonction de la précision souhaitée. Chaque bit de la sous-chaîne binaire est traité de façon

indépendante par les opérateurs de croisement et de mutation, ce qui permet d'explorer les 2^m possibilités de codage.

Lorsqu'on utilise le codage binaire pour représenter un ensemble discret de valeurs, dont le cardinal n'est pas une puissance de 2, les opérateurs génétiques peuvent faire apparaître des combinaisons binaires ne correspondant pas à un codage valide. Par exemple, si on utilise le codage binaire pour représenter l'ensemble de trois couleurs {rouge,vert,bleu} selon :

rouge → | 0 | 0 | vert → | 0 | 1 | bleu → | 1 | 0 |

une opération de croisement entre deux chromosomes décrivant les couleurs «vert» et «bleu» fait apparaître deux nouveaux chromosomes, dont un qui ne représente pas une couleur valide. Le même problème se pose lors d'une mutation du premier bit d'un chromosome représentant la couleur «vert».

On doit alors faire intervenir une contrainte de validité sur les chromosomes résultant des opérations de croisement et de mutation. Cette solution présente deux inconvénients. Tout d'abord, le temps de calcul est augmenté, ce qui nuit globalement à l'efficacité de l'algorithme. Ensuite, que doit on faire lorsqu'on détecte un codage non valide ? Recommencer l'opération jusqu'à obtention d'un résultat valide, ou remplacer le code non valide par un code par défaut ?

Pour contourner ce problème, certains auteurs ont proposé une autre méthode de codage, appelée codage en base n, et ont décrit les opérateurs génétiques associés [HM95a, LRG96]. Dans cette méthode de codage, les éléments constituant un chromosome sont des chiffres exprimés dans une base de numération n, ce qui permet de représenter n valeurs discrètes.

Les opérateurs de croisement et de mutation adaptés au codage en base n sont de simples extensions des opérateurs standard. Pour le croisement, les chromosomes sont scindés puis recombinés entre deux éléments codés en base n. Pour la mutation, l'élément initial est remplacé par un chiffre en base n tiré au sort.

2.5 Conclusion

Dans ce chapitre, nous avons décrit brièvement le principe des algorithmes génétiques, qui permettent d'aborder la majorité des problèmes d'optimisation en utilisant un formalisme simple. Dans le chapitre suivant de cette thèse, nous verrons comment utiliser les algorithmes génétiques pour optimiser la structure et les paramètres de fonctionnement d'un contrôleur flou.

Nous avons vu également que les paramètres régissant l'évolution de la population (nombre d'individus, probabilités de croisement et de mutation) ont une influence importante sur la convergence d'un algorithme génétique. Nous verrons dans le dernier chapitre de cette thèse comment on peut utiliser un contrôleur flou pour ajuster dynamiquement les paramètres d'un algorithme génétique afin de mieux contrôler sa convergence.

Chapitre 3

Optimisation d'un contrôleur flou par Algorithme Génétique

3.1 Introduction

Comme nous l'avons vu dans le chapitre 1, les contrôleurs flous permettent de piloter des systèmes complexes ou difficilement modélisables en utilisant une méthode de raisonnement de type «si 'condition' alors 'action'». Récemment, des auteurs ont prouvé qu'il est possible de reproduire le fonctionnement de n'importe quel contrôleur continu standard – avec une précision arbitraire – grâce à un contrôleur flou [Wan92, Kos92, FY94, Yin94, Cas95, GF95]. Pourtant, lorsque le système à commander est connu de façon imprécise, la conception du contrôleur flou n'est pas chose triviale.

Le fonctionnement d'un contrôleur flou dépend principalement des caractéristiques de trois sous-systèmes : la fuzzification, les règles floues et la défuzzification. La phase de fuzzification est parfaitement spécifiée lorsque sont définies les fonctions d'appartenance des termes linguistiques décrivant les entrées. Les règles floues sont issues d'une analyse des connaissances fournies par un expert humain et de certaines caractéristiques du système commandé. La méthode utilisée pour la défuzzification doit permettre de transformer une fonction d'appartenance en une valeur de commande envoyée au système.

Plusieurs méthodes de conception, permettant de spécifier les différents paramètres dont dépend le fonctionnement d'un contrôleur flou, ont été décrites dans la littérature. Elles sont pour la plupart basées sur un apprentissage qui permet de définir de façon

itérative le meilleur jeu de paramètres pour une structure donnée de contrôleur. Pour l'instant, les chercheurs se sont concentrés principalement sur les approches suivantes :

- Optimisation des fonctions d'appartenance,
- Optimisation des règles floues,
- Optimisation simultanée des fonctions d'appartenance et des règles floues.

Pour optimiser les fonctions d'appartenance, Karr [Kar91b, KG93] utilise un algorithme génétique standard (codage binaire, croisement et mutation). Le nombre de termes linguistiques est fixé a priori et les fonctions d'appartenance sont codées dans un chromosome en spécifiant les bornes des intervalles flous. Plusieurs variantes de cette méthode ont été proposées par la suite :

- Herrera utilise un algorithme génétique dans lequel les paramètres des fonctions d'appartenance sont codés par des nombres réels [HLV95c, HLV95a].
- Shimojima et al. proposent d'utiliser des fonctions d'appartenance radiales [SFH95], alors que Liska utilise des fonctions de forme Gaussienne [LM94].
- Tang et al. ajoutent au codage binaire un bit de contrôle pour chaque fonction d'appartenance, ce qui permet éventuellement d'éliminer les fonctions inutiles [TMC94, MTKH97].

Thrift est le premier à décrire une méthode d'optimisation des règles floues par algorithme génétique, en utilisant trois bits pour coder chaque règle [Thr91]. A la même époque, Karr propose une méthode permettant de faire intervenir dans le contrôleur flou à la fois des règles spécifiées par un expert humain et des règles optimisées par un algorithme génétique [Kar91a]. Quelques améliorations ont été apportées par la suite :

- Herrera utilise un codage des paramètres des règles floues par des nombres réels [HLV97].
- Hoffmann utilise l'algorithme génétique de Messy pour optimiser les règles floues [HP95].
- Lim et al. optimisent les règles floues grâce à un codage par paires [LRG96].
- Liska utilise un codage en nombres réels pour décrire une règle d'inférence à plusieurs entrées et une seule sortie. Un bit supplémentaire permet de supprimer éventuellement des règles floues inutiles [LM94].

Lee et Takagi proposent, en 1993, une méthode d'optimisation qui prend en compte simultanément les fonctions d'appartenance de la fuzzification et les règles floues [LT93a, HLTT93]. Les différents paramètres sont codés dans un chromosome unique sous forme binaire.

Par la suite, Chwee et Lee [NL94] optimisent un contrôleur flou comportant deux entrées et une seule sortie. Dans leur méthode, les chromosomes sont des chaînes exprimées en base sept, ce qui permet de représenter au mieux les sept fonctions d'appartenance utilisées dans la variable linguistique de sortie.

Dans un article de Homaifar et McCormick [HM95a], on découvre un algorithme génétique chargé d'optimiser un contrôleur comportant deux entrées, une sortie et vingt-cinq règles floues. Ces auteurs utilisent un codage des chromosomes en base cinq.

Dans sa thèse, Leitch [Lei95] présente un codage particulier des chromosomes : chaque paramètre est décrit par une sous-chaîne de longueur variable, les sous-chaînes étant séparées par des marqueurs spéciaux. Ce codage permet de modifier simplement le nombre de règles floues à optimiser.

Dans la même optique que les travaux de Karr [Kar91a], Herrera et al. [HLV95a, CH96] ont proposé une méthode dans laquelle les règles floues sont construites en deux parties, l'une venant d'une expertise humaine, l'autre étant optimisée par un algorithme génétique.

Toutes les méthodes citées précédemment visent à l'optimisation des phases de fuzzification et d'inférence floue. Jusqu'à présent, la phase de défuzzification n'a pas retenu l'attention des chercheurs. Pourquoi avoir laissé cette partie du contrôleur à l'écart alors que toutes les approches basées sur une optimisation simultanée des différents sous-ensembles s'avèrent être les plus efficaces ? En toute logique, une méthode traitant globalement les trois sous-ensembles d'un contrôleur flou devrait mener à de meilleures solutions.

Dans ce chapitre, nous présentons tout d'abord des méthodes existantes qui permettent d'optimiser la fuzzification et les règles floues. Ensuite, nous examinons comment la phase de défuzzification peut être optimisée par un algorithme génétique. Nous étudions également différentes méthodes de codage permettant de regrouper tous les paramètres d'un contrôleur flou dans un seul chromosome.

3.2 Optimisation des fonctions d'appartenance par un Algorithme Génétique

La paramétrisation automatique des fonctions d'appartenance par un algorithme génétique a été proposée initialement par Karr pour réaliser un contrôleur de pH [Kar91b, KG93]. Depuis, de nombreux auteurs ont entamé des recherches dans ce domaine [LT93a, LT93b, TMC94, FNU95, SFH95, HLV95c, MTKH97, HL97].

Dans cette section, nous présentons les méthodes de codage des fonctions d'appartenance. Nous voyons tout d'abord comment il est possible de représenter les fonctions d'appartenance sur un chromosome. Ensuite, nous abordons le problème du choix entre un codage binaire et un codage en nombres réels.

3.2.1 Représentation des fonctions d'appartenance

Dans un contrôleur flou, les règles de raisonnement sont appliquées sur des termes linguistiques. Ces termes, qui permettent de qualifier une variable linguistique, sont définis par l'intermédiaire de fonctions d'appartenance. La représentation se fait donc en deux étapes : on doit d'abord représenter chaque fonction d'appartenance, puis la variable linguistique dans sa totalité.

Comme nous l'avons vu dans le chapitre 1, les fonctions d'appartenance les plus classiquement utilisées sont de type triangulaire, trapézoïdal ou Gaussien.

3.2.1.1 Représentation des fonctions d'appartenance triangulaires

Une fonction d'appartenance triangulaire dans un univers du discours fini $[a, b]$ peut être définie par :

$$\mu_A(x) = \begin{cases} 0, & x \leq x_1 \\ \frac{x-x_1}{x_2-x_1}, & x_1 \leq x \leq x_2 \\ \frac{x_3-x}{x_3-x_2}, & x_2 \leq x \leq x_3 \\ 0, & x_3 \leq x \end{cases} \tag{3.1}$$

ou encore, de façon simplifiée, par :

$$\mu_A(x) = max(min(\frac{x - x_1}{x_2 - x_1}, \frac{x_3 - x}{x_3 - x_2}), 0) \quad . \tag{3.2}$$

On constate que cette fonction dépend des trois paramètres x_1, x_2 et x_3 qui prennent leurs valeurs dans l'intervalle $[a, b]$, comme on le voit sur la figure 3.1.

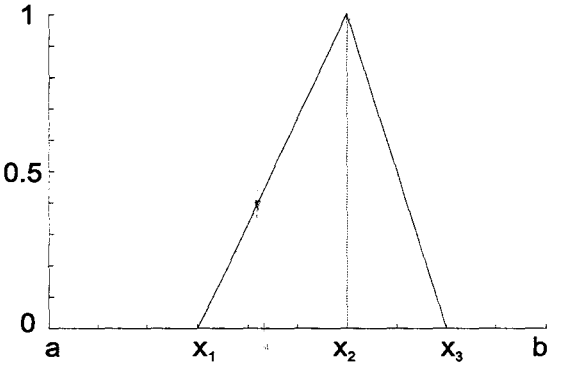


FIG. 3.1 : Fonction d'appartenance triangulaire

Cette fonction d'appartenance est parfaitement définie par ces trois paramètres, on peut donc utiliser la représentation suivante :

$c_{[a,b]}(x_1)$	$c_{[a,b]}(x_2)$	$c_{[a,b]}(x_3)$

où $c_{[a,b]}(x_k)$ est un codage (en binaire, en base n ou en nombre réel) de la variable x_k qui prend ses valeurs dans l'intervalle $[a, b]$. Les valeurs codées doivent respecter les inégalités suivantes :

$$x_1 \leq x_2 \leq x_3. \tag{3.3}$$

Lee et Takagi ont proposé une autre méthode de représentation des fonctions d'appartenance triangulaires [LT93a, LT93b]. En notant x_c l'abscisse du sommet de la fonction d'appartenance, les deux points extrêmes sont repérés par leurs abscisses $(x_c - x_1)$ et $(x_c + x_2)$ où x_1 et x_2 prennent leurs valeurs dans l'intervalle $[0, (b - a)/2]$ (cf. figure 3.2).

Cette représentation présente l'avantage de ne pas imposer de condition aux limites sur les valeurs des paramètres. La représentation d'une fonction triangulaire selon Lee et Takagi est ainsi donnée par :

$c_{[a,b]}(x_c)$	$c_{[0,(b-a)/2]}(x_1)$	$c_{[0,(b-a)/2]}(x_2)$

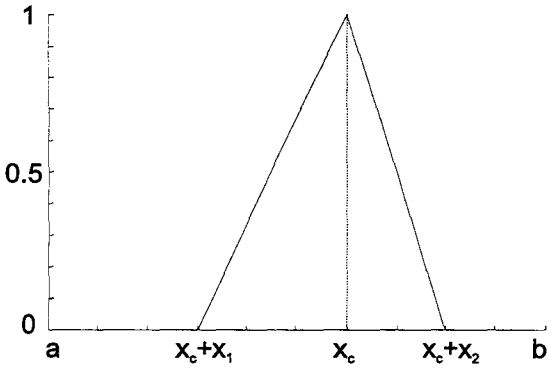


FIG. 3.2 : Codage d'une fonction triangulaire par Lee

Dans un contrôleur flou, la variable linguistique complète peut être définie par l'ensemble des fonctions d'appartenance des termes linguistiques, comme le montre la figure 3.3.

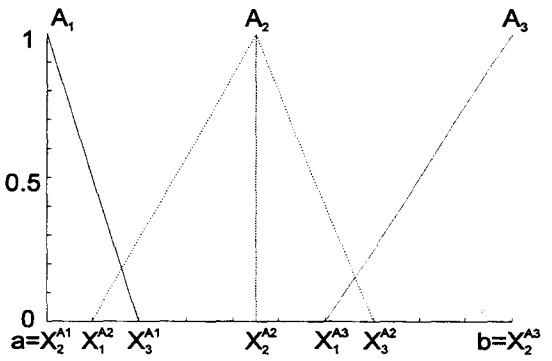


FIG. 3.3 : Variable linguistique complète

Lorsque les termes linguistiques constituent une partition floue de l'entrée du système, on peut simplifier la représentation. En effet, certains paramètres interviennent plusieurs fois dans la définition des fonctions d'appartenance. La figure 3.4 représente une partition floue définie par 5 fonctions d'appartenance triangulaires. Ce type de variable linguistique a été proposé par Lee et Yubazaki [LT93a, YOAH95].

Une partition floue comportant n termes linguistiques est complètement définie par l'intermédiaire de n points $(x_1 = a, x_2, \cdots, x_n = b)$ en utilisant les fonctions d'appartenance triangulaires suivantes :

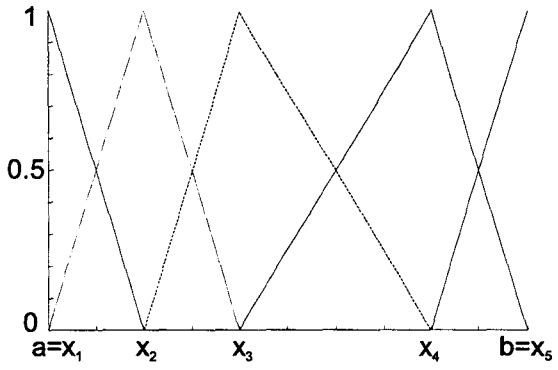


FIG. 3.4 : Partition floue avec 5 fonctions d'appartenance triangulaires

$$\mu_{A_1}(x) = \begin{cases} 0, & x < x_1 \\ \frac{x_2 - x}{x_2 - x_1}, & x_1 \leq x \leq x_2 \\ 0, & x > x_2 \end{cases}$$

$$\mu_{A_i}(x) = \begin{cases} 0, & x < x_i \\ \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} \leq x < x_i \quad i = 2, \cdots, n-1 \\ \frac{x_{i+1} - x}{x_{i+1} - x_i}, & x_i \leq x \leq x_{i+1} \\ 0, & x > x_{i+1} \end{cases} \tag{3.4}$$

$$\mu_{A_n}(x) = \begin{cases} 0, & x < x_{n-1} \\ \frac{x - x_{n-1}}{x_n - x_{n-1}}, & x_{n-1} \leq x \leq x_n \\ 0, & x > x_n \end{cases}$$

La partition floue peut donc être représentée par le codage des n points :

$c_{[a,b]}(x_2)$	$\cdots$	$c_{[a,b]}(x_i)$	$\cdots$	$c_{[a,b]}(x_{n-1})$

et par les conditions aux limites :

$$a = x_1 \leq x_2 \leq \cdots \leq x_n = b. \tag{3.5}$$

Cette représentation limite fortement l'efficacité des traitements, en raison des $n-1$ inégalités gérant les conditions aux limites. Nous proposons une représentation basée sur les différences d_i entre deux points caractéristiques successifs x_{i+1} et x_i. Cela correspond

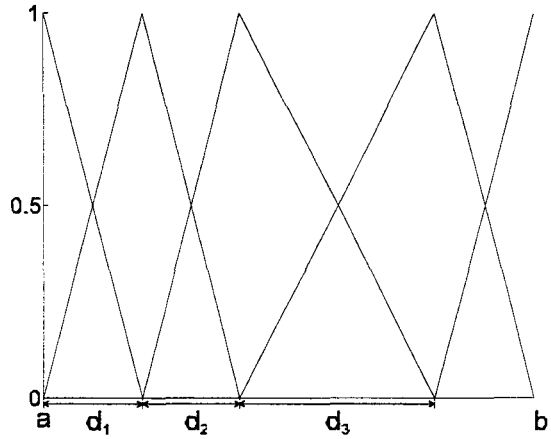


FIG. 3.5 : Partition floue représentée en utilisant les différences d'abscisses

à une extension de la méthode de Lee [LT93a, LT93b] à la représentation d'une variable linguistique. Nous représentons la partition floue comme l'indique la figure 3.5.

En utilisant les différences d'abscisses plutôt que les abscisses elles-mêmes, on aboutit à la représentation suivante :

$c_{[0,(b-a)]}(d_1)$	$\cdots$	$c_{[0,(b-a)]}(d_i)$	$\cdots$	$c_{[0,(b-a)]}(d_{n-1})$

avec, pour seule condition limite imposée :

$$d_1 + d_2 + \cdots + d_{n-1} \leq (b-a). \tag{3.6}$$

3.2.1.2 Représentation des fonctions d'appartenance trapézoïdales

Une fonction d'appartenance trapézoïdale dans un univers du discours fini $[a, b]$ peut être définie par :

$$\mu_A(x) = \begin{cases} 0, & x \leq x_1 \\ \frac{x-x_1}{x_2-x_1}, & x_1 \leq x \leq x_2 \\ 1, & x_2 \leq x \leq x_3 \\ \frac{x_4-x}{x_4-x_3}, & x_3 \leq x \leq x_4 \\ 0, & x_5 \leq x \end{cases} \tag{3.7}$$

ou encore par :

$$\mu_A(x) = max(min(\frac{x-x_1}{x_2-x_1}, 1, \frac{x_4-x}{x_4-x_3}), 0) \tag{3.8}$$

Cette fonction est définie par quatre paramètres x_1, x_2, x_3 et x_4 qui prennent leurs valeurs dans l'intervalle $[a, b]$. La figure 3.6 illustre cette fonction. Notons que lorsque $x_2 = x_3$, la fonction trapézoïdale dégénère en une fonction triangulaire.

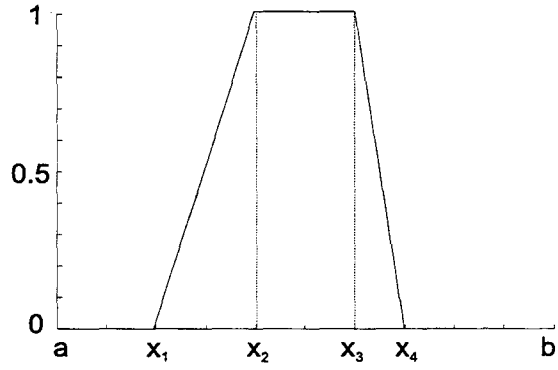


FIG. 3.6 : Fonction d'appartenance trapézoïdale

La représentation de la fonction d'appartenance trapézoïdale fait intervenir ces quatre paramètres :

$c_{[a,b]}(x_1)$	$c_{[a,b]}(x_2)$	$c_{[a,b]}(x_3)$	$c_{[a,b]}(x_4)$

et les conditions aux limites associées sont :

$$x_1 \leq x_2 \leq x_3 \leq x_4. \tag{3.9}$$

Une variable linguistique dont les termes constituent une partition floue définie par 5 fonctions d'appartenance trapézoïdales est représentée sur la figure 3.7.

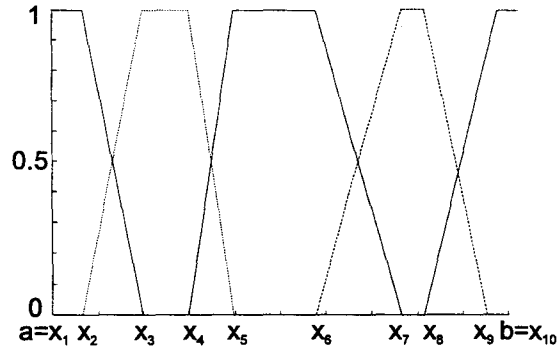


FIG. 3.7 : Partition floue avec 5 fonctions d'appartenance trapézoïdales

Plus généralement, une partition floue avec n fonctions d'appartenance trapézoïdales est définie par $2n$ points $(a = x_1, x_2, \cdots, x_{2n} = b)$ et par les équations suivantes :

$$\mu_{A_1}(x) = \begin{cases} 1, & x_1 \leq x \leq x_2 \\ \frac{x_3 - x}{x_3 - x_2}, & x_2 < x < x_3 \\ 0, & x \geq x_3 \end{cases}$$

$$\mu_{A_i}(x) = \begin{cases} 0, & x \leq x_{2i-2} \\ \frac{x - x_{2i-2}}{x_{2i-1} - x_{2i-2}}, & x_{2i-2} \leq x < x_{2i-1} \\ 1, & x_{2i-1} \leq x < x_{2i} \quad i = 2, \cdots, n-1 \\ \frac{x_{2i+1} - x}{x_{2i+1} - x_{2i}}, & x_{2i} \leq x \leq x_{2i+1} \\ 0, & x > x_{2i+1} \end{cases} \tag{3.10}$$

$$\mu_{A_n}(x) = \begin{cases} 0, & x \leq x_{2n-2} \\ \frac{x - x_{2n-2}}{x_{2n-1} - x_{2n-2}}, & x_{2n-2} < x < x_{2n-1} \\ 1, & x_{2n-1} \leq x \leq x_{2n-2} \\ 0, & x > x_{2n} \end{cases}$$

Dans ce cas, la représentation est :

$c_{[a,b]}(x_2)$	$\cdots$	$c_{[a,b]}(x_{2i})$	$\cdots$	$c_{[a,b]}(x_{2n-1})$

et les conditions aux limites sont de la forme :

$$a = x_1 \leq x_2 \leq \cdots \leq x_{10} = b \ . \tag{3.11}$$

Nous pouvons également représenter cette partition floue en utilisant les différences d'abscisses $d_{i-1} = x_i - x_{i-1}$, comme le montre la figure 3.8.

La représentation associée à cette procédure prend la forme :

$c_{[0,(b-a)]}(d_1)$	$\cdots$	$c_{[0,(b-a)]}(d_{2i-1})$	$\cdots$	$c_{[0,(b-a)]}(d_{2n-2})$

pour laquelle l'unique condition aux limites s'écrit :

$$d_1 + d_2 + \cdots + d_{2n-2} \leq (b - a) \ . \tag{3.12}$$

3.2.1.3 Représentation des fonctions d'appartenance Gaussiennes

Une fonction d'appartenance Gaussienne dans un univers du discours fini $[a, b]$ peut être exprimée sous la forme :

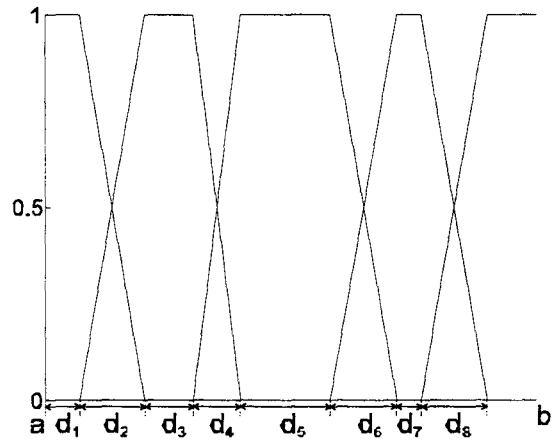


FIG. 3.8 : Codage d'une partition floue utilisant les différences d'abscisses

$$\mu_A(x) = e^{\frac{-(x-x_1)^2}{2\sigma^2}} \tag{3.13}$$

Cette fonction est définie par deux paramètres x_1, σ qui prennent respectivement leurs valeurs dans l'intervalle $[a, b]$ et et dans l'intervalle $[0, \infty[$. En pratique, l'écart type σ prend ses valeurs dans un intervalle plus restreint de valeurs $[c, d]$, qui est fonction de l'application. La figure 3.9 montre une telle fonction.

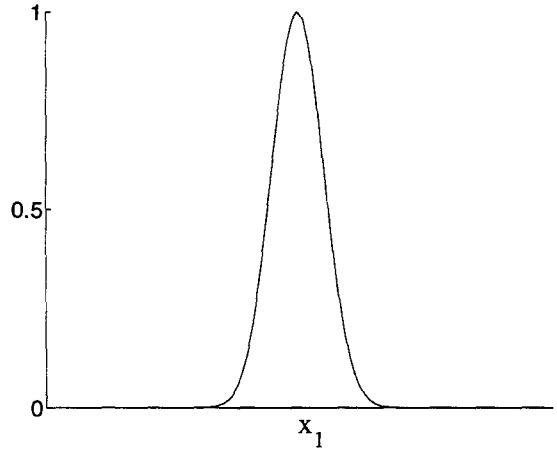


FIG. 3.9 : Fonction d'appartenance Gaussienne

La représentation d'une fonction d'appartenance Gaussienne est alors donnée par le codage suivant :

$c_{[a,b]}(x_1)$	$c_{(c,d]}(\sigma)$

Lorsque les termes linguistiques sont caractérisés par des fonctions d'appartenance Gaussiennes, la variable linguistique ne peut pas être une partition floue de l'univers du discours. Avec des fonctions Gaussiennes, il existe toujours une zone de recouvrement entre deux termes linguistiques. On ne peut donc pas simplifier la représentation en utilisant les intersections entre deux fonctions d'appartenance adjacentes. La variable linguistique complète est représentée par la totalité des fonctions d'appartenance de ses termes :

$c_{[a,b]}(x_1)$	$c_{[c,d]}(\sigma_1)$	$\cdots$	$c_{[a,b]}(x_n)$	$c_{[c,d]}(\sigma_n)$

Nous avons vu dans le chapitre 1 que la variable linguistique utilisée pour fuzzifier une entrée d'un contrôleur flou doit satisfaire une contrainte concernant la séparation de ses différents termes linguistiques. Il faut donc limiter autant que possible la zone de recouvrement entre deux termes linguistiques non consécutifs. De plus, au voisinage du maximum de chaque fonction d'appartenance Gaussienne, il faut que les autres fonctions d'appartenance prennent des valeurs aussi faibles que possible.

Pour que la variable linguistique satisfasse ces deux conditions, nous imposons deux contraintes à chaque fonction d'appartenance. Les valeurs prises par la fonction d'appartenance $\mu_{A_i}(x)$ aux abscisses des maxima des fonctions adjascentes $\mu_{A_{i-1}}$ et $\mu_{A_{i+1}}$ doivent être inférieures à un seuil de recouvrement s_r fixé a priori. Ces deux contraintes sont représentées sur la figure 3.10.

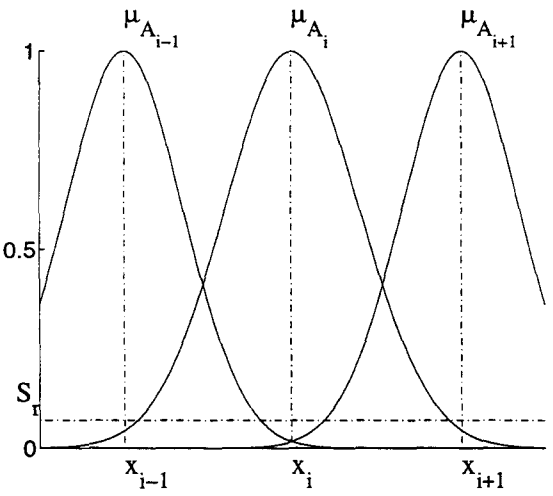


FIG. 3.10 : Contrainte de non-recouvrement des fonctions d'appartenance

Toutes ces contraintes sont imposées par les inégalités suivantes :

$$\mu_{A_{i+1}}(x_i) \leq s_r \text{ et } \mu_{A_i}(x_{i+1}) \leq s_r, \quad i = 1, \cdots, n-1 \tag{3.14}$$

3.2.1.4 Représentation d'autres fonctions d'appartenance

Il existe d'autres fonctions d'appartenance, telle que la fonction «courbe généralisée en cloche», définie par :

$$\mu_A(x) = \frac{1}{1 + |\frac{x-x_1}{\alpha}|^{2\beta}}, \quad \alpha > 0, \ \beta > 0 \tag{3.15}$$

où le paramètre x_1 indique la position du mode de la fonction, α et β permettant d'ajuster respectivement l'étendue et l'aspect de la courbe. La figure 3.11 montre une telle fonction.

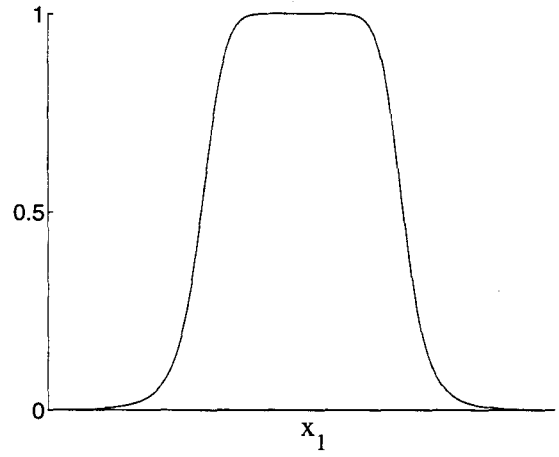


FIG. 3.11 : Fonction d'appartenance de type courbe généralisée en cloche

La représentation d'une telle fonction d'appartenance fait intervenir le codage des trois paramètres sur trois intervalles de valeurs dépendant de l'application :

$c_{[a,b]}(x_1)$	$c_{[c,d]}(\alpha)$	$c_{[e,f]}(\beta)$

De façon générale, l'utilisation de fonctions non standard implique le codage de nombreux paramètres indépendants pour chaque fonction d'appartenance. Dès lors, la variable linguistique complète ne peut être représentée que par la totalité des paramètres, puisqu'on ne peut pas exploiter de redondance.

3.2.2 Codage des variables linguistiques

Les variables linguistiques sont représentées par un certain nombre de paramètres, comme nous l'avons vu dans la section précédente. Le type de codage utilisé pour ces différents paramètres au sein d'un même chromosome dépend à la fois de la précision souhaitée sur les valeurs et de l'étendue de la gamme des valeurs admissibles.

Lorsqu'un paramètre prend ses valeurs dans un intervalle borné $[a, b]$, on utilise souvent un codage sous forme de chaîne binaire, qui permet de fixer aisément la précision requise sur les valeurs du paramètre. Par contre, lorsque le paramètre prend ses valeurs dans un intervalle non borné $[a, \infty[$, on est amené à utiliser un codage sous forme de nombre réel.

3.2.2.1 Codage binaire des paramètres

Ce type de codage permet d'utiliser un algorithme génétique standard durant la phase d'optimisation. Nous rappelons brièvement le principe de codage déjà présenté dans le chapitre 2.

Une chaîne binaire $c = (b_{m-1} \cdots b_0)$ comportant m bits permet de coder 2^m valeurs régulièrement espacées dans un intervalle $[a, b]$. Ces valeurs sont données par l'expression :

$$\hat{p}_c = a + (\sum_{i=0}^{m-1} b_i \cdot 2^i) \cdot \frac{b-a}{2^m - 1} \quad . \tag{3.16}$$

Une valeur particulière du paramètre p est codée par la chaîne binaire c décrivant la valeur $\hat{p}_c$ la plus proche de p dans l'intervalle $[a, b]$.

Le nombre m de bits nécessaires pour coder une valeur dépend uniquement de la précision souhaitée et de l'étendue $b - a$ de l'intervalle contenant les valeurs codées, la différence entre une valeur codée et la valeur initiale étant majorée par $(b - a)/(2^m - 1)$.

3.2.2.2 Codage en nombre réel des paramètres

Le codage des paramètres par un vecteur de nombres réels peut être utilisé dans deux cas particuliers :

- lorsque le nombre de paramètres est important et que la précision souhaitée est élevée. Dans ce cas, le codage en nombres réels est plus adapté.

- lorsque l'un des paramètres à coder prend ses valeurs dans un domaine non borné, par exemple dans un intervalle du type $[a, +\infty]$.

Si on utilise un codage par des nombres réels, on ne peut plus utiliser les opérateurs génétiques standards. Ces derniers sont remplacés par les opérateurs arithmétiques uniformes ou non uniformes décrits dans la section 2.4.1 du chapitre 2.

3.3 Optimisation des règles floues par un Algorithme Génétique

Les règles floues permettent d'interpréter les variables linguistiques qui caractérisent les entrées du contrôleur. Plusieurs auteurs se sont penchés sur le problème de l'optimisation des règles d'inférence utilisées dans un contrôleur flou. Certaines approches d'optimisation sont basées sur l'utilisation de réseaux de neurones, qui permettent un apprentissage des règles utiles [Jan93, Kas93, Kas96].

L'optimisation des règles floues par un algorithme génétique a été proposée pour la première fois par Karr [Kar91a] qui a présenté une méthode dans laquelle certaines règles sont définies par un expert alors que d'autres sont issues d'un processus d'optimisation. D'autres auteurs ont apporté par la suite quelques améliorations qui reposent principalement sur une modification de la représentation des règles dans les chromosomes traités par l'algorithme génétique [Thr91, HP95, LM94, LRG96, HLV97].

Dans cette section, nous présentons tout d'abord quelques méthodes de représentation des règles floues. Ces représentations permettent d'optimiser les règles exploitées par la méthode d'inférence de Mamdani, qui est la plus largement utilisée. Ensuite nous verrons comment ces représentations peuvent être codées pour être traitées par divers algorithmes génétiques.

3.3.1 Représentation des règles floues

Dans la section 1.4.4, nous avons noté les règles floues de la façon suivante :

Règle 1 : Si ($\mathcal{X}_1$ est A_{11}) et $\cdots$ et ($\mathcal{X}_m$ est A_{1m}) ; alors ($\mathcal{Y}$ est B_1)

Règle 2 : Si ($\mathcal{X}_1$ est A_{21}) et $\cdots$ et ($\mathcal{X}_m$ est A_{2m}) ; alors ($\mathcal{Y}$ est B_2)

$\ldots$

Règle n : Si ($\mathcal{X}_1$ est A_{n1}) et $\cdots$ et ($\mathcal{X}_m$ est A_{nm}) ; alors ($\mathcal{Y}$ est B_n)

où $\mathcal{X}_1 \in X_1,\ldots,\mathcal{X}_m \in X_m$ sont des qualifications linguistiques des m entrées, A_{ij} le i^{eme} terme linguistique qualifiant l'entrée numéro j et $B_1, \cdots, B_n$ sont des nombres flous définis par les n règles sur l'univers du discours Y.

Pour simplifier la compréhension des méthodes de représentation d'un ensemble de règles floues, nous proposons d'utiliser une notation plus générale. Supposons que le contrôleur flou comporte N_e entrées, qualifiées chacune par N_{te} termes linguistiques. Lorsque les variables linguistiques qualifiant les entrées ne contiennent pas le même nombre de termes linguistiques, N_{te} désigne le nombre de termes linguistiques de la variable qui en comporte le plus. Nous supposons également que la sortie est qualifiée par une variable linguistique définie par N_{ts} termes.

Dans ces conditions, on peut définir au maximum ${N_{te}}^{N_e}$ règles floues, que nous choisissons de repérer par N_e indices variant entre 1 et N_{te}. Une règle est alors notée de la façon suivante :

Règle $i_1, i_2, \cdots, i_{N_e}$: Si ($\mathcal{X}_1$ est A_{1i_1}) et $\cdots$ et ($\mathcal{X}_{N_e}$ est $A_{N_e i_{N_e}}$) ; alors ($\mathcal{Y}$ est B_j)

dans laquelle chaque indice i_k varie entre 1 et N_{te}, A_{uv} désigne le u^{eme} terme linguistique qualifiant l'entrée numéro v et B_j désigne le j^{eme} terme linguistique qualifiant la sortie.

Cette notation peut être étendue pour permettre la représentation de règles comportant plusieurs conclusions, ce qui est nécessaire quand le contrôleur pilote plusieurs sorties. Nous supposons alors que les N_s sorties sont qualifiées par N_{ts} termes linguistiques. Si les variables linguistiques de sortie ne contiennent pas le même nombre de termes, N_{ts} désigne le nombre maximum de termes. La notation d'une règle devient :

Règle $i_1, i_2, \cdots, i_{N_e}$: Si ($\mathcal{X}_1$ est A_{1i_1}) et $\cdots$ et ($\mathcal{X}_{N_e}$ est $A_{N_e i_{N_e}}$) ;
alors ($\mathcal{Y}_1$ est B_{1j_1}) ; $\cdots$; ($\mathcal{Y}_{N_s}$ est $B_{N_s j_{N_s}}$)

dans laquelle chaque indice j_k varie entre 1 et N_{ts} et B_{uv} désigne le u^{eme} terme linguistique qualifiant la sortie numéro v.

3.3.1.1 Règles floues pour deux entrées et une sortie

Nous décrivons tout d'abord un exemple très simple de représentation, dans le cas d'un contrôleur avec deux entrées et une sortie, dont les règles floues possèdent deux prémisses et une seule conclusion :

$$\text{Règle } i_1, i_2 : \quad \text{Si } (\mathcal{X}_1 \text{ est } A_{1i_1}) \text{ et } (\mathcal{X}_2 \text{ est } A_{2i_2}) \quad ; \text{ alors } (\mathcal{Y} \text{ est } B_j)$$
$$i_1 = 1, \cdots, N_{te},\ i_2 = 1, \cdots, N_{te},\ j = 1, \cdots, N_{ts}$$

où $A_{11}, \cdots, A_{1N_{te}}$ sont les termes linguistiques qualifiant la première entrée $\mathcal{X}_1$, $A_{21}, \cdots, A_{2N_{te}}$ sont les termes linguistiques pour la deuxième entrée $\mathcal{X}_2$ et $B_1, \cdots, B_{N_{ts}}$ sont les termes linguistiques qui qualifient la sortie $\mathcal{Y}$.

On peut définir au maximum ${N_{te}}^2$ règles floues pour ce contrôleur, chaque règle pouvant avoir N_{ts} conclusions possibles si elle est définie. Parmi toutes les règles possibles, certaines ne sont pas utiles. Afin de représenter les règles non utilisées, on choisit de les marquer en considérant que la sortie n'est pas qualifiée. Cela correspond à ajouter un $N_{ts} + 1$ ème terme linguistique pour qualifier la sortie, auquel on affecte de façon conventionnelle un indice zéro.

Pour représenter la totalité des règles floues, on dresse ensuite un tableau des conclusions correspondant à toutes les configurations possibles de prémisses. La conclusion d'une règle repérée par les indices i_1 et i_2 est alors représentée par un nombre $R_{i_1 i_2}$ défini selon :

$$R_{i_1 i_2} = \begin{cases} 0, & \text{si la règle } (i_1, i_2) \text{ est inutile} \\ j \in \{1 \ldots N_{ts}\}, & \text{si la conclusion de la règle } (i_1, i_2) \text{ est } B_j \end{cases} \tag{3.17}$$

Un exemple de tableau de ce type est présenté sur la figure 3.12.

Pour constituer la représentation de l'ensemble des règles, on réorganise le tableau bidimensionnel en liste de valeurs en le balayant ligne par ligne. Cela fournit comme représentation :

R_{11}	R_{12}	$\cdots$	$R_{N_{te}N_{te}-1}$	$R_{N_{te}N_{te}}$

3.3.1.2 Règles floues pour deux entrées et plusieurs sorties

Lorsque le contrôleur à deux entrées pilote plusieurs sorties, chaque règle floue comporte plusieurs conclusions. Une règle est alors exprimée de la façon suivante :

	A_{21}	A_{22}	$\cdots$	A_{2i_2}	$\cdots$	$A_{2N_{te}}$
A_{11}	R_{11}	R_{12}		$\cdots$		$R_{1N_{te}}$
A_{12}	R_{21}	R_{22}		$\cdots$		$R_{2N_{te}}$
$\vdots$				$\vdots$		
A_{1i_1}		$\cdots$		$R_{i_1i_2}$		$\cdots$
$\vdots$				$\vdots$		
$A_{1N_{te}}$	$R_{N_{te}1}$			$\cdots$		$R_{N_{te}N_{te}}$

FIG. 3.12 : Règles floues sous forme de tableau

$$\text{Règle } i_1, i_2 : \quad \text{Si } (\mathcal{X}_1 \text{ est } A_{1i_1}) \text{ et } (\mathcal{X}_2 \text{ est } A_{2i_2}) \quad ;$$
$$\text{alors } (\mathcal{Y}_1 \text{ est } B_{1j_1}) ; \cdots ; (\mathcal{Y}_{N_s} \text{ est } B_{N_sjN_s})$$

où B_{uv} désigne le u^{eme} terme linguistique qualifiant la sortie numéro v.

Pour chaque sortie, repérée par un indice j, on constitue un tableau des règles similaire à celui de la figure 3.12. Une valeur $R^j_{i_1i_2}$ contenue dans ce tableau est définie par :

$$R^j_{i_1i_2} = \begin{cases} 0, & \text{si la règle } (i_1, i_2) \text{ est inutile pour la sortie } j \\ k \in \{1 \ldots N_{ts}\}, & \text{si la conclusion de la règle } (i_1, i_2) \text{ est } B_{jk} \end{cases} \tag{3.18}$$

La représentation de l'ensemble des règles est obtenue en balayant les tableaux constitués pour toutes les sorties du contrôleur :

R^1_{11}	R^1_{12}	$\cdots$	$R^1_{N_{te}N_{te}}$	R^2_{11}	$\cdots$	$R^{N_s}_{N_{te}N_{te}}$

3.3.1.3 Règles floues pour plusieurs entrées et plusieurs sorties

Quand le contrôleur flou dispose de plus de deux entrées, le principe de représentation reste le même. Supposons que le contrôleur a N_e entrées, et que les règles floues sont exprimées par :

$$\text{Règle } i_1, i_2, \cdots, i_{N_e} : \quad \text{Si } (\mathcal{X}_1 \text{ est } A_{1i_1}) \text{ et } \cdots \text{ et } (\mathcal{X}_{N_e} \text{ est } A_{N_ei_{N_e}}) \quad ;$$
$$\text{alors } (\mathcal{Y}_1 \text{ est } B_{1j_1}) ; \cdots ; (\mathcal{Y}_{N_s} \text{ est } B_{N_sjN_s})$$

Pour chaque sortie, on constitue un tableau à N_e dimensions, chaque dimension correspondant à une entrée du contrôleur. Ce tableau, pour la sortie numéro j, contient les termes $R^j_{i_1i_2\cdots i_{N_e}}$ définis par :

$$R^j_{i_1 i_2 \cdots i_{N_e}} = \begin{cases} 0, & \text{si la règle } (i_1, i_2, \cdots, i_{N_e}) \text{ est inutile pour la sortie } j \\ k \in \{1 \ldots N_{ts}\}, & \text{si la conclusion de la règle } (i_1, i_2, \cdots, i_{N_e}) \text{ est } B_{jk} \end{cases} \tag{3.19}$$

La représentation est obtenue en balayant les tableaux contenant les conclusions des règles afin de constituer une séquence monodimensionnelle. On constate que la quantité d'informations augmente très rapidement avec le nombre d'entrées, ce qui constitue la principale limitation de ce mode de représentation.

3.3.2 Codage des règles floues

Afin de permettre un traitement par un algorithme génétique, la représentation des règles décrite précédemment doit être codée pour être insérée dans un chromosome. Plusieurs types de codage ont été décrits dans la littérature.

3.3.2.1 Codage des règles floues par un chromosome binaire

Le codage le plus simple consiste à représenter les règles par une chaîne binaire, chaque élément de la représentation étant codé sur m bits. Nous avons vu qu'un élément $R^j_{i1_i2\cdots i_{N_e}}$ prend ses valeurs dans l'ensemble $\{0, \cdots, N_{ts}\}$, où N_{ts} désigne le nombre de termes linguistiques décrivant chaque sortie. Une sous-chaîne binaire décrivant une de ces valeurs doit donc comporter au minimum $m = \mathrm{E}(\log_2(N_{ts} + 1))$ bits, $\mathrm{E}(x)$ désignant la partie entière de x.

Lorsque $N_{ts} + 1$ n'est pas une puissance de deux, certaines configurations binaires ne constituent pas un codage valide. On doit alors ajouter une condition sur la valeur représentée par la sous-chaîne binaire, qui doit toujours être inférieure ou égale à N_{ts}. Cette condition, qui doit être testée à chaque fois que l'on a réalisé une opération sur la chaîne binaire, constitue un des principaux inconvénients du codage binaire des règles floues, car elle diminue fortement l'efficacité des traitements.

3.3.2.2 Codage des règles floues par un chromosome en base n

Dans l'article [HM95a], Homaifar décrit une procédure d'optimisation des règles floues d'un contrôleur dont les sorties sont qualifiées par 5 termes linguistiques. Le nombre 5

n'étant pas une puissance de deux, il propose d'utiliser un codage faisant intervenir une représentation des nombres en base 5, ce qui permet de coder efficacement les 5 valeurs possibles d'une conclusion de règle floue. Dans cet article, Homaifar ne tient pas compte d'éventuelles règles inutiles.

	NM	NP	EZ	PP	PM
NM	1	4	3	2	1
NP	5	2	4	3	2
EZ	1	2	4	5	1
PP	4	3	1	2	2
PM	1	1	3	4	5

FIG. 3.13 : Conclusions des règles floues

Par exemple, le codage des règles floues décrites par le tableau présenté sur la figure 3.13 est réalisé en exprimant par un chiffre en base 5 le nombre $R_{i_1 i_2} - 1$, selon :

0_5	3_5	2_5	1_5	0_5	4_5	1_5	3_5	2_5	$\cdots$	2_5	0_5	1_5	1_5	0_5	0_5	2_5	3_5	4_5

Pour obtenir un codage homogène des différents éléments optimisés dans le contrôleur flou, Homaifar propose également une méthode de codage en base 5 des fonctions d'appartenance.

Afin de pouvoir coder également les règles inutiles, Lim propose une autre méthode de codage [LRG96]. Les conclusions des règles utiles sont codées par un chiffre en base 5, comme pour la méthode de Homaifar, mais seules les conclusions utiles apparaissent dans le codage final. Pour ce faire, chaque règle utile est codée par un couple de valeurs (r, s) dans lequel r repère la position de la règle dans le tableau contenant toutes les règles possibles, et s indique l'indice du terme linguistique correspondant à la conclusion.

Par exemple, les règles floues décrites par le tableau de la figure 3.14, sont codées par les couples :

$(2, 0_5)$	$(7, 1_5)$	$(8, 2_5)$	$(10, 0_5)$	$\cdots$	$(14, 4_5)$	$(16, 2_5)$	$(17, 3_5)$	$(22, 4_5)$

Le principal avantage de ce type de codage est qu'il permet de faire intervenir des règles inutiles tout en conservant un nombre total de règles constant. En effet, lorsqu'une règle inutile est codée par un chiffre au même titre qu'une autre règle, une opération

	NM	NP	EZ	PP	PM
NM	0	0	1	0	0
NP	0	0	2	3	0
EZ	1	2	3	4	5
PP	0	3	4	0	0
PM	0	0	5	0	0

FIG. 3.14 : Conclusions des règles floues

de mutation peut faire disparaître ou apparaître une règle floue durant l'évolution de la population.

Hoffmann [HP95] suggère d'utiliser l'algorithme génétique de Messy, qui a été décrit initialement par Goldberg [GKD89, GKD91], dans lequel une chaîne binaire telle que (1010) est traduite par des paires (r, s), r indiquant la position d'un bit et s sa valeur : $\{(1,1)(2,0)(3,1)(4,0)\}$.

Dans le codage proposé par Hoffmann, pour représenter une règle d'indices (i_1, i_2) dont la conclusion est le terme linguistique d'indice j, on code le triplet (i_1, i_2, j) en : $\{(1,i_1)(2,i_2)(3,j)\}$. Par exemple, les règles floues décrites par le tableau de la figure 3.15 sont codées en :

$\{(1,1_5)(2,1_5)(3,4_5)\}$	$\{(1,2_5)(2,1_5)(3,4_5)\}$	$\cdots$	$\{(1,3_5)(2,3_5)(3,1_5)\}$

	N(1)	EZ(2)	P(3)
N(1)	4	4	3
EZ(2)	1	2	3
P(3)	3	2	1

FIG. 3.15 : Conclusions des règles floues

3.3.2.3 Codage des règles floues par un chromosome en nombre réel

Très peu d'auteurs ont retenu le principe de codage des règles floues par un nombre réel. Le seul intérêt est de disposer d'un type de codage homogène pour tous les paramètres du contrôleur que l'on souhaite optimiser. Herrera et al. [HLV97] ont opté pour ce codage

afin de représenter dans un même chromosome les paramètres de la fuzzification et les règles floues.

Nous verrons dans la suite de ce mémoire qu'il est possible de regrouper les différents paramètres dans un même individu, caractérisé par trois chromosomes, sans qu'il soit nécessaire d'utiliser systématiquement le même codage.

3.4 Optimisation de la défuzzification par un Algorithme Génétique

Dans le contrôleur flou, la défuzzification est également une phase de traitement très importante puisqu'elle fournit la valeur non floue présentée sur la sortie. Depuis que Zadeh [Zad68] a, le premier, remarqué la limitation des procédures de défuzzification, de nombreuses méthodes ont été décrites en détails dans la littérature [KM78, KGN85, YF93a]. On peut retenir de ces différentes études quatre méthodes principales : COA (centre de région), COG (centre de gravité), MOM (moyenne des maxima) et MC (critère maximum).

Des études comparatives ont montré la supériorité des procédures COA et MOM par rapport à la méthode MC [KGN85, YF93a]. Le COA donne de meilleurs résultats que le MOM en régime permanent de fonctionnement, alors que ce dernier se révèle plus efficace dans les régimes transitoires. Filev et Yager [FY91] sont les premiers à avoir reconnu que l'optimisation de la défuzzification ne peut pas être réalisée sans avoir recours à une procédure d'apprentissage.

Ils ont proposé une méthode de défuzzification appelée BADD (BAsic Defuzzification Distributions) qui inclut une procédure d'apprentissage. Dans cette procédure, le paramètre réglant le fonctionnement de la méthode de défuzzification SLIDE (Semi Linear Defuzzification [YF93b]) est ajusté durant la phase d'apprentissage grâce à un filtrage de Kalman.

Plus récemment, Jiang et al. [JL96] ont également utilisé le filtrage de Kalman durant une phase d'apprentissage tout en utilisant une formulation plus générale pour l'opération de défuzzification. Dans un article récent, Song fait appel à une technique neuronale pour l'apprentissage des paramètres [SL96].

3.4.1 Les méthodes de défuzzification

Soit B un sous-ensemble flou d'un univers du discours discret $Y = \{y_1, \cdots, y_i, \cdots, y_n\}$, de fonction d'appartenance μ_B :

$$B = \sum_{i=1}^{n} \mu_B(y_i)/y_i \tag{3.20}$$

Nous notons $H(B)$ la hauteur de B, qui est la valeur maximale que prend la fonction d'appartenance sur l'univers du discours Y :

$$H(B) = max\{\mu_B(y_i)\}, \qquad i = 1, \cdots, n \tag{3.21}$$

Nous utiliserons également les trois ensembles M, S_α et I_α, constitués des indices de certains des éléments de Y. M est l'ensemble des indices des éléments de Y pour lesquels la fonction d'appartenance μ_B est maximale :

$$M = \{\, i \mid \mu_B(y_i) = H(B)\}, \qquad i = 1, \cdots, n \tag{3.22}$$

S_α est l'ensemble des indices des éléments de l'$\alpha-$coupe de B, défini par :

$$S_\alpha = \{\, i \mid \mu_B(y_i) \geq \alpha\}, \qquad i = 1, \cdots, n \tag{3.23}$$

et I_α est l'ensemble des indices des éléments du complémentaire de l'$\alpha-$coupe de B, donné par :

$$I_\alpha = \{\, i \mid \mu_B(y_i) < \alpha\}, \qquad i = 1, \cdots, n \ . \tag{3.24}$$

Les principales méthodes de défuzzification (COA, MOM, BADD [FY91], SLIDE, M-SLIDE, G et GD) sont décrites dans le tableau de la figure 3.16. Les méthodes G (Gaussienne) et GD (Différence de Gaussienne) ont été décrites par Jiang et Li dans [JL96].

La formulation la plus générale de l'opération de défuzzification dans le cas discret peut être exprimée par l'équation suivante :

$$y = \frac{\sum_{i=1}^{n} y_i \cdot \mu_B(y_i) \cdot T_i}{\sum_{i=1}^{n} \mu_B(y_i) \cdot T_i} \tag{3.25}$$

dans laquelle T_i est un coefficient de pondération appliqué à chaque terme.

La plupart des méthodes standard de défuzzification peuvent être exprimées grâce à cette équation, en choisissant les valeurs adéquates pour les coefficients T_i. Les coefficients

Méthode	Expression
COA	$\frac{\sum_{i=1}^{n} y_i \cdot \mu_B(y_i)}{\sum_{i=1}^{n} \mu_B(y_i)}$
MOM	$\frac{\sum_{i \in M} y_i \cdot \mu_B(y_i)}{\sum_{i \in M} \mu_B(y_i)}$
BADD	$\frac{\sum_{i=1}^{n} y_i \cdot \mu_B^{\gamma}(y_i)}{\sum_{i=1}^{n} \mu_B^{\gamma}(y_i)} \quad \gamma > 0$
SLIDE	$\frac{(1-\beta)\sum_{i \in I_\alpha} y_i \cdot \mu_B(y_i) + \sum_{i \in S_\alpha} y_i \cdot \mu_B(y_i)}{(1-\beta)\sum_{i \in I_\alpha} \mu_B(y_i) + \sum_{i \in S_\alpha} \mu_B(y_i)} \quad \beta > 0$
M-SLIDE	$\frac{(1-\beta)\sum_{i \notin M} y_i \cdot \mu_B(y_i) + \sum_{i \in M}(1+\beta \sum_{j \notin M} \frac{\mu_B(y_j)}{H(B)}) y_i \cdot \mu_B(y_i)}{(1-\beta)\sum_{i \notin M} \mu_B(y_i) + \sum_{i \in M}(1+\beta \sum_{j \notin M} \frac{\mu_B(y_j)}{H(B)}) \mu_B(y_i)}$
G	$\frac{\sum_{i=1}^{n} y_i \cdot \mu_B(y_i) e^{-\beta(\mu_B(y_i) - H(B))^2}}{\sum_{i=1}^{n} \mu_B(y_i) e^{-\beta(\mu_B(y_i) - H(B))^2}}$
GD	$\frac{\sum_{i=1}^{n} y_i \cdot \mu_B(y_i) e^{[\sum_{j=0} N\beta_j \cdot (\mu_B(y_i) - 0.5)^j]}}{\sum_{i=1}^{n} \mu_B(y_i) [\sum_{j=0} N\beta_j \cdot (\mu_B(y_i) - 0.5)^j]}$

FIG. 3.16 : Principales méthodes de défuzzification

T_i correspondant aux méthodes décrites dans le tableau de la figure 3.16 sont précisés dans le tableau 3.17.

Méthode	Coefficient T_i
COA	$T_i = 1, \quad i = 1, \cdots, n$
MOM	$T_i = \begin{cases} 0, & (i \notin M) \\ 1, & (i \in M) \end{cases}$
BADD	$T_i = \mu_B^{\gamma-1}(y_i), \quad i = 1, \cdots, n$
SLIDE	$T_i = \begin{cases} 1-\beta, & (i \in I_\alpha) \\ 1, & (i \in S_\alpha) \end{cases}$
M-SLIDE	$T_i = \begin{cases} 1-\beta, & (i \notin M) \\ 1+\beta \sum_{j \notin M} \frac{\mu_B(y_j)}{H(B)}, & (i \in M) \end{cases}$
G	$T_i = e^{-\beta(\mu_B(y_i) - H(B))^2}, \quad i = 1, \cdots, n$
GD	$T_i = \sum_{j=0} N\beta_j \cdot (\mu_B(y_i) - 0.5)^j, \quad i = 1, \cdots, n$

FIG. 3.17 : Coefficients pour chaque méthode de défuzzification

Yager et Filev ont montré dans l'article [YF93b] que la méthode M-SLIDE est une combinaison linéaire des méthodes MOM et COA, définie par l'équation :

$$y_{MSLIDE} = \beta(y_{MOM} - y_{COA}) + y_{COA} \tag{3.26}$$

Les performances des procédures de défuzzification BADD, SLIDE, M-SLIDE, G et

GD peuvent être réglées en modifiant la valeur d'un paramètre. Dans les articles de Yager et Jiang, ce paramètre est ajusté grâce à une phase d'apprentissage [YF93b, JL96]. Ces auteurs utilisent un filtrage de Kalman qui exploite la différence entre la valeur obtenue par la méthode de défuzzification et une valeur idéale.

3.4.2 Optimisation de la défuzzification par AG

Lorsque les valeurs idéales des sorties sont connues a priori pour un certain nombre de sous-ensembles de test, l'opération de défuzzification peut être optimisée par un algorithme génétique. Ainsi, les paramètres de la méthode de défuzzification (paramètres α et β pour la méthode SLIDE, paramètre β pour la méthode M-SLIDE, ...) sont ajustés de telle sorte que pour chaque sous-ensemble de test, la valeur obtenue en sortie soit la plus proche possible de la valeur idéale fixée initialement.

La fonction de coût de l'algorithme génétique est construite de telle sorte qu'elle présente un minimum global lorsque les sorties calculées $\hat{s}_i$ sont égales aux sorties idéales s_i pour les n sous-ensembles de test. On peut par exemple utiliser la somme des différences au carré entre les deux séries de données s_i et $\hat{s}_i$ définie par :

$$f_{eval} = \sqrt{\sum_{i=1}^{n} [\hat{s}_i - s_i]^2} \ . \tag{3.27}$$

Pour valider cette méthode, nous avons utilisé l'exemple décrit initialement par Yager dans [YF93b] qui a également été utilisé par Jiang par la suite [JL96]. Pour obtenir la meilleure précision possible sur les paramètres estimés, nous avons choisi de les coder par un nombre réel dans le chromosome exploité par l'algorithme génétique. Sur ces données, à l'issue de la phase d'optimisation, nous avons obtenu des résultats identiques à ceux de Yager et Jiang.

Comme l'ont souligné d'autres auteurs [Mab93, SL96], les valeurs idéales dont souvent inconnues dans la pratique, ce qui élimine toute possibilité d'apprentissage par un filtrage de Kalman basé sur une comparaison. Lorsque l'optimisation est réalisée par un algorithme génétique, la fonction de coût peut être construite en utilisant des critères de performances plus généraux, qui ne font pas forcément intervenir de valeurs idéales connues a priori.

3.5 Optimisation globale d'un contrôleur flou par AG mixte

Dans les sections précédentes de ce chapitre, nous avons présenté les méthodes de codage de paramètres permettant l'optimisation de chaque partie d'un contrôleur flou. Nous avons mis en évidence les résultats suivants :

- L'optimisation des fonctions d'appartenance consiste à trouver les positions relatives de chaque terme linguistique au sein d'une variable linguistique. La précision du codage n'est pas un critère prépondérant, mais il faut prévoir un moyen d'éliminer éventuellement certains termes linguistiques inutiles. Un codage des paramètres par une chaîne binaire s'avère bien adapté.
- L'optimisation des règles floues consiste à sélectionner parmi toutes les règles possibles celles qui s'avèrent pertinentes. Différentes méthodes de représentation sont utilisables, toutefois le codage des règles en base n est la méthode la plus efficace.
- Afin d'obtenir une version optimale de l'opération de défuzzification, on doit ajuster un nombre relativement limité de paramètres. Par contre, pour obtenir un résultat satisfaisant, il faut coder ces paramètres avec une précision importante.

Pour l'instant, les auteurs qui ont proposé une optimisation simultanée de plusieurs éléments d'un contrôleur flou ont sélectionné une seule méthode de codage pour regrouper tous les paramètres dans un seul chromosome. Par exemple, Homaifar représente les paramètres des fonctions d'appartenance en utilisant le codage en base 5 qu'il introduit pour représenter les règles floues [HM95a].

Nous proposons une méthode de codage mixte, qui permet de représenter au mieux les différents paramètres caractérisant une solution possible. Un individu est caractérisé par trois chromosomes : un chromosome codé en binaire, un chromosome codé en base n et un chromosome codé en nombres réels. Ce codage mixte est représenté sur la figure 3.18.

Nous décrivons les opérations de croisement et de mutation permettant d'utiliser ce principe de codage dans un algorithme génétique. Chaque chromosome est affecté par des opérateurs génétiques spécifiques, mais l'algorithme génétique utilise une seule fonction d'évaluation pour mesurer la qualité d'un individu caractérisé par ces trois chromosomes.

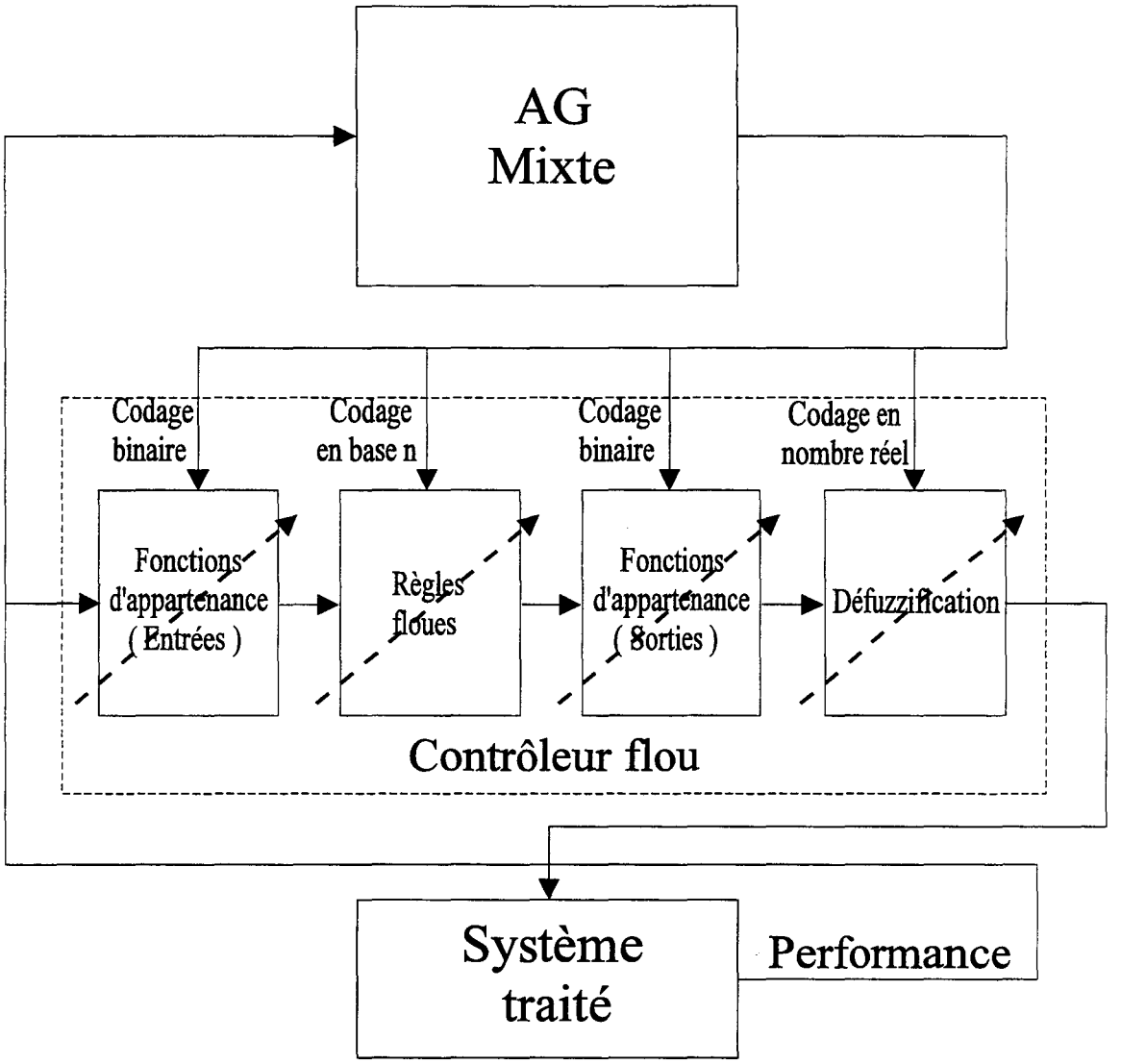


FIG. 3.18 : Codage mixte des paramètres

3.5.1 Codage des fonctions d'appartenance

Nous choisissons une méthode de codage des fonctions d'appartenance exploitant les différences entre les abscisses de deux points caractéristiques successifs de la fonction. La variable linguistique codée est une partition floue d'un intervalle borné $[a, b]$, dont les termes linguistiques ont des fonctions d'appartenance triangulaires, à l'exception des deux fonctions extrêmes qui sont trapézoïdales. La figure 3.19 présente une telle variable linguistique.

En général, le nombre de termes linguistiques nécessaires pour qualifier une entrée n'est pas connu a priori. On choisit alors un nombre initial de termes relativement important, en

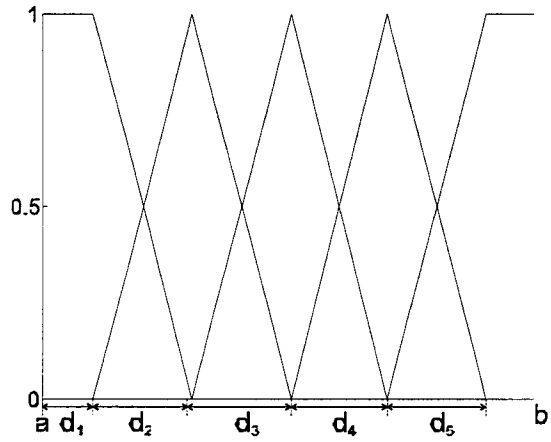


FIG. 3.19 : Partition floue retenue

supposant que le processus d'optimisation éliminera les termes superflus. Dans le codage, il faut prévoir un moyen d'éliminer les termes inutiles, ce que nous faisons en ajoutant un bit de validité dans la sous-chaîne binaire codant une fonction d'appartenance.

Par exemple, pour coder la variable linguistique représentée sur la figure 3.19, nous utilisons le chromosome binaire suivant :

b_1	$c_{m,[a,b]}(d_1)$	b_2	$c_{m,[a,b]}(d_2)$	b_3	$c_{m,[a,b]}(d_3)$	b_4	$c_{m,[a,b]}(d_4)$	b_5	$c_{m,[a,b]}(d_5)$

où b_i est le bit de validité du terme linguistique numéro i et $c_{m,[a,b]}(d)$ désigne le codage sur m bits d'une valeur d de l'intervalle $[a, b]$.

Les valeurs codées doivent respecter une seule inégalité, qui permet d'éviter le débordement des termes linguistiques en dehors de l'intervalle $[a, b]$:

$$d_1 + \cdots + d_5 \leq (b - a) \tag{3.28}$$

Si un bit de validité prend une valeur nulle durant le processus d'optimisation, le terme linguistique correspondant est éliminé dans la définition de la variable linguistique. Les termes linguistiques voisins sont étendus pour maintenir une continuité de la partition lorsqu'un terme linguistique est éliminé. Par exemple, la figure 3.20 représente la variable linguistique définie précédemment lorsque les termes linguistiques d'indices 2 et 5 sont inutiles ($b_2 = b_5 = 0$).

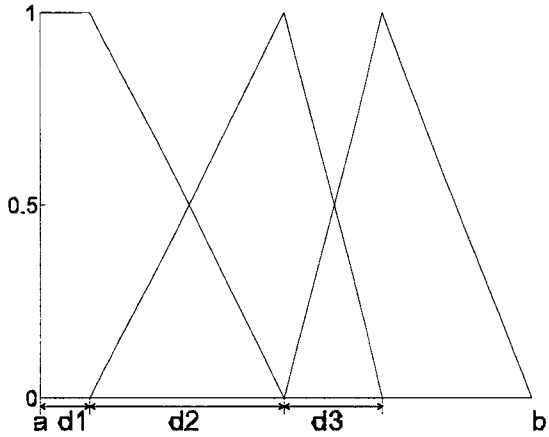


FIG. 3.20 : Variable linguistique avec deux termes inutiles

3.5.2 Codage des règles floues

Nous utilisons un chromosome codé en base n pour représenter les règles floues d'un contrôleur disposant de N_e entrées, qualifiées chacune par N_{te} termes linguistiques, et de N_s sorties, qualifiées chacune par N_{ts} termes. Une règle définie par :

$$\begin{aligned} \text{Règle } i_1, i_2, \cdots, i_{N_e} : \quad & \text{Si } (\mathcal{X}_1 \text{ est } A_{1i_1}) \text{ et } \cdots \text{ et } (\mathcal{X}_{N_e} \text{ est } A_{N_e i_{N_e}}) \quad ; \\ & \text{alors } (\mathcal{Y}_1 \text{ est } B_{1j_1}) ; \cdots ; (\mathcal{Y}_{N_s} \text{ est } B_{N_s j_{N_s}}) \end{aligned}$$

est codée par une succession de sous-chaînes en base $N_{ts}+1$, chaque sous-chaîne B_j permettant le codage des conclusions de règles pour la sortie j :

B_1	$\cdots$	B_j	$\cdots$	B_{N_s}

Chaque sous-chaîne B_j est constituée de ${N_{te}}^{N_e}$ chiffres en base $N_{ts}+1$ qui codent les indices des termes linguistiques qualifiant les conclusions de toutes les règles pour la sortie j :

$R^j_{1\cdots 1}$	$\cdots$	$R^j_{N_{te}\cdots N_{te}}$

où les $R^j_{i_1\cdots i_{N_e}}$, pour $(i_1 = 1, \cdots, N_{te}, \ \cdots, \ i_k = 1, \cdots, N_{te})$, sont des chiffres exprimés en base $N_{ts}+1$.

3.5.3 Codage des paramètres de la défuzzification

Comme nous l'avons vu précédemment, les paramètres ajustant les processus de défuzzification doivent être codés par des nombres réels, pour autoriser une précision élevée.

Selon la méthode de défuzzification choisie, on est amené à coder plus ou moins de paramètres. Par exemple, pour la méthode M-SLIDE, un seul paramètre suffit, alors que pour la méthode SLIDE, deux paramètres sont nécessaires.

Dans les exemples présentés par la suite, nous utilisons toujours la méthode M-SLIDE, qui tout en étant relativement performante, ne nécessite l'ajustement que d'un seul paramètre. Ainsi, le seul paramètre β est codé dans le chromosome en tant que nombre réel :

Pour d'autres applications, dans lesquelles l'étape de fuzzification serait plus critique, on peut tout à fait coder un nombre plus important de paramètres dans un chromosome constitué de nombres réels. Par exemple, si on utilise la méthode de défuzzification généralisée de Jiang et Li, on peut coder un paramètre β_k pour chaque élément de l'univers du discours discret caractérisant une sortie [JL96].

3.5.4 Opérations génétiques

Puisque la méthode de codage des paramètres que nous avons retenue fait intervenir trois chromosomes pour un seul individu, il convient de spécifier les opérateurs génétiques permettant d'assurer l'évolution de la population.

Comme le chromosome d'optimisation de contrôleur flou comprend trois chaînes différentes, une chaîne binaire, une chaîne en base n et une chaîne en nombre réel, les opérations génétiques ont les trois types.

- Crossover et mutation binaire, comme les relations entre la fonction d'évaluation et les paramètres qu'on veut optimiser, on préfère le crossover et la mutation standard.
- Crossover et mutation en base n sont décrits dans 2.4.2. Nous utilisons seulement le crossover et la mutation standard.
- Crossover et mutation en nombre réel sont présentés dans 2.4.1. le crossover simple et la mutation uniforme sont choisis.

3.5.5 Fonction d'évaluation

La fonction d'évaluation permettant d'optimiser la structure d'un contrôleur flou dépend fortement de l'application dans laquelle intervient ce contrôleur. Il n'est donc pas possible de donner une formulation générale de cette fonction qui puisse s'adapter à toutes les particularités d'un problème. Cependant, on peut préciser certaines règles permettant de choisir au mieux une fonction d'évaluation.

Notons sous forme vectorielle $\vec{E} = (E_1, E_2, \cdots, E_{N_e})^T$ les entrées d'un contrôleur flou, et $\vec{S} = (S_1, S_2, \cdots, S_{N_s})^T$ ses sorties. Si chaque entrée E_i prend ses valeurs dans un intervalle $[a_i, b_i]$, le vecteur $\vec{E}$ appartient à un hypercube H_e de R^{N_e}. Il en va de même pour les sorties, le vecteur $\vec{S}$ prenant ses valeurs dans un hypercube H_s de R^{N_s}.

Le contrôleur flou définit une fonction injective f (mapping), de H_e vers H_s, qui associe à tout vecteur d'entrée $\vec{E}$ un vecteur de sortie $\vec{S}$. Cette fonction f dépend également des N_p paramètres de fonctionnement du contrôleur flou (limites des fonctions d'appartenance, règles floues, paramètres de la défuzzification), symbolisés par un vecteur $\vec{P}$ à N_p composantes. Ainsi, la fonction f est définie par :

$$\begin{aligned} f \; : \; H_e \times R^{N_p} &\rightarrow H_s \\ (\vec{E}, \vec{P}) &\rightarrow \vec{S} \end{aligned} \tag{3.29}$$

Pour évaluer la performance du contrôleur flou pour un certain jeu de paramètres définis par un vecteur $\vec{P}$, on place sur les entrées une succession de vecteurs $\vec{E}_k$ et on enregistre les vecteurs de sortie $\vec{S}_k$ correspondants. La fonction d'évaluation est construite en utilisant ces deux séries de valeurs.

Par exemple, si on connaît pour chaque vecteur d'entrée $\vec{E}_k$ la sortie souhaitée $\vec{S}'_k$, la fonction d'évaluation peut être calculée par la somme des normes des différences entre les sorties obtenues et les sorties souhaitées :

$$Eval = \sum_{k=1}^{l} ||\vec{S}'_k - f(\vec{E}_k, \vec{P})|| \quad , \tag{3.30}$$

où $||x||$ désigne une norme définie sur R^{N_s}.

On utilise également la somme des carrés des normes des différences en tant que fonction d'évaluation, ce qui s'exprime par :

$$Eval = \sqrt{\sum_{k=1}^{l} ||\vec{S}'_k - f(\vec{E}_k, \vec{P})||^2} \quad . \tag{3.31}$$

Lorsque les valeurs idéales des sorties du contrôleur flou pour un vecteur d'entrée donné ne sont pas connues a priori, la fonction d'évaluation peut être construite à partir de critères établis sur les sorties du système piloté par le contrôleur flou.

3.6 Conclusion

Dans ce chapitre, nous avons présenté une méthode de représentation des paramètres de fonctionnement d'un contrôleur, permettant d'optimiser leurs valeurs par un algorithme génétique. Cette représentation fait intervenir trois types de codage, à savoir un codage binaire, un codage en base n et un codage en nombres réels.

Pour démontrer l'efficacité de la méthode proposée, nous décrivons dans les chapitres suivants de cette thèse deux applications sur lesquelles nous l'avons utilisée.

Chapitre 4

Réglage dynamique d'un PID par un contrôleur flou

Dans le chapitre précédent, nous avons présenté une méthode de synthèse d'un contrôleur flou basée sur une optimisation par un algorithme génétique. Nous décrivons maintenant une première application de ce procédé d'optimisation dans un problème standard de réglage : l'ajustement dynamique des coefficients d'un contrôleur de type PID.

4.1 Introduction

Le régulateur PID (Proportionnel, Intégral, Dérivé) est encore largement utilisé dans le milieu industriel malgré l'émergence d'autres méthodes de régulation. Ce régulateur linéaire est basé sur une structure très simple dont le fonctionnement ne dépend que de trois coefficients, qui sont les gains appliqués sur les signaux proportionnel (K_p), intégral (K_i) et dérivé (K_d).

De nombreuses méthodes de réglage statique d'un PID ont été décrites dans la littérature [ÅH89, HÅH91, RL95, LR95], la plus connue étant certainement la méthode de Ziegler-Nichols [ÅH88]. Dans chacune de ces méthodes, les trois gains sont fixés en suivant une procédure de réglage qui garantit un fonctionnement optimal selon un ou plusieurs critères. Dans tous les cas, la fonction de transfert du régulateur PID reste linéaire.

Plus récemment, des auteurs ont proposé des méthodes de réglage dynamique des coefficients d'un PID [HTXW93, ZTI93, Miz95, KGT95, BdM95, Lim95]. Les trois gains sont

continuellement ajustés par un régulateur annexe qui exploite pour ce faire des mesures réalisées sur le processus commandé. Dans ce cas, la fonction de transfert globale (PID + contrôleur annexe) n'est plus linéaire. Dans [ZTI93], Zhao utilise un régulateur annexe de type contrôleur flou à deux entrées. Il montre que le régulateur non-linéaire équivalent est bien plus performant qu'un PID classique réglé par la méthode de Ziegler-Nichols.

Nous avons appliqué la méthode de synthèse d'un contrôleur flou par algorithme génétique afin de spécifier la structure d'un régulateur annexe similaire à celui proposé par Zhao. Les solutions du problème d'optimisation sont évaluées par le biais d'une fonction construite à l'aide de critères de performance mesurés sur la réponse à l'échelon du système contrôlé par le PID. A la fin de ce chapitre, nous comparons les réponses à l'échelon de systèmes contrôlés par trois PID : un PID fixe réglé par Ziegler-Nichols, le PID flou de Zhao et celui dont les paramètres sont réglés dynamiquement par un contrôleur flou synthétisé par notre méthode.

4.2 Le régulateur PID

4.2.1 Structure d'un régulateur PID

Le régulateur PID est un système linéaire du second ordre à une entrée et une sortie, dont la fonction de transfert dans le domaine de Laplace est donnée par :

$$G(p) = K_p + K_i/p + K_d \cdot p \ , \tag{4.1}$$

dans laquelle K_p, K_i et K_d sont appelés respectivement gain proportionnel, intégral et dérivé. La commande $c(t)$ du système est ainsi constituée par la somme pondérée de trois termes dépendant de l'erreur $\epsilon(p)$ entre la sortie $s(p)$ du processus et la consigne $e(p)$, de la dérivée et de l'intégrale de cette erreur.

On utilise également une autre expression de la fonction de transfert :

$$G(p) = K_p \cdot (1 + 1/(T_i \cdot p) + T_d \cdot p) \ , \tag{4.2}$$

où $T_i = K_p/K_i$ est la constante de temps intégrale et $T_d = K_d/K_p$ la constante de temps dérivée.

Dans le domaine temporel, l'évolution de la commande $c(t)$ fournie par le PID est déterminée en fonction de celle de l'erreur $\epsilon(t)$ en utilisant l'expression suivante :

$$c(t) = K_p \cdot \epsilon(t) + K_i \cdot \int \epsilon(t)dt + K_d \cdot \frac{d\epsilon(t)}{dt} \quad . \tag{4.3}$$

Pour contrôler un processus industriel, on utilise en général une version numérique du régulateur PID qui reçoit en entrée la séquence des échantillons $\epsilon(k)$ correspondant à l'erreur et fournit en sortie la commande échantillonnée $c(k)$ calculée par :

$$c(k) = K_p \cdot \epsilon(k) + K_i \cdot T_s \cdot \sum_{i=1}^{n} \epsilon(i) + \frac{K_d}{T_e} \Delta\epsilon(k) \quad , \tag{4.4}$$

où T_e désigne la période d'échantillonnage, $\sum_{i=1}^{n} \epsilon(i)$ est l'approximation de l'intégrale de l'erreur et $\Delta\epsilon(k) = \epsilon(k) - \epsilon(k-1)$ l'approximation de sa dérivée.

4.2.2 Réglage statique des gains d'un PID

Le problème du réglage des gains pondérant les différents termes du régulateur PID est un problème assez complexe abondamment traité dans la littérature.

La méthode la plus utilisée est celle de Ziegler-Nichols, car elle ne nécessite pas de connaître un modèle du processus commandé. Durant la phase de calibration, on remplace le PID par un simple amplificateur à gain réglable, comme le montre la figure 4.1. On augmente progressivement la valeur du gain afin d'amener le système en régime d'auto-oscillation (phénomène de pompage). La plus petite valeur du gain permettant d'obtenir un régime oscillant est notée K_u. On mesure alors la période des oscillations, notée T_u.

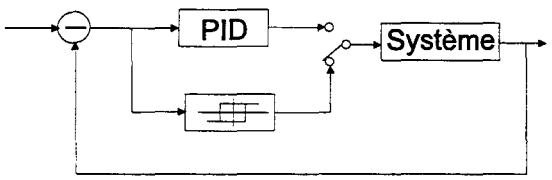


FIG. 4.1 : Réglage par Ziegler-Nichols des gains d'un PID

Les valeurs du gain proportionnel K_p et des constantes de temps intégrale T_i et dérivée T_p sont calculées par les formules suivantes :

$$K_p = 0,6K_u, \qquad T_i = 0,5T_u, \qquad T_d = 0,125T_u, \tag{4.5}$$

dans laquelle K_u est le gain et T_u la constante de temps qui ont été estimés lors de la phase de calibration.

4.2.3 Indices de performance d'un PID

Afin de définir la qualité de la régulation, on se base en général sur l'analyse de la réponse indicielle de l'ensemble régulateur PID plus système. La figure 4.2 montre l'allure générale d'une réponse à l'échelon :

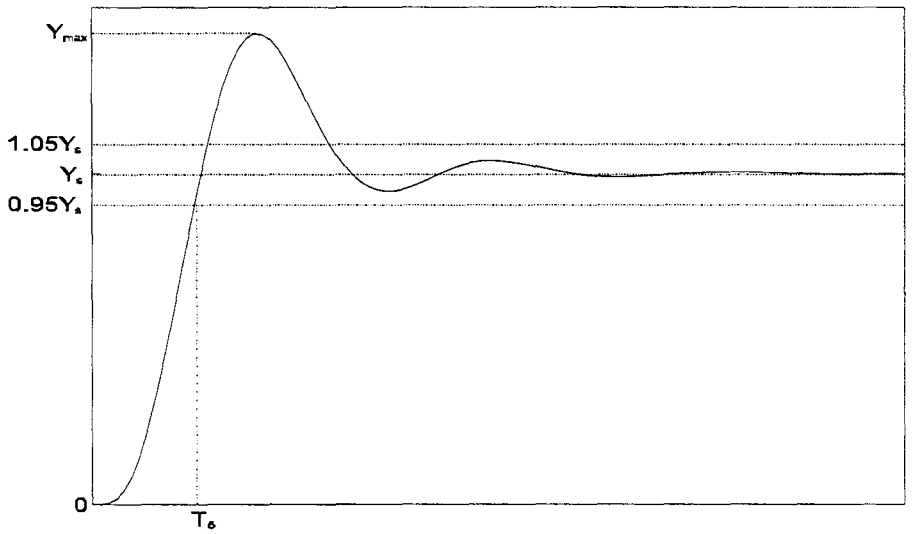


FIG. 4.2 : Réponse à l'échelon d'un ensemble PID plus système

Différents indices de performance peuvent être évalués à partir de cette réponse temporelle. De façon générale, on cherche à quantifier la différence entre la réponse réelle du système asservi et une réponse idéale qui serait un échelon. Les indices couramment utilisés sont définis de la façon suivante :

- Y_{os}, *pourcentage de dépassement.* Avant de se stabiliser, la sortie du système passe par un régime transitoire oscillant de part et d'autre de la valeur finale. On définit le pourcentage de dépassement par :

$$Y_{os} = \frac{s_{max} - s(\infty)}{s(\infty)} \times 100\% \ , \tag{4.6}$$

où $s(\infty)$ est la valeur finale de la sortie et s_{max} est sa valeur maximale, qui est atteinte en général lors de la première oscillation.

- T_5, *temps de stabilisation à 5%.* Le temps de stabilisation à 5% de la valeur finale est une estimation de la durée du régime transitoire. On considère que le système a atteint son régime permanent lorsque sa sortie est égale à la valeur finale à 5% près.

Cette durée est définie par :

$$T_5 = \min\{t_0 > 0 \mid \forall t \geq t_0,\ s(\infty) \cdot (1 - 5\%) \leq s(t) \leq s(\infty) \cdot (1 + 5\%)\} \ . \quad (4.7)$$

- *Intégrales faisant intervenir l'erreur.* Pour évaluer la différence existant entre la réponse réelle et une réponse idéale de type échelon, on peut calculer l'intégrale d'un terme positif faisant intervenir l'erreur. Un indice calculé de cette façon prend une valeur d'autant plus élevée que la réponse réelle est éloignée de la réponse idéale. En pratique, l'intégrale est calculée sur un intervalle $[0, T]$ suffisamment étendu pour contenir tout le régime transitoire. L'intégrale de la valeur absolue de l'erreur $\epsilon(t)$ donnée par :

$$IAE = \int_0^T |\epsilon(t)| \cdot dt \ , \quad (4.8)$$

On utilise également l'intégrale de l'erreur quadratique, définie par :

$$ISE = \int_0^T \epsilon^2(t) \cdot dt \ . \quad (4.9)$$

Pour pénaliser les systèmes dont le régime transitoire dure trop longtemps, on utilise également l'intégrale du produit de l'erreur par le temps, donnée par :

$$ITAE = \int_0^T t \cdot |\epsilon(t)| \cdot dt \ . \quad (4.10)$$

Dans [Dor74], on peut trouver une liste plus complète de mesures de performances d'un système asservi. Dans notre étude, nous nous sommes limités aux quatre premiers indices de performance définis ci-dessus (Y_{os}, T_5, IAE et ISE).

4.3 Réglage dynamique des gains d'un PID

Récemment, plusieurs auteurs ont décrit des méthodes de réglage dynamique des gains d'un PID par un contrôleur flou [HTXW93, ZTI93, Miz95, KGT95, BdM95, Lim95]. On obtient ainsi un PID qualifié d'adaptatif, puisque ses paramètres de fonctionnement dépendent de l'état du système. Le PID adaptatif réalisé selon ce principe n'est plus un régulateur linéaire.

Dans la plupart de ces études, le contrôleur flou utilisé pour piloter le PID est défini par les auteurs à partir d'une série d'expériences. Pour tenter d'améliorer le réglage dynamique

des gains, Wang et al. [WK92] utilisent un contrôleur flou dans lequel les règles floues ont été optimisées par un algorithme génétique standard. Alander et al. [AMT97] utilisent également un contrôleur flou optimisé par un algorithme génétique à partir de critères définis dans le domaine fréquentiel.

He propose une structure de PID adaptatif dans lequel le réglage dynamique est superposé à un réglage statique réalisé par la méthode de Ziegler-Nichols [HTXW93]. Les trois gains du PID sont donnés par les formules :

$$K_p = 1,2 \cdot \alpha \cdot K_u, \qquad T_i = \frac{0,75}{1+\alpha} \cdot T_u, \qquad T_d = 0,25 \cdot T_i, \tag{4.11}$$

dans lesquelles α est un coefficient réglé par le contrôleur flou. On constate aisément que les formules précédentes correspondent au réglage de Ziegler-Nichols lorsque α vaut 1/2. La structure du PID adaptatif proposé par He est représentée sur la figure 4.3.

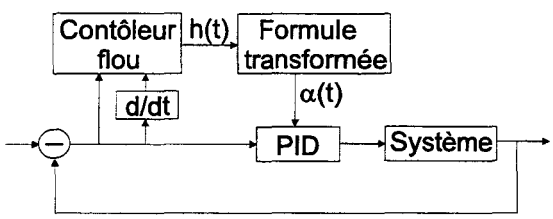


FIG. 4.3 : PID adaptatif proposé par He

Le contrôleur flou utilisé par He est défini à partir d'une série d'expériences, et non par un procédé d'optimisation. Les entrées sont fuzzifiées en utilisant sept termes linguistiques dont les fonctions d'appartenance sont des Gaussiennes, les règles floues sont fixes, et la défuzzification est de type COA.

4.3.1 PID adaptatif proposé par Zhao

Dans [ZTI93], Zhao a également décrit un PID adaptatif dans lequel un contrôleur flou ajuste dynamiquement les trois gains. La structure décrite par Zhao est représentée sur la figure 4.4.

Les gains K_p, K_i et K_d du PID sont calculés à partir de trois valeurs intermédiaires fournies par le contrôleur flou, en utilisant les expressions suivantes :

$$K_p = (K_{p,max} - K_{p,min}) \cdot K'_p + K_{p,min} \tag{4.12}$$

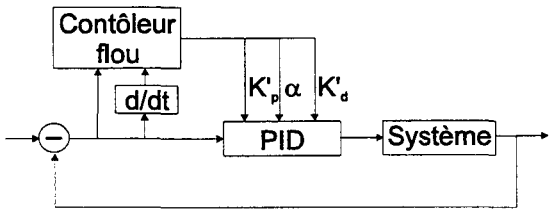


FIG. 4.4 : PID adaptatif proposé par Zhao

$$K_d = (K_{d,max} - K_{d,min}) \cdot K'_d + K_{d,min} \tag{4.13}$$

$$K_i = (K_p^2/(\alpha \cdot K_d)) \tag{4.14}$$

où K'_p et K'_d prennent leurs valeurs dans l'intervalle $[0, 1]$ et α permet de régler la constante de temps intégrale en fonction de la constante de temps dérivée. Les valeurs extrêmes des gains sont définies à partir des paramètres K_u et T_u des formules de Ziegler-Nichols, par les expressions :

$$K_{p,min} = 0,32 \cdot K_u, \qquad K_{p,max} = 0,6 \cdot K_u, \tag{4.15}$$

$$K_{d,min} = 0,08 \cdot K_u \cdot T_u, \qquad K_{d,max} = 0,15 \cdot K_u \cdot T_u \ . \tag{4.16}$$

Zhao utilise une version échantillonnée du régulateur PID, dont l'entrée est alimentée par une série d'échantillons $\epsilon(k)$. Le contrôleur flou utilise également les échantillons $\epsilon(k)$ ainsi que l'estimation de leur dérivée temporelle donnée par la différence $\Delta\epsilon(k) = \epsilon(k) - \epsilon(k-1)$ entre deux échantillons successifs. Les deux entrées du contrôleur flou sont limitées à l'intervalle $[-1, 1]$ puis qualifiées par deux variables linguistiques identiques comportant sept termes. Les fonctions d'appartenance des sept termes linguistiques $(NG, NM, NP, EZ, PP, PM, PG)$ sont représentées sur figure 4.5.

Les sorties du contrôleur flou sont qualifiées par des variables linguistiques dont les fonctions d'appartenance sont particulières. Pour les sorties K'_p et K'_d, Zhao utilise uniquement deux termes linguistiques P (pour petit) et G (pour grand), dont les fonctions d'appartenance sont logarithmiques et définies par :

$$\mu_P(x) = \min(1, -\frac{1}{4}\log x) \tag{4.17}$$

$$\mu_G(x) = \min(1, -\frac{1}{4}\log(1-x)) \tag{4.18}$$

Ces fonctions d'appartenance sont représentées sur la figure 4.6(a). La sortie α est qualifiée par quatre termes linguistiques (P, PM, M, G), qui définissent des singletons, comme le montre la figure 4.6(b).

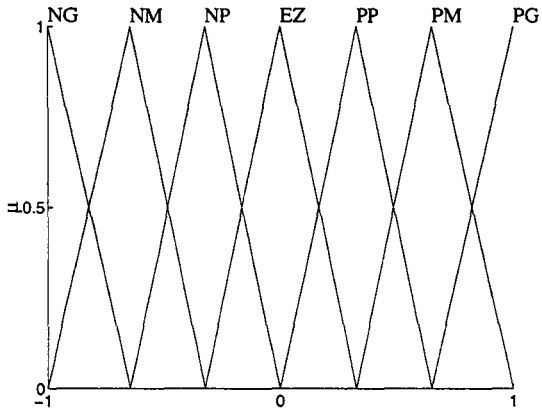
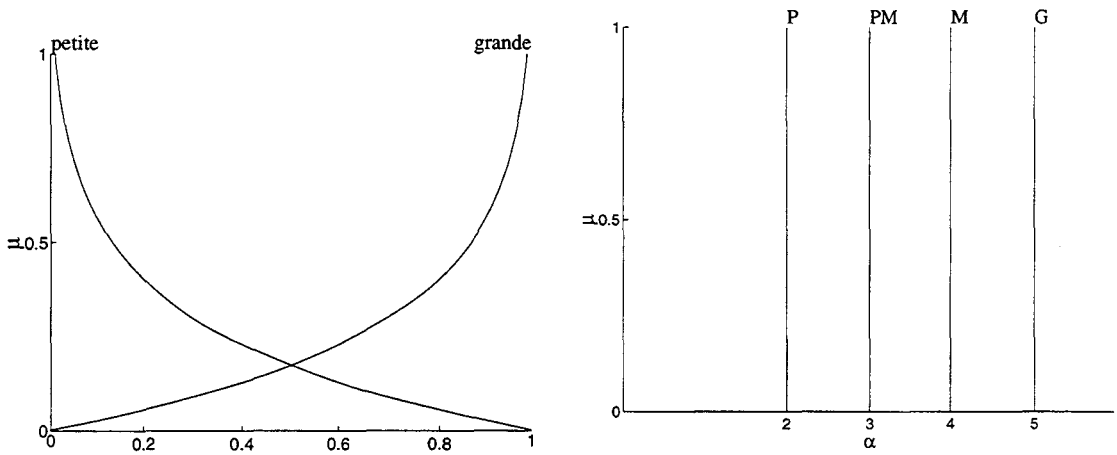


FIG. 4.5 : Fonctions d'appartenance des termes linguistiques qualifiant les entrées



(a) Sorties K'_p et K'_d (b) Sortie α

FIG. 4.6 : Fonctions d'appartenance pour les sorties

Les 49 règles floues utilisées par Zhao comportent toutes deux prémisses et trois conclusions, et sont exprimées sous la forme :

Règle i : Si ($\epsilon(k)$ est A_i) et ($\Delta\epsilon(k)$ est B_i) alors (K'_p est C_i), (K'_d est D_i), ($\alpha = \alpha_i$)
$i = 1, \cdots, 49$

Ces règles sont représentées sous forme de tableaux dans la figure 4.7. Les sous-figures (a) et (b) montrent les conclusions des règles floues pour K'_p et K'_d qui sont qualifiées par P et G, et la sous-figure (c) présente les conclusions des règles pour la sortie α, qui peut prendre quatre valeurs.

Zhao utilise la méthode d'inférence de Larsen, et la méthode COA pour la défuzzification de K'_p et K'_d. La sortie α ne doit pas être défuzzifiée puisque ses termes linguistiques sont des ensembles non flous.

		$\Delta\epsilon(k)$						
	K'_p	NG	NM	NP	EZ	PP	PM	PG
$\epsilon(k)$	NG	G	G	G	G	G	G	G
	NM	P	G	G	G	G	G	P
	NP	P	P	G	G	G	P	P
	EZ	P	P	P	G	P	P	P
	PP	P	P	G	G	G	P	P
	PM	P	G	G	G	G	G	P
	PG	G	G	G	G	G	G	G

(a) Conclusions des règles floues pour K'_p

		$\Delta\epsilon(k)$						
	K'_d	NG	NM	NP	EZ	PP	PM	PG
$\epsilon(k)$	NG	P	P	P	P	P	P	P
	NM	G	G	P	P	P	G	G
	NP	G	G	G	P	G	G	G
	EZ	G	G	G	G	G	G	G
	PP	G	G	G	P	G	G	G
	PM	G	G	P	P	P	G	G
	PG	P	P	P	P	P	P	P

(b) Conclusions des règles floues pour K'_d

		$\Delta\epsilon(k)$						
	α	NG	NM	NP	EZ	PP	PM	PG
$\epsilon(k)$	NG	2	2	2	2	2	2	2
	NM	3	3	2	2	2	3	3
	NP	4	3	3	2	3	3	4
	EZ	5	4	3	3	3	4	5
	PP	4	3	3	2	3	3	4
	PM	3	3	2	2	2	3	3
	PG	2	2	2	2	2	2	2

(c) Conclusions des règles floues pour α

FIG. 4.7 : Règles floues utilisées dans la méthode de Zhao

Dans le PID adaptatif décrit précédemment, les règles floues intervenant dans le contrôleur ont été entièrement définies en utilisant des connaissances a priori sur le fonctionnement d'un PID. On sait, par exemple, qu'au début du régime transitoire le contrôleur doit réagir au plus vite à la modification de l'entrée. On doit donc disposer de gains proportionnel et intégral important, mais d'un gain dérivé assez faible. Cette connaissance est formalisée par la règle floue :

Règle 1 : Si ($\epsilon(k)$ est PG) et ($\Delta\epsilon(k)$ est PG) alors (K'_p est G), (K'_d est P), ($\alpha = 2$)

L'inconvénient de cette méthode de synthèse, qui repose uniquement sur une expertise humaine, est que le PID obtenu n'est pas conçu spécifiquement pour piloter au mieux un système donné. Dans la section suivante, nous montrons comment il est possible d'optimiser le contrôleur flou réglant les paramètres du PID afin qu'il soit aussi bien adapté que possible aux caractéristiques du système commandé.

4.4 Synthèse du contrôleur flou par algorithme génétique

Afin d'optimiser la structure du contrôleur flou utilisé dans le PID adaptatif de Zhao, nous utilisons la méthode de synthèse décrite dans le chapitre 3. Nous décrivons tout d'abord le codage utilisé pour représenter les différents paramètres du contrôleur, puis nous présentons le résultat de la synthèse pour l'un des trois exemples de système utilisés par Zhao dans [ZTI93].

4.4.1 Codage des paramètres du contrôleur flou

Nous avons utilisé la méthode de codage mixte des paramètres présentée dans le chapitre précédent. Une solution possible du problème d'optimisation est ainsi représentée par un individu caractérisé par trois chromosomes, codés respectivement en binaire, en base n et par des nombres réels.

4.4.1.1 Codage des fonctions d'appartenance des entrées

Dans le contrôleur flou proposé par Zhao, les entrées $\epsilon(k)$ et $\Delta\epsilon(k)$ prennent leurs valeurs dans l'univers du discours $[-1, 1]$. Zhao a défini les termes linguistiques et les règles floues de façon à garantir la symétrie du fonctionnement quel que soit le signe des entrées. Pour simplifier la structure du contrôleur flou tout en conservant un fonctionnement symétrique, nous utilisons comme entrées les valeurs absolues des deux termes $\epsilon(k)$ et $\Delta\epsilon(k)$.

Notre contrôleur flou dispose ainsi de deux entrées $|\epsilon(k)|$ et $|\Delta\epsilon(k)|$. Ces entrées sont qualifiées par deux variables linguistiques comprenant au maximum cinq termes définis sur un univers du discours $[0, 1]$.

Les termes linguistiques décrivant les entrées sont représentés par des sous-chaînes binaires dans lesquelles nous codons les différences entre les maxima des fonctions d'appartenance de deux termes successifs. Chaque différence d_i est codée sur 8 bits, ce qui permet d'obtenir une précision de $7,825 \times 10^{-3}$. Nous ajoutons un bit pour chaque terme, ce qui permet d'éliminer éventuellement des termes inutiles au cours du processus d'optimisation. Ainsi, chacune des deux variables linguistiques est codée par une chaîne binaire :

b_1	$c_{8,[0,1]}(d_1)$	$\cdots$	b_5	$c_{8,[0,1]}(d_5)$

avec la condition aux limites imposée sur la somme des différences d_i :

$$d_1 + \cdots + d_5 \leq 1 \ . \tag{4.19}$$

4.4.1.2 Codage des fonctions d'appartenance des sorties

Le contrôleur flou utilisé par Zhao dispose de trois sorties, dont la combinaison permet de régler les trois gains du PID. Pour qualifier les sorties K'_p et K'_d, Zhao utilise uniquement deux termes linguistiques dont les fonctions d'appartenance sont logarithmiques. Dans [ZTI93], Zhao ne justifie pas ce choix relativement original.

Nous n'avons pas retenu cette méthode car les fonctions d'appartenance logarithmiques ne sont pas facilement paramétrables si on souhaite partitionner efficacement l'univers du discours. Pour qualifier les deux sorties K'_p et K'_d, nous utilisons deux variables linguistiques comprenant cinq termes {NG, NP, EZ, PP, PG} définis sur un univers du discours $[0, 1]$. Les fonctions d'appartenance sont triangulaires, sauf pour les termes situés aux

limites de l'intervalle $[0, 1]$ pour lesquels les fonctions sont trapézoïdales. Ces fonctions d'appartenance sont représentées sur la figure 4.8.

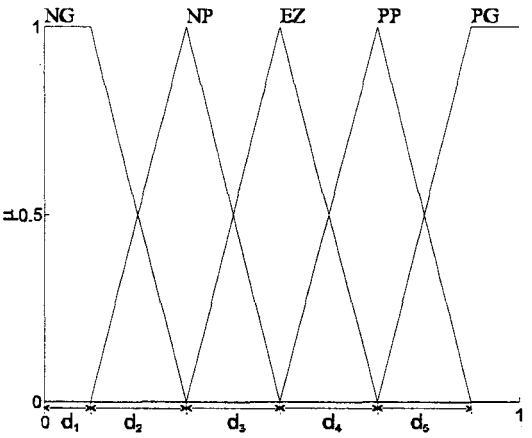


FIG. 4.8 : Fonctions d'appartenance des sorties K'_p et K'_d

Pour qualifier la sortie α, nous utilisons la même variable linguistique que Zhao. Elle est définie par quatre termes linguistiques {P, PM, M, G}, signifiant respectivement «Petit», «Petit Moyen», «Moyen» et «Grand», qui sont des singletons. Nous n'optimisons pas ces termes linguistiques, car nous voulons que les deux contrôleurs flous restent comparables.

Une chaîne binaire est utilisée pour représenter les paramètres des variables linguistiques de chacune des sorties K'_p et K'_d. Les différences d_i sont codées sur 8 bits pour obtenir une précision identique à celle des entrées. Nous ajoutons également un bit supplémentaire pour permettre l'élimination des termes inutiles. La chaîne binaire décrivant la variable linguistique d'une sortie est la suivante :

b_1	$c_{8,[0,1]}(d_1)$	$\cdots$	b_5	$c_{8,[0,1]}(d_5)$

Ce codage doit respecter la condition imposée sur les valeurs des différences d_i :

$$d_1 + \cdots + d_5 \leq 1 \ . \tag{4.20}$$

4.4.1.3 Codage des règles floues

Les règles utilisées dans notre contrôleur flou sont similaires à celles du contrôleur de Zhao. Les 25 règles possibles ont toutes deux prémisses et trois conclusions, et sont exprimées de la façon suivante :

$$\text{Règle } i_1, i_2 : \quad \text{Si } (|\epsilon(k)| \text{ est } A_{i_1}) \text{ et } (|\Delta\epsilon(k)| \text{ est } B_{i_2})$$
$$\text{alors } (K'_p \text{ est } C_{j_1}), (K'_d \text{ est } D_{j_2}), (\alpha = E_{j_3})$$
$$i_1 = 1, \cdots, 5 \qquad i_2 = 1, \cdots, 5$$

où les A_{i_1} et B_{i_2} désignent respectivement les termes linguistiques des entrées $|\epsilon(k)|$ et $|\Delta\epsilon(k)|$, et C_{j_1}, D_{j_2} et E_{j_3} les termes linguistiques des sorties K'_p, K'_d et α. Les indices j_1 et j_2 varient donc entre 1 et 5, et l'indice j_3 varie entre 1 et 4.

Pour représenter les conclusions des règles floues, nous utilisons un codage en base 6 pour les sorties K'_p et K'_d, ce qui permet de représenter par les valeurs $\{0, 1, 2, 3, 4, 5\}$ les conclusions possibles {-, NG, NP, EZ, PP, PG}, «-» signifiant la suppression de cette règle pour la sortie correspondante. Pour la sortie α, le codage s'effectue en base 5, l'ensemble $\{0, 1, 2, 3, 4\}$ représentant les conclusions {-, P, PM, M, G}.

Les trois conclusions de la règle d'indices (i_1, i_2) sont ainsi représentées par le triplet $(R^{K'_p}_{i1,i2}, R^{K'_d}_{i1,i2}, R^{\alpha}_{i1,i2})$, les deux premières valeurs étant exprimées en base 6 et la dernière en base 5. Le codage de toutes les règles floues est donné par :

$R^{K'_p}_{1,1}$	$R^{K'_d}_{1,1}$	$R^{\alpha}_{1,1}$	$\cdots$	$R^{K'_p}_{5,5}$	$R^{K'_d}_{5,5}$	$R^{\alpha}_{5,5}$

4.4.1.4 Codage de la défuzzification

Zhao utilise une méthode de défuzzification de type COA (Center Of Area). Nous préférons utiliser une méthode paramétrable afin d'ajuster au mieux la valeur de la sortie résultant de l'inférence floue. Dans les exemples présentés par la suite, nous avons retenu la méthode M-SLIDE, dont le fonctionnement est réglé par un seul paramètre β, codé par un nombre réel :

β

4.4.2 Paramétrisation de l'optimisation

L'algorithme génétique exploitant le codage mixte des paramètres décrit précédemment doit permettre d'optimiser la structure du contrôleur. Pour ce faire, on doit disposer d'une fonction permettant d'évaluer une solution possible du problème d'optimisation. On doit également fixer les valeurs des paramètres régissant l'évolution de la population traitée par cet algorithme génétique : taille de la population, probabilités de croisement et de mutation.

4.4.3 Fonction d'évaluation

Nous avons vu, à la fin du chapitre 3, qu'il est souvent impossible de construire une fonction d'évaluation en utilisant uniquement les sorties du contrôleur flou. Dans l'exemple qui nous intéresse, on ne peut pas définir de critère d'évaluation sans faire intervenir le système piloté par le contrôleur flou. Du point de vue du processus d'optimisation, ce système est constitué du PID dont on règle le fonctionnement et du processus que contrôle ce PID.

La fonction d'évaluation doit permettre de mesurer l'efficacité *globale* du couple (PID adaptatif, processus asservi). On synthétise alors un contrôleur flou adapté au système global, ce qui nécessite une nouvelle procédure d'optimisation à chaque changement de processus.

Pour évaluer les performances de l'ensemble (PID + système), nous faisons intervenir les critères de qualité définis à partir de la réponse à un échelon du système asservi. La fonction d'évaluation doit être d'autant plus élevée que les critères qui ont été présentés dans la section 4.2.3 sont proches de zéro, ce qui correspond à une régulation performante. Pour ce faire, nous utilisons la fonction d'évaluation définie par l'expression suivante :

$$f_{eval} = \frac{1}{1 + a \cdot Y_{os} + b \cdot T_5 + c \cdot IAE + d \cdot ISE} \tag{4.21}$$

où a, b, c, d sont des coefficients de pondération.

Ces coefficients sont fixés avant le processus d'optimisation de façon à prendre plus ou moins en compte chaque critère de performance. Dans les exemples traités par la suite, les quatre coefficients de pondération sont fixés à la valeur 1.

4.4.3.1 Paramètres de l'algorithme génétique

Le choix des paramètres régissant l'évolution de la population dans un algorithme génétique est assez complexe. Nous décrirons dans le chapitre 5 une méthode d'ajustement dynamique de ces paramètres.

Dans l'exemple qui nous intéresse dans ce chapitre, nous avons vérifié que le choix des paramètres n'est pas critique. On peut alors simplement fixer les valeurs de ces paramètres et les conserver constants durant tout le processus d'optimisation.

Nous utilisons une population constituée de 20 individus. La probabilité de croisement

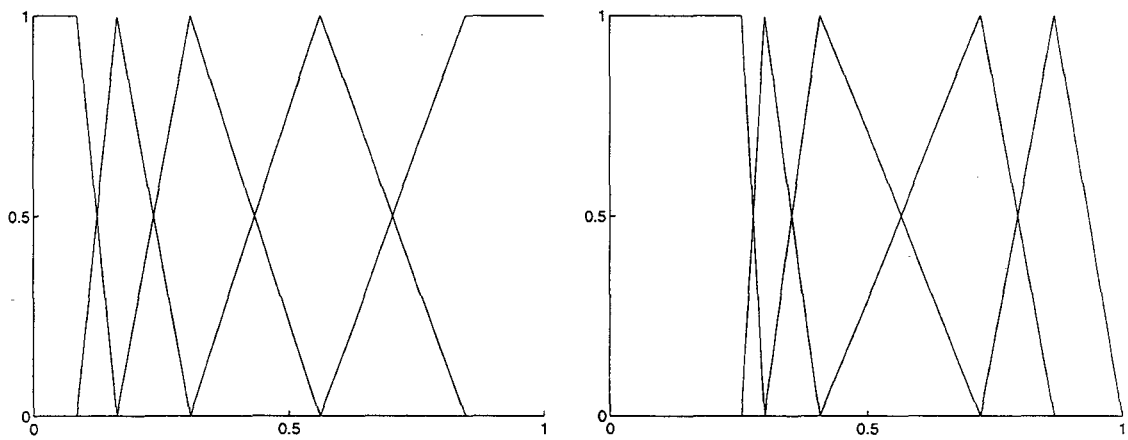
est fixée à 0, 6 pour les trois types de codage. Les probabilités de mutation pour les codages binaire et en base n sont toutes deux fixées à 0, 001, et la probabilité de mutation pour le chromosome en nombre réel est fixée à 0, 03.

4.4.4 Résultat de l'optimisation

Nous présentons les résultats obtenus à l'issue du processus d'optimisation du contrôleur flou pour le même système du second ordre que celui utilisé comme exemple par Zhao dans [ZTI93]. La fonction de transfert de ce système du deuxième ordre est donnée par :

$$G_1(p) = \frac{e^{-0.5p}}{(p+1)^2} \quad . \tag{4.22}$$

Les fonctions d'appartenance des différents termes linguistiques qualifiant les entrées $|\epsilon(k)|$ et $|\Delta\epsilon(k)|$ sont représentées sur la figure 4.9. Les termes linguistiques utilisés pour qualifier l'entrée $|\Delta\epsilon(k)|$ sont assez éloignés de la valeur zéro, comme on peut le voir sur la figure 4.9(b). Par contre, le processus d'optimisation a accordé plus d'importance aux qualifications de l'entrée $|\epsilon(k)|$ proches de la valeur zéro (cf figure 4.9(a)).

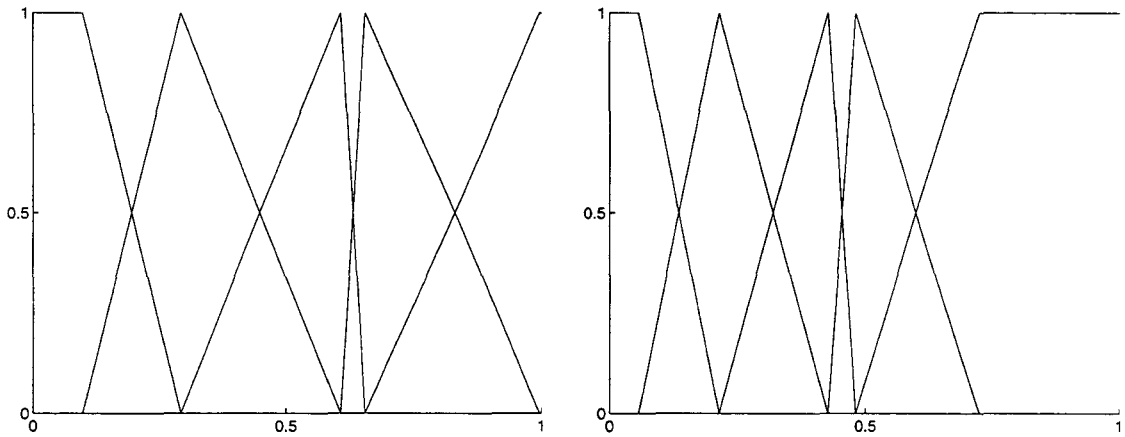


(a) Entrée $\epsilon(k)$ (b) Entrée $\Delta\epsilon(k)$

FIG. 4.9 : Fonctions d'appartenance pour les entrées $\epsilon(k)$ et $\Delta\epsilon(k)$

Sur la figure 4.10, nous avons représenté les fonctions d'appartenance des termes linguistiques pour les sorties K'_p et K'_d.

Dans les tableaux de la figure 4.11, nous avons représenté les conclusions des règles floues définies par le processus d'optimisation. Il est assez difficile d'interpréter ce résultat du fait du nombre important de règles utilisées dans le contrôleur flou.



(a) Sortie K'_p (b) Sortie K'_d

FIG. 4.10 : Fonctions d'appartenance pour les sorties K'_p et K'_d

Le processus d'optimisation a fourni une valeur $0,3041$ pour le coefficient β intervenant dans la méthode de défuzzification M-SLIDE.

4.5 Comparaison des résultats

Pour montrer l'efficacité du PID adaptatif dans lequel le contrôleur flou résulte d'un processus d'optimisation, nous avons comparé ses performances à celles de deux autres régulateurs PID. Le premier est un PID non adaptatif dont les coefficients ont été réglés par la méthode de Ziegler-Nichols, en utilisant une boucle de réaction pour la mise en auto-oscillation. Le second PID est celui proposé par Zhao.

Nous avons utilisé les exemples de systèmes définis par Zhao dans [ZTI93] : un système du second ordre avec retard pur, un système du troisième ordre et un système du quatrième ordre avec un pôle triple. Les fonctions de transfert de ces systèmes sont les suivantes :

$$G_1(p) = \frac{e^{-0.5p}}{(p+1)^2} \tag{4.23}$$

$$G_2(p) = \frac{4.228}{(p+0.5)(p^2+1.64p+8.456)} \tag{4.24}$$

$$G_3(p) = \frac{27}{(p+1)(p+3)^3} \tag{4.25}$$

Pour chaque exemple, nous avons optimisé les paramètres du contrôleur flou assurant le réglage dynamique du PID. Pour comparer qualitativement les performances, nous représentons les réponses des trois couples (PID + système) pour un même signal d'entrée

		$\lvert\Delta\epsilon(k)\rvert$				
	K'_p	P	MP	M	MG	G
$\lvert\epsilon(k)\rvert$	P	PP	NG	NP	–	NP
	MP	PP	PG	EZ	EZ	EZ
	M	PG	PP	NG	EZ	NP
	MG	EZ	EZ	EZ	NG	PP
	G	EZ	NP	EZ	–	PG

(a) Conclusions des règles floues pour K'_p

		$\lvert\Delta\epsilon(k)\rvert$				
	K'_d	P	MP	M	MG	G
$\lvert\epsilon(k)\rvert$	P	PP	EZ	NP	NP	NG
	MP	EZ	–	EZ	NG	NG
	M	PG	PP	PP	NG	NP
	MG	EZ	PP	NP	NP	NP
	G	–	NP	EZ	EZ	NG

(b) Conclusions des règles floues pour K'_d

		$\lvert\Delta\epsilon(k)\rvert$				
	α	P	MP	M	MG	G
$\lvert\epsilon(k)\rvert$	P	4	4	3	2	3
	MP	–	3	4	4	3
	M	–	4	4	3	3
	MG	4	3	4	5	4
	G	5	3	4	2	3

(c) Conclusions des règles floues pour α

FIG. 4.11 : Règles floues issues du processus d'optimisation

constitué de plusieurs échelons : un échelon passant de 0 à 1 suivi d'un échelon passant de 1 à 0, puis un échelon de 0 à -1 suivi d'un échelon de -1 à 0. Pour comparer quantitativement les trois PID, nous présentons dans un tableau les critères de performance Y_{os}, T_5, IAE, ISE calculés lors des différents fronts.

4.5.1 Comparaison pour le système du second ordre

Les réponses du système du second ordre avec retard pur pour chacun des trois PID sont présentées sur la figure 4.12. Pour le PID de Ziegler-Nichols, les paramètres sont fixés aux valeurs suivantes : $K_p = 2,808$; $T_i = 1,64$; $T_d = 0,41$. On constate aisément que le PID adaptatif dont le contrôleur flou a été optimisé est le plus performant des trois, puisque la réponse du système est la plus proche de la réponse idéale de type échelon.

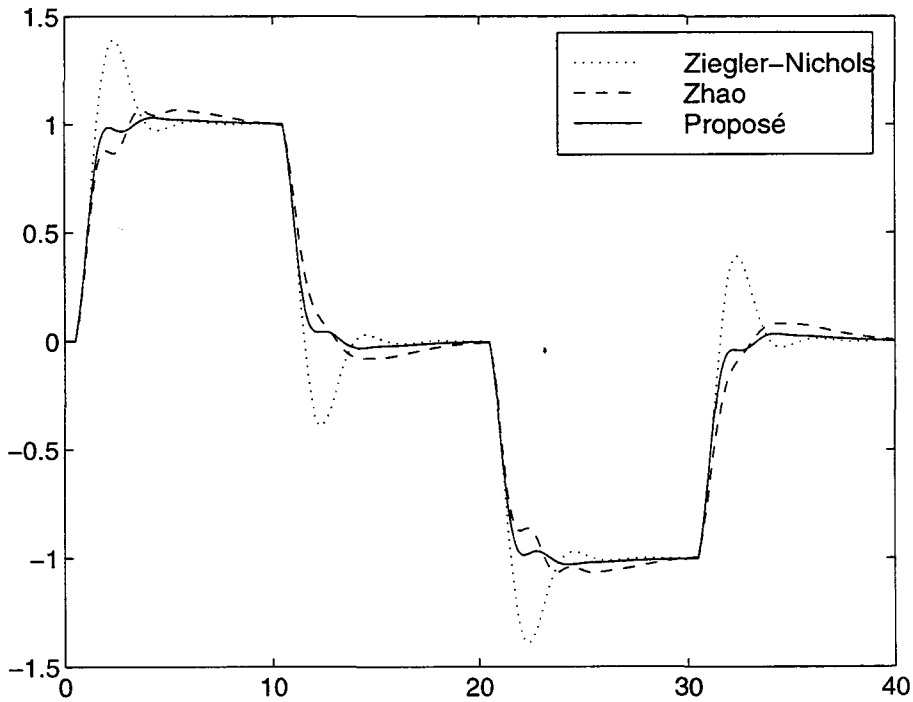


FIG. 4.12 : Réponses du système du second ordre pour les trois PID

La comparaison quantitative présentée dans le tableau de la figure 4.13 confirme les résultats de la comparaison qualitative. Selon les quatre critères, le PID adaptatif dont le contrôleur flou est optimisé est plus performant que le PID de Zhao, lui même étant plus performant que le PID réglé par la méthode de Ziegler-Nichols.

Transition	PID de Ziegler-Nichols	PID de Zhao	PID optimisé
Front montant	$Y_{os} = 0,3899$ $T_5 = 3,7$ $IAE = 1,7270$ $ISE = 1,1478$	$Y_{os} = 0,0684$ $T_5 = 6,8$ $IAE = 1,6572$ $ISE = 1,0547$	$Y_{os} = 0,0307$ $T_5 = 1,8$ $IAE = 1,3243$ $ISE = 0,9856$
Front descendant	$Y_{os} = 0,3899$ $T_5 = 3,7$ $IAE = 1,7270$ $ISE = 1,1478$	$Y_{os} = 0,0801$ $T_5 = 6,8$ $IAE = 1,7670$ $ISE = 1,1200$	$Y_{os} = 0,0331$ $T_5 = 1,9$ $IAE = 1,3612$ $ISE = 0,9914$

FIG. 4.13 : Indices de performance, système du second ordre

4.5.2 Comparaison pour le système du troisième ordre

Les réponses du système du troisième ordre pour les trois PID sont présentées sur la figure 4.14. Pour le PID de Ziegler-Nichols, les paramètres sont fixés aux valeurs suivantes : $K_p = 2,2096$; $T_i = 1,034$; $T_d = 0,2585$. On constate encore que le PID adaptatif dont le contrôleur flou a été optimisé est le plus performant des trois régulateurs.

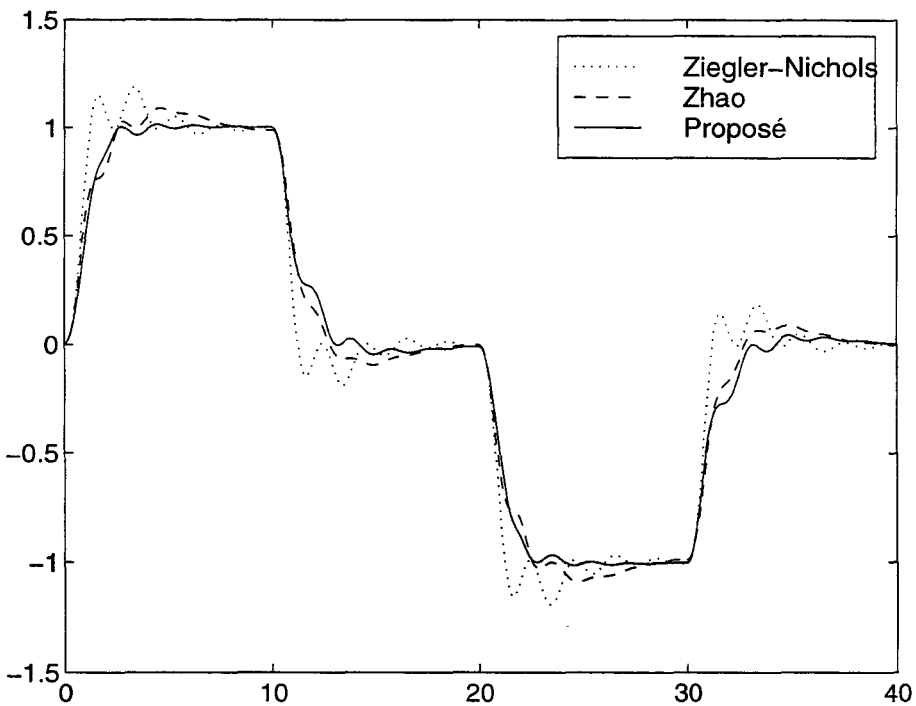


FIG. 4.14 : Réponses du système du troisième ordre pour les trois PID

La comparaison quantitative est présentée dans le tableau de la figure 4.15. On constate encore que, selon les quatre critères, le PID adaptatif proposé est plus performant que les PID de Zhao et de Ziegler-Nichols.

Transition	PID de Ziegler-Nichols	PID de Zhao	PID optimisé
Front montant	$Y_{os} = 0,1882$ $T_5 = 5,5$ $IAE = 1,1848$ $ISE = 0,6932$	$Y_{os} = 0,0911$ $T_5 = 6,7$ $IAE = 1,3996$ $ISE = 0,7809$	$Y_{os} = 0,0155$ $T_5 = 2,2$ $IAE = 1,2142$ $ISE = 0,8223$
Front descendant	$Y_{os} = 0,1882$ $T_5 = 5,5$ $IAE = 1,1848$ $ISE = 0,6932$	$Y_{os} = 0,0938$ $T_5 = 6,4$ $IAE = 1,5089$ $ISE = 0,8340$	$Y_{os} = 0,0460$ $T_5 = 2,7$ $IAE = 1,4240$ $ISE = 0,8210$

FIG. 4.15 : Indices de performance, système du troisième ordre

4.5.3 Comparaison pour le système du quatrième ordre

Les réponses du système du quatrième ordre comportant un pôle triple pour les trois PID sont tracées sur la figure 4.16. Pour le PID de Ziegler-Nichols, les paramètres sont fixés aux valeurs suivantes : $K_p = 3,072$; $T_i = 1,352$; $T_d = 0,338$. Le PID adaptatif dont le contrôleur flou a été optimisé est encore le plus performant des trois.

Les critères de performance évalués sur ces trois réponses sont présentés dans le tableau de la figure 4.17. On constate également que, selon les quatre critères, le PID adaptatif proposé est plus performant que les PID de Zhao et de Ziegler-Nichols.

4.6 Conclusion

Dans ce chapitre, nous avons présenté un exemple d'application de la méthode de synthèse d'un contrôleur flou décrite dans le chapitre 3. Le contrôleur flou synthétisé est utilisé pour régler dynamiquement les paramètres d'un régulateur linéaire de type PID.

Les différents paramètres du contrôleur flou sont codés par trois chromosomes afin d'être optimisés par un algorithme génétique. Durant le processus d'optimisation, les so-

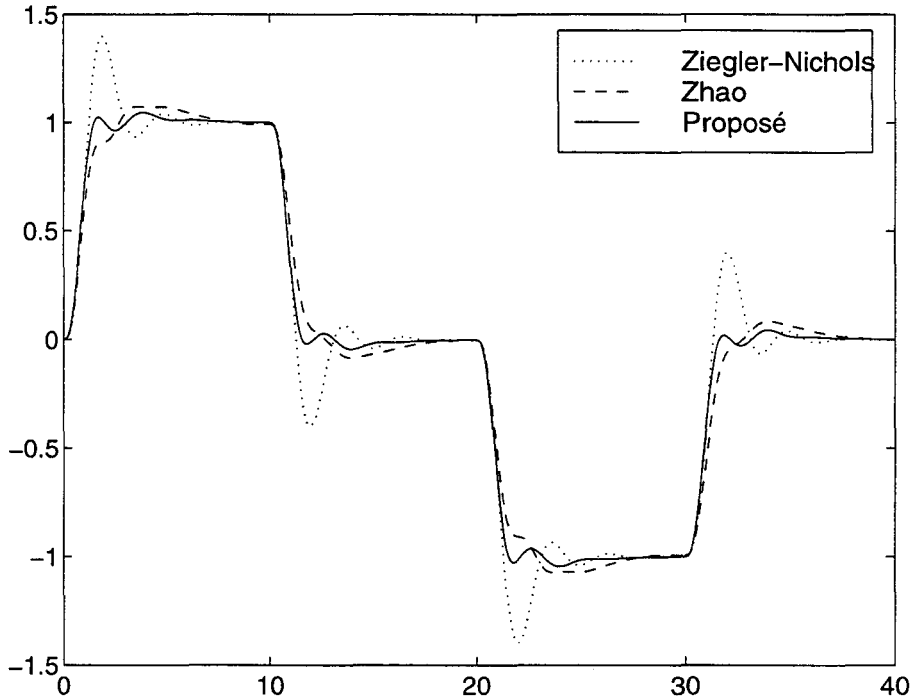


FIG. 4.16 : Réponses du système du quatrième ordre pour les trois PID

Transition	PID de Ziegler-Nichols	PID de Zhao	PID optimisé
Front montant	$Y_{os} = 0,3982$ $T_5 = 3,7$ $IAE = 1,4118$ $ISE = 0,8646$	$Y_{os} = 0,0738$ $T_5 = 5,8$ $IAE = 1,3476$ $ISE = 0,8057$	$Y_{os} = 0,0471$ $T_5 = 1,3$ $IAE = 1,0203$ $ISE = 0,7137$
Front descendant	$Y_{os} = 0,3982$ $T_5 = 3,7$ $IAE = 1,4118$ $ISE = 0,8646$	$Y_{os} = 0,0845$ $T_5 = 5,6$ $IAE = 1,3715$ $ISE = 0,8475$	$Y_{os} = 0,0442$ $T_5 = 1,3$ $IAE = 1,0136$ $ISE = 0,7130$

FIG. 4.17 : Indices de performance, système du quatrième ordre

lutions possibles sont évaluées par l'intermédiaire d'une fonction qui fait intervenir quatre critères de performance calculés sur la réponse du système global.

Les simulations réalisées sur trois systèmes différents nous montrent que le contrôleur flou optimisé par l'algorithme génétique est plus performant que le PID proposé par Zhao et que celui dont les paramètres sont fixés par la méthode de Ziegler-Nichols. Cependant, nous avons utilisé une méthode d'optimisation par algorithme génétique en supposant un

réglage fixe des paramètres pilotant le fonctionnement de l'algorithme évolutif.

Cette stratégie a été rendue possible du fait que, dans ces exemples, il apparaît que l'ajustement de ces paramètres (probabilités de croisement, de mutation et taille de la population) n'est pas crucial. Toutefois, un réglage dynamique des paramètres de fonctionnement d'un algorithme génétique peut être envisagé afin d'en améliorer les performances. Cette stratégie fait l'objet du chapitre suivant.

Chapitre 5

Réglage de la convergence d'un algorithme génétique par contrôleur flou : un exemple

Dans ce chapitre, nous décrivons l'application pour laquelle nous avons initialement développé la méthode de synthèse d'un contrôleur flou présentée dans le chapitre 3. Dans cette application, dont le contexte a été présenté brièvement dans le préambule de ce mémoire, nous utilisons un algorithme génétique pour traiter des données issues d'un système de vision.

5.1 Introduction

Dans le cadre d'une étude proposée au Laboratoire d'Automatique I3D par la Direction de la Recherche de RENAULT, nous avons développé un système de vision permettant de mesurer la position angulaire d'un objet par rapport à un repère fixe. Ce système est utilisé pour estimer la position angulaire de la remorque d'un camion par rapport au tracteur. Nous considérons que la remorque est en rotation libre autour d'une rotule, ce qui permet de définir sa position par l'intermédiaire des trois angles d'Euler.

Deux caméras, situées au niveau des rétroviseurs du camion et dirigées vers l'arrière, permettent d'obtenir deux vues différentes de la remorque sur laquelle nous avons placé des marques facilement repérables lors du traitement des images et dont les positions sont

connues avec précision dans un référentiel lié à la remorque. Le système de vision permet de définir précisément les positions des projections de ces marques dans les référentiels liés aux deux images. Pour estimer les angles définissant la position de la remorque, nous recherchons les paramètres de la transformation géométrique qui relie les coordonnées des marques détectées dans les images à celles des marques dessinées sur la remorque.

Dans ce chapitre, nous montrons comment on peut transformer le problème de l'estimation de la position angulaire en un problème d'optimisation faisant intervenir simultanément toutes les informations (paramètres des caméras, positions des marques sur la remorque, mesure des positions des marques dans les deux images). La solution, constituée des trois angles intervenant dans les transformations géométriques, est ensuite déterminée par un algorithme génétique dont la convergence doit être la plus rapide possible pour respecter les contraintes de temps de calcul imposées par l'application.

Pour cette application particulière, nous avons comparé les performances de quatre algorithmes génétiques différents :

- Un algorithme génétique standard, utilisant un codage binaire des trois angles dans un chromosome.
- Un algorithme génétique adaptatif proposé par Srinivas, dans lequel les probabilités de croisement et de mutation sont réglées dynamiquement.
- Un algorithme génétique dont les probabilités sont réglées par un contrôleur flou de structure fixe.
- Un algorithme génétique dont les paramètres sont réglés par un contrôleur flou optimisé par un algorithme génétique selon la méthode présentée dans le chapitre 3.

Nous comparons les algorithmes selon deux critères. Nous vérifions tout d'abord l'exactitude de la solution fournie, en nous basant sur des images de synthèse pour lesquelles tous les paramètres sont connus précisément. Ensuite, nous prenons en compte le nombre d'itérations nécessaires à chaque algorithme pour obtenir la solution.

5.2 Estimation de la position angulaire par un algorithme génétique

Dans cette section, nous établissons les relations géométriques reliant les coordonnées des marques détectées dans les images aux coordonnées des marques dessinées sur la remorque. Ensuite, nous voyons comment ramener l'estimation des trois angles à un problème de maximisation d'une fonction, qui est résolu par un algorithme génétique.

5.2.1 Modèle géométrique du système de vision

Si nous supposons que la remorque tourne librement autour d'un point fixe O, sa position peut être exactement définie par trois angles d'Euler [HM95b]. Ces trois angles (α, β, γ) décrivent les trois rotations permettant de passer d'un repère fixe $R_s = (O, \vec{X_s}, \vec{Y_s}, \vec{Z_s})$ lié au camion et centré sur le point O à un repère mobile $R_m = (O, \vec{X_m}, \vec{Y_m}, \vec{Z_m})$ lié à la remorque et également centré sur le point O.

Les images sont acquises par des caméras fixées sur le camion et visant une partie de la remorque. Nous définissons deux autres repères liés à chaque caméra. Le repère optique $R_f = (F, \vec{X_f}, \vec{Y_f}, \vec{Z_f})$ est centré sur le foyer F de l'objectif de la caméra. Son axe $\vec{X_f}$ est vertical et $\vec{Z_f}$ est orienté vers la remorque selon l'axe optique.

Nous utilisons également le repère image $R_i = (I, \vec{X_i}, \vec{Y_i}, \vec{Z_i})$, centré sur le point I situé à l'intersection du plan image et de l'axe optique. Le centre de l'image I est situé à la distance focale f du centre optique F. Les coordonnées d'un point dans ce repère sont définies en nombre de pixels, et non en unités de longueur. Dans les transformations géométriques, nous faisons donc intervenir deux facteurs d'échelle k_x et k_y définis en fonction des caractéristiques du capteur CCD de la caméra.

Nous notons θ l'angle séparant l'axe optique IF de l'axe FO liant le centre optique à la rotule de la remorque. Les différents repères sont représentés sur la figure 5.1.

Une marque M est dessinée sur la remorque au point de coordonnées $(x_{m,M}, y_{m,M}, z_{m,M})$ dans le repère R_m. Cette marque se projette dans l'image sur le point m dont les coordonnées dans le repère R_i sont notées $(x_{i,m}, y_{i,m}, 0)$. Pour déterminer les coordonnées du point m dans le repère image, nous calculons tout d'abord les coordonnées dans le repère R_f du vecteur directeur $\vec{u}$ de la droite FM issue du point M et passant par le centre

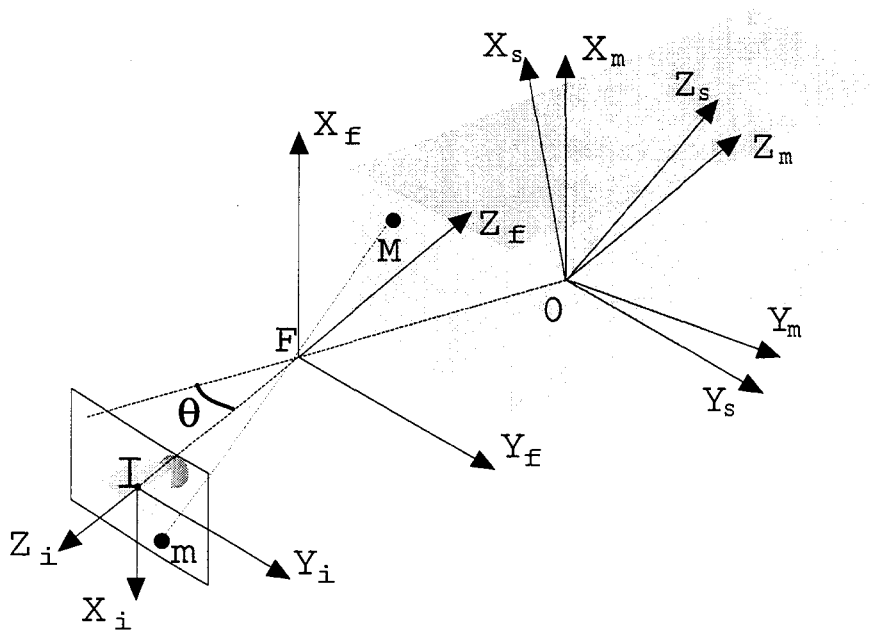


FIG. 5.1 : Modèle géométrique du système de vision

optique de la caméra. Ce vecteur est calculé par la transformation suivante :

$$\vec{u} = K.\Theta.A.\overrightarrow{OM} + K.\overrightarrow{FO} \quad , \tag{5.1}$$

dans laquelle la matrice K fait intervenir le changement d'échelle, la matrice Θ prend en compte la différence d'orientation des repères R_f et R_m et la matrice A décrit les rotations. Ces matrices sont définies par :

$$K = \begin{pmatrix} -k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & -\frac{1}{f} \end{pmatrix} \quad , \quad \Theta = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix} \quad ,$$

$$A = \begin{pmatrix} e \cdot g & a \cdot b \cdot g - d \cdot c & d \cdot b \cdot g + a \cdot c \\ e \cdot c & a \cdot b \cdot c + d \cdot g & d \cdot b \cdot c - a \cdot g \\ -b & a \cdot e & d \cdot e \end{pmatrix}$$

où $a = \sin\alpha$, $b = \sin\beta$, $c = \sin\gamma$, $d = \cos\alpha$, $e = \cos\beta$, $g = \cos\gamma$.

Le point m est situé à l'intersection de la droite $(F, \vec{u})$ et du plan image. La coordonnée $z_{f,m}$ du point m dans le repère optique R_f est constante et égale à $-f$. Les deux coordonnées de m dans le repère image sont ainsi données par :

$$x_{i,m} = x_u/f$$

$$y_{i,m} = y_u/f \ . \tag{5.2}$$

En résumant toutes les opérations, on peut définir les coordonnées du point m par l'intermédiaire de deux fonctions f_x et f_y, dont les paramètres sont les coordonnées de la marque M dans le repère lié à la remorque, les caractéristiques du système de prise de vue symbolisées par C, et les trois angles définissant la position de la remorque α, β, et γ. Nous utilisons ainsi les expressions suivantes :

$$\begin{cases} x_{i,m} = f_x(M, C, \alpha, \beta, \gamma) \\ y_{i,m} = f_y(M, C, \alpha, \beta, \gamma) \end{cases} \tag{5.3}$$

5.2.2 De l'estimation angulaire à un problème d'optimisation

Lorsqu'on connaît tous les paramètres du système de prise de vue ainsi que la position angulaire de la remorque, les positions des marques dans les images sont calculées à partir de leurs positions sur la remorque en utilisant les fonctions f_x et f_y selon les équations 5.3. Inversement, lorsqu'on connaît les positions de n marques détectées dans les images, pour estimer les angles α, β et γ il suffit théoriquement de résoudre le système composé de $2n$ équations dont ils constituent les trois inconnues :

$$\begin{cases} x_{i,m_1} - f_x(M_1, C, \alpha, \beta, \gamma) = 0 \\ y_{i,m_1} - f_y(M_1, C, \alpha, \beta, \gamma) = 0 \\ \qquad\qquad\qquad\qquad\qquad\quad \ldots \\ x_{i,m_n} - f_x(M_n, C, \alpha, \beta, \gamma) = 0 \\ y_{i,m_n} - f_y(M_n, C, \alpha, \beta, \gamma) = 0 \end{cases} \tag{5.4}$$

Ce système est non linéaire puisque les angles interviennent en tant qu'arguments de fonctions trigonométriques. Si on utilise une seule marque, le système est constitué uniquement de deux équations, il a donc une infinité de solutions. Par contre, pour plus de deux marques, le système est sur-dimensionné puisqu'on dispose de plus d'équations que d'inconnues. Comme les coordonnées des points qui interviennent dans les équations sont connues de façon imprécise, on aboutit à un système qui n'a pas de solution exacte.

Avec n marques, comme il n'existe pas de solution exacte, on cherche simplement une *bonne* solution, pour laquelle tous les termes $|x_{i,m_k} - f_x(M_k, C, \alpha, \beta, \gamma)|$ et $|y_{i,m_k} - f_y(M_k, C, \alpha, \beta, \gamma)|$, $k = 1, \ldots, n$, prennent des valeurs aussi proches que possible de zéro.

Ainsi, la résolution du système d'équations peut se ramener à un problème de recherche de l'extrémum d'une fonction définie à partir des termes intervenant initialement dans les équations. Cette fonction, dont les paramètres sont les trois angles α, β et γ, doit atteindre un extrémum absolu quand les angles correspondent à la meilleure estimation de la position angulaire de la remorque.

Cette approche se justifie d'autant plus que nous utilisons deux caméras dans cette application. Les angles estimés doivent permettre de minimiser l'ensemble des termes :

$$\begin{aligned} &|x_{i_1,m_k} - f_x(M_k, C_1, \alpha, \beta, \gamma)| \ , \\ &|y_{i_1,m_k} - f_y(M_k, C_1, \alpha, \beta, \gamma)| \ , \\ &|x_{i_2,m_k} - f_x(M_k, C_2, \alpha, \beta, \gamma)| \ , \\ &|y_{i_2,m_k} - f_y(M_k, C_2, \alpha, \beta, \gamma)| \ , \end{aligned} \tag{5.5}$$

où $k = 1, \ldots, n$, x_{i_1,m_k} et y_{i_1,m_k} sont les coordonnées de la marque M_k vue dans l'image de la caméra 1, et x_{i_2,m_k} et y_{i_2,m_k} les coordonnées de cette même marque vue dans l'image de la caméra 2. Les paramètres de prise de vue pour chaque caméra sont résumés par les caractéristiques C_1 et C_2. Dans ces équations, nous avons supposé que toutes les marques sont visibles par les deux caméras, ce qui n'est pas toujours le cas dans notre application. Lorsqu'une marque n'est pas visible dans une image, il faut supprimer du système les deux équations faisant intervenir cette marque pour la caméra correspondante.

La fonction construite à partir des termes issus du système d'équations doit présenter un extrémum global pour la meilleure estimation possible des trois angles. Malheureusement, vu le nombre important de paramètres intervenant dans cette fonction, il est probable qu'elle possèdera un nombre important d'extréma locaux, correspondant à des solutions du problème ne satisfaisant pas simultanément toutes les contraintes.

Pour éviter de tomber dans un extrémum local lors de la recherche de la solution, nous avons choisi d'utiliser un algorithme génétique. Il nous faut donc construire une fonction d'évaluation qui présente un maximum absolu pour la meilleure estimation possible des trois angles. De nombreuses expressions sont possibles pour définir cette fonction, mais le choix doit être guidé par plusieurs contraintes [Gol89] :

- La fonction doit permettre d'évaluer *tous* les individus testés par l'algorithme génétique. En particulier, elle doit permettre de comparer deux «mauvaises» solutions

entre elles.

- La fonction doit permettre de comparer précisément une «très bonne» solution avec une «bonne» solution.
- La fonction doit être régulière, elle ne doit donc pas présenter des discontinuités marquées.

Pour ces différentes raisons, nous avons choisi d'utiliser l'expression suivante :

$$f_{eval}(\alpha,\beta,\gamma) = \sum_{k=1}^{n} \left[\delta(k,1).e^{-d_1^2(k)/\sigma^2} + \delta(k,2).e^{-.d_2^2(k)/\sigma^2}\right] \quad , \tag{5.6}$$

dans laquelle $d_1(k)$ désigne la distance euclidienne entre la k-ième marque détectée dans l'image 1 et la projection de la marque réelle M_k dans cette même image obtenue en utilisant le modèle des équations 5.3, $d_2(k)$ est défini de la même façon pour l'image de la caméra 2. Par exemple, pour la caméra 1, la distance $d_1(k)$ entre la marque détectée et la projection de la marque réelle est donnée par :

$$d_1(k) = \sqrt{(x_{i_1,m_k} - f_x(M_k, C_1, \alpha, \beta, \gamma))^2 + (y_{i_1,m_k} - f_y(M_k, C_1, \alpha, \beta, \gamma))^2} \tag{5.7}$$

Les termes $\delta(k,1)$ et $\delta(k,2)$ intervenant dans l'expression de la fonction d'évaluation permettent d'éliminer un terme lorsque la marque M_k n'est pas vue par la caméra correspondante. Ils sont définis de la façon suivante :

$$\begin{cases} \delta(k,c) = 1 & \text{si la marque } M_k \text{ est vue par la caméra } c \\ \delta(k,c) = 0 & \text{si la marque } M_k \text{ n'est pas vue par la caméra } c \end{cases} \tag{5.8}$$

- L'écart type σ qui intervient dans tous les termes Gaussiens de la fonction décrite par l'équation 5.6 permet de régler la sélectivité de l'évaluation. Dans notre étude, nous avons fixé la valeur de σ, mais on peut tout à fait envisager de faire varier l'écart type au cours des itérations de l'algorithme génétique afin de mieux séparer les solutions durant les dernières itérations.

5.2.3 Détection des marques par traitement d'image

La fiabilité et la précision de la méthode d'estimation décrite précédemment repose sur la qualité de l'opération de détection des marques dans l'image. Durant cette phase de traitement, chaque marque doit être identifiée et différenciée des autres marques présentes dans l'image. Sa position doit être déterminée avec la plus grande précision possible.

Dans cette application, nous maîtrisons entièrement l'aspect visuel de la remorque et des marques : nous pouvons choisir le niveau de gris de la surface de la remorque, celui des marques et concevoir des marques de formes différentes. Cela permet de simplifier considérablement les traitements permettant la différenciation : chaque marque est caractérisée par son niveau de gris connu a priori. On pourrait également, si nécessaire, différencier les marques en utilisant des formes géométriques différentes (rond, carré ...).

Les marques sont détectées en recherchant des zones de petites dimensions dans l'image dont le niveau de gris est constant. Une fois qu'une marque est localisée dans l'image, sa position précise est déterminée en calculant le centre de gravité des pixels constituant la région. Cette méthode classique permet d'estimer la position de la marque avec une précision sub-pixel.

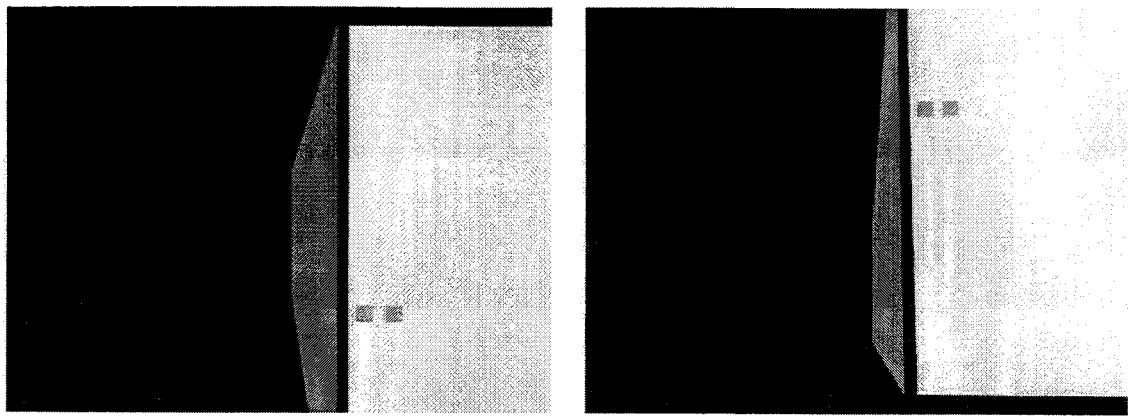
5.2.4 Validation par des images de synthèse

Pour valider notre méthode d'estimation angulaire, mous avons développé un système de simulation utilisant des images de synthèse. Nous avons synthétisé un ensemble d'images pour des valeurs de l'angle de lacet α et de l'angle de roulis β comprises entre 0 et 15 degrés. Nous considérons que l'angle de tangage γ est nul dans cette simulation, pour limiter le nombre d'images de test qui seront utilisées durant la phase d'optimisation. Cette approximation est justifiée puisque l'angle de tangage de la remorque d'un camion est toujours très faible.

La figure 5.2 montre deux exemples d'images synthétiques correspondant à la scène observée par la caméra gauche. Dans ces images, nous avons défini deux marques de forme carrée. Chaque marque est caractérisée par un niveau de gris spécifique qui est connu a priori. Les paramètres optiques et géométriques du système simulé sont les suivants pour la caméra gauche :

Distance focale	$f = 35,0mm$
Position de la rotule O	$(x_{f,O}, y_{f,O}, z_{f,O}) = (-46.67cm, 156.40cm, 575.0cm)$
Positions des marques	$(x_{M_1}, y_{M_1}, z_{M_1}) = (0cm, -117.0cm, -375.0cm)$
	$(x_{M_2}, y_{M_2}, z_{M_2}) = (0cm, -104.0cm, -375.0cm)$

La configuration correspondant à la caméra droite est obtenue par symétrie par rapport à l'axe longitudinal du camion.



(a) $\alpha = 0°, \beta = 0°$ (b) $\alpha = 1°, \beta = 15°$

FIG. 5.2 : Images de synthèse, caméra gauche

5.3 Réglage des paramètres d'un algorithme génétique

Dans le chapitre 2, nous avons présenté le principe des algorithmes génétiques en nous basant sur un exemple simple. Grâce à l'application décrite dans le chapitre 4, nous avons également pu vérifier l'efficacité de cette méthode d'optimisation pour la résolution d'un problème complexe. Dans ces deux exemples, les paramètres régissant le fonctionnement de l'algorithme génétique, à savoir la taille de la population N_p et les probabilités de croisement p_c et de mutation p_m, étaient constants durant toute la durée du processus d'optimisation.

Dans le chapitre 2, nous avons décrit le problème de la convergence prématurée d'un algorithme génétique, qui dépend fortement du choix des paramètres. Si les probabilités de croisement et de mutation sont trop faibles, l'algorithme a tendance à converger trop rapidement vers des minima locaux de la fonction d'évaluation. Par contre, si ces deux probabilités sont trop élevées, l'algorithme génétique évolue difficilement vers une solution stable, ce qui augmente le nombre total d'itérations nécessaires. Dans ces conditions, comment peut on définir au mieux les paramètres d'un algorithme génétique ?

Plusieurs méthodes permettant de choisir les paramètres ont été décrites dans la littérature. Certaines techniques consistent à fixer les paramètres qui restent constants pendant toute la durée du processus d'optimisation [Jon75, Gre86, SCED89]. Pour améliorer la convergence, plusieurs auteurs ont proposé d'ajuster dynamiquement la valeur des paramètres durant l'évolution de la population [HM91, Bäc92, AMM94, SP94, XV94, GAF95, HLV95b, Bäc96, HL96b, HYFS96, WWSJ96]. Les principales méthodes de réglage sont

décrites ci-après.

Dans l'application qui nous intéresse, à savoir l'estimation des angles définissant la position de la remorque, le temps de calcul nécessaire pour réaliser une mesure doit être le plus faible possible, ce qui revient à limiter au maximum le nombre d'itérations nécessaires pour aboutir à la convergence de l'algorithme génétique. Nous voyons comment on peut atteindre cet objectif en effectuant un réglage dynamique des paramètres de l'algorithme génétique par un contrôleur flou.

5.3.1 Réglage fixe des paramètres

Dans [Jon75], DeJong a appliqué un algorithme génétique standard pour résoudre plusieurs problèmes typiques d'optimisation numérique. Il a proposé un ensemble de cinq fonctions d'évaluation appelées «fonctions de test de DeJong» (*DeJong Test Suite*) qui permettent de mesurer l'efficacité d'un algorithme génétique, qui doit en rechercher le maximum absolu. Pour la plupart des problèmes standard, DeJong conseille d'utiliser des paramètres fixés à :

$$N_p = 100, \qquad p_c = 0,6, \qquad p_m = 10^{-3} \ . \tag{5.9}$$

Afin de mieux s'adapter au problème traité, Schaffer et al. [SCED89] proposent une formule donnant la probabilité de mutation p_m en fonction de la taille de la population N_p et de la longueur m de la chaîne binaire constituant le chromosome :

$$p_m \approx \frac{1.75}{N_p \cdot \sqrt{m}} \ . \tag{5.10}$$

Greenwell et al. [GAF95] ont également déterminé une formule donnant la valeur p_m qui maximise la probabilité que l'algorithme génétique trouve le maximum global de la fonction d'évaluation.

Grefenstette considère que le choix des paramètres d'un algorithme génétique est également un problème d'optimisation. En plus des trois paramètres habituels, il utilise trois valeurs supplémentaires (*Generation Gap* (G), *Scaling Window* (W) et *Selection Strategy* (S)) qui permettent de stabiliser l'évolution de la population. Il incorpore également dans le processus d'optimisation le choix de la stratégie de sélection. Grefenstette code ces six paramètres dans un chromosome dont la valeur optimale est obtenue par un deuxième

algorithme génétique (*méta-AG*). La fonction d'évaluation est construite grâce à l'analyse des performances de l'algorithme génétique sur les fonctions de test de DeJong. Le principe de cette méthode est résumé par le synoptique de la figure 5.3.

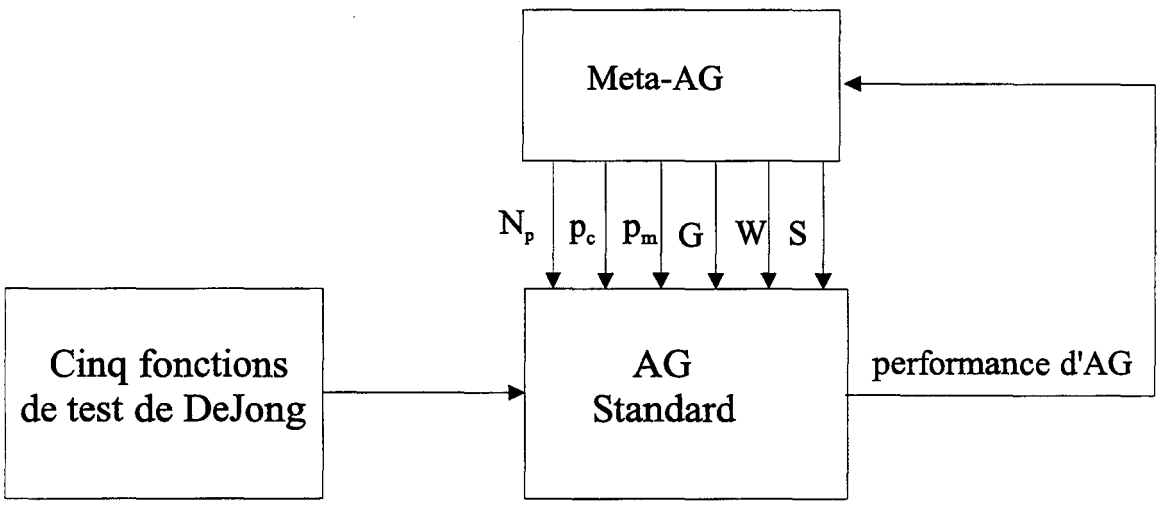


FIG. 5.3 : Optimisation des paramètres d'AG par méta-AG

Le concept de *Scaling Window* est très intéressant car il permet de traiter des problèmes dans lesquels les bornes de la fonction d'évaluation ne sont pas connues précisément. Dans [Gol89], Goldberg a décrit les principes de base permettant de construire une fonction d'évaluation. Lorsque la fonction f à maximiser est bornée, il conseille d'utiliser comme fonction d'évaluation f_{eval} la différence entre la fonction f et sa valeur minimale f_{min}. Grefenstette définit la valeur minimale de la fonction f en conservant le minimum des évaluations effectuées sur tous les individus durant les W générations précédentes.

Grefenstette a recherché les paramètres permettant un fonctionnement optimal de l'algorithme génétique en utilisant deux fonctions d'évaluations différentes. La première mesure la performance en ligne de l'algorithme, elle prend en compte toutes les itérations réalisées. La deuxième mesure la performance hors-ligne, ce qui correspond à étudier uniquement la qualité de la convergence.

En utilisant les paramètres définis par Grefenstette on obtient en définitive des résultats très similaires à ceux obtenus avec les paramètres proposés initialement par DeJong. Cela tend à prouver qu'un réglage fixe des paramètres, même optimal selon certains critères, ne permet pas d'améliorer les performances d'un algorithme génétique de façon significative.

5.3.2 Réglage dynamique des paramètres

Le théorème des schémas permet de spécifier l'utilité des différentes phases de traitement intervenant dans un algorithme génétique. L'opération de sélection tend à conserver les individus les plus performants dans les nouvelles générations, alors que le croisement et la mutation permettent d'explorer l'espace de recherche. En changeant les paramètres en cours de fonctionnement, on peut modifier le comportement de l'algorithme génétique durant le processus d'optimisation. Durant les premières itérations, on utilise des probabilités de croisement et de mutation assez élevées, ce qui permet d'explorer efficacement l'espace des solutions. Par la suite, on choisit des valeurs plus faibles, ce qui tend à stabiliser la population aux alentours des bonnes solutions.

Pour ce faire, Hesser et al. [HM91] modifient la formule proposée par Schaffer [SCED89], en faisant intervenir le numéro t de la génération traitée par l'algorithme génétique :

$$p_m = p_{m_0} \cdot e^{-\gamma \frac{t}{2}} \quad , \tag{5.11}$$

où p_{m_0} désigne une probabilité initiale déterminée par une formule similaire à celle de Schaffer et γ est une constante qui permet de régler la décroissance en fonction du temps. Cette méthode d'ajustement dynamique ne tient pas compte de l'évolution réelle de la population.

Davis propose de modifier les probabilités de croisement et de mutation en fonction de l'adaptation des individus constituant la population [Dav89, Dav91]. De la même façon, Srinivas et al. utilisent des critères évalués sur la population, comme la moyenne $\overline{f}$ de la fonction d'évaluation ou le maximum f_{max} de cette fonction, afin de modifier dynamiquement les probabilités d'évolution [SP94]. Srinivas a été le premier à utiliser des opérateurs génétiques dont les paramètres sont adaptés à *chaque* individu, selon la valeur prise par la fonction d'évaluation f.

Les probabilités de croisement sont ainsi ajustées pour chaque chromosome en fonction de son adaptation, selon les expressions :

$$p_c = \begin{cases} k_1 \cdot (f_{max} - f)/(f_{max} - \overline{f}) & \text{si} \quad f' \geq \overline{f} \\ k_3 & \text{si} \quad f' < \overline{f} \end{cases} \tag{5.12}$$

$$p_m = \begin{cases} k_2 \cdot (f_{max} - f)/(f_{max} - \overline{f}) & \text{si} \quad f \geq \overline{f} \\ k_4 & \text{si} \quad f < \overline{f} \end{cases} \tag{5.13}$$

où $\overline{f}$ est la valeur moyenne de fonction d'évaluation sur toute la population, f_{max} la valeur maximum de cette fonction et f la valeur de la fonction pour le chromosome affecté par l'opération génétique. k_1, k_2, k_3 et k_4 sont des constantes prenant leurs valeurs dans l'intervalle $[0, 1]$. Dans [SP94], Srinivas a fixé ces valeurs à $k_1 = k_3 = 1$ et $k_2 = k_4 = 0.5$.

Dans les équations 5.12 et 5.13, le terme $(f_{max} - \overline{f})$ diminue quand la population converge vers un optimum (local ou global) de la fonction d'évaluation. On peut ainsi utiliser ce critère, qui concerne toute la population, pour analyser la convergence de l'algorithme. Inversement, $1/(f_{max} - \overline{f})$ est un critère permettant de mesurer la diversité de la population traitée par l'algorithme génétique. Dans ces mêmes équations, Srinivas utilise le terme $(f_{max} - f)$ qui permet de régler les probabilités individu par individu. Quand un individu a des performance supérieures à la moyenne $(f \geq \overline{f})$ et qu'il s'approche de la solution ($f_{max} - f$ est proche de zéro), les probabilités de croisement et de mutation utilisées pour cet individu sont faibles.

Bäck a appliqué le concept de *stratégie d'évolution* (en Anglais Evolution Strategies) aux algorithmes génétiques [Bäc92, Bäc96]. Selon ce principe, un individu est caractérisé par ses propres possibilités d'évolution. Dans le cadre des algorithmes génétiques, cela signifie que les probabilités de croisement et de mutation sont codées dans le chromosome et qu'elles évoluent ainsi avec la population. En suivant un principe similaire, Arabas a décrit un algorithme dans lequel la taille de la population évolue au cours des générations [AMM94]. Il considère que chaque chromosome a une certaine durée de vie, et qu'il s'éteint naturellement au bout de quelques itérations.

5.3.3 Réglage dynamique par un contrôleur flou

Nous avons vu que les méthodes de réglage dynamique des paramètres d'un algorithme génétique sont basées sur le principe suivant :

- Durant les premières itérations, les probabilités de croisement et de mutation sont fixées à des valeurs relativement élevées pour permettre un exploration efficace de l'espace des solutions du problème traité.
- Durant les dernières itérations, lorsque l'algorithme converge, on doit diminuer les valeurs des probabilités pour stabiliser la population autour des solutions trouvées durant la phase d'exploration.

Dans la plupart des méthodes décrites précédemment, l'évolution des paramètres est régie par une heuristique déduite du problème particulier que traite l'algorithme génétique.

On remarque que les valeurs des paramètres sont choisies en fonction de qualifications imprécises ou incomplètes de l'état de l'algorithme : début du processus, premières itérations, phase de convergence, etc. Ces qualifications prennent un sens précis lorsqu'on connaît plus précisément le problème traité. En remarquant que ces qualifications imprécises peuvent être exploitées par le formalisme de la logique floue, Lee et Takagi ont été les premiers à proposer un contrôle dynamique des paramètres d'un algorithme génétique utilisant un contrôleur flou [LT93a]. Par la suite, d'autres auteurs ont proposé des méthodes de réglage basées sur ce même principe [XV94, HLV95b, HL96b, HL96a, HYFS96, WWSJ96].

Dans [LT93a], Lee et Takagi se sont basés sur les critères de performance en ligne et hors ligne définis par Grefenstette pour obtenir la structure du contrôleur flou. Ces mesures de performance sont exploitées par un méta-AG qui optimise certaines parties du contrôleur flou. L'algorithme génétique principal traite les fonctions de test de DeJong. La structure complète proposée par Lee et Takagi est représentée sur le synoptique de la figure 5.4.

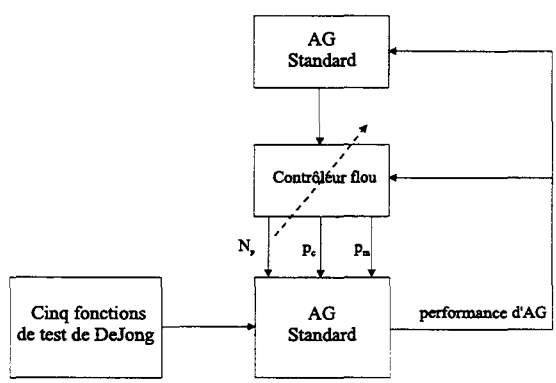
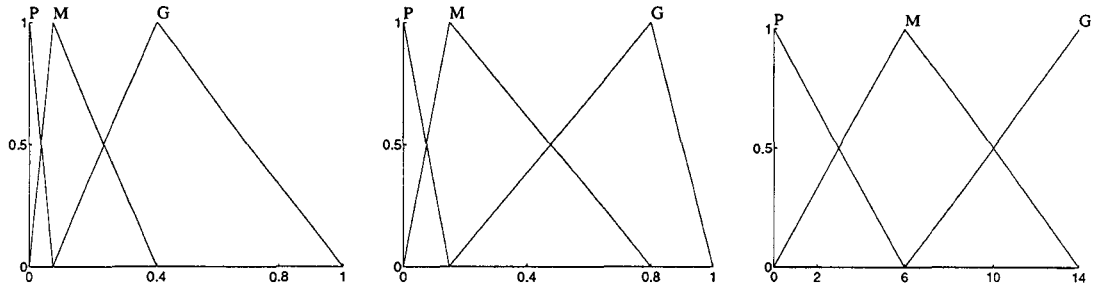


FIG. 5.4 : AG adaptatif décrit par Lee et Takagi

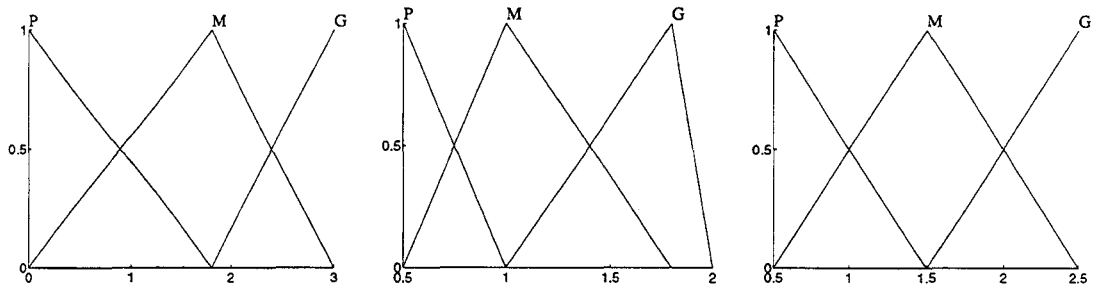
A l'entrée du contrôleur flou, on trouve trois mesures de performance évaluées sur la population traitée par l'algorithme génétique : $\overline{f}/f_{max}$, $f_{min}/\overline{f}$ et Δf_{max}. Δf_{max} est la différence entre la valeur maximale de la fonction d'évaluation pour la population de la génération t et la valeur maximale pour la population de la génération précédente $t-1$. Le contrôleur flou a également trois sorties qui définissent la variation de la taille de population ΔN_p, la variation de la probabilité de croisement Δp_c et de la probabilité de

mutation Δp_m. Les fonctions d'appartenance pour les entrées et les sorties, issues d'une optimisation par le méta-AG, sont représentées sur les figures 5.5 et 5.6.



(a) $\overline{f}/f_{max}$ (b) $f_{min}/\overline{f}$ (c) Δf_{max}

FIG. 5.5 : Fonctions d'appartenance pour les entrées



(a) ΔN_p (b) Δp_c (c) Δp_m

FIG. 5.6 : Fonctions d'appartenance pour les sorties

Les règles floues, définies également par le méta-AG à partir des fonctions de test de DeJong, sont représentées dans le tableau de la figure 5.7. Lee et Takagi utilisent le processus d'inférence max-prod dans le contrôleur flou, et la méthode de défuzzification COA.

D'autres auteurs ont suivi une approche similaire en exploitant d'autres critères de performance [XV94, HLV95b, HL96b]. Dans [WWSJ96], Wang et al. utilisent la variation moyenne de fonction d'évaluation de la population $\Delta f(t)$ et la différence $\Delta_2 f(t) = \Delta f(t) - \Delta f(t-1)$ entre la génération en cours et la génération précédente comme entrées du contrôleur flou. Hsu et al. [HYFS96] proposent une structure d'algorithme génétique parallèle, basée sur des opérateurs locaux «sub-exchange» et «sub-copy», dans laquelle tous les paramètres sont réglés par un contrôleur flou.

Herrera et Lozano [HL96b, HL96a] décrivent une méthode de réglage des paramètres par un contrôleur flou pour un algorithme génétique exploitant un codage par des nombres

règles	$\overline{f}/f_{max}$	$f_{min}/\overline{f}$	Δf_{max}	ΔN_p	Δp_c	Δp_m
1	P	P	P		P	P
2	P	P	M		M	
3	P	P	G	G	P	M
4	P	M	P	G		
5	P	M	M	P	G	M
6	P	M	G	G		G
7	P	G	P	P	M	M
8	P	G	M		P	
9	P	G	G	M	P	
10	M	P	P	P	M	M
11	M	P	M	M	M	G
12	M	P	G	P		M
13	M	M	P	P		M
14	M	M	M	M	M	
15	M	M	G	G	P	
16	M	G	P		P	G
17	M	G	M			M
18	M	G	G			M
19	G	P	P	M		P
20	G	P	M		P	
21	G	P	G	P	G	G
22	G	M	P	G		
23	G	M	M	G	P	
24	G	M	G	G	P	G
25	G	G	P		M	
26	G	G	M			
27	G	G	G		M	P

FIG. 5.7 : Règles floues pour ΔN_p, Δp_c et Δp_m

réels. Ils ajoutent un critère de performance qui n'est pas calculé à partir de la fonction d'évaluation, mais à partir des valeurs codées dans les chromosomes. Le contrôleur flou traite ainsi deux entrées, $PD = f_{max}/\overline{f}$ et $ED = (\overline{d} - d_{min})/(d_{max} - d_{min})$, où $\overline{d}$, d_{min} et d_{max} représentent des distances entre individus. La distance entre deux individus est définie en considérant que chaque chromosome est un vecteur dont les coordonnées sont les valeurs codées par des nombres réels. $\overline{d}$ désigne la distance moyenne entre tous les individus de la population et l'individu pour lequel la fonction d'évaluation est maximale, d_{min} et d_{max} désignent respectivement le minimum et le maximum des distances entre tous les individus de la population et l'individu pour lequel la fonction d'évaluation est maximale. L'inconvénient principal de cette méthode est qu'elle ne peut s'appliquer que dans le cas d'un codage par des nombres réels.

5.4 Réglage dynamique pour chaque individu

Dans la méthode proposée par Lee et Takagi, les probabilités de croisement et de mutation réglées par le contrôleur flou sont utilisées pour faire évoluer de façon similaire tous les individus de la population. En effet, les critères d'évaluation qui constituent les entrées du contrôleur sont calculés à partir de tous les individus. De plus, l'optimisation du contrôleur flou est réalisée à partir des fonctions de test de DeJong, et non en utilisant les caractéristiques propres du problème traité par l'algorithme génétique.

Nous proposons une structure de réglage dynamique des paramètres d'un algorithme génétique dans laquelle les probabilités de croisement et de mutation sont ajustées pour chaque individu. Nous utilisons ainsi une méthode d'évaluation similaire à celle proposée par Srinivas, tout en exploitant les mesures de performance de façon plus fine grâce à un contrôleur flou. Ce principe de réglage dynamique est présenté sur la figure 5.8.

La première entrée du contrôleur est un critère évalué sur la population entière, $\overline{f}/f_{max}$, dont la valeur augmente lorsque l'algorithme génétique arrive en phase de convergence. La deuxième entrée est un critère évalué sur chaque individu, f/f_{max}, qui prend une valeur maximale pour le meilleur individu de la population. Le contrôleur flou dispose de deux sorties, qui règlent les probabilités de croisement et de mutation pour chaque individu.

Nous décrivons par la suite deux contrôleurs flous différents permettant d'assurer le

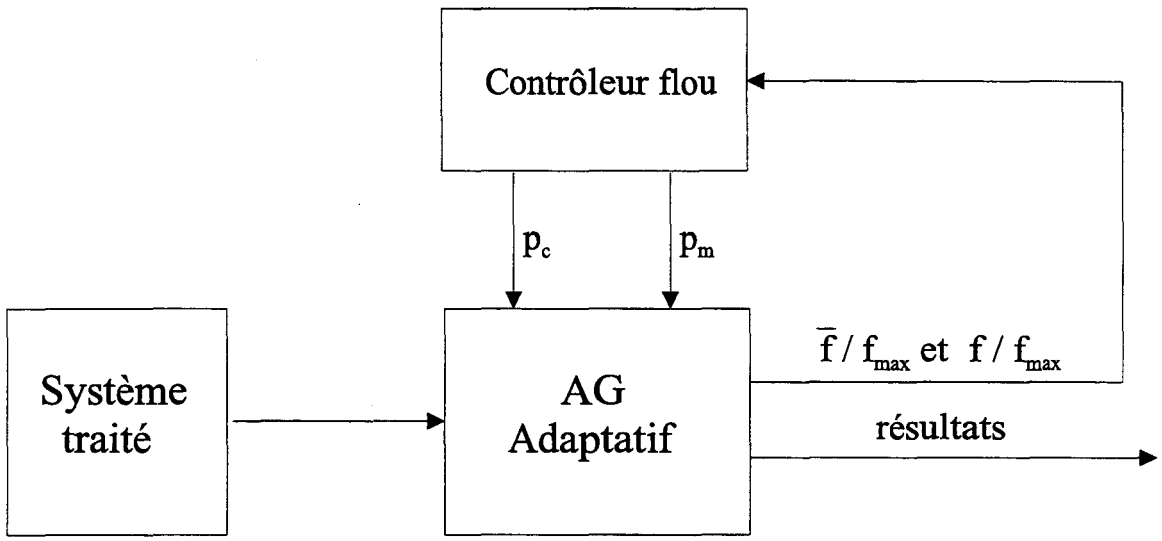


FIG. 5.8 : Réglage des paramètres pour chaque individu

réglage de l'algorithme génétique. Le premier a une structure fixe, que nous avons définie à partir de règles simples formalisant les connaissances décrites précédemment concernant le réglage dynamique des probabilités de croisement et de mutation. Le deuxième contrôleur flou proposé est défini en suivant la méthode de synthèse décrite dans le chapitre 3.

5.4.1 Contrôleur flou de structure fixe

Les deux entrées du contrôleur, $\overline{f}/f_{max}$ et f/f_{max}, sont qualifiées par deux partitions identiques de l'univers du discours $[0, 1]$, contenant trois termes (P, M, G) signifiant respectivement Petit, Moyen et Grand. Les fonctions d'appartenance de ces trois termes linguistiques sont représentées sur la figure 5.9. Les termes ont été choisis afin de constituer une partition uniforme de l'univers du discours, puisqu'on ne dispose pas d'information a priori sur l'évolution des deux critères.

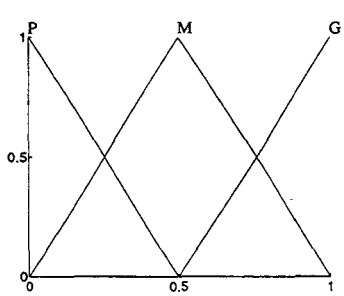
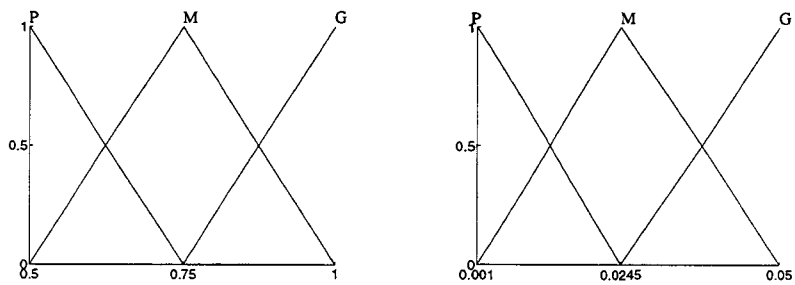


FIG. 5.9 : Fonctions d'appartenance des termes qualifiant $\overline{f}/f_{max}$ et f/f_{max}

Les sorties du contrôleur flou, définissant les probabilités p_c et p_m utilisées par l'algorithme génétique pour traiter un individu, sont qualifiées par trois termes linguistiques, notés également (P, M, G). La probabilité de croisement p_c prend ses valeurs dans l'intervalle $[0.5, 1]$, les fonctions d'appartenance des termes linguistiques correspondants sont représentées sur la figure 5.10(a). La probabilité de mutation, dont la valeur est située dans l'intervalle $[0.001, 0.05]$, est qualifiée par les termes linguistiques dont les fonctions d'appartenance sont présentées sur la figure 5.10(b).



(a) Termes linguistiques pour p_c (b) Termes linguistiques pour p_m

FIG. 5.10 : Fonctions d'appartenance des termes linguistiques des sorties

Les règles floues utilisées durant le processus d'inférence sont les mêmes pour les deux sorties. Ces règles sont présentées dans le tableau de la figure 5.11.

		$\overline{f}/f_{max}$		
	p_c ou p_m	P	M	G
f/f_{max}	P	G	G	G
	M	G	M	P
	G	M	P	P

FIG. 5.11 : Règles floues pour p_c et p_m

Ces règles sont issues d'une analyse des variations des deux critères pour un algorithme génétique standard. Par exemple, lors des dernières itérations, le premier critère $\overline{f}/f_{max}$ se rapproche de la valeur 1. Il est donc qualifié par le terme linguistique *Grand*. Si l'individu en cours de traitement par l'algorithme génétique est performant, le deuxième critère f/f_{max} prend également une valeur proche de 1, qualifiée par le terme *Grand*. Dans ce cas, pour tenter de conserver cet individu sans lui apporter de modification, on choisit des probabilités p_c et p_m qualifiées par *Petit* :

Règle 1 : Si ($\overline{f}/f_{max}$ est *Grand*) et (f/f_{max} est *Grand*) ;
alors (p_c est *Petit*), (p_m est *Petit*)

Dans le contrôleur de structure fixe, nous utilisons la méthode d'inférence de Mamdani, et la procédure de défuzzification par centre de région (COA). Les performances de l'algorithme génétique piloté par ce contrôleur flou seront décrites dans la dernière section de ce chapitre.

5.4.2 Contrôleur flou optimisé

Certaines parties du contrôleur flou décrit précédemment, comme les fonctions d'appartenance des entrées et des sorties, ont été définies de façon arbitraire. En effet, on ne dispose pas d'informations a priori sur les qualifications linguistiques décrivant les deux critères de performance. Pour déterminer au mieux le contrôleur flou permettant de piloter l'algorithme génétique utilisé dans notre application, nous utilisons la méthode de synthèse décrite dans le chapitre 3. Le principe de l'optimisation est présenté sur la figure 5.12. L'algorithme génétique situé en haut de la hiérarchie est utilisé uniquement durant la phase de synthèse du contrôleur flou. Durant le fonctionnement normal, il n'intervient plus dans le système.

5.4.2.1 Codage des paramètres du contrôleur flou

Les variables linguistiques utilisées pour décrire les deux entrées et les deux sorties sont toutes définies par trois termes linguistiques (P, M, G). Les fonctions d'appartenance des termes linguistiques P, M et G sont triangulaires. Chaque variable linguistique est donc codée en utilisant uniquement deux différences entre les maxima de deux fonctions d'appartenances successives, comme le montre la figure 5.13. Toutes les valeurs sont codées par des chaînes binaires de 8 bits, et regroupées dans un seul chromosome. Pour les deux entrées, la chaîne binaire code une valeur appartenant à l'intervalle $[0, 1]$. Pour la sortie fournissant la probabilité de croisement, la valeur codée appartient à l'intervalle $[0.5, 1]$ et pour la sortie correspondant à la probabilité de mutation, l'intervalle codé est $[0.001, 0.05]$.

Les codages sont présentés dans le tableau de la figure 5.4.2.1, ainsi que les conditions imposées sur les valeurs codées.

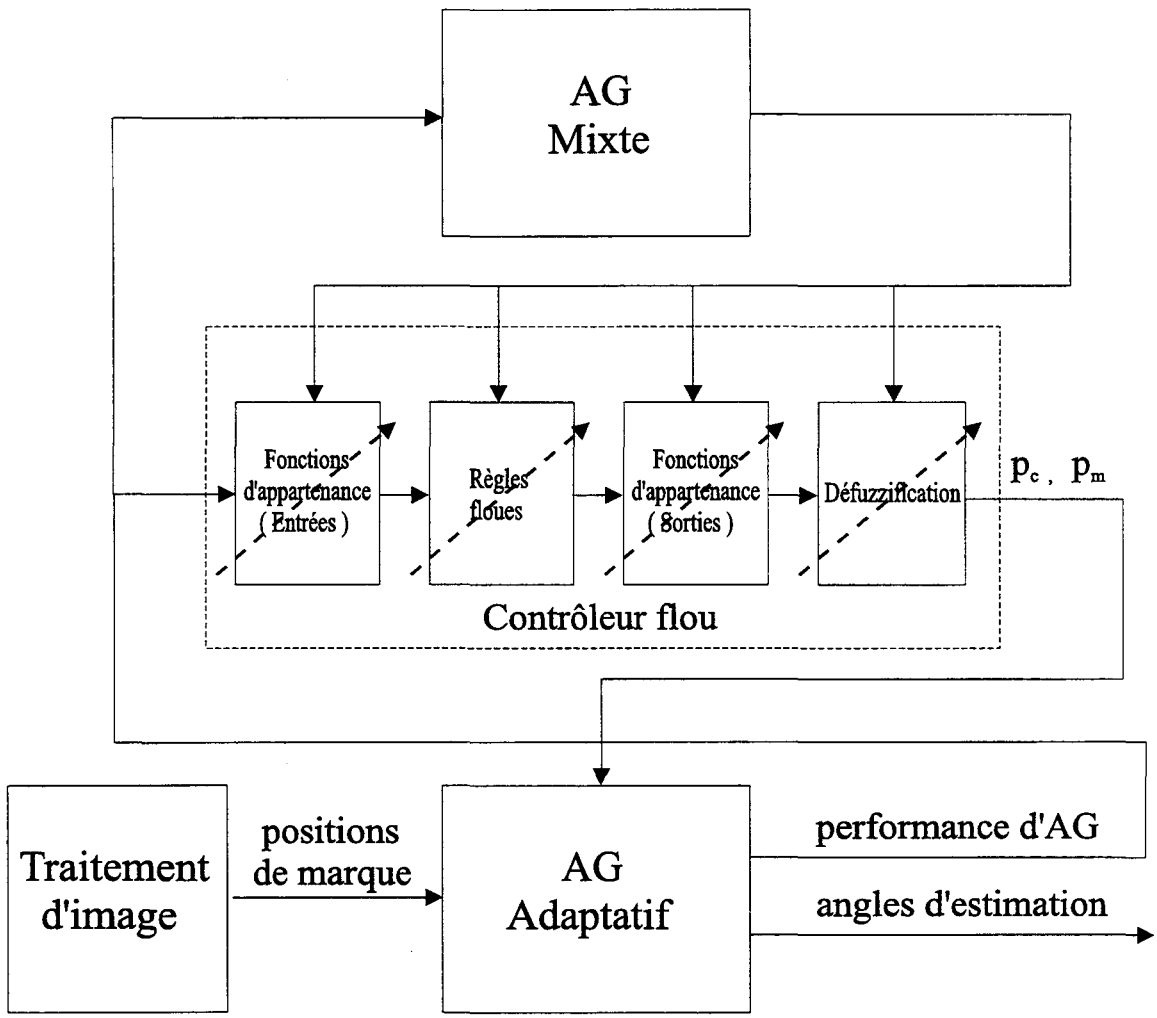


FIG. 5.12 : La structure hiérarchique permettant d'optimiser le système d'estimation

Les règles floues utilisées ont toutes deux prémisses et deux conclusions, et sont exprimées de la façon suivante :

$$\begin{aligned} \text{Règle } i_1, i_2 : \quad & \text{Si } (\overline{f}/f_{max} \text{ est } A_{i_1}) \text{ et } (f/f_{max} \text{ est } B_{i_2}) \\ & \text{alors } (p_c \text{ est } C_{j_1}), (p_m \text{ est } D_{j_2}) \\ & i_1 = 1, \cdots, 3 \qquad i_2 = 1, \cdots, 3 \end{aligned}$$

où les A_{i_1} et B_{i_2} désignent respectivement les termes linguistiques des entrées $\overline{f}/f_{max}$ et f/f_{max}, et C_{j_1} et D_{j_2} les termes linguistiques des sorties p_c et p_m. Les indices j_1 et j_2 varient donc entre 1 et 3.

Pour représenter les conclusions des règles floues, nous utilisons un codage en base 4 pour les sorties, ce qui permet de représenter par les valeurs $\{0, 1, 2, 3\}$ les conclusions possibles {-, P, M, G}, «-» signifiant la suppression de cette règle pour la sortie corres-

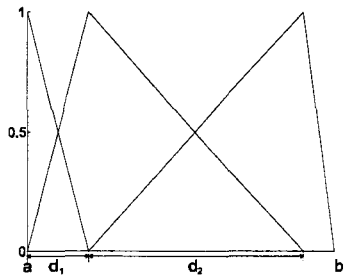


FIG. 5.13 : Codage des fonctions d'appartenance

Entrée	Codage	Condition
$\overline{f}/f_{max}$	$c_{8,[0,1]}(d_1)$ $c_{8,[0,1]}(d_2)$	$d_1 + d_2 \leq 1$
f/f_{max}	$c_{8,[0,1]}(d_3)$ $c_{8,[0,1]}(d_4)$	$d_3 + d_4 \leq 1$
Sortie	**Codage**	**Condition**
p_c	$c_{8,[0.5,1]}(d'_1)$ $c_{8,[0.5,1]}(d'_2)$	$d'_1 + d'_2 \leq 0.5$
p_m	$c_{8,[0.001,0.05]}(d'_3)$ $c_{8,[0.001,0.05]}(d'_4)$	$d'_3 + d'_4 \leq 0.049$

pondante. Les trois conclusions de la règle d'indices (i_1, i_2) sont ainsi représentées par le couple $(R^{p_c}_{i1,i2}, R^{p_m}_{i1,i2})$, les deux valeurs étant exprimées en base 4. Le codage de toutes les règles floues est donné par :

$R^{p_c}_{1,1}$	$R^{p_m}_{1,1}$	$R^{p_c}_{1,2}$	$R^{p_m}_{1,2}$	$\cdots$	$R^{p_c}_{3,3}$	$R^{p_m}_{3,3}$

Nous cherchons à optimiser la méthode de défuzzification M-SLIDE, pour laquelle nous codons le seul paramètre (noté β_d pour ne pas le confondre avec l'angle de roulis) dans un chromosome en nombres réels :

β_d

5.4.2.2 Phase de synthèse du contrôleur flou

Nous cherchons à optimiser le fonctionnement de l'ensemble (algorithme génétique + contrôleur flou) de telle sorte que le nombre d'itérations nécessaire à l'estimation des angles soit le plus faible possible. Pour cela, nous devons construire une fonction de coût favorisant les solutions menant à un nombre minimum d'itérations.

Les performances de l'algorithme sont évaluées en faisant fonctionner le système complet sur des séries d'images de synthèse, dans lesquelles les angles réels sont connus avec

précision. Nous avons utilisé un ensemble de 256 images de test, pour lesquelles les angles de lacet α et de roulis β sont compris entre 0 et 15 degrés. Pour chaque réglage possible du contrôleur flou, qui correspond à une solution possible du problème d'optimisation, nous estimons les angles sur toutes les images de synthèse.

Nous avons fixé à 400 le nombre maximal d'itérations réalisées pour une estimation des angles. Nous introduisons également un critère d'arrêt, en considérant que le processus est terminé lorsque l'erreur entre les angles estimés et les angles réels (qui sont connus puisqu'on utilise des images de synthèse) est inférieure à 0.2 degrés. En notant α_e et β_e les angles estimés par l'algorithme génétique, le critère d'arrêt est alors défini par les deux conditions :

$$\begin{aligned} |\alpha - \alpha_e| &\leq 0.2 \\ |\beta - \beta_e| &\leq 0.2 \ . \end{aligned} \tag{5.14}$$

Nous utilisons la fonction d'évaluation suivante :

$$f_{eval} = 400 - \frac{\sum_{\alpha=0}^{15} \sum_{\beta=0}^{15} gen(\alpha, \beta)}{256} \tag{5.15}$$

dans laquelle 400 correspond au nombre maximum d'itérations et $gen(\alpha, \beta)$ représente le nombre d'itérations nécessaires pour obtenir la valeur correcte des angles. Cette fonction évalue en fait la différence entre le nombre maximal d'itérations et le nombre moyen d'itérations qui ont été nécessaires pour estimer les angles sur toute la série d'images.

L'estimation de la performance pour un couple (p_c, p_m) par la fonction d'évaluation définie dans l'équation 5.15 nécessite de réaliser un certain nombre d'itérations pour chacune des 256 images synthétiques. Le temps de calcul, même sur un ordinateur très performant, n'est pas négligeable. Durant le processus complet d'optimisation, on doit réaliser cette mesure de performance pour les N_p individus de la population, et ce à chaque génération. En utilisant cette méthode, avec une taille de population de 20 individus durant 50 générations, il faut plusieurs jours de calcul sur un ordinateur compatible PC équipé d'un Pentium II à 400MHz ...

Pour diminuer le temps de calcul, nous modifions la fonction d'évaluation afin de ne prendre en compte que quelques images pour évaluer les performances obtenues avec un certain couple de paramètres. La nouvelle fonction d'évaluation est ainsi donnée par :

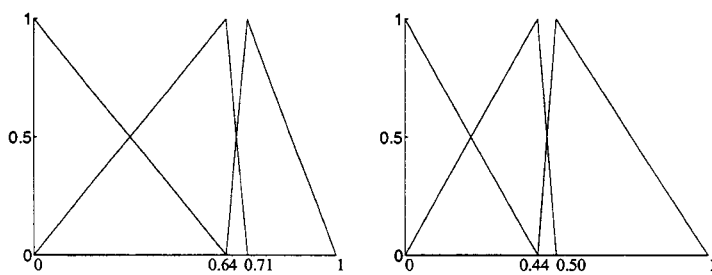
$$f_{eval} = 400 - \frac{\sum_{k=1}^{N_e} gen(\alpha(k), \beta(k))}{N_e} \ , \tag{5.16}$$

dans laquelle N_e désigne le nombre d'images sur lesquelles on évalue les performances du système et $\alpha(k)$ et $\beta(k)$ désignent deux valeurs exprimées en degrés tirées au sort dans l'intervalle $[0, 15]$ degrés pour chaque image servant durant l'évaluation. Cette fonction permet de diminuer le temps de calcul d'un facteur $256/N_e$ par rapport à la fonction d'évaluation initiale.

Pour le méta-AG qui assure la synthèse du contrôleur flou, nous avons utilisé les paramètres fixes suivants :

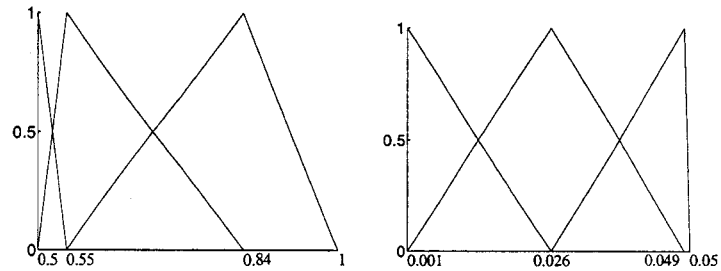
Méthode de sélection	roulette de casino standard
Population N_p	20
Probabilités de croisement	0.6
Probabilités de mutation	Chromosome binaire : 0.001 Chromosome base n : 0.001 Chromosome réel : 0.03

Après 80 itérations du méta-AG, nous obtenons le contrôleur flou défini par les éléments suivants. Les fonctions d'appartenance pour les entrées correspondant aux deux critères de performance sont présentées sur la figure 5.14, les fonctions d'appartenance des termes linguistiques qualifiant les sorties p_c et p_m sont représentées sur la figure 5.15, les règles floues sont décrites par le tableau de la figure 5.16 et le paramètre β_d de la procédure de défuzzification M-SLIDE vaut $0,278$.



(a) $\overline{f}/f_{max}$ (b) f/f_{max}

FIG. 5.14 : Les fonctions d'appartenance des entrées après optimisation



(a) p_c (b) p_m

FIG. 5.15 : Les fonctions d'appartenance des sorties après optimisation

		$\overline{f}/f_{max}$		
	p_c	P	M	G
f/f_{max}	P	G	M	M
	M	M	G	M
	G	P	P	P

(a) Règles floues pour p_c

		$\overline{f}/f_{max}$		
	p_m	P	M	G
f/f_{max}	P	G	G	P
	M	-	P	P
	G	G	G	P

(b) Règles floues pour p_m

FIG. 5.16 : Règles floues optimisées

5.5 Comparaison des algorithmes génétiques

Pour montrer l'efficacité du réglage dynamique des paramètres d'un algorithme génétique par un contrôleur flou, nous avons comparé les performances de quatre algorithmes génétiques différents : un algorithme génétique standard, l'AG adaptatif de Srinivas, l'AG piloté par un contrôleur flou de structure fixe et l'AG piloté par un contrôleur flou optimisé.

5.5.1 Choix des paramètres des algorithmes génétiques

Pour l'algorithme génétique standard, nous utilisons des probabilités fixées aux valeurs retenues par DeJong et une population constituée de 20 individus :

Population N_p	20
Probabilité de croisement	0.6
Probabilité de mutation	0.001

Pour l'algorithme génétique adaptatif décrit par Srinivas, nous utilisons les paramètres suivants :

Population N_p	20
Probabilité de croisement	$\begin{cases} (f_{max} - f)/(f_{max} - \overline{f}) & \text{si} \quad f' \geq \overline{f} \\ 1 & \text{si} \quad f' < \overline{f} \end{cases}$
Probabilité de mutation	$\begin{cases} 0.5 \cdot (f_{max} - f)/(f_{max} - \overline{f}) & \text{si} \quad f \geq \overline{f} \\ 0.5 & \text{si} \quad f < \overline{f} \end{cases}$

Pour les algorithmes génétiques pilotés par contrôleur flou, de structure fixe ou optimisé, les probabilités de croisement et de mutation sont ajustées par le contrôleur entre deux valeurs extrèmes. Nous utilisons les paramètres suivants :

Population N_p	20
Probabilité de croisement minimale	0.5
Probabilité de croisement maximale	1
Probabilité de mutation minimale	0.001
Probabilité de mutation maximale	0.05

5.5.2 Résultats de la comparaison

Les performances des quatre algorithmes sont comparées sur la base du nombre moyen d'itérations nécessaires pour l'évaluation d'une position angulaire. Les différents algorithmes sont testés sur la même série d'images de synthèse, correspondant aux 256 configurations possibles obtenues lorsque les deux angles α et β varient entre 0 et 15 degrés. Le nombre moyen d'itérations est présenté dans le tableau de la figure 5.17, ainsi que le rapport (nombre d'itérations de l'AG standard / nombre d'itérations de l'AG testé) qui correspond au gain de temps de calcul.

	AG standard	AG Srinivas	AG + CF fixe	AG + CF optimal
Itérations	203.906	165.656	74.875	52.625
Rapport	1	1.230	2.723	3.875

FIG. 5.17 : Comparaison des nombres moyens d'itérations

On constate que les algorithmes génétiques dont les paramètres sont réglés par un contrôleur flou sont au moins deux fois et demi plus rapides qu'un algorithme génétique

standard. La meilleure performance en termes de temps de calcul est obtenue avec l'algorithme génétique réglé dynamiquement par un contrôleur flou dont la structure a été optimisée, qui est environ quatre fois plus rapide qu'un algorithme génétique standard.

Nous avons également comparé les performances en termes de précision sur la mesure des angles. Pour chaque image de synthèse, nous réalisons 400 itérations avec chacun des quatre algorithmes génétiques, puis nous retenons comme solution l'individu pour lequel la fonction d'évaluation est maximale. Nous avons calculé l'erreur moyenne $\bar{\epsilon}$ entre les valeurs estimées et les valeurs idéales sur les 256 images de synthèse pour les quatre algorithmes, ainsi que l'erreur maximale ϵ_{max}. Les erreurs sur l'estimation de l'angle de roulis β sont présentées dans le tableau de la figure 5.18.

	AG standard	AG Srinivas	AG + CF fixe	AG + CF optimal
$\bar{\epsilon}$	0.1144	0.0805	0.0563	0.0428
ϵ_{max}	2.0000	0.2610	0.3193	0.1989

FIG. 5.18 : Comparaison des erreurs moyennes et maximales

On constate que les trois algorithmes génétiques dont les paramètres sont réglés dynamiquement ont des performances similaires en termes de précision. Par contre, l'erreur maximale pour l'algorithme génétique standard est de 2 degrés, ce qui correspond à une image de synthèse pour laquelle l'estimation n'a pas été obtenue au bout des 400 itérations.

5.6 Conclusion

Dans ce chapitre, nous avons décrit une application de la méthode de synthèse d'un contrôleur flou présentée dans le chapitre 3. Le contrôleur flou assure le réglage dynamique des probabilités de croisement et de mutation d'un algorithme génétique. Dans notre application, l'algorithme génétique est utilisé pour estimer les trois angles définissant la position angulaire d'une remorque à partir de marques détectées par un système de vision.

L'algorithme génétique dont les paramètres sont ajustés dynamiquement par un contrôleur flou est à peu près trois fois et demi plus rapide qu'un algorithme génétique standard. Pour l'application présentée dans ce chapitre, ce gain de temps de calcul nous permet d'en-

visager l'utilisation de la méthode d'estimation dans un contexte de traitement rapide des images.

Conclusion générale et perspectives

Dans cette thèse, nous avons présenté une méthode de synthèse d'un contrôleur flou par un algorithme génétique qui optimise simultanément :

- Les fonctions d'appartenance des termes linguistiques permettant la qualification des entrées du contrôleur,
- Les règles floues intervenant dans le processus d'inférence,
- Les fonctions d'appartenance des termes linguistiques décrivant l'état des sorties,
- Les paramètres définissant la méthode de défuzzification.

Dans le troisième chapitre de ce mémoire, nous avons présenté en détails cette méthode de synthèse. Nous avons tout d'abord décrit le principe de codage mixte des paramètres du contrôleur flou dans un seul individu caractérisé par trois chromosomes dont l'un est codé en binaire, l'autre en base n et le dernier en nombres réels. Chaque chromosome regroupe toutes les informations qui peuvent être représentées en utilisant un même mode de codage. Nous avons ensuite présenté les opérateurs génétiques permettant à l'algorithme d'exploiter le codage mixte.

Dans le quatrième chapitre, nous avons présenté une première utilisation de cette méthode de synthèse d'un contrôleur flou. L'application décrite est le réglage dynamique des gains d'un régulateur standard de type PID. L'algorithme génétique nous a permis de définir au mieux la structure du contrôleur flou qui assure ce réglage dynamique. Les performances du système asservi, analysées par l'intermédiaire de critères calculés sur sa réponse à l'échelon, permettent de définir la fonction d'évaluation utilisée par l'algorithme génétique durant la phase d'optimisation.

Le cinquième chapitre est consacré à la description d'une application dans laquelle le contrôleur flou assure le réglage dynamique des paramètres d'un algorithme génétique. Cet algorithme génétique est utilisé pour exploiter les données issues d'un système de

vision afin de déterminer la position angulaire d'un objet observé par deux caméras. Les performances de l'algorithme génétique sont mesurées selon deux critères, à savoir le nombre d'itérations permettant d'obtenir une estimation de la position, et la précision obtenue sur les valeurs estimées.

Sur la base de ces deux exemples, nous pouvons proposer une méthode relativement générale permettant de synthétiser un contrôleur flou dont la fonction est de régler dynamiquement les paramètres d'un système. La spécification du contrôleur flou est réalisée en deux étapes. Durant la première étape, une analyse du système permet de définir les entrées et les sorties du contrôleur flou ainsi que plusieurs critères de performance du système qui seront utilisés soit durant la phase de synthèse (performance hors ligne), soit durant le fonctionnement (performance en ligne). Durant la deuxième étape, la méthode de synthèse proposée dans cette thèse permet de définir les différents éléments du contrôleur flou.

Phase d'analyse du système

Lorsqu'on dispose d'un modèle précis du fonctionnement d'un système, on utilise en général cette connaissance a priori pour régler ses paramètres de manière à optimiser ses performances. Cependant, dans de nombreuses situations pratiques, on ne connaît pas avec suffisamment de précision ce fonctionnement. Si on souhaite régler dynamiquement les paramètres d'un tel système, en suivant la méthode que nous proposons, on doit analyser les performances du système puisque le réglage dynamique a comme objectif de le maintenir constamment dans les meilleures conditions de fonctionnement. Les performances sont mesurées de deux façons différentes.

On doit tout d'abord définir des critères de performance *en ligne* qui permettent de qualifier à tout instant l'état de fonctionnement du système. Ces critères, qui peuvent être de simples mesures ou des qualifications exprimées de façon symbolique, sont exploités par le contrôleur flou afin de régler dynamiquement les paramètres. Dans l'exemple du chapitre quatre, la performance en ligne du PID est mesurée par deux valeurs, l'erreur ϵ entre la sortie du processus et la consigne, et la dérivée temporelle $\Delta\epsilon$ de cette erreur. Dans le chapitre cinq, la performance en ligne de l'algorithme génétique est mesurée par deux critères, à savoir l'adaptation moyenne de la population traitée et l'adaptation relative de

l'individu en cours de traitement.

On doit ensuite définir des critères de performance *hors ligne*, qui permettront de spécifier un contrôleur flou de structure optimale. Durant la phase de synthèse du contrôleur flou, l'algorithme génétique exploite ces critères de performance hors ligne, par le biais d'une fonction dévaluation, afin de définir au mieux la stratégie de réglage dynamique. Dans le chapitre quatre, nous avons retenu comme critères de performance hors ligne différents indices calculés sur la réponse à un échelon du couple PID + système asservi. Dans l'application décrite dans le chapitre cinq, les performances hors ligne de l'algorithme génétique sont mesurées en tenant compte uniquement du nombre d'itérations permettant d'obtenir une estimation de la position angulaire.

En résumé, a l'issue de cette première étape, les caractéristiques suivantes doivent être clairement spécifiées :

- Les sorties du contrôleur flou, qui correspondent aux paramètres du système devant être réglés dynamiquement.
- Les entrées du contrôleur flou, qui correspondent à un certain nombre de critères de performance en ligne.
- Un ou plusieurs critères de performance hors ligne, qui permettront de synthétiser le contrôleur flou.

Phase de synthèse du contrôleur flou

Durant cette phase, les différentes parties du contrôleur flou sont spécifiées grâce au procédé d'optimisation par algorithme génétique décrit dans le chapitre trois. Pour ce faire, on doit tout d'abord définir les codages utilisés pour représenter tous les paramètres du contrôleur que l'on souhaite optimiser. Cela ne doit théoriquement poser aucun problème, puisque le processus d'optimisation que nous avons décrit peut traiter des paramètres codés en binaire, en base n, ou par des nombres réels. Pour définir assez simplement le mode de codage, on peut utiliser les quelques règles suivantes :

- Le processus d'optimisation permet de définir au mieux les fonctions d'appartenance des termes qualifiant les entrées lorsque celles-ci correspondent à des mesures réalisées sur le système. Le choix d'un type de codage est guidé principalement par la

précision souhaitée lors de la prise en compte d'une entrée : codage par une chaîne binaire de longueur fixe pour une précision modérée, codage par un nombre réel pour une précision élevée. Lorsqu'un critère de performance en ligne est exprimé sous forme symbolique, l'entrée du contrôleur flou correspondante est utilisée directement par le processus d'inférence, aucun codage n'est alors nécessaire.

- Les règles floues sont également définies par l'intermédiaire du processus d'optimisation. Si certaines règles sont connues a priori, elles peuvent être utilisées directement. Dans ce cas, elles ne sont pas codées dans les individus définissant une solution possible du problème d'optimisation. Pour les règles devant être définies par le processus d'optimisation, on utilise un codage en base n, la base étant définie en fonction du nombre de termes linguistiques décrivant les conclusions des règles.
- La synthèse par algorithme génétique permet également de définir les fonctions d'appartenance des termes linguistiques qualifiant les sorties, qui correspondent aux paramètres du système. Lorsqu'une sortie prend uniquement un certain nombre de valeurs connues a priori, on ne définit pas de variable linguistique pour la qualifier et aucun codage n'est alors nécessaire. Sinon, le codage des paramètres des termes linguistiques est similaire à celui utilisé pour les entrées.
- L'algorithme génétique permet d'optimiser la méthode de défuzzification des conclusions issues du processus d'inférence. Plusieurs méthodes de défuzzification peuvent être utilisées, qui dépendent d'un nombre plus ou moins important de paramètres. En général, les paramètres, dont les valeurs doivent être obtenues avec précision, sont codés par l'intermédiaire de nombres réels.

Avant d'entamer le processus d'optimisation, on doit également définir la fonction d'évaluation utilisée dans l'algorithme génétique. Cette fonction, qui permet de tester l'efficacité d'une solution possible, est construite à partir des critères de performance hors ligne sélectionnés durant l'étape d'analyse du système. Comme nous l'avons déjà indiqué dans le chapitre cinq, cette fonction doit satisfaire principalement deux conditions :

- Elle doit permettre d'évaluer *toutes* les solutions, y compris les «mauvaises solutions». Il ne faut pas que cette fonction devienne identiquement nulle pour les individus les moins performants.

- Elle doit permettre de séparer précisément les «bonnes» des «très bonnes» solutions. En d'autres termes, cette fonction doit présenter des maxima très marqués.

Perspectives

Nous pensons que le travail décrit dans cette thèse peut se poursuivre principalement selon deux directions. Une première voie de recherche possible est l'amélioration de la méthode de synthèse du contrôleur flou. D'autre part, afin de valider complètement cette méthode, on doit tenter de l'appliquer pour résoudre d'autres problèmes de réglage dynamique des paramètres.

Amélioration de la méthode de synthèse

Plusieurs étapes de la méthode de synthèse décrite dans le chapitre trois peuvent certainement être améliorées. Par exemple, lors de la définition du contrôleur flou, de nombreuses caractéristiques doivent être choisies de façon intuitive : nombre de critères de performances mesurés sur le système (nombre d'entrées), nombre de termes linguistiques qualifiant les entrées ou les sorties.

Concernant ce dernier point, nous avons prévu dans le codage des paramètres du contrôleur flou une méthode pour supprimer éventuellement des termes linguistiques, ce qui permet théoriquement d'éliminer des termes inutiles. Sur les différents exemples que nous avons traités, cette possibilité n'a jamais été exploitée par le processus d'optimisation. Nous avons cependant souvent constaté que certains termes linguistiques deviennent inutiles dans la pratique car leur fonction d'appartenance prend des valeurs non nulles sur un intervalle très limité de l'univers du discours. Leur élimination totale simplifierait la structure du contrôleur flou sans incidence majeure sur les performances.

Nous pensons qu'il faudrait également préciser la façon de faire intervenir des connaissances a priori dans le processus de synthèse. Lorsque des règles sont connues par avance, nous n'en tenons pas compte dans le codage, ce qui fait que le processus optimise uniquement les règles initialement inconnues. Parfois, on aboutit à des contradictions entre les règles définies a priori et les règles issues de la synthèse. Pour éviter ce problème, on pourrait par exemple faire intervenir les règles connues dans la fonction d'évaluation, ce qui

permettrait de défavoriser les règles contradictoires durant le processus d'optimisation.

Autres applications

La méthode de synthèse décrite dans ce travail peut être utilisée dans de nombreuses applications dans lesquelles les traitements dépendent du choix de nombreux paramètres. En particulier, dans le domaine du traitement des images numériques, on sait que les algorithmes utilisent souvent des valeurs fixées par l'opérateur ou le programmeur : seuils, coefficients de pondération, etc.

A court terme, nous envisageons d'utiliser le principe de réglage dynamique présenté dans ce travail pour ajuster les paramètres de la méthode de modélisation par contours actifs. De nombreuses variables régissent l'évolution d'un contour actif durant le processus de minimisation de son énergie : coefficients de tension, de torsion, de prise en compte de l'énergie externe, ... La précision obtenue sur la position du contour à la fin du processus itératif, ainsi que le nombre d'itérations permettant d'aboutir à ce résultat, dépendent fortement du choix des paramètres. La méthode de réglage décrite dans cette thèse semble très adaptée à la résolution de ce problème.

Une autre application possible est le réglage des coefficients d'un filtre de détection de contours. Une technique originale de détection de contours a été proposée en 1996 par Wan, chercheur au Laboratoire I3D. Les contours sont mis en évidence par un filtre récursif dont les performances en termes de précision et de localisation sont ajustées par l'intermédiaire de deux paramètres α et β. Comment choisir au mieux ces deux coefficients en fonction du voisinage de l'image en cours de traitement ? La méthode décrite dans ce mémoire devrait nous permettre d'aborder ce problème.

Table des figures

Bibliographie

[ÅH88] K.J. Åström et T. Hägglund. *Automatic tuning of PID controllers.* Instrument Society of America, USA, 1988.

[ÅH89] K.J. Åström et T. Hägglund. An industrial adaptive PID controllers. Dans *Proc. 1989 IFAC Symp. Adaptive System in Control and Signal Processing*, pages 283–298, 1989.

[AMM94] J. Arabas, Z. Michalewicz, et J. Mulawka. GAVaPS - a genetic algorithm with varying population size. Dans *Proc. of the First IEEE Conference on Evolutionary Computation*, pages 73–78, 1994.

[AMT97] J.T. Alander, G. Moghadampour, et P. Törmänen. Evaluating the benefit of fuzzy logic for PID-control by means of genetic algorithms-case : frequency. Dans *Proc. of the Third Nordic Workshop on Genetic Algorithms and their Applications(3NWGA)*, pages 321–331, 1997.

[AT94] P. Andrey et P. Tarroux. Unsupervised image segmentation using a distributed genetic algorithm. *Pattern Recognition*, 27(5) :659–673, 1994.

[Bäc92] T. Bäck. Self-adaption in genetic algorithms. Dans *Proc. of the First European Conference on Artificial Life*, pages 263–271, 1992.

[Bäc96] T. Bäck. *Evolutionary Algorithms in the Theory and Practice.* Oxford University Press, 1996.

[BdM95] M.A.D. Bayens et M.J.G. Van de Molengraft. An approach to the design of a fuzzy self-tuning PID controller. *Journal A*, 36(2) :37–44, 1995.

[BM93] B. Bouchon-Meunier. *La logique floue.* Presses Universitaires de FRANCE, 1993.

[BM95] B. Bouchon-Meunier. *La logique floue et ses applications.* Addison-Wesley France, 1995.

[BZP94] S.M. Bhandarkar, Y. Zhang, et W.D. Potter. An edge detection technique using genetic algorithm-based optimization. *Pattern Recognition*, 27 :1159–1180, 1994.

[Cas95] J.L. Castro. Fuzzy logic controllers are universal approximators. *IEEE Trans. on Systems, Man, and Cybernetics*, 25(4) :629–635, 1995.

[CH96] O. Cordón et F. Herrera. A hybrid genetic algorithm-evolution strategy process for learning fuzzy logic controller knowledge bases. *Genetic Algorithms and Soft Computing*, pages 251–278, 1996.

[Che96] C.H. Chen. *Fuzzy logic and neural network handbook.* McGraw-Hill, 1996.

[Dav89] L. Davis. Adapting operator probabilities in genetic algorithms. Dans *Proceedings of the third Int'l Conf. on Genetic Algorithms*, pages 61–69, 1989.

[Dav91] L.D. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 1991.

[Dor74] R.C. Dorf. *Modern Control Systems.* Addison-Wesley,Reading,MA., 1974.

[DP94] D. Dubois et H. Prade. Principes de base de la théorie des ensembles flous et enjeux liés à la logique floue. Dans *Les Perspectives et Applications Industrielles de la Logique Floue en Europe*, 1994.

[DUC96] K.K. Delibasis, P.E. Undrill, et G.G. Cameron. Genetic algorithm implementation of stack filter design for image restoration. Dans *IEE Proc. Vision Image Signal Processing*, pages 177–183, 1996.

[ES93] L.J. Eshelman et J.D. Schaffer. Real-coded genetic algorithms and interval schemata. Dans *Foundations of Genetic Algorithms 2*, pages 187–202, 1993.

[FNU95] T. Furuhashi, K. Nakaoka, et Y. Uchikawa. An efficient finding of fuzzy rules using a new approach to genetic based machine learning. Dans *Proceedings of the Fourth IEEE International Conference on Fuzzy Systems*, volume 2, pages 715–722, 1995.

[FY91] D.P. Filev et R.R. Yager. A generalized defuzzification method via BAD distributions. *Int. J. Intell. Syst.*, 6 :687–697, 1991.

[FY94] D.P. Filev et R.R. Yager. On the analysis of fuzzy logic controllers. *Fuzzy Sets and Systems*, 68 :39–66, 1994.

[GAF95] R.N. Greenwell, J.E. Angus, et M. Finck. Optimal mutation probability for genetic algorithms. *Mathl. Comput. Modelling*, 21(8) :1–11, 1995.

[GF95] S. Galichet et Laurent Foulloy. Fuzzy controllers : Synthesis and equivalences. *IEEE Trans. on Fuzzy Systems*, 3(2) :140–148, 1995.

[GKD89] D.E. Goldberg, B. Korb, et K. Deb. Messy genetic algorithms motivation, analysis and first results. *Complex Systems*, 3 :493–530, 1989.

[GKD91] D.E. Goldberg, B. Korb, et K. Deb. Don't worry, be messy. Dans *Proceedings of the Fourth IEEE International Conference on Fuzzy Systems*, pages 24–30, 1991.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, New York, 1989.

[Gre86] J.J. Grefenstette. Optimization of control parameters for genetic algorithm. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-16(1) :122–128, 1986.

[GY95] J.K. George et B. Yuan. *Fuzzy Sets and Fuzzy Logic : Theory and Applications.* Englewood Cliffs, N.J. Prentice-Hall, 1995.

[HÅH91] C.C. Huang, K.J. Åström, et W.K. Ho. Refinements of the ziegler-nichols tuning formula. *IEE Proc. Part D*, 138 :111–118, 1991.

[HL96a] F. Herrera et M. Lozano. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. Dans *Genetic Algorithms and Soft Computing*, pages 95–125, 1996.

[HL96b] F. Herrera et M. Lozano. Adaptive genetic algorithms based on fuzzy techniques. Dans *Information Proceedings and Management of Uncertainly in Knowledge-Based Systems*, pages 775–980, 1996.

[HL97] III H.W. Lewis. *The Foundations of Fuzzy Control.* Plenum Press, New York and London, 1997.

[HLTT93] T. Hessburg, M.A. Lee, H. Takagi, et Tomizuka. Automatic design of fuzz-systems using algorithms and its application to lateral vehicle guidance. Dans

Applications of Fuzzy Logic Technology, volume SPIE-2061, pages 452–463, 1993.

[HLV95a] F. Herrera, M. Lozano, et J.L. Verdegay. Generating fuzzy rules from examples using genetic algorithms. Dans *Fuzzy Logic and Soft Computing*, pages 11–20. World Scientific, Singapore, 1995.

[HLV95b] F. Herrera, M. Lozano, et J.L. Verdegay. Tacking fuzzy genetic algorithms. Dans *Genetic Algorithms in Engineering and Computer Science*. John Wiley and Sons, 1995.

[HLV95c] F. Herrera, M. Lozano, et J.L. Verdegay. Turning fuzzy logic controllers by genetic algorithms. *International Journal of Approximate Reasoning*, 12 :299–315, 1995.

[HLV95d] F. Herrera, M. Lozano, et J.L. Verdegay. The use of fuzzy connectives to design real-coded genetic algorithms. *Mathware and Soft Computing*, pages 239–251, 1995.

[HLV97] F. Herrera, M. Lozano, et J.L. Verdegay. A learning progress for fuzzy control rules using genetic algorithms. *Fuzzy Sets and Systems*, page To appear, 1997.

[HM91] J. Hesser et R. Männer. Towards an optimal mutation probability in genetic algorithms. Dans *Parallel Problem Solving from Nature*, pages 23–32, 1991.

[HM95a] A. Homaifar et Ed. McCormik. Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Trans. on Fuzzy Systems*, 3(2) :129–139, 5 1995.

[HM95b] R. Horaud et O. Monga. *Vision par Ordinateur*. HERMES, 2^{nd} edition, 1995.

[Hol75] J.H. Holland. *Adaptation in natural and artificial systems*. Ann Arbor : The University of Michigan Press, 1975.

[HP95] F. Hoffmann et G. Pfister. A new learning method for the design of hierarchical fuzzy controllers using messy genetic algorithms. Dans *IFSA'95*, 1995.

[HTXW93] S.Z. He, S.H. Tan, F.L. Xu, et P.Z. Wang. PID self-tuning control using a fuzzy adaptive mechanism. Dans *Proc. 1993 IEEE International Conference on Fuzzy Systems*, pages 708–713, 1993.

[HYFS96] C.C. Hsu, S.I. Yamada, H. Fujikawa, et K. Shida. A fuzzy self-tuning parallel genetic algorithm for optimization. *Computers ind. Enging.*, 30 :883–893, 1996.

[Jan93] J.-S.R. Jang. ANFIS : Adaptive-network-based fuzzy inference systems. *IEEE Trans. Syst. Man and Cybern.*, 23(3) :665–685, 1993.

[JL96] T. Jiang et Y. Li. Generalized defuzzification strategies and their parameter learning procedures. *IEEE Trans. on Fuzzy Systems*, 4(1) :64–71, 1996.

[Jon75] K.A. De Jong. An analysis of the behavior of a class of genetic adaptive systems. ph. D Dissertation 76-9381, University of Michigan, 1975.

[Kar91a] C.L. Karr. Applying genetics to fuzzy logic. *AI Expert*, 6(3) :38–43, 1991.

[Kar91b] C.L. Karr. Design of an adaptive fuzzy linguistic controller using a genetic algorithm. Dans *Proc. of the Fourth Int. Conf. Genetic Algorithms*, pages 450–457, 1991.

[Kas93] N.K. Kasabov. Learning fuzzy production rules for approximate reasoning. Dans *Proc. Internat. Conf. on Artificial Neural Networks ICANN'93*, pages 337–342, 1993.

[Kas96] N.K. Kasabov. Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. *Fuzzy Sets and Systems*, 82 :135–149, 1996.

[KG93] C.L. Karr et E.J. Gentry. Fuzzy control of pH using genetic algorithms. *IEEE Trans. on Fuzzy Systems*, 1(1) :46–53, 2 1993.

[KGN85] J.B. Kiszka, M.M. Gupta, et P.N. Nikiforuk. Energetic stability of fuzzy dynamic systems. *IEEE Trans. on Syst. Man Cybern.*, SMC-15(5) :783–792, 1985.

[KGT95] R. Ketata, D.D. Geest, et A. Titli. Realization of PID controls by fuzzy control methods. *Fuzzy Sets and Systems*, 71 :113–129, 1995.

[KK97] D. Kim et C. Kim. Forecasting time series with genetic fuzzy predictor ensemble. *IEEE Trans. on Fuzzy Systems*, 5 :523–535, 1997.

[KM78] W.J.M. Kickert et E.H. Mamdani. Analysis of a fuzzy logic controller. *Fuzzy Sets and Systems*, 1 :24–44, 1978.

[Kos92] B. Kosko. Fuzzy systems as universal approximators. Dans *Proc. 1st IEEE Conference on Fuzzy Systems*, pages 1153–1162, 1992.

[Lei95] D.D. Leitch. *A New Genetic Algorithm for the Evolution of Fuzzy Logic.* PhD thesis, Oxford University, 1995.

[Lim95] C.M. Lim. Implementation of a fuzzy PID scheme for multi-variable process control. *Journal of the Institution of Engineers*, 35(1) :31–35, 1995.

[LK89] C.B. Lucasius et G. Kateman. Applications of genetic algorithms in chemometrics. Dans *Proceedings of the Third Int'l Conf. on Genetic Algorithms*, pages 170–176, 1989.

[LM94] J. Liska et S.S. Melsheimer. Complete design of fuzzy logic systems using genetic algorithms. Dans *Proceedings of the Third IEEE International Conference on Fuzzy Systems*, pages 1377–1382, 1994.

[LR95] W.L. Lo et A.B. Rad. Comparison of two auto-tuning predictive PI controllers. *International Journal of Modelling and Simulation*, 15(3) :98–106, 1995.

[LRG96] M.H. Lim, S. Rahardja, et B.H. Gwee. A GA paradigm for learning fuzzy rules. *Fuzzy Sets and Systems*, 82 :178–186, 1996.

[LS95] L. Lam et C.Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16 :945–954, 1995.

[LT93a] M.A. Lee et H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. Dans *Proceedings of the 5th Int'l Conf. on Genetic Algorithms*, pages 76–83, 1993.

[LT93b] M.A. Lee et H. Takagi. Embedding a priori knowledge into an integrated fuzzy system design method based on genetic algorithms. Dans *Proceedings of the 5th IFSA Congress*, pages 1293–1296, 1993.

[MA75] E.H. Mamdani et S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man Mach. Studies*, 7(1) :1–13, 1975.

[Mab93] S. Mabuchi. A proposal for a defuzzification strategy by the concept of sensitivity analysis. *Fuzzy Sets and Systems*, 55 :1–14, 1993.

[MC96] C.A. Murthy et N. Chowdhury. In search of optimal clusters using genetic algorithms. *Pattern Recognition Letters*, 17 :825–832, 1996.

[Mic92] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolutionary Programs.* Springer-Verlag, 2^{nd} edition, 1992.

[Miz95] M. Mizumoto. Realization of PID controls by fuzzy control methods. *Fuzzy Sets and Systems*, 70 :171–182, 1995.

[MTKH97] K.F. Man, K.S. Tang, S. Kwong, et W.A. Halang. *Genetic Algorithms for Control and Signal Processing.* Springer, 1997.

[NL94] K. Chwee Ng et Y. Li. Design of sophisticated fuzzy logic controllers using genetic algorithms. Dans *Proc. of ICCI94/Fuzzy Systems*, pages 1708–1712, 1994.

[Ost77] J.J. Ostergaad. Fuzzy logic control of a heat exchange process. *in Fuzzy Automata and Decision Processes, M.M. Gupta, G.N. Saridis, and B.R. Gaines, Eds.*, pages 285–320, 1977.

[RL95] A.B. Rad et W.L. Lo. Adaptive PID control of systems with unknown time delay. *Int. J. Systems Sci.*, 26(3) :619–638, 1995.

[SCED89] J.D. Schaffer, R.A. Caruna, L.J. Eshelman, et R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. Dans *Proceedings of the third Int'l Conf. on Genetic Algorithms*, pages 51–60, 1989.

[Sch97] P. Scheunders. A genetic c-means clustering algorithm applied to color image quantization. *Pattern Recognition*, 30 :859–866, 1997.

[SFH95] K. Shimojima, T. Fukuda, et Y. Hasegawa. Self-turning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm. *Fuzzy Sets and Systems*, 71 :295–309, 1995.

[SK88] M. Sugeno et G.T. Kang. Structure identification of fuzzy model. *Fuzzy Sets Syst.*, 28 :15–33, 1 1988.

[SL96] Q. Song et R.P. Leland. Adaptive learning defuzzification techniques and applications. *Fuzzy Sets and Systems*, 81 :321–329, 1996.

[SM84] M. Sugeno et K. Murakami. Fuzzy parking control of model car. *in 23rd IEEE Conf. on Decision and Control*, 1984.

[SP94] M. Srinivas et L. M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. on Systems, Man, and Cybernetics*, 24 :656–667, 4 1994.

[Sug85] M. Sugeno. *Industrial Application of Fuzzy Control.* Elsevier Science Pub. Co., 1985.

[Tam86] T. Tamakawa. High speed fuzzy controller hardware system. *Proc. 2nd Fuzzy System Symp.*, pages 122–130, 1986.

[Tam92] K.Y. Tam. Genetic algorithms, function optimization and facility layout design. *European Journal of Operational Research*, 63 :322–346, 1992.

[TH95] A. Toet et W.P. Hajema. Genetic contour matching. *Pattern Recognition Letters*, 16 :849–856, 1995.

[Thr91] P. Thrift. Fuzzy logic synthesis with genetic algorithms. Dans *Proc. of the Fourth Int. Conf. Genetic Algorithms*, pages 509–513, 1991.

[TM92] E.G. Talbi et T. Muntean. Evaluation et etude comparative d'algorithmes d'optimisation combinatoire : Application au problème de placement de processus. Dans *Rapport de Recherche de Laboratoire de Génie Informatique / Institut IMAG, Grenoble*, pages 1–36, 1992.

[TMC94] K.S. Tang, K.F. Man, et C.Y. Chan. Fuzzy control of water pressure using genetic algorithm. Dans *Proc IFAC Workshop on Safety, Reliability and Applications of Emerging Intelligent Control Technologies*, pages 15–20, 1994.

[TS83] T. Takaki et M. Sugeno. Derivation of fuzzy control rules from human operator's control action. Dans *Proc. of IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*, pages 55–60, Marseilles, France, 1983.

[TS85] T. Takaki et M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Syst. Man and Cybern.*, SMC15(1) :116–132, 1985.

[TW86] M. Togai et H. Watanabe. Expert system on a chip : An engine for real-time approximate reasoning. *IEEE Expert Syst. Mag.*, 1 :55–62, 1986.

[WCP97a] W. Wu, F. Cabestaing, et J.G. Postaire. Angular estimation by image analysis and genetic algorithm. In *Proceedings of the IEEE International Conference on Syst. Man and Cybern.*, 1997.

[WCP97b] W. Wu, F. Cabestaing, et J.G. Postaire. Estimation de la position angulaire d'un cube par analyse d'image et algorithmes génétiques. In *AGIS'97*, 1997.

[WCP97c] W. Wu, F. Cabestaing, et J.G. Postaire. Position estimation by image analysis and fuzzy genetic algorithms. In *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications*, pages 223–232, 1997.

[WK92] P. Wang et D.P. Kwok. Optimal fuzzy PID control based on genetic algorithms. In *Proc. of the 1992 Int. Conf. on Industrial Electronics, Control, and Instrumentation*, volume 2, pages 977–981, 1992.

[WM78] D. Willaeys et N. Malvache. Use of fuzzy model for process control. In *IEEE International Conference on Cybernetics and Society*, 1978.

[WMH77] D. Willaeys, N. Malvache, et P. Hammad. Utilization of fuzzy sets for systems modelling and control. In *IEEE International Conference on Decision and Control*, 1977.

[WMM97] D. Willaeys, P. Mangin, et N. Malvache. Use of fuzzy sets for systems modelling and control : Application to the speed control of a strongly perturbed motor. In *IFAC/IFIP International Conference on Digital Computer Applications to Process Control*, 1997.

[Wri91] A. Wright. Genetic algorithms for real parameter optimization. In *Foundations of Genetic Algorithms, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems*, pages 205–218, 1991.

[WWSJ96] P.Y. Wang, G.S. Wang, Y.H. Song, et A.T. Johns. Fuzzy logic controlled genetic algorithms. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, pages 972–979, 1996.

[XV94] H.Y. Xu et G. Vukovich. Fuzzy evolutionary algorithms and automatic robot trajectory generation. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 595–600, 1994.

[Wan92] L.X. Wang. Fuzzy systems are universal approximators. Dans *Proc. 1st IEEE Conference on Fuzzy Systems*, pages 1163–1169, 1992.

[WK92] P. Wang et D.P. Kwok. Optimal fuzzy PID control based on genetic algorithms. Dans *Proc. of the 1992 Int. Conf. on Industrial Electronics, Control, and Instrumentation*, volume 2, pages 977–981, 1992.

[WM78] D. Willaeys et N. Malvache. Use of fuzzy model for process control. Dans *IEEE International Conference on Cybernetics and Society*, 1978.

[WMH77] D. Willaeys, N. Malvache, et P. Hammad. Utilization of fuzzy sets for systems modelling and control. Dans *IEEE International Conference on Decision and Control*, 1977.

[WMM97] D. Willaeys, P. Mangin, et N. Malvache. Use of fuzzy sets for systems modelling and control : Application to the speed control of a strongly perturbed motor. Dans *IFAC/IFIP International Conference on Digital Computer Applications to Process Control*, 1997.

[Wri91] A. Wright. Genetic algorithms for real parameter optimization. Dans *Foundations of Genetic Algorithms, First Workshop on the Foundations of Genetic Algorithms and Classifier Systems*, pages 205–218, 1991.

[WWSJ96] P.Y. Wang, G.S. Wang, Y.H. Song, et A.T. Johns. Fuzzy logic controlled genetic algorithms. Dans *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems*, pages 972–979, 1996.

[XV94] H.Y. Xu et G. Vukovich. Fuzzy evolutionary algorithms and automatic robot trajectory generation. Dans *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 595–600, 1994.

[YF93a] R.R. Yager et D.P. Filev. On the issue of defuzzification and selection based on a fuzzy set. *Fuzzy Sets and Systems*, 55 :255–271, 1993.

[YF93b] R.R. Yager et D.P. Filev. Slide : A simple adaptive defuzzification method. *IEEE Trans. on Fuzzy System*, 1(1) :69–78, 1993.

[Yin94] H. Ying. Sufficient conditions on general fuzzy systems as function approximators. *Fuzzy Sets and Systems*, 68 :39–66, 1994.

[YOAH95] N. Yubazaki, M. Otani, T. Ashida, et K. Hirota. Dynamic fuzzy control method and its application to positioning of induction motor. Dans *Proc. of the Fourth IEEE International Conference on Fuzzy Systems*, volume 3, pages 1095–1102, 1995.

[YOTN87] R.R. Yager, S. Ovchinnikov, R.M. Tong, et H.T. Nouyen. *Fuzzy Sets and Application : Selected Paper by L.A. Zadeh.* J. Wiley, New York, 1987.

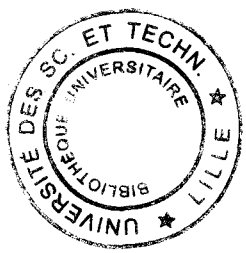
[Zad65] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8 :338–353, 1965.

[Zad68] L.A. Zadeh. Fuzzy algorithm. *Information and Control*, 12 :94–102, 1968.

[Zad94a] L.A. Zadeh. Fuzzy logic, neural networks and soft computing. *Comm. ACM*, pages 77–84, 3 1994.

[Zad94b] L.A. Zadeh. Soft computing and fuzzy logic. *IEEE Software*, 11(6) :48–56, 1994.

[ZTI93] Z.Y. Zhao, M. Tomizuka, et S. Isaka. Fuzzy gain scheduling of PID controllers. *IEEE Trans. on Systems, Man, and Cybernetics*, 23(5) :1392–1398, 1993.



Résumé

Dans cette thèse, nous présentons une méthode de synthèse d'un contrôleur flou basée sur une optimisation globale par un Algorithme Génétique. Tous les paramètres réglant le fonctionnement du contrôleur flou (fonctions d'appartenance, règles floues, défuzzification) sont optimisés simultanément.

Nous proposons une méthode de codage mixte, dans laquelle un individu est représenté par trois chromosomes décrivant les différents paramètres du contrôleur. Nous définissons les opérations génétiques de croisement et de mutation utilisées dans cet Algorithme Génétique.

Nous présentons deux exemples dans lesquels nous avons exploité cette méthode de synthèse. Dans ces deux applications, le contrôleur flou permet d'ajuster dynamiquement les paramètres d'un système complexe.

Le premier exemple décrit la synthèse d'un contrôleur flou utilisé pour modifier dynamiquement les coefficients d'un régulateur de type PID. Dans le deuxième exemple, le contrôleur flou ajuste les paramètres d'un Algorithme Génétique utilisé pour estimer la position angulaire d'un objet.

Mots clés : Synthèse d'un contrôleur flou, Algorithme Génétique, Réglage dynamique d'un PID, Estimation de position angulaire, Codage mixte

Abstract

In this work, we describe a method for designing a Fuzzy Logic Controller using global optimization by a Genetic Algorithm. All the parameters defining the Fuzzy Controller (membership functions, fuzzy rules, defuzzification) are optimized simultaneously.

We propose a mixed coding method, in which an individual is described by three chromosomes representing all the parameters of the Fuzzy Controller. We define the genetic operations of crossover and mutation used in this Genetic Algorithm.

We present two examples in which we used this method of controller design. In each application, the Fuzzy Logic Controller is used to dynamically adjust the parameters of a complex system.

The first example describes the synthesis of a Fuzzy Controller which modifies dynamically the coefficients of a PID regulator. In the second example, the Fuzzy Controller adjusts the parameters of a Genetic Algorithm used to estimate the angular position of an object.

Keywords : Fuzzy Logic Controller Design, Genetic Algorithm, Dynamic tuning of a PID, Angular Position Estimation, Mixed coding