ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DE LA PRODUCTION AUTOMATISÉE
M.ing.

PAR
THÉORÊT, Claude

ÉLABORATION D'UN LOGICIEL D'ENSEIGNEMENT ET D'APPLICATION DE LA LOGIQUE FLOUE DANS UN CONTEXTE D'AUTOMATE PROGRAMMABLE

MONTRÉAL, LE 16 AVRIL 2009

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Michel Rioux, directeur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M. Guy Gauthier, président du jury
Département de génie de la production automatisée à l'École de technologie supérieure

M. Mickaël Gardoni, membre du jury
Département de génie de la production automatisée à l'École de technologie supérieure

M. Guy Charbonneau, membre du jury externe
Département des Technologies du génie électrique au cégep du Vieux Montréal

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 27 MARS 2009

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Tout d'abord, je tiens à remercier mon directeur de recherche, M. Michel Rioux, pour le soutien, la confiance et la disponibilité dont il a toujours fait preuve à mon égard. Il a su garder ma motivation élevée, par de judicieux conseils et un engagement hors du commun.

Mes remerciements vont également aux membres du jury qui ont pris le temps nécessaire à l'évaluation et au jugement de mon travail.

Mes remerciements vont aussi à M. Guy Charbonneau du département des Technologies du génie électrique du cégep du Vieux Montréal pour sa collaboration et ses conseils éclairés.

Je remercie également le cégep du Vieux Montréal, pour avoir eu la gentillesse de me permettre d'utiliser les équipements du département des Technologies du génie électrique ainsi que son personnel pour sa collaboration.

Je remercie la direction de l'École de technologie supérieure, ainsi que mes collègues du service des enseignements généraux, et tout particulièrement le directeur du service, M. Luc Favreau, qui m'ont offert un contexte favorable à l'achèvement de ce mémoire.

Je remercie très chaleureusement mon ami Bruno Lamarche, pour son amitié sincère, pour ses conseils, et tous les services qu'il a pu me rendre durant ce mémoire.

Finalement, je tiens à remercier ma famille pour son soutien, et tout spécialement ma conjointe, Manon, qui m'a permis de réaliser ce mémoire grâce à ses encouragements, sa compréhension et son aide au quotidien. Je voudrais aussi remercier ma fille Marie-Jeanne et mon fils Louis-Simon pour les sacrifices et leur patience exprimée pendant tout le parcours de ce mémoire.

ÉLABORATION D'UN LOGICIEL D'ENSEIGNEMENT ET D'APPLICATION DE LA LOGIQUE FLOUE DANS UN CONTEXTE D'AUTOMATE PROGRAMMABLE

THÉORÊT, CLAUDE

RÉSUMÉ

La logique floue, ou plus généralement le traitement des incertitudes, a pour objet d'étude la représentation des connaissances imprécises et le raisonnement proche du langage humain de tous les jours. La logique floue permet d'obtenir une loi de commande souvent efficace, sans devoir faire appel à des développements théoriques importants. Elle présente l'intérêt d'incorporer des connaissances linguistiques sur la manière de piloter un processus difficile en prenant compte les expériences acquises par les utilisateurs et opérateurs du processus à commander. Plutôt que d'utiliser une approche traditionnelle fondée sur les lois de commande classique, on utilise des contrôles ayant une loi de commande basée sur les notions de la logique floue. Ces contrôleurs flous ont surtout démontré des performances plus robustes, par rapport aux systèmes traditionnels, dans les situations où le modèle mathématique du procédé était mal connu ou lorsque le comportement du procédé varie ou est non linéaire.

Malgré sa présence grandissante dans les applications industrielles, la logique floue est méconnue des techniciens qui œuvrent dans le domaine de la commande industrielle. Or, il n'existe pas de logiciel pédagogique pour l'apprentissage des notions de la logique floue. Il existe, certes, des logiciels professionnels pour la mise en œuvre des systèmes flous, par exemple Matlab®, mais rien qui ne préconise une approche pédagogique. Notre projet de recherche propose un logiciel d'enseignement et d'application de la logique floue dans un contexte d'automate programmable. Le logiciel permet l'apprentissage rapide des concepts de base de la logique floue. Il vise à montrer les techniques d'application issues de cette nouvelle technologie pour la conduite des procédés. Le logiciel permet l'interconnexion avec un automate programmable pour effectuer un contrôle en temps réel.

Un contrôleur à logique floue a été élaboré à l'aide du logiciel pour contrôler un procédé simulé et réel. Les résultats de simulation et d'expérimentation présentés démontrent bien les performances du contrôleur à logique floue. Des données expérimentales viennent valider le fonctionnement du logiciel proposé.

Mots-clés : Logique floue, contrôleur à logique floue, logiciel d'enseignement, Automate programmable, implantation.

DEVELOPMENT OF SOFTWARE FOR TEACHING AND APPLICATION OF FUZZY LOGIC IN THE CONTEXT OF PROGRAMMABLE LOGIC CONTROLLER

THÉORÊT, CLAUDE

ABSTRACT

Fuzzy logic, generally known as the treatment of uncertainties, aims to study the representation of imprecise knowledge and the reasoning akin to every day human language. Fuzzy logic allows to acquire a control law that is often efficient without requiring any significant theoretical development. It allows the integration of linguistic knowledge about the way to manage a complex process by taking into account the lessons learnt by the users and operators of the process to be controlled. Rather than using a traditional approach based on classical control laws, controls with a control law based on the notions of fuzzy logic are being utilized. These fuzzy controllers have primarily shown more robust performance than traditional systems in situations where the mathematical model of the process was unfamiliar but also when the behaviour of the process varies or is non-linear.

Despite its growing presence in many industrials applications, technicians working in the field of industrial control are often unfamiliar with fuzzy logic. Regrettably, there is no educational software on the market to teach about the concepts of fuzzy logic. Admittedly, there are several professional software available for the implementation of fuzzy systems, such as Matlab®, but nothing facilitating an educational approach. Our research project proposes a software for the purpose of teaching and applying fuzzy logic within the context of a programmable logic controller (PLC). The software allows one to rapidly learn the basic concepts of fuzzy logic. It also demonstrates the application techniques resulting from this new technology for the conduct of processes. In addition, it allows the interconnection with a PLC to perform real-time control.

A fuzzy controller was developed with the help of this software in order to control a simulated and real process. Indeed, the simulation and experimentation results demonstrate the performance of the fuzzy logic controller. Experimental data confirm the performance of the proposed software.

Keywords : Fuzzy logic, fuzzy logic controller, educational software, Programmable logic controller, implementation.

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

API	Automate Programmable Industriel
BR	Base des règles
CDG	Centre de gravité
CF	Contrôleur flou ou contrôle flou
DA	Direct Action
FLFB	Bloc de fonction à logique floue
FIS	*Fuzzy Inference System*
G_p	Gain statique du procédé
GS	*Gain Scheduling*
$G_{(s)}$	Fonction de transfert du procédé
h_i	Résultat de l'implication du calcul de la moyenne pondérée
$ht_{(A)}$	Hauteur de l'ensemble flou A
K_{dE}	Gain de la variation de l'erreur du contrôleur flou
K_E	Gain de l'erreur du contrôleur flou
K_p	Gain proportionnel
K_S	Gain de la sortie du contrôleur flou
LF	Logique floue
max	t-norme selon le modèle de Mamdani
min	t-conorme selon le modèle de Mamdani
M_D	Facteur de mise à l'échelle du contrôleur PD flou
M_i	Facteur de mise à l'échelle du contrôleur PI flou

$noy_{(A)}$	Noyau de l'ensemble flou A
OPC	***O**bject Linking and Embedding for **P**rocess **C**ontrol*
PD	Proportionnel Dérivé
PI	Proportionnel Intégral
PID	Proportionnel Intégral Dérivé
R_s	Signal de consigne d'un système de commande
SEF	Sous-ensemble flou
SIF	Système d'inférence floue
$supp_{(A)}$	Support de l'ensemble flou A
Te	Temps d'échantillonnage
TEI	Technologie de l'électronique industrielle
TSK	Takagi-Sugeno-Kang
t_d	Temps de délai du procédé
T_D	Temps de dérivation
T_i	Temps d'intégration
TGÉ	Technologies du génie électrique
U	Univers du discours
U_s	Signal de commande d'un système de commande classique
U_{sf}	Signal de commande d'un système de commande flou
y_0	Solution précise selon la méthode du centre de gravité
Y_s	Signal de mesure d'un système de commande
z_0	Solution précise d'une solution floue
z_{s0}	Solution précise d'une solution floue selon la méthode de Sugeno

e	Erreur entre la référence et le signal mesuré
Δe	Variation de l'erreur
$\Delta^2 e$	Accélération de l'erreur
φ_A	Fonction caractéristique de la variable A
μ_A	Fonction d'appartenance de la variable A
$\mu_{A(x)}$	Degré d'appartenance de x à fonction d'appartenance de la variable A
τ	Constante de temps du procédé
$\cap$	Opérateur d'intersection
$\cup$	Opérateur d'union
T	Norme triangulaire représentant la conjonction
$\perp$	Conorme triangulaire représentant la disjonction

INTRODUCTION

C'est en 1965, à l'université de Berkeley, que Lofti Zadeh a développé la logique floue (LF) en s'appuyant sur sa théorie mathématique des ensembles flous (Fuzzy set) pour modéliser les connaissances imprécises et incertaines (Borne, 1998). Dès cet instant, plusieurs chercheurs voient dans les travaux de Zadeh une nouvelle méthode technologique permettant la création de machines intelligentes. C'est le cas de Mamdani qui fut le premier, au début des années 1970, à employer la LF pour contrôler le fonctionnement d'une chaudière à vapeur (Borne, 1998; Sangalli, 2001).

Après un développement académique, c'est dans les années 1980 que les premières applications industrielles de cette nouvelle méthode technologique font leur apparition au Japon dans le domaine de la commande industrielle. Depuis cette époque, la méthode a gagné l'Europe et les États-Unis et a été mise en œuvre dans de nombreux projets industriels (Borne, 1998).

Les applications développées à base de LF ont démontré leur capacité à résoudre efficacement des problèmes dont les informations initiales disponibles manquaient de précisions (Bouchon-Meunier, Foulloy et Ramdani, 1998). Les récentes observations sur la croissance des applications basées sur la LF, ainsi que l'analyse des facteurs qui ont influé l'évolution de la méthode, nous confirment que cette méthode est prometteuse (Kosko, 1992).

Depuis les années 1990, la LF commence à être enseignée aux niveaux collégial et universitaire, dont des écoles d'ingénieurs (Borne, 1998). De plus, la LF est utilisée dans des outils d'aide à la décision pour des activités de recherche et développement (Bouchon-Meunier et Zadeh, 1995).

Il existe certains logiciels d'aide à la conception de systèmes basés sur la LF, mais ceux-ci sont destinés à un usage professionnel et permettent surtout de concevoir des systèmes très

complexes (systèmes experts). Ces logiciels s'adressent à des utilisateurs chevronnés, sont toutefois dispendieux et difficiles à apprendre. Ils sont également fondés sur l'implantation d'un modèle fermé, cachant les étapes de fonctionnement aux utilisateurs.

De plus, certains logiciels nécessitent l'apprentissage de concepts mathématiques avancés ou la maîtrise de notions avancées de programmation (Akcayol *et al.*, 2004). Finalement, certains nécessitent des équipements spécifiques pour leur mise en œuvre sur des procédés industriels. Pour toutes ces raisons, ces logiciels d'aide sont mal adaptés à un usage pédagogique où l'on veut démontrer comment appliquer les concepts de base de la LF. Conséquemment, l'enseignement de la portion de mise en œuvre de la méthode est plus ou moins abordée, ce qui pourrait expliquer la difficulté de la méthode à percer dans l'industrie.

Malgré la puissance de la méthode de la LF et sa forte croissance des dernières années, son utilisation au sein des entreprises demeure modeste. L'une des raisons à cela est que la méthode est méconnue par la plupart des techniciens et des ingénieurs sortant des institutions d'enseignement. Ceci s'explique par un manque d'outils pédagogiques efficaces, tant au niveau collégial (technologies du génie électrique) qu'universitaire (programmes de génie).

Ce projet vise deux objectifs. Tout d'abord, nous voulons promouvoir l'utilisation de la méthode de la LF dans le développement d'applications industrielles. Pour ce faire, nous désirons élaborer un logiciel ayant un environnement convivial permettant l'apprentissage rapide des concepts de base de la LF. Il servira à montrer les techniques d'application issues de cette nouvelle technologie pour la conduite des procédés.

Le deuxième objectif de ce projet est de faire connaître les contrôleurs flous, pour être en mesure de développer une application dans un automate programmable industriel (API). Pour atteindre ce deuxième objectif, nous réaliserons un contrôleur flou sous forme de bloc de fonction intégré dans un API. Nous traiterons du choix de la structure du contrôleur flou et de son réglage. Nous nous appuierons sur les outils de conception des contrôleurs PID classiques pour faciliter l'interprétation et la mise en œuvre du contrôleur flou (CF).

Ce projet trouve toute sa pertinence en comblant le manque entre les besoins de personnel qualifié dans ce domaine en ce qui concerne les industries, et l'offre des institutions d'enseignement. Ainsi, on peut s'attendre à un impact économique intéressant pour plusieurs entreprises en rendant disponible du personnel technique familier avec cette technologie et en pouvant augmenter sensiblement le rendement de multiples procédés industriels.

Pour atteindre nos objectifs, la principale hypothèse est la suivante. Nous pensons qu'un matériel pédagogique mieux adapté contribuerait significativement à réduire le temps d'apprentissage de la LF et de sa mise en œuvre. Par matériel mieux adapté, nous entendons un logiciel simple, interactif, ayant une gamme limitée de fonctionnalités et étant accompagné d'une documentation de soutien.

Le **chapitre 1** de ce mémoire consiste essentiellement à présenter le contexte général du projet. Le **chapitre 2** passe en revue la littérature sur la LF et les contrôleurs flous. Dans le **chapitre 3**, nous présentons une vue d'ensemble bibliographique allant de la définition à la mise en œuvre des systèmes flous en passant par ses concepts de base. Dans le **chapitre 4**, nous étudions les contrôleurs classique et flou dans un contexte de contrôle de procédé. Le contenu du **chapitre 5** traite essentiellement du logiciel d'apprentissage et de la mise en œuvre de la logique floue, *Fuzzy Kernel*. Il traite également du contrôleur flou de type PID développé dans le cadre de notre projet sous forme de bloc de fonction intégré dans un API. Le **chapitre 6** expose les procédés utilisés pour les tests de validation et de performance. Des résultats expérimentaux exprimant les performances obtenues sont présentés au **chapitre 7**. Finalement, la conclusion résume les principales contributions de notre projet.

CHAPITRE 1

MISE EN CONTEXTE GÉNÉRALE DU PROJET

Ce chapitre présente le contexte général du projet. Dans la première partie, nous présenterons les notions de base de la régulation de procédé et de la LF afin d'introduire le vocabulaire du domaine. Cette section se termine par les considérations pour la conception d'un CF. La seconde partie est consacrée à la présentation de la solution proposée. On y retrouve, en autres, les facteurs de succès, la contribution attendue et les limites de la solution.

1.1 Régulation de procédés

La régulation de procédé existe depuis plus de 2000 ans alors que les Grecs et les Babyloniens utilisaient ce principe pour réguler les niveaux d'eau dans leur canalisation (Godoy, 2007). La régulation de procédés industriels, quant à elle, est apparue au cours des années 1930 et c'est la compagnie « Taylor Instrument Companies » qui l'utilisa dans la commercialisation du « Fullscope », le premier contrôleur ayant de véritables fonctionnalités proportionnelle, intégrale et dérivée : le contrôleur PID. L'application de la régulation de procédé s'est étendue aujourd'hui aux secteurs manufacturiers : pharmaceutique, agro-alimentaire, robotique, électronique et bien plus encore (Kristiansson et Lennartson, 2006).

On appelle régulateur (ou contrôleur) la partie d'un système de commande comparant le signal de mesure (Y_s) avec le signal de consigne (R_s) afin de générer un signal de commande (U_s) approprié à l'intention de l'élément final (Figure 1.1). Le contrôleur peut être très simple ou extrêmement complexe. Bien sûr, tout est fonction de la nature du procédé, de la précision et du temps de réaction requis.

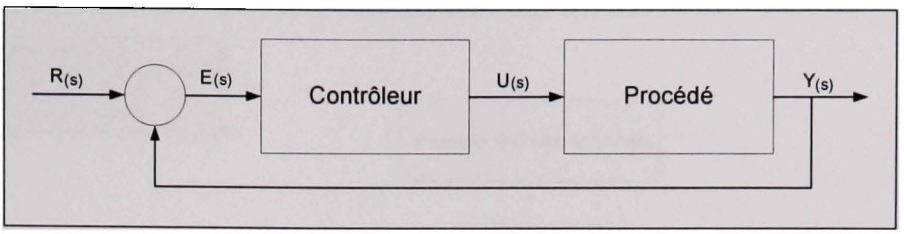


Figure 1.1 Schéma de principe d'un système industriel asservi.

On peut classer les contrôleurs selon deux grandes familles : les contrôleurs classiques et les contrôleurs avancés. La Figure 1.2 présente une carte conceptuelle de la régulation de procédés permettant de situer les différents contrôleurs dans l'ensemble de ce domaine. Cet outil de schématisation permet de représenter un ensemble de concepts reliés sémantiquement entre eux.

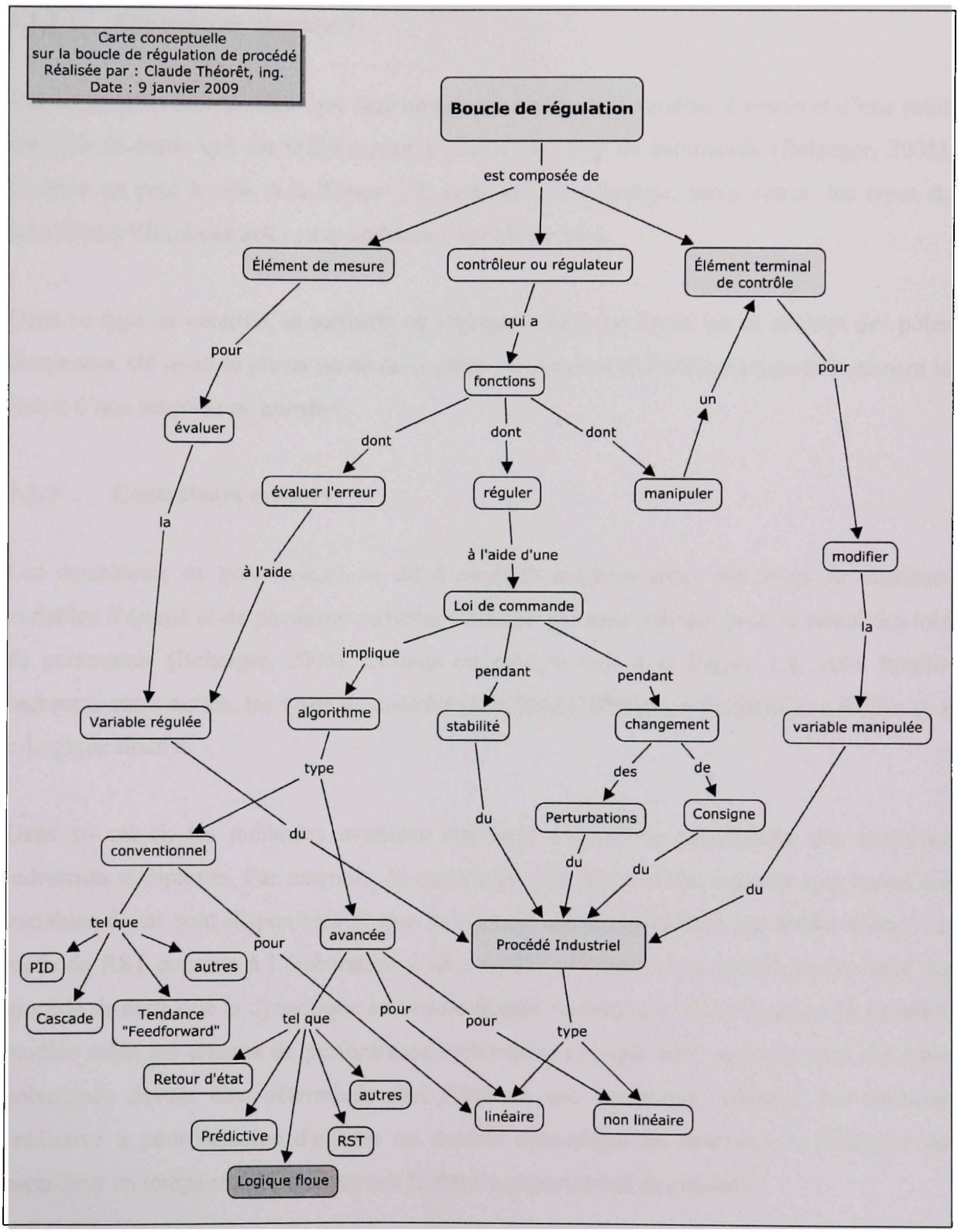


Figure 1.2 Carte conceptuelle sur la régulation de procédés.

1.1.1 Contrôleurs classiques

Les contrôleurs de type classique sont constitués d'une seule variable d'entrée et d'une seule variable de sortie qui est utilisée pour le calcul des lois de commande (Belanger, 2005). Comme on peut le voir à la Figure 1.2, cette famille regroupe, entre autres, les types de contrôleurs PID, à cascades ou à tendance « Feedforward ».

Dans ce type de contrôle, la méthode de réglage utilisée est basée sur le concept des pôles dominants. On tente de placer un ou deux pôles, ce qui permet d'utiliser un modèle prenant la forme d'une fonction de transfert.

1.1.2 Contrôleurs avancés

Les contrôleurs de type avancé ou dit *à méthode moderne* sont constitués de plusieurs variables d'entrée et de plusieurs variables de sortie qui sont utilisées pour le calcul des lois de commande (Belanger, 2005). Comme on peut le voir à la Figure 1.2, cette famille regroupe, entre autres, les types de contrôleurs « Retour d'état », « Prédictif », « RST » et à « Logique floue ».

Dans ce cas-ci, les méthodes avancées ont pour objectif de commander des systèmes industriels complexes. Par exemple, la commande par retour d'état suppose que toutes les variables d'état sont disponibles et que le système est commandable par retour d'état. La méthode RST consiste à l'élaboration d'un contrôleur à partir d'un modèle polynomial du procédé de sorte que la dynamique en boucle fermée corresponde à une fonction de transfert modèle selon les critères de performance recherchés. Le sigle RST vient du nom des trois polynômes devant être déterminés afin d'obtenir une commande efficace. La méthode prédictive a pour principe d'utiliser un modèle dynamique du processus à l'intérieur du contrôleur en temps réel afin de prévoir le futur comportement du procédé.

Bien que les recherches scientifiques conduites aujourd'hui dans le domaine de la commande industrielle se concentrent sur les contrôleurs de type avancé, l'industrie des procédés met l'accent sur l'exploitation maximale des contrôleurs de type classique, dont le PID. Ceux-ci ont fait l'objet de nombreuses études, leurs équations différentielles sont connues et leurs réglages peuvent être relativement simples. Conséquemment, depuis un demi-siècle, on retrouve des contrôleurs PID dans la majorité des systèmes de contrôles (Li, Ang et Chong, 2006).

Toutefois, les contrôleurs de type PID classique ont atteint le maximum de leur possibilité. En effet, ceux-ci sont souvent mal réglés ou tout simplement utilisés à défaut d'autres choix, ce qui occasionne, entre autres, une diminution de la performance (Kristiansson et Lennartson, 2006). Pour pallier ces limites, on vient donc ajouter un sous-système basé sur de la LF, ce qui permettra d'améliorer les performances globales du système de contrôle.

1.2 Logique floue (LF)

Les systèmes de contrôle ont contribué à l'atteinte des différents objectifs globaux des entreprises, soit à diminuer les coûts de production ou à augmenter la quantité ou la qualité de leurs produits (Harinath, 2007). L'une des avenues empruntées afin d'améliorer les systèmes de contrôle est d'introduire de la LF à l'intérieur de la boucle de contrôle (Maussion, 1998). En effet, la LF a démontré sa capacité à améliorer les performances des systèmes de contrôle non linéaires et adaptatifs (Ruel, 1994). La LF ne remplacera pas les autres systèmes traditionnels; elle est plutôt un outil supplémentaire qui permet d'obtenir des performances impossibles à atteindre autrement dans certains cas (Ruel, 1994). En général, les solutions sont plus simples, plus près des raisonnements humains et plus faciles à élaborer.

L'être humain résout souvent des problèmes complexes à l'aide de données approximatives. Les règles de la LF étant semblables à celles du raisonnement humain en ce qu'elles sont approximatives – en effet, comme la LF tient compte des données du monde naturel, tout

comme les humains, elle représente les données par des règles plutôt que par des équations précises –, une approche par LF semble tout à fait appropriée. Tout comme le soulignent les auteurs Chevrie et Guély (1998), « plutôt que de modéliser le système, il est souvent plus intéressant de modéliser le comportement d'un opérateur humain face au système », ce qui démontre bien l'importance de l'intérêt de la LF.

Bien avant les années 1990, le besoin d'une technologie capable d'imiter le raisonnement humain s'est fait sentir. La LF ne remplit pas cet objectif à elle seule, mais a démontrée qu'en l'introduisant dans les contrôleurs PID classiques, entre autres, on pouvait obtenir des systèmes de contrôle plus intelligents, plus stables et surtout plus performants. C'est pourquoi il est devenu primordial que les institutions d'enseignement, collégiale et universitaire, fassent la promotion de cette technologie en l'introduisant dans leurs cours théoriques et pratiques.

1.2.1 Systèmes à commande floue

Les méthodes de réglage et de commande de régulation industrielle sont des domaines d'application fréquents de la logique floue (Dobritoiu, 2003). Ces méthodes permettent d'obtenir une loi de commande efficace, sans faire appel à des développements théoriques importants. Elle incorpore des connaissances linguistiques sur la manière de piloter un processus difficile en prenant compte les expériences acquises par les utilisateurs et les opérateurs du processus à commander. Pour ce faire, on utilise des contrôles ayant une loi de commande basée sur les notions de la LF que l'on nommera *contrôleurs flous* ou *contrôle flou* (CF). La Figure 1.3 représente une commande floue simplifiée.

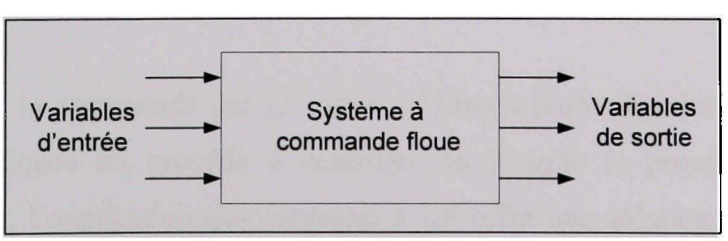


Figure 1.3 Représentation simplifiée d'une commande floue.

Un CF se caractérise par le type de relation établie entre les entrées et les sorties. En effet, cette relation est obtenue par l'application de règles qualitatives plutôt que purement mathématiques. Ces règles sont établies par les experts et les opérateurs du domaine dans lequel on applique le contrôleur à LF. La transposition du savoir-faire de l'expert, sous forme de règles d'inférences floues que l'on nommera *la base des règles* (BR), permet de rendre plus compréhensible le fonctionnement du système pour l'utilisateur.

Lorsque les BR permettent une description du système à l'aide de variables linguistiques, on parlera alors d'un contrôleur de type *Mamdani*. Par contre, dans le cas où toutes les règles de production modélisant le système sont des constantes, des polynômes ou des fonctions non linéaires des entrées du contrôleur, on parlera alors d'un contrôleur de type *Sugeno*. Les variables d'entrée et de sortie nécessaires à la commande d'un procédé sont des paramètres réels qui prennent leurs valeurs dans un univers bien déterminé. Ils émanent également des avis des experts et des opérateurs.

Les principaux objectifs des applications des CF sont les suivants : amélioration du comportement dynamique du système (en comparaison avec les contrôleurs classiques) et adaptation aux variations de paramètres du système. Les principes de conception des CF font l'objet d'un certain nombre de publications.

Les techniques classiques et modernes nécessitent habituellement la connaissance d'un modèle mathématique qui représente la dynamique du système à contrôler qui soit le plus fidèle possible à la réalité. Par ailleurs, plusieurs procédés industriels sont souvent des systèmes non linéaires (Harinath, 2007). Par conséquent, les équations mathématiques qui décrivent les systèmes sont complexes et non linéaires.

De façon générale, la commande par LF offre un intérêt particulier lorsqu'il n'y a pas de modèles mathématiques du procédé à contrôler ou lorsque le procédé à contrôler est complexe. En effet, l'application des contrôles à LF offre une solution attrayante pour les

systèmes complexes ayant un comportement fortement non linéaire en raison de sa capacité à produire un raisonnement sur des termes proches du langage humain.

Lorsque le contrôleur PID classique est incapable de stabiliser ou de réguler un processus, il faut, soit changer la structure du système de commande, soit proposer d'autres algorithmes de commande plus sophistiqués. Un CF est tout à fait approprié dans ce genre de situation étant donné que le système doit réagir rapidement, avec une erreur minimale, et se stabiliser à la valeur désirée. La structure globale d'un procédé en boucle fermée piloté par une commande floue (ou contrôleur flou) est décrite à la Figure 1.4.

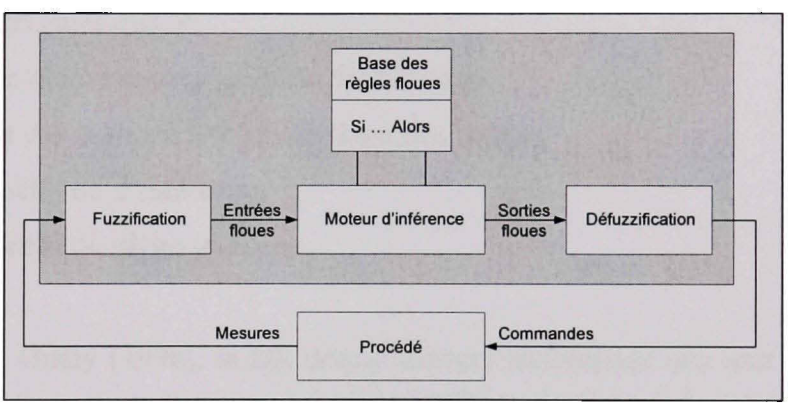


Figure 1.4 Structure de commande floue.

Le CF est contenu dans le cadre grisé. Un système flou est donc formé de trois étapes comme indiqué sur la Figure 1.4. La première étape de *fuzzification* transforme les valeurs numériques en degrés d'appartenance aux différents ensembles flous de la partition. La seconde concerne le module d'inférence qui est constitué de deux blocs, soit le moteur d'inférence et la BR. Enfin, une étape de *défuzzification* permet d'inférer une valeur précise, utilisable en commande par exemple, à partir du résultat de l'agrégation des règles. Ce concept sera vu plus en détail au chapitre 3, portant sur les concepts fondamentaux de la LF.

1.2.2 Conception d'un contrôleur flou

Un CF peut être considéré comme la mise en œuvre d'un algorithme de commande d'un procédé déterminé. Pour sa conception, il n'est pas nécessaire de connaître le modèle mathématique du procédé. On doit plutôt déterminer les grandeurs mesurables et analyser les comportements dynamiques du procédé vis-à-vis de la variation de la grandeur de commande. Pour faire un bon résumé de la conception d'un CF, il est souhaitable de se référer à la Figure 1.5. Les étapes principales pour concevoir un CF sont les suivantes :

1. Étude et description du processus à réguler;
2. Choix de la structure du CF;
3. Détermination des caractéristiques de *fuzzification*;
4. Établissement des règles d'inférences (base des règles);
5. Choix de la méthode d'inférence;
6. Choix du procédé de *défuzzification*.

Selon Chevrie et Guély (1998), la BR donne souvent satisfaction dès leur premier essai. Il arrive cependant que l'on ait besoin de modifier ou de mettre au point cette BR. Par conséquent, la méthodologie est habituellement suivie d'une validation par simulation à l'aide d'un logiciel. Cette validation assure, d'une part, une mise au point qui permet d'améliorer la *fuzzification* et l'établissement des règles d'inférence et, d'autre part, permet de faciliter l'implantation réelle du système flou.

Toutefois, comme mentionné précédemment, le réglage d'un système flou étant réalisé par des experts ou opérateurs, on comprendra que plus les systèmes seront complexes, plus la dimension de la BR du système flou sera grande et plus difficile sera l'élaboration des règles d'inférence floue. De plus, il est parfois impossible d'obtenir toutes les informations de commande du procédé à réguler pour diverses raisons, par exemple les informations non fiables ou fournies sous forme numérique non traduisible en termes linguistiques, etc. (Tong-Tong, 1995). Aussi, parce que les règles correspondent à un savoir empirique et non

mathématique, les CF fournissent des résultats qui ne sont pas prévisibles (Sangalli, 2001). Pour toutes ces raisons, il est clair que la validation par simulation du CF dans le cas d'applications critiques est d'une importance capitale. La Figure 1.5 présente une carte conceptuelle de la conception d'un CF permettant de situer les différentes étapes de conception, ainsi que la place où intervient la simulation du contrôleur dans l'ensemble de la conception.

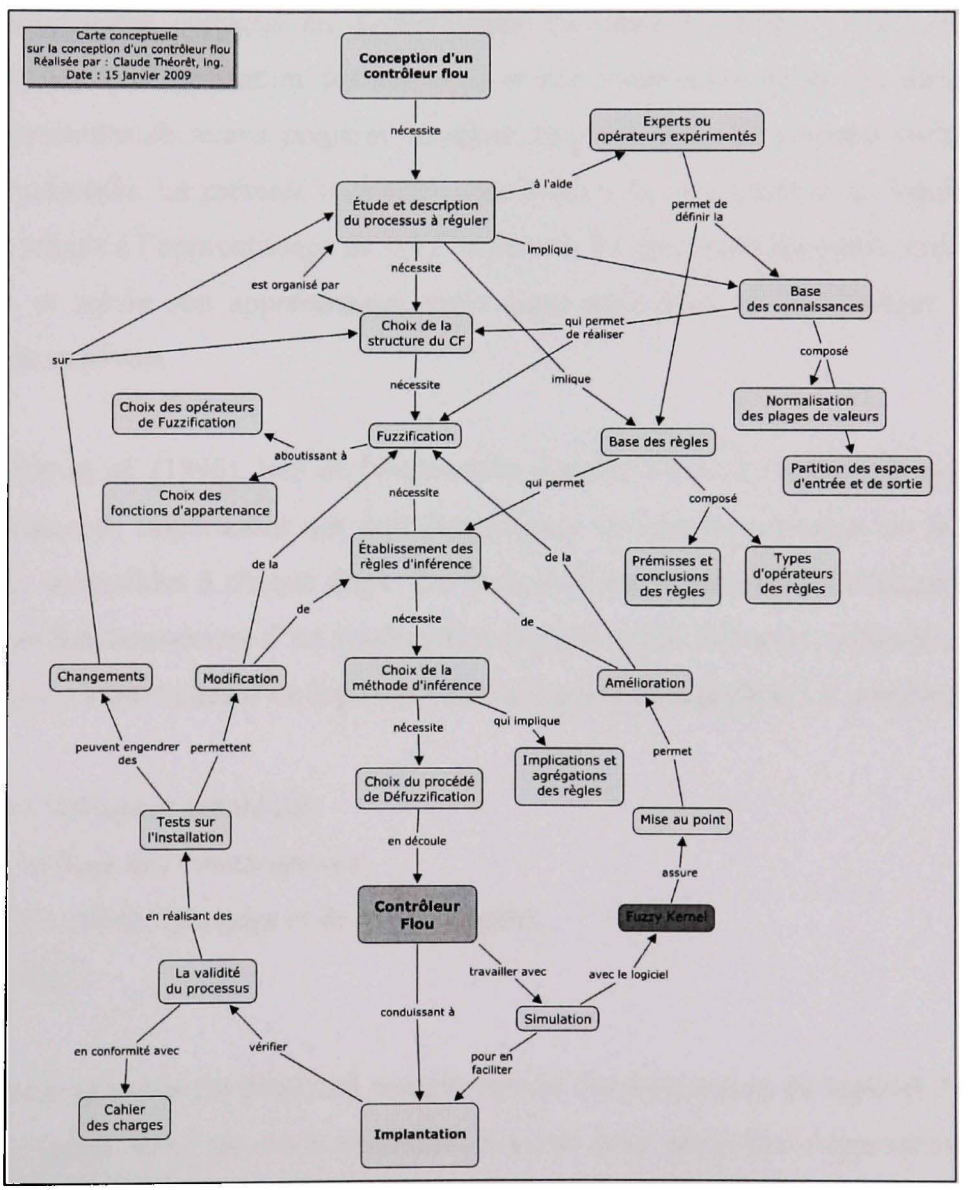


Figure 1.5 Carte conceptuelle sur la conception d'un CF.

1.3 Proposition pour l'apprentissage des concepts de la logique floue

En raison de ce qui précède, l'élaboration et la mise en œuvre d'un logiciel efficace, permettant l'enseignement et l'application de la LF, deviennent une nécessité absolue. L'objectif principal de cette recherche est d'élaborer et de valider un logiciel d'enseignement et d'application de la logique floue dans un contexte d'automate programmable. Nous voulons, par ce projet de recherche, mettre en application ce logiciel dans le cadre du nouveau programme collégial en Technologies du génie électrique (TGÉ). Ce logiciel d'enseignement et d'application des concepts et des fondements de la LF, dans le milieu collégial, permettra de mieux préparer l'étudiant finissant aux changements technologiques du milieu industriel. La présente recherche s'intéresse à la conception d'un logiciel qui soit réellement adapté à l'apprentissage de la LF et fournit à l'apprenant les outils, non seulement pour gérer et suivre son apprentissage, mais aussi pour faire de l'ordinateur un soutien pédagogique motivant.

Selon Dunlop *et al.* (1996), lors de l'élaboration d'outils d'aide à l'apprentissage de la LF, une des exigences importantes qui doit être retenue est que les concepts de la LF soient présentés et accessibles à chaque étape. De plus, l'utilisateur doit pouvoir interpréter toutes les étapes de fonctionnement d'un système flou (*fuzzification*, inférence, *défuzzification*). Les caractéristiques essentielles d'un logiciel d'aide à l'apprentissage de la LF comprennent :

- interface utilisateur graphique;
- éditeur de base des connaissances;
- outil interactif de débogage et de mise au point;
- aide en ligne.

Ces caractéristiques ont été prises en compte lors du développement du logiciel. Nous avons élaboré un logiciel avec un environnement convivial pour permettre l'apprentissage rapide des concepts de base de la logique floue. En fonction des intérêts, les apprenants seront

capables de connaître et mettre en œuvre une commande avancée à logique floue par rapport à une commande classique sous format simple et interactif.

Nous proposons, avec ce logiciel, d'outiller les enseignants de niveau collégial, par conséquent les étudiants, à la théorie de la logique floue en leur donnant un outil qui leur permettra de s'approprier les notions de base. Ce logiciel est un outil pédagogique, certes, mais son originalité réside dans le fait qu'il permet la réalisation d'exemples pratiques. Il permet une mise en œuvre de ces concepts au sein d'une interface accessible en prenant en charge tous les aspects des calculs.

La réalisation de ce projet doit s'effectuer en deux étapes. La toute première étape est de réaliser le logiciel d'enseignement et d'application de la LF. Cependant, avant de pouvoir élaborer un logiciel d'enseignement, il faut connaître les concepts fondamentaux de la LF ainsi que les logiciels existants. À l'heure actuelle, il existe très peu de logiciels d'enseignement et d'application de la LF dans un contexte d'automate programmable. De prime abord, les logiciels existants permettent la mise en œuvre de systèmes flous, mais en délaissant l'aspect pédagogique, ce qui les rend donc mal adaptés pour l'enseignement.

Pour la réalisation du logiciel, nous avons d'abord colligé toutes les informations pertinentes au domaine de la LF, en faisant largement appel aux méthodes de mise en œuvre, sans pour autant délaisser le contexte pédagogique appliqué à la LF. Par la suite, tous ces éléments ont été réunis suivant un processus de synthèse. Cette synthèse a permis de réfléchir sur les éléments essentiels retenus sur la LF pour une mise en œuvre simple et conviviale. C'est ainsi que nous avons pu définir la structure du logiciel et tous les éléments s'y rattachant pour une prise en main rapide.

Les auteurs Girault et d'Ham (2005) spécifient que « Le travail expérimental est essentiel pour l'apprentissage des sciences expérimentales, car c'est la seule situation où peuvent être mis en œuvre certains objectifs d'apprentissage spécifiques. », ce qui démontre bien l'importance d'avoir un outil pédagogique permettant la réalisation d'exemples pratiques.

Par conséquent, la deuxième étape de ce projet devra être développée, soit l'élaboration de deux blocs de fonction intégrés dans un API qui permettent la mise en œuvre pratique d'application à LF. La première fonctionnalité développée offre un bloc de fonction qui est destiné à permettre l'intégration d'un système flou. L'idée de ce bloc de fonction est de réaliser un système flou de notre choix comportant deux entrées et une sortie intégré dans un API. Le deuxième bloc de fonction est plutôt dédié au domaine du contrôle de procédé. Nous avons réalisé un CF de type PID sous forme de bloc de fonction intégré dans un API. Ainsi, le logiciel permet l'interconnexion avec un API pour effectuer un contrôle en temps réel. L'étudiant aura ainsi la possibilité d'apprendre à créer un contrôleur basé sur la LF et cela en toute simplicité.

Cependant, pour élaborer un CF simple et efficace, il faut tout d'abord connaître les différents types de CF. Il existe une panoplie de CF, il peut donc devenir très ardu de choisir le meilleur. Dans le cadre de ce projet, nous ne rechercherons pas nécessairement le meilleur contrôleur au point de vue de la performance, mais plutôt un CF simple d'apprentissage, soit un CF capable de satisfaire le deuxième objectif de ce projet.

Il faut toutefois poser la question : pourquoi choisir un CF comme structurant pour l'enseignement de la LF ? On a vu précédemment comment la LF s'inscrit au cœur même de la commande industrielle. La LF apparaît comme une alternative aux contrôles classiques, par exemple le contrôleur PID, lorsque ceux-ci ont des performances amoindries. De plus, pour l'objectif recherché dans le cadre de ce projet, le principal avantage du choix d'un CF comme structurant, est que les notions de régulation de procédé sont enseignées dans les cégeps et les universités. Toutes les activités théoriques et pratiques reliées à la LF serviront à illustrer notamment comment celle-ci a permis des développements industriels spectaculaires à partir d'algorithmes très simples de traduction de connaissances linguistiques en entités numériques. Ils serviront à présenter des applications de la LF qui ont eu un impact sur les techniques de contrôle des processus automatiques en permettant de faire le lien entre les contrôleurs PID classiques et les CF. L'objectif final est de permettre à l'étudiant de maîtriser les techniques permettant l'automatisation des procédés industriels.

Lorsque tous ces développements auront été réalisés, la validation du logiciel et du CF sera réalisée sur un banc d'essai du département des Technologies du génie électrique du cégep du Vieux Montréal. Dans la section qui suit, nous mettrons en évidence l'apport de notre projet pour l'enseignement de la LF.

1.4 Contribution attendue du projet

En 1993, le renouveau de l'enseignement collégial s'est fait en utilisant une nouvelle pratique : l'approche par compétences. Créé en 2002 par le ministère de l'Éducation, le nouveau programme de Technologie de l'électronique industrielle (TEI) a pour objectif de former des personnes aptes à exercer la profession de Technologue en électronique industrielle en utilisant l'approche par compétences. L'une des principales tâches des technologues en électronique industrielle est l'installation et la mise en route d'appareils dans les systèmes de contrôle-commande. En collaboration avec des ingénieurs, ils participent également à la conception ou à la modification d'un système automatisé ou d'une installation électrique (Devis du ministère de l'éducation, 2002).

Le cégep du Vieux Montréal, dans le cadre de ses programmes en Technologies du génie électrique, forme des technologues à la fine pointe de la technologie et cela dans un environnement stimulant. Il est parmi les collèges qui bâtissent et testent le nouveau programme (2007) en Technologies du génie électrique. L'un des objectifs importants pour les cégeps est aussi l'accroissement de l'expertise dans les domaines technique et pédagogique.

L'une des compétences du programme en TEI demande de « Participer à la conception d'un projet de contrôle-commande ». Dans ces circonstances, les enseignants du département de TEI ont choisi, en autre, de faire l'apprentissage des CF pour atteindre l'objectif de cet énoncé de compétence du nouveau programme. C'est là que s'insère notre projet pour combler une lacune et permettre l'apprentissage des systèmes de contrôle avancé dont les CF et cela en respectant les éléments de compétence établis par le nouveau programme. Notre

projet a été intégré dans le cadre du cours 243-502-VM « Programmer un système de régulation » du nouveau programme de TEI du cégep du Vieux Montréal pour y faire l'apprentissage théorique et pratique de la LF et du CF.

En collaboration avec le cégep du Vieux Montréal, la démarche pédagogique ainsi que le matériel didactique découlant des résultats de la recherche ont été présentés aux enseignants impliqués dans l'élaboration du nouveau programme. Cette rencontre a permis de donner la possibilité à ces enseignants de s'approprier les notions de la LF. De plus, les contenus didactiques et la démarche pédagogique employée ont fait l'objet d'une présentation de type magistral à une classe cible. Cette présentation a permis aux enseignants impliqués de structurer l'énoncé de compétence ainsi que les éléments de compétence sur les concepts et les fondements de la logique floue dans le domaine particulier de l'électronique industrielle. Cet énoncé de compétence est disponible à l'annexe I.

L'énoncé de compétence pour l'apprentissage de la LF sert d'outil de référence afin d'établir si le projet de notre recherche est en mesure de s'acquitter de ses obligations à un niveau jugé acceptable. L'énoncé de compétence permet d'établir les objectifs d'apprentissage de la LF. De plus, cet énoncé de compétence fait état des connaissances, des aptitudes et du jugement dont ont besoin de faire preuve les étudiants afin de rendre les services et de procéder aux interventions qui relèvent de l'exercice de leur profession.

Notre projet permet de fournir un soutien pédagogique complémentaire aux enseignants et aux étudiants. Notre projet contribue à :

- offrir aux étudiants un soutien pour stimuler leur curiosité sur la LF;
- inciter d'autres collèges à intégrer la LF dans leur programme;
- assurer aux enseignants un soutien technique et pédagogique;
- offrir aux enseignants un outil pédagogique pour l'enseignement et l'application de la LF;
- favoriser la collaboration entre le milieu universitaire et le milieu collégial;
- améliorer les compétences des futurs technologues afin de permettre d'avoir de meilleurs indices de performance dans les procédés, par conséquent une amélioration des profits des entreprises.

Notre projet permet ainsi de favoriser l'atteinte des objectifs d'apprentissage relatif aux énoncés de compétences du nouveau programme.

1.5 Limites du projet

Le logiciel de ce projet est conçu pour des utilisateurs débutants. Mais comme il intègre la plupart des caractéristiques de base très similaires à celles des logiciels professionnels proposés par l'industrie, il ne déroutera donc pas un utilisateur expérimenté. Il est donc important de noter que notre logiciel ne remplacera pas les autres logiciels de type professionnel; il sera plutôt un outil complémentaire comportant un modèle pédagogique mieux adapté pour permettre aux étudiants de connaître les concepts et les limites de la commande à base de logique floue. L'objectif d'un tel logiciel est clair : convaincre les enseignants et les étudiants de la simplicité de la logique floue et de sa mise en œuvre.

Cependant, l'apprentissage du fonctionnement d'un système flou n'est pas trivial pour un novice, il faut donc offrir à l'usager un ensemble de fonctions typiques prédéfinies (valeurs par défauts) dans le logiciel de façon à ce qu'il puisse, s'il le désire, les utiliser sans avoir à les adapter. De ce fait, nous avons volontairement limité certains paramètres du système flou,

comme le nombre d'entrée et de sortie. De même, le type d'inférence ainsi que la méthode de *défuzzification* ont été limités. Au plan éducatif, ces choix sont avantageux, alors qu'au plan opérationnel ils ne sont pas contraignants.

La LF est une infime partie du domaine très vaste de l'intelligence artificielle. Notre contribution est limitée à la mise en forme du raisonnement humain dans un contexte industriel sous forme d'un contrôleur implanté dans un automate programmable. Les automates programmables sont des équipements de base de l'investissement de production. En Amérique du Nord, Allen-Bradley®, filiale de la Rockwell International Corporation, est un chef de file en automatisation (Avishai, 1989). Les produits Allen-Bradley® constituent plus de 66 % de la part du marché industriel nord-américain. Allen-Bradley® se distingue par des commandes en temps réel et des solutions pour systèmes avec communication en réseaux d'une grande fiabilité. Ils peuvent satisfaire ainsi à pratiquement toutes les exigences qui se présentent dans le domaine de l'automatisation industrielle. Pour toutes ces raisons, nous avons choisi d'intégrer le système flou dans l'API ControlLogix® de la compagnie Allen-Bradley®.

Pour des raisons de souplesse et de facilité d'utilisation de l'outil de développement, ainsi que la capacité d'une bonne intégration avec les produits disponibles et la réduction des coûts et du temps de développement, ce projet a été réalisé à l'aide de l'outil de développement Microsoft Visual Basic 6.0®. Visual Basic 6.0® est un environnement de programmation performant, qui allie les avantages d'un langage de programmation connu, souple et facile à utiliser, qui permet une bonne intégration avec les produits industriels disponibles. Ce langage permet la programmation événementielle dans les états et les formulaires.

1.6 Conclusion

Notre projet de recherche a pour objectif d'élaborer un logiciel spécialisé et convivial permettant l'apprentissage des notions de la LF et d'intégrer un CF dans un API pour contrôler un processus en temps réel. Notre projet est donc un outil pédagogique pour

enseigner les notions de la LF intégrant une structure flexible et une interface graphique simple et facile d'utilisation afin de minimiser le temps d'apprentissage. Il est également un outil pour aider les étudiants à apprendre et à comparer l'application des contrôleurs classiques et flous. De plus, notre projet de recherche va permettre aux apprenants des notions de la LF, d'explorer et de tester leurs connaissances dans différents environnements d'apprentissage.

Un des grands avantages de notre logiciel est qu'il ne demande aucune connaissance particulière en langage de programmation et que toute personne ayant des notions élémentaires en informatique peut facilement élaborer ses propres systèmes flous. Les étudiants, en créant leurs propres systèmes flous, peuvent ainsi mieux répondre aux besoins des compétences de l'apprentissage de la LF et de les intégrer aisément dans la pratique.

En conclusion, le logiciel est à la fois un outil pédagogique et un outil de mise en œuvre de la LF. En vue de favoriser l'expérimentation des systèmes à logique floue au collégial, ce logiciel propose un soutien à ceux qui souhaitent l'utiliser dans le cadre de leurs cours. Cette démarche novatrice constitue une des originalités de notre projet.

C'est donc dans cette perspective que cette recherche vise une amélioration et une actualisation de l'enseignement. Le logiciel d'enseignement et d'application de la LF, dans une approche d'apprentissage par compétences, est donc un outil de base qui permettra de couvrir complètement certains objectifs de différents cours. Par ailleurs, cette recherche contribuera certainement à favoriser des collaborations entre les écoles de génie et les départements des Technologies du génie électrique des cégeps, en plus de répondre aux nouveaux besoins pédagogiques et techniques durant l'implantation des nouveaux programmes. Nous allons dans le chapitre suivant présenter la revue de littérature.

CHAPITRE 2

REVUE DE LITTÉRATURE

Ce chapitre consiste essentiellement à passer en revue la littérature et situer les présents travaux dans l'ensemble des travaux de recherche qui ont été faits sur la LF et les CF. Mais avant d'entreprendre la revue de littérature, nous présenterons un bref historique de la logique floue, ainsi que son évolution. Deux principaux sujets théoriques sont utilisés dans le cadre de ce projet soient la théorie de la LF et la théorie des CF, c'est pourquoi cette revue de littérature vise principalement deux objectifs.

Le premier objectif de cette revue de littérature est de recenser tous les outils d'aide à l'apprentissage de la LF. Le travail de mise au point de produits informatiques pour l'enseignement est une tâche complexe qui nécessite la collaboration de plusieurs intervenants ayant des points de vue souvent très différents. Pour réaliser un logiciel, il est important de spécifier l'ensemble des actions et réactions qui l'animeront. Pour ce faire, on pourra s'inspirer du travail réalisé dans ce domaine.

Le deuxième objectif de cette revue de littérature est de trouver un modèle de CF simple d'apprentissage, peu complexe et qui ne nécessite pas une mise en œuvre ardue. Cependant, il existe plusieurs modèles de CF. La littérature offre donc une multitude d'articles sur les CF de toutes sortes; par contre, il devient difficile d'en cerner un qui soit simple et efficace. Cependant, avant de se lancer trop loin dans les différents modèles de CF existant et dans les différentes façons de les utiliser, il faut se rappeler l'objectif premier de la réalisation du CF, soit mettre en application la LF dans un contexte de commande de procédé réel. En d'autres termes, le CF sert de structurant pour l'enseignement des notions de la LF.

2.1 Historique de la logique floue

Dubois *et al.* (1993) résument les balbutiements de la LF. Ils mentionnent qu'au siècle dernier, le philosophe américain Charles Peirce fut l'un des premiers chercheurs modernes à noter (et à regretter) que « Les logiciens ont beaucoup trop négligé l'étude de l'imprécision, ne soupçonnant pas l'importance de son rôle en mathématiques ». Ils mentionnent également que dans les années 30, Jan Lukasiewicz développa la logique avec des valeurs de vérité intermédiaires; de même, le philosophe Max Black proposa des profils de cohérence (les ancêtres des fonctions de la LF) dans le but de caractériser les symboles vagues.

Le mathématicien Karl Menger a été le premier à employer le terme « ensemble flou » dans le titre d'un article écrit en français, en 1951 (Dubois, Prade et Yager, 1993). Menger étudiait une géométrie basée sur des points dans le plan et dans l'espace dont la localisation était associée à des fonctions de diffusion de probabilités. En 1966, il reconnaissait un certain rapport entre ce qu'il définit comme étant « des ensembles nébuleux » et l'idée des « ensembles flous » qui, à cette époque, était toute récente.

C'est en 1965 que le professeur Lotfi A. Zadeh de l'Université de Californie (Berkeley) a développé sa théorie des ensembles flous avec la publication d'un article intitulé Fuzzy Sets (Dubois, Prade et Yager, 1993). Plusieurs autres chercheurs ont travaillé à l'élaboration du développement de la théorie de la LF : Kampé de Fériet *et al.* (1969), Shilkret (1971), Cohen (1973), N. Rescher (1976), Shafer (1976), B. Schweizer et A. Sklar (1983). Zadeh a observé que la logique binaire de l'informatique conventionnelle était incapable de manipuler des données représentant des idées humaines vagues ou subjectives, comme « une personne séduisante » ou « assez grande ». La logique informatique binaire (logique booléenne classique) ne voyait la réalité que dans des termes aussi simples que « marche/arrêt » ou « oui/non ».

En effet, avec la logique classique pour le concept d'appartenance à un ensemble, il n'existe que deux états possibles d'appartenance : soit un élément appartient à un ensemble, soit il n'appartient pas à cet ensemble. Pourtant dans la vie de tous les jours, nous ressentons intuitivement le besoin de nuancer certaines affirmations, c'est le concept d'appartenance graduelle. Le concept d'appartenance graduelle d'un élément à un ensemble a conduit au développement d'une logique basée sur la gradation des propriétés et au calcul d'incertitude, appelé *Théorie des possibilités* qui spécifie la gradation des concepts de possibilité et de certitude (Zadeh, 1978).

La LF, ou plus généralement le traitement des incertitudes, a pour but la représentation des connaissances imprécises et le raisonnement proche du langage humain de tous les jours. Elle a recourt le plus souvent aux connaissances d'experts ou d'opérateurs qualifiés travaillant sur le processus à contrôler. Le concept des ensembles flous et la *Théorie des possibilités* offrent conjointement une structure permettant de traiter des prédicats dont la valeur est dépendante du degré, ainsi que des caractéristiques exprimant une incertitude graduelle. La LF a été conçue pour permettre aux ordinateurs d'effectuer des distinctions valables entre des niveaux de gris, en travaillant de façon similaire au raisonnement humain. Les technologies floues sont ainsi conçues pour incorporer la théorie de la LF dans le traitement de données moderne, de façon à créer des systèmes et des produits plus conviviaux.

En 1974, Ebrahim Mamdani expérimentait un système de contrôle construit en utilisant la théorie des ensembles flous dans le but de contrôler une machine à vapeur et des chaudières par la combinaison et la synthèse d'un ensemble de règles de contrôle linguistique obtenu à partir de l'expérience humaine des opérateurs (Mann, Bao-Gang et Gosine, 1999; Sangalli, 2001). Le fonctionnement du système de contrôle se fonde sur l'article de Lotfi Zadeh de 1973, « Algorithmes flous pour les systèmes complexes et les processus de décision » (Borne, 1998). Il s'en suit une émergence des applications en Europe.

Au Japon, les premières applications de la théorie de la LF furent implantées en 1985 grâce au chercheur Michio Sugeno. Dès lors, les Nippons voient l'ouverture, en 1987, du premier métro contrôlé grâce à la LF à Sendaï dans le nord du Japon. Les commandes à LF rendent les voyages en métro beaucoup plus confortables grâce à des accélérations et des freinages effectués en douceur. En fait, le chauffeur n'a qu'à appuyer sur un bouton pour mettre le train en route. La LF a aussi progressé dans le contrôle des appareils électroménagers. La lessiveuse à linge « à LF » possède plus de 400 cycles préprogrammés; cependant, malgré sa complexité technologique, elle reste très facile à utiliser. Il suffit à l'utilisateur d'appuyer sur le bouton de départ et le reste est pris en charge par la machine; elle estime automatiquement le type du tissu, le volume et le degré de saleté du linge, choisit le cycle le plus approprié et le débit d'eau.

Nous pourrions allonger cette liste et citer les exemples de poste de télévision, de distributeur de billets de banque et même de systèmes de gestion financière. La popularité remarquable de la LF au Japon pourrait même surprendre le docteur Zadeh, son fondateur (Oyagi, 1991). La mise en marché de logiciels et de matériels permettant le développement d'applications utilisant la LF (les cartes spécialisées, les cartes pour les automates programmables, etc.) a fait augmenter le nombre d'applications.

À partir de 1990, les applications apparaissent en grand nombre. Par ailleurs, les fabricants intègrent de plus en plus la technologie de la LF dans les appareils de grande consommation (appareils de photos, vidéo, ...). Ces appareils sont présentés comme faisant intervenir un réglage par LF ou « fuzzy-logic inside » ou encore « fuzzy-control ». Ces appareils comportent une nouvelle technologie souvent non maîtrisée par le personnel technique qui devra intervenir sur ceux-ci, ce qui demandera de nouvelles exigences professionnelles.

Au Québec, dans le secteur industriel, la compagnie Alcan de Jonquière utilise un module à LF de la compagnie Omron pour contrôler la tension pour l'enroulement des rouleaux d'un convoyeur. De même, la scierie Max Meilleur à Ferme-Neuve utilise ce même module pour le contrôle de positionnement d'un chariot qui transporte les billes de bois jusqu'aux scies.

Dans ces cas, la LF sert principalement à optimiser les rampes d'accélération et de décélération des contrôles de vitesse des moteurs. Pour ces deux industries, les techniques conventionnelles ont été impuissantes tandis que la technologie de la LF a résolu leur problème. En conclusion, la LF, mise au service d'appareils ménagers ou industriels, permet d'émuler les raisonnements humains, de quantifier des informations imprécises et de prendre des décisions basées sur des données vagues et incomplètes afin d'obtenir des actions valables.

2.2 Axes de développement

2.2.1 Logique floue dans le domaine de la recherche

Aujourd'hui, la LF fait l'objet de nombreuses recherches dans plusieurs domaines. En fait, plusieurs recherches proposent de nouvelles approches basées sur les concepts de la LF pour aborder leurs problématiques. Un nombre croissant de domaines est concerné par l'exploitation de la théorie de la LF.

> « Si d'un point de vue historique la commande floue a joué un rôle précurseur, elle est aujourd'hui rejointe par bon nombre de disciplines telles que la recherche opérationnelle et l'aide à la décision, le mesurage et l'évaluation subjective, l'analyse de données, la classification et la reconnaissance de formes, le traitement et l'interprétation des images, ..., la robotique, mais aussi l'économie, la gestion, la psychologie, l'ergonomie, etc. » (AFIA - Association Française d'Intelligence Artificielle, 2003)

Étant donné que le réglage découle donc a priori des connaissances du fonctionnement du système et du savoir-faire des experts ou des opérateurs qualifiés travaillant sur le processus, plusieurs recherches s'orientent vers le développement de méthode originale pour déterminer automatiquement la BR d'un système flou. Jusqu'à présent, les techniques de réglage sont essentiellement empiriques et les performances dépendent de l'expertise des intervenants, ce qui en fait son talon d'Achille. En effet, comme il n'existe aucune méthode pour établir les relations de comportement d'un système, la mise en œuvre d'un système à LF n'en est que

plus difficile. Par contre, ce savoir-faire ou cette expertise doit absolument exister pour rendre possible l'utilisation d'un CF (Hernandez, 2007). L'élaboration des règles floues est d'autant plus facile lorsque ce savoir-faire existe.

Par conséquent, certaines recherches ont comme problématique de développer différentes approches pour améliorer ou faciliter la mise en œuvre de systèmes utilisant la LF. Abdelnour *et al.* (1991) ont proposé une procédure de conception qui ne nécessite aucune modélisation comportementale du processus pour établir la base des règles. Une table de conversion 3-D est utilisée pour déterminer la loi de commande en rapport aux trois entrées (e, Δe et Δ^2e). Nakoula (1997) a développé un système qui repose sur la conception d'une méthodologie de modélisation explicite basée sur la génération automatique de systèmes d'inférence floue (SIF) à partir de données d'apprentissage.

Les méthodes de conception relèvent du cas par cas, d'autant plus que l'élaboration d'un système flou varie selon le contexte. En effet, les circonstances et les conditions qui entourent le fonctionnement du système auront une influence sur la conception du système flou. À la limite, le fait de négliger l'effet du contexte pourrait limiter les performances du système (Eksioglu, 2000). De ce fait, certaines recherches proposent une structure cognitive tenant compte de l'effet du contexte et se rapprochant le plus possible du raisonnement de l'expert. Eksioglu (2000), par sa recherche propose une architecture contenant des modules cognitifs, appelée *structure cognitive pour le flou contextuel*, qui permet d'intégrer l'effet du contexte dans un système à LF.

Toutefois, comme mentionné précédemment, le réglage d'un système flou étant réalisé par des experts ou opérateurs, on comprendra que plus les systèmes seront complexes, plus la dimension de la BR du système flou sera grande et plus difficile sera l'élaboration des règles d'inférences floues. Par conséquent, un système complexe qui aurait un nombre très élevé de règles entraînerait une limitation des performances du système flou. Le sous-problème qui se pose alors consiste à réduire le nombre de règles. De ce fait, des recherches présentent des approches visant à minimiser le nombre des règles d'inférence d'un système flou sans en

amoindrir ses performances. Citons par exemple, les travaux de Rattan *et al.* (2000) qui ont proposés une méthode de réduction de la BR pour la conception d'un contrôleur PID flou qui prend avantage des propriétés des contrôleurs flous de types PI et PD en modifiant la structure et les BR du contrôleur.
La LF ne cesse d'évoluer et de faire l'objet de nombreuses recherches. Avec tous ces développements et activités de recherche, la LF représente donc un intérêt certain.

2.2.2 Logique floue dans l'industrie

Dès son introduction dans le monde industriel, les méthodes de commande floue ont attiré l'attention. La communauté scientifique, quant à elle, a fait preuve de réactions controversées : peu d'enthousiasme, du scepticisme, voire même de l'hostilité (Longchamp, 2007). Les doutes de la communauté scientifique proviennent du fait qu'il n'existe pas, dans la littérature, de théorie générale qui caractérise rigoureusement la stabilité et la robustesse du système de contrôle, contrairement aux contrôles classiques. Les mathématiques classiques sur lesquelles est bâtie l'automatique traditionnelle sont beaucoup plus riches que celles de la LF.

En dépit de ces réactions du milieu scientifique, les contrôles à LF offrent une solution attrayante pour les systèmes complexes ayant un comportement fortement non linéaire. Également, en raison de sa capacité à produire un raisonnement sur des termes proches du langage humain, elle a permis d'obtenir une explosion d'applications dans différents champs du milieu industriel (Eksioglu, 2000). Par ailleurs, étant donné que la loi de commande est exprimée en utilisant un langage linguistique proche du raisonnement humain, les opérateurs peuvent facilement interpréter les effets de celle-ci et par conséquent analyser clairement le comportement du système en boucle fermée.

Depuis quelques années, on voit émerger, notamment aux États-Unis, plusieurs applications utilisant des systèmes de contrôle basées sur la théorie des ensembles flous (Tong-Tong, 1995). Les besoins matériels et logiciels pour la mise en œuvre de ces systèmes flous

nécessitent un micro-ordinateur et un microprocesseur. Au tout début, les systèmes flous étaient exécutés sur des microprocesseurs classiques. Au milieu des années 80, Togai et Yamakawa ont été les premiers à initier l'apparition des microprocesseurs flous numériques et analogiques (Alsaleh, 1998). L'arrivée de ces microprocesseurs a permis d' élargir la sphère des applications à des domaines comme l'électronique grand public où des traitements plus rapides sont nécessaires.

En ce qui concerne les performances, la LF obtient des résultats remarquables lorsque la théorie est simple et s'applique à des systèmes complexes, et lorsqu'il n'y a pas de modèles mathématiques du procédé à contrôler. De plus, les contrôles à LF offrent une robustesse de la commande floue vis-à-vis des incertitudes, et également, ils offrent la possibilité d'avoir une commande auto-adaptative aux variations du procédé.

Pour ce qui est des principes d'un contrôle à LF et des concepts mathématiques, ils sont faciles à saisir et à maîtriser. Les concepts mathématiques entrant en jeu sont élémentaires. Quant à l'implantation de tels contrôleurs, une fois que les relations de comportement du procédé ont été établies par un expert, elle est actuellement directe, car il existe sur le marché des outils de développement professionnels performants.

2.2.3 Enseignement de la logique floue

Depuis 2006, le nouveau programme de TEI est en cours d'implantation dans les cégeps du Québec. Le ministère a établi une liste de compétences qui constituent les objectifs et standards du nouveau programme. L'une des principales tâches des technologues en électronique industrielle est l'installation et la mise en route d'appareils dans les systèmes de contrôle-commande (Devis du ministère de l'éducation, 2002).

Dans le nouveau programme, chaque cégep doit définir sa propre grille de cours de façon à respecter les compétences spécifiées. Il en résulte que les grilles de cours deviennent différentes d'un cégep à l'autre et il n'y a plus de tronc commun obligatoire entre programmes

connexes. Par contre, chaque cégep peut adapter sa grille de cours en fonction de certains besoins spécifiques des employeurs de sa région, des possibilités de regroupement de cours entre ses propres programmes, de projets pédagogiques originaux, bref, de se distinguer des autres institutions (Devis du ministère de l'éducation, 2002).

Nous avons vu précédemment que la LF apparaît comme une alternative aux contrôles traditionnels. Plusieurs applications industrielles à base de LF, qui ont été réalisées ces dernières années, ont prouvé leur efficacité pour résoudre différentes problématiques dans lesquelles les connaissances disponibles étaient incertaines et imprécises (Bouchon-Meunier, Foulloy et Ramdani, 1998). À partir de maintenant, la LF apparaît comme un outil supplémentaire mis à la disposition de tout développeur de système qui doit posséder toutes les connaissances nécessaires à la conception des systèmes automatiques. Elle est également considérée comme un outil crucial pour planifier la recherche et le développement (Bouchon-Meunier et Zadeh, 1995). Il est donc essentiel de pouvoir faire l'apprentissage des notions de base de la LF. De plus, l'une des compétences du programme en TEI demande de « Participer à la conception d'un projet de contrôle-commande ».

Dans cette perspective, plusieurs cégeps offrant le programme de TEI enseignent la LF au sein de leurs cours, par exemple le cégep du Vieux Montréal, le cégep de Limoilou, le cégep de Lévis-Lauzon pour ne nommer que ceux-là. À cet effet, les cégeps font l'apprentissage des CF pour atteindre l'objectif de cet énoncé de compétence du nouveau programme. C'est là que s'insère notre projet pour combler une lacune et permettre l'apprentissage des systèmes de contrôle avancé tels que les CF et cela en respectant les éléments de compétence établis par le nouveau programme.

2.3 Logiciels d'aide à l'apprentissage de la logique floue

La littérature montre le potentiel de la LF comme une des alternatives aux contrôles traditionnelles. De plus, elle montre l'importance de l'existence des outils d'exploitation de la LF non seulement pour l'intégration et la mise en œuvre en usine, mais aussi comme un

outil d'aide à l'apprentissage (Akcayol *et al.*, 2004; Dunlop *et al.*, 1996; Foulloy, Boukezzoula et Galichet, 2006). Toutefois, la recherche bibliographique a démontré que peu de travaux traitent des logiciels d'aide à l'apprentissage de la LF. Tous ceux qui sont intéressés par la LF, sont confrontés à un manque d'outil de mise en œuvre à la fois technique et pédagogique(Akcayol *et al.*, 2004).

En effet, ils sont souvent désemparés par ce manque d'outils simples d'utilisation, conviviaux et peu dispendieux pour la mise en œuvre de systèmes flous. Selon Bélanger (2005), la mise au point d'un CF est une tâche ardue. Elle mentionne également qu'il est très difficile de suivre à la trace l'exécution d'un système flou en termes de règles activées et du degré d'appartenance des variables d'entrées. Par manque d'outils, elle a développé son propre outil sous la forme d'une *S-Function* de Matlab® codée en C pour faciliter la mise en œuvre de son contrôleur de son projet. Néanmoins, l'encapsulation de fonctions C pour la mise en œuvre d'un système flou nécessite un travail fastidieux. Cet exemple démontre l'importance d'avoir un outil simple et convivial pour la mise en œuvre de systèmes flous. Cela dit, il n'en demeure pas moins qu'il existe dans la littérature des articles où il est question de logiciels d'aide à l'apprentissage de la LF, et les plus représentatifs seront présentés ci-après.

Selon la littérature, les logiciels associés à la LF sont conçus de plusieurs façons selon les besoins des applications auxquels ils veulent répondre. Certains répondent à un besoin de mise en œuvre optimum en offrant une très grande panoplie de réglages et d'options de configuration. C'est le cas par exemple des logiciels professionnels tels que Fuzzy designer®[1] et Fuzzy-Control++®[2], qui offrent non seulement un ensemble de ressources telles qu'un

---

[1] Fuzzy Designer® peut être utilisé en combinaison avec les API Logix5000 de la compagnie Allen-Bradley®.

[2] FuzzyControl++® peut être utilisé en combinaison avec les API SIMATIC de la compagnie Siemens®.

niveau de capacité de commande évoluée, mais aussi un jeu de composants intégrés permettant l'exécution en temps réel du système flou.

Ces logiciels, de type professionnel, sont très performants. Ils permettent de réaliser des systèmes flous multivariable. L'inconvénient majeur des logiciels professionnels est l'apprentissage parfois long et difficile à réaliser suivant la complexité de l'application. Un autre inconvénient des logiciels professionnels connus est qu'il est nécessaire de disposer d'un matériel supplémentaire, parfois dispendieux, pour la mise en œuvre expérimentale. En plus de leurs coûts, ces logiciels professionnels présentent l'inconvénient de ne pas permettre une adaptation rapide à l'apprentissage des notions de base de la LF.

D'autres logiciels cherchent essentiellement à développer une approche pédagogique mieux adaptée pour l'apprentissage de la LF. Le logiciel FisPro présenté par Serge Guillaume du Cemagref se concentre principalement sur la compréhension du fonctionnement de la LF (Guillaume, 2005). Bien que FisPro intègre des outils à vocation pédagogique pour illustrer le mécanisme de raisonnement et d'autres permettant de mesurer la performance d'un système flou, il ne permet pas l'expérimentation et l'intégration en temps réel. De plus, il a l'inconvénient de ne pas permettre une adaptation rapide à l'apprentissage des notions de base de la LF à cause de sa convivialité et son environnement graphique peu efficace.

Celui présenté par Akcayol *et al.* (2004), bien qu'offrant différentes approches pédagogiques en ce qui concerne l'interface graphique, s'occupe particulièrement de la compréhension de la commande basée sur les performances de celle-ci dans un contexte de système dédié. Le logiciel ne permet pas une architecture ouverte au niveau du choix de l'application. De plus, les variables d'entrée et de sortie sont dédiées. L'application est un contrôle de moteur, dont les deux variables d'entrée et la variable de sortie du système flou sont, respectivement, l'erreur, la variation de l'erreur, et une sortie incrémentale.

Des logiciels commerciaux bien connus tels que Matlab/Simulink® et LabView® offrent une boîte à outils « fuzzy logic » (Foulloy, Boukezzoula et Galichet, 2006; Keller, 2000). Bien

que ces logiciels permettent l'apprentissage du processus de modélisation, de conception de systèmes à commande classique et à LF, seuls quelques étudiants de niveau avancé peuvent effectuer cet apprentissage dans un temps limité. Par conséquent, il convient plutôt aux enseignants et étudiants universitaires désirant faire l'apprentissage de la LF. Par contre, un des grands désavantages de ce logiciel est qu'il demande des connaissances particulières dans un langage de programmation. Ce type de logiciels ne permet donc pas de minimiser le temps pour l'apprentissage des notions des contrôleurs flous. L'apprentissage à l'aide de ces logiciels pourrait être long et coûteux à la fois pour les professeurs et les étudiants.

Des logiciels disponibles depuis peu, tels que FIDE$_{TM}$ élaboré par Aptronix et FuzzyTECH développé par Inform, proposent une conception pour générer les codes sources de plusieurs microcontrôleurs comme le 6805, le 68HC05, le 68HC11, le 8051 et le 80C196 (Alsaleh, 1998). De plus, certains logiciels, dont *FUDGE*® (FUzzy Design GEnerator) de la compagnie Motorola, permettent de programmer soi-même directement les inférences floues ainsi que les fonctions d'appartenance dans un langage de programmation, par exemple le C++, l'assembleur ou le Java™ (Belanger, 2005). Néanmoins, la complexité de l'utilisation de ces logiciels s'en trouve augmentée considérablement, ce qui élèvera conséquemment les exigences professionnelles nécessaires aux interventions les concernant.

En conclusion, les logiciels existants sont mal adaptés pédagogiquement pour faire l'apprentissage de toutes les étapes de fonctionnement d'un système à LF. La plupart de ces logiciels sont fondés sur l'implantation d'un modèle fermé dont les étapes de fonctionnement ne sont pas explicites et auxquels les utilisateurs n'ont pas accès. Ceci rend l'utilisabilité et la convivialité de ces logiciels très peu efficaces. Certains logiciels nécessitent l'apprentissage de concepts mathématiques avancés, ou la maîtrise des notions avancées de programmation, qui ne sont pas enseignés au niveau collégial.

2.4 Contrôleurs flous

Cette section consiste en une revue littéraire des principaux modèles de CF. Nous présentons les modèles de CF couramment utilisés en industrie et en recherche. Ces modèles sont présentés, car ils sont à l'origine de beaucoup de modèles construits pour le domaine du contrôle. Nous présentons les différentes structures des commandes en régulation de procédé, car il n'est pas possible de réaliser un CF sans connaître initialement certaines notions théoriques fondamentales.

2.4.1 Principales structures des contrôleurs flous

Au sein de la littérature, on constate qu'il existe différents modèles de commandes floues; on ne citera ici que les plus utilisés et ceux desquels nous nous sommes inspirés par la suite. La commande floue se divise en deux classes : les systèmes à commande floue, communément appelés les *contrôleurs flous*, et les contrôleurs PID flous qui sont une application spécifique des systèmes à commande floue calqués habituellement sur les performances des contrôleurs PID classiques (Tong-Tong, 1995). La majorité des applications au cours des deux dernières décennies appartiennent à la classe des contrôleurs PID flous (Mann, Bao-Gang et Gosine, 1999).

Les contrôleurs PID flous peuvent être divisés en trois types en ce qui concerne leurs applications : le type « direct action » (DA), le type « gain scheduling » (GS) et une combinaison des types DA et GS (Mann, Bao-Gang et Gosine, 1999). La majorité des applications utilisant des contrôleurs PID flous appartiennent au type DA.

Le type DA traite des systèmes flous purs, dans lesquels les signaux et le modèle du système de contrôle sont décrits par des sous-ensembles flous et des relations respectives. Cependant, la plupart des applications utilisent des descriptions de système basées sur des modèles numériques, par exemple, équations différentielles, avec un élément non linéaire de contrôle, le CF. La Figure 2.1 illustre la structure du contrôleur PID flou de type DA.

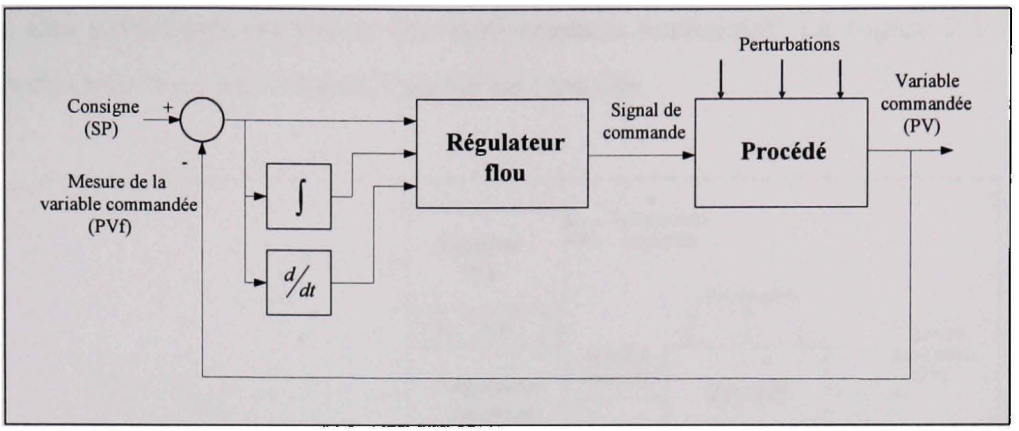


Figure 2.1 Contrôleur PID flou de type DA.

Dans le type GS, l'inférence floue est utilisée pour régler les paramètres d'un contrôleur PID classique. Le préréglage (self-tuning) s'effectue en analysant le comportement du procédé et les paramètres optimaux sont choisis selon les performances souhaitées. La Figure 2.2 illustre la structure du contrôleur PID flou de type GS.

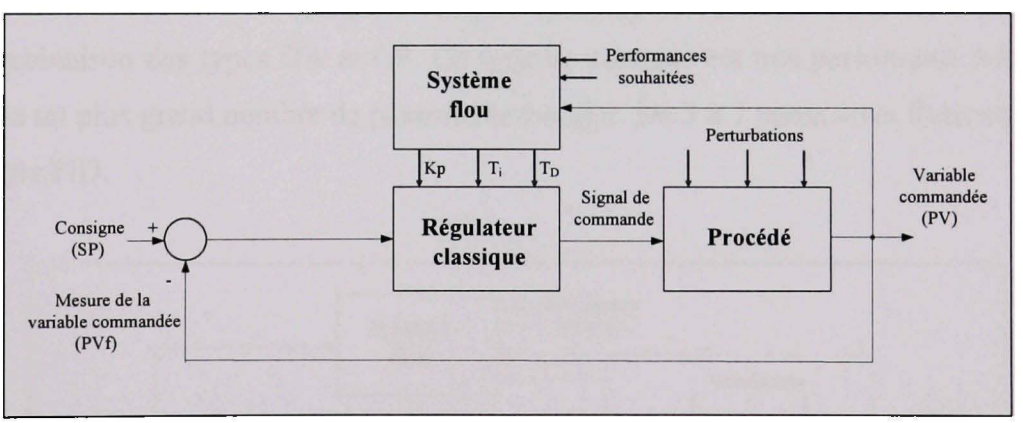


Figure 2.2 Contrôleur PID flou de type GS.

Le troisième type est semblable au deuxième type à la seule différence que le système flou est utilisé pour modifier les paramètres selon le comportement du procédé; le système flou observe en permanence l'erreur du contrôleur classique. Les paramètres sont adaptés de manière dynamique, et évolue continuellement selon une loi de commande adaptative et non

linéaire. Ces paramètres dépendent des performances souhaitées. La Figure 2.3 illustre la structure du contrôleur auto adaptatif par LF de type GS.

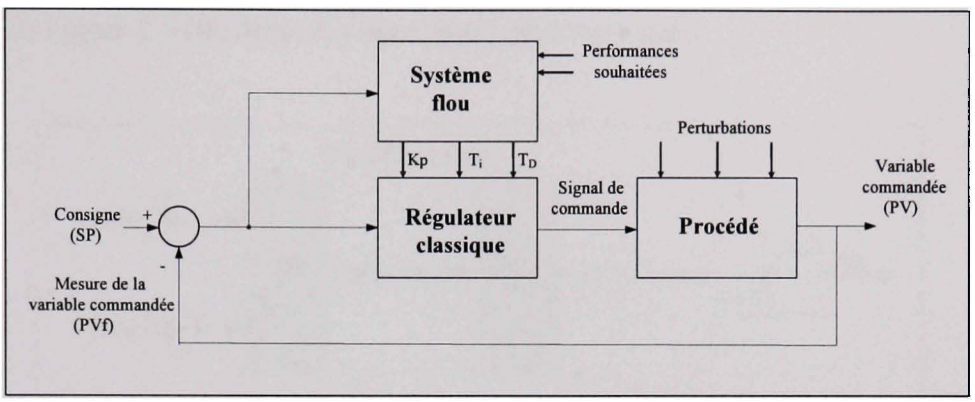


Figure 2.3 Contrôleur autoadaptatif par logique floue.

Le quatrième et dernier groupe utilise un système flou en parallèle avec un contrôleur PID classique pour aider ce dernier lors des perturbations. Le système flou ajoute une correction à celle du contrôleur PID classique pour réduire l'influence des perturbations. Ce contrôleur est une combinaison des types DA et GS. Ce type de contrôle est très performant, toutefois, il nécessite un plus grand nombre de paramètres à régler. De 3 à 7 paramètres flous en plus des 3 réglages PID.

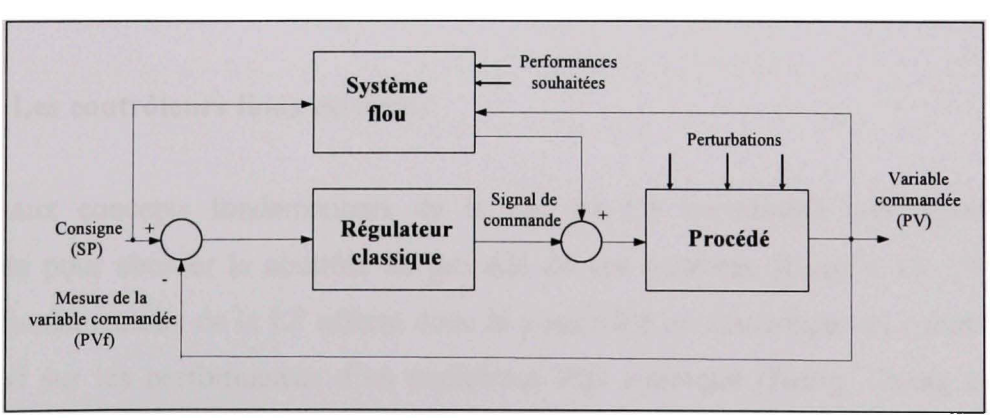


Figure 2.4 Contrôleur classique avec correction floue.

Selon Pivonka (2002) les CF peuvent être classés selon leur type de sortie : les contrôleurs flous à sortie directe et les contrôleurs flous à sortie incrémentale. Ils sont aussi appelés contrôleurs à action directe et contrôleur à action incrémentale (Hernandez, 2007). Les Figure 2.5 et Figure 2.6 illustrent ces deux types de contrôleurs.

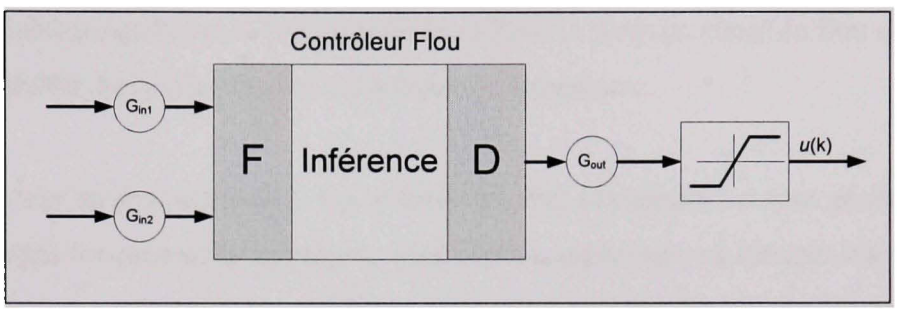


Figure 2.5 Contrôleur à sortie directe.

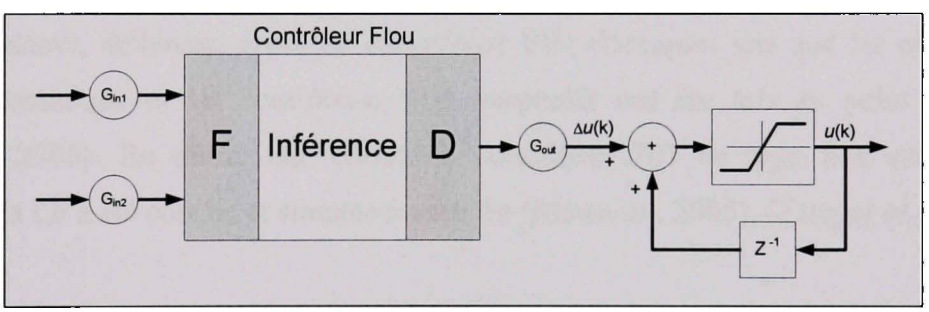


Figure 2.6 Contrôleur à sortie incrémentale.

2.4.2 Les contrôleurs flous existants

Associés aux concepts fondamentaux de la LF, les CF constituent une approche très intéressante pour aborder le contrôle de procédé de ces systèmes (Feng *et al.*, 2001). Les concepts fondamentaux de la LF offrent donc la possibilité de développer un contrôleur PID flou calqué sur les performances d'un contrôleur PID classique (Juang, Chang et Huang, 2008; Khan et Rapal, 2006; Lee, 1990). Étant donné que le contrôleur PID flou de type DA est le plus couramment utilisé, notre étude est limitée à ce type de contrôleur.

Aujourd'hui, de nombreuses recherches théoriques et pratiques ont montré la validité et la stabilité d'un CF; on peut mentionner notamment les travaux de Mohan et Sinha (2006), Carvajal *et al.* (2000), qui ont démontré que sous certaines conditions les CF offrent des performances de loin supérieures à leurs homologues, les contrôleurs PID classiques, lorsqu'ils sont utilisés directement sur des systèmes non linéaires. Les auteurs Lam *et al.* (2008) ont démontré qu'il existe une condition suffisante pour un contrôle flou qui permet de garantir la stabilité du système selon la méthode de Lyapunov.

En raison de leur structure linéaire, les contrôleurs PID classiques ne sont généralement pas efficaces lorsque les processus impliqués sont d'ordre supérieur, ou lorsque les systèmes ont des délais, ou encore lorsque les systèmes sont non linéaires. Ils ont également une piètre performance lorsque les systèmes sont complexes et vagues, sans modèles mathématiques précis, ainsi que lorsque les systèmes ont des incertitudes (Mansouri, 2005). Pour améliorer les performances, différents types de contrôleurs PID classiques tels que les contrôleurs à réglage automatique et les contrôleurs PID adaptatifs ont été mis au point récemment (Mansouri, 2005). En outre, une classe de contrôleur PID de type non conventionnel employant la LF a été conçue et simulée à cette fin (Mansouri, 2005), (Tang *et al.*, 2001).

Il est également possible d'en connaître beaucoup plus sur les différents modèles de CF pouvant être utilisés pour ce projet. Une étude comparative complète de ces structures a été menée par Mann *et al.* (1999). Les chercheurs Mohan et Sinha (2006) ont proposé un type de contrôleur PID flou claqué sur les performances d'un contrôleur PID classique. Ce type de contrôleur utilise trois entrées, l'erreur, la variation de l'erreur et l'accélération de l'erreur, ainsi qu'une sortie de type incrémentale. Ces contrôleurs PID flous sont essentiellement implantés en technologie numérique et il faut les discrétiser, donc approximer les opérateurs dérivé et intégrale. Pour ce faire, il est nécessaire de déterminer la dérivée de l'erreur entre un signal de consigne et le signal mesuré et l'intégrale de cette erreur (Bühler, 1994).

Le modèle le plus simple est le CF par interconnexion de type PI+PD proposé par Pivonka (2002). Ce CF est composé de l'association entre les deux contrôleurs flous de base (PI flou,

PD flou). Cette combinaison permet aux contrôleurs d'avoir de meilleures performances en régime transitoire et également en régime permanent. De plus, ce contrôleur a l'avantage d'avoir une BR réduite par rapport aux autres modèles de CF.

Par ailleurs, il existe d'autres modèles de contrôleurs flous qui s'inspirent de la commande PID floue standard. Parmi ces contrôleurs, on distingue le contrôleur PI+D flou (commande floue proportionnelle, intégrale plus dérivée) qui utilise une loi de commande PI floue additionnée à une loi de commande D floue (Tang *et al.*, 2001), (Misir, Malki et Guanrong, 1996). Ensuite, le contrôleur PID flou optimisé par un algorithme génétique (GA) qui modifie les fonctions d'appartenance du CF (Kermiche et Abbassi, 2008). En terminant, citons le modèle de contrôleur PID classique dont ses paramètres sont adaptés par un CF (Tipsuwanpom *et al.*, 2004). Chacun offre des particularités différentes, mais le CF qui attire principalement l'attention pour ce projet est le CF par interconnexion proposé par Pivonka (2002).

2.5 Synthèse de la revue de littérature

De façon générale, la commande par LF offre un intérêt particulier lorsqu'il n'y a pas de modèles mathématiques du procédé à contrôler ou lorsque le procédé à contrôler est complexe. En effet, l'application des contrôles à LF offre une solution attrayante pour les systèmes complexes ayant un comportement fortement non linéaire en raison de sa capacité à produire un raisonnement sur des termes proches du langage humain.

La revue de littérature a montré l'importance d'avoir des outils pédagogiques permettant l'apprentissage de la LF. En effet, bien que dans l'esprit de tout le monde, la LF soit simple de compréhension, sa mise en œuvre est toute autre. Les outils d'aide doivent permettre d'effectuer une mise en œuvre optimale. De plus, d'un point de vue pédagogique, ces outils doivent offrir à l'utilisateur la possibilité d'interpréter toutes les étapes de fonctionnement d'un système flou.

La revue de littérature a généralement été peu contributive en ce qui concerne les outils d'aide à l'apprentissage de la LF. Elle a montré que quelques chercheurs se sont penchés sur le problème que pose l'apprentissage des concepts de la LF. Bien que les concepts de la LF paraissent tout à fait simples, un fait qui apparaît de manière récurrente doit être pris en considération : il semble évident à la lecture des articles que la plupart des concepteurs de logiciels professionnels ne se rendent pas compte des efforts et des compétences nécessaires pour le développement des systèmes experts à LF et, par conséquent, ne permet pas une mise en œuvre simple. Précisons tout de même, à la décharge des compagnies, que l'aspect pédagogique n'est pas la vocation principale de ces logiciels.

Toutefois, ceci a pour conséquence de créer de la frustration chez les personnes qui veulent introduire les concepts de la LF dans leurs projets. L'utilisation des logiciels actuels leur impose des efforts colossaux afin d'obtenir un premier projet, et l'achat, à des coûts élevés, d'équipements nécessaires à l'implantation d'un système flou. Par conséquent, pour de nombreux partisans de la LF, en plus d'entraîner des coûts supplémentaires, l'apprentissage de la LF entre dans le cadre d'un apprentissage long des différentes fonctions et des nombreux paramètres à configurer pour l'utilisation des logiciels actuels de type professionnel. Malgré l'effort de la communauté, il existe un réel manque d'outils d'enseignement et de mise en œuvre de la LF. Les travaux de ce projet présentés dans ce document viennent combler cette importante lacune.

La revue de littérature a montré un bon éventail de modèles de CF. Notons que tous les contrôleurs que nous avons détaillés ne sont pas exhaustifs, mais représentent les contrôleurs les plus usuels. Nous avons détaillé le principe d'un CF et ses différents modèles. Ces notions présentées serviront également pour les travaux que nous aborderons dans les chapitres suivants. Les contrôleurs flous peuvent être implantés de différentes manières. Rappelons qu'il existe trois types : le type DA, le type GS et une combinaison des types DA et GS. La plupart des applications utilisent les contrôleurs PID flous de type DA.

Il ressort de cette revue de littérature que la majorité de ces applications appartiennent à la classe de structure des contrôleurs PID flous de type DA à deux entrées. Les articles consultés ont par ailleurs tous montré que le CF par interconnexion de type PI+PD était le plus simple d'application et de mise en œuvre. De plus, il nécessite une BR réduite par rapport aux autres modèles de CF. Ainsi, malgré les nombreux avantages des autres CF, dans le cadre de ce projet, le CF par interconnexion de type PI+PD a été choisi pour l'élaboration du CF de ce projet. La réalisation de ce CF offre la possibilité d'être calqué sur les performances d'un contrôleur PID classique; ainsi, le lien entre le contrôleur PID classique et le CF de ce projet pourra être fait. Nous traitons de la structure des CF par interconnexion de type PI+PD dans le chapitre 4. Dans le chapitre suivant, nous allons présenter les concepts fondamentaux de la LF.

CHAPITRE 3

CONCEPTS FONDAMENTAUX DE LA LOGIQUE FLOUE

Cette section présente les concepts fondamentaux de la LF[3]. Cette présentation ne veut pas se substituer aux nombreux ouvrages qui décrivent la LF de manière exhaustive. Il s'agit plutôt d'une introduction nécessaire qui permet la compréhension de la structure et des éléments essentiels retenus pour une mise en œuvre du logiciel développé dans le cadre de ce projet. Dans ce qui suit, nous présenterons d'abord la théorie des ensembles flous, puis nous préciserons le raisonnement en LF, nous définirons ensuite les éléments qui constituent un système flou, et finalement nous présenterons une application complète de la LF.

Ainsi, le résultat principal de ce chapitre est de colliger toutes les informations pertinentes au domaine de la LF, en faisant largement appel aux méthodes de mise en œuvre. Cette synthèse permettra de retenir les éléments essentiels sur la LF pour l'élaboration du logiciel.

3.1 Théorie des ensembles flous

3.1.1 Notions de base

La LF est une technologie pour le traitement de connaissances imprécises basées sur des termes linguistiques; elle s'approche donc de la démarche humaine en ce sens que les variables traitées ne sont pas des variables logiques binaires, mais plutôt des variables linguistiques se rapprochant du langage humain de tous les jours. Dans le but de distinguer

---

[3] Le lecteur peu familier avec les concepts de la logique floue et des systèmes d'inférence floue est invité à consulter, pour une présentation plus complète, les ouvrages cités dans la bibliographie.

la LF de la logique classique, nous présentons un bref rappel de la théorie des ensembles classiques.

Un ensemble classique se caractérise par ses frontières abruptes entre deux catégories d'éléments; ceux qui font partie de l'ensemble et ceux qui ne le font pas (vrai ou faux, blanc ou noir). À titre d'exemple, on désire classer un groupe d'individus par leur taille en définissant la catégorie des petits par une taille inférieure à 1,65 mètre et la catégorie des grands par une taille supérieure à 1,65 mètre. La Figure 3.1 (a) illustre deux fonctions en théorie des ensembles classiques, soient : une variable x (la *taille*) et un univers du discours U, les *individus*, avec A l'ensemble *petit* et B l'ensemble *grand.* Dans la théorie des ensembles classiques, soit qu'un élément appartient totalement à un ensemble soit qu'il ne lui appartient pas du tout. Autrement dit, tout membre de l'univers U est soit *petit* ou soit *grand.*

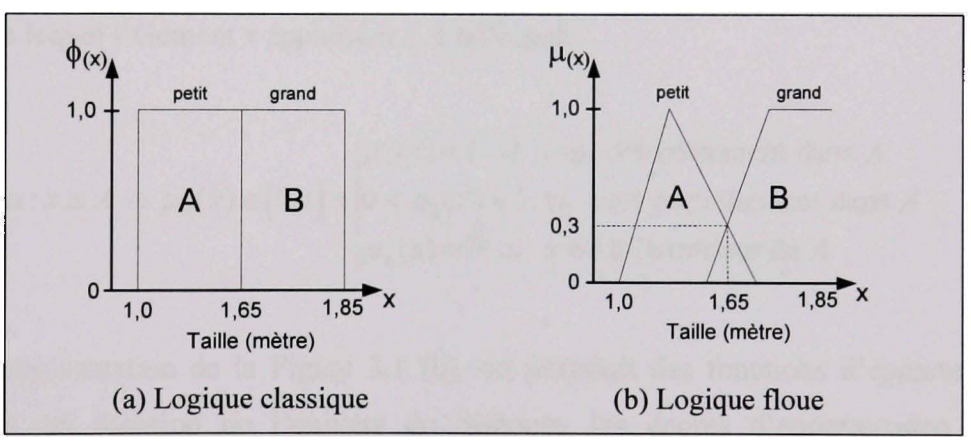


Figure 3.1 Logique classique vs logique floue.

Les fonctions qui caractérisent les deux sous-ensembles A et B, appelées les *fonctions caractéristiques*, ne permettent pas la représentation de tout cas intermédiaire d'éléments x sur la frontière de A et B (taille moyenne par exemple). On écrira pour un élément x appartenant à A :

$$\varphi : x \in A \rightarrow \varphi_A(x) \in \{0,1\} = \begin{vmatrix} \varphi_A(x) = 1 \;\; si \;\; x \in A \\ \varphi_A(x) = 0 \;\; si \;\; x \notin A \end{vmatrix} \tag{3.1}$$

Pour imiter l'esprit humain, la structure de la logique classique est limitée, étant donné qu'on ne peut exprimer des faits qu'avec *vrai* ou *faux* (0 ou 1). En effet, la logique classique ne considère pas la représentation intermédiaire; par contre, la théorie des ensembles flous donne une représentation de ces catégories vagues. En conclusion, la logique classique présente l'avantage de la simplicité, mais est assez éloignée de la logique utilisée naturellement par l'être humain.

L'approche floue se base sur des descriptions qualitatives du comportement d'un système. Ces descriptions, appelées *fonctions d'appartenance*, sont représentées par des variables linguistiques telles que *chaud*, *froid*, *jeune*, *vieux*, etc. Ces *fonctions d'appartenance* ont le grand avantage de constituer une représentation beaucoup plus proche du raisonnement humain puisqu'elle permet de faire intervenir des notions telles que « *plutôt petit* », « *assez grand* ». Un ensemble flou[4] est défini par une *fonction d'appartenance*, $\mu_A(x)$, qui décrit le degré avec lequel l'élément x appartient à $\boldsymbol{A}$ telle que :

$$\mu : x \in A \rightarrow \mu_A(x) \in [0,1] = \begin{cases} \mu_A(x) = 1 \;\; si \;\; x \text{ est complétement dans } A \\ 0 < \mu_A(x) < 1 \;\; si \;\; x \text{ est partiellement dans } A \\ \mu_A(x) = 0 \;\; si \;\; x \text{ est à l'extérieur de } A \end{cases} \qquad (3.2)$$

Dans la représentation de la Figure 3.1 (b), on introduit des fonctions d'appartenance qui définissent, en fonction de l'univers du discours, les degrés d'appartenance à chaque ensemble flou. Les limites ne varient pas soudainement, mais progressivement. Par exemple, un homme de 1,65 mètre fera partie à la fois de l'ensemble *grand* et à la fois à l'ensemble *petit*; cependant, avec un degré d'appartenance différent. Dans ce cas précis, l'individu appartient à l'ensemble *petit* avec une valeur $\mu_A(x)$ = 0,30 et à l'ensemble *grand* avec une

[4] Nous emploierons indifféremment les termes « ensemble flou » et « sous-ensemble flou ».

valeur $\mu_B(x) = 0{,}30$. Autrement dit, en théorie des ensembles flous, un élément appartient à un ensemble à un degré variant de 0 à 1 inclusivement. Un ensemble flou est totalement déterminé par sa fonction d'appartenance. Il est à noter que les ensembles classiques sont considérés comme un cas particulier des ensembles flous.

3.1.2 Notions caractéristiques d'un ensemble flou

Pour un ensemble flou, il existe diverses notions complémentaires qui sont utilisées dans les applications courantes (Uncu, 2003). Il s'agit de sa hauteur, son support et son noyau.

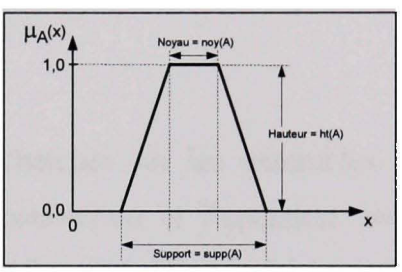


Figure 3.2 Notions caractéristiques d'un ensemble flou.

La hauteur d'un ensemble flou A, $ht(A)$, dans un univers du discours U est la borne supérieure de la fonction d'appartenance, qui est définie comme suit :

$$ht(A) = \max\left(\mu_A(x); x \in U\right) \tag{3.3}$$

Le support d'un ensemble flou A, $supp(A)$, dans un univers du discours U, est l'ensemble des éléments du domaine pour lesquels la réponse, $\mu_A(x)$, est non nulle, par rapport à A. Le support d'un ensemble flou est défini comme suit :

$$\text{supp}(A) = \left\{x \in U; \mu_A(x) \neq 0\right\} \tag{3.4}$$

Le noyau d'un ensemble flou A, $noy(A)$, dans un univers du discours U est l'ensemble des éléments du domaine pour lesquels le degré d'appartenance est égal à 1, par rapport à A. Le noyau d'un ensemble flou est défini comme suit :

$$\text{noy}(A) = \left\{x \in U; \mu_A(x) = 1\right\} \tag{3.5}$$

3.2 Opérateurs à logiques floues

Maintenant que nous avons une idée de ce que sont les ensembles flous, nous pouvons présenter les opérations de base sur les ensembles flous. Comme dans le cas des ensembles « classiques », les opérations logiques d'union (OU), d'intersection (ET) et de complémentarité (NON) peuvent être appliquées aux ensembles flous (Uncu, 2003). Leur définition n'est pas unique. Dans la littérature, il existe de nombreuses variantes pour ces opérateurs[5]; les définitions les plus souvent rencontrées sont : la fonction *max* et la fonction *min* (Mamdani et Zadeh), la fonction *produit* et la fonction *somme moins le produit* (Sugeno). Nous ne donnons ici que les plus communément utilisées, à savoir celles basées sur la LF de Zadeh.

Dans ses toutes premières recherches sur les ensembles flous, L. A. Zadeh a suggéré l'opérateur *minimum* pour l'intersection et l'opérateur *maximum* pour l'union de deux ensembles flous. Afin de clarifier ceci, nous présentons les trois principaux opérateurs logiques, en prenant comme exemple les deux ensembles flous de la Figure 3.3 soient : ***A*** un nombre flou; environ 10 et ***B*** un intervalle flou entre 12 et 18. Il est à noter que selon la théorie des ensembles flous, l'ensemble ***A*** est plus flou que l'ensemble ***B***.

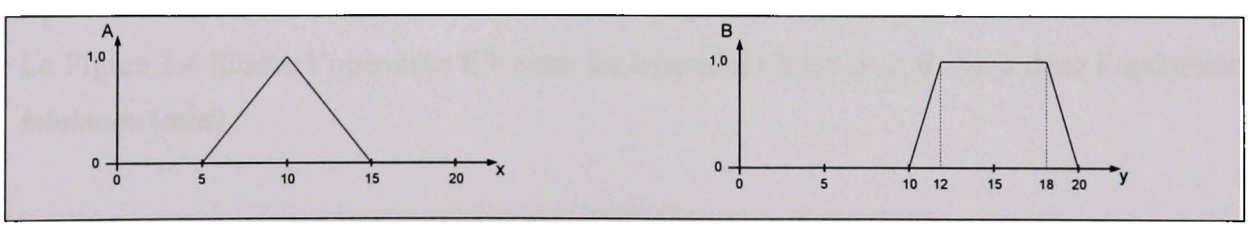


Figure 3.3 Ensembles flous A et B.

[5] Le lecteur peut consulter l'annexe I pour les opérateurs en logique floue les plus répandus.

3.2.1 Opérateur logique ET

L'opérateur logique **ET** classique est défini de la manière suivante : *A* **ET** *B* est VRAI si et seulement si *A* est VRAI et *B* est VRAI. Cette loi est aussi notée *A·B* que l'on nommera *conjonction*. Le Tableau 3.1 montre la table de vérité de l'opération **ET** classique.

Tableau 3.1
Table de vérité de l'opérateur logique ET classique

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

On définit l'opérateur **ET** flou comme l'intersection de deux ensembles flous ***A*** et ***B***, respectivement étant le plus petit ensemble flou contenant ***A*** et ***B***. Selon l'approche Zadeh, l'intersection de deux ensembles flous ***A*** et ***B*** de même référentiel ***U*** est définie par l'équation suivante :

$$\forall(x,y) \in XxY : \mu_{A\cap B}(x,y) = \min(\mu_A(x), \mu_B(y)) \tag{3.6}$$

La Figure 3.4 illustre l'opérateur **ET** entre les ensembles flous ***A*** et ***B***. On a donc l'opérateur *minimum* (***min***).

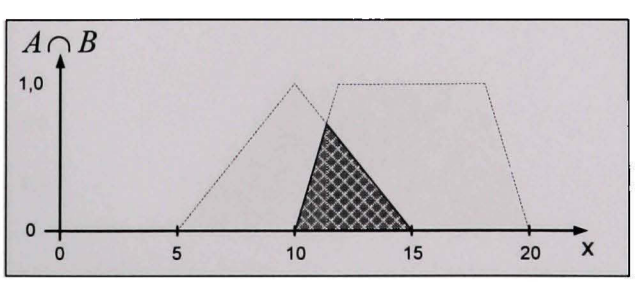


Figure 3.4 Intersection des ensembles flous A et B.

3.2.2 Opérateur logique OU

L'opérateur logique **OU** classique est défini de la manière suivante : *A* **OU** *B* est VRAI si et seulement si *A* est VRAI ou *B* est VRAI. (si *A* est vrai et que *B* est vrai aussi, alors *A* **OU** *B* est vrai). Cette loi est aussi notée *A+B* que l'on nommera *disjonction inclusive*. Le Tableau 3.2 montre la table de vérité de l'opération **OU** classique.

Tableau 3.2
Table de vérité de l'opérateur logique OU classique

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

On définit l'opérateur **OU** flou comme l'union de deux ensembles flous ***A*** et ***B***, respectivement étant le plus grand ensemble flou contenu dans ***A*** et dans ***B*** d'autre part. Selon l'approche Zadeh, l'union de deux ensembles flous ***A*** et ***B*** de même référentiel ***U*** est définie par l'équation suivante :

$$\forall (x,y) \in XxY : \mu_{A \cup B}(x,y) = \max(\mu_A(x), \mu_B(y)) \tag{3.7}$$

On parle d'opérateur *maximum* (***max***). Cette opération est représentée à la Figure 3.5.

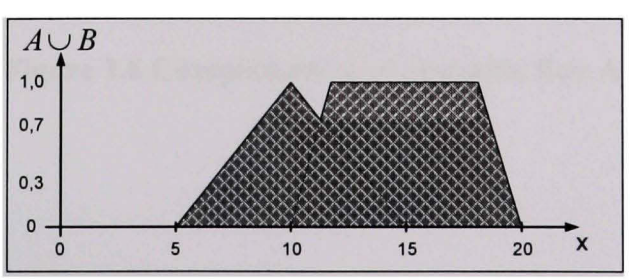


Figure 3.5 Union des ensembles flous A et B.

3.2.3 Opérateur logique NON

L'opérateur logique **NON** est défini de la manière suivante : le contraire de A est VRAI si et seulement si A est FAUX. Le contraire de A est noté $\overline{A}$ que l'on nomme la *négation* ou le *complément*. Le Tableau 3.3 montre la table de vérité de l'opération **NON** classique.

Tableau 3.3
Table de vérité de l'opérateur logique NON classique

A	$\overline{A}$
0	1
1	0

Selon l'approche Zadeh, le complément d'un ensemble flou A est défini par l'équation suivante :

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x) \tag{3.8}$$

La Figure 3.6 montre l'opérateur **NON** de l'ensemble flou A.

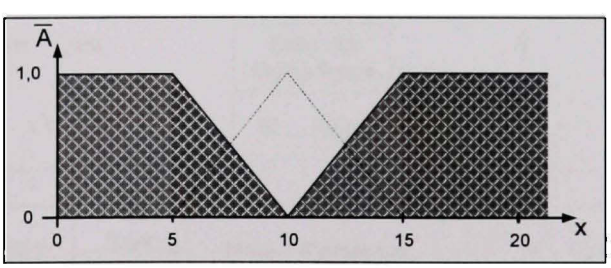


Figure 3.6 Complément de l'ensemble flou A.

3.3 Systèmes flous

La conception d'un système flou est constituée de trois étapes principales : la conversion des entrées en valeurs floues, l'évaluation des règles et la conversion du résultat des règles en une valeur numérique de sortie. La première étape, appelée la *fuzzification*, permet de découper une entrée en zones – fonctions d'appartenance que l'on désignera par des variables linguistiques. La seconde, soit l'inférence des règles, consiste en une évaluation des règles d'inférence entre les entrées et les sorties. La troisième étape, soit la *défuzzification*, consiste à convertir le résultat flou de l'inférence des règles en une valeur de sortie finale précise.

Un système flou est donc formé de trois étapes comme indiqué sur la Figure 3.7. La première, l'étape de *fuzzification* transforme les valeurs numériques en degrés d'appartenance aux différents ensembles flous de la partition. La seconde étape concerne le module d'inférence, qui est constitué de deux blocs, le moteur d'inférence et la *base des règles*. Enfin, l'étape de *défuzzification* qui permet d'inférer une valeur nette (précise), utilisable en commande par exemple, à partir du résultat de l'agrégation des règles.

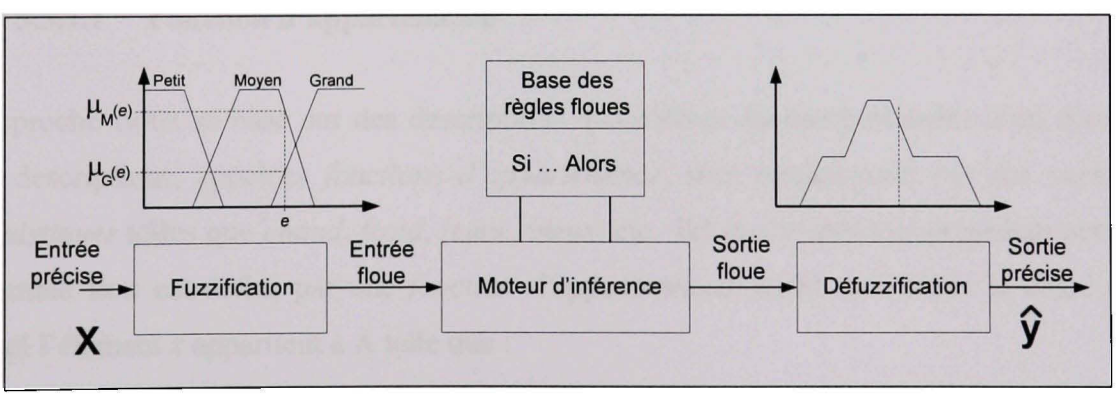


Figure 3.7 Système flou.

(Tiré de Serge Guillaume - FISPRO, 2007)

3.3.1 *Fuzzification*

La *fuzzification* est le premier traitement qui entre en compte dans la structure d'un système flou. Elle consiste à donner un degré d'appartenance à une valeur réelle d'entrée en fonction des ensembles flous. Dans un système flou, il faut rendre floues (fuzzifier) les entrées et les sorties du système. Pour le système flou, la *fuzzification* des variables est une étape importante du processus de mise en œuvre. Les caractéristiques de cette étape sont habituellement déterminées par des experts ou des opérateurs qualifiés travaillant sur le processus et recourant le plus souvent à leurs connaissances. De plus, les performances du système flou seront influencées par la *fuzzification* (Flaus, 1994). Les étapes de la *fuzzification* consistent à :

1. Établir les *variables linguistiques*;
2. Établir les quantificateurs flous (nombre de *valeurs linguistiques*);
3. Attribuer une signification numérique à chaque quantificateur flou : *fonction d'appartenance*.

3.3.1.1 Fonction d'appartenance

L'approche floue se base sur des descriptions qualitatives du comportement d'un système. Ces descriptions, appelées *fonctions d'appartenance*, sont représentées par des *variables linguistiques* telles que *chaud, froid, jeune, vieux*, etc. Tel que mentionné précédemment, un ensemble flou est défini par une *fonction d'appartenance,* $\mu_A(x)$, qui décrit le degré avec lequel l'élément x appartient à **A** telle que :

$$\mu : x \in A \rightarrow \mu_A(x) \in [0,1] = \begin{cases} \mu_A(x) = 1 \;\; si \;\; x \text{ est complétement dans } A \\ 0 < \mu_A(x) < 1 \;\; si \;\; x \text{ est partiellement dans } A \\ \mu_A(x) = 0 \;\; si \;\; x \text{ est à l'extérieur de } A \end{cases} \qquad (3.9)$$

3.3.1.2 Univers du discours et classes d'appartenance

L'univers du discours d'une variable couvrira l'ensemble de la gamme d'une variable d'entrée ou de sortie. La LF est basée sur des variables floues dites *variables linguistiques* à *valeurs linguistiques* dans l'univers du discours **U**. L'univers du discours est partitionné en plusieurs sous-ensembles flous représentés par leurs *fonctions d'appartenance*. À chaque fonction d'appartenance, une valeur linguistique est associée. Chaque *valeur linguistique* constitue alors un ensemble flou de l'univers du discours.

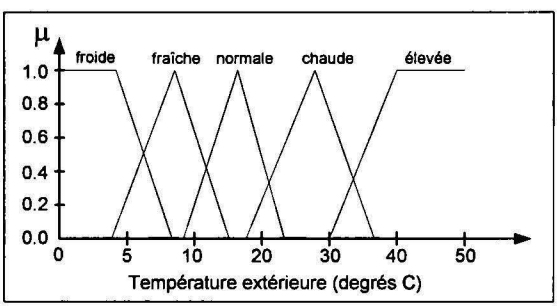


Figure 3.8 Univers du discours « température ».

La Figure 3.8 représente l'univers du discours de la *variable linguistique « température »* dont la gamme est de 0 °C à 50 °C. Les prédicats *« froide, fraîche, normale, chaude* et *élevée »* sont les *valeurs linguistiques* de la *variable linguistique « température »*. Pour chacun de ces prédicats, on associe une *fonction d'appartenance* $\mu_{prédicat}(x)$. À titre d'exemple, si l'on désire représenter la manière de classer un groupe d'individus par leur âge par les ensembles flous *jeune*, *moyen* et *vieux*, on pourrait procéder comme dans la Figure 3.9.

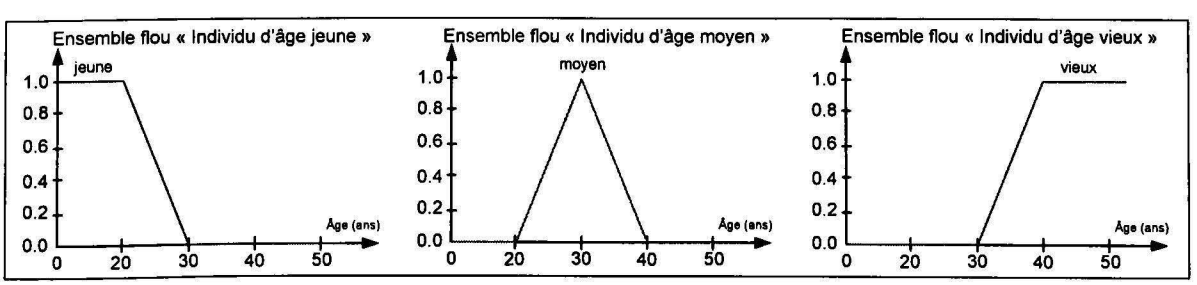


Figure 3.9 Ensembles flous jeune, moyen et vieux.

En regroupant les trois fonctions d'appartenance, la partition floue forme l'univers du discours *âge* qui est illustrée à la Figure 3.10.

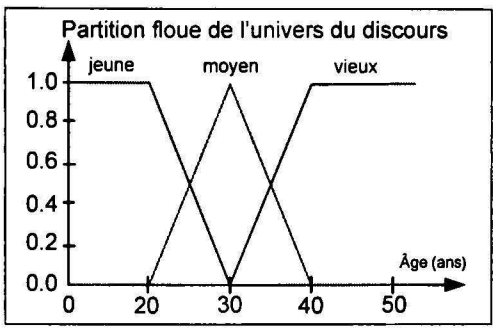


Figure 3.10 Partition floue de l'univers du discours « âge ».

Nous avons dit précédemment que l'opération de *fuzzification* consiste à déterminer le degré d'appartenance d'une valeur, mesurée par exemple, à un ensemble flou. Ainsi, un individu âgé de 28 ans appartient à l'ensemble *jeune* à 25 % et à l'ensemble *moyen* à 75 %, et il n'appartient pas à l'ensemble *vieux*. Le degré d'appartenance de l'individu est donc $\mu_{jeune}(x) = 0.25$, $\mu_{moyen}(x) = 0.75$ et $\mu_{vieux}(x) = 0$ (voir Figure 3.11).

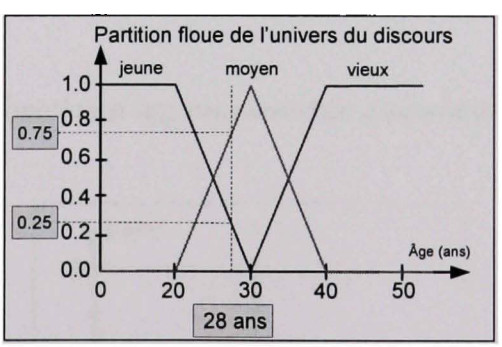


Figure 3.11 Degré d'appartenance d'un individu de 28 ans.

3.3.1.3 Les différentes formes des fonctions d'appartenance

L'univers du discours est partitionné en plusieurs sous-ensembles flous représentés par leurs *fonctions d'appartenance*. Cette *fonction d'appartenance* peut être représentée par plusieurs formes, mais les plus usuelles sont triangulaires, trapézoïdales ou gaussiennes. Lorsque

l'univers du discours est défini par des segments de droite, exemple par des formes triangulaires ou trapézoïdales, cette partition est dite *linéaire par morceaux*. Ces dernières formes sont souvent utilisées, car elles sont simples et comportent des zones où la notion est vraie, des zones où elle est fausse, ce qui rend plus naturelle l'acquisition de l'expertise.

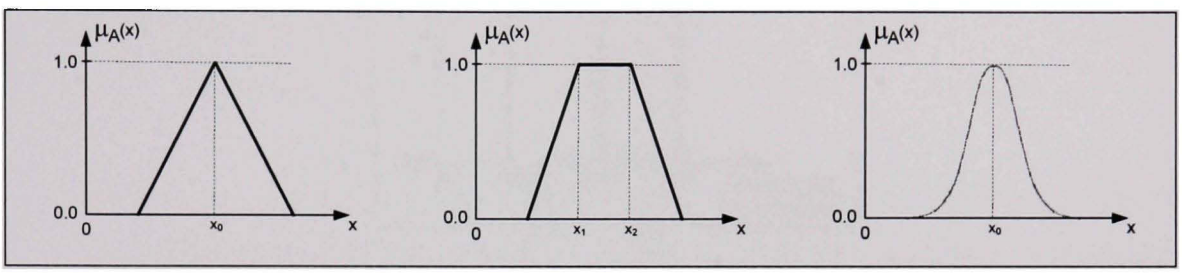


Figure 3.12 Les principales formes de fonctions d'appartenance.

Lorsqu'une fonction d'appartenance est partout nulle, sauf en un point, on a un *singleton.* Un *singleton* est défini par une *fonction d'appartenance, $\mu_A(x)$*, qui décrit le degré avec lequel l'élément *x* appartient à **A** telle que :

$$\mu : x \in A \rightarrow \mu_A(x) \in \{0,1\} = \begin{cases} \mu_A = 1 \ \ pour\ x = x_0 \\ \mu_A = 0 \ pour\ x \neq x_0 \end{cases} \qquad (3.10)$$

La Figure 3.13 illustre une *fonction d'appartenance* de type *singleton.*

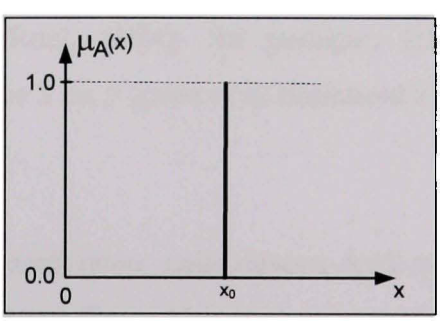


Figure 3.13 Fonction d'appartenance de type singleton.

Les *fonctions d'appartenance* de type *singleton* sont utilisées plus particulièrement pour une grandeur de sortie d'un système flou. La Figure 3.14 illustre les fonctions d'appartenance de

type *singleton* pour un système de contrôle d'un feu de circulation. Par exemple, la couleur du feu tricolore est de la classe « Le feu est vert ».

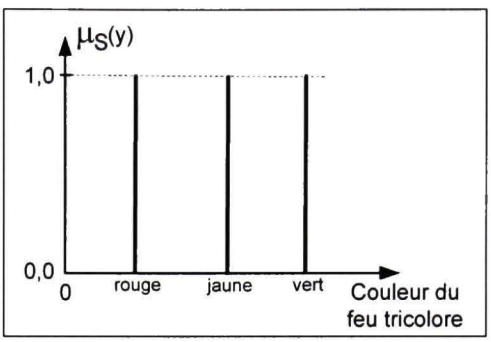


Figure 3.14 Fonctions d'appartenance de type singleton pour une sortie.

3.3.1.4 Considération sur les fonctions d'appartenance

Après avoir déterminé les variables et les valeurs linguistiques des fonctions d'appartenance, il faut choisir le nombre et la forme des fonctions d'appartenance. Le nombre de fonctions d'appartenance dans un univers du discours dépend essentiellement de l'application et de l'expertise. Toutefois, il est d'usage courant de partitionner l'univers du discours en 3, 5, 7, 9 ou 11 zones. Un plus grand nombre de zones augmente la sensibilité de la commande floue. En revanche, la littérature précise qu'un univers de discours partitionné en un trop grand nombre de zones diminue la cadence du fonctionnement du système flou sans augmenter les performances pour autant (Ruel, 1994). En pratique, les systèmes flous utiliseront, habituellement, une partition de 3 ou 5 zones et se limiteront à une partition ne dépassant pas 7 zones.

Lors de l'élaboration de la *fuzzification*, nous devons également attribuer une signification numérique à chaque quantificateur flou – *Fonction d'appartenance*. On sait qu'un ensemble flou est caractérisé par un support et un noyau. Les ensembles flous de type trapézoïdal, par conséquent, utilisent deux points pour le support et deux points pour le noyau. La Figure 3.15 montre un exemple d'un ensemble flou B avec la partition suivante : 43, 70, 85 et 110.

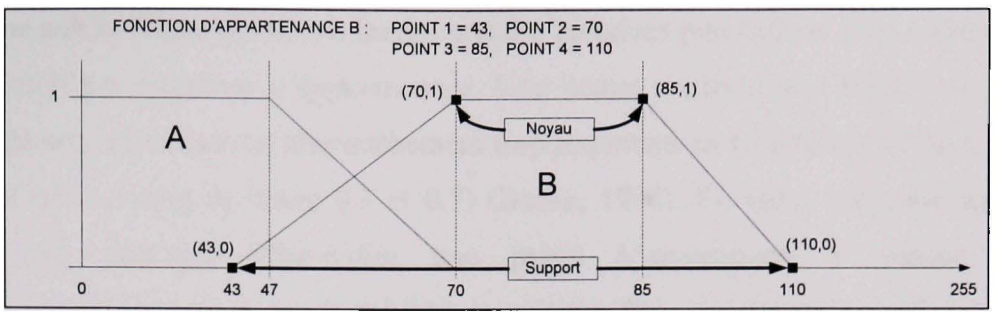


Figure 3.15 Partition d'un ensemble flou de type trapézoïdal.

L'assignation des valeurs numériques de chaque quantificateur flou consiste à définir les valeurs numériques du noyau et du support de chaque *fonction d'appartenance.* Il est à noter que pour une forme triangulaire le noyau est nul; par le fait même, les valeurs numériques du noyau sont identiques. Il existe plusieurs façons de définir les valeurs numériques de chaque *fonction d'appartenance*; elles peuvent être symétriques et distribuées de manière équidistante. Elles peuvent également être symétriques et distribuées de manière non équidistante. Et finalement, elles peuvent être non symétriques et non équidistantes. La Figure 3.16 illustre les différentes répartitions d'un univers du discours.

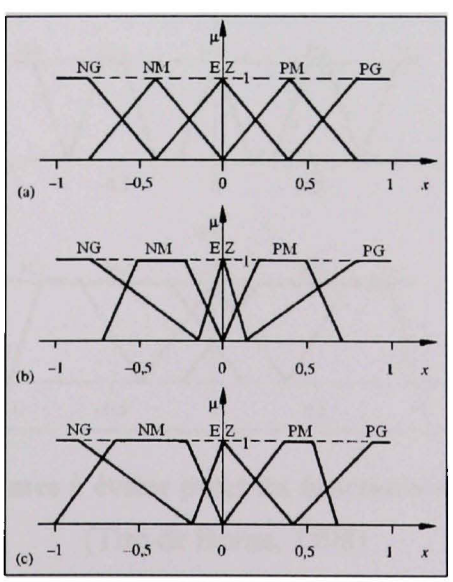


Figure 3.16 Différentes formes possibles pour les fonctions d'appartenance (a) symétriques et équidistantes, (b) symétriques et non équidistantes et (c) non symétriques et non équidistantes.

(Tiré de Borne, 1998)

Quelle que soit la forme choisie, il faudra prendre certaines précautions dans l'élaboration et la disposition des *fonctions d'appartenance*. Une bonne partition sera formée de *fonctions d'appartenance* en évitant un chevauchement trop important ou trop faible de deux *fonctions d'appartenance* contiguës (entre 0.3 et 0.7) (Borne, 1998). En effet, une zone morte dans l'univers du discours, c'est-à-dire une partie n'appartenant à aucune *fonction d'appartenance*, conduira à une instabilité du système flou. À l'inverse, un chevauchement trop important conduira à un appauvrissement des performances du système flou. De manière générale, le chevauchement des zones est de 50 % par rapport à l'axe des ordonnées.

Un autre critère est d'éviter, pour les formes simples, de permettre qu'une partie de l'univers du discours puisse être représentée par plus de deux variables linguistiques dont la fonction d'appartenance est non nulle. Ce qui créerait ainsi de l'imprécision lors de l'étape de calcul de la *fuzzification*. De même, on préférera les triangles ou trapèzes aux gaussiennes ou autres fonctions compliquées, dévoreuses d'espace mémoire et nécessitant un temps de calcul plus long lorsque le contrôleur est en fonction.

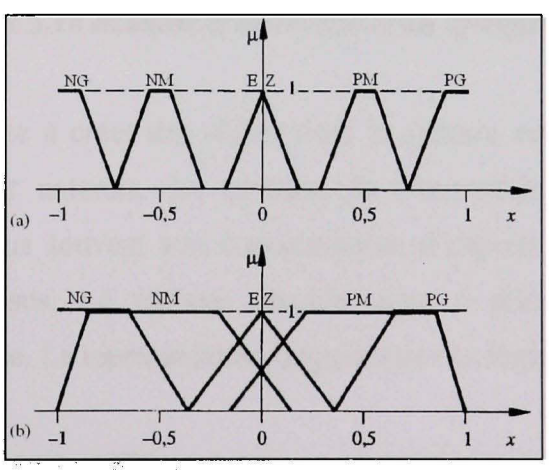


Figure 3.17 Formes à éviter pour les fonctions d'appartenance.
(Tiré de Borne, 1998)

En conclusion, la *fuzzification* consiste à déterminer le degré d'appartenance de chaque *variable linguistique*, pour chaque cycle, sur laquelle porteront les règles d'inférence qui détermineront la *défuzzification*.

3.3.2 Différentes méthodes d'inférences floues

L'inférence floue est le processus d'élaboration des relations qui existent entre les variables d'entrées (exprimées comme variables linguistiques) et la variable de sortie (également exprimée comme variable linguistique). Ces relations fournissent ensuite *la base des règles* **(BR)** à partir de laquelle les décisions peuvent être prises. Le processus de l'inférence floue implique tous les éléments qui sont décrits dans les sections précédentes : les fonctions d'appartenance, les opérateurs à LF et les règles de type « Si...Alors ». Le module d'inférence est constitué de deux blocs, le moteur d'inférence et la *base des règles*.

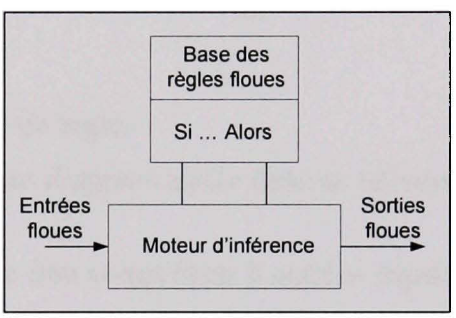


Figure 3.18 Module d'inférence d'un système flou.

La première étape consiste à créer des règles dont la syntaxe est très stricte bien que les termes utilisés paraissent naturels. La création de l'ensemble des règles de contrôle linguistique recourt le plus souvent aux connaissances d'experts ou d'opérateurs qualifiés travaillant sur le processus. Le réglage découle donc a priori des connaissances du fonctionnement du système. La représentation linguistique des règles est la suivante :

Si *antécédent 1* **ET** *antécédent 2* **ALORS** *conséquence 1*

où :

Antécédent : variable d'entrée

Conséquence : variable de sortie

ET : opérateur de base (parfois pour certaines applications l'opérateur est **OU**)

Exemple:

> **Si** température élevée **ET** sol sec **ALORS** durée d'arrosage longue **OU**
> **Si** température normale **ET** sol humide **ALORS** durée d'arrosage courte

La quantité de règles dépend du nombre d'entrées et du nombre de zones de chacune d'elles. Si l'on considère n univers de discours U_i pour les entrées du système flou et que chaque univers U_i est partitionné en m_i fonctions d'appartenance floues, alors le nombre maximal de règles est donné par l'équation 3.11 (Elqaq, 2005; Tong-Tong, 1995).

$$r_{\max} = \prod_{i=1}^{n} m_i \tag{3.11}$$

où :

r_{max} : nombre maximal de règles

m_i : nombre de fonctions d'appartenance dans un univers du discours

À titre d'exemple, un système flou comportant 2 entrées segmentées chacune en 5 fonctions d'appartenance (ou ensembles flous) génèrera une base de 25 règles maximum. Du fait que plusieurs règles doivent être évaluées à chaque cycle de fonctionnement du système flou, le nombre d'entrées aura une incidence sur le nombre de calculs que devra effectuer le système à chaque cycle. De plus, l'élaboration des règles étant réalisée par des experts ou opérateurs qualifiés travaillant sur le processus et recourant le plus souvent à leurs connaissances, on comprendra que plus la dimension de la BR du système flou est grande, plus difficile sera l'élaboration des règles d'inférence et plus la cadence du fonctionnement du système flou diminuera.

> « Un expert humain peut réagir efficacement pendant le contrôle d'un système sans avoir à chaque fois à utiliser toutes les règles. Il peut en effet contextualiser la base, ce qui lui permettra de sélectionner les règles appropriées au cas qui l'occupe. Cette optimisation est nécessaire à cause des capacités limitées de l'homme de mémoriser une grande quantité de règles. » (Eksioglu, 2000, p. 65)

Par conséquent, un système complexe qui aurait un nombre très élevé de règles entraînerait une limitation des performances du système flou. Lorsqu'on a établi la base des règles qui décrivent le fonctionnement, il faut choisir la méthode d'inférence floue, c'est-à-dire la méthode que le système doit employer pour calculer la fonction de sortie à l'aide des relations qui existent entre les variables d'entrées et la variable de sortie utilisant la base des règles.

L'inférence floue se compose de deux étapes : l'implication des antécédents et l'agrégation des règles. L'implication des antécédents consiste à déterminer par un opérateur flou le poids de chacune des conséquences des règles impliquées. Pour déterminer l'implication, il faut utiliser un opérateur de conjonction traduisant le **ET**. Quant à l'agrégation, elle permet de faire la synthèse des solutions de chaque implication. Cette synthèse nous permet de déterminer la variable de sortie finale floue. Pour déterminer cette variable, à l'aide de l'agrégation, il faut utiliser un opérateur de disjonction traduisant le **OU**. En prenant par exemple un ensemble de règles floues du genre :

R_1 : **Si** $(x \in A_1)$ **ET** $(y \in B_1)$ **ALORS** $(z \in C_1)$ **OU**
R_2 : **Si** $(x \in A_2)$ **ET** $(y \in B_2)$ **ALORS** $(z \in C_2)$ **OU**
R_3 : **Si** $(x \in A_3)$ **ET** $(y \in B_3)$ **ALORS** $(z \in C_3)$ **OU**
R_4 : **Si** $(x \in A_4)$ **ET** $(y \in B_4)$ **ALORS** $(z \in C_4)$

L'implication sous forme de conjonction de la représentation numérique des règles est définie par une norme triangulaire, appelée la *t-norme* (Bouchon-Meunier, Foulloy et Ramdani, 1998). Par conséquent, pour les quatre règles ci-dessus, on a donc :

$$f_{R_i}(x,y) = T\left(\mu_{A_i}(x), \mu_{B_i}(y)\right) \tag{3.12}$$

où

f_{R_i} est la fonction d'appartenance *i* de la relation floue (règle) *i*;

T est la norme triangulaire représentant la conjonction.

L'agrégation sous forme de disjonction de la représentation numérique de la synthèse des solutions de chaque implication est définie par une conorme triangulaire, appelée la *t-conorme*. Ainsi, l'ensemble flou inféré pour la variable de sortie U_{sf} est défini par le *modus ponens* généralisé suivant :

$$\mu_C(z) = \perp\left[T\left(\mu_{A_i}(x), \mu_{B_i}(y)\right)\right] \tag{3.13}$$

où

$\perp$ est la conorme triangulaire représentant la disjonction.

Dans ce cas, la variable de sortie floue U_{sf} est définie par ses degrés d'appartenance $\mu_C(z)$ aux différents ensembles C_i.

Dans la littérature, il existe plusieurs types de systèmes d'inférence floue qui peuvent être mis en œuvre dans les systèmes flous (Yen *et al.*, 1995). Par exemple, la méthode de *Mamdani* qui utilise les opérateurs *min* et *max* pour l'implication et l'agrégation, ou encore la méthode de Larsen qui utilise un *produit* et l'opérateur *max*, ou celle de *Sugeno* (Takagi, Yamaguchi et Sugeno, 1992) qui utilise une fonction linéaire ou une constante pour les signaux de sortie (Dadone, 2001). Les deux méthodes qui sont les plus largement utilisées en pratique sont la *méthode de Mamdani* et la *méthode de Sugeno* (Jihong, 1993). Ces deux types de systèmes d'inférence varient quelque peu dans la façon dont les résultats sont déterminés.

Les *modèles de Mamdani* permettent une description linguistique du système par une base des règles basée sur l'approche proposée par Zadeh (Nakoula, 1997). La méthode d'inférence floue de type *Mamdani* est la méthodologie la plus couramment utilisée. Rappelons que la *méthode Mamdani* a été parmi les premières à être utilisée dans les systèmes de contrôle construit en utilisant la théorie des ensembles flous. Bien que le processus d'inférence de type *Mamdani* n'ait pas été retenu dans la mise en œuvre du logiciel découlant de ce projet, nous la décrivons tout de même dans la section suivante.

Les *modèles linguistiques*, ou *modèles de Mamdani* ont certaines limites. Entre autres, Nakoula (1997, p. 31) spécifie que « Les modèles linguistiques ne permettent pas d'intégrer directement des connaissances objectives sur le système qui ne sont pas exprimées par des ensembles flous », ce qui démontre l'intérêt de la *méthode de Sugeno*. *Sugeno* et ses collaborateurs ont proposé un modèle de type numérique, dans lequel les parties conséquentes des règles du CF sont une représentation fonctionnelle numérique, ou dans le cas simplifié, par des nombres réels au lieu des sous-ensembles flous (Nakoula, 1997). Ceci permet de prendre en compte des connaissances exprimées sous forme analytique décrivant la structure physique d'un système. Avec ce modèle, utilisant des nombres réels pour la sortie, l'ensemble flou de l'inférence des conséquences sera un ensemble flou discret avec un nombre fini de points, ce qui peut simplifier grandement le calcul de l'algorithme de *défuzzification* (Khan et Rapal, 2006). Nous présenterons les principes de base de la *méthode de Sugeno* à la section 3.3.2.2.

3.3.2.1 Inférence floue de type Mamdani

Nous avons vu qu'on définit l'intersection (conjonction) et l'union (disjonction) à l'aide de deux relations floues, la t-norme et la t-conorme. Leurs définitions ne sont pas uniques (Alata, 2001). Dans le *modèle de Mamdani*, la conjonction (des antécédents) est usuellement interprétée par l'opération *min* (Zadeh) et la disjonction (des règles) comme le *max*, appelée la *méthode max-min*. La t-norme et la t-conorme, selon le *modèle de Mamdani* sont définies comme suit :

$$\textbf{t-norme :}\ \mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x)) \tag{3.14}$$

$$\textbf{t-conorme :}\ \mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x)) \tag{3.15}$$

où

$\cap$ est l'opérateur d'intersection représentant la conjonction *min* (t-norme).

$\cup$ est l'opérateur d'union représentant la disjonction *max* (t-conorme).

Ainsi, l'implication utilise l'opérateur *min* et l'agrégation l'opérateur *max*. La Figure 3.19 illustre le modèle d'inférence de *Mamdani*.

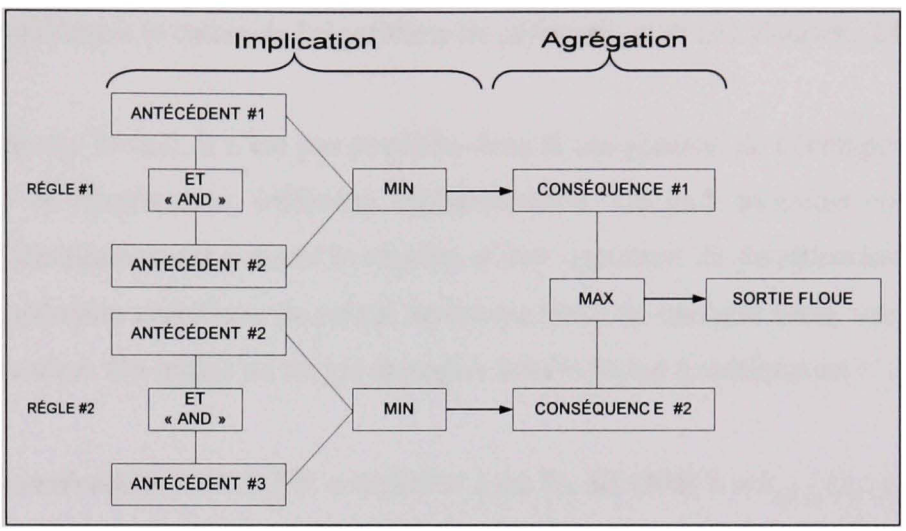


Figure 3.19 Méthode d'inférence MAX-MIN (Mamdani).

3.3.2.2 Inférence floue de type Sugeno

Le processus d'inférence floue de type *Mamdani*, dont nous avons fait référence à la section précédente, est connu comme la méthodologie la plus commune. Par contre, plusieurs contrôleurs flous utilisent des règles d'inférences où la variable de sortie est obtenue en relation avec les entrées, appelée *la méthode d'inférence floue de type Sugeno* (Uncu, 2003). Dans cette section, nous discuterons du processus d'inférence floue de type *Sugeno*, ou Takagi-Sugeno-Kang (TSK). Introduite en 1985 par les chercheurs Takagi, Sugeno et Kang, elle est semblable à la *méthode Mamdani* à bien des égards (Al-Zahrani, 2005). Les deux premières parties de l'inférence floue, qui traitent de la *fuzzification* des entrées et de l'application de l'opérateur flou, sont exactement les mêmes. La principale différence entre les *méthodes de Mamdani* et *Sugeno* est que les fonctions d'appartenance des sorties de type *Sugeno* sont soit linéaires ou soit constantes. Une règle typique dans un processus d'inférence floue de type *Sugeno* a la forme suivante :

Si *antécédent 1* = x **ET** *antécédent 2* = y **ALORS** Sortie $z = ax + by + c$

Pour un modèle de type *Sugeno* d'ordre zéro, la sortie z est égale à une constante (a=b=0). Avec ce modèle, utilisant des nombres réels pour la sortie, l'ensemble flou de l'inférence des conséquences sera un ensemble flou discret avec un nombre fini de points, ce qui peut simplifier grandement le calcul de l'algorithme de *défuzzification* (Al-Zahrani, 2005).

D'un point de vue formel, il n'est pas possible, dans le cas général, de décomposer un CF de type *Sugeno* en *fuzzification*, inférence, *défuzzification*. On peut toutefois considérer son principe de fonctionnement comme le résultat d'une opération de *fuzzification* symbolique suivi d'une opération spécifique de calcul agrégeant ainsi, en quelque sorte, une inférence et une *défuzzification*. On utilise ici un jeu de règles dont la forme générique est :

Si *antécédent 1* est A_i **ET** *antécédent 2* est B_j, **ALORS** $u = h_{g(i,j)}(x_0, y_0)$

Exemple :
Si température est élevée **ET** humidité est sèche, **ALORS** $u = 45 + 0.5 \cdot x + 5 \cdot y$

3.3.3 Règles floues

3.3.3.1 Évaluation des règles

L'évaluation des règles est une étape très importante du système d'inférence. Elle permet de calculer la valeur de la variable de sortie finale floue à partir des entrées floues issues de la *fuzzification* et de l'ensemble de la base des règles en utilisant une *méthode d'inférence* (Mizumoto, 1995). Dans un premier temps, nous allons présenter une évaluation des règles selon le *modèle de Mamdani*, appelée *la méthode max-min*. Considérerons, la suite de raisonnements flous suivante :

Règle 1 : A_1 **ET** $B_1 \Rightarrow C_1$

Règle 2 : A_2 **ET** $B_2 \Rightarrow C_2$

$\vdots$

Règle n : A_n **ET** $B_n \Rightarrow C_n$

Fait : x_0 et y_0

Conséquence : C'

où

A_i est un ensemble flou de X;

B_i est un ensemble flou de Y;

C_i est un ensemble flou de Z;

$x_0 \in X$ et $y_0 \in Y$

La Figure 3.20 illustre l'évaluation des règles selon *la méthode max-min.*

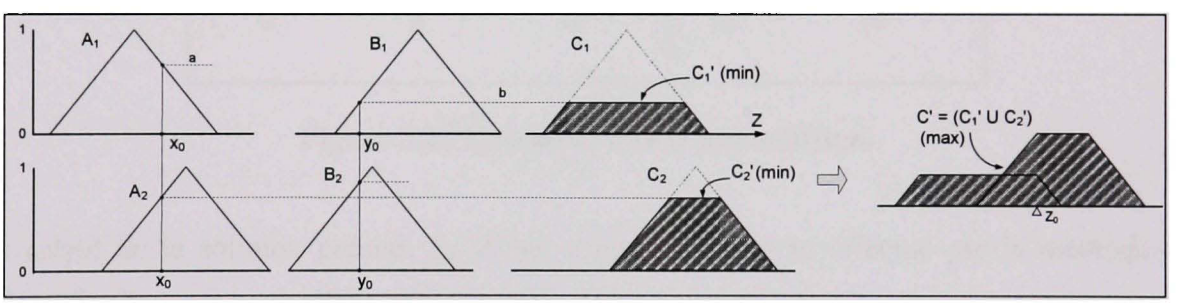


Figure 3.20 Évaluation des règles selon la méthode max-min (Mamdani).

(Tiré de Mizumoto, 1995)

Le résultat de l'implication C_i' est donné par l'opérateur *min*, il est défini par :

$$\mu_{C_i'}(z) = \min\left(\mu_{A_i}(x), \mu_{B_i}(y)\right) \tag{3.16}$$

La conséquence finale C' est donnée par l'agrégation utilisant l'opérateur *max*. Elle est définie comme suit :

$$C' = \max\left(C_1', C_2', \ldots, C_n'\right) \tag{3.17}$$

Finalement, le calcul de l'évaluation des règles utilisant le *modèle de Mamdani* est généralisé par :

$$\mu_{C'}(z) = \max\left[\min\left(\mu_{A_i}(x), \mu_{B_i}(y)\right)\right] \tag{3.18}$$

où

$\mu_{C'}(z)$ est la valeur de la variable de sortie floue.

La Figure 3.21 illustre un exemple de variable de sortie floue.

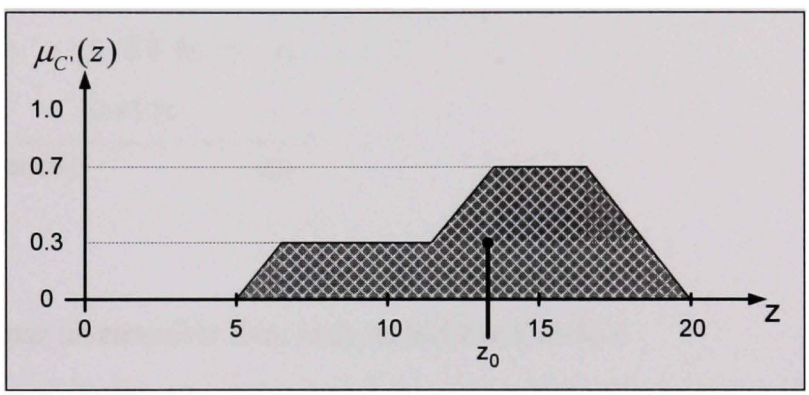


Figure 3.21 Ensemble flou d'une solution.

Le calcul de la solution précise, z_0, d'une solution floue sera effectué par la méthode de *défuzzification* que nous allons présenter dans la section suivante.

Nous allons maintenant présenter *la méthode d'inférence floue de type Sugeno* qui peut être obtenue en remplaçant les fonctions d'appartenance de la sortie soit par des fonctions linéaires ou soit par des constantes. À partir de la *méthode Mamdani*, nous pouvons définir une méthode de raisonnement flou simplifiée (Mizumoto, 1995) de *type Sugeno* dont la forme de raisonnement flou est la suivante :

Règle 1 :	A_1 **ET** $B_1 \Rightarrow$	z_1
Règle 2 :	A_2 **ET** $B_2 \Rightarrow$	z_2
	⋮	
Règle n :	A_n **ET** $B_n \Rightarrow$	z_n
Fait :	x_0 et y_0	
Conséquence :		z_{s0}

où

z_{si} n'est pas un ensemble flou, mais un nombre réel de Z.

Lorsque la *méthode de Sugeno* est d'ordre zéro, z_{si} est une constante; en conséquence, z_{si} est considéré comme une forme de *fonction d'appartenance* de *type singleton*. La Figure 3.22 illustre l'évaluation des règles selon *la méthode Sugeno*.

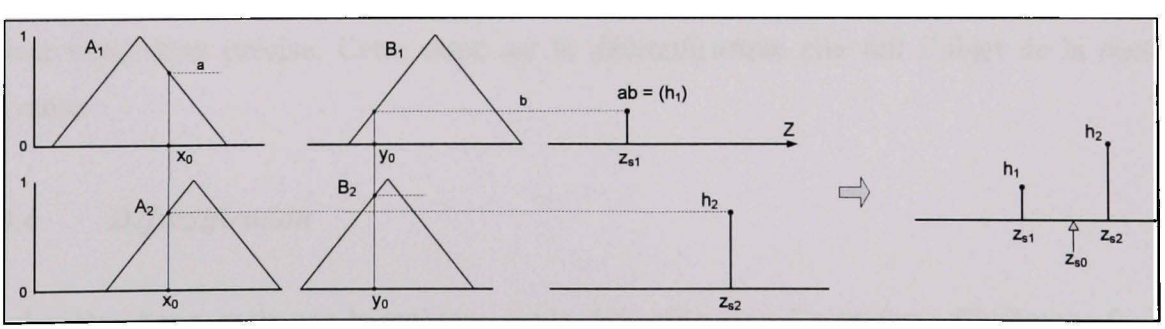


Figure 3.22 Évaluation des règles selon la méthode Sugeno.

(Tiré de Mizumoto, 1995)

Le résultat de l'implication h_i est donné par l'opérateur *min* qui est défini par :

$$h_i = \min\left(\mu_{A_i}(x), \mu_{B_i}(y)\right) \tag{3.19}$$

La conséquence finale z_{s0} est donnée par la moyenne pondérée du produit des z_{si} par le résultat de l'implication h_i. Elle est définie comme suit :

$$z_{s0} = \frac{h_1 \cdot z_1 + h_2 \cdot z_2 + ... + h_n \cdot z_n}{h_1 + h_2 + ... + h_n} \approx z_0 \tag{3.20}$$

L'implication h_i, appelé *le degré de vérité* ou *le degré d'activation*, peut être considéré comme le degré avec lequel z_{si} est obtenu.

Pratiquement, la *méthode de Sugeno* est beaucoup plus facile à réaliser vu que l'équation de la conséquence finale est donnée par une moyenne pondérée. Par contre, le *modèle de Mamdani* offre une meilleure interprétation sémantique que le modèle de Sugeno. En conclusion, la seconde étape d'un système flou, qui consiste en une évaluation des règles d'inférence entre les entrées et les sorties, nous a donc permis d'obtenir le sous-ensemble flou de la solution. Par contre, pour utiliser pratiquement ce résultat, il faut lui donner une valeur numérique précise. Cette étape est la *défuzzification*, elle fait l'objet de la section suivante.

3.3.4 *Défuzzification*

La dernière étape, mais non la moindre, est la *défuzzification*. Le système d'inférence fournit une sortie floue par l'évaluation des règles à la suite d'une ou plusieurs entrées réelles. Se pose alors le problème de lui donner une représentation floue approximative ou une valeur précise, c'est la *défuzzification*. La *défuzzification* est généralement l'une des tâches les plus gourmandes en temps d'opération du traitement flou (Uncu, 2003). Il existe de nombreuses

méthodes de *défuzzification*, mais seulement quatre sont pratiquées. Les méthodes les plus couramment utilisées sont :

- la méthode du centre de gravité;
- la méthode des maximums;
- la méthode des surfaces;
- la méthode des hauteurs.

Bien que le choix soit d'une façon ou d'une autre subjectif, nous nous limitons ici à la présentation de la méthode du centre de gravité, traditionnellement utilisée par les contrôleurs flous (Yen *et al.*, 1995).

3.3.4.1 *Défuzzification* par la méthode du centre de gravité

En général, la méthode du centre de gravité (CDG) est basée sur le calcul de l'abscisse correspondant au centre de gravité de la surface du sous-ensemble flou de la solution déterminée par l'agrégation de l'action des règles floues. La Figure 3.23 illustre une solution floue β dont la solution précise, z_0, est donnée par la méthode du centre de gravité.

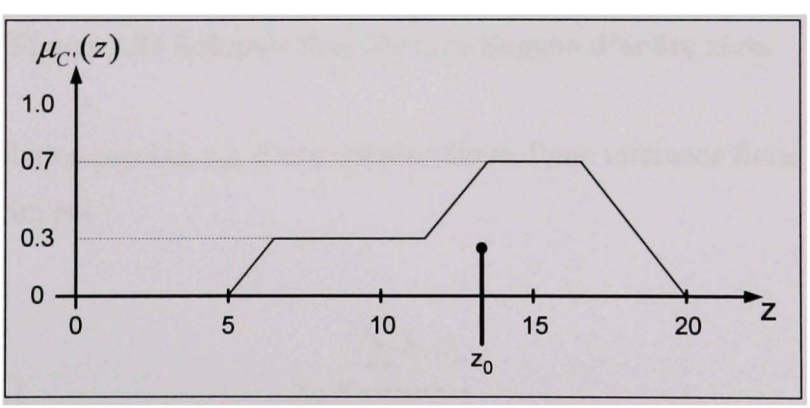


Figure 3.23 Méthode du centre de gravité.

Le calcul de la solution précise, z_0, d'une solution floue à l'aide de la méthode du centre de gravité est défini par :

$$z_0 = \frac{\int_z f_\beta(z) \cdot z \, dz}{\int_z f_\beta(z) \, dz} \tag{3.21}$$

Dans le cas où la méthode d'inférence de *type Sugeno d'ordre zéro* est utilisée, la *défuzzification* est donnée par la moyenne pondérée. On calcule individuellement les sorties, z_{si}, relatives à chaque règle selon le principe de la moyenne des maxima, puis on réalise leur moyenne pondérée. La Figure 3.24 illustre une solution floue dont la solution précise, z_{s0}, est donnée par la moyenne pondérée.

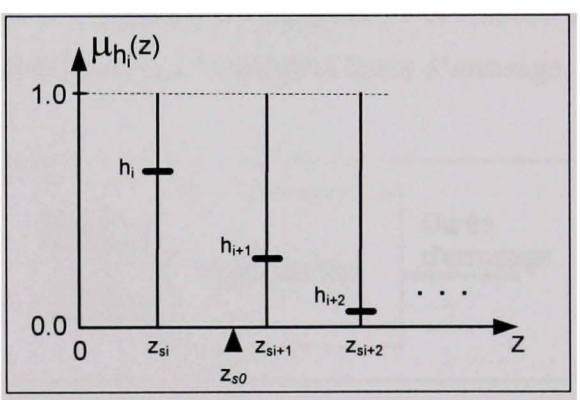


Figure 3.24 Solution floue de type Sugeno d'ordre zéro.

Le calcul de la solution précise, z_{s0}, d'une solution floue d'une inférence floue par la *méthode de Sugeno* est défini par :

$$z_{s0} = \frac{\sum_1^n h_i \cdot z_{si}}{\sum_1^n h_i} \tag{3.22}$$

En conclusion, il n'y pas vraiment de bonne ou de mauvaise méthode de *défuzzification*. Le choix de la méthode de *défuzzification* est conditionné par un compromis entre la facilité et la performance, selon la sensibilité du maître d'œuvre et sa perception du problème.

3.4 Illustration de la logique floue à l'aide d'un exemple

Dans le but de mieux comprendre la problématique envisagée, nous proposons, avant l'étude rigoureuse des divers aspects relatifs à la mise en œuvre des contrôleurs à LF, d'examiner l'exemple qui suit dans le but de donner une bonne idée de la LF en vue de permettre de mieux la comprendre.

3.4.1 *Fuzzification* du système flou

Afin de mettre en évidence le principe fondamental de la LF, nous présentons un exemple simple, celui d'un système de contrôle d'arrosage selon la température extérieure. On souhaite commander la durée d'arrosage à l'aide d'un système flou. Le système a deux entrées, soit la mesure de la température extérieure et la mesure de l'humidité du sol. Le système possède un signal de sortie qui contrôle la durée d'arrosage.

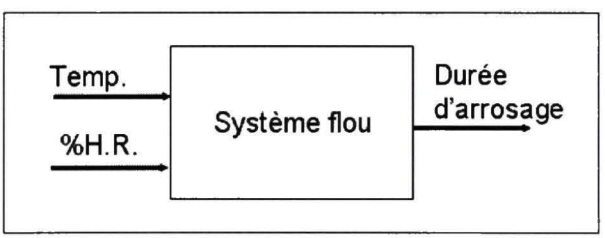


Figure 3.25 Schéma du système flou d'un contrôle d'arrosage.

Les étapes de *fuzzification* consistent à établir les fonctions d'appartenance dont voici les étapes :

1. Établir les *variables linguistiques*;
2. Établir les quantificateurs flous (nombre de *valeurs linguistiques*);
3. Attribuer une signification numérique à chaque quantificateur flou : *fonction d'appartenance*.

3.4.1.1 Établissement des variables linguistiques du système flou

Les entrées : température extérieure, degré d'humidité du sol.
La sortie : durée d'arrosage.

3.4.1.2 Établissement des quantificateurs flous et les fonctions d'appartenance

En premier lieu, nous établissons les quantificateurs flous et les fonctions d'appartenance pour les entrées du système. Pour la *fuzzification* de la température externe, nous choisissons cinq intervalles flous et des fonctions d'appartenance de types trapézoïdal et triangulaire. La gamme de l'univers du discours de la *variable linguistique « température »* est de 0 °C à 50 °C. Les prédicats *« froide, fraîche, normale, chaude* et *élevée »* sont les *valeurs linguistiques* de la *variable linguistique « température »*. Pour chacun de ces prédicats, on associe une *fonction d'appartenance* $\mu_{prédicat}(x)$. La Figure 3.26 illustre la répartition floue de la variable d'entrée température extérieure.

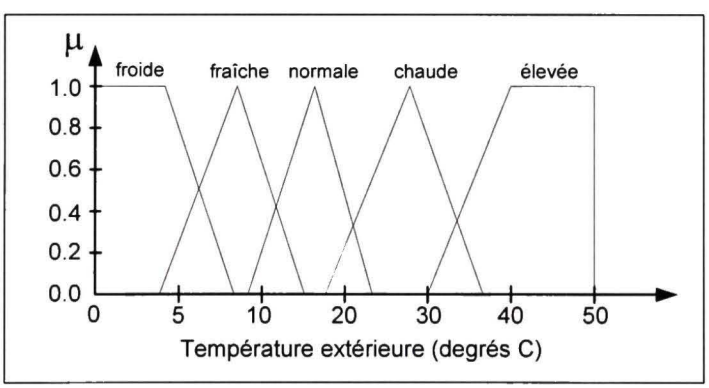


Figure 3.26 Répartition floue de la variable d'entrée température extérieure.

Nous choisissons trois intervalles flous et des fonctions d'appartenance de types trapézoïdal et triangulaire pour la *fuzzification* de l'humidité du sol en définissant le *« sec »* comme correspondant à une humidité de 30 %, le prédicat *« humide »* comme étant une humidité comprise entre 28 % et 62 % et le *« trempé »* comme étant une humidité supérieure à 65 %. La Figure 3.27 illustre la répartition floue de la variable d'entrée humidité du sol.

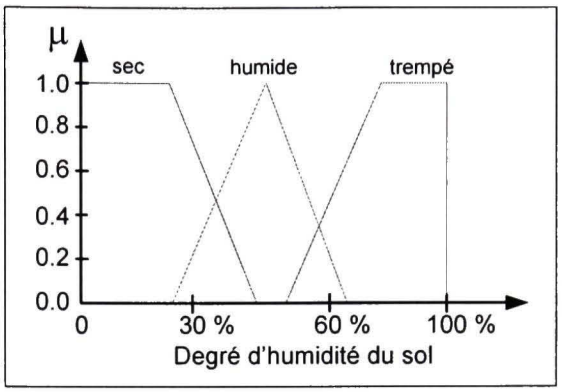


Figure 3.27 Répartition floue de la variable d'entrée humidité du sol.

Par la suite, nous établissons les quantificateurs flous et les fonctions d'appartenance pour la sortie du système. Nous choisissions trois intervalles flous et des fonctions d'appartenance de type trapézoïdale en définissant le prédicat « *courte* » comme correspondant à une durée d'arrosage inférieure à 12 minutes, le prédicat « *moyenne* » comme étant une durée d'arrosage comprise entre 25 et 35 minutes et le prédicat « *longue* » comme étant une durée d'arrosage supérieure à 42 minutes. La Figure 3.28 illustre la répartition floue de la variable de sortie durée d'arrosage.

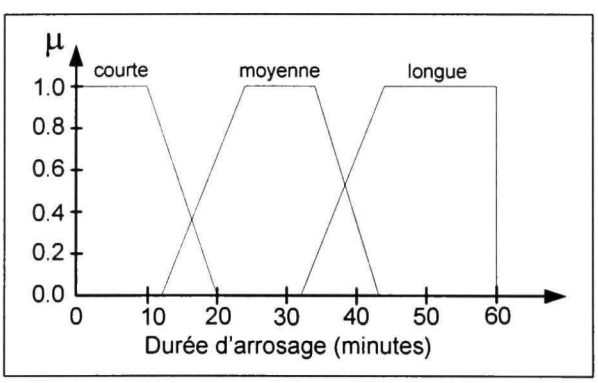


Figure 3.28 Répartition floue de la variable de sortie durée d'arrosage.

3.4.2 Établissement de la base des règles du système flou

La base des règles est définie par l'expertise d'un expert. L'expérience acquise sur la manipulation d'un système d'arrosage a permis de définir les quinze règles du système d'arrosage. Voici quatre exemples de la base des règles représentées sous une forme linguistique :

1. **Si** la température est « *normale* » **ET** le degré d'humidité du sol est « *sec* » **ALORS** la durée d'arrosage est « *longue* »;
2. **Si** la température est « *froide* » **ET** le degré d'humidité du sol est « *humide* » **ALORS** la durée d'arrosage est « *courte* »;
3. **Si** la température est « *chaude* » **ET** le degré d'humidité du sol est « *sec* » **ALORS** la durée d'arrosage est « *longue* »;
4. **Si** la température est « *chaude* » **ET** le degré d'humidité du sol est « *humide* » **ALORS** la durée d'arrosage est « *moyenne* ».

On peut représenter la base des règles sous forme de tableau ou de matrice. La base des règles complète est la suivante :

Température

Degré d'humidité du sol

	froide	fraîche	normale	chaude	élevée
trempé	courte	courte	courte	courte	courte
humide	courte	moyenne	moyenne	moyenne	moyenne
sec	longue	longue	longue	longue	longue

Durée d'arrosage

Figure 3.29 Base des règles sous forme de matrice.

3.4.3 Méthode d'inférence du système flou

La méthode d'inférence choisie est celle de *Mamdani*. Par conséquent, l'opérateur ***ET*** est réalisé par le calcul du *minimum*, tandis que l'opérateur ***OU*** est réalisé par le calcul du *maximum*.

3.4.4 *Défuzzification* du système flou

L'étape de *défuzzification* se fait à l'aide de la méthode de calcul du centre de gravité.

3.4.5 Exemple de calcul de toutes les étapes du système flou

On sait que la *fuzzification* consiste à déterminer le degré d'appartenance d'une valeur pour une variable d'entrée à un ensemble flou. Par exemple, prenons une température de 20°C et un degré d'humidité de 30 %, illustrées à la Figure 3.30.

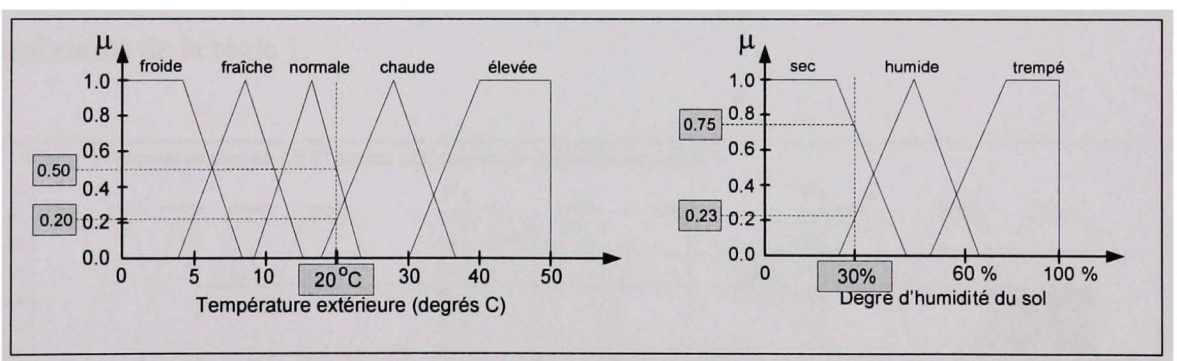


Figure 3.30 Exemple de *fuzzification*.

Dans ce cas, la température est « *normale* » à 0.5 et est « *chaude* » à 0.20. Tandis que le degré d'humidité est « *sec* » à 0.75 et est « *humide* » à 0.23. Le degré d'appartenance de la température mesurée (20°C) est donc $\mu_{froide}(x) = 0$, $\mu_{fraîche}(x) = 0$, $\mu_{normale}(x) = 0.50$, $\mu_{chaude}(x) = 0.20$ et $\mu_{élevée}(x) = 0$. De manière succincte, le degré d'appartenance de l'humidité mesurée (30 %) est $\mu_{sec}(y) = 0.75$ et $\mu_{humide}(y) = 0.23$.

L'implication consiste à déterminer le degré de vérité pour chaque conséquence. Il faut s'assurer de la pertinence de la fonction d'appartenance des conséquences. Les quatre règles d'inférence sollicitées sont les suivantes :

1. **Si** la température est « *normale* » **ET** le degré d'humidité du sol est « *sec* » **ALORS** la durée d'arrosage est « *longue* »;
2. **Si** la température est « *froide* » **ET** le degré d'humidité du sol est « *humide* » **ALORS** la durée d'arrosage est « *moyenne* »;
3. **Si** la température est « *chaude* » **ET** le degré d'humidité du sol est « *sec* » **ALORS** la durée d'arrosage est « *longue* »;
4. **Si** la température est « *chaude* » **ET** le degré d'humidité du sol est « *humide* » **ALORS** la durée d'arrosage est « *moyenne* ».

L'ensemble flou de conclusion de la règle 1 est construit en réalisant le « *minimum* » entre les deux degrés d'appartenance des antécédents. La Figure 3.31 illustre la conclusion de l'implication de la règle 1.

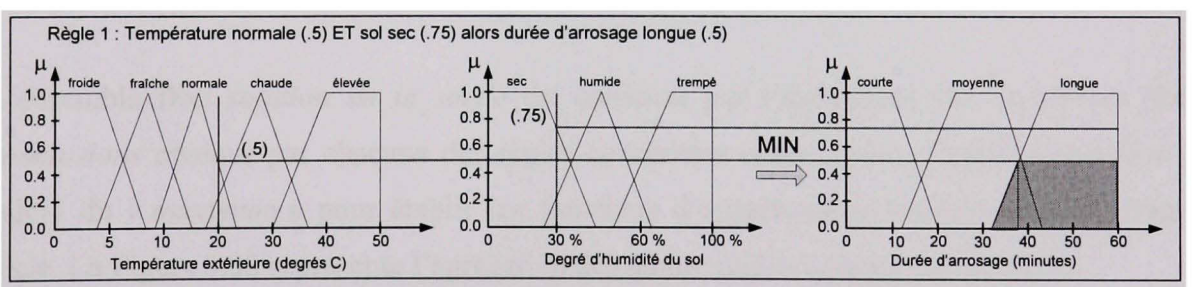


Figure 3.31 Règle 1 sollicitée par le système flou.

La Figure 3.32 illustre les conclusions des implications des règles 2, 3 et 4 sollicitées par le système flou pour les deux variables d'entrées.

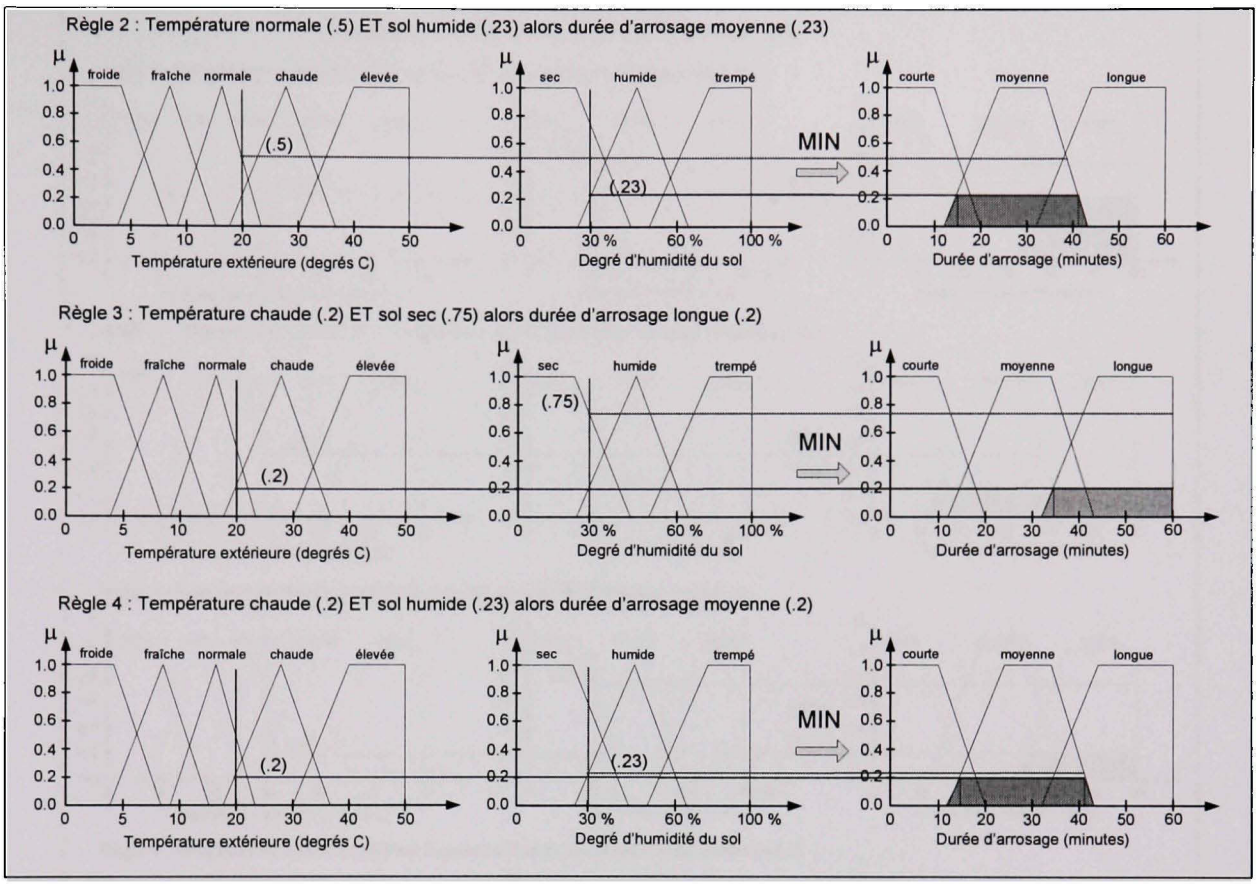


Figure 3.32 Règles 2, 3 et 4 sollicitées par le système flou.

L'ensemble flou *solution de la sortie* est construit par l'agrégation des *ensembles flous conclusions* obtenus par chacune des règles concernant cette sortie. L'agrégation utilise le calcul du « *maximum* » pour établir les fonctions d'appartenance résultantes pour chaque règle. La Figure 3.33 représente l'agrégation des quatre règles agissant sur la sortie.

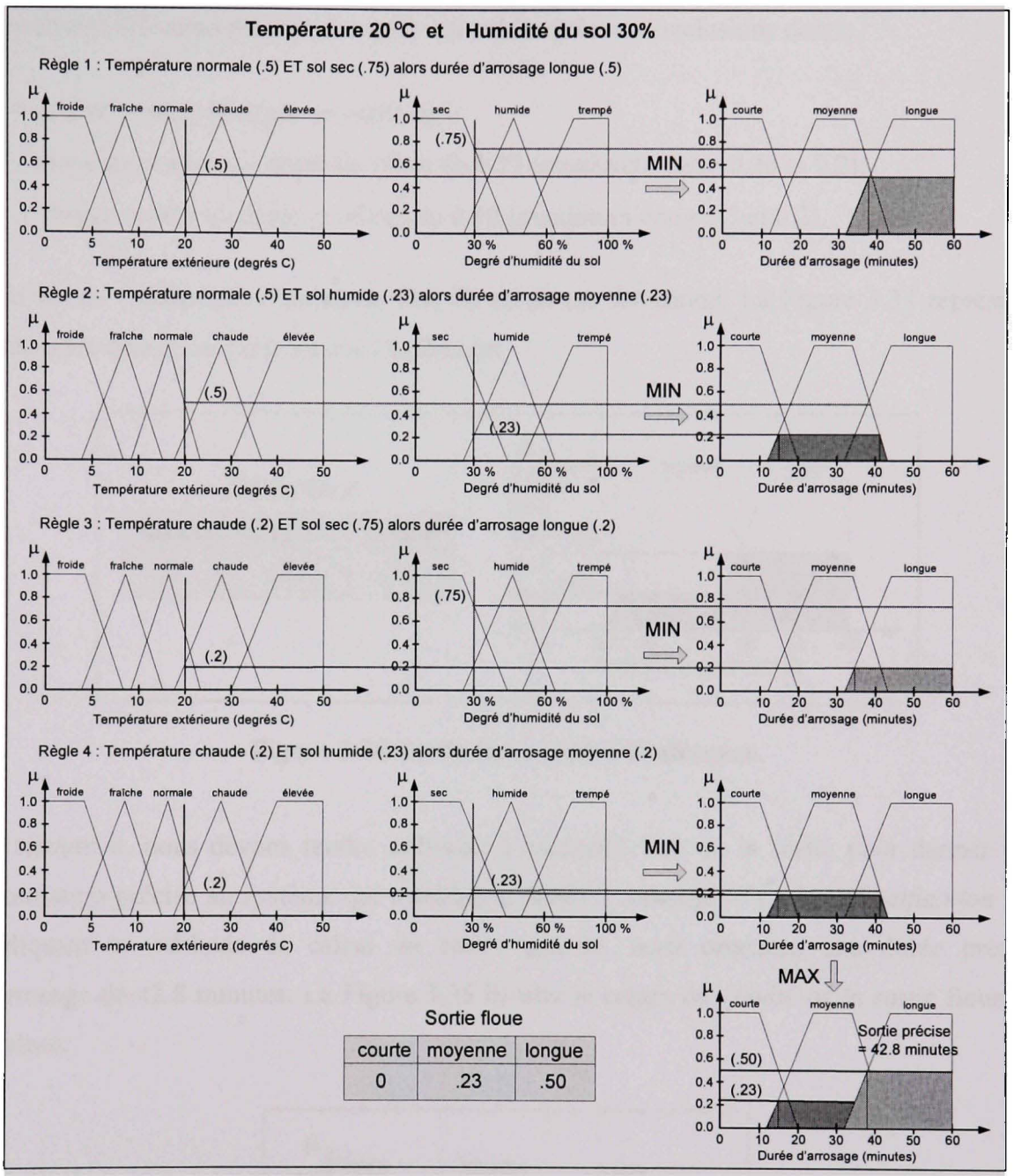


Figure 3.33 Agrégation des règles sur la sortie.

L'opérateur ***OU*** appliqué sur les règles qui ont les mêmes conclusions donne :

- « *courte* » avec un degré de vérité de 0;
- « *moyenne* » avec un degré de vérité de 0.23 (maximum entre 0.23 et 0.2);
- « *longue* » avec un degré de vérité de 0.50 (maximum entre 0.5 et 0.2).

À la fin de l'inférence, l'ensemble flou de sortie est déterminé. La Figure 3.34 représente l'ensemble flou de la sortie suite à l'inférence.

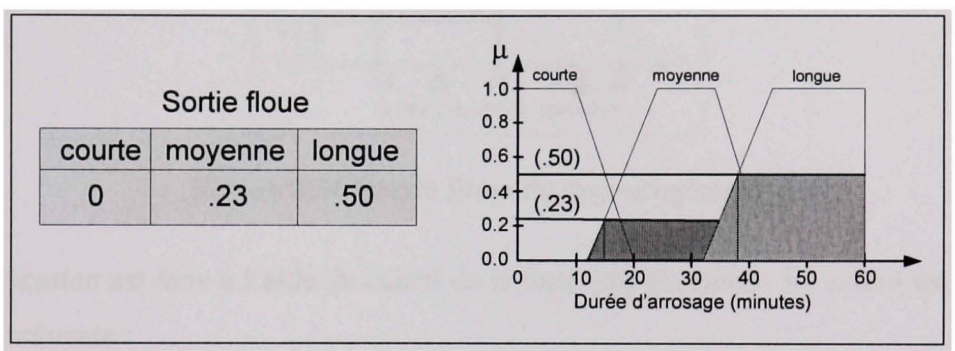


Figure 3.34 Sortie floue suite à l'inférence.

En terminant, nous devons rendre utilisable l'ensemble flou de la sortie pour donner une information précise au système qui contrôle la durée d'arrosage, c'est la *défuzzification*. En appliquant la méthode de calcul du centre gravité, nous obtenons une durée précise d'arrosage de 42.8 minutes. La Figure 3.35 illustre le centre de gravité de la sortie floue du système.

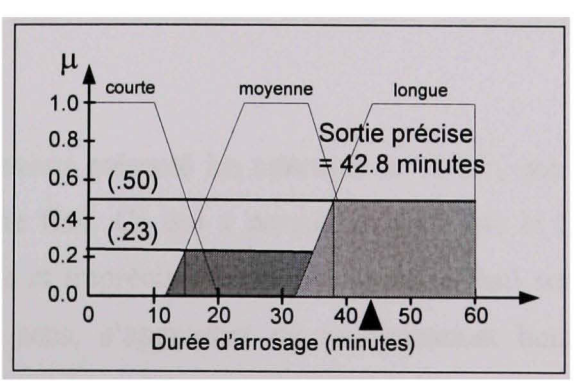


Figure 3.35 Centre de gravité de la sortie floue.

Selon la précision requise, le centre de gravité peut être calculé approximativement en utilisant la *méthode d'inférence de type Sugeno* d'ordre zéro. À cette fin, les formes des fonctions d'appartenance de la sortie sont de type *singleton*. La Figure 3.36 représente le résultat d'une *inférence floue de type Sugeno* d'ordre zéro.

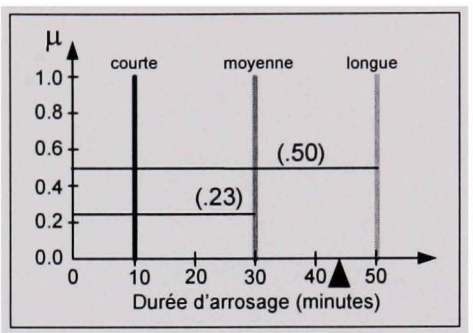


Figure 3.36 Sortie floue de type singleton.

La *défuzzification* est faite à l'aide du calcul de la moyenne pondérée. Le calcul est donné par l'équation suivante :

$$\begin{aligned} \textit{sortie précise } (z_{s0}) &= \frac{\sum_i (\textit{sortiefloue})_i \times (\textit{valeur singulière})_i}{\sum_i (\textit{sortiefloue})_i} \\ &= \frac{(0.23\times 30)+(0.50\times 50)}{(0.23+0.50)} = 43.69 \text{ minutes} \end{aligned} \tag{3.23}$$

Le système flou impose donc une durée d'arrosage de 43.69 minutes sur le système de contrôle d'arrosage.

3.5 Conclusion

Dans ce chapitre, nous avons présenté les concepts de la LF, une description générale des constituants d'un système flou. Ce qui a permis de voir que la LF sert à représenter des connaissances incertaines et imprécises. Ainsi, le système flou sert à prendre une décision pouvant, en un certain sens, s'approcher du raisonnement humain. D'autre part, cette décision peut être prise sans toutefois avoir à estimer les entrées et les sorties de manière précise, mais plutôt qu'à partir de prédicats vagues.

Enfin, nous avons présenté un exemple détaillé d'un système de contrôle utilisant un système flou pour son contrôle. Ce qui nous a permis de voir que la LF nécessite l'aide d'un expert qui devient très précieuse lors de la conception d'un système flou. La LF est souvent utilisée en combinaison avec d'autres techniques. Dans le prochain chapitre, il sera question du CF proposé dans le cadre de ce projet.

CHAPITRE 4

MODÈLE DU CONTRÔLEUR FLOU PROPOSÉ

Dans ce chapitre, nous exposons les principales approches de la conception d'un CF. Ainsi, le chapitre est réparti en trois sections principales. Dans la première, nous allons présenter les principales approches d'un contrôleur PID classique et flou de base élaborées dans le cadre d'un contrôle de procédé. Ensuite, dans la seconde section, nous considérons le cas le plus simple, celui d'un contrôleur PID flou par interconnexion. Finalement, la troisième section traite de la mise en œuvre de ce dernier, qui sera calqué sur les performances d'un contrôleur PID classique.

4.1 Contrôleurs classique et flou

Dans cette section, les deux types de contrôleurs (classique et flou) sont expliqués pour démontrer la technique et les différences entre eux. Dans le cadre de notre projet, nous nous attachons à la mise en œuvre des contrôleurs flous dans le cadre de la régulation de procédé. Cette section permettra de mieux cerner les concepts utilisés et aussi de mieux comprendre les choix effectués dans de précédentes recherches tels que décrits dans la revue de littérature.

4.1.1 Contrôleur PID classique

Dans le but d'explorer le comportement du CF en relation avec le contrôleur PID classique, nous présentons ici un bref aperçu des propriétés de ce dernier. Il est à priori intéressant d'observer la qualité de performance d'un contrôleur utilisant une loi de commande classique (PID), car la structure du CF est basée sur celle d'un contrôleur PID classique.

Les contrôleurs PID classiques ont été largement utilisés dans l'industrie en raison de leur simplicité, de leur faible coût et de l'efficacité pour les systèmes linéaires. Jusqu'à présent, quatre configurations sont utilisées pour le contrôle PID, comme illustré dans la Figure 4.1.

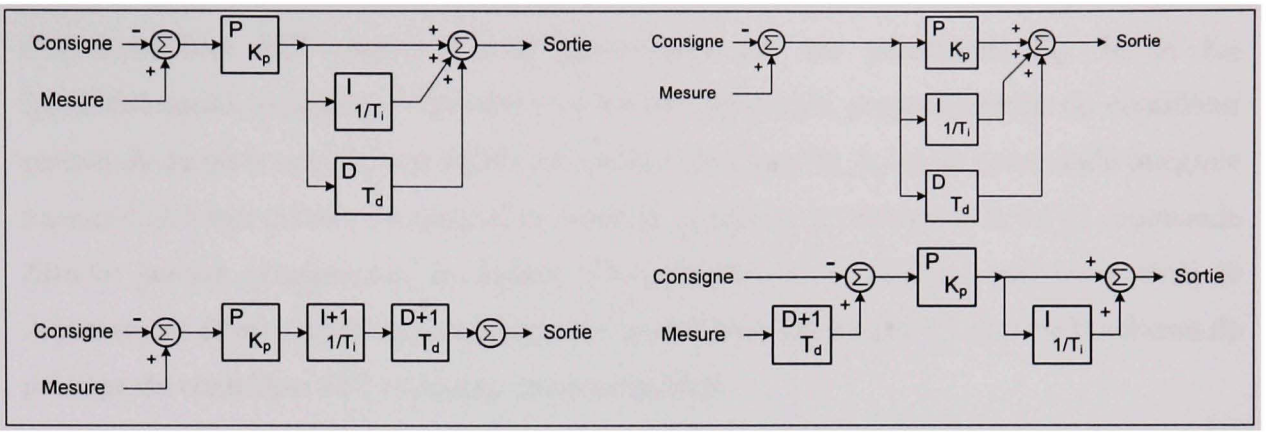


Figure 4.1 Exemples de configurations de contrôleurs PID classiques.

Le contrôleur PID classique de type standard est décrit par l'équation[6] suivante :

$$u(t) = k_p \cdot e(t) + \frac{k_p}{T_i}\int_o^t e(t)dt + k_p \cdot T_D \cdot \frac{de(t)}{dt} \tag{4.1}$$

où

k_p, T_i et T_D sont les paramètres à régler du contrôleur;

$e(t)=r(t)-y(t)$ l'erreur du système;

$u(t)$ la sortie du contrôleur.

Posons $k_D = k_p \cdot T_D$ et $k_i = k_p/T_i$, l'équation devient :

$$u(t) = k_p \cdot e(t) + k_i \int_o^t e(t)dt + K_D \cdot \frac{de(t)}{dt} \tag{4.2}$$

---

[6] L'équation présentée est celle d'un contrôleur de type standard ou modèle ISA. Plusieurs implantations des calculs sont possibles : les structures, par exemple, série, parallèle, mixte, etc. Pour une présentation plus complète le lecteur peut consulter l'annexe 1.

Selon la théorie des contrôles automatiques conventionnels (Kuo, 1991), les performances d'un contrôleur PID classique sont déterminées par ses paramètres K_p, K_I et K_D (proportionnelle, intégrale et dérivée). La loi de commande proportionnelle du contrôleur permet de garantir une réponse rapide du système de contrôle, la loi de commande intégrale permet d'éliminer l'erreur du système de contrôle en régime permanent et la loi de commande dérivée permet d'augmenter le facteur d'amortissement du système, réduisant ainsi le dépassement et les oscillations de la réponse du système. La Figure 4.2 montre le schéma de principe du contrôleur PID classique de type standard.

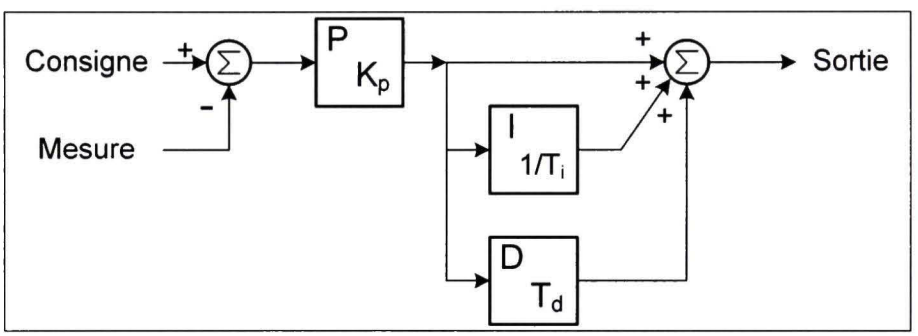


Figure 4.2 Schéma de principe du contrôleur PID classique.

4.1.2 Structure de base d'un contrôleur PID flou

La LF est utilisée dans une foule de domaines, principalement pour prendre des décisions ou pour réguler des procédés. Plusieurs applications dans le domaine de la commande de procédé utilisent un CF dont le fonctionnement est calqué sur celui d'un contrôleur PID classique. En effet, l'élaboration du modèle du contrôleur PID flou est faite en utilisant l'équation de base d'un contrôleur PID classique. La structure du CF sera choisie d'abord pour reproduire la structure générique d'un contrôleur PID classique. Le contrôleur PID classique de type standard est décrit par l'équation suivante :

$$u(t) = k_p \cdot e(t) + \frac{k_p}{T_i} \int_o^t e(t)dt + k_p \cdot T_D \cdot \frac{de(t)}{dt} \tag{4.3}$$

Par conséquent, la sortie à action directe d'un tel contrôleur est fonction de l'erreur, de l'intégrale de l'erreur ainsi que de sa dérivée.

$$u(t) = f\left[e(t), \int e(t), \frac{de(t)}{dt} \right] \tag{4.4}$$

Le CF doit donc traiter l'intégrale et la dérivée de l'erreur. En analyse numérique, il existe toute une famille d'algorithmes permettant d'approcher la valeur numérique d'une intégrale et de la dérivée. Toutes consistent à approcher l'intégrale ou la dérivée par une formule dite de quadrature, du type approximation rectangulaire ou approximation trapézoïdale. La seconde méthode d'Euler est mise à profit pour numériser les termes intégral et dérivé, conduisant à l'équation suivante :

$$U(z) = k_p \cdot E(z) + \frac{k_p}{T_i} \cdot \frac{T(z+1)}{2(z-1)} \cdot E(z) + k_p \cdot T_D \cdot \frac{(z-1)}{Tz} \tag{4.5}$$

Cependant, l'intégration numérique peut conduire à une approximation erronée lorsque la période d'échantillonnage est beaucoup plus petite que la constante d'intégration T_i. Il existe bien sûr une solution pour contrer ce fâcheux phénomène; par contre, elle nécessite une grande capacité de mémoire. Plutôt que d'implanter l'algorithme directement à partir de l'équation 4.5, il est souhaitable de modifier l'équation en supprimant l'intégrale de celle-ci. Nous utilisons une *forme incrémentale* du contrôleur numérique, dans laquelle ce n'est pas la grandeur de commande *U(z)* qui est évaluée, mais plutôt la variation (*incrément*) de *U(z)*. Le signal de contrôle progressif généré par un contrôleur PID discret est donné par :

$$\Delta u(kT) = u(kT) - u\left[(k-1)T\right] = K_p \cdot e(kT) + K_i \cdot \dot{e}(kT) + K_D \cdot \ddot{e}(kT) \tag{4.6}$$

$$\Delta u(kT) = u(kT) - u\left[(k-1)T\right] = K_p \cdot d(kT) + K_i \cdot v(kT) + K_D \cdot a(kT) \tag{4.7}$$

où K_p, K_I et K_D sont respectivement le gain proportionnel, le gain de l'intégrale et le gain de la dérivée du contrôleur PID classique. La variation de l'erreur, l'erreur et l'accélération de l'erreur sont données par les équations respectives suivantes :

$$v(kT) = \left\{ d(kT) - d\left[(k-1)T \right] \right\} / T \tag{4.8}$$

$$d(kT) = e(kT) \tag{4.9}$$

$$a(kT) = \left\{ v(kT) - v\left[(k-1)T \right] \right\} / T \tag{4.10}$$

où

e(kT) est le signal de l'erreur;

T est la période d'échantillonnage.

La Figure 4.3 illustre le schéma de principe d'un contrôleur PID flou standard.

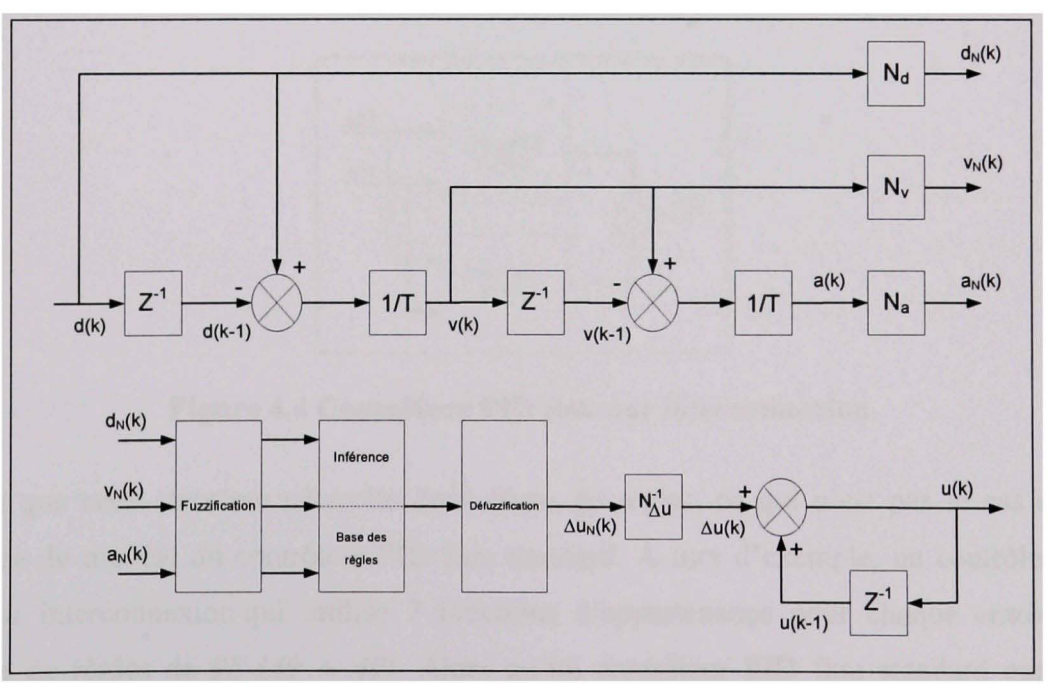


Figure 4.3 Contrôleur PID flou standard.

(Tiré d'Abdelnour, 1991)

Cependant, si théoriquement il est possible d'élaborer la loi de commande d'un contrôleur PID dans un CF, en pratique cette élaboration peut aboutir à l'obtention d'un très grand nombre de règles floues (Mansouri, 2005). En effet, l'élaboration d'un tel type de contrôleur implique une base des règles à trois dimensions. En outre, cela nécessitera un travail plus ardu de la part de l'expert lors de la mise en œuvre de ce type de contrôleur.

4.2 Contrôleur PID flou simplifié

Dans le but de réduire le nombre de règles et d'avoir les mêmes performances, Jihong (1993) et Skoczowski *et al.* (2005) ont présenté une structure plus simple d'un contrôleur PID flou. Cette structure plus simple d'un contrôleur PID flou est obtenue par interconnexion d'un contrôleur PI flou et un contrôleur PD flou. Le système de contrôle réalisé se divise en deux parties. D'une part, il y a un système de contrôle PI flou et, d'autre part, il y a un système de contrôle PD flou (Figure 4.4). Ce dernier a pour avantage d'avoir les mêmes performances que le contrôleur PID flou standard sans avoir les inconvénients d'un nombre élevé de règles d'inférence.

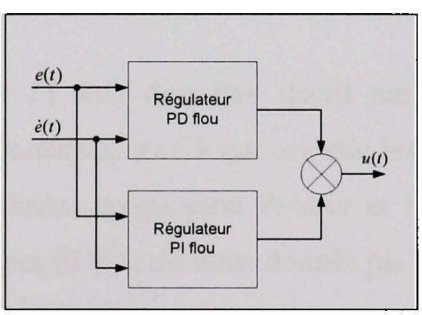


Figure 4.4 Contrôleur PID flou par interconnexion.

Notons que cette structure nécessite deux bases de règles, ce qui n'est pas le cas dans la structure du modèle du contrôleur PID flou standard. À titre d'exemple, un contrôleur PID flou par interconnexion qui utilise 7 fonctions d'appartenance pour chaque entrée a un nombre de règles de *98 (49 + 49)*. Alors qu'un contrôleur PID flou standard exploitant directement les trois entrées $e(t), \dot{e}(t)$ *et* $\ddot{e}(t)$ a un nombre de règles de *343 (7 x 7 x 7)*. Le contrôleur PID flou par interconnexion utilise les deux types de contrôleurs, celui à action

directe est utilisé pour le contrôleur de type PD tandis que celui à action incrémentale est pour le contrôleur de type PI. Dans la suite, nous abordons la conception de chacun de ces contrôleurs séparément.

4.2.1 Élaboration du contrôleur flou de type PI

Dans le cadre de notre projet, nous avons choisi d'élaborer notre CF en utilisant l'approche simplifiée, c'est-à-dire le contrôleur par interconnexion, en raison de sa simplicité de sa base des règles. L a première partie du CF par interconnexion est le CF de type PI. Si l'on considère l'équation qui décrit le contrôleur PI classique :

$$u(t) = k_p \cdot e(t) + \frac{k_p}{T_i} \int_o^t e(t)dt \tag{4.11}$$

La variable de commande *u(t)* d'un contrôleur PI classique est donc fonction de l'erreur et de son intégrale :

$$u(t) = f\left[e(t), \int e(t)\right] \tag{4.12}$$

Par conséquent, le contrôleur PI flou doit être décrit par la même équation. Selon la littérature, il est préférable de concevoir un CF qui possède les excellentes caractéristiques du contrôleur PI classique en utilisant uniquement l'erreur et la variation de l'erreur comme entrées. La fonction du contrôleur PI flou est donc donnée par l'équation suivante :

$$u(t)_{flou} = f\left[e(t), \int e(t)\right] \tag{4.13}$$

où

e(t) et $\int e(t)$ sont, respectivement, l'erreur et l'intégrale de l'erreur;

$u(t)_{flou}$ la sortie du contrôleur flou.

Si on varie la vitesse de sortie plutôt que la sortie, on obtient :

$$\frac{du(t)_{flou}}{dt} = d\left\{f\left[e(t), \int e(t)\right]\right\} / dt \tag{4.14}$$

Ainsi, un CF de type PI peut donc être vu comme une fonction qui associe une variation de la variable commandée à une erreur et sa variation.

$$\Delta u(t)_{flou} = f(e(t), \Delta e(t)) \tag{4.15}$$

où

$e(t)$ et $\Delta e(t)$ sont, respectivement, l'erreur et la variation de l'erreur;

$u(t)_{flou}$ la sortie du contrôleur flou.

L'objectif du CF de type PI est de remplacer les lois de commande proportionnelle et intégrale par un système flou (*fuzzification*, inférence et *défuzzification*). Par conséquent, le CF de type PI est caractérisé par l'équation suivante :

$$u(k)_{flou} = D\left\{F\left(\Delta e(k) + e(k)\right)\right\} + u(k-1)_{flou} \tag{4.16}$$

où

F et D sont, respectivement, les opérations de *fuzzification* et de *défuzzification.*

Selon Dobritoiu (2003), la normalisation est essentielle lors de la mise en œuvre d'un système flou. Un système flou qui a été normalisé fournit une sortie stable, en associant un poids équitable à chacune des règles, qui est basée sur les signaux d'entrées. Par contre, si la normalisation des signaux d'entrées est omise, la sortie du contrôleur sera établie par la prédominance d'une entrée par rapport à l'autre. Dans ces conditions, les résultats pourraient être biaisés si le poids des règles associées à la variable qui prédomine est plus important.

Afin de normaliser les signaux d'entrées et de sortie, le système est muni de gain servant de facteur d'échelle. La normalisation est le processus de mise à échelle des valeurs réelles entrées et sortie du contrôleur dans un domaine normalisé. Les gains G_{in1}, G_{in2} et G_{out1} sont les facteurs de normalisation de $e(k)$, $\Delta e(k)$ et $\Delta u(k)$ respectivement. Ces facteurs d'échelle

jouent un rôle semblable à celui des gains K_P et K_I dans un contrôleur PI classique. L'équation 4.17 montre la loi de commande normalisée du CF de type PI.

$$u(k)_{flou} = G_{out} \cdot D\left\{F\left(G_{in2} \cdot \Delta e(k) + G_{in1} \cdot e(k)\right)\right\} + u(k-1)_{flou} \tag{4.17}$$

L'illustration de cette équation est donnée à la Figure 4.5.

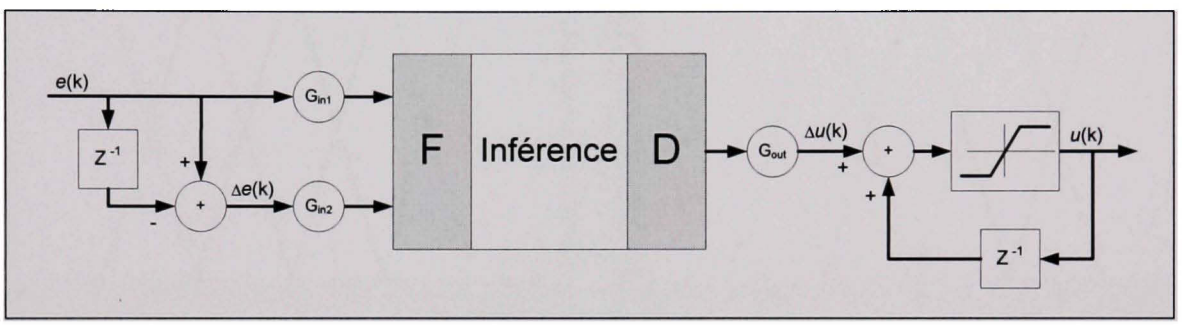


Figure 4.5 Schéma de principe d'un CF de type PI normalisé.

Comme, il y a un risque de dépassement des valeurs maximales, une saturation a été placée à la sortie du CF de type PI pour assurer en tout temps une valeur de sortie *u(k)* comprise dans l'intervalle de l'univers du discours de la sortie.

4.2.2 Choix des fonctions d'appartenance

Le CF est réalisé avec des sous-ensembles flous pour *e(k)* et *Δe(k)* de forme triangulaire ou trapézoïdale, des sous-ensembles flous en singleton pour *Δu(k)* et une *défuzzification* de type Sugeno. Dans le cadre d'une commande de processus, trois à cinq intervalles s'avèrent suffisants (Harinath, 2007). Les sous-ensembles flous sont régulièrement répartis et forment une partition stricte.

L'erreur et la variation de l'erreur sont divisées en cinq fonctions d'appartenance. Le domaine de l'erreur est généralisé de -100 % à +100 %, alors que celui de la variation de l'erreur est de -100 %/s à +100 %/s. La Figure 4.6 montre les fonctions d'appartenance de l'erreur et la variation de l'erreur.

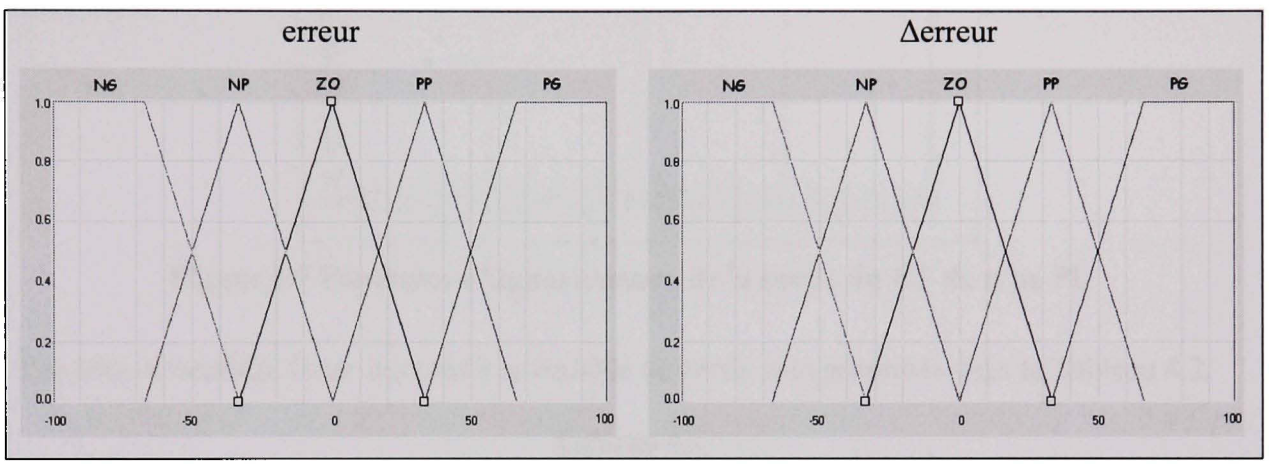


Figure 4.6 Fonctions d'appartenance des entrées du CF de type PI.

Les sous-ensembles flous associés aux deux variables d'entrées sont présentés dans le Tableau 4.1.

Tableau 4.1
Termes linguistiques des entrées du CF de type PI

Variable linguistique	Termes linguistiques
erreur :	$\{NG, NP, ZO, PP, PG\}$
Δerreur :	$\{NG, NP, ZO, PP, PG\}$

Quant à la sortie, elle est divisée en cinq fonctions d'appartenance sous forme de singletons. Son domaine est normalisé de -100 %/s à +100 %/s. La Figure 4.7 montre les fonctions d'appartenance de la sortie système flou.

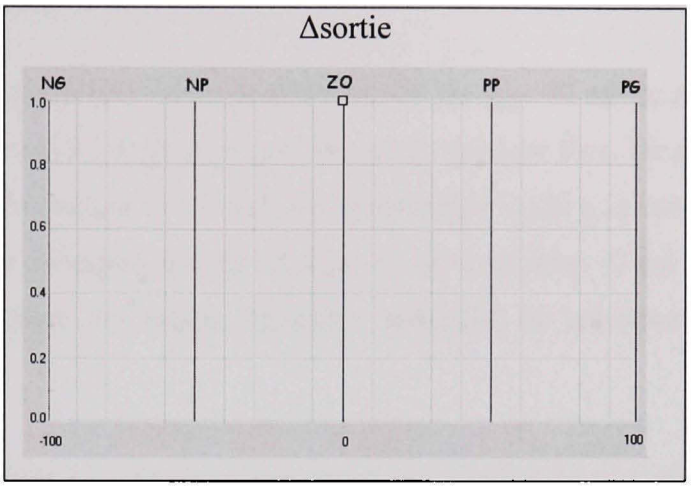


Figure 4.7 Fonctions d'appartenance de la sortie du CF de type PI.

Les sous-ensembles flous associés à la variable de sortie sont présentés dans le Tableau 4.2.

Tableau 4.2
Termes linguistiques de la sortie du CF de type PI

Variable linguistique	Termes linguistiques
Δsortie :	$\{NG, NP, ZO, PP, PG\}$

Les noms des sous-ensembles flous sont les suivants :

Tableau 4.3
Abréviations utilisées pour les termes linguistiques

NG = Négatif Grand	NP = Négatif Petit
ZO = Zéro	
PP = Positif Petit	PG = Positif Grand

4.2.3 Évaluation des règles

Comme mentionné précédemment, le principe du CF de type PI est de remplacer les lois de commande proportionnelle, intégrale et dérivée par un système flou. De même, un contrôleur se doit de maintenir la mesure de la variable commandée égale à la consigne, et ce, malgré les perturbations, les changements de charge ou de consigne. C'est le rôle du module d'inférence avec sa base des règles. Plusieurs méthodes de sélection des règles ont été étudiées.

Une méthode, proposée par King et Mamdani, pour élaborer les règles d'inférence du contrôleur PI flou, consiste à se servir des connaissances pragmatiques du réglage d'un contrôleur PI classique (Lee, 1990). Pour choisir les règles, il suffit d'observer la réponse d'un système en boucle fermée à la suite d'un changement de charge, de consigne ou sa trajectoire dans le plan de phase. La Figure 4.8 illustre un plan de phase d'une réponse indicielle d'un système en boucle fermée.

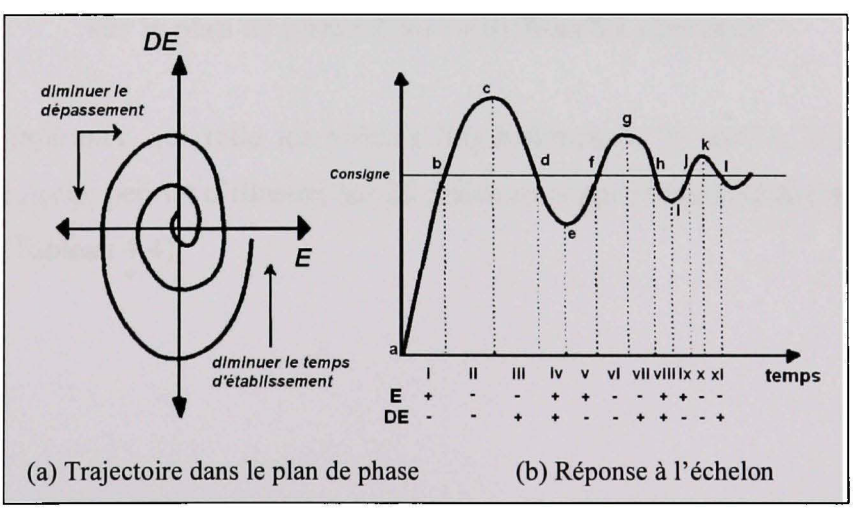


Figure 4.8 Plan de phase justifiant les règles d'inférence.
(Tiré de Lee, 1990)

On sait que, par exemple, si l'erreur est grande, il faut commander une forte correction à la sortie, peu importe la variation de l'erreur (*région i*). Si l'erreur est négative et la variation de l'erreur est positive (*région iii*), la commande floue consistera donc à imposer une correction

faible puisque la mesure se rapproche de la consigne. De même, si l'erreur est nulle et la vitesse est nulle, la sortie doit être maintenue à sa valeur actuelle. La base des règles sera donc constituée de l'analyse fonctionnelle d'un contrôleur PI classique par une personne expérimentée. La prise en compte de toutes les situations possibles du plan de phase permet en définitive d'obtenir la table des règles appelée *matrice d'inférence*. La Figure 4.9 montre la *matrice d'inférence* pour le CF de type PI calqué sur le plan de phase d'un contrôleur PID classique pour une partition de 5 sous-ensembles flous pour chaque variable d'entrée *e(k)* et *Δe(k)*.

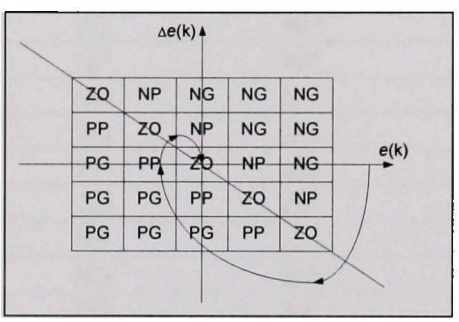


Figure 4.9 Base des règles pour le CF de type PI calqué sur le plan de phase d'un contrôleur PI classique.

La *matrice d'inférence* qui relie les valeurs linguistiques de la sortie, *Δu(t)*, à celle des entrées, *e(k)* et *Δe(k)*, permet d'illustrer les 25 règles sous une forme linguistique de type « SI … ALORS » (Tableau 4.4).

Tableau 4.4
Base des règles pour un CF de type PI sous forme linguistique

Règle no	**SI**	Δerreur	**ET**	erreur	**ALORS**	Δsortie
1		PG		NG		ZO
2		PP		NG		PP
3		ZO		NG		PG
4		NP		NG		PG
5		NG		NG		PG
6		PG		NP		NP
7		PP		NP		ZO
8		ZO		NP		PP
9		NP		NP		PG
10		NG		NP		PG
11		PG		ZO		NG
12		PP		ZO		NP
13		ZO		ZO		ZO
14		NP		ZO		PP
15		NG		ZO		PG
16		PG		PP		NG
17		PP		PP		NG
18		ZO		PP		NP
19		NP		PP		ZO
20		NG		PP		PP
21		PG		PG		NG
22		PP		PG		NG
23		ZO		PG		NG
24		NP		PG		NP
25		NG		PG		ZO

Le Tableau 4.5 représente la base des règles pour un CF de type PI sous forme de matrice.

Tableau 4.5
Base des règles pour un CF de type PI sous forme de matrice

		e(k)				
		NG	NP	ZO	PP	PG
Δe(k)	PG	ZO	NP	NG	NG	NG
	PP	PP	ZO	NP	NG	NG
	ZO	PG	PP	ZO	NP	NG
	NP	PG	PG	PP	ZO	NP
	NG	PG	PG	PG	PP	ZO

Finalement, le calcul de l'évaluation des règles utilisant le *modèle de Mamdani* est généralisé par :

$$\mu_{C'}(z) = \max\left[\min\left(\mu_{A_i}(x), \mu_{B_i}(y)\right)\right] \tag{4.18}$$

où

$\mu_{C'}(z)$ est la valeur de la variable de sortie floue.

4.2.4 Conversion du résultat flou en une valeur numérique

La conséquence finale z_{s0} est donnée par la moyenne pondérée du produit des z_{si} par le résultat de l'implication h_i. Elle est définie comme suit :

$$z_{s0} = \frac{h_1 \cdot z_1 + h_2 \cdot z_2 + \ldots + h_n \cdot z_n}{h_1 + h_2 + \ldots + h_n} \approx z_0 \tag{4.19}$$

4.2.5 Élaboration du contrôleur flou de type PD

La deuxième partie du CF par interconnexion est le CF de type PD. Tout ce qui a été dit pour le CF de type PI reste valable pour le CF de type PD. Si l'on considère l'équation qui décrit le contrôleur PD classique :

$$u(t) = k_p \cdot e(t) + k_p \cdot T_D \cdot \frac{d}{dt} e(t) \tag{4.20}$$

La variable de commande *u(t)* d'un contrôleur PD classique est donc fonction de l'erreur et de sa variation de l'erreur :

$$u(t) = f\left[e(t), \frac{d}{dt} e(t) \right] \tag{4.21}$$

Par conséquent, le contrôleur PD flou doit être décrit par la même équation. La fonction du contrôleur PD flou est donc donnée par l'équation suivante :

$$u(t)_{flou} = f\left[e(t), \frac{d}{dt} e(t) \right] \tag{4.22}$$

où

$e(t)$ et $\frac{d}{dt} e(t)$ sont, respectivement, l'erreur et la variation de l'erreur;

$u(t)_{flou}$ la sortie du contrôleur flou.

L'objectif du CF de type PD est de remplacer les lois de commande proportionnelle et dérivée par un système flou (*fuzzification*, Inférence et *défuzzification*). Par conséquent, le CF de type PI est caractérisé par l'équation suivante :

$$u(k)_{flou} = D\left\{ F\left(e(k) + \Delta e(k) \right) \right\} \tag{4.23}$$

où

F et D sont, respectivement, les opérations de *fuzzification* et de *défuzzification.*

Conformément à ce qui a été vu au paragraphe 4.2.1, le système est muni de gains servant de facteur d'échelle pour la normalisation des signaux d'entrées et de sortie. Les gains G_{in1}, G_{in2} et G_{out1} sont les facteurs de normalisation de *e(k)*, *Δe(k)* et *u(k)* respectivement. Ces facteurs d'échelle jouent un rôle semblable à celui des gains K_p et K_D dans un contrôleur PD classique. L'équation 4-24 montre la loi de commande normalisée du CF de type PD.

$$u(k)_{flou} = G_{out} \cdot D\left\{F\left(G_{in1} \cdot e(k) + G_{in2} \cdot \Delta e(k)\right)\right\} \tag{4.24}$$

L'illustration de cette équation est donnée à la Figure 4.10.

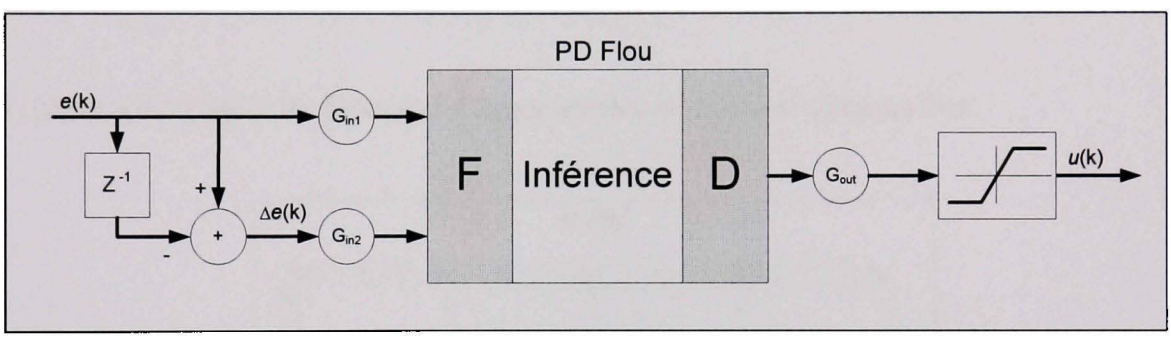


Figure 4.10 Schéma de principe d'un CF de type PD.

Comme, il y a un risque de dépassement des valeurs maximales, une saturation a été placée à la sortie du CF de type PD pour assurer en tout temps une valeur de sortie *u(k)* comprise dans l'intervalle de l'univers du discours de la sortie.

4.2.6 Choix des fonctions d'appartenance

L'élaboration des sous-ensembles flous pour *e(k)*, *Δe(k)* et *u(k)* s'effectue de la même façon que celle du CF de type PI. Les sous-ensembles flous sont régulièrement répartis et forment une partition stricte. La Figure 4.11 montre les fonctions d'appartenance de l'erreur et la variation de l'erreur.

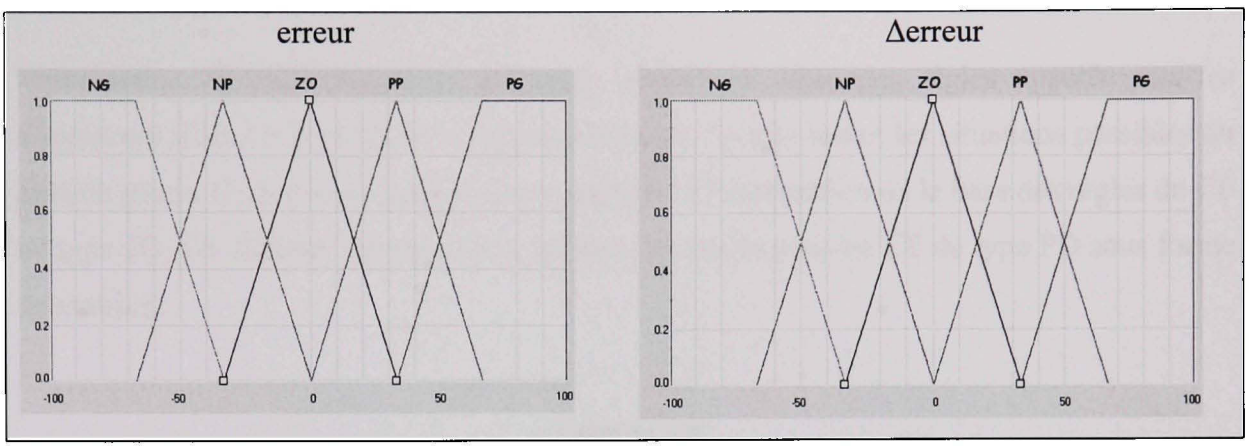


Figure 4.11 Fonctions d'appartenance des entrées du CF de type PD.

La Figure 4.12 montre les fonctions d'appartenance de la sortie système flou.

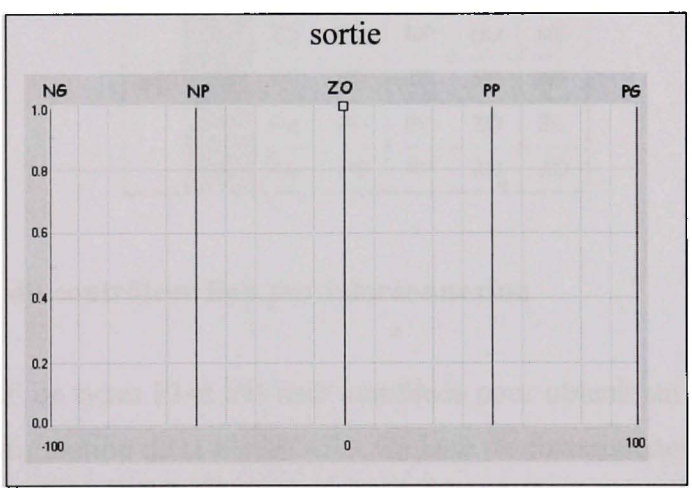


Figure 4.12 Fonctions d'appartenance de la sortie du CF de type PD.

4.2.7 Évaluation des règles

Nous avons dit précédemment, que pour obtenir la table des règles appelée *matrice d'inférence* d'un CF de type PI, il fallait prendre en compte toutes les situations possibles du plan de phase. Cette même constatation s'applique à l'élaboration de la base des règles du CF de type PD. Le Tableau 4.6 représente la base des règles pour un CF de type PD sous forme de matrice.

Tableau 4.6
Base des règles pour un CF de type PD sous forme de matrice

		e(k)				
		NG	NP	ZO	PP	PG
Δ*e*(k)	PG	ZO	ZO	NP	NG	NG
	PP	ZO	ZO	NP	NG	NG
	ZO	PP	PP	ZO	NP	NP
	NP	PG	PG	PP	ZO	ZO
	NG	PG	PG	PP	ZO	ZO

4.2.8 Structure du contrôleur flou par interconnexion

En conclusion, les CF de types PI et PD sont combinés pour obtenir un contrôleur PID flou en fonction de la configuration de la Figure 4.13. Sa base de connaissances se compose d'une base des règles à deux dimensions pour chaque contrôleur de types PI et PD.

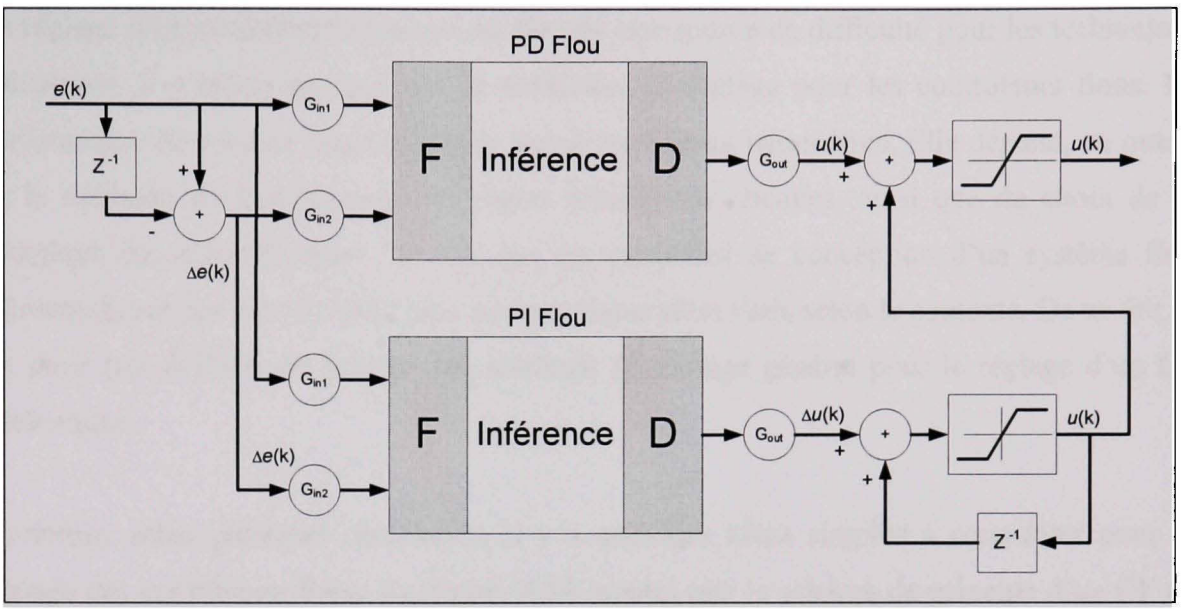


Figure 4.13 Structure du contrôleur PID flou par interconnexion.

La combinaison parallèle des CF de types PI et PD peut être utilisée pour l'optimisation des paramètres de réglage, mais il convient de noter que, en raison de la présence de partie proportionnelle double dans ce contrôleur les paramètres ajustés différeraient de ceux des contrôleurs PID classiques.

4.3 Réglage et mise au point d'un contrôleur flou

La mise au point d'un CF consiste à faire des compromis afin d'obtenir la réponse désirée selon un critère de qualité choisi. Donc, il s'agit de trouver le point d'équilibre entre les caractéristiques qui influencent le régime transitoire du procédé :

- la stabilité du procédé;
- l'erreur maximale;
- le temps de rétablissement désiré.

Le réglage d'un contrôleur classique ou flou est une source de difficulté pour les techniciens industriels. Il n'existe presque pas de méthodes de réglage pour les contrôleurs flous. La performance de celui-ci sera influencée par de nombreux paramètres. Elle dépend, en outre, de la méthode de *fuzzification*, des règles d'inférence retenues, ainsi que du choix de la technique de *défuzzification*. On sait que les méthodes de conception d'un système flou relèvent du cas par cas, d'autant plus que son élaboration varie selon le contexte. De ce fait, il est donc très difficile de trouver une méthode de réglage général pour le réglage d'un CF quelconque.

Toutefois, selon plusieurs chercheurs, il y a quelques idées simples à considérer pour le réglage des contrôleurs flous. La Figure 4.14 montre que le schéma de principe d'un CF de type PI est composé d'un système flou et de gains. Ces derniers ont été ajoutés pour normaliser les entrées et la sortie du CF. Les termes employés pour désigner ces gains sont : G_{in1}, G_{in2} et G_{out}. D'après Dobritoiu (2003), la normalisation des grandeurs d'entrées et de sortie est une étape clé du bon fonctionnement. Les gains d'entrées permettent de doser l'action des entrées sur le CF, tandis que le gain de sortie influence la commande qui est directement envoyée au procédé. Il existe un fort lien entre les gains d'entrées et le gain de sortie puisque le réglage des gains d'entrées influe sur la sortie du CF. À la limite, un mauvais choix de réglage aurait pour conséquence de biaiser les résultats.

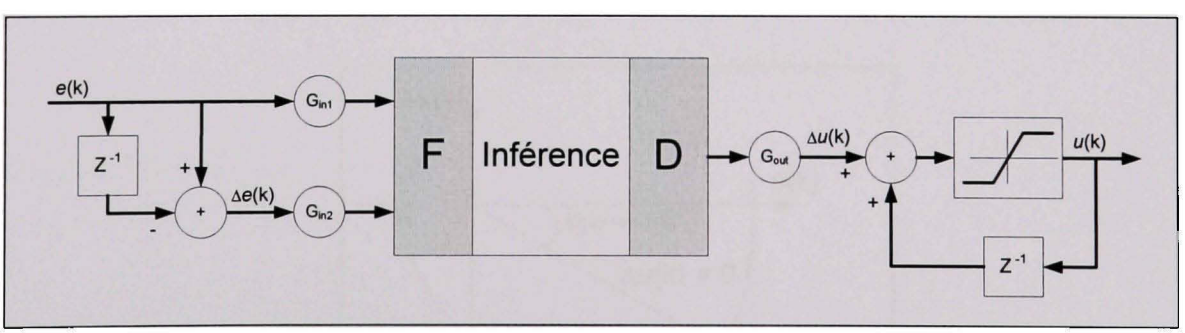


Figure 4.14 Schéma de principe d'un CF de type PI.

L'équation 4.25 montre la loi de commande du CF de type PI.

$$u(k)_{flou} = G_{out} \cdot D\left\{F\left(G_{in2} \cdot \Delta e(k) + G_{in1} \cdot e(k)\right)\right\} + u(k-1)_{flou} \tag{4.25}$$

Si l'on considère l'équation qui décrit le contrôleur PI classique :

$$u(t) = k_p \cdot e(t) + \frac{k_p}{T_i}\int_o^t e(t)dt \tag{4.26}$$

En dérivant, nous obtenons :

$$u'(t) = k_p\left(e'(t) + \frac{1}{T_i}e(t)\right) \tag{4.27}$$

À la limite, nous avons

$$u'(t) = k_p\left(e'(t) + \frac{1}{T_i}e(t)\right) = 0 \tag{4.28}$$

La solution de l'équation 4.28 est

$$e'(t) = -\frac{1}{T_i}e(t) \tag{4.29}$$

Étant donné que le k_p est toujours positif, la correction de l'erreur dépend seulement du temps d'intégration du contrôleur PI.

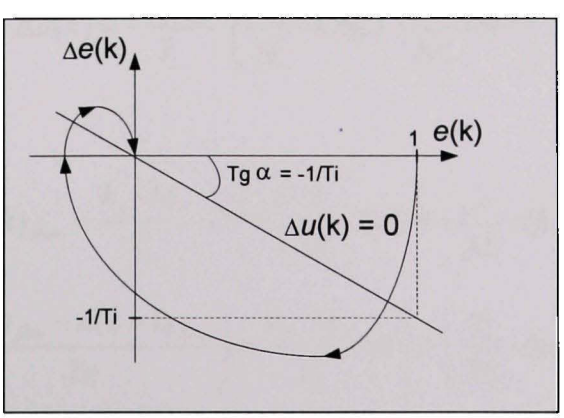


Figure 4.15 Plan de phase du contrôleur PI.

En représentant l'équation 4.27 du contrôleur PI classique sous forme discrète, nous obtenons l'équation suivante :

$$\Delta u(k) = k_p \left(\Delta e(k) + \frac{1}{T_i} e(k) \right) \tag{4.30}$$

où

$$\Delta u(k) = \left(\frac{u(k) - u(k-1)}{Te} \right) \tag{4.31}$$

$$\Delta e(k) = \left(\frac{e(k) - e(k-1)}{Te} \right) \tag{4.32}$$

et *Te* : temps d'échantillonnage

Alors,

$$\Delta u(k) = \frac{k_p}{T_i} \cdot \left(T_i \cdot \Delta e(k) + e(k) \right) \tag{4.33}$$

Donc, les entrées du système flou sont *e(k)* et *Δe(k)*. Pour normaliser les entrées, nous devons établir un facteur de mise à l'échelle pour la gamme de l'univers du discours, que l'on nommera M_i, ($M_i > 0$).

$$\Delta u(k) = \frac{k_p \cdot M_i}{T_i} \cdot \left(\frac{T_i}{M_i} \cdot \Delta e(k) + \frac{1}{M_i} e(k) \right) \tag{4.34}$$

Pour un CF, on a :

$$\Delta u(k)_{flou} = \frac{k_p \cdot M_i}{T_i} \cdot D \left\{ F \left(\frac{T_i}{M_i} \cdot \Delta e(k) + \frac{1}{M_i} e(k) \right) \right\} \tag{4.35}$$

$$\Delta u(k)_{flou} = \left(\frac{u(k)_{flou} - u(k-1)_{flou}}{Te} \right) = \frac{k_p \cdot M_i}{T_i} \cdot D \left\{ F \left(\frac{T_i}{M_i} \cdot \Delta e(k) + \frac{1}{M_i} e(k) \right) \right\} \tag{4.36}$$

La sortie du CF de type PI est donc définie par l'équation suivante :

$$u(k)_{flou} = \frac{k_p \cdot M_i \cdot Te}{T_i} \cdot D\left\{F\left(\frac{T_i}{M_i} \cdot \Delta e(k) + \frac{1}{M_i} e(k)\right)\right\} + u(k-1)_{flou} \tag{4.37}$$

Et ainsi, à l'aide de l'équation 4.25, nous pouvons associer par comparaison la valeur des gains du CF de type PI.

$$G_{in1} = \frac{1}{M_i} \tag{4.38}$$

$$G_{in2} = \frac{T_i}{M_i} \tag{4.39}$$

$$G_{out} = \frac{k_p \cdot M_i \cdot Te}{T_i} \tag{4.40}$$

Le réglage de ces paramètres est important pour obtenir la meilleure performance du CF de type PD. L'optimisation des paramètres de réglage est également disponible pour le contrôleur flou de type PD avec l'univers du discours unifié où les paramètres sont les mêmes que celles de contrôleur PD classique. La Figure 4.16 montre le schéma de principe d'un CF de type PD.

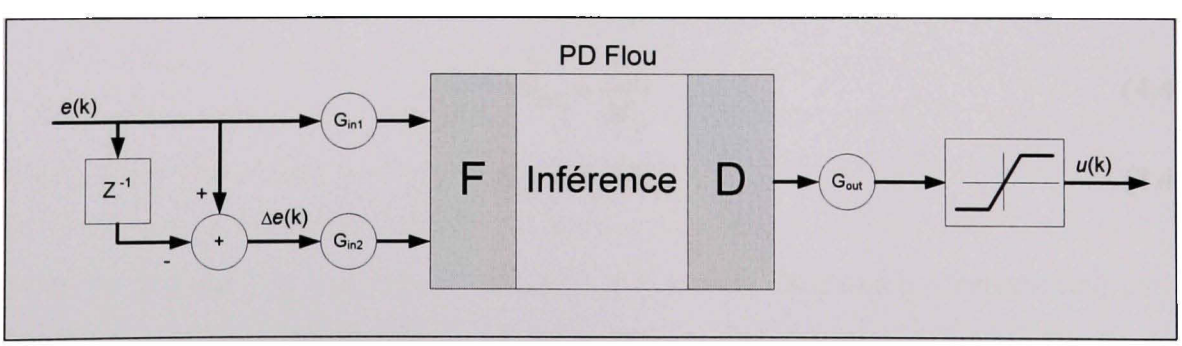


Figure 4.16 Schéma de principe d'un CF de type PD.

L'équation 4.41 montre la loi de commande du CF de type PD.

$$u(k)_{flou} = G_{out} \cdot D\left\{F\left(G_{in1} \cdot e(k) + G_{in2} \cdot \Delta e(k)\right)\right\} \tag{4.41}$$

La structure de base d'un CF de type PD est inspirée de celle du contrôleur de type PD discret. Rappelons que, sous forme discrète, ce contrôleur est décrit par l'équation :

$$u(k) = k_p\left(e(k) + T_D \cdot \Delta e(k)\right) \tag{4.42}$$

Pour normaliser les entrées, nous devons établir un facteur de mise à l'échelle pour la gamme de l'univers du discours, que l'on nommera M_D, ($M_D > 0$).

$$u(k) = k_p \cdot M_D\left(\frac{1}{M_D} \cdot e(k) + \frac{T_D}{M_D} \cdot \Delta e(k)\right) \tag{4.43}$$

Pour un CF, on a :

$$u(k)_{flou} = k_p \cdot M_D \cdot D\left\{F\left(\frac{1}{M_D} \cdot e(k) + \frac{T_D}{M_D} \cdot \Delta e(k)\right)\right\} \tag{4.44}$$

Donc, à l'aide de l'équation 4.41, nous pouvons associer par comparaison la valeur des gains du CF de type PD.

$$G_{in1} = \frac{1}{M_D} \tag{4.45}$$

$$G_{in2} = \frac{T_D}{M_D} \tag{4.46}$$

$$G_{out} = k_p \cdot M_D \tag{4.47}$$

En résumé, les paramètres de réglage du contrôleur PID flou par interconnexion sont donc au nombre de sept :

- les deux gains d'entrées G_{in1} et G_{in2} du contrôleur PI;
- le gain de sortie G_{out} du contrôleur PI;
- les deux gains d'entrées G_{in1} et G_{in2} du contrôleur PD;
- le gain de sortie G_{out} du contrôleur PD;
- sans oublier la période d'échantillonnage Te du contrôleur, puisque celui-ci est, par nature, discret.

Le Figure 4.7 montre les formules de réglage pour le contrôleur PID flou par interconnexion.

Tableau 4.7
Paramètres de réglage du CF par interconnexion

	G_{in1}	G_{in2}	G_{out}	Période d'échantillonnage
PI	$\frac{1}{M_i}$	$\frac{T_i}{M_i}$	$\frac{k_p \cdot M_i \cdot Te}{T_i}$	T_e
PD	$\frac{1}{M_D}$	$\frac{T_D}{M_D}$	$k_p \cdot M_D$	

Les gains d'entrées et de sortie du contrôleur PID flou sont fonctions des paramètres k_p, T_i et T_D du contrôleur classique, ainsi que de la période d'échantillonnage T_e. Cette méthode, quoiqu'imparfaite, permet de donner un ordre de grandeur aux valeurs des paramètres ou à tout le moins une valeur de départ à partir des réglages du contrôleur PID classique. Par contre, les réglages du contrôleur PID classique doivent absolument exister pour rendre possible l'utilisation de cette méthode. Dans ces conditions, la meilleure méthode pour améliorer le réglage du contrôleur PID flou est la méthode essais/erreurs.

4.4 Conclusion

Dans ce chapitre, nous avons présenté un bref aperçu de la structure de base du contrôleur PID flou. Nous avons mis en avant-plan les principaux résultats de la littérature concernant la mise en œuvre d'un CF. Nous avons également montré à partir des modèles d'un contrôleur PI et PD classique qu'il est possible d'élaborer un CF calqué sur la loi de commande du contrôleur classique. Cette loi de commande, très utilisée en industrie, est la base de ce projet d'application. Cette dernière approche (CF de type PI et PD), sera exploitée par la suite pour valider notre logiciel dans le cadre d'un contrôle de niveau à l'aide d'un CF de type PI et PD.

CHAPITRE 5

MISE EN ŒUVRE DU LOGICIEL

Ce chapitre traite essentiellement du logiciel d'apprentissage *Fuzzy Kernel*, de la mise en œuvre de la LF, ainsi qu'un CF de type PID sous forme de bloc de fonction intégré à un API. Le logiciel et le CF de type PID ont été développés tous les deux dans le cadre de ce projet. Dans un premier temps, une description détaillée de notre logiciel *Fuzzy Kernel* et de son fonctionnement est présentée. La description fonctionnelle de notre logiciel montre que celui-ci est simple et convivial. Notamment, elle montre que toutes les informations pertinentes au domaine de la LF ont été intégrées dans notre logiciel *Fuzzy Kernel* en faisant largement appel aux méthodes de mise en œuvre, sans pour autant délaisser le contexte pédagogique appliqué à la LF. Nous développerons ensuite le CF de type PID sous forme de bloc de fonction intégré dans un API de la famille ControlLogix® de la compagnie Allen-Bradley®. La forme du contrôleur et sa structure interne sont exposées.

5.1 Structure de développement

Le logiciel *Fuzzy Kernel* est destiné à permettre l'intégration d'un CF dans l'automate programmable ControlLogix 5000® de la compagnie Allen-Bradley®. Il propose pour cela un bloc fonction FLFB (Fuzzy Logic Function Bloc) intégré au logiciel RSLogix5000®, réalisant le calcul des inférences floues. Les règles floues et les fonctions d'appartenance sont intégrées dans l'application par saisie d'une variable de type FUZZY interne de ce bloc.

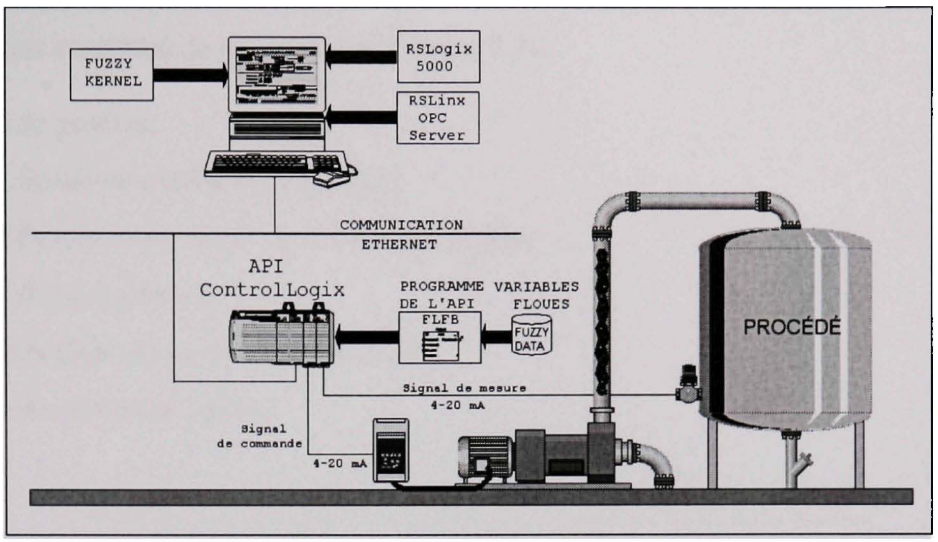


Figure 5.1 Schéma de la structure de développement.

5.2 Présentation du logiciel *Fuzzy Kernel*

Le logiciel *Fuzzy Kernel* développé met en œuvre tous les développements théoriques et pratiques décrits dans le mémoire. Le logiciel *Fuzzy Kernel* permet de créer des systèmes d'inférence floue et de les utiliser à des fins de contrôle dans les systèmes de commande à base de LF. Ses interfaces graphiques intuitives permettent de modifier la structure du système flou en quelques clics de souris. Le logiciel *Fuzzy Kernel* est un logiciel d'aide à l'apprentissage et d'application de la LF. Il permet de simuler et d'effectuer la mise en œuvre des systèmes à LF.

Fuzzy Kernel présente également la particularité de pouvoir s'interfacer et s'intégrer à l'environnement existant d'un programme dans un API. Il offre la possibilité d'exporter et de formater les données floues du système vers un API. Par conséquent, il est donc un complément au logiciel RSLogix 5000® permettant de réaliser des traitements en LF afin d'optimiser la commande des procédés à partir des automates programmables industriels ControlLogix®.

Le logiciel est constitué de six modules (Figure 5.2) :

a) module de gestion;
b) module de développement graphique;
c) module de développement de la base des règles;
d) module de compilation;
e) module de mise au point en mode local;
f) module de communication.

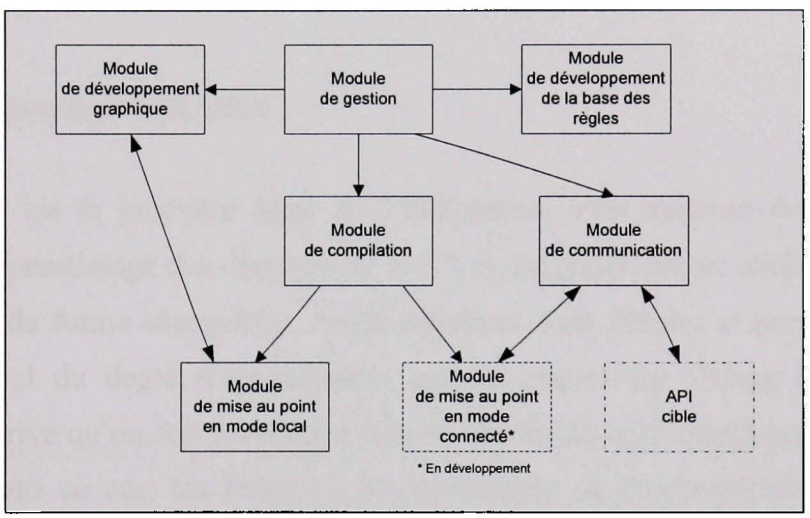


Figure 5.2 Architecture générale du logiciel *Fuzzy Kernel.*

Ce logiciel se présente sous la forme d'un simulateur autonome qui permet d'intégrer un système flou dans un bloc de fonction qui s'intègre dans tous les programmes RSLogix 5000®. Il inclut les outils de mise en œuvre et de mise au point. Les outils que nous avons utilisés pour développer ce logiciel sont le langage de programmation Microsoft Visual Basic®, dans sa version 6.0, pour le développement de l'interface graphique et un composant ActiveX®, OPC automation 2.0, pour la communication entre le logiciel et la plateforme de mise en œuvre.

Les fonctions du logiciel se caractérisent par :

- 2 grandeurs physiques utilisées en entrée;
- 2 à 5 fonctions d'appartenance graphiques qui permettent de caractériser les grandeurs physiques d'entrées sous forme de termes linguistiques associés prédéfinis;
- 25 règles linguistiques qui permettent de déterminer l'état à appliquer sur la sortie (2 antécédents et 1 conclusion par règle);
- 2 à 5 variables numériques en sortie, résultats de l'application de la fonction sur les grandeurs d'entrées.

5.2.1 Interface de *fuzzification*

La *fuzzification* est la première étape de l'élaboration d'un système flou. Dans le but d'améliorer l'apprentissage des concepts de la LF, il est préférable de choisir des fonctions d'appartenance de forme triangulaire. Leurs équations sont simples et permettent ainsi de faciliter le calcul du degré d'appartenance qui est réalisé par l'étape de *fuzzification*. Cependant, il arrive qu'on soit intéressé d'étendre une fonction d'appartenance sur l'univers du discours. Dans ce cas, les fonctions d'appartenance de forme trapézoïdale sont plus appropriées, car elles permettent de mieux couvrir l'ensemble des valeurs à *fuzzifier*, c'est-à-dire à rendre floue. Le logiciel *Fuzzy Kernel* permet la combinaison de fonction d'appartenance de formes triangulaires et trapézoïdales. Il est à noter que la forme triangulaire est réalisée à l'aide de la fonction d'appartenance trapézoïdale dont le noyau est nul.

Le logiciel utilise un assistant pour la réalisation d'un projet. Cet assistant permet de définir les noms, l'étendue de l'univers du discours, ainsi que le nombre de fonctions d'appartenance des variables d'entrées et de la variable de la sortie. La Figure 5.3 montre un exemple de données de l'assistant.

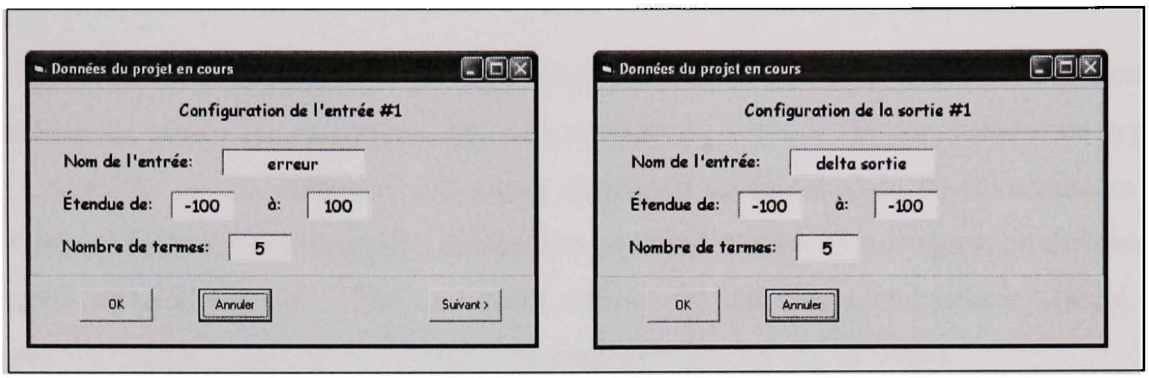


Figure 5.3 Exemple de données de l'assistant.

La description des fonctions d'appartenance s'effectue de façon intuitive grâce à un environnement graphique convivial. Les fenêtres d'ensembles flous permettent la répartition de chaque fonction d'appartenance à l'aide de quatre points d'ancrage. De plus, elles permettent le choix des termes linguistiques au choix de l'utilisateur et associés au projet en cours. La Figure 5.4 présente les fenêtres des ensembles flous d'une entrée et d'une sortie respectivement.

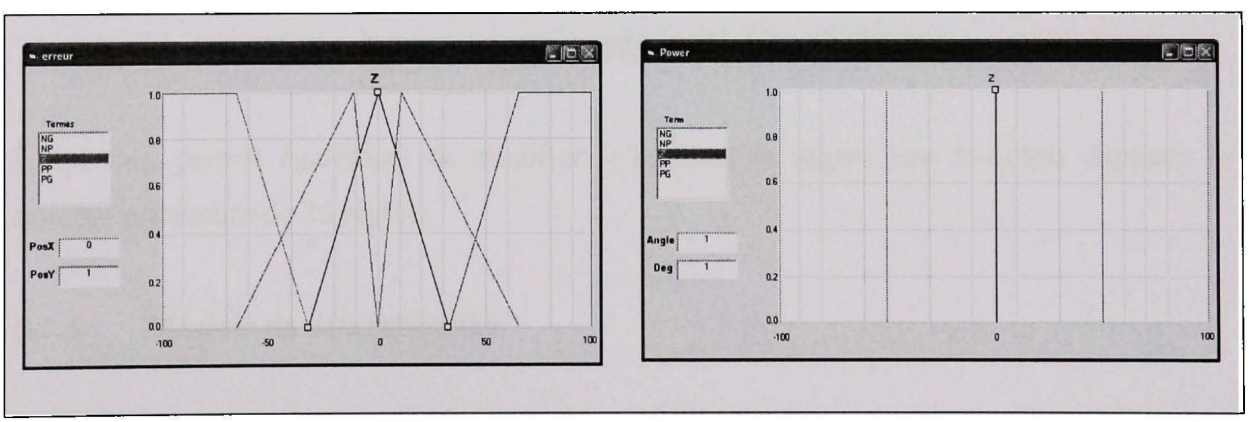


Figure 5.4 Ensembles flous d'une entrée et d'une sortie.

5.2.2 Interface des règles d'inférence

Il s'agit d'un tableau qui permet de caractériser les relations entre les classes d'événements possibles en entrée et les commandes correspondantes à l'aide de conclusions du type : «SI...ALORS...». La définition des règles s'effectue de façon aisée en sélectionnant les fonctions d'appartenance associées aux entrées par leurs termes linguistiques, en définissant la sortie affectée par cette règle également définie par son terme linguistique associé. La Figure 5.5 illustre le tableau de la fenêtre de configuration des règles floues.

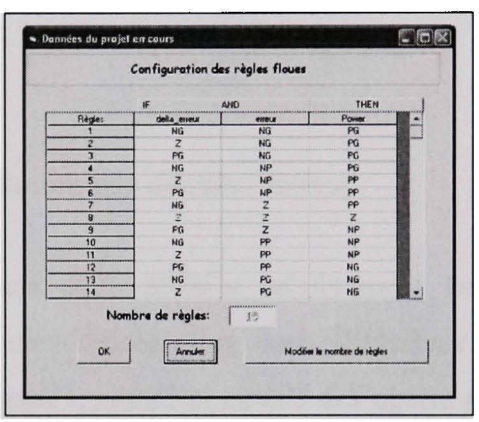


Figure 5.5 Fenêtre de configuration des règles floues

L'interface permet également de modifier le nombre de règles sans toutefois dépasser le nombre maximum de 25 règles.

5.2.3 Module de *défuzzification*

Le calcul de la valeur de sortie floue est réalisé en deux étapes principales, l'implication et l'agrégation des règles. Nous avons choisi le modèle d'inférence de *type Sugeno*, c'est-à-dire l'opérateur *min* et *max* pour l'implication et l'agrégation des règles. Pour des raisons d'interprétabilité, dans *Fuzzy Kernel*, les *fonctions d'appartenance* de la sortie sont de formes *singleton*, c'est-à-dire, une constante au lieu de la traditionnelle combinaison linéaire des entrées. La *défuzzification* est donc un modèle de *type Sugeno d'ordre zéro*, c'est-à-dire que

la sortie est égale à une constante. De plus, pratiquement, la méthode Sugeno est beaucoup plus facile à réaliser vu la simplicité de l'étape de *défuzzification.*

Dans ce cas, la *défuzzification* est donnée par la moyenne pondérée. On calcule individuellement les sorties, *Cj*, relatives à chaque règle selon le principe de la moyenne des maxima, puis on réalise leur moyenne pondérée.

$$z_{s0} = \frac{\sum_{j=1}^{m} W_j \cdot C_j}{\sum_{j=1}^{m} W_j} \tag{5.1}$$

5.2.4 Module de compilation des données floues

Les fonctions d'appartenance sont définies relativement aux entrées par des formes trapézoïdales. Elles sont toutes représentées dans l'interface graphique sous forme des abscisses et pentes *point1*, *pente1*, *point2* et *pente2* des quatre points d'inflexion successifs. La Figure 5.6 représente où les points d'ancrage consécutifs peuvent éventuellement être positionnés.

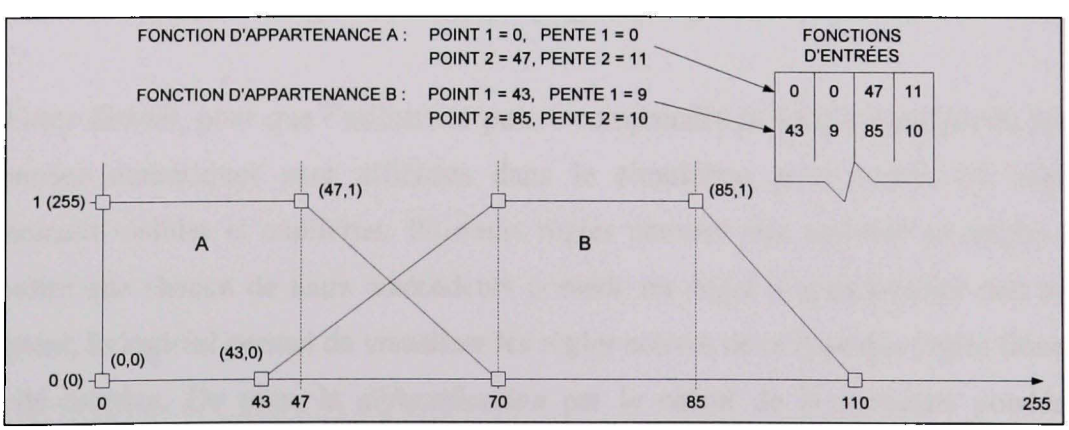


Figure 5.6 Points d'ancrage des fonctions d'appartenance.

Le module de compilation du logiciel *Fuzzy Kernel* calcule les quatre composantes, *point1*, *pente1*, *point2* et *pente2* de chaque fonction d'appartenance des entrées et de la sortie. Ce module travaille avec les fenêtres graphiques des d'entrées et de sortie du logiciel pour obtenir la valeur des ancrages de toutes les fonctions d'appartenance. Par la suite, une compilation des règles d'inférence est effectuée. En terminant, toutes les données floues sont placées dans trois variables de type tableau, *InputFonction*, *Rules* et *Singletons*.

5.2.5 Interface d'apprentissage et mise au point en mode local

De manière générale, la fenêtre de mise au point en mode local est utilisée pour étudier différents scénarios d'opération du système flou. Il s'agit principalement d'un algorithme qui effectue les trois étapes de calcul du système flou (*fuzzification*, évaluation des règles, *défuzzification*). Lors de sa mise en marche, l'algorithme de calcul est appelé périodiquement avec une fonction de temporisation (*timer)* de la librairie de l'environnement Visual Basic®. La mise au point de la fonction floue se réalise aisément à partir de l'écran de mise au point, en particulier grâce à la possibilité de simuler, dévaluer et d'optimiser le fonctionnement en mode local. Beaucoup d'implantations utilisant des méthodes d'apprentissage automatique conduisent malheureusement à des systèmes de type « boîte noire ». Dans ce cas, il est très difficile de suivre à la trace l'exécution d'une simulation en termes du degré d'appartenance des entrées et sorties et des règles activées (Belanger, 2005).

Dans *Fuzzy Kernel*, pour que l'utilisateur puisse comprendre le fonctionnement du système, les données numériques sont affichées dans le simulateur pour rendre les règles de raisonnement visibles et explicites. Plusieurs règles peuvent être activées en même temps, c'est-à-dire que chacun de leurs antécédents possède un degré d'appartenance non nul. Par conséquent, le logiciel permet de visualiser les règles actives de la base des règles floues sous forme de matrice. De plus, la *défuzzification* par le calcul de la moyenne pondérée est représentée de manière dynamique. Cette démarche novatrice constitue une des originalités du logiciel. La Figure 5.7 illustre cette fenêtre de mise au point en mode local.

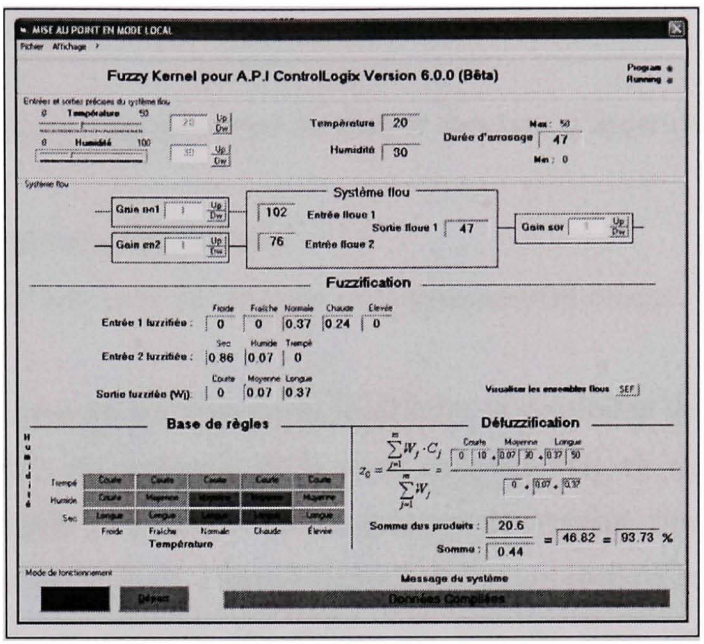


Figure 5.7 Fenêtre de mise au point en mode local.

Plusieurs articles notent l'importance des gains d'entrées et de sorties d'un système flou, car les gains servent, d'une part, à normaliser les signaux d'entrées et de sorties et, d'autre part, à déterminer l'influence de l'action des entrées sur le système flou ainsi que l'influence de l'action du signal de sortie sur la commande qui est directement envoyée au procédé. Par conséquent, le logiciel *Fuzzy Kernel* propose un système flou ayant la possibilité d'avoir des gains en entrées et en sortie. Une fois l'ensemble des écrans de paramétrage remplis et compilés, il est possible de simuler en mode local le fonctionnement du système d'inférence floue. L'écran de mise au point en mode local offre la possibilité de :

- varier des valeurs pour les différentes variables d'entrée;
- varier le gain des variables d'entrées et de sortie;
- lancer la simulation avec la touche « marche ».

Les résultats obtenus sont :

- les valeurs du degré d'appartenance de chaque fonction d'appartenance d'entrée et de sortie;
- les différentes règles qui sont actives;
- la valeur qui serait appliquée sur la sortie en fonctionnement normal.

De plus, l'écran de mise au point en mode local offre la possibilité de visualiser les sous-ensembles flous (SEF) des entrées et de la sortie en temps réel, en cliquant sur le bouton « SEF » dans la section « Entrées et sorties précises du système flou ». Ceci permet de visualiser graphiquement le degré d'appartenance à une valeur réelle d'entrée en fonction des sous-ensembles flous de chacune des entrées du système flou. De plus, cela permet également de visualiser la valeur nette inférée par l'agrégation résultant de la *défuzzification*. La Figure 5.8 et la Figure 5.9 illustrent, respectivement, les sous-ensembles flous des entrées et de la sortie en temps réel d'un système flou.

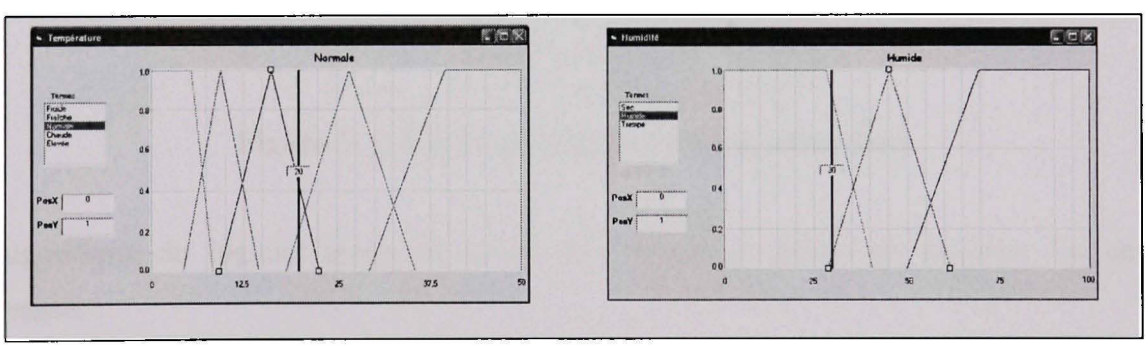


Figure 5.8 Sous-ensembles flous des entrées en temps réel.

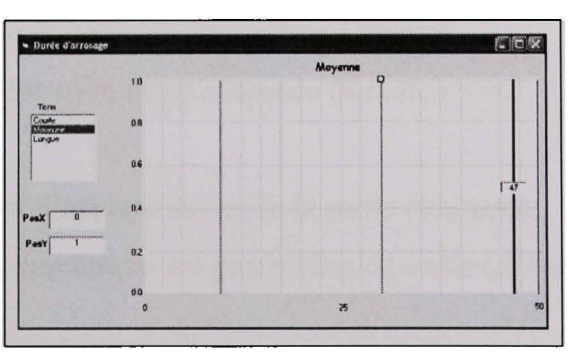


Figure 5.9 Sous-ensembles flous de la sortie en temps réel.

L'implantation de l'algorithme a été développée entièrement avec des variables entières en format 8 bits. Ce choix permet ainsi une utilisation d'un vaste choix de plateformes externes pour la mise en œuvre en temps réel du système flou. La conception d'un système flou est constituée de trois étapes principales : la conversion des entrées en valeurs floues, l'évaluation des règles et la conversion du résultat des règles en une valeur numérique de sortie. La Figure 5.10 illustre les étapes principales d'un système flou.

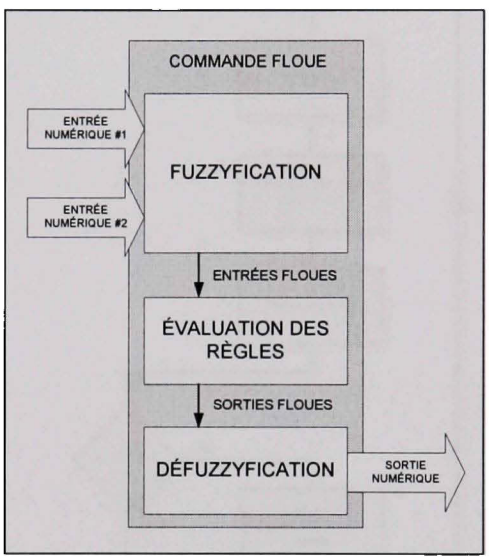


Figure 5.10 Étapes principales d'un système flou.

L'algorithme du logiciel assure le calcul des inférences floues en réalisant les étapes suivantes :

- lecture des valeurs numériques des paramètres d'entrées;
- évaluation de chaque fonction d'appartenance (*fuzzification*);
- évaluation des règles;
- obtention des fonctions d'appartenances de la sortie résultante;
- obtention de la valeur numérique des paramètres de sortie (*défuzzification*).

La Figure 5.11 montre l'algorithme de simulation du système flou.

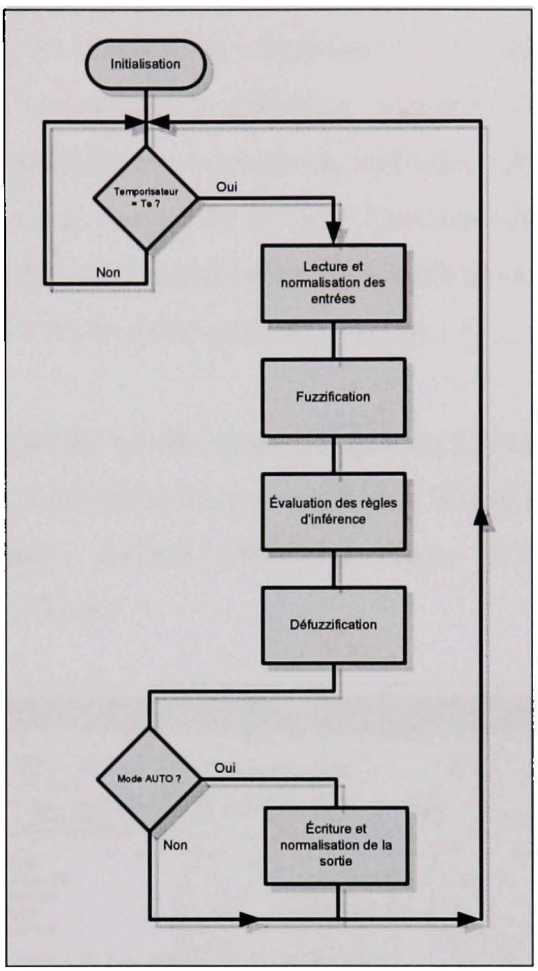


Figure 5.11 Algorithme de simulation du système flou.

5.2.6 Interface de communication

Dans notre projet de recherche, l'ordinateur hôte qui exécute le logiciel *Fuzzy Kernel* est un Pentium 4 de 1,7 GHz et de 521 Mo de mémoire RAM. L'API cible est un ControlLogix® modèle 1756-L61 avec 1024 Ko de mémoire RAM. L'interface de communication permet le transfert des données floues vers la plateforme de mise en œuvre. L'interface de communication est un composant « **O**bject Linking and Embedding for **P**rocess **C**ontrol » (OPC) qui établit le lien entre le logiciel *Fuzzy Kernel* et la plateforme externe, ici l'automate programmable industriel ControlLogix®.

Le composant de communication OPC est un mode de connexité ouverte standardisée de l'industrie créé grâce à la collaboration d'un certain nombre de sociétés d'automatisme, constructeurs de logiciel et de matériel, en collaboration avec Microsoft. Les serveurs OPC fournissent une méthode permettant à différents logiciels d'accéder aux données de dispositifs de contrôle de procédé, par exemple un automate. OPC est le protocole le plus employé dans l'industrie, car il permet de faciliter l'interopérabilité entre les applications d'automatisation et de contrôle, les systèmes et les dispositifs des entreprises. C'est donc pour toutes ces raisons que nous l'avons choisi pour notre projet.

L'interface est de type client OPC et elle nécessite donc un serveur OPC pour communiquer avec la plateforme externe. L'utilitaire de communication RSLinx® de la compagnie Allen-Bradley® est utilisé comme serveur OPC. La Figure 5.12 illustre l'interface de téléchargement des données floues.

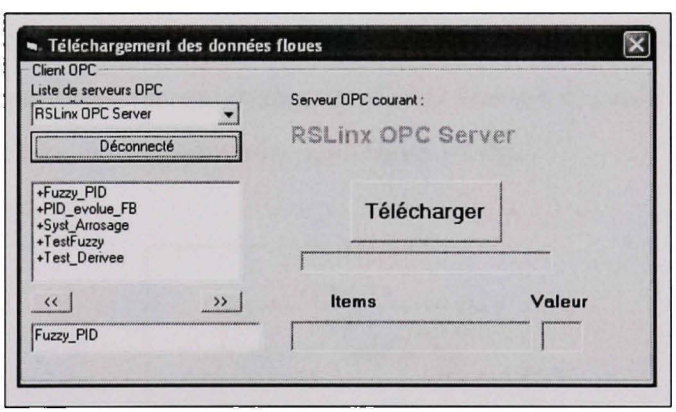


Figure 5.12 Interface de téléchargement des données floues.

La communication entre l'ordinateur hôte et l'API cible est faite à partir d'une connexion réseau Ethernet. L'ordinateur et l'API sont connectés au réseau avec des cartes Ethernet utilisant un protocole de communication de type OPC en mode client-serveur. La connexion entre l'ordinateur et l'API permet de télécharger les données de l'application floue de l'hôte vers la cible. La connexion permet aussi de télécharger les signaux de l'API cible vers l'ordinateur afin de visualiser les signaux d'entrées et de sortie.

5.3 Présentation du bloc de fonction à logique floue FLFB

Une des contributions du projet de maîtrise est de faire la réalisation et l'intégration d'un bloc de fonction à LF pouvant être intégré dans un API. Ce bloc de fonction à logique floue (FLFB) est intégré à l'API et doit traiter les signaux provenant de l'externe. L'implantation du bloc de fonction FLFB a été développée entièrement dans le logiciel RSLogix 5000®. Il s'agit d'une instruction complémentaire définie par l'utilisateur (Add-on) : l'utilisateur a la possibilité de créer des instructions personnalisées qui viennent s'ajouter au large éventail de fonctionnalités déjà intégrées dans la plateforme de commande Logix.

Le FLFB est destiné à permettre l'intégration d'un système flou dans l'automate programme ControlLogix5000®. Le FLFB a été conçu dans un souci de simplicité dans le but de permettre l'apprentissage rapide des concepts de la LF. Il intègre un mécanisme d'inférence rapide (grâce à la programmation en entiers de 8 bits) et il utilise des fonctions d'appartenance de sortie de type « Singleton ». L'algorithme de programmation du bloc FLFB est le même que celui du simulateur de *Fuzzy Kernel* illustrée à la Figure 5.11. La Figure 5.13 montre le bloc de fonction à logique floue FLFB.

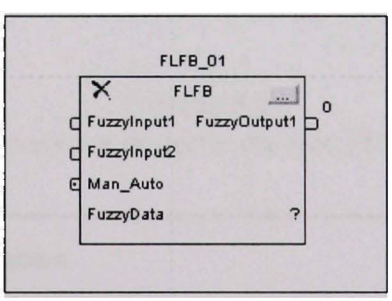


Figure 5.13 Bloc de fonction à logique floue FLFB.

5.3.1 Description des paramètres du bloc de fonction FLFB

Le bloc de fonction FLFB comporte trois signaux d'entrée, soit les deux entrées (FuzzyInput1 et FuzzyInput2) du système flou et un signal booléen (Man_Auto) pour le contrôle du bloc de fonction. Le signal Man_Auto permet de sélectionner le mode de

fonctionnement du bloc de fonction. Le Tableau 5.1 donne la description des paramètres d'entrée.

Tableau 5.1
Paramètres d'entrée du bloc FLFB

Paramètre d'entrée	**Type de données**	**Description**
FuzzyInput1	Réelle	Entrée 1 de variable procédé du système flou. Mise à l'échelle à l'aide de *LimInfInput1* et de *LimSupInput1* : *LimInfInput1* correspondant à 0 % et *LimSupInput1* correspondant à 100 %. Cette valeur provient en général d'un module d'entrée analogique.
FuzzyInput2	Réelle	Entrée 2 de variable procédé du système flou. Mise à l'échelle à l'aide de *LimInfInput2* et de *LimSupInput2* : *LimInfInput2* correspondant à 0 % et *LimSupInput2* correspondant à 100 %. Cette valeur provient en général d'un module d'entrée analogique.
Man_Auto	Booléenne	Requête programme pour le passage en mode auto. Activé (Man_Auto = 1) par le programme utilisateur pour demander le mode Auto.

De plus, le bloc de fonction FLFB est muni d'un signal de sortie (FuzzyOutput1) dont la valeur du signal provient du module d'inférence. Le Tableau 5.2 donne la description du paramètre de sortie.

Tableau 5.2
Paramètres de sortie du bloc FLFB

Paramètre de sortie	**Type de données**	**Description**
FuzzyOutput1	Réelle	Sortie variable de contrôle du système flou mise à l'échelle à l'aide de *LimInfOutput1* et de *LimSupOutput1* : *LimInfOutput1* correspondant à 0 % et *LimSupOutput1* correspondant à 100 %. Cette sortie contrôle en général un module de sortie analogique. Elle peut aussi contrôler une boucle secondaire.

Finalement, le bloc de fonction utilise une table d'information essentielle pour faire le traitement de l'inférence du système flou. Le Tableau 5.3 donne la description du paramètre interne.

Tableau 5.3
Paramètres interne du bloc PID_FLOU

Paramètre interne	Type de données	Description
FuzzyData	Fuzzy	Table d'information pour le bloc de fonction FLFB.

Pour normaliser l'utilisation du bloc de fonction FLFB, nous avons créé une table d'information prédéfinie pour le fonctionnement du bloc nommée « FUZZY ». Cette table d'information contient toutes les informations nécessaires pour le fonctionnement du système flou. De plus, cette variable sert également de cible lors du téléchargement des données de l'application floue vers l'API, à l'aide de l'interface de communication de *Fuzzy Kernel*. La table d'information FUZZY est un type de données défini par l'utilisateur, son contenu est illustré à la Figure 5.14.

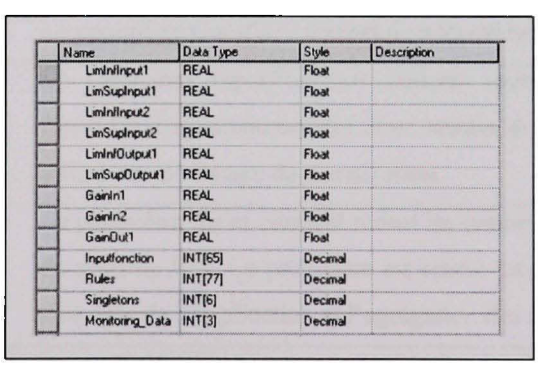


Figure 5.14 Table d'information du bloc de fonction FLFB.

Le Tableau 5.4 donne la description de la table d'information de type FUZZY.

Tableau 5.4
Paramètres de la table d'information de type FUZZY

Paramètre d'entrée	**Type de données**	**Description**
LimInfInput1	Réelle	Le paramètre *LimInfInput1* permet la mise à l'échelle de l'entrée 1 du système flou. Le paramètre *LimInfInput1* correspondant à 0 %.
LimSupInput1	Réelle	Le paramètre *LimSupInput1* permet la mise à l'échelle de l'entrée 1 du système flou. Le paramètre *LimSupInput1* correspondant à 100 %.
LimInfInput2	Réelle	Le paramètre *LimInfInput2* permet la mise à l'échelle de l'entrée 2 du système flou. Le paramètre *LimInfInput2* correspondant à 0 %.
LimSupInput2	Réelle	Le paramètre *LimSupInput2* permet la mise à l'échelle de l'entrée 2 du système flou. Le paramètre *LimSupInput2* correspondant à 100 %.
LimInfOutput1	Réelle	Le paramètre *LimInfOutput1* permet la mise à l'échelle de la sortie du système flou. Le paramètre *LimInfOutput1* correspondant à 0 %.
LimSupOutput1	Réelle	Le paramètre *LimSupOutput1* permet la mise à l'échelle de la sortie du système flou. Le paramètre *LimSupOutput1* correspondant à 100 %.
GainIn1	Réelle	Le paramètre *GainIn1* permet de normaliser l'entrée 1 du système flou.
GainIn2	Réelle	Le paramètre *GainIn2* permet de normaliser l'entrée 2 du système flou.
GainOut1	Réelle	Le paramètre *GainOut1* permet de normaliser la sortie du système flou.
InputFonction	Tableau d'entiers	Le paramètre *InputFonction* contient toutes les données associées aux fonctions d'appartenance des deux entrées du système flou. Ce paramètre est utilisé lors de l'étape de *fuzzification*.
Rules	Tableau d'entiers	Le paramètre *Rules* contient toutes les données associées à la base des règles du système flou. Ce paramètre est utilisé lors de l'étape d'inférence, c'est-à-dire, lors de l'implication et l'agrégation des règles du module d'inférence.
Singletons	Tableau d'entiers	Le paramètre *Singletons* contient toutes les données associées aux fonctions d'appartenance de la sortie du système flou. Ce paramètre est utilisé lors de l'étape de *défuzzification*.
Monitoring_Data	Tableau d'entiers	Le paramètre *Monitoring_Data* contient les valeurs numériques des signaux d'entrées et de sortie du système flou. Ce paramètre est utilisé lorsque *Fuzzy Kernel* est en mode connecté. Il permet de visualiser le fonctionnement du système flou en temps réel.

5.3.2 Principe de fonctionnement du bloc de fonction FLFB

Au début de chaque calcul, les valeurs des deux entrées (FuzzyInput1 et FuzzyInput2) du système flou sont normalisées. De cette façon, les influences des signaux sont identiques pour la première étape qui est la *fuzzification*. Les valeurs des signaux normalisés sont des valeurs binaires d'une résolution de 8 bits, ce qui permet d'avoir un mécanisme d'inférence rapide. Par la suite, le bloc de fonction assure le calcul des inférences floues en établissant les relations qui existent entre les variables d'entrées et la variable de sortie en utilisant la base des règles floues. Le résultat de cette étape permet de calculer la valeur de la variable de sortie finale floue à partir des signaux d'entrées floues issus de la *fuzzification*. Finalement, le bloc de fonction fournit une sortie réelle en appliquant une méthode de *défuzzification* de type Sugeno.

L'API cible est capable d'exécuter deux types de tâches, soit une tâche de type continue et une tâche de type périodique en temps réel à très haute vitesse. Le temps d'exécution d'une tâche continue varie selon la grosseur et la complexité des programmes. Le désavantage de ce mode est qu'il présente une latence provenant du traitement des interruptions. À l'opposé, une tâche périodique exécute une tâche à une fréquence déterminée et ne possède aucune latence. Elle peut être configurée avec une fréquence d'exécution comprise entre 0,1 ms et 2 000 s. La tâche périodique est prioritaire et synchronisée par des interruptions aux coups d'horloge. La tâche continue, quant à elle, est apériodique et asynchrone.

Selon la littérature, les applications en temps réel à commande floue doivent très souvent faire face à des calculs comportant des dérivées, par exemple, traiter la variation de l'erreur d'une position angulaire. Cependant, la dérivée numérique peut conduire à une approximation erronée si la période d'échantillonnage se modifie avec le temps. En d'autres mots, la précision du calcul d'une dérivée numérique est fortement liée à la période d'échantillonnage. Dans ces circonstances, la tâche périodique est donc privilégiée pour exécuter une application cible dont la période d'échantillonnage doit être précise. De plus, la tâche périodique est requise pour garantir le bon fonctionnement d'un système de contrôle.

Par conséquent, le bloc FLFB devra être utilisé à l'intérieur d'une tâche périodique de n'importe quelque programme RSLogix 5000®. Le choix du temps d'échantillonnage est basé sur la littérature, qui mentionne souvent un temps d'échantillonnage de 100 millisecondes.

5.4 Présentation du bloc de fonction PID_FLOU

Une deuxième contribution du projet est de faire la réalisation et l'intégration d'un CF sous forme de bloc de fonction pouvant être intégré dans un API. Malgré les nombreux avantages des différents contrôleurs flous, dans le cadre de ce projet, le CF par interconnexion PI et PD a été choisi pour l'élaboration de notre CF. Nous utiliserons ces deux types de contrôleurs : le contrôleur flou de type PI et le contrôleur flou de type PD.

La démarche proposée ici a la particularité d'utiliser deux blocs de fonction FLFB, un pour le contrôleur flou de type PI et l'autre pour le contrôleur flou de type PD. Un bloc de fonction « PID_FLOU » a donc été créé à partir de ces deux blocs de fonction FLFB. Tel qu'expliqué auparavant, l'idée de développer un système flou PID de la sorte est de minimiser le nombre de règles et ainsi diminuer la quantité d'information nécessaire a priori afin d'élaborer la base des règles.

La mise en œuvre du CF est basée sur le contrôleur PID flou par interconnexion défini dans le chapitre 3. La Figure 5.15 montre l'architecture générale du contrôleur PID flou par interconnexion.

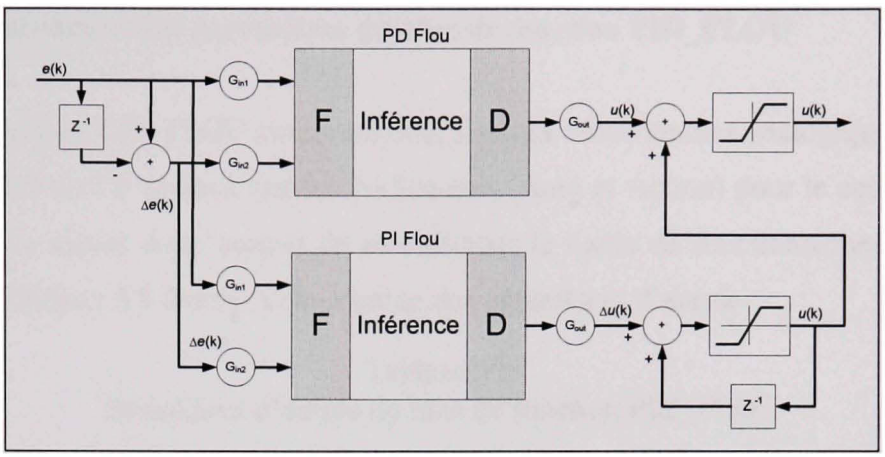


Figure 5.15 Structure du CF PID par interconnexion.

L'implantation du CF a été développée entièrement dans le logiciel RSLogix 5000®. Il s'agit d'une instruction complémentaire définie par l'utilisateur (Add-on) : l'utilisateur a la possibilité de créer des instructions personnalisées qui viennent s'ajouter au large éventail de fonctionnalités déjà intégrées dans la plateforme de commande Logix. Il peut créer simplement des bibliothèques standardisées, qui permettent à la fois réduire le temps de développement d'un projet et de fournir une certaine cohérence, de façon à réduire les dépenses liées au démarrage des équipements et à la formation. La Figure 5.16 montre l'instruction personnalisée PID_FLOU du projet.

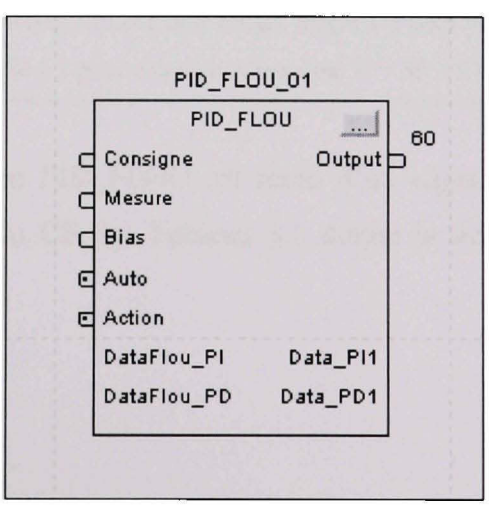


Figure 5.16 Bloc de fonction PID_FLOU.

5.4.1 Description des paramètres du bloc de fonction PID_FLOU

Le bloc de fonction PID_FLOU comporte cinq signaux : trois entrées analogiques (Consigne, Mesure et *Bias*) du CF et deux entrées booléennes (Auto et Action) pour le contrôle du bloc de fonction. Le signal Auto permet de sélectionner le mode de fonctionnement du bloc de fonction. Le Tableau 5.5 donne la description des paramètres d'entrée.

Tableau 5.5
Paramètres d'entrée du bloc de fonction PID_FLOU

Paramètre d'entrée	Type de données	Description
Consigne	Réelle	Valeur courante du point de consigne. La valeur du point de consigne est utilisée pour contrôler la variable de contrôle en mode automatique (Auto = 1).
Mesure	Réelle	Entrée de la variable procédé mise à l'échelle. Cette valeur provient en général d'un module d'entrée analogique.
Bias	Réelle	Valeur opérateur manuelle de la variable de sortie (Output). La variable de sortie est égale à cette valeur en mode manuel (Auto = 0). Si le système est en mode manuel, le bloc de fonction met Output = *Bias* à la fin de l'exécution de chaque instruction.
Auto	Booléenne	Requête programme pour le passage en mode auto. Activé par le programme utilisateur qui demande le mode Auto.
Action	Booléenne	Inversion du calcul d'erreur. Réglé à 0 pour calculer l'erreur par E = PV – SP ; réglé à 1 pour calculer l'erreur par E = SP – PV.

De plus, le bloc de fonction PID_FLOU est muni d'un signal de sortie (Output) dont la valeur du signal provient du CF. Le Tableau 5.6 donne la description des paramètres de sortie.

Tableau 5.6
Paramètres de sortie du bloc de fonction PID_FLOU

Paramètre de sortie	Type de données	Description
Output	Réelle	Sortie variable de contrôle du contrôleur flou mise à l'échelle de 0 % à 100 %. Cette sortie contrôle en général un module de sortie analogique ou une boucle secondaire.

Finalement, le bloc de fonction PID_FLOU comprend deux paramètres internes servant de table d'information contenant les données du CF. Le Tableau 5.7 donne la description des paramètres interne.

Tableau 5.7
Paramètres interne du bloc de fonction PID_FLOU

Paramètre interne	Type de données	Description
DataFlou_PI	Fuzzy	Table d'information pour le bloc de fonction FLFB du contrôleur PI flou.
DataFlou_PD	Fuzzy	Table d'information pour le bloc de fonction FLFB du contrôleur PD flou.

La façon la plus simple de mettre en œuvre l'instruction PID_FLOU est de créer une routine dans un programme, à l'intérieur d'une tâche périodique. Lorsque l'instruction PID_FLOU est utilisée dans une tâche périodique, elle utilise automatiquement la vitesse d'actualisation de la tâche périodique comme durée de rafraîchissement.

5.4.2 Principe de fonctionnement du bloc de fonction PID_FLOU

L'algorithme du contrôleur PID_FLOU régule la sortie de la variable de contrôle afin de maintenir la variable mesurée du procédé au point de consigne lorsque l'instruction s'exécute en mode Auto. Si le contrôleur PID_FLOU est en mode manuel (Auto = 0), le bloc de fonction met la sortie Output égale à la valeur de *Bias* à la fin de l'exécution de chaque instruction.

5.5 Conclusion

On a vu dans ce chapitre les différentes interfaces constituant le logiciel *Fuzzy Kernel*, leur fonctionnement, ainsi que le principe de l'algorithme de mise au point. Nous avons également présenté les deux blocs de fonction permettant l'intégration de la LF dans l'API ControlLogix®. Nous verrons dans le chapitre suivant les procédures de tests qui permettront une exploitation d'une application en temps réel ainsi que les caractéristiques d'un banc expérimental d'un procédé de niveau.

CHAPITRE 6

PROCÉDÉS UTILISÉS POUR LES TESTS

Dans ce chapitre, nous allons présenter les procédés tests qui ont été utilisés. Dans un premier temps, nous allons valider le fonctionnement du module d'inférence du logiciel. Pour tester efficacement le logiciel *Fuzzy Kernel*, nous allons comparer le fonctionnement de son module d'inférence avec ceux des logiciels *FUDGE*® et Matlab®. Par la suite, les caractéristiques de la boucle de contrôle sont présentées ainsi que ceux des deux contrôleurs, un contrôleur PID classique et un contrôleur PID flou par interconnexion. Ces derniers ont été utilisés afin de comparer les performances du CF avec un contrôleur PID classique dans une boucle de contrôle. En terminant, nous présentons l'organisation d'une séance de travaux pratiques effectuée auprès d'un groupe d'étudiants de niveau collégial dans le but d'effectuer une évaluation préliminaire de l'aspect pédagogique du logiciel.

6.1 Validation du module d'inférence de *Fuzzy Kernel*

Pour valider le module d'inférence du logiciel *Fuzzy Kernel*, on utilisera le CF de type PI. Le CF sera réalisé dans les logiciels *FUDGE*® et Matlab®, version 2008r, en respectant les caractéristiques du module d'inférence de *Fuzzy Kernel*. En se basant sur ce CF, le comportement du contrôleur sera simulé et comparé.

6.1.1 Module d'inférence du logiciel *Fuzzy Kernel*

C'est le module qui décide de l'implication et de l'agrégation, c'est-à-dire, celui qui évalue la base des règles du système flou. On rappelle que le module d'inférence du logiciel *Fuzzy Kernel* utilise une implication et une agrégation de type *max-min* (Mamdani) et une *défuzzification* de type Sugeno d'ordre zéro. Donc, le calcul de la sortie précise (non floue) utilise alors l'opérateur de Sugeno.

Le module d'inférence est intégré dans la fenêtre de mise au point en mode local. La Figure 6.1 illustre cette fenêtre.

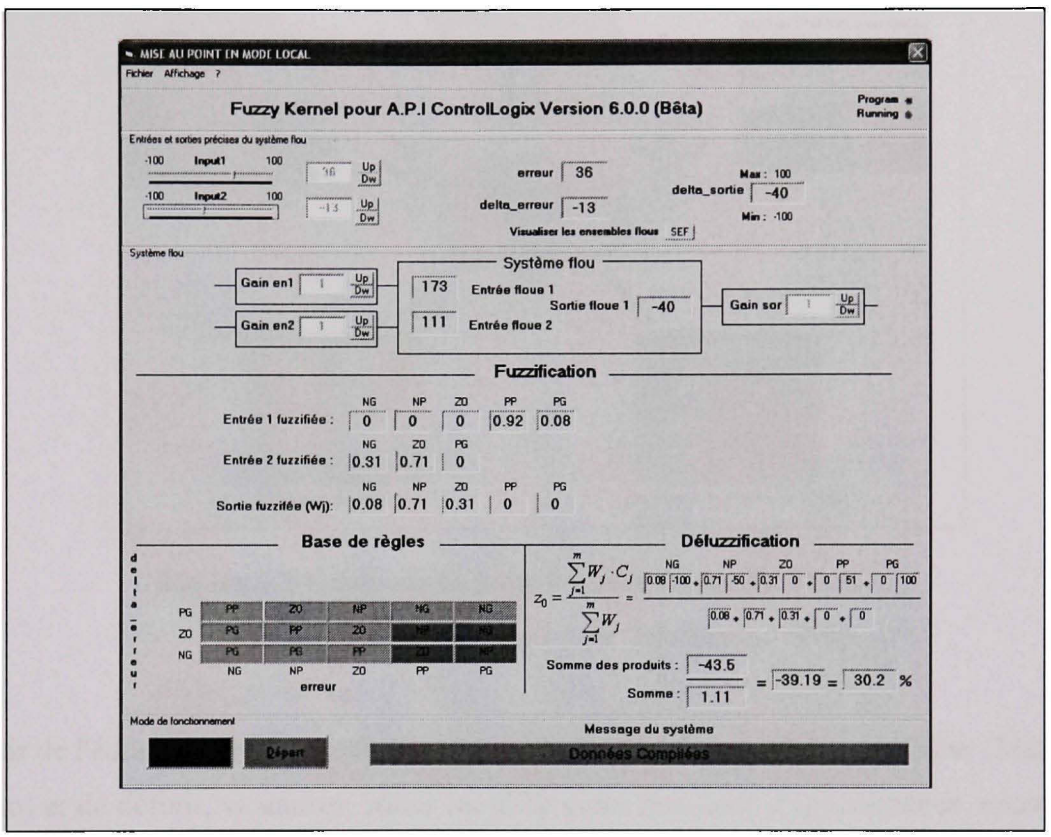


Figure 6.1 Fenêtre de mise au point en mode local du logiciel *Fuzzy Kernel.*

6.1.2 Module d'inférence du logiciel Matlab®

L'interface graphique d'inférence (*Fuzzy Inference System*, ou **FIS** en abrégé) de la boîte à outils de Matlab® permet de définir complètement le système flou. Cette interface permet de générer des fichiers « .fis », qui correspondent à des systèmes d'inférences floues et dont font partie les règles floues. Cette interface possède trois éditeurs (de fichier .fis, de règles et de fonctions d'appartenance) qui permettent de saisir l'ensemble des données du FIS ainsi que deux interfaces graphiques qui permettent de visualiser les inférences directement sur la base des règles, ainsi que des surfaces de contrôle.

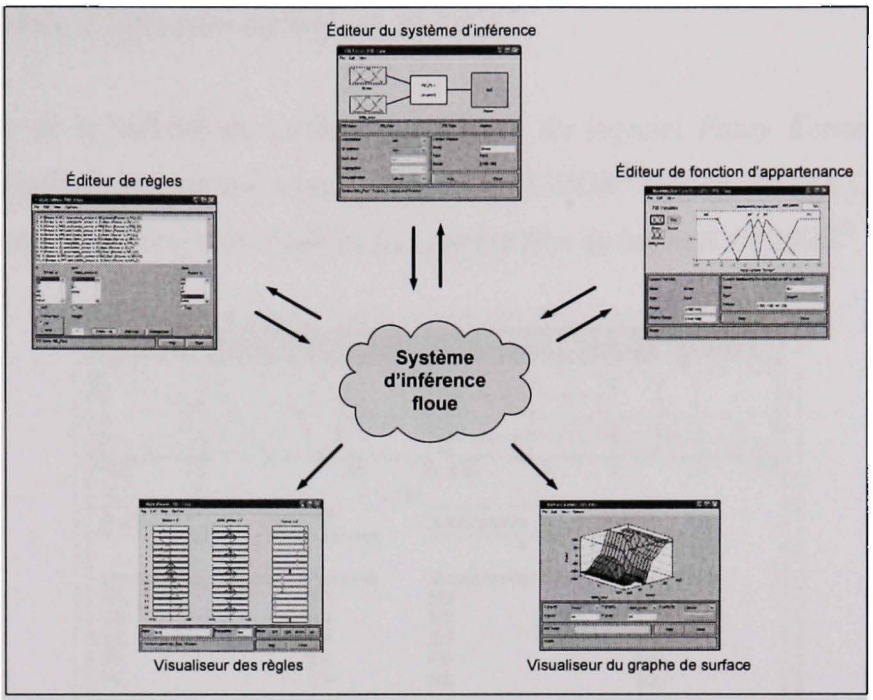


Figure 6.2 Outils de la boîte à outils Fuzzy de Matlab®.

(Tiré du manuel « Fuzzy Logic Toolbox », 1995)

À partir de l'éditeur correspondant, il est possible de choisir le type de contrôleur (Mamdani, Sugeno) et de définir, visualiser, éditer les différentes fonctions d'appartenance, construire la base des règles, choisir les méthodes d'implication (max, min, …), etc.

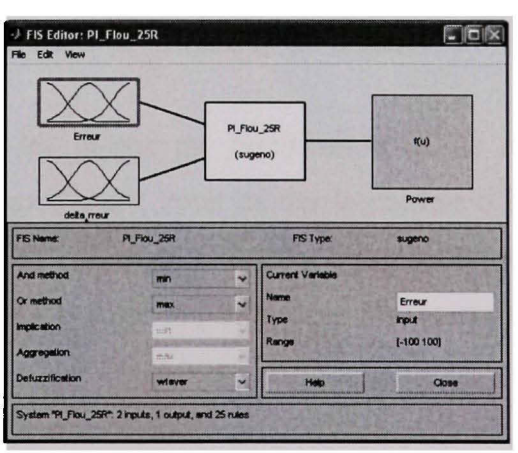


Figure 6.3 Interface graphique d'inférence de Matlab®.

(Tiré du logiciel « Matlab® »)

6.1.3 Module d'inférence du logiciel *FUDGE®*

Pour s'assurer de la validité du module d'inférence du logiciel *Fuzzy Kernel*, nous avons comparé ses résultats avec un autre logiciel nommé *FUDGE®* (FUzzy Design GEnerator). La Figure 6.4 illustre la fenêtre d'évaluation du système flou du logiciel *FUDGE®*.

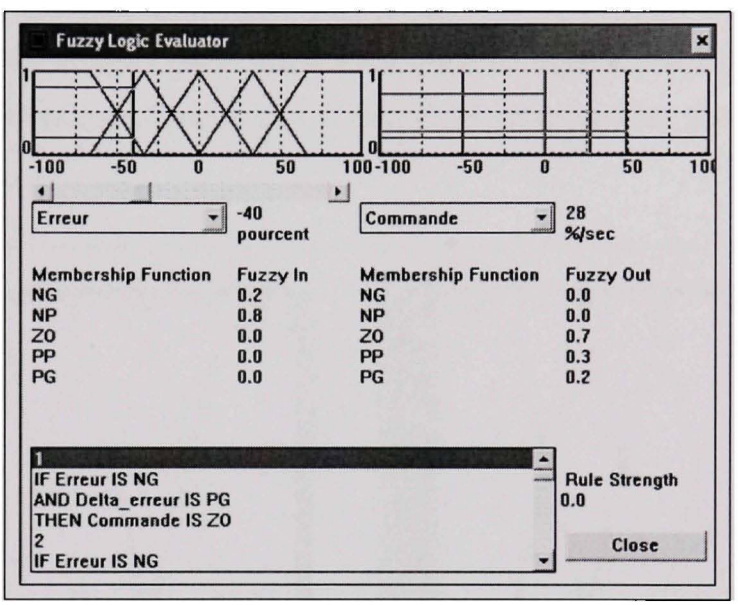


Figure 6.4 Fenêtre d'évaluation du système flou de *FUDGE®*.
(Tiré du logiciel « *FUDGE®* »)

6.1.4 Procédure de test pour la validation du logiciel

L'objectif est de valider les étapes de *fuzzification* et de *défuzzification* du module d'inférence du logiciel *Fuzzy Kernel*. Pour valider le module d'inférence, nous utilisons le contrôleur de type PI développé au chapitre 4. Pour tester différents scénarios et ainsi valider l'ensemble des 25 règles, le CF a été implanté dans les logiciels *FUDGE®* et Matlab®. Le

fichier généré par le module d'inférence floue de Matlab® s'appelle « PI_Flou_25R.fis »[7]. Nous avons implanté le même contrôleur PI flou dans le logiciel *FUDGE*®[8].

Étant donné la simplicité des équations des fonctions d'appartenance, le calcul de la *fuzzification* s'effectue facilement. Par conséquent, un total de 21 tests a été exécuté pour valider l'étape de *fuzzification* du logiciel. En ce qui concerne l'étape de *défuzzification*, un total de 25 tests a été exécuté pour valider cette étape du logiciel. La Figure 6.5 illustre les fonctions d'appartenance des entrées du CF de type PI implanté dans les logiciels *FUDGE*® et Matlab®.

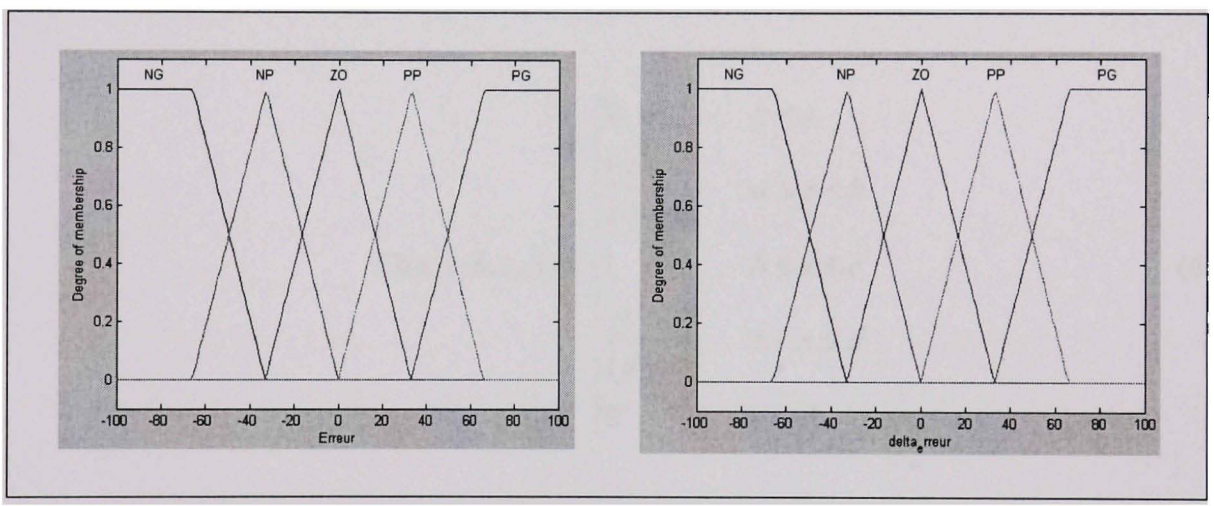


Figure 6.5 Fonctions d'appartenance des entrées du CF dans Matlab®.

La première étape consiste à déterminer, pour chaque variable d'entrées réelles, les degrés d'appartenance aux ensembles flous qui lui sont associés. Les fonctions d'appartenance sont définies relativement aux entrées par des formes trapézoïdales. La Figure 6.6 illustre la fonction d'appartenance de type trapézoïdal souvent appelé la fonction π.

[7] Le lecteur trouvera en annexe V le fichier complet du système flou utilisé dans le logiciel Matlab®.

[8] Le lecteur trouvera en annexe VII le fichier texte complet du système flou utilisé dans le logiciel *FUDGE*®.

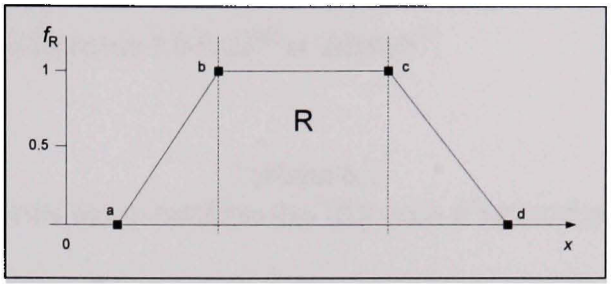


Figure 6.6 Fonction d'appartenance trapézoïdale (π).

La représentation graphique de la relation floue d'une forme trapézoïdale est définie par la fonction d'appartenance f_R suivante :

$$\Pi(x;a,b,c,d)=\begin{cases}0 & x<a\\ \dfrac{(x-a)}{(b-a)} & a\le x<b\\ 1 & b\le x\le c\\ \dfrac{(c-x)}{(d-c)} & c<x\le d\\ 0 & x>d\end{cases} \qquad (6.1)$$

Lors de la *fuzzification*, pour chaque variable d'entrée réelle, on calcule ses degrés d'appartenance aux ensembles flous qui lui sont associés. Malheureusement, l'interface graphique d'inférence du logiciel Matlab® ne permet pas de visualiser les degrés d'appartenance aux ensembles flous. Par conséquent, pour évaluer le degré d'appartenance à un sous-ensemble flou de type trapézoïdale, nous devons utiliser les instructions suivantes :

```
>>    x= 25;
>>    mfparam=[a b c d];
>>    mftype='trapmf';
>>    u=evalmf(x,mfparam,mftype);
```

où x : valeur à tester

a, b, c et d : valeurs du noyau et du support de la fonction d'appartenance.

trapmf : forme trapézoïdale de la fonction d'appartenance.

Le Tableau 6.1 montre les valeurs des paramètres des fonctions d'appartenance du CF de type PI implanté dans les logiciels *FUDGE*® et Matlab®.

Tableau 6.1
Valeurs des paramètres des fonctions d'appartenance

Fonctions d'appartenance	Paramètres			
Erreur & delta_erreur	a	b	c	d
NG	-100	-100	66	33
NP	-66	-33	-33	0
ZO	-33	0	0	33
PP	0	33	33	66
PG	33	66	100	100

La deuxième étape consiste à valider le module de *défuzzification*, et, par le fait même, le module d'inférence des règles floues du logiciel *Fuzzy Kernel*. Dans l'étape de *défuzzification*, on réalise l'opération inverse, à savoir, obtenir une valeur réelle de la sortie à partir de la base des règles obtenues dans l'étape d'inférence. Pour un système flou possédant m entrées et n sorties, l'ensemble des règles floues est défini par une matrice de règles possédant autant de lignes que d'ensembles flous de chacune des entrées et $(m+n+2)$ colonnes. Le Tableau 6.2 donne les règles de la base reliées au CF de type PI implanté dans le système flou.

Tableau 6.2
Base des règles du CF

1. If (Erreur is NG) and (delta_erreur is PG) then (delta_U is ZO) (1)
2. If (Erreur is NG) and (delta_erreur is PP) then (delta_U is PP) (1)
3. If (Erreur is NG) and (delta_erreur is ZO) then (delta_U is PG) (1)
4. If (Erreur is NG) and (delta_erreur is NP) then (delta_U is PG) (1)
5. If (Erreur is NG) and (delta_erreur is NG) then (delta_U is PG) (1)
6. If (Erreur is NP) and (delta_erreur is PG) then (delta_U is NP) (1)
7. If (Erreur is NP) and (delta_erreur is PP) then (delta_U is ZO) (1)
8. If (Erreur is NP) and (delta_erreur is ZO) then (delta_U is PP) (1)
9. If (Erreur is NP) and (delta_erreur is NP) then (delta_U is PG) (1)
10. If (Erreur is NP) and (delta_erreur is NG) then (delta_U is PG) (1)
11. If (Erreur is ZO) and (delta_erreur is PG) then (delta_U is NG) (1)
12. If (Erreur is ZO) and (delta_erreur is PP) then (delta_U is NP) (1)
13. If (Erreur is ZO) and (delta_erreur is ZO) then (delta_U is ZO) (1)
14. If (Erreur is ZO) and (delta_erreur is NP) then (delta_U is PP) (1)
15. If (Erreur is ZO) and (delta_erreur is NG) then (delta_U is PG) (1)
16. If (Erreur is PP) and (delta_erreur is PG) then (delta_U is NG) (1)
17. If (Erreur is PP) and (delta_erreur is PP) then (delta_U is NG) (1)
18. If (Erreur is PP) and (delta_erreur is ZO) then (delta_U is NP) (1)
19. If (Erreur is PP) and (delta_erreur is NP) then (delta_U is ZO) (1)
20. If (Erreur is PP) and (delta_erreur is NG) then (delta_U is PP) (1)
21. If (Erreur is PG) and (delta_erreur is PG) then (delta_U is NG) (1)
22. If (Erreur is PG) and (delta_erreur is PP) then (delta_U is NG) (1)
23. If (Erreur is PG) and (delta_erreur is ZO) then (delta_U is NG) (1)
24. If (Erreur is PG) and (delta_erreur is NP) then (delta_U is NP) (1)
25. If (Erreur is PG) and (delta_erreur is NG) then (delta_U is ZO) (1)

Dans Matlab®, pour évaluer la valeur de la sortie réelle, on utilise les instructions suivantes :

```
>>    sys_Flou= readfis('PI_Flou_25R.fis');
>>    x=[-55 30];
>>    s=evalfis(x,sys_flou);
```

où sys_Flou : fichier qui contient le système flou.

x : valeurs d'entrée à tester.

s : valeur de la sortie réelle de l'inférence floue.

Une simulation similaire est effectuée dans le logiciel *Fuzzy Kernel* à l'aide de la fenêtre de mise au point en mode local. La section *fuzzification* de cette fenêtre (Figure 6.7) permet de visualiser les degrés d'appartenance des variables d'entrée du système flou. Tandis que la section *défuzzification* permet de visualiser l'inférence du système flou.

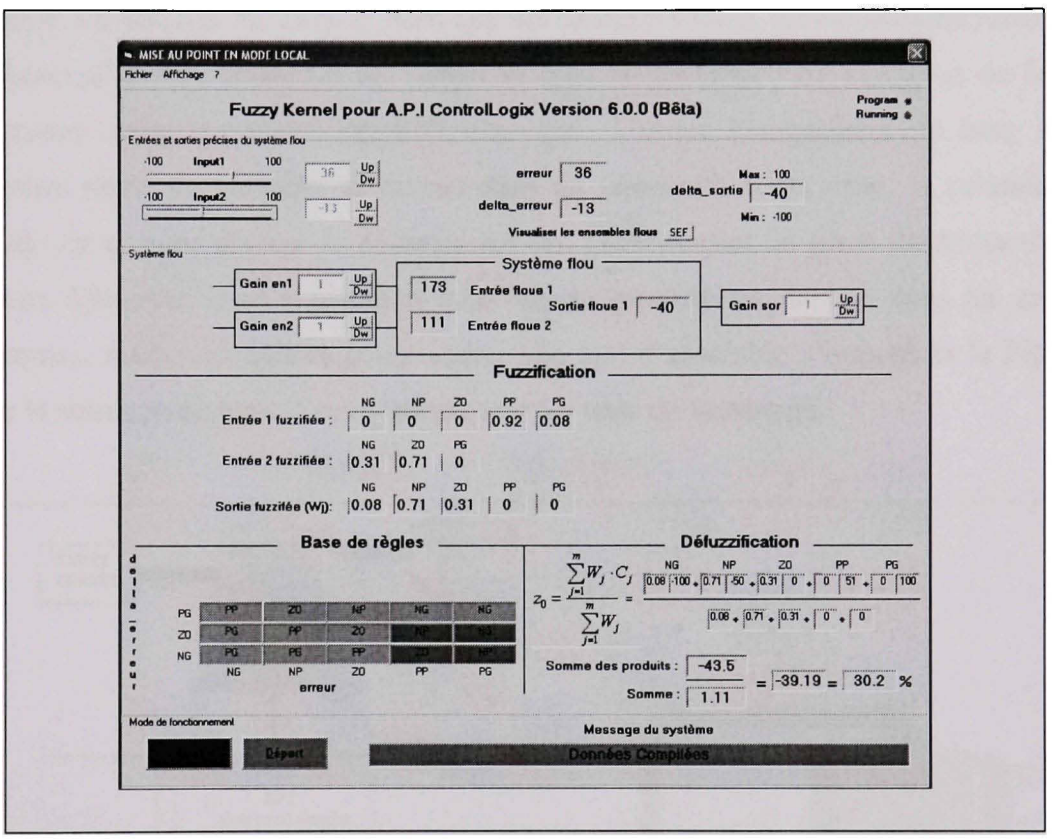


Figure 6.7 Fenêtre de mise au point en mode local du logiciel *Fuzzy Kernel*.

Grâce aux logiciels *FUDGE*® et Matlab®, nous pouvons valider le fonctionnement du logiciel *Fuzzy Kernel* proposé dans ce projet. Des résultats de simulation similaires sont comparés entre eux et les différences de valeurs sont commentées. Les résultats de la comparaison sont présentés dans le chapitre 7.

6.2 Validation du contrôleur PID flou

Bien que le logiciel soit fonctionnel lors des essais de simulation, il faut tout de même vérifier si le CF est fonctionnel sur un procédé physique. Un procédé de niveau a été choisi pour la validation du contrôleur PID flou intégré dans l'API ControLogix® que nous avons développé. Ce système est simple. Bien que ses caractéristiques intrinsèques ne justifient pas l'utilisateur d'un CF, il permet de valider le bloc de fonction PID_FLOU et de faire une comparaison avec un contrôleur PID classique. L'étude comparative se base sur une application réelle de contrôle de niveau dans un réservoir. À cet effet, la validation sera effectuée sur un banc d'essai du département des Technologies du génie électrique du cégep du Vieux Montréal. Une explication détaillée du procédé est donnée dans les prochains paragraphes, mais pour mieux comprendre, une vue d'ensemble s'impose et la Figure 6.8 illustre la structure du banc d'essai utilisé pour les tests de validation.

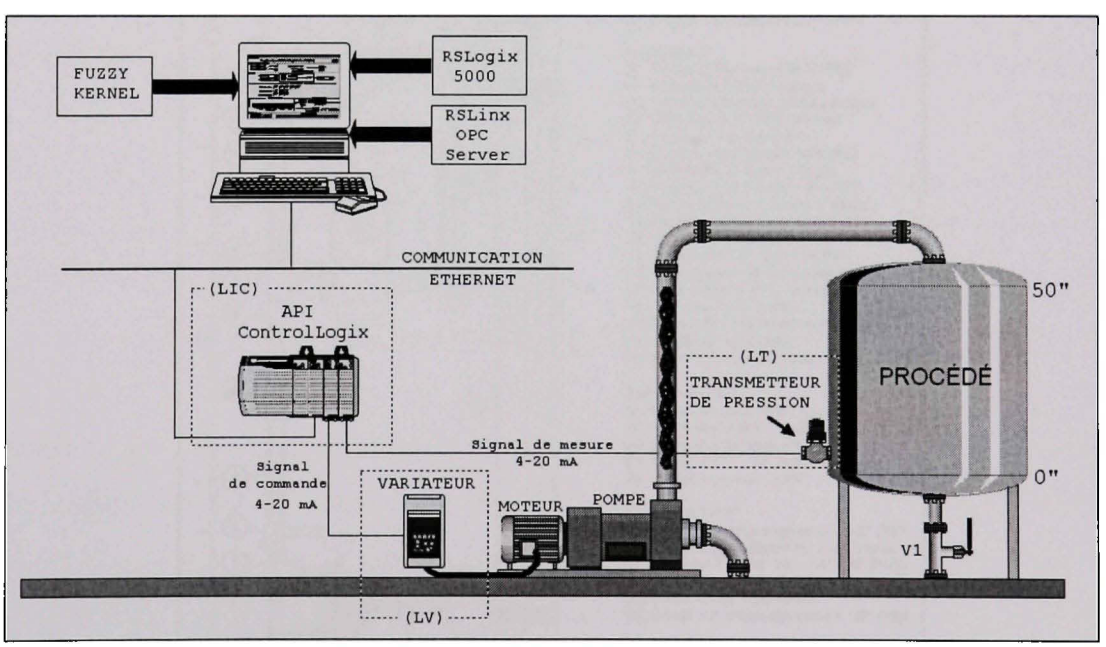


Figure 6.8 Schéma de la structure du banc d'essai.

Tout d'abord, l'élément final de contrôle (LV) est une pompe centrifuge de 1,5 HP de marque Goulds® modèle 1ST1F2C4 entraînée par un moteur triphasé d'une puissance de 1/2 HP. La commande du moteur est assurée par un variateur de vitesse PowerFlex 40® de la

compagnie Allen-Bradley®. La variable mesurée, c'est-à-dire le niveau d'eau du réservoir, est effectuée en mesurant la pression hydrostatique de la colonne à l'aide d'un transmetteur de pression (LT) de marque Rosemount® modèle 3051 série S. Le transmetteur fournit un signal analogique en courant de 4 à 20 mA pour un niveau de 0 à 50 pouces d'eau. Le transmetteur de pression est branché à une entrée analogique d'un module de l'API; par conséquent, le niveau recueilli est transmis à la partie contrôle. Le contrôle de niveau (LIC) est réalisé, premièrement, par un contrôleur PID classique et par la suite par le CF qui seront implantés dans l'API ControlLogix®. Le signal de commande fourni par un module analogique de l'API est de type standard 4 à 20 mA. Ce signal est acheminé au variateur et il permet de contrôler la vitesse du moteur, donc par le fait même le débit de la pompe. Une vanne manuelle (V1) fait office de charge en contrôlant le débit de sortie du réservoir. La Figure 6.9 illustre le dessin d'implantation du procédé de niveau.

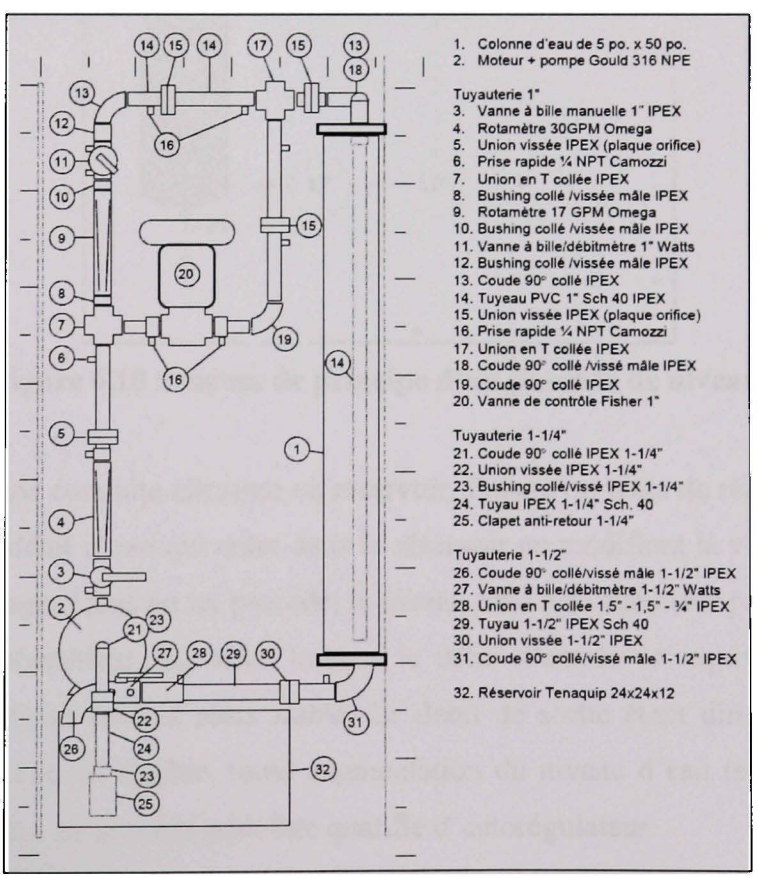


Figure 6.9 Dessin d'implantation du procédé.

6.2.1 Procédé de niveau

Nous allons détailler dans ce qui suit les caractéristiques et le modèle du système, les caractéristiques des deux contrôleurs utilisés ainsi que leurs méthodes de réglages. Le dispositif expérimental est un procédé de niveau linéaire. L'objectif final de ce procédé consiste à réguler le niveau du liquide dans le réservoir. Ce système est un système mono-entrée/mono-sortie. Sa seule entrée est la commande de la pompe qui contrôle le débit du fluide d'entrée du réservoir. La sortie du procédé est une valeur numérique venant du capteur de niveau. Tout d'abord, analysons le procédé de niveau dont le schéma de principe est fourni à la Figure 6.10.

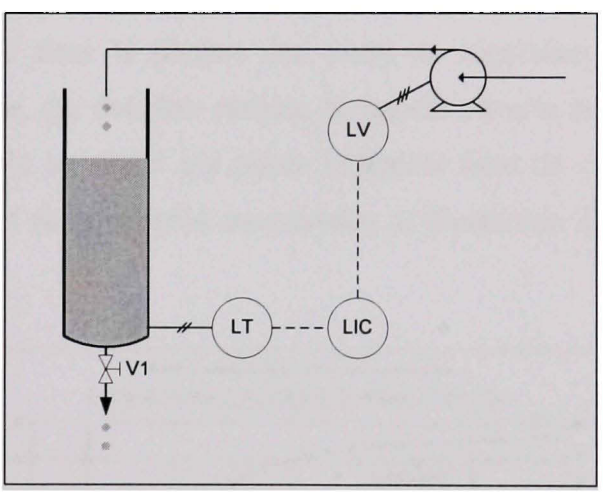


Figure 6.10 Schéma de principe d'un procédé de niveau.

On remarque qu'une conduite alimente un réservoir. Puisqu'on tente de réguler le niveau, on manipule donc le débit d'eau qui entre dans le réservoir en modifiant la vitesse d'un moteur entraînant une pompe. Dans un tel procédé, le niveau tend à se stabiliser par lui-même, et ce, sans l'aide d'un contrôleur. En effet, lorsque le débit d'entrée est équivalent au débit de sortie, le niveau d'eau devient alors stable. Le débit de sortie étant directement relié à la hauteur de la colonne du liquide, toute augmentation du niveau d'eau tend à augmenter le débit de sortie. Donc, ce procédé peut être qualifié d'autorégulateur.

6.2.1.1 Modèle mathématique du procédé

Afin de procéder adéquatement à la mise au point d'un contrôleur, il est tout d'abord nécessaire de procéder à l'analyse qualitative du procédé. Nous avons utilisé l'approche bien connue, dite *la réponse indicielle de Ziegler-Nichols*[9], introduite par John G. Ziegler et Nathaniel B. Nichols (1942), destinée à ajuster principalement les paramètres *Kp*, *Ti* et T_D d'un contrôleur classique. Cette méthode fait partie des méthodes de réglage empirique, particulièrement adaptées à des processus pour lesquels il est difficile d'obtenir des modèles mathématiques simples du comportement.

Notre choix est justifié par le fait que cette méthode, qui est connue et très populaire en industrie, est enseignée dans la plupart des cours de régulation de procédé au niveau collégial. Cette approche, qui doit être réalisée en boucle ouverte *in situ*, consiste à imposer une variation instantanée au signal qui pilote l'élément final de commande et à observer ensuite le comportement de la variable commandée. L'illustration de ce test est donnée à la Figure 6.11.

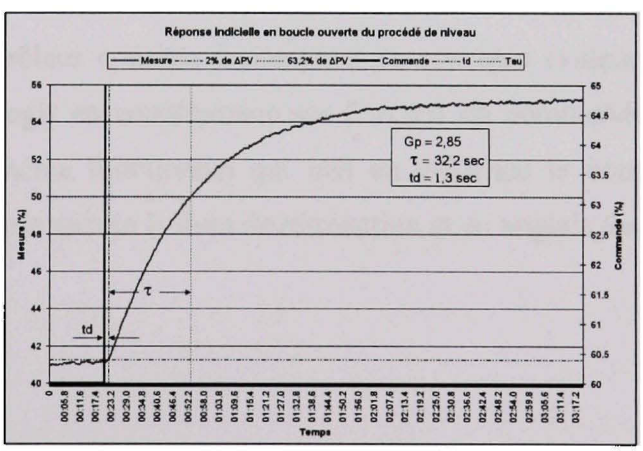


Figure 6.11 Réponse indicielle en boucle ouverte du procédé de niveau.

[9] La méthode Ziegler-Nichols est décrite à l'annexe 3.

La réponse indicielle du procédé de niveau nous donne les informations suivantes :

Tableau 6.3
Caractéristiques du procédé de niveau

le gain du procédé, G_p	2,85
la constante de temps, τ	32,2 secondes
le temps de délai, t_d	1,3 seconde

Ces valeurs sont importantes puisqu'elles permettent de prédire le comportement du procédé. Rappelons que la fonction de transfert d'un système est caractérisée par un modèle d'ordre 1 approximatif affecté par un retard pur. La fonction de transfert de notre procédé test est présentée à l'équation 6.2.

$$G_{(s)} = \frac{2,85 \cdot e^{-1,3 \cdot s}}{32,2s + 1} \tag{6.2}$$

6.2.2 Boucle de contrôle

Le principe d'un contrôleur consiste à comparer la consigne (valeur désirée) à la mesure (valeur actuelle) et à agir en conséquence sur l'organe de commande. On représente cette disposition par un schéma fonctionnel qui met en évidence le bouclage de la sortie sur l'entrée. Ce concept porte encore le nom de rétroaction et en anglais *feedback*.

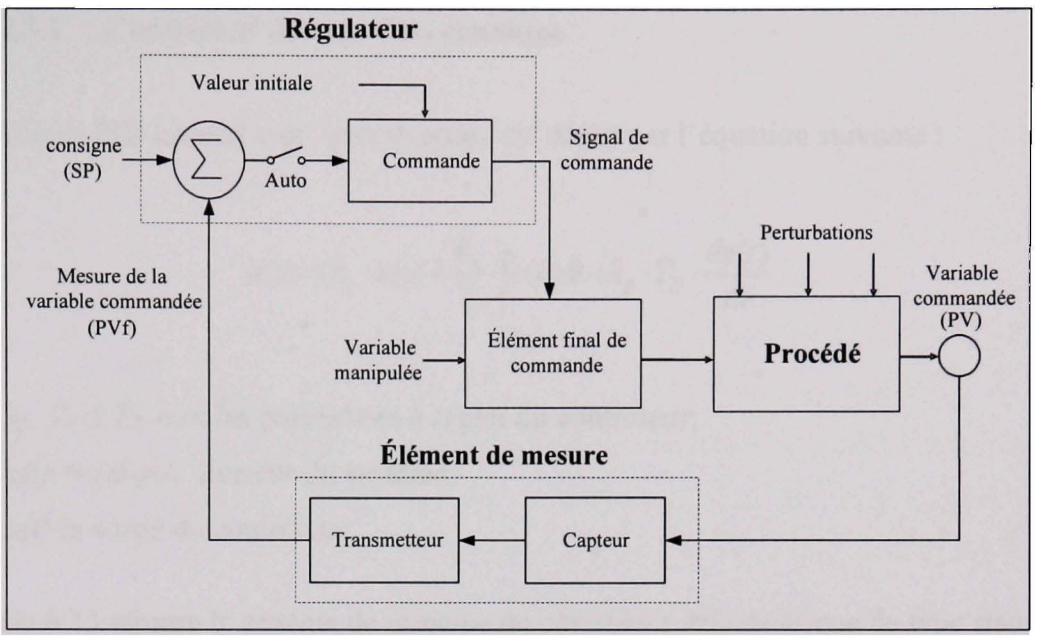


Figure 6.12 Schéma fonctionnel d'une boucle de contrôle.

Après cette étape de modélisation, nous abordons dans la suite la mise en œuvre des contrôleurs PID classiques et PID flous. L'analyse des caractéristiques intrinsèques de ces contrôleurs nous permettra de tester et de valider notre projet.

6.2.3 Contrôleur flou pour remplacer un contrôleur PID classique

Dans cette section, nous allons analyser le comportement du CF de type numérique avec la méthode min-max, en faisant ressortir les relations entre le contrôleur PID flou par interconnexion et le contrôleur PID classique. Nous allons étudier l'influence des fonctions d'appartenance, des règles de contrôle, ainsi que des paramètres de mise à l'échelle, sur la performance d'un CF en fonction de la méthode de conception classique du contrôleur PID.

6.2.3.1 Contrôleur de type PID classique

Le contrôleur PID classique de type standard est décrit par l'équation suivante :

$$u(t) = k_p \cdot e(t) + \frac{k_p}{T_i} \int_o^t e(t)dt + k_p \cdot T_D \cdot \frac{de(t)}{dt} \tag{6.3}$$

où

k_p, T_i et T_D sont les paramètres à régler du contrôleur;

$e(t)=y_r(t)-y(t)$ l'erreur du système;

$u(t)$ la sortie du contrôleur.

La Figure 6.13 montre le schéma de principe du contrôleur PID classique de type standard.

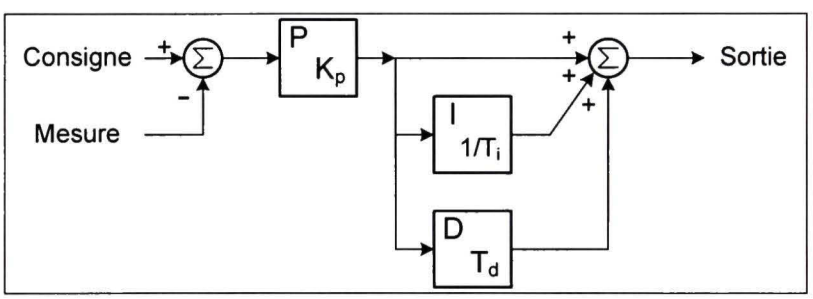


Figure 6.13 Schéma de principe du contrôleur PID classique.

6.2.3.2 Réglage du contrôleur PID classique

Dans le cadre d'une régulation à LF, nous n'utilisons pas directement le modèle d'un contrôleur PID, mais plutôt un contrôleur par interconnexion. Toutefois, l'implantation du CF est calquée sur les performances d'un contrôleur de type PID classique. Pour cette raison, nous allons présenter les réglages nécessaires à la mise au point d'un contrôleur PID classique. La mise au point d'un contrôleur PID classique consiste à faire des compromis afin d'obtenir la réponse désirée selon un critère de qualité choisi. Donc, il s'agit de trouver le point d'équilibre entre les caractéristiques qui influencent le régime transitoire du procédé :

la stabilité du procédé, l'erreur maximale et le temps de rétablissement désiré. Le contrôleur PID classique de type standard est décrit par l'équation suivante :

$$u(t) = k_p \cdot e(t) + \frac{k_p}{T_i} \int_o^t e(t)dt + k_p \cdot T_D \cdot \frac{d}{dt} e(t) \tag{6.4}$$

D'après la méthode du test de la réponse indicielle, les informations extraites permettent de procéder à la mise au point du contrôleur PI et PID classique de type standard. De manière quantitative, il s'agit de proposer les actions (P, I, D) du contrôleur et de leurs paramètres (K_p, T_i, T_d) répondant le mieux possible aux spécifications d'un cahier de charges. Il existe plusieurs méthodes pour faire la mise au point d'un contrôleur PID classique. Elles vont de la méthode empirique jusqu'aux équations mathématiques évoluées, en passant par les méthodes pratiques basées sur des recherches (Ziegler et Nichols, 1942).

Pour le réglage du contrôleur PID classique de ce projet, nous avons utilisé la méthode de *réponse indicielle de Ziegler-Nichols*. En effet, nous pouvons utiliser cette méthode décrite à l'annexe IV afin d'estimer les paramètres de réglage. Le Tableau 6.4 montre les formules à utiliser selon le mode de régulation préconisé.

Tableau 6.4
Formules de réglage des paramètres d'un contrôleur standard

Mode du contrôleur	K_p	T_i	T_D
P	$\frac{\tau}{t_d} \cdot \frac{1}{Gp}$	-	-
PI	$0,9 \cdot \frac{\tau}{t_d} \cdot \frac{1}{Gp}$	$3,3 \cdot t_d$	-
PID	$1,2 \cdot \frac{\tau}{t_d} \cdot \frac{1}{Gp}$	$2,0 \cdot t_d$	$0,5 \cdot t_d$

Cette méthode, quoiqu'imparfaite, permet de donner un ordre de grandeur aux valeurs des paramètres ou à tout le moins une valeur de départ. Par la suite, la meilleure méthode pour améliorer le réglage du contrôleur est la méthode essais/erreurs. Pour ce projet, nous utilisons seulement le contrôleur en modes PI et PID. En conséquence, d'après les équations du Tableau 6.4, on aura :

Tableau 6.5
Valeurs de réglage des paramètres d'un contrôleur standard

Mode	Gain proportionnel	Bande proportionnelle	Temps d'intégration				Temps de dérivé	
PI	K_p = 7,82	B.P. = 12,78	T_i = 4,3	**s**	K_i = 0,231	**rep/s**		
			T_i = 0,072	**min**	K_i = 13,860	**rep/min**		
PID	K_p = 10,4	B.P. = 9,59	T_i = 2,6	**s**	K_i = 0,385	**rep/s**	T_D = 0,65	**s**
			T_i = 0,043	**min**	K_i = 23,077	**rep/min**	T_D = 0,0108	**min**

Rappelons que le modèle du procédé de niveau est caractérisé par l'équation :

$$G_{(s)} = \frac{2,85 \cdot e^{-1,3 \cdot s}}{32,2s+1} \tag{6.5}$$

6.2.3.3 Implantation du contrôleur PID classique dans l'API ControlLogix®

Le contrôleur PID classique est implanté dans une tâche périodique de l'API ControlLogix® tel que suggéré par le manufacturier. Une tâche périodique exécute une fonction à une fréquence donnée. Lorsque le délai alloué à une tâche périodique arrive à terme, celle-ci interrompt la tâche continue, s'exécute une fois, et redonne la commande à la tâche continue, à l'endroit où cette dernière a été interrompue.

Nous utilisons l'instruction PIDE du ControlLogix® qui offre des capacités plus évoluées que l'instruction PID standard. Elle se base sur la variation de l'erreur de l'algorithme PID. L'algorithme de contrôle PID calcule la valeur de la variable de contrôle en sommant les

termes Delta P_{Term}, Delta I_{Term}, Delta D_{Term} et CV à partir de l'exécution précédente de l'instruction (c.-à-d. CV_{n-1}).

$$Calculated\,CV = CV_{n-1} + P_{term} + I_{term} + D_{term} \tag{6.6}$$

La Figure 6.14 illustre le contrôleur PID classique de l'API ControlLogix®.

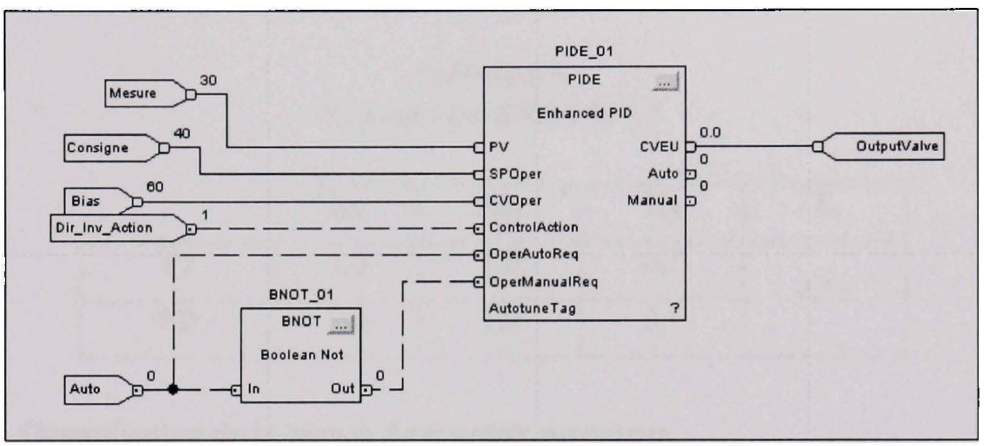


Figure 6.14 Bloc de fonction du contrôleur PID classique.

6.2.3.4 Implantation du contrôleur flou dans l'API ControlLogix®

Le CF a été implanté dans une tâche périodique de l'API ControlLogix®. La Figure 6.14 illustre le CF par interconnexion PI et PD intégré dans l'API.

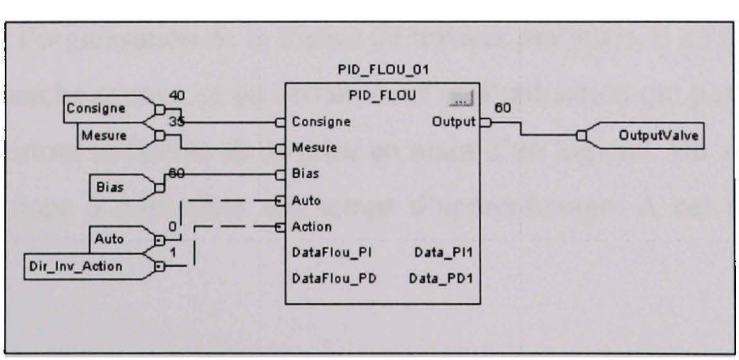


Figure 6.15 Bloc de fonction du contrôleur PID flou.

Pour le réglage du CF, il a suffi de procéder par itérations. Les gains K_E, K_{dE} et K_S ont d'abord été choisis pour que l'influence de chacun soit faible afin de ne pas provoquer d'instabilité au démarrage. Afin de diminuer l'influence des signaux de sortie, les signaux sont divisés par un gain de 10. La plage résultante est de ±10 %/seconde. Toutefois, les sorties sont saturées pour des valeurs supérieures à +100 %/s et inférieures à -100 %/s. Le Tableau 6.6 donne les valeurs pour les contrôleurs flous de types PI et PD.

Tableau 6.6
Réglages par défaut du CF

	K_E	K_{dE}	K_S	T_e
PI	1,0	1,0	0,1	0,1 s
PD	1,0	1,0	0,1	

6.3 Organisation de la séance de travaux pratiques

L'atteinte de l'énoncé de la compétence relative aux connaissances théoriques et pratiques de la LF a été évaluée à l'aide de notre projet de recherche durant l'automne 2008. Deux expérimentations en classe ont eu lieu en novembre et décembre 2008 avec 21 étudiants issus d'une classe de finissants dans le programme de Technologie de l'électronique industrielle du cégep du Vieux Montréal.

Avant de présenter l'organisation de la séance de travaux pratiques, il ne faudrait pas omettre qu'un apprenant cherche avant tout un certain confort d'utilisation qui passe évidemment par la convivialité et surtout la facilité de la prise en main d'un logiciel. Un apprenant cherche à gagner du temps, donc à minimiser son temps d'apprentissage. À cet effet, un document

intitulé « Démarrer avec *Fuzzy Kernel* » a été élaboré pour l'apprentissage rapide du logiciel[10].

Premièrement, les élèves ont travaillé, individuellement, à l'étude d'un système d'arrosage à LF expérimental suite à un exposé magistral des notions de la LF. Pour cela, ils ont utilisé, durant 1 heure, le logiciel du projet pour sa partie interface de conception.

Deuxièmement, ils ont utilisé un API ControlLogix® pour la mise en œuvre et l'analyse des performances du système d'arrosage dans un contexte de contrôle de processus en temps réel. Pour ce faire, ils ont utilisé le logiciel du projet pour la réalisation et l'apprentissage du système d'arrosage à LF. Cette expérimentation s'étend sur une séance de 2 heures alors que les étudiants travaillaient en binômes. Ils étaient déjà familiers avec l'API, car ils avaient déjà travaillé sur celui-ci dans un contexte traitant de la programmation d'automatismes industriels. De plus, ils connaissaient déjà les logiciels associés à cet API.

6.4 Conclusion

Dans ce chapitre, nous avons présenté en détail les procédures de test pour valider le fonctionnement du logiciel *Fuzzy Kernel*. Pour cette validation, nous avons codé des procédures de tests dans des fichiers Matlab® sous forme de scripts pour pouvoir les exécuter aussi souvent que nécessaire. Nous avons également présenté les procédures de tests pour valider les performances du contrôleur PID flou. Afin de procéder adéquatement à la mise au point des contrôleurs, nous avons décrit et fait une analyse qualitative du dispositif expérimental, à savoir, le procédé de niveau. Ainsi, les caractéristiques intrinsèques du procédé ont pu être déterminées. Ces caractéristiques ont servi de base pour le calcul des réglages du contrôleur PID classique. En terminant, la séance de travaux pratiques qui a

[10] Le lecteur trouvera en annexe VI le document complet du guide de démarrage du logiciel *Fuzzy Kernel*.

permis d'apporter des améliorations au logiciel a été présentée. Le prochain chapitre présente et valide les résultats obtenus pour la simulation d'un CF de type PI. La validation de la simulation est réalisée par la comparaison des résultats obtenus avec *Fuzzy Kernel* et ceux obtenus par simulation avec les logiciels *FUDGE*® et Matlab®.

CHAPITRE 7

ANALYSE ET INTERPRÉTATION DES RÉSULTATS

Ce chapitre est consacré à la présentation des résultats obtenus du module d'inférence du logiciel *Fuzzy Kernel* avec celui des logiciels *FUDGE*® et Matlab®. De plus, nous présentons les résultats obtenus avec le CF dans une boucle de contrôle. Ces derniers sont comparés avec ceux d'un contrôleur PID classique. En terminant, nous présentons les résultats de la séance de travaux pratiques faite auprès d'un groupe d'étudiants.

7.1 Module d'inférence versus Module d'inférence de Matlab®

Pour tester différents scénarios afin de vérifier la validité du module d'inférence du logiciel *Fuzzy Kernel*, les variables d'entrées ont été variées dans le module de « mise au point en mode local » du logiciel. La sortie générée par le module a été comparée avec celle calculée par le logiciel *FUDGE*®, ainsi que par le module FIS du logiciel Matlab®.

7.1.1 Résultats obtenus

Cette section présente et valide les résultats obtenus pour la simulation d'un CF de type PI. La validation de la simulation est réalisée par la comparaison des résultats obtenus avec *Fuzzy Kernel* avec des résultats de simulation obtenus avec les logiciels *FUDGE*® et Matlab®. À l'issue de cette simulation, la *fuzzification* et la *défuzzification* sont calculées. Les Tableau 7.1 et Tableau 7.2 montrent les résultats obtenus pour chacune des étapes de calcul du système flou.

Tableau 7.1
Comparaison de la *fuzzification* du signal de l'erreur

	ERREUR																				
	Valeur	Fuzzy Kernel					Excel					Fudge					Matlab				
		NG	NP	ZO	PP	PG	NG	NP	ZO	PP	PG	NG	NP	ZO	PP	PG	NG	NP	ZO	PP	PG
1	-100	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
2	-90	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
3	-80	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
4	-70	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
5	-60	0.79	0.21	0	0	0	0.82	0.18	0	0	0	0.8	0.2	0	0	0	0.82	0.18	0	0	0
6	-50	0.48	0.52	0	0	0	0.52	0.48	0	0	0	0.5	0.5	0	0	0	0.51	0.49	0	0	0
7	-40	0.2	0.8	0	0	0	0.21	0.79	0	0	0	0.2	0.8	0	0	0	0.21	0.79	0	0	0
8	-30	0	0.91	0.09	0	0	0	0.91	0.09	0	0	0	0.9	0.1	0	0	0	0.91	0.09	0	0
9	-20	0	0.6	0.4	0	0	0	0.61	0.39	0	0	0	0.6	0.4	0	0	0	0.61	0.39	0	0
10	-10	0	0.29	0.71	0	0	0	0.3	0.7	0	0	0	0.3	0.7	0	0	0	0.30	0.70	0	0
11	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
12	10	0	0	0.72	0.28	0	0	0	0.7	0.3	0	0	0	0.7	0.3	0	0	0	0.70	0.30	0
13	20	0	0	0.41	0.59	0	0	0	0.39	0.61	0	0	0	0.4	0.6	0	0	0	0.39	0.61	0
14	30	0	0	0.11	0.89	0	0	0	0.09	0.91	0	0	0	0.1	0.9	0	0	0	0.09	0.91	0
15	40	0	0	0	0.84	0.16	0	0	0	0.79	0.21	0	0	0	0.8	0.2	0	0	0	0.79	0.21
16	50	0	0	0	0.53	0.47	0	0	0	0.48	0.52	0	0	0	0.5	0.5	0	0	0	0.48	0.52
17	60	0	0	0	0.22	0.78	0	0	0	0.18	0.82	0	0	0	0.2	0.8	0	0	0	0.18	0.82
18	70	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
19	80	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
20	90	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
21	100	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Tableau 7.2
Comparaison de la *fuzzification* du signal de la variation de l'erreur

	VARIATION DE L'ERREUR																				
	Valeur	Fuzzy Kernel					Excel					Fudge					Matlab				
		NG	NP	ZO	PP	PG	NG	NP	ZO	PP	PG	NG	NP	ZO	PP	PG	NG	NP	ZO	PP	PG
1	-100	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
2	-90	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
3	-80	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
4	-70	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
5	-60	0.79	0.21	0	0	0	0.82	0.18	0	0	0	0.8	0.2	0	0	0	0.82	0.18	0	0	0
6	-50	0.48	0.52	0	0	0	0.52	0.48	0	0	0	0.5	0.5	0	0	0	0.51	0.49	0	0	0
7	-40	0.2	0.8	0	0	0	0.21	0.79	0	0	0	0.2	0.8	0	0	0	0.21	0.79	0	0	0
8	-30	0	0.91	0.09	0	0	0	0.91	0.09	0	0	0	0.9	0.1	0	0	0	0.91	0.09	0	0
9	-20	0	0.6	0.4	0	0	0	0.61	0.39	0	0	0	0.6	0.4	0	0	0	0.61	0.39	0	0
10	-10	0	0.29	0.71	0	0	0	0.3	0.7	0	0	0	0.3	0.7	0	0	0	0.30	0.70	0	0
11	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
12	10	0	0	0.72	0.28	0	0	0	0.7	0.3	0	0	0	0.7	0.3	0	0	0	0.70	0.30	0
13	20	0	0	0.41	0.59	0	0	0	0.39	0.61	0	0	0	0.4	0.6	0	0	0	0.39	0.61	0
14	30	0	0	0.11	0.89	0	0	0	0.09	0.91	0	0	0	0.1	0.9	0	0	0	0.09	0.91	0
15	40	0	0	0	0.84	0.16	0	0	0	0.79	0.21	0	0	0	0.8	0.2	0	0	0	0.79	0.21
16	50	0	0	0	0.53	0.47	0	0	0	0.48	0.52	0	0	0	0.5	0.5	0	0	0	0.48	0.52
17	60	0	0	0	0.22	0.78	0	0	0	0.18	0.82	0	0	0	0.2	0.8	0	0	0	0.18	0.82
18	70	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
19	80	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
20	90	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
21	100	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Tableau 7.3
Comparaison de la *défuzzification*

			Logiciel Fuzzy Kernel	Logiciel Fudge	Logiciel Matlab
	Erreur	Delta Erreur	Delta U	Delta U	Delta U
1	-100	-100	100.00	100.00	100.00
2	-50	-100	100.00	100.00	100.00
3	0	-100	100.00	100.00	100.00
4	50	-100	27.00	25.00	24.20
5	100	-100	0.00	0.00	0.00
6	-100	-50	100.00	100.00	100.00
7	-50	-50	100.00	100.00	100.00
8	0	-50	74.00	75.00	75.80
9	50	-50	0.00	0.00	0.00
10	100	-50	-26.00	-25.00	-24.20
11	-100	0	100.00	100.00	100.00
12	-50	0	74.00	75.00	75.80
13	0	0	0.00	0.00	0.00
14	50	0	-74.00	-75.00	-75.80
15	100	0	-100.00	-100.00	-100.00
16	-100	50	27.00	25.00	24.20
17	-50	50	0.00	0.00	0.00
18	0	50	-74.00	-75.00	-75.80
19	50	50	-100.00	-100.00	-100.00
20	100	50	-100.00	-100.00	-100.00
21	-100	100	0.00	0.00	0.00
22	-50	100	-26.00	-25.00	-24.20
23	0	100	-100.00	-100.00	-100.00
24	50	100	-100.00	-100.00	-100.00
25	100	100	-100.00	-100.00	-100.00

7.1.2 Comparaison des résultats flous du simulateur avec les logiciels de référence

Le CF de type PI implanté dans les logiciels *FUDGE*® et Matlab® permet donc de vérifier la validité du fonctionnement des modules de *fuzzification* et de *défuzzification* du logiciel *Fuzzy Kernel*. Les Tableau 7.1 et Tableau 7.2 présentent les résultats obtenus de la *fuzzification* pour le signal de l'erreur et le signal de la variation de l'erreur. Les résultats obtenus par les logiciels de référence sont très semblables aux résultats obtenus avec le logiciel *Fuzzy Kernel*. À la suite des résultats, nous pouvons confirmer que l'implication des antécédents du module de *fuzzification* du logiciel *Fuzzy Kernel* fonctionne correctement selon l'opérateur *min*. Ces résultats sont très satisfaisants même si le peu d'exemples utilisés ne permet pas de les rendre totalement représentatifs.

Le Tableau 7.3 présente les résultats obtenus de la *défuzzification* pour le CF. Les résultats obtenus par les logiciels de référence sont très semblables aux résultats obtenus avec le logiciel *Fuzzy Kernel*. Finalement, tous les tests réalisés sur les logiciels de référence ont montré que les résultats coïncident pratiquement avec les résultats obtenus à l'aide de *Fuzzy Kernel*. Par conséquent, on peut conclure que le système d'inférence de *Fuzzy Kernel* fonctionne bien selon le modèle de Sugeno.

7.2 Contrôleur flou versus contrôleur classique

Dans cette section, nous allons analyser le comportement du CF par interconnexion en faisant ressortir les relations entre le CF et le contrôleur PID classique. Nous allons étudier l'influence des réglages, des règles de contrôle, ainsi que des paramètres de mise à l'échelle, sur la performance d'un CF en fonction de la méthode de conception classique du contrôleur PID.

7.2.1 Réglage de départ

Avant la mise en service du contrôleur, c'est-à-dire avant de le placer en mode automatique, il est nécessaire de s'assurer que les conditions initiales du procédé soient respectées; les conditions initiales ont une influence sur la dynamique du procédé, donc par le fait même sur les performances des contrôleurs. Toutes les vannes manuelles sont ouvertes au maximum, ne provoquant ainsi aucune restriction à l'écoulement du fluide. Le niveau du réservoir est d'abord stabilisé à 40 % pour un signal de commande de 60 % puis le contrôleur est placé en mode automatique. Pour le procédé de niveau, les conditions initiales choisies sont les suivantes :

- toutes les vannes sont ouvertes;
- commande à 60 % avec le contrôleur en mode manuel;
- ajustement de l'ouverture de la vanne V1 pour avoir un niveau stable de 40 % dans le réservoir.

De cette façon, les caractéristiques intrinsèques du procédé resteront sensiblement les mêmes pour tous les tests effectués sur celui-ci. En utilisant exactement le même système, ainsi que les mêmes réglages initiaux, il sera plus facile de comparer les performances des deux contrôleurs. Le réglage des paramètres K_p, T_i et T_D du contrôleur PID classique est fait à l'aide de la méthode *Zigler-Nichols* à partir de la réponse indicielle du procédé. Cette méthode a été présentée au chapitre précédent. Le Tableau 7.4 donne les valeurs pour un contrôleur classique PI et PID.

Tableau 7.4
Réglages du contrôleur classique selon Zielgler-Nichols

	K_P	T_I	T_D
PI	7,82	0,072 minute	-
PID	10,4	0,043 minute	0,0108 minute

Les réglages choisis sont ceux du contrôleur PID pour un temps d'échantillonnage de 0,1 seconde. Pour le réglage du CF, les réglages par défaut ont été choisis. Le Tableau 7.5 donne les valeurs pour les contrôleurs flous de types PI et PD.

Tableau 7.5
Réglages par défaut du CF

	K_E	K_{dE}	K_S	T_e
PI	1,0	1,0	0,1	0,1 s
PD	1,0	1,0	0,1	

Par la suite, des changements de consigne sont provoqués. Les changements de consigne sont de 40 % à 80 % et de 80 % à 40 %. En respectant les réglages de départ, cela permet de comparer les performances du CF avec celle du contrôleur PID classique.

7.2.2 Résultats obtenus

Lors des changements de consigne, les résultats obtenus ont montré que les performances du CF étaient légèrement supérieures à celles obtenues avec un contrôleur PID classique. Les Figure 7.1 et Figure 7.2 représentent la réponse du procédé de niveau à une variation de la valeur de la consigne pour un échelon positif de 40 % et un échelon négatif, également de 40 %, pour les contrôleurs PI et PID respectivement.

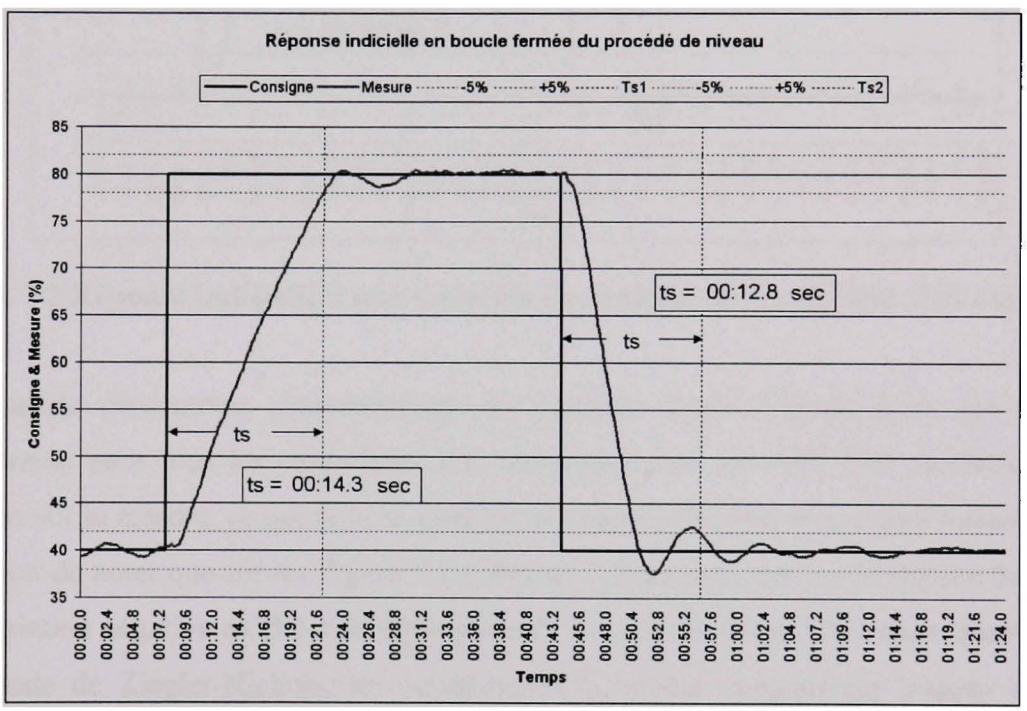


Figure 7.1 Réponse indicielle à une variation de consigne du contrôleur PI classique.

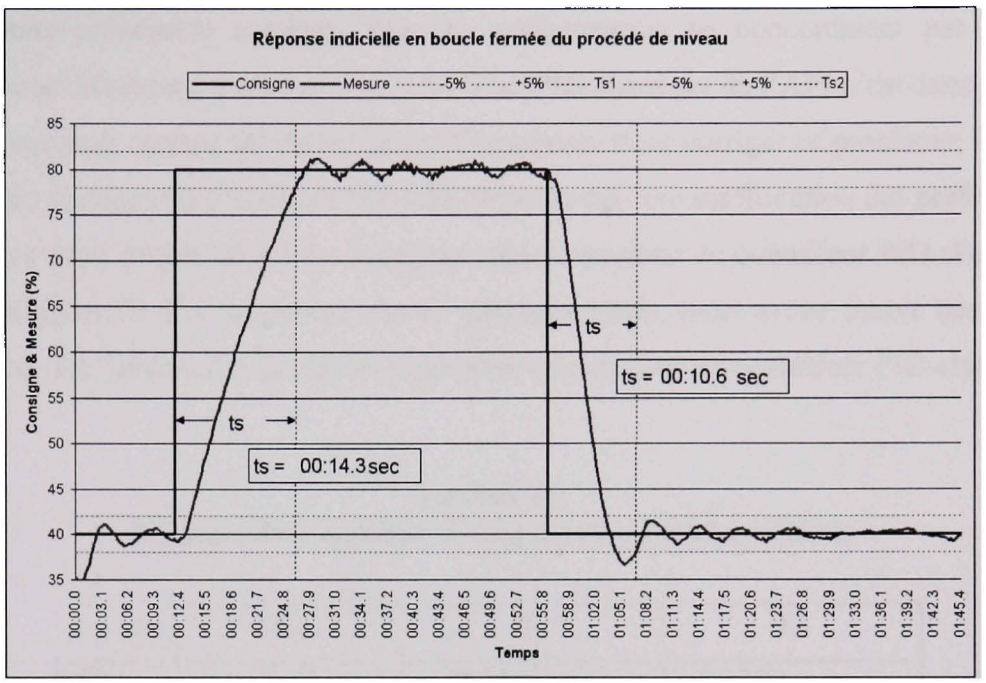


Figure 7.2 Réponse indicielle à une variation de consigne du contrôleur PID classique.

Les résultats démontrent clairement que la méthode Zigler-Nichols n'est pas adaptée correctement pour tous les contrôleurs PID classiques dans les API. Des oscillations sont présentes sur la mesure, ce qui rend le système très peu stable pour ne par dire instable. Il est intéressant de noter que sur les Figure 7.1 et Figure 7.2, on peut voir sur le régime transitoire de la variation négative qu'il y a un dépassement. Avec le contrôleur PID classique réglé par la méthode de Ziegler-Nichols, les conditions d'instabilité apparaissent lorsque le mode dérivé est utilisé. Le système est au bord de l'instabilité. Une raison qui peut être à l'origine de ces problèmes est que la méthode Z-N est basée sur une méthode empirique et que la moindre erreur de manipulation ou d'interprétation des caractéristiques lors du test peut entraîner une très grande erreur dans les calculs des réglages. En effet, par exemple, une faible erreur d'interprétation du temps de délai du procédé donne des réglages trop agressifs qui provoquent une réponse indicielle brusque ou oscillatoire. D'autre part, il est à noter que les contrôleurs issus de l'utilisation de la méthode de Z-N privilégient la rapidité au détriment de la stabilité.

Les résultats précédents montrent que les performances ne concordaient pas avec les performances théoriques attendues. Le contrôleur PID classique de l'API n'est donc pas réglé correctement pour obtenir les meilleures performances. Pour corriger ce problème, il faudrait modifier les réglages du contrôleur PID afin de permettre une amélioration des performances. Nous avons donc décidé de refaire le réglage des paramètres du contrôleur PID classique en utilisant la méthode d'essais/erreur. Après quelques essais, nous avons trouvé des réglages satisfaisants. Le Tableau 7.6 donne les nouvelles valeurs pour un contrôleur PID classique.

Tableau 7.6
Réglages du contrôleur classique selon Zielgler-Nichols

	K_P	T_I	T_D
PID	2,77	0,28 minute	0,0328 minute

La Figure 7.3 représente la réponse du procédé de niveau à une variation de la valeur de la consigne pour un échelon positif de 40 % et un échelon négatif, également de 40 % pour les nouveaux réglages (Tableau 7.7).

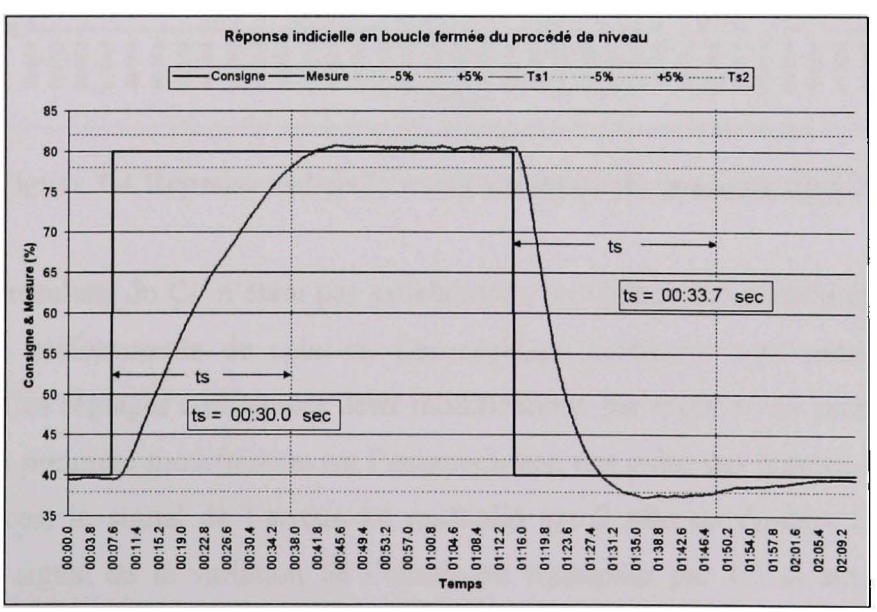


Figure 7.3 Réponse indicielle améliorée du contrôleur PID classique.

On constate donc une nette amélioration de la réponse indicielle à la variation de consigne positive. Par contre, bien que les oscillations de la réponse indicielle à la variation de consigne négative aient disparu un léger dépassement subsiste encore. Toutefois, la stabilité du procédé a été grandement améliorée. Les tests similaires ont été effectués sur le contrôleur PID flou. La Figure 7.4 représente la réponse du procédé de niveau à une variation de la valeur de la consigne pour un échelon positif de 40 % et un échelon négatif, également de 40 %, pour le contrôleur PID flou.

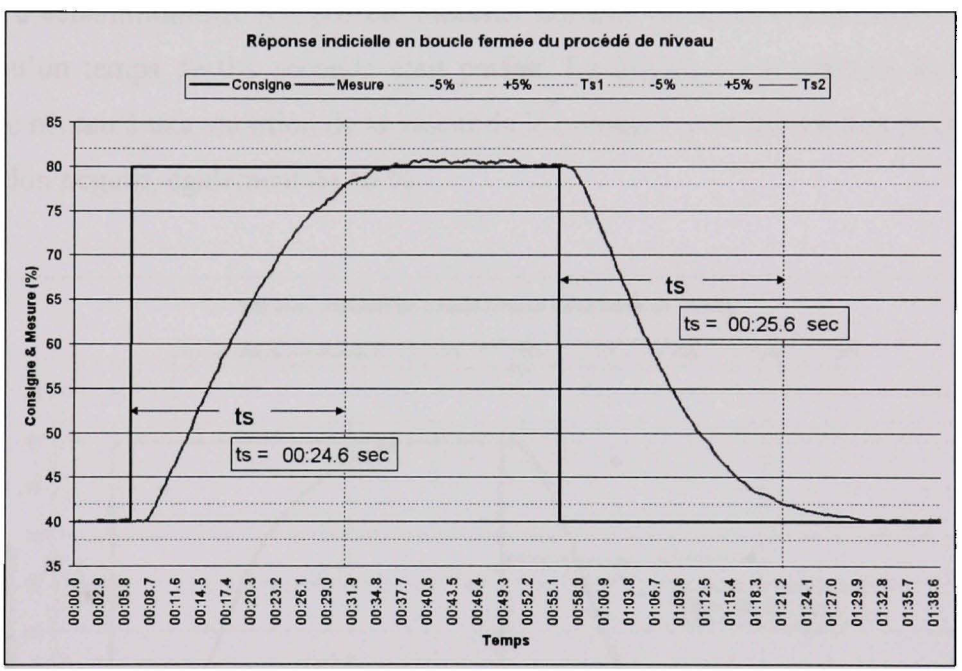


Figure 7.4 Réponse indicielle à une variation de consigne du CF.

Les premiers résultats du CF n'étant pas satisfaisants, un réglage des gains a été réalisé pour améliorer les performances de celui-ci. Les réglages améliorés sont présentés dans le Tableau 7.7. Ces réglages comportent deux modifications par rapport aux premiers réglages par défaut. La première modification est l'augmentation des gains des entrées du CF de type PI. Dans ce cas, le signal de l'erreur est multiplié par 2 afin de doubler l'amplitude de l'erreur et le signal de la variation de l'erreur est multiplier par 1,5 et ainsi obtenir une résolution (influence) augmentée.

Tableau 7.7
Réglages optimaux du CF

	K_E	K_{dE}	K_S	T_e
PI	2,0	1,5	0,1	0,1 s
PD	1,0	1,0	0,1	

Le temps d'échantillonnage n'a pas été modifié, puisque qu'après expérimentation, il a été observé qu'un temps de 0,1 seconde était parfait. La Figure 7.5 représente la réponse du procédé de niveau à une variation de la valeur de la consigne pour un échelon positif de 40 % et un échelon négatif, également de 40 %.

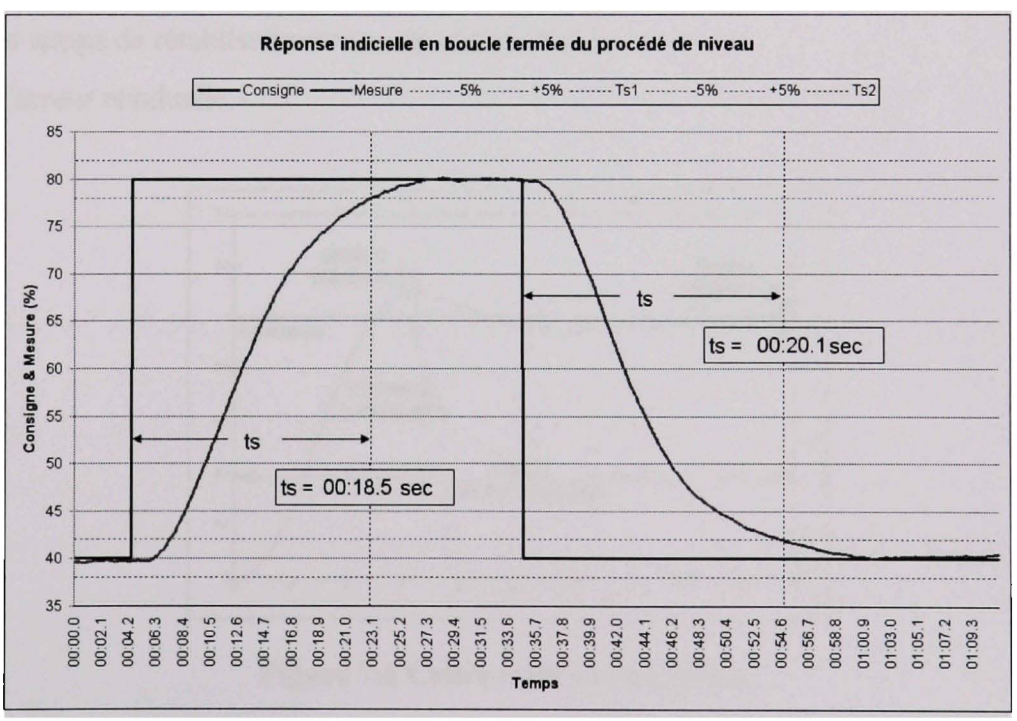


Figure 7.5 Réponse indicielle à une variation de consigne du CF.

Le second test a donné de bons résultats. Le temps total requis pour ramener la mesure à la consigne est optimal tout en maintenant une erreur nulle en régime permanent.

7.2.3 Critères de comparaison des performances

Afin de comparer les contrôleurs, il faut définir des critères de comparaison. Un contrôleur se doit de maintenir la mesure de la variable commandée égale à la consigne, et ce, malgré les perturbations, les changements de charge ou de consigne. Selon les critères de performance choisis, il existe plusieurs approches pour optimiser les réglages d'un contrôleur. Nous allons présenter dans ce paragraphe les critères de performance les plus populaires dans le domaine du contrôle de procédé. Il est raisonnable de s'attendre à ce que des erreurs soient présentes temporairement, car le système de régulation réagit aux erreurs. Il existe trois grandes caractéristiques (Figure 7.6) que l'on peut observer lors de la réponse d'un système asservi. De manière qualitative, les critères à satisfaire sont les suivants :

- l'erreur maximale;
- le temps de rétablissement (ou de stabilisation);
- l'erreur résiduelle.

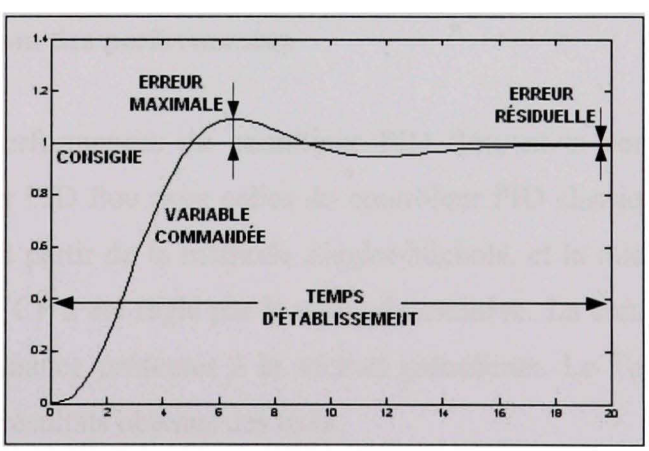


Figure 7.6 Critères de comparaison.

Après une perturbation ou un changement de consigne, le système doit réagir rapidement, avec une erreur minimale, et se stabiliser à la valeur désirée. Voici la définition des critères de comparaison :

Erreur maximale : Valeur maximale de dépassement entre la consigne et la variable commandée (mesure).

%Erreur maximale : Erreur maximale par rapport à la variation de mesure ($\%E_{max}=E_{max}/\Delta$mesure).

Erreur résiduelle : Différence entre la consigne et la variable commandée (mesure) après un changement de consigne ou une perturbation externe sur le procédé.

Temps d'établissement : Temps requis pour que le système atteigne une erreur inférieure à 5 % de la valeur finale de la mesure.

7.2.4 Comparaison des performances

Afin d'évaluer les performances du contrôleur PID flou, nous comparons les meilleurs résultats du contrôleur PID flou avec celles du contrôleur PID classique. Le contrôleur PID classique a été réglé à partir de la méthode Ziegler-Nichols, et la mise au point final par la méthode intuitive. Le CF a été réglé par la méthode intuitive. La comparaison est basée sur les critères de performance présentés à la section précédente. Le Tableau 7.8 présente un résumé des meilleurs résultats obtenus des tests.

Tableau 7.8
Résumé des meilleurs résultats obtenus

Variation (40 % à 80 %)	Erreur maximale	% Erreur maximale	Erreur résiduelle	Temps de stabilisation (±5 %)
PID classique (Ziegler-Nichols)	1,10	2,750 %	±1 %	14,3 secondes
PID classique (amélioré)	0,77	1,925 %	0,0 %	30,0 secondes
PID flou	0,0	0,0 %	0,0 %	18,5 secondes
Variation (80 % à 40 %)	**Erreur maximale**	**% Erreur maximale**	**Erreur résiduelle**	**Temps de stabilisation (±5 %)**
PID classique (Ziegler-Nichols)	3,13	7,825	±0,5 %	10,6 secondes
PID classique (amélioré)	2,58	6,450 %	0,0 %	37,7 secondes
PID flou	0,0	0,0 %	0,0 %	20,1 secondes

Par rapport au contrôleur PID classique, on voit clairement (Tableau 7.8) que le CF présente une plus grande rapidité. Pour ce qui est de l'erreur résiduelle, elle est nulle dans les deux cas. Le CF par interconnexion satisfait les critères de performance désirés, c'est-à-dire, avoir un procédé stable, une erreur minimale et un temps de stabilisation le plus rapide possible.

7.3 Résultats de la séance de travaux pratiques

Suite à la séance de travaux pratiques effectués avec un groupe d'étudiants cibles, nous avons pu apporter des corrections et des améliorations au logiciel. Nous faisons ainsi une liste non exhaustive des dites améliorations qui ont été apportées au logiciel.

- pour rendre l'apprentissage de la LF plus rapide, nous avons ajouté la BR du système flou sous forme matricielle dans la fenêtre de mise au point. Cette matrice est animée, par conséquent, le logiciel permet de visualiser les règles actives de la base des règles floues;

- la *défuzzification* par le calcul de la moyenne pondérée est représentée de manière dynamique;
- l'écran de mise au point en mode local offre la possibilité de visualiser les sous-ensembles flous des entrées et de la sortie du système flou lors de la mise au point. Ceci permet de visualiser graphiquement le degré d'appartenance à une valeur réelle d'entrée en fonction des sous-ensembles flous de chacune des entrées du système flou. De plus, cela permet également de visualiser la valeur nette inférée par l'agrégation résultante de la *défuzzification*.

De plus, après observation du groupe d'étudiants cibles, nous avons pu constater que même sans consulter le didacticiel, au moins la moitié du groupe était capable de suivre une démonstration et de travailler sans aide particulière lors des exercices. Cette approche nous a permis de déterminer l'efficacité de l'apprentissage de la LF à l'aide de notre projet de recherche en comparaison avec les outils non pédagogiques. L'approche a permis également de favoriser l'apprentissage d'une commande avancée, telle que le CF, dans un contexte de régulation de processus.

Cette séance de travaux pratiques d'une durée de 3 heures, a permis aux étudiants de mettre en œuvre rapidement un système à commande floue et de montrer l'intérêt de la LF dans un environnement industriel (API). À titre de comparaison, nous estimons, selon notre expérience, que la durée d'une séance de travaux pratiques identique utilisant le logiciel Matlab® aurait été de 9 heures. Toutefois, il est à noter que le développement de ces travaux pratiques n'aurait pas pu être fait sur un procédé réel sans avoir recours à un matériel supplémentaire pouvant effectuer l'acquisition de données externes. L'ajout de ce matériel aurait nécessité un apprentissage et une mise en œuvre supplémentaires, ceci aurait facilement doublé le temps nécessaire à la réalisation de ces travaux pratiques.

7.4 Conclusion

Nous avons démontré expérimentalement la validité du module d'inférence du logiciel *Fuzzy Kernel*. Le CF de type PI a été implanté et testé dans deux logiciels professionnels, *FUDGE*® et Matlab®. Les résultats des tests ont montré que le module d'inférence du logiciel *Fuzzy Kernel* développé est fonctionnel.

Les performances des contrôleurs PID classiques et flous ont été analysées dans une application réelle, à savoir, le procédé de niveau. Les résultats ont montré que le CF permet de commander un procédé réel efficacement. On peut donc confirmer que le contrôleur développé permet la commande d'un procédé physique. Toutefois, l'objectif final n'est pas d'analyser ses performances, mais plutôt de valider son fonctionnement. Ainsi, il pourra être utilisé dans d'autres applications que celle analysée par ce projet.

CONCLUSION

Le but de ce projet était d'élaborer un logiciel d'enseignement et d'application de la logique floue (LF) dans un contexte d'automate programmable. L'engouement envers la logique floue que le milieu industriel a connu, dès le début des années 90, n'était qu'une première onde qui reflétait fidèlement l'orientation de cette technologie. La réaction enthousiaste des marchés internationaux s'explique par les besoins de tout milieu envers cette technologie attendue depuis longtemps; cette dernière créerait des machines vraiment intelligentes et conviviales. Par exemple, la commande du métro de Sendai au Japon qui contrôle l'accélération et décélération suivant 19 règles traduisant les limitations sur la vitesse, la sécurité, le confort, de consommation d'énergie et la traçabilité. Que dire de la lessiveuse à linge « à logique floue » qui, possédant plus de 400 cycles préprogrammés, est toutefois contrôlée par seulement un bouton de départ? Il appert que cette nouvelle méthode technologique s'implante sur le marché de l'électronique industrielle : le milieu scolaire doit former une main-d'œuvre compétente et habilitée à intervenir dans un contexte de pointe.

L'implantation au niveau collégial d'une compétence visant les notions de la logique floue devient une réalité incontournable. Dans l'atteinte des différents objectifs globaux des entreprises, les systèmes de contrôle ont contribué à diminuer les coûts de production ou à augmenter la quantité ou à améliorer la qualité de leurs produits. Jusqu'à tout récemment, cette tâche était dévolue aux contrôleurs PID classiques. Toutefois, ces contrôleurs ont atteint le maximum de leur possibilité. En effet, ceux-ci sont souvent mal réglés ou tout simplement utilisés à défaut d'autres choix, ce qui occasionne, entre autres, une diminution de la performance. L'une des avenues empruntées afin d'améliorer les systèmes de contrôle est d'introduire de la LF à l'intérieur de la boucle de contrôle. En effet, la LF a démontré sa capacité à améliorer les performances des systèmes de contrôle non linéaires et adaptatifs.

Au même moment cependant, la LF est méconnue des enseignants de cégep, et par le fait même, des étudiants finissants des programmes techniques en Technologies du génie électrique. Ainsi, la LF est peu enseignée au niveau collégial, ceci s'explique par un manque

d'outils pédagogiques efficaces. En effet, tous ceux qui sont intéressés par la logique floue sont confrontés à un manque d'outil de mise en œuvre à la fois technique et pédagogique. Ils sont souvent désemparés par un manque d'outil simple d'utilisation, convivial et peu dispendieux pour la mise en œuvre de système flou, si bien que les démarches plus avancées ne se rendent qu'aux concepts théoriques. Les logiciels existants ne sont pas adaptés pédagogiquement pour faire l'apprentissage de toutes les étapes de fonctionnement d'un système à logique floue. Tous ces logiciels sont fondés sur l'implantation d'un modèle fermé dont les étapes de fonctionnement ne sont pas explicites et auquel les utilisateurs n'ont pas accès. Un autre inconvénient majeur des logiciels professionnels est l'apprentissage souvent long et difficile à réaliser selon la complexité de l'application. Par conséquent, pour de nombreux partisans de la logique floue, en plus d'entraîner des coûts supplémentaires, l'apprentissage de la logique floue entre dans le cadre d'un apprentissage long des différentes fonctions et des nombreux paramètres à configurer pour l'utilisation des logiciels de type professionnel. Cette recherche propose donc un logiciel qui vient combler cette importante lacune.

Comme l'apprentissage du fonctionnement d'un système flou n'est pas trivial pour un novice, nous avons fait des choix technologiques et pédagogiques pour contrer cette difficulté. Dans le but d'optimiser le temps d'apprentissage, nous avons volontairement limité certains paramètres du système flou, comme le nombre d'entrées et de sorties. Cette limitation a pour avantage de minimiser la base des règles (BR) à élaborer. On comprendra que plus les systèmes sont complexes, plus la dimension de la BR du système flou sera grande et plus difficile sera l'élaboration des règles d'inférences floues. De même, le type d'inférence ainsi que la méthode de *défuzzification* ont été limités. La méthode d'inférence utilise les opérateurs *min* et *max* pour l'implication et l'agrégation, tandis que la méthode de Sugeno d'ordre zéro a été préconisée pour la *défuzzification*. Ces deux opérateurs peuvent être représentés par une méthode graphique simple qui facilite leur compréhension. Quant à la méthode de Sugeno d'ordre zéro, elle offre une méthode de calcul simple, rapide et très explicite pour la *défuzzification*. Au plan éducatif, ces choix sont avantageux, alors qu'au plan opérationnel, ils ne sont pas contraignants.

En Amérique du Nord, Allen-Bradley®, filiale de la Rockwell International Corporation, est un chef de file en automatisation. Les produits Allen-Bradley® constituent une part majoritaire du marché industriel nord-américain. Pour toutes ces raisons, nous avons choisi d'intégrer le système flou dans l'automate programmable industriel (API) ControlLogix® de la compagnie Allen-Bradley®. Pour des raisons de souplesse et de facilité d'utilisation de l'outil de développement, ainsi que la capacité d'une bonne intégration avec les produits disponibles et la réduction des coûts et du temps de développement, ce projet a été réalisé à l'aide de l'outil de développement Microsoft Visual Basic 6.0®. Ceci permet une bonne intégration avec les produits industriels disponibles. Le contrôleur PID flou proposé est une combinaison de deux contrôleurs flous (CF) standards (PI flou, PD flou). Pour un apprentissage simplifié, l'apprenant peut, s'il le désire, utiliser seulement la partie PI du contrôleur. En conséquence, cette option peut améliorer significativement l'apprentissage de la LF et des CF en diminuant le nombre de paramètres à régler. De cette façon, le contrôleur PID flou proposé permet d'envisager une transition progressive du contrôleur PID classique au contrôleur PID flou.

Les fruits de ce projet offrent une solution pédagogique complète et innovatrice intégrée dans le cadre de l'enseignement collégial. Ils offrent une grande ouverture fonctionnelle avec la possibilité d'un déploiement rapide de l'enseignement de la LF et des CF dans plus de 20 cégeps québécois impliqués dans la formation des futurs technologues du programme de Technologie de l'électronique industrielle. *Fuzzy Kernel* est une première solution pour l'enseignement de la LF et des CF articulée autour d'une véritable approche pédagogique; il offre notamment un environnement unique, simple et convivial pour faciliter l'accès et la prise en main des utilisateurs et surtout un accès inégalé à tous les résultats de calculs intermédiaires tout en ne nécessitant aucun codage. Ce projet constitue un véritable catalyseur pédagogique et peut se montrer utile dans des situations très différentes. *Fuzzy Kernel* s'impose comme la plateforme pédagogique pour l'enseignement de la LF la plus performante du domaine. Il permet à l'utilisateur de concevoir, gérer, modifier et simuler des systèmes d'inférence floue en quelques minutes, au lieu des heures généralement nécessaires avec les offres s'appuyant exclusivement sur les logiciels professionnels traditionnels.

En ce qui concerne les principales caractéristiques et fonctionnalités intéressantes du logiciel, nous pouvons débuter par mentionner qu'il n'est pas une boîte noire, dans le sens où des traitements ne seraient pas visibles. Beaucoup d'attention a été accordée au module de mise au point dans le but d'avoir un outil le plus convivial possible en intégrant les principales étapes de calcul d'un système flou. Une visualisation détaillée de tous les calculs des différents composants du système flou et l'animation de la base des règles sous forme matricielle est affichée pour démontrer le fonctionnement détaillé d'un système à logique floue. *Fuzzy Kernel* offre également un assistant qui guide l'utilisateur novice lors de la création d'un système flou. Il permet de configurer toutes les informations de base nécessaires pour créer un système flou. En plus de disposer d'un module d'analyse et de simulation, *Fuzzy Kernel* présente également la particularité de pouvoir s'interfacer et s'intégrer à l'environnement existant d'un programme dans un API grâce au bloc de fonction FLFB. Il offre la possibilité d'exporter et de formater les données floues du système vers un API. Le contrôleur PID flou proposé, qui est réalisé à l'aide de deux blocs de fonction FLFB, permet de réaliser des traitements en LF afin d'optimiser la commande de procédés réels à partir des API. Le contrôleur flou implantable dans un API est réalisé à l'aide de deux blocs de fonction FLFB. Le contrôleur flou optimise les performances de contrôle en remplaçant les modèles de contrôleurs classiques. Ces travaux auront un impact positif sur l'enseignement de la LF, notamment l'apprentissage rapide de la LF et des CF en toute simplicité et par conséquent de promouvoir la LF.

Pour valider le logiciel de ce projet, un contrôleur PI flou a été implanté dans les logiciels de *FUDGE*® et Matlab®, servant de référence. Les résultats obtenus montrent hors de tout doute que le logiciel de ce projet, *Fuzzy Kernel*, fonctionne conformément selon la méthode d'inférence de type Sugeno. Le contrôleur PID flou proposé a été testé dans une application réelle. Les résultats ont démontré que le CF fonctionnait conformément aux critères de performance recherchés. Par conséquent, ce projet démontre que le contrôleur PID flou proposé est capable de remplacer parfaitement un contrôleur PID classique. Ceci permet d'affirmer que le choix du CF comme structurant pour l'enseignement de la LF dans un contexte de régulation de procédé est tout à fait approprié.

Nous sommes persuadé que notre solution pédagogique, comprenant le logiciel et le matériel pédagogique, possède un potentiel de bénéfices significatifs pour le système d'enseignement québécois et, par ricochet, pour l'industrie québécoise. En effet, pour accompagner le logiciel, un guide de démarrage rapide a été réalisé. Ce document a pour objectif, entre autres, de faciliter l'utilisation du logiciel et de permettre la réalisation d'un système flou complet. Cette solution pédagogique a servi à l'enseignement de la LF dans le cadre d'un cours de niveau collégial. Lors de ce cours, nous avons vérifié trois choses : le niveau de réduction du temps d'apprentissage, le niveau d'intérêt suscité pour l'apprentissage de la LF et finalement, l'impact de l'apprentissage de la LF sur les apprenants en utilisant le guide de démarrage. Non seulement le logiciel a démontré son utilité pour l'apprentissage et la mise en œuvre simples d'un système flou, mais encore il a permis de leur faciliter l'acquisition des connaissances et de leur rendre l'apprentissage stimulant. Nous estimons une réduction du temps d'apprentissage d'environ 80 % par rapport au temps requis avec des logiciels commerciaux. On a pu constater qualitativement que l'utilisation de la solution pédagogique proposée a permis de maintenir un intérêt soutenu tout au long de la séance. Le système flou a été réalisé aisément par l'entremise des interfaces graphiques simples. Nous avons constaté un enthousiaste et stimulé la curiosité envers la LF.

Suite à l'utilisation de *Fuzzy Kernel*, la logique floue a rapidement fait des adeptes. Un étudiant raconte « J'ai bien aimé la logique floue, je trouve cela très intéressant. Je vais choisir la logique floue pour mon travail d'exposé oral en français ...». Incontestablement, *Fuzzy Kernel* aidera les apprenants à mieux comprendre les notions théoriques et pratiques de base de la LF. Un enseignant explique « J'avais suivi, il y a quelques années, un séminaire sur les concepts de la logique floue; malheureusement, je n'avais pas saisi toutes les subtilités des concepts. Par contre, l'utilisation du logiciel *Fuzzy Kernel* m'a permis de comprendre la logique floue...».

Même si dédiée à l'enseignement de la LF et des CF au niveau collégial, la solution des travaux de ce projet peut aussi offrir une plateforme intéressante pour répondre aux problématiques d'enseignement et de recherche et développement des établissements universitaires et des écoles de génie. En plus d'un usage similaire à celui des cégeps, un tel outil pourra être utilisé dans un contexte plus large dont en autres, la prise de décision. La forte présence des boucles de contrôle dans une multitude d'établissements industriels en fait une des toutes premières références dans le domaine des logiciels pour l'enseignement de la LF et des CF. Ce projet nous permet de croire que d'ici quelques années, nous aurons contribué, par la formation des futurs techniciens, à promouvoir la LF et les CF dans le milieu industriel. Par conséquent, les futurs techniciens auront, en la LF, un outil supplémentaire pour l'optimisation des boucles de contrôle en industrie.

ANNEXE I

Énoncé de la compétence de la logique floue

Énoncé de la compétence

Connaître et mettre en œuvre une commande avancée à logique floue par rapport à une commande classique.

Éléments de la compétence

- Donner une base théorique indispensable à la compréhension des principes de bases de la logique floue.
- Présenter les méthodes et outils nécessaires à l'intégration de la logique floue dans les commandes de processus industriels.
- Analyser les étapes de fonctionnement d'un système flou.
- Présenter les phases de conception, de synthèse et de mise en œuvre d'un contrôleur flou.
- Élaborer un contrôleur flou de type PID dans un système contrôle-commande.
- Analyser les performances d'un contrôleur flou de type PID dans un contexte de régulation de procédé réelle.

Stratégie pédagogique

Les notions de la logique floue, et par le fait même celles des contrôleurs flous, font partie intégrante de deux énoncés de compétence du programme et seront vues dans le cadre du cours 243-502-VM « Programmer un système de régulation ».

Énoncés de compétence :

- 043A Programmer des parties commande;
- 043H Participer à la conception d'un projet de contrôle-commande.

La stratégie pédagogique visera, plus particulièrement, deux éléments de compétence de l'ensemble des deux énoncés. L'élément de compétence 3 de l'énoncé *043A* : « Tester le fonctionnement des programmes » et l'élément de compétence 2 de l'énoncé *043H* : « Déterminer les stratégies de contrôle-commande à utiliser ».

Toutes les activités théoriques et pratiques reliées à la logique floue serviront à illustrer notamment comment celle-ci a permis des développements industriels spectaculaires à partir d'algorithmes très simples de traduction de connaissances linguistiques en entité numérique. Ils serviront à présenter des applications de la logique floue qui ont eu un impact sur les techniques de contrôle des processus automatiques en permettant de faire le lien entre les contrôleurs PID classiques et les contrôleurs flous. L'objectif final est de permettre à l'étudiant de maîtriser les techniques permettant l'automatisation des procédés industriels.

Afin d'atteindre les compétences d'apprentissage, les activités sont fortement orientées vers la réalisation, à partir des principes de base de la logique floue, d'un correcteur flou et de l'analyse de ses performances.

Contenu du cours

INTRODUCTION (0 h 30)

- Rappels sur les objectifs de la régulation.
- Historique de la logique floue.
- Domaines d'application de la logique floue.

PRINCIPES DE BASE DE LA LOGIQUE FLOUE (3 h)

- Introduction à la logique floue.
- Les bases de la commande floue :

  - Les variables d'entrée / sortie d'un algorithme flou;
  - Les fonctions d'appartenance;
  - Les opérateurs flous;
  - Univers de discours;
  - Schéma de la commande floue.

- Structure d'une commande floue :

  - Base des règles et définitions;
  - Interface de *fuzzification*;
  - Mécanisme d'inférence;
  - Interface de *défuzzification*.

- Exemples d'applications industrielles.

REGULATION FLOUE (1 h 30)

- Caractéristiques d'un correcteur flou.
- Méthodologie pour la conception d'un correcteur flou.
- Réglage d'un correcteur flou.

TRAVAUX PRATIQUES

Application des concepts de la logique floue : (2 h)

- Élaboration d'un système flou à l'aide d'un outil d'aide pédagogique.
- Interprétation des étapes de fonctionnement d'un système flou.
- Simulation d'un système flou à l'aide d'un outil d'aide pédagogique.

Application à la commande de procédés industriels : (4 h)

- Élaboration d'un régulateur flou dans un automate programmable industriel.
- Définition des variables d'entrée, de la variable de sortie et des règles floues.
- Utilisation d'un régulateur flou de type PID dans un processus de contrôle.
- Comparaison des performances d'un régulateur flou de type PID avec celle d'une commande classique PID.

Source : Ce texte a été tiré des documents du cours 243-502-VM « Programmer un système de régulation » de Claude Théorêt et de Guy Charbonneau du cégep du Vieux Montréal.

ANNEXE II

Opérateurs en logique floue

Il existe de nombreuses variantes des opérateurs en logique floue. Les définitions les plus souvent rencontrées sont : le max et le min (Mamdani et Zadeh), le produit et la somme moins le produit (Sugeno). Le tableau suivant nous donne les variantes les plus communément utilisées, à savoir celles basées sur la logique floue de Zadeh.

Tableau II-1
Opérateurs logiques floues les plus utilisés

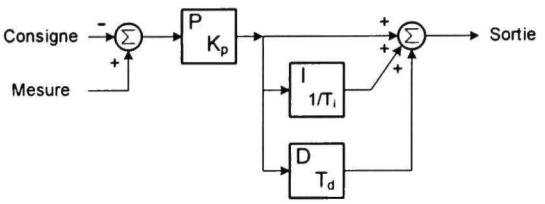
Appellation	Intersection « ET » (t-norme)	Union « OU » (t-conorme)	Complément « NON »
Zadeh	$\mu_{A\cap B}(x)=\min(\mu_A(x),\mu_B(x))$	$\mu_{A\cup B}(x)=\max(\mu_A(x),\mu_B(x))$	$\mu_{\bar{A}}(x)=1-\mu_A(x)$
Probabiliste	$\mu_{A\cap B}(x)=\mu_A(x)\times\mu_B(x)$	$\mu_{A\cup B}(x)=\mu_A(x)+\mu_B(x)-\mu_A(x)\times\mu_B(x)$	$\mu_{\bar{A}}(x)=1-\mu_A(x)$
Lukasiewicz	$\mu_{A\cap B}(x)=\max(\mu_A(x)+\mu_B(x)-1,0)$	$\mu_{A\cap B}(x)=\min(\mu_A(x)+\mu_B(x),1)$	$\mu_{\bar{A}}(x)=1-\mu_A(x)$

ANNEXE III

Différent modèle régulateur classique

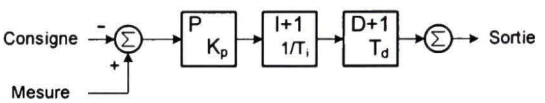
1. Modèle Standard

a) Modèle théorique, idéal.
b) Non-interactif: le bloc I n'a pas d'influence sur le bloc D et vice-versa.
c) Toutes les formules de réglage sont données en fonction de ce modèle.



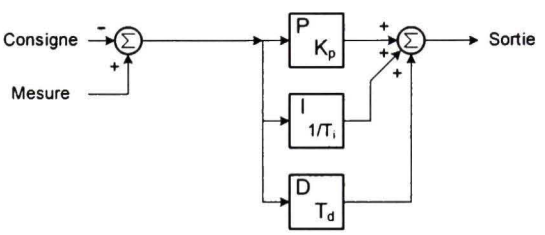
2. Modèle Série

a) Interactif.
b) Très utilisé.



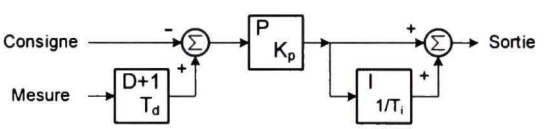
3. Modèle Parallèle

a) Non-interactif.
b) Utilisé par Bailey et Allen-Bradley.



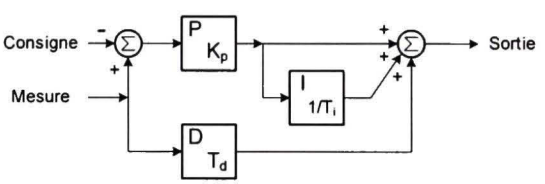
4. Modèle Différentiel sur la mesure

a) Interactif: formules Série.
b) dC/dt: D insensible, se comporte comme un PI.



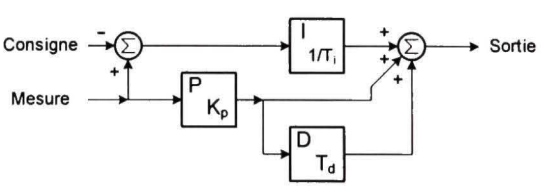
5. Modèle Dérivé sur la mesure

a) Non-interactif: formules Standard.
b) dC/dt: D insensible, se comporte comme un PI.



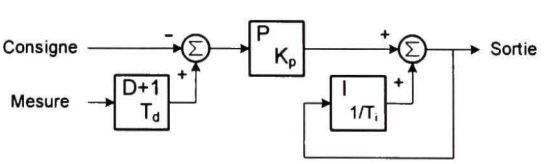
6. Modèle Proportionnel et Dérivé sur la mesure

a) Non-interactif: formules Standard.
b) dC/dt: P et D insensibles, seul I corrige doucement.
c) Honeywell: ÉQUATION B



7. Modèle Intégral sur la sortie

a) ≈ formules Série.
b) dC/dt: D insensible.
c) Moore Micro-352



ANNEXE IV

Test de la réponse indicielle de Ziegler-Nichols

Cette approche, qui doit être réalisée en boucle ouverte *in situ*, consiste à imposer une variation instantanée au signal qui pilote l'élément final de commande et à observer ensuite le comportement de la variable commandée. Après la stabilisation du système, il faut imposer une seconde variation au signal de l'élément final, mais cette fois, le saut doit être de signe contraire (on revient ainsi à l'état initial). Pour que le test soit valide, il est nécessaire que le saut imposé modifie le procédé d'un état stable à un second état stable. Afin d'analyser les résultats ultérieurement, il est judicieux de posséder un enregistreur graphique qui inscrit sur papier tous les détails de la réponse indicielle du procédé.

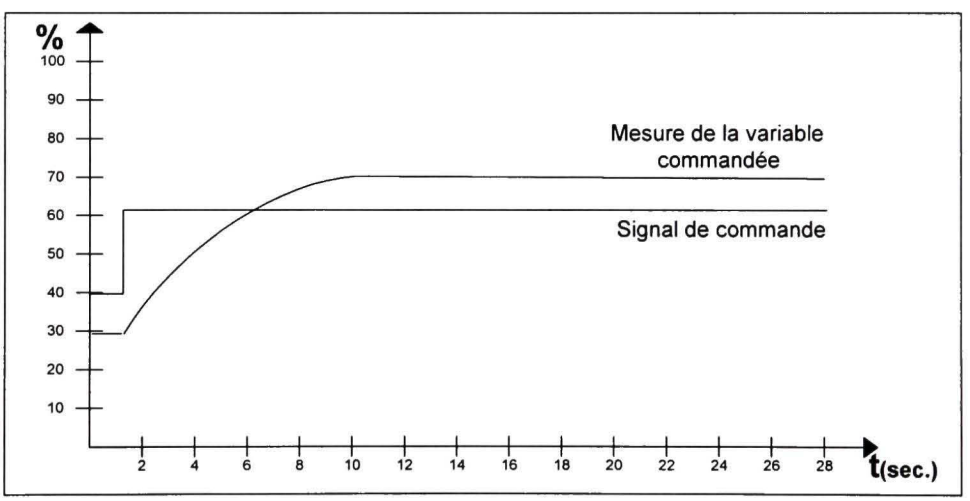


Figure IV-1 Réponse indicielle du procédé en boucle ouverte.

La réponse indicielle nous renseigne sur les informations suivantes :

- le gain du procédé, *Gp*;
- la constante de temps, τ;
- le temps de délai, *td.*

Ces valeurs sont importantes puisqu'elles permettent de prédire le comportement du procédé.

Le gain du procédé est le rapport entre la variation de la grandeur mesurée et la variation du signal de commande.

$$Gp = \frac{\text{variation du signal mesuré}(\Delta PV)}{\text{variation du signal de commande}(\Delta OUT)} \qquad \text{(IV.1)}$$

Le gain de procédé est une information très importante, car il permet de déterminer la sensibilité du système asservi en réponse au signal de commande. Dans un procédé de régulation de niveau, le gain de procédé est influencé par la densité des liquides en présence. En effet, la hauteur de la colonne de liquide, qui est nécessaire pour assurer un débit de sortie équivalent au débit d'entrée, sera très différente selon que le liquide est léger ou lourd. Donc, le gain de procédé nous renseigne sur la valeur avec laquelle le procédé réagit à une perturbation.

La constante de temps (τ ou *tau*) du procédé permet de déterminer la rapidité avec laquelle le système aurait atteint le prochain état stable (la valeur finale pour une variation du signal de commande donnée) et ce, en conservant la vitesse initiale. En réalité, le temps nécessaire pour atteindre la valeur finale est plus élevé puisque le rythme n'est pas maintenu; il est plutôt réduit graduellement.

Le tau est un paramètre important lorsque le système réagit à une perturbation. En effet, la vitesse de réaction du système est d'abord dictée par la constante de temps. Si des perturbations continuelles sont présentes, le procédé débute sans cesse un nouveau régime transitoire dont la pente est proportionnelle au tau.

Dans un procédé de régulation de niveau, la modification du diamètre du réservoir influence directement la constante de temps, et ce, sans changer le gain de procédé. Le gain est inchangé, car il est fonction de la hauteur de la colonne de liquide (ou de sa densité). Par contre, la constante de temps est grandement affectée, car la quantité de liquide nécessaire varie en fonction de la capacité du réservoir. Il est plus usuel de définir la constante de temps comme étant le temps nécessaire pour atteindre 63,2 % du changement total.

Le temps de délai est le retard que l'on observe entre le moment où un signal est appliqué au procédé (une variation du signal de commande ou une perturbation) et le moment où on observe une réaction sur la variable mesurée. Un temps de délai pur fait que retarder le signal, et ce, sans l'altérer. Le temps de délai est la conséquence directe du retard relié au transport de matériel ou d'énergie. Par contre, ce type de délai est très nuisible, car il retarde l'information et qu'il est alors impossible d'anticiper les réactions du procédé.

Il existe deux approches permettant de déterminer les paramètres du modèle équivalent du procédé. Nous présentons, ici, la première approche de Ziegler-Nichols, dite *la méthode 2 % et 63,2 %*. Tout d'abord, il importe de déterminer le point où le procédé dépasse approximativement 2 % de la variation totale et d'y associer le temps de délai. Ensuite, on reprend l'opération en localisant, cette fois, le point où le procédé atteint 63,2 % de la variation totale.

La différence entre les deux points localisés symbolise la constante de temps *tau*. Se référer à la Figure IV-2 pour de bien visualiser les paramètres ainsi trouvés. On utilise la valeur à 2 % car elle permet d'isoler facilement le temps de délai. Aussi, il est aisé de situer graphiquement une telle donnée.

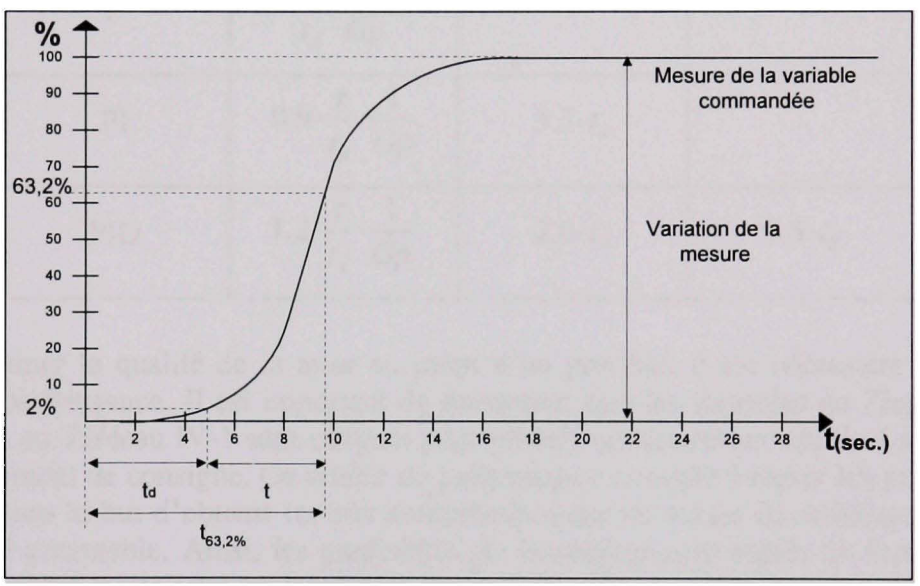


Figure IV-2 ***Mesure des caractéristiques du procédé.***

La fonction de transfert de système peut être approchée par un modèle d'ordre 1 affecté par un retard pur :

$$FT = G_{(s)} = \frac{Gp \cdot e^{-td \cdot s}}{1 + \tau s} \qquad \text{(IV.2)}$$

Les informations extraites du test de la réponse indicielle permettent de procéder à la mise au point du contrôleur classique de type standard. En effet, il suffit de consulter le tableau suivant afin de connaître les formules à utiliser selon le mode de régulation préconisé.

Tableau IV-1
Formules de réglage des paramètres d'un contrôleur standard

Mode du contrôleur	K_p	T_i	T_D
P	$\frac{\tau}{t_d}\cdot\frac{1}{Gp}$	-	-
PI	$0.9\cdot\frac{\tau}{t_d}\cdot\frac{1}{Gp}$	$3.3\cdot t_d$	-
PID	$1.2\cdot\frac{\tau}{t_d}\cdot\frac{1}{Gp}$	$2.0\cdot t_d$	$0.5\cdot t_d$

Pour déterminer la qualité de la mise au point d'un procédé, il est nécessaire d'avoir des critères de performance. Il est important de remarquer que les formules de Ziegler-Nichols représentées au Tableau IV-1 sont conçues pour obtenir un amortissement de 4 à 1 à la suite d'un changement de consigne. Ce critère de performance consiste à régler les paramètres du contrôleur dans le but d'obtenir un bon compromis entre un temps de stabilisation court et une stabilité acceptable. Ainsi, les paramètres du contrôleur sont réglés de façon à obtenir une erreur qui décroisse avec un facteur de quatre à un, lors des premiers dépassements de la consigne par la mesure.

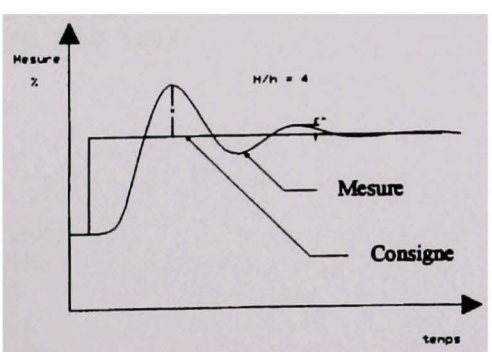


Figure IV-3 Exemple d'un décroissement 4 à 1.

ANNEXE V

Régulateur flou de type PI développé dans le logiciel Matlab®

```
[System]
Name='PID_Flou_25R'
Type='sugeno'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=25
AndMethod='min'
OrMethod='max'
ImpMethod='prod'
AggMethod='sum'
DefuzzMethod='wtaver'

[Input1]
Name='Erreur'
Range=[-100 100]
NumMFs=5
MF1='NG':'trapmf',[-100 -100 -66 -33]
MF2='NP':'trapmf',[-66 -33 -33 0]
MF3='ZO':'trapmf',[-33 0 0 33]
MF4='PP':'trapmf',[0 33 33 66]
MF5='PG':'trapmf',[33 66 100 100]

[Input2]
Name='delta_erreur'
Range=[-100 100]
NumMFs=5
MF1='NG':'trapmf',[-100 -100 -66 -33]
MF2='NP':'trapmf',[-66 -33 -33 0]
MF3='ZO':'trapmf',[-33 0 0 33]
MF4='PP':'trapmf',[0 33 33 66]
MF5='PG':'trapmf',[33 66 100 100]

[Output1]
Name='Power'
Range=[0 1]
NumMFs=5
MF1='NG':'constant',[-100]
MF2='NP':'constant',[-50]
MF3='ZO':'constant',[0]
MF4='PP':'constant',[50]
MF5='PG':'constant',[100]

[Rules]
1 5, 3 (1) : 1
1 4, 4 (1) : 1
1 3, 5 (1) : 1
```

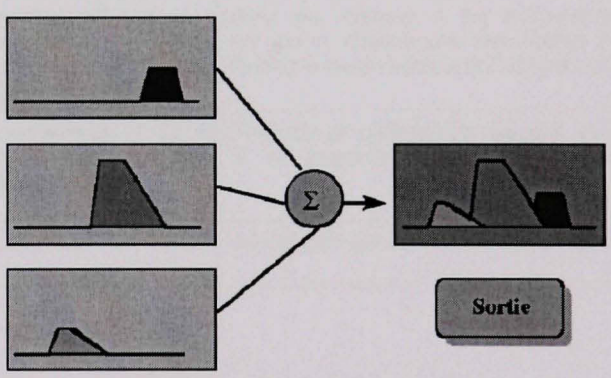
```
1 2, 5 (1) : 1
1 1, 5 (1) : 1
2 5, 2 (1) : 1
2 4, 3 (1) : 1
2 3, 4 (1) : 1
2 2, 5 (1) : 1
2 1, 5 (1) : 1
3 5, 1 (1) : 1
3 4, 2 (1) : 1
3 3, 3 (1) : 1
3 2, 4 (1) : 1
3 1, 5 (1) : 1
4 5, 1 (1) : 1
4 4, 1 (1) : 1
4 3, 2 (1) : 1
4 2, 3 (1) : 1
4 1, 4 (1) : 1
5 5, 1 (1) : 1
5 4, 1 (1) : 1
5 3, 1 (1) : 1
5 2, 2 (1) : 1
5 1, 3 (1) : 1
```

ANNEXE VI

Guide de démarrage « Débuter avec *Fuzzy Kernel* »

FUZZY KERNEL

Logiciel d'enseignement et d'application de la logique floue



10/7/2008

Débuter avec Fuzzy Kernel

Claude Théorêt, ing.

Version 2.1.0

FUZZY KERNEL

PRÉSENTATION

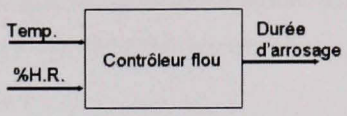
Ce document présente les étapes de base à suivre pour l'élaboration d'un système à logique floue à l'aide de Fuzzy Kernel.

Le logiciel FUZZY KERNEL est un logiciel d'aide à l'apprentissage et d'application de la logique floue. Il permet de simuler et effectuer la mise en œuvre des systèmes à logique floue. Il est aussi un complément aux logiciels RSLogix 5000 permettant de réaliser des traitements en logique floue afin d'optimiser la commande des procédés à partir des automates programmables industriels ControlLogix. Ce logiciel se présente sous la forme d'un simulateur autonome intégrer ainsi d'une fonction qui s'intègre dans tout programme RSLogix 5000. Il inclut les outils de mise en œuvre et de mise au point.

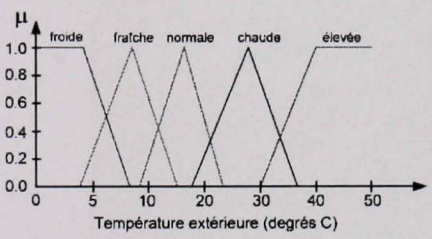
Convention du document:	
Ce symbole :	Indique :
	Le texte qui suit ce symbole fait référence à des informations complémentaires. Le texte qui suit ce symbole peut vous fournir de précieux conseils en vue de faciliter la compréhension et l'utilisation du logiciel Fuzzy Kernel.
	**Avertissement** !!! Le texte qui suit ce symbole fait référence à un avertissement important. Il est conseillé de lire attentivement l'avertissement.
Remarque : Si le bouton de la souris n'est pas précisé dans le texte, cela signifie que vous devez cliquer sur le bouton gauche de la souris.	

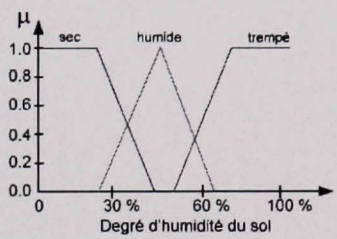
1 CRÉER UN SYSTÈME FLOU

Lors de cette section, vous allez crée un système flou qui va vous permettre de contrôler un système d'arrosage selon la température extérieure. Le système a deux entrées, soit la mesure de la température extérieure et la mesure de l'humidité du sol. Le système possède un signal de sortie qui contrôle la durée d'arrosage. Nous aurons 15 règles d'inférence qui font varier la durée du temps d'arrosage.

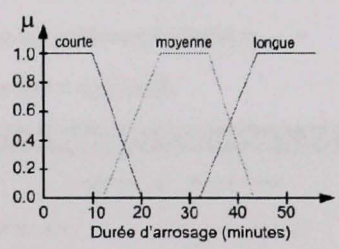


Les entrées :





La sortie :



Les règles d'inférence :

Température

Humidité du sol	froide	fraîche	normale	moyenne	élevée
trempé	courte	courte	courte	courte	courte
humide	courte	moyenne	moyenne	moyenne	moyenne
sec	longue	longue	longue	longue	longue

Durée d'arrosage

1.1 Création d'un nouveau projet

Cette section pratique vous montre comment créer votre premier projet à logique floue.

1. Démarrez Fuzzy Kernel.

La fenêtre « ***Gestionnaire des données floues*** » apparaît.

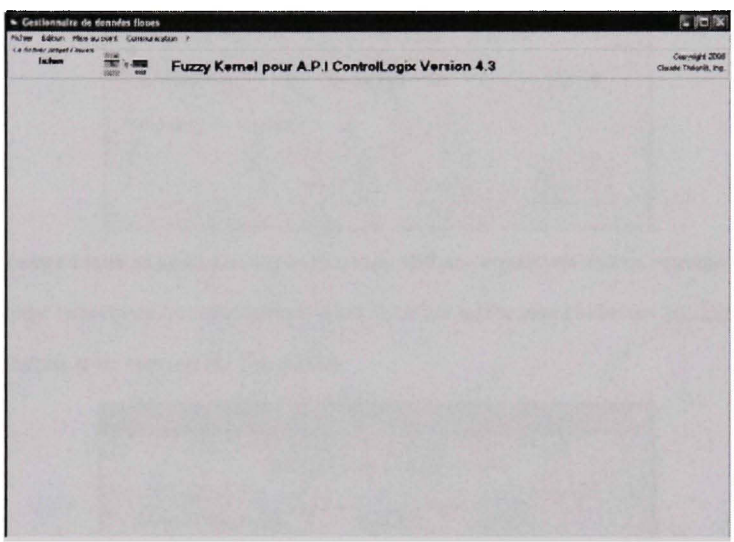


FIGURE 1 GESTIONNAIRE DE DONNÉES FLOUES

2. Dans le menu « ***Fichier*** », cliquez sur « ***Nouveau Projet*** ».

La fenêtre « ***Génération d'un nouveau projet*** » apparaît.

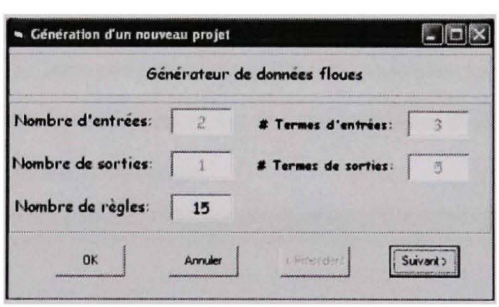


FIGURE 2 FENÊTRE GÉNÉRATION D'UN NOUVEAU PROJET – GÉNÉRATEUR DE DONNÉES FLOUES

3. Dans le champ « ***Nombre de règles*** », entrez 15 et cliquez sur le bouton « ***Suivant*** ».

L'entrée #1 sera associée à la mesure de la température.

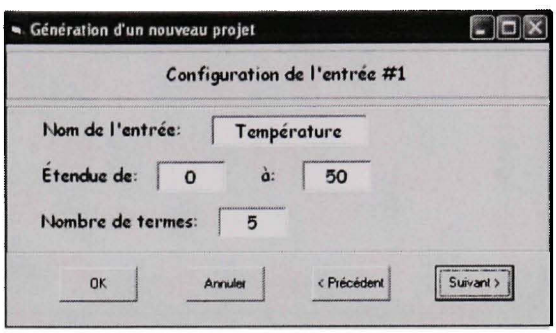


FIGURE 3 FENÊTRE GÉNÉRATION D'UN NOUVEAU PROJET – CONFIGURATION DE L'ENTRÉE #1

4. Vérifiez que vos sélections correspondent bien à celles indiquées ci-dessus et cliquez sur « ***Suivant*** ».

L'entrée #2 sera associée à la mesure de l'humidité.

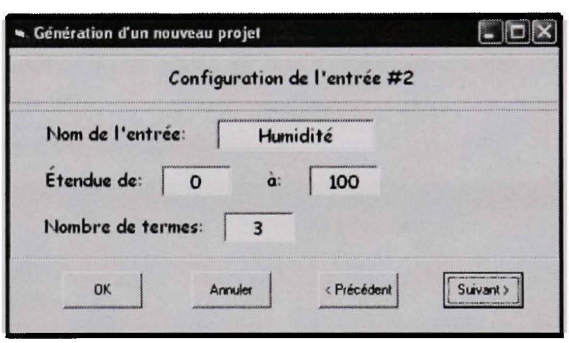


FIGURE 4 FENÊTRE GÉNÉRATION D'UN NOUVEAU PROJET – CONFIGURATION DE L'ENTRÉE #2

5. Vérifiez que vos sélections correspondent bien à celles indiquées ci-dessus et cliquez sur « ***Suivant*** ».

La sortie #1 sera associée à la durée d'arrosage.

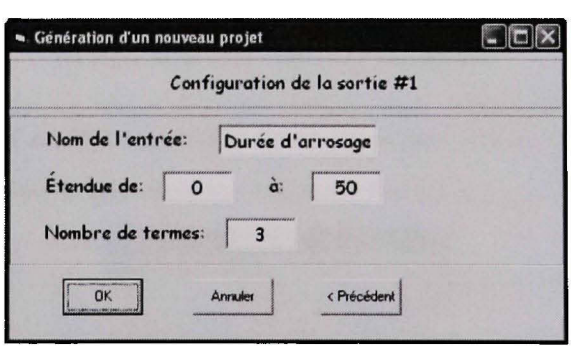


FIGURE 5 FENÊTRE GÉNÉRATION D'UN NOUVEAU PROJET – CONFIGURATION DE LA SORTIE #1

6. Vérifiez que vos sélections correspondent bien à celles indiquées ci-dessus et cliquez sur « ***OK*** ».

Les fenêtres des données floues des entrées et de la sortie apparaissent.

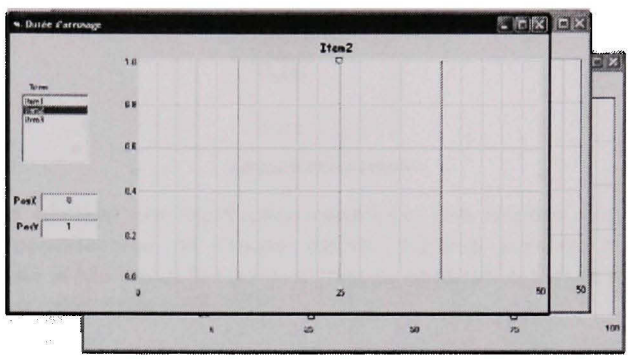


FIGURE 6 FENÊTRES DES DONNÉES FLOUES DES ENTRÉES ET DE LA SORTIE

1.2 Définir les entrées et la sortie

Cette section pratique vous montre comment définir les entrées et la sortie d'un projet à logique floue. Une entrée est caractérisée par son domaine, et sa partition floue, c'est-à-dire les sous-ensembles flous (SEF) qui la composent. Le domaine de variation d'une entrée est fixe et il est compris dans l'intervalle [0,1]. Les noms des SEF est par défaut item1, item2, etc.

1. Sélectionnez la fenêtre d'entrée « ***Température***».

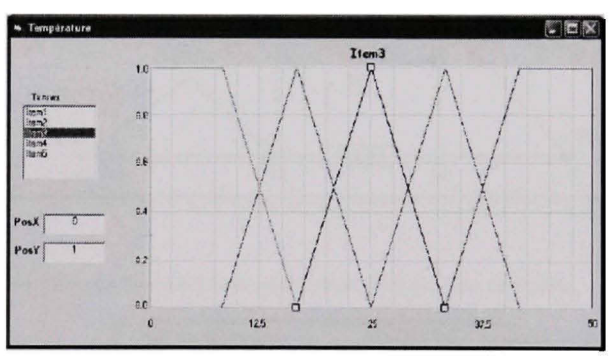
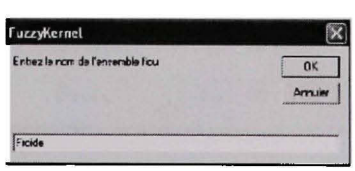


FIGURE 7 FENÊTRE DE L'ENTRÉE FLOUE « TEMPÉRATURE»

Pour la clarté des règles d'inférence, les noms des SEF sont importants, car ils apparaissent dans la base des règles. Pour modifier le nom d'un SEF, il faut double cliquer sur son nom dans la fenêtre « ***Termes*** ».

2. Double cliquez sur l'item1 et saisissez le nouveau nom « ***Froide*** »



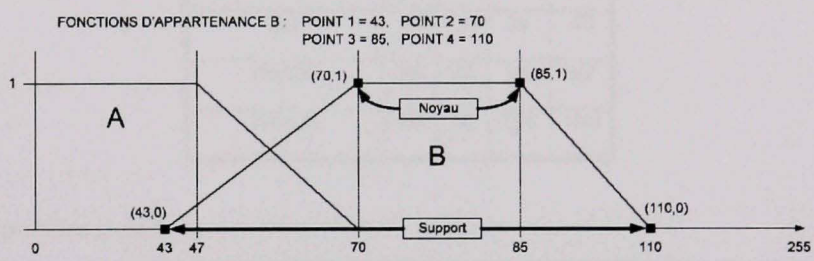
3. Modifiez les autres noms des SEF selon la figure suivante.

FIGURE 8 NOMS DES SEF

Pour chaque SEF, il faut attribuer une signification numérique. Une fonction d'appartenance est créée en spécifiant le degré d'appartenance de chaque donnée d'entrée possible. Pour modifier la fonction d'appartenance, cliquer sur le SEF désiré (les points d'ancrage apparaissent) et déplacer les points d'ancrage selon les limites désirées à l'aide de la souris.

Caractéristiques d'un ensemble flou d'entrée

Dans Fuzzy Kernel, un ensemble flou d'entrée est caractérisé par un support et un noyau. Les SEF sont de type trapézoïdal, par conséquent, ils utilisent deux points d'ancrage pour le support et deux points d'ancrage pour le noyau. Voici un exemple d'un SEF B avec la partition suivante : 43, 70, 85 et 110.



4. Modifiez les SEF de l'entrée température pour avoir la partition suivante.

SEF du domaine Température	ANCRAGE			
	1	2	3	4
Froide	0	0	5	8
Fraîche	4	9	9	14
Normale	9	16	16	23
Chaude	18	27	27	36
Élevée	30	40	50	50

Vous obtenez la partition ci-dessous.

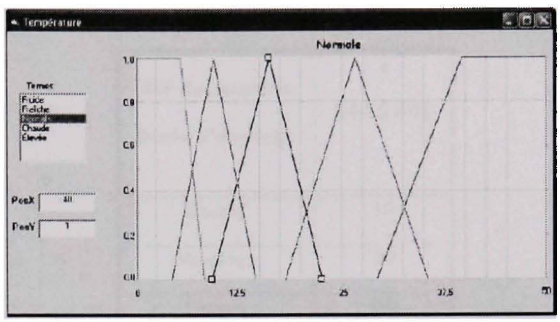


FIGURE 9 PARTITION DES SEF POUR L'ENTRÉE « TEMPÉRATURE »

5. Sélectionnez la fenêtre de l'entrée floue « ***Humidité*** ». Si la fenêtre n'est pas ouverte, il est possible de l'ouvrir à l'aide de l'option « ***Les ensembles d'entrée*** » du menu « ***Édition*** ».

6. Modifiez les noms et les fonctions d'appartenance des SEF de l'entrée « ***humidité*** » pour avoir la partition suivante.

SEF du domaine Humidité	ANCRAGE			
	1	2	3	4
Sec	0	0	28	40
Humide	28	45	45	62
Trempé	50	70	100	100

Vous obtenez la partition ci-dessous.

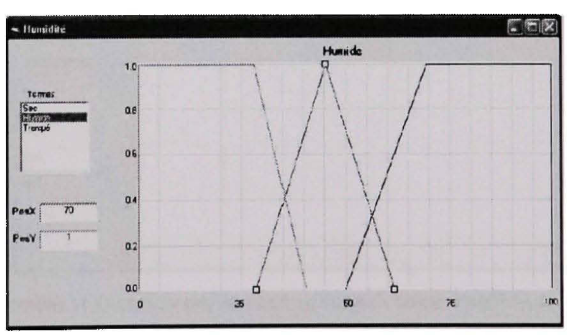


FIGURE 10 PARTITION DES SEF POUR L'ENTRÉE « HUMIDITÉ »

7. Sélectionnez la fenêtre de sortie floue « ***Durée d'arrosage*** ». Si la fenêtre n'est pas ouverte, il est possible de l'ouvrir à l'aide de l'option « ***Les ensembles de sortie*** » du menu « ***Édition*** ».

8. Modifiez les noms et les fonctions d'appartenance des SEF de la sortie « ***Durée d'arrosage*** » pour avoir la partition suivante.

SEF du domaine Durée d'arrosage	ANCRAGE
Courte	10
Moyenne	30
Longue	50

Caractéristiques d'un ensemble flou de sortie

Dans Fuzzy Kernel, un ensemble flou de sortie est caractérisé par un « singleton ». Le calcul de la sortie précise (non floue) utilise, alors, l'opérateur de Sugeno.

$$\hat{y}_i = \frac{\sum_{j=1}^{m} W_j \cdot C_j}{\sum_{j=1}^{m} W_j}$$

Vous obtenez la partition ci-dessous.

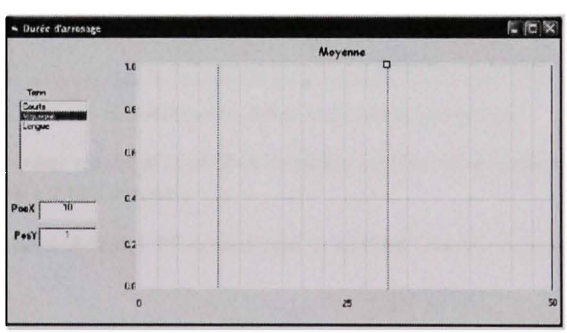


FIGURE 11 PARTITION DES SEF POUR LA SORTIE « DURÉE D'ARROSAGE »

1.3 Définir la base des règles du module d'inférence

Cette section pratique vous montre comment définir la base des règles du module d'inférence.

 La base des règles d'inférence

L'inférence floue est le processus d'élaboration des relations qui existent entre les variables d'entrées (exprimées comme variables linguistiques) et la variable de sortie (également exprimée comme variable linguistique). Ces relations fournissent ensuite *la base des règles* (**BR**) à partir de laquelle les décisions peuvent être prises. Le processus de l'inférence floue implique tous les SEF des entrées et de la sortie.

1. Pour créer les règles, cliquez sur l'option « ***Les règles floues*** » du menu « ***Édition*** ».

La fenêtre « ***Configuration des règles floues*** » apparaît.

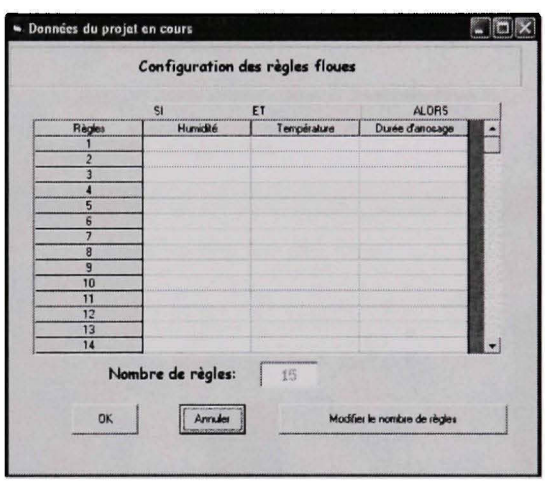


FIGURE 12 FENÊTRE DE CONFIGURATION DES RÈGLRS FLOUES

La première étape consiste à créer des règles dont la syntaxe est très stricte malgré que les termes utilisés paraissent naturels. La syntaxe est la suivante :

Si antécédent 1 **ET** antécédent 2 **ALORS** conséquence 1

Où:

- Antécédent: variable d'entrée,
- Conséquence: variable de sortie.

Nous allons insérer la première règle :

Si Humidité = Sec **ET** Température = Froide **ALORS** Durée d'arrosage = Longue

2. Cliquez sur le champ de la règle 1 sous le domaine « ***Humidité*** ».

La liste des SEF du domaine « ***Humidité*** » apparaît.

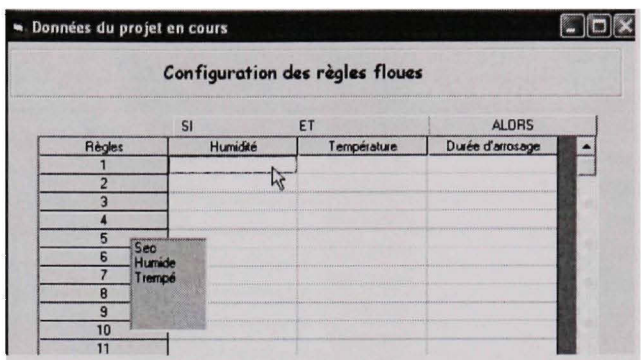


FIGURE 13 LISTE DES SEF DU DOMAINE « HUMIDITÉ »

3. Dans la liste, cliquer sur le nom du SEF « ***Sec*** ».
4. Cliquez sur le champ de la règle 1 sous le domaine « ***Température*** ».

La liste des SEF du domaine « ***Température*** » apparaît.

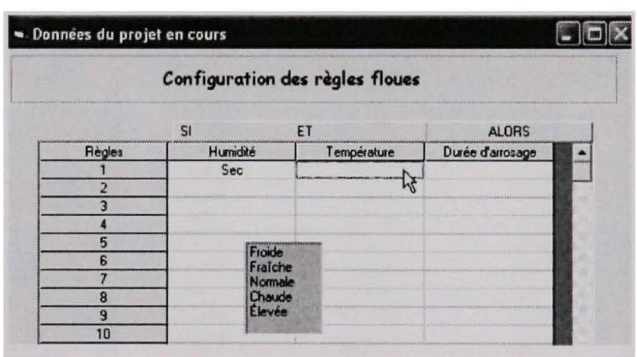


FIGURE 14 LISTE DES SEF DU DOMAINE « TEMPÉRATURE »

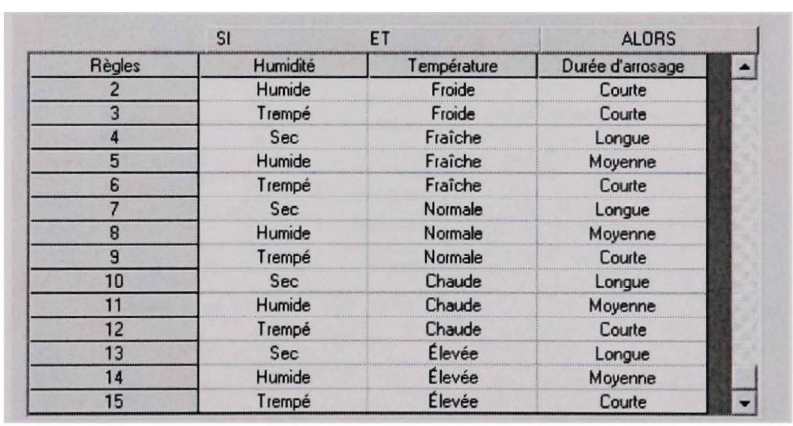
5. Dans la liste, cliquez sur le nom du SEF « ***Froide*** ».
6. Cliquez sur le champ de la règle 1 sous le domaine « ***Durée d'arrosage*** » et cliquez sur le nom du SEF « ***Longue*** ».

La première règle est programmée.

7. Créez les règles 2 à 15 selon le tableau suivant.

Règle no	SI	Humidité	ET	Température	ALORS	Durée d'arrosage
1		Sec		Froide		Longue
2		Humide		Froide		Courte
3		Trempé		Froide		Courte
4		Sec		Fraîche		Longue
5		Humide		Fraîche		Moyenne
6		Trempé		Fraîche		Courte
7		Sec		Normale		Longue
8		Humide		Normale		Moyenne
9		Trempé		Normale		Courte
10		Sec		Chaude		Longue
11		Humide		Chaude		Moyenne
12		Trempé		Chaude		Courte
13		Sec		Élevée		Longue
14		Humide		Élevée		Moyenne
15		Trempé		Élevée		Courte

Vous obtenez la base des règles ci-dessous.



	SI	ET	ALORS
Règles	Humidité	Température	Durée d'arrosage
2	Humide	Froide	Courte
3	Trempé	Froide	Courte
4	Sec	Fraîche	Longue
5	Humide	Fraîche	Moyenne
6	Trempé	Fraîche	Courte
7	Sec	Normale	Longue
8	Humide	Normale	Moyenne
9	Trempé	Normale	Courte
10	Sec	Chaude	Longue
11	Humide	Chaude	Moyenne
12	Trempé	Chaude	Courte
13	Sec	Élevée	Longue
14	Humide	Élevée	Moyenne
15	Trempé	Élevée	Courte

FIGURE 15 BASE DES RÈGLES FLOUES

8. Pour sauvegarder le système flou, cliquez sur l'option « ***Sauvegarder un projet*** » du menu « ***Fichier*** ».

1.4 Mise au point en mode local

L'option mise au point en mode local permet d'explorer le fonctionnement ou d'effectuer une simulation du système flou. Cette section pratique vous montre comment effectuer la simulation du système flou en mode local.

Pour simuler le projet, vous devez compiler les données floues

1. Pour compiler, cliquez sur l'option « ***Compilation des données floues*** » du menu « ***Communication*** ».
2. Un message du système indiquant que la compilation a été complétée, cliquez sur « ***OK*** ».

3. Cliquer sur l'option « ***Mode Local*** » du menu « ***Mise au point*** » pour ouvrir la fenêtre de mise au point en mode local.

La fenêtre « ***Mise au point en mode local*** » apparaît.

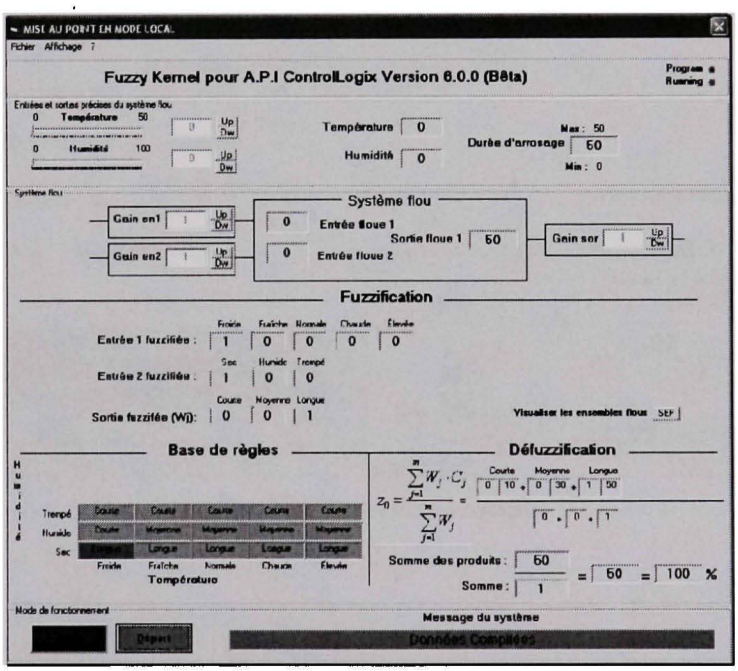


FIGURE 16 FENÊTRE DE MISE AU POINT EN MODE LOCAL

4. Cliquez sur le bouton « ***Départ*** » pour démarrer le système flou.

Dans la section « Entrées et sorties précises du système flou », les valeurs des variables d'entrée sont saisies manuellement en déplaçant un curseur ou en cliquant sur les icônes « Up » et « Dw » pour augmenter ou diminuer la valeur. La valeur précise (non flou) de la sortie est affichée. La section « Calcul d'inférence » affiche plusieurs informations intermédiaires permettant de comprendre les étapes du raisonnement flou. Le système affiche le degré d'appartenance au SEF de chaque entrée, ainsi que la valeur d'inférence pour la sortie. La sortie précise, avec une défuzzification de type Sugeno et une agrégation de type Min-Max, s'obtient par une simple moyenne des conclusions des règles, pondérées par la somme des degrés de vérité des SEF de la sortie. Ces valeurs sont indiquées dans les champs « Somme des Produits » et Somme » respectivement.

5. Modifiez les valeurs d'entrée pour avoir une température de 20 degrés et une humidité de 30%.

Le système affiche une durée d'arrosage de 47 minutes, ainsi que toutes les informations intermédiaires du raisonnement flou (voir Figure 17).

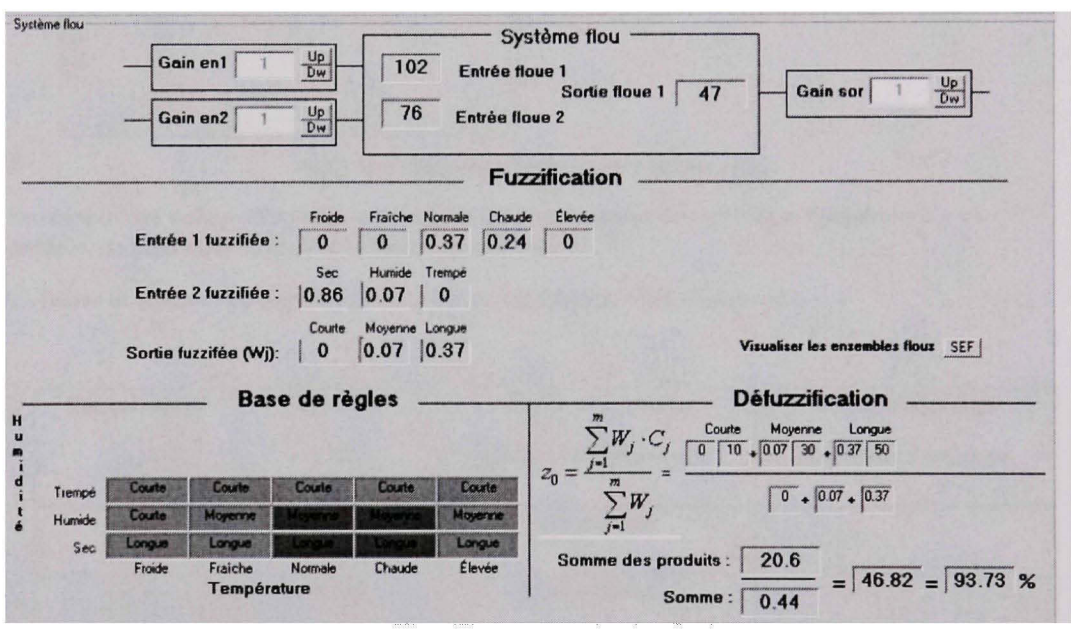


FIGURE 17 RÉSULTAT DU CALCUL D'INFÉRENCE

6. Pour visualiser les SEF des entrées et de la sortie en temps réel, cliquez sur le bouton « ***SEF*** » dans la section « ***Fuzzification*** ».

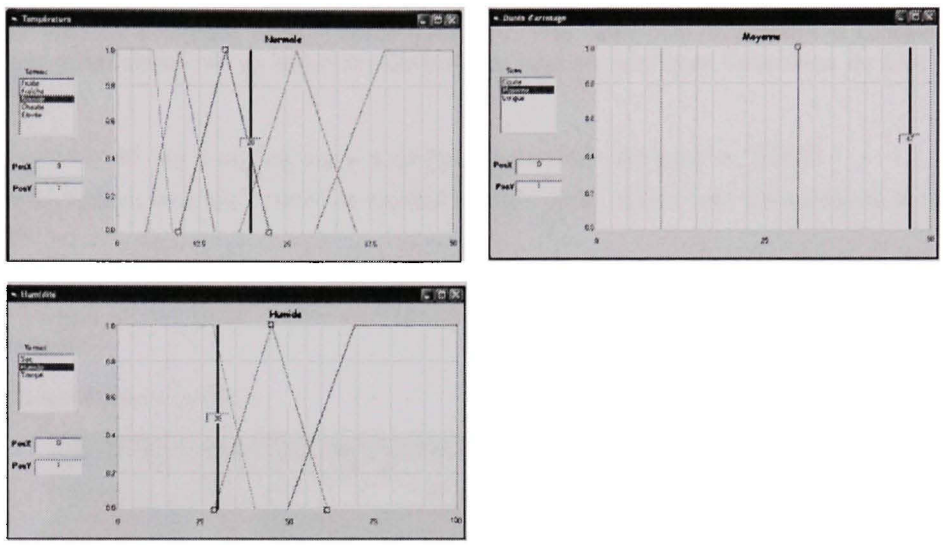


FIGURE 18 SEF DES ENTRÉES ET DE LA SORTIE EN TEMPS RÉEL

Pour l'ensemble des valeurs d'entrées comprissent dans le domaine des entrées « ***Température*** » et « ***Humidité*** », on peut obtenir la durée d'arrosage en minute.

7. Tester et valider votre système flou avec les données du tableau suivant :

Température (°C)	**Humidité (%)**	**Durée d'arrosage Résultats Théoriques**	**Durée d'arrosage Résultats Pratiques**
20 °C	30 %	47 minutes	
20 °C	50 %	29 minutes	
36 °C	75 %	10 minutes	
12 °C	40 %	30 minutes	

2 CRÉER UN BLOC DE FONCTION À LOGIQUE FLOUE

Lors de cette section, vous allez crée un projet dans un automate ControlLogix qui va vous permettre d'utiliser le bloc de fonction à logique floue, appelé « FLFB »[1]. Vous allez ainsi apprendre à concevoir, créer et charger des programmes sur un automate ControlLogix tout en contrôlant l'exécution du bloc de fonction « FLFB ».

2.1 Lancement du logiciel de programmation RsLogix 5000

Lors de cette section, vous allez lancer le logiciel RSLogix 5000 qui va vous permettre de programmer un processeur.

1. Vous devez dans l'environnement Windows XP démarrer RSLogix 5000 en choisissant l'option « RSLogix 5000» du menu «Démarrer/Programmes/Rockwell Software/RSLogix 5000 Enterprise Series».

L'écran RSLogix5000 apparaît.

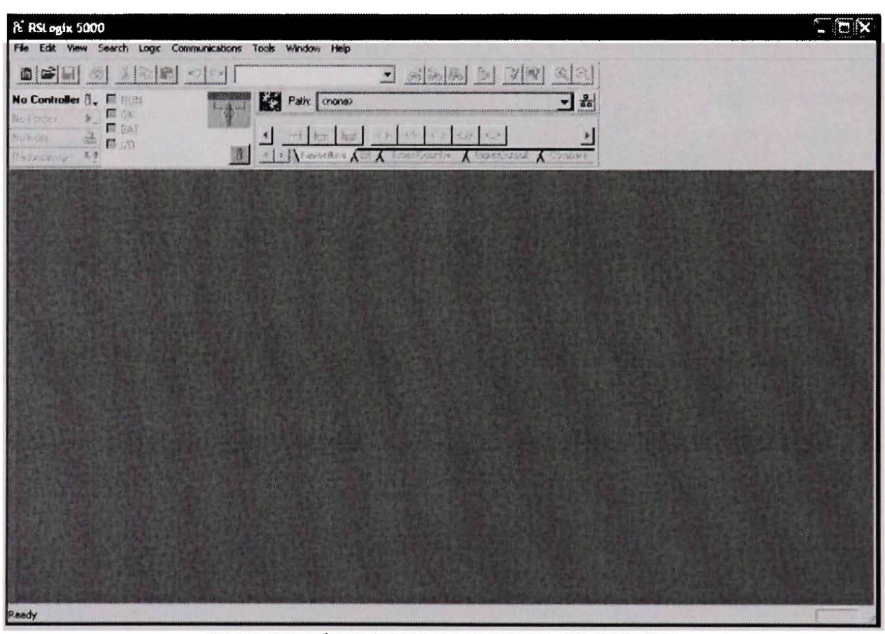


FIGURE 19 FENÊTRE PRINCIPALE DU LOGICIEL RSLOGIX 5000

[1] FLFB, signifie « Fuzzy Logic Function Bloc » en anglais

Cette section pratique vous montre comment créer votre premier projet d'automate.

2. Dans le menu ***File*** (Fichier), sélectionnez ***New*** (Nouveau).
3. La boîte de dialogue New Controller (Nouvel automate) apparaît.
4. Vérifiez que vos sélections correspondent bien à celles indiquées ci-dessous et cliquez sur ***OK***. (Note : la sélection peut variée selon la configuration logicielle et matérielle)

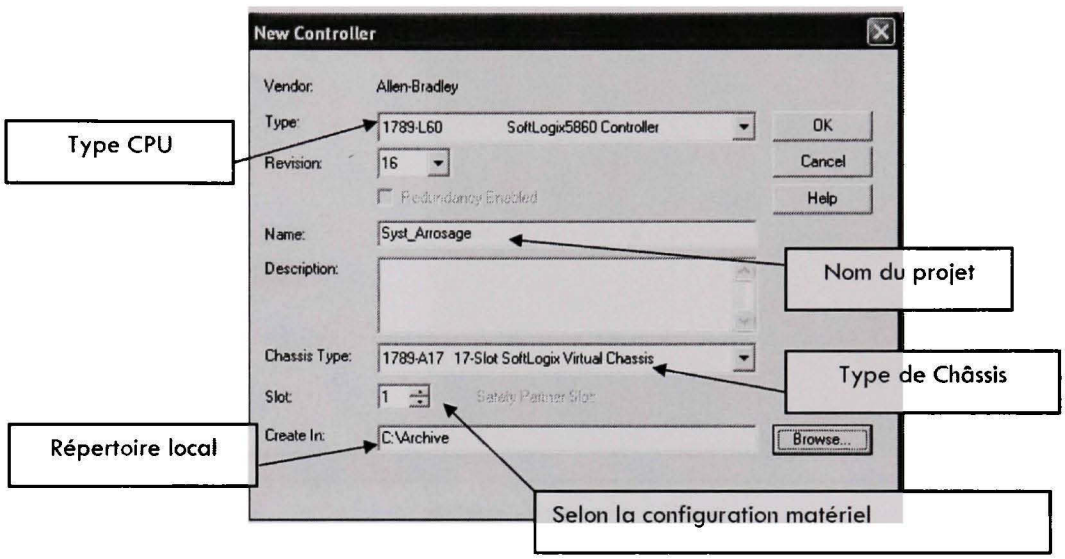


FIGURE 20 FENÊTRE PROPRIÉTÉS DU CONTRÔLEUR

L'arborescence de l'automate apparaît à gauche dans la fenêtre RSLogix5000 avec un dossier appelé Controller Syst_Arrosage. Vous avez à présent créé votre premier projet d'automate. A ce stade, aucune E/S, aucune base de données de points, ni aucune logique n'est associée à l'automate.

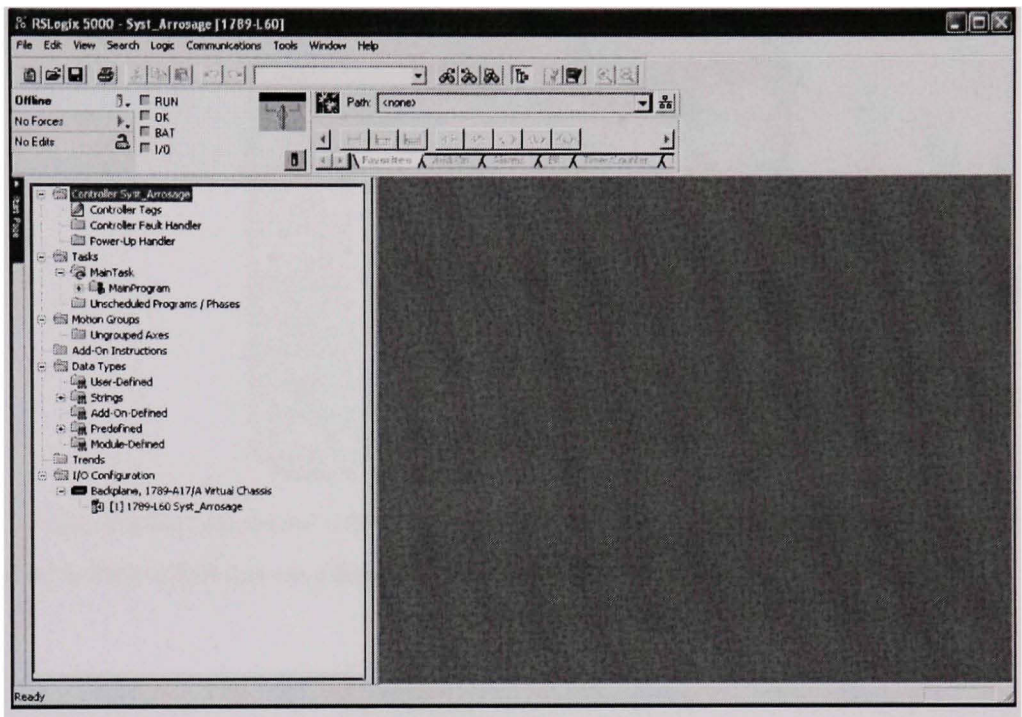
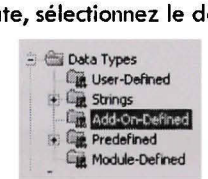


FIGURE 21 ARBORESCENCE DU PROJET

2.2 Ajout du bloc de fonction FLFB

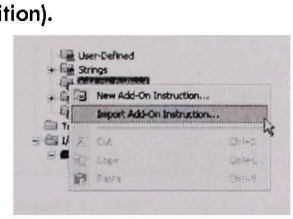
Lors de cette session, vous devez importer le bloc de fonction « FLFB » dans le projet. Les automates Logix peuvent être programmés à l'aide des graphes de blocs fonctionnels (Function Block), des graphes de fonctionnement séquentiel (Sequential Function Charts) et du texte structuré (Structured Text), vous pouvez sélectionner la langue du programme la mieux adaptée à votre application.

1. Dans l'arborescence de l'automate, sélectionnez le dossier ***Add-On-Defined*** (ajout de définition).



2. Une fois développé, le dossier ***MainProgram*** apparaît comme ci-dessous.

3. Cliquez avec le bouton droit de la souris sur le dossier ***Add-On-Defined*** et sélectionnez ***Import Add-On-Defined*** (Importer nouvelle définition).



La fenêtre ***Import Add-On-Defined*** (Importer Nouvelle définition) apparaît.

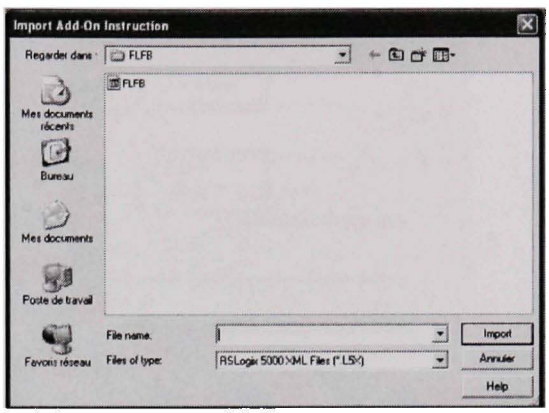
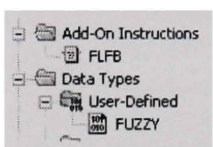


FIGURE 22 FENÊTRE D'IMPORTATION DE NOUVELLE DÉFINITION

4. Saisissez le bloc de fonction « ***FLFB*** » et cliquez sur ***Import*** pour importer le bloc de fonction.

Le bloc de fonction « ***FLFB*** » et son « ***Data Types*** » sont maintenant importés.



2.3 Créer une tâche périodique pour le bloc « FLFB »

Lors de cette session, vous devez créer une tâche périodique dans le projet. Vous devez, obligatoirement, exécuter le bloc de fonction « ***FLFB*** » à l'intérieur d'une tâche périodique.

1. Cliquez avec le bouton droit de la souris sur le répertoire ***Task*** (tâche) et sélectionnez ***New Task*** (nouvelle tâche).

La fenêtre ***New Task*** (nouvelle tâche) apparaît.

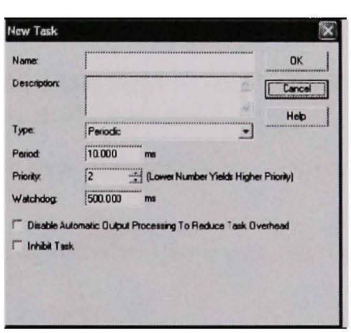


FIGURE 23 FENÊTRE NOUVELLE TÂCHE

2. Configurez la nouvelle tâche périodique pour qu'elle corresponde aux données ci-dessous. Lorsque vous avez terminé, cliquez sur ***OK***.

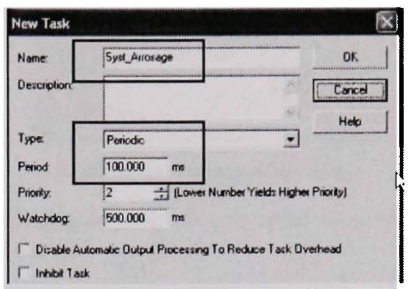


FIGURE 24 FENÊTRE NOUVELLE TÂCHE

3. Il n'y a pour l'instant aucun programme défini dans la tâche périodique ***Syst_Arrosage***. Pour ajouter un programme, cliquez avec le bouton droit de la souris sur ***Syst_Arrosage*** et sélectionnez ***New Program*** (nouveau programme).

4. Pour le nom du programme entrez « ***Controle*** », puis cliquez sur ***OK***.

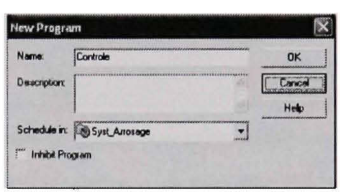
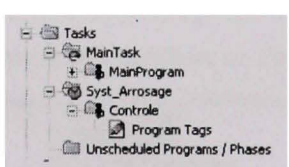


FIGURE 25 FENÊTRE NOUVEAU PROGRAMME

5. Cliquez sur le signe + pour dérouler le programme ***Controle*** que vous venez d'ajouter à la tâche ***Syst_Arrosage***. Vous remarquerez qu'il n'y a encore aucun sous-programme attribué dans ce programme.



Ensuite, nous devons ajouter une routine « ***Syst_Flou1*** » dans la quelle le bloc de fonction « ***FLFB*** » sera exécuter. N'oubliez pas qu'un programme doit toujours avoir un sous-programme principal.

6. Pour ajouter une routine, cliquez avec le bouton droit de la souris sur ***Controle*** et sélectionnez ***New Routine*** (nouvelle routine).

7. Configurez la nouvelle routine pour qu'elle corresponde aux données ci-dessous. Lorsque vous avez terminé, cliquez sur ***OK***.

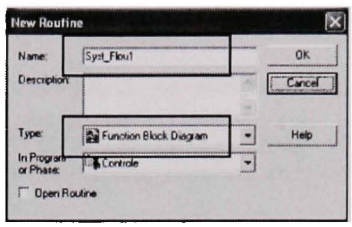


FIGURE 26 FENÊTRE NOUVELLE ROUTINE

8. Pour assigner la routine au programme, cliquez avec le bouton droit sur le dossier ***Controle*** et sélectionnez ***Properties*** (Propriétés).

9. Cliquez sur l'onglet ***Configuration*** de la fenêtre ***Program Properties.***

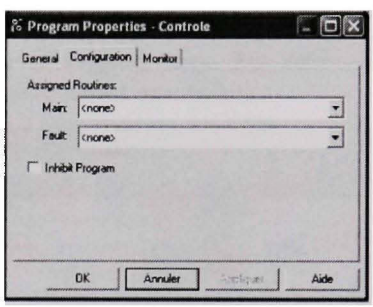


FIGURE 27 FENÊTRE PROPRIÉTÉS DU PROGRAMME

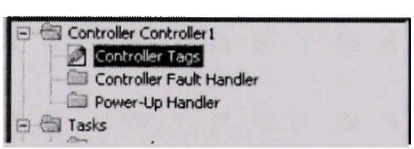
10. Dans le champ ***Main*** (Principale), sélectionnez « ***Syst_Flou1*** » puis cliquez sur ***OK***.

L'icône apparaît à côté de la routine « ***Syst_Flou1*** ».

2.4 Création de points (TAG) pour le programme

Au cours de cette section pratique, vous allez créer les points requis pour le programme.

1. Dans l'arborescence de l'automate, double-cliquez sur ***Controller Tags*** (Points d'automate).



2. La fenêtre tag Monitor/Editor (Contrôleur/éditeur de points) apparaît. Veuillez noter, en bas à gauche de la fenêtre, les deux points intitulés ***Monitor Tags*** (Visualiser/Contrôler les points) et ***Edit Tags*** (Éditer les points) comme ci-après.

Onglets ***Monitor Tags*** et ***Edit Tags*** (Contrôler les points et Éditer les points)

Lorsque l'onglet « ***Monitor Tags*** » (Contrôler les points) est sélectionné, la ou les valeur(s) réelle(s) des points s'affiche(nt). Par ex., si vous visualisez un bouton de saisie, le logiciel affiche le point activé ou désactivé du bouton.

Lorsque l'onglet « ***Edit Tags*** » (Éditer les points) est sélectionné, il est possible de créer de NOUVEAUX points et de modifier les propriétés actuelles des points.

Si vous avez des difficultés à créer ou à modifier les paramètres des points, vérifiez que l'onglet « ***Edit Tags*** » (Éditer les points) est bien sélectionné.

3. Créez les points (Tags) associées au programme ***Syst_Arrosage*** qui correspond au tableau suivant. Respectez les noms, les types ainsi que les descriptions.

Scope: Syst_Arrosage Show... Show All

Name	Alias For	Base Tag	Data Type	Style	Description
+ Temperature			DINT	Decimal	
+ Humidite			DINT	Decimal	
+ Duree_Arrosage			DINT	Decimal	
+ Data_Flou			FUZZY		
Auto			BOOL	Decimal	

2.5 Ajout du bloc « FLFB »

Lors de cette session, vous allez ajouter un bloc de fonction « FLFB » dans la routine de la tâche périodique.

1. Double cliquez avec le bouton gauche de la souris sur le répertoire ***Syst_Flou1***.

La fenêtre d'édition ***Contrôle – Syst_Flou1*** apparaît.

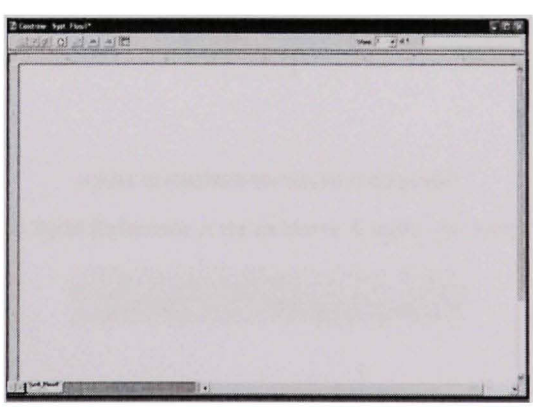
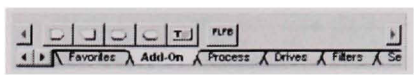


FIGURE 28 FENÊTRE D'ÉDITION DE PROGRAMME

2. Cliquez sur l'onglet « ***Add-On*** » de la barre d'outils des instructions.



3. Cliquez sur l'instruction ***FLFB*** .

L'instruction ***FLFB*** apparaît dans la fenêtre d'édition ***Contrôle – Syst_Flou1***.

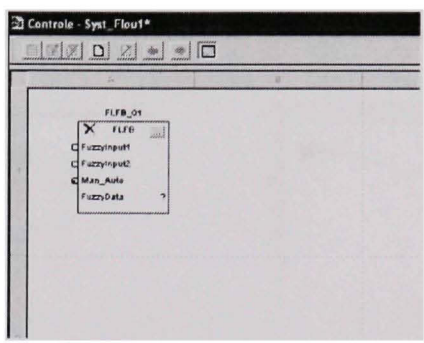


FIGURE 29 FENÊTRE D'ÉDITION DE PROGRAMME

4. Déplacez le bloc de fonction vers le bas et la droite.

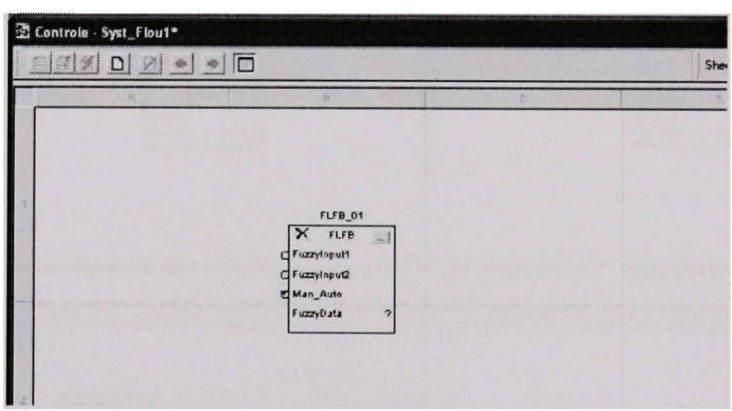
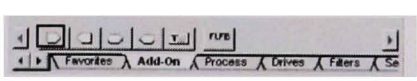
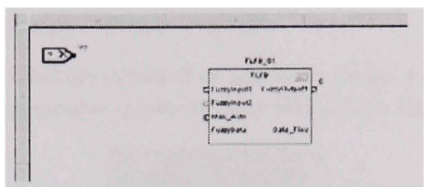


FIGURE 30 FENÊTRE D'ÉDITION DE PROGRAMME

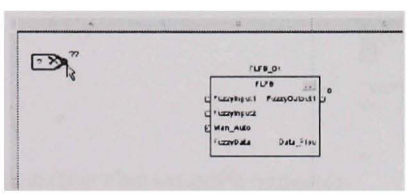
5. Cliquez sur l'instruction « ***Input Reference*** » de la barre d'outils des instructions.

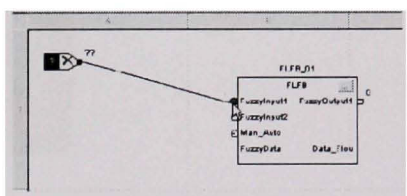


L'instruction « ***Input Reference*** » apparaît dans fenêtre d'édition ***Contrôle – Syst_Flou1***.

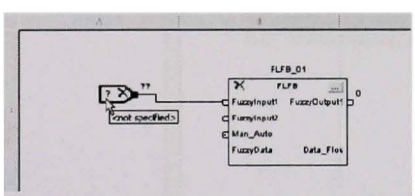


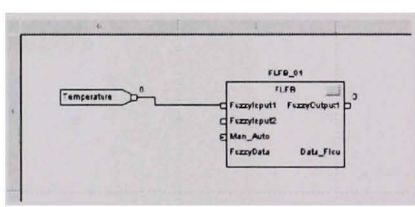
6. À l'aide de la souris, cliquez sur l'instruction « ***Input Reference*** » et effectuez le branchement à l'entrée « ***FuzzyInput1*** » du Bloc de fonction « ***FLFB*** ».



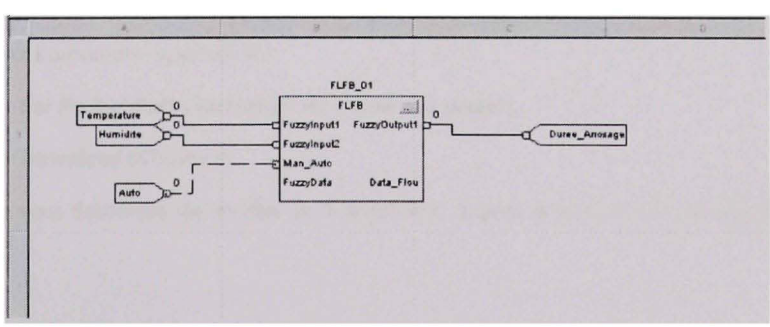


7. Double cliquez sur l'instruction « ***Input Reference*** » et associée cette entrée avec le point « ***Temperature*** ».





8. Effectuez les branchements des entrées et de la sortie qui respectent la figure suivante.



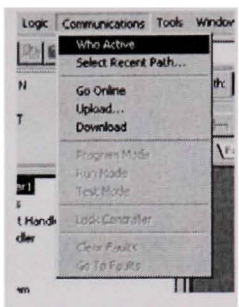
9. Dans le champ « ***FuzzyData*** », sélectionnez le point « ***Data_Flou*** ».

10. Sauvegardez le programme en cliquant sur l'icône Save (Sauvegarder) de la barre d'outils.

2.6 Chargement du projet sur l'automate

La présente section vous explique comment charger le projet.

1. Assurez-vous que RSLogix 5000 est démarré et que votre projet « ***Syst_Arrosage*** » est ouvert. Dans le menu ***Communications*** (Communications), sélectionnez ***Who Active*** (Qui est actif).



L'écran ***Who Active*** (Qui est actif) apparaît.

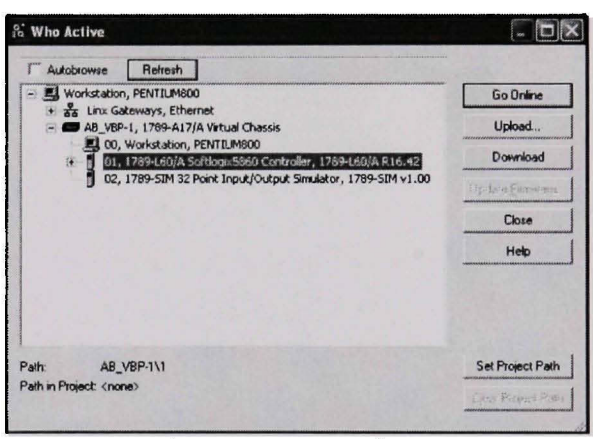
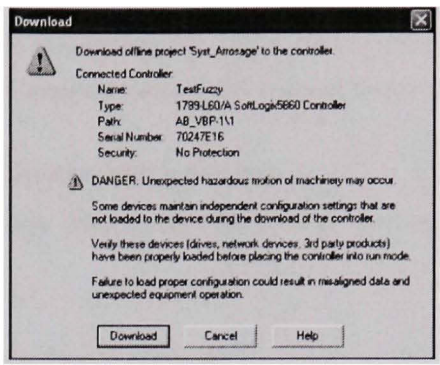


FIGURE 31 FENÊTRE FURETEUR DE CONTRÔLEUR (WHO ACTIVE)

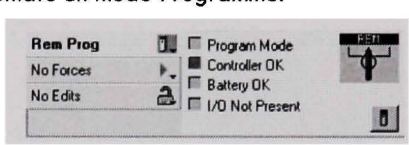
2. Sélectionnez l'automate approprié.
3. Cliquez sur ***Set Project Path*** (Activation du chemin du projet).
4. Cliquez sur ***Download*** (Charger).
5. Le système vous demande de vérifier le chargement, cliquez à nouveau sur ***Download***.



Le chargement du projet débute.

> Si votre automate était en mode RUN avant le chargement, il est possible que le système vous demande de rétablir le mode RUN. Si tel est le cas, cliquez sur ***YES***.

6. A ce stade, l'automate est en ligne et les DEL d'état imitent les voyants de votre automate. Voici ci-dessous un exemple d'automate en mode Programme.



2.7 Chargement des données floues sur l'automate

La présente section vous explique comment charger les données floues sur l'automate à l'aide du logiciel Fuzzy Kernel.

1. Assurez-vous que l'automate est en mode « ***Rem Prog*** ».
2. Créez un lien DDE/OPC dans RSLinx vers l'automate et son programme et donnez lui le nom de « ***Syst_Arrosage*** ».
3. Démarrez Fuzzy Kernel.

La fenêtre « ***Gestionnaire des données floues*** » apparaît.

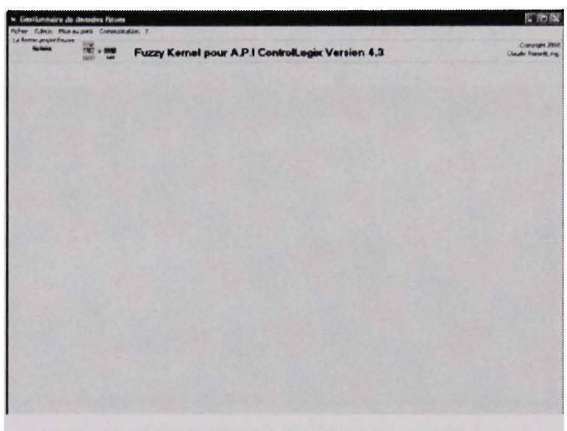


FIGURE 32 GESTIONNAIRE DE DONNÉES FLOUES

4. Dans le menu « ***Fichier*** », cliquez sur « ***Télécharger un Projet*** ».

La fenêtre « ***Ouvrir*** » apparaît.

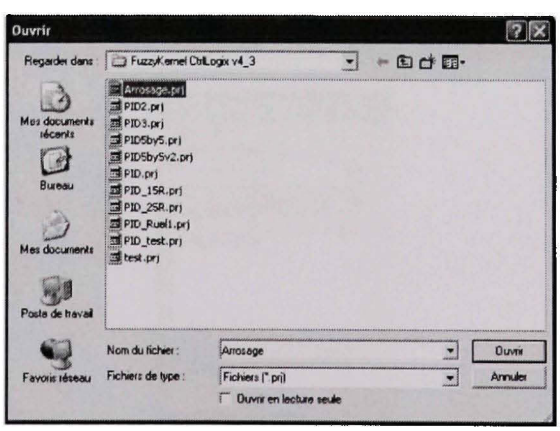


FIGURE 33 FENÊTRE OUVRIR UN PROJET

5. Sélectionnez le projet du chapitre 1 « ***Arrosage.prj*** » et cliquez sur « ***Ouvrir*** ».
6. Compilez en cliquant sur l'option « ***Compilation des données floues*** » du menu « ***Communication*** ».
7. Un message du système indiquant que la compilation a été complétée, cliquez sur « ***OK*** ».

8. Cliquez sur l'option « ***Téléchargement*** » du menu « ***Communication*** » pour ouvrir la fenêtre de téléchargement.

La fenêtre «***Téléchargement des données floues***» apparaît.

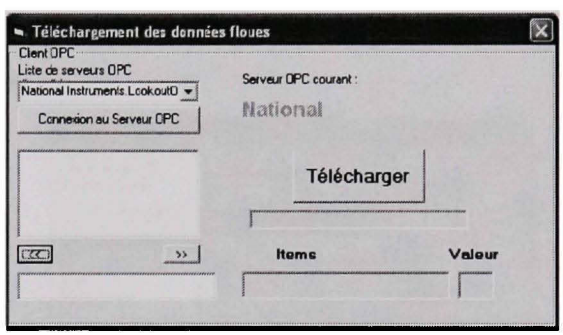


FIGURE 34 FENÊTRE DE TÉLÉCHARGEMENT DES DONNÉES FLOUES

9. Dans le champ « ***List of OPC Servers available*** », cliquez sur le menu déroulant et sélectionnez le serveur « ***RSLinx OPC Server*** ».
10. Cliquez sur le bouton « ***Connect to OPC Server*** ».

Le client OPC se branche sur le serveur OPC RSLinx et donne la liste des liens existants.

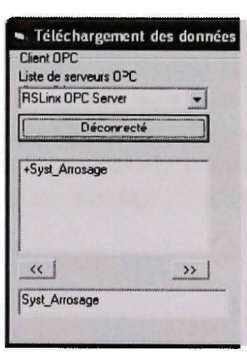


FIGURE 35 FENÊTRE DE TÉLÉCHARGEMENT DES DONNÉES FLOUES

11. Développez l'arborescence du lien « ***Syst_Arrosage*** » en double cliquant sur le + ou en sélectionnant le lien et en cliquant sur le bouton >>.

12. Développez l'arborescence du lien « ***Online*** » en double cliquant sur le +.

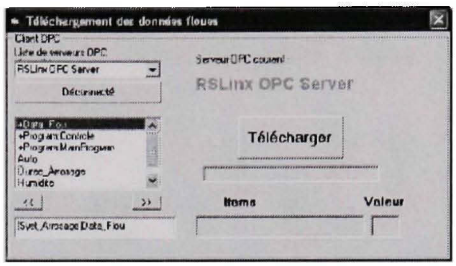


FIGURE 36 FENÊTRE DE TÉLÉCHARGEMENT DES DONNÉES FLOUES

13. Sélectionnez le point « ***Data_Flou*** » et cliquez sur le bouton « ***Télécharger*** ».

Le téléchargement s'effectue.

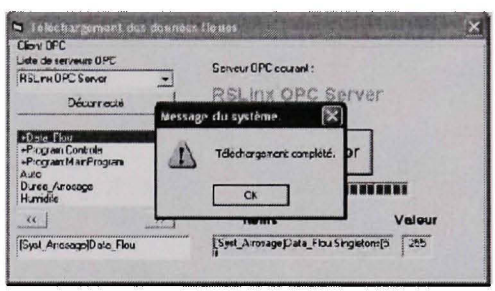


FIGURE 37 FENÊTRE DE TÉLÉCHARGEMENT DES DONNÉES FLOUES

14. Cliquez sur le bouton « ***Ok*** ».

Les données du système flou sont maintenant dans l'automate programmable. Il ne reste plus qu'à placer l'automate en mode « ***Run*** » et tester le bloc de fonction « ***FLFB*** » en simulant les valeurs d'entrée des variables « ***Température*** » et « ***Humidité*** ». Pour l'ensemble des valeurs d'entrées comprissent dans le domaine des entrées « ***Température*** » et « ***Humidité*** », on peut obtenir la durée d'arrosage en minute. Vous pouvez comparer les valeurs obtenues avec l'option mise au point en mode local du logiciel Fuzzy Kernel et celles obtenues avec le bloc de fonction « ***FLFB*** » de l'automate ControlLogix. Attention !!! Le bloc de fonction « ***FLFB*** » est muni d'une entrée « ***Man_Auto*** » qui doit être à un niveau logique 1 pour fonctionner.

ANNEXE VII

Régulateur flou de type PI développé dans le logiciel *FUDGE*®

```
*      FUzzy Development and Generation Environment (FUDGE) Version V1.02
*      User readable knowledge base file
*      Alex DeCastro & Jason Spielman & John Dumas
*      Copyright Motorola 1994

      INPUTS
Input #1
        INPUT NAME          MIN          MAX                UNITS
            Erreur         -100          100             pourcent

MEMBERSHIP FUNCTION          P1           P2           P3           P4
                NG         -100         -100          -66          -33
                NP          -66          -33          -33            0
                ZO          -33            0            0           33
                PP            0           33           33           66
                PG           33           66          100          100
                 ~            0            0            0            0
                 ~            0            0            0            0
                 ~            0            0            0            0
Input #2
        INPUT NAME          MIN          MAX                UNITS
      Delta_erreur         -100          100       d_erreur_sec

MEMBERSHIP FUNCTION          P1           P2           P3           P4
                NG         -100         -100          -66          -33
                NP          -66          -33          -33            0
                ZO          -33            0            0           33
                PP            0           33           33           66
                PG           33           66          100          100
                 ~            0            0            0            0
                 ~            0            0            0            0
                 ~            0            0            0            0
     OUTPUTS
Output #0
       OUTPUT NAME          MIN          MAX                UNITS
          Commande         -100          100              %/sec

MEMBERSHIP FUNCTION          P1
                NG         -100
                NP          -50
                ZO            0
                PP           50
                PG          100
                 ~            0
                 ~            0
                 ~            0
     RULES
```

```
Rule #1
IF Erreur IS NG
AND Delta_erreur IS PG
THEN Commande IS ZO

Rule #2
IF Erreur IS NG
AND Delta_erreur IS PP
THEN Commande IS PP

Rule #3
IF Erreur IS NG
AND Delta_erreur IS ZO
THEN Commande IS PG

Rule #4
IF Erreur IS NG
AND Delta_erreur IS NP
THEN Commande IS PG

Rule #5
IF Erreur IS NG
AND Delta_erreur IS NG
THEN Commande IS PG

Rule #6
IF Erreur IS NP
AND Delta_erreur IS PG
THEN Commande IS NP

Rule #7
IF Erreur IS NP
AND Delta_erreur IS PP
THEN Commande IS ZO

Rule #8
IF Erreur IS NP
AND Delta_erreur IS ZO
THEN Commande IS PP

Rule #9
IF Erreur IS NP
AND Delta_erreur IS NP
THEN Commande IS PG

Rule #10
IF Erreur IS PG
AND Delta_erreur IS NG
THEN Commande IS ZO

Rule #11
IF Erreur IS ZO
AND Delta_erreur IS PG
THEN Commande IS NG
```

```
Rule #12
IF Erreur IS ZO
AND Delta_erreur IS PP
THEN Commande IS NP

Rule #13
IF Erreur IS ZO
AND Delta_erreur IS ZO
THEN Commande IS ZO

Rule #14
IF Erreur IS ZO
AND Delta_erreur IS NP
THEN Commande IS PP

Rule #15
IF Erreur IS ZO
AND Delta_erreur IS NG
THEN Commande IS PG

Rule #16
IF Erreur IS PP
AND Delta_erreur IS PG
THEN Commande IS NG

Rule #17
IF Erreur IS PP
AND Delta_erreur IS PP
THEN Commande IS NG

Rule #18
IF Erreur IS PP
AND Delta_erreur IS ZO
THEN Commande IS NP

Rule #19
IF Erreur IS PP
AND Delta_erreur IS NP
THEN Commande IS ZO

Rule #20
IF Erreur IS PP
AND Delta_erreur IS NG
THEN Commande IS PP

Rule #21
IF Erreur IS PG
AND Delta_erreur IS PG
THEN Commande IS NG

Rule #22
IF Erreur IS PG
AND Delta_erreur IS PP
THEN Commande IS NG
```

```
Rule #23
IF Erreur IS PG
AND Delta_erreur IS ZO
THEN Commande IS NG

Rule #24
IF Erreur IS PG
AND Delta_erreur IS NP
THEN Commande IS NP

Rule #25
IF Erreur IS NP
AND Delta_erreur IS NG
THEN Commande IS PG
```

BIBLIOGRAPHIE

Abdelnour, G. M., C. H. Chang, F. H. Huang et J. Y. Cheung. 1991. « Design of a fuzzy controller using input and output mapping factors ». *IEEE Transactions on Systems, Man and Cybernetics,* vol. 21, n° 5, p. 952-60.

AFIA - Association Française d'Intelligence Artificielle. 2003. « Conférences AFIA ». In *Rencontres Francophones sur la Logique Floue et ses Applications : la Logique Floue et ses Applications*. En ligne. <http://afia.lri.fr/node.php?lang=fr&node=565>. Consulté le 24 juillet 2008.

Akcayol, M. A., C. Elmas, O. A. Erdem et M. Kurt. 2004. « An educational tool for fuzzy logic controller and classical controllers ». *Computer Applications in Engineering Education,* vol. 12, n° 2, p. 126-35.

Al-Zahrani, Khalid Mousa. 2005. « Fuzzy Takagi-Sugeno and LMS based control techniques ». M.S., Saudi Arabia, King Fahd University of Petroleum and Minerals (Saudi Arabia).

Alata, Mohanad. 2001. « Control of nonlinear systems using Sugeno fuzzy approximators ». Ph.D., Canada, Concordia University (Canada).

Alsaleh, Hassan. 1998. « Utilisation classique et extensions des fonctionnalités des processeurs flous ». Ph.D., France, Université de Savoie (France), 176 p.

Belanger, Melanie. 2005. « Commande de l'orientation d'un satellite basee sur la logique floue ». M.Sc.A., Canada, Universite de Sherbrooke (Canada).

Borne, Pierre. 1998. *Introduction à la commande floue*. Coll. « Collection Sciences et technologies 6 ». Paris: Technip, vi, 102 p.

Bouchon-Meunier, B., et Lotfi Asker Zadeh. 1995. *La logique floue et ses applications*. Coll. « Vie artificielle ». Paris: Addison-Wesley France, xii, 257 p.

Bouchon-Meunier, Bernadette, Laurent Foulloy et Mohammed Ramdani. 1998. *Logique floue : exercices corrigés et exemples d'applications*. Toulouse: Cépaduès, 200 p.

Bühler, Hansruedi. 1994. *Réglage par logique floue*. Coll. « Collection Électricité ». Lausanne: Presses polytechniques et universitaires romandes, 181 p.

Carvajal, James, Guanrong Chen et Haluk Ogmen. 2000. « Fuzzy PID controller: Design, performance evaluation, and stability analysis ». *Information Sciences,* vol. 123, n° 3, p. 249-270.

Chevrie, François., Guély, François. 1998. *La logique floue*. Coll. « Cahier technique », CT191. France: Groupe Schneider, 28 p.

Cohen, L.J. 1973. « A note on inductive logic ». *The J. of Philosophy,* vol. LXX, p. 27-40.

Dadone, Paolo. 2001. « Design optimization of fuzzy logic systems ». Ph.D., United States -- Virginia, Virginia Polytechnic Institute and State University.

Devis du ministère de l'éducation. 2002. *Technologie de l'électronique industrielle : Rapport d'analyse de situation de travail.* Coll. « Ministère de l'éducation », 2000-00-0572. Gourvernement du Québec, 123 p.

Dobritoiu, Manuela Carmen. 2003. « Stabilisateur de réseaux électriques à base de logique floue ». M.Sc.A., Canada, École Polytechnique, Montréal (Canada).

Dubois, Didier, Henri Prade et Ronald R. Yager. 1993. *Readings in fuzzy sets for intelligent systems*. San Mateo, Calif.: Morgan Kaufman Publishers., xii, 916 p.

Dunlop, J. A., K. J. Burnham, A. Edge, D. J. G. James et W. J. Wilber. 1996. « Educational developments in the teaching of fuzzy logic ». In *Proceedings of the 1996 IEE Colloquium on Fuzzy Logic Controllers in Practice, Nov 15 1996* (Stevenage, Engl). Vol. 200, p. 8-1. IEE.

Eksioglu, Kamil Murat. 2000. « Modélisation de la dépendance contextuelle des concepts flous : La structure SFC ». Ph.D., Canada, Université de Sherbrooke (Canada), 99 p.

Elqaq, Eyad. 2005. « Fuzzy observer and performance measures for model-based fuzzy closed loop control system ». Ph.D., United States -- Illinois, University of Illinois at Chicago.

Feng, Zheng, Wang Qing-Guo, Lee Tong Heng et Huang Xiaogang. 2001. « Robust PI controller design for nonlinear systems via fuzzy modeling approach ». *Systems, Man and Cybernetics, Part A, IEEE Transactions on,* vol. 31, n° 6, p. 666-675.

Flaus, Jean-Marie. 1994. *La régulation industrielle : régulateurs PID, prédictifs et flous*. Coll. « Traité des nouvelles technologies ». Paris: Hermès, 348 p.

Foulloy, L., R. Boukezzoula et S. Galichet. 2006. « An educational tool for fuzzy control ». *IEEE Transactions on Fuzzy Systems,* vol. 14, n° 2, p. 217-21.

Girault, Isabelle et d'Ham, C. 2005. « Analyse de changements induits par la technologie dans des TP de sciences expérimentales effectués avec un laboratoire distant. ». *Skholê,* vol. Hors-série, n° 2, p. 55-63.

Godoy, Emmanuel. 2007. *Régulation industrielle*. Coll. « Technique et ingénierie. Série EEA ». Paris: Dunod : L' Usine nouvelle, xi, 525 p.

Guillaume, Serge. 2005. « Représentation des connaissances et systèmes d'inférence floue ». Mémoire d'Habilitation à Diriger des recherches, Toulouse, France, Université Paul Sabatier, 36 p.

Harinath, Eranda. 2007. « Design and tuning of fuzzy PID controllers for multivariable process systems ». M.Eng., Canada, Memorial University of Newfoundland (Canada).

Hernandez, Gustavo. 2007. « Generalisation de solutions de controle flou pour usage industriel ». M.Sc.A., Canada, Universite de Sherbrooke (Canada).

Jihong, Lee. 1993. « On methods for improving performance of PI-type fuzzy logic controllers ». *Fuzzy Systems, IEEE Transactions on,* vol. 1, n° 4, p. 298-301.

Juang, Yau-Tarng, Yun-Tien Chang et Chih-Peng Huang. 2008. « Design of fuzzy PID controllers using modified triangular membership functions ». *Information Sciences,* vol. 178, n° 5, p. 1325-1333.

Kampé de Fériet, J. , Forte, B., Benvennuti, P. 1969. « Forme générale de l'opération de composition continue d'une information ». *Académie Sci.* (Paris). p. 529-534.

Keller, J. P. 2000. « Teaching PID and fuzzy controllers with LabVIEW ». *International Journal of Engineering Education,* vol. 16, n° 3, p. 202-11.

Kermiche, Salah, et Hadj Ahmed Abbassi. 2008. « Fuzzy PID controller parameters optimized by genetic algorithms ». In *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications, ICTTA, Apr 7-11 2008* (Damascus, Syrian Arab Republic). p. 4530018. Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States.

Khan, A. A., et N. Rapal. 2006. « Fuzzy PID Controller: Design, Tuning and Comparison with Conventional PID Controller ». In *Engineering of Intelligent Systems, 2006 IEEE International Conference on.* p. 1-6.

Kosko, Bart. 1992. *Neural networks and fuzzy systems : a dynamical systems approach to machine intelligence*. Englewood Cliffs, N.J.: Prentice-Hall, xxvii, 449 p.

Kristiansson, B., et B. Lennartson. 2006. « Evaluation and simple tuning of PID controllers with high-frequency robustness ». *Journal of Process Control,* vol. 16, n° 2, p. 91-102.

Kuo, Benjamin C. 1991. *Automatic control systems*, 6th. Englewood Cliffs, N.J., : Prentice-Hall, xviii, 760 p.

Lam, H. K., M. Narimani, J. C. Y. Lai et F. H. F. Leung. 2008. « Stability analysis of T-S fuzzy-model-based control systems using fuzzy Lyapunov function ». In *2008 IEEE 16th International Conference on Fuzzy Systems (FUZZ-IEEE), 1-6 June 2008* (Piscataway, NJ). p. 931-8. IEEE.

Lee, C. C. 1990. « Fuzzy logic in control systems: fuzzy logic controller. I ». *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 20, n° 2, p. 404-418.

Li, Yun, Kiam Heong Ang et Gregory C. Y. Chong. 2006. « Patents, software, and hardware for PID control: An overview and analysis of the current art ». *IEEE Control Systems Magazine,* vol. 26, n° 1, p. 42-54.

Longchamp, Roland. 2007. *Commande numérique de systèmes dynamiques : cours d'automatique*, 2e éd. entièrement rev. et augm. Lausanne, Suisse: Presses polytechniques et universitaires romandes, xvii, 765 p.

Mann, G. K. I., Hu Bao-Gang et R. G. Gosine. 1999. « Analysis of direct action fuzzy PID controller structures ». *Systems, Man, and Cybernetics, Part B, IEEE Transactions on,* vol. 29, n° 3, p. 371-388.

Mansouri, Badr. 2005. « Contribution à la synthèse de lois de commandes en poursuite de trajectoire pour les systèmes de type Takagi Sugeno incertains ». Ph.D., France, Université de Reims (France), 158 p.

Misir, D., H. A. Malki et Chen Guanrong. 1996. « Design and analysis of a fuzzy proportional-integral-derivative controller ». *Fuzzy Sets and Systems,* vol. 79, n° 3, p. 297-314.

Mizumoto, M. 1995. « Realization of PID controls by fuzzy control methods ». *Fuzzy Sets and Systems,* vol. 70, n° 2-3, p. 171-82.

Mohan, B. M., et A. Sinha. 2006. « The simplest fuzzy PID controllers: mathematical models and stability analysis ». *Soft Computing,* vol. 10, n° 10, p. 961-75.

Nakoula, Yassar. 1997. « Apprentissage des modèles linguistiques flous, par jeu de règles pondérées ». Ph.D., France, Université de Savoie (France), 158 p.

Oyagi, Masayuki. 1991. *La logique floue en clair*. vol. 5; Issue 4. Tokyo (Japon): Omron Corporation, 75 p.

Pivonka, P. 2002. « Comparative analysis of fuzzy PI/PD/PID controller based on classical PID controller approach ». In *2002 IEEE World Congress on Computational*

Intelligence. 2002 IEEE International Conference on Fuzzy Systems. FUZZ-IEEE'02, 12-17 May 2002 (Honolulu, HI). Vol. vol.1, p. 541-6. IEEE.

Rattan, K. S., et D. Van Cleave. 2000. « Design and implementation of a reduced rule fuzzy logic PID controller ». In *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American*. p. 465-469.

Rescher, N. 1976. *Plausible Reasoning : An Introduction to the theory and pratice of plausibilistic inference*. Assen/Amsterdam: Van Gorcum.

Ruel, Michel. 1994. *Introduction à la logique floue : logique à valeurs multiples*. Lévis: Cégep de Lévis-Lauzon., 163 p.

Sangalli, Arturo. 2001. *Éloge du flou : aux frontières des mathématiques et de l'intelligence artificielle*. Montréal: Presses de l'Université de Montréal, 206 p.

Schweizer, B., Sklar, A. 1983. *Probalilistic Metric Spaces*. Amsterdam, North-Holland.

Shafer, G. 1976. *Mathematical Theory of Evidence*. Princeton, N.J.: Princeton University Press.

Shilkret, N. 1971. « Maxitive measure and integration ». *Indag. Math.*, p. 109-116.

Skoczowski, S., S. Domek, K. Pietrusewicz et B. Broel-Plater. 2005. « A method for improving the robustness of PID control ». *Industrial Electronics, IEEE Transactions on,* vol. 52, n° 6, p. 1669-1676.

Takagi, T., T. Yamaguchi et M. Sugeno. 1992. « Conceptual fuzzy sets ». In *Fuzzy Engineering Toward Human Friendly Systems, 13-15 Nov. 1991* (Yokohama, Japan). p. 261-72. IOS Press.

Tang, K. S., Man Kim Fung, Chen Guanrong et S. Kwong. 2001. « An optimal fuzzy PID controller ». *IEEE Transactions on Industrial Electronics,* vol. 48, n° 4, p. 757-65.

Tipsuwanpom, Runghimmawan T., T. Runghimmawan, S. Intajag et V. Krongratana. 2004. « Fuzzy logic PID controller based on FPGA for process control ». In *Industrial Electronics, 2004 IEEE International Symposium on*. Vol. 2, p. 1495-1500 vol. 2.

Tong-Tong, Jean-Raphaël. 1995. *La logique floue*. Paris: Hermès, 160 p.

Uncu, Ozge. 2003. « Type 2 fuzzy system models with type 1 inference ». Ph.D., Canada, University of Toronto (Canada).

Yen, John, Reza Langari, Lotfi Asker Zadeh et IEEE Neural Networks Council. 1995. *Industrial applications of fuzzy logic and intelligent systems*. New York

Piscataway, N.J.: Institute of Electrical and Electronics Engineers
IEEE Press., xviii, 356 p.

Zadeh, L. A. 1978. « Fuzzy sets as a basis for a theory of possibility ». *Fuzzy Sets and Systems,* vol. 1, n° 1, p. 3-28.

Ziegler, J. G., et N. B. Nichols. 1942. « Optimum settings for automatic controllers ». *American Society of Mechanical Engineers -- Transactions,* vol. 64, n° 8, p. 759-765.