UNIVERSITE DE SHERBROOKE

Faculté de génie

Département de génie électrique et de génie informatique

# CONTRÔLE DE L'ÉQUILIBRE ET DE LA TRAJECTOIRE D'UNE BICYCLETTE TÉLÉCOMMANDÉE PAR LA LOGIQUE FLOUE

Mémoire de maîtrise ès sciences appliquées

Spécialité : génie électrique

Said BERRIAH

Sherbrooke (Québec), Canada Janvier 2000

## RÉSUMÉ

Les dernières années ont enregistrées un intérêt grandissant accordé aux systèmes de contrôle par la logique floue. Celle ci, a été introduite principalement pour imiter les stratégies de contrôle des opérateurs humains. Dans ce genre de cas, les connaissances et l'expérience des opérateurs humains sont utilisées et le contrôleur final peut performer aussi bien que le meilleur des opérateurs humains. Vu sous cet angle, le contrôle par la logique floue convient très bien dans les situations ou le processus est trop complexe pour être modélisable, ou le nombre de variables accessibles et mesurables est réduit, ou le système est fortement non linéaire. Le but de notre travail est de développer et de tester un contrôleur flou pour ce genre de systèmes.

Le système en question, est un prototype d'un modèle réduit d'une bicyclette télécommandée développé au département de génie électrique et de génie informatique de l'Université de Sherbrooke. La dynamique de ce système est telle que le maintien de l'équilibre et de la trajectoire est impossible sans supervision. C'est dans ce but, qu'un contrôleur basé sur la logique floue a été conçu. Ce contrôleur permet la supervision du pilotage humain de modèle réduit à travers une télécommande ou le lien série d'un PC.

Dans la première partie de ce mémoire, nous allons présentés les concepts de base de la logique floue. Dans la deuxième partie, nous présenterons les grandes lignes des systèmes de commandes basées sur la logique floue. La dernière partie, traite du contrôleur flou implanté pour le pilotage du modèle réduit de la bicyclette.

## REMERCIEMENT

Je tiens à exprimer ma gratitude à M. Gérard Lachiver, directeur du département de génie électrique et de génie informatique de la faculté de génie, qui a su faciliter le déroulement et l'orientation de ce travail par ces conseils avisés relatifs à ce texte.

Je tiens aussi à remercier M. Bruno Paillard, Professeur au département de génie électrique et génie informatique, pour son aide et ses conseils.

Merci aussi à M. Sidi Ouldmoha, étudiant au doctorat au département de génie électrique et génie informatique pour sa contribution au bon déroulement de ce travail, ainsi que toutes les personnes qui ont rendu possible la réalisation de ce mémoire par leurs conseils et critiques.

## TABLE DES MATIÈRES

## LISTE DES FIGURES

### CHAPITRE 1



### CHAPITRE 2



### CHAPITRE 3

# INTRODUCTION

Ce mémoire traite de l'application de la logique floue au contrôle et la supervision des systèmes automatiques. Dans la première partie du mémoire, nous présentons les notions de base des ensembles flous et du raisonnement flou. La deuxième partie est consacrée aux systèmes de commande basés sur la logique floue. Dans la troisième partie, nous présentons une application du contrôle flou à la supervision du pilotage d'un modèle réduit d'une bicyclette télécommandée.

Introduite en 1965 par L.A. Zadeh [14], professeur à l'Université de Californie à Berkeley, la théorie des ensembles flous est restée marginale pendant longtemps, n'intéressant qu'un cercle restreint d'adeptes parmi les chercheurs. Cette traversée du désert que connaissent la plupart des nouvelles théories (du fait que très souvent elles bousculent l'ordre établi) a durée au moins une dizaine d'années. Mais la théorie des ensembles flous va connaître au terme de cette période de gestation, un essor prodigieux dans la vague de l'intelligence artificielle mais surtout grâce à des applications pratiques industrielles dans le domaine de la commande et du contrôle des automatismes et des procédés.

Utilisée dans de nombreuses disciplines mathématiques (intégration, théorie de la mesure, théorie de l'information, statistiques, calcul différentiel, topologie, etc.) où elle s'est révélée très féconde, la théorie des ensembles flous a vue ses premières applications dans le domaine du contrôle et des automatismes industriels. Nous citons les travaux pionniers de E. H. Mamdani [8], ainsi que L. P. Holmblad et J. J. Ostergraad [3] qui ont développé un contrôleur flou pour un four à ciment, ce cas est typique du champ d'applications de cette technique dans le sens que les données d'entrée se présentent avec beaucoup d'imprécision et qu'un modèle mathématique de ce système non-linéaire est entièrement inconnu. Il faut citer aussi le système développé par Sugeno[10] pour le stationnement d'un véhicule.

La majorité des applications de la logique floue, on vu le jour au Japon, dont un pourcentage non négligeable est utilisé pour la régulation d'appareils ménagers (machines à laver, mise au point d'appareils photos) ou de processus industriels (tunneliers, contrôle du débit et de la température de l'eau, fours de verrerie, métro ...). Les grandes entreprises électriques japonaises ont ainsi contribué à la promotion de la logique floue et il n'est pas rare de voir des produits estampillés « *Fuzzy logic Inside* ». Cet engouement a aujourd'hui gagné l'Europe et les États-Unis.

On parle de commande floue lorsque la partie commande d'un automatisme est réalisée en logique floue. Le choix d'une commande floue est directement lié à la souplesse de la logique floue en général, à la simplicité de sa mise en œuvre et sa capacité de description et de représentation des connaissances ou phénomènes vagues imprécis ou incomplets sans recours aux modèles mathématiques complexes. On conçoit l'intérêt de cette approche dans la régulation ou l'asservissement des processus industriels, pour lesquels les informations sont souvent imprécises, incertaines, voir seulement qualitatives, ou contenues dans des boucles de régulation parfois incomplètes.

Un modèle réduit d'une bicyclette télécommandée a été développé au Département de génie électrique et génie informatique de l'Université de Sherbrooke. La dynamique de ce système est telle que le maintien de l'équilibre et de la trajectoire par un simple pilotage manuel à distance est très difficile et même impossible. Le but de notre travail est de développer et tester des algorithmes de contrôle basés sur la logique floue capables de maintenir l'équilibre et la trajectoire du modèle réduit de la bicyclette.

# CHAPITRE 1

## CONCEPTS DE BASE DE LA LOGIQUE FLOUE

Le mode de pensée d'un être humain est généralement fondé sur un raisonnement empirique, où l'analogie et l'intuition jouent un rôle prépondérant. En fait, nos sens et notre jugement ne nous permettent d'évaluer certaines grandeurs que de manière imprécise ou vague : par exemple, la température de l'eau de notre baignoire nous apparaîtra assez chaude ou très froide, ou comprise dans un certain intervalle de température, sans que nous puissions donner directement une valeur exacte de cette température.

Dans ce chapitre nous introduisons les concepts mathématiques des ensembles flous en faisant ressortir leurs propriétés spécifiques. La caractéristique fondamentale d'un ensemble classique est la frontière abrupte entre deux catégories d'éléments : ceux qui sont dans l'ensemble et qui lui appartiennent, et ceux qui sont à l'extérieur et qui ne lui appartiennent pas. Un ensemble classique peut être modélisé par une fonction $\varphi_A$ de type tout ou rien appelée fonction caractéristique de l'ensemble $A$, sous-ensemble d'un ensemble de référence $U$ appelé également univers de référence ou ensemble universel [6].

La fonction $\varphi_A$ définie sur $U$ et à valeur dans l'ensemble à deux éléments $\{0,1\}$ est telle que $\forall x \in U,\ \varphi_A(x) = 1$ si $x \in A$ et $\varphi_A(x) = 0$ si $x \notin A$.

La théorie des ensembles flous se propose de généraliser cette fonction d'appartenance pour des catégories vagues, de telle sorte que la transition entre éléments appartenant et éléments non appartenant à l'ensemble soit graduelle au lieu d'être abrupte.

## 1.1 Notions d'ensembles flous

Soit l'ensemble de référence $U$, on définit un ensemble flou $A$ dans $U$ par la donnée d'une application $\mu_A$ de $U$ dans l'intervalle réel [0,1]. A tout élément $x \in U$ on associe une valeur $\mu_A(x)$, telle que :

$$\mu_A : U \rightarrow [0,1]; \quad 0 \leq \mu_A(x) \leq 1 \tag{1.1}$$

L'application $\mu_A$ est appelée fonction d'appartenance de l'ensemble flou $A$, généralisant ainsi le concept d'appartenance et la notion de fonction caractéristique que nous venons de rappeler ci-dessus. A tout élément $x$ de $U$ la valeur $\mu_A(x)$ associée n'est pas nécessairement égale à 0 ou à 1, elle est a priori quelconque et désigne le degré d'appartenance de $x$ à l'ensemble $A$ [4]. Reprenons l'exemple de la température de l'eau d'une baignoire pour voir la différence entre un ensemble classique et un ensemble flou. La température est définie dans l'ensemble de référence $U$ de [0, 50] C∘. La température d'intérêt est 30 C∘ qui se trouve dans l'ensemble [20, 40] C∘. Une température entre 20 et 40 C∘ est ainsi considérée comme approximativement 30 C∘.

Le degré d'appartenance dans un ensemble classique prend les valeurs $\mu_{temperature}(x) = 0$ si la température se trouve n'importe où à l'extérieur de l'intervalle [20,40] C∘. Quand la température est à l'intérieur de l'intervalle donné, peu importe sa valeur, le degré d'appartenance $\mu_{temperature}(x)$ est toujours égal à 1 (voir figure 1.1).

Dans un ensemble flou la fonction d'appartenance change graduellement. Le degré d'appartenance devient de plus en plus élevé au fur et à mesure que la température approche les 30 C∘. Dans cet exemple une fonction d'appartenance triangulaire a été choisie, cependant elle peut prendre d'autres formes ( voir figure 1.2).

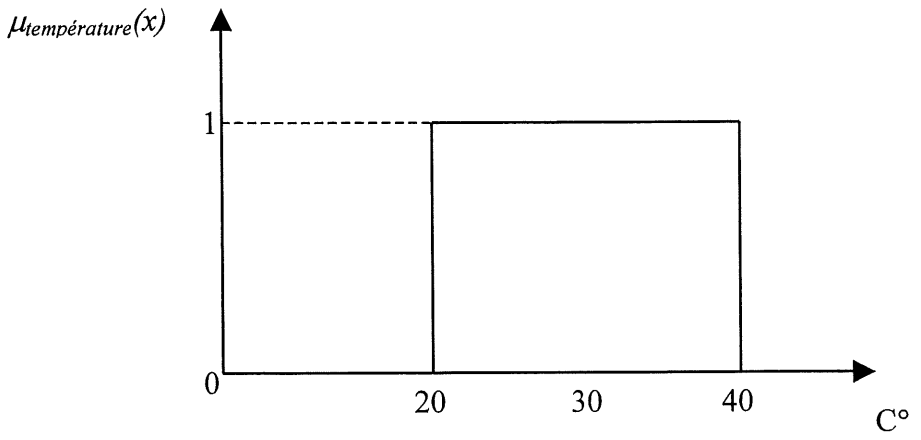



Figure 1.1 Ensemble classique [20,40] C°.

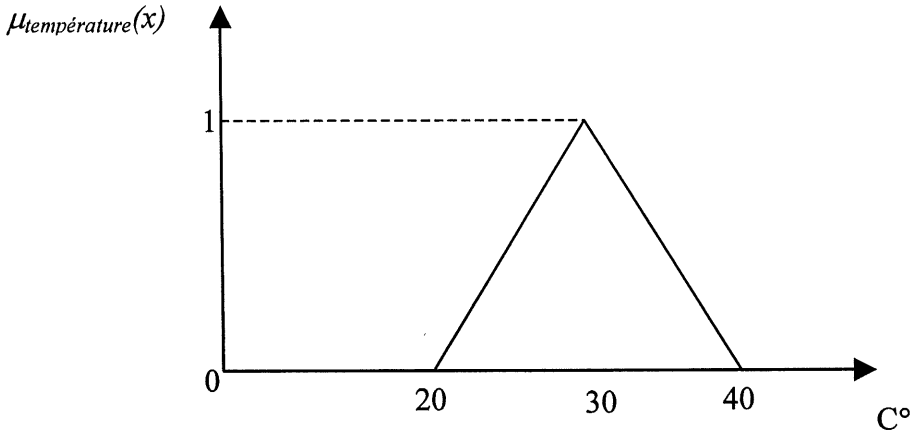



Figure 1.2 Ensemble flou [20,40] C°

## 1.2 Opérations de base sur les ensembles flous

Différentes opérations mathématiques s'appliquent aux ensembles flous. Les notions d'égalité, d'union, d'intersection, d'inclusion, de complémentarité et de produit cartésien sont discutées dans cette section. Les opérations mathématiques entre les ensembles flous sont essentiellement des généralisations des opérations entre les ensembles classiques[Lee91].

### 1.2.1 L'égalité

Deux ensembles flous $A$ et $B$ définis dans l'univers de référence $U$, sont égaux si leurs fonctions d'appartenance $\mu_A(x)$ et $\mu_B(x)$ sont identiques sur tout l'univers de référence $U$ :

$$\forall x \in U : \mu_A(x) = \mu_B(x). \tag{1.2}$$

### 1.2.2 L'inclusion

Un ensemble flou $A$ est un sous-ensemble d'un ensemble flou $B$ si $\forall x \in U : \mu_A(x) \leq \mu_B(x)$. On dit que $A$ est inclus dans $B$ et on note $A \subseteq B$ . Lorsque $A$ est un sous-ensemble propre ou strict, on écrit $A \subset B$. L'inclusion signifie que si un élément de $x$ de $U$ satisfait un tant soit peu une caractéristique floue $A$, il satisfait a priori, à un degré supérieur ou égal, une caractéristique floue $B$.

### 1.2.3 L'union

L'union de deux ensembles flous $A$ et $B$ du même univers de référence $U$ est l'ensemble flou $A \cup B$ de fonction d'appartenance :

$$\forall x \in U : \mu_{A \cup B}(x) = \mu_A(x) \oplus \mu_B(x) \tag{1.3}$$

Où $\oplus$ est une co-norme triangulaire. En pratique $\oplus$ est choisie parmi les deux opérations suivantes :

- Maximum: $x \oplus y = max\ (x,y)$
- Somme algébrique : $x \oplus y = x + y - xy$

Comme illustration définissons deux ensembles $A$, $B$ avec repectivement les degrés d'appartenance $\mu_A$ et $\mu_B$ comme le montre la figure 1.3. L'union des deux ensembles $A$ et $B$ selon le principe du maximum est l'ensemble qui possède le degré d'appartenance $\mu_{A \cup B}$ .

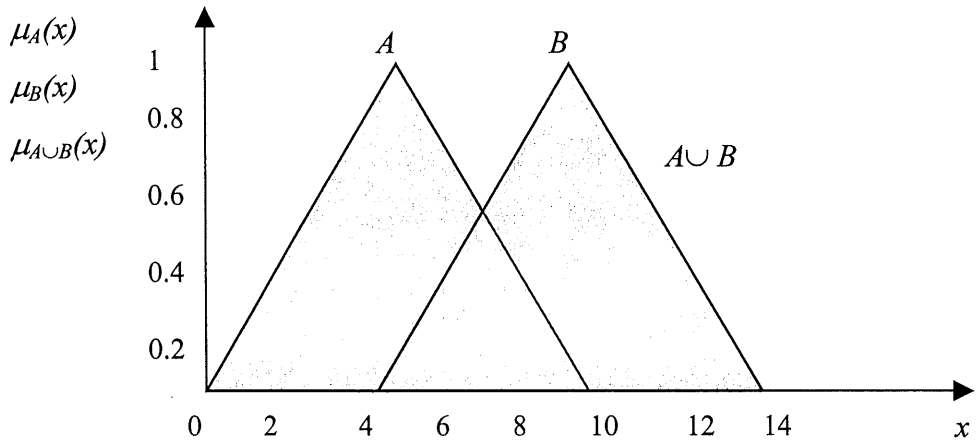


Figure 1.3 Union de deux ensembles flous

1.2.4 L'intersection

L'intersection de deux ensembles flous $A$ et $B$ du même univers de référence $U$ est l'ensemble flou $A \cap B$ de fonction d'appartenance :

$$\forall x \in U : \mu_{A\cap B}(x) = \mu_A(x) * \mu_B(x) \qquad (1.4)$$

Où * est une norme triangulaire. En pratique * est choisie parmi les deux opérations suivantes:

- Minimum : $x * y = min(x,y)$
- Produit algébrique : $x * y = xy$

Illustrons l'intersection entre les deux ensembles $A$ et $B$ décrit dans la section 1.2.3, l'intersection des deux ensembles $A$ et $B$ selon le principe du minimum est l'ensemble qui possède le degré d'appartenance $\mu_{A\cap B}$ comme le représente la figure 1.4.

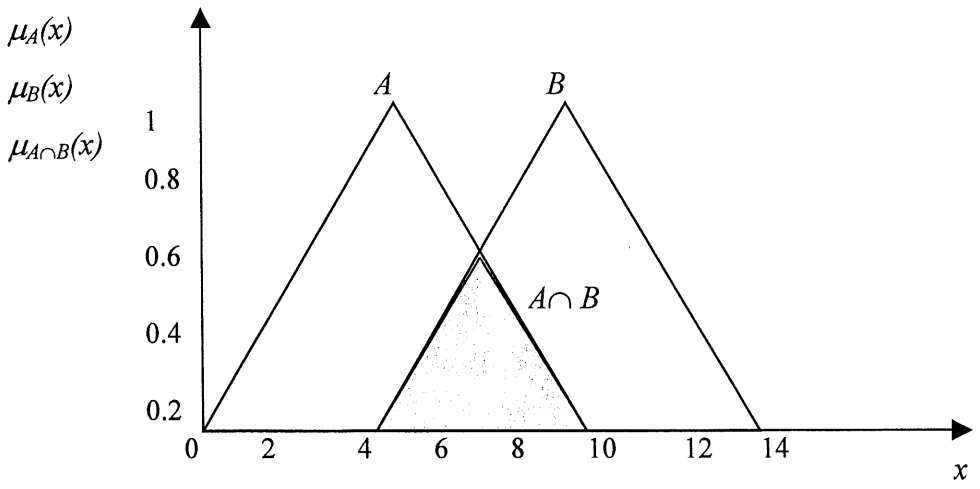



Figure 1.4 Intersection de deux ensembles flous

1.2.5 La complémentarité

Le complément $\neg A$ d'un ensemble flou $A$ de l'univers $U$ est défini comme l'ensemble flou de fonction d'appartenance :

$$\forall x \in U \quad \mu_{\neg A}(x) = 1 - \mu_A(x) \tag{1.5}$$

## 1.3 Variable linguistique

Il peut être très difficile ou même impossible de décrire certaines applications par des valeurs numériques. Dans ce cas, il est plus facile et plus naturel de travailler avec les valeurs et les descriptions linguistiques d'une variables plutôt qu'avec des descriptions numériques. La notion essentielle de variable linguistique a été introduite par Zadeh [14]; elle suggère d'emblée que les valeurs de cette variable ne sont pas numériques, mais plutôt symboliques, en termes de mots ou d'expressions du langage naturel. Par exemple les valeurs de la variable linguistique *Taille* pour décrire la taille d'une certaine population peuvent être : *Très petit*, *petit* , *moyen*, *grand* et *très grand.*

Une variable linguistique est décrite généralement par le triplet $(x,U,T(x))$, $x$ est une étiquette définie dans l'univers de référence $U$ et $T(x)$ représente l'ensemble fini ou non des ensembles flous de $U$. Pour l'exemple des tailles d'une population donnée, on aura $x$ =*Taille* définie dans l'ensemble de référence $U$ = [0,150] et *T(taille)*={*très petit*, *petit*, *moyen*, *grand* et *très grand* }(voir figure 1.5).

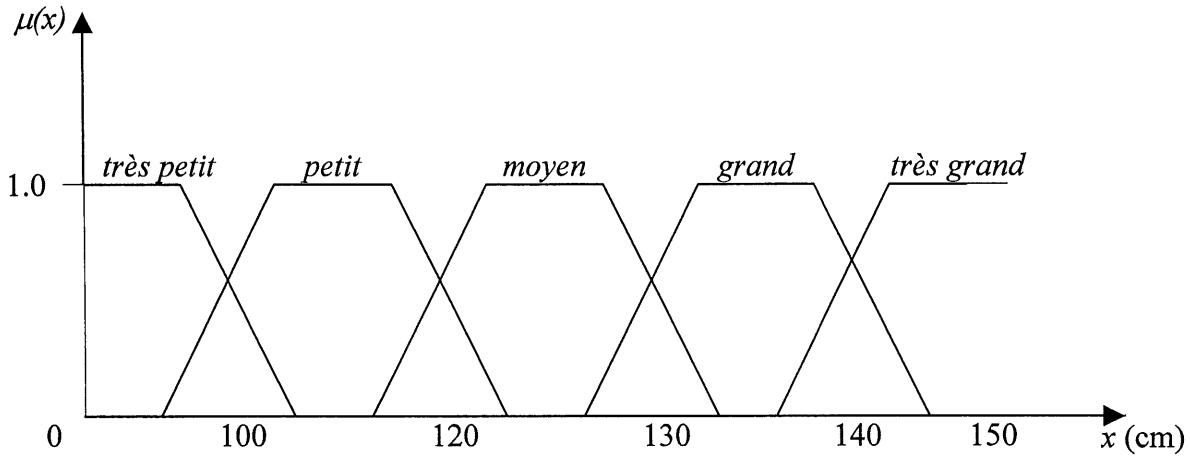



Figure 1.5 Variable linguistique *Taille*

## 1.4 Relations floues

La notion de relation floue intervient lorsqu'une correspondance entre deux ou plusieurs objets ne peut être établie avec précision et sans ambiguïté. Une telle relation est suggérée par les expressions suivantes :

- Pierre est beaucoup plus grand que Paul.
- Le coût de l'opération est nettement supérieur au bénéfice escompté.
- Le prix des articles est très inférieur aux étiquettes correspondantes.
- Le nombre proposé était beaucoup plus petit que 500.
- Paris est très loin de Tokyo.

Une relation floue est une extension de la notion classique de relation basée sur le principe de l'absence ou de présence d'interactions entre les éléments de deux ou plusieurs ensembles. Une relation floue est définie formellement à partir du produit cartésien.

Soit $U = U_1 \times U_2$ le produit cartésien de deux univers $U_1$ et $U_2$. On appelle relation floue entre $U_1$ et $U_2$ un ensemble flou $R$ dans $U_1 \times U_2$ (relation floue binaire). Cette définition peut se généraliser à un produit cartésien d'ordre $n$; dans ce cas, $R$ est un ensemble flou dans $U_1 \times U_2 \times \ldots \times U_n$ (relation floue $n$-aire) [13].

La fonction d'appartenance $\mu_R(x,y)$ ou $\mu_R(x_1,\ldots,x_n)$ est une application du produit cartésien $U$ dans l'intervalle [0,1] selon notre choix initial, ou dans un ensemble ordonné $L$ plus général (un treillis, par exemple); on note : $\mu_R : U_1 \times U_2 \times \ldots \times U_n \rightarrow [0,1]$. Pour $n = 1$, $U = U_1$; on a : $\mu_R : U \rightarrow [0,1]$ où $R$ est une relation floue unaire, c'est à dire un ensemble flou dans $U$.

### 1.4.1 Produit cartésien d'ensembles flous

Soit $n$ ensemble flous $A_1, A_2, \ldots, A_n$ de référentiels respectifs $U_1, U_2, \ldots, U_n$; le produit cartésien des ensembles flous $A_i = 1, \ldots, n$ est un ensemble flou noté $A_1 \times A_2 \times \ldots \times A_n$ dont la fonction d'appartenance est :

$$\mu_{A_1 \times \ldots \times A_n}(x_1,\ldots,x_n) = \min(\mu_{A_1}(x_1),\ldots,\mu_{A_n}(x_n)) \qquad (1.6)$$

On considère deux univers de référence $U_1$ = {a,b,c,d}et $U_2$ = {1,2,3,4}. On définit la relation floue suivante:

$R$ = { 0.3/(a,1), 0.3/(a,2), 0.2/(b,2), 0.2/(b,3), 0.1(c,3), 0.1/(c,4), 0.4/(d,4) }

Cette relation peut être représentée par la matrice suivante:

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| a | 0.3 | 0.3 | 0 | 0 |
| b | 0 | 0.2 | 0.2 | 0 |
| c | 0 | 0 | 0.1 | 0.1 |
| d | 0 | 0 | 0 | 0.4 |

On peut également représenter la relation $R$ par un diagramme sagittal :

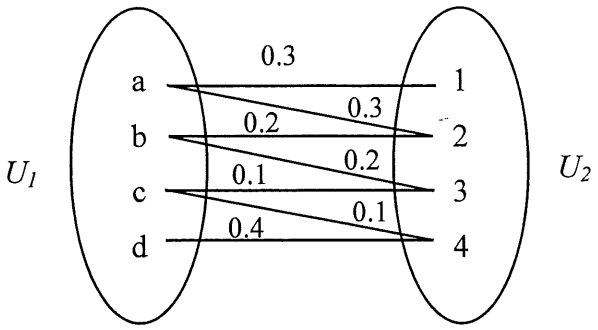



Figure 1.6 Exemple de diagramme sagittal flou

1.4.2 Composition des relations floues

Les relations floues les plus importantes de point de vue des applications sont les compositions. Différentes relations floues peuvent être combinées par les opérations de composition [6].

Il existe différents types d'opération permettant la composition de deux ou de plusieurs relations floues. Pour la composition de deux relations floues $R$ entre $U_1$ et $U_2$ et $S$ entre $U_2$ et $U_3$, les opérateurs de composition, on définit deux types de composition : La composition *max-min* et la composition *max-produit*.

Dans la composition *max-min*, la relation floue composée de $R$ et de $S$ est une relation floue entre $U_1$ et $U_3$ notée $R \circ S$ est dont la fonction d'appartenance est définie par :

$$\mu_{R \circ S}(x,z) = \max_{y \in U_2}[min(\mu_R(x,y), \mu_s(y,z))] \tag{1.7}$$

La composition *max-produit* est définie par :

$$\mu_{R \circ S}(x,z) = \max_{y \in U_2}[(\mu_R(x,y) * \mu_s(y,z))] \tag{1.8}$$

Où * représente l'opérateur produit.

Lorsque les ensembles $U_1$, $U_2$, $U_3$ sont finis, la composition peut être représentée par un produit matriciel, notamment dans le cas de l'opération *max-min* où *max* joue le rôle de l'addition et *min* celui du produit. La composition est nécessaire lorsqu'il faut établir une relation floue entre un objet et un autre, par l'intermédiaire de leurs relations floues respectives avec un troisième objet..

Soit par exemple, la phrase suivante : « *Pierre est beaucoup plus grand que Paul qui est plutôt petit par rapport à Jean* ». La composition des relations « *beaucoup plus grand* » et « *plutôt petit* » nous fournit une indication vague sur la taille de Pierre par rapport à celle de Jean.

Soit $U_1$ = (a,b,c), $U_2$ = (1,2,3), $U_3$ = $(\alpha, \beta, \gamma)$ et les relations floues $R$ entre $U_1$ et $U_2$ et $S$ entre $U_2$ et $U_3$ telles que :

| | 1 | 2 | 3 |
|---|---|---|---|
| a | 0.1 | 0.2 | 0.3 |
| b | 0 | 0.6 | 0.4 |
| c | 0.2 | 0.3 | 0.2 |

| | $\alpha$ | $\beta$ | $\gamma$ |
|---|---|---|---|
| 1 | 0.2 | 0.4 | 0.3 |
| 2 | 0 | 0 | 0.3 |
| 3 | 0.1 | 0 | 0.5 |

Le calcul des éléments de $R \circ S$ s'apparente à celui d'un produit matriciel ordinaire en remplaçant le produit de deux termes par min et l'addition par max. A titre d'exemple, voici le calcul détaillé du premier élément de $R \circ S$ :

$$\mu_{R \circ S}(a,\alpha) = \max_{y \in U_2}[min(\mu_R(a,y), \mu_s(y,\alpha))] \qquad (1.9)$$

On a :

$\min(\mu_R(a,1), \mu_S(1,\alpha)) = \min(0.1, 0.2) = 0.1$

$\min(\mu_R(a,2), \mu_S(2,\alpha)) = \min(0.2, 0) = 0$

$\min(\mu_R(a,3), \mu_S(3,\alpha)) = \min(0.3, 0.1) = 0.1$

$\max(0.1, 0, 0.1) = 0.1$

En utilisant le même procédé pour les autres éléments, on a :

$$R \circ S = \begin{array}{c|ccc} & \alpha & \beta & \gamma \\ a & 0.1 & 0.1 & 0.3 \\ b & 0.1 & 0 & 0.4 \\ c & 0.2 & 0.2 & 0.3 \end{array}$$

1.4.3 Quantificateurs flous

Les quantificateurs flous sont définis par le doublet $\mu_Q$, $U_Q$ tel que [9] :

$$Q = \{\mu_Q, U_Q\} \tag{1.10}$$

Où $\mu_Q$ est la fonction d'appartenance de $Q$ dans l'univers ordonné $U_Q$. Quand l'ensemble flou de $U_Q$ est normalisé, on appelle $Q$ une quantité floue.

Des exemples de quantificateurs flous sont « *un peu* », « *tout* », « *il existe* », « *presque tout* » avec les fonctions d'appartenance correspondantes $\mu_{un\ peu}(x)$, $\mu_{tout}(x)$, $\mu_{presque\ tout}(x)$, $\mu_{il\ existe}(x)$ où les fonctions d'appartenance sont définies comme :

$$\mu_{tout}(x) = (-\infty, +\infty) \tag{1.11}$$

$$\mu_{presque\ tout}(x) = +\infty \tag{1.12}$$

$$\mu_{il\ existe}(x) = x \tag{1.13}$$

$$\mu_{un\ peu}(x) = \frac{1}{x} \tag{1.14}$$

## 1.5 Modificateurs flous

Le modificateur linguistique est un opérateur $m$ qui permet de modifier des variables linguistiques [15]. Le modificateur flou peut modifier les variables linguistiques, les valeurs linguistiques et les quantificateurs flous. Le modificateur flou m est défini par la fonction $f_m$ telle que :

$$f_m : U \rightarrow U \tag{1.15}$$

Les modificateurs les plus fréquemment appliqués à la fonction d'appartenance sont la négation, la concentration et la diffusion. La négation de la fonction d'appartenance $\mu$ est la fonction d'appartenance $\mu_m$ telle que :

$$\mu_m = 1 - \mu \tag{1.16}$$

Si on prend par exemple, comme univers de référence la taille d'une population donnée, la négation de l'expression « *Paul est grand* » est définie par la phrase « *Paul est non grand* » ou l'opérateur de négation est représenté par l'expression « *non* ».

La concentration est spécifiée par le modificateur « *très* » qui transforme la fonction d'appartenance $\mu$ à la fonction $\mu_m$ telle que :

$$\mu_m = \mu^2 \tag{1.17}$$

La diffusion est spécifiée par le modificateur « *plus ou moins* ». Il transforme la fonction d'appartenance $\mu$ à la fonction $\mu_m$ telle que :

$$\mu_m = \mu^{1/2} \tag{1.18}$$

Il existe d'autres modificateurs linguistiques qui permettent de changer la forme de la fonction d'appartenance.

## 1.6 Raisonnement et implication floue

### 1.6.1 Raisonnement flou

La base du raisonnement logique est constituée d'énoncés ( axiomes, formules, propositions) et de règles d'inférence. Une règle d'inférence est une procédure qui établit un lien entre une ou plusieurs prémisses (énoncés initiaux) et une conclusion (énoncé final). Rappelons que par ailleurs les deux règles essentielles de la logique classique sont le modus ponens et le modus tollens basés sur l'implication $p \Rightarrow q$, où $p$ et $q$ sont deux propositions binaires ( $p$ est l'antécédent et $q$ la conclusion). Elles sont définies de la façon suivante [13]:

Modus ponens :

$p \Rightarrow q$ vrai et $p$ vrai alors $q$ vrai

Modus tollens :

$p \Rightarrow q$ faux et $p$ faux alors $q$ faux

En logique classique les propositions p et q sont précises et l'observation des faits permet une vérification immédiate. Prenons la règle suivante : « *Si la surface habitable de la maison est de 200 $m^2$ alors sont coût est de 1 million de dollars* ». Dans cette règle : $p$ = « *la surface habitable est 200 $m^2$* » et la conclusion $q$ = « *le coût est de 1 million de dollars* », elle peut être vérifiée en mesurant la surface concernée (on suppose que la surface concernée obéit à une convention précise indiquant les pièces à prendre en compte). Cet exemple est une illustration de la règle du modus ponens, le modus tollens pourrait être vérifié de façon similaire.

En logique floue ou les propositions sont vagues il n'est plus possible d'appliquer ces règles d'inférence telles qu'elles sont utilisées en logique classique. Soit, par exemple, la règle floue suivante « *Si la maison est spacieuse alors sont coût est élevé* » ; on voit que la vérification

de la conclusion n'est plus immédiate, car non seulement l'antécédent est vague, la conclusion est elle-même vague. Quelle conclusion peut-on tirer si la mesure (fait observé) indique que la maison est relativement spacieuse ou très grande ou que sa surface habitable vaut 350 $m^2$ ?

Le raisonnement flou introduit par Zadeh [14]et dont l'outil de base est le concept de relation floue, fournit une extension des règles d'inférence classiques (modus ponens et modus tollens généralisés) afin de permettre la représentation des connaissances vagues, imprécises et/ou incertaines.

### 1.6.2 Implication floue

L'implication de la logique classique est une proposition constituée de deux propositions $p$ et $q$ et notée $p \Rightarrow q$ et vraie dans tout les cas sauf dans celui où $p$ est vraie et $q$ fausse. En logique floue, nous avons déjà fait état des propositions floues de la forme : « Si $X$ est $A$ alors $Y$ est $B$ » où $X$ et $Y$ sont deux variables linguistiques sans interactions et concernant des sujets différents (taille et coût par exemple), alors que $A$ et $B$ sont deux étiquettes (petite et élevé, par exemple), éléments respectifs des ensembles $T(X)$ et $T(Y)$. la première partie « $X$ est $A$ » constitue la prémisse ou l'antécédent, la dernière « $Y$ est $B$ » la conclusion ou la conséquence. L'ensemble flou $A \rightarrow B$ est une relation floue $R$ entre les univers $U$ et $V$ et sa fonction d'appartenance $\mu_{A\rightarrow B}(x,y)$ avec $(x,y) \in U \times V$ , représente la valeur de vérité de l'implication floue. La fonction $\mu_{A\rightarrow B}(x,y)$ dépend des fonctions d'appartenance $\mu_A(x)$ et $\mu_B(x)$, selon une relation caractéristique du type d'implication considéré. Les implications les plus utilisées sont les suivantes :

$$\mu_{A\rightarrow B}(x,y) = \min(1,1-\mu_A(x)+\mu_B(y)) \tag{1.19}$$

$$\mu_{A\rightarrow B}(x,y) = \max(1-\mu_A(x),\mu_B(y)) \tag{1.20}$$

Comme on l'a décrit ci-dessus, une règle floue est décrite sous la forme « Si Prémisse Alors Action ». La prémisse peut être soit simple où de la forme d'une conjonction (ET) de plusieurs prémisses. Par contre une disjonction (OU) est traitée comme plusieurs règles en parallèle. Les

règles prennent deux formes. La première forme est composée de plusieurs entrées et une sortie, la deuxième forme comprend plusieurs entrées et plusieurs sorties. La première forme peut être généralisée comme un ensemble des règles suivantes :

SI $x_1$ est $A_1$ ET $x_2$ est $B_1$...ET $x_n$ est $C_1$ ALORS $y$ est $M_1$

OU

SI $x_1$ est $A_2$ ET $x_2$ est $B_2$...ET $x_n$ est $C_2$ ALORS $y$ est $M_2$

.

.

OU

SI $x_1$ est $A_n$ ET $x_2$ est $B_n$...ET $x_n$ est $C_n$ ALORS $y$ est $M_m$

Où $x_1, x_2, ..., x_n \in U_1 \times U_2 \times ... \times U_n$ et $y \in V$ sont les variables et $A_1$, ...$A_m$, $B_1$,...$B_m$, $C_1$, ...$C_m$ sont les étiquettes floues définies dans $U_n$ et $M_m$ est défini dans $V$.

La deuxième forme avec plusieurs entrées et plusieurs sorties est démontrée par l'ensemble des règles suivantes :

SI $x_1$ est $A_1$ ET $x_2$ est $B_1$...ET $x_n$ est $C_1$ ALORS $y_1$ est $M_1$, $y_2$ est $N_1$ ...$y_n$ est $O_1$

OU

SI $x_1$ est $A_2$ ET $x_2$ est $B_2$...ET $x_n$ est $C_2$ ALORS $y$ est $M_2$, $y_2$ est $N_2$ ...$y_n$ est $O_2$

.

.

OU

SI $x_1$ est $A_n$ ET $x_2$ est $B_n$...ET $x_n$ est $C_n$ ALORS $y$ est $M_m$, $y_2$ est $N_n$ ...$y_n$ est $O_n$

Où $x_1, x_2, ..., x_n \in U_1 \times U_2 \times ... \times U_n$ et $y_1, ..., y_n \in V_1 \times V_2 \times ... V_n$ sont les variables et $A_1, ... A_m$, $B_1, ... B_m$, $C_1, ... C_m$ sont les étiquettes floues définies dans $U_n$ et $M_m$, $N_m$, $O_m$ définis dans $V_n$.

1.6.3 Base de règles proposée par Sugeno

L'ensemble des règles proposé par Sugeno [11] est une modification du raisonnement flou introduit dans les sections précédentes. L'implication floue dans ce cas est une fonction de variables d'entrée. Les règles sont écrites sous la forme :

SI $x_1$ est $A_1$ ET $x_2$ est $B_1 ...$ ET $x_n$ est $C_1$ ALORS $f_1(x_1, x_2, ... x_n)$

OU

SI $x_1$ est $A_2$ ET $x_2$ est $B_2 ...$ ET $x_n$ est $C_2$ ALORS $f_2(x_1, x_2, ... x_n)$

.

.

OU

SI $x_1$ est $A_n$ ET $x_2$ est $B_n ...$ ET $x_n$ est $C_n$ ALORS $f_n(x_1, x_2, ... x_n)$

Les $x_1, ... x_n$ sont les variables et $A_1, ... A_m$, $B_1, ... B_m$, $C_1, ... Cm$ sont les étiquettes floues avec les fonctions d'appartenance linéaires.

La partie des conséquence des règles est représentée par une fonction linéaire de la forme :

SI $x_1$ est $A_1$ ET $x_2$ est $B_1 ...$ ET $x_n$ est $C_1$ ALORS $y_n = p_0 + p_1 x_1 + p_2 x_2 ... p_n x_n)$

# CHAPITRE 2

## SYSTEME FLOU ET CONTRÔLEUR FLOU

Dans un premier temps, la théorie de la logique floue a été développée pour des besoins de formaliser la représentation et le traitement de connaissances imprécises ou approximatives. Celles-ci étaient décrites sous la forme de règles, transcription de l'analyse d'un expert.

On conçoit l'intérêt de faire entrer l'approche floue dans la régulation ou l'asservissement de certains processus industriels, pour lesquels les informations disponibles sont souvent imprécises, incertaines, qualitatives et parfois incomplètes. Le savoir-faire de l'opérateur, constitué entre autres de règles souvent simples, lui permet de conduire chaque machine, prise isolément, plus correctement parfois qu'un algorithme classique. L'intérêt de la « commande floue » (bien que cette expression soit abusive puisque la commande est parfaitement définie) est de « faire entrer » l'expert dans le processus.

Durant la dernière décennie, le contrôle flou, initié par les travaux pionniers de Mamdani [7] qui est le premier à avoir expérimenté la théorie de Zadeh sur une chaudière à vapeur, a émergé comme étant l'un des domaines les plus actifs pour la recherche dans l'application de la théorie des ensembles flous, la logique floue et le raisonnement approximatif. Ces applications vont du contrôle industriel de processus au diagnostic médical.

Dans ce chapitre nous allons introduire l'architecture de base et la méthodologie de la conception des contrôleurs flous.

## 2.1 Structure d'une commande floue

Les contrôleurs flous sont des systèmes à base de règles où la logique et les concepts flous (ensembles flous, variables linguistiques et raisonnement approximatif ) sont utilisés comme structure de contrôle (figure 2.1). Cette structure est composée de quatre blocs distincts qui sont : la fuzzification, le moteur d'inférence, la défuzzification ainsi la base de règles.

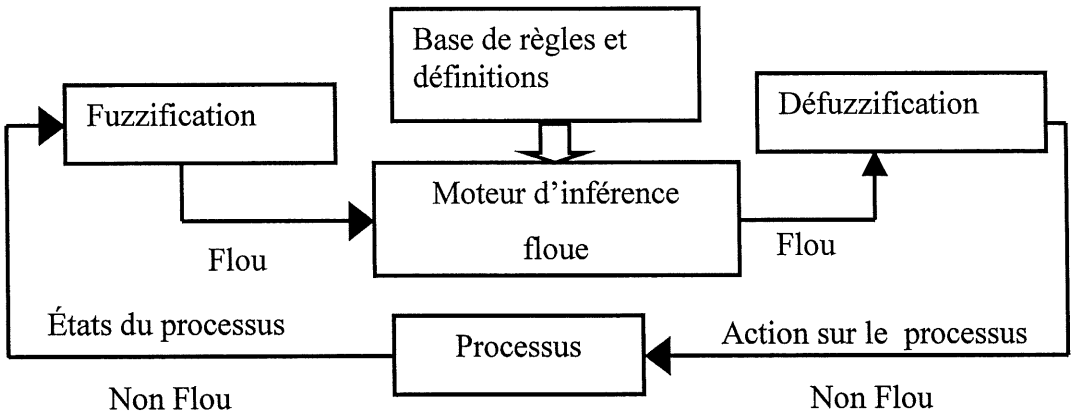



Figure 2.1 Structure conventionnelle d'un contrôleur flou

### 2.1.1 Base de règles et définitions

On regroupe dans ce bloc, d'existence virtuelle, l'ensemble des définitions utilisées dans un contrôleur flou (univers de référence, les étiquettes floues, choix des opérateurs flous...), ainsi que la base de règles, transcription sous forme de règles «SI...ALORS» de la stratégie du contrôle. Cette collection de règles caractérise la relation entrées-sorties du système.

La création et l'utilisation d'une base de règles nécessitent l'existence, pour chaque univers de référence considéré, des ensembles flous particuliers, La définition de ces ensembles flous fait l'objet de la partition floue.

La partition floue d'un univers de référence $U$ consiste à définir $n$ ensembles $F_i$ de façon à recouvrir $U$. C'est-à-dire que, pour tout élément $x$ de $U$, il faut assurer une appartenance minimale $\varepsilon$ à l'union des $F_i$ [2].

$$\bigcup_{i=1}^{n} F_i \supseteq U_\varepsilon = \{x \in U; \mu_{U_\varepsilon}(x) = \varepsilon\} \tag{2.1}$$

La condition (2.1) se traduit au niveau des fonctions d'appartenance par la condition :

$$\forall\, x \in U; \mu_{F_i}(x) \vee \ldots \vee \mu_{F_n}(x) \geq \varepsilon \tag{2.2}$$

Où $\vee$ est un opérateur d'union.

Comme la fonction *max* minore toutes les fonctions utilisées comme opérateurs d'union, pour assurer une partition floue de niveau $\varepsilon$, il faut et il suffit que tout élément $x$ de $U$ possède un degré d'appartenance à l'union des *Fi*, avec pour opérateur d'union la fonction *max*, supérieur ou égal à $\varepsilon$.

Par exemple, si l'on considère l'univers de référence d'une température comme étant l'intervalle [0°C,60°C], on peut choisir trois ensembles flous *Faible*, *Moyenne* et *Élevée* de cet univers de référence, présenté dans la figure 2.2.

Le nombre des ensembles flous à définir dans une partition d'un univers de référence est fixé par l'expert. Plus ce nombre est important et plus on définit de classes sur cet univers de référence ce qui permet d'augmenter la sensibilité de la commande floue.

Une fois la partition des univers de référence réalisée, il est possible de définir la base de règles. Celle-ci caractérise les relations entre les classes d'événements possibles en entrée et les commandes correspondantes.

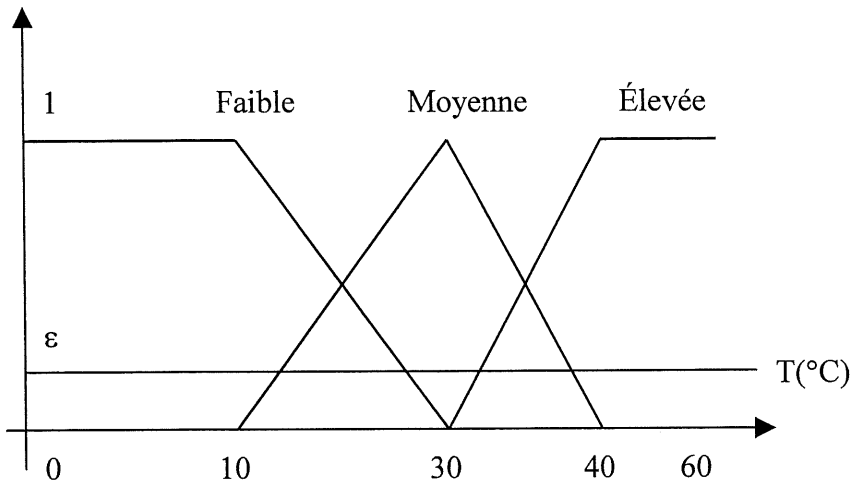



Figure 2.2 Exemple de partition floue

Par conséquent, si l'on considère $n$ univers de référence $U$, pour les prémisses des règles floues et si pour chaque univers $U$, on définit une partition en $m_i$, ensembles flous, le nombre maximum de règles $r_{max}$ est de :

$$r_{max} = \prod_{i=1}^{n} m_i \tag{2.3}$$

Le nombre de règles définies par l'expert peut être inférieur à $r_{max}$. C'est le cas en particulier s'il existe des configurations d'ensembles flous impossibles à obtenir pour le système. De plus, le nombre des ensembles flous définissant la partition de l'univers de référence de la commande n'est pas forcément égal au nombre de règles. En effet, il est possible de considérer des règles différentes aboutissant à la même conclusion.

Enfin, on peut remarquer qu'une augmentation de la sensibilité de la commande floue obtenue par une partition plus fine des univers de référence des prémisses aboutit à un accroissement important du nombre des règles à définir par l'expert.

2.1.2 Fuzzification

La Fuzzification est l'opération de la transformation d'une entrée numérique, $x_0$, en un ensemble flou $A$ dans $U$ afin d'assurer la compatibilité entre l'entrée du contrôleur flou et le moteur d'inférence. Pour ce faire, on utilise un opérateur dit de fuzzification qui associe à une mesure de la variable $x_0$ une fonction d'appartenance particulière $\mu_A(x_0)$.

Le choix de l'opérateur de fuzzification dépend de la confiance que l'on accorde aux mesures effectuées. Ainsi, si la mesure de la variable est exacte, l'ensemble flou $A$ doit être représenté par un fait précis. Par conséquent, on utilise comme opérateur de fuzzification la transformation dite du singleton. La fonction d'appartenance de l'ensemble flou $A$ est alors définie par :

$\mu_A : U \rightarrow U,$

$\mu_A(x)$=1 si $x = x_0$

$\mu_A(x)$= 0 si $x \neq x_0$,

La figure 2.3 monte l'aspect de cette fonction d'appartenance.

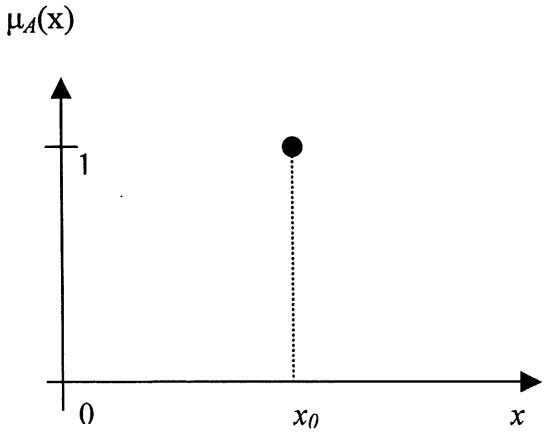



Figure 2.3 Aspect d'une fonction d'appartenance de type singleton

Ainsi l'ensemble flou $A$ réalisé par cette méthode de fuzzification ne comprend que l'élément $x_0$.

Par contre, si la mesure de la variable est incertaine, par exemple à cause de bruits, l'ensemble flou $A$ doit être représenté par un fait imprécis. On utilise alors une méthode de fuzzification qui associe à la variable mesurée $x_0$ une fonction d'appartenance telle que, par exemple :

$$\mu_A(x) = max\left\{0\ ;\ 1 - \frac{|x - x_0|}{\varepsilon}\right\} \tag{2.4}$$

La représentation graphique de cette fonction d'appartenance est donnée en figure 2.4

Cet ensemble flou comprend donc la mesure $x_0$ avec une appartenance unité et les valeurs voisines de $x_0$.

La base du triangle $\varepsilon$ est fonction de l'importance relative des erreurs de mesure. En effet, plus elles sont importantes, plus la mesure de la variable $x_0$, devient imprécise, et donc, plus le triangle doit s'élargir. Pour les cas extrêmes, s'il n'y a pas de bruit ($\varepsilon = 0$) on retrouve la fuzzification par la méthode du singleton; si on ne peut accorder aucune confiance à la mesure effectuée si $\varepsilon = \infty$ l'ensemble flou $A$ devient dans ce cas complètement incertain sur l'univers de référence considéré et on a $x_0 = U$.

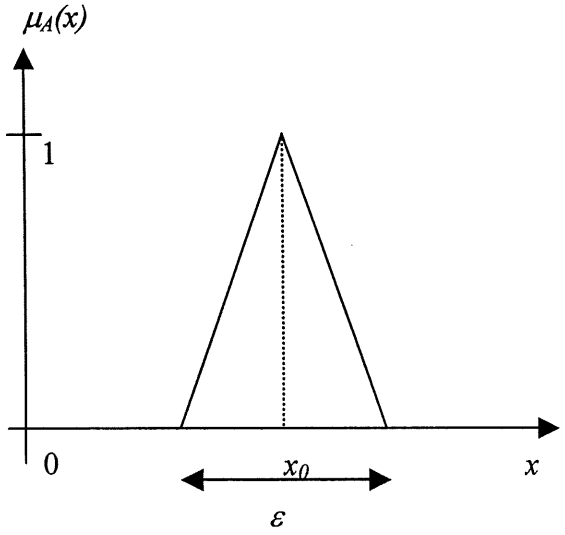



Figure 2.4 Méthode de fuzzification pour une mesure incertaine.

### 2.1.3 Mécanisme d'inférence

Le moteur d'inférence est le noyau du contrôleur flou, il utilise les principes de la logique floue et du raisonnement pour combiner les règles floues. La valeur floue $B'$ dans $V$ est induite par une valeur floue $A'$ dans $U$. La sortie du moteur d'inférence peut prendre deux formes:[Lee1991]

- $M$ ensemble flous $B^l$ ($l$=1,2,...$M$) de la forme :

$$\forall y \in V \ \mu_{B'}(y) = \sup_{x \in U}(\mu_{A'_1 \times \ldots \times A'_n \to B^l}(x,y) * \mu_{A'}(x)) \tag{2.5}$$

- Un ensemble flou $B'$ formé de l'union des $M$ ensembles flous $B^l$. C'est à dire:

$$\forall y \in V \ \mu_{B'}(y) = \mu_{B^1}(y) \oplus \ldots \oplus \mu_{B^M}(y) \tag{2.6}$$

### 2.1.4 Défuzzification

L'ensemble flou $Y$ de l'univers de référence $V$ ayant été calculé par le mécanisme d'inférence, l'interface de défuzzification a pour objectif de le transformer en une valeur non floue permettant ainsi la commande effective du système. Cette opération est effectuée par l'opérateur de défuzzification. Cet opérateur est nécessaire car dans la majorité des applications, des commandes de contrôle ordinaires et non linguistiques sont utilisées pour l'actuation du processus.

Afin de présenter les méthodes de défuzzification les plus utilisées, considérons l'ensemble flou $Y$ de l'univers de référence $V$ représenté par la figure 2.5.

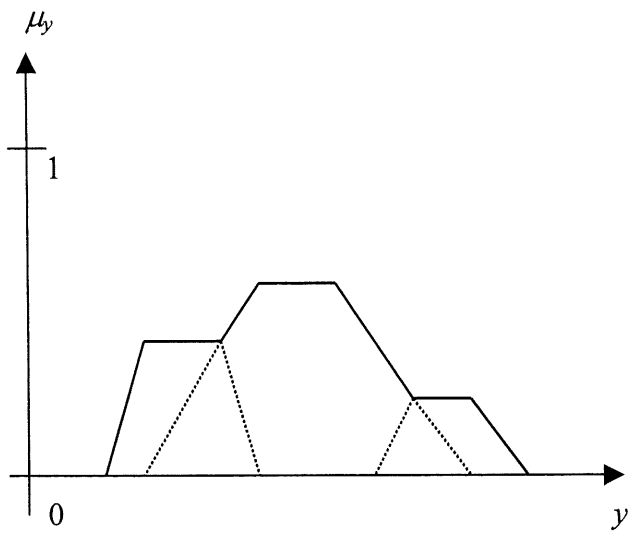



Figure 2.5 Définition de l'ensemble flou $Y$

La fonction d'appartenance de ce ensemble flou représente la pertinence de chaque action $Y$ envisageable.

La première méthode correspond à l'utilisation du principe du maximum. Celui-ci consiste à choisir comme sortie $y_0$ de l'interface de défuzzification une des valeurs possédant la plus grande appartenance à l'ensemble flou $Y$. Par cette méthode, on applique une des actions les plus pertinentes.

$$D_F : V \rightarrow V ;$$
$$Y=\{y \in V;\ \mu_Y(y)\} \rightarrow D_F(Y) = y_0; \qquad (2.7)$$
$$\text{avec } \mu_Y(y_0) = \sup_y \{\mu_Y(y)\}$$

Ainsi, pour l'ensemble flou $Y$ de la figure 2.5, la figure 2.6 présente la sortie $y_0$ obtenue par cette méthode de défuzzification.

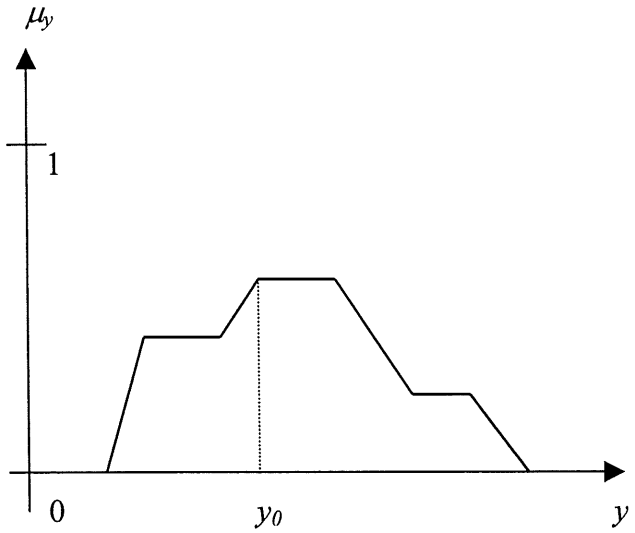



Figure 2.6 Sortie de la commande floue pour une défuzzification par le principe du maximum.

Il se peut, comme dans le cas de la figure 2.5, que la fonction d'appartenance de l'ensemble flou $Y$ possède plusieurs maxima identiques. Dans ce cas, et afin d'éviter un choix arbitraires de la valeur $y_0$, on choisit comme opérateur de défuzzification une transformation qui effectue la moyenne de ces maxima, c'est-à-dire la moyenne des actions les plus pertinentes.

$$D_F : V \rightarrow V,\ Y = \{y \in V ; \mu_Y(y)\} \longrightarrow D_F(Y) = y_0 = \frac{\int_S y\, dy}{\int_S dy} \tag{2.8}$$

avec $S = \{y_i \in V ; \mu_Y(y_i) = sup\{\mu_Y(y)\}\}$

La figure 2.7 donne, pour cette méthode, la sortie $y_0$ pour l'ensemble flou $Y$ de la figure 2.5.

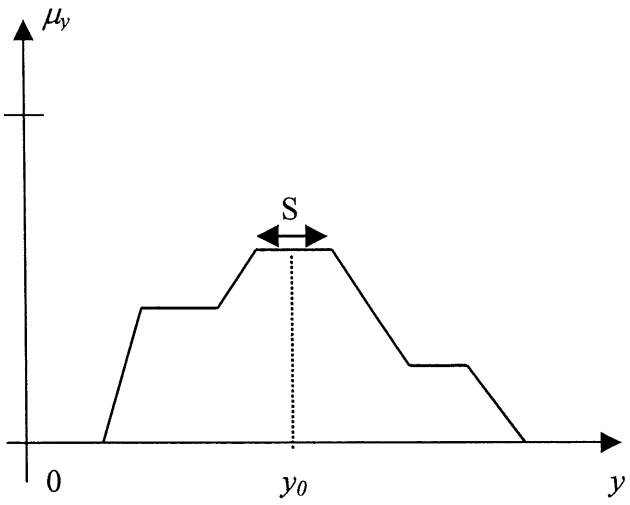



Figure 2.7 Sortie de la commande floue pour une défuzzification par le principe de la moyenne des maxima.

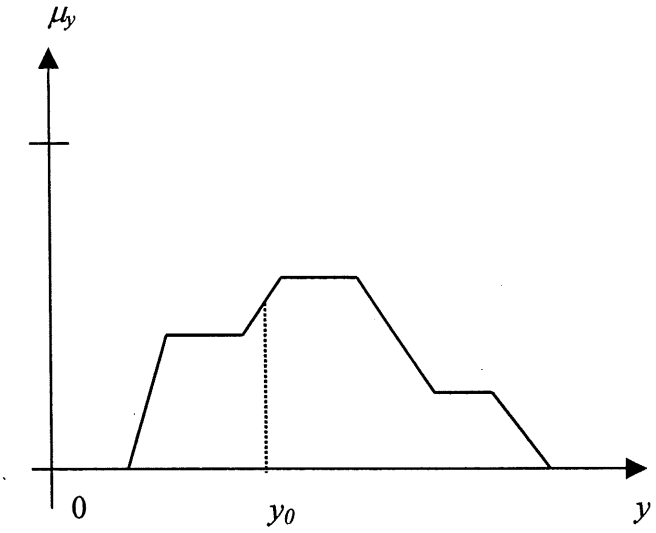



Figure 2.8 Sortie de la commande floue pour la défuzzification barycentrique

Pour prendre en compte l'influence de l'ensemble des valeurs $y$ proposées par la solution floue $Y$ (c'est-à-dire possédant une appartenance non nulle à ce sous ensemble), on utilise une méthode de défuzzification dite barycentrique.

$$D_F : V \rightarrow V,\ Y = \{y \in V ; \mu_Y(y)\} \longrightarrow D_F(Y) = y_0 = \frac{\int_S y\ \mu_Y(y)dy}{\int_S \mu_Y(y)dy} \qquad (2.9)$$

On constate que cette méthode effectue la moyenne des actions $y$ pondérées par leur appartenance $\mu_Y(y)$. La figure 2.8 représente l'application de cette méthode au ensemble flou $Y$ de la figure 2.5.

La forme des règles floues, le choix du moteur d'inférence ainsi que les stratégies de fuzzification et de défuzzification font qu'il existe plusieurs types de modèles flous. Selon que le modèle flou est utilisé pour représenter un système de prise de décision, de diagnostic ou de contrôle, un modèle flou s'apprête mieux que les autres.

## 2.2 Application à la commande d'un système de premier ordre

Le but de cet exemple est de présenter les différentes étapes de la conception d'une commande floue, ainsi que la construction graphique de la commande pour une entrée particulière.

Le processus que l'on désire commander est un système du premier ordre dont la constante de temps $\tau$ et le gain statique $k$ sont incertains, mais supposés positifs. Pour représenter cette contrainte, les paramètres de la commande floue seront établis pour des valeurs nominales de $k$ et $\tau$. La figure 2.9 présente la fonction de transfert du système étudié.

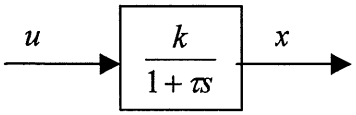



Figure 2.9 Système du premier ordre

On considère les définitions suivante : $u$ est la commande du système et $x$ est la sortie. Ils sont reliés par l'équation différentielle suivante :

$$\dot{x} = -\frac{1}{\tau}x + \frac{k}{\tau}u. \tag{2.10}$$

On commande le système par une boucle de régulation classique présentée par la figure 2.10.

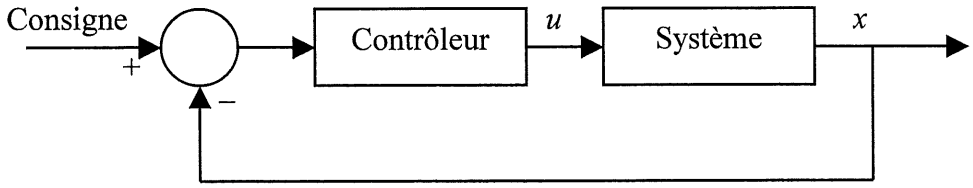



Figure 2.10 Boucle de régulation utilisée

Les variables de régulation envisagées sont l'erreur entre la consigne et la sortie du système et la variation de cette erreur.

Erreur = $e$ = Consigne - $x$

Variation de l'erreur = $\Delta e = e(t)-e(t-T)$, $T$ constante

On considère pour chaque variable d'entrée (l'erreur et la variation de l'erreur) et deux ensembles flous : « Positive » et « Négative ». Ils sont définis sur l'ensemble des réels. On admet trois actions possibles pour la variation de la sortie de la commande floue définie par les ensembles flous : « Négative », « Nulle » et Positive » sur l'univers de référence $[-\Delta U, \Delta U]$. La figure 2.11 présente l'aspect de ces différents ensembles flous.

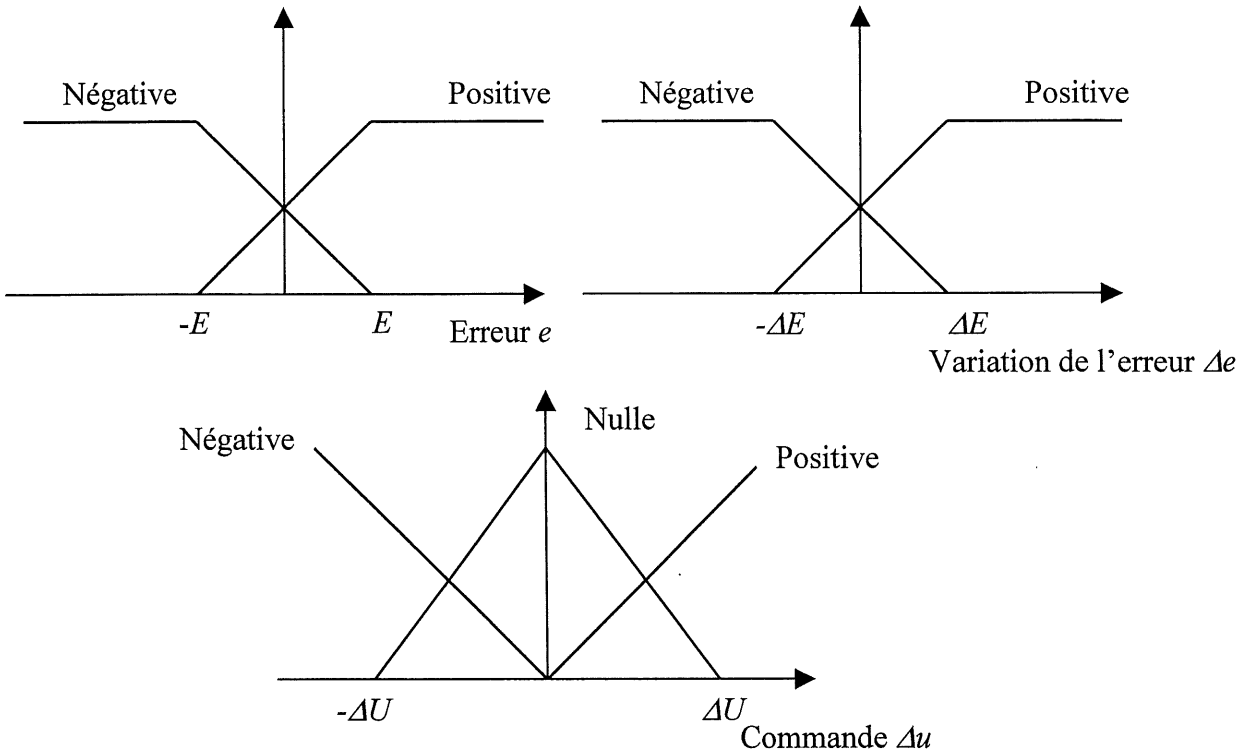



Figure 2.11 Présentation des ensembles flous utilisés

Nous voulons que notre contrôleur ai l'effet d'un contrôleur proportionnel-intégral (PI), dans ce cas le système doit réagir de façon à suivre les variations de la commande Par conséquent, pour obtenir une erreur et une variation de l'erreur nulles, on utilise la base de règles définie dans le tableau 2.1.

| **Règles n°** | **Si *e* est** | **Et *Δe* est** | **Alors *Δu* est** |
|---|---|---|---|
| 1 | Négative | Négative | Négative |
| 2 | Négative | Positive | Nulle |
| 3 | Positive | Négative | Nulle |
| 4 | Positive | Positive | Positive |

Tableau 2.1 Base de règles de la commande floue

La fonction d'appartenance de l'ensemble flou relatif à la variation de la commande $\Delta u$ correspond au maximum des commandes élémentaires $\mu_{R_i}(x_0, \Delta u)$, définis à partir du vecteur de mesure $x_0$. Ces commandes élémentaires sont construites en utilisant l'opérateur d'intersection (utilisation du ET dans les prémisses), l'implication floue (règle floue) et le produit cartésien (utilisé dans le mécanisme d'inférence). La construction graphique de la solution floue suit les étapes suivantes :

- Étape 1 : Pour chaque règle $R_i$ : Définir $\mu_{E_i}(e_0)$ et $\mu_{\Delta E_i}(\Delta e_0)$, les degrés d'appartenance de l'erreur et de sa variation aux ensembles flous définis dans les univers de référence $e$ et $\Delta e$.

- Étape 2 : Reporter le minimum des deux valeurs sur l'ensemble flou $\Delta U_i$ (opérateur d'intersection : fonction minimum).

- Étape 3 : Construire la commande floue élémentaire pour la règle $R_i$ (implication floue et mécanisme d'inférence).

- Étape 4 : Prendre le maximum des solutions élémentaires (agrégation des règles par l'utilisation de l'opérateur d'union).

- Étape 5 : Appliquer l'opérateur de défuzzification à l'ensemble flou obtenu.

Ces étapes de la construction graphique de la commande floue sont représentées par la figure 2.12.

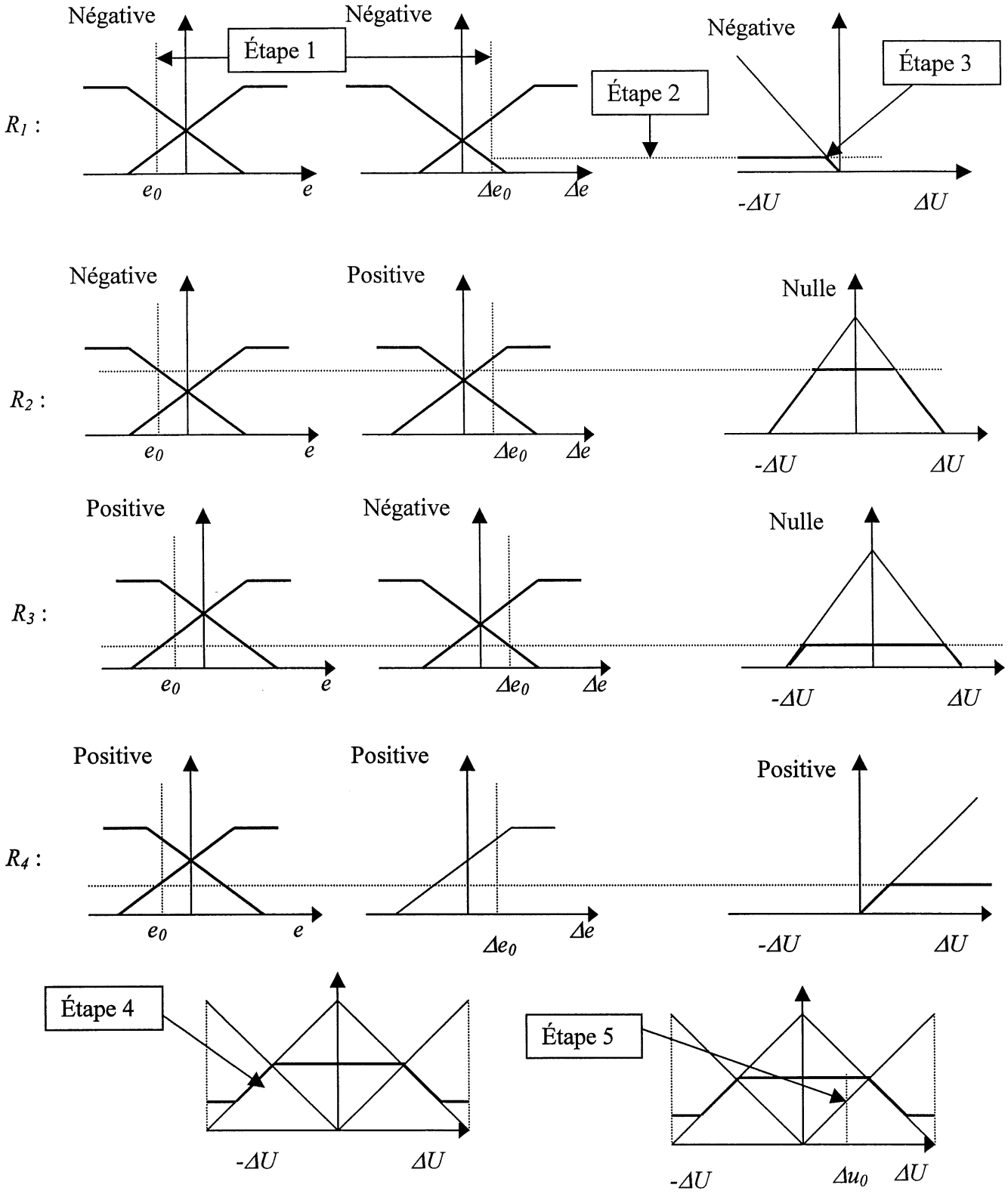



Figure 2.12 Construction de la commande floue $\Delta u_0$

# CHAPITRE 3

## CONTRÔLE DE L'EQUILIBRE ET DE LA TRAJECTOIRE D'UNE BICYCLETTE

Le contrôle d'une bicyclette est un problème riche qui offre un nombre considérable de défis à la recherche actuelle dans les domaines de la robotique, de la mécanique et des systèmes intelligents. Un modèle réduit d'une bicyclette équipée d'un système de stabilisation a été développé au Département de génie électrique et génie informatique de l'Université de Sherbrooke.

Le prototype est conçu pour illustrer les concepts du contrôle numérique et le contrôle intelligent avec une emphase sur la stabilisation de la trajectoire et le contrôle d'équilibre. L'actuateur est un servomoteur haute performance qui dirige le guidon, le capteur principal est un gyromètre piézo-électrique placé dans la tête du cycliste. Le système prend en compte les commandes du pilote à travers la télécommande (direction et vitesse) ou bien à travers un ordinateur personnel via le lien série, ainsi que l'information provenant du gyromètre et du capteur de vitesse de la roue arrière. Il agit sur le servomoteur qui dirige la roue avant et sur le circuit de puissance pour le moteur de propulsion (voir figure 3.1).

Une architecture de contrôle intelligent basée un contrôleur flou a été élaborée pour obtenir un comportement stable de la bicyclette et assister le pilotage humain. Comme on le sait l'être humain n'a nul besoin de méthodes mathématiques rigoureuses pour piloter un vélo bien qu'il adopte toujours une stratégie de contrôle précise et efficace.

Le pilote et le vélo forment un tout. C'est la connaissance des interactions entre les deux entités qui assure le comportement souhaité. C'est dans ce sens qu'un contrôleur flou a été développé pour permettre le pilotage du vélo. La base de connaissances de notre système flou contient les paramètres des fonctions d'appartenance et les règles d'inférence. Cette base a été construite en se basant sur notre connaissance de la dynamique du prototype.

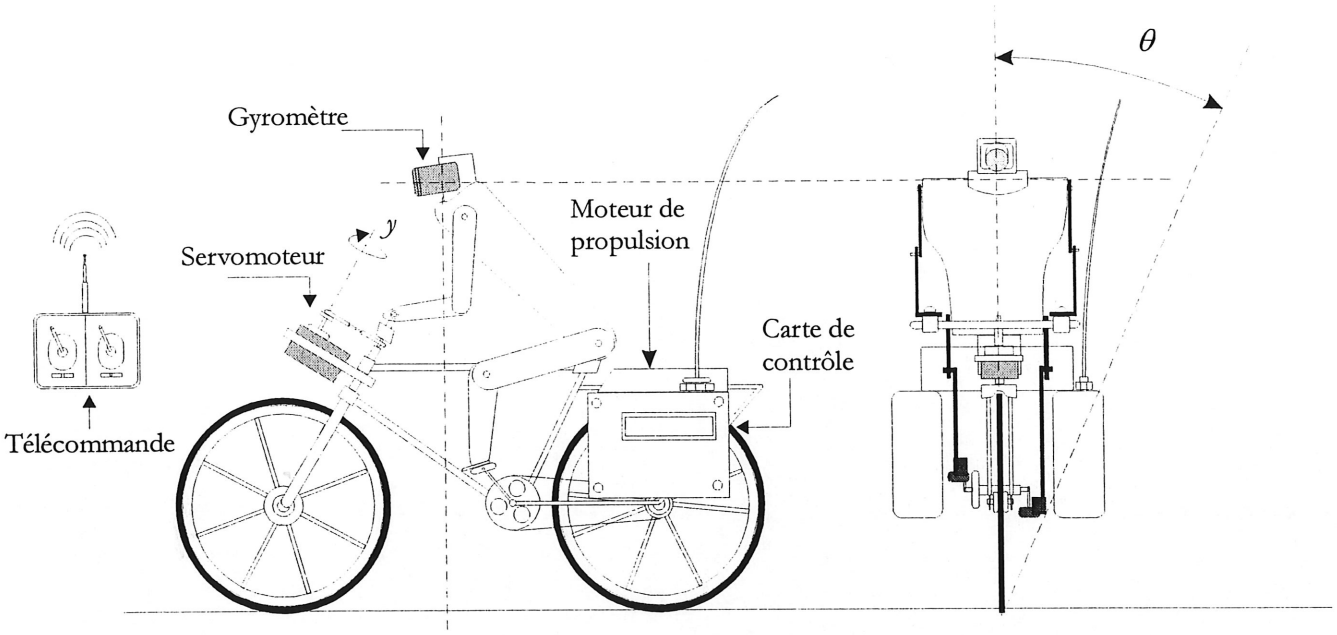
### 3.1 Aspect matériel

Le prototype réalisé est schématisé par la figure 3.1, sa construction, en tube d'acier, lui donne un cadre léger et rigide. Le cycliste est penché vers l'avant, plaçant plus de poids sur la roue avant et donnant au vélo un meilleur contrôle directionnel. Le prototype tel qu'il est construit, possède un centre de gravité fixe.

Le modèle construit contient un capteur principal qui est un gyromètre piézo-électrique monté dans la tête du cycliste et qui permet de mesurer la vitesse angulaire $\dot{\theta}$ entre la verticale et la trajectoire, un capteur optique intégré dans le moteur de propulsion placé à l'arrière du vélo permet de mesurer la vitesse $v$ de déplacement. L'actuateur principal est un servomoteur haute performance placé à l'avant du vélo qui dirige la roue avant. Une télécommande avec deux manches une pour la vitesse et l'autre pour la direction permet de piloter le vélo à distance.

La carte de contrôle est embarquée sur le vélo. Elle est construite autour d'un microcontrolleur 68HC11 de Motorola, cette carte est appelée «Handy board ». C'est une carte commerciale développée au Media Lab du MIT, cette carte correspond aux besoins du projet. Le système de contrôle contient les éléments suivants :

- Un ensemble de batteries contenant 8 cellules AA qui gènere une tension de 9.6 Volts, permettant l'alimentation de la carte de contrôle ainsi que tout les composants du prototype à part le moteur propulsion qui est lui, alimenté par une serie de 4 cellules C au Cadium-Nickel générant une tension de 4.8 Volts.
- Deux entrées analogiques 8 bits, une pour l'intégration du signal gyromètre et l'autre pour la surveillance de l'état des batteries.
- Deux canaux d'entrée PWM pour la mesure des signaux de commande de la direction et de la vitesse provennant du recepteur radio.
- Deux canaux de sortie PWM pour la génération des signaux de commandes pour le servomoteur et le moteur de propulsion.



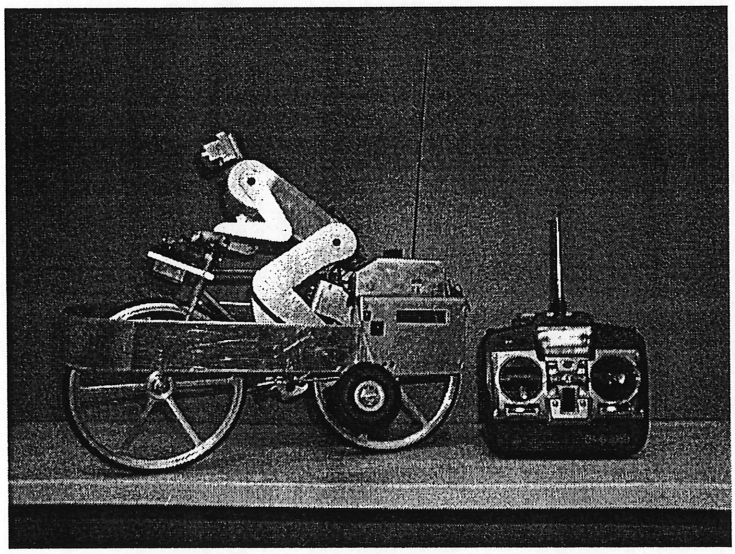



Figure 3.1 Schéma général du prototype.

- Une entrée numérique pour la mesure de la vitesse de déplacement du vélo à travers l'encodeur optique.
- Un lien RS232 qui permet la communication avec un ordinateur PC et l'enregistrement des signaux de fonctionnement pertinents en temps réel.
- Un afficheur à cristaux liquides permet la visualisation des paramètres de pilotage tel que la vitesse par exemple.

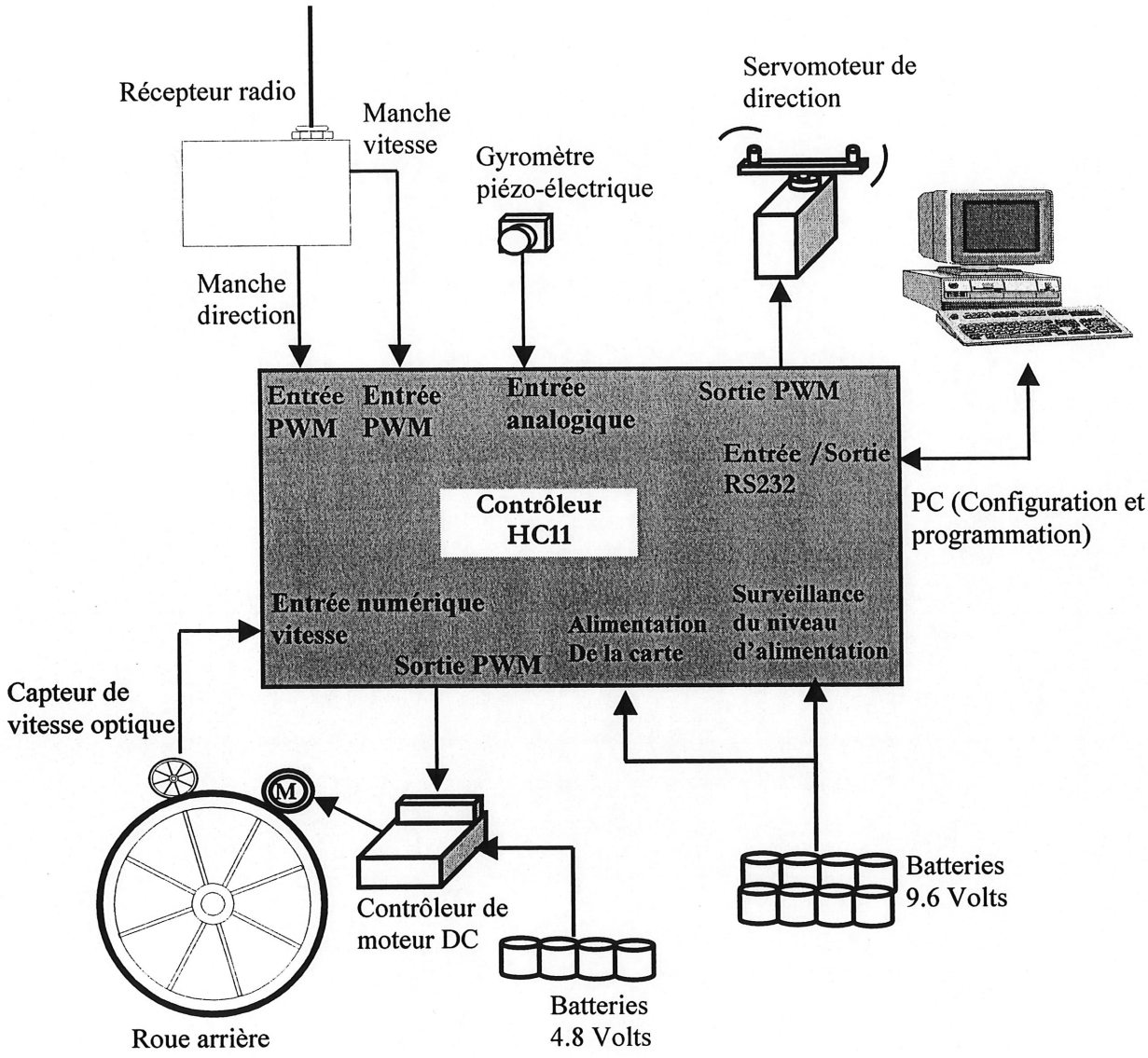



Figure 3.2 Entrées-Sorties de la carte de contrôle

### 3.2 Organisation générale du programme de contrôle

La structure générale du programme de contrôle et les différents systèmes logiciels est représentée par la figure suivante :

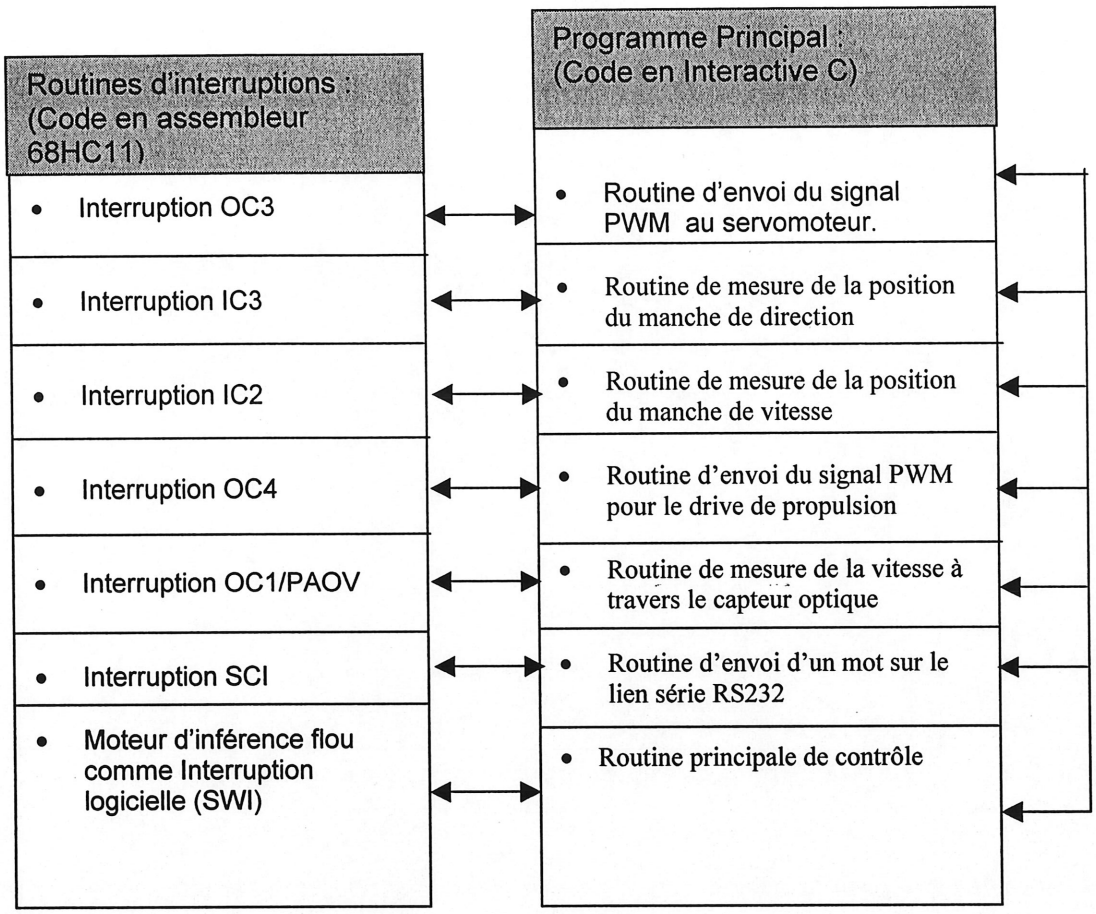



Figure 3.3 Structure générale du programme de contrôle

Le programme principal (codé en interactive C) contient la routine principale de contrôle. Celle ci fait appel aux différentes routines de mesure et de commande d'une façon itérative. Comme représenté par la figure 3.3 chaque routine de mesure ou de commande met à jours ses variables locales en faisant appel à la routine d'interruption qui lui est associée. Ces variables seront transmises par la suite à la routine principale pour le calcule de la commande.

### 3.3 Entrées-sorties du 68HC11

Les ports du 68HC11 sont utilisés de la façon suivante :

| Fonction | Port 68HC11 |
|---|---|
| Entrée PWM(Sortie Rx, Canal de direction) | A0 |
| Entrée PWM (Sortie Rx, Canal de vitesse) | A1 |
| Sortie PWM (connectée à l'entrée du drive de moteur) | A5 |
| Sortie PWM (connectée au servomoteur du guidon) | A4 |
| Entrée capteur optique | A7 |
| Entrée analogique gyromètre | E3 |
| Entrée analogique | E0 |

Tableau 3.1 Entrées-Sorties du 68HC11

### 3.4 Organisation générale des interruptions

« *Interactive C* », un environnement d'exécution interprété comportant un noyau temps réel est utilisé pour la gestion des systèmes logiciels. La plupart des sous systèmes logiciels fonctionnent sous interruption. La table 1.3 décrit toutes les interruptions qui sont mises en œuvre dans le programme, ainsi que leurs contraintes temporelles (période de répétition).

| Interruption | Période de répétition | Traitement |
|---|---|---|
| OC3 | 20 ms | Génération du signal PWM du servo de direction |
| OC4 | 20 ms | Génération du signal PWM pour le moteur de propulsion |
| IC3 | 20 ms | Mesure du signal PWM du manche de direction |
| IC2 | 20 ms | Mesure du signal PWM du manche de la vitesse |
| OC1/PAOV | 20 ms | Mesure de vitesse (optique) |

| | | |
|---|---|---|
| SCI | 1.0 ms | Réception RS232 |

Table 3.2 Organisation des interruptions.

### 3.5 Architecture de contrôle

La structure générale du contrôle est représentée par la figure 3.4. La commande de direction $y_d$ générée par la télécommande ou à travers le lien série, est prise en compte et multipliée par un facteur proportionnel pour la normaliser dans l'univers de référence de la vitesse angulaire. La valeur résultante est considérée comme la consigne de croisement ou décroisement angulaire par rapport à la verticale désirée par le pilote $\dot{\theta}_d$. La vitesse angulaire $\dot{\theta}_m$ est mesurée à travers le gyromètre piézo-électrique. A partir des paramètres $\dot{\theta}_d$ et $\dot{\theta}_m$, on déduit l'erreur $\varepsilon$ et sa dérivée $\dot{\varepsilon}$. Le contrôleur flou prend en compte les paramètres $\varepsilon$ et $\dot{\varepsilon}$ et génère une commande de direction pour le servomoteur. Cette commande est ajustée à travers le gain $g$ de telle façon quelle soit agressive à petite vitesse (de l'ordre de 80 tours/minute) et moins agressive à grande vitesse (de l'ordre de 300 tours/minute).

Les décalages *offset_Dir* et *offset_gyro*, qui sont dus généralement à la dérive thermique, sont calculés en effectuant la moyenne sur les premiers 256 échantillons au début du pilotage. L'intégration du paramètre $\dot{\varepsilon}$ est effectuée à travers l'équation suivante :

$$\varepsilon(i+1) = \varepsilon(i) + \tau \times \dot{\varepsilon}(i) \qquad (3.1)$$

où $i$ est l'indice de l'échantillon, $\varepsilon(0) = 0$ et $\tau$ est la période d'échantillonage qui est de l'ordre de 10ms.

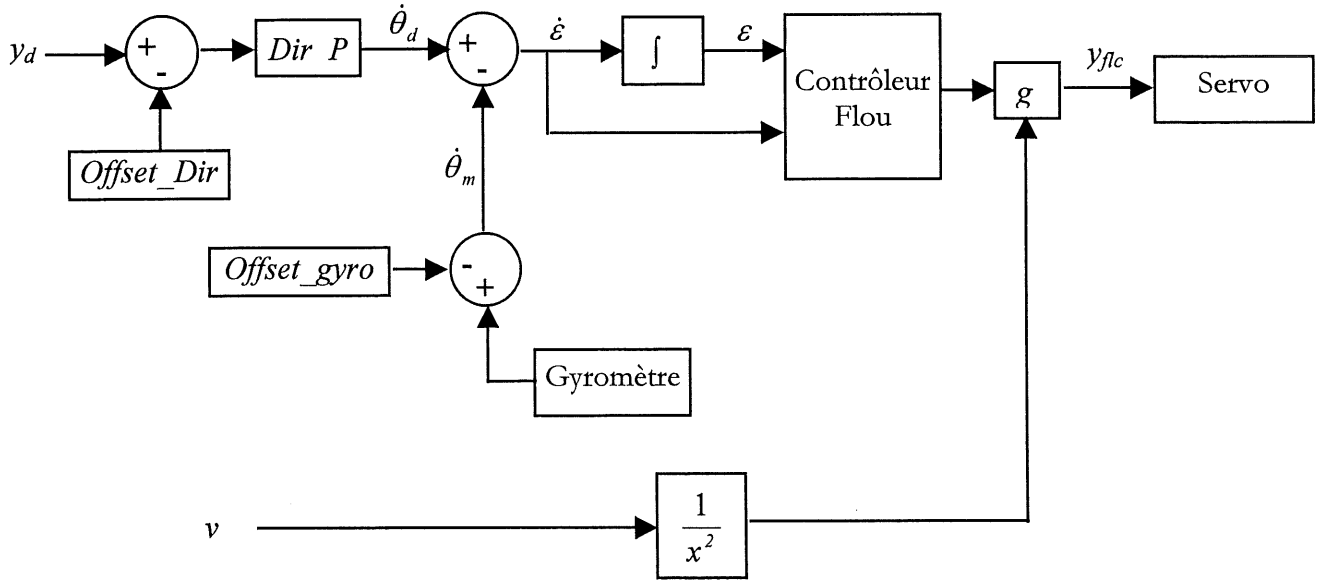



Figure 3.4 Structure générale du contrôleur.

La commande de vitesse est générée à travers le manche de direction ou le lien série et elle est directement envoyée au moteur de propulsion. La sortie du contrôleur flou est pondérée au gain $g$ qui est inversement proportionel au carré de la vitesse mesurée dans le but d'avoir une plage d'action de la commande sur le guidon qui est petite à grande vitesse, et une plage d'action plus large à petite vitesse.

### 3.6 Description du contrôleur flou

Un contrôleur flou de type Sugeno est utilisé pour la génération de la commande de direction en fonction des paramètres $\dot{\varepsilon}$ et $\varepsilon$. L'inférence utilisée est de type min-max. Le schéma général du contrôleur est représenté par la figure 3.5.

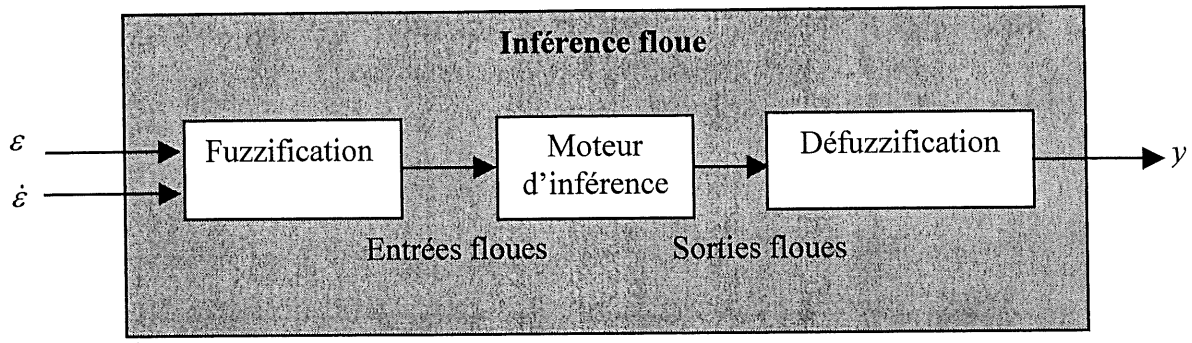



Figure 3.5 Structure du contrôleur flou.

La fonction de fuzzification produit, à partir des variables d'entrées $\dot{\varepsilon}$ et $\varepsilon$, un ensemble d'entrées floues après les avoir mises à l'échelle dans un format de 8 bits non signés (compris entre 0x00 et 0xff). Elle assigne à chaque variable d'entrée un degré d'appartenance à chaque fonction d'appartenance. Les fonctions d'appartenance utilisées sont de type triangulaire comme le représente la figure suivante:

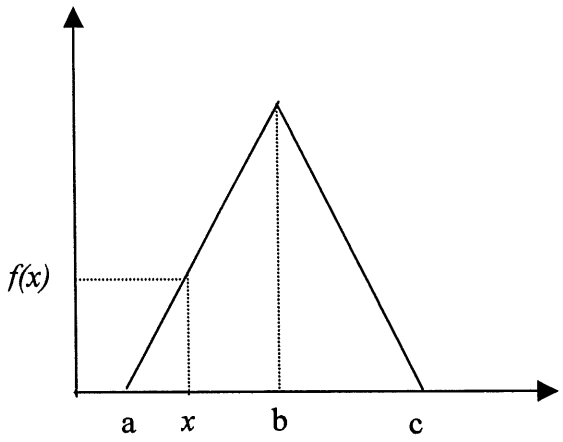



Figure 3.6 Fonction d'appartenance triangulaire

Dans la figure 3.6 : *a* et *c* définissent la position des extrémités de la fonction sur l'axe des abscisses, *b* définit la position du sommet sur le même axe et *f(x)* le degré d'appartenance d'une mesure *x*.

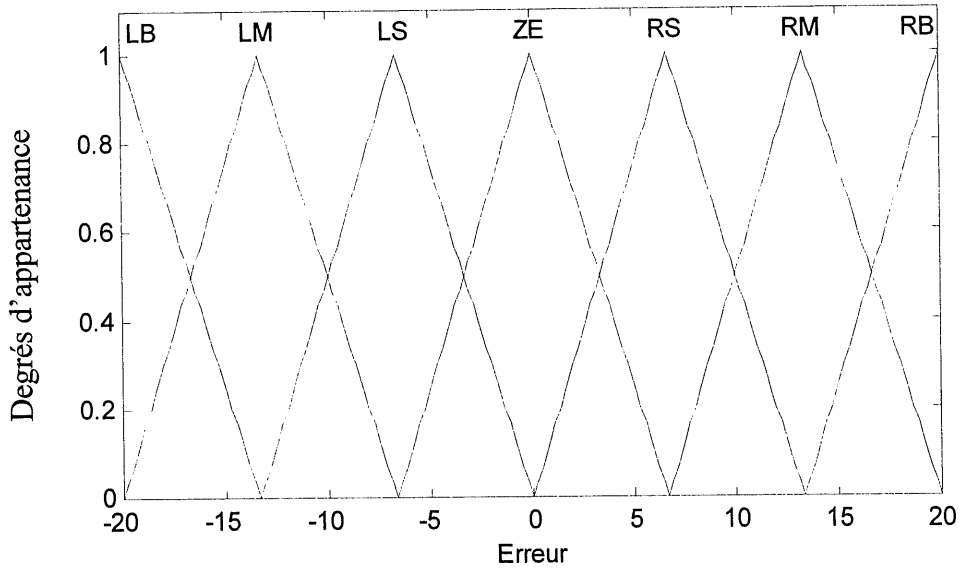



Figure 3.7 Définition des fonctions d'appartenance pour l'entrée $\varepsilon$

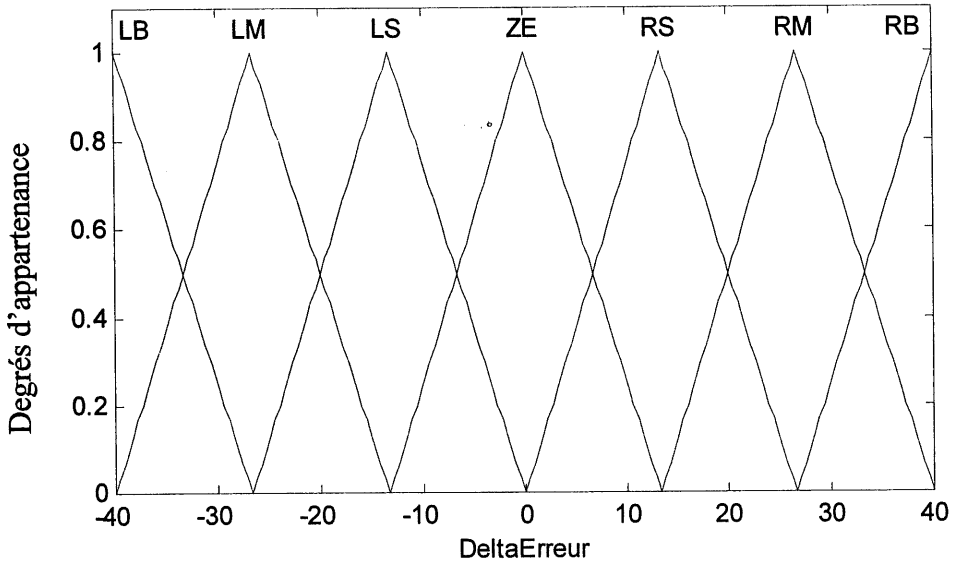



Figure 3.8 Définition des fonctions d'appartenance pour l'entrée $\dot{\varepsilon}$

Comme le montre les figures 3.7 et 3.8, nous avons défini 7 fonctions d'appartenance pour chaque entrées $\dot{\varepsilon}$ et $\varepsilon$. Où l'étiquette LB signifie « *Left Big* », l'étiquette RS signifie « *Right small* » et ainsi de suite.

A partir des paramètres *a, b* et *c,* on calcule les deux pentes $s_1$ et $s_2$ pour chaque fonction d'appartenance en utilisant les formules suivantes :

$$s_1 = \frac{2^n - 1}{b - a}; \qquad s_2 = \frac{2^n - 1}{c - b} \tag{3.2}$$

Où $n$ = 8 bits.

Selon cette représentation, chaque fonction d'appartenance est définie par quatre mots de 8 bits qui sont : *a*, *b*, $s_1$ et $s_2$. Ces paramètres sont stockés en mémoire sous forme de table comme le montre la figure suivante :

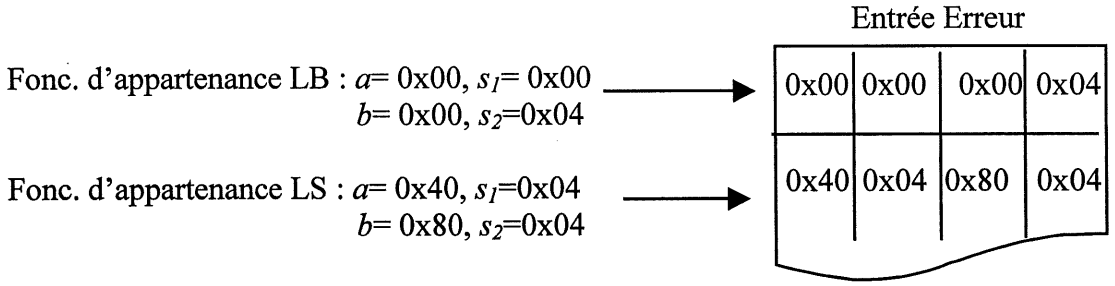



Figure 3.9 : Format des fonctions d'appartenance.

A partir de la représentation citée ci dessus, le moteur d'inférence calcule le degré d'appartenance d'une entrée quelconque *x* en utilisant les équations suivantes :

$x < a$ : degré = 0;

$a \leq x < b$ : degré = $(x - a) \times (s_1)$;

$x \geq b$ : degré = $255-((x - b) \times (s_2))$.

Pour la sortie commande de direction, nous avons définis 7 fonctions d'appartenance de type singleton comme le montre la figure 3.10. Où les étiquettes LVS et RVS signifient « *Left Very Small* » et « *Right Very Small* ».

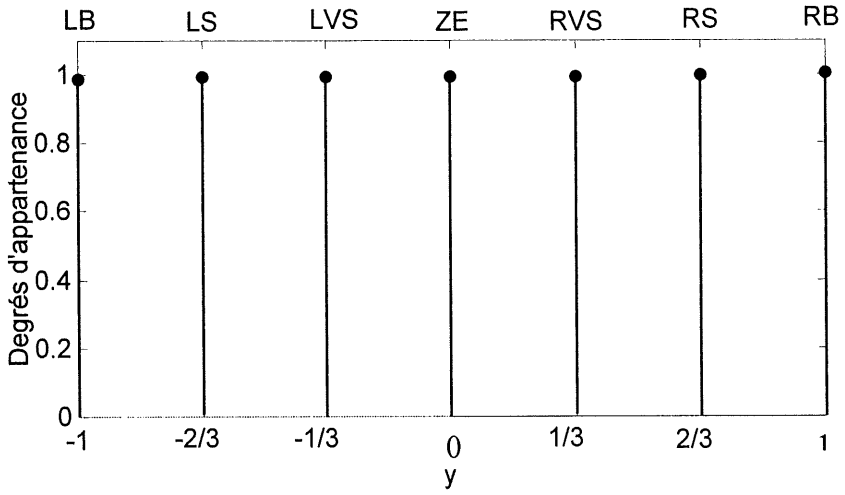



Figure 3.10 : Définition des fonctions d'appartenance pour la sortie *y*.

Le moteur d'inférence prend en considération les entrées floues calculées durant l'étape de fuzzification et génère une sortie floue, en évaluant un ensemble de règles prédéfini et cela en utilisant une inférence de type min-max. Les règles sont exprimées dans un format « *si-alors* », si les deux entrées sont vraies (appelées aussi antécédents) , alors une fonction floue de sortie (appelée aussi conséquence) est exécutée au minimum des degrés d'appartenance des antécédents. Pour les règles avec la même conséquence, le moteur d'inférence va choisir celle qui possède le plus grand degré (voir figure 3.11).

### 3.6.1 Source de dérivation des règles pour le contrôleur flou

Il existe plusieurs méthodes pour dériver les règles d'un contrôleur flou. Les règles peuvent être dérivées à partir de l'expertise qu'on possède du processus, ou bien à partir de l'observation du processus, ou par apprentissage en utilisant des méthodes telles que les réseaux neuroflous ou l'apprentissage par renforcement.

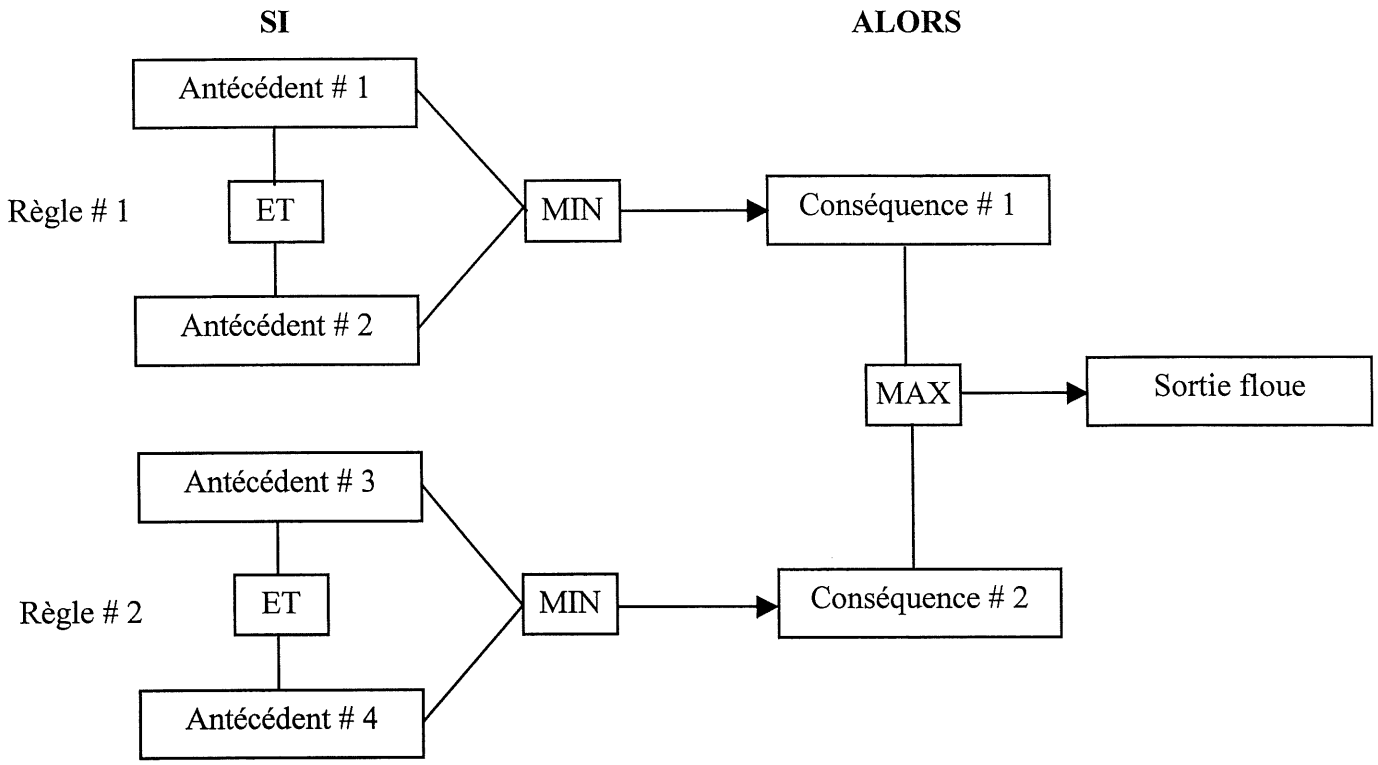



Figure 3.11 : Moteur d'inférence de type min-max.

Pour dériver les règles de notre contrôleur flou nous avons commencé par reproduire un simple contrôleur proportionnel dérivé flou. La table 3.3 représente l'ensemble des règles pour ce type de contrôleur. La dérivation des règles dans ce cas, se fait d'une manière simple et intuitive en analysant, cellule par cellule, la position du système $\varepsilon$ et son gradient $\dot{\varepsilon}$ par rapport à la consigne de vitesse angulaire $\dot{\theta}_d$, donnée par le pilote. Par exemple, si l'erreur $\varepsilon$ est « *Right Small* » et sa dérivée est « *Right Small* » aussi, dans ce cas le système est légèrement en dessous de sa consigne et s'en éloigne lentement. Par conséquent, la commande doit être positive. Cependant, une commande trop forte risque de provoquer un dépassement important ce qui peut expliquer le choix de la valeur « *Right Small* » de $y$ .

| Direction $y$ | | Erreur | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | LB | LM | LS | ZE | RS | RM | RB |
| Dérivée de l'erreur | LB | LB | LB | LB | LB | LS | LVS | ZE |
| | LM | LB | LB | LB | LB | LVS | ZE | RB |
| | LS | LB | LB | LB | LS | ZE | RS | RB |
| | ZE | LB | LB | LVS | ZE | RS | RB | RB |
| | RS | LB | LVS | ZE | RS | RS | RB | RB |
| | RM | LS | ZE | RVS | RS | RB | RB | RB |
| | RB | ZE | RVS | RS | RB | RB | RB | RB |

Table 3.3 : Base des règles pour un contrôleur proportionnel dérivé flou.

A partir de la base des règles du contrôleur proportionnel dérivé décrit précédemment et en se basant sur l'observation de la conduite manuelle du vélo, nous avons établis 49 règles (7 étiquettes pour $\varepsilon$ × 7 étiquettes pour $\dot{\varepsilon}$ ) pour générer la commande de direction adéquate qui permet au vélo de maintenir son équilibre et de prévenir les chutes. L'une de ces règles est que pour prévenir une chute la fourche du guidon doit tourner dans le sens de l'angle d'inclinaison. Ces règles représentent la base de notre expertise. La table 3.4 résume la base des règles.

| Direction $y$ | | Erreur | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | LB | LM | LS | ZE | RS | RM | RB |
| Dérivée de l'erreur | LB | LB | LB | LB | LB | LVS | LVS | ZE |
| | LM | LB | LB | LB | LB | LVS | ZE | RB |
| | LS | LB | LB | LB | LS | ZE | RS | RB |
| | ZE | LB | LB | LS | ZE | RS | RB | RB |
| | RS | LB | LS | ZE | RS | RB | RB | RB |
| | RM | LB | ZE | RVS | RB | RB | RB | RB |
| | RB | ZE | RVS | RVS | RB | RB | RB | RB |

Table 3.4 : Base des règles pour le contrôle d'équilibre.

Des exemples de règles ainsi qu'une représentation de l'inférence de la sortie à partir des règles sont représentés par la figure 3.15. La surface de contrôle issue de ces règles est représentée par la figure 3.12. On peut bien voir que la commande de direction est agressive aux bornes de l'intervalle de l'angle et de la vitesse angulaire là ou le vélo tend vers une chute ou un déséquilibre.

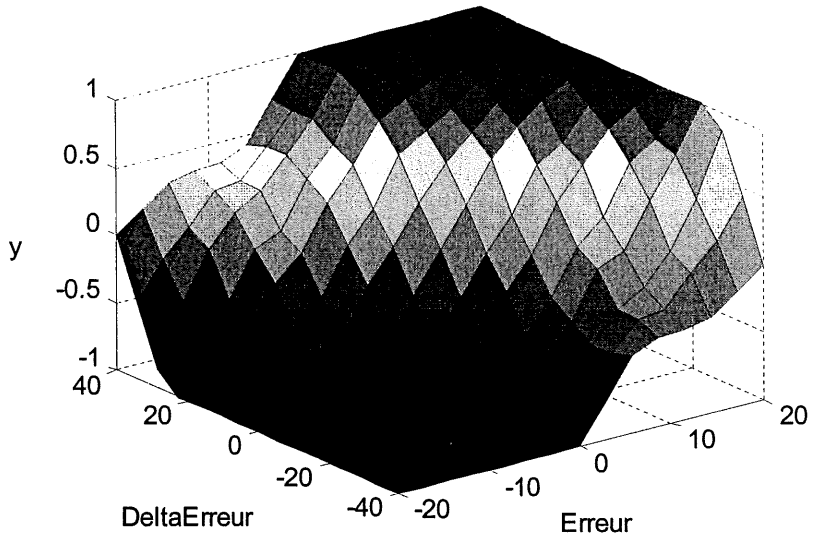



Figure 3.12 : Surface de contrôle $y=f(\varepsilon, \dot{\varepsilon})$.

Toutes les règles (antécédant et conséquence) sont codées dans un format 8 bits non signé. Comme le montre la figure 3.13, le premier bit est pour désigner si le mot est un antécédant ou une conséquence. Pour l'antécédant les bits 4 et 5 désignent l'entrée assignée (0b00 pour l'entrée angle et 0b01 pour l'entrée dérivée de l'erreur). Les trois derniers bits désigne quelle fonction d'appartenance est utilisée. Pour la partie conséquence les trois derniers bits désignent le singleton utilisé.

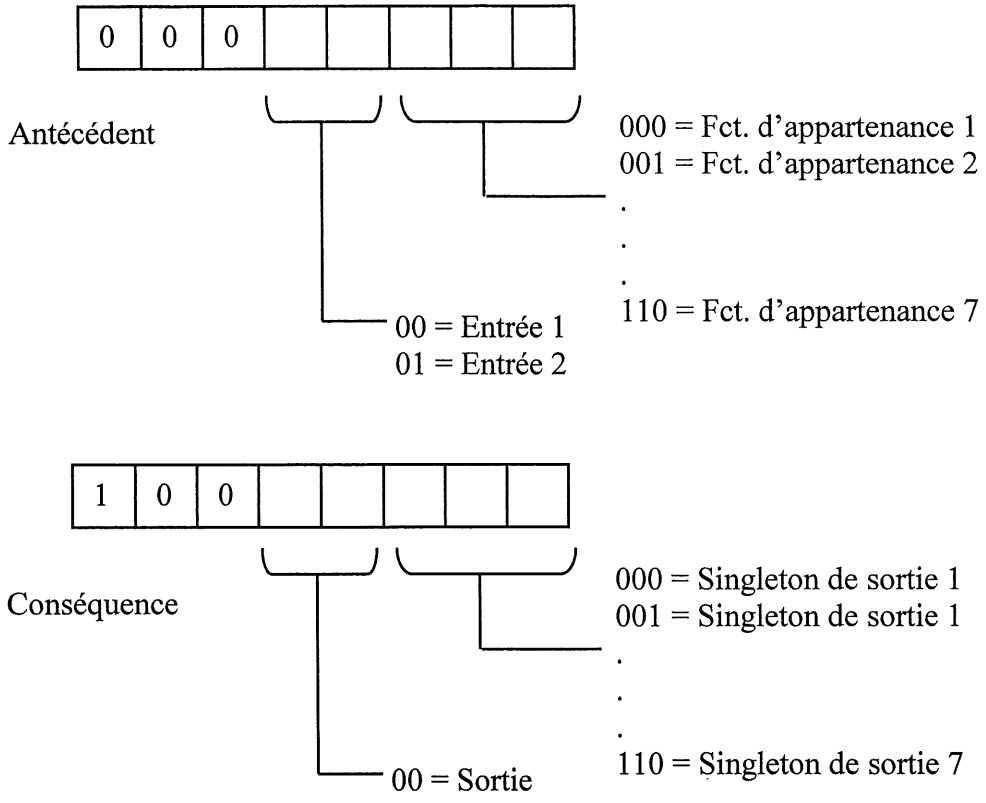



Figure 3.13 : Format des antécédents et des conséquences.

Un exemple de représentation de la table de règles avec les fonctions d'appartenance de l'entrée 1 et 2 et les singleton de la sortie est montré par la figure 3.14.

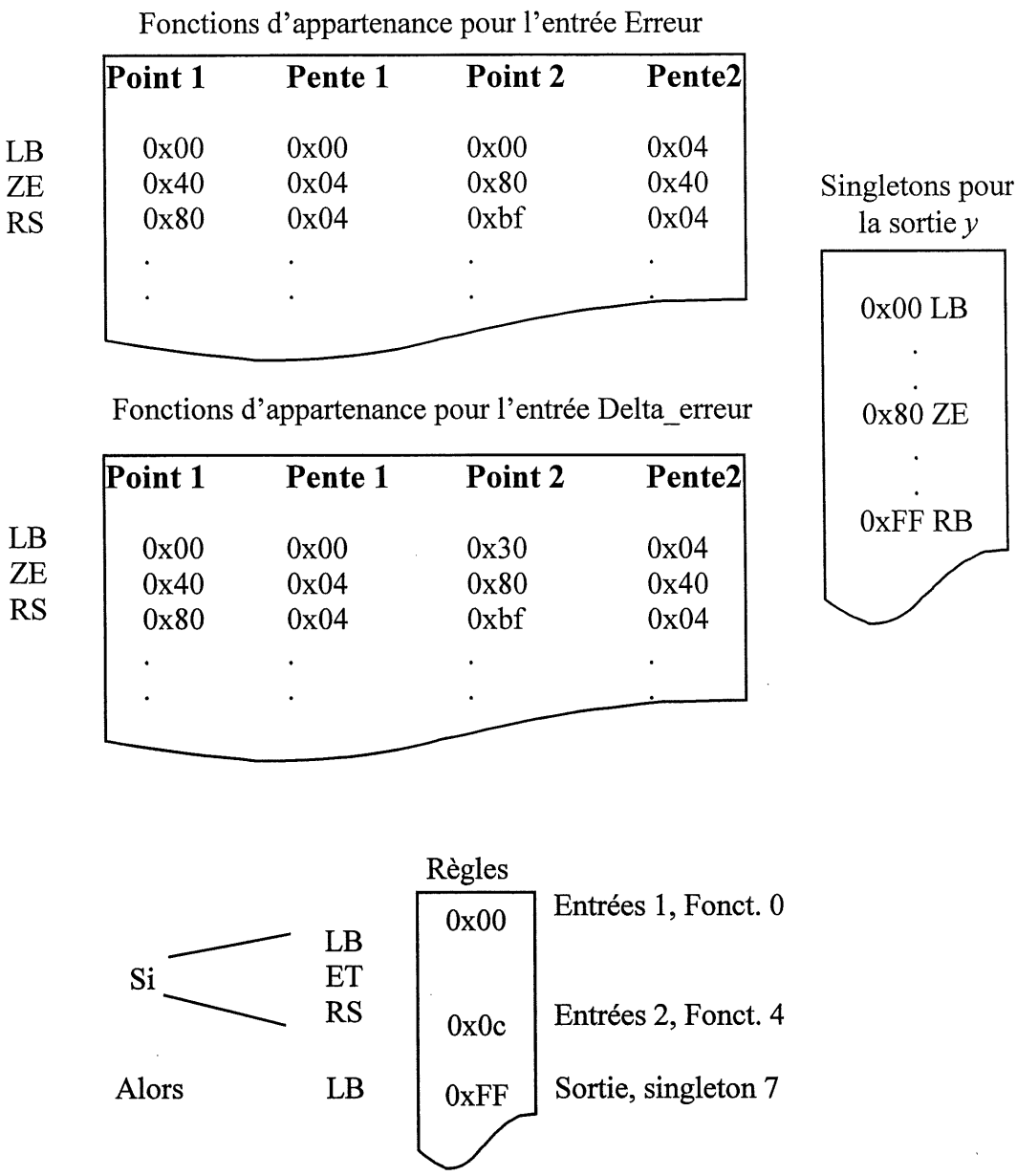



Figure 3.14: Exemple de représentation de la table des règles avec les fonctions d'appartenance d'entrée et les singletons de sortie.

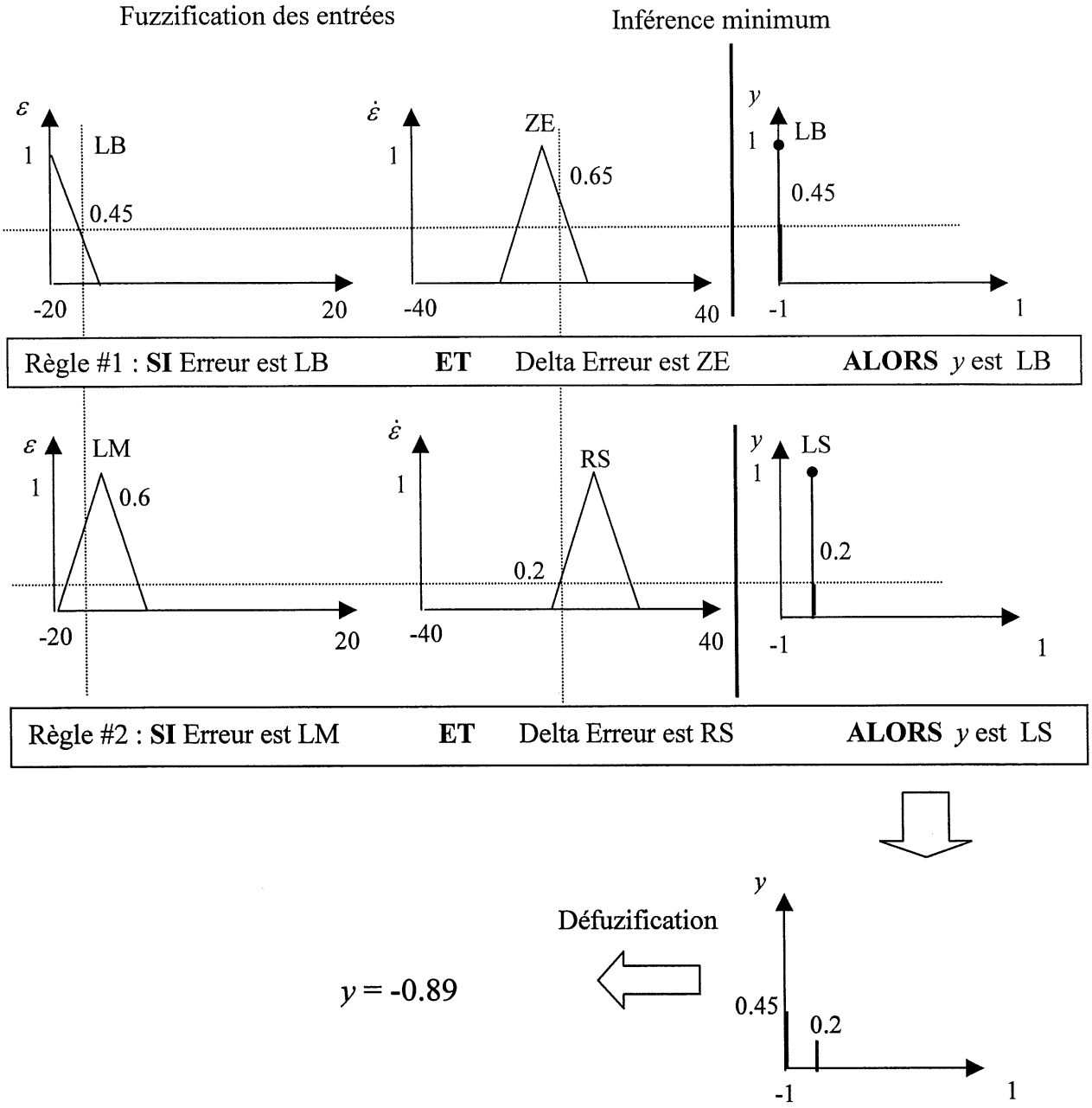



Figure 3.15 Exemple d'inférence de la commande de direction $y$ à travers le contrôleur flou.

La dernière tâche du contrôleur est de traduire les sorties floues en une sortie de commande réelle pour le servomoteur, c'est l'étape de défuzzification. La méthode de défuzzification utilisée est celle du centre de gravité, comme décrit ci-dessous les valeurs des sorties floues sont pondérées au singleton.

$$y = \frac{\sum_{i=1}^{m} (F_i \times S_i)}{\sum_{i=1}^{m} F_i} \tag{3.5}$$

Où $i$=1,$m$ est l'indice de règles, $F_i$ représente les sorties floues et $S_i$ les singletons.

Un exemple de l'inférence de la commande de direction $y$ à travers le contrôleur flou est représenté par la figure 3.15.

La commande de direction $y$ calculée par le contrôleur flou est comprise dans l'intervalle [-1,1]. Comme nous l'avons déjà mentionné, cette commande est pondérée par le gain $g$, qui est inversement proportionnel au carré de la vitesse de déplacement du vélo. De telle façon qu'à grande vitesse nous avons des commandes de direction assez faibles avec une valeur du gain $g$ qui est de l'orde de 10 pour la vitesse la plus grande et à vitesse faible nous avons des commandes assez fortes avec une valeur du gain de l'ordre de 35 pour la vitesse la plus élevée. La stabilité observé à grande vitesse est due à la proportionnalité de la force de basculement appliquée au vélo à l'accélération radiale subie par le vélo. A position du guidon égale (donc à trajectoire égale), cette accélération radiale est proportionnelle au carré de la vitesse sur la trajectoire. Ainsi pour appliquer une force de basculement constante, indépendamment de la vitesse sur la trajectoire, il faut diviser la commande $y$ par le carré de la vitesse. En d'autres termes, le gain $g$ a pour effet, la contraction ou l'expansion de l'univers de référence de la commande de direction $y$. Dans le cas où $g = 1$, aucun effet n'est produit sur l'univers de référence. Si $g < 1$, nous allons obtenir une contraction de l'univers de référence. Dans le cas où $g > 1$, une expansion de l'univers de référence est produite. Pour ce qui est de notre contrôleur, il y a toujours une expansion vu que $g$ est toujours supérieur à 1.

## 3.7 Conduite expérimentale de la bicyclette

Pour observer le fonctionnement du contrôleur flou avec la base de règles établi dans la section 3.5, nous avons effectués des expériences de conduite de la bicyclette avec différentes manœuvres. Celles-ci illustrent la stabilité du système (maintien de l'équilibre) dans différents cas de figure : les deux premiers essais représentent une conduite en ligne rectiligne à grande et faible vitesse, et les deux derniers représentent une conduite en ligne courbe à grande et faible vitesse. Durant les expériences de conduite, nous avons effectué un enregistrement en temps réel des signaux que nous avons jugé pertinents pour l'observation du fonctionnement du contrôleur.

### 3.7.1 Conduite en ligne rectiligne à grande vitesse

La figure 3.16 représente un exemple de conduite de la bicyclette en ligne rectiligne avec une vitesse de déplacement quasi constante de l'ordre de 250 tours/minute (représentée par le graphique 4 de la figure). Dans ce cas, nous avons donné une commande de direction constante de 0 degré à travers la télécommande (graphique 2). La zone 1 de la figure est une zone de stabilité qui représente le maintien de l'équilibre après le démarrage, en d'autres termes, cette zone représente le régime permanent pour le système après la période transitoire du démarrage. Pour voir le fonctionnement du contrôleur, nous avons essayé de déséquilibrer le vélo en donnant un coup sur le corps du cycliste ce qui est représenté dans la zone de perturbation par le graphique 1 qui illustre le changement de la vitesse angulaire par rapport à la verticale donné par le gyromètre, le contrôleur réagit avec la commande adéquate sur le servomoteur, celle-ci est comprise dans l'intervalle [-10°, 10°], car vue que la vitesse est assez grande le gain de sortie est faible, de l'ordre de 10, ce qui illustré par le graphique 3 de la figure. Suite à cela le vélo reprends son équilibre (son régime permanent) ce qui est démontré dans la troisième zone de la figure.

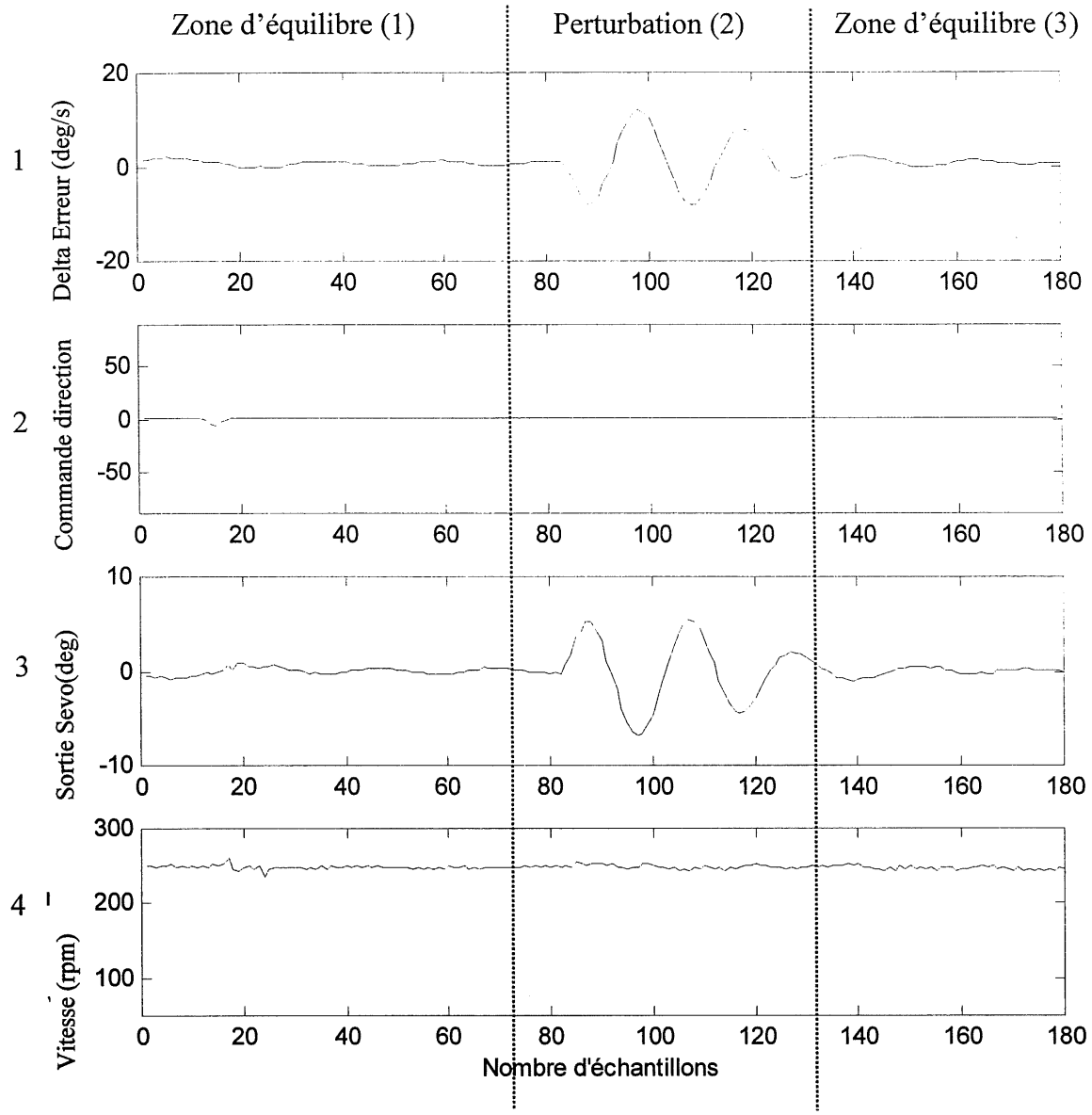



Figure 3.16 Enregistrement de signaux pour une trajectoire rectiligne à grande vitesse.

### 3.7.2 Conduite en ligne rectiligne à faible vitesse

La figure 3.17 illustre la même manœuvre que dans la dernière section, c'est à dire, une commande de direction de 0 degré générée à travers la télécommande (représenté par le graphique 2 de la figure), sauf que dans ce cas la vitesse de déplacement est plus faible, elle est de l'ordre de 100 tours/minute (représenté par le graphique 4 de la figure). De même, pour observer le fonctionnement du contrôleur, nous avons essayé de perturber le vélo de sa

position d'équilibre en régime permanent qui vient après la période de démarrage, cette perturbation est illustrée par le graphique 1 de la figure, qui représente le changement de la vitesse angulaire mesuré par le gyromètre. La commande générée par le contrôleur pour reprendre l'équilibre est représentée par le graphique 3. Puisque la vitesse est faible, le gain de sortie dans ce cas est important (de l'ordre de 30) et donc la commande envoyée au servomoteur est comprise dans l'intervalle [-30°,30°]. Dans ce cas, on peut remarquer que le temps de réponse du système est grand par rapport à une conduite à vitesse élevée. Ceci s'explique par l'instabilité en conduite à faible vitesse qui est dû essentiellement, au « slew rate » limité du servomoteur, qui aux faibles vitesses doit avoir des déplacement plus rapides et plus important qu'aux hautes vitesses, la qualité et la performance du servomoteur utilisé sont critiques. La troisième zone de la figure représente une reprise de l'équilibre.

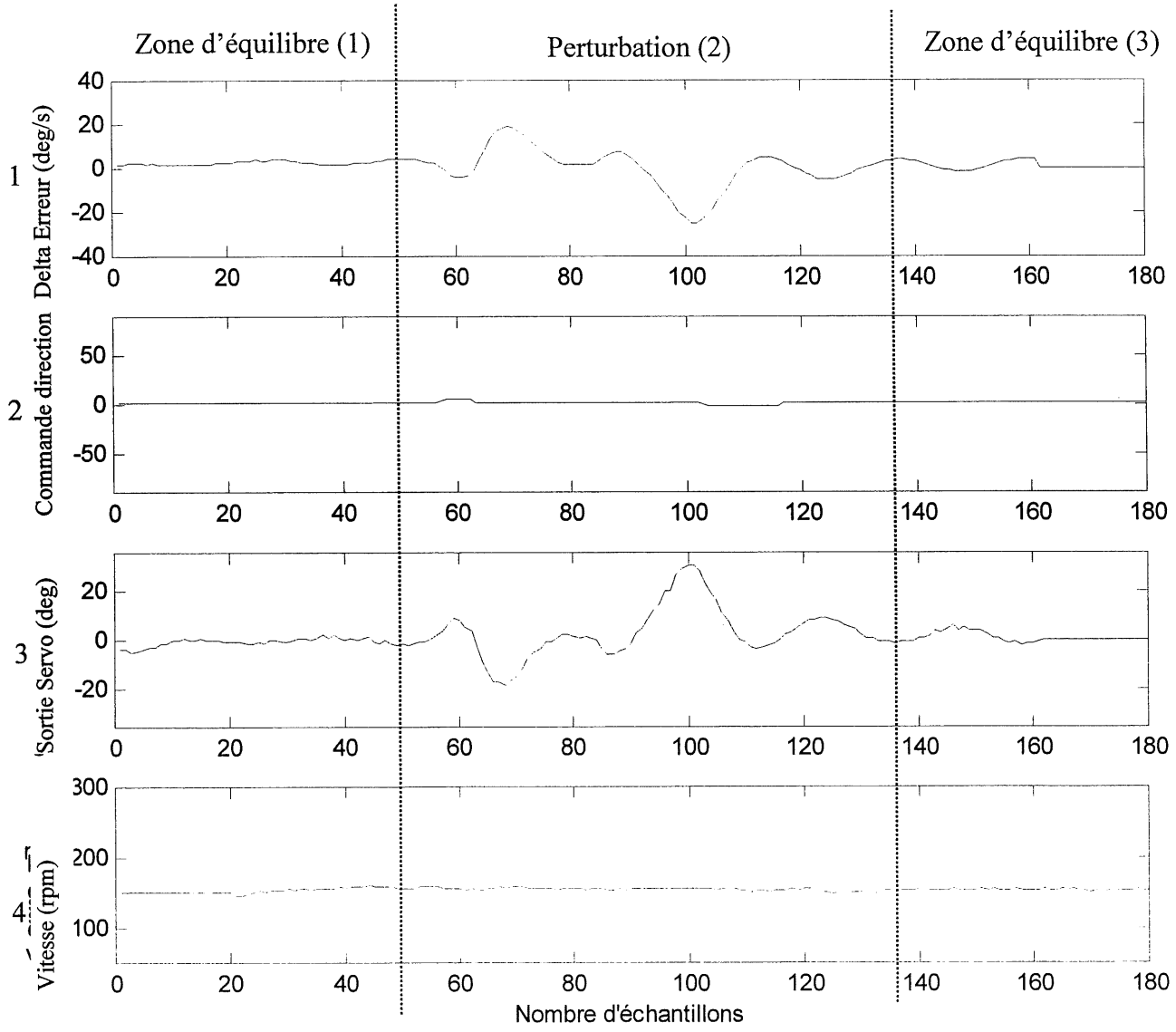



Figure 3.17 Enregistrement des signaux pour une trajectoire rectiligne à vitesse faible.

### 3.7.3 Conduite en ligne courbe à forte vitesse

Dans les deux essais précédents, la commande de direction envoyé à la bicyclette à travers la télécommande était nulle, ce qui correspond à une trajectoire rectiligne. Dans les deux manœuvres suivantes, la consigne de direction est non nulle, sa valeur est de l'ordre de 10 degrés (converti en taux de roulis), ce qui correspond à une trajectoire courbe ou à un virage. Le graphique 4 de la figure 3.18 montre un essai de conduite à vitesse élevée ( 250 tours/minute). Après la période de stabilisation au démarrage, représentée par la zone 1 de la figure, nous avons perturber la bicyclette sur sa trajectoire, et cela en donnant un coup au

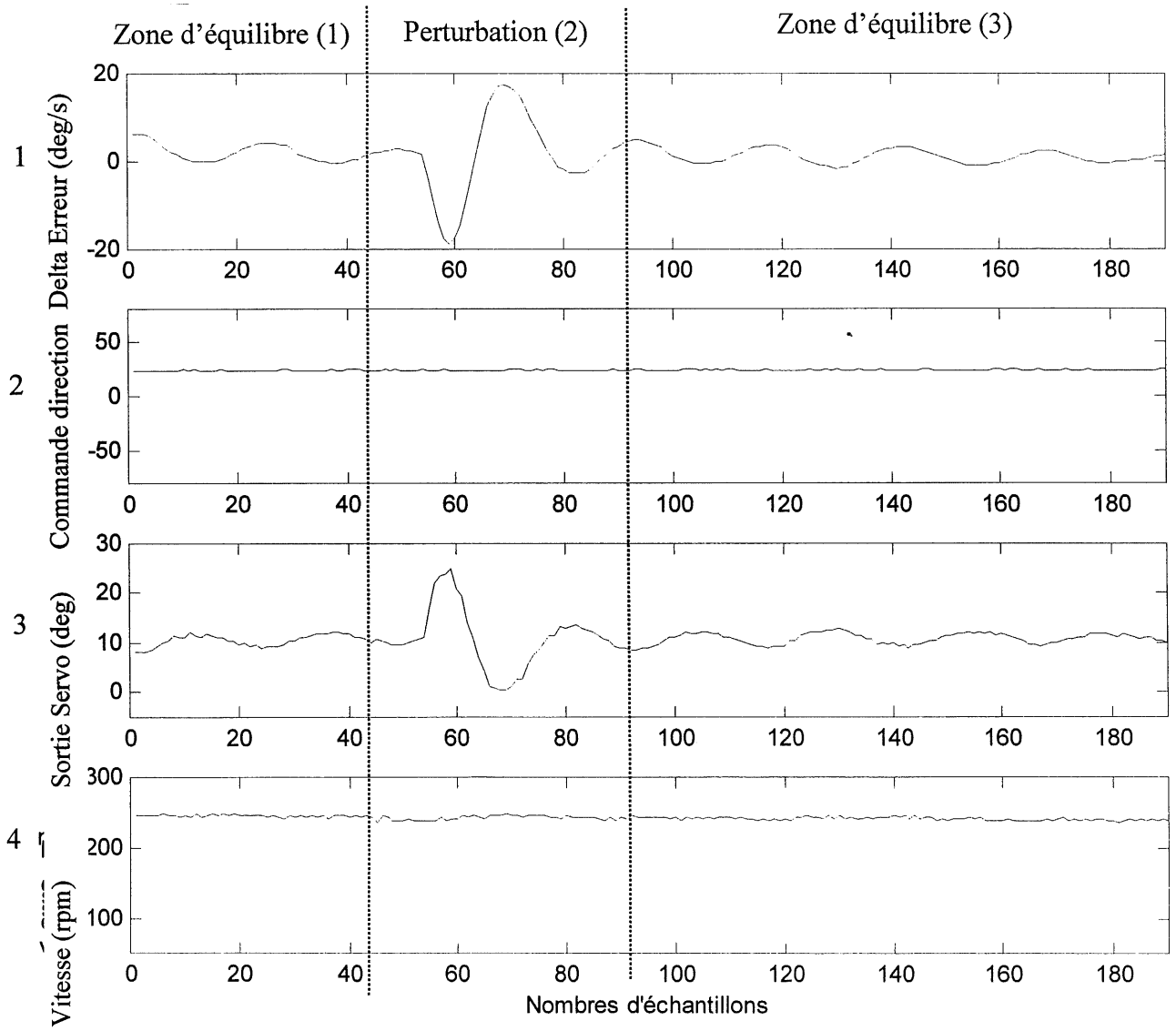



Figure 3.18 Enregistrement de signaux pour une trajectoire courbe à vitesse élevée

corps du cycliste. La commande envoyée au servomoteur pour reprendre l'équilibre est montrée par le graphique 3 de la figure, elle est comprise dans l'intervalle [5°,25°], elle est centrée sur la consigne de direction (10 degrés). Dans ce cas, on a un gain de sortie faible, qui est de l'ordre de 15. Dans ce cas aussi on bien voir qu'on a un temps de réponse faible, dû à une meilleur stabilité pendant le guidage à grande vitesse. Après la zone de perturbation, le vélo reprend sa position originale d'équilibre centrée autour de la commande de direction.

### 3.7.4 Conduite en ligne courbe à faible vitesse

De la même manière que l'essai précédent, la figure 3.19 représente un essai de conduite en ligne courbe avec une commande de direction de l'ordre de 10 degrés, envoyée toujours à travers la télécommande, mais dans ce cas à vitesse faible autour de 150 tours/minute comme le montre le graphique 4 de la figure. Avec la même manœuvre nous avons déstabilisés la bicyclette de sa consigne, après la période de démarrage. Le graphique 1 de la figure illustre le changement de la vitesse angulaire par rapport à la position initiale, mesuré par le gyromètre suite à cette perturbation (zone 2 de la figure). Dans ce cas, la commande sur le guidon pour la reprise de l'équilibre est plus agressive qu'à grande vitesse et elle est comprise dans l'intervalle [-5°,35 °], elle est toujours centrée par rapport à la consigne qui est de 10 degrés, comme montré par le graphique 3 de la figure. L'agressivité de la commande de sortie est due au gain de sortie, qui est de 35 calculé par le contrôleur pour reprendre l'équilibre. On remarque aussi que dans ce cas, le temps de réponse est plus grand, ce qui est dû, comme on l'a déjà expliqué, à l'instabilité dans la conduite pendant les déplacements à vitesse faible. Suite à cette période de perturbation, la bicyclette reprend son équilibre autour de sa position d'équilibre initial, ce qui est représenté par la zone 3 de la figure.

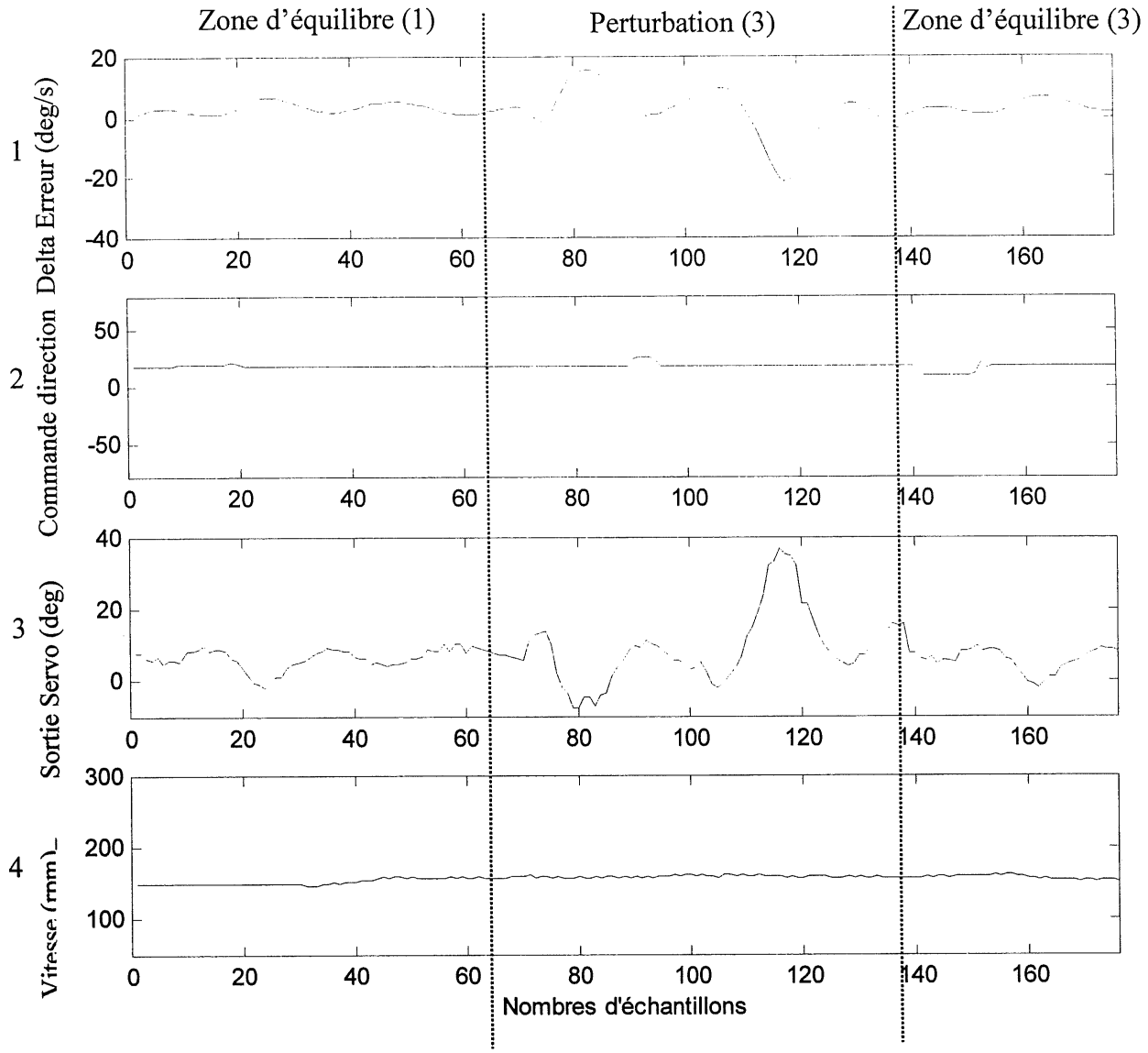



Figure 3.19 Enregistrement de signaux pour une trajectoire courbe à faible vitesse

Les exemples de conduite cités précédemment, montrent que le contrôleur flou conçu permet le pilotage supervisé de la bicyclette dans différents cas de figure mais pas dans tous les cas, par exemple le prototype tel qu'il est conçu actuellement ne peut pas effectuer des virages trop serrés (des virages au delà de 30 degrés par rapport à la verticale). Pour faire de telles manœuvres, un servomoteur avec des meilleures performances est indispensable. En particulier, une caractéristique primordiale est le « slew rate » (vitesse de déplacement max) du servomoteur.

La question qui se pose maintenant, pourquoi l'utilisation d'un algorithme de contrôle basé sur la logique floue alors qu'un contrôleur proportionnel dérivé avec glissement du gain pourrait faire la même tâche? La réponse est que contrairement aux contrôleurs conventionnels, les non-linéarités et les exceptions peuvent être incluses de façon naturelle dans la structure internes des contrôleurs flous. La robustesse du contrôle flou à la variabilité du comportement dynamique du système (masse/répartition des masses), et donc la possibilité d'avoir un contrôle efficace, pour contrôler un système inconnu à priori ou variable, est primordiale dans le cas d'un système naturellement instable, qui ne peut donc exister que s'il est contrôlé, et dont il est impossible d'identifier le comportement en boucle ouverte.

Cet ensemble de réglages peut être implanté d'une façon naturelle dans les règles des contrôleurs flous. En plus, d'autres types de non-linéarités locales peuvent être prises en compte facilement vu qu'un contrôleur flou peut être considéré comme une association non linéaire entre la sortie du processus et son entrée. Les contrôleurs flous sont aussi utilisés pour exprimer les objectifs du contrôle de manière qualitative, comme le cas de notre application alors qu'avec les méthodes du contrôle conventionnel, il est nécessaire de quantifier nos objectifs par des valeurs numériques précises ou par des expressions mathématiques.

L'autre avantage des contrôleurs flous c'est qu'ils offrent un moyen systématique et efficace pour la réutilisation des connaissances des experts humains ou des connaissances tirées de l'observation du processus, exprimées de façon linguistique. Ils s'adaptent bien au cas des systèmes mal modélisables tel que notre problème. L'autre mérite des contrôleurs

flous c'est qu'ils sont faciles à comprendre et simples à implanter, étant donné qu'ils émulent les stratégies de contrôle de l'humain. D'où un coût de développement avantageux.

La base de connaissance de notre contrôleur flou a été construite à partir de la base de règles d'un contrôleur proportionnel dérivé flou. L'approche adoptée pour atteindre nos objectifs de contrôle est de modifier et régler la structure de cette base en observant le comportement du prototype en faisant des manœuvres de conduite. Il existe d'autres approches, plus systématiques pour faire l'adaptation des contrôleurs flous. Le but du contrôle flou adaptatif est de trouver un contrôleur capable de s'autoconfigurer pour s'adapter aux changements des paramètres du processus et aux situations nouvelles et inconnues. Il existe plusieurs approches pour rendre un contrôleur flou adaptatif. Elles sont, pour la plupart, inspirées soit des méthodes du contrôle adaptatif conventionnel soit des techniques d'apprentissage des systèmes neuro-flous.

Bien qu'à priori l'adaptation peut porter aussi bien sur la structure que sur les paramètres du contrôleur, la plupart des algorithmes adaptatifs sont des algorithmes d'adaptation paramétriques. Un contrôleur flou adaptatif est en général composé de trois blocs principaux : un contrôleur flou classique de bas niveau, un moniteur de performance pour estimer les paramètres du processus et/ou pour évaluer le degré de succès de la configuration de la structure actuelle par rapport aux objectifs de contrôle et à la fin un bloc d'adaptation qui corrige les paramètres du contrôleur afin de minimiser le degré d'insatisfaction du moniteur de performance. Ce type d'approches d'adaptation systématique, peuvent être testées sur notre application dans le cadre de projets futurs, pour atteindre un niveau de performance meilleur et une structure de contrôle adaptative en fonction des changements effectués sur le prototype.

# CONCLUSION

Dans la première partie de ce mémoire nous avons introduit la logique floue et discuté les concepts de base rattachés à cette méthode. Dans la deuxième partie nous avons présenté la structure d'une commande floue et son application pour les systèmes asservis. La troisième partie du mémoire est consacrée à la présentation de la mise en œuvre d'un contrôleur flou sur une plate forme matérielle. Celle ci est un prototype de bicyclette télécommandée, développé au département de génie électrique et de génie informatique.

Le contrôleur mis en œuvre permet la supervision en temps réel du guidage d'un opérateur humain et gardé l'équilibre du vélo en tout temps, avec une emphase sur le maintien de la trajectoire voulue par le pilote à travers la télécommande ou un ordinateur. L 'avantage de l'utilisation d'une structure de contrôle basée sur la logique floue est sa capacité à reproduire un raisonnement humain sous forme de règles linguistiques simples, chose qui s'apparente bien à notre problème puisque nous disposons d'aucun modèle mathématique qui nous décrit la méthode à adopter pour le guidage du vélo.

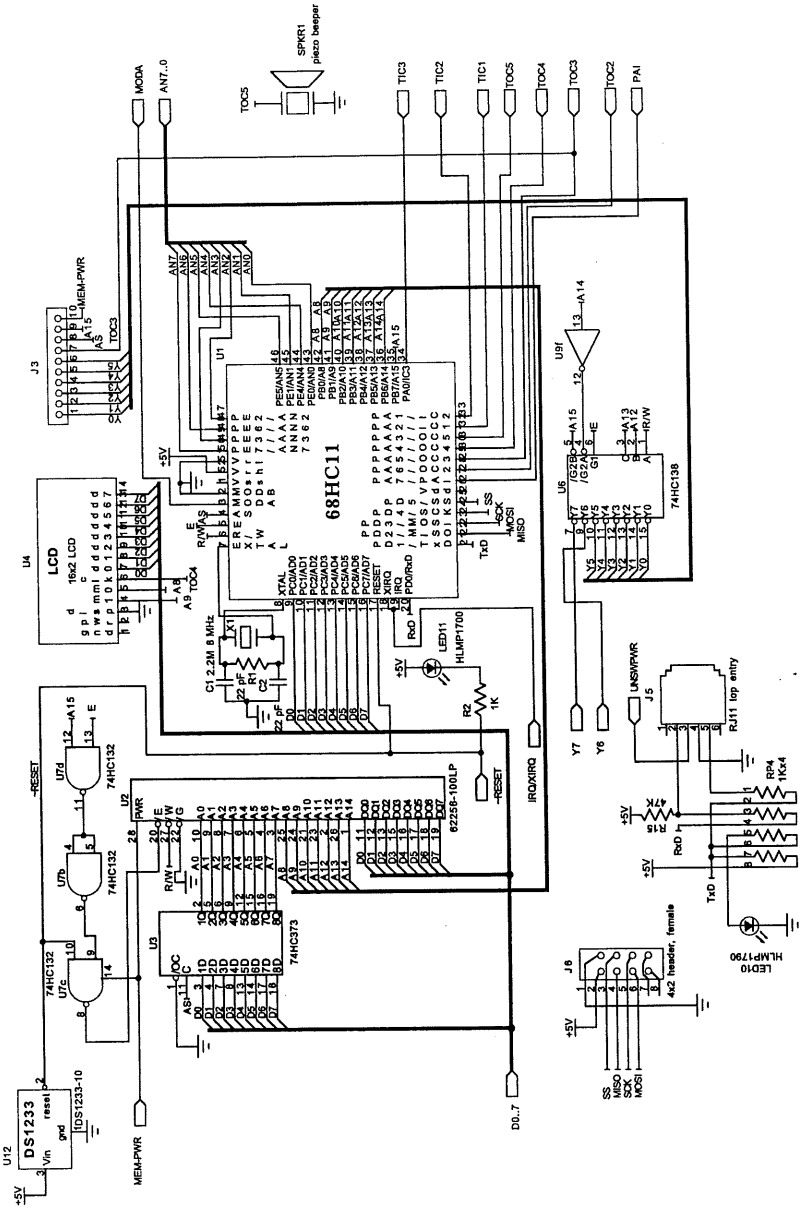
La base de connaissance de notre contrôleur flou est dérivée à partir de la base de règles d'un contrôleur proportionnel dérivé flou et en se basant sur l'observation du guidage manuel de la bicyclette, nous avons établi la base de règles finale qui permet d'avoir la supervision du pilotage humain de la bicyclette. L'un des principes qui nous a permis d'établir cette base de règles est que pour éviter une chute, la fourche du guidon du vélo doit tourner dans le sens de l'angle de la descente, principe que nous utilisons régulièrement pour piloter un vélo à l'échelle réelle.

Pour des meilleures performances de contrôle (surtout pour les virages serrés) l'utilisation d'un servomoteur avec un meilleur « slew rate » (vitesse de déplacement max ) devient indispensable. Aussi, une adaptation de la base de connaissance du contrôleur est fortement recommandé, cela peut être réalisé en utilisant une approche d'apprentissage par essais/erreurs telle que la technique du renforcement ou bien en développant un réseau neuro-flou pour adapter les paramètres de la base de connaissance.

# ANNEXE 1

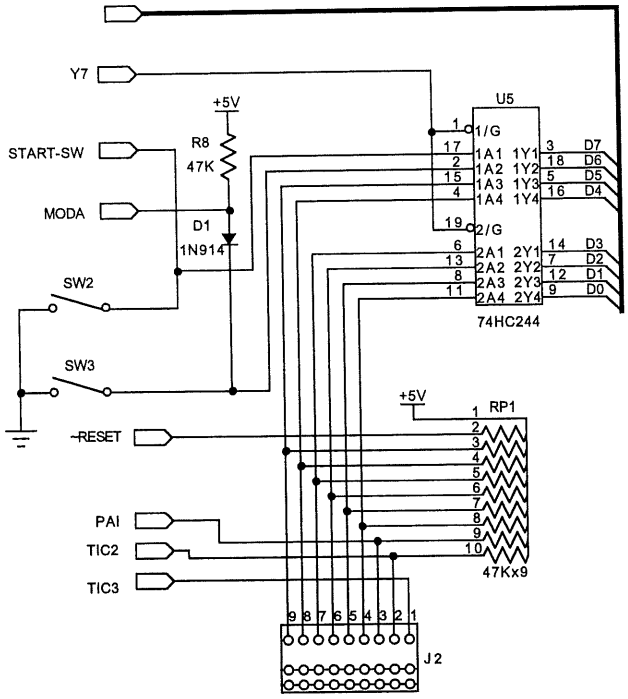
## Schémas de la carte de contrôle et des différentes composantes*

### 1. Schéma de la carte et de la mémoire

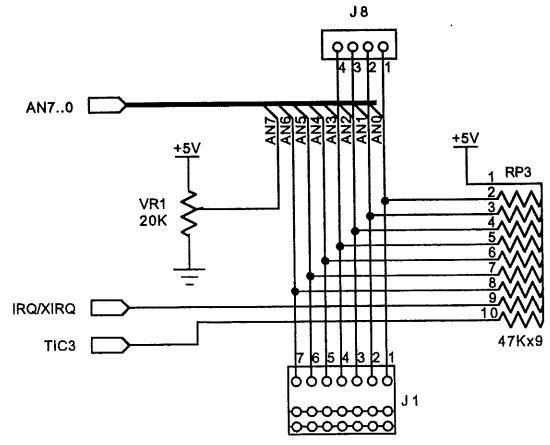



* Copie autorisée du manuel de la Handy board

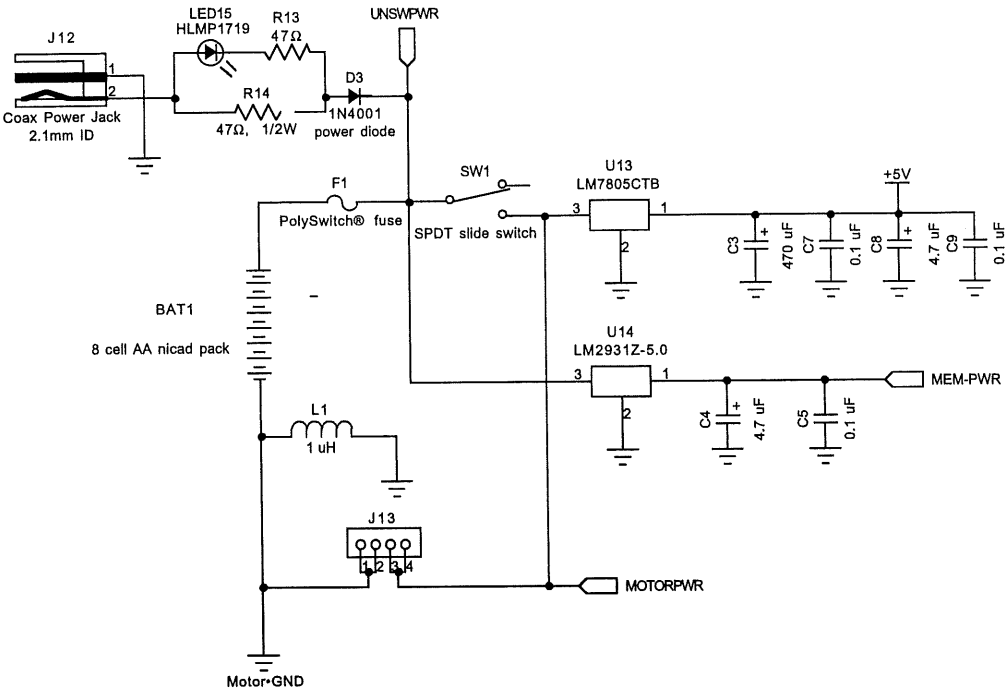
## 2. Entrées et Sorties numérique



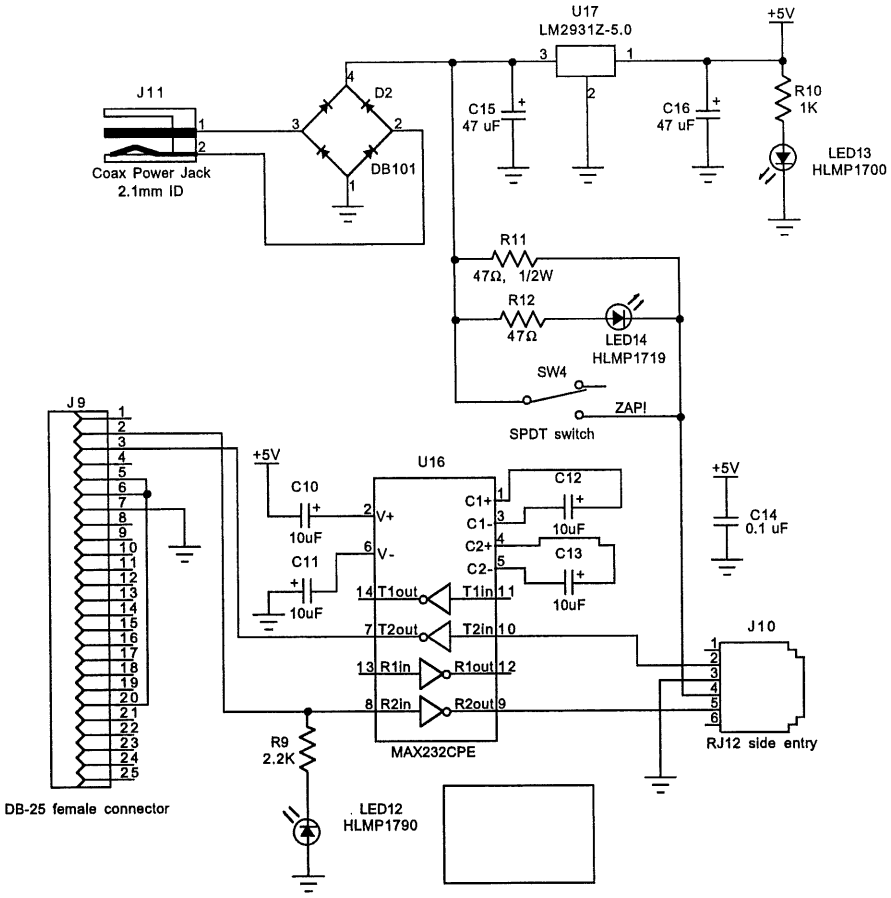

## 3. Entrées et Sorties analogiques



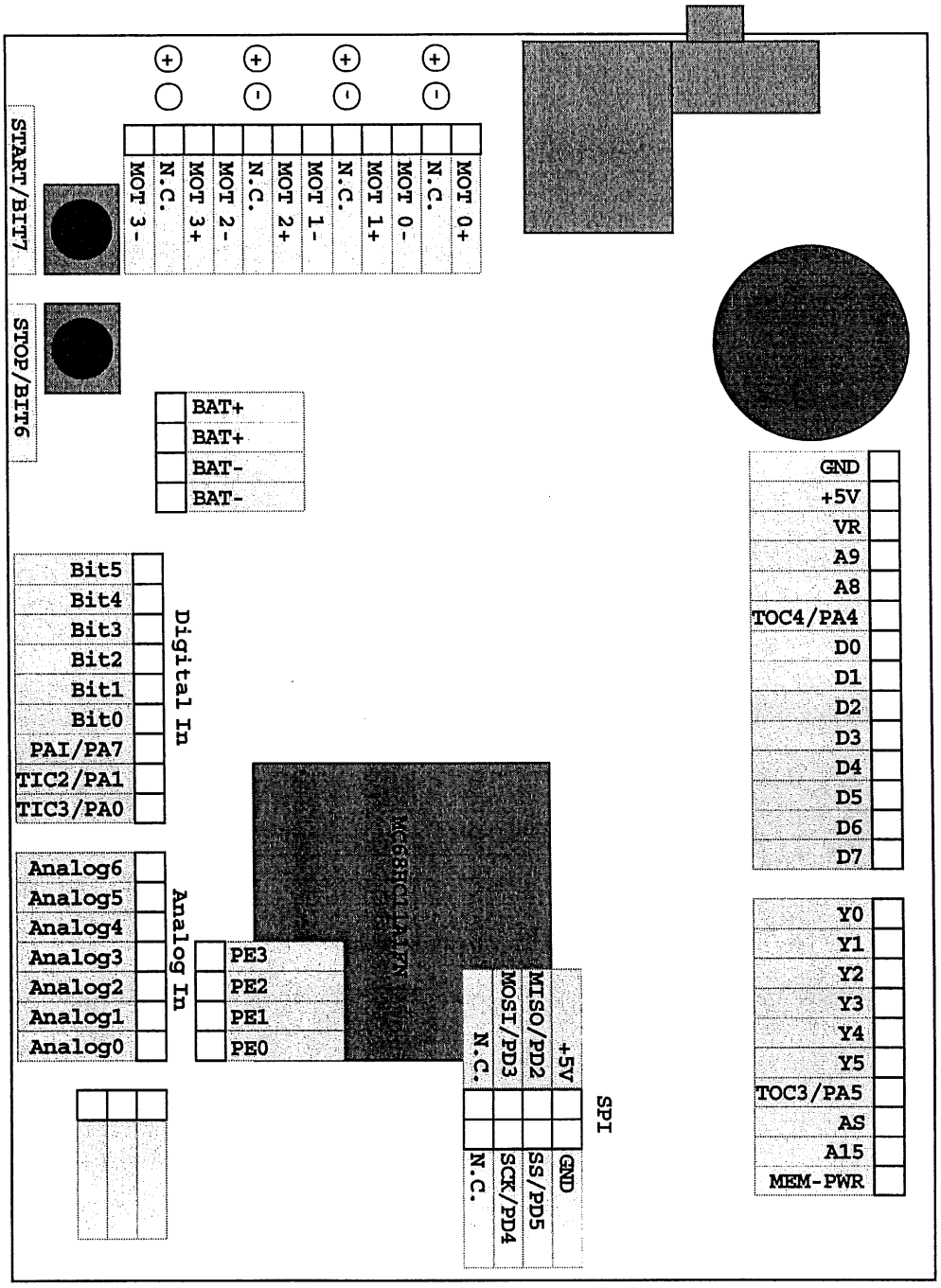
# 3. Circuit d'alimentation de la carte



4. Circuit d'interface série et recharge des batteries d'alimentation de la carte



# 5. Diagramme général de la carte



## ANNEXE 2

## CODE DU CONTRÔLEUR ET DES SYSTEMES D'ENTREES/SORTIES

```
/*****************************************************/
/* Programme pour le contrôle flou de l'équilibre et */
/* de la trajectoire du vélo                         */
/* Said berriah                                      */
/* Décembre 1999                                     */
/*****************************************************/


/* Variables globales*/

float error,error_dot,y, TAU = 0.01;
float P_in=100.0;
int DIR_in,v,batt_lev;
int MIN_SERVO_WAVETIME = 1400;
int MAX_SERVO_WAVETIME = 4680;
int SERVO_RANGE =(MAX_SERVO_WAVETIME-MIN_SERVO_WAVETIME);
int SERVO_offset=(MAX_SERVO_WAVETIME+MIN_SERVO_WAVETIME)/2;
float rexcursion = 3.14159;
float dexcursion = 180.;
int MIN_WAVETIME = 3400;
int MAX_WAVETIME = 4000;
int WAVE_RANGE=(MAX_WAVETIME-MIN_WAVETIME);
int WAVE_offset=(MAX_WAVETIME+MIN_WAVETIME)/2;
float G0_offset,Offset_dir,error;

/*********************************************/
/* Routine de mesure de la position du manche  */
/* de direction                                */
/*********************************************/

void setic3()
{
  bit_set(0x1022,1);/*activer l'interruption IC3*/
  width=0;          /*remise à zero de la variable de mesure*/
}

int readic3()
{ float c;
  setic3();
  c=((3280.0/1679.0)*((float)width-2982.5))+3040.0;/*mise à l'échelle*/
  DIR_in=(int)c;/*arrondir la valeur mesurer*/
  return(DIR_in);
}

/*****************************************************/
/* Routine d'envoi du signal PWM sur le canal TOC3/PA5 */
/* Pour le servo (direction)                           */
/*****************************************************/

void servo_on()/*initialisation de la commande*/
{
  servo_a5_init(1);
```

```
}
void servo_off()/*stop la commande*/
{
  servo_a5_init(0);
}

/* Commandes de direction du servo*/

int servo(int period) /* argument en nombre de cycles d'impulsions */
{
  if(period>MAX_SERVO_WAVETIME)
  return (servo_a5_pulse=MAX_SERVO_WAVETIME);
  else if(period<MIN_SERVO_WAVETIME)
  return (servo_a5_pulse=MIN_SERVO_WAVETIME);
  else
  return(servo_a5_pulse=period);
}

int servo_rad(float angle) /* argument en radians*/
{
  return servo(radian_to_pulse(angle));
}

int servo_deg(float angle) /* argument en degrees*/
{
  return servo(degree_to_pulse(angle));
}


int radian_to_pulse(float angle) /*Conversion du radians en cycles
d'implusions*/
{
  return ((int)(angle*((float)SERVO_RANGE)/rexcursion)+MIN_SERVO_WAVETIME);
}

int degree_to_pulse(float angle) /*Conversion du degrés en cycles d'impulsions
*/
{
  return ((int)((angle*((float)SERVO_RANGE))/dexcursion)+SERVO_offset);
}

/*********************************************************/
/* Envoi de du signal PWM au drive de propulsion         */
/* sur la ligne PA7                                      */
/*********************************************************/

void pwm_on()/*initialisation de l'envoi*/
{
  wave_a7_init(1);
}
void pwm_off()/*stop l'envoi*/
{
  wave_a7_init(0);
}

int pwm(int period) /* argument en nombre de cycles d'impulsions */
{
```

```
  if(period>MAX_WAVETIME)
  return (a7_pulse=MAX_WAVETIME);
  else if(period<MIN_WAVETIME)
  return (a7_pulse=MIN_WAVETIME);
  else
  return(a7_pulse=period);
}

/*********************************************************/
/* Routine de mesure de la vitesse à travers le capteur  */
/* optique TIC2                                          */
/*********************************************************/

void enable_encoder() /*initialisation de l'encodeur*/
{
  bit_set(0x1022,2);
  port1_shaft_count=0;
}

void disable_encoder()/*décative l'encodeur*/
{
  bit_clear(0x1022,2);
}

void reset_encoder()/*reset de l'encodeur*/
{
  port1_shaft_count=0;

}

int read_encoder()/*lecture de l'encodeur*/
{
  return(port1_shaft_count);
}

void readv()/*mesure de la vitesse chaque 20 ms*/
{
 int v1,v2;
 enable_encoder();
 v1=read_encoder();
 sleep(0.02);
 v2=read_encoder();
 v=v2-v1;
 reset_encoder();

}

/*********************************************************/
/*Routine d'envoi d'un mot sur le lien série RS232       */
/*********************************************************/

void serial_putchar(int c)
{
   while (!(peek(0x102e) & 0x80));  /* attend que le buffer soit vide*/
   poke(0x102f, c);  /* envoi charactère */
}
```

```
/*********************************************************/
/* Routine de mesure de l'état des baterries             */
/*********************************************************/

get_batt()
{
 batt_lev=analog(1);
 return(batt_lev);
}

/*********************************************************/
/* Calcul de la valeur absolue                           */
/*********************************************************/

int abs(int arg)  /* fonction valeur absolue*/
{  if (arg < 0)
      return -1*arg;
    else
      return arg;
}

/*********************************************************/
/* Routine principale                                    */
/* Routine de contrôle                                   */
/*********************************************************/

void main()
/* Fonction de calcul de l'offset du gyromètre sur
   les premiers 256*/
float sum=0.0;
int i;
for (i=0;i<=256;i++)
{
 sum=sum+(float)analog(3);}
 G0_offset=sum/256.0;
}
/* Fonction de calcul de l'offet du manche de direction
   sur les premiers 256 */

float sum=0.0;
int i;
for (i=0;i<=256;i++)
{ readic3();
  sum=sum+(float)DIR_in;}
  Offset_dir=sum/256.0;
}


beep();
beep();

error=0.0;
```

```
servo_on();/*active le servo*/
pwm_on();  /*active la propulsion*/


/* Boucle de contrôle*/

while(1)
{
  readic3();
  DIR_in=DIR_in-Offset_dir /*retranche l'offset direction*/
  DIR_in=DIR_in*P_in; /* Multiplie par le gain de manche */
  teta_dot_d=DIR_in;
  G0=analog(3)-(int)G0_offset;/*retranche l'offet direction*/
  if(abs(G0)<=1) /* mise à l'echelle*/
  {G0=0;}
  teta_dot=(float) G0*(360.0/255.0);/*normalisation*/
  error_dot=teta_dot_d-teta_dot;/*calcul dérivée de l'erreur*/
  error = error + TAU * error_dot;/*integration*/
  if (error > 30.0) /*saturation*/
  error = 30.0;
  if (error < -30.0)
  error = -30.0;
  flc_input2=(255*(error_dot+10))/20;/*normalisation sur 8 bits*/
  flc_input1 =(255*((int)error + 10))/20;/*normalisation sur 8 bits*/
  flc_exec(0); /* inference flou*/
  y=(2.0/255.0)*((float)flc_output1)-128.0);/*normalisation de la commande*/
  v=readv(); /*mesure de la vitesse*/
  if (v<100) /* gain sechule*/
  y=y*g1/(v*v);
  if ((v>100)||(v<200))
  y=y*g2/(v*v);
  if(v>200)
  y=y*g3/(v*v);
  servo_deg(y); /* envoi de la commande*/
  get_batt(); /* niveau de batteries*/

/* envoi des signaux pertient au lien série sur 8 bits*/

serial_putchar(batt_lev);/*niveau de la batterie*/
serial_putchar(flc_input2);/* gyromètre*/
```

```
serial_putchar(flc_output1);/* commande sur le servo*/
serial_peutchar(v);/*vitesse mesurée*/
}}

*************************************************************************
*
* SERVO_A5.ASM                                                          *
*                                                                       *
* Utilise le PORT A Bit pour la generation du signal PWM pour le servo*  *
                                                                        *
*                                                                       *
* Définitions :                                                         *
* IC global "servo_a5_pulse" -- largeur d'impulsion dans 500ns unites *  *
                                                                        *
*IC fonction "servo_a5_init(int enable)" -- appele avec 1 pour activer*
*                                                    0 pour desactiver*
*                                                                       *
*************************************************************************

BASE    EQU     $1000   ; base de registre

PORTA   EQU     $1000   ; Port A data register
CFORC   EQU     $100B   ; Timer Compare Force Register
TCNT    EQU     $100E   ; Timer Count Register
TOC3    EQU     $101A   ; Timer Output Compare register 3
TCTL1   EQU     $1020   ; Timer Control register 1
TMSK1   EQU     $1022   ; main Timer interrupt Mask register 1
TFLG1   EQU     $1023   ; main Timer interrupt Flag register 1
PACTL   EQU     $1026   ; Pulse Accumulator Control register

TOC3INT EQU     $E4     ; Timer Output Compare 3

SERVO_BIT EQU   $20     ; PA5 bit
SERVO_PORT EQU  PORTA   ; servo output port
PULSE_DEFAULT EQU 3040  ; valeur initiale quand le servo est allumé

        ORG     MAIN_START

* C variables
variable_servo_a5_pulse         FDB     0

* Variables internes
servo_a5_pulse                  FCB     0       ; si 0 généré un gap


subroutine_initialize_module:
        ldx     #$bf00

* Installation de l'interruption TOC3

        ldd     #servo_a5_int
        std     TOC3INT,X

        ldd     #PULSE_DEFAULT
        std     variable_servo_a5_pulse ; initialise la periode du servo
```

```
        clrb                            ; run "servo_disable" et sort


*************************************************
*  Routine d'initialisation d'interruption pour TOC3*
*************************************************

subroutine_servo_a5_init:

        ldx     #BASE
        tstb
        beq     servo_disable

servo_enable

* Commence avec une impulsion positive

        ldd     TCNT,X
        addd    variable_servo_a5_pulse
        std     TOC3,X                  ; fin de l'impulsion positive

        bset    TCTL1,X $30             ; si un match TOC3 devient high
        bset    CFORC,X SERVO_BIT       ; force un match
        bclr    TCTL1,X $10             ; si un match, TOC3 devient low
        bset    TFLG1,X SERVO_BIT       ; clear interrupt flag
        bset    TMSK1,X SERVO_BIT       ; enable TOC3 interrupt

        clr     servo_a5_pulse          ; demande à l'int de faire un gap

        rts

servo_disable
        bclr    TMSK1,X SERVO_BIT       ; désactive l'interruption TOC3

        rts

*******************************
* Routine d'interruption TOC3 *
*******************************

servo_a5_int

        ldx     #BASE
        bclr    TFLG1,X $FF-SERVO_BIT   ; clear interrupt flag

* S'il y a interruption et bit vrai, alors set up pour un front descendant
        tst     servo_a5_pulse
        bne     setup_pulse

setup_gap
        ldd     TOC3,X                  ; temps du front descendant
        addd    #40000                  ; 20 ms après
        subd    variable_servo_a5_pulse ; largeur base de l'impulsion
        std     TOC3,X
```

```
        ldaa    #1
        staa    servo_a5_pulse          ; la prochaine impulsion

        rti

setup_pulse

        ldd     TOC3,X
        addd    variable_servo_a5_pulse
        std     TOC3,X                  ; fin de l'impulsion positive

        bset    TCTL1,X $30             ; si match, TOC3 High
        bset    CFORC,X SERVO_BIT       ; force un match
        bclr    TCTL1,X $10             ; si match, TOC3 low

        clr     servo_a5_pulse          ; faire un gap

        rti

*************************************************************************
*                                                                       *
* PWM_A7.ASM                                                            *
*                                                                       *
* Utilisation:                                                          *
* Generation de signal PWM pour la propulsion sur la ligne PA7          *
* Definition:                                                           *
* IC global "a7_pulse" --                                               *
* IC fonction "wave_a7_init(int enable)" -- appele avec 1 pour activer*
*                                                        0 pour desactiver*
*                                                                       *
*************************************************************************

BASE    EQU     $1000   ; Base de registre
PORTA   EQU     $1000   ; Port A data register
CFORC   EQU     $100B   ; Timer Compare Force Register
OC1M    EQU     $100C   ; Output Compare 1 Mask register
OC1D    EQU     $100D   ; Output Compare 1 Data register
TCNT    EQU     $100E   ; Timer Count Register
TOC1    EQU     $1016   ; Timer Output Compare register 1
TMSK1   EQU     $1022   ; main Timer interrupt Mask register 1
TFLG1   EQU     $1023   ; main Timer interrupt Flag register 1
PACTL   EQU     $1026   ; Pulse Accumulator Control register

TOC1INT EQU     $E8     ; Timer Output Compare 1 vector

SERVO_BIT EQU   $80     ; PA7 bit
SERVO_PORT EQU  PORTA   ; output port
PULSE_DEFAULT EQU 3000  ; Valeur initiale quand le moteur est etteint

        ORG     MAIN_START

* C variables
variable_a7_pulse       FDB     0

* Variables interne
a7_pulse                FCB     0       ; si 0, génère un gap
```

```
subroutine_initialize_module:
        ldx     #$bf00

* Installation de l'interruption TOC1

        ldd     #a7_int
        std     TOC1INT,X

        ldd     #PULSE_DEFAULT
        std     variable_a7_pulse ; Initialise l'impulsion

        clrb

*******************************
* Routine d'initialisation    *
*******************************

subroutine_wave_a7_init:
        ldx     #BASE

        tstb
        beq     motor_disable

motor_enable
        bset    PACTL,X $80             ; PA7 en sortie
        bset    OC1M,X SERVO_BIT        ; Demande à A7 de générer l'impulsion

* begin with positive-going pulse
        ldd     TCNT,X
        addd    variable_a7_pulse
        std     TOC1,X                  ; fin de l'impulsion positive

        bset    OC1D,X SERVO_BIT        ;
        bset    CFORC,X SERVO_BIT       ; force un match
        bclr    OC1D,X SERVO_BIT        ; au prochain match, le bit devient low
        bset    TFLG1,X SERVO_BIT       ; clear interrupt flag
        bset    TMSK1,X SERVO_BIT       ; enable TOC1 interrupt

        clr     a7_pulse          ; tell int routine to make gap

        rts

motor_disable
        bclr    PACTL,X $80             ; désactive le bit de sortie du moteur
        bclr    TMSK1,X SERVO_BIT       ; désactive TOC1 interrupt

        rts

*********************************
* Routine d'interruption de PA7   *
*********************************
a7_int
        ldx     #BASE
        bclr    TFLG1,X $FF-SERVO_BIT   ; clear interrupt flag

* Si interruption et bit vrai, set up pour un front descendant
        tst     a7_pulse
```

```
        bne     setup_pulse

setup_gap
        ldd     TOC1,X                  ; temps du front descendant
        addd    #40000                  ; 20 ms après
        subd    variable_a7_pulse ; largeur de l'impulsion
        std     TOC1,X

        ldaa    #1
        staa    a7_pulse          ; prochaine impulsion

        rti

setup_pulse
        ldd     TOC1,X
        addd    variable_a7_pulse
        std     TOC1,X                  ; fin de l'impulsion positive

        bset    OC1D,X SERVO_BIT        ; bit de sortie on
        bset    CFORC,X SERVO_BIT       ; force un match


        bclr    OC1D,X SERVO_BIT        ; prochain match le bit devient low

        clr     a7_pulse          ; faire un gap

        rti

*********************************************************
* Routine pour la mesure de la vitesse à travers       *
* le capteur optique                                    *
*********************************************************

* Base de registres
#include <6811regs.asm>

* program equates

* variables globales

      ORG   MAIN_START

variable_port1_shaft_count    FDB   0

subroutine_initialize_module:
#include <ldxibase.asm>

*Installation de l'interruption
      LDD   #Port1_ShaftInt
      STD   TIC2INT,X




*Initialisation de l'interruption TIC2
```

```
      LDX   #BASE
      BSET  TCTL2,X %00001111 /* Interrupts TIC2 front montant*/
      BCLR  TFLG1,X %00000011 /* Clear IC3,IC2 Flags */
      BCLR  TMSK1,X %00000011 /* Clear Mask pour les Interruptions */

Port1_ShaftInt:
      LDX   variable_port1_shaft_count
      INX
      STX   variable_port1_shaft_count
      LDX   #BASE
      BCLR  TFLG1,X %11111101
      RTI

*********************************************************
* Routine d'interruption pour la mesure du PWM du manche*
* de direction    TIC3                                  *
*********************************************************


* Definition de la base des registres *

#include "C:\pcode\include\6811regs.asm"

* variables globales*

      ORG   MAIN_START

variable_frst     FDB    0
variable_width    FDB    0

* Installtion de la routine d'interruption IC3*

subroutine_initialize_module:

#include "c:\pcode\include\ldxibase.asm"


      LDD   #IC3Int
      STD   TIC3INT,X
      LDX   #BASE
      LDAA  #%00000000
      STAA  TMSK2,X
      BSET  TCTL2,X %00000001 /* Interrupts TIC3 front motant*/
      BSET  TFLG1,X %00000001 /* Clear IC3 Flags */
      BCLR  TMSK1,X %00000001 /* Clear Mask pour Interrupt */
      RTS


*Interruption IC3*

IC3Int:
      LDX   #BASE
      BRSET TCTL2,X %00000001 edg1 /*si front montant saute*/
      LDAA  #%00000001/*mesure de la largeur*/
      STAA  TCTL2,X
      LDD TIC3,X
      SUBD variable_frst
```

```
      STD variable_width
      BRA exit

edg1  LDAA  #%00000010 /* mesure front descendant*/
      STAA  TCTL2,X
      LDD   TIC3,X
      STD   variable_frst
exit  BSET  TFLG1,X %00000001
      RTI


*****************************************************************
* Defintion de la base de connaissance du contrôleur flou       *
* Fonctions d'appartenances d'entrées et de sorties             *
* Définitions des règles                                        *
*****************************************************************


INPUT_MFS   EQU   *          ; Fonctions d'appartenance d'entrée
IN0MF       EQU   *          ;     Erreur
            FCB   $00,$00,$00,$06       ;           LB
            FCB   $00,$06,$2D,$06       ;           LM
            FCB   $2D,$06,$59,$07       ;           LS
            FCB   $59,$07,$80,$07       ;           ZE
            FCB   $80,$07,$A6,$06       ;           RS
            FCB   $A6,$06,$D2,$06       ;           RM
            FCB   $D2,$06,$FF,$00       ;           RB
            FCB   $00,$00,$00,$00       ;
IN1MF       EQU   *          ;     Deltaerreur
            FCB   $00,$00,$00,$06       ;           LB
            FCB   $00,$06,$2D,$06       ;           LM
            FCB   $2D,$06,$56,$06       ;           LS
            FCB   $56,$06,$80,$06       ;           ZE
            FCB   $80,$06,$A9,$06       ;           RS
            FCB   $A9,$06,$D2,$06       ;           RM
            FCB   $D2,$06,$FF,$00       ;           RB
            FCB   $00,$00,$00,$00       ;
SGLTN_POS   EQU   *          ; Fonctions d'appartenance de sortie
OUT0MF      EQU   *          ;     y
            FCB   $00        ;     LB
            FCB   $28        ;     LM
            FCB   $59        ;     LS
            FCB   $A7        ;     ZE
            FCB   $80        ;     RS
            FCB   $D7        ;     RM
            FCB   $FF        ;     RB
            FCB   $00        ;
RULE_START  EQU   *          ;     Rèlges:
            FCB   $00        ;     <Erreur = LB
            FCB   $08        ;     <Deltaerreur = LB
            FCB   $80        ;     >y = LB
            FCB   $0A        ;     <Deltaerreur = LS
            FCB   $80        ;     >y = LB
            FCB   $00        ;     <Erreur = LB
            FCB   $09        ;     <Deltaerreur = LM
            FCB   $80        ;     >y = LB
            FCB   $00        ;     <Erreur = LB
```

```
FCB   $0B          ;     <Deltaerreur = ZE
FCB   $80          ;     >y = LB
FCB   $0C          ;     <Deltaerreur = RS
FCB   $80          ;     >y = LB
FCB   $00          ;     <Erreur = LB
FCB   $0D          ;     <Deltaerreur = RM
FCB   $80          ;     >y = LB
FCB   $00          ;     <Erreur = LB
FCB   $0E          ;     <Deltaerreur = RB
FCB   $80          ;     >y = LB
FCB   $01          ;     <Erreur = LM
FCB   $08          ;     <Deltaerreur = LB
FCB   $80          ;     >y = LB
FCB   $01          ;     <Erreur = LM
FCB   $09          ;     <Deltaerreur = LM
FCB   $80          ;     >y = LB
FCB   $01          ;     <Erreur = LM
FCB   $0A          ;     <Deltaerreur = LS
FCB   $80          ;     >y = LB
FCB   $01          ;     <Erreur = LM
FCB   $0B          ;     <Deltaerreur = ZE
FCB   $80          ;     >y = LB
FCB   $01          ;     <Erreur = LM
FCB   $0C          ;     <Deltaerreur = RS
FCB   $82          ;     >y = LS
FCB   $01          ;     <Erreur = LM
FCB   $0D          ;     <Deltaerreur = RM
FCB   $83          ;     >y = ZE
FCB   $0E          ;     <Deltaerreur = RB
FCB   $85          ;     >y = RM
FCB   $02          ;     <Erreur = LS
FCB   $08          ;     <Deltaerreur = LB
FCB   $80          ;     >y = LB
FCB   $02          ;     <Erreur = LS
FCB   $09          ;     <Deltaerreur = LM
FCB   $80          ;     >y = LB
FCB   $02          ;     <Erreur = LS
FCB   $0A          ;     <Deltaerreur = LS
FCB   $80          ;     >y = LB
FCB   $02          ;     <Erreur = LS
FCB   $0B          ;     <Deltaerreur = ZE
FCB   $82          ;     >y = LS
FCB   $0C          ;     <Deltaerreur = RS
FCB   $83          ;     >y = ZE
FCB   $02          ;     <Erreur = LS
FCB   $0D          ;     <Deltaerreur = RM
FCB   $85          ;     >y = RM
FCB   $0E          ;     <Deltaerreur = RB
FCB   $85          ;     >y = RM
FCB   $03          ;     <Erreur = ZE
FCB   $00          ;     <Erreur = LB
FCB   $80          ;     >y = LB
FCB   $02          ;     <Erreur = LS
FCB   $09          ;     <Deltaerreur = LM
FCB   $80          ;     >y = LB
FCB   $03          ;     <Erreur = ZE
FCB   $0A          ;     <Deltaerreur = LS
```

```
FCB   $82          ;     >y = LS
FCB   $03          ;     <Erreur = ZE
FCB   $0B          ;     <Deltaerreur = ZE
FCB   $83          ;     >y = ZE
FCB   $0C          ;     <Deltaerreur = RS
FCB   $84          ;     >y = RS
FCB   $03          ;     <Erreur = ZE
FCB   $0D          ;     <Deltaerreur = RM
FCB   $86          ;     >y = RB
FCB   $0E          ;     <Deltaerreur = RB
FCB   $86          ;     >y = RB
FCB   $04          ;     <Erreur = RS
FCB   $08          ;     <Deltaerreur = LB
FCB   $81          ;     >y = LM
FCB   $04          ;     <Erreur = RS
FCB   $09          ;     <Deltaerreur = LM
FCB   $81          ;     >y = LM
FCB   $04          ;     <Erreur = RS
FCB   $0A          ;     <Deltaerreur = LS
FCB   $83          ;     >y = ZE
FCB   $04          ;     <Erreur = RS
FCB   $0B          ;     <Deltaerreur = ZE
FCB   $84          ;     >y = RS
FCB   $04          ;     <Erreur = RS
FCB   $0C          ;     <Deltaerreur = RS
FCB   $86          ;     >y = RB
FCB   $04          ;     <Erreur = RS
FCB   $0D          ;     <Deltaerreur = RM
FCB   $86          ;     >y = RB
FCB   $04          ;     <Erreur = RS
FCB   $0E          ;     <Deltaerreur = RB
FCB   $86          ;     >y = RB
FCB   $05          ;     <Erreur = RM
FCB   $08          ;     <Deltaerreur = LB
FCB   $81          ;     >y = LM
FCB   $05          ;     <Erreur = RM
FCB   $09          ;     <Deltaerreur = LM
FCB   $83          ;     >y = ZE
FCB   $05          ;     <Erreur = RM
FCB   $0A          ;     <Deltaerreur = LS
FCB   $84          ;     >y = RS
FCB   $05          ;     <Erreur = RM
FCB   $0B          ;     <Deltaerreur = ZE
FCB   $86          ;     >y = RB
FCB   $05          ;     <Erreur = RM
FCB   $0C          ;     <Deltaerreur = RS
FCB   $86          ;     >y = RB
FCB   $0D          ;     <Deltaerreur = RM
FCB   $86          ;     >y = RB
FCB   $05          ;     <Erreur = RM
FCB   $0E          ;     <Deltaerreur = RB
FCB   $86          ;     >y = RB
FCB   $06          ;     <Erreur = RB
FCB   $08          ;     <Deltaerreur = LB
FCB   $83          ;     >y = ZE
FCB   $06          ;     <Erreur = RB
FCB   $09          ;     <Deltaerreur = LM
```

```
            FCB   $86          ;     >y = RB
            FCB   $06          ;     <Erreur = RB
            FCB   $0A          ;     <Deltaerreur = LS
            FCB   $86          ;     >y = RB
            FCB   $06          ;     <Erreur = RB
            FCB   $0B          ;     <Deltaerreur = ZE
            FCB   $86          ;     >y = RB
            FCB   $06          ;     <Erreur = RB
            FCB   $0C          ;     <Deltaerreur = RS
            FCB   $86          ;     >y = RB
            FCB   $06          ;     <Erreur = RB
            FCB   $0D          ;     <Deltaerreur = RM
            FCB   $86          ;     >y = RB
            FCB   $06          ;     <Erreur = RB
            FCB   $0E          ;     <Deltaerreur = RB
            FCB   $86          ;     >y = RB
END_OF_RULE FCB   $ff
NUMINP      EQU   $02
NUMOUT      EQU   $01
DEFVER      FCC   '1.02'

*******************************************************************************
* Le contrôleur flou
*******************************************************************************

            ORG     MAIN_START

#include    "icfuzz.asm"


*******************************************************************************
*
*  Variables IC                                                    *
*
*******************************************************************************
variable_flc_input1
FUZ_IN1     FDB     $00
variable_flc_input2
FUZ_IN2     FDB     $00
variable_flc_input3
FUZ_IN3     FDB     $00
variable_flc_input4
FUZ_IN4     FDB     $00
variable_flc_input5
FUZ_IN5     FDB     $00
variable_flc_input6
FUZ_IN6     FDB     $00
variable_flc_input7
FUZ_IN7     FDB     $00
variable_flc_input8
FUZ_IN8     FDB     $00
variable_flc_output1
FUZ_OUT1    FDB     $00
variable_flc_output2
FUZ_OUT2    FDB     $00
variable_flc_output3
FUZ_OUT3    FDB     $00
```

```
variable_flc_output4
FUZ_OUT4    FDB     $00

**Struture de données et variable

CURRENT_INS EQU     * ;Stocke les entrées sur 8 bits
            FCB     0
            FCB     0
            FCB     0
            FCB     0
            FCB     0
            FCB     0
            FCB     0
            FCB     0
FUZ_INS     EQU     *
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
FUZ_OUTS    EQU     *    ;Stock les sorties floues
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
```

```
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
            FDB     0
COG_OUTS    EQU     *       ;Sorties deffuzifier
            FCB     0
            FCB     0
            FCB     0
            FCB     0
SUM_OF_FUZ  FDB     2           ;somme sur 11-bit pour la sortie
SUM_OF_PROD FDB     0           ;produits de somme sur 19 bits
            FCB     0
COGDEX      FCB     1           ;numero de sortie courante pour le COG
LP_COUNT    FCB     1           ;Index pour la boucle flou
SUMDEX      FCB     1           ;Index pour la boucle de somme

subroutine_flc_exec:
*           SEI
            LDAB    #NUMINP
            LDY     #CURRENT_INS
            LDX     #FUZ_IN1+1
CPY_LOOP    LDAA    0,X
            STAA    0,Y
            INX
            INX
            INY
            DECB
            BNE     CPY_LOOP

***** Inference flou commence ici*********
* Programme suppose qu'il y a au moins une entrée valide
*
FUZZIFY     LDY     #FUZ_INS                    ;Pointe vers la table ;des
                                                 entrées
            LDX     #CURRENT_INS+NUMINP-1       ;Pointe vers vers la fin   ;de
                                                la table d'entrée
PUSH_LOOP   LDAA    0,X                         ;Prend la valeur
            PSHA                                ;et la pousse dans la ;pile
            CPX     #CURRENT_INS                ;Derniere valeur?
            BEQ     DONE_PUSH                   ;Si oui fait un ;branchement
            DEX                                 ;sinon pointe au ;prochain byte
            BRA     PUSH_LOOP                   ;branche vers la  ;prochaine
                                                valeur

DONE_PUSH   LDX     #INPUT_MFS    ;Pointe vers les specs des MF ;d'entrées

NXTIN_LP    PULA                  ;Prend la prochaine valeur courante
            LDAB    #7            ;Pour 8 passage dans la boucle
            STAB    LP_COUNT      ;Initialise le compteur de boucle

GRAD_LOOP   EQU     *             ;Stocke le degré pour une valeur ;d'une entrée

*****************************************************************
* GET_GRADE - Routine pour fuzzifier une entrée discrete     *
```

```
*   Entrée la valeur dans le registre A et x pionte vers      *
*   les pentes la position des MFs                             *
*   Retourne le grade dans B, X pointe au spec de la prochaine MF *
*   (X+4) et A est inchangé.                                   *
*****************************************************************
GET_GRADE   PSHA                ;Sauve la valeur dans A
            CLRB                ;dans la case grade = 0
            SUBA    2,X         ;Valeur d'entrée D pt2 -> A
            BLS     NOT_SEG2    ;Si valeur < pt2
            LDAB    3,X         ;Pente 2
            BEQ     HAV_GRAD    ;Skip si pente verticale
            MUL                 ;(In D pt1) * slp2 -> A:B
            TSTA                ;verifie si > $FF
            BEQ     NO_FIX      ;Si superieur à 8 = 0
            CLRB                ;Grade limite à 0
            BRA     HAV_GRAD    ;Dans la region limite du seg 2
NO_FIX      SUBB    #$FF        ;B D $FF
            NEGB                ;$FF D B
            BRA     HAV_GRAD    ;($FF D((In D pt2) * slp2))
NOT_SEG2    ADDA    2,X         ;Restore la valeur initiale
            SUBA    0,X         ;Valeur d'entrée D pt1 -> A
            BLO     HAV_GRAD    ;Si < pt1 donc grade = 0
            LDAB    1,X         ;Pente 1
            BEQ     VERT_SLP    ;Skip si pente verticale
            MUL                 ;(In D pt1) * slp1 -> A:B
            TSTA                ;Verifie si > $FF
            BEQ     HAV_GRAD    ;Resultat dans B
VERT_SLP    LDAB    #$FF        ;Region limite ou pente verticale
HAV_GRAD    INX                 ;Pointe vers la prochaine MF
            INX
            INX
            INX
            PULA                                ;Restore le registre A

            STAB    0,Y                         ;Sauve l'entrée flou
            INY                                 ;Pointe vers la ;prochaine
            DEC     LP_COUNT                    ;
            BPL     GRAD_LOOP                   ;Pour les 8 étiquettes
            CPY    #NUMINP*8+FUZ_INS            ;Est ce fait pour toutes ;les ins
                                                flous
            BNE     NXTIN_LP                    ;Si non, process la ;prochaine
                                                entrées


* Evaluation des règles

            LDX     #FUZ_OUTS       ;Pointe vers la premiere sortie floue
            LDAA    #8*NUMOUT       ;Nombre de sortie floues
CLR_OUTS    CLR     0,X             ;Clear la sortie floue
            INX                     ;Pointe vers la prochaine
            DECA                    ;Index de boucle
            BNE     CLR_OUTS        ;Continue jusqu'à ce que toute les ;sorties
                                    floues=0
            LDY     #RULE_START     ;Pointe vers la 1 règles
RULE_TOP    LDAA    #$FF            ;Commence à traiter la premiere ;règles
```

```
IF_LOOP     LDAB    0,Y             ;Prend le byte de la règle 000X XAAA; ;Si X
                                    est A
            BMI     THEN_LOOP       ;Si MSB=1, sort vers then loop
            INY                     ;Pointe vers le prochain byte de ;règle
            LDX     #FUZ_INS        ;Pointe vers les sorties floues
            ABX                     ;Pointe vers les specs des entrées ;floues
            CMPA    0,X             ;Si cette entrée floue est plus ;petite
            BLS     IF_LOOP         ;Sinon ne pas remplacer la plus ;faible
            LDAA    0,X             ;Remplacer la plus faible si oui.
            BNE     IF_LOOP         ;<zero, fait le prochain byte


FIND_THEN   LDAB    0,Y             ;Prend le prochain byte de règle
            BMI     FIND_IF         ;MSB=1 => partie consequence
            INY                     ;Pointe vers le prochain byte
            BRA     FIND_THEN       ;boucle pointe vers la partie ;consequence
FIND_IF     INY                     ;Pointe vers la partie antecedant
            LDAB    0,Y             ;Pointe vers le prochain byte.
            BPL     RULE_TOP        ;MSB=0 =>partie antecedant
            CMPB    #$FF            ;$FF si pas de markeur de regles
            BNE     FIND_IF
            BRA     DEFUZ           ;si toute les regles sont inferées, ;deffuzifi

THEN_LOOP   LDX     #FUZ_OUTS       ;Pointe vers la sortie floue
            ANDB    #$7F
            ABX

            CMPA    0,X             ;Compare avec la sortie flou
            BLO     NOT_HIER        ;Branch si c'Est pas superieur
            STAA    0,X             ;Grade est + grand donc update
NOT_HIER    INY                     ;Pointeur de regles vers le next byte
            LDAB    0,Y             ;Prends le byte de regle
            BPL     RULE_TOP        ;SI MSB=0 c une nouvelle règle
CHK_END     CMPB    #$FF            ;Check le flag de fin des re`gles
            BNE     THEN_LOOP       ;Si non $FF,doit etre une partie ;consequence


* Maintenant toute les regles evaluer doit deffuzifier les sorties

DEFUZ       LDY     #SGLTN_POS      ;Pointe vers le 1ier singleton de ;sortie
            LDX     #FUZ_OUTS       ;Pointe vers la 1ere sortie floue
            CLR     COGDEX          ;Index de boucle de 0->1
COG_LOOP    LDAB    #8              ;
            STAB    SUMDEX          ;
            LDD     #$0000          ;
            STD     SUM_OF_FUZ      ;Somme des sorties floues
            STD     SUM_OF_PROD+1   ; les 16-bits de piods faibel de la ;somme de
                                    produits
            STAA    SUM_OF_PROD     ;les 8 bits de poids forts
SUM_LOOP    LDAB    0,X             ;Prend une sortie floue
            CLRA                    ;Clear les 8-bits de poinds forts
            ADDD    SUM_OF_FUZ      ;Add à la somme des sorties floues
            STD     SUM_OF_FUZ      ;Update les variables de la ram
            LDAA    0,X             ;Prend les sorties floues encore
            LDAB    0,Y             ;Prend la position des singeleton
            MUL                     ;
```

```
            ADDD    SUM_OF_PROD+1 ;Low 16-bits de la somme des produits
            STD     SUM_OF_PROD+1 ;Update low 16-bits
            LDAA    SUM_OF_PROD   ;Upper 8-bits
            ADCA    #0            ;Add carry du 16-bit add
            STAA    SUM_OF_PROD   ;Upper 8-bits de  la somme des 24-bit
            INY                   ;Pointe vers la prochaine position du
                                  ;singeleton
            INX                   ;Pointe vers la prochiane sortie ;floue
            DEC     SUMDEX        ;
            BNE     SUM_LOOP      ;Pour toutes les étiquettes
            PSHX                  ;Save index
            CLRA                  ;
            LDX     SUM_OF_FUZ    ;
            BEQ     SAV_OUT       ;Branch si denominateur est 0
            TST     SUM_OF_PROD   ;voir si c'Est > 16-bit
            BNE     NUM_BIG       ;si non zero, # est > 16-bits
            LDD     SUM_OF_PROD+1 ;Numerateur pour la division
            IDIV                  ;Resultat dans les low 8-bits de X
            XGDX                  ;Resultat maintenant dans B
            TBA                   ;Move le resultat dans A
            BRA     SAV_OUT       ;Go save output
NUM_BIG     LDD     SUM_OF_PROD   ;les upper 16 de 24-bit
            TST     SUM_OF_PROD+2 ;Check for round ing error
            BPL     NO_ROUND      ;SI MSB clear, ne pas arrondir
            ADDD    #1
NO_ROUND    FDIV                  ;D/X -> X, utilise les upper 8 de 16
            XGDX                  ;Resultat maintenant dans A
SAV_OUT     LDX     #COG_OUTS     ;Pointe vers 1ier sortie de defuz
            LDAB    COGDEX
            ABX
            STAA    0,X           ;Update defuzzified output
            PULX                  ;Recouvre l'index
            INCB                  ;Incremente l'indexe
            STAB    COGDEX        ;Update
            CMPB    #NUMOUT
            BNE     COG_LOOP
*****
* Le moteur d'inference a complété un passage de toute les règles
*****
            LDAB  #NUMOUT-1
            LDX     #COG_OUTS
            LDY     #FUZ_OUT1
CPY_LOOP1   LDAA    0,X
            CLR     0,Y
            STAA    1,Y
            INX
            INY
            INY
            DECB
            BPL     CPY_LOOP1
*           CLI
            RTS
```

## BIBLIOGRAPHIE


[1] A. V. Beznos et A. M. Formalsky, *Control of Autonomous Motion of Two-Wheel Bicycle with Gyroscopic Stabilisation*, Proceedings of the 1998 IEEE International Conference on Robotics ans Automation, p.2670-2676, Mai 1998.

[2] L. Dubois, J. Rozinoer et P. Borne, *Introduction à la commande floue*, Éditions Techenip, p.24-29, 1995.

[3] L. P. Holmblad, J. J. Ostergaad, C*ontrol of cement kiln by Fuzzy Logic*, Fuzzy Information and Decision Processes, North-Holland Publishing Company, p.389-399, 1982.

[4] C. C. Lee, *Fuzzy logic in control systems :Fuzzy logic controllers -Part I,* IEEE Transactions on systems, Man, and Cybernitics, vol. 20, N°2, 1990.

[5] C. C. Lee, *Fuzzy logic in control systems :Fuzzy logic controllers -Part 2,* IEEE Transactions on systems, Man, and Cybernitics, vol. 20, N°2, 1990.

[6] C. T. Lin, G. Lee, *Neuro-fuzzy systems,* Prentice Hall Edition, p.145,156,551-554, 1994.

[7] E. H. Mamdani, W. J. Kickert, *Analysis of Fuzzy Logic Controller*, Fuzzy Sets and Systems, North-Holland Publishing Company, vol. 1, p.29-43, 1978.

[8] E. H. Mamdani, T. J. Proczyk, *A Linguistic Self Organizing Controller*, Automatica p.15-30, 1978.

[9] P. C. Rhodes, S. M. Menani, *Towards a Fuzzy logic Programming System : a 1st order Fuzzy logic*, Knowledge Based Systems, vol.5, no. 2, Butterworth-Heinemann Ltd. P.106-116,1992.

[10] M. Sugeno, K. Murakamati, *An experimental study of parking control using a model car*, Industrial Application of Fuzzy control, M, (ed.), Elsevier Science Publishers B.V. (North-Holland), 1985.

[11] M. Sugeno, T. Takagi, *Fuzzy Identification of Systems and Its Applications to Modeling and Control,* IEEE Transactions on systems, Man, and Cybernitics, vol.15, no. 1, p.116-132, 1985.

[12] N. Takashi, J. Nishino, *Modeling of the Running Bicycle and Collision Avoidance Control*, Memories of the Fukui University (Japan), 1996.

[13] J. R.Tong, *La logique floue,* Edition Hèrmes, p. 111-117, 1995.

[14] L. A. Zadeh, *Fuzzy sets as a basis for a theory of possibility*, Fuzzy sets and Systems, vol. 1, North-Holland Publishing Company, P.3-28, 1978.

[15] L. A. Zadeh, *Knowledge Representation in Fuzzy logic*, IEEE Transactions on Knowledge and Data Engineering, vol. 1, no. 1, p.89-100, 1989.