



FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT D'AUTOMATIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Spécialité: **Commande des Systèmes**
Filière: **Génie électrique**

Présenté par
AMRANI Rachid
BOUMEGHAR Saïd

Mémoire dirigé par ALLAD Mourad

Thème

Etude et Développement d'une Plate Forme de Commande (TOR, PID et Floue) de la Station de Pression Sous Environnement LabVIEW

Mémoire soutenu publiquement le 02/07/2015 devant le jury composé de :

M. MOULA Belkacem

MCB, UMMTO, Président

M. Mourad ALLAD

MAA, UMMTO, Encadreur

M. CHARIF Moussa

MAA, UMMTO, Examineur

M. SAIDI Khayreddine

MAA, UMMTO, Examineur

Remerciements

A travers ce modeste travail, nous tenons à remercier vivement notre promoteur M. Mourad ALLAD pour l'intéressante documentation qu'il à mise à notre disposition, pour ses conseils précieux et pour toutes les commodités et aisances qu'il nous a apporté durant notre étude et réalisation de ce projet.

Nos remerciements les plus vifs s'adressent aussi aux messieurs le président et les membres de jury d'avoir accepté d'examiner et d'évaluer notre travail.

Nous exprimons également notre gratitude à tous les professeurs et enseignants qui nous ont guidé a travers notre parcours, sans omettre bien sur de remercier profondément tous ceux qui ont contribué de près ou de loin à la réalisation du présent travail.

Et enfin, que nos chers parents et familles, et bien avant tout, trouvent ici l'expression de nos remerciement les plus sincères et les plus profonds en reconnaissance de leurs sacrifices, aides, soutien et encouragement afin de nous assurer cette formation dans les meilleurs conditions.

Dédicaces

*À
Mes chers parents*

Said

Dédicaces

« L'eau coule grâce à sa source

L'arbre pousse grâce à ses racines »

A

Mes parents

*Pour les sacrifices déployés à mon égard ; pour leur patience
Leur amour et leur confiance en moi*

*Ils ont tout fait pour mon bonheur et ma réussite.
Qu'ils trouvent dans ce modeste travail, le témoignage de ma
Profonde affection et de mon attachement indéfectible.
Nulle dédicace ne puisse exprimer ce que je leur dois
Que dieu leur réserve la bonne santé et une longue vie.*

A Mes frères et à mes sœurs

*Qui n'ont cessé d'être pour moi des exemples de persévérance, de courage et de
générosité.*

A

Mes amis

*En témoignage de mes sincères reconnaissances pour les efforts
Qu'ils ont consentis pour me soutenir au cours de mes études.
Que dieu nous garde toujours unis*

A

Toute personne qui nous a aidé à faire ce projet.

Rachid

Nomenclature

$X(t)$: Mesure

$\varepsilon(t)$: Erreur

$Y(t)$: Signal de commande

Y_0 : Talon réglable

θ : Constante de temps

G_s : Gain statique

p : Opérateur de Laplace

ω_n : Pulsation propre

$D\%$: Dépassement

h : Coefficient d'amortissement

τ : Retard

P : Régulateur proportionnel

PI : Régulateur proportionnel Intégral

PID : Régulateur proportionnel Intégral Dérivé

G_r : Gain de régulateur

T_i : Temps d'intégration

T_d : Temps dérivatif

$\varepsilon(p)$: Erreur

$W(p)$: Consigne

θ_d : La constante de temps désiré

$z(t)$: Perturbation

T_e : Période d'échantillonnage

G_m : Marge de gain

φ_m : Marge de phase

CAN : Convertisseur Analogique/Numérique

CNA : Convertisseur Numérique /Analogique

t_M : Temps de montée

t_r : Temps de réponse

PE : Capteur de pression

PT : Transmetteur de pression.

PR : Enregistreur de pression (Pressure Recorder)..

PIC : Régulateur indicateur de niveau (Level Indicator Controller).

SP : SetPoint (mesure)

PV : Process Variable (mesure)

W : Consigne

μ : Degré d'appartenance

Sommaire

Introduction générale	1
------------------------------------	---

Chapitre 01 : Identification et Régulation des systèmes

1.1. Introduction	3
1.2. Modélisation et identification.....	3
1.2.1. Modélisation.....	4
1.2.2. Identification des procédés	5
1.3. Régulation	12
1.3.1. Définition	12
1.3.2. Objectif de la régulation	12
1.3.3. Chaîne d'une régulation	12
1.3.4. Les Formes fondamentale de régulation	13
1.3.5. Régulateur PID	15
1.3.6. Régulation TOR (Tout ou Rien).....	19
1.4. Qualité d'une régulation.....	20
1.5. Régulation numérique	22
1.5.1. Intérêt de la régulation numérique	23
1.5.2. Choix du pas d'échantillonnage	24
1.5.3. Théorème de Shannon	24
1.5.4 Application aux boucles d'asservissement.....	24
1.6. Conclusion.....	25

Chapitre 02 : Logique floue

2.1. Introduction	26
2.2. Sous-ensembles flous	27
2.3. Variables Linguistiques.....	28
2.4. Opérateurs de la logique floue	28
2.4.1. Opérateur NON	28

2.4.2. Opérateur ET	29
2.4.3. Opérateur OU	30
2.5. Réglage et commande par logique floue	31
2.5.1. Introduction	31
2.5.2. Structure d'une commande floue	31
2.5.3. Fuzzification.....	32
2.5.4. Inférence.....	34
2.5.5. Défuzzification	38
2.6. Différents types de régulateurs flous.....	39
2.6.1. Régulateur de type Mamdani.....	39
2.6.2. Régulateur de type Sugeno.....	40
2.7. Exemple d'application.....	40
2.8. Conclusion	42

Chapitre 03 : Présentation du logiciel LabVIEW.

3.1. Introduction	44
3.2. Présentation des instruments virtuels	45
3.2.1. Face – avant.....	45
3.2.2. Diagramme	46
3.2.3. Icône et connecteur	46
3.3. Les différentes palettes de LabVIEW	47
3.3.1. Palette d'outils.....	47
3.3.2. Palette des commandes.....	48
3.3.3. Palette des fonctions.....	48
3.4. Structure des données dans LabVIEW	49
3.4.1. Les tableaux sous LabVIEW	49
3.4.2. Waveform.....	50
3.4.3. Clusters.....	51
3.5. Structure de programmation dans LabVIEW	51

3.5.1. Structures Séquence	51
3.5.2. Structure Condition	52
3.5.3. Structures d'itération	53
3.6. Traitements de données dans LabVIEW	54
3.7. Graphes dans LabVIEW.....	56
3.7.1. Graphe	56
3.7.2. Graphe déroulant	57
3.7.3. Graphe XY	57
3.8. Conclusion.....	58

Chapitre 04 : Description matérielle

4.1. Introduction	59
4.2. Description de l'unité	59
4.3. Contrôle automatique de la vanne proportionnelle	61
4.4. Transducteur de pression (capteur de pression)	61
4.4.1. Méthodes de mesure de pression.....	61
4.4.2. Transducteur de pression de la station PUP-4.....	61
4.5. Conditionnement du signal du capteur.....	62
4.6. Amplificateur de puissance de la vanne proportionnelle	63
4.7. Carte d'acquisition Labjack	65
4.7.1. Description de la carte Labjack U3_LV	65
4.8. Boucle de régulation PID	69
4.9. Boucle de régulation TOR.....	70
4.10. Boucle de régulation floue	70
4.11. Conclusion.....	70

Chapitre 05 : Simulation et résultats expérimentaux

5.1. Introduction	71
5.2. Plateforme de commande	71
5.2.1. Programmes Labjack sous LabVIEW	72
5.2.2. Réalisation d'une poursuite	73
5.2.3. Archivage des données	74
5.2.4. Programme d'une alarme sur LabVIEW	74
5.3. Caractéristique statique du processus.....	75
5.4. Identification des paramètres du modèle.....	75
5.5. Application de la régulation PI.....	76
5.5.1. Programme d'un régulateur PID parallèle sur LabVIEW	76
5.5.2. Paramètres du régulateur PID	77
5.5.3. Résultats simulation PID avec Labview.....	79
5.6. Application de la régulation floue	80
5.6.1. Programme d'un régulateur flou sur LabVIEW	80
5.6.2. Fonctions d'appartenance	81
5.6.3. Résultats de simulation de la régulation floue	82
5.7. Résultats expérimentaux (TOR, PID, flou).....	83
5.8. Conclusion.....	88

Introduction Générale

L'automatique est l'art de modéliser, identifier et d'analyser puis de commander les systèmes. C'est aussi celui de traiter l'information et de prendre des décisions.

L'Automatique est la science qui traite des lois de régulation des systèmes commandés. Commander un objet (on dit aussi le contrôler ou l'asservir) signifie influencer son comportement pour lui faire effectuer une tâche définie à l'avance. Afin de réaliser en pratique cette « influence », les ingénieurs ont mis au point des mécanismes appropriés faisant appel à des principes théoriques généraux, eux-mêmes s'exprimant à l'aide de divers outils mathématiques.

La commande automatique a considérablement évolué, notamment durant les trois dernières décennies. Son champ d'application s'est élargi pour couvrir actuellement la plupart des industries (pétrochimiques, agroalimentaires, textiles, cimenteries, aérospatial, etc...).

Les bases théoriques de la logique floue ont été établies en 1965 par le professeur Lotfi A. Zadeh de l'Université de Californie de Berkeley. A cette époque, la théorie de la logique floue n'a pas été prise au sérieux. En effet, les ordinateurs avec leurs fonctionnements exact part Tout ou Rien (1 et 0) ont commencé à se répandre sur une large échelle. Par contre la logique floue permet de traiter des variables non exactes dont la valeur peut varier entre 1 et 0. Dès 1975, on trouve les premières applications au niveau des systèmes de réglage. A partir de 1985 environ, ce sont les Japonais qui commencent à utiliser la logique floue dans des produits industriels pour résoudre des problèmes de réglage et de commande [6].

Le langage de programmation LabVIEW (Laboratory Virtual Instrument Engineering Workbench) est un environnement de programmation à caractère universel bien adapté pour la mesure, les tests, l'instrumentation et l'automatisation. C'est un programme dont le but est de

contrôler et commander des processus allant du simple capteur ou de l'actionneur, jusqu'à une chaîne de fabrication complète.

Comme nous allons le voir par la suite, LabVIEW dispose d'un nombre de fonction graphique qui permettent facilement d'acquérir des données, de les traiter et d'afficher les résultats.

Ce mémoire est subdivisé en cinq chapitres:

Le premier chapitre consiste à faire des rappels sur l'identification et la régulation des systèmes.

Le second chapitre est consacré à la logique floue, et à l'étude des différentes étapes nécessaires pour la réalisation d'un régulateur flou.

Le troisième chapitre inclut la présentation de logiciel LabVIEW, c'est-à-dire le contenu et les blocs que nous allons utiliser pour la réalisation de notre plateforme de commande.

La description matérielle est effectuée au quatrième chapitre. Dans ce dernier on va faire la présentation de la station de pression PUP-4/EV, puis on présente la carte d'acquisition Labjack-U3 et ses différents blocs de programmation sous LabVIEW.

Le dernier chapitre présente les différents résultats de simulations et expérimentaux obtenus après identification de notre système, implémentation de trois régulateurs (TOR, PID, Flou) et analyse des résultats obtenus.

Enfin nous terminons notre travail par une conclusion générale et perspective.

Chapitre 1

Identification et Régulation **des Systèmes**

Chapitre 1

Identification et Régulation des Systèmes

1.1. Introduction

Le développement des modèles mathématiques est un problème majeur pour l'application des techniques avancées de l'analyse, telles que la commande, l'optimisation, la surveillance et le diagnostic des processus. Identifier un système c'est de le représenter par un modèle, à partir de la connaissance expérimentale des entrées-sorties, de manière à obtenir une identité de comportement.

L'objectif de l'identification est de calculer les paramètres d'un modèle du procédé, de façon à ce que le comportement de procédé et celui de modèle soient identique, et ceci pour toutes les séquences de variables d'entrées habituellement utilisées.

La régulation des procédés est d'assurer le fonctionnement selon des critères prédéfinis par un cahier des charges tels que la stabilité, la précision et la rapidité. Obtenir un débit de fluide constant dans une conduite en fonction des besoins, ou faire évoluer une température d'un four selon un profil déterminé, se sont des exemples de la régulation.

1.2. Modélisation et identification

1.2.1. Modélisation [1]

La recherche d'un modèle mathématique pour un procédé est nécessaire et doit aboutir à un modèle représentant correctement le comportement du procédé. Cependant le modèle ne doit ni être trop complexe au risque d'être incompatible avec le correcteur disponible, ni être trop simple pour ne pas masquer certains aspects néfastes au bon fonctionnement. Simplifier une petite constante de temps ou un retard dans un modèle ou ne pas prendre en compte la variation de gain statique est effectivement source d'instabilité du procédé.

1.2.1.1. Objectif de la modélisation

La modélisation a pour but :

- D'avoir une meilleure compréhension des phénomènes.
- Dimensionnement d'une installation.
- Formation des opérateurs.
- Conception du système de commande pour :
 - La mise au point de la stratégie de commande.
 - Conception de la loi de commande et son réglage.
 - Conception de capteurs logiciels ou estimateur d'état du système.

1.2.1.2. Différents types de modèle

Dans la modélisation, il est important de distinguer les différents types de modèles qui ont pour objectif de décrire le système avec plus ou moins de détails.

On distingue trois types de modèles qui sont :

a. Modèle de connaissance « Boîte blanche »

Ils sont élaborés à partir des lois de la physique ou de la chimie. Leur objectif principal est d'expliquer un phénomène par une relation mathématique.

Les équations physiques ne sont pas toujours données par le fournisseur, elles conduisent souvent à des développements mathématique trop complexes pour être exploitées au sens de l'automatique.

b. Modèle de comportement « Boîte noire »

Ce sont des modèles linéaires, dont la validité reste limitée à des petites variations autour de point de fonctionnement. Les petites variations de l'entrée autour d'un point de fonctionnement peuvent être reliées à des petites variations de la sortie par un modèle dynamique linéaire.

c. Modèle intermédiaire « Boîte grise »

Ils constituent un hybride entre les deux types précédents. On peut les considérer comme des modèles de connaissance simplifiés.

1.2.2. Identification des procédés

1.2.2.1. Méthode du premier ordre [2]

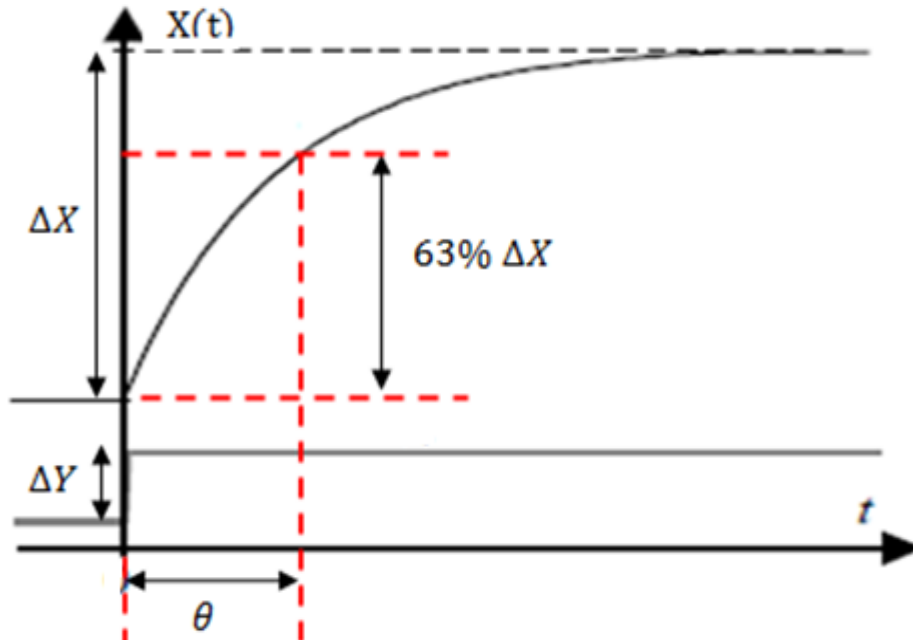


Figure 1.1 : Système du premier ordre

Après avoir exercé à $t=0$ un échelon d'amplitude ΔY en entrée d'un procédé, on obtient la courbe $X(t)$ (voire figure 1.1).

L'allure de cette courbe est celle de la réponse indicielle d'un système du premier ordre :

$$\text{Qui est sous la forme suivante : } G(p) = \frac{G_s}{\theta p + 1} \quad (1.1)$$

Pour déterminer les paramètres de modèle, on procède comme suit:

- Le gain statique est mesuré directement

$$G_s = \frac{\Delta X}{\Delta Y} \quad (1.2)$$

- Pour déterminer la constante de temps θ : On trace conjointement la droite d'ordonnée $(0.63\Delta X)$ parallèle à l'axe des abscisses.

1.2.2.2. Modèle du second ordre

La réponse d'un système du second ordre soumis à un échelon est indiquée sur la figure 1.2.

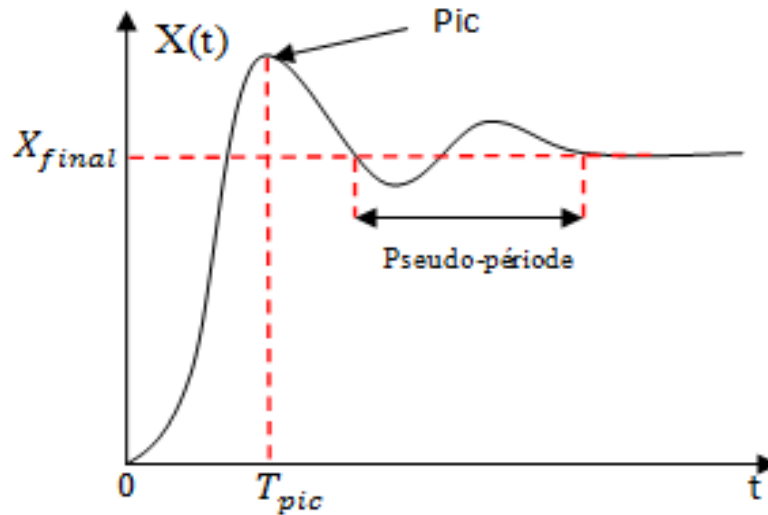


Figure 1.2 : Réponse indicielle d'un système du deuxième ordre

L'allure de cette courbe est celle de la réponse indicielle d'un système du deuxième ordre :

Qui est sous la forme suivante : $G(p) = \frac{Gs}{1 + \frac{2h}{\omega_n}p + \frac{p^2}{\omega_n^2}}$ (1.3)

Pour déterminer les paramètres de modèle on procède comme suit :

- Le facteur d'amortissement « h » s'obtient à partir de la mesure du dépassement.

$$D\% = 100e^{\frac{-\pi h}{\sqrt{1-h^2}}} \quad (1.4)$$

- La pulsation propre « ω » s'obtient à partir de l'une des deux équations suivantes :

$$T_{pic} = \frac{\pi}{\omega_n \sqrt{1-h^2}} \quad (1.5)$$

$$T_{pseudo-périodique} = \frac{2\pi}{\omega_n \sqrt{1-h^2}} \quad (1.6)$$

1.2.2.3. Méthode de Broïda [2]

La méthode de Broïda consiste en une identification en boucle ouverte d'un procédé autorégulant adapté au modèle :

$$G(p) = \frac{G_S}{\theta p + 1} e^{-\tau p} \quad (1.7)$$

Pour appliquer la méthode d'identification de Broïda, la réponse obtenue à un incrément de commande doit être en forme de « S » bien prononcé, comme celle de la figure 1.3.

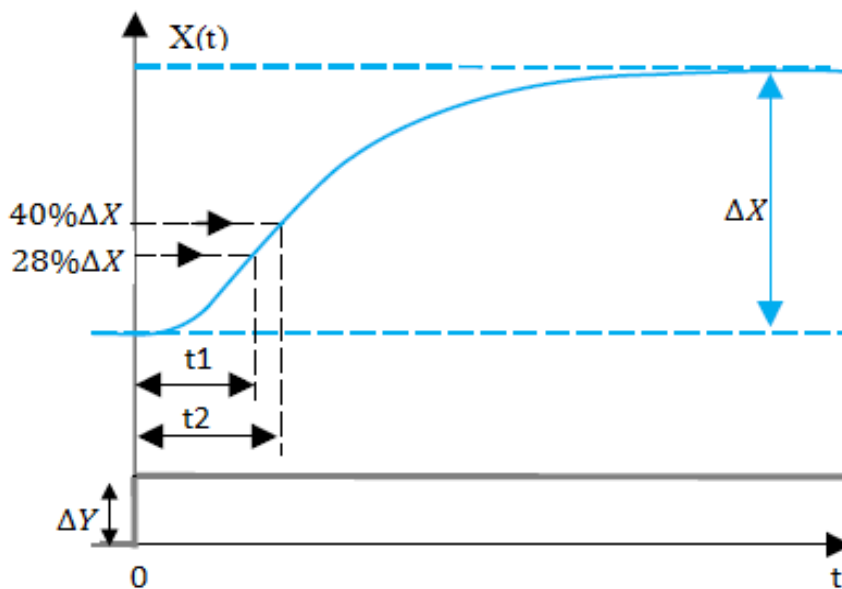


Figure 1.3 : Courbe en « S » analysée par la méthode de Broïda

Détermination du modèle de Broïda :

Le procédé est représenté par le modèle de Broïda : $G(p) = \frac{G_S}{\theta p + 1} e^{-\tau p}$

Le gain statique (G_S) :

$$G_S = \frac{\Delta X}{\Delta Y} \quad (1.8)$$

- La constante de temps (θ) :

$$\theta = 5.5(t_2 - t_1) \quad (1.9)$$

- Le retard ou temps mort (τ) :

$$\tau = (2.8 t_1 - 1.8 t_2) \quad (1.10)$$

1.2.2.4. Méthode de Strejc [2]

La méthode de Strejc d'identification en boucle ouverte s'applique à un procédé naturellement stable ou procédé autorégulant.

Pour appliquer la méthode d'identification de Strejc, la réponse obtenue à un changement en échelon de la commande doit être en forme « S » avec un point d'inflexion bien marqué comme celle de la figure 1.4.

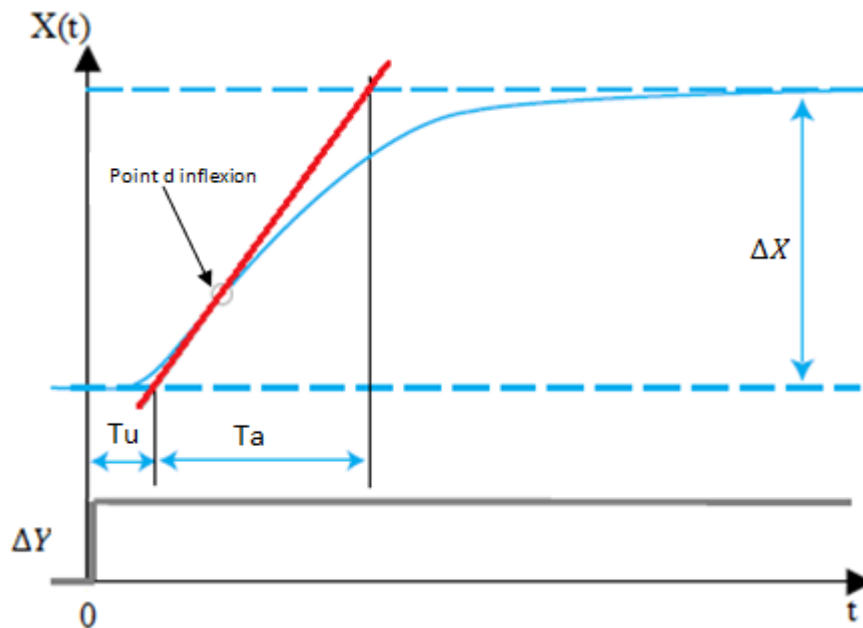


Figure 1.4 : Courbe en « S » analysée par le modèle de Strejc

Le procédé est représenté par le modèle de Strejc : $G(s) = \frac{Gs}{(\theta p + 1)^n} e^{-\tau p}$ (1.11)

- Le gain statique est calculé comme suit :

$$G_s = \frac{\Delta X}{\Delta Y} \quad (1.12)$$

Détermination de (n) :

Tout d'abord on trace le mieux possible la tangente au point d'inflexions de la réponse indicielle, afin de définir les deux grandeurs T_u et T_a .

Après on calcule le rapport $\mu = \frac{T_u}{T_a}$ (1.13)

On choisira de tableau (1.1) la valeur de $\frac{T_u}{T_a}$ immédiatement inférieur de $\mu = \frac{T_u}{T_a}$ qu'on a déjà calculée. Cette ligne permet d'obtenir l'ordre (n) du modèle.

- Détermination de la constante de temps (θ) :

De tableau (1.1) on cherche le rapport ($\frac{\theta}{T_a}$) qui correspond à l'ordre (n) d'où on obtient la valeur(θ) .

- Détermination de temps de retard (τ) :

$$\tau = T_u, \text{ mesuré} - \frac{T_u}{T_a} \left| \text{tableau} . T_a, \text{ mesuré} \right. \quad (1.14)$$

Le tableau suivant permet d'obtenir la constante de temps (θ) et l'ordre (n).

L'ordre (n)	$\frac{T_u}{T_a}$	$\frac{\theta}{T_a}$
1	0	1
2	0.105	0.37
3	0.22	0.27
4	0.32	0.22
5	0.41	0.20
6	0.49	0.18
7	0.57	0.19
8	0.64	0.15
9	0.71	0.14
10	0.77	0.13

Tableau 1.1 : Tableau de Strejc

1.2.2.5. Méthode de réglage PID [2]

Un régulateur est constitué d'un comparateur pour observer l'écart entre la mesure et la consigne, et d'un correcteur (algorithme) qui permet d'obtenir une loi d'évolution de la mesure du procédé conforme au cahier des charges.

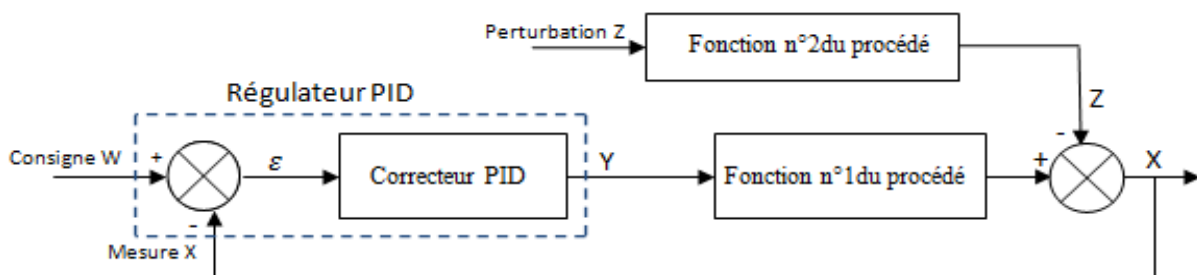


Figure 1.5 : Régulateur PID

Le régulateur doit maintenir une mesure égale à une consigne quelles que soient les perturbations qui agissent sur le procédé. Le réglage des paramètres de son algorithme dépend des performances attendues telles que le temps de réponse et la précision lors de la régulation (application de perturbation), et aussi de poursuite (changement de consigne).

1.2.2.6. Méthode basée sur un modèle de réponse à l'échelon

▪ Méthode de Ziegler-Nichols

C'est une méthode empirique qui permet d'ajuster les paramètres d'un régulateur PID pour commander un processus à partir de mesure sur sa réponse indicielle.

La réponse à un échelon d'amplitudes E_0 , sans oscillation, sera assimilée à celle de premier

ordre avec retard : $G(p) = \frac{Gs}{\theta p + 1} e^{-\tau p}$ (1.15)

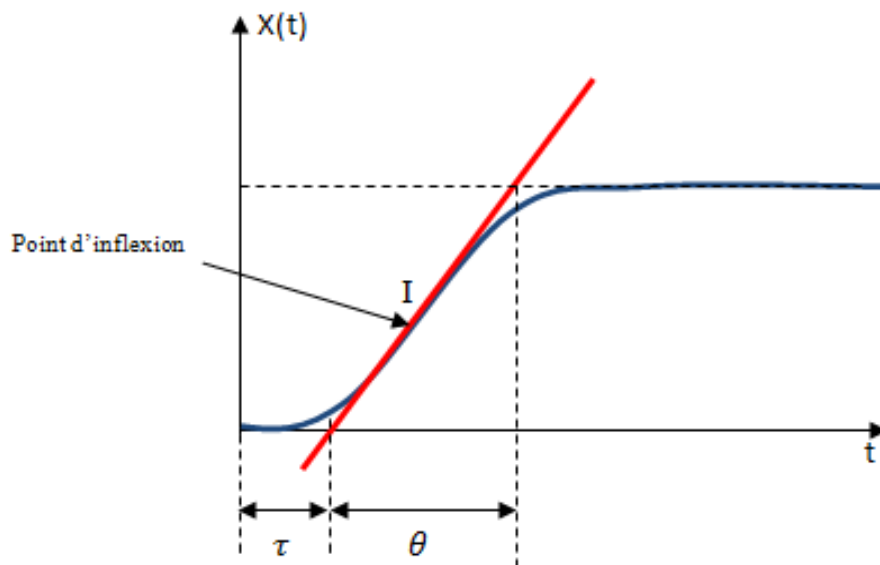


Figure 1.6 : Identification pour (Ziegler-Nichols)

La tangente au point d'inflexion « I » est définie comme étant :

$$I = \frac{Gs}{\theta} \quad (1.16)$$

Pour définir les paramètres des correcteurs P, PI, PID, il suffit d'appliquer les relations indiquées sur le tableau 1.2 suivant :

Type de correcteur	G_r	T_i	T_d
P	$\frac{E_0}{\tau.I}$		
PI	$\frac{0.9}{\tau.I}$	0.33τ	
PID	$\frac{1.2}{\tau.I}$	2τ	0.5τ

Tableau 1.2 : Calcul des paramètres des correcteurs par la méthode de Ziegler-Nichols

Ziegler-Nichols propose des règles de correcteur P, PI ou PID pour avoir une réponse en boucle fermée satisfaisante, avec dépassement initial de l'ordre 30 à 40% et avec un rapport d'amplitudes d'oscillations de 0.25.

▪ **Méthode de modèle de référence [2]**

Pour la boucle de régulation suivante

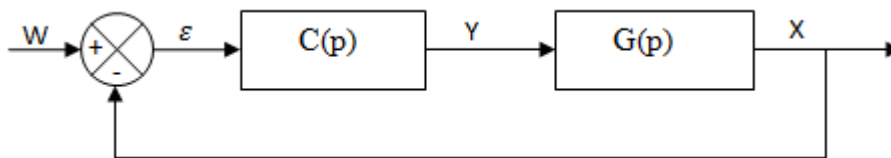


Figure 1.7 : Boucle de régulation en boucle fermée

En imposant un modèle de référence en boucle fermée du premier ordre $F(p) = \frac{1}{1+\theta_d P}$, la fonction de transfert du correcteur devient : $C(p) = \frac{1}{\theta_d \cdot P \cdot G(p)}$; soit $C(p) = \frac{1+\theta P}{G_s \cdot \theta_d \cdot P}$

θ_d : Constante de temps désiré.

$F(p)$: La fonction de transfert en boucle fermée.

$G(p)$: La fonction de transfert du procédé.

La fonction de transfert $C(p)$ correspond à un régulateur PI, ($C(p) = G_r + \frac{1}{T_i p}$) dont les valeurs des coefficients dépendront de la structure de régulateur réel disponible.

Les valeurs du gain G_r et de la constante de temps intégrale T_i de régulateur sont calculées avec les coefficients G_s et θ_d du procédé et de la constante θ_d fixée.

	PI Parallèle	PI Série	P
Gr	$\frac{1}{G_s} \frac{\theta}{\theta d}$	$\frac{1}{G_s} \frac{\theta}{\theta d}$	$\frac{1}{G_s} \frac{\theta}{\theta d}$
P	$G_s \theta d$	θ	

Tableau 1.3 : Paramètres du régulateur

1.3. Régulation

1.3.1. Définition [5]

La régulation est essentielle dans le domaine de l'industrie car elle permet de contrôler nombreuses actions nécessaires au fonctionnement des systèmes. La régulation peut également compenser un certain nombre de mauvais fonctionnement liés à la conception ou à la mise en œuvre de ces circuits. Toutefois l'objectif principal est d'adapter au plus proche la production aux besoins.

Le système mis en œuvre pour la régulation est constitué principalement de :

- capteur,
- Régulateur,
- Organe de réglage.

1.3.2. Objectif de la régulation [2]

La régulation a pour but de garantir le bon fonctionnement du procédé selon un objectif détaillé dans le cahier des charges. La régulation a pour objectif de contrôler une ou plusieurs grandeurs mesurées quelles que soient les perturbations qui agissent sur le procédé, tout en assurant la stabilité, la précision et la rapidité du procédé.

1.3.3. Chaîne de régulation [1]

Afin d'obtenir le fonctionnement désiré, la régulation doit agir en continu sur le procédé.

Pour cela il faut observer la grandeur à maîtriser (observation), comparer cette grandeur à celle désirée et déterminer l'action à entreprendre (réflexion) puis agir sur une ou plusieurs grandeurs incidentes du procédé (action). On obtient alors une chaîne de régulation (figure 1.7)

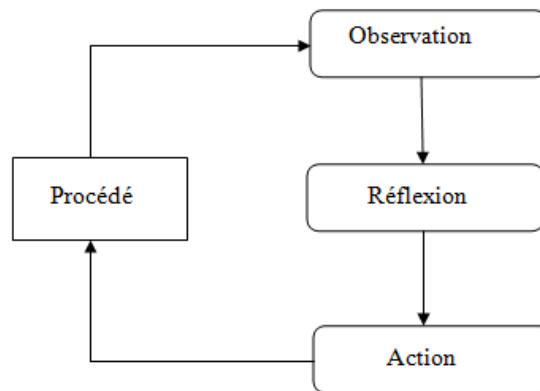


Figure 1.8 : Chaîne de régulation

1.3.4. Forme fondamentale de régulation

1.3.4.1. Régulation en boucle ouverte [2]

En boucle ouverte l'observation n'est pas celle de la grandeur à maîtriser mais celle d'une grandeur incidente. La réflexion est l'étape où la commande prend en compte une relation préétablie entre la grandeur observée et la grandeur incidente sur laquelle on agit. L'action modifie alors la grandeur à maîtriser.

Cette chaîne de régulation est dite ouverte car l'action ne modifie pas la grandeur observée.

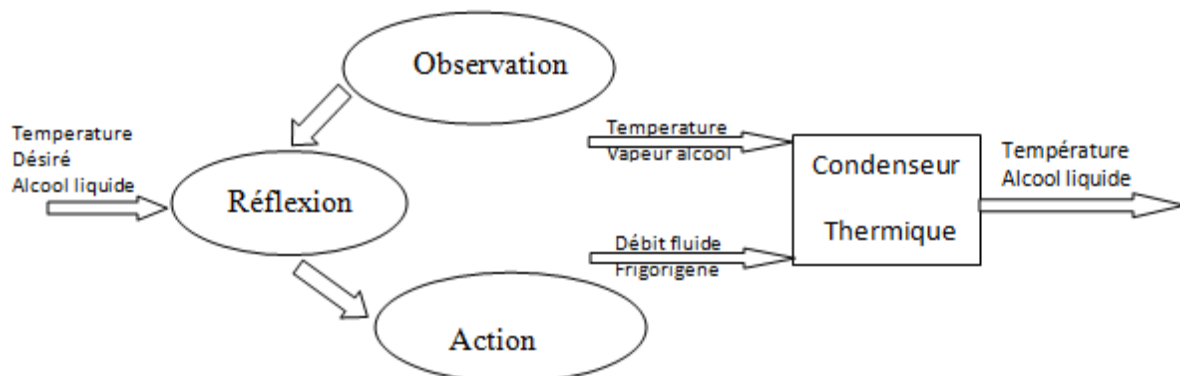


Figure 1.9 : Régulation en chaîne ouverte

Cette régulation forme une boucle ouverte car l'action ne modifie pas la grandeur mesurée.

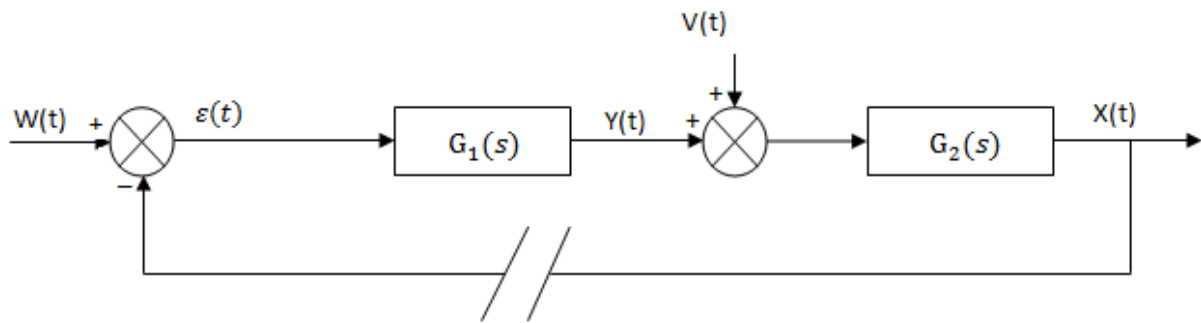


Figure 1.10 : Diagramme fonctionnel en boucle ouverte

1.3.4.2. Régulation en boucle fermée [2]

En boucle fermée l'observation se porte sur la grandeur à maîtriser. L'étape de réflexion détermine l'écart entre la grandeur observée et la grandeur à maîtriser. En fonction de cet écart, des règles d'évolutions fixées, on en déduit l'action à entreprendre. L'action modifie la grandeur incidente réglante et donc la grandeur à maîtriser.

Cette chaîne de régulation est dite fermée car l'action modifie la grandeur observée.

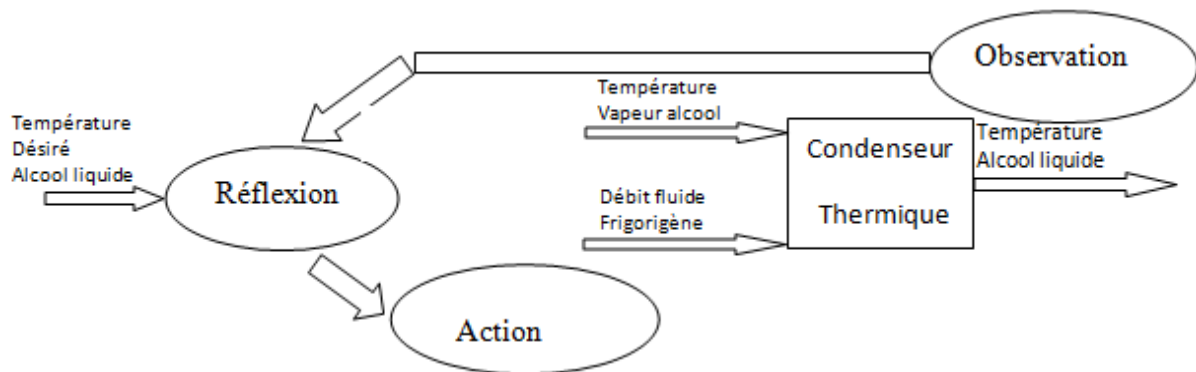


Figure 1.11 : Régulation en chaîne fermée

Cette régulation forme une boucle fermée car l'action modifie la grandeur mesurée. Elle comporte une contre réaction ou retour d'information.

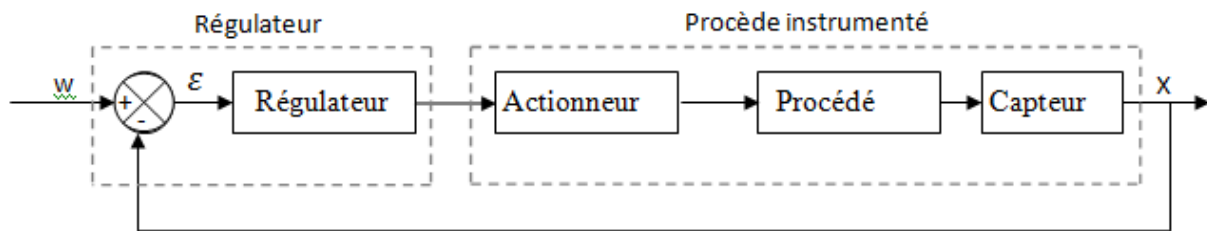


Figure 1.12 : Constitution d'une chaîne fermée de régulation [1]

	Boucle ouverte	Boucle fermée
Avantage	-Pas de problème de stabilité. -Simple et rapide à mettre en place	-Pas de problème de précision et de temps de réponse. -L'effet des perturbations est pris en compte -La linéarité de procédé améliorée
Inconvénient	-Impossible de régler un procédé intégrateur. -L'objectif n'est pas toujours atteint. -Précision et temps de réponse.	-La stabilité doit être étudiée -Etudes et mise au point peuvent être complexes

Tableau 1.4 : Comparaison entre chaîne ouverte et boucle fermée [2]

1.3.5 : Régulateur PID [5]

Le régulateur PID a pour fonction les trois actions suivantes :

- Proportionnelle,
- Intégrale,
- Dérivée.

Pour un régulateur PID, il existe plusieurs possibilités d'associer l'action intégrale et dérivée à l'action proportionnelle. On note : PID série, PID parallèle et PID mixte.

1.3.5.1 : Aspect fonctionnel et structurel d'un régulateur PID

a) Action proportionnelle

L'action proportionnelle applique une correction instantanée pour tout écart entre la mesure et la consigne, plus la perturbation est grande, plus la correction apportée est grande.

L'équation de régulateur en fonction de temps et donnée comme suit :

$$Y(t) = K_C \cdot \varepsilon(t) \quad (1.17)$$

L'effet de l'action proportionnelle est de réduire l'écart statique. Plus l'action est grande plus l'écart est réduit, mais un excès d'actions conduit à l'instabilité de la boucle.

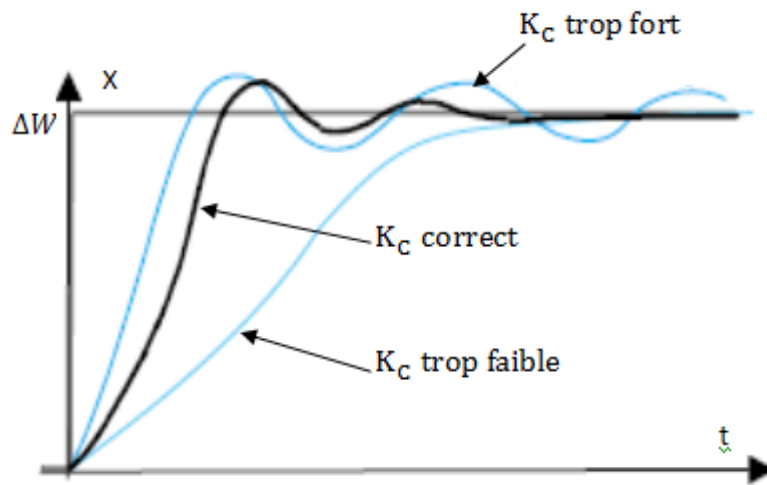


Figure 1.13 : Influence de l'action proportionnelle [2]

b) Action intégrale

Cette action est complémentaire à l'action proportionnelle, elle permet de stabiliser dans le temps l'action proportionnelle, plus l'erreur mesurée est constante plus la correction est constante. L'action intégrale est conditionnée par le temps d'intégration T_i .

$$Y(t) = \frac{1}{T_i} \int_0^t \varepsilon(t) dt \quad (1.18)$$

L'action intégrale permet de supprimer l'écart statique, plus la constante T_i est petite plus l'action intégrale est forte, une action excessive (T_i trop faible ou K_c trop grand) conduit à une instabilité de la boucle (augmentation de déphasage).

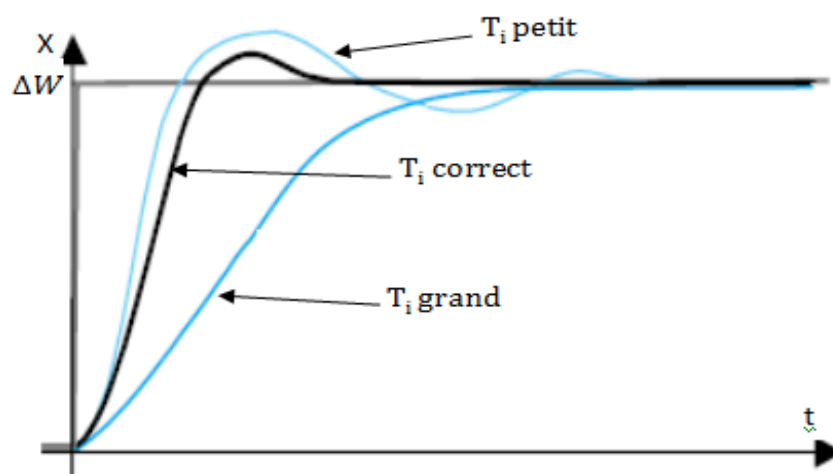


Figure 1.14 : Influence de l'action intégrale [2]

c) Action dérivée

Cette action permet d'anticiper la réponse de la régulation en cas de perturbations rapide ou de modification de la consigne.

$$Y(t) = T_d \frac{d \varepsilon(t)}{dt} \quad (1.19)$$

Avec T_d : constante de temps dérivée

Effet de l'action dérivée :

Plus la constante T_d est grande plus l'action dérivée est forte,

L'action dérivée bien dosée permet de :

- réduire le dépassement ou les oscillations obtenues en action proportionnelle seule.
- d'accélérer la réponse de la mesure.
- d'améliorer la stabilité de la boucle.

Un excès de l'action dérivée conduit à l'instabilité du système bouclé.

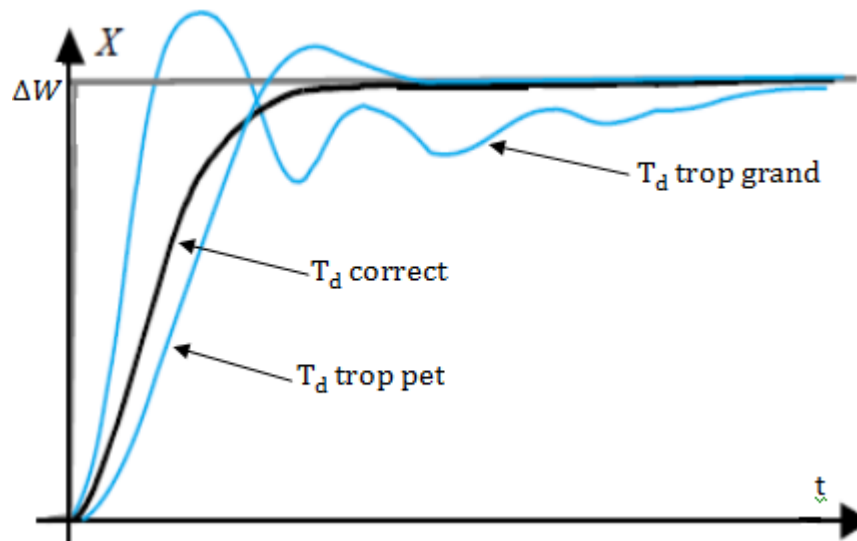


Figure 1.15 : Influence de l'action dérivée [2]

d) Principe de l'action PI

A l'action proportionnelle, on ajoute une action égale à l'intégrale de l'écart divisée par une constante de temps T_i . L'action ne s'annule plus quand l'erreur est nulle. Cette action introduit un retard par rapport à une action proportionnelle pure. Il est possible d'atteindre la valeur de la consigne au prix d'oscillations lors du démarrage du processus.

On peut limiter celles-ci en introduisant une autre correction (action dérivée) qui permet d'anticiper les dépassement de consigne.

$$Y(t) = K_c \left(\varepsilon(t) + \frac{1}{T_i} \int_0^t \varepsilon(t) dt \right) \quad (1.20)$$

e) Principe de l'action proportionnelle dérivée

En principe, le régulateur ne fonctionne pas en action dérivée pure. Mais on lui associe une action proportionnelle. Son équation est donnée comme suit :

$$Y(t) = K_c \left(\varepsilon(t) + T_d \frac{d\varepsilon(t)}{dt} \right) \quad (1.21)$$

f) Principe de l'action proportionnelle intégrale et dérivée

C'est le type d'algorithme le plus utilisé. Il est en général disponible sur la plupart des régulateurs actuellement utilisés (régulateurs PID).

Il permet la stabilisation de la mesure au point de consigne en un temps minimum.

Son équation est donnée comme suit :

$$Y(t) = K_c \left(\varepsilon(t) + \frac{1}{T_i} \int_0^t \varepsilon(t) dt + T_d \frac{d\varepsilon(t)}{dt} \right) \quad (1.22)$$

1.3.5.2 Différentes structures PID [4]

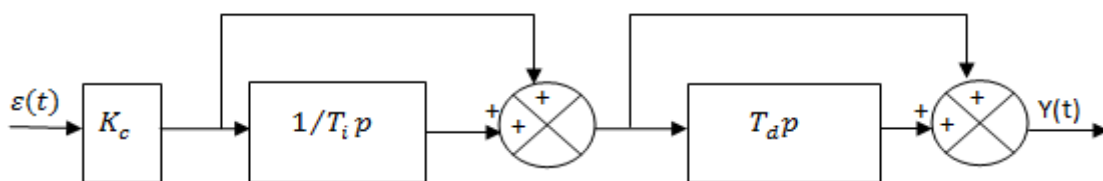


Figure 1.16 : Structure série d'un PID

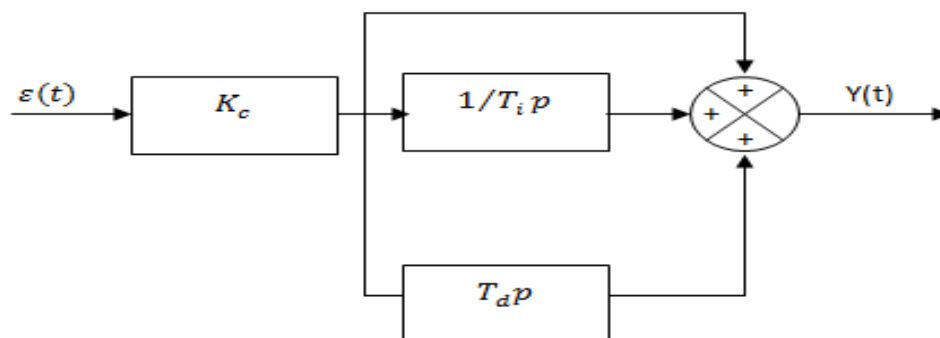


Figure 1.17 : Structure mixte d'un PID

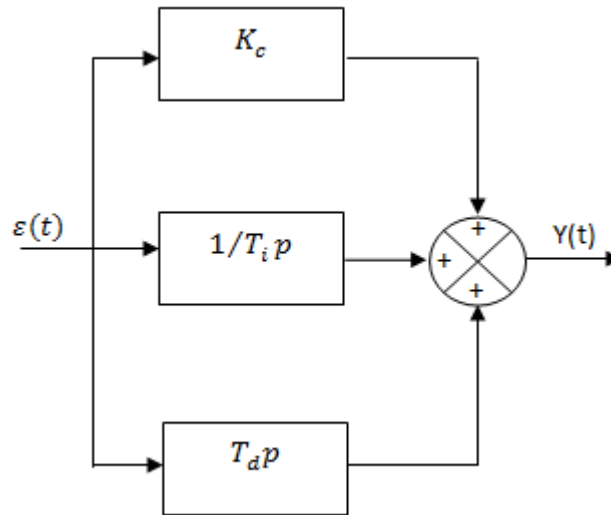


Figure 1.18 : Structure parallèle d un PID

1.3.6. Régulation TOR (Tout ou Rien)

Mode de régulation à deux positions qui sont généralement les positions extrêmes de l'élément de contrôle final (ouvert et fermé). C'est par exemple les positions extrêmes marche et arrêt d'un compresseur frigorifique.

Un régulateur «Tout ou Tien» est un régulateur qui élabore une action de commande discontinue qui prend deux positions ou deux états 0 et 1 (ou 0 et 100%).

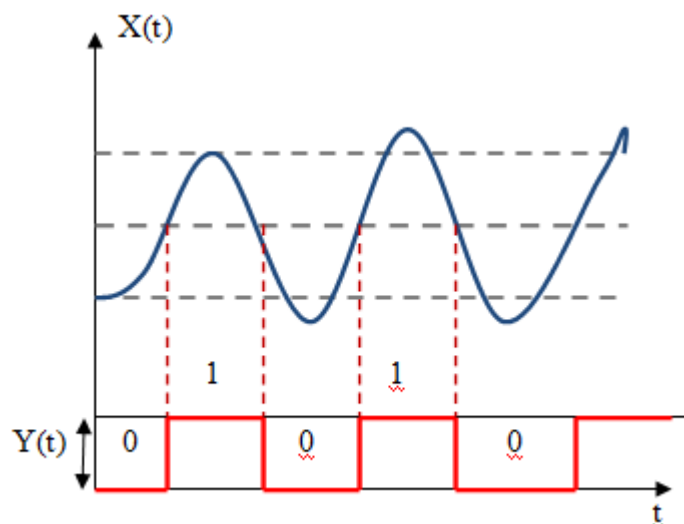


Figure 1.19 : Régulation en Tout ou Rien

1.4. Qualité d'une régulation [2]

La première qualité à assurer d'une régulation est la stabilité puisque toute instabilité conduit à la perte de contrôle du procédé.

La précision, statique ou dynamique, est souvent la deuxième qualité attendue d'une régulation. La rapidité est une qualité opposée à la précision dynamique et liée à l'amortissement.

1.4.1. Stabilité

- **Stabilité dans le domaine temporelle**

La boucle de régulation est stable lorsqu'elle est soumise à un incrément de consigne ou d'une grandeur perturbatrice, la mesure retrouve un état stable.

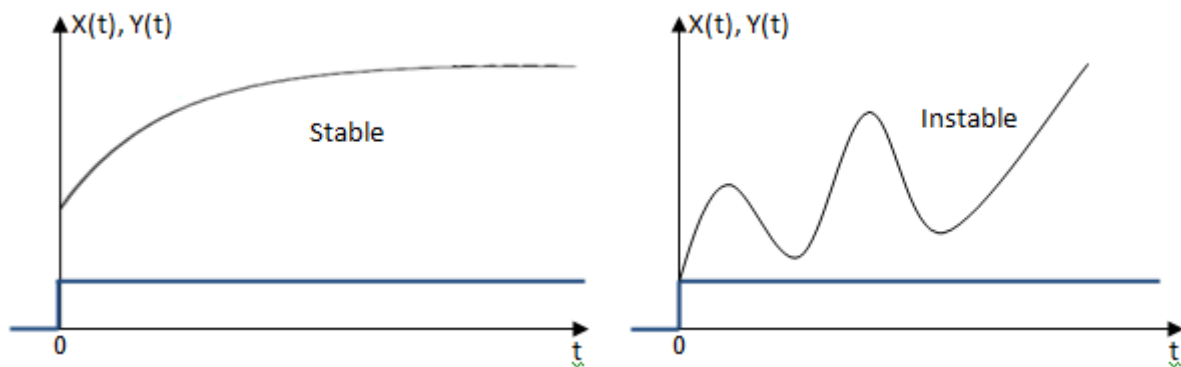


Figure 1.20 : Exemple sur la stabilité

- **Stabilité dans domaine fréquentiel**

Un système asservi à retour unitaire est stable si, en décrivant la courbe représentative dans le diagramme de Black de la fonction de transfert en chaîne ouverte dans le sens des pulsations croissantes, on laisse le point critique (0 dB , -180°) à sa droite. Il est instable dans le cas contraire.

- **Degré de stabilité [3]**

Il ne suffit pas qu'un système soit stable, il faut qu'il soit suffisamment stable. En effet l'évaluation de la fonction de transfert d'un système n'est pas toujours parfaite (petites constantes de temps ou légers temps morts négligés, hypothèses simplificatrices, incertitudes sur les paramètres ou les mesures lors des identifications). La courbe représentative de la

fonction de transfert doit donc passer assez loin de point critique, et l'évaluation de cet « éloignement » est effectuée à l'aide de deux critères :

$$\text{Marge de gain : } G_m = 20 \lg \left[\frac{1}{|A(j\omega_\pi)|} \right] = 20 \lg A_m \quad (1.23)$$

$$\text{Marge de phase : } \varphi_m = \pi + \arg[A(j\omega_1)] \quad (1.24)$$

Où ω_1 est la pulsation pour laquelle le module $|A(j\omega_1)| = 1$.

Un système est stable pour $G_m > 0$ et $\varphi_m > 0$.

Valeurs courantes des marges : $8dB < G_m < 15dB$ et $40^\circ < \varphi_m < 60^\circ$.

1.4.2. Rapidité

Elle traduit la durée du régime transitoire et s'évalue, au choix, par :

- Le temps de réponse T_r correspondant au temps que met la mesure à rester dans une zone à $\pm 5\%$ de la variation finale, soit entre 95% et 105% .
- Le temps de montée T_m correspondant au temps nécessaire à la mesure pour passer de 10% à 90% de la valeur finale.

1.4.3. Précision

Elle est évaluée par son aptitude à obtenir une mesure proche de la consigne en régime permanent ou en régime transitoire lors d'un changement de consigne ou d'une perturbation.

- **En régime permanent**

La précision d'une régulation de maintien ou de poursuite se chiffre par la différence entre la consigne et la mesure en régime permanent. Plus cet écart est petit, plus la régulation est précise.

- **En régime transitoire**

La précision dynamique d'une régulation est évaluée par l'écart maximal obtenu pour le premier dépassement lors d'un changement de consigne.

La précision dynamique d'une régulation de maintien est évaluée par l'écart maximal obtenu entre la consigne et la mesure lors du régime transitoire.

1.5. Régulation numérique

La commande numérique présente de nombreux avantages, en particulier la fiabilité, et la souplesse dans la programmation des algorithmes, qui permet d'obtenir facilement des lois de commande variées. Le correcteur est donc réalisé par un algorithme implémenté sur un microcalculateur, il reçoit des informations sur l'état du procédé uniquement aux instants d'échantillonnage, à travers un convertisseur analogique-numérique (CAN), le résultat des calculs est transmis à un convertisseur numérique-analogique (CNA), qui élabore un signal de commande.

La commande par ordinateur, ou processeur, d'un procédé nécessite la mise en œuvre d'un certain nombre d'éléments.

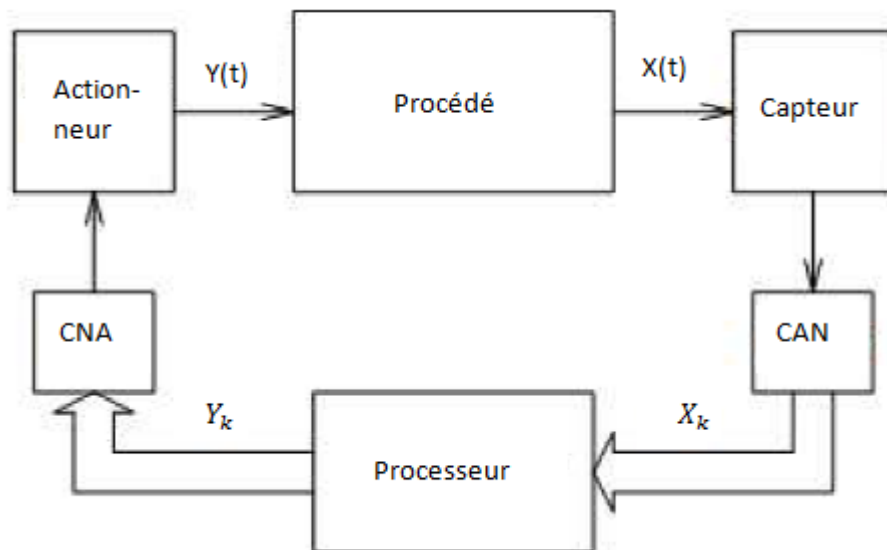


Figure 1.21 : Structure de commande d'un système continu par ordinateur.

- Un processeur (ordinateur) qui élabore la commande Y_k et réalise l'échantillonnage.
- Des convertisseurs analogique-numériques, CAN (échantillonnage).
- Des capteurs ou organes de mesure qui transmettent au calculateur les informations recueillies sur le système continu, à travers les convertisseurs analogiques/numériques.
- Des convertisseurs numérique-analogiques, CNA.

Un convertisseur N/A consistera à maintenir constante sur l'intervalle $[kT, (k+1)T]$ la commande soit : $Y(t) = Y(kT)$, $\forall t \in [kT, (k+1)T]$

Un tel convertisseur est appelé Bloqueur d'Ordre Zéro (BOZ).

Remarque : Le principe de la commande numérique reste toujours identique à celui de la régulation analogique.

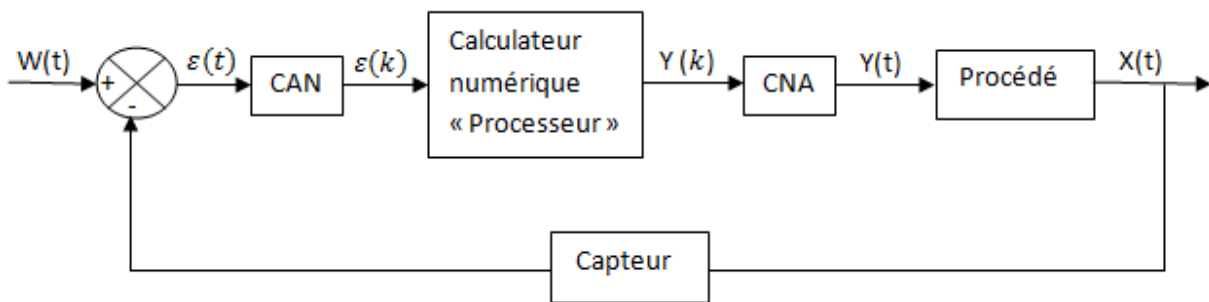


Figure 1.22 : Structure d'une régulation numérique

1.5.1. Intérêt de la régulation numérique [3]

La commande numérique d'un processus continu précédé par son BOZ et dont la sortie est échantillonnée est mis en œuvre comme suit :

- Choix de la période d'échantillonnage,
- Détermination du correcteur numérique à partir d'un cahier des charges,

Il élabore en temps réel la commande qu'il enverra à chaque instant ou pas d'échantillonnage au processus. L'évolution du système dépend alors à chaque instant des résultats élaborés par le calculateur ; on parle de commande par calculateurs en temps réel.

1.5.2. Choix du pas d'échantillonnage [3]

Le découpage temporel de l'information qu'est l'échantillonnage est certainement la caractéristique la plus importante de l'insertion des calculateurs dans la commande des processus. Pour une fréquence d'échantillonnage $f_e = \frac{1}{\Delta}$, il ne faut pas que Δ soit

trop petit sinon l'ordinateur consacre inutilement trop de temps au système qu'il pilote, en le corrigeant par petits coups souvent répétés en perdant de temps, ainsi qu'il doit pas être trop grand sinon l'ordinateur ne recevra pas certaines informations importantes en provenance de la sortie.

1.5.3. Théorème de Shannon [3]

Afin de bien restaurer après numérisation, il faut respecter la condition de Shannon suivante :

$$f_e = 2f_{max} \quad (1.25)$$

En pratique pour satisfaire la condition de Shannon avec une bonne marge de sécurité on prendra : $\frac{T_H}{25} < \Delta < \frac{T_H}{5}$ soit $5f_H < \frac{1}{\Delta} < 25f_H$ (1.26)

1.5.4 Application aux boucles d'asservissement

Lorsque, après bouclage on obtient une fonction de transfert du premier ou du deuxième ordre, il est aisé de définir une plage convenable pour la valeur du pas d'échantillonnage Δ .

En effet, prenons pour valeur de f_H la fréquence de coupure f_C de la boucle.

- **Système de premier ordre [3]**

$$G(p) = \frac{1}{1+\theta p}$$

$$f_H = f_C = \frac{1}{2\pi\theta}, \text{ donc } T_H = 2\pi\theta = 6.3\theta \quad (1.27)$$

$$\text{En appliquant (1.25) } \frac{6.3\theta}{25} < \Delta < \frac{6.3\theta}{5}$$

$$\text{En arrondissant les valeurs } 0.25 < \frac{\Delta}{\theta} < 1.25 \quad (1.28)$$

D'où on prend $\frac{\Delta}{\theta}$ voisin de 1

- **Système de deuxième ordre [3]**

$$G(p) = \frac{1}{1 + \frac{2h}{\omega_n}p + \frac{p^2}{\omega_n^2}}$$

$$f_H = f_C = \frac{\omega_n}{2\pi}, \text{ donc } T_H = \frac{2\pi}{\omega_n} \quad (1.29)$$

$$\text{En appliquant (1.25) } \frac{2\pi}{25\omega_n} < \Delta < \frac{2\pi}{5\omega_n}$$

$$\text{En arrondissant les valeurs } 0.25 < \Delta\omega_n < 1.25 \quad (1.30)$$

D'où on prend $\Delta\omega_n$ voisin de 1.

1.6 Conclusion

Dans ce chapitre nous avons vu que, identifier un système dynamique réel revient à caractériser son modèle. Parmi les nombreuses méthodes d'identifications existantes, nous avons présentées la méthode de Broïda pour un procédé autoréglant, et la méthode de Strejc pour un modèle d'ordre supérieur. Cette démarche permet de trouver un modèle de comportement traduisant le plus fidèlement le procédé autour d'un point de fonctionnement.

Après avoir présenté la définition de la régulation, son principe et son objectif qui consiste à maintenir à des niveaux prédéterminés. Nous avons entamé l'étude de régulateur PID dont nous avons expliqué les effets des actions P, I et D et les différentes structures de ce régulateur, à savoir, la structure parallèle, la structure série et la structure mixte et on termine par les méthodes de réglage de ce régulateur.

Chapitre 2

Logique Floue

Chapitre 02

Logique Floue

2.1 Introduction

La logique floue est une extension de la logique booléenne créée par Lotfi Zadeh en 1965 en se basant sur sa théorie mathématique des ensembles flous, qui est une généralisation de la théorie des ensembles classiques. En introduisant la notion de degré dans la vérification d'une condition, permettant ainsi à une condition d'être dans un autre état qu'est vrai ou faux. La logique floue confère une flexibilité très appréciable aux raisonnements qui l'utilisent, ce qui rend possible la prise en compte des imprécisions et des incertitudes.

Elle est de grande actualité. Il y a surtout des réalisations dans le domaine de réglage et de la commande de processus industriels, liés à l'énergie, les transports, la transformation de matière, l'électroménager, la robotique, etc. Cette nouvelle méthode de réglage a été introduite à large échelle d'abord au Japon. Cependant, plus récemment elle est de plus en plus appliquée [6].

Exemple

Quelques règles de conduite qu'un conducteur suit, en supposant qu'il tienne à son permis :

Si le feu est rouge...	si ma vitesse est élevée...	et si le feu est proche...	alors je freine fort.
Si le feu est rouge...	si ma vitesse est faible...	et si le feu est loin...	alors je maintiens ma vitesse.
Si le feu est orange...	si ma vitesse est moyenne...	et si le feu est loin...	alors je freine doucement.
Si le feu est vert...	si ma vitesse est faible...	et si le feu est proche...	alors j'accélère.

Intuitivement, il semble donc que les variables d'entrées à l'instar de cet exemple sont appréciées par le cerveau de manière approximative, correspondant ainsi au degré de vérification d'une condition de la logique floue.

2.2 Sous-ensembles flous

Un sous-ensemble flou F est défini sur un ensemble de valeur et le référentiel U . Il est caractérisé par une fonction d'appartenance :

$$\mu : x \in u \longrightarrow \mu(x) \in [0,1] \quad (1.1)$$

Qui qualifie le degré d'appartenance de chaque élément de U à F .

Exemple

Evaluation de la température de l'eau d'un récipient par les mots

Froide : F, Tiède : T, Chaude : C

- En logique classique

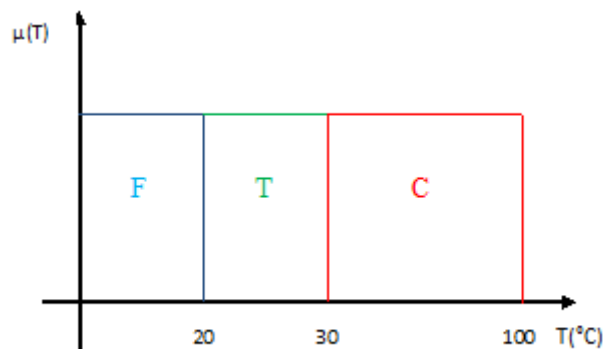


Figure 2.1 : Fonctions d'appartenances en logique classique

- En logique floue

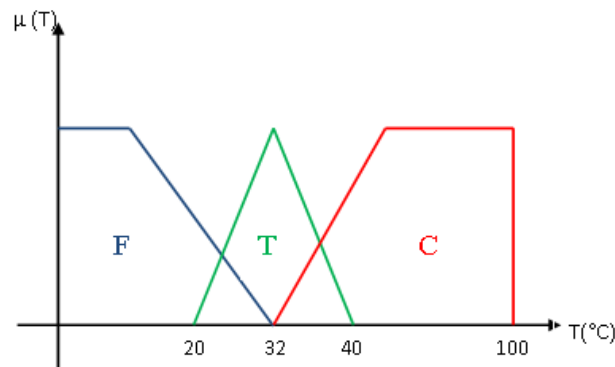


Figure 2.2 : Fonctions d'appartenances en logique floue

On voit sur la figure 2.1 que la logique classique ne peut utiliser que le 0 et le 1 ainsi l'eau est d'abord totalement froide puis tiède et enfin chaude. En dessous sur la figure 2.2 nous pouvons observer la représentation graphique de trois fonctions d'appartenance **Froide**, **Tiède** et **Chaude**. Ces fonctions nous permettent de superposer sur des plages de température données par les qualificatifs froide, tiède et chaude. On se rapproche donc du raisonnement humain.

2.3 Variables Linguistiques

La description d'une certaine situation, d'un phénomène ou d'un procédé contient en général des expressions floues comme : quelque, beaucoup, souvent, chaud, froid, rapide, lent, grand, petit, etc.

Les expressions de ce genre forment les variables linguistiques de la logique floue. Afin de permettre un traitement numérique, il est indispensable de les soumettre à une définition à l'aide de fonctions d'appartenance.

2.4 Opérateurs de la logique floue [6]

Les variables linguistiques sont liées entre elles au niveau des inférences par des opérateurs **ET** ou **OU**. Il s'agit d'opérateurs de la logique floue qui interviennent sur les fonctions d'appartenance représentant les variables linguistiques. De plus il existe un opérateur **NON** (complément, négation, inverse).

2.4.1 Opérateur NON

Selon la théorie des ensembles, l'ensemble complémentaire.

$$c = \bar{a} = \text{NON}(a) \quad (2.2)$$

Est défini par les éléments de x qui n'appartiennent pas à l'ensemble a .

Dans le cas de la logique floue, cette définition peut être exprimée par les fonctions d'appartenance de la manière suivante :

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x), \forall x \in u \quad (2.3)$$

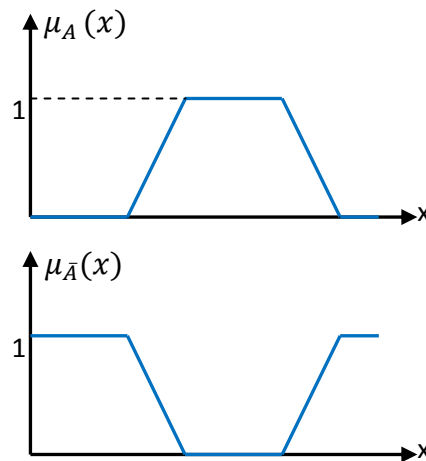


Figure 2.3: Opérateur NON

2.4.2 Opérateur ET

L'opérateur ET correspond à l'intersection de deux ensembles a et b et on écrit :

$$c = a \cap b = a \text{ ET } B \quad (2.4)$$

Dans le cas de la logique floue, l'opérateur ET est réalisé dans la plupart des cas par la formation du minimum, appliquée aux fonctions d'appartenance $\mu_A(x)$ et $\mu_B(x)$ des deux ensembles A et B à savoir :

$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)), \quad \forall x \in u \quad (2.5)$$

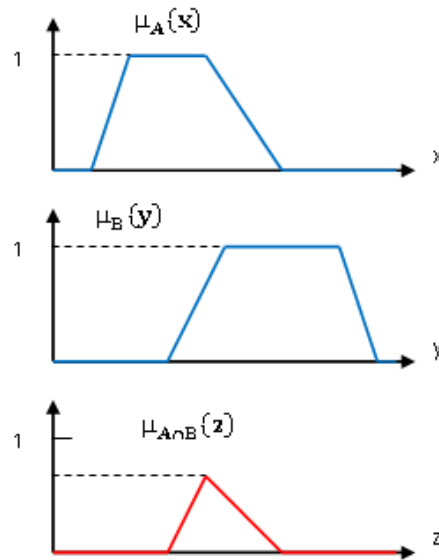


Figure 2.4: Opérateur ET

2.4.3 Opérateur OU

L'opérateur OU correspond à l'union de deux ensembles a et b, on a donc :

$$c = a \cup b = a \text{ OU } b \quad (2.6)$$

La réalisation de l'opérateur OU au niveau de la logique floue se fait en général par la formation du maximum, appliquée aux fonctions d'appartenance $\mu_A(x)$ et $\mu_B(x)$ des deux ensembles A et B. On a donc l'opérateur maximum.

$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)), \quad \forall x \in u \quad (2.7)$$

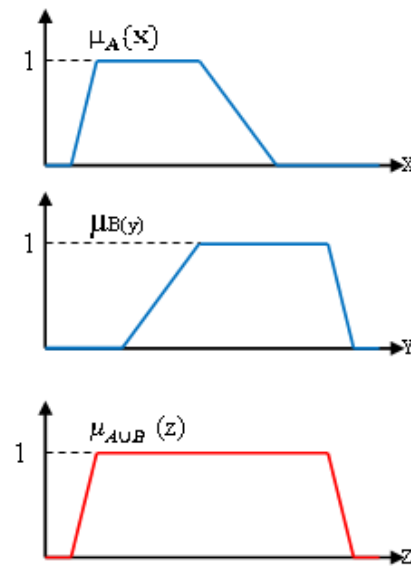


Figure 2.5 : Opérateur OU

2.5 Réglage et commande par logique floue

2.5.1 Introduction

Comme déjà mentionné, un domaine d'application de la logique floue qui devient de plus en plus important, est celui du réglage et de la commande de processus industriels [6]. Les algorithmes de réglage conventionnels sont alors remplacés par une série de règles linguistiques de la forme **SI... ALORS...**

En effet, cette méthode permet d'obtenir une loi de réglage souvent très efficace sans devoir faire des études théoriques approfondies. Par des inférences avec plusieurs règles, il est possible de tenir compte des expériences acquises par les opérateurs pour la conduite de processus.

2.5.2 Structure d'une commande floue [6]

La structure conventionnelle d'une commande floue est présentée à la figure 2.6. Elle est composée de quatre blocs distincts dont les définitions sont données ci-dessous.

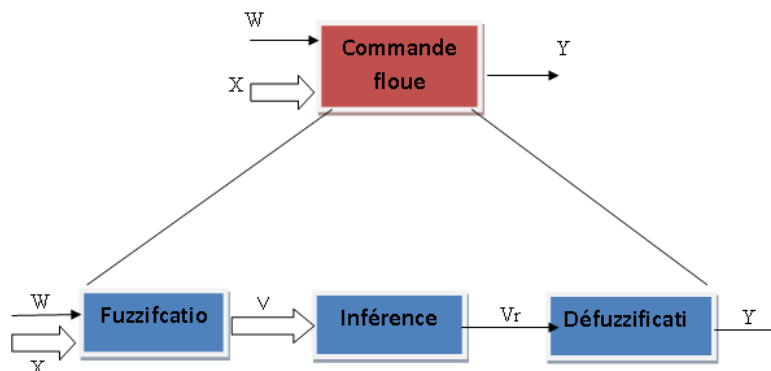


Figure 2.6 : Structure d'une commande floue

On peut distinguer trois parties :

- Fuzzification.
- Inférence.
- Défuzzification.

La commande par logique floue fournit le signal de commande Y . Elle reçoit à son entrée la grandeur de consigne W et une ou plusieurs grandeurs mesurées, réunies dans le vecteur X . La fuzzification fournit une série de variables floues réunies par le vecteur V et à la sortie de bloc d'inférence on obtient une information floue pour la variable V_r .

On procède tout d'abord à la partition en sous-ensembles flous des différents univers de discours que le système impose. Ensuite on détermine la base de règles qui va caractériser le fonctionnement désiré du système.

Il faut transformer les variables réelles, c'est à dire celles qui ont une réalité physique, en variables floues, on appelle cette étape la fuzzification. On utilise ensuite ces variables floues dans un mécanisme d'inférence qui crée et détermine les variables floues de sortie en utilisant les opérations sur les fonctions d'appartenance.

Enfin, on opère à la défuzzification qui consiste à extraire une valeur réelle de sortie à partir de la fonction d'appartenance du sous-ensemble flou de sortie établi par le mécanisme d'inférence.

2.5.3 Fuzzification

2.5.3.1 Généralités [6]

Comme on l'a montré à la figure 2.6 la commande par logique floue se compose de trois parties, dont la première est la fuzzification. Il s'agit de la conversion analogique/digitale, ainsi que de traitement des grandeurs mesurées X et de leur transformation en variables linguistiques avec la définition des fonctions d'appartenance.

La fuzzification proprement dite consiste à définir les fonctions d'appartenance pour les différentes variables. On réalise ainsi le passage des grandeurs physique (grandeurs déterminées) en variables linguistiques (variables floues) qui peuvent alors être traitées par les inférences.

2.5.3.2 Différentes formes possibles pour les fonctions d'appartenance

Fonction d'appartenance	Forme algébrique	Forme graphique
Fonction gaussienne	Elle est définie par deux paramètres $\{m, \sigma\}$ $\mu(x) = \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right)$	
Fonction sigmoïdale	Elle est définie par deux points $\{a, c\}$ $\mu(x) = \frac{1}{1 + \exp(-a(x-c))}$	
Singleton	Un fait certain à une fonction d'appartenance égale à 1 (le sous ensemble est alors représenté par un singleton) $\mu_{x_0}(x) = 1 \text{ si } x = x_0$ $\mu_{x_0}(x) = 0 \text{ sinon}$	
Fonction triangulaire	Elle est définie par trois paramètres $\{a, b, c\}$ qui déterminent les coordonnées des trois sommets $\mu(x) = \text{Max}\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right)$	
Fonction trapézoïdale	Elle est définie par quatre paramètres $\{a, b, c, d\}$ $\mu(x) = \text{Max}\left(\min\left(\frac{x-a}{b-a}, 1, \frac{c-x}{c-d}\right), 0\right)$	

Dans le cas du réglage par logique floue, on utilise en général des formes trapézoïdales et triangulaires pour les fonctions d'appartenance. Bien qu'il n'existe pas de règles précises pour la définition des fonctions d'appartenance.

2.5.4 Inférence [6]

2.5.4.1 Généralités

La stratégie de réglage dépend essentiellement des inférences adoptées. Elles lient les grandeurs mesurées, qui sont les variables d'entrées V (transformées en variables linguistiques à l'aide de la fuzzification), à la variable de sortie V_r (voir la figure 2.6).

Les méthodes d'inférence permettent la réalisation de différents opérateurs ET, OU, ALORS, intervenant dans les règles d'inférence et s'appliquant aux fonctions d'appartenance.

Trois méthodes sont généralement utilisées :

- Méthode d'inférence Max-Min.
- Méthode d'inférence Max-Prod.
- Méthode d'inférence Somme-Prod.

2.5.4.2 Méthode d'inférence Max-Min

La méthode d'inférence max-min réalise, au niveau de la condition, l'opérateur OU par la formation de maximum et l'opérateur ET par la formation du minimum. La conclusion dans chaque règle, introduite par ALORS, lie le facteur d'appartenance de la condition avec la fonction d'appartenance de la variable de sortie X_R par l'opérateur ET, réalise dans le cas présent par la formation du minimum. Enfin l'opérateur OU qui lie les différentes règles est réalisé par la formation du maximum.

La figure 2.7 représente graphiquement le principe de la méthode d'inférence Max-Min.

La condition (X_1 PG ET X_2 EZ) de la première règle implique pour $X_1=0.44$ et $X_2= -0.67$ les facteurs d'appartenance $\mu_{PG}(X_1=0.44) = 0.67$ et $\mu_{EZ}(X_2= -0.67) = 0.33$, ce qui implique que la condition prend le facteur d'appartenance $\mu_{c1} = 0.33$ (minimum des deux valeurs à cause de l'opérateur ET). La fonction d'appartenance $\mu_{EZ}(X_R)$ pour la variable de sortie est donc écrêtée à 0.33 (à cause de la formation du minimum lié à ALORS).

La deuxième règle implique par sa condition (X_1 EZ ET X_2 NG) les facteurs d'appartenance $\mu_{EZ}(X_1=0.44) = 0.33$ et $\mu_{NG}(X_2= -0.67) = 0.67$. Ainsi la condition possède le facteur d'appartenance $\mu_{c2} = 0.67$ (maximum des deux valeurs à cause de l'opérateur OU).

La fonction d'appartenance $\mu_{NG}(X_R)$ est donc écrêtée à 0.67 (à cause de la formation du minimum lié à ALORS).

Puisque chacune des règles exige une intervention, il faut encore déterminer la fonction d'appartenance résultante $\mu_{RES}(X_R)$. Elle s'obtient par la formation du maximum des deux fonctions d'appartenance, étant donné que les deux règles sont liées par l'opérateur OU. Cette fonction d'appartenance est hachurée à la figure 2.7.

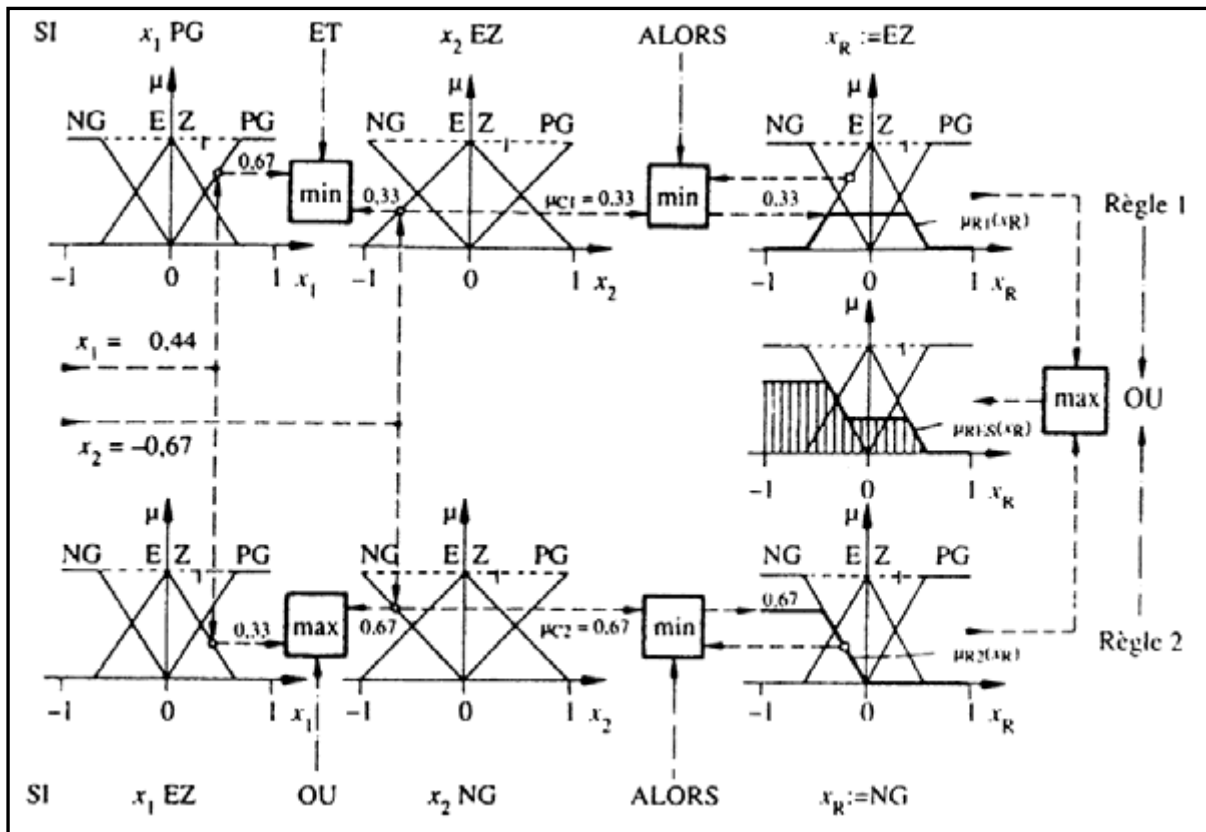


Figure 2.7 : Méthode d'inférence Max-Min

2.5.4.3 Méthode d'inférence Max-Prod

La méthode d'inférence Max-Prod réalise en général, au niveau de la condition, l'opérateur OU par la formation de maximum et l'opérateur ET par la formation du minimum. Par contre la conclusion dans chaque règle, introduite par ALORS, qui lie le facteur d'appartenance de la condition avec la fonction d'appartenance de la variable de sortie X_R par l'opérateur ET, est réalisé cette fois-ci par la formation de produit. L'opérateur OU qui lie les différentes règles est réalisé par la formation du maximum.

Comme on le voit, le OU, liant les règles est réalisé par la formation du maximum et le ALORS est réalisé par la formation du produit, d'où la désignation de cette méthode d'inférence par Max-Prod.

La représentation graphique du principe de la méthode d'inférence Max-Prod se trouve à la figure 2.8.

Comme dans le cas du paragraphe précédent, la première condition prend le facteur d'appartenance $\mu_{c1} = 0.33$. La fonction d'appartenance $\mu_{EZ}(X_R)$ pour la variable de sortie est cette fois –ci multipliée par ce facteur (à cause de la formation du produit lié à ALORS). La deuxième condition possède le facteur d'appartenance $\mu_{c2} = 0.67$, comme dans le paragraphe précédent. La fonction d'appartenance $\mu_{NG}(X_R)$ est multipliée par ce facteur (formation du produit), et l'on obtient la fonction d'appartenance partielle $\mu_{R2}(X_R)$.

La fonction d'appartenance résultante $\mu_{RES}(X_R)$ s'obtient également par la formation du maximum des deux fonctions d'appartenance partielle, réalisant ainsi l'opérateur OU.

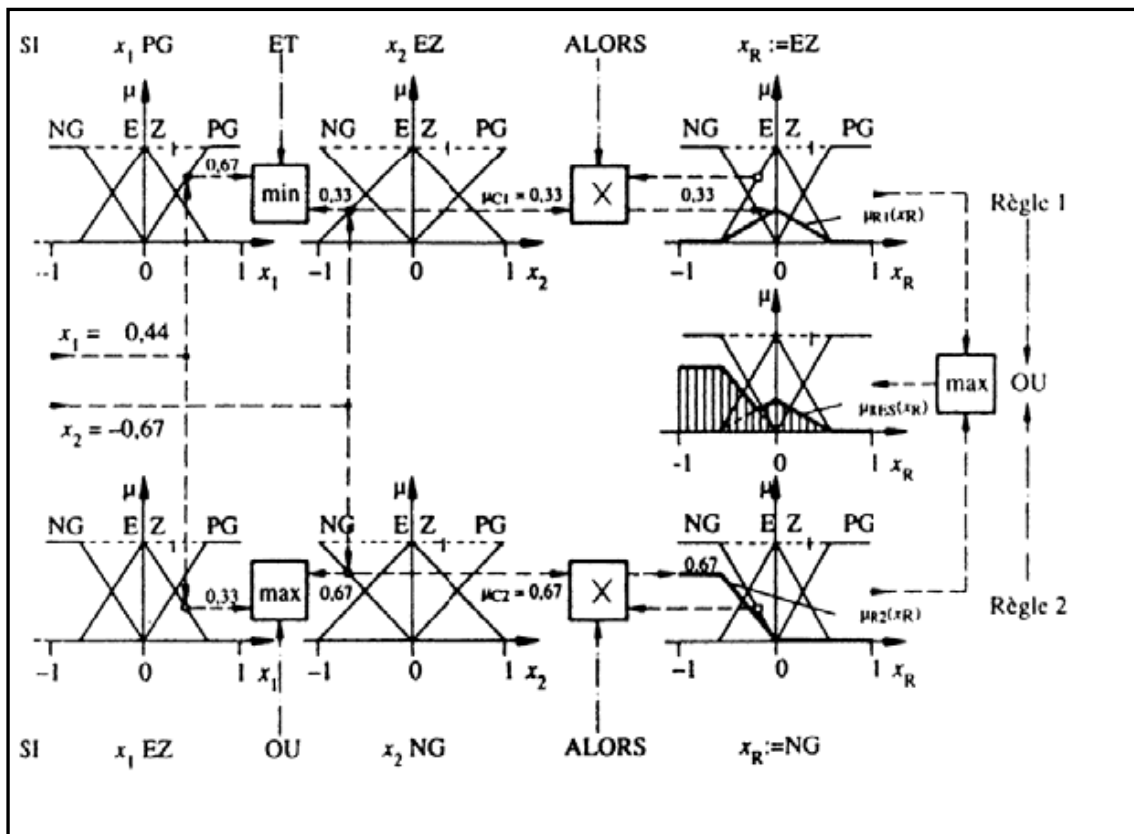


Figure 2.8 : Méthode d'inférence Max-Prod

2.5.4.4 Méthode d'inférence Somme-Prod

Par opposition aux méthodes d'inférences précédentes, la méthode d'inférence Somme-Prod réalise, au niveau de la condition, l'opérateur OU par la formation de la somme, plus précisément par la valeur moyenne, tandis que l'opérateur ET est réalisé par la formation du produit. La conclusion de chaque règle, précédée par ALORS, liant le facteur d'appartenance de la condition avec la fonction d'appartenance de la variable de sortie par l'opérateur ET, est réalisé par la formation de produit. L'opérateur OU qui lie les différentes règles est réalisé par la formation de la somme, donc de la valeur moyenne.

Dans ce cas, le OU liant les règles est réalisé par la formation de la somme et le ALORS est réalisé par la formation du produit, ainsi s'explique la désignation par Somme-Prod de cette méthode d'inférence.

La méthode d'inférence Somme-Prod est représentée graphiquement à la figure 2.9.

Avec les facteurs d'appartenance $\mu_{PG}(X_1=0.44) = 0.67$ et $\mu_{EZ}(X_2 = -0.67) = 0.33$, la première condition prend le facteur d'appartenance $\mu_{C1} = 0.22$ (produit des deux valeurs) la fonction d'appartenance $\mu_{EZ}(X_R)$ pour la variable de sortie est multipliée par ce facteur (puisque ALORS est réalisé par la formation du produit). On obtient ainsi la fonction d'appartenance $\mu_{R1}(X_R)$.

La condition de la deuxième règle possède le facteur d'appartenance $\mu_{C2} = 0.5$, à cause de la formation de la somme des deux facteurs d'appartenance $\mu_{EZ}(X_1=0.44) = 0.33$ et

$\mu_{NG}(X_2 = -0.67) = 0.67$ pour l'opérateur OU. Ainsi la fonction d'appartenance $\mu_{NG}(X_R)$ est multipliée par le facteur $\mu_{C2} = 0.5$. Il en résulte ainsi la fonction d'appartenance partielle $\mu_{R2}(X_R)$. La fonction d'appartenance résultante $\mu_{RES}(X_R)$, hachurée à la figure 2.9, s'obtient par la formation de la somme (valeur moyenne) des deux fonctions d'appartenance partielles.

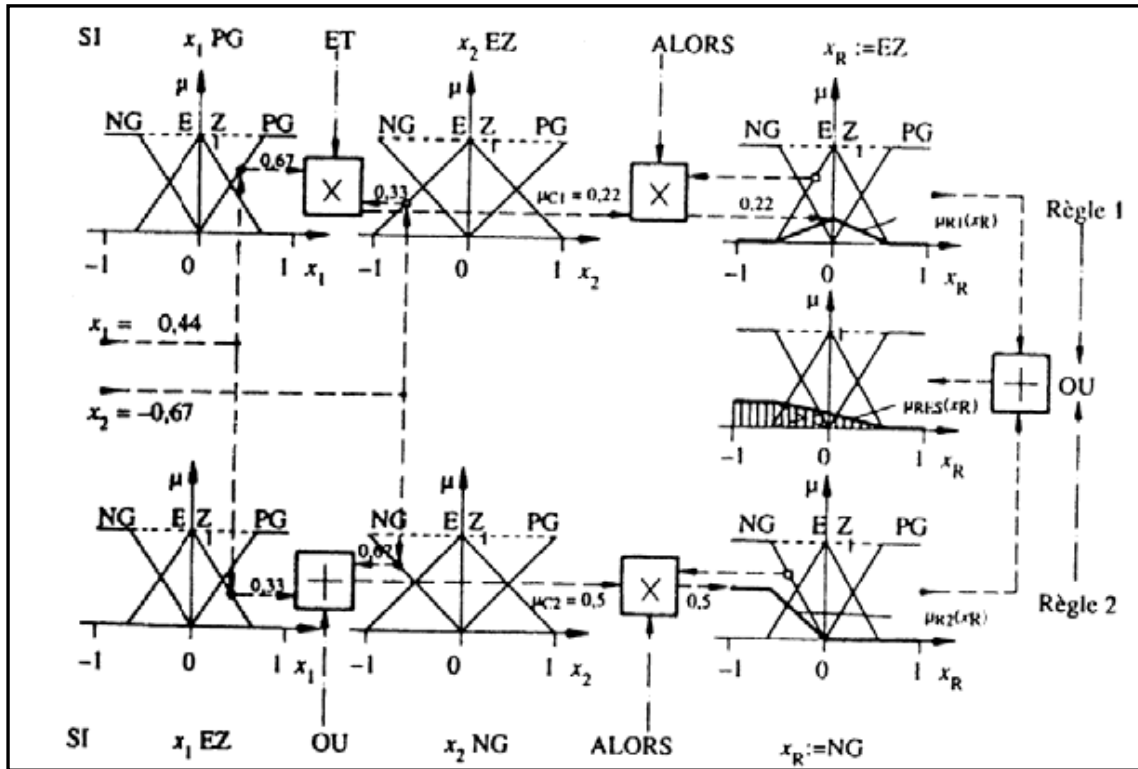


Figure 2.9 : Méthode d'inférence Somme-Prod

2.5.5 Défuzzification

La défuzzification consiste à transformer le sous ensemble flou de sortie en grandeur physique permettant la commande du système.

Nous allons présenter brièvement les deux principales méthodes de défuzzification : la méthode du valeur maximale et la méthode du centre de gravité (COG).

2.5.5.1 Défuzzification par centre de gravité

La méthode de défuzzification la plus utilisée est celle de la détermination du centre de gravité de la fonction d'appartenance résultante $\mu(x)$. Dans ce contexte, il suffit de calculer l'abscisse X_0 . La figure 2.10 montre le principe de défuzzification [6].

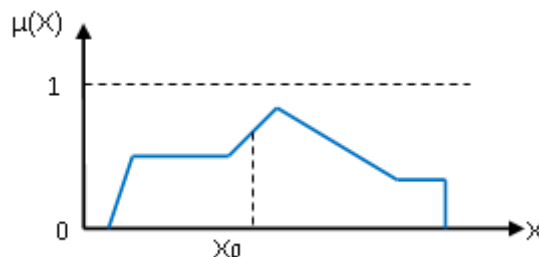


Figure 2.10 : Défuzzification par centre de gravité

$$x_0 = \frac{\int_u x \cdot \mu(x) dx}{\int_u \mu(x) dx} \quad (2.8)$$

U : domaine d'intégration

2.5.5.2 Défuzzification par valeur maximale

La défuzzification par centre de gravité exige en général une envergure de calcul assez importante. Par conséquent, il sera utile de disposer d'une méthode de défuzzification plus simple.

Cette méthode consiste à choisir comme valeur de sortie celle correspondant à l'abscisse du maximum de la fonction d'appartenance comme le montre la figure 2.11.

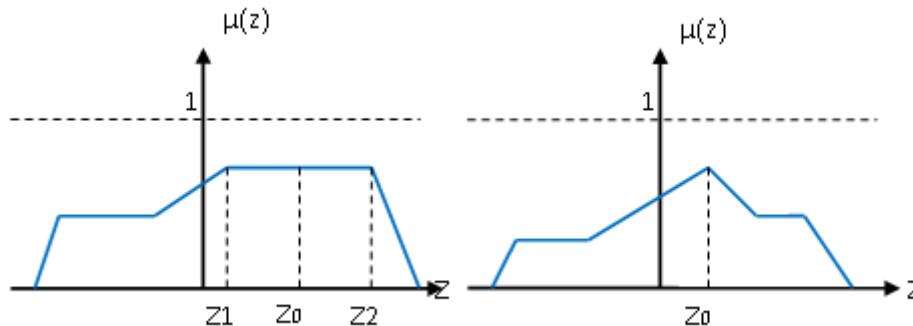


Figure 2.11 : Défuzzification par valeur maximale.

$$Z_0 = \frac{\int_s y dy}{\int_s dy} \quad (2.9)$$

Lorsque la fonction d'appartenance est écrêtée, toute valeur entre Z_1 et Z_2 peut être utilisée comme l'indique la figure 2.11. Afin d'éviter cette indétermination, on prend la moyenne des abscisses du maximum.

2.6 Différents types de régulateurs flous

2.6.1 Régulateur de type Mamdani

Dans les modèles linguistiques, appelés aussi modèles flous de type Mamdani, les Antécédents et les conséquences des règles sont des propositions floues. La forme générale des règles de Mamdani est :

$$\text{Si } x_1 = A_1 \text{ ET } x_2 = A_2 \text{ ALORS } y=B_1 \quad (2.10)$$

Généralement les régulateurs flous de type Mamdani, sont des régulateurs à deux entrées, l'erreur et sa variation. Et une sortie, qui représente la variation de la commande.

La structure d'un contrôleur flou est donnée par la figure 2.12.

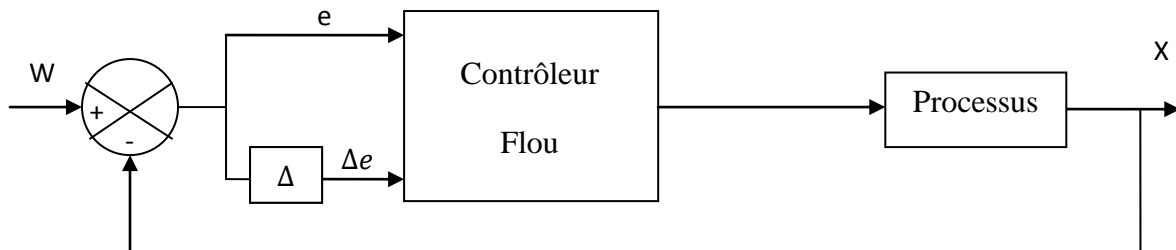


Figure 2.12 : Structure d'un régulateur flou

2.6.2 Régulateur de type Sugeno

Le modèle de Sugeno est connu sous le nom de TSK, car il a été proposé par Takagi, Sugeno et Kang en 1988. Dans ce type de régulateurs les sorties sont des fonctions linéaires.

Le système flou de type Takagi-Sugeno, utilise des règles écrites de la manière suivante :

$$\text{Si } x_1 = A_1 \text{ ET } x_2 = A_2 \text{ ALORS } y = z_1 x_1 + z_2 x_2 + z_0 \quad (2.11)$$

2.7 Exemple d'application

- **Description du problème**

On souhaite commander l'installation de chauffage d'un immeuble à l'aide d'un contrôleur flou. On dispose de deux sondes de température : l'une à l'extérieur de l'immeuble ; l'autre à l'intérieur (grandeur interne). Sur la base de ces deux mesures et en faisant appel aux règles d'inférences, le contrôleur flou doit régler la puissance de l'installation de chauffage.

- **Fuzzification de la température externe**

On choisit deux intervalles flous et des fonctions d'appartenance de type trapézoïdales en définissant le « froid » comme correspondant à une température inférieure à 5 °C , et le « chaud » comme étant une température supérieure à 20 °C (figure 2.13).

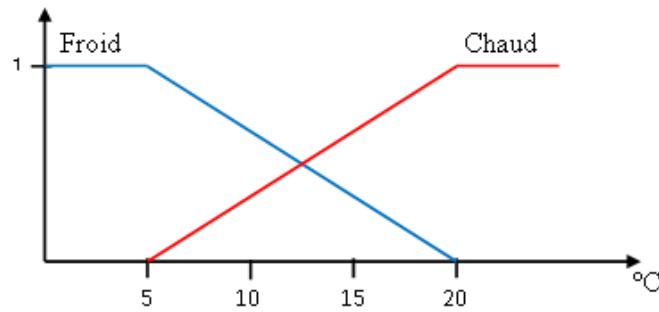


Figure 2.13 : Fuzzification de la température externe

- **Fuzzification de la température interne**

On choisit trois intervalles flous et des fonctions d'appartenance de type trapézoïdales en définissant le « froid » comme correspondant à une température inférieure à 15 °C, le « bon » comme étant une température comprise entre 18 °C et 22 °C et le « chaud » comme étant une température supérieure à 25 °C (figure 2.14).

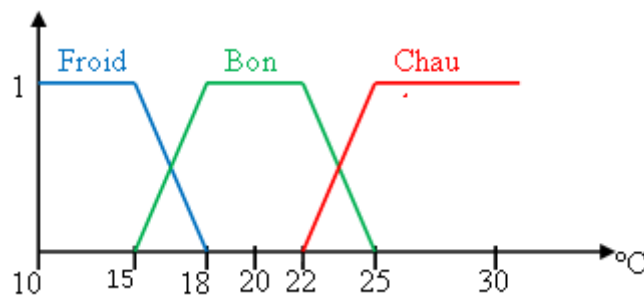


Figure 2.14 : Fuzzification de la température interne

- **Fuzzification de la puissance**

On choisit quatre intervalles flous pour définir la puissance de l'installation avec des fonctions d'appartenance en forme de raies. On définit les valeurs suivantes :

Valeur de Puissance en %

Nulle 0

Faible 33

Moyenne 67

Maximale 100

Ce qui définit les fonctions d'appartenance illustrées à la figure (2.15).

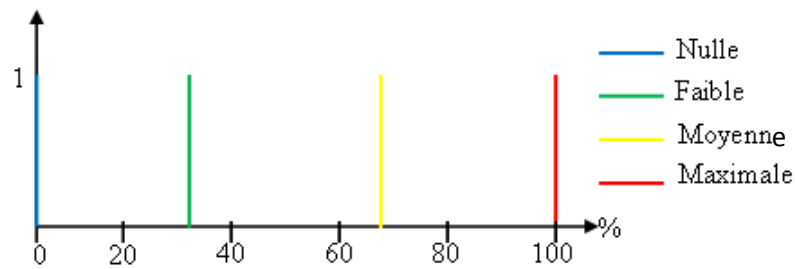


Figure 2.15 : Fuzzification de la puissance

• Règles d'inférences

L'expérience acquise sur l'installation de chauffage a permis de définir les six règles suivantes :

1. **Si** la température extérieure est « froide » **et** la température intérieure est « froide » **alors** mettre la puissance au « maximum »
2. **Si** la température extérieure est « froide » **et** la température intérieure est « bonne » **alors** mettre une puissance « moyenne »
3. **Si** la température extérieure est « froide » **et** la température intérieure est « chaude » **alors** mettre une puissance « faible »
4. **Si** la température extérieure est « chaude » **et** la température intérieure est « froide » **alors** mettre une puissance « moyenne »
5. **Si** la température extérieure est « chaude » **et** la température intérieure est « bonne » **alors** mettre une puissance « faible »
6. **Si** la température extérieure est « chaude » **et** la température intérieure est « chaude » **alors** mettre une puissance « nulle »

2.8. Conclusion

Les outils fournis par la logique floue permettent une modélisation des phénomènes pouvant en un certain sens s'approcher du raisonnement humain. Le fait de transcender le « Tout ou Rien » introduit une souplesse faisant la puissance des outils flous dans de nombreux domaines.

Au milieu des années 80, les applications industrielles firent florès de manière spectaculaire, et ce essentiellement en Asie de Sud-est, l'Europe et l'Amérique, restant assez circonspectes sur le sujet. Elles vont du contrôle d'un métro automatique à l'élimination du tremblement pour les caméras vidéo en passant par le réglage de cycles sur une machine à laver. En outre

la flexibilité des modèles flous permet également des applications dans des domaines tels que la médecine (aide au diagnostic), la finance (prévision boursières, opérations de change), la météorologie, etc.

Chapitre 3

Présentation du logiciel LabVIEW

Chapitre 03

Présentation du logiciel LabVIEW

3.1. Introduction [8]

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) est un langage de programmation graphique qui utilise des icônes au lieu de lignes de texte pour créer des applications. Contrairement aux langages de programmation textuels où ce sont les instructions qui déterminent l'ordre d'exécution du programme, LabVIEW utilise la programmation par flux de données ; c'est le flux des données transitant par les nœuds sur le diagramme qui détermine l'ordre d'exécution des VIs et des fonctions. Les VIs, ou instruments virtuels, sont des programmes LabVIEW qui imitent les instruments physiques.

Dans LabVIEW, on construit une interface utilisateur à l'aide d'un ensemble d'outils et d'objets. L'interface utilisateur d'un VI est appelée la face-avant. Ensuite, vous créez le code en utilisant des représentations graphiques de fonctions pour commander les objets de la face-avant. Ce code source graphique est aussi appelé code G ou code du diagramme.

D'une certaine manière, le diagramme ressemble à un organigramme.

3.2. Présentation des instruments virtuels

Les programmes LabVIEW portent la dénomination d'instruments virtuels ou VIs, car leur apparence et leur fonctionnement imitent ceux d'instruments réels tels que les oscilloscopes et les multimètres. Chaque VI utilise des fonctions qui manipulent les entrées de l'interface utilisateur ou d'autres sources et qui affichent ces informations ou les déplacent vers d'autres fichiers ou ordinateurs [8].

Un VI contient les trois composantes suivantes [8] :

- **Face-avant** : Sert d'interface utilisateur.
- **Diagramme** : Contient le code source graphique qui définit les fonctionnalités du VI.
- **Icône et connecteur** : Identifie l'interface au VI pour que vous puissiez utiliser celui-ci dans un autre VI. Un VI à l'intérieur d'un autre VI est appelé sous-VI. Un sous-VI correspond à un sous-programme dans des langages de programmation textuels.

3.2.1 Face – avant [8]

La face-avant est l'interface utilisateur du VI. La figure suivante montre un exemple de face-avant.

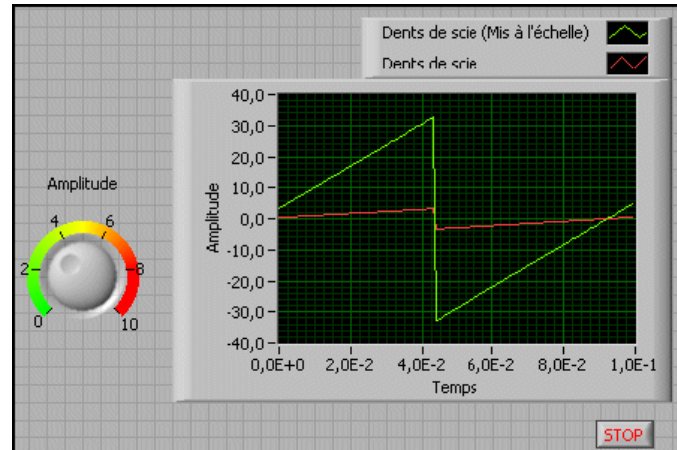


Figure 3.1: Exemple de face avant

Vous construisez la face-avant avec des commandes et des indicateurs qui sont respectivement les terminaux d'entrée et de sortie interactifs du VI. Les commandes sont des boutons rotatifs, des boutons-poussoirs, des cadrans et autres mécanismes d'entrée. Les indicateurs sont des graphes, des LED et d'autres afficheurs de sortie. Les commandes simulent les mécanismes d'entrée des instruments et fournissent des données au diagramme du VI. Les indicateurs simulent les mécanismes de sortie d'instruments et affichent les données que le diagramme acquiert ou génère.

3.2.2. Diagramme [8]

Après avoir construit la face-avant, vous devez ajouter le code en utilisant les représentations graphiques des fonctions pour commander les objets de la face-avant. Le diagramme contient ce code source graphique, aussi appelé code G ou code du diagramme. Les objets de la face-avant apparaissent en tant que terminaux sur le diagramme. La figure suivante montre un exemple de diagramme.

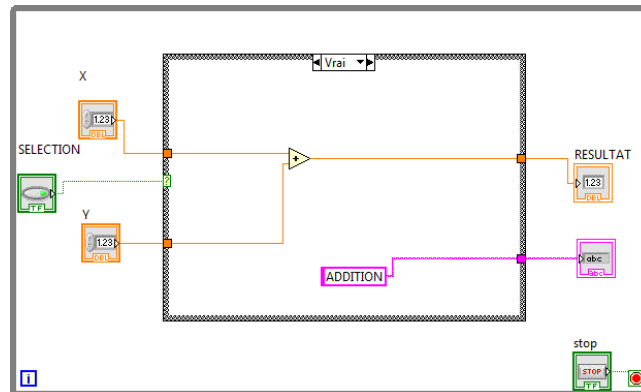


Figure 3.2 : exemple de diagramme

3.2.3. Icône et connecteur

Une fois que vous avez construit la face-avant et le diagramme d'un VI, créez son icône et son connecteur pour pouvoir l'utiliser en tant que sous-VI. L'icône et le connecteur correspondent au prototype de fonction des langages de programmation textuels.

- **Icône**

Chaque VI affiche une icône, comme celle qui est représentée ci-dessous, dans le coin supérieur droit des fenêtres de la face-avant et du diagramme.

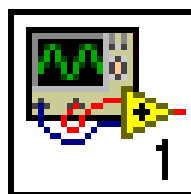


Figure 3.3 : Icône

Une icône est la représentation graphique d'un VI. Elle peut contenir du texte, des images ou les deux. Si vous utilisez un VI comme sous-VI, l'icône identifie le sous-VI sur le diagramme du VI. Vous pouvez double-cliquer sur l'icône pour la personnaliser ou la modifier.

- **Connecteur**

Vous devez également construire un connecteur, représenté ci-dessous, pour utiliser le VI comme sous-VI.

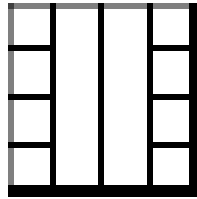


Figure 3.4 : Connecteur

Le connecteur est un groupe de terminaux qui correspond aux commandes et aux indicateurs de ce VI ; ce groupe est semblable à la liste de paramètres d'un appel de fonction dans les langages de programmation textuels. Le connecteur définit les entrées et sorties que vous pouvez connecter au VI que vous voulez utiliser comme sous-VI. Un connecteur reçoit des données sur ses terminaux d'entrée et transmet ces données au diagramme par les commandes de sa face-avant. Il reçoit les résultats sur ses terminaux de sortie par les indicateurs de sa face-avant.

3.3. Différentes palettes de LabVIEW

Pour construire les faces-avant et les diagrammes de VIs, LabVIEW comprend trois palettes :

La palette **Commandes**, la palette **Fonctions** et la palette **Outils**.

3.3.1. Palette d'outils

La palette **Outils** est disponible sur la face-avant et le diagramme. Un outil est un mode de fonctionnement spécial du curseur de la souris. Le curseur prend l'apparence de l'icône de l'outil sélectionné sur la palette. Utilisez les outils pour faire fonctionner et modifier la face-avant et les objets du diagramme.

Si la sélection automatique de l'outil est activée et que vous déplacez le curseur sur les objets de la face-avant ou du diagramme, LabVIEW sélectionne automatiquement l'outil correspondant dans la palette **Outils**.



Figure 3.5 : Palette d'outils

3.3.2. Palette des commandes

La palette **Commandes** est disponible uniquement sur la face-avant. Elle contient les commandes et les indicateurs nécessaires pour créer la face-avant. Les commandes et les indicateurs sont situés dans des sous-palettes en fonction du type de commande et d'indicateur.

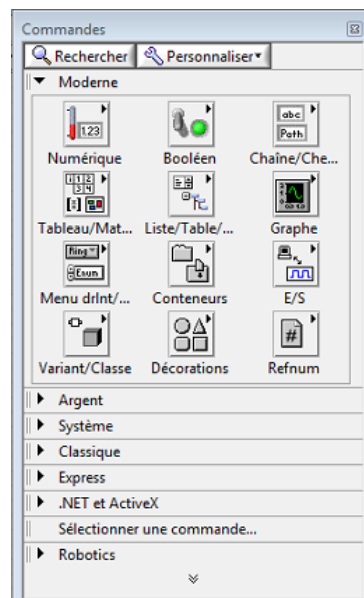


Figure 3.6 : Palette des commandes

3.3.3. Palette des fonctions

La palette **Fonctions** est disponible uniquement sur le diagramme. Elle contient les VIs et les fonctions nécessaires pour construire le diagramme. Les VIs et les fonctions sont situés dans des sous-palettes en fonction du type du VI ou de la fonction.

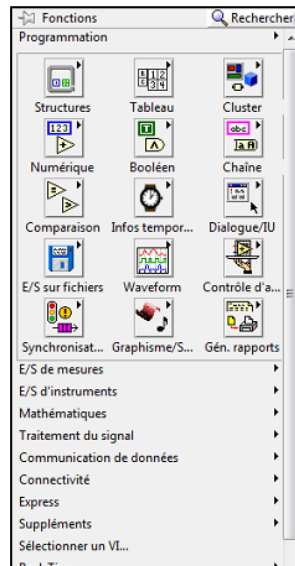


Figure 3.7 : Palette des fonctions

3.4. Structure des données dans LabVIEW

LabVIEW utilise un langage fortement typé et toutes données ou structure de données ne peuvent être manipulées qu'avec des fonctions admettant ce type, enfaîte dans LabVIEW on trouve les types de base scalaire, les types entiers (signés ou non, codés sur 8, 16 ou 32 bits), le type réel (codé sur 16, 32 ou 64 bits), le type booléen et le type chaîne de caractères (figure 3.6). Il est important de noter que les éléments représentant ces données, ainsi que les liaisons issues de ces éléments, sont de forme et de couleur différente.

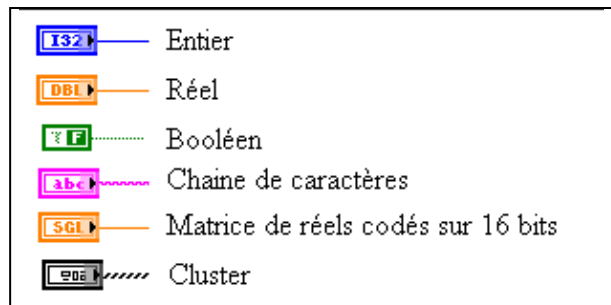


Figure 3.8 : Différents types de structures des données dans LabVIEW

3.4.1. Tableaux sous LabVIEW

Les tableaux sont des ensembles ordonnés d'éléments de même type : des booléens, chaînes de caractères, réels, waveforms ou clusters. On ne peut pas avoir un tableau de tableaux. Pour cela, on peut utiliser un tableau multidimensionnel ou un tableau de clusters dans lequel chaque cluster peut contenir un ou plusieurs tableaux.

Un tableau peut être de plusieurs dimensions. Chaque dimension peut être indexée de 0 à N-1 avec N le nombre d'éléments. Une colonne est un tableau à 1 dimension. Le tableau est dit à 2 dimensions lorsqu'il possède un certain nombre de colonnes et un certain nombre de lignes.

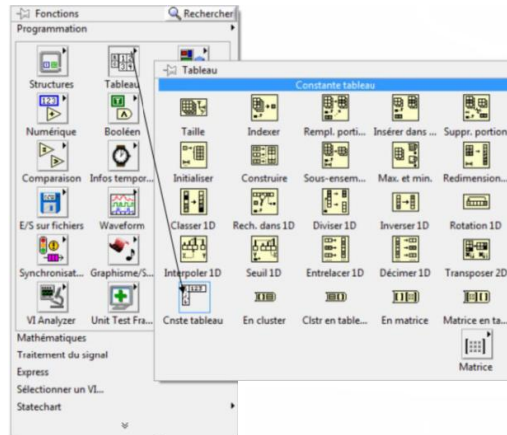


Figure 3.9 : Création de tableaux dans le diagramme

3.4.2 Waveform

Le type Waveform est un type correspondant à un cluster possédant un champ date (absolue ou relative) t_0 , un champ flottant représentant l'intervalle de temps dt séparant chaque valeur, et un tableau de valeurs réelles Y .

Le Waveform est une structure qui facilite l'utilisation des données issues des captures et générations des signaux.

La sous palette Waveform donne accès aux fonctions suivantes :

- Composantes d'une Waveform ($x, \Delta x, [y]$)
- Construire une Waveform
- E/S sur fichiers de Waveform
- Mesure sur Waveform
- Génération de Waveform

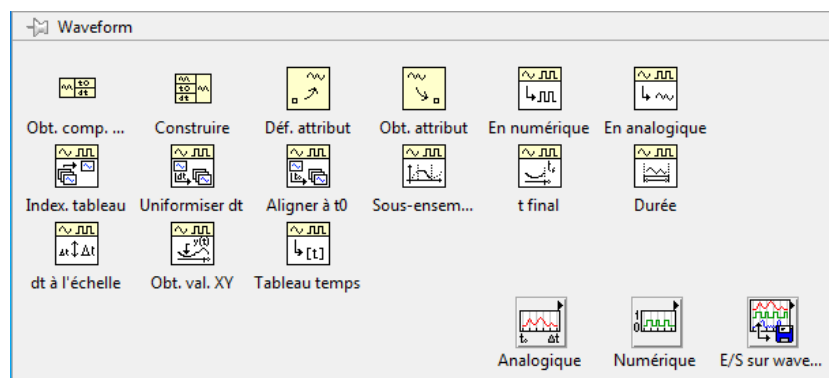


Figure 3.10 : Sous palette Waveform

3.4.3. Clusters

Les clusters ou agrégats sont des objets pouvant contenir des éléments de différents types, ordonnés de 0 à n-1.

Lorsqu'on supprime un élément du cluster, l'ordre se réajuste automatiquement.

Ils permettent de rassembler un ensemble de données et éviter l'emploi de nombreux fils et ainsi faciliter la lisibilité du VI. Un cluster est comme une gaine dans laquelle on fait passer les différents fils d'un câble téléphonique ou des fils électriques. On passe alors la gaine au lieu des fils individuels. Un cluster est équivalent au type Struct du langage C. Les clusters sont recommandés lorsqu'on doit transmettre beaucoup de fils d'une partie d'un code à une autre ou à des VIs.

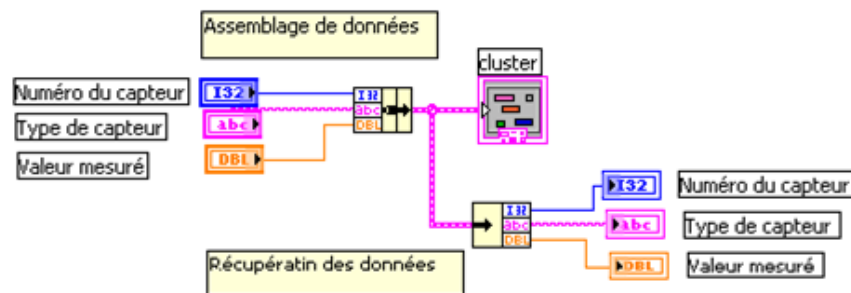


Figure 3.11 : Exemple de manipulation des données avec le type cluster

3.5. Structure de programmation dans LabVIEW

Les structures sont des représentations graphiques des boucles et des conditions des langages de programmation textuels. On utilise des structures dans le diagramme pour répéter des blocs de code et pour exécuter le code de manière conditionnelle ou dans un ordre spécifique.

3.5.1. Structures Séquence

Les structures Séquence contiennent un ou plusieurs sous-diagrammes, ou étapes, qui s'exécutent dans l'ordre séquentiel. Dans chaque étape d'une structure Séquence, comme dans le reste du diagramme, la dépendance des données détermine l'ordre d'exécution des nœuds. Les structures Séquence ne sont pas couramment utilisées dans LabVIEW.

Il existe deux types de structures Séquence : la structure Séquence déroulée et la structure Séquence empilée.

- **La structure Séquence déroulée**

Illustrée ci-après, affiche toutes les étapes à la fois et les exécute de gauche à droite quand toutes les données câblées aux étapes sont disponibles, jusqu'à ce que la dernière étape ait été exécutée. Les valeurs de données quittent chaque étape lorsque celle-ci finit de s'exécuter.

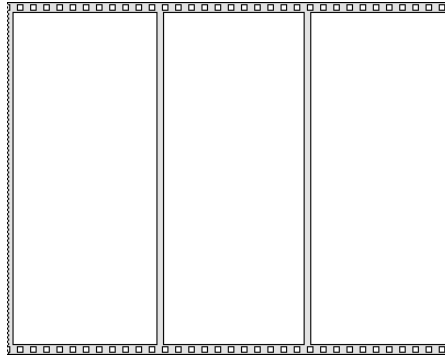


Figure 3.12 : Structure séquence déroulée

- **La structure Séquence empilée**

Représentée ci-après, empile les étapes, si bien que vous ne pouvez voir qu'une étape à la fois, et exécute l'étape 0, puis l'étape 1, et ainsi de suite, jusqu'à l'exécution de la dernière étape.

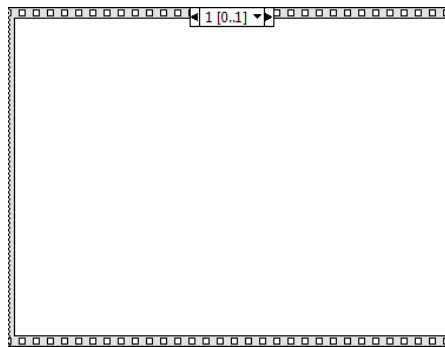


Figure 3.13 : structure séquence empilée

3.5.2. Structure Condition

La structure peut être considérée comme un IF. Elle traite tout type de données (entrée numérique, chaîne, booléen, énumération...).

Une structure comporte plusieurs sous-diagrammes dont un seul sera exécuté selon la condition qui est réalisée. La condition à tester est câblée à l'entrée « ? ».

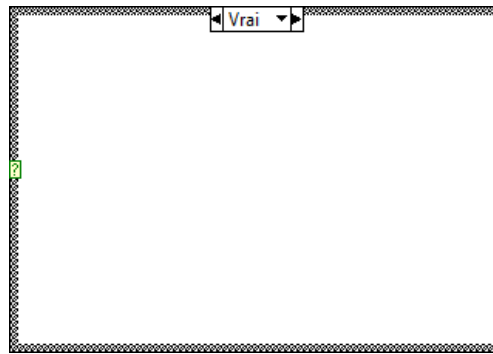


Figure 3.14 : Structure condition

3.5.3. Structures d'itération

Le langage G de LabVIEW possède 2 structures permettant de répéter l'exécution d'un Ensemble de VIs; les boucles For (pour) et While (tant que).

- **Boucle For**

La boucle For s'exécute un nombre de fois défini par le nombre N. L'indice i d'itération va toujours de 0 à (N-1).

Comme en langage C, c'est une structure itérative permettant d'exécuter une partie de programme un nombre déterminé de fois, connu à l'avance.

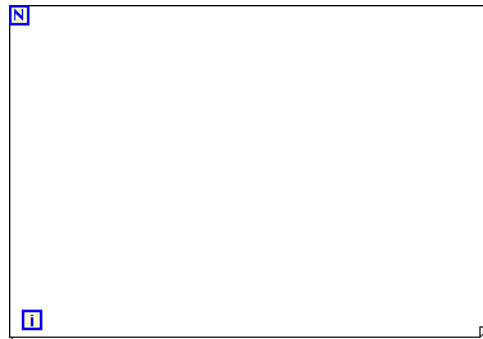


Figure 3.15 : Boucle four

- **Boucle While**

Pour la boucle For, le terminal de décompte N est connu à l'avance. La valeur associée à ce terminal détermine le nombre d'itérations.

Cette boucle exécute un ensemble de VIs jusqu'à ce qu'une condition d'arrêt soit remplie.

La boucle While s'exécute au moins une fois. Elle s'exécute indéfiniment jusqu'à ce que le terminal d'arrêt reçoive une valeur logique 1 ou T (vraie).

Cette condition peut se réaliser par programmation ou par l'appui sur un bouton stop sur la face-avant qui envoie un 1 logique sur le terminal d'arrêt de la boucle While.

La boucle While s'exécute aussi longtemps que la valeur du terminal conditionnel reste TRUE (vrai).



Figure 3.16 : boucle while

Dans l'exemple suivant (Figure 3.17), nous traçons un signal aléatoire uniforme dans un graphe déroulant. Les différentes itérations sont cadencées par une période de 200 ms.

La boucle ne s'arrête que lorsqu'on appuie sur le bouton STOP.

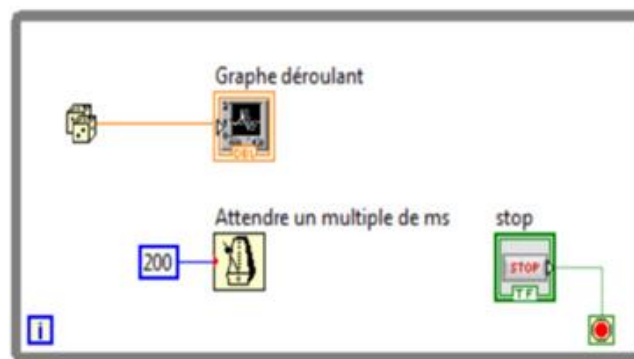


Figure 3.17 : Exemple d'utilisation de la boucle while

3.6. Traitements de données dans LabVIEW

- **Boîtes à outils mathématique**

LabVIEW contient aussi des boîtes de calcul mathématique qui servent à introduire des commandes complexes.

La boîte de calcul est un nœud en mode texte, à l'instar du nœud MathScript, qui permet d'effectuer des calculs dans le diagramme d'un VI.

Les boîtes de calcul permettent d'éviter des opérations graphiques qui nécessitent l'utilisation d'un grand nombre de VIs.

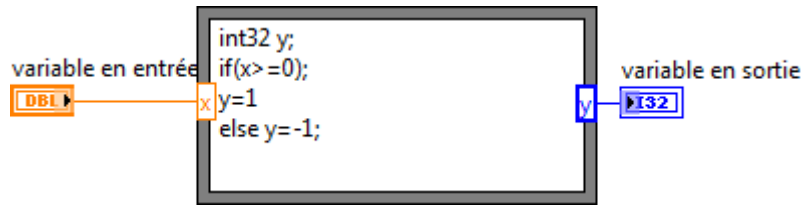


Figure 3.18 : Exemple d'utilisation de la boîte de calcul

- **Bibliothèque de contrôle PID**

La sous-palette PID contient beaucoup de structures de ce régulateur avec toutes les options que l'on trouve généralement dans les systèmes de régulation.

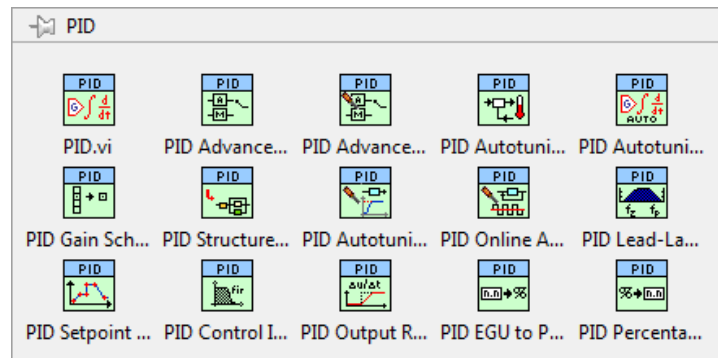


Figure 3.19 : Sous palette PID

- **Bibliothèque de logique floue**

Elle est destinée à accélérer le développement d'applications de contrôle pour les systèmes non linéaires ou complexes. Elle propose une interface graphique prête à l'emploi pour la conception de contrôleurs de logique floue et des VI pour créer ces contrôleurs dans LabVIEW.

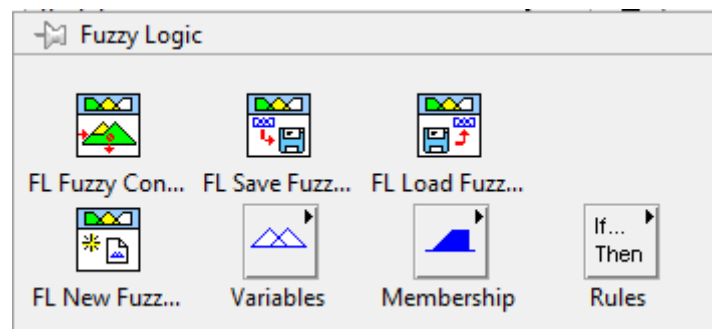


Figure 3.20 : Sous palette logique floue

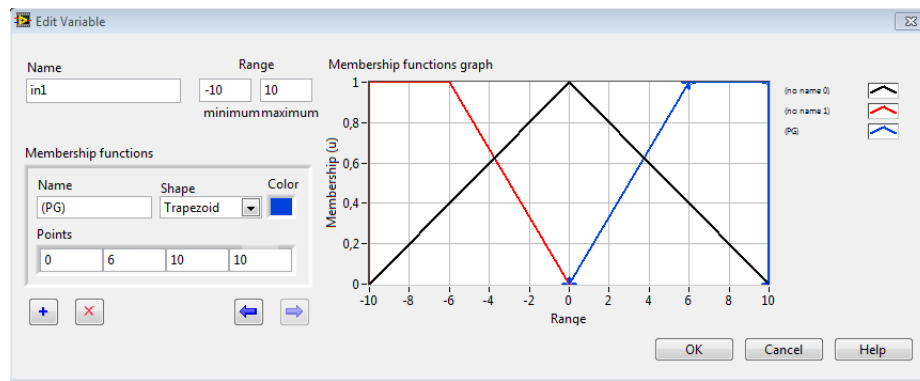


Figure 3.21 : Interface graphique pour la conception d'une commande floue

3.7. Graphes dans LabVIEW [7]

Il existe 3 types principaux de graphes dans LabVIEW :

- le graphe déroulant qui affiche les courbes avec leurs valeurs dans le temps (point par point).
- le graphe qui affiche les courbes une fois que toutes leurs valeurs sont connues et mémorisées dans des tableaux.
- le graphe XY ou courbe paramétrique de Y en fonction de X.

Ces graphes sont définis dans la face-avant ou panel d'un VI.

3.7.1 Graphe

Le graphe affiche un ou plusieurs tracés de mesures échantillonnées de manière constante. Le graphe ne trace que les fonctions à une seule valeur, par exemple dans $y = f(x)$ avec des points distribués également le long de l'axe des X, comme les waveforms acquises qui varient dans le temps. La figure suivante montre un exemple de graphe.

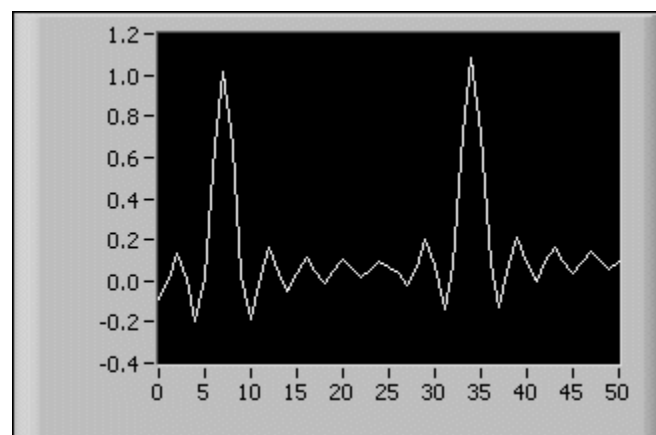


Figure 3.22 : Exemple de graphe

Le graphe peut afficher des tracés contenant n'importe quel nombre de points. Le graphe accepte aussi plusieurs types de données, ce qui minimise la manipulation des données avant l'affichage.

3.7.2 Graphe déroulant

Le graphe déroulant est un type particulier d'indicateur numérique qui affiche un ou plusieurs tracés d'échantillons acquis à une vitesse constante.

La figure suivante montre un exemple de graphe déroulant.

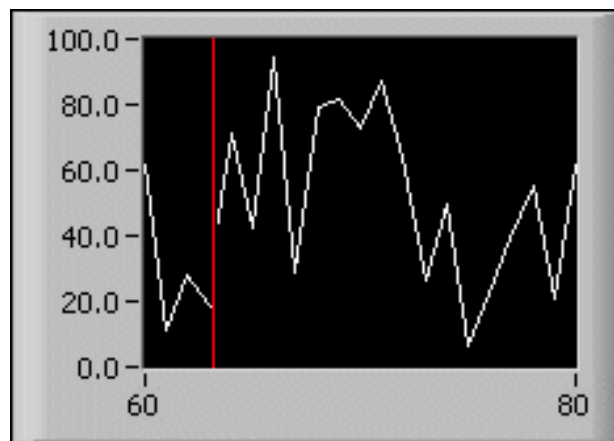


Figure 3.23 : Exemple de graphe déroulant

3.7.3 Graphe XY

Le graphe XY est un objet graphique cartésien à usage général qui trace des fonctions à valeurs multiples, comme des formes circulaires ou des waveforms avec une base de temps qui varie. Le graphe XY affiche des ensembles de points, échantillonnés régulièrement ou irrégulièrement.

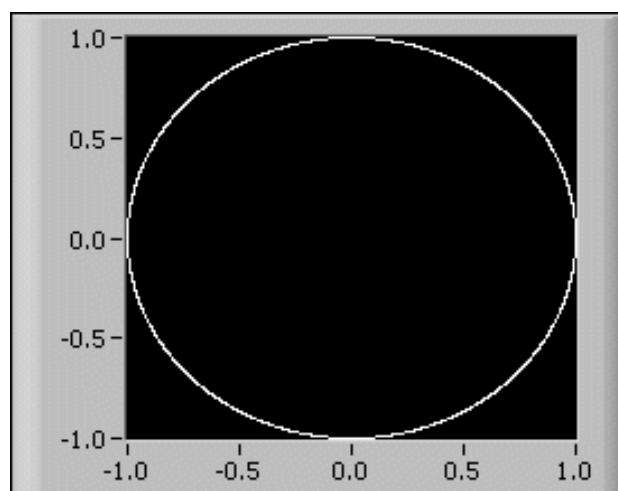


Figure 3.24 : Exemple de graphe XY

3.8. Conclusion

Dans ce chapitre nous avons présenté d'une façon générale le contenu du logiciel LabVIEW qu'on trouve très vaste en fonctions prédéfinies pour programmer toutes applications voulues.

Ce chapitre nous a permis d'explorer l'environnement graphique de LabVIEW et de connaître les différents blocs que nous utiliserons plus tard pour programmer notre plate forme de contrôle de la station de pression.

Chapitre 4

Description matérielle

Chapitre 4

Description matérielle

4.1. Introduction

Quantitativement, la pression est la seconde grandeur physique mesurée industriellement après la température. Du vide poussé de quelques Pascals absolus aux très grandes pressions de plusieurs milliers de bar, les techniques de mesures développées pour les applications sont très variées.

4.2. Description de l'unité

L'unité de pression PUP-4 (figure 3.1) est constituée d'un réservoir d'air, un compresseur actionné par un moteur électrique qui fournit de l'air servant à atteindre une pression souhaitée et la maintenir.

L'actionneur de l'unité est une vanne proportionnelle (proportional valve) à commande électrique située sur le côté de refoulement.

Un transducteur de pression qui fournit le signal de rétroaction, il se trouve sur un côté du réservoir.

L'unité comporte un manomètre (manometer) pour la lecture de pression, elle comporte aussi une vanne manuelle perturbatrice (Hand valve).

Enfin une valve de pression maximale soupape de sécurité (Safety valve), située en série sur le coté de refoulement qui sert à éviter les pressions, dangereuses à l'intérieur du réservoir et le blocage du compresseur.

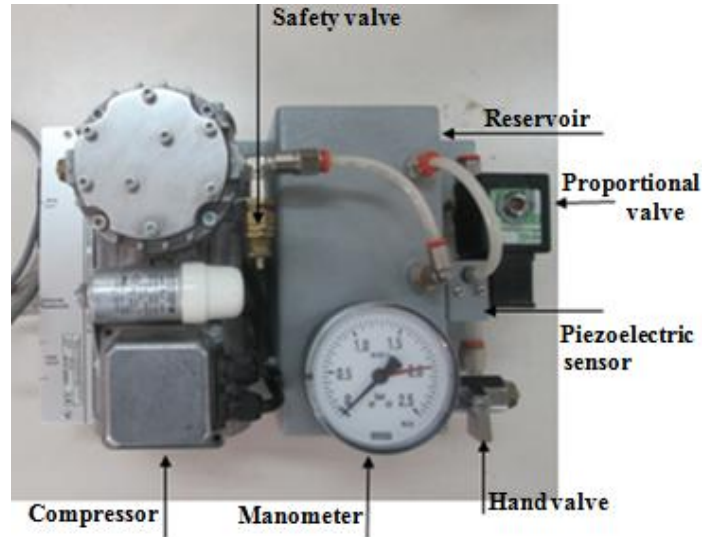


Figure 3.1 : Processus de pression PUP-4

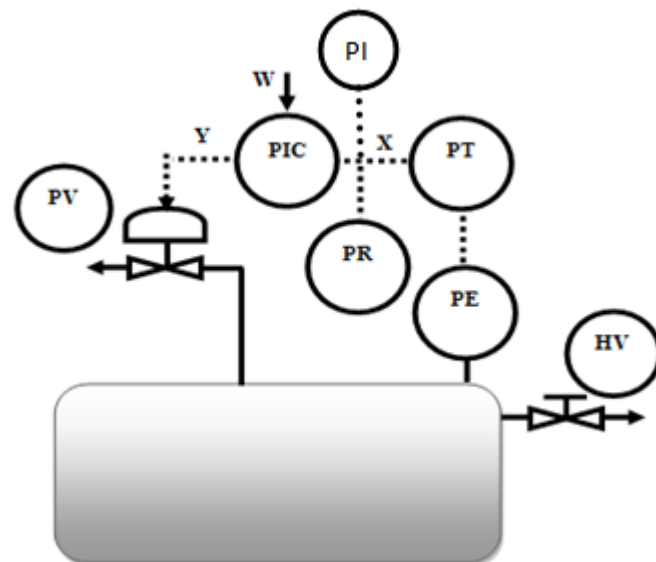


Figure 3.2 : Schéma P&ID de la station et la plate forme

PV : Vanne proportionnelle

PIC : Régulateur indicateur de pression

PT : Transmetteur de pression

PR : Enregistreur de pression

PE : Capteur de pression

HV : Vanne manuelle

PI : Indicateur de pression

4.3. Contrôle automatique de la vanne proportionnelle

Le contrôle automatique de pression est très répandue en industrie, sa structure de base est identique à celle des autres automatismes, sauf que la vanne proportionnelle qui est l'actionneur à contrôler nécessite un amplificateur de puissance.

On peut appliquer les différents régulateurs qu'on connaît déjà :

- On/Off, régulation Tout Ou Rien.
- Proportionnelle.
- Proportionnelle et intégrale.
- Proportionnelle, intégrale et dérivée.
- Régulateur flou

4.4. Transducteur de pression (capteur de pression)

En physique on définit la pression comme le rapport (Force /Surface), dans le système international l'unité de mesure de la pression est le pascal (Pa) qui correspond à 1N/m^2 .

En général dans l'industrie, on trouve d'autres unités de pression : Bar, PSI,

4.4.1. Méthodes de mesure de pression

- **pression absolue** : Pression mesurée par rapport au vide parfait.
- **Pression relative** : Pression mesurée par rapport à la pression atmosphérique.
- **Pression différentielle** : La différence de pression mesurée entre deux sources de pression.

4.4.2. Transducteur de pression de la station PUP-4

Le transducteur est fabriqué par la société Micro-Switch Honey Well

Les caractéristiques les plus importantes de ce transducteur sont :

Champ de mesure	0 à 30 PSI (0 à 2 bars)
F.S.O	195 Mv
Linéarité	0,25% FSO
Temps de réponse	1 ms

Tableau 3.1 : Caractéristique du transducteur

La (figure 3.3) montre en détail les différents composants du transducteur utilisé dans la station

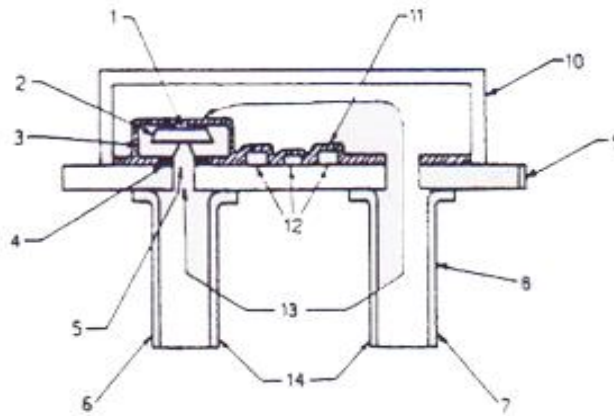


Figure 3.3. Transducteur RC/EV

- 1 Élément capteur
- 2 Cavité du capteur
- 3 Cristal du silicium
- 4 Point de relâchement
- 5 Entrée alternative (transducteur absolu ou transducteur différentiel)
- 6,7 Deux orifices
- 8 Placage de laiton (alliage de cuivre et de zinc)
- 9 Substrat de céramique
- 10 Revêtement de céramique
- 11 Couche de pyralène
- 12 Composant du circuit
- 13 Pression
- 14 Orifice double (différentielle)

4.5. Conditionnement du signal du capteur

Pour faire l'acquisition du signal de sortie du transducteur, il faut d'abord le conditionner pour le rendre lisible par les cartes d'acquisitions. La plage de conditionnement disponible est (0~ 10V). La (figure 3.4) montre le schéma électronique du conditionneur :

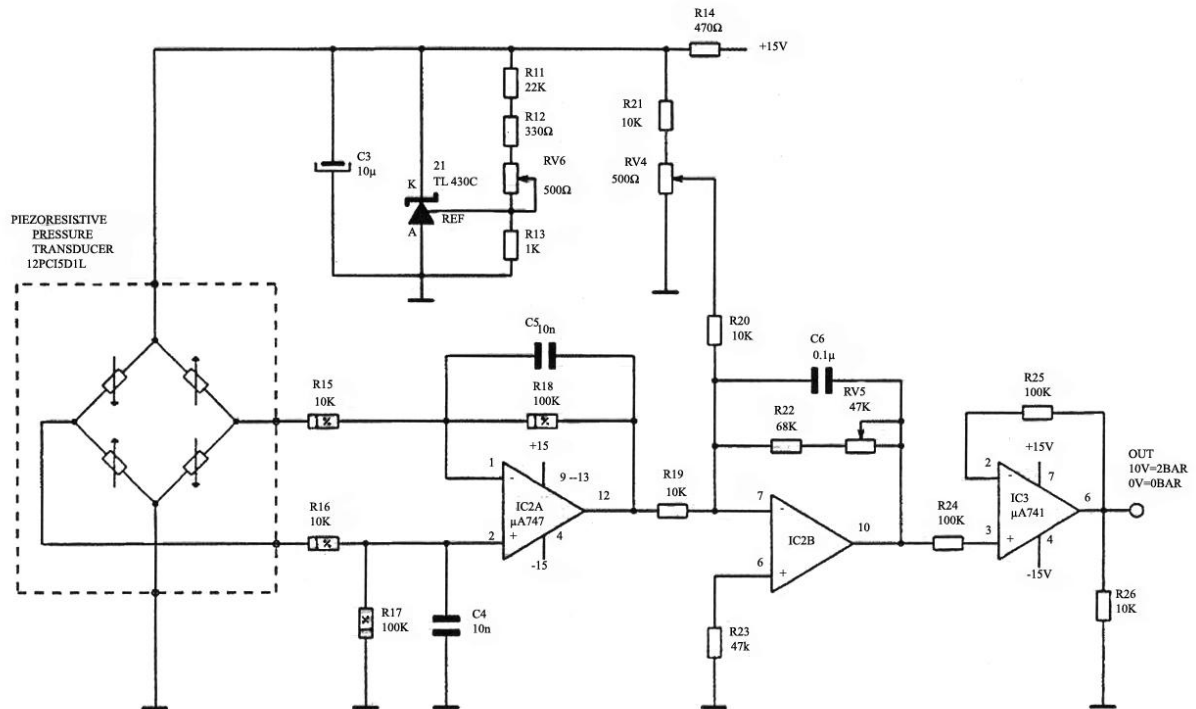


Figure 3.4 : Schéma électronique du conditionneur

4.6. Amplificateur de puissance de la vanne proportionnelle

L'organe réglant (vanne proportionnelle) est l'élément qui assure la régulation de pression de la station. Cette vanne est alimentée par (0~ 24V) .

Pour fournir cette tension il faut réaliser un amplificateur de puissance comme le montre la figure 3.5 (entrée(0~ 10V), sortie(0~ 24V)).

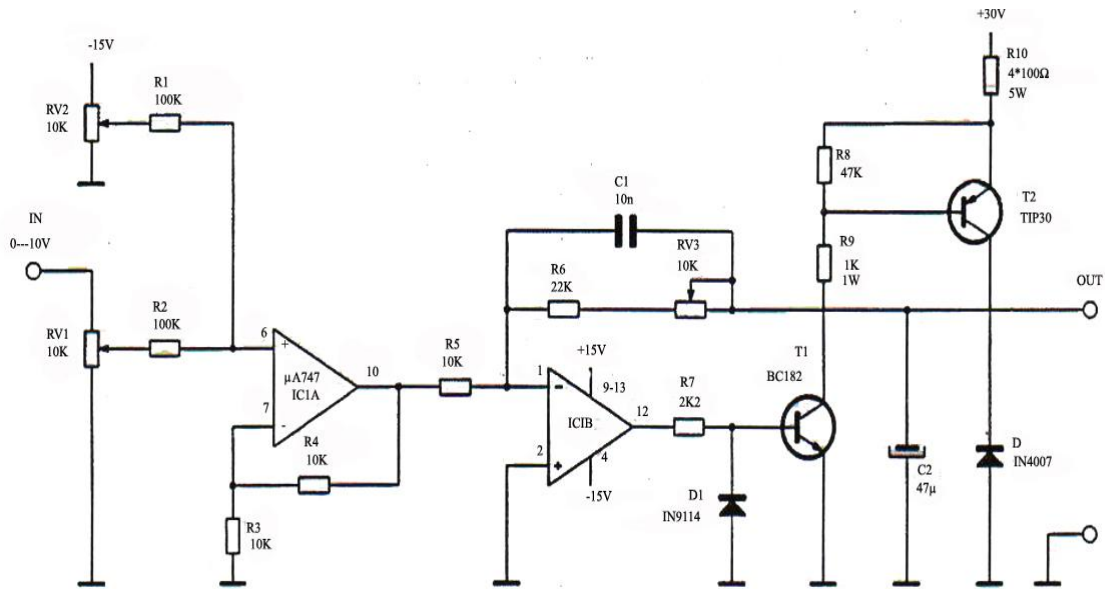


Figure 3.5 : Schéma électronique de l'amplificateur de puissance

L'amplificateur et le conditionneur sont rassemblés sous un seul boîtier comme le montre la (figure 3.6).

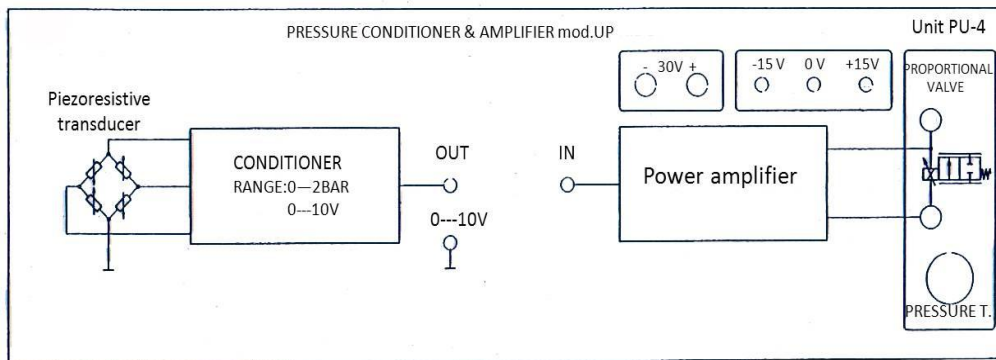


Figure 3.6 : Bloc conditionneur et amplificateur de puissance

4.7. Carte d'acquisition Labjack

Pour commander notre système avec un ordinateur, on fait appel aux cartes d'acquisitions qui assurent la communication entre l'ordinateur et la station PUP-4.

La carte "Labjack U3-LV" est un dispositif d'acquisition de données à connexion USB. Doté de 16 entrées/sorties. Il se prêtera à de multiples applications. La (figure 3.6) montre un Labjack U3-LV (low voltage).

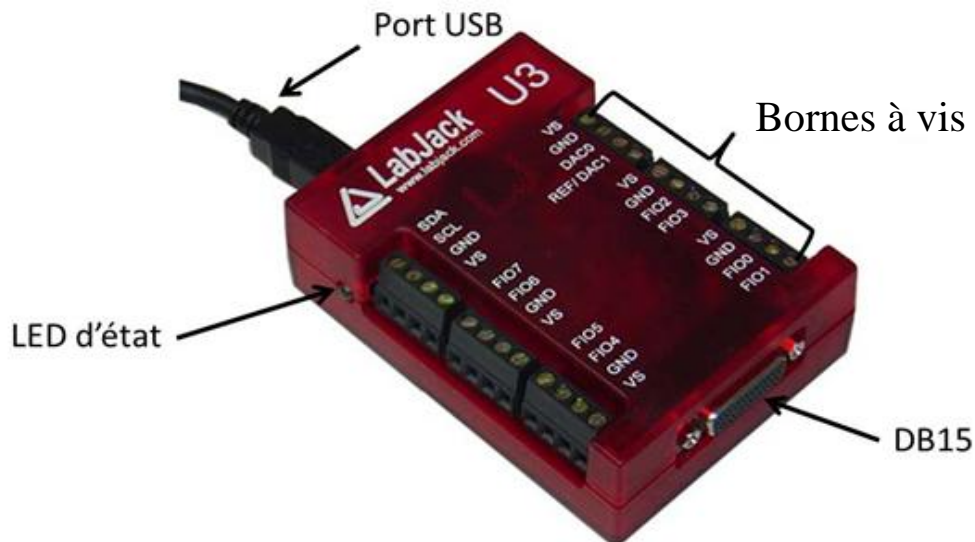


Figure 3.7 : Carte d'acquisition Labjack

4.7.1. Description de la carte Labjack U3_LV

La carte se compose :

- **Port USB (universal serial bus)** : qui assure l'alimentation et la communication.
- **LED d'état** : Led verte située sur le bord gauche de l'appareil, elle clignote lors de la réinitialisation de l'appareil puis reste allumée d'une façon permanente.
- **GND et SGND (Ground)** : Les bornes GND et SGND disponible sur les bornes à vis et le DB15 offre une masse commune qui est reliée à la masse de l'ordinateur.
- **VS (voltage supply)** : les bornes de VS sont conçues comme des sorties d'alimentation d'une valeur de 5V (tension fournie par le câble USB).
- **FIO (flexible input/output), EIO** : Les ports FIO et EIO du Labjack peuvent être configurés individuellement comme entrée numérique, sortie numérique ou une entrée analogique.

Les 8 premières lignes (FIO₀ ~ FIO₇) sont disponibles sur les bornes à vis, les autres lignes (EIO₀ ~ EIO₇) sont disponibles sur le DB15.

- **AIN (analog input)**: On peut configurer sur le Labjack jusqu'à 16 entrées analogiques ($FIO_0 \sim FIO_7$) et ($EIO_0 \sim EIO_7$). La plage de variation du signal d'entrée analogique est de (0 ~ 2.44V).

- **DAC (data acquisition converter)** : Le Labjack U3 dispose de 2 sorties analogiques (DAC0 et DAC1) qui sont disponibles sur les bornes à vis. Chaque sortie analogique peut délivrer une tension comprise entre (0 ~ 5V) volts.

- **Digital I / O** : On peut configurer jusqu'à 20 canaux d'entrées/sorties numériques. Si une borne est configurée en sortie il délivre soit 0V comme 0 logique, ou 3.3V comme 1 logique.

- **DB15 (data bus)** : Le connecteur DB 15 fait sortir 12 entrée/sortie numériques On peut l'utiliser comme un bus d'extension ou les 8 EIO sont des lignes de données et le 4 CIO sont les lignes de commandes.

4.7.2. Descriptions des différents blocs de programmation du Labjack

- **List all** : Renvoie tous les périphériques connectés d'un DeviceType et Connection Type données.

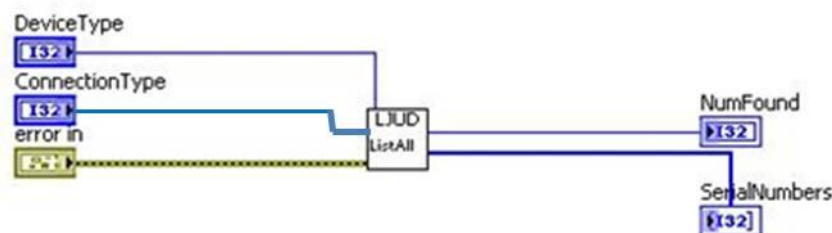


Figure 3.8 : Bloc listAll

Device Type : Le type de Labjack à rechercher.

Connexion Type : Le type de connexion utilisé des Labjack à rechercher.

NumFound : Le nombre de Labjack trouvé.

SerialNumbers : Table des numéros de séries des Labjack trouvés.

Error In : Renvoi le code de l'erreur produite si il y'a eu lieu, 0 sinon.

- **LJUD OPENS** : Utilisé pour appeler un Labjack, cependant, si ce dernier est ouvert il reste jusqu'à ce que l'application se termine.

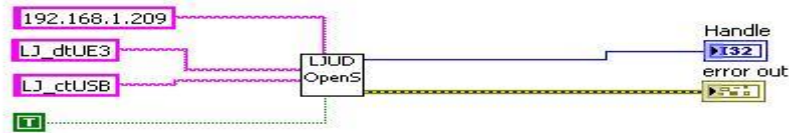


Figure 3.9 : Bloc opens

Device Type et Connection Type : chaînes de caractère pour introduire le type de Labjack et le type de connexion utilisé (USB uniquement pour le U3).

Adress : Adresse IP si on utilise Ethernet ou le numéro de série du Labjack.

First Found : Commande boolienne si elle est égale à 1, Adress et Connection type sont ignorés, et le premier Labjack connecté s’ouvre.

Handle : Code renvoyé par l’Opens et transmit à d’autre bloc pour identifier le Labjack ouvert.

Error Out : Renvoi le code de l’erreur produite si il y’a eu lieu, 0 sinon.

- **LJUD EGETS** : Utiliser pour configurer l’une des bornes du Labjack, soit en entrée analogique, entrée numérique ou sortie numérique.

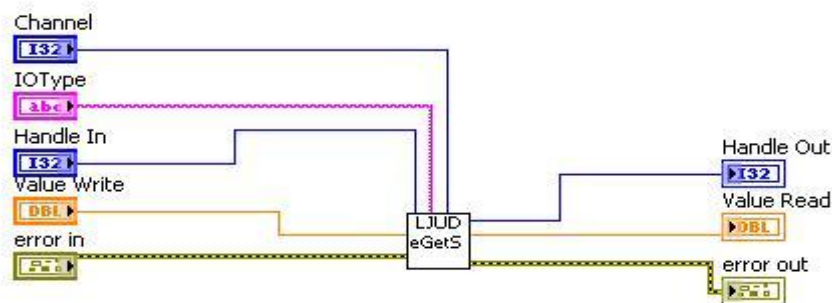


Figure 3.10 : Bloc EGets

Chanel : Le numéro de la borne à configurer (FIO₀ ~ FIO₇)

IOType : Le type de commande

Type de demande	Commande
Entrée analogique	LJ_ioGET_AIN
Sortie analogique	LJ_ioPUT_DAC
Entrée numérique	LJ_ioGET_DIGITAL_BIT
Sortie numérique	LJ_ioPUT_DIGITAL_BIT

Tableau 3.2 : Liste des commandes

Pour les sorties, analogiques, on peut configurer que les deux bornes DAC0 et DAC 1.

Handle In et handle Out : handle in, reçoit le code approprié du Labjack envoyé par le bloc LJUD opens. Handle out, renvoie ce code pour d'autres blocs.

Value Write, value Read : Si on configure la borne en entrée, on utilise Read qui nous permet d'afficher la valeur de la mesure, sinon on utilise Write pour faire sortir une valeur désirée.

Error In, error Out : Error in, reçoit les erreurs des précédents blocs. Error out, renvoie toutes les erreurs.

- **Add request** : Ajouter un élément à la liste des commandes à effectuer.
- **Go One** : Après avoir utilisé add request pour faire une liste de commande, on appelle Go One, pour les exécuter.
- **Get result** : Affiche les résultats des requêtes exécutées par Go One.
- **Erreur to String** : Convertit l'erreur code en une chaîne de caractère.
- **E AIN, E DAC, E DI, E DO** : Des fonctions simples, au lieu d'utiliser le bloc eGets avec une commande spécifiée, ces blocs remplacent les différentes commandes comme l'illustre le tableau ci-dessous :

Bloc	Fonction
E AIN	Entrée analogique
E DAC	Sortie analogique
E DI	Entrée numérique
E DO	Sortie numérique

Tableau 3.3 : Blocs des commandes simples

Ces blocs nous servira pour programmer une entrée analogique pour récupérer le signal de mesure, une sortie analogique pour envoyer le signal de commande, et une sortie numérique pour agir sur une vanne Tout ou Rien.

On rappelle que le signal d'entrée du Labjack à une plage de variation de (0V ~ 2.44V) et le signal de sortie est de (0V ~ 5V), mais, la plage de conditionnement du capteur de pression de la station est de (0V ~ 10V) et la tension utile de la vanne proportionnelle est de (0V ~ 24V). Donc, pour adapter ses signaux il faut les conditionner, pour cela on réalise 2 circuits d'adaptations à base des amplificateurs opérationnels.

4.8. Boucle de régulation PID

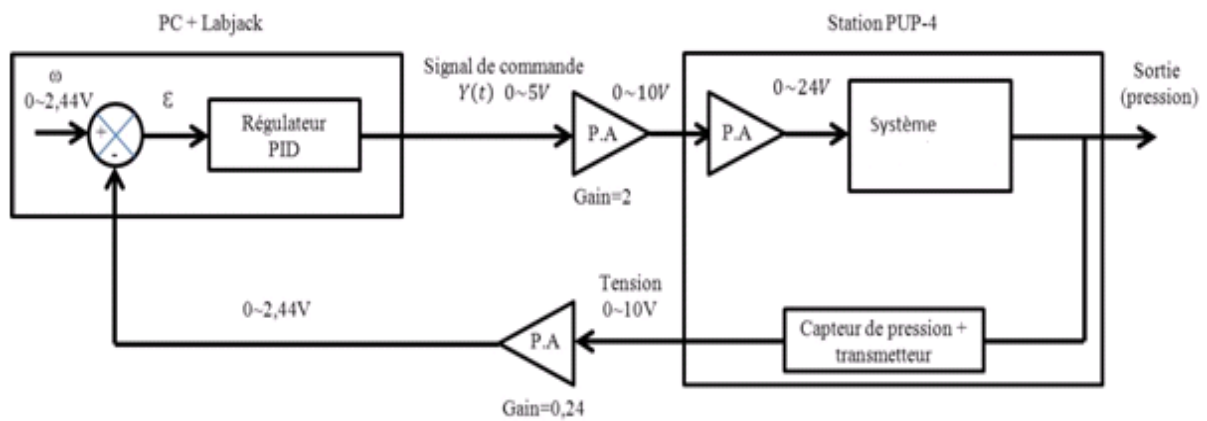


Figure 3.15 : Boucle de régulation PID

PA : Amplificateur de puissance (Power Amplifier).

4.9. Boucle de régulation TOR

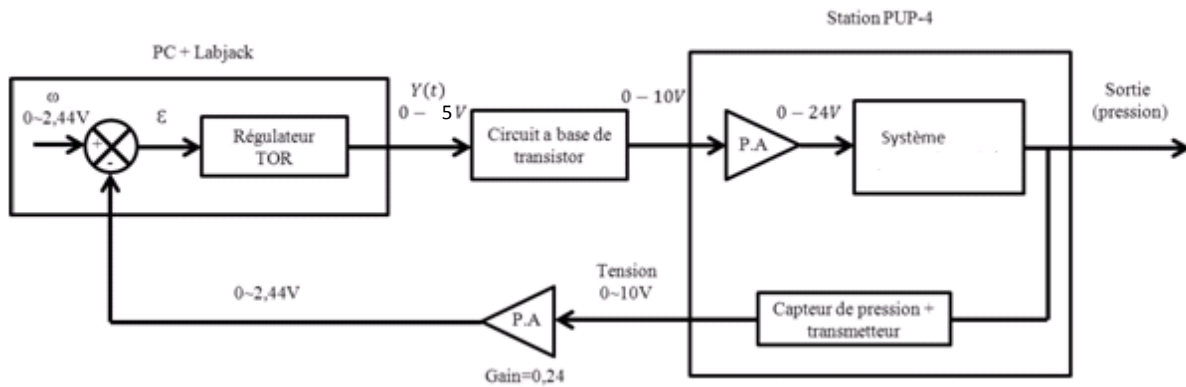


Figure 3.16 : Boucle de régulation TOR

4.10. Boucle de régulation floue

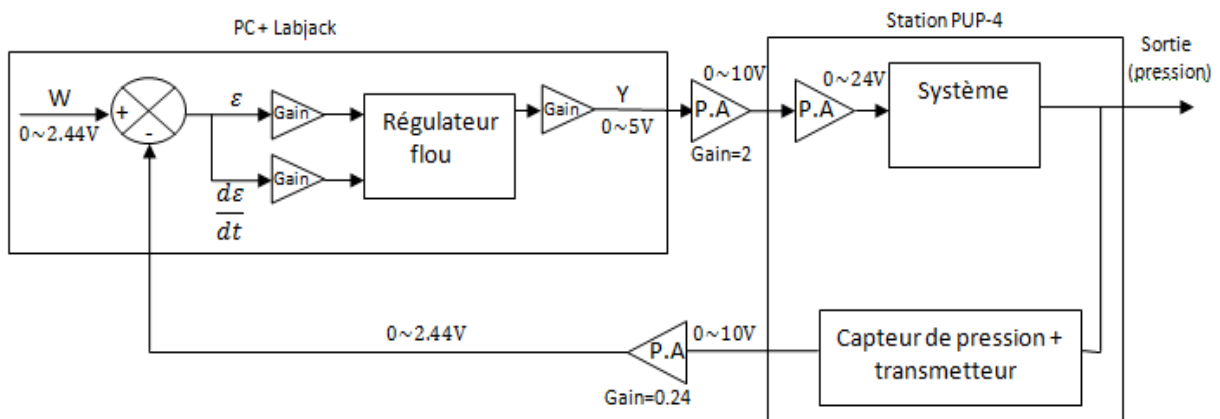


Figure 3.17 : Boucle de régulation floue

4.11. Conclusion

Dans ce chapitre nous avons présenté la station de pression PUP-4 qui sera l'objectif de notre étude. Puis on a décrit la carte Labjack et ses différents blocs de programmation sous LabVIEW. Mais, pour exploiter la station et la carte Labjack et réaliser une boucle de régulation de pression il faut adapter tous les signaux, ce qui nous a amené à exploiter des composants électroniques pour réaliser les circuits d'adaptation.

Chapitre 5

Simulation et Résultats **Expérimentaux**

Chapitre 5

Simulation et Résultats Expérimentaux

5.1. Introduction

Ce chapitre concerne la partie application de notre travail, où on va exploiter les connaissances acquises dans les chapitres précédents pour identifier et implémenter des régulateurs (TOR, PID, Flou,) pour la station de pression PUP-4 d'Electronica Veneta.

Cette étude a pour but, de régler la pression à une valeur bien déterminée. Notre objectif est de trouver, la meilleure commande qui va répondre au cahier des charges imposé, et de faire une comparaison entre les régulateurs Flou et PID.

5.2. Plateforme de commande

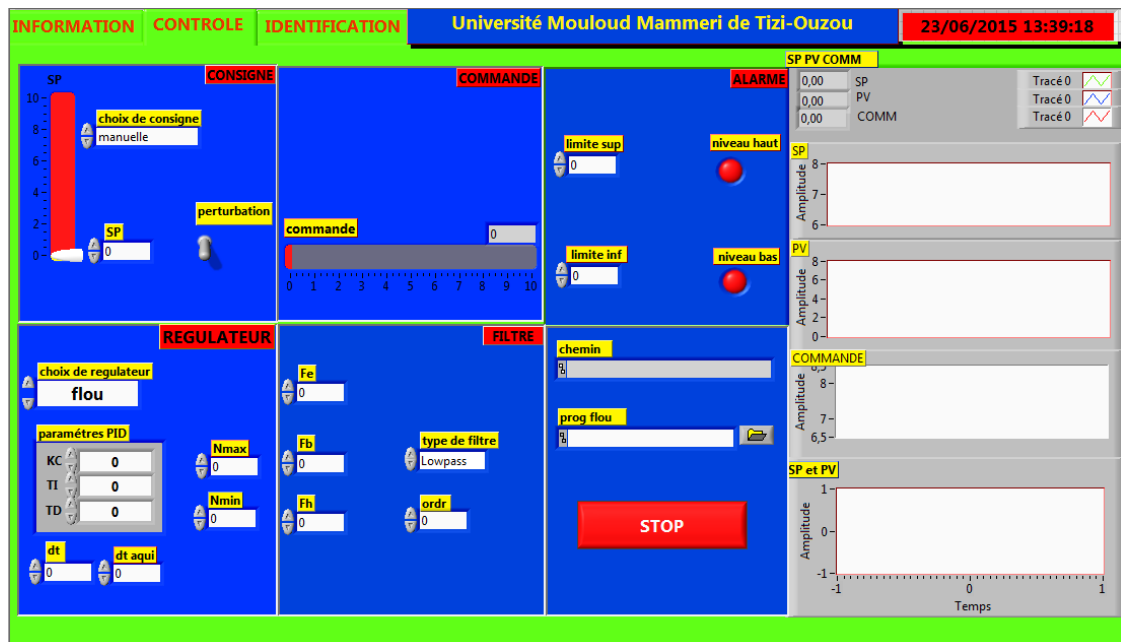


Figure 5.1 : Interface de commande

Nous avons structuré une plateforme de commande en 6 parties essentielles d'une façon à faire apparaître toutes les commandes et les mesures possibles.

- Partie consigne : L'opérateur peut choisir entre, donner une consigne manuelle ou, programmer une poursuite.
- Partie commande : Elle consiste à afficher l'évolution de la commande en temps réel.
- Partie régulateur : Dans cette partie on choisit le type du régulateur (TOR, PID, FLOU), et offre la possibilité de faire varier leur paramètres.
- Partie visualisation : Cette partie regroupe les différents graphes, une fenêtre pour visualiser les signaux (consigne, mesure, commande) séparément, une autre pour visualiser l'évolution de la mesure par rapport à la consigne.
- Partie filtrage : Choisir un filtre et introduire ses paramètres.
- Partie alarme : Alarme réglable pour prévenir l'opérateur.

5.2.1. Programmes Labjack sous LabVIEW

- **Programme pour acquisition de données**

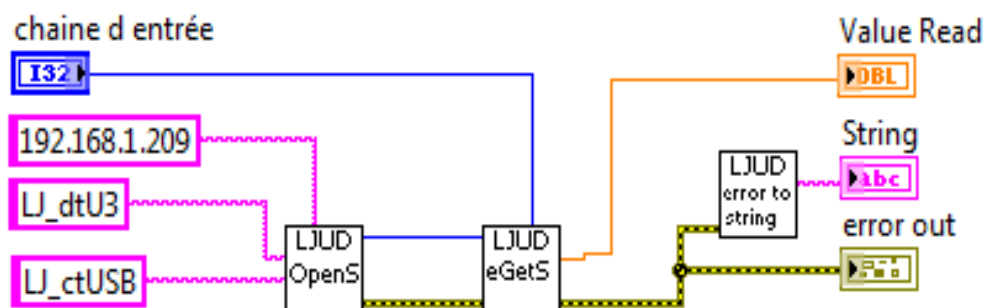


Figure 5.2 : Programme pour acquisition

Le bloc LJUD Opens permet d'appeler la carte Labjack U3.

- **Programme pour la sortie analogique**

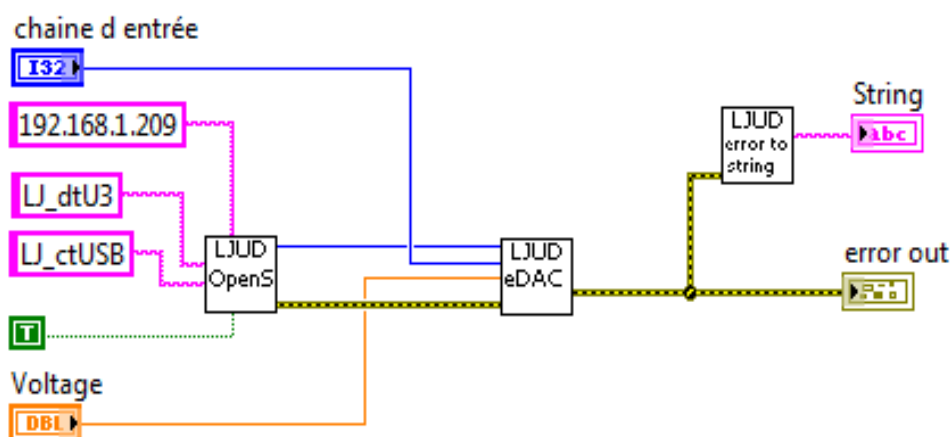


Figure 5.3 : Programme sortie analogique

- Pour la sortie analogique on utilise directement le bloc eDAC.
- Chaîne d'entrée nous permet de sélectionner DAC0 ou DAC1.

- Programme pour la sortie numérique

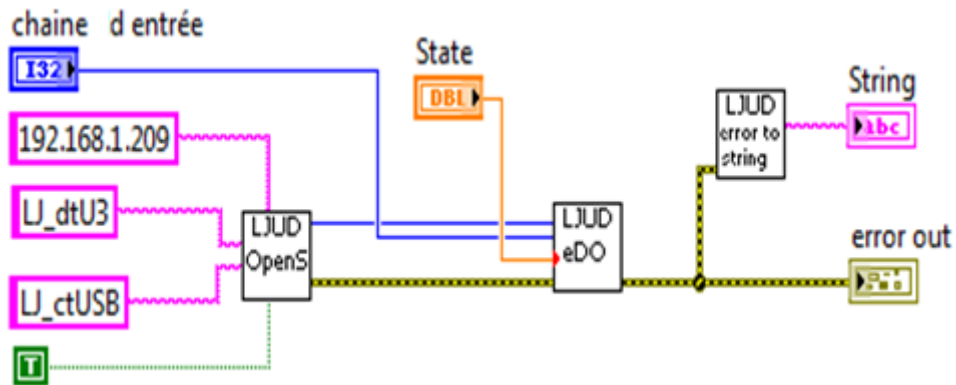


Figure 5.4 : Programme pour sortie numérique

Le bloc LJUD eDO nous permet de faire sortir un signal numérique de (0-3.22V) sur l'un des ports spécifiés sur chanel.

5.2.2. Réalisation d'une poursuite

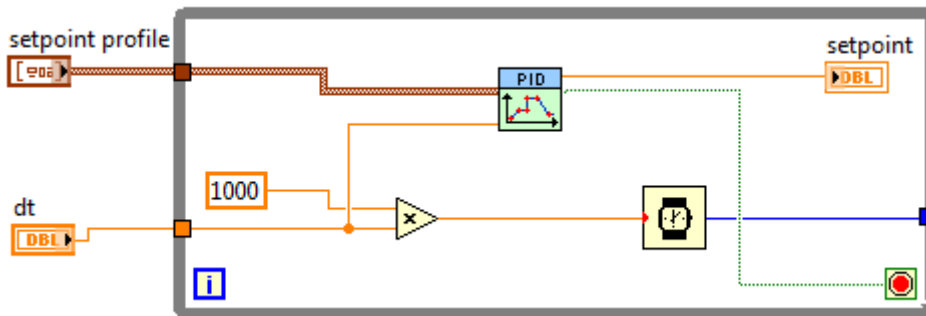


Figure 5.5 : Programme de la poursuite avec Labview

Le bloc setpoint génère des consignes différentes dans le temps.

On introduit notre poursuite dans le cluster d'entrée sous forme de deux variables (la valeur de la consigne (Volt) et sa durée (Sec)).

5.2.3. Archivage des données

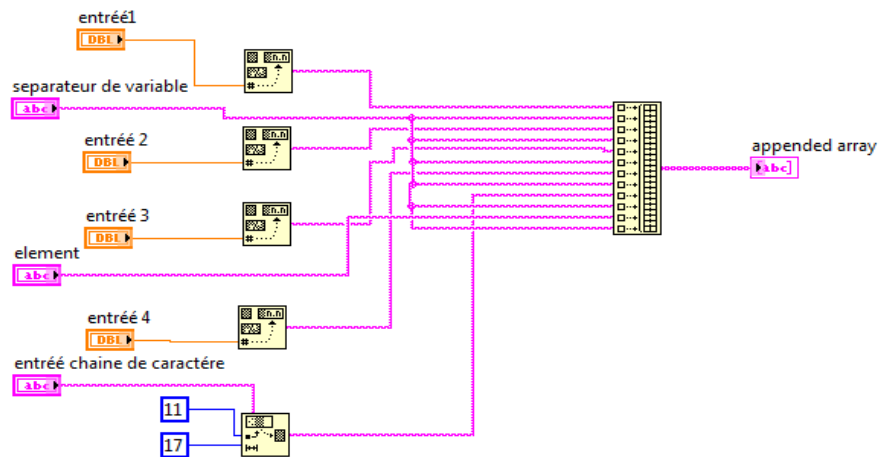


Figure 5.6 : Programme pour l'archivage

Ce programme converti des entiers à des chaînes de caractère pour créer un fichier d'archivage.

5.2.4. Programme d'une alarme sur LabVIEW

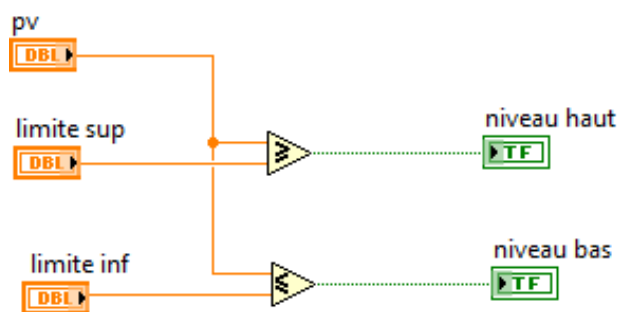


Figure 5.7 : Programme d'une alarme

On compare le signal de mesure (PV) à une limite basse et à une limite haute. On récupère l'information sous forme d'un booléen qu'on va l'utiliser pour faire clignoter des Leds ou faire activer des effets sonores.

5.3. Caractéristique statique du processus

Afin de déterminer la zone linéaire, on trace la caractéristique statique du processus.

Dans notre cas l'entrée est une tension délivrée par une alimentation continue variable de (0V~10V) appliquée sur une vanne régulatrice, la sortie est la pression délivrée par le capteur de pression piézorésistif. Les résultats obtenus sont représentés dans le (tableau 5.1).

Tension d'entrée U_e (V)	1.19	1.8	2.96	3.25	4.06	4.61	5.98	6.73	7.25	8.04	9.57	9.92
Pression (V)	0.28	0.29	0.33	0.35	0.42	0.48	0.93	1.88	3.38	5.70	9.60	10

Tableau 5.1 : Caractéristique statique.

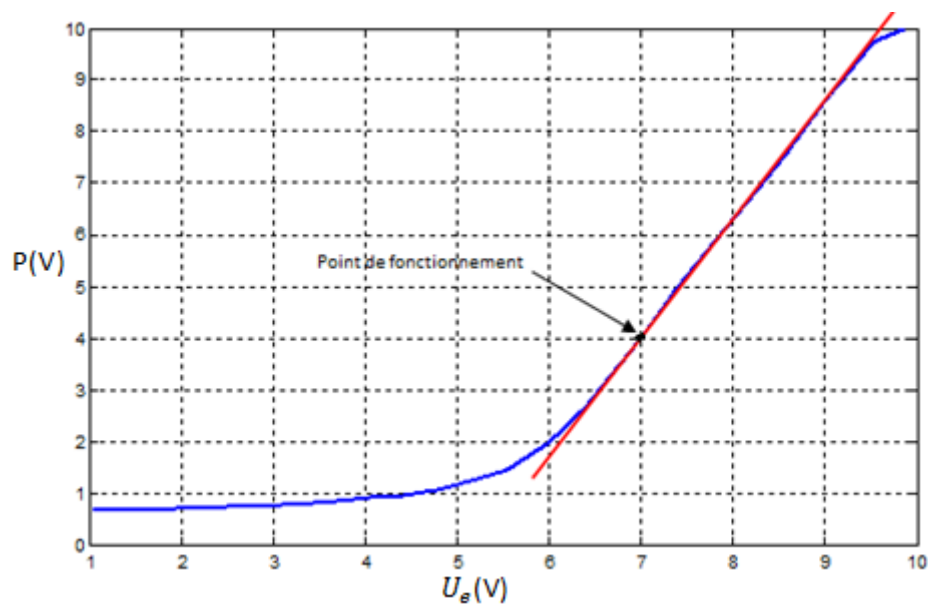


Figure 5.8 : Caractéristique statique

La courbe montre que notre système n'est pas linéaire (contient des zones linéaires).

On choisit la zone linéaire entre 6.5 et 9V.

5.4. Identification des paramètres du modèle

Une fois que nous avons choisis la zone linéaire, on choisit un point de fonctionnement dans cette zone, par exemple 7V et on fait une augmentation de 2% à 5% de cette tension, ici on l'augmente jusqu'à 7.59V comme le montre la (figure 5.9).

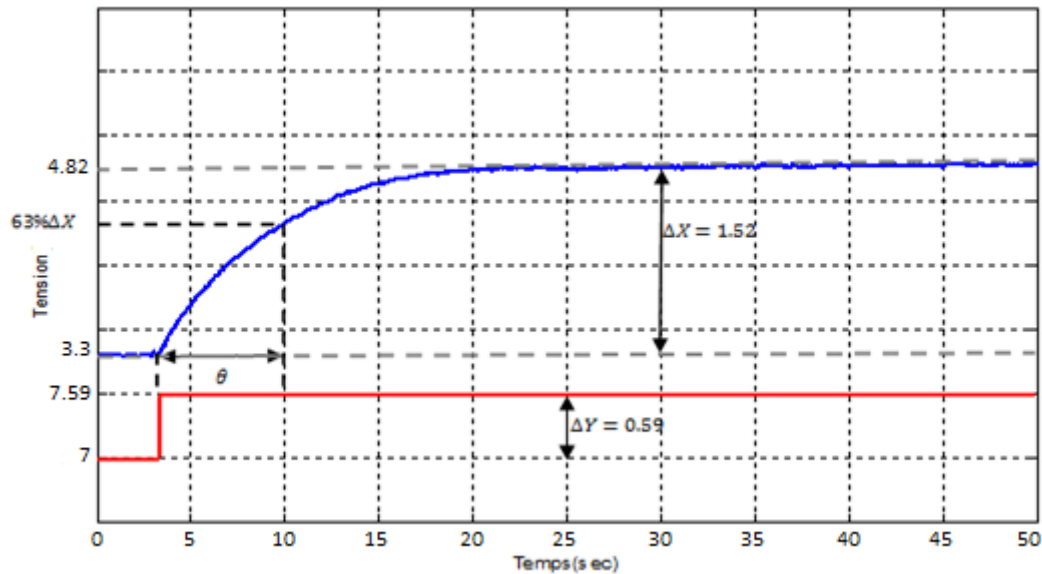


Figure 5.9 : Identification des paramètres du modèle

La courbe obtenue est la réponse d'un système de 1^{er} ordre sans retard, pour obtenir sa fonction de transfert, on utilise la méthode d'identification du modèle de 1^{er} ordre présentée au (chapitre 1).

$$G(p) = \frac{G_S}{(1+\theta p)} \quad (5.1)$$

$$G_S = \frac{\Delta x}{\Delta y} = \frac{4.82-3.3}{7.59-7} = \frac{1.52}{0.59} = 2.57 \quad (5.2)$$

$$\theta \approx 7\text{Sec}$$

Donc on obtient le modèle suivant :

$$G(p) = \frac{2.57}{(1+7p)} \quad (5.3)$$

5.5. Application de la régulation PI

5.5.1. Programme d'un régulateur PID parallèle sur LabVIEW

Pour programmer le PID sous LabVIEW il suffit de traduire l'équation de commande donnée au chapitre 1.

$$Y(t) = K_c \varepsilon(t) + \frac{1}{T_i} \int_0^t \varepsilon(t) dt + T_d \frac{d\varepsilon(t)}{dt} + Y_0 \quad (5.4)$$

y_0 : C'est une constante qui présente la valeur du point nominal de fonctionnement. Ce point est ajustable, le plus souvent il est choisi au milieu de la zone de fonctionnement du régulateur par exemple $Y_0=5V$ pour les signaux $0\sim 10V$ [10].

Les deux blocs dérivateur et intégrateur fonctionnent point par point, tel que la durée que mettent ces derniers pour passer d'un point à l'autre est donnée par dt (période d'échantillonnage).

Pour réaliser le PI, il suffit d'éliminer l'action dérivée en donnant 0 à Td.

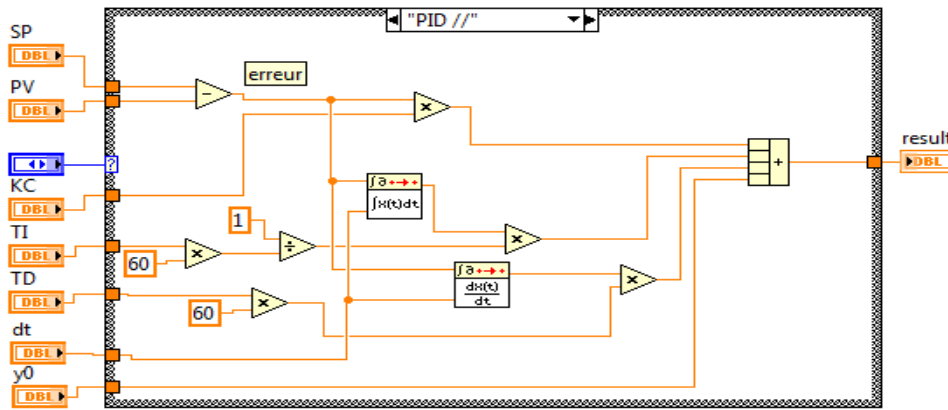


Figure 5.10 : Programme d'un régulateur PID parallèle

5.5.2. Paramètres du régulateur PID

Nous avons vu au chapitre 1 plusieurs méthodes pour calculer les paramètres du régulateur PID.

Parmi ces méthodes on trouve la méthode du modèle de référence.

- **Méthode du modèle de référence**

Après identification de notre système nous avons obtenu le modèle suivant :

$$G(p) = \frac{2.57}{(1+7p)} \quad (5.5)$$

Dans notre modèle de référence, on prend la constante de temps désiré en boucle fermée $\theta_d = 5s$.

Pour un PI parallèle nous avons :

$$G_r = \frac{1}{G_s} \frac{\theta}{\theta_d} \quad (5.6)$$

$$T_i = \theta_d G_s \quad (5.7)$$

Après l'application numérique on trouve les résultats suivants :

$$G_r = 0.54, T_i = 12.84 \text{ sec (0.21min)}$$

- **Stabilité du système**

Tracé de Bode

il est utilisé afin de visualiser rapidement la marge de gain, la marge de phase et sert à étudier la stabilité du système.

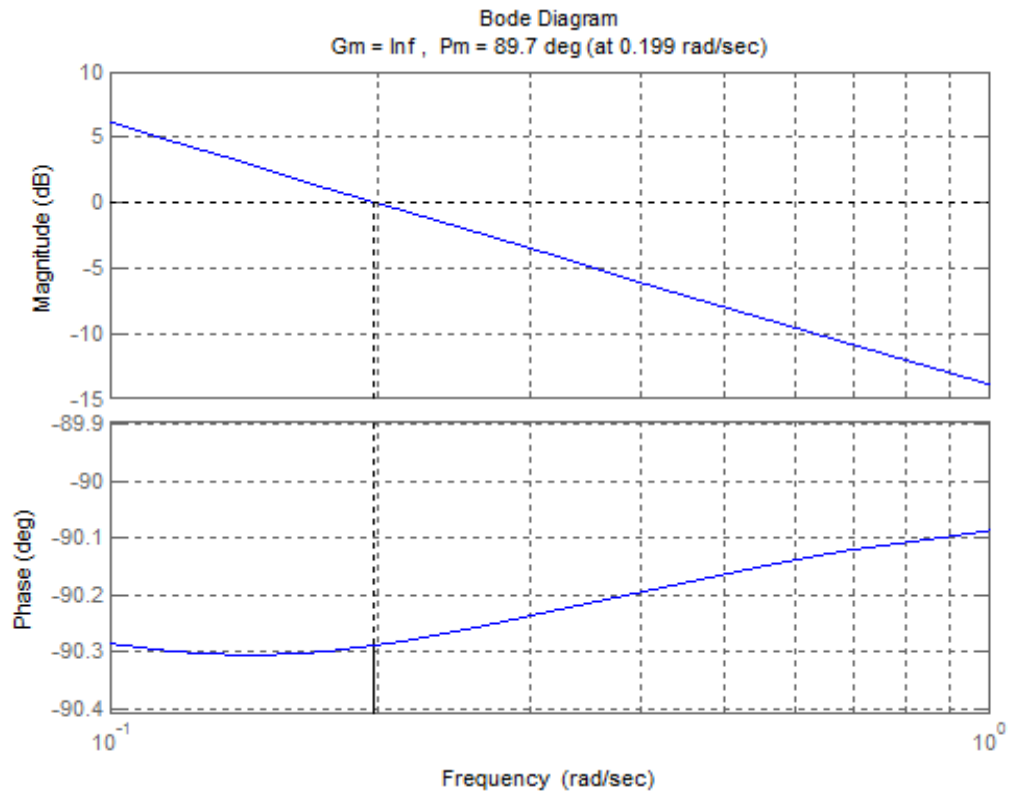


Figure 5.11 : Diagramme de Bode

$$G_m = \text{inf}$$

$$G_m = 89.7$$

On conclut que le système est stable.

5.5.3. Résultats de simulation PID avec Labview

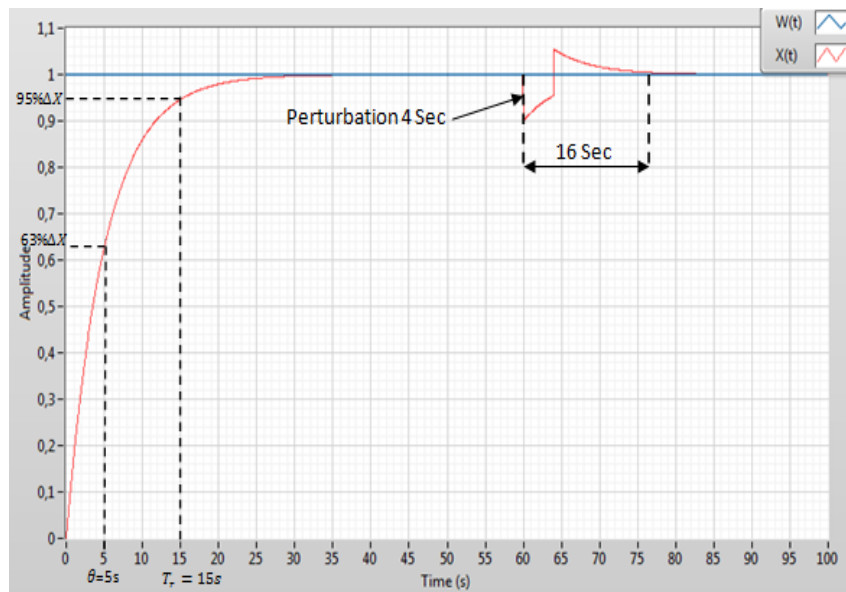


Figure 5.12 : Réponse à un échelon avec application d'une perturbation de 4 Sec
 On remarque que la réponse est sans dépassement avec un temps de réponse $T_r = 15$ Sec
 et que la perturbation est rejeteé après 16 Sec.

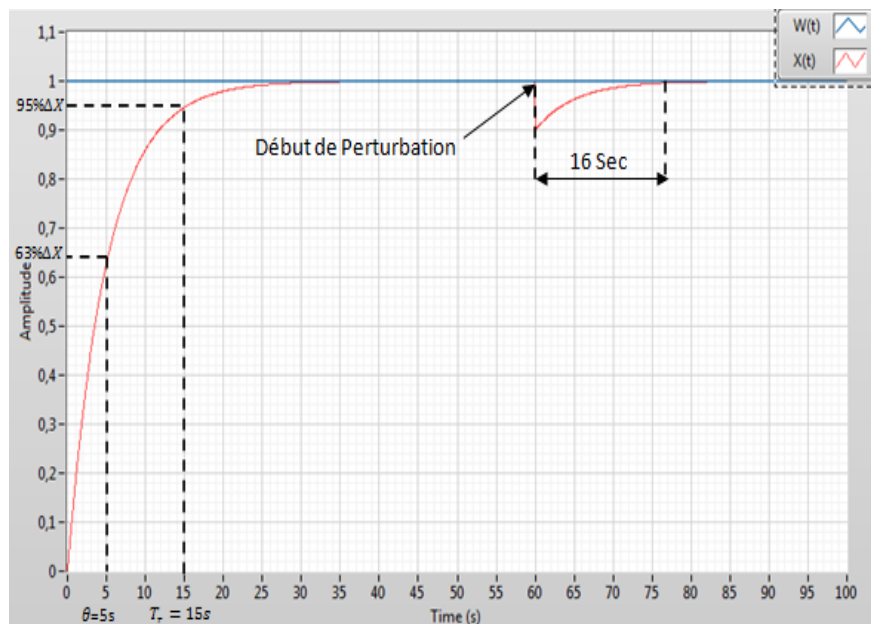


Figure 5.13 : Réponse à un échelon avec application d'une perurbation permanente

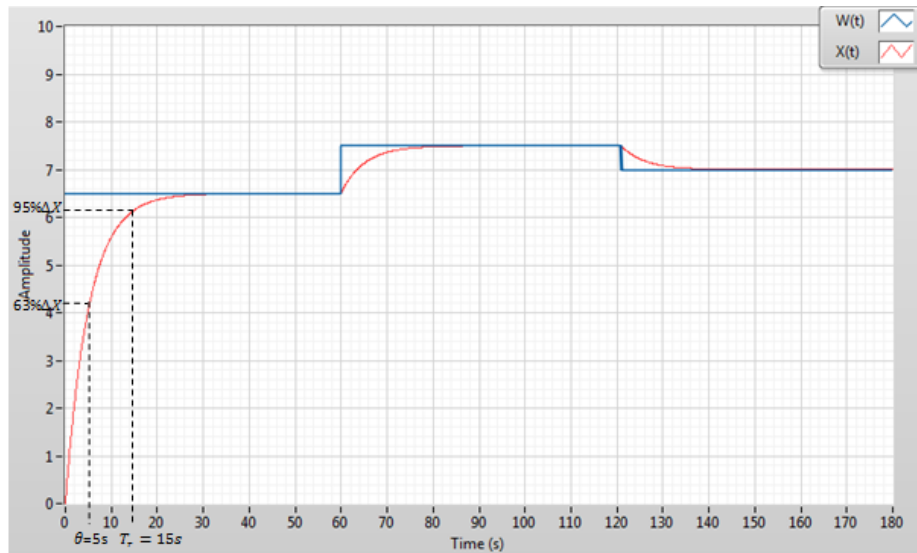


Figure 5.14 : Réalisation d’une poursuite

Selon la figure 5.13 on remarque un bon suivi de consigne et un bon rejet de perturbation qui dure 16 Sec malgré qu’elle est permanente.

La figure 5.14 montre un bon suivi pour changement de consigne.

5.6. Application de la régulation floue

5.6.1. Programme d’un régulateur flou sur LabVIEW

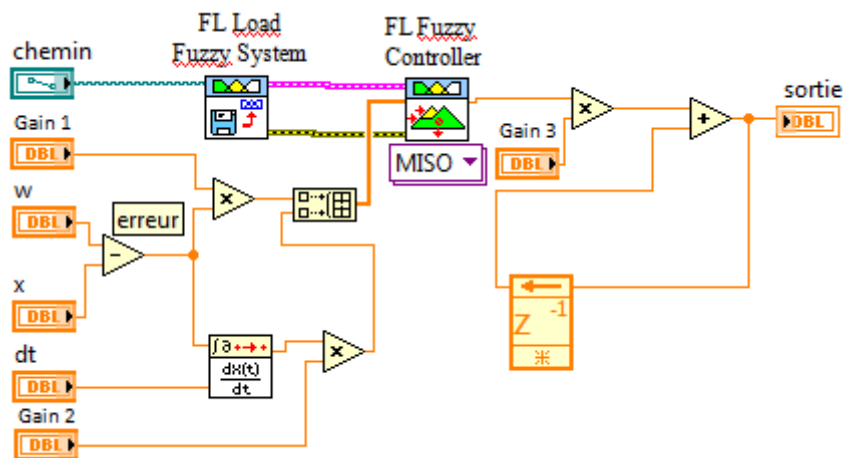


Figure 5.15 : Programme d’un régulateur flou sur LabVIEW

Le type de variable d’entrée ou de sortie, est spécifié par Input/Output, du VI **FL Fuzzy Controller.vi**. Et grâce au VI **FL Load Fuzzy System.vi**, nous récupérons le système flou créé par l’outil **Control Design and Simulation**.

5.6.2. Fonctions d'appartenance

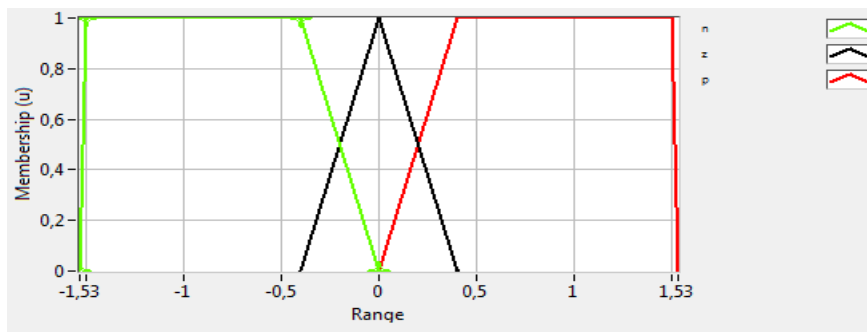


Figure 5.16 : Fonctions d'appartenance de l'erreur

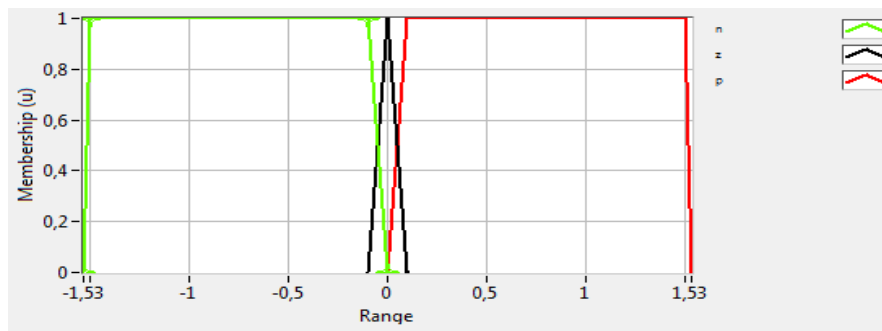


Figure 5.17 : Fonctions d'appartenance de la dérivée de l'erreur

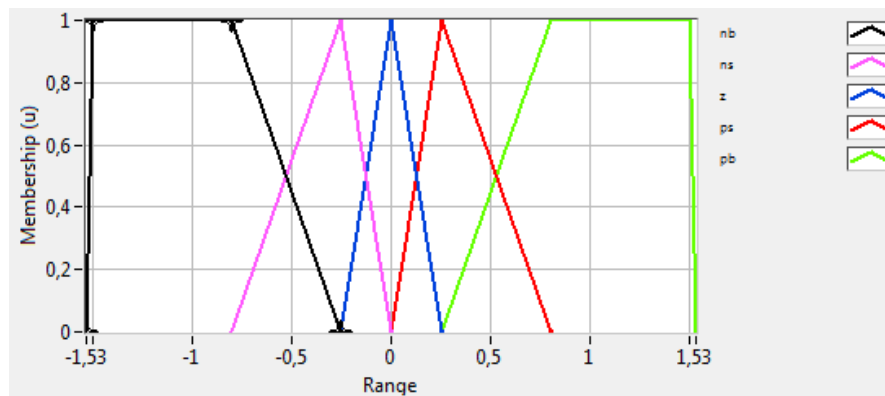


Figure 5.18 : Fonctions d'appartenance de sortie

$\frac{d\varepsilon}{dt} \backslash \varepsilon$	N	Z	P
N	NB	NS	Z
Z	NS	Z	PS
P	Z	PS	PB

Tableau 5.2 : Tableau des règles d'inférences

5.6.3. Résultats de simulation de la régulation floue

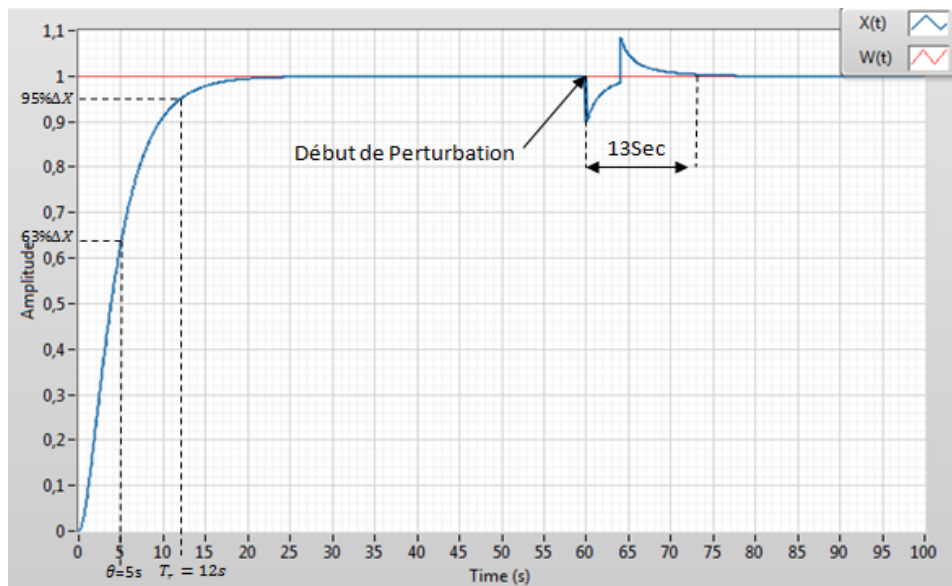


Figure 5.19 : Réponse à un échelon avec application d'une perturbation de 4 Sec

On remarque que la réponse est sans dépassement avec un temps de réponse $T_r \approx 12$ Sec et que la perturbation est rejeté après 13 Sec.

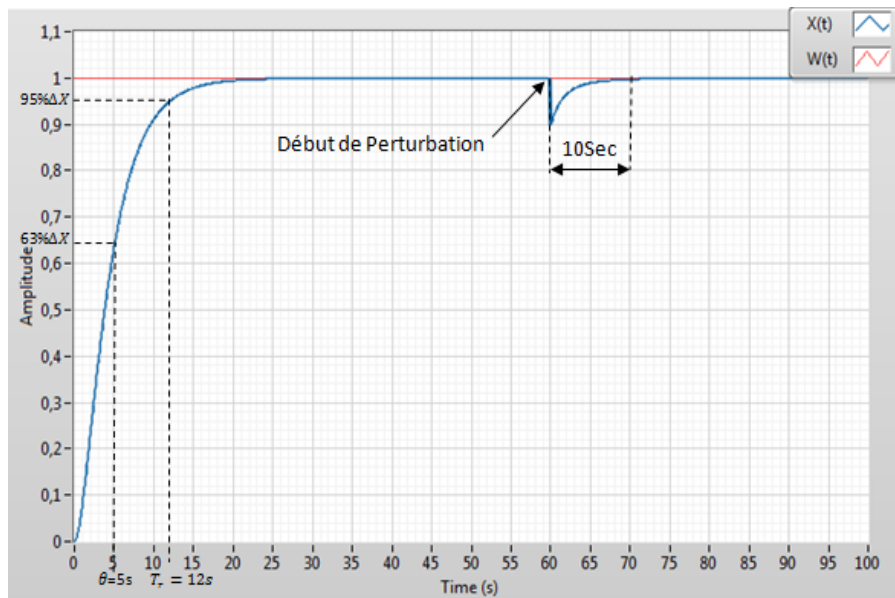


Figure 5.20 : Réponse à un échelon avec application d'une perurbation permanente

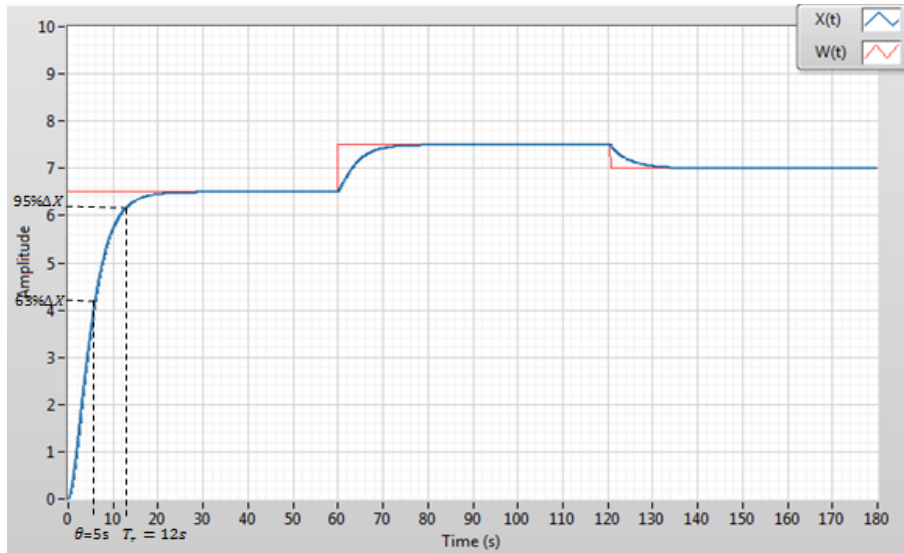


Figure 5.21 : Réalisation d'une poursuite

On remarque un bon suivi, pour le changement de consigne.

5.7. Résultats expérimentaux (TOR, PID, flou)

A. Programme d'un régulateur TOR sur LabVIEW

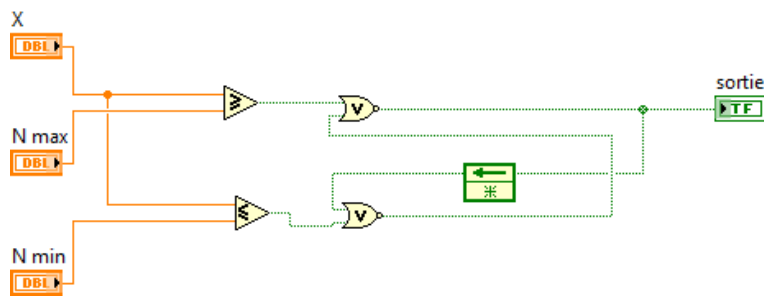


Figure 5.22 : Programme d'un régulateur Tout ou Rien

Avec ce programme, la sortie booléen est mise à 1 si $X \leq N_{min}$ et mise à 0 si $X \geq N_{max}$.

- 1^{er} Essai

On prend $N_{max}=7$ et $N_{min}=3$.

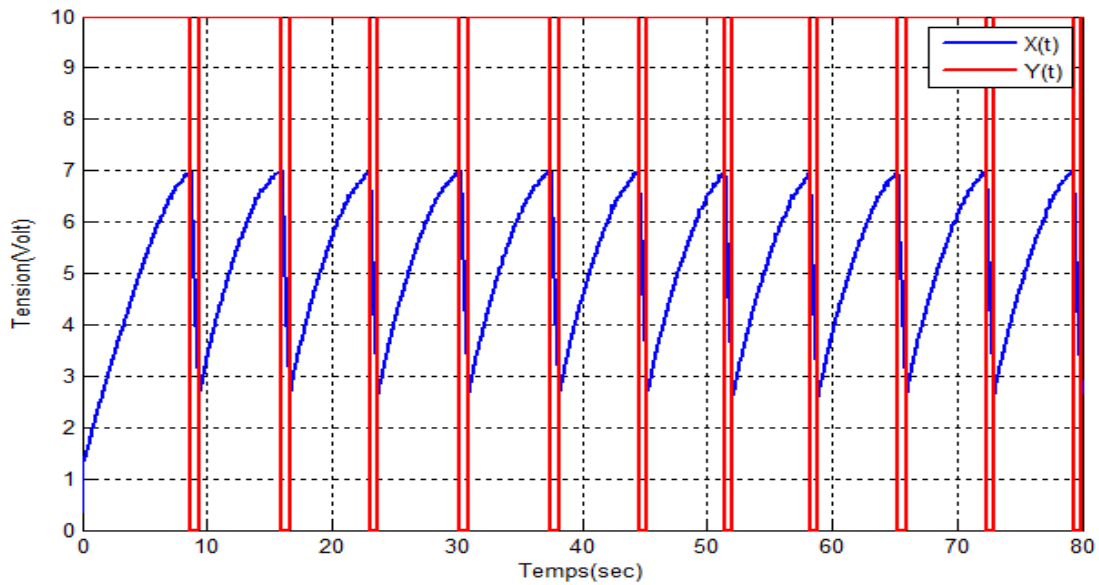


Figure 5.23 : Commande Tout ou Rien ($N_{max}=7$ et $N_{min}=3$).

- 2^{ème} Essai

On prend $N_{max}=5$ et $N_{min}=4$

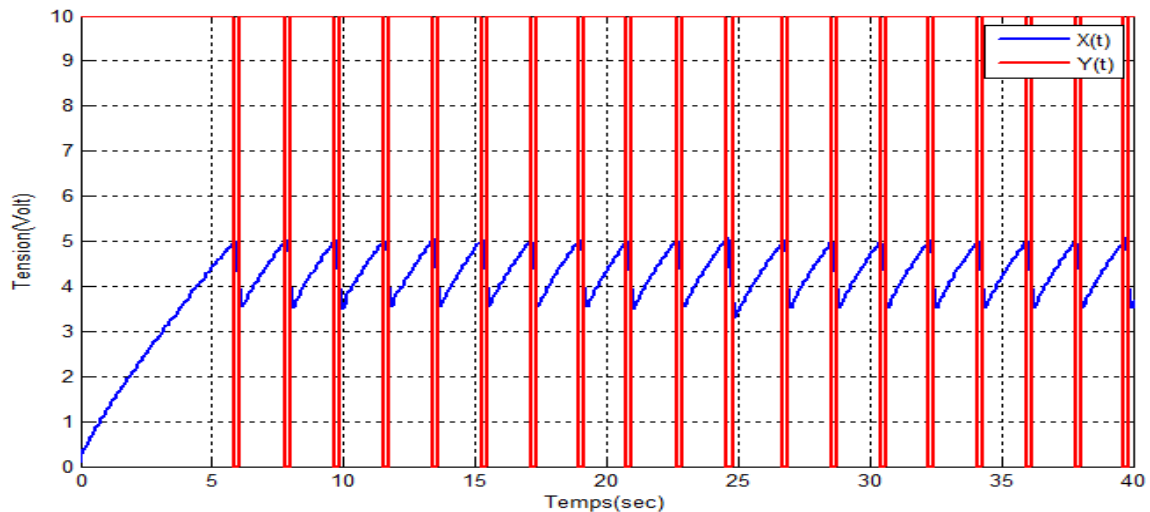


Figure 5.24 : Commande Tout ou Rien ($N_{max}=5$ et $N_{min}=4$).

Remarque

Selon les deux figures, on peut dire que la régulation TOR est intéressante pour des plages de variation importante. Sinon on risque de bousiller l'actionneur (Vanne proportionnelle), car la commande est sollicitée à chaque fois.

B. Résultats expérimentaux PID

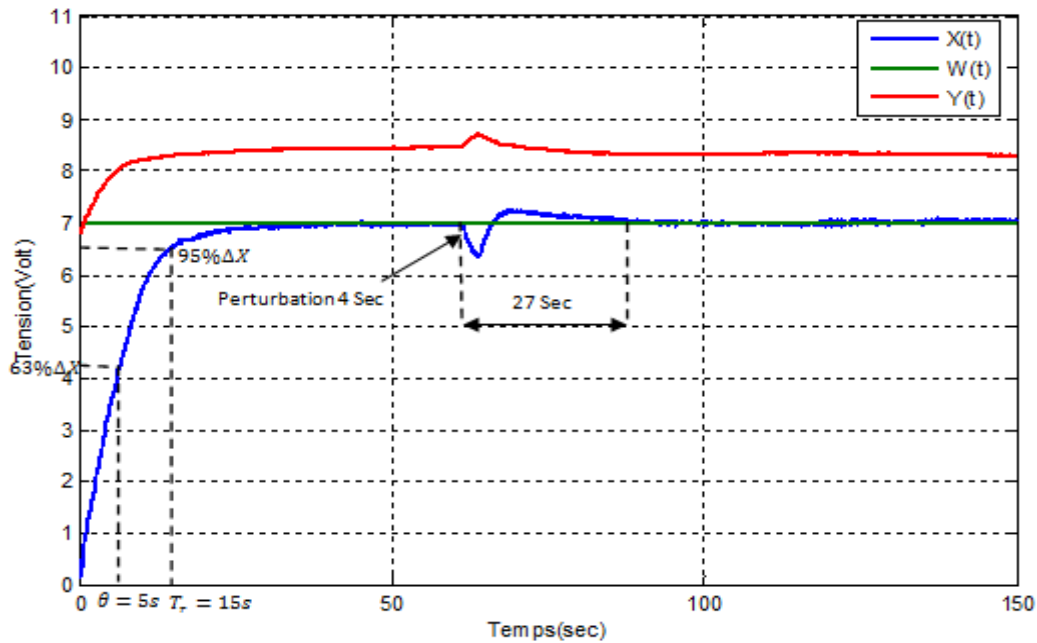


Figure 5.25 : Réponse du système pour une perturbation de 4 Sec

Lorsqu'on atteint le régime permanent, on applique une petite perturbation de 4 Sec à l'instant 60Sec (rejeter après 27 Sec). On remarque que la commande augmente pour corriger la perturbation et garder le même état initial.

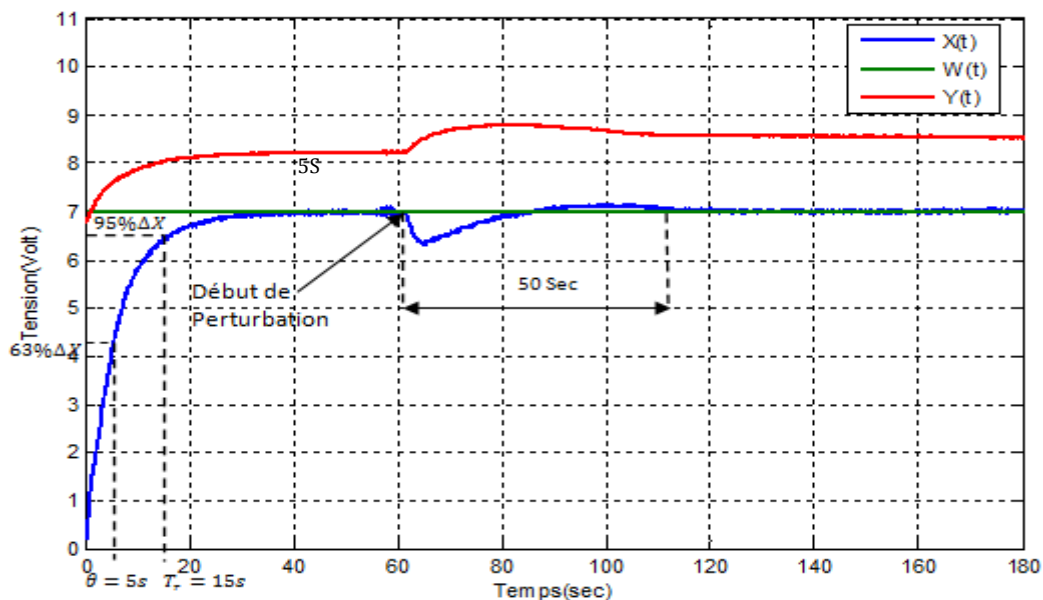


Figure 5.26 : Réponse du système pour une perturbation permanente

À l'instant 60 Sec, on applique une perturbation permanente.

On remarque que le régulateur PI a compensé cette perturbation après 50 Sec.

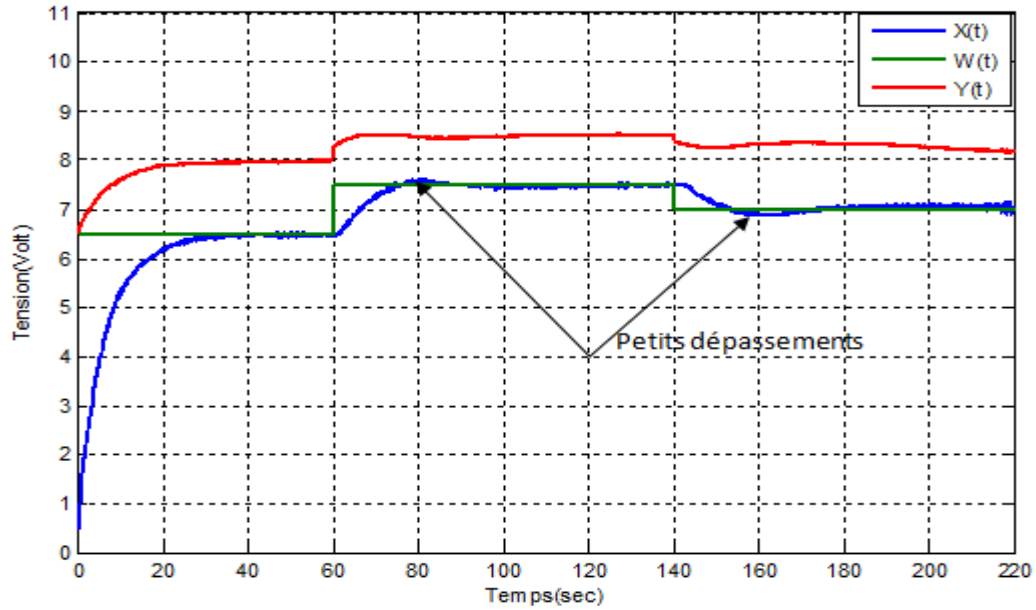


Figure 5.27 : Réalisation d'une poursuite

Au début la consigne est égale à 6.5 Volt et à l'instant 60 Sec on a augmenté la consigne à 7.5 Volt et à l'instant 140 Sec on l'a diminué à 7 Volt.

On remarque que la commande change au même temps, Pour ramener le signal de mesure à suivre la consigne.

Le signal de mesure présente des petits dépassements lors de changement de consigne.

C. Résultats expérimentaux (Régulateur flou)

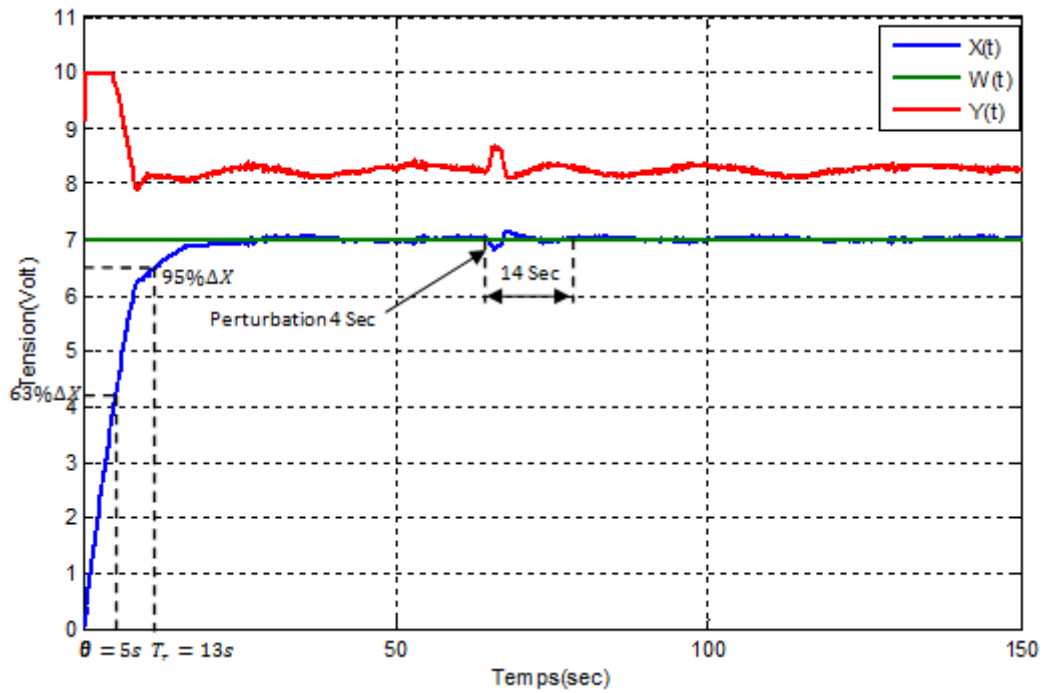


Figure 5.28 : Réponse du système après une perturbation de 4 Sec

A l'instant $t=60\text{Sec}$ on a appliqué une perturbation de 4Sec, le régulateur flou l'a rejeté après 14Sec.

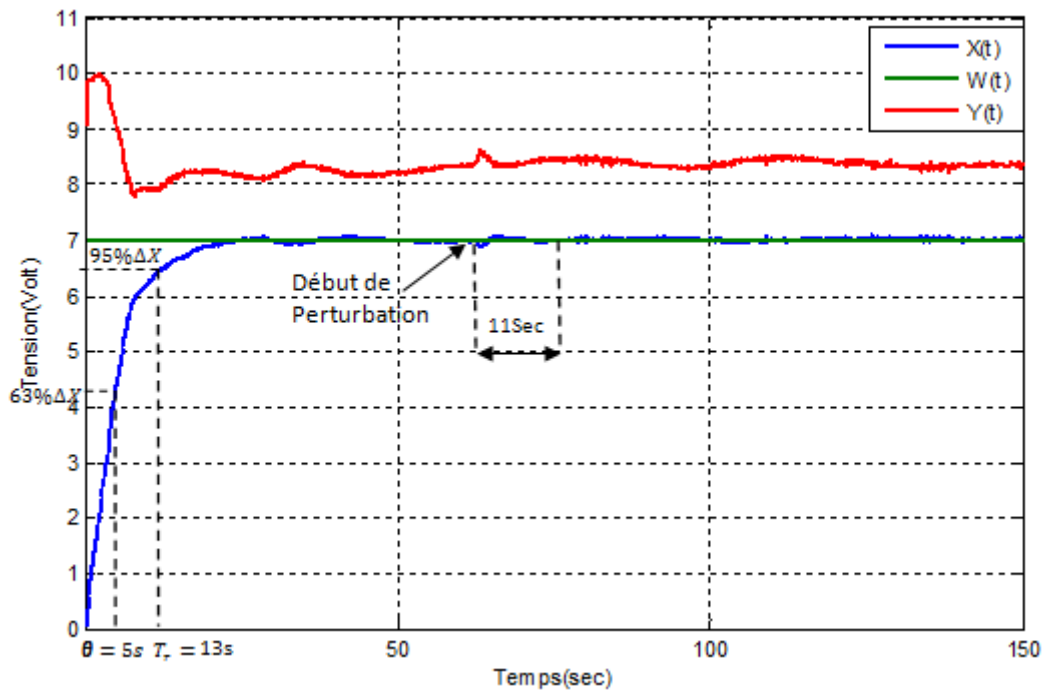


Figure 5.29 : Réponse du système après une perturbation permanente

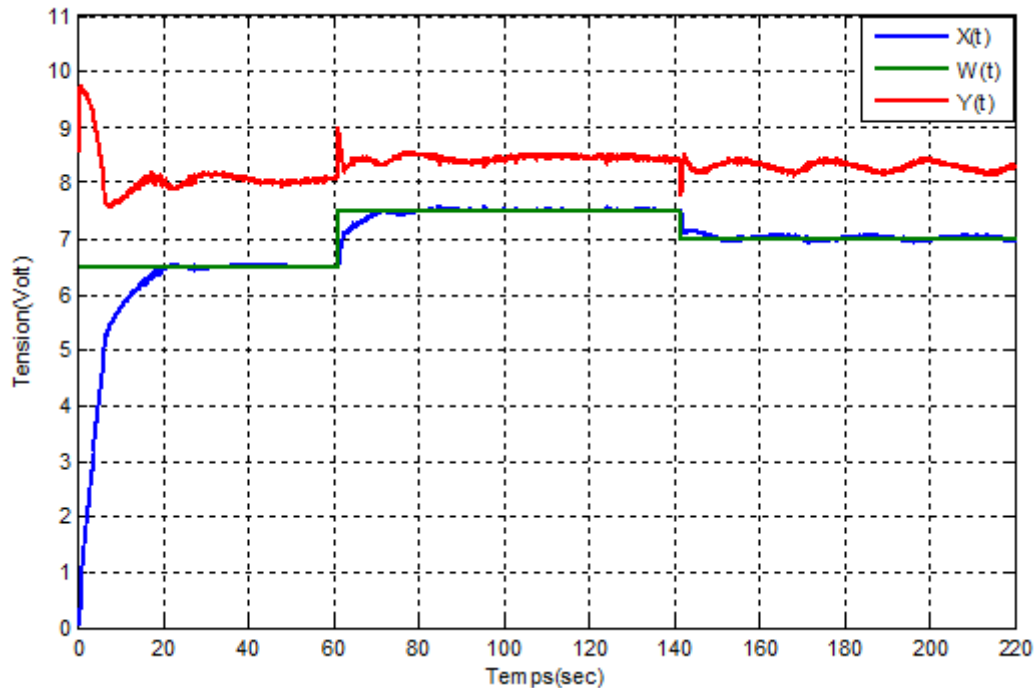


Figure 5.30 : Réalisation d'une poursuite

Au début la consigne est égale à 6.5 Volt et à l'instant 60 Sec on augmente la consigne à 7.5 Volt puis à l'instant 140 Sec on la diminue à 7 Volt. On note un bon suivi.

On remarque que le signal de mesure est sans dépassements lors de changement de consigne.

Si on fait une comparaison entre les deux commandes (PI et floue), on remarque que la commande floue corrige mieux les perturbations et en plus elle répond plus rapidement que la commande PI. La remarque la plus importante, la commande par la logique floue ne présente pas de dépassement lors de changement de consigne.

5.8. Conclusion

Nous avons présenté les différentes simulations ainsi les différents résultats expérimentaux obtenus à travers une carte d'acquisition Lbjack sous environnement Lbview, en utilisant deux techniques de commande l'une est PI et l'autre la commande floue.

Selon les résultats obtenus théoriquement et pratiquement on peut dire que la commande par la logique floue donne des résultats plus satisfaisants que le PI concernant le rejet de perturbation et lors de changement de consigne que ne présente pas de dépassement.

Conclusion Générale

La commande floue est une méthode intéressante et occupe aujourd'hui une place importante dans la commande des procédés industriels. Elle utilise des techniques s'appuyant sur le savoir-faire humain, exprimées en mots et en phrases ordinaires, plutôt que sur des équations.

Nous nous sommes intéressés dans notre travail à l'étude et développement d'une plateforme de commande pour la station de pression PUP-4 d'Electronica Veneta. Notre étude est divisée en trois parties : une centrée sur l'identification de notre station de pression, ensuite nous avons appliqué trois techniques de commande (TOR, PID et Flou). Une étude comparative entre la commande Floue et la commande classique PID a été discuté.

Les résultats obtenus nous ont permis de confirmer l'avantage de la commande floue par rapport à la commande classique PID, notamment dans le rejet des perturbations et lorsqu'on effectue des changements de consigne.

Ce travail a été bénéfique pour nous car nous a permis d'exploiter les connaissances déjà acquises durant notre formation et de les concrétiser pratiquement.

Notre plate forme est ouverte, ce qui conduit à la possibilité d'implémenter d'autres techniques de commande telles que : Réseaux de neurones, neuroflou, mode glissant,..etc.

nous espérons que nos efforts puissent servir à quelque chose et que ce mémoire soit un bon guide pour les promotions futures.

Références Bibliographiques

- [1] **PROUVOST. P**, « *Automatique contrôle et régulation* ». Edition Dunod, 2004.
- [2] **PROUVOST.P**, « *Instrumentation et Régulation en 30 fiches* ».Edition Dunod, Paris, 2010.
- [3]**PROVOUST. P**, « *Contrôle Régulation, Exercices et problèmes résolus* », Editions Nathan, Paris, 1997.
- [4] **RIVOIRE.M / FERRIER.J.L** « *Commande par ordinateur Identification* », Editions Eyrolles, Paris, 1997.
- [5] **AUCLERC.M/RENE.P** « *Régulation et automatisme des systèmes frigorifiques*», Edition Dunod, Paris2010.
- [6] **BUHLER.H**, « *Réglage par logique floue* » presses polytechnique et universitaire romandes, 1994.
- [7] **NADIA.M/MOHAND.M**, « *Apprendre et maîtriser LabVIEW par ses applications* », Springer-Verlag Berlin Heidelberg 2014
- [8] **NATIONAL INSTRUMENTS**, «*Principes de base de LabView* », Edition de Août 2006.
- [9] **CHAUVASEAU C**, « *Amplificateur Opérationnel- Cours et Problèmes*», version du 11 février 2013.
- [10] **FOUAD.G/MAREK.Z/TAIEB.B**,« *Système asservis : commande et régulation*», Edition Eyrolles, 1993

Logiciels Utilisés

MATLAB 2010.

LabVIEW 2013.


Plate forme de commande

1. Face avant (information)

INFORMATION CONTROLE IDENTIFICATION Université Mouloud Mammeri de Tizi-Ouzou 23/06/2015 13:39:18

Université Mouloud MAMMERRI de TIZI OUZOU
Faculté de génie électrique et informatique
Département d automatique

FI00 : Configurée en entrée, utilisée pour faire l'acquisition.
FI01: Configurée en sortie, utilisée pour envoyer la perturbation (sortie digital).
DAC0 : Sortie analogique, utilisée pour récupérer le signal de commande.



erreur labjack

String FI00

String 2 FI01

String 3 DAC0

error out code
no error 0
source

error out 2 code
no error 0
source

error out 3 code
no error 0
source

2. Face avant (identification)

INFORMATION CONTROLE IDENTIFICATION Université Mouloud Mammeri de Tizi-Ouzou 23/06/2015 01:26:28

Chemin

commande

dt aquis

STOP

FILTRE

Fe

Fb type de filtre Lowpass

Fh ordre

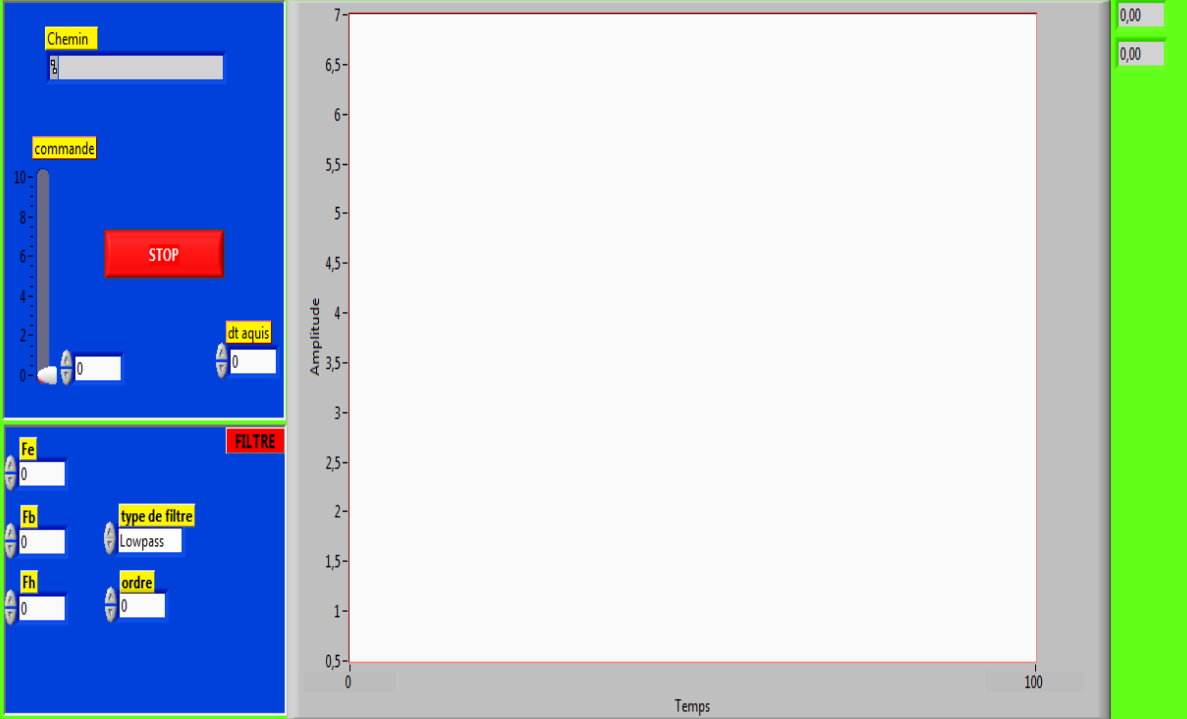
Amplitude

Temps

Tracé 0

0,00

0,00



3. Face avant (contrôle)

INFORMATION
CONTROLE
IDENTIFICATION

Université Mouloud Mammeri de Tizi-Ouzou
23/06/2015 13:39:18

CONSIGNE

SP 10

choix de consigne
manuelle

SP 0

perturbation

COMMANDE

commande

0

0 1 2 3 4 5 6 7 8 9 10

ALARME

limite sup 0

limite inf 0

niveau haut

niveau bas

REGULATEUR

choix de regulateur
flou

paramètres PID

KC 0

TI 0

TD 0

Nmax 0

Mmin 0

dt 0

dt acqui 0

FILTRE

Fe 0

Fb 0

Fh 0

ordr 0

type de filtre
Lowpass

chemin

prog flou

STOP

SP PV COMM

0,00 SP

0,00 PV

0,00 COMM

Tracé 0 █

Tracé 0 █

Tracé 0 █

Amplitude

SP et PV

Temps

Amplitude

COMMANDE

Temps

Amplitude

PV

Temps

Amplitude

SP

Temps

Diagramme (interface contrôle)

