République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de **L**'enseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITÉ **M**OULOUD **M**AMMERI **D**E **T**IZI-**O**UZOU

FACULTÉ **D**U **G**ÉNIE **É**LECTRIQUE **E**T **D'I**NFORMATIQUE
DÉPARTEMENT D'AUTOMATIQUE

Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Science et Technologie**
Filière : **Automatique**
Spécialité : **Automatique et Informatique Industrielle**

Présenté par
Amokrane ALLACHE

Thème

Commande Neuro-floue d'un aérotherme à base de la carte Arduino.

Mémoire soutenu publiquement le 25/06/2024 devant le jury composé de :

M Mourad ALLAD
MCB, UMMTO, Président

M Rabah MELLAH
Professeur, UMMTO, Encadrant

Mme Karima AMOURA
MCB, UMMTO, Examinateur

Mme Karima HOUACINE
MCB, UMMTO, Examinateur

Ce travail a été réalisé au Laboratoire de Conception et Conduite des Systèmes de Production (L2CSP)

République **A**lgérienne **D**émocratique et **P**opulaire
Ministère de **L**'enseignement **S**upérieur et de la **R**echerche **S**cientifique

UNIVERSITÉ **M**OULOUD **M**AMMERI **D**E **T**IZI-**O**UZOU

FACULTÉ **D**U **G**ÉNIE **É**LECTRIQUE **E**T **D**'**I**NFORMATIQUE
DÉPARTEMENT D'AUTOMATIQUE

Mémoire de Fin d'Etudes de MASTER PROFESSIONNEL

Domaine : **Science et Technologie**
Filière : **Automatique**
Spécialité : **Automatique Industrielle**

Présenté par
Yasmina BENSAI

Thème

Commande Neuro-floue d'un aérotherme à base de la carte Arduino.

Mémoire soutenu publiquement le 25/06/2024 devant le jury composé de :

M Mourad ALLAD
MCB, UMMTO, Président

M Rabah MELLAH
Professeur, UMMTO, Encadrant

Mme Karima AMOURA
MCB, UMMTO, Examinateur

Mme Karima HOUACINE
MCB, UMMTO, Examinateur

Ce travail a été réalisé au Laboratoire de Conception et Conduite des Systèmes de Production (L2CSP)

Remerciements

Avant tous, on remercie DIEU le tout puissant de nous avoir donné la santé, le courage et la patience durant toutes ces années d'études et grâce à qui ce travail a pu être réalisé.

On tiens, tout d'abord, à exprimer notre gratitude, notre plus grande reconnaissance et nos profond respect à notre promoteur, Monsieur Rabah MELLAH, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, qui nous a apporté son savoir scientifique indéniable, sa confiance, ses judicieux conseils et sa disponibilité, tout au long de l'élaboration de ce travail. Qu'il trouve ici l'expression de notre profonde reconnaissance.

On tiens particulièrement à remercier Monsieur Ahcene TRIKI, Maître de conférences à l'université Mouloud MAMMERI de Tizi-Ouzou, pour sa disponibilité, ses conseils judicieux, et son aide précieuse pour l'élaboration de notre travail expérimental. Nous lui en sommes très reconnaissant.

Nos remerciements s'adressent à tous les membres du Laboratoire de Conception et Conduite des Systèmes de Production (L2CSP) (UMMTO), pour leur sympathie et l'excellente ambiance de travail qu'ils ont crées.

Nous Remercions aussi Monsieur Ahmed MAIDI, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, pour son aide et ses précieux conseils.

Nos remerciements vont s'adressé également au président et aux membres du jury qui nous font l'honneur d'évaluer notre travail.

Nos reconnaissances vont également s'adressé à tous les enseignants qui ont contribué à notre formation tout au long de notre cursus scolaire et universitaire.

Nos remerciements les plus chaleureux vont à Nos chers parent et nos familles respectives pour leurs encouragements, leur patience, et leur grand soutien durant toutes ces années d'études.

Nous Remercions tous ceux qui ont contribué, de près ou de loin, à la réalisation de ce travail de fin d'étude.

Dédicaces

Du plus profond de mon cœur, je dédie ce travail à tous ceux qui me sont chers,

A ma chère mère,
Quoi que je fasse ou que je dise, je ne saurai point te remercier comme il se doit. Ton affection me couvre, ta bienveillance me guide et ta présence à mes côtés a toujours été ma source de force pour affronter les différents obstacles.

A mon cher père,
Ma ressource inépuisable de force, mon guide dans la vie et le meilleur ami que l'on pourrait avoir. Merci de m'avoir appris tout ce que je sais et d'avoir fait de moi ce que je suis.

A mes chères sœurs,
Rakelle, Mélissa, Thafath, pour leur soutien moral et leurs conseils tout au long de mes études.

A mon adorable petite sœur Thiziri,
Qui sait toujours comment procurer la joie et le bonheur pour toute la famille.

A ma grand-mère,
A qui je souhaite une bonne santé.

A mon chéri, ami et binôme Amokrane et sa famille,
Qui a toujours été là pour moi, à me soutenir et à m'encourager. Et pour sa patience et sa compréhension tout au long de ce projet. Merci.

A mes chères amies,
Pour leurs aides et supports dans les moments difficiles

A toute la promotion Automatique Industrielle 2024.

A toute ma famille.

Yassmina

A mon cher père,
Ma source de vie, bonheur et d'amour. A celui qui n'a jamais cessé, de formuler des prières à mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs.

A ma chère mère,
Ma source de bonheur et d'affection, et ma raison de vivre. A celle, qui a toujours été là pour moi, qui n'a cessé de croire en moi, qui m'a soutenu et encouragé durant toutes ces années.

« A vous mes parents, les êtres les plus chères à mes yeux, je vous remercie d'avoir fait de moi celui qui je suis aujourd'hui, aucun mot ni phrase ne pourra exprimer mes respects, mes considérations et ma grande admiration pour vous. Puisse ce travail vous témoigne mon affection et mon profond amour. Merci pour tout et que dieu vous garde pour moi »

A mon cher petit frère Youcef,
Mon bras droit, qui a toujours été là à mes côtés dans les moments difficiles, qu'il trouve ici mon affection et mon profond amour pour lui.

A tous mes grands frères, grandes sœurs et leurs enfants,
Pour leurs soutiens et leur encouragement, qu'ils trouvent ici mes respects pour eux, et que dieu leurs donne une vie joyeuse.

A ma chérie, amie et binôme Yasmina et sa famille,
Qui a toujours été là pour moi, à me soutenir et à m'encourager. Et pour sa patience et sa compréhension tout au long de ce projet. Merci.

A mes grands-mères, mes oncles, mes tantes et leurs enfants,
Que dieu leur donne une longue et joyeuse vie.

A tous mes cousins et cousines,
Que dieu vous protège et vous offre la chance et le bonheur.

A tous mes amis,
Pour leur soutiens et aide dans les moments difficiles.

A toute la promotion Automatique et informatique industrielle 2024.

A toute ma famille,

Je dédie ce modeste travail.

Amokrane

Table des matières

Table des figures

Liste des tableaux

Liste des symboles

Symbole	Désignation
V	Volt, est l'unité de mesure de la tension électrique dans le Système international d'unités (SI).
W	Watt, est l'unité de mesure de la puissance dans le Système international d'unités (SI).
cm	Centimètre, st une unité de mesure de longueur dans le système métrique.
m	Mètre, est l'unité de base de longueur dans le Système international d'unités (SI).
C	Celsius, est une unité de mesure de la température dans le système métrique.
t	Temps, est une grandeur mesurable qui permet de quantifier la durée des événements et les intervalles entre eux.
mV	Millivolt, est l'unité de mesure de la tension électrique.
K1	Gain d'entrée du contrôleur ANFIS.
K2	Gain d'entrée du contrôleur ANFIS.
K3	Gain de sortie du contrôleur ANFIS.

Notation

Acronymes

SISO	Single Input, Single Output.
GND	Ground.
OUT	Output.
Adj	adjustment.
DEL	Une diode électroluminescente
AC	Alternating Current.
AVR	Automatic Voltage Regulator.
MHz	Mega Hertz.
KB	Kilo Byte.
Ko	Kilo octet.
SRAM	Static Random Access Memory.
EEPROM	Electrically Erasable Programmable Read Only Memory.
RISC	Reduced Instruction Set Computing.
PWM	Pulse Width Modulation.
I2C	Inter-Integrated Circuit.
SCL	Serial Clock Line.
SDA	Serial Data Line.
UART	Universal Asynchronous Receiver Transmitter.
RX	Receive Data.
TX	Transmit Data.
SPI	Synchronous Peripheral Interface.
MOSI	Master Out Slave In.
MISO	Master In Slave Out.
SS	Slave Select.
SCLK	serial clock.
USB	Universal Serial Bus.
IDE	Integrated Development Environment.
PID	Proportionnel, Intégral, Dérivé.
ANFIS	Adaptive Neuro-Fuzzy Inference System.
RNA	Réseau de Neurones Artificiels.
LF	Logique Floue.
SIF	Système d'inférence flou.
NF	Neuro-flou.
PC	Personal Computer.

Introduction générale

L'aérotherme, également appelé chauffage à air pulsé, est un dispositif de chauffage largement utilisé dans divers secteurs industriels et commerciaux. Il fonctionne en chauffant l'air ambiant à l'aide d'une résistance électrique, puis cet air chauffé est distribué à travers un système de ventilation. Depuis son invention au début du XXe siècle par Henri Arquembourg [1], sa progression est liée à l'évolution des technologies de chauffage, elle passe de premiers modèles rudimentaires à combustion, aux systèmes électriques modernes à haute efficacité énergétique, en passant par l'utilisation de pompes à chaleur. Ce processus permet de chauffer de grands volumes d'air rapidement et uniformément, offrant ainsi un contrôle précis de la température ambiante. Dans l'industrie, ils jouent un rôle crucial dans de nombreux processus de fabrication, de séchage et de conditionnement, pour assurer la qualité des produits et la productivité des opérations. Par exemple, dans le domaine agroalimentaire, les aérothermes sont utilisés pour le séchage de produits tels que les céréales, les fruits et les légumes. Dans les entrepôts logistiques, ils assurent le maintien de températures adéquates pour le stockage de marchandises sensibles aux variations thermiques. En outre, ils sont largement utilisés dans des usages commerciaux tel que les gymnases et les ateliers, ou personnelles comme les habitations.

La commande de l'aérotherme a été au centre de nombreuses recherches visant à optimiser ses performances et à offrir un contrôle précis de la température. A cet effet, les contrôleurs doivent être trés performants car les aérothermes présentent des systèmes complexe avec une dynamique thérmique non linéaires. De plus, dans certaines applications industrielles et commerciales, leurs interactions avec d'autres systèmes tels que la ventilation, la climatisation et les systèmes de contrôle d'accès, peut introduire des couplages et des contraintes supplémentaires. Pour cela, les régulateurs appropriés à l'aérorherme

doivent être robustes et adaptatifs pour faire face à ces contraintes. Ces dernières décennies, l'utilisation des techniques modernes tel que l'intelligence artificielle (logique floue, réseaux de neurones, ...) pour le contrôle des systèmes a suscité un intérêt croissant et a fait objet de plusieurs travaux de recherche [2][3]. D'une part, la logique floue permet d'avoir de bonnes performances grâce à son efficacité de gérer les incertitudes présentes dans les systèmes non linéaires et à sa capacité d'offrir une modélisation intuitive de ces derniers. Par contre, la génération des fonctions d'appartenances pour les systèmes flous est un problème complexe et leurs performances dépendent fortement des ces fonctions [2]. D'autre part, les réseaux de neurones sont également trés utilisé dans le contrôle des systèmes grâce à leurs traitement massif en parallèle, leur capacité d'auto-apprentissage et leur structure flexible [4]. Néanmoins, leurs conception et leurs configuration peuvent être complexes, surtout pour les réseaux profonds avec de nombreux neurones et couches. Par conséquent, dans ce présent travail, nous allons nous intéresser à des systèmes neuro-flou, qui sont des systèmes flous entraîné par un algorithme d'apprentissage inspiré des réseaux de neurones. Ce choix est motivé par le fait que la combinaison entre la logique floue et les réseaux de neurones offre des contrôleurs plus performants, et cela en tirant profit de ces deux approches.

L'objectif de ce projet de fin d'étude est d'implémenter une strategie de commande qui combine la logique floue et les réseaux de neurones de telle maniére à synthétiser un régulateur neuro-flou, en utilisant le logiciel MATLAB-Simulink et Arduino Support Package, destiné à piloter un aérotherme réalisé au laboratoire de conception et de conduite des systèmes de production (L2CSP), afin d'assurer la poursuite des profils de température. A cet effet notre travail est organisé comme suit :

Dans le premier chapitre, nous avons présenté notre aérotherme et sa boucle de commande avec son principe de fonctionnement. Par la suite, on a fait une déscription détaillée de chaque organe du procédé. Ensuite, nous avons présenté les outils de l'environnement MATLAB utilisés pour l'identification du système afin de synthétiser un contrôleur PID, à savoir les outils "System Identification Toolbox" pour l'identification, et "PID Tuner" pour la synthèse du contrôleur PID.

Le deuxième chapitre décrit les différentes techniques de l'intelligence artificielle à savoir : la logique floue et les réseaux de neurones, les structures neuro-floues. Ensuite, nous nous concentrons en particulier sur la projection des systèmes flous dans les réseaux de neurones afin de concevoir un contrôleur neuro-flou de type ANFIS.

Le troisième chapitre est consacré à l'implémentation des contrôleurs proposés sur logiciel MATLAB, via l'environnement Simulink. La suite du chapitre présente les résultats expérimentaux obtenus en utilisant le contrôleur classique PID et le contrôleur adaptatif ANFIS.

Enfin, nous clôturons notre travail par une conclusion générale et quelques perspectives.

Chapitre 1

Description de l'aérotherme et de la boucle de commande

1.1 Introduction

Un aérotherme est un procédé thermique servant à chauffer un espace spécifique, en utilisant des échanges thermiques efficaces pour assurer une température uniforme. Le procédé sur lequel nous allons travailler est réalisé au laboratoire de recherche. Dans ce chapitre, nous allons commencer par la description du principe de fonctionnement du procédé thermique, ensuite nous allons définir les differents organes qui le constituent. Et on finit par une conclusion.

1.2 Principe de fonctionnement du procédé

Le procédé thermique, aérotherme, à concevoir doit se rapprocher au maximum du fonctionnement d'un procédé thermique réel, tout en se distinguant par une simplicité et un temps de réponse faible.

À ce titre, notre aérotherme présente plusieurs avantages, à savoir : le fluide utilisé (l'air ambiant) qui est facile à manipuler et doté d'une disponibilité permanente [5]. Et l'énergie utilisée est de forme électrique qui est une source d'énergie garentie et fiable, et qui est considerée comme une énergie propre, contrairement au gaz ou fioul.

Le système à commander est un système de classe monovariable (SISO), tel que la puissance électrique de la résistance de chauffage et la température d'un courant d'air sont respectivement l'entrée et la sortie du sytème. Un séche cheveux génère un débit d'air chaud maintenu constant canalisé dans un tube. À l'extrimité de ce dernier on place un capteur de température qui va mesurer la température du flux d'air. cette derniére est acquise à travers une carte Arduino pour servir d'information au régulateur neuro-flou conçu et implémenté sur le logiciel MATLAB-Simulink. Une commande faite par le régulateur est envoyée sous forme d'un signal électrique par la carte arduino pour commander l'amplificateur de puissance, qui va commander à son tour la résistance chauffante du sèche cheveux en vue de faire varier la température de l'air souflé, comme le montre la figure 1.1.

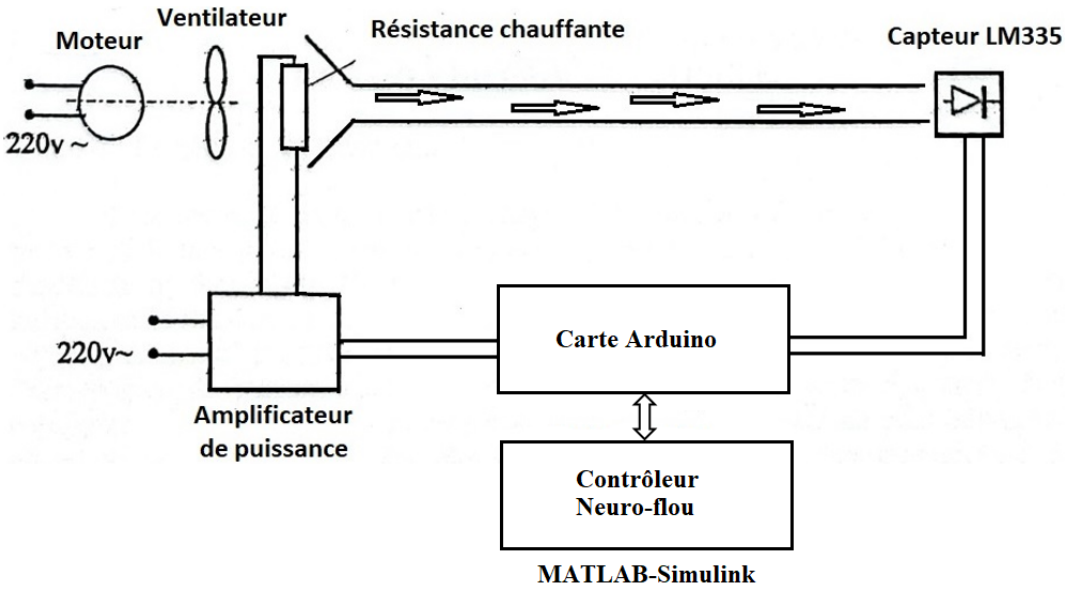


FIGURE 1.1 – Shéma de principe de la boucle de commande de l'aérotherme

Ainsi, la figure 1.2 montre le shéma fonctionel de base du système à commander qu'on a deduit de la figure 1.1.

où :

- Yd est le signal de consigne (Température désirée).
- $e(t)$ est le signal de l'erreur.
- $U(t)$ est le signal de commande.
- $T(°C)$ est la grandeur de commande (Température).
- $Y(t)$ est le signal de retour.

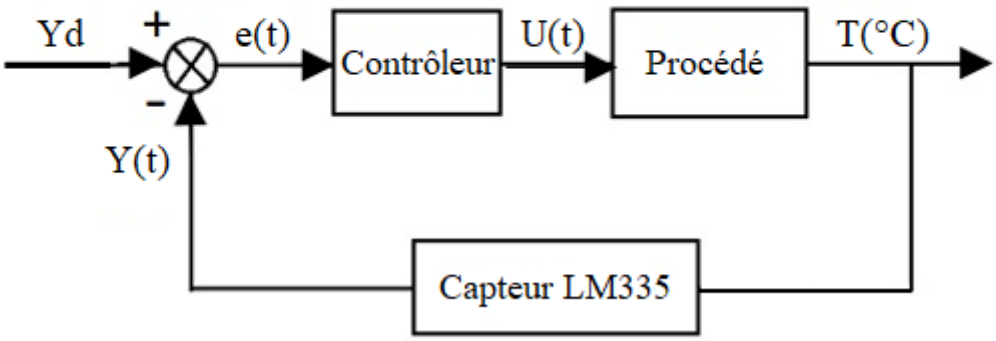


FIGURE 1.2 – Shéma fonctionel du procédé bouclé

1.3 Description des differents organes du procédé bouclé

1.3.1 Générateur d'air chaud

Il s'agit d'un séchoir à cheveux doté d'un moteur à courant continu, qui traine un ventilateur pour générer un débit d'air, le tout couplé à une résistance chauffante de 500W pour réchauffer ce dernier. Un tube cylindrique en plastique de 7,5 cm de diamètre et de 1m de longueur a été ajouté à la suite du séchoir pour simuler une conduite.

Dans sa version de base, le séche-cheveux peut fonctionner avec deux vitesses et puissances de chauffage différentes, qu'on peut séléctionner à l'aide d'un redresseur à diode commutable. Mais pour les besoins de notre étude, à savoir générer un débit d'air constant chauffé à des températures variables, l'alimentation de la résistance chauffante a été rendu indépendante de celle du moteur.

1.3.2 Capteur de température

Les capteurs sont des organes de prélèvement d'informations [6]. Ils élaborent, à partir d'une grandeur physique, une autre grandeur physique de nature différente. Dans notre

cas, elles sont respectivement la température et la tension électrique (Figure 1.3).

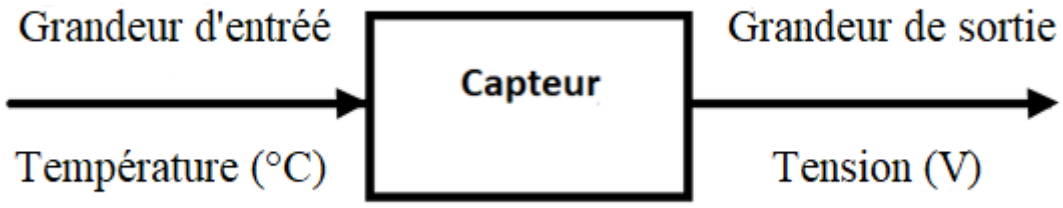


FIGURE 1.3 – Entrée et sortie du capteur LM335

où :

- la grandeur d'entrée est appelée la mesurande.
- la grandeur de sortie est appelée la réponse du capteur.

Ils jouent un rôle important dans la théorie du contrôle en fournissant des informations en temps réel sur l'évolution des systèmes à contrôler. Afin de concevoir notre régulateur, l'utilisation d'un capteur de température est indisponsable pour la mesure du signal de sortie.

Pour notre procédé, nous avons opté pour le capteur de température LM335. Ce dernier est un capteur de température linéaire analogique fabriqué par Texas Instruments. Il comporte une diode semi-conductrice qui produit une tension proportionnelle à la température en degrés Kelvin à raison de 10mV par degré, soit 0.01V/°K. Le LM335 peut mesurer des températures dans une large plage allant généralement de -40°C à 100°C [7]. Il est aussi réputé pour sa stabilité à long terme, ce qui le rend approprié pour des applications où une mesure précise de la température est requise pendant des périodes prolongées sans nécessiter de recalibration fréquente. Ce qui est le cas de notre procédé.

Branchement

Le capteur est fixé sur le bout du tube cylindrique en plastique. Son branchement est relativement simple, la figure 1.4 montre son branchement sur une carte à microcontrôleur à savoir l'Arduino.

Le LM335 se présente sous la forme d'un boitier TO92 à trois pattes. La patte 1 est nomée "Gnd", elle est reliée directement à la masse. La patte 2 est identifiée "Out", c'est la sortie du capteur, elle est connectée à une entrée analogique de la carte arduino pour l'acquisition des mesures. Elle est aussi branchée, avec une résistance en série, sur les 5V de la carte. Enfin, la patte 3 est désignée par "Adj", ce qui signfie ajustement, elle est utilisée pour calibrer le capteur afin d'obtenir des lectures de température plus précises.

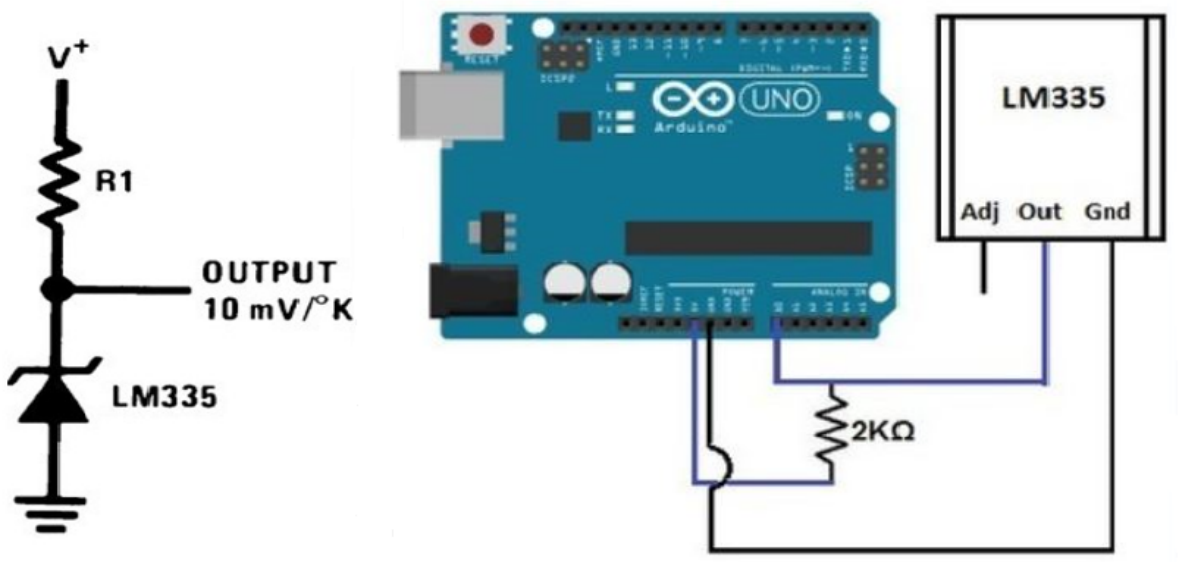


FIGURE 1.4 – Branchement du capteur LM335 sur la carte arduino

1.3.3 Amplificateur de puissance

Il forme l'interface entre la carte de commande et l'élément chauffant. Il reçoit de la carte arduino le signal de commande élaborer sous forme d'une tension qui varie entre 0 et 5 volts et est chargé de délivrer la puissance voulue à la résistance de chauffage. Ce dispositif est basé sur un optotriac. Ce dernier, est un dispositif semi-conducteur utilisé pour isoler électriquement un circuit de commande basse tension d'un circuit de puissance haute tension (figure 1.5).

L'optotriac utilisé est un optotriac MOC3041, est composé de deux éléments principaux à savoir :

- Une diode émettrice de lumière (DEL) : Cette diode est une DEL infrarouge (IR) qui convertit le signal électrique en lumière infrarouge.

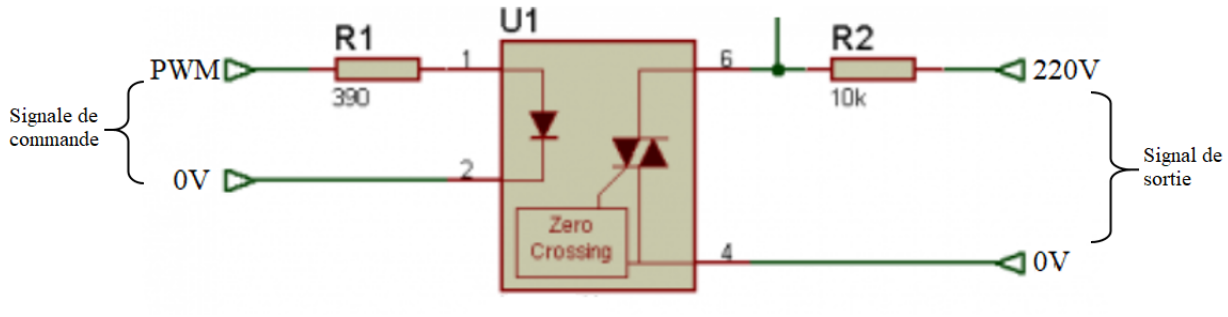


FIGURE 1.5 – Shéma électrique sur Proteus de l'amplificateur de puissance.

- Un triac de puissance : Le triac est un dispositif de commutation bidirectionnel utilisé pour contrôler le courant alternatif (AC). Il peut être activé par une impulsion de courant provenant de la DEL.

Fonctionnement du l'optotriac MOC3041

Son fonctionnement repose sur le principe de l'isolation optoélectronique : lorsque le courant est appliqué à la diode émettrice de lumière (DEL), celle-ci émet de la lumière infrarouge. Cette dernière, traverse le boîtier de l'optotriac et atteint le phototriac intégré et l'active, ce qui permet à un courant de gâchette de circuler à travers le triac de puissance. À son tour, ce dernier, conduit le courant alternatif à travers lui-même jusqu'à ce que le courant soit inversé ou qu'il soit coupé.

1.3.4 Carte d'acquisition

La carte d'acquisition joue un rôle fondamental dans la chaine de commande. Sur le plan fonctionel, elle assure deux rôles principaux : D'une part, une carte d'acquisition. Elle récupère les températures mesurées par le LM335. D'autre part, un émetteur de commande. Elle permet d'envoyer le signal de commande, générer sur simulink par le régulateur, afin de commander la resistance chauffante. Pour l'aérotherme conçu, on a choisit la carte Arduino Mega2560 comme carte d'acquisition. (figure 1.6).

L'arduino est une plate-forme d'objets interactifs à usage créatif constituée de deux parties : Hardware, qui est une carte électronique. Et software, qui est un environnement de programmation "IDE ARDUINO" [8].

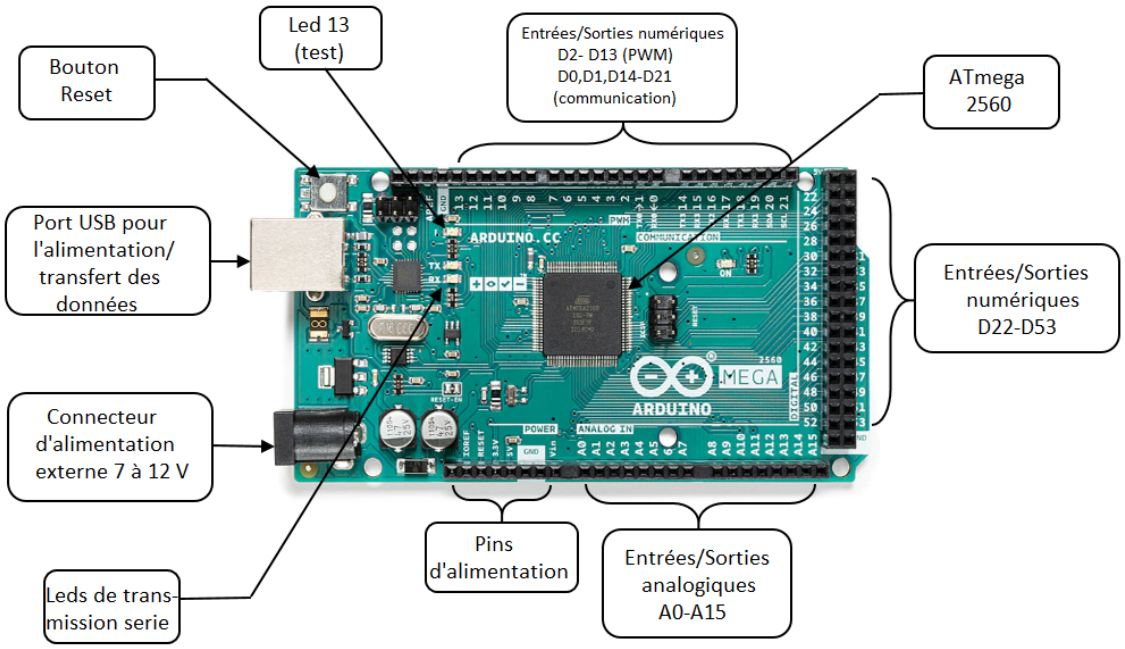


FIGURE 1.6 – Architecture de la carte Arduino Mega 2560

Partie hardware

La carte électronique Arduino Mega2560 est l'une des cartes les plus populaires de la famille Arduino, connue pour sa puissance et sa flexibilité. Sa grande capacité de mémoire, son nombre élevé de broches d'entrée/sortie et sa compatibilité avec une large gamme de shields, lui permettent d'être idéale dans de nombreux projets, notamment : La robotique, des projets d'apprentissage automatique, et surtout pour des procédés industriels comme dans notre cas.

Elle est constituée de :

1. **Microcontrôleur**

   Le microcontrôleur est de type ATmega2560, il appartient à la famille AVR, fabriqué par Microchip Technology, avec une architecture 8 bits RISC. C'est un microcontrôleur puissant, il est doté d'une fréquence d'horloge de 16 MHz et trois différentes mémoires : une Flash de 256 KB pour stocker le programme de l'utilisateur, une SRAM de 8 Ko pour les variables et les données temporaires, et enfin une mémoire EEPROM de 4 Ko pour stocker des données qui doivent être conservées même après une mise hors tension (figure 1.7).

FIGURE 1.7 – microcontrôleur ATmega2560

2. **Les broches d'entrées-sorties**

   La carte Arduino Mega2560 dispose d'un ensemble de broches configurables par programmation, qui fonctionnent comme des entrées ou des sorties, en utilisant les instructions PinMode(), et DigitalWrite(), DigitalRead(), analogRead(), analogWrite() du langage Arduino. Elles permettant de communiquer et interagir avec des composants externes.

   Elle possède 54 broches entrées-sorties numériques (D0-D53). Elles admettent et délivrent des signaux logiques (tension comprise entre 0 et 5V). Certaines broches ont des fonctions spécialisées reparties comme suit [9] [10] :

   - D2 - D13 : Elles fournissent une impulsion PWM à l'aide de l'instruction "analogWrite()".
   - D20 , D21 : Deux broches d'interface série I2C reparties comme suit : SDA->20, SCL->21.
   - DO, D1, D14 - D19 : 8 broches dont 4 entrées (Rx0,Rx1,Rx2,Rx3) et 4 sorties (Tx0,Tx1,Tx2,Tx3) pour former 4 ports de communication serie UART.
   - D50 - D53 : 4 broches entrées-sorties pour le bus série normalisé SPI (MISO ->50, MOSI->51, SCK->52, SS->53).

   Et elle dispose aussi de 16 broches entrées-sorties analogiques (A0-A16). Chacune pouvant fournir une mesure d'une résolution de 10 bits (1024 niveaux soit de 0 à 1023). Ces broches mesurent entre le 0V (valeur 0) et le 5V (valeur 1023).

3. **Alimentation**

   La carte Arduino peut-être alimentée via quatre entrées différentes :

   - La premiere, via le port USB qui fournit une source de tension de 5V directement au microcontrôleur sans passer par le régulateur de tension.
   - La deuxième, via la prise jack, une alimentation externe de 6V-20V, qui alimente le régulateur de tension à la carte. Pour protéger la carte et le régulateur des chutes de tension et de la surchauffe la plage idéale recommandée pour alimententation est entre 7V et 12V.
   - La troisième, via la pin Vin(+) et GND(-). Les recommandations d'alimentations sont les mêmes que pour la prise jack.
   - La quatrième, dirctement sur la pin 5V et GND. Si la source de tension utilisée est précise.

4. **Les leds**

   - LED d'alimentation. Elle est allumée quand la carte est mise sous tension.
   - LEDs de communication. Deux leds RX et Tx qui s'activent lors de transmission et la réception des données.
   - LED 13 appelée de test. Elle est connectée à la broche 13. La LED est allumée, lorsque la broche est au niveau haut (5v), et elle est éteinte lorsque la broche est au niveau bas (0v).

Partie software

Afin de communiquer avec MATLAB, les broches de la carte arduino doivent être spécifier dans un programme. A cet effet, ce dernier est conçu dans un logiciel appelé IDE Arduino.

L'IDE Arduino, ou environnement de développement intégré Arduino, est un logiciel open-source, écrit en Java inspiré du langage Processing. Il permet l'édition, en langage C, des programmes, leur compilation et leur téléversement dans la carte Arduino, il permet aussi de communiquer avec cette dernière grâce au moniteur série.

La figure (figure 1.8) montre les différentes parties de la fenêtre principale du logiciel

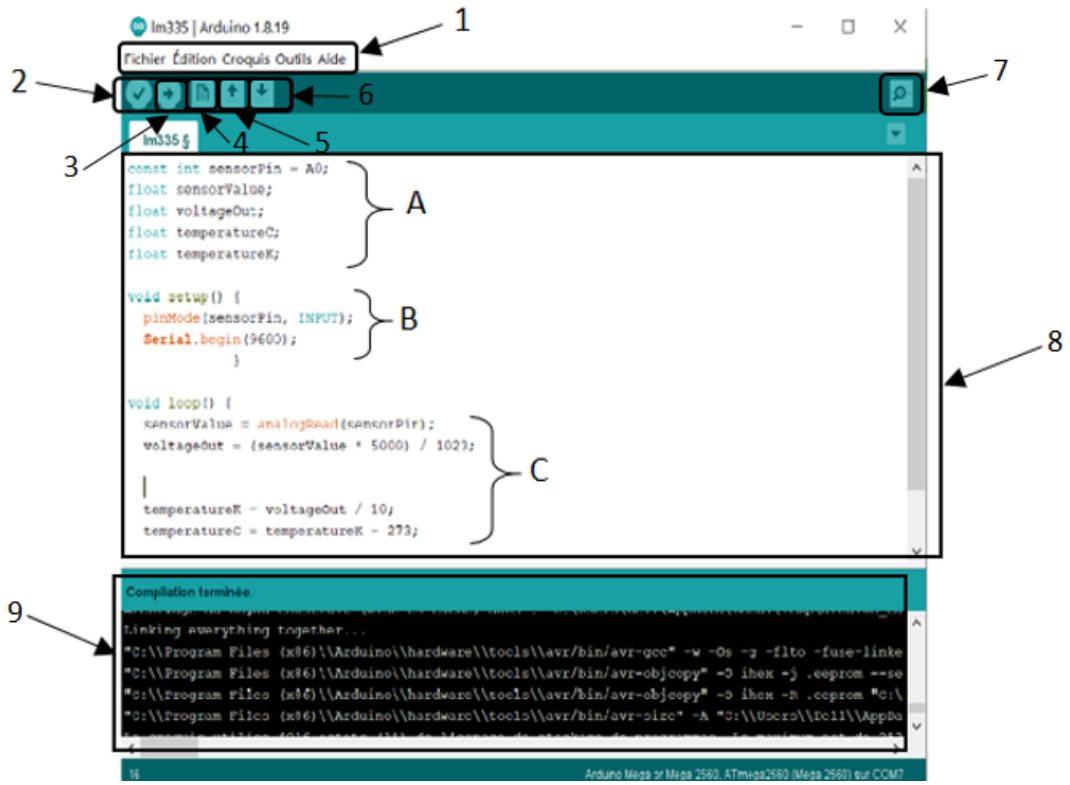


FIGURE 1.8 – Les différentes parties de la fenêtre principale du logiciel IDE Arduino.

IDE Arduino :

L'interface du logiciel se présente de la façon suivante :

1. Les options de configuration du logiciel.
2. Bouton de vérification et de compilation du programme.
3. Bouton téléversement du programme dans la carte.
4. Bouton pour l'ouverture d'une nouvelle fenêtre pour un nouveau programme.
5. Bouton pour l'ouverture d'un programme préalablement enregistré.
6. Bouton pour enregistré le programme courant.
7. Bouton pour ouvrir le moniteur série.
8. Zone de texte, pour la rédaction du programme.
   - A : Cette partie est dédiée à la déclaration des variables et des constantes.
   - B : Cette partie est consacrée à la fonction « Void Setup () » : Toutes les instructions comprise entre les deux accolades seront exécuter une seule fois au démarrage du programme.

- C : Cette partie est réservée à la boucle principale « Voidloop() » : Elle est exécutée en boucle après l'exécution de la fonction Void setup(). Elle contient généralement le code principal du programme qui doit être exécuté de manière répétée.

9. Console d'affichage des messages de compilation.

1.4 Identification du procédé

Dans la suite de notre travail, nous allons synthétisé un régulateur neuro-flou ANFIS qu'on va implémenter par la suite sur le logiciel MATLAB SIMULINK dans le but de contrôler notre sytème. Afin de démontrer son efficacité, nous souhaitons faire une comparaison avec un contrôleur classique (PID). A cet effet, la synthése de ce dernier nécessite d'abord l'identification du modéle mathématique approprié à l'aérotherme. Ce modèle servira de base pour ajuster les paramètres P, I et D. De plus, ce modèle sera aussi utile pour la simulation du contrôleur neuro-flou sur MATLAB SIMULINK.

L'identification est l'opération de détermination des caractéristiques dynamiques d'un système à partir des données expérimentales ou de mesures. Son objectif principal est de créer des modèles mathématiques ou dynamiques qui représentent fidèlement le comportement du système réel. Elle se fait en utilisant différentes techniques telles que la régression, l'estimation des paramètres, l'analyse de Fourier, les méthodes de moindres carrés. Néanmoins, dans notre travail, nous allons utiliser la boite à outils d'identification des systèmes que MATLAB nous offre, nommé "System Identification Toolbox" (figure 1.9).

System Identification Toolbox est une boîte à outils logicielle proposée dans MATLAB. Elle est conçue pour faciliter le processus d'identification. Cette boîte à outils contient des fonctions détaillées du processus d'identification, telles que le prétraitement et l'estimation, elle contient également des fonctions pour analyser les performances du modèle estimé [11]. Pour l'identification du modéle mathématique de notre système on a suivi les étapes suivantes :

1. Les données de mesure entrée-sortie, issues de la carte d'acquisition Arduino Mega2560, sont importées dans l'interface de la Toolbox, en spécifiant dans "import data" le domaine des données (temporel ou fréquentiel).
2. Traitement des données d'acquisition (filtrage).
3. Dans "Estimate->", on choisi le type de modèle ("process model" dans notre cas), et la spécification des paramétres (nombre de pôles et leurs types, nombre de zéro du système, le retard, etc) et l'estimation de son ordre.
4. Validation du modèle estimé.

Le modéle obtenu est :

$$G(s) = \frac{2127.548\,s + 34.794}{2202.496\,s^2 + 117.551\,s + 1)} \tag{1.1}$$

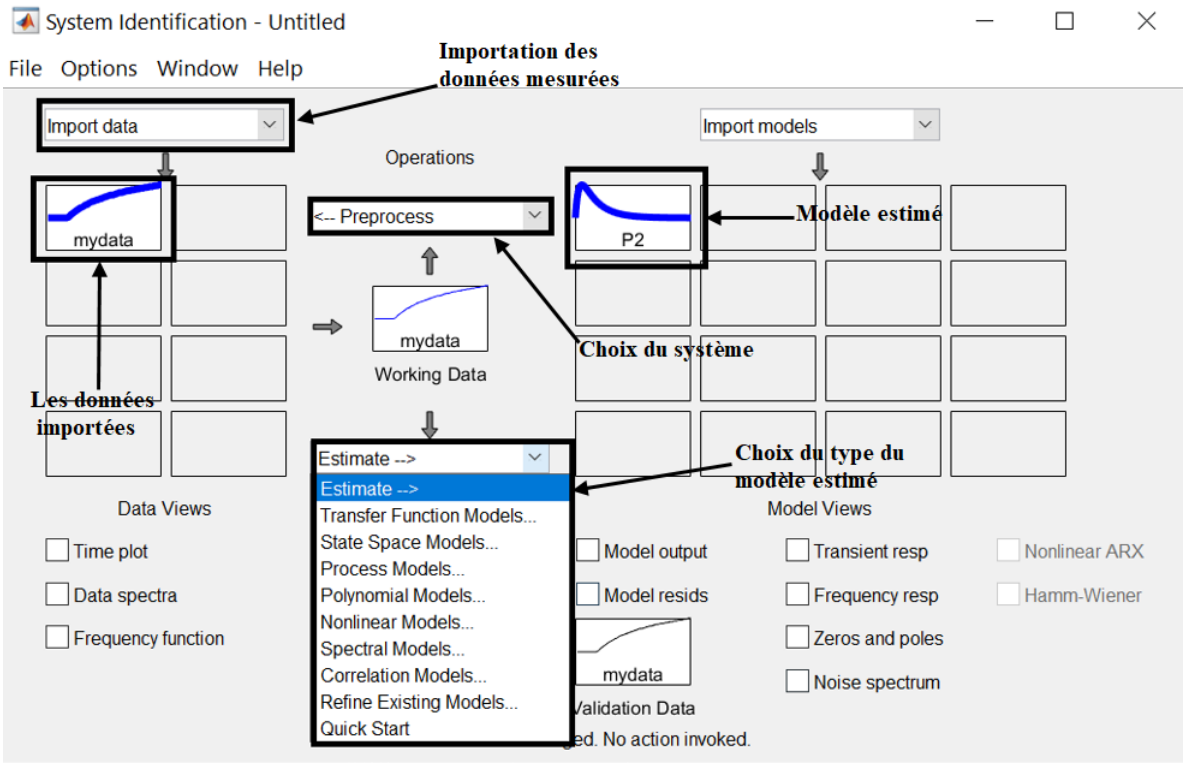


FIGURE 1.9 – Interface de System Identification Toolbox.

1.5 Synthèse du contrôleur PID

Pour la synthèse d'un contrôleur PID, plusieurs méthodes ont été proposées pour le réglage des paramètres du régulateur, chacune adaptée à des situations et des systèmes spécifiques. On cite : la méthode de Ziegler-Nichols, méthode de Cohen-Coon et d'autres méthodes basées sur l'optimisation, etc. Mais dans notre cas, Nous avons choisi une méthode qui utilise un outil logiciel, qui est l'application PID Tuner de l'environnement MATLAB (figure 1.10).

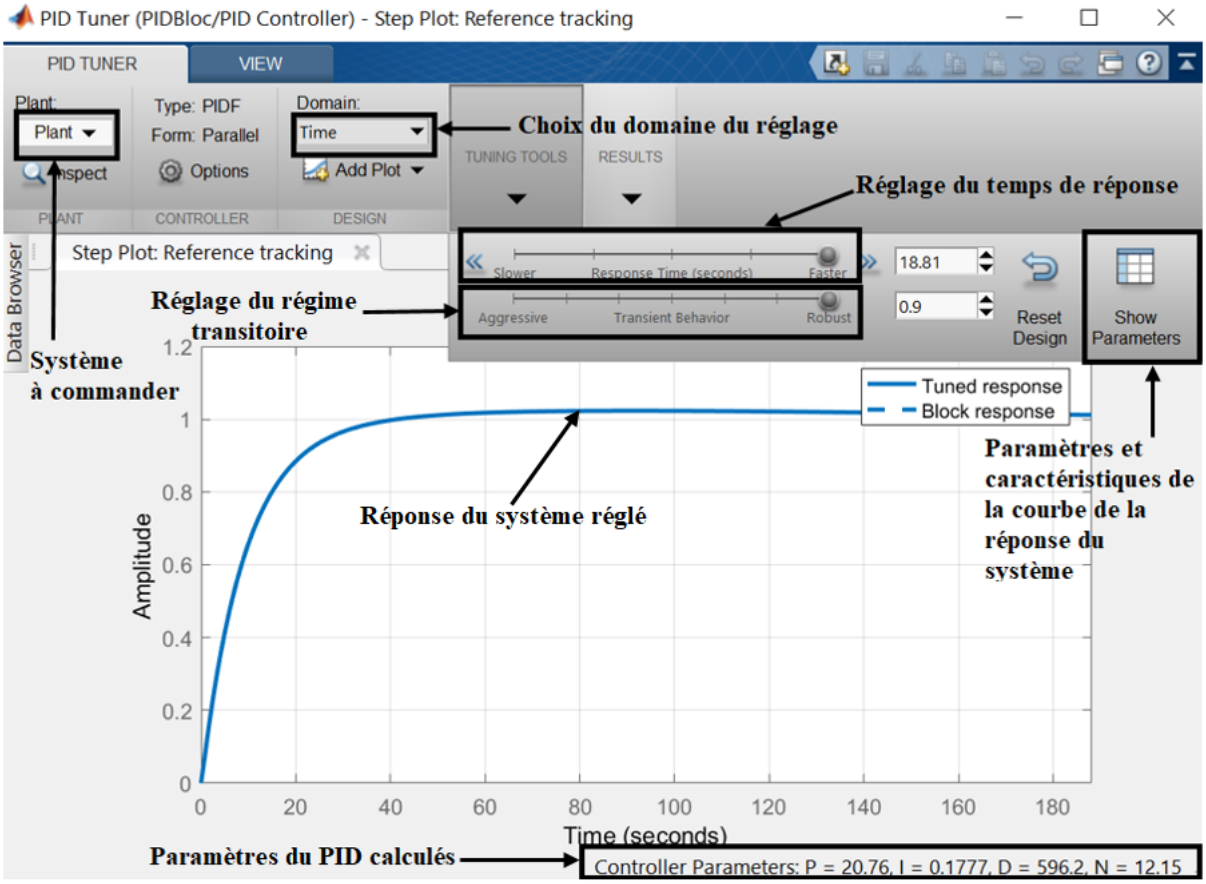


FIGURE 1.10 – Interface de PID Tuner.

PID Tuner est un outil intégré sur MATLAB. Il assure le réglage de manière interactive des gains proportionnel (P), intégral (I) et dérivé (D) d'un contrôleur PID, afin d'atteindre un bon équilibre entre performance et robustesse. Cet outil offre plusieurs fonctionnalités, il spécifie les objectifs de performance du système, tels que le temps de montée, le dépassement et le temps de stabilisation et il ajuste automatiquement les paramètres du régulateur PID pour répondre aux exigences. L'application permet aux utilisateurs de vi-

sualiser facilement les réponses du système en temps réel sous forme de courbes en offrant une interface graphique conviviale avec des fonctionnalités avancées et permet également d'ajouter des perturbations en entrée et en sortie du système.

Les paramètres du contrôleur PID obtenus sont les suivants (Tableau (1.1)) :

P	0.2072
I	0.0013
D	0.9894

TABLE 1.1 – Les paramètres du PID

1.6 Conclusion

Dans ce chapitre, Nous avons fait une déscription des éléments composant l'aérotherme et leur fonctionnement respectif, nous avons aussi présenté sa boucle de commande et son fonctionnement avec la carte arduino Mega2560. Par la suite, nous avons fait une identification du système afin de pouvoir synthétiser un contrôleur PID à l'aide des outils de l'environnement MATLAB, "System Identification Toolbox" pour l'identification et "PID Tuner" pour la synthèse de contrôleur.

Dans le chapitre suivant, nous allons proposer un contrôleur adaptatif neuro-flou (ANFIS).

Chapitre 2

Commande neuro-flou de l'aérotherme

2.1 Introduction

L'utilisation des techniques de l'intelligence artificielle telles que la logique floue, les réseaux de neurones dans la commande des processus industriels a connu un essor important au cours de ces dernières décennies. Ces techniques sont utlisées afin de résoudre différents problèmes liés à la commande, notamment la nécessité d'avoir un modèle aussi fidèle que possible, en tenant compte des erreurs de modélisation pouvant significativement affecter les performances des lois de commande conventionnelles [12].

Une des solutions pour pallier ces problèmes consiste à introduire une technique intelligente qui combine la logique floue (LF) avec les réseaux de neurones artificiels (RNAs) pour former un réseau neuro-flou. Cette hybridation permet de tirer profit des avantages des deux approches. D'une part, les RNAs offrent une capacité d'apprentissage et de généralisation permettant une représentation efficace de la connaissance. D'autres part, la LF permet de traduire l'expérience humaine en un ensemble de règles linguistiques et facilite le traitement des connaissances imprécises. L'utilisation des réseaux neuro-flous pour le contrôle de notre aèrotherme apparaît comme une solution appropriée, compte tenu de la nature dynamique non linéaire du système ainsi que des incertitudes engendrées par les variations des conditions environnementales. Cette approche offre la possibilité de mieux modéliser et anticiper les réponses du système, permettant ainsi une régulation plus efficace des variations de température et une amélioration de sa stabilité globale.

Dans la première partie de ce chapitre, nous allons présenter les principes de la logique floue. Par la suite, dans la deuxième partie, nous discuterons des réseaux de neurones artificiels, leurs différentes architectures ainsi que leur fonctionnement. Enfin, la dernière partie du chapitre, sera consacrée à la synthèse d'un contrôleur neuro-flou de type ANFIS (Adaptative Neural Fuzzy Inference System) pour commander notre système.

2.2 La logique floue

La logique floue est une extension de la logique classique (booléenne). Elle traite la notion de la vérité partielle plutôt que de la vérité absolue. Contrairement à la logique binaire, où une proposition peut être soit vraie soit fausse c'est-à-dire 1 ou 0, la logique floue permet qu'une proposition soit partiellement vraie ou partiellement fausse en lui assignant des degrés d'appartenance compris entre [0 1]. Elle a été introduite par Lotfi A. Zadeh en 1965 [13], où il stipule dans son article que cette logique permet de modéliser des situations où les frontières entre les catégories ne sont pas nettement définies, et qu'elle offre un moyen de traiter l'incertitude et l'imprécision présentes dans de nombreux systèmes du monde réel en se rapprochant le plus possible du raisonnement humain, du fait que ce dernier a la capacité de gérer l'incertitude et l'imprécision et utilise des concepts vagues et s'adapte aux différentes situations.

La logique floue est la technique de modélisation et de manipulation des concepts flous modélisés par des ensembles flous, elle permet d'associer à des variables des degrés d'appartenance à des sous-ensembles flous, avec des valeurs comprises entre 0 et 1 en prenant ainsi en considération toutes les incertitudes sur la variable.

2.2.1 Ensemble flou

Un ensemble flou est une classe d'objets caractérisée par une fonction d'appartenance qui attribue à chaque objet un degré d'appartenance variant entre 0 et 1 [13]. Ainsi, plus le degré d'appartenance est proche de 1, plus l'appartenance est forte.
Le concept de sous-ensemble flou a été introduit par L.A. Zadeh afin de représenter ma-

thématiquement l'imprécision relative à certaines classes d'objets.

Sous-ensemble flou

Un sous-ensemble flou A de X est défini par une fonction d'appartenance $\mu_A(x)$, qui associe à chaque élément x de X un degré d'appartenance compris entre 0 et 1.

$$\mu_A(x) \in [0\ \ 1] \tag{2.1}$$

Contrairement à un sous ensemble classique qui a une fonction d'appartenance, notée A, représentée par deux valeurs possibles : la valeur 1 si x appartient à A ou la valeur 0 si x n'appartient pas à A. Par conséquent :

$$\mu_A(x) = \begin{cases} 1 & si\ \ x\ \ \in A \\ 0 & si\ \ x\ \ \notin A \end{cases} \tag{2.2}$$

2.2.2 Variable linguistique

Une variable linguistique, ou variable floue, est une variable dont les valeurs appartiennent à des ensembles flous pouvant représenter des mots du langage naturel tels que : (négatif, zéro, positif) ou (petit, très petit, moyen, grand, très grand). Une variable floue permet de modéliser les connaissances imprécises ou vagues sur une variable dont la valeur précise est inconnue [14]. Ainsi, elle peut prendre simultanément plusieurs valeurs linguistiques. Ces dernières constituent alors des ensembles flous dans l'univers du discours.

Univers du discours

L'univers de discours représente le domaine de variation de la variable linguistique. Il est défini comme un ensemble U qui représente l'ensemble complet de toutes les valeurs potentielles de la variable floue. Ces valeurs peuvent être numériques ou symboliques.

2.2.3 Fonction d'appartenance

Une fonction d'appartenance est une courbe qui traduit comment chaque variable linguistique de l'univers du discours est associée à un degré d'appartenance définit dans l'intervalle [0 1], où 0 représente l'appartenance nulle et 1 représente l'appartenance totale.

Les fonctions d'appartenance peuvent être représentées sous plusieurs formes selon l'application. Les formes les plus utilisées dans le réglage par logique floue sont les fonctions triangulaires, trapézoïdales, sigmoïdes et gaussiennes [15].

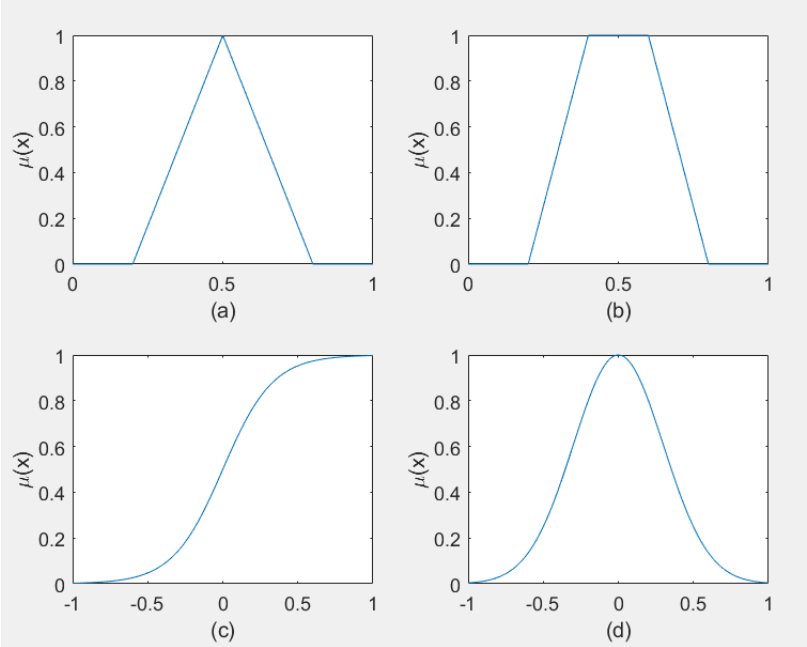


FIGURE 2.1 – Fonctions d'appartenance usuelles.

Fonction triangulaire

Elle est définie par trois paramètres (a,b et c) qui déterminent les coordonnées des trois sommets (figure 2.1(a)).

$$\mu_A(x) = max\left(min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{2.3}$$

Fonction trapézoïdale

Elle est caractérisée par quatre paramètres (a,b,c et d) qui définissent les coordonnées des quatre sommets (figure 2.1(b)).

$$\mu_A(x) = max\left(min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \tag{2.4}$$

Fonction sigmoïde

Cette fonction est définie par deux paramètres (a et c) (figure 2.1(c)).

$$\mu_A(x) = \left(\frac{1}{1+\exp\left(-a(x-c)\right)}\right) \tag{2.5}$$

Fonction gaussienne

La fonction gaussienne est caractérisée par deux paramètres (σ et m) (figure 2.1(d)).

$$\mu_A(x) = \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right) \tag{2.6}$$

2.2.4 Opérations sur les ensembles flous

Les variables linguistiques sont liées entre elles au niveau des règles d'inférence par les opérateurs suivant : ET (Intersection), OU (Union) et NON (complément à 1), ils permettent d'écrire des combinaisons logiques entre notions floues, c'est-à-dire de faire des calculs sur des degrés de vérité. Dans le cas de la logique binaire, ces opérateurs sont définis de façon simple et univoque. En revanche, la définitions de ces opérateurs, dans le cas de la logique floue n'est plus univoque car ils sont interprétés respectivement par : Minimum, Maximum et Complément à 1 (Tableau (2.1)).

Opérateurs	Logique classique	Logique floue
$C = A\ ET\ B$	$C = A \cap B$	$\mu_c(x) = Min(\mu_A(x), \mu_B(x))$
$C = A\ OU\ B$	$C = A \cup B$	$\mu_c(x) = Max(\mu_A(x), \mu_B(x))$
$C = NON\ (A)$	$C = \overline{A}$	$\mu_c(x) = 1 - \mu_A(x))$

TABLE 2.1 – Opérateurs dans les logiques classique et floue

2.2.5 Règles d'inférence

Les règles d'inférence constituent un système de règles floues. Elles permettent de décrire, sous forme de règles linguistiques, la relations entre les variables floues d'entrée et

les variables floues de sortie d'un système, au moyen de différents opérateurs flous. Ces règles sont décrites sous la forme suivante :

Si condition 1 ET/OU condition 2 (ET/OU. . .) alors action sur les sorties OU

Si condition 3 ET/OU condition 4 (ET/OU. . .) alors action sur les sorties OU

$\vdots$

Si condition $n-1$ ET/OU condition n (ET/OU. . .) alors action sur les sorties.

2.2.6 Commande par logique floue

La synthèse d'un contrôleur classique demande la connaissance précise du modèle du système à contrôler. De plus, les valeurs de ses entrées doivent être mesurées avec la plus grande précision possible afin d'éviter toutes les erreurs liées à l'état du système qu'elles décrivent. Contrairement au contrôleur classique, le contrôleur flou ne demande aucune de ces deux spécifications. Il n'est pas nécessaire de connaître le modèle analytique du processus pour le concevoir. Par conséquent, il ne traite pas de relations mathématiques bien définies mais utilise des inférences avec plusieurs règles, en se basant sur des variables linguistiques, ces inférences sont alors traitées par des opérateurs flous. Dans ce qui suit, nous allons présenter les composants essentiels d'un régulateur flou (Figure 2.2).

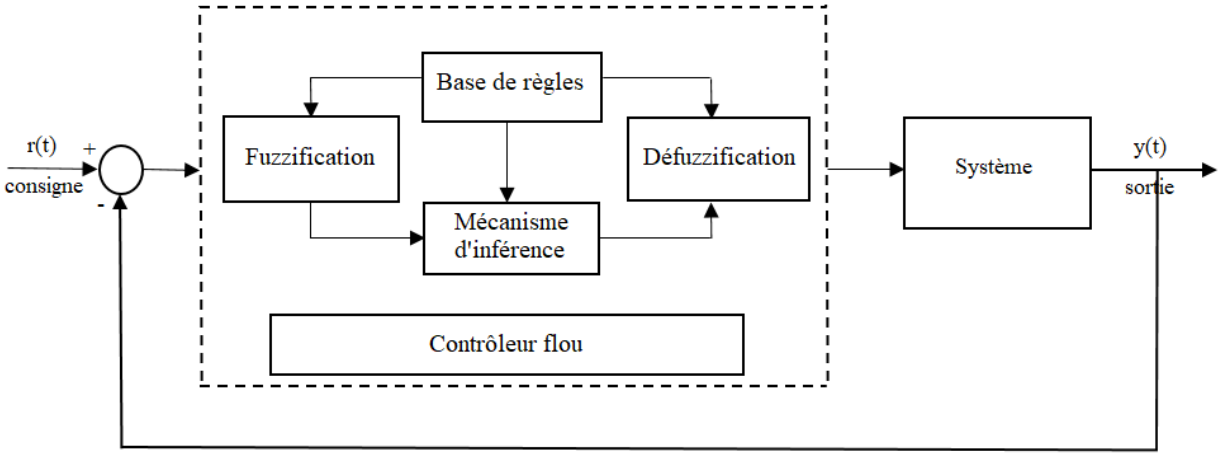


FIGURE 2.2 – Schéma fonctionnel d'un régulateur flou.

Fuzzification

Dans les problèmes de commande floue, les données observées sont habituellement des grandeurs physiques. Or le traitement de ces données est basé sur la théorie des ensembles flous, ceci nécessite donc une procédure de transformation nomée "fuzzification".
La fuzzification représente le passage des grandeurs réelles (ou physiques) générées par un capteur en des variables linguistiques en vue d'un traitement d'inférence. Elle caractérise le degré avec lequel ces grandeurs appartiennent à un sous-ensemble flou, en utilisant différentes formes de fonctions d'appartenance dont le choix de la forme et du nombre est déterminé selon le domaine d'application et la facilité de calculs.

Pour illustrer le mécanisme de la fuzzification, nous allons donner un exemple en fixant comme valeur d'entrée x=0.45 définie par l'ensemble des variables linguistiques : NG=Négatif Grand, NM=Négatif Moyen, NP= Négatif Petit, ZE= Zéro Environ, PP =Positif Petit, PM= Positif Moyen, PG= Positif Grand et l'univers de discours associé est [-1 1]. Le résultat de la fuzzification sera présenté sur la figure 2.3. On remarque que pour cette valeur d'entrée correspond les ensembles flous PP et PM avec les degrés d'appartenances $\mu_{PP}(e) = 0.60$ et $\mu_{PM}(e) = 0.40$.

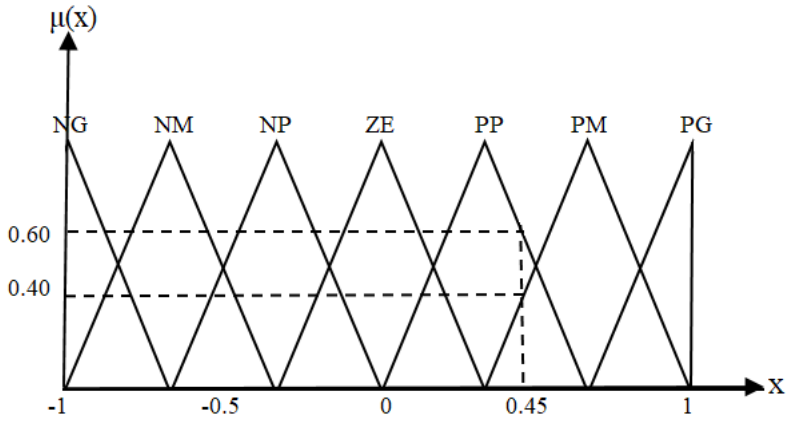


FIGURE 2.3 – Exemple de fuzzification

Base de règles floues

La base de règles d'un système flou est une base de connaissances renfermant des règles floues. Ces dernières décrivent le comportement d'une sortie du système lorsqu'elle

est soumise à diverses entrées. Elles sont données sous la forme suivante :

Si condition 1 ET/OU condition 2 (ET/OU. . .) alors action sur les sorties

Si condition 3 ET/OU condition 4 (ET/OU. . .) alors action sur les sorties

$$\vdots$$

Si condition $n-1$ ET/OU condition n (ET/OU. . .) alors action sur les sorties.

Mécanisme d'inférence

Le mécanisme d'inférence est une étape qui consiste à transformer, à l'aide des opérateurs flous, la partie floue issue par la fuzzification en une nouvelle partie floue. Il permet, à partir d'un fait observé de la base des règles floues, d'obtenir une conclusion raisonnable en exploitant le raisonnement approximatif [16]. Ce processus implique l'évaluation des degrés d'appartenance des entrées aux ensembles flous définis dans les règles.
Pour le réglage par logique floue, on utilise en générale une des méthodes suivantes [16] :

- *Méthode d'inférence max-min de Mamdani.*

  Elle réalise, au niveau de la condition, l'opérateur OU par fonction maximum et l'opérateur ET par la fonction minimum.

- *Méthode d'inférence max-prod de Larsen.*

  Cette méthode réalise en générale, au niveau de la condition, l'opérateur OU par fonction maximum et l'opérateur ET par la fonction produit.

- *Méthode d'inférence somme-prod de Sugeno.*

  Par opposition aux méthodes d'inférence précédentes, la méthode d'inférence somme-prod réalise, au niveau de la condition, l'opérateur OU par la fonction somme, tandis que l'opérateur ET est réalisé par la fonction produit.

Défuzzification

Les méthodes d'inférence fournissent une fonction d'appartenance résultante, il s'agit donc d'une information floue. Par cette étape, se fait alors le retour aux grandeurs de sorties réelles pour former un signal de contrôle pour le système à commander. À cet effet, elle représente alors l'inverse de la fuzzification d'où le nom "Défuzzification".

Plusieurs stratégies de défuzzification existent, les plus utilisées sont :

- *Méthode de centre de gravité.*

  C'est la méthode la plus utilisée [4]. la valeur de sortie est estimée par l'abscisse du centre de gravité de la surface générée par la fonction d'appartenance résultante .

- *Méthode du maximum.*

  Cette méthode, s'applique uniquement dans le cas où la fonction d'appartenance associée à l'ensemble de sortie n'admet qu'un seul maximum, On choisit comme sortie l'abscisse correspondant à ce maximum.

- *Méthode de la moyenne des maxima.*

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

2.3 Réseaux de neurones artificiels (R.N.A)

Les réseaux de neurones artificiels notés R.N.A, sont des systèmes de traitement de l'information dont la structure s'inspire de celle du système nerveux. Ils sont considérés comme une représentation mathématique simplifiée du cerveau humain[17]. Ils sont destinés à effectuer des tâches auxquelles les approximateurs traditionnels semblent moins adaptés. Les RNAs sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des signaux d'entrées, diversement pondérés, qu'il reçoit [18].

L'origine de l'inspiration des réseaux de neurones artificiels remonte à l'année 1890 où W. James a introduit le concept de mémoire associative. Quelques années plus tard, en 1943, J. Mc Culloch et W. Pitts ont publié un article [19] où ils proposent un modèle mathématique simplifié d'un neurone biologique. Ils ont montré que des RN formels simples peuvent réaliser des fonctions logiques, arithmétiques et symboliques complexes. Ils supposent que l'impulsion nerveuse est le résultat d'un calcul simple effectué par chaque neurone, et que la pensée est née grâce à l'effet collectif d'un réseau de neurones interconnectés. Ils ont connu des débuts prometteurs vers la fin des années 50, mais le manque

d'approfondissement de la théorie a gelé ces travaux jusqu'aux années 80.

Parmi les principaux avantages des réseaux de neurones artificiels, on citera leur grande capacité d'apprentissage de manière itérative, ce qui leur permet de s'adapter à des tâches spécifiques sans nécessiter d'une programmation explicite, ainsi que une grande capacité de traitement et de calcul, ce qui les rend efficaces pour des applications nécessitant un traitement en temps réel ou une grande quantité de données. Ils sont constitués d'un nombre fini de neurones qui sont arrangés sous forme de couches. Des poids pondérés établissent des connexions entre les neurones de deux couches adjacentes, permettant ainsi une transmission d'informations d'une couche à l'autre. Nous distinguons trois types de couches : *Couche d'entrée, Couche cachée, Couche de sortie*

2.3.1 Neurone biologique

Le neurone est la cellule fondamentale du système nerveux central [4]. Cette cellule est responsable de la transmission et du traitement de l'information au niveau du cerveau. Lorsque ce dernier est soumis à certains stimulis, les signaux transmis aux cellules nerveuses sont traités pour aboutir à une réponse du cerveau qui se traduit par un comportement observable. Le neurone est composé de quatre parties principales :

- *Le corps cellulaire (soma)* : C'est la partie centrale du neurone où se trouvent le noyau cellulaire. Et c'est l'unité qui se charge d'analyser et de traiter l'ensemble des informations transmises par les dendrites.
- *Les dendrites* : Ce sont de petites extensions en forme de branches qui partent du corps cellulaire. Leur rôle se joue dans la réception des informations et signaux issus des autres neurones.
- *L'axone* : C'est une longue projection unique qui s'étend depuis le corps cellulaire. Il a pour mission de transmettre les signaux électriques vers d'autres neurones.
- *Les synapses* : Ce sont les zones de contact, permettant la transmission des signaux électriques entre les axones d'un neurone et les dendrites des autres neurones.

2.3.2 Neurone formel

Une analogie directe, faite par Culloch et Pitts, avec le neurone biologique a donné naissance au neurone formel. Ce dernier représente un modèle simplifié du neurone biologique (figure2.4). Il est définit comme étant un élément de traitement (opérateur mathématique) possédant n entrées $\{x_1, x_2, ..., x_n\}$ (Dendrites) pondérées par des coefficients appelés poids synaptiques $\{w_1, w_2, ..., w_n\}$ et s la sortie du corps cellulaire artificiel.

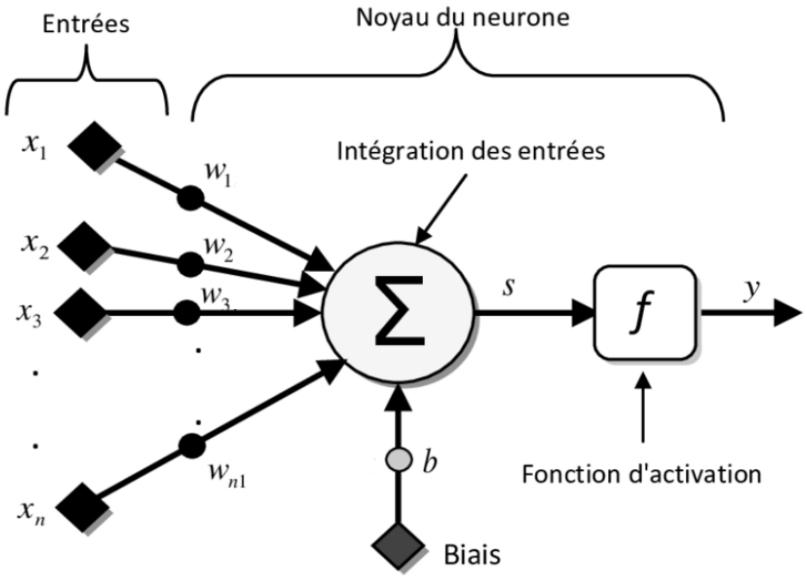


FIGURE 2.4 – Modèle général d'un neurone formel.

La sortie du neurone y est définie par l'équation suivante :

$$y = f(s) \tag{2.7}$$

avec

$$s = \sum_{i=1}^{n} w_i x_i + b \tag{2.8}$$

Où b et f sont respectivement le biais du neurone (le seuil d'activation du neurone) et la fonction d'activation.

Conformément au modèle biologique, les fonctions d'activation sont généralement croissantes et bornées. Elles peuvent prendre plusieurs formes à savoir : *fonction identité, saturation, fonction seuil, fonction gaussienne, fonction sigmoïde, fonction tangente hyperbolique.*

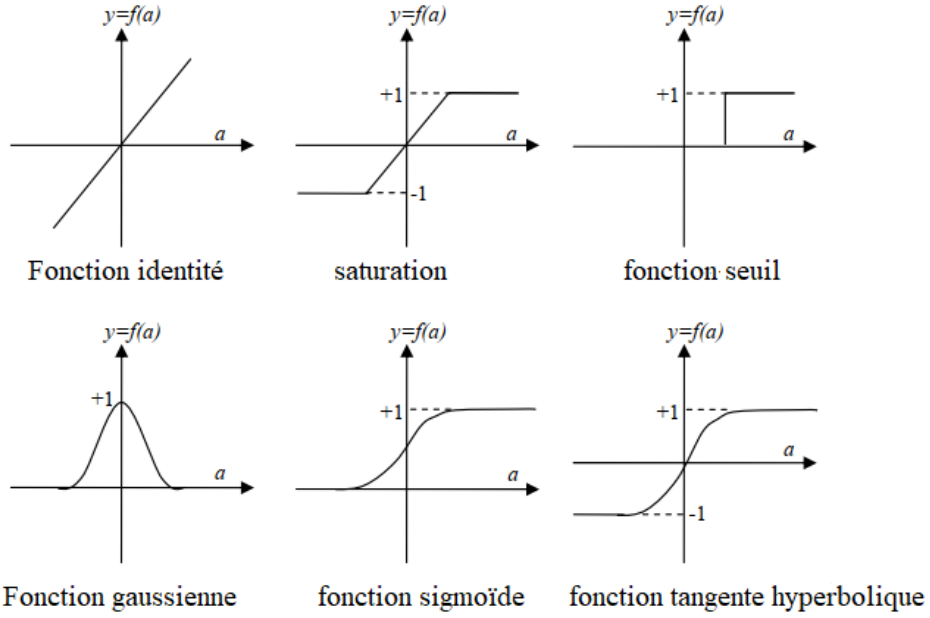


FIGURE 2.5 – Formes usuelles de la fonction d'activation

2.3.3 Architectures des réseaux de neurones artificiels

Un RNA est constitué d'une multitude de neurones connectés entre eux par un très grand nombre d'interconnexions. Ces dernières définissent la topologie de l'architecture du réseau. La diversité des interconnexions engendre une variété de types d'architectures des RNAs, chacune adaptée à des applications spécifiques et offrant des capacités d'apprentissage distinctes. On distingue deux grandes familles d'architectures à savoir : les architectures bouclés et les architectures non bouclés.

Architectures non bouclés

Dans ce type d'architecture, aussi, on distingue deux familles : les réseaux à une seule couche et les réseaux à multi-couches.

- ***Architecture non bouclée à une seule couche*** **:** C'est un réseau simple unidirectionnel. Il possède une seule couche d'entrée, qui reçoit les stimulis à traiter par l'intermédiaire des nœuds sources, et une seule couche de sortie, qui se charge de transmettre les résultats du traitement au milieu extérieur (figure 2.6). Ces réseaux sont utilisés dans les problèmes de classification et de filtrage linéaire [4]. Parmi les principaux types de réseaux non-bouclés à une seule couche, on cite le Perceptron

et l'ADALINE [20].

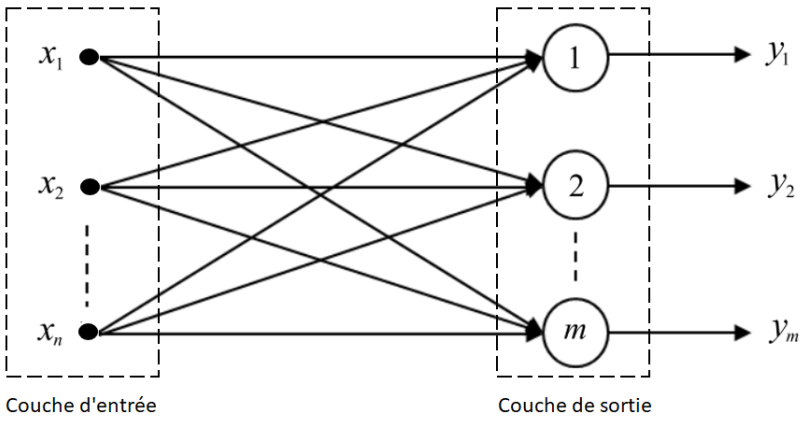


FIGURE 2.6 – Réseau de neurones artificiels non bouclé mono-couche.

- ***Architecture non-bouclée à multi-couches*** **:** Ce type de d'architecture se caractérise par la présence d'une ou plusieurs couches cachées (figure 2.7), leur nombre dépend de la nature et de la complexité du problème à résoudre, en plus de la couche d'entrée et une de sortie [15]. le rôle de ces couches cachées est d'effectuer un prétraitement des signaux d'entrée, reçus par la couche d'entrée et de transmettre les résultats correspondants à la couche de sortie. Ce type de RNAs est utilisé dans les problèmes de classification des formes, robotique, l'approximation des fonctions, l'identification des systèmes et le contrôle des procédés. Parmi les principaux réseaux non-bouclés multi-couches, on trouve le Perceptron multicouches et la fonction de base radiale [20].

Architecture bouclée (récurrente)

Les réseaux récurrents possèdent une structure similaire à celle des réseaux unidirectionnels (Architecture non-bouclée) mais les couches de sorties sont utilisées comme des entrées de rétroaction pour d'autres couches (figure 2.8), permettant ainsi aux réseaux d'être bidirectionnel (les informations cirulent des deux sens). Ces réseaux sont utilisés généralement pour la prédiction temporelle, l'identification, l'optimisation et le contrôle des processus. Parmi les principaux réseaux de neurones bouclés, on cite les réseaux de Hopfield et de Perceptron [20].

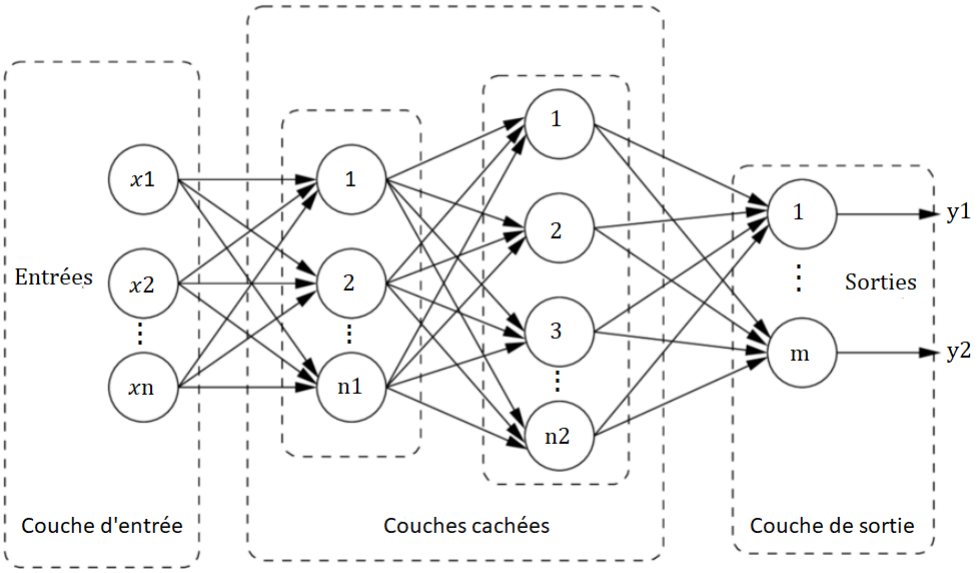


FIGURE 2.7 – Réseau de neurones artificiels non bouclé multi-couche.

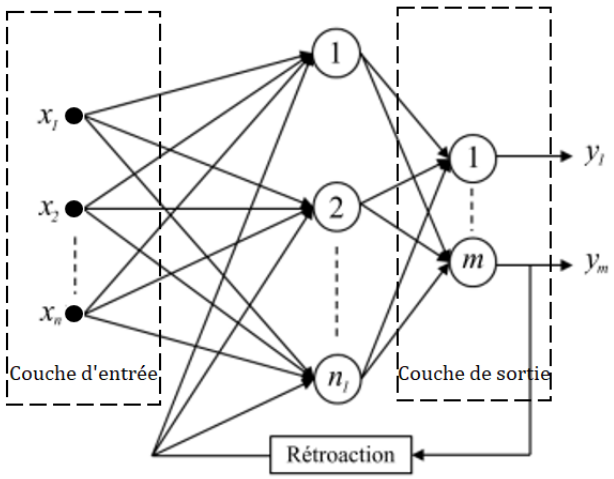


FIGURE 2.8 – Réseau de neurones artificiels bouclé.

2.4 Processus d'apprentissage

Les réseaux de neurones artificiels se caractérisent par leur grande capacité d'apprentissage [4]. L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré en modifiant les poids d'interconnexion [21].

Le processus d'apprentissage peut s'effectuer selon plusieurs règles et différentes manières. A cet effet, on distingue trois types d'apprentissages : Supervisé, non supervisé et par renforcement.

2.4.1 Apprentissage supervisé

L'apprentissage supervisé suppose l'éxistance d'un expert qui fournit les sorties désirées décrivant la fonction du réseau. Ensuite, ce dernier compare les sorties obtenues aux sorties désirées, et la différence entre les deux sorties, appelée l'erreur, sera utilisée pour adapter les paramètres du réseau de façon à corriger son comportement afin d'atteindre les performances souhaitées (figure 2.9).

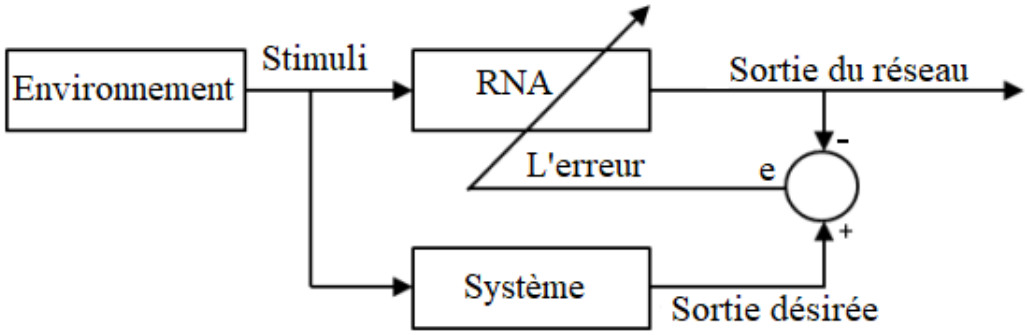


FIGURE 2.9 – Illustration de l'apprentissage supervisé.

Cet apprentissage est le plus utilisé. mais son inconvénient reste que l'agent n'est pas immédiatement autonome, puisqu'il a besoin d'un superviseur qui, dans un premier temps, lui indique la marche à suivre dans des situations qu'il pourra rencontrer. Parmi les algorithmes d'apprentissage supervisé les plus connus, on cite l'algorithme de rétro-propagation de l'erreur.

Algorithme de rétro-propagation

La méthode de rétro-propagation de l'erreur est une technique d'apprentissage supervisé. Elle permet d'ajuster les poids et les biais du RNA afin de minimiser l'erreur quadratique entre la sortie calculée du réseau et sa sortie réelle.

Soit une réponse spécifique désirée D_p à chaque vecteur d'entrée X_p. La mise à jour des coefficients synaptiques W s'effectuent progressivement jusqu'à ce que l'erreur entre les sorties du réseau et les résultats désirés soit minimisée. Pour ce faire considérons le problème de minimisation de la fonction coût définie pour n exemples $(X_p : D_p)$ et Y_p la

sortie du réseau, par l'erreur qui peut être quadratique de la forme.

$$E(W) = \frac{1}{2}\sum_{p}(Y_p - D_p)^2 \tag{2.9}$$

L'objectif consiste à calculer les poids W afin de minimiser $E(W)$.

$$W = ArgMin(E) \tag{2.10}$$

A fin de résoudre l'équation (2.10), généralement on fait appel à des méthodes de gradient partiel, total ou stochastique (Tableau 2.2).

$$W(t+1) = W(t) - \lambda\nabla_W(E) \tag{2.11}$$

Méthode	Équation
gradient partiel	$W(t+1) = W(t) + \lambda(D_p - f(WX_P))f'(WX_P)X_p$
gradient total	$W(t+1) = W(t) - \lambda\nabla_W(E(t))$
gradient stochastique	$W(t+1) = W(t) - \lambda\nabla_W(E_p(t))$

TABLE 2.2 – Diffirentes méthodes du gradient

où :

- $E_p = (Y_p - D_p)^2$ est l'erreur calculée sur un seul exemple sélectionné au hasard à chaque instant t.
- f est une fonction d'activation.
- λ (t) est le pas de gradient (fixe, décroissant, ou adaptatif).

Pour calculer $\frac{\partial E_p}{\partial W_{ij}}$, on exploite la rétro propagation du gradient, qui est une méthode astucieuse de calcul du gradient de la fonction coût en utilisant la dérivation composée pour rétro-propager l'erreur, comme le montre l'équation suivante :

$$\frac{\partial E_p}{\partial W_{ij}} = \frac{\partial E_p}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial W_{ij}} = \left(\frac{\partial E_p}{\partial \sigma_j}\right) y_i \tag{2.12}$$

On pose $\delta = \frac{\partial E_p}{\partial \sigma_j}$ d'où :

$$W(t+1) = W_{ij}(t) - \lambda\delta_j yi \tag{2.13}$$

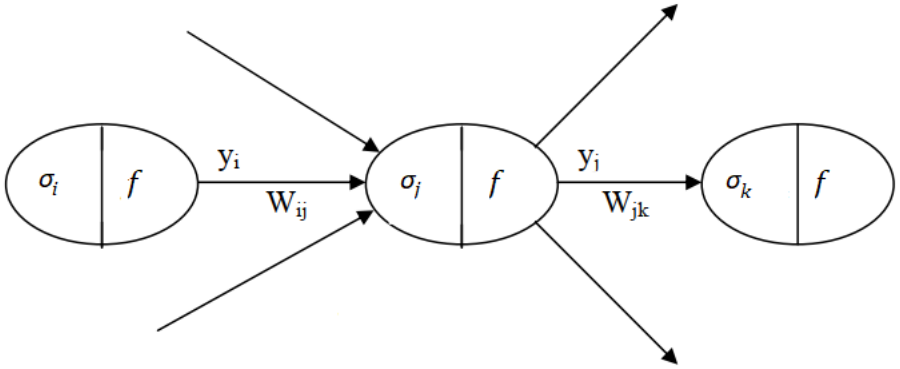


FIGURE 2.10 – Rétro-propagation du gradient

avec :

$$\delta_j = \frac{\partial E_p}{\partial \sigma_j} = \sum_k \left(\frac{\partial E_p}{\partial \sigma_k}\right)\left(\frac{\partial \sigma_k}{\partial \sigma_j}\right) = \sum_k \delta_k \left(\frac{\partial \sigma_k}{\partial \sigma_j}\right) = \sum_k \delta_k W jk \left(\frac{\partial yj}{\partial \sigma_j}\right) \tag{2.14}$$

d'où :

$$\delta_j = \left(\sum_k \delta_k W jk\right) f'(\delta_j) \tag{2.15}$$

et :

$$\delta_j = \left(\frac{\partial E_p}{\partial \sigma_j}\right) = \left(\frac{\partial E_p}{\partial yj}\right)\left(\frac{\partial yj}{\partial \sigma_j}\right) \tag{2.16}$$

d'où

$$\delta_j = 2(y_i - D_j) f'(\sigma_j) \tag{2.17}$$

Cependant δ_j se calcule de proche en proche par rétro propagation de l'erreur.

2.4.2 Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, effectué sous contrôle d'un expert, l'apprentissage non supervisé est autodidacte [12] i.e. il ne nécessite aucune information sur les sorties désirées (figure 2.11). Il repose sur un critère interne de conformité du compor-

tement du réseau par rapport à des spécifications générales et non sur des observations externes. Dans ce cas, l'apprentissage est basé sur des probabilités, le réseau va se modifier en fonction des régularités statistiques de l'entrée.

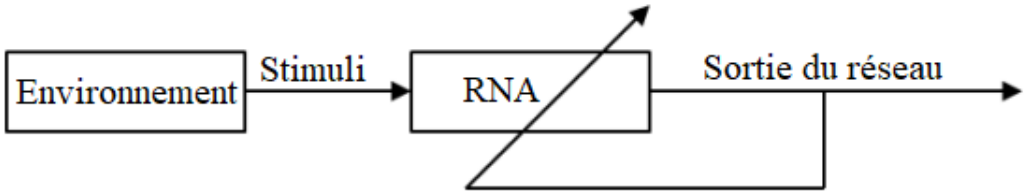


FIGURE 2.11 – Illustration de l'apprentissage non supervisé

2.4.3 Apprentissage par renforcement

Dans ce type d'apprentissage, on entraîne le réseau par tâtonnement en précédant par essais et erreurs. Le réseau est alors stimulé par l'environnement et ses réponses sont sanctionnées ou récompensées afin de l'inciter à adopter le bon comportement.

Le choix d'utiliser une telle architecture de réseau de neurones ou une autre, tel ou tel type d'apprentissage dépend de l'application mais aussi des capacités de traitement du système sur lequel ces architectures vont être implantées.

2.5 Réseaux neuro-flous

2.5.1 Systèmes neuro-flous

Jusqu'aux années 90, les deux techniques de l'intelligence artificielle à savoir les réseaux de neurones et la logique floue apparaissaient comme des approches bien distinctes, ayant chacune leurs avantages et inconvénients et leurs domaines spécifiques. Mais elles ont le même objectif, les deux tentent de modéliser le fonctionnement du cerveau humain. Les RNAs tentent de modéliser l'architecture du cerveau en créant une modélisation de

l'entité du cerveau : le neurone. Les systèmes d'inference flous (SIF) modélisent le cerveau par son mode de fonctionnement (apprentissage et déduction). D'une part, les RNAs constituent des approximateurs universels grâce à leur capacité d'apprentissage, mais leur structures et leur paramètres n'ont pas toujours d'interprétation physique et la connaissance humaine ne peut pas être exploitée pour les construire. D'autre part, les SIFs sont construits à partir des connaissances humaines, et ils sont dotés d'une capacité descriptive élevée due à l'utilisation de variables linguistiques, néanmoins, aucune méthode formelle n'existe permettant de définir les fonctions d'appartenance ou de construire la base de règles pour les systèmes. Il est donc naturel de construire des systèmes hybrides, appelé réseaux neuro-flous, qui combine les concepts des réseaux de neurones et ceux de la logique floue (figure 2.12).

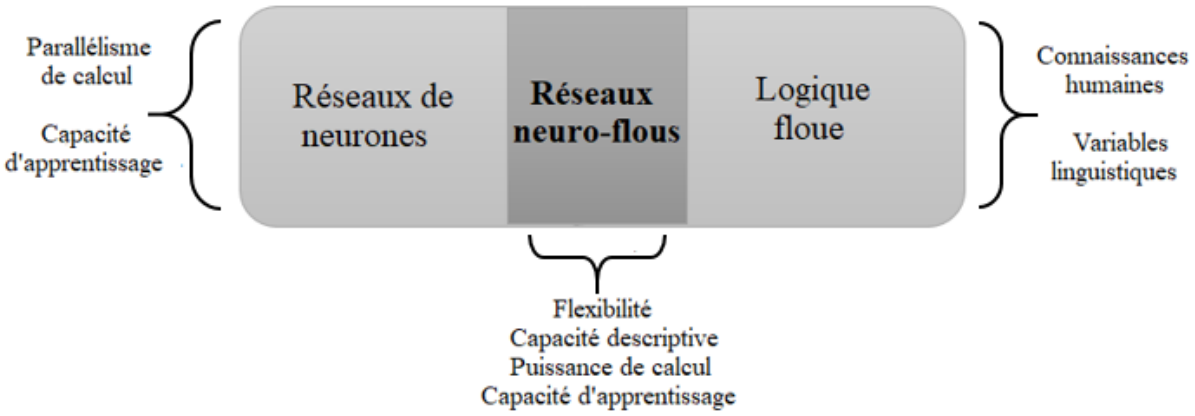


FIGURE 2.12 – Réseaux neuro-flous.

Une telle combinaison permet d'utiliser les méthodes d'apprentissage supervisé, comme par exemple la rétropropagation du gradient, pour optimiser et adapter les règles linguistiques et les fonctions d'appartenance du système floue [14]. Un des avantages principaux de la commande par les réseaux neuro-flous est qu'elle n'exige pas des informations sur le modèle mathématique du système à commander. La commande NF permet la résolution de plusieurs problèmes de commande dont le modèle mathématique est inconnu ou présente des incertitudes dynamiques. Plusieurs combinaisons de ces deux méthodes ont été développées dans la littérature qui ont donné naissance aux systèmes neuro-flous.

2.5.2 Type de combinaisons neuro-floues

Il existe trois grandes familles de combinaisons de réseaux de neurones avec la logique floue.

Système flou neuronal

Dans les système flous neuronaux, la logique floue est le principal moteur du traitement des données et de la prise de décision, tandis que les réseaux de neurones sont utilisés pour affiner ou optimiser les ensembles flous et les fonctions d'appartenances.

Système neuronal flou

Dans les systèmes neuronaux flous, les principes des réseaux de neurones, tels que l'apprentissage et la propagation, sont prédominants, tandis que la logique floue est utilisée pour gérer l'incertitude ou pour définir des règles dans le processus de prise de décision.

Système neuro-flou hybride

Les systèmes neuro-flous hybride sont synthétisés à partir des combinaisons synergiques de réseaux de neurones et de logique floue, qui visent à tirer profit des avantages de ces deux modéles pour la modélisation, la prise de décision et le contrôle dans des environnements complexes et incertains. Ces systèmes exploitent la puissance des réseaux de neurones pour leurs capacités de traitement et les avantages de la logique floue pour sa capacité de raisonnement flexible pour la prise de décision [4].

2.5.3 Types d'implémentation des réseaux neuro-flous

Il existe une multitude de systèmes NF. Mais selon l'interaction entre la logique floue et les réseaux de neurones, et compte tenu de nos objectifs, nous avons retenu deux types : le système d'inférence neuro-flous et le système d'inférence neuro-flou organisé en réseau adaptatif (ANFIS).

Système d'inférence neuro-flou

Un système d'inférence neuro-flou (INF) est une combinaison entre la logique floue et les réseaux de neurones artificiels. Son objectif principal est de réaliser un processus de raisonnement flou en utilisant une architecture de RNA de manière à ce que les paramètres de la logique flou soient représentés par les poids du réseau de neurones [15]. Ainsi, le système INF permet d'ajuster automatiquement et d'optimiser les fonctions d'appartenance, et cela en modifiant les poids de connexion du RN via un algorithme d'apprentissage approprié.

Système d'inférence neuro-flou adaptatif (ANFIS)

Le système ANFIS (Adaptative Network based Fuzzy Inference System) est un réseau adaptatif proposé par Jang en 1993 [22]. Ce système appartient à la famille des systèmes neuro-flous hybrides, qui combinent les avantages de la logique floue avec ceux des réseaux de neurones dans un seul réseau. Il peut être vu comme un réseau de neurones non bouclé pour lequel chaque couche est un composant d'un système flou. Le modèle ANFIS est le modèle le plus utilisé en pratique [15]. Des applications notamment dans le traitement du signal, le filtrage adaptatif et la commande des systèmes ont été réalisées avec cette architecture.

L'architecture ANFIS décrit le fonctionnement d'un système flou, avec des règles qui peuvent être de type Takagi Sugeno ou Mamdani, sous forme d'un réseau de neurones, entraîné à l'aide d'un algorithme d'apprentissage de rétropropagation. Le rôle de l'apprentissage est l'ajustement des paramètres de ce système d'inférence flou (partie prémisse et partie conséquence des règles). Ce système hybride neuro-flou se compose de cinq couches où les nœuds adaptatifs sont situés à la première et la quatrième couche. Afin de présenter l'architecture de base et le fonctionnement du ANFIS utilisé dans ce travail, on prend comme exemple un réseau neuro-flou avec un modèle d'inférence floue du premier ordre.

Le modèle illustré sur la figure 2.13 définit un ANFIS à deux entrées x_1 et x_2 et une seule sortie y. Chaque entrée possède deux fonctions d'appartenances respectivement A_1,A_2 et B_1,B_2. Il est modélisé par un système flou de type Takagi-Sugeno composé des

deux régles suivantes :

- $Si \quad x_1 \ est \ A_1 \ et \ x_2 \ est \ B_1 \quad alors \quad y_1 = F_1(x_1, x_2) = a_1 x_1 + b_1 x_2 + c_1$ (2.18)

- $Si \quad x_1 \ est \ A_2 \ et \ x_2 \ est \ B_2 \quad alors \quad y_2 = F_2(x_1, x_2) = a_2 x_1 + b_2 x_2 + c_2$ (2.19)

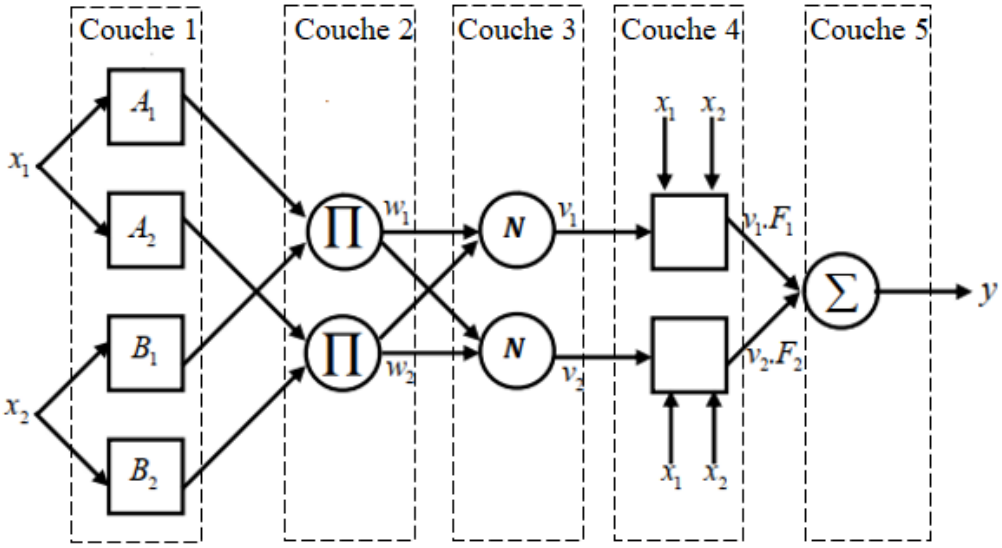


FIGURE 2.13 – Architecture d'un réseau ANFIS équivalent au modèle TSK.

Le réseau comporte 5 couches dont la fonction de chacune est décrite comme suit :

Couche 1 : Fuzzification

Chaque nœud de cette couche possède des paramètres ajustables, et il est doté d'une fonction d'appartenance qui est identique à un sous-ensemble flou de l'univers du discours des entrées.

$$O_{1j} = \mu_{A_j}(x_1), \quad pour \ \ j = 1, 2 \tag{2.20}$$

$$O_{1j} = \mu_{B_j}(x_2), \quad pour \ \ j = 1, 2 \tag{2.21}$$

où :

O_{1j} sont les degrés d'appartenance des variables x_1 et x_2 respectivement aux sous-ensembles flous A_j et B_j, qui peuvent être définis par différentes formes de fonctions d'appartenance.

Couche 2 : Degré d'activation

Chaque nœud de cette couche est un noeud circulaire fixe appelé $\prod$ qui engendre en sortie le produit de ses entrées. Ce produit représente le degré d'activation approprié à une règle floue.

$$O_{2j} = W_j = \mu_{A_j}(x_1)\mu_{B_j}(x_2), \quad pour \;\; j = 1, 2. \tag{2.22}$$

Couche 3 : Normalisation

Chaque nœud de cette couche est un noeud circulaire fixe appelé N. Sa sortie représente le degré d'activation normalisé de la j^{ime} règle.

$$O_{3j} = v_j = \frac{w_j}{w_1 + w_2}, \quad pour \;\; j = 1, 2. \tag{2.23}$$

Couche 4 : Calcul des sorties des règles

Chaque nœud de cette couche est un noeud carré et adaptatif avec une fonction réalisant le calcul suivant :

$$O_{4j} = v_j F_j = v_j(a_j x_1 + b_j x_2 + c_j), \quad pour \;\; j = 1, 2. \tag{2.24}$$

Où :

v_j est la sortie de la couche 3, (a_j, b_j, c_j) sont les paramètres de sortie ajustables de la j^{ime} règle. Ces paramètres sont appelés paramètres de la conséquence. Ils sont identifiés dans le processus d'apprentissage, en utilisant un algorithme basé sur des méthodes d'optimisation telle que la descente du gradient.

Couche 5 : Défuzzification

C'est un nœud unique de forme circulaire fixe qui calcule la sortie globale qui est la somme de tous les signaux provenant de la couche 4, c'est-à-dire :

$$O_{5j} = y = \sum_j v_j F_j, \quad pour \;\; j = 1, 2 \tag{2.25}$$

On remarque que la sortie globale du réseau est équivalente à la sortie du modèle TSK. La généralisation à un système à n entrées ne pose aucun problème particulier. Le nombre de noeuds de la couche 1 est toujours égal au nombre total de termes linguistiques définis.

2.6 Commande neuro-flou du procédé thermique

Afin d'assurer la poursuite des profils de température avec l'aérotherme, les algorithmes de contrôle doivent être stables, robustes et rapides. Cependant, l'aérotherme et sa boucle de commande sont des systèmes complexe avec une dynamique thérmique non linéaire qui ne peut pas parfois être disponible en raison des incertitudes du modèle, car ce dispositifs de chauffage peut être utiliser dans des environnements inconnus. De plus, dans certaines applications industrielles et commerciales, leurs interactions avec d'autres systèmes peut introduire des contraintes supplémentaires. L'utilisation des contrôleurs adaptatifs ainsi que les techniques de l'intelligence artificielle est alors une solution adéquate pour remédier à ces contraintes. Plusieurs travaux ont été menés dans ce domaine. L'auteur dans [3] a proposé un contrôleur neuro-flou en cascade pour un système de contrôle de la température de l'eau. Dans [23], un contrôleur PID flou adaptatif a été proposé pour améliorer la précision de la température du système de climatisation. Un contrôleur à base d'un réseau de neurones à rétropropagation pour une régulation de température a été proposé dans [24]. Ces travaux avaient pour objectif d'améliorer les performances du suivi des profils de température.

Dans ce travail de fin d'étude, nous proposons un contrôleur adaptatif neuro-flou (ANFIS) pour l'aérotherme réalisé (figure 2.14), en utilisant une carte arduino mega2560. Ce type de contrôleur ne nécessite aucune connaissance du modéle du système. De plus, il est trés robuste et il s'adapte aux différentes perturbations, en ajustant les paramètres de la conséquence du réseau neuro-flou dans un temps très court afin de le faire converger rapidement vers les performances désirées.

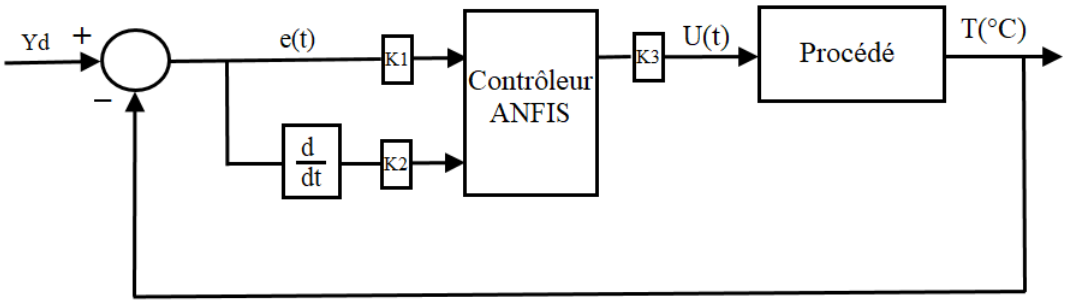


FIGURE 2.14 – Shéma fonctionnel avec le contrôleur ANFIS

2.6.1 Synthèse du contrôleur ANFIS

Dans cette section, nous développons le contrôleur ANFIS pour le procédé thermique. Considérons un réseau neuro-flou de premier ordre avec un système d'inférence floue de type Takagi-Sugeno, comportant deux entrée $x1$ et $x2$ et une seule sortie y (figure 2.15).

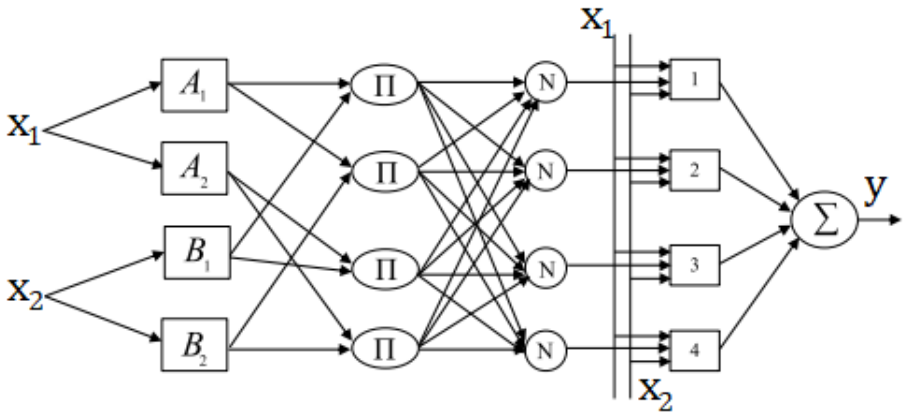


FIGURE 2.15 – Structure du régulateur ANFIS.

Les règles floues sont définies ainsi :

- $Si \quad x_1 \ est \ A_1 \ et \ x_2 \ est \ B_1 \quad alors \quad u_1 = a_1x_1 + b_1x_2 + c_1$ (2.26)
- $Si \quad x_1 \ est \ A_1 \ et \ x_2 \ est \ B_2 \quad alors \quad u_2 = a_2x_1 + b_2x_2 + c_2$ (2.27)
- $Si \quad x_1 \ est \ A_2 \ et \ x_2 \ est \ B_1 \quad alors \quad u_3 = a_3x_1 + b_3x_2 + c_3$ (2.28)
- $Si \quad x_1 \ est \ A_2 \ et \ x_2 \ est \ B_2 \quad alors \quad u_4 = a_4x_1 + b_4x_2 + c_4$ (2.29)

Où $u_j = a_j x_1 + b_j x_2 + c_j$ pour $j = 1, 2..., 4$, $x1$ et $x2$ sont respectivement l'erreur de température et sa dérivée : $[x1, x2] = [e, \Delta e]$. Ai et Bi sont des sous-ensembles flous. aj, bj, et cj sont les paramètres de la conséquence de la règle j déterminés dans le processus d'apprentissage.

Nous associons deux ensembles flous pour chacune des entrées $x1$ et $x2$ nommés N(Negative) et P(Positive). μ_P et μ_N sont les degrés d'appartenance des variables xi aux sous ensembles flous Ai et Bi, définis par les fonctions d'appartenance suivantes :
Pour i=1,2 :

$$\mu_P(xi) = \begin{cases} 0 & si \ \ xi < -1 \\ 0.5xi + 0.5 & si \ \ -1 < xi < 1 \\ 1 & si \ \ xi > 1 \end{cases} \tag{2.30}$$

$$\mu_N(xi) = \begin{cases} 1 & si \ \ xi < -1 \\ -0.5xi + 0.5 & si \ \ -1 < xi < 1 \\ 0 & si \ \ xi > 1 \end{cases} \tag{2.31}$$

Pour la diffuzification, on utilise la méthode de centre de gravité. la valeur numérique de la sortie y est donnée par :

$$y = \sum_{j=1}^{4} v_j u_j \tag{2.32}$$

où :

$$v_j = \frac{w_j}{w_1 + w_2 + w_3 + w_4}, \quad pour \ \ j = 1.2, ..., 4. \tag{2.33}$$

avec :

$$\begin{cases} w1 = \mu_P(x_1).\mu_P(x_2) = \mu_{A_1}(x_1).\mu_{B_1}(x_2) \\ w2 = \mu_P(x_1).\mu_N(x_2) = \mu_{A_1}(x_1).\mu_{B_2}(x_2) \\ w3 = \mu_N(x_1).\mu_P(x_2) = \mu_{A_2}(x_1).\mu_{B_1}(x_2) \\ w4 = \mu_N(x_1).\mu_N(x_2) = \mu_{A_2}(x_1).\mu_{B_2}(x_2) \end{cases}$$

2.6.2 Algorithme d'apprentissage

L'apprentissage concerne l'identification des paramètres des prémisses et des conséquences. L'algorithme d'apprentissage commence par construire un réseau initial, en suite nous appliquons une méthode d'apprentissage par rétro-propagation de l'erreur en utilisant une règle hybride d'apprentissage qui combine un algorithme de descente de gradient avec une estimation par moindres carrées. Ainsi, posons y_d et y respectivement les sorties désirée et réelle du système. Dans ce cas, nous considérons que les paramètres de la prémisse (couche 1) sont ceux proposés par l'expert, alors que ceux de la conséquence sont ajustés en minimisant la fonction objectif suivante :

$$J = \frac{1}{2}e^2 = \frac{1}{2}(y_d - y)^2 \tag{2.34}$$

Soit ϕ_j le vecteur des paramètres à ajuster. L'objectif est de trouver les paramètres a_j, b_j et c_j du vecteur ϕ_j en utilisant la méthode de la descente du gradient comme suit :

$$\phi_j(k+1) = \phi_j(k) - \alpha(k)\frac{\partial J}{\partial \phi_j} \tag{2.35}$$

on a :

$$\frac{\partial J}{\partial \phi_j} = -e\frac{\partial y}{\partial \phi_j} = -e\frac{\partial y}{\partial u}\frac{\partial u}{\partial \phi_j} \tag{2.36}$$

À partir des équations (2.35) et (2.36), on aura :

$$\phi_j(k+1) = \phi_j(k) + \alpha(k)\frac{\partial y}{\partial u}\frac{\partial u}{\partial \phi_j}e \tag{2.37}$$

En utilisant les équations du filtre de Kalman étendu [25], on peut estimé le terme $\frac{\partial y}{\partial u}$

puisque on peut pas le calculer. L'équation (2.37) peut être écrite comme suit :

$$\phi_j(k+1) = \phi_j(k) + K'_j(\psi_j)^T e \tag{2.38}$$

avec :

$$(\psi_j)^T = \frac{\partial u}{\partial \phi_j} = \begin{bmatrix} \frac{\partial u}{\partial a_j} \\ \frac{\partial u}{\partial b_j} \\ \frac{\partial u}{\partial c_j} \end{bmatrix} \tag{2.39}$$

$$K'_j = \alpha(k)\frac{\partial y}{\partial u} \tag{2.40}$$

L'équation (2.38) peut être identifiée à l'équation du filtre de Kalman étendu :

$$\phi_j(k+1) = \phi_j(k) + K(k)e \tag{2.41}$$

où $K(k)$ est le gain de Kalman défini par :

$$K(k) = \frac{P(k)H^T(k)}{H(k)P(k)H^T(k) + R(k)} \tag{2.42}$$

où $H(k)$ est la matrice jacobienne (la matrice d'observation de système), $P(k)$ est la matrice d'estimation de covariance de l'erreur et $R(k)$ est la matrice de covariance du bruit de procédé. En prenant $H^T(k) = (\psi_j^T)$, $P(k) = \lambda_1$ et $R(k) = \lambda_2$, le gain $K(k)$ peut être écrit sous la forme :

$$K(k) = \frac{\lambda_1}{(\psi_j)\lambda_1(\psi_j)^T + \lambda_2}(\psi_j^T) \tag{2.43}$$

Donc l'équation (2.41) devient :

$$\phi_j(k+1) = \phi_j(k) + \frac{\lambda_1}{(\psi_j)\lambda_1(\psi_j)^T + \lambda_2}(\psi_j^T)e \tag{2.44}$$

Par identification entre les équations (2.38) et (2.44), nous obtenons :

$$K'_j = \frac{\lambda_1}{(\psi_j)\lambda_1(\psi_j)^T + \lambda_2} \tag{2.45}$$

Par conséquent, le vecteur des paramètres ϕ_j peut être estimé par la formule suivante :

$$\phi_j(k+1) = \phi_j(k) + K'_j(\psi_j)^T e \tag{2.46}$$

2.7 Conclusion

Dans ce chapitre, nous avons présenté les concepts fondamentaux de la logique floue, des réseaux de neurones ainsi que les systèmes neuro-flous. Dans la premiére partie du chapitre, nous avons présenté les notions de base de la logique floue en décrivant la structure interne d'un système d'inférence flou. Par la suite, nous avons abordé le concept générale des réseaux de neurones artificiels et leur apprentissage par l'algorithme de rétro-propagation. Dans la deuxième partie du chapitre, nous avons présenté quelques structures de base des combinaisons de la logique flou avec les réseaux de neurones, plus particulièrement la structure ANFIS, ainsi que l'algorithme d'apprentissage utilisé dans notre travail. La dernière partie du chapitre a été consacrée pour la synthèse du contrôleur ANFIS proposé.

Dans le chapitre qui suit, nous allons implémenter le contrôleur proposé en utilisant l'environnement Simulink de MATLAB.

Chapitre 3

Implémentation et résultats

3.1 Introduction

Dans ce chapitre, une attention particuliére est accordée au problème d'implémentation des contrôleurs PID et ANFIS développés précédemment afin d'apporter une contribution à la synthèse de la commande en ligne de l'aérotherme, en utilisant l'environnement Simulink du logiciel MATLAB. Ensuite, nous allons présenter les résultats expérimentaux obtenus.

3.2 Implémentation des algorithmes de contrôle

Le contrôle et l'acquisition des données à l'aide des dispositifs électroniques (cartes arduino) interfacés avec un PC ont été largement utilisés dans différentes applications industrielles [26]. Cependant, l'implémentation des algorithmes de contrôle proposés précédemment (PID et ANFIS) sur MATLAB est réalisée selon une méthodologie de conception basée sur l'utilisation de l'environnement Simulink de MATLAB (figure 3.1).

Le logiciel MATLAB-Simulink permet à travers l'arduino Support Package de communiquer avec la carte Arduino [27], puisque ce package contient des blocs et des bibliothèques qui permettent une intégration fluide avec l'Arduino. Il offre également la possibilité de créer des blocs spécifiques dans Simulink, simplifiant ainsi l'intégration et la communication directe avec la carte Arduino.

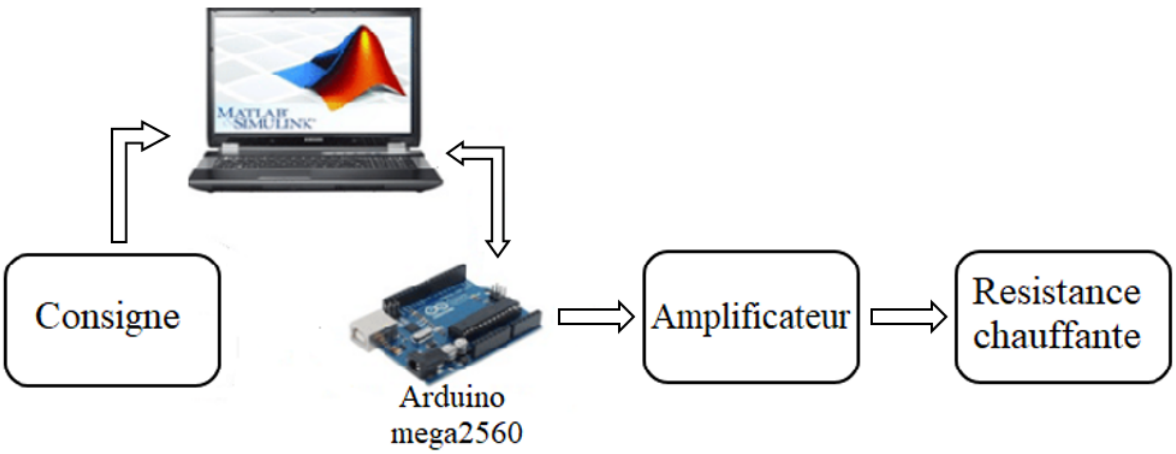


FIGURE 3.1 – Schéma synoptique de l'asservissement à implémenter

Sur MATLAB-Simulink, on ajoute un bloc spécifique à l'Arduino : Analog Input pour lire la température mesurée avec le capteur LM335 connecté à la carte. Un bloc pour le régulateur ANFIS, pour réguler la température, qui prendra comme entrée la différence entre la température mesurée et la température désirée. La sortie du bloc régulateur sera connectée à un bloc spécifiques à l'Arduino : Digital Output pour envoyer les commandes numériques et contrôler la puissance de l'aérotherme via une sortie PWM de l'Arduino.

Cette méthode permet un contrôle en temps réel de la température de l'aérotherme, en utilisant l'environnement convivial de MATLAB et Simulink pour le développement et le déploiement de l'algorithme de commande.

3.3 Mise en œuvre expérimentale

Dans MATLAB-Simulink, la température désirée (Consigne) est définie afin d'être maintenue dans la pièce voulue.

la figure (3.2) illustre le système thermique conçu. Ce dernier a été commandé par une carte Arduino mega2560 suivant deux stratégies de commande PID et neuro-floue qui sont implémentés via le logiciel MATLAB-Simulink. Les températures (T) sont mesurées par un capteur de température LM335. Ces dernières sont aquises par la carte arduino, afin d'être utilisés pour calculer le signal d'erreur. Le signal d'erreur est injecté dans les contrôleurs en vue de calculer les signaux de commande PWM qui seront envoyer par

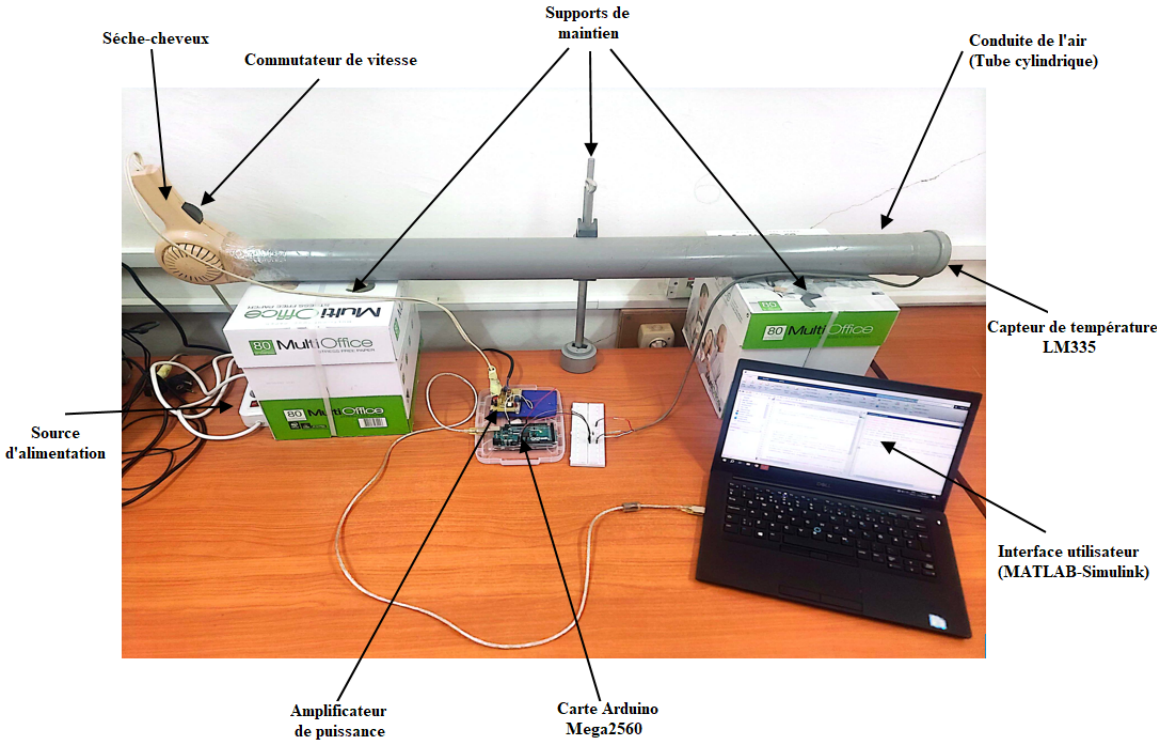


FIGURE 3.2 – Banc de test expérimental

la carte puis amplifier par un amplificateur de puissance basé sur un optotriac de type MOC3041, dans le but de contrôler la puissance de la resistance chauffante.

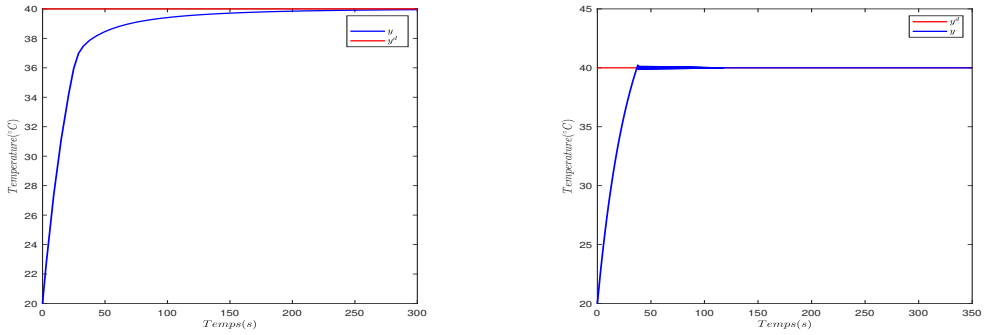
3.4 Résultats expérimentaux

Les résultats expérimentaux ont été menés pour vérifier l'efficacité et les performances du contrôleur ANFIS proposé. Afin de confirmer la robustesse de ce dernier, nous avons comparé dans des conditions identiques les performances du contrôleur neuro-flou à celles du contrôleur classiques PID.

Par contre, avant d'effectuer la commande en ligne sur le procédé, nous avons procédé à une simulation des algorithmes dans différentes situations.

3.4.1 Commande du procédé simulé

Premièrement, on fixe la consigne à 40°C. Les résultats de simulations montrant le suivi des profits de températures sont illustrés dans la figure 3.3.

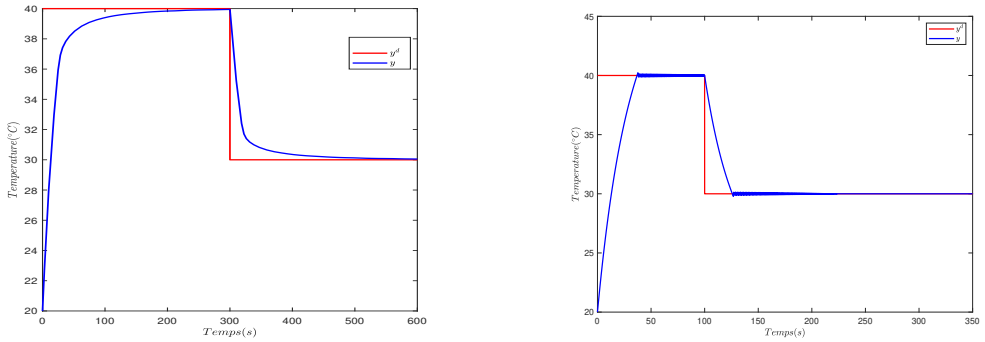


(a) (b)

FIGURE 3.3 – Comportements du suivi des profits de températures avec : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

Pour le contrôleur PID, nous pouvons constater, à partir de la figure 3.3 (a), que la temperature désirée est atteinte aprés un temps de réponse de 200 secondes. Par contre, on remarque que, pour le contrôleur ANFIS, le régime permanent est atteint aprés un temps de réponse <50 secondes (figure 3.3 (b)).

Deuxièmement, une consigne variable (30°C puis 40°C) est injectée. Les résultats de simulations montrant le suivi des profits de températures sont illustrés dans la figure 3.4.



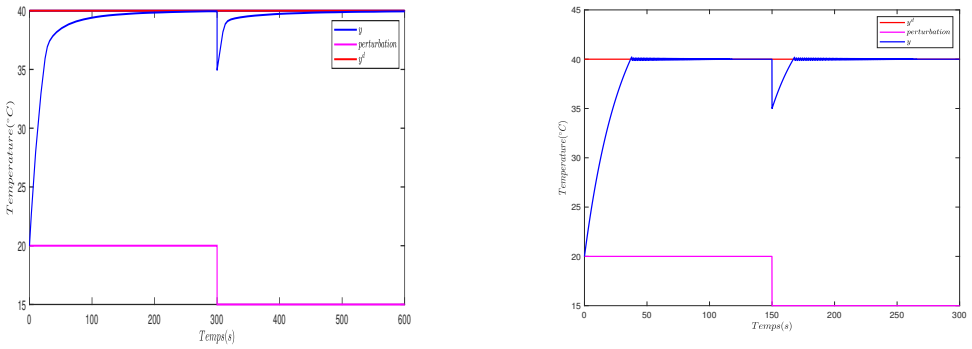
(a) (b)

FIGURE 3.4 – Comportements du suivi des profits de températures pour une consigne variable avec : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

Pour le contrôleur PID, nous pouvons constater, à partir de la figure 3.4 (a), que la sortie y suit la consigne y^d aprés 200 secondes dans chaque palier. Mais, on remarque que, pour le contrôleur ANFIS, la sortie désirée est atteinte aprés un temps de réponse <50 secondes dans chaque palier (figure 3.3 (b)).

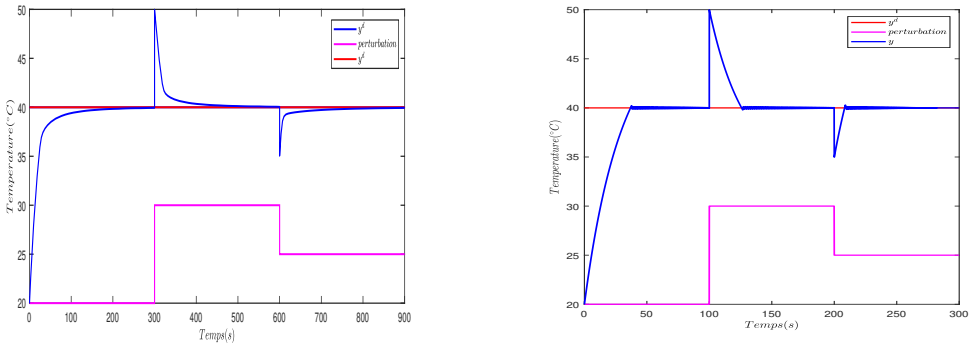
Troisièmement, le test de la robustesse du contrôleur ANFIS proposé, a été effectuée en introduisant différentes perturbations sur notre système.

Les résultats de simulations relatifs aux suivis des températures désirées sont illustrés respectivement dans les figures 3.5, 3.6, 3.7, 3.8 :



(a) (b)

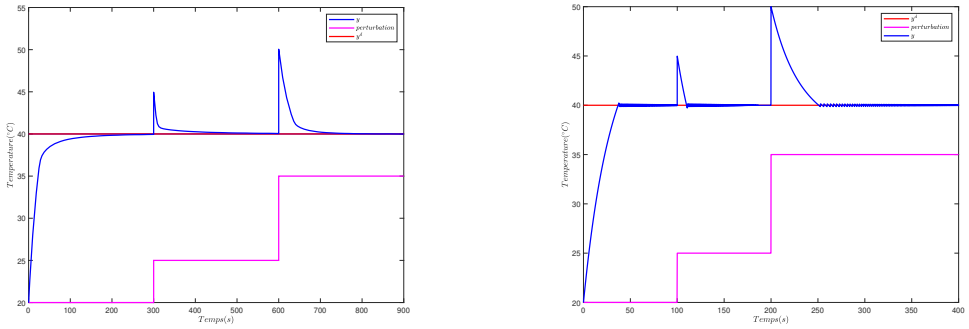
FIGURE 3.5 – Comportements du suivi des profits de températures avec présence de perturbation : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.



(a) (b)

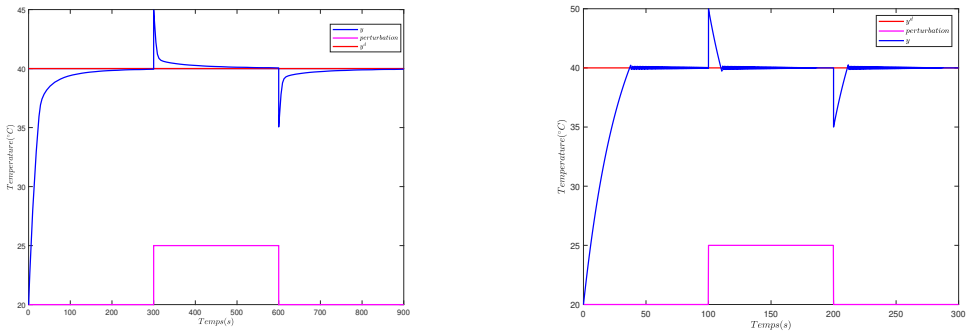
FIGURE 3.6 – Comportements du suivi des profits de températures avec présence de perturbations : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

À partir de ces résultats, nous pouvons constater que le contrôleur ANFIS offre de très bonnes performances de poursuite avec une robustesse acceptable comparativement au contrôleur conventionnel PID. Le contrôleur neuro-flou proposé assure une meilleure efficacité en termes de robustesse et de rapidité car la température désirée est atteinte beaucoup plus rapidement et plus fidéle que le contrôleur classique, et cela grâce à l'algorithme d'apprentissage et la puissance de calcul ainsi que la flexibilité du réseau neuro-flou.



(a) (b)

FIGURE 3.7 – Comportements du suivi des profits de températures avec présence de perturbations : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.



(a) (b)

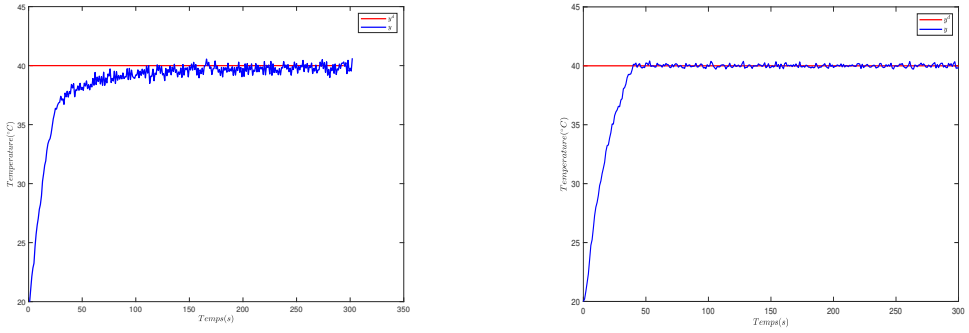
FIGURE 3.8 – Comportements du suivi des profits de températures avec présence de perturbations : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

3.4.2 Commande du procédé en ligne

Comme dans le cas de la simulation, on fixe la consigne à 40°C. Les résultats expérimentaux obtenus montrant le suivi des profits de températures sont illustrés dans la figure 3.9.

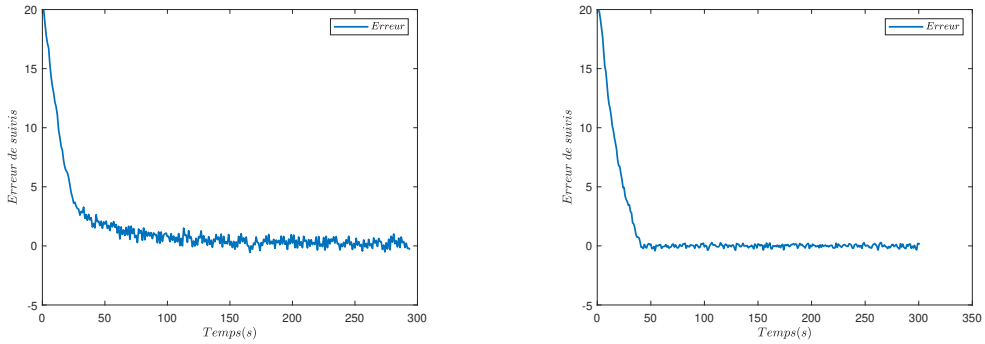
L'analyse de la figure 3.9, nous permet de constater que les résultats expérimentaux sont presque identiques avec ceux obtenus en simulation (figure 3.3).

Pour le contrôleur PID, nous pouvons remarquer, à partir de la figure 3.9(a), que la température désirée est atteinte aprés un temps de réponse de 175 secondes ; Cependant, une petite erreur de suivi peut être remarquée, comme indiqué dans la figure 3.10(a). En revanche, on constate que, pour le contrôleur ANFIS, la consigne désirée est atteinte aprés seulement 50 secondes (figure 3.9(b)) avec une très petite erreur, comme indiqué dans la



(a) (b)

FIGURE 3.9 – Comportements du suivi des profits de températures avec : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.



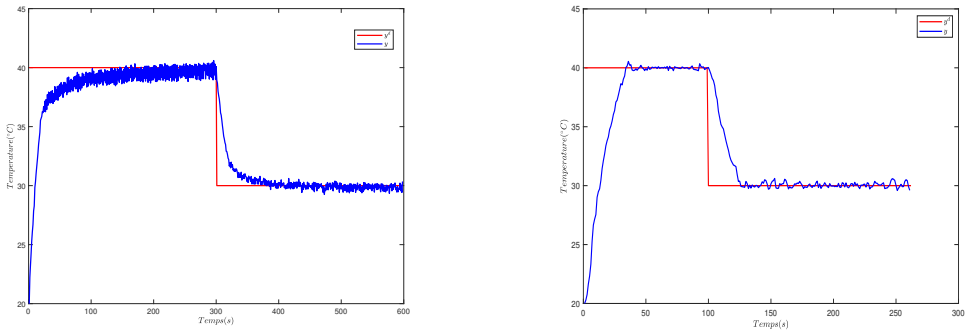
(a) (b)

FIGURE 3.10 – Comportements des erreurs du suivis des profits de températures avec : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

figure 3.10(b). On peut constater aussi que le signal de sortie est plus lisse dans le cas du contrôleur ANFIS.

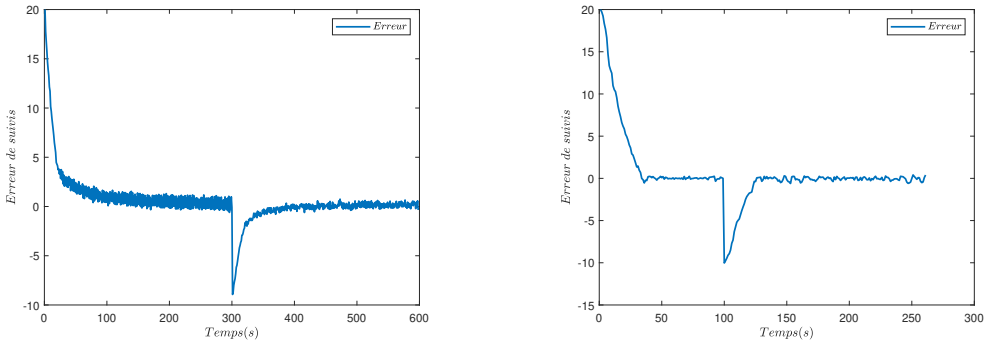
Par la suite, une consigne variable (30°C puis 40°C) est injectée. Les résultats expérimentaux obtenus montrant le suivi des profits de températures sont illustrés dans la figure 3.11.

Pour le contrôleur PID, nous pouvons constater, à partir de la figure 3.11(a), que la sortie y suit la consigne y^d aprés 175 secondes dans chaque palier ; Néanmoins, une petite erreur de suivi peut être remarquée, comme indiqué dans la figure 3.12(a). Par contre, pour le contrôleur ANFIS, la sortie désirée est atteinte aprés un temps de réponse $<$50 secondes dans chaque palier (figure 3.3(b)), avec une très petite erreur, comme indiqué dans la figure 3.12(b). On peut constater aussi que le signal de sortie est plus lisse dans



(a) (b)

FIGURE 3.11 – Comportements du suivi des profits de températures pour une consigne variable avec : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.



(a) (b)

FIGURE 3.12 – Comportements des erreurs du suivi des profits de températures pour une consigne variable avec : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

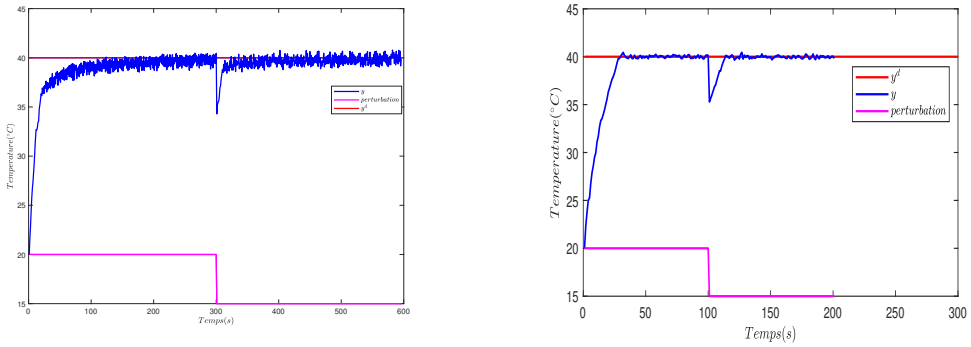
le cas du contrôleur ANFIS.

Dans le but de tester la robustesse du contrôleur ANFIS proposé, on a introduit différentes perturbations sur notre système, comme suit :

Perturbations sur Simulink

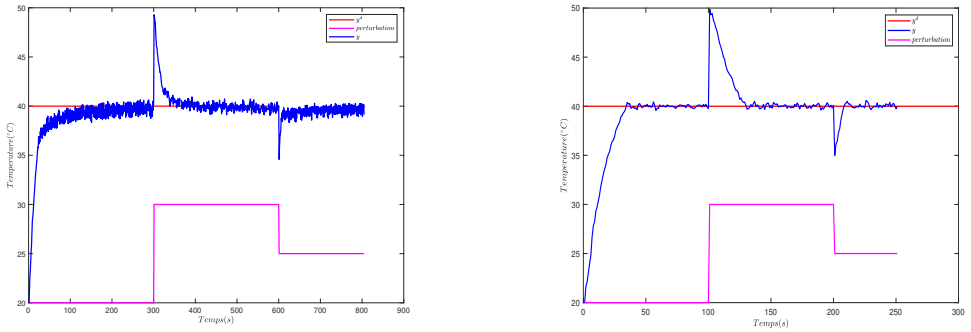
On a injecté des perturbations directement sur simulink afin de perturber la sortie. Les résultats expérimentaux relatifs aux suivis des températures désirées sont illustrés respectivement dans les figures 3.13, 3.14, 3.15, 3.16 :

À partir de tous ces résultats graphiques on peut remarquer que les résultats obtenus en simulation sont presque identiques avec ceux obtenus expérimentalement. Et on constate que le contrôleur ANFIS offre de meilleures performances en termes de suivi et



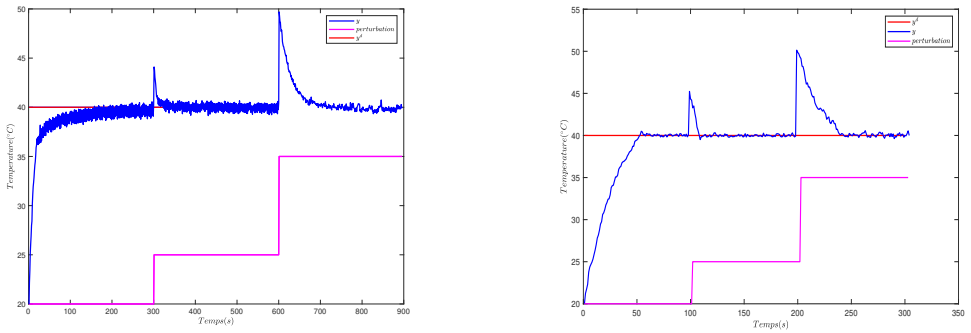
(a) (b)

FIGURE 3.13 – Comportements du suivi des profits de températures avec présence de perturbation : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.



(a) (b)

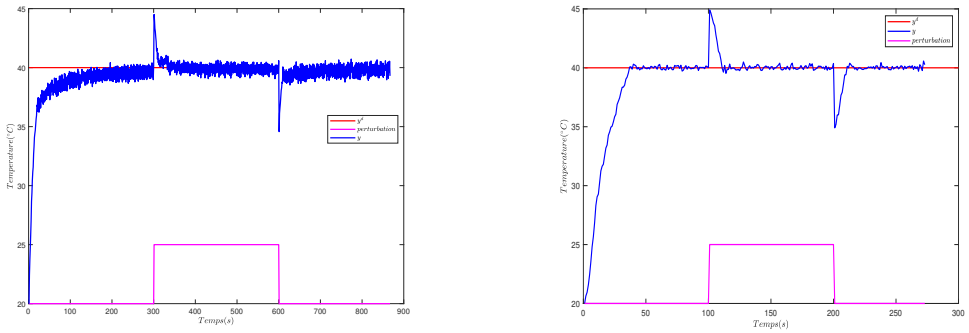
FIGURE 3.14 – Comportements du suivi des profits de températures avec présence de perturbations : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.



(a) (b)

FIGURE 3.15 – Comportements du suivi des profits de températures avec présence de perturbations : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

de robustesse par rapport au contrôleur PID conventionnel dans le même cas. Le contrôleur neuro-flou proposé est plus efficace en termes de rapidité et de fidélité, atteignant la température désirée plus rapidement et de manière plus précise que le contrôleur classique, grâce à l'algorithme d'apprentissage, la puissance de calcul et la flexibilité du réseau



(a) (b)

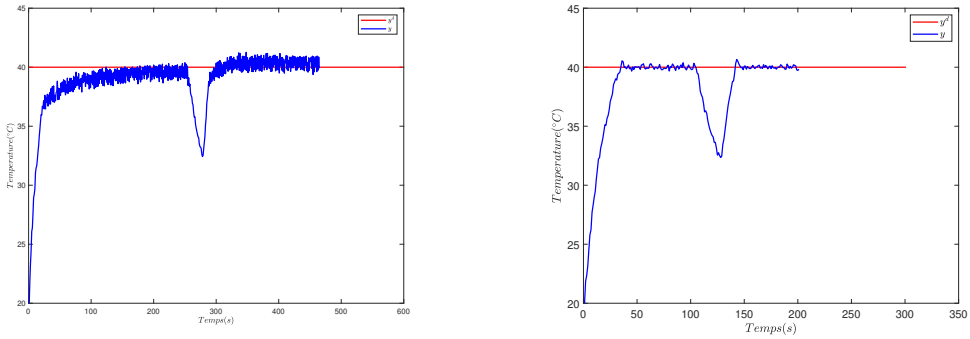
FIGURE 3.16 – Comportements du suivi des profits de températures avec présence de perturbations : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

neuro-flou.

Perturbations physiques(réelles)

Premièrement, les perturbations sont provoquées par diminution réelle de la température, obtenues par une modification momentanée de la puissance de chauffage à l'aide du redresseur à diode commutable.

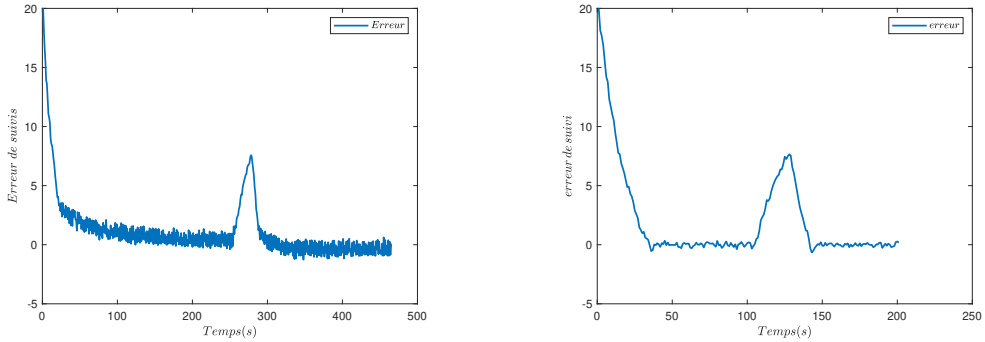
Les résultats expérimentaux sont illustrés respectivement dans la figure (3.17)



(a) (b)

FIGURE 3.17 – Comportements du suivi des profits de températures avec présence de perturbation réelle : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

Pour le contrôleur PID, nous pouvons constater, à partir de la figure 3.17(a), que la sortie y suit la consigne y^d aprés un temps de réponse considérable ; Néanmoins, une erreur de suivi peut être remarquée, comme indiqué dans la figure 3.18(a). Par contre, pour le contrôleur ANFIS, la sortie désirée est atteinte rapidement (figure 3.17(b)), avec une très

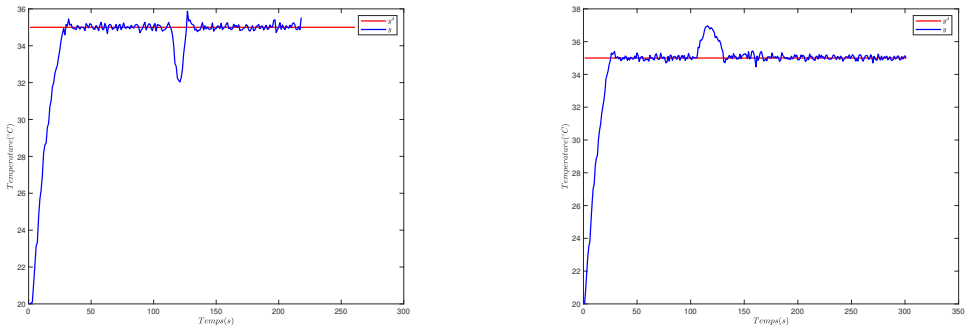


(a) (b)

FIGURE 3.18 – Comportements des erreurs du suivi des profits de températures avec présence de perturbation réelle : (a) Le contrôleur PID, (b) Le contrôleur ANFIS.

petite erreur, comme indiqué dans la figure 3.18(b). On peut constater que le signal de sortie est beaucoup plus lisse dans le cas du contrôleur ANFIS.

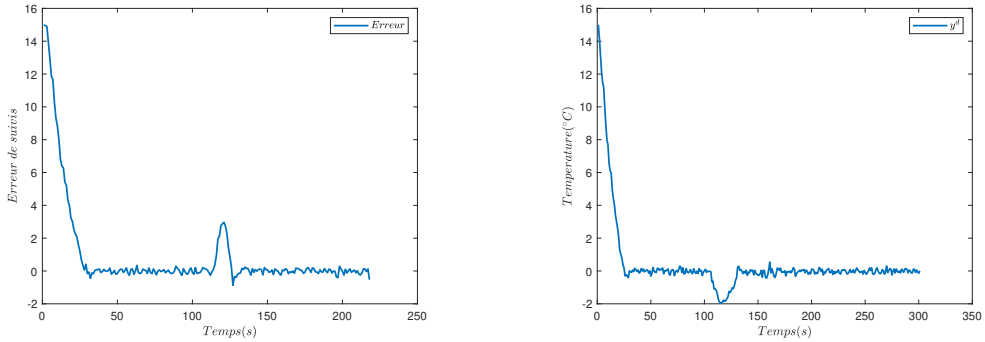
Deuxièmement, on fixe la consigne à 35°C, les perturbations sont provoquées par des variations de la température au niveau du capteur, obtenues en injectant respectivement deux flux d'air chaud et froid produits par un sèche-cheveux externe à une distance $d = 60$ centimètres. Les résultats expérimentaux sont illustrés dans la figure (3.19).



(a) (b)

FIGURE 3.19 – Comportements du suivi des profits de températures avec présence de perturbation réelle avec le contrôleur ANFIS : (a) Diminution de température, (b) Augmentation de température.

Ces résultats expérimentaux montrant les performances de poursuite des températures désirées et de régulation dans le cas ou des perturbations externes s'applique au système (figure (3.19)), avec des très petites erreurs, comme indiqué dans la figure 3.20. On peut constater que le contrôleur ANFIS proposé dans ce travail offre de trés bonnes performances, en termes de poursuite des températures désirées et de rejet des perturbations.



(a) (b)

FIGURE 3.20 – Comportements des erreurs du suivi des profits de températures avec présence de perturbation réelle avec le contrôleur ANFIS : (a) Diminution de température, (b) Augmentation de température.

On peut conclure alors qu'il est trés rapide, performant et robuste.

Ces résultats peuvent être expliqués par le contrôleur ANFIS proposé, qui applique un algorithme de contrôle intelligent, peut d'une part compenser les non-linéarités du système thermique, et d'autre part s'adapter rapidement aux variations dynamiques du système lorsque la température varie brusquement, et cela en ajustant rapidement les paramètres de la conséquence du réseau neuro-flou en temps réel. De plus, la méthodologie de conception du contrôleur, utilisée dans ce travail, a conduit à des résultats satisfaisants en termes de précision et de stabilité. L'utilisation de l'environnement MATLAB-Simulink, avec son package Arduino Support Package, a permis de développer un algorithme robuste, offrant une grande précision tout en optimisant le temps d'exécution.

3.5 Conclusion

Dans ce chapitre, nous avons présenté la démarche suivie pour concevoir et implémenter les contrôleurs proposés sur MATLAB-Simulink, ainsi que les résultats obtenus lors de la commande du système thermique. la première partie du chapitre détaille la méthodologie adoptée, basée sur l'utilisation de MATLAB/Simulink et du Arduino Support Package, qui a permis de réduire considérablement le temps de conception de l'algorithmes et d'obtenir un contrôleur précis et robuste. Ensuite, on a expliqué brièvement la mise en œuvre du système conçu. La dernière partie est consacrée à la présentation des résultats

expérimentaux obtenus par les commandes PID et neuro-floue. Ces résultats mettent en évidence l'efficacité de l'approche ANFIS par rapport à la méthode conventionnelle (PID).

Conclusion générale

Le travail présenté dans le cadre de ce projet de fin d'étude a pour objectif d'aborder les techniques avancées d'intelligence artificielle, en se concentrant particulièrement sur les techniques neuro-floues appropriées à la commande des systèmes thermiques, et plus spécifiquement des aérothermes. Ces méthodes représentent une alternative prometteuse aux approches classiques, en répondant aux exigences de performance et de robustesse tout en surmontant les contraintes inhérentes à la commande de ces systèmes. L'aérotherme, en raison de sa nature non linéaire et de sa complexité, pose des défis significatifs pour le commander. A cet effet, il est essentiel que les contrôleurs soient adaptatifs afin de maintenir la stabilité et la transparence du système. Dans cette optique, nous avons développé un contrôleur neuro-flou adaptatif de type ANFIS (Adaptive Neuro-Fuzzy Inference System). La mise en œuvre de ce contrôleur s'appuie sur l'utilisation d'une carte Arduino, qui a permis d'acquérir puis transmettre le plus fidèlement possible les informations provenant du capteur de température. Elle permis aussi d'envoyer le signal de commande, générer sur simulink, pour commander la puissance la résistance chauffante.

La première partie a été consacrée à la présentation de l'aérotherme et son principe de fonctionnement, ainsi que une déscription détaillé des differents organes du procédé bouclé. Enfin, nous avons expliqué les techniques utilisées sur l'environnement MATLAB-Simulink pour l'identification du système (System Identification Toolbox) et la synthèse du contrôleur PID (PID Tuner).

Dans la deuxième partie de ce travail, nous avons présenté des techniques d'intelligence artificielle, Nous avons d'abord introduit les concepts fondamentaux de la logique floue, puis ceux des réseaux de neurones artificiels et des réseaux neuro-flous. Par la suite,

nous avons développé un régulateur neuro-flou adaptatif de type ANFIS pour le système thermique conçu. Cette approche intègre les avantages de la logique floue et des réseaux de neurones en un seul réseau. La capacité d'apprentissage des réseaux de neurones a été exploitée pour l'adaptation en ligne des paramètres du réseau neuro-flou, tandis que la logique floue a fourni une puissante capacité d'inférence du raisonnement, permettant ainsi de concevoir un contrôleur hautement performant, en utilisant un algorithme d'apprentissage basé sur le filtre de Kalman étendu, ce qui nous a permis d'atteindre les performances désirées en un temps très court.

La troisième partie de ce travail a été dédiée à la présentation des techniques utilisées pour la mise en œuvre du contrôleur proposé sur arduino. La conception et l'implémentation de l'algorithme de contrôle sur l'environnement Simulink de MATLAB en utilisant le package de communication avec la carte arduino "Arduino Support Package". Les résultats expérimentaux ont démontré l'efficacité du contrôleur neuro-flou proposé. En effet, ce dernier a surpassé le contrôleur PID classique en termes de poursuite, de stabilité et de robustesse face aux variations dynamiques du système. Ces performances supérieures s'expliquent par la capacité du régulateur ANFIS à s'adapter aux variations dynamiques du système ainsi qu'aux perturbations externes, telles que les changements brusques de température.

Comme perspectives à ce travail, il serait intéressant de proposer un autre régulateur ANFIS, en complément de celui déjà proposé, pour contrôler la vitesse du moteur qui traine le ventilateur du séche cheveux. Cela permettrait d'améliorer à la fois la rapidité et la robustesse du système.

Bibliographie

[1] R. Dupuy, *Henri Arquembourg*, Association des centraliens, article No 11juin1952.

[2] A. B. Patil, A. V. Salunkhe, *Adaptive Neuro Fuzzy Controller for Process Control System*, 2008 IEEE Region 10 Colloquium and the Third International Conference on Industrial and Information Systems, pp 1-5, Kharagpur, INDIA December 8-10, 2008.

[3] P. P. Bhogle, B. M. Patre, L. M. Waghmare, V. M. Panchade, *Neuro Fuzzy Temperature Controller*, Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation, Harbin, China, August 5-8, 2007.

[4] H. Khati, *Commande d'une Architecture de Téléopération par la Carte FPGA*, thèse de doctorat, département Automatique, Université Mouloud Maameri Tizi-Ouzou, 2020.

[5] S. Guermah, *«Commande adaptative d'un aérotherme par calculateur».* thèse de magister, département Automatique, Université Mouloud Maameri Tizi-Ouzou, 1998.

[6] S. Alkama, *Cours : capteurs et chaines de mesure* , Département Automatique, Université de Mouloud Maameri Tizi-Ouzou, 2022.

[7] Texas instrument :*LM335 datasheet, product information and support.*

[8] BookSprint, *flossmanuals arduino,* 2011.

[9] E. Bartmann, *le grand livre d'arduino*, 2eme édition, 2015.

[10] J. M. Hughes, *Arduino : le guide complet*, 2016.

[11] D. Saputra, A. Ma'arif, H. Maghfiroh, P. Chotikunnan, *Design and Application of PLC-based Speed Control for DC Motor Using PID with Identification System and*

MATLAB Tuner, International Journal of Robotics and Control Systems Vol.3, No.2, pp. 233-244, 2023.

[12] R. Mellah, *Contribution à la commande adaptative neurofloue. Application à la Robotique.* thèse doctorat, Université des Sciences et de la Technologie Houari Boumediene, Alger, Mai 2006.

[13] L. A. Zadeh, *Fuzzy Sets*, Information and Control, Vol. 8, pp. 338-353, 1965.

[14] S. Zeghlache, *Cours : Commande Intelligente*, Département de Génie Electrique, Université Mohammed boudiaf - M'sila.

[15] K. Houacine, *commande neuro-floue d'une machine asynchrone dans une chaine de propulsion d'un véhicule électrique*, thèse de doctorat, département Automatique, Université Mouloud Maameri Tizi-Ouzou, 2016.

[16] N. Talbi, *Conception des Systèmes d'Inférence Floue par des Approches Hybrides : Application pour la Commande et la Modélisation des Systèmes Nonlinéaires*, thèse de doctorat, Université de Constantine 1, 2014.

[17] R. Fuller, *Neural Fuzzy Systems*, Abo Akademi University, Turku, 1995.

[18] L. Cherroun *Navigation Autonome d'un Robot Mobile par des Techniques Neuro-Floues*, thèse de doctorat, Université Mohamed Khider - Biskra, 2014

[19] J. Mc Culloch, W. Pitts *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biology, Vol 52, No l/2, pp 99-115, 1990.

[20] I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, S. F. D. R. Alves, *Artificial Neural Networks, A Practical Course*, Springer, 2016.

[21] C. Touzet, *Les réseaux de neurones artificiels, introduction au connexionnisme, Cours, Exercices et Travaux pratiques*, Juillet 1992.

[22] JSR. Jang, *ANFIS : Adaptive-Network-Based Fuzzy Inference System,* IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 3, pp.665-685, 1993.

[23] L. Song, H. Chen, H. Wu, Z. Yang, *Adaptive Fuzzy Controller for Air Conditioning System* D. Jin and S. Lin (Eds) : Advances in Mechanical and Electronic Engineering, Lecture Notes in Electrical Engineering, vol 176, pp 255–260, Springer, Berlin, Heidelberg, 2012.

[24] M. Khalid, S. Omatu, *A neural network controller for a temperature control system*, IEEE Control Systems Magazine, vol 12, no 3, pp 58-64, June 1992.

[25] R. E. Kalman, *A New Approach to Linear Filtering and Prediction Problems*, Transactions of the ASME - Journal of Basic Engineering Vol. 82, p 35-45 , 1960.

[26] H. Hamil, M. S. Azzaz, S. Sakhi, R. Kaibou, A. Hamil. *Design of a thermal stabilizer based on USB data acquisition and control using LabVIEW and PIC microcontroller*, International Conference on Advances in Electronics, Control and Communication Systems (ICAECCS), pp 1-6 Blida, Algeria, 2023.

[27] MATLAB & SIMULINK R2024a. *Get Started with Arduino Hardware*, mathworks, MATLAB, 2024.