République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOULOUD MAMMERI DE TIZI-OUZOU

FACULTÉ DU GÉNIE ÉLECTRIQUE ET D' INFORMATIQUE
DÉPARTEMENT D'AUTOMATIQUE

Mémoire de Fin d'Études de MASTER ACADÉMIQUE

Domaine : **Sciences et Technologies**
Filière : **Automatique**
Spécialité : **Automatique et informatique industrielle.**

Présenté par
AMOURA Mounia Aini

Thème

Implémentation d'une commande Neuro-floue à une architecture de télé opération position-position en présence d'un retard fixe de transmission

Mémoire soutenu publiquement le 30 /06/ 2024 devant le jury composé de :

M	**HAMMOUCHE S**
Grade	,UMMTO , Président
M	**MESSAR Y**
Grade	, UMMTO, Encadrant
M	**MELLAH R**
Grade	, UMMTO, Co-encadrant
M	**SERSOUR**
Grade	,UMMTO, Examinateur
M	**AMROUNE**
Grade	, UMMTO, Examinateur

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La téléopération est une technologie permettant à un opérateur humain de contrôler à distance un système robotique en temps réel, en reproduisant les mouvements ou les commandes effectuées par l'opérateur. Cette technologie trouve des applications dans divers domaines tels que la médecine, l'industrie, et les environnements hostiles ou dangereux, où la présence humaine directe est impraticable ou risquée [?]. Une des configurations les plus courantes est la téléopération de type maître-esclave, où le dispositif maître (opéré par l'humain) envoie des commandes au dispositif esclave (robot), qui exécute les actions correspondantes [?].

Cependant, un des défis majeurs de la téléopération est le retard de transmission entre le maître et l'esclave. Ce retard peut être causé par la distance géographique, les contraintes de communication réseau, ou les limitations techniques des systèmes de transmission [?]. Le retard fixe peut dégrader les performances du système de téléopération en introduisant des instabilités et en réduisant la précision des opérations. Pour remédier à ce problème, diverses méthodes de compensation de retard ont été proposées dans la littérature, mais elles présentent souvent des limitations en termes de robustesse et de complexité [?].

L'approche neuro-floue combine les avantages des réseaux de neurones artificiels et des systèmes flous pour traiter les incertitudes et les non-linéarités des systèmes complexes [?]. Les réseaux de neurones permettent d'apprendre et de généraliser à partir de données, tandis que les systèmes flous modélisent les connaissances imprécises ou incomplètes à l'aide de règles linguistiques [?]. L'application des techniques neuro-floues à la téléopération promet d'améliorer la robustesse et la flexibilité des systèmes de contrôle en présence de retards de transmission.

La commande neuro-floue dans une architecture de téléopération position-position vise à synchroniser les mouvements du maître et de l'esclave malgré les retards de transmission. Cette approche consiste à ajuster en temps réel les paramètres du contrôleur en fonction des erreurs de position et des retards observés, en utilisant des algorithmes d'apprentissage basés sur la descente du gradient ou d'autres techniques d'optimisation [?].

L'objectif de ce mémoire est de concevoir et de mettre en œuvre une commande neuro-floue pour une architecture de téléopération position-position en présence d'un retard fixe de

transmission. Nous allons d'abord analyser les défis associés aux retards de transmission et leurs impacts sur la performance des systèmes de téléopération. Ensuite, nous introduirons les concepts fondamentaux des réseaux de neurones et des systèmes flous, ainsi que leurs applications dans le domaine du contrôle robotique. Enfin, nous présenterons la méthodologie proposée pour la conception du contrôleur neuro-flou, suivie de son implémentation et de l'évaluation des performances à travers des simulations et des expériences pratiques [**?**].

Ce mémoire est structuré comme suit : le chapitre 1 présente une revue de la littérature sur les systèmes de téléopération, les techniques de compensation de retard . Le chapitre 2 décrit la méthodologie de développement du contrôleur neuro-flou, incluant les algorithmes d'apprentissage et les critères de performance. Le chapitre 3 présente les résultats des expériences, suivis d'une discussion sur les performances obtenues et les améliorations potentielles.

Chapitre 1

Description d'une architecture de téléopération position-position en présence d'un retard de transmission fixe

1.1 Introduction

La téléopération, ou téléopérabilité, est une technique qui permet à un opérateur de contrôler à distance un système robotique . L'un des défis majeurs de la téléopération est la transmission de l'information entre l'opérateur et le robot, en particulier lorsque des retards sont présents dans le système. Dans ce chapitre, nous abordons le problème spécifique du retard fixe de transmission dans une architecture de téléopération position-position, en utilisant une commande neuro-floue pour atténuer les effets de ce retard.

1.2 Définition

1.2.1 Téléopération

La téléopération en robotique est un procédé où un opérateur contrôle à distance un robot ou un système robotique par le biais d'une interface utilisateur, permettant ainsi de commander ses mouvements et actions tout en restant éloigné physiquement de l'appareil [10]. Ce concept est largement utilisé dans des environnements où la présence humaine est difficile ou dangereuse, offrant une solution sécurisée et efficace pour diverses applications industrielles, médicales et militaires[11, 12].

1.2.2 Système de téléopération

Un système de téléopération permet à un opérateur humain de contrôler à distance un dispositif ou un robot. L'opérateur envoie des commandes de contrôle au dispositif distant, qui exécute ces commandes et renvoie des données de retour à l'opérateur. Ces données de retour peuvent inclure des informations sur l'état du dispositif, son environnement, ou d'autres informations pertinentes pour l'opération en cours.

Pour fonctionner efficacement, un système de téléopération doit avoir une communication fiable entre l'opérateur et le dispositif distant, une latence minimale pour assurer des réponses en temps réel, des capteurs et des actionneurs appropriés pour permettre le contrôle du dispositif, ainsi qu'une interface utilisateur intuitive pour faciliter le contrôle par l'opérateur[9].

En résumé, un système de téléopération permet à un opérateur de contrôler à distance un dispositif, ce qui ouvre de nombreuses possibilités d'application dans divers domaines.

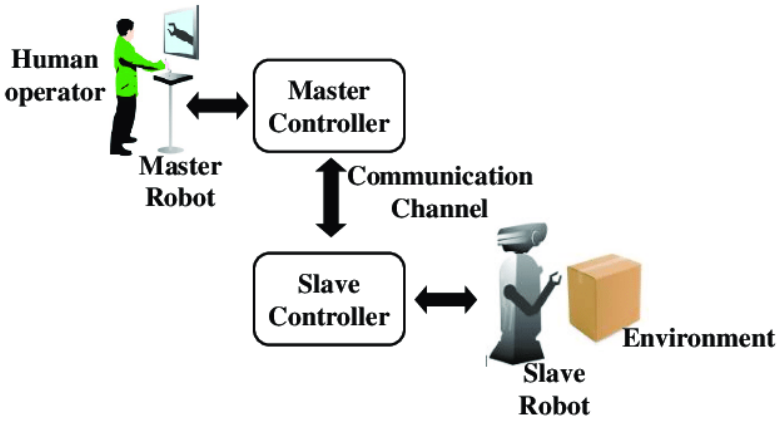

FIGURE 1.1 – Représentation générale d'un système de téléopération bilatérale.

1.3 Architecture d'un système de téléopération

La téléopération en robotique est un système sophistiqué qui permet à un opérateur humain de contrôler à distance un robot ou un système robotique. Ce système implique plusieurs composantes essentielles [13] [14] :

Interface utilisateur avancée

Une interface conviviale est cruciale pour permettre à l'opérateur de communiquer efficacement avec le robot. Cette interface peut prendre la forme d'un joystick, d'une manette, d'un ordinateur ou même de dispositifs de réalité virtuelle pour une immersion accrue.

Transmission de données

Les commandes de l'opérateur doivent être transmises au robot, tandis que les données provenant des capteurs du robot doivent être renvoyées à l'opérateur. Cela nécessite des systèmes de transmission de données robustes et fiables, souvent basés sur des technologies sans fil ou filaires.

Capteurs et caméras

Le robot est équipé de capteurs et de caméras pour fournir à l'opérateur des informations en temps réel sur l'environnement dans lequel le robot évolue. Cela permet à l'opérateur de prendre des décisions éclairées lors de la téléopération.

Système de contrôle

Un système de contrôle avancé est nécessaire pour traduire les commandes de l'opérateur en mouvements et actions précis du robot. Ce système doit prendre en compte la cinématique du robot, les contraintes environnementales et les objectifs de la mission.

Sécurité et redondance

Étant donné que la téléopération peut souvent se dérouler dans des environnements dangereux ou imprévisibles, des mesures de sécurité robustes sont essentielles. Cela peut inclure des fonctions d'arrêt d'urgence, des systèmes de surveillance à distance et des mécanismes de redondance pour assurer le bon fonctionnement du système en cas de défaillance.

Ensemble, ces composantes permettent à la téléopération en robotique d'être utilisée dans une variété d'applications, telles que l'exploration spatiale, l'inspection d'infrastructures, les interventions chirurgicales à distance et les opérations de sauvetage en cas de catastrophe.

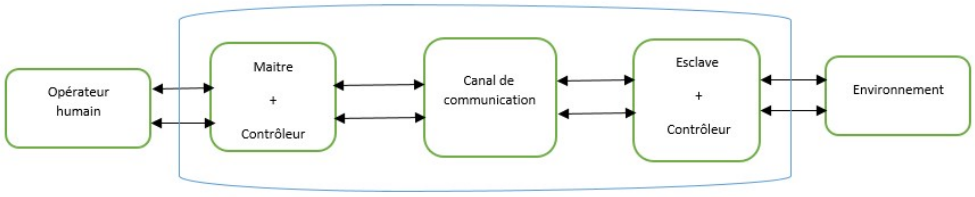

FIGURE 1.2 – Représentation d'un système de téléopération sous forme d'un réseau.

1.4 Représentation mathématique d'un système de télé-opération

1.4.1 Représentation d'un système maître-esclave à un degré de liberté

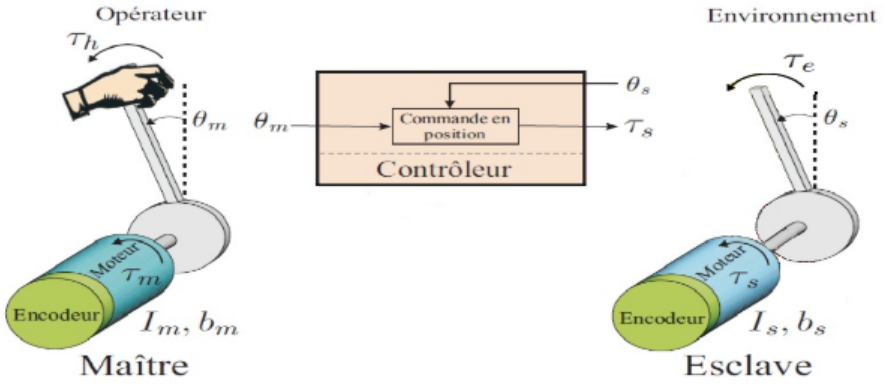

FIGURE 1.3 – système de téléopération à 1 degré de liberté (contrôle position- position

Les systèmes de téléopération sont souvent constitués de dispositifs à plusieurs degrés de liberté . Cependant, un modèle simplifié à un degré de liberté a été défini dans la littérature pour faciliter l'étude du problème. Mathématiquement, un système de téléopération à un degré de liberté comporte deux sous-systèmes, maître et esclave, qui échangent des signaux de vitesses, de forces (couples) et d'accélérations. La dynamique des dispositifs maître et esclave est décrite par les équations suivantes [9] :

$$\begin{cases} I_m\ddot{\theta}_m + I_m\dot{\theta}_m & = \tau_m + \tau_h \\ I_s\ddot{\theta}_s + I_s\dot{\theta}_s & = \tau_s - \tau_e \end{cases} \tag{1.1}$$

où I_m et b_m, I_s et b_s représentent respectivement les masses et les frottements visqueux des robots maître et esclave. θ_m et θ_s désignent les positions du maître et de l'esclave, τ_h est la force (couple) appliquée par l'opérateur sur le dispositif maître, et τ_e est la force (couple) exercée par l'environnement sur l'esclave. τ_m et τ_s sont les forces (couples) fournies par les actionneurs pour commander les mouvements du maître et de l'esclave.

La dynamique de l'objet en interaction avec le robot esclave lors d'un contact rigide est donnée par l'équation suivante :

$$\tau_s = I_w\ddot{\theta}_s + b_w\dot{\theta}_s + C_w\theta_s \tag{1.2}$$

où I_w, b_w et C_w représentent respectivement la masse, les frottements visqueux et la rigidité de l'objet.

La dynamique de l'opérateur humain est représentée par l'équation suivante :

$$\tau_{\text{op}} - \tau_m = I_{\text{op}}\ddot{\theta}_m + b_{\text{op}}\dot{\theta}_m + C_{\text{op}}\theta_m \tag{1.3}$$

où M_op, B_op et C_op représentent respectivement la masse, les frottements visqueux et la rigidité de l'opérateur, et F_op désigne la force générée par les muscles de l'opérateur.

1.4.2 Représentation hybride

Un système de téléopération peut être représenté par un réseau à deux ports, inspiré des réseaux électriques, qui relie la force (ou couple) de l'opérateur τ_h et sa position θ_m aux variables de force τ_e et de position θ_s du manipulateur esclave [9].

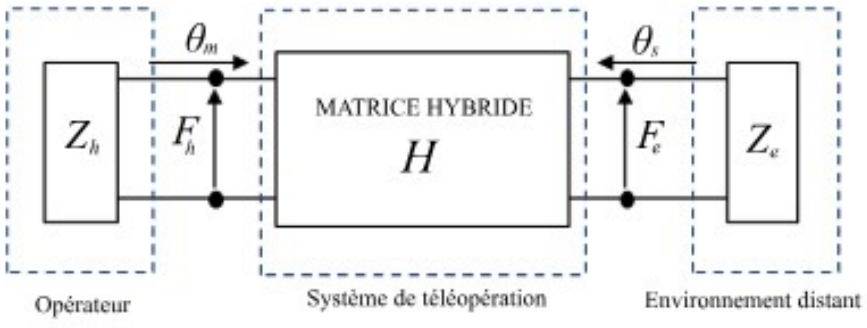

FIGURE 1.4 – Représentation d'un système de téléopération par un modèle à deux ports.

La figure 1.4 illustre un système de téléopération modélisé par un réseau à deux ports sous une forme hybride, mettant en relation l'opérateur humain, caractérisé par une impédance Z_h, avec l'environnement distant d'impédance Z_e. Ce réseau est décrit par la relation suivante, proposée par Hannaford [9] :

$$\begin{bmatrix} F_h \\ \theta_s \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} \theta_m \\ F_e \end{bmatrix}$$

La matrice H est appelée matrice d'admittance hybride, et elle caractérise complètement le système de téléopération, y compris la dynamique du maître et de l'esclave, les contrôleurs

et les communications. Les paramètres h_{ij} de la matrice hybride H définissent quatre critères de performance pour évaluer la transparence d'un système de téléopération :

1. Pour $F_e = 0$ (mouvement libre : pas d'interactions entre l'esclave et l'environnement) :

   $h_{11} = Z_{\min} = \frac{F_h}{\theta_m}$: Représente l'impédance minimale ressentie par l'opérateur en mouvement libre. Idéalement, elle doit être égale à zéro.

   $h_{21} = X_T = \frac{\theta_m}{\theta_s}$: Est le suivi de position en mouvement libre. Idéalement, ce terme doit être égal à 1.

2. Pour $\theta_s = 0$ (contact rigide) :

   $FT = -\frac{F_e}{F_h} = \frac{h_{21}}{h_{12}h_{21} - h_{11}h_{22}}$: Représente le suivi de force lors d'un contact rigide.

   $Z_{\max} = \frac{F_h}{\theta_m} = \frac{h_{11}h_{22} - h_{12}h_{21}}{h_{22}}$: Est l'impédance maximale transmissible à travers le système de téléopération lors d'un contact rigide. Idéalement, elle doit tendre vers l'infini.

1.5 Caractéristiques d'un système de téléopération bilatérale

Un système de téléopération bilatérale est caractérisé par deux critères essentiels :

1.5.1 Stabilité

La stabilité est cruciale pour garantir un fonctionnement sûr et efficace de la téléopération. Elle se définit par la capacité du système à assurer que tout déplacement ou effort limité exercé par l'opérateur sur l'interface maître entraîne un déplacement ou un effort borné réalisé par le robot esclave dans l'environnement distant. Cependant, étudier la stabilité est complexe en raison de l'interaction avec un opérateur humain dont le comportement est incertain, ainsi qu'avec un environnement caractérisé par des non-linéarités importantes. De plus, les délais de transmission causés par le canal de communication peuvent entraîner une instabilité du système. Par conséquent, les lois de commande du système de téléopération doivent être robustes face à ces incertitudes dynamiques du système maître-esclave et aux retards de transmission pour assurer la stabilité [16].

1.5.2 transparence

La transparence est l'autre critère essentiel d'un système de téléopération, se référant à sa capacité à refléter fidèlement les interactions entre l'esclave et l'environnement sur l'opérateur. L'objectif est que l'opérateur ressente les efforts comme s'il effectuait directement la tâche à la place de l'esclave. Une transparence parfaite est définie lorsque les réponses de position et de force du robot maître sont égales à celles du robot esclave, assurant ainsi un suivi parfait des forces et des positions . Cette égalité peut être exprimée par les équations suivantes [9] :

$$\begin{cases} F_h(t) & = -F_e(t) \\ \theta_s(t) & = \theta_m(t) \end{cases} \tag{1.4}$$

La matrice hybride idéale correspondante est définie comme suit :

$$H_{\text{ideale}} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{1.5}$$

1.6 Architecture téléopération position-position

La téléopération en mode position-position représente un schéma de contrôle bilatéral dans le domaine de la robotique, où l'opérateur exerce un contrôle direct sur la position d'un système distant en fonction de sa propre position. Cette approche nécessite l'intégration de capteurs de position pour la mesure des positions de l'opérateur et du système distant, ainsi que l'utilisation de dispositifs de commande pour la génération de signaux de commande en fonction de ces positions mesurées. Ces signaux de commande sont ensuite utilisés pour contrôler les actionneurs du système distant, permettant à l'opérateur d'ajuster sa position de manière précise et intuitive par rapport au système distant [15].

Son fonctionnement repose sur l'échange de signaux de position seulement entre le maître et l'esclave à travers deux canaux de communication C1 et C4. L'avantage principal dans cette méthode est aucun capteur de force n'est nécessaire sur le site distant [31,36,37,43], les erreurs de position entre le maître et l'esclave sont alors injectées dans les contrôleurs du maître (Cm) et de l'esclave (Cs) pour créer le retour d'effort et commander les mouvements de l'esclave respectivement. Le schéma de base du contrôle position-position est illustré sur la Figure 1.5 :

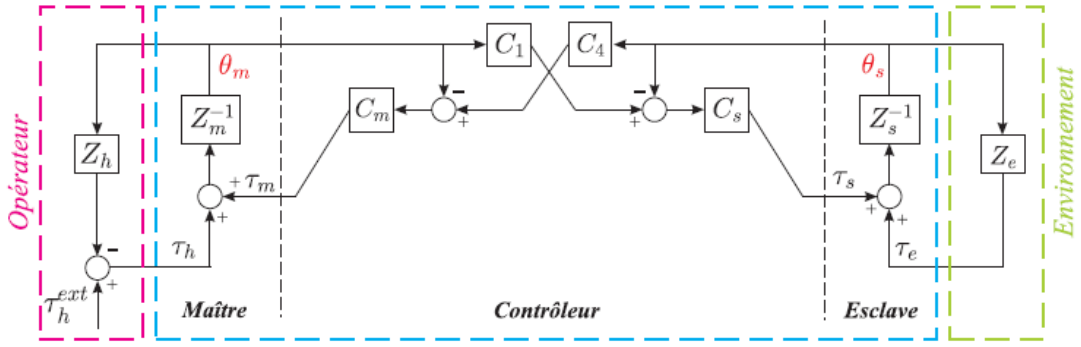

FIGURE 1.5 – Schéma de contrôle position-position.

Les quatre critères de performances qui caractérisent cette méthode de contrôle sont définis comme suit :

$$h_{11} = Z_{\min} = Z_m + \frac{Z_s \times C_m}{Z_s + C_s} = Z_m + \frac{C_m}{C_s} \times Z_s \tag{1.6}$$

$$h_{21} = X_T = \frac{C_s}{Z_s + C_s} = 1 \tag{1.7}$$

$$F_T = \frac{Z_m + C_m}{C_s} = \frac{C_m}{C_s} \tag{1.8}$$

$$Z_{\max} = Z_m + C_m = C_m^* \tag{1.9}$$

L'architecture position-position offre plusieurs avantages significatifs [17] :

• Intuitivité et facilité d'utilisation : L'opérateur contrôle directement la position du système distant en fonction de sa propre position, ce qui est plus naturel et intuitif que d'autres modes de contrôle. Cela réduit la charge cognitive de l'opérateur et facilite l'apprentissage et l'utilisation du système.

• Précision du contrôle : En contrôlant la position directement, l'opérateur peut effectuer des mouvements précis et subtils avec le système distant, ce qui est crucial pour les applications nécessitant un contrôle fin, telles que la chirurgie robotique ou la manipulation d'objets délicats.

• Feedback immédiat : Comme l'opérateur reçoit un retour direct sur la position du système distant, il peut ajuster sa propre position en conséquence, ce qui permet un contrôle en temps réel et une réactivité aux changements de l'environnement ou des conditions de travail.

• Pas besoin de capteurs de force sur le système distant : Contrairement à d'autres architectures qui nécessitent des capteurs de force pour mesurer les interactions forces/torques, l'architecture position-position utilise uniquement des capteurs de position, ce qui simplifie la conception et la mise en œuvre du système.

• Transparence cinématique : Dans certains cas, l'architecture position-position peut offrir une certaine transparence cinématique, où le système distant se déplace exactement comme l'opérateur l'ordonne, sans introduire de délais ou de distorsions indésirables.

• Robustesse face aux perturbations : Cette architecture peut être plus robuste face aux perturbations externes, car elle se concentre principalement sur le contrôle de la position plutôt que sur des interactions plus complexes telles que la force ou le couple.

Cependant, malgré ses avantages, l'architecture position-position peut présenter des inconvénients, tels que la possibilité de sentir l'impédance du système distant, ce qui peut affecter la perception de l'environnement par l'opérateur. De plus, elle peut ne pas être aussi efficace pour des tâches nécessitant un contrôle de la force précis.

1.7 Retard Fixe de Transmission

1.7.1 Definition

Un retard fixe de transmission est un délai constant et prévisible lors du transfert de données ou de signaux entre deux points dans un système. Dans la téléopération, ce retard peut affecter la synchronisation entre le maître et l'esclave, nécessitant des stratégies de contrôle spécifiques pour maintenir la performance et la stabilité[18].

1.7.2 Effet du retard de transmission

Le retard fixe de transmission peut entraîner un décalage entre les commandes émises par l'opérateur et les actions effectuées par le robot esclave. Ce décalage peut rendre la téléopération moins précise et moins fluide. Pour illustrer cet effet, supposons que le maître

envoie une commande de déplacement à l'instant t. En raison du retard, cette commande n'atteindra l'esclave qu'à l'instant $t+\tau$, où τ est le retard fixe de transmission. Par conséquent, l'esclave commencera à se déplacer seulement après ce délai, créant un décalage entre les mouvements du maître et de l'esclave[19].

1.7.3 Stratégies de contrôle

Dans le cadre de la téléopération avec un retard fixe de transmission, diverses stratégies de contrôle peuvent être déployées pour garantir un suivi précis et stable du mouvement entre le maître et l'esclave malgré ce délai. Une approche courante consiste à utiliser des techniques de prédiction pour estimer le comportement futur de l'esclave en fonction des commandes actuelles du maître et du retard de transmission. Cette prédiction permet de compenser le retard et de maintenir la synchronisation entre les deux systèmes, assurant ainsi un suivi plus précis [21].

Une autre approche consiste à concevoir des algorithmes de contrôle robustes capables de minimiser l'impact du retard sur la performance globale du système. Ces algorithmes peuvent inclure des mécanismes de correction en temps réel qui ajustent les commandes en fonction des écarts entre les positions attendues et réelles de l'esclave, ce qui contribue à maintenir une bonne coordination entre le maître et l'esclave[20].

De plus, l'utilisation de structures de contrôle avancées telles que les réseaux de neurones ou les systèmes neuro-flous peut permettre une adaptation efficace aux variations du système et du retard de transmission. Ces approches offrent une meilleure précision et une meilleure stabilité du système de téléopération en compensant les effets du retard et en s'adaptant aux changements dynamiques du système, améliorant ainsi les performances globales de la téléopération[22].

1.7.4 Introduction du Retard dans l'Architecture

Pour introduire le retard fixe de transmission dans une architecture de téléopération position-position, les signaux de position envoyés par le maître doivent être retardés avant d'être transmis à l'esclave. Cela peut être réalisé en utilisant une fonction de retard dans le système de transmission, qui décale les signaux de position de τ secondes, où τ est le retard fixe de transmission.

Une fois le retard introduit dans le système, les commandes peuvent être ajustées en fonction du retard pour compenser son effet sur la performance du système. En utilisant des méthodes de compensation de retard telles que la commande prédictive ou la commande

neuro-floue, il est possible d'atténuer les effets négatifs du retard et d'améliorer la synchronisation entre le maître et l'esclave dans une architecture de téléopération position-position[23].

1.8 Conclusion

Dans ce chapitre, nous avons exploré l'architecture d'une téléopération position-position avec un retard fixe de transmission. Nous avons défini la téléopération et ses éléments constitutifs, puis examiné les caractéristiques clés d'un système de téléopération bilatérale, notamment la stabilité et la transparence. Nous avons également abordé les différentes stratégies de contrôle visant à compenser le retard, telles que la prédiction et la correction en amont, ainsi que la commande à retardateur. Enfin, nous avons souligné l'importance de prendre en compte le retard lors de la conception de l'architecture de téléopération pour garantir un suivi précis des mouvements du maître par l'esclave.

Chapitre 2

Synthèse d'un régulateur neuro-flou

2.1 Introduction

Ce chapitre traite de la synthèse d'un régulateur neuro-flou pour un système maître-esclave à un degré de liberté. Nous commençons par présenter les bases de la logique floue et des réseaux de neurones artificiels. Nous explorons ensuite l'algorithme de rétro-propagation et d'autres méthodes d'apprentissage. Nous abordons les systèmes neuro-flous, en détaillant leur structure et leurs types d'implémentation. Enfin, nous examinons l'analyse de stabilité de l'algorithme de contrôle neuro-flou.

2.2 Généralités sur la logique floue

2.2.1 Logique floue

La logique est l'étude du raisonnement, visant à dériver de nouvelles propositions à partir de propositions existantes. Dans la logique classique, une proposition peut être soit vraie, soit fausse, c'est-à-dire que sa valeur est 1 ou 0. Bien que largement utilisée dans le domaine scientifique, la logique classique présente des limites en raison de ses valeurs de vérité absolue. Pour surmonter ces limitations, la logique floue a été introduite. Elle représente une transition de la vérité absolue à la vérité partielle, exprimée par un nombre compris entre 0 et 1.

La logique floue permet de tirer des conclusions imprécises à partir de prémisses floues. Fondée par Zadeh en 1965, cette logique vise à modéliser de manière réaliste le comportement humain, qui est capable de raisonner et de prendre des décisions rationnelles en présence d'incertitude et d'imprécision, tout en exécutant des tâches mentales et physiques sans mesures ni calculs exacts.

La logique floue est une technique de modélisation et de manipulation des concepts flous,

représentés par des ensembles flous. Elle associe à des variables des degrés d'appartenance à des sous-ensembles flous, avec des valeurs comprises entre 0 et 1, prenant ainsi en compte toutes les incertitudes liées à la variable[9].

2.2.2 Sous-ensemble flou

Un sous-ensemble classique A d'un ensemble U est défini par une fonction d'appartenance $\mu_A(x)$, qui indique pour chaque élément x de U s'il appartient ou non à A [9].

$$\mu_A(x) = \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{si } x \notin A \end{cases} \tag{2.1}$$

Cette fonction d'appartenance peut seulement prendre deux valeurs : 1 si x est un élément de A, ou 0 si x n'en est pas un.

En revanche, un sous-ensemble flou A d'un ensemble U est caractérisé par une fonction d'appartenance $\mu_A(x)$ qui peut prendre n'importe quelle valeur dans l'intervalle de 0 à 1. Le degré d'appartenance de l'élément x au sous-ensemble flou A est ainsi exprimé comme une valeur continue comprise entre 0 et 1[9].

$$\mu_A(x) \in [0, 1] \tag{2.2}$$

2.2.3 Variable linguistique

Une variable linguistique représente une situation vague à l'aide de termes ou d'expressions comme : (négatif, zéro, positif) ou (petit, très petit, moyen, grand, très grand). Une valeur linguistique attribuée à une variable linguistique est un ensemble flou défini sur l'univers de discours[9].

2.2.4 Fonction d'appartenance

Les variables linguistiques sont exprimées par des fonctions d'appartenance, de sorte que chaque variable floue x d'un sous-ensemble flou A soit liée à une fonction d'appartenance qui indique le degré auquel x appartient à A. Ces fonctions d'appartenance peuvent prendre diverses formes selon l'application. Les types les plus couramment utilisés en théorie du contrôle flou incluent les fonctions triangulaires, trapézoïdales, sigmoïdes et gaussiennes [9].Comme le montre la Figure 2.1

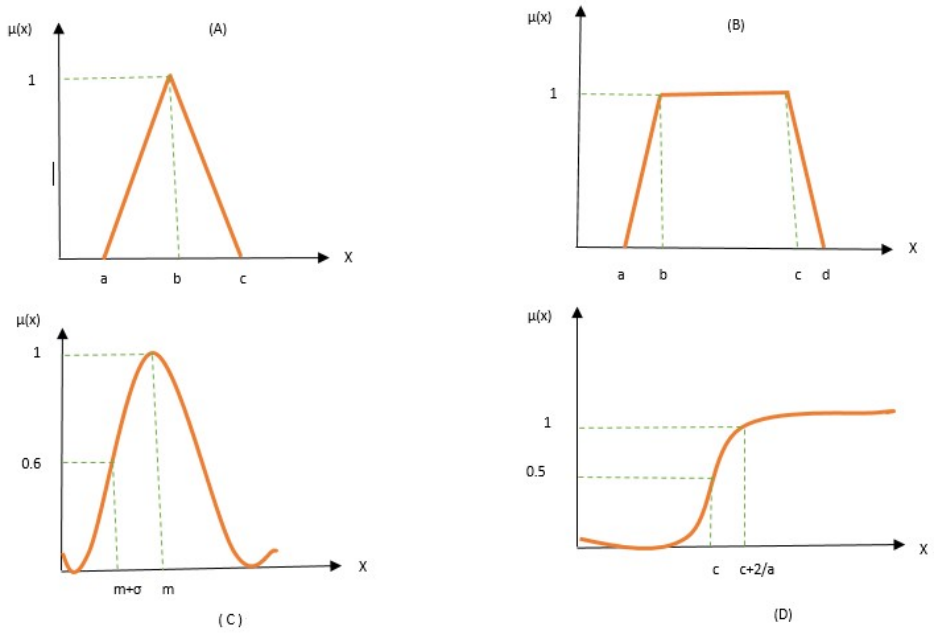

FIGURE 2.1 – fonctions d'appartenance.

Fonction triangulaire

La figure (A) est définie par trois paramètres (a, b et c) qui spécifient les coordonnées des trois sommets. La fonction d'appartenance $\mu(x)$ est donnée par :

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

Fonction trapézoïdale

La figure (B) est une fonction caractérisée par quatre paramètres (a, b, c et d) qui définissent les positions des quatre sommets. La fonction d'appartenance $\mu(x)$ est exprimée par :

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

Fonction gaussienne

La figure (C) est définie par deux paramètres (σ et m). La fonction d'appartenance $\mu(x)$ est donnée par :

$$\mu(x) = \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right) \tag{2.5}$$

Fonction sigmoïde

La fonction sigmoïde (D) est caractérisée par deux paramètres (a et c), et elle est définie par l'expression suivante :

$$\mu(x) = \frac{1}{1 + \exp(-a(x - c))} \tag{2.6}$$

2.2.5 Univers de discours

L'univers de discours désigne la plage de valeurs possibles pour une variable linguistique, décrivant ainsi le domaine d'activité du processus à contrôler. Il est crucial de le prendre en compte lors de la configuration du régulateur [9].

2.2.6 Règle d'inférence

Les règles d'inférence constituent l'ensemble des principes flous qui établissent des liens entre les variables floues d'entrée et de sortie d'un système en utilisant divers opérateurs flous. Elles prennent 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 ... Si condition n ET/OU condition n + 1 (ET/OU...) alors action sur les sorties." [9].

2.2.7 Opérations sur les ensembles flous

En théorie des ensembles classique, les variables sont reliées à travers des opérateurs logiques comme ET (intersection), OU (union) et NON (complément à 1). En logique floue, ces opérateurs sont interprétés respectivement par Minimum, Maximum et Complément à 1, et sont définis par leurs fonctions d'appartenance. Ces opérateurs sont décrits dans le tableau ci-dessous, illustrant leurs équivalences entre les logiques classique et floue.[9]

	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 = $\bar{A}$	$\mu_C(x) = 1 - \mu_A(x)$

TABLE 2.1 – Opérateurs dans les logiques classique et floue.

2.2.8 Commande floue

Les approches classiques de contrôle sont souvent limitées, notamment pour modéliser des systèmes non linéaires, partiellement inconnus ou présentant des incertitudes dynamiques imprévisibles. Pour surmonter ces défis, la logique floue est utilisée, permettant de modéliser ces comportements imprévisibles en utilisant une approche similaire au raisonnement humain, telle que proposée par Zadeh. L'intégration de la logique floue dans le contrôle des systèmes est un domaine de recherche actif, offrant une méthode de commande efficace pour des systèmes complexes et fortement non linéaires sans nécessiter un modèle mathématique précis. Cette approche repose sur des inférences utilisant des règles floues basées sur des variables linguistiques. Nous présenterons ci-dessous les composants essentiels d'un régulateur flou[24].

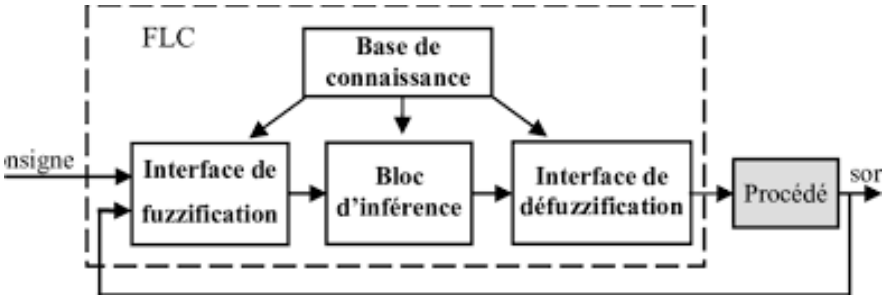

FIGURE 2.2 – Schéma fonctionnel d'un régulateur flou

Fuzzification

La fuzzification implique la conversion de valeurs numériques, issues par exemple de capteurs, en variables linguistiques. Elle évalue le degré d'appartenance de ces valeurs à un ensemble flou, utilisant diverses fonctions d'appartenance dont le choix est déterminé par l'expert en fonction du contexte et des contraintes de calcul[9].
Par exemple, si nous avons une mesure de température en degrés Celsius provenant d'un capteur, nous pouvons la transformer en une variable linguistique. Cette variable pourrait être qualifiée par plusieurs termes linguistiques tels que chaud, froid, très froid, tempéré, très chaud, etc.Si le capteur nous renvoie 17°C, après fuzzification, la température sera chaude à 20%, tempérée à 60% et froide à 0% [25].

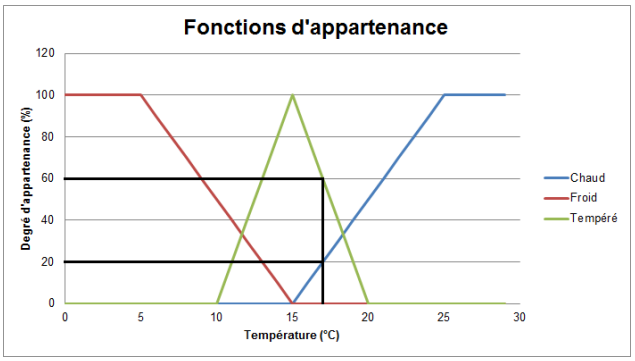

FIGURE 2.3 – Exemple de fuzzification

Base de règles

La base de règles d'un système flou décrit de manière qualitative le comportement d'une sortie en fonction de diverses entrées. Ces règles sont généralement formulées sous la forme suivante [25] : "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, ..., Si condition n ET/OU condition n + 1 (ET/OU...) alors action sur les sorties." Les entrées peuvent représenter des paramètres tels que l'erreur, la variation de l'erreur ou la somme d'erreurs, tandis que la sortie peut correspondre à une action de contrôle ou une action de variation de contrôle.

Mécanisme d'inférence

Le mécanisme d'inférence en logique floue est une étape essentielle qui transforme les informations floues issues de la fuzzification en une forme plus précise. Cette transformation s'appuie sur trois éléments clés : une base de règles contenant des règles floues décrivant le comportement du système, une base de données définissant les fonctions d'appartenance utilisées dans ces règles, et un mécanisme de raisonnement qui applique ces règles pour aboutir à une conclusion logique.

Dans un régulateur flou de type Mamdani, la sortie est exprimée par des variables linguistiques, tandis que dans un régulateur de type Takagi-Sugeno, la sortie est une valeur numérique ou une fonction mathématique. Cette distinction est importante car elle influence la manière dont l'inférence est réalisée et comment la sortie est interprétée.

L'inférence floue nécessite l'utilisation d'opérateurs flous pour combiner les informations selon les règles. Parmi les méthodes d'inférence les plus couramment utilisées, on trouve les méthodes MAX-MIN de Mamdani et somme-produit de Sugeno. Ces méthodes permettent de prendre en compte de manière pondérée la contribution de chaque règle pour obtenir une

sortie floue finale qui représente la réponse du système de manière précise et adaptable aux variations de l'environnement[9].

Défuzzification

La défuzzification, contrairement à la fuzzification, qui convertit les valeurs numériques en ensembles flous, la défuzzification vise à obtenir des valeurs concrètes pour guider les actions du système. Plusieurs approches de défuzzification existent, mais la méthode du centre de gravité est la plus répandue en raison de sa simplicité et de son efficacité.

La défuzzification comprend deux principales étapes. Tout d'abord, elle implique la fusion des différentes variables linguistiques générées par les règles d'inférence, en utilisant des opérateurs logiques pour combiner les valeurs. Ensuite, elle consiste à déterminer une valeur numérique unique à partir de ces variables combinées. Pour cela, deux méthodes populaires sont utilisées : la méthode de la moyenne des maximas et la méthode du centre de gravité.

La méthode de la moyenne des maximas consiste à calculer la moyenne des valeurs numériques correspondant aux points où la fonction d'appartenance est maximale. Cette approche permet d'obtenir une valeur de sortie qui représente au mieux les informations fournies par les différentes règles. En revanche, la méthode du centre de gravité calcule le centre de gravité de la surface délimitée par les fonctions d'appartenance combinées. Cette approche offre une vision plus globale de la sortie floue, en prenant en compte l'ensemble des valeurs possibles et leurs degrés d'appartenance respectifs[26].

2.3 Généralités sur les réseaux de neurones artificiels

2.3.1 Réseaux de neurones

Les réseaux de neurones artificiels (RNA) sont perçus comme des représentations mathématiques simplifiées du cerveau humain. Ils fonctionnent comme des systèmes cellulaires physiques capables de capter, de mémoriser et d'exploiter des connaissances empiriques grâce à un algorithme d'apprentissage. Ces réseaux se composent principalement de neurones artificiels, connectés entre eux par des liaisons dotées de coefficients de pondération, ainsi que d'un algorithme d'apprentissage.

Les réseaux de neurones artificiels offrent plusieurs avantages notables, dont une grande capacité de calcul et une aptitude à apprendre de manière itérative. Pendant la phase d'apprentissage, les poids des connexions évoluent pour permettre au réseau de reproduire les sorties souhaitées à partir d'entrées connues. Lorsque de nouvelles entrées sont fournies, le réseau est capable de générer des résultats optimaux en fonction de ces nouvelles données.

Les modèles de réseaux de neurones artificiels se distinguent par leurs méthodes d'apprentissage (supervisé ou non supervisé), leurs architectures (récurrentes ou non récurrentes) et leurs types de sortie (binaire ou continue) [9].

2.3.2 Neurone biologique

Le neurone est l'élément de base du système nerveux central, dont la fonction principale est de transmettre des impulsions électriques générées par des réactions physico-chimiques dans des conditions spécifiques. Il est constitué de quatre parties essentielles : les dendrites, le corps cellulaire, l'axone et les synapses.

Les dendrites captent en permanence les signaux provenant d'autres neurones. Le corps cellulaire traite toutes les informations reçues par les dendrites. L'axone a pour rôle de transmettre les signaux vers d'autres neurones connectés. Les synapses sont les jonctions qui permettent le transfert des signaux électriques des axones d'un neurone aux dendrites des autres neurones[9].

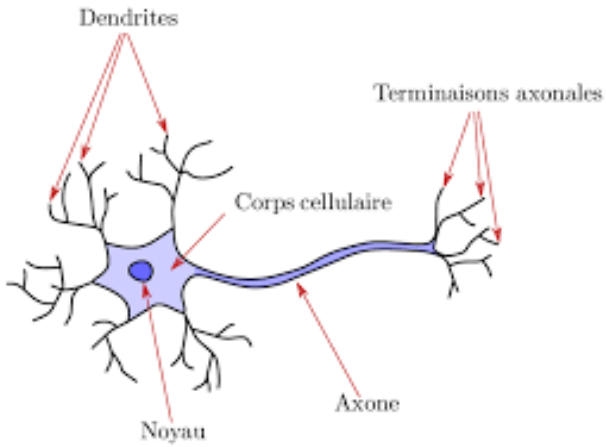

FIGURE 2.4 – Neurone biologique.

2.3.3 Neurone formel

Un neurone formel est une abstraction mathématique qui s'inspire du fonctionnement des neurones biologiques pour le traitement de l'information dans les réseaux de neurones artificiels. Ce modèle simplifié est constitué de plusieurs éléments clés : les entrées pondérées, la fonction d'activation et la sortie. Dans cette section, nous détaillerons le fonctionnement d'un neurone formel et les concepts mathématiques qui le sous-tendent[27].

Structure et Fonctionnement

Un neurone formel peut être décrit par les composants suivants [28] :

Entrées (Inputs) : Ce sont les signaux reçus par le neurone. Chaque entrée x_i est associée à un poids w_i. Les entrées peuvent être représentées sous la forme d'un vecteur $X = [x_1, x_2, \ldots, x_n]$.

Poids (Weights) : Chaque connexion d'entrée est pondérée par un coefficient w_i, qui indique l'importance relative de cette entrée. Les poids peuvent également être représentés par un vecteur $W = [w_1, w_2, \ldots, w_n]$.

Somme pondérée (Weighted Sum) : Le neurone effectue une sommation pondérée des entrées. Cette somme pondérée est calculée comme suit :

$$z = \sum_{i=1}^{n} w_i x_i = w \cdot x \tag{2.7}$$

où $w \cdot x$ représente le produit scalaire des vecteurs de poids et d'entrées.

Biais (Bias) : Un biais b est ajouté à la somme pondérée pour ajuster la sortie du neurone. Le biais permet au modèle de mieux s'adapter aux données en déplaçant la fonction d'activation vers la gauche ou la droite :

$$z = w \cdot x + b \tag{2.8}$$

Fonction d'activation (Activation Function) : La somme pondérée z passe ensuite à travers une fonction d'activation $\phi(z)$, qui introduit de la non-linéarité dans le modèle. Cette non-linéarité permet au réseau de neurones de modéliser des relations complexes. Cette fonction peut prendre plusieurs formes à savoir : signe, tout ou rien, fonction affine, plus ou moins à seuil, saturation, fonction gaussienne et fonction arc-tangente.

Sortie (Output) : La sortie du neurone y est le résultat de la fonction d'activation appliquée à la somme pondérée :

$$y = \phi(z) = \phi(w \cdot x + b) \tag{2.9}$$

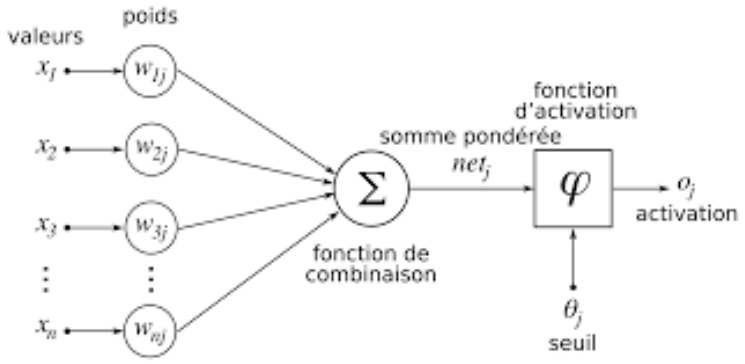

FIGURE 2.5 – Modèle général d'un neurone formel.

2.3.4 Architectures des réseaux de neurones artificiels

Les architectures des réseaux de neurones artificiels (RNA) déterminent comment les neurones sont organisés et interconnectés. Les choix d'architecture ont un impact significatif sur les capacités de modélisation, la performance et l'efficacité du réseau. Voici un aperçu des principales architectures de réseaux de neurones artificiels.

Architecture non-bouclée à une seule couche

L'architecture non-bouclée à une seule couche, également connue sous le nom de perceptron, est une structure de réseau de neurones fondamentale et simple. Elle se compose d'une couche d'entrée recevant les données d'entrée et d'une couche de sortie générant la sortie du réseau, sans aucune couche cachée. Durant la propagation avant, les données d'entrée sont pondérées et passent à travers une fonction d'activation pour produire la sortie du neurone. Cette architecture est couramment utilisée pour des tâches de classification binaire et des opérations logiques simples en raison de sa simplicité et de sa facilité de mise en œuvre. Cependant, elle est limitée dans sa capacité à modéliser des fonctions non linéaires complexes en raison de sa nature linéaire [9].

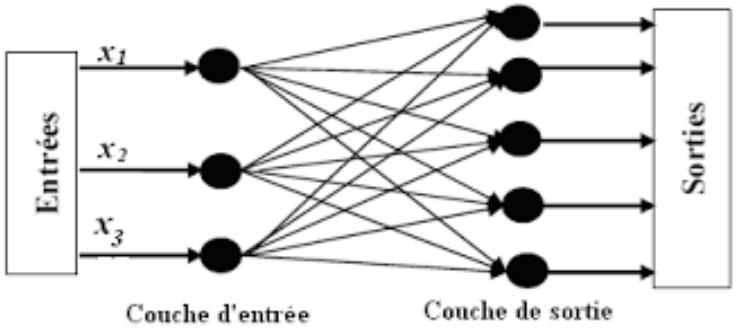

FIGURE 2.6 – Réseau de neurones artificiels à une seule couche

Architecture non-bouclée à multi-couches

L'architecture non bouclée à multicouches, également connue sous le nom de perceptron multicouche (MLP), est un type de réseau de neurones artificiels où les données circulent dans une seule direction, de la couche d'entrée à la couche de sortie, sans rétroaction. Elle se compose de plusieurs couches de neurones : une couche d'entrée, une ou plusieurs couches cachées, et une couche de sortie (Figure 2.7). Chaque neurone d'une couche est connecté à tous les neurones de la couche suivante. Les neurones de chaque couche appliquent une transformation linéaire suivie d'une fonction d'activation non linéaire aux signaux qu'ils reçoivent. Cette structure permet au réseau de capturer des relations complexes dans les données en ajustant les poids des connexions pendant le processus d'apprentissage. Les poids sont ajustés en minimisant l'erreur entre les prédictions du réseau et les valeurs réelles grâce à un algorithme appelé rétropropagation. Cette architecture est utilisée pour diverses applications comme la reconnaissance de motifs, la classification et la prédiction [9].

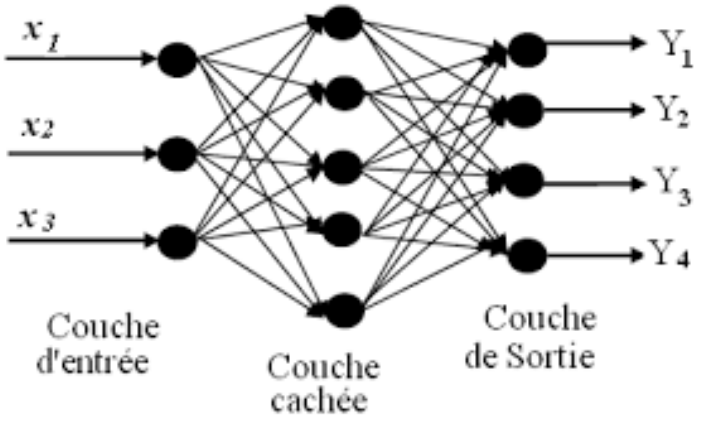

FIGURE 2.7 – Réseau de neurones artificiels à multi-couches

Architecture bouclée (récurrente)

Les réseaux de neurones récurrents (RNN) sont une classe de réseaux de neurones où les connexions entre les unités forment des boucles, permettant de conserver une mémoire des informations précédentes. Cette architecture est idéale pour les données séquentielles ou temporelles, comme les séries chronologiques, les textes et les signaux audio. Les RNN comportent une couche d'entrée pour les données séquentielles, des couches récurrentes où les neurones sont connectés à eux-mêmes pour conserver les états passés, et une couche de sortie pour les prédictions. À chaque pas de temps, l'état du neurone récurrent est mis à jour en fonction de l'entrée actuelle et de l'état précédent. Les variantes comme les LSTM (Long Short-Term Memory) et les GRU (Gated Recurrent Unit) sont conçues pour mieux gérer

les dépendances à long terme et éviter les problèmes de vanishing et exploding gradients. Les RNN, notamment avec ces variantes, sont puissants pour des applications telles que la traduction automatique, la reconnaissance vocale et l'analyse de séries temporelles, malgré leur complexité et leur besoin en ressources computationnelles[29].

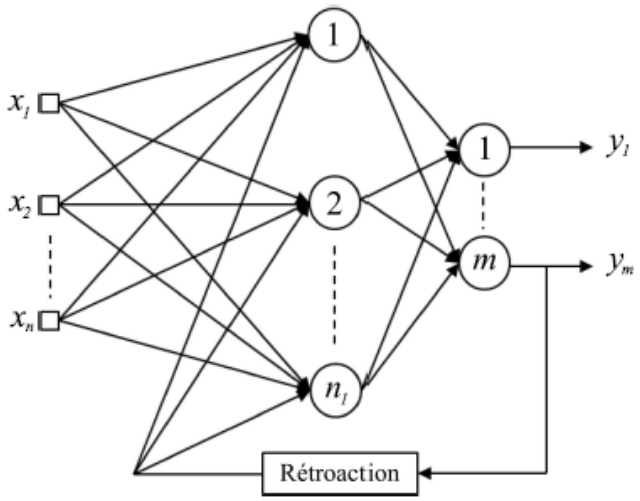

FIGURE 2.8 – Réseau de neurones artificiels bouclé.

2.4 Processus d'apprentissage

Une des caractéristiques principales des réseaux de neurones artificiels est leur capacité d'apprentissage. Un réseau de neurones artificiels peut apprendre et établir une relation entre les entrées et les sorties afin de généraliser une solution et produire une sortie proche de celle désirée. Le processus d'apprentissage implique plusieurs étapes structurées visant à ajuster les poids et les seuils synaptiques des neurones artificiels pour améliorer la généralisation des solutions générées par les sorties du réseau. Ce processus ordonné est connu sous le nom d'algorithme d'apprentissage. Il existe plusieurs méthodes d'apprentissage, parmi lesquelles on distingue l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement [9].

2.4.1 Apprentissage supervisé

L'apprentissage supervisé est une méthode d'apprentissage automatique où un modèle est formé à partir de données étiquetées, chaque exemple comprenant des entrées et des sorties connues. Le processus commence par la préparation et la séparation des données en jeux d'entraînement, de validation et de test. Le modèle est ensuite construit et entraîné en

utilisant la propagation avant pour générer des prédictions et la rétropropagation pour ajuster les poids afin de minimiser l'erreur de prédiction. Après ajustement des hyperparamètres et validation pour éviter le surajustement, le modèle est évalué sur le jeu de test en utilisant diverses métriques de performance. Bien que très efficace et capable d'atteindre une haute précision, l'apprentissage supervisé dépend fortement de la disponibilité de données étiquetées de qualité et peut être sujet au surajustement si les données d'entraînement sont insuffisantes.

Parmi les algorithmes d'apprentissage supervisé les plus connus, on cite l'algorithme de rétro-propagation de l'erreur[9].

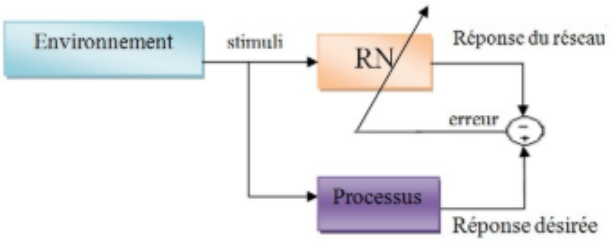

FIGURE 2.9 – Apprentissage supervisé

Algorithme de rétro-propagation

La méthode de rétro-propagation de l'erreur est une technique d'apprentissage supervisé qui ajuste les poids et les biais des réseaux de neurones artificiels pour minimiser l'erreur quadratique entre les sorties prédites et les sorties réelles. Pour chaque paire entrée/sortie, une erreur est calculée. Si la réponse du réseau diffère de la réponse attendue, les poids et les biais sont modifiés en temps réel pour réduire cette erreur [9].

Prenons un réseau de neurones à trois couches (n, m, p) comme illustré dans la Figure 2.10, avec q représentant le vecteur d'entrée et y^q le vecteur des sorties désirées. La fonction à minimiser est l'erreur quadratique E^q entre la sortie prédite O^q et la sortie souhaitée y^q :

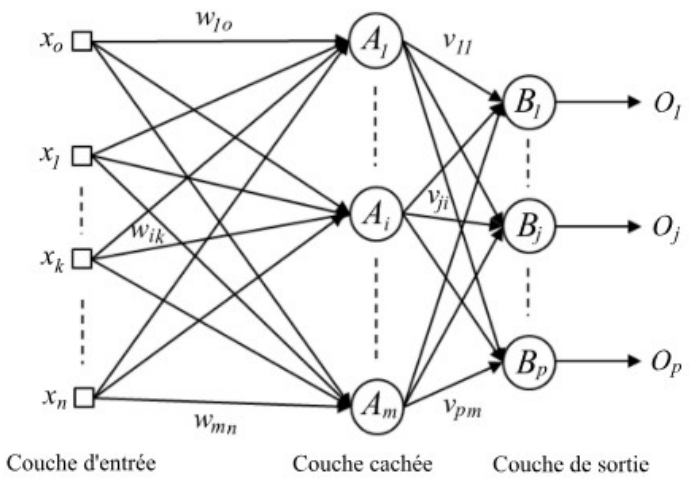

FIGURE 2.10 – Réseau de neurones à trois couches

$$E^q = \frac{1}{2}\sum_j \left(O - y_q^j\right)^2 \tag{2.10}$$

Soit v_{ji} les poids entre les neurones cachés et les neurones de sortie, et w_{ik} les poids entre les neurones d'entrée et les neurones cachés. L'objectif est de calculer ces poids pour minimiser E^q. Les poids v_{ji} et w_{ik} sont mis à jour comme suit :

$$v_{ji} \to v_{ji} + \triangle v_{ji}, \tag{2.11}$$

$$w_{ik} \to w_{ik} + \triangle w_{ik} \tag{2.12}$$

Tout d'abord, les poids v_{ji} sont ajustés par la méthode de descente du gradient. Pour un neurone de sortie j et un neurone caché i, les poids v_{ji} sont mis à jour par :

$$\triangle v_{ji} = -\eta \sum_{q=1}^{N} \frac{\partial E_q}{\partial v_{ji}} = -\eta \sum_{q=1}^{N} \delta_i^q z_i^q \tag{2.13}$$

où z_i^q est l'activation du neurone caché i.

$$z_i^q = f_i\left(\sum_{k=0}^{n} w_{ik} x_k^q\right) \tag{2.14}$$

Et :

$$\delta_q^j = \left(O_q^j - y_q^j\right) f'i \left(\sum_{k=1}^{m} v_{jk} z_q^k\right) \tag{2.15}$$

Pour un neurone caché i et un neurone d'entrée k, les poids w_{ik} sont mis à jour ainsi :

$$\triangle^q w_{ik} = -\eta \frac{\partial E_q}{\partial w_{ji}} \tag{2.16}$$

Avec :

$$\frac{\partial E^q}{\partial w_{ik}} = \frac{\partial E^q}{\partial O_i^q} \frac{\partial O_i^q}{\partial w_{ik}} \tag{2.17}$$

Où η est le taux d'apprentissage, et O_i^q est la sortie du neurone caché i.

$$O_i^q = f_i \left(\sum_{k=0}^{n} w_{il} x_l^q\right) = z_i^q \tag{2.18}$$

Nous avons :

$$\frac{\partial O_i^q}{\partial w_{ik}} = f_i' \left(\sum_{l=0}^{n} w_{ik} x_l^q\right) x_k^q \tag{2.19}$$

Soit :

$$\delta_i^q = \frac{\partial E^q}{\partial O_q^i} = \sum_{j=1}^{p} \frac{\partial E^q}{\partial O_j^q} \frac{\partial O_j^q}{\partial O_i^q} \tag{2.20}$$

Où j est un neurone de la couche de sortie. Le terme $\frac{\partial E^q}{\partial O_j^q}$ est calculé précédemment, donc :

$$O_i^q = f_j' \left(\sum_{l=1}^{m} v_{il} z_l^q\right) v_{ji} \tag{2.21}$$

Ainsi, δ_i^q pour le neurone caché i est calculé à partir des valeurs déjà connues de δ_j^q pour chaque neurone de sortie j.

Cet algorithme est appelé rétro-propagation car il commence par propager les entrées x^q à travers le réseau jusqu'à la couche de sortie, calcule ensuite les δ_j^q pour chaque neurone de sortie j, et propage ces δ_j^q vers la couche cachée pour calculer les δ_i^q pour chaque neurone caché i.

2.4.2 Apprentissage non supervisé

L'apprentissage est qualifié de non supervisé lorsque seules les valeurs d'entrée sont disponibles. Il repose sur un critère interne de conformité du comportement 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 se modifie en fonction des régularités statistiques de l'entrée. L'algorithme d'apprentissage utilise une mesure prédéterminée de la qualité de la représentation de la connaissance pour ajuster les paramètres du réseau. Ce type d'apprentissage est représenté schématiquement à la Figure 2.11 .

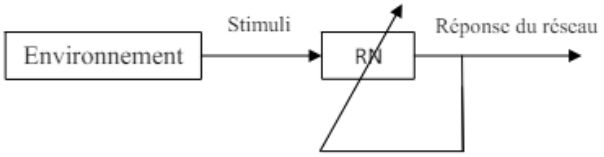

FIGURE 2.11 – Apprentissage non supervisé.

2.4.3 Apprentissage par renforcement

Les algorithmes d'apprentissage basés sur cette méthode ajustent les paramètres du réseau de neurones artificiels (RNA) en fonction des informations acquises lors de l'interaction avec l'environnement afin d'évaluer les performances d'apprentissage. Le processus d'apprentissage se fait par essais et erreurs : la réponse du réseau pour une entrée est alors évaluée pour déterminer si elle est satisfaisante ou non. Si la sortie est satisfaisante, les poids et les seuils sont progressivement ajustés.

Ce type d'apprentissage est appelé apprentissage par renforcement lorsqu'on ne possède que des indications imprécises, c'est-à-dire que l'on indique seulement au réseau si la sortie est correcte ou non. Dans ce cas, le réseau tend à maximiser un indice de performance appelé signal de renforcement. Le système est ainsi capable de savoir si la réponse qu'il fournit est correcte, même s'il ne connaît pas la réponse exacte attendue[9].

2.5 Réseaux neuro-flous

2.5.1 Systèmes neuro-flous

Les systèmes flous sont construits à partir des connaissances humaines en modélisant le fonctionnement du cerveau humain et utilisent des variables linguistiques, ce qui leur confère une capacité descriptive élevée. Cependant, il n'existe aucune méthode formelle pour définir

les fonctions d'appartenance ou construire la base de règles pour ces systèmes. Par ailleurs, les réseaux de neurones artificiels se distinguent par leurs structures flexibles et leurs capacités d'apprentissage à partir de données expérimentales, rendant ces réseaux utiles dans de nombreuses applications telles que la reconnaissance de formes, la modélisation et le contrôle des procédés. Toutefois, leurs structures et paramètres manquent souvent d'interprétation physique et ne peuvent pas exploiter la connaissance humaine pour leur construction.

Pour surmonter ces limitations, la combinaison des réseaux de neurones et de la logique floue permet de créer des systèmes neuro-flous plus efficaces. Ces systèmes hybrides allient les capacités d'apprentissage des réseaux de neurones à la capacité d'inférence de la logique floue, permettant d'adapter ou d'optimiser les règles linguistiques et les fonctions d'appartenance. L'apprentissage dans les systèmes neuro-flous ajuste les paramètres et s'adapte aux changements de conditions, avec des inférences extraites après convergence. Un avantage notable de ces systèmes est qu'ils n'exigent pas de connaissances sur le modèle mathématique du système à commander, ce qui les rend utiles pour résoudre des problèmes de commande dont le modèle mathématique est inconnu ou présente des incertitudes dynamiques. Dans notre travail, nous utilisons un contrôleur neuro-flou pour piloter un système de téléopération sans recourir à des modèles mathématiques précis du maître et de l'esclave, afin de s'adapter aux variations dynamiques de ces systèmes, notamment lorsque l'esclave est en contact avec un environnement inconnu.

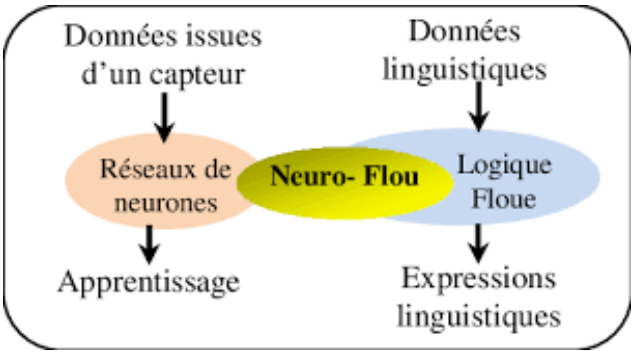

FIGURE 2.12 – reseaux-neuro-flous.

2.5.2 Types de combinaisons neuro-floues

Il existe trois principales catégories de combinaisons entre réseaux de neurones et logique floue, chacune caractérisée par son fonctionnement et son architecture [9].

Système flou neuronal

Les techniques floues sont intégrées dans ces réseaux pour étendre les capacités du processus d'apprentissage et d'exécution des réseaux de neurones.

Système neuronal flou

Les systèmes neuronaux flous préservent les propriétés fondamentales et les architectures de base des réseaux de neurones tout en rendant flous certains de leurs éléments. Dans ce type de réseaux, un neurone peut devenir flou, et sa réponse peut être une relation floue plutôt qu'une réponse binaire. Cette approche est largement utilisée dans les applications de reconnaissance de formes, où la capacité des neurones flous à traiter des informations incertaines ou partielles est particulièrement utile.

Système neuro-flou hybride

Dans ce type de combinaisons, les techniques floues et les réseaux de neurones jouent un rôle central. En exploitant les avantages individuels de ces deux approches, elles s'intègrent et se complètent mutuellement pour atteindre un objectif commun. Les systèmes neuro-flous hybrides tirent parti de la puissance des réseaux de neurones pour leurs capacités de traitement avancées, tout en bénéficiant des avantages de la logique floue pour son raisonnement flexible et adaptatif dans la prise de décision. Ce type de système est particulièrement utilisé dans les applications de contrôle des procédés et de reconnaissance de formes, où la combinaison des deux techniques permet d'obtenir des performances supérieures.

2.5.3 Types d'implémentation des réseaux neuro-flous

Système d'inférence neuro-floue

Un système d'inférence neuro-floue est mis en œuvre sous la forme d'un réseau de neurones, où les paramètres de la logique floue sont incarnés par les poids synaptiques du réseau. Ainsi, la structure neuronale permet l'ajustement automatique et l'optimisation des fonctions d'appartenance en modifiant les poids des connexions du réseau neuronal à l'aide d'un algorithme d'apprentissage adapté. Cette approche offre une flexibilité et une capacité d'adaptation accrues, permettant au système d'apprendre et de s'ajuster à partir des données d'entrée, améliorant ainsi ses performances au fil du temps [9].

Système d'inférence neuro-floue adaptatif (ANFIS)

Les systèmes d'inférence neuro-floue adaptatifs font partie de 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 intégré.

La structure ANFIS, proposée par Jang en 1993 [30], décrit le fonctionnement d'un système flou avec des règles pouvant être de type Takagi-Sugeno ou Mamdani, organisées sous forme d'un réseau de neurones. Ce réseau est entraîné à l'aide d'un algorithme d'apprentissage de rétro-propagation, qui ajuste et optimise les paramètres de la prémisse et de la conséquence du réseau. Le système ANFIS se compose de cinq couches, où les nœuds adaptatifs contenant des paramètres se trouvent dans la première et la quatrième couche, tandis que les autres nœuds sans paramètres restent fixes[9].

2.5.4 Méthodes d'apprentissage des réseaux neuro-flous

Les algorithmes utilisés pour les réseaux de neurones artificiels et les réseaux neuro-flous, notamment en robotique, offrent des systèmes intelligents capables de s'adapter à des environnements dynamiques inconnus grâce à des mécanismes d'apprentissage en ligne ou hors ligne.

L'apprentissage en ligne implique l'ajustement des paramètres du réseau neuro-flou en temps réel, simultanément avec le fonctionnement du système à contrôler, en utilisant des algorithmes d'optimisation locale et globale. Les données sont présentées de manière séquentielle par rapport au temps, et l'algorithme estime et prédit les paramètres du réseau en fonction des actions passées et présentes. En revanche, dans l'apprentissage hors ligne, les paramètres du réseau sont mis à jour une seule fois et restent fixes par la suite, quelle que soit l'évolution du système à contrôler dans le temps [9].

Dans notre travail, nous allons utiliser un contrôleur neuro-flou de type ANFIS pour piloter le système de téléopération , en utilisant une carte Arduino Due .

2.5.5 Structure ANFIS

La fusion des réseaux de neurones et des systèmes flous dans un seul réseau, comme le réseau ANFIS, permet de bénéficier simultanément des avantages de chaque approche. La structure du réseau neuro-flou ANFIS se divise en deux parties : une partie prémisse et une partie conséquence, connectées par une base de règles floues sous forme de réseau. Cette structure permet d'adapter les règles floues aux changements de conditions, ce qui permet aux systèmes ANFIS d'optimiser et d'ajuster automatiquement les fonctions d'appartenance grâce à un algorithme d'apprentissage. Un exemple de réseau neuro-flou utilisant un modèle

d'inférence floue du premier ordre de Sugeno, avec deux entrées (x et y)et une seule sortie (f), est représenté dans la Figure 2.13 qui suit.Ce réseau comporte 5 couches, chacune ayant une fonction spécifique qui sera décrite ci-dessous.

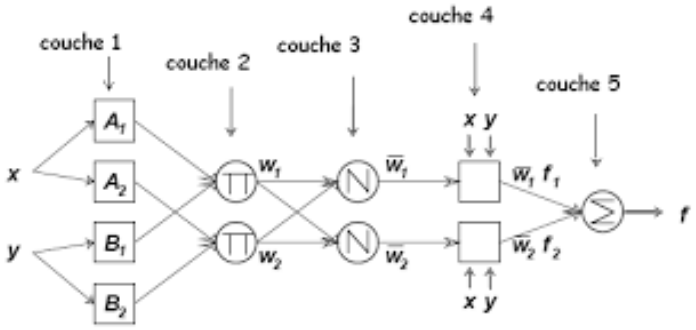

FIGURE 2.13 – Architecture d'un réseau ANFIS.

Les deux règles floues sont définies comme suit [9] :

$$\text{si } x \text{ est } A_1 \text{ et } y \text{ est } B_1 \text{ alors } z_1 = p_1x + q_1y + r_1 \tag{2.22}$$

$$\text{si } x \text{ est } A_2 \text{ et } y \text{ est } B_2 \text{ alors } z_2 = p_2x + q_2y + r_2 \tag{2.23}$$

Couche 1 : Fuzzification

Chaque nœud i de cette couche est un nœud adaptatif équipé d'une fonction d'appartenance qui possède des paramètres ajustables.

$$O_{1,j} = \mu_{Aj}(x), \text{ pour } j = 1, 2 \tag{2.24}$$

$$O_{1,j} = \mu_{B_{j-2}}(x), \text{ pour } j = 3, 4 \tag{2.25}$$

$O_{1,j}$ sont les degrés d'appartenance des variables x et y 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

Génère le degré d'activation approprié à la règle.

$$O_{2,j} = w_j = \mu_{A_j}(x)\mu_{B_j}(y), \text{ pour } j = 1, 2 \tag{2.26}$$

Couche 3 : Normalisation

Chaque nœud de cette couche est un nœud fixe appelé N. Sa sortie représente le degré d'activation normalisé de la j-ème règle.

$$O_{3,j} = \overline{w}_j = \frac{w_j}{w_1 + w_2}, \text{ pour } j = 1, 2 \tag{2.27}$$

Couche 4 : Calcul des sorties des règles

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

$$O_{4,j} = \overline{w}_j z_j = w_j(p_j x + q_j y + r_j) \tag{2.28}$$

Où w_j est la sortie de la couche 3 qui représente le degré d'activation normalisé de la règle et (p_j, q_j, r_j) sont les paramètres de sortie ajustables de la règle j. Ces paramètres sont appelés paramètres de la conséquence.

Couche 5 : Défuzzification

Cette couche comprend un seul nœud fixe, qui calcule la sortie globale qui est la somme de tous les signaux provenant de la couche 4 :

$$O_{5,j} = z = \sum_j \overline{w}_j z_j = \frac{\sum_j w_j z_j}{\sum_j w_j} \tag{2.29}$$

Les paramètres de la conséquence (p_j, q_j et r_j) sont identifiés dans le processus d'apprentissage, en utilisant un algorithme basé sur des méthodes d'optimisation telle que la descente du gradient.

2.6 Commande neuro-floue du système maître-esclave à un degré de liberté

La commande neuro-floue est une approche innovante de la commande des systèmes dynamiques, combinant les principes de la logique floue et des réseaux de neurones artificiels. Dans le contexte du système maître-esclave à un degré de liberté, cette méthode est particulièrement pertinente pour contrôler le mouvement d'un système esclave en réponse aux mouvements d'un système maître, malgré la présence d'un retard fixe de transmission.

La mise en œuvre de la commande neuro-floue dans ce système repose sur l'utilisation d'une carte Arduino Due, un microcontrôleur puissant capable de gérer les calculs nécessaires à la commande en temps réel. La carte Arduino Due est choisie pour sa rapidité d'exécution, ce qui est crucial pour compenser efficacement le retard de transmission et assurer une réponse rapide du système esclave[**?**].

Le fonctionnement de la commande neuro-floue implique plusieurs étapes. Tout d'abord, les signaux de position du système maître sont capturés par des capteurs et transmis à la carte Arduino Due. Ensuite, ces signaux sont traités par un algorithme neuro-flou, qui combine la logique floue pour la prise de décision et les réseaux de neurones pour l'adaptabilité du système[31].

L'algorithme neuro-flou ajuste les paramètres de contrôle en fonction des signaux reçus, permettant au système esclave de suivre fidèlement les mouvements du système maître malgré le retard de transmission. Cette approche offre une grande robustesse et une capacité d'adaptation aux variations de l'environnement, ce qui en fait une solution idéale pour les systèmes de téléopération nécessitant une grande précision et une réponse rapide[31].

2.6.1 Contrôle position-position

L'architecture de contrôle position-position repose sur l'échange de signaux de position entre le poste maître et celui de l'esclave, permettant ainsi de commander précisément les positions du maître et de l'esclave à l'aide de deux contrôleurs ANFIS distincts[9].

Dans ce schéma de commande, représenté dans la Figure 2.14, les variables θ_m et θ_s correspondent respectivement aux positions du maître et de l'esclave. Le terme T_e représente le couple d'interaction de l'esclave avec son environnement, tandis que T_h désigne le couple exercé par l'opérateur sur le maître. Les signaux u_m et u_s sont les commandes envoyées aux actionneurs pour contrôler les mouvements du maître et de l'esclave.

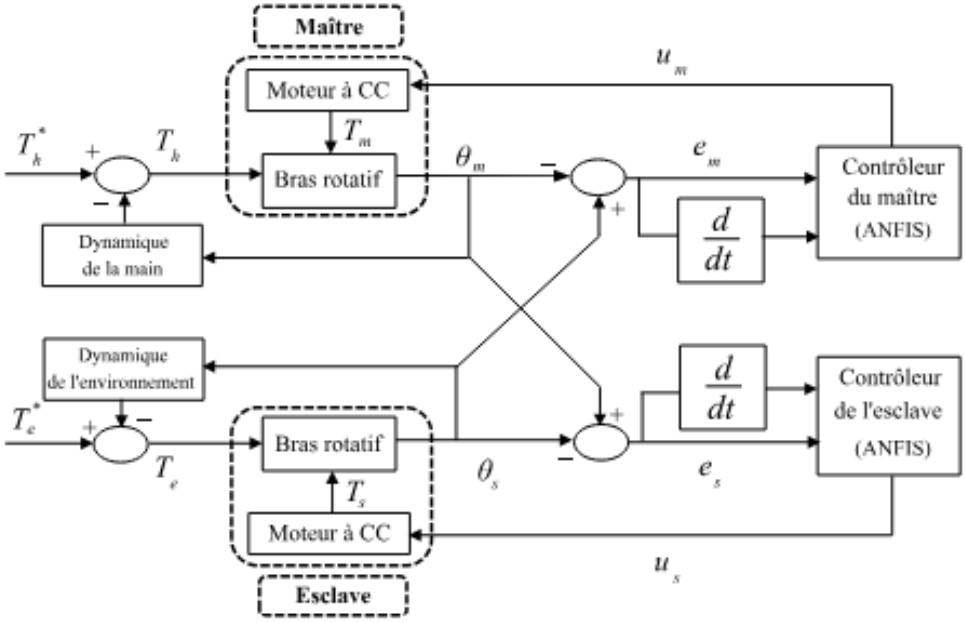

FIGURE 2.14 – Schéma de commande position-position.

2.6.2 Algorithme de commande ANFIS

Dans cette partie, nous concevons le contrôleur ANFIS pour le système maître en adoptant la stratégie de commande position-position. Ce processus est également appliqué au contrôleur du système esclave en mode position, ainsi qu'aux contrôleurs du maître en mode couple et de l'esclave en mode position pour la stratégie de commande à quatre canaux.

Nous considérons un réseau neuro-flou de premier ordre avec un système d'inférence floue de type Takagi-Sugeno, comprenant deux entrées x_1 et x_2 et une seule sortie f.

Les règles floues sont définies comme suit [9] :

$$\text{Règle 1 : si } x_1 \text{ est } A_1 \text{ et } x_2 \text{ est } B_1 \text{ alors } f_1 = p_1x_1 + q_1x_2 + r_1 \tag{2.30}$$

$$\text{Règle 2 : si } x_1 \text{ est } A_1 \text{ et } x_2 \text{ est } B_2 \text{ alors } f_2 = p_2x_1 + q_2x_2 + r_2 \tag{2.31}$$

$$\text{Règle 3 : si } x_1 \text{ est } A_2 \text{ et } x_2 \text{ est } B_1 \text{ alors } f_3 = p_3x_1 + q_3x_2 + r_3 \tag{2.32}$$

$$\text{Règle 4 : si } x_1 \text{ est } A_2 \text{ et } x_2 \text{ est } B_2 \text{ alors } f_4 = p_4x_1 + q_4x_2 + r_4 \tag{2.33}$$

Où $f_j = p_jx_1 + q_jx_2 + r_j$ pour $j = 1, 2, ..., 4$, x_1 et x_2 sont l'erreur de position du maître et sa dérivée : $[x_1, x_2] = [e, \Delta e]$. A_i et B_i sont des sous-ensembles flous, p_j , q_j et r_j sont les paramètres de la conséquence de la règle j déterminés dans le processus d'apprentissage.

Pour la commande neuro-floue du système maître-esclave à un degré de liberté, nous as-

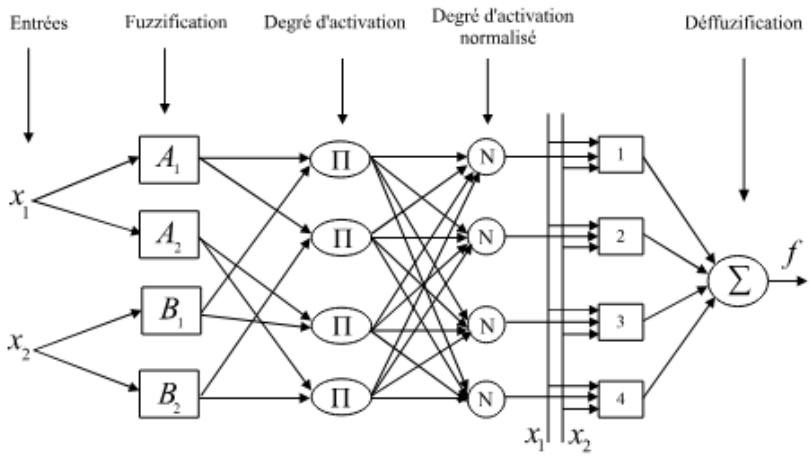

FIGURE 2.15 – Structure du régulateur ANFIS.

socions deux ensembles flous pour chacune des entrées x_1 et x_2, nommés N (Negative) et P (Positive). Les degrés d'appartenance des variables x_i aux sous-ensembles flous A_i et B_i sont μ_P et μ_N, définis par les fonctions d'appartenance suivantes :

$$\mu_P(x_i) = \begin{cases} 0, & \text{si } x_i < -1, \\ 0.5x_i + 0.5, & \text{si } -1 < x_i < 1, \\ 1, & \text{si } x_i > 1. \end{cases} \tag{2.34}$$

$$\mu_N(x_i) = \begin{cases} 1, & \text{si } x_i < -1, \\ -0.5x_i + 0.5, & \text{si } -1 < x_i < 1, \\ 0, & \text{si } x_i > 1. \end{cases} \tag{2.35}$$

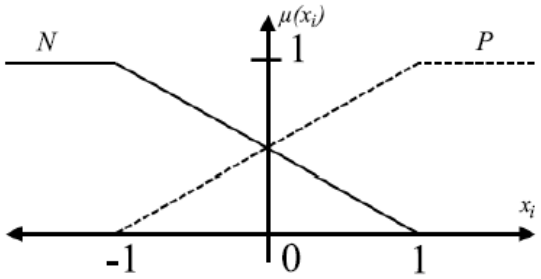

FIGURE 2.16 – Fonctions d'appartenance

En utilisant la méthode de déffuzification par centre de gravité, la valeur numérique de la sortie u est donnée par :

$$u = \frac{\sum_{j=1}^{4} f_j w_j}{\sum_{j=1}^{4} w_j} \tag{2.36}$$

Où :

$$w_1 = \mu_P(x_1) \cdot \mu_P(x_2) = \mu_{A_1}(x_1) \cdot \mu_{B_1}(x_2) \tag{2.37}$$

$$w_2 = \mu_P(x_1) \cdot \mu_N(x_2) = \mu_{A_1}(x_1) \cdot \mu_{B_2}(x_2) \tag{2.38}$$

$$w_3 = \mu_N(x_1) \cdot \mu_P(x_2) = \mu_{A_2}(x_1) \cdot \mu_{B_1}(x_2) \tag{2.39}$$

$$w_4 = \mu_N(x_1) \cdot \mu_N(x_2) = \mu_{A_2}(x_1) \cdot \mu_{B_2}(x_2) \tag{2.40}$$

2.6.3 Algorithme d'apprentissage

Pour identifier et ajuster les paramètres des contrôleurs ANFIS, nous utilisons l'algorithme d'apprentissage. Ici, un algorithme d'apprentissage est développé pour chaque contrôleur ANFIS, tel que$(k = s)$ pour le premier contrôleur ANFIS du robot esclave, $(k = m)$ pour le second contrôleur ANFIS du robot maître que les paramètres des locaux sont fixés. Cet algorithme permet d'ajuster les paramètres de conséquence en minimisant la fonction objective suivante :

$$j = \frac{1}{2}(y_d^k - y^k)^2 \tag{2.41}$$

Où y_d est la sortie désirée et y est la sortie réelle du système à contrôler.

Soit Φ_j le vecteur des paramètres à ajuster. L'objectif est de trouver les paramètres p_j, q_j et r_j du vecteur Φ_j en utilisant la méthode de la descente du gradient [9] comme suit :

$$\Phi_j(k+1) = \Phi_j(k) - \alpha(k)\frac{\partial J}{\partial \Phi_j} \tag{2.42}$$

Nous avons :

$$\frac{\partial J}{\partial \Phi_j} = -e\frac{\partial y}{\partial \Phi_j} = -e\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j} \tag{2.43}$$

À partir des équations (2.42) et (2.43), nous aurons :

$$\Phi_j(k+1) = \Phi_j(k) + \alpha(k)\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j}e \tag{2.44}$$

Dans notre cas, le terme $\frac{\partial y}{\partial u}$ ne peut pas être calculé, mais il peut être estimé en utilisant les équations du filtre de Kalman étendu.

L'équation (2.44) peut être écrite comme suit :

$$\Phi_j(k+1) = \Phi_j(k) + K_{0j}(\Psi_j)^T e \tag{2.45}$$

Où :

$$(\Psi_j)^T = \frac{\partial u}{\partial \Phi_j} = \begin{bmatrix} \frac{\partial u}{\partial p_j} \\ \\ \frac{\partial u}{\partial q_j} \\ \\ \frac{\partial u}{\partial r_j} \end{bmatrix} \tag{2.46}$$

$$K_{0j} = \alpha(k)\frac{\partial y}{\partial u} \tag{2.47}$$

L'équation (2.45) peut être identifiée à l'équation du filtre de Kalman étendu [9] :

$$\Phi_j(k+1) = \Phi_j(k) + K(k)e \tag{2.48}$$

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

$$K(k) = P(k)H^T(k)\left(H(k)P(k)H^T(k) + R(k)\right)^{-1} \tag{2.49}$$

Avec $H(k)$ 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 processus.

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 :

$$K(k) = \lambda_1(\Psi_j)^T \left(\lambda_1(\Psi_j)(\Psi_j)^T + \lambda_2\right)^{-1} \tag{2.50}$$

Donc l'équation (2.48) devient :

$$\Phi_j(k+1) = \Phi_j(k) + \lambda_1(\Psi_j)^T \left(\lambda_1(\Psi_j)(\Psi_j)^T + \lambda_2\right)^{-1} e \tag{2.51}$$

Par identification entre les équations (2.45) et (2.51), nous obtenons :

$$K_{0j} = \lambda_1(\Psi_j)^T \left(\lambda_1(\Psi_j)(\Psi_j)^T + \lambda_2\right)^{-1} \tag{2.52}$$

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

$$\Phi_j(k+1) = \Phi_j(k) + K_{0j}(\Psi_j)^T e \tag{2.53}$$

Pour une période d'échantillonnage T_e très courte, nous avons :

$$\dot{\Phi}_j = \frac{\Phi_j(k+1) - \Phi_j(k)}{T_e} = \frac{K_{0j}(\Psi_j)^T e}{T_e} = K_1(\Psi_j)^T e \tag{2.54}$$

Où $K_1 = \frac{K_{0j}}{T_e}$.
Donc :

$$\dot{\Phi}_j = K_1(\Psi_j)^T e = (\Psi_j)^T e_u \tag{2.55}$$

Où $e_u = K_1 e$.

Soit $\Phi_{ej} = \Phi_{jd} - \Phi_j$, où Φ_j est le vecteur des paramètres et Φ_{jd} est le vecteur des paramètres désirés.

Ce qui implique :

$$\dot{\Phi_{ej}} = \dot{\Phi_{jd}} - \dot{\Phi_j} = -(\Psi_j)^T e_u \tag{2.56}$$

Avec $e_u = u_d - u$ est l'erreur entre la sortie désirée du contrôleur u_d et la sortie calculée du contrôleur u. Pour une variation linéaire, elle est définie par :

$$e_u = u_d - u = \sum_{j=1}^{4} \left((\Psi_j)\Phi_{jd} - (\Psi_j)\Phi_j\right) = \sum_{j=1}^{4} \left((\Psi_j)(\Phi_{jd} - \Phi_j)\right) = \sum_{j=1}^{4} (\Psi_j)\Phi_{ej} \tag{2.57}$$

2.6.4 Analyse de stabilité de l'algorithme de contrôle

Considérons la fonction de Lyapunov suivante [9] :

$$V_j = \frac{1}{2}\sum_{j=1}^{4} (\Phi_{ej})^T (\Phi_{ej}) \tag{2.58}$$

En dérivant V_j par rapport au temps, nous aurons :

$$\dot{V_j} = \sum_{j=1}^{4} \left(\dot{\Phi_{ej}}\right)^T (\Phi_{ej}) \tag{2.59}$$

À partir des équations (2.56), (2.57) et (2.59), nous obtenons :

$$\dot{V_j} = \sum_{j=1}^{4} \left(-(\Psi_j)^T e_u\right)^T (\Phi_{ej}) = -(e_u)^T \sum_{j=1}^{4} ((\Psi_j)(\Phi_{ej})) = -(e_u)^T (e_u) \tag{2.60}$$

Par conséquent :

$$\dot{V_j} = -(e_u)^T (e_u) \leq 0 \tag{2.61}$$

À partir de l'équation (2.61), nous obtenons $\dot{V_j} \leq 0$, par conséquent, nous pouvons conclure que le système est asymptotiquement stable au sens de Lyapunov [9].

2.7 Conclusion

Ce chapitre explore la synthèse d'un régulateur neuro-flou, une approche avancée pour concevoir des systèmes de commande. Il combine les principes de la logique floue et des réseaux de neurones artificiels. Nous débutons par les bases théoriques, en présentant la logique floue et les réseaux de neurones artificiels.

Ensuite, nous examinons l'apprentissage, soulignant l'importance des méthodes supervisées, non supervisées et par renforcement dans la synthèse des régulateurs neuro-flous. L'algorithme de rétro-propagation est crucial pour ajuster les poids des connexions neuronales dans le réseau.

Nous plongeons ensuite dans les réseaux neuro-flous, décrivant leur structure et leur fonctionnement, ainsi que les différents types de combinaisons neuro-floues possibles. L'adaptabilité est une caractéristique clé des réseaux neuro-flous, illustrée par les systèmes d'inférence neuro-flous adaptatifs (ANFIS).

En conclusion, la synthèse d'un régulateur neuro-flou offre une voie prometteuse pour des systèmes de commande adaptatifs et robustes, capables de s'ajuster à des environnements complexes. L'intégration de la logique floue et des réseaux de neurones ouvre de nouvelles perspectives dans l'automatique et la robotique.

Chapitre 3

Application d'une Stratégie de Commande Neuro-floue à une Architecture de Téléopération Position-Position

3.1 Introduction

La téléopération permet de contrôler des systèmes à distance, avec des applications allant de la chirurgie robotique aux explorations spatiales. Un des défis majeurs de la téléopération est la gestion des retards de transmission, qui peuvent dégrader la performance du système et réduire la précision des tâches. Pour pallier ce problème, nous avons mis en œuvre une stratégie de commande neuro-floue pour une architecture de téléopération position-position, visant à améliorer la précision et la robustesse du système malgré les retards fixes de transmission. Ce chapitre détaille la mise en œuvre matérielle et logicielle ainsi que les résultats expérimentaux obtenus.

3.2 Mise en Œuvre

3.2.1 Matériel Utilisé

Pour cette expérimentation, nous avons utilisé les composants suivants :

Arduino Due :

Microcontrôleur principal pour la génération des signaux de commande.
Offrant une grande puissance de calcul grâce à son processeur ARM Cortex-M3.

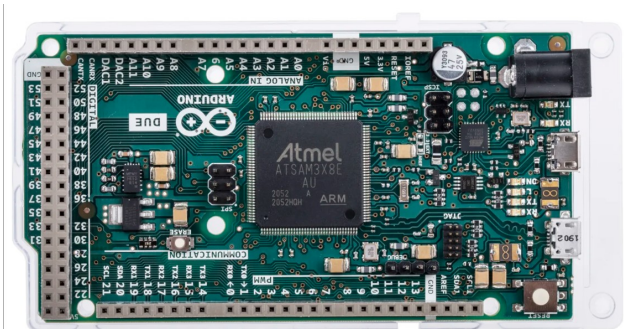

FIGURE 3.1 – Carte Arduino Due

Arduino Mega :

Utilisé pour l'acquisition des données des encodeurs des moteurs.
Dispose de nombreuses broches d'entrée/sortie analogiques et numériques pour une acquisition précise et rapide.

FIGURE 3.2 – Carte Arduino Mega 2560

Moteurs avec Encodeurs :

Deux moteurs DC équipés d'encodeurs pour mesurer la position angulaire.
Les encodeurs fournissent des retours de position en impulsions, converties en radians pour l'analyse.

FIGURE 3.3 – Moteur avec Encodeur

Module de Commande de Moteur L298N :

Contrôle la direction et la vitesse des moteurs.
Capable de piloter deux moteurs DC avec une alimentation externe pour un contrôle précis du mouvement.

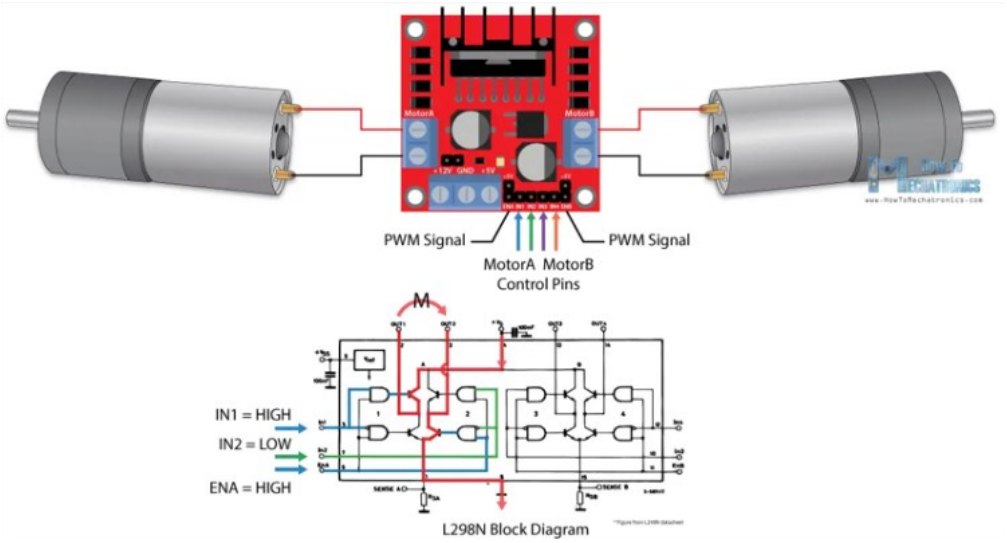

FIGURE 3.4 – Module L298N.

Breadboard et Câbles de Connexion :

Utilisés pour réaliser les connexions entre les composants.
Permettent une configuration flexible et modifiable du circuit.

Alimentation Électrique :

Fournit l'énergie nécessaire aux moteurs et aux cartes Arduino.
Une source de 3.3V est utilisée pour alimenter le module L298N et les moteurs.

3.2.2 Câblage et Fonctionnement

Le schéma de câblage est essentiel pour assurer la synchronisation et le bon fonctionnement des différents composants. Les détails de câblage sont fournis ci-dessous illustrant la figure 3.5.

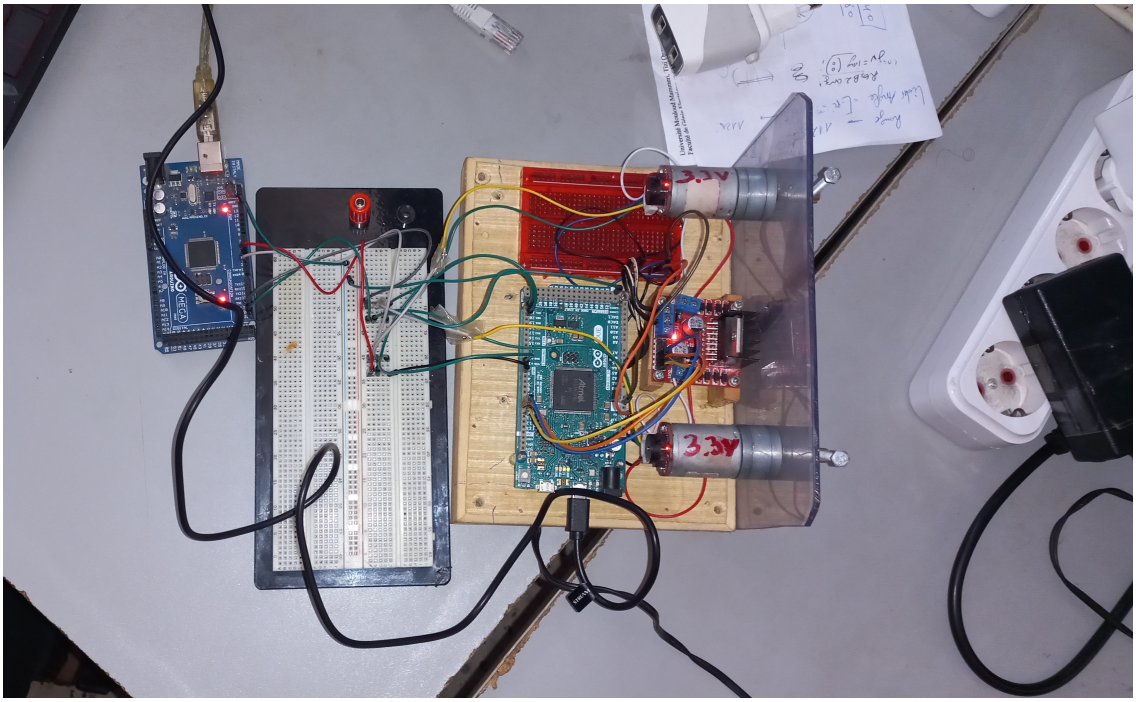

FIGURE 3.5 – Plate-forme expérimentale du système de téléopération position-position

Détails du Câblage :

Connexion des Moteurs et du Module L298N :

Les moteurs sont connectés aux sorties du module L298N.
Les entrées de commande du module L298N sont reliées aux broches PWM de l'Arduino Due.

Broches Utilisées sur l'Arduino Due :

PWM pour Moteur 1

Nous avons utilisés :
IN1 : Pin 6
IN2 : Pin 7
ENA : Pin 9

PWM pour Moteur 2

Nous avons utilisés :
IN3 : Pin 8
IN4 : Pin 11
ENB : Pin 10

Connexion des Encodeurs :

Les signaux des encodeurs des moteurs sont connectés aux entrées analogiques de l'Arduino Mega.
Les signaux sont traités pour convertir les impulsions des encodeurs en positions angulaires (en radians).

Broches Utilisées sur l'Arduino Mega :

Encodeur Moteur 1

Sont :
Signal A : Pin 2
Signal B : Pin 3

Encodeur Moteur 2

Sont :
Signal A : Pin 18
Signal B : Pin 19

Communication entre Arduino Due et Arduino Mega :

Une communication série est établie entre les deux cartes Arduino.
L'Arduino Due envoie les signaux de commande ajustés par le contrôleur neuro-flou, tandis

que l'Arduino Mega transmet les données de position en temps réel.

Fonctionnement du Système

Le système de téléopération fonctionne sur le principe maître-esclave :

Moteur Maître

Contrôlé directement par l'utilisateur via une interface de commande. Les commandes de position sont envoyées à ce moteur.

Moteur Esclave

Suit les mouvements du moteur maître. Les commandes de suivi sont générées en fonction de la position actuelle du moteur maître et ajustées par la commande neuro-floue pour compenser les retards de transmission.

3.2.3 Stratégie de Commande Neuro-floue

La commande neuro-floue utilise des règles floues pour modéliser l'expertise humaine et un réseau neuronal pour ajuster dynamiquement les paramètres des règles.

Modélisation Floue

Les règles floues sont basées sur des entrées et sorties floues. Par exemple :
Si l'erreur de position est grande et la variation de l'erreur est positive ; alors augmenter la commande du moteur esclave.

Entraînement du Réseau Neuronal

Le réseau neuronal est entraîné avec des données expérimentales pour minimiser l'erreur de suivi. Il ajuste les gains des règles floues en temps réel pour améliorer la précision du système.

3.3 Résultats Expérimentaux

3.3.1 Graphique de Position

Le graphiquede la figure 3.6 montre les positions en radians des moteurs maître et esclave au cours du temps :

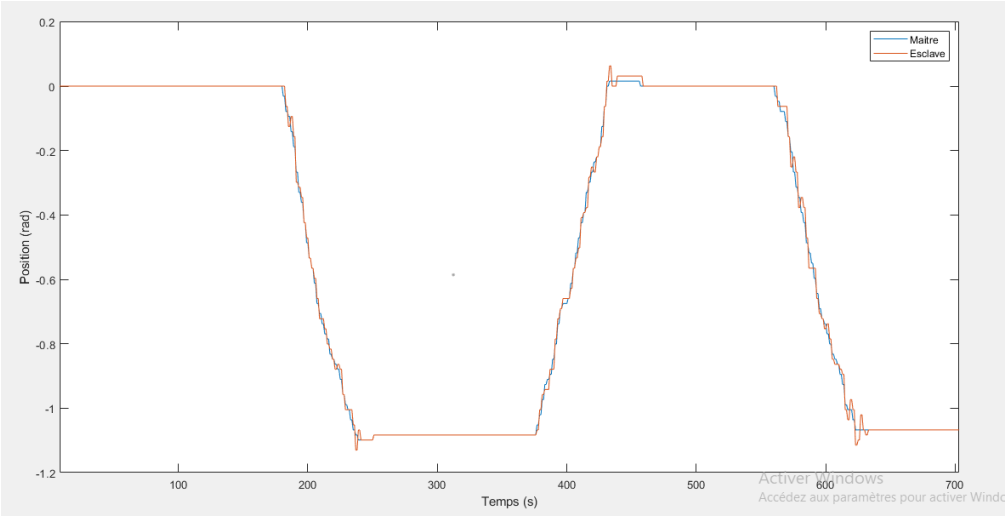

FIGURE 3.6 – Comportement de suivi de mouvement entre le maître et l'esclave

Analyse des Résultats

Les positions des moteurs maître et esclave sont très proches, indiquant un suivi précis malgré le retard de transmission. Les petites variations sont rapidement corrigées par le contrôleur neuro-flou, démontrant son efficacité.

3.3.2 Graphique de l'Erreur

Le graphique de la figure 3.7 présente l'erreur de position entre le moteur maître et le moteur esclave :

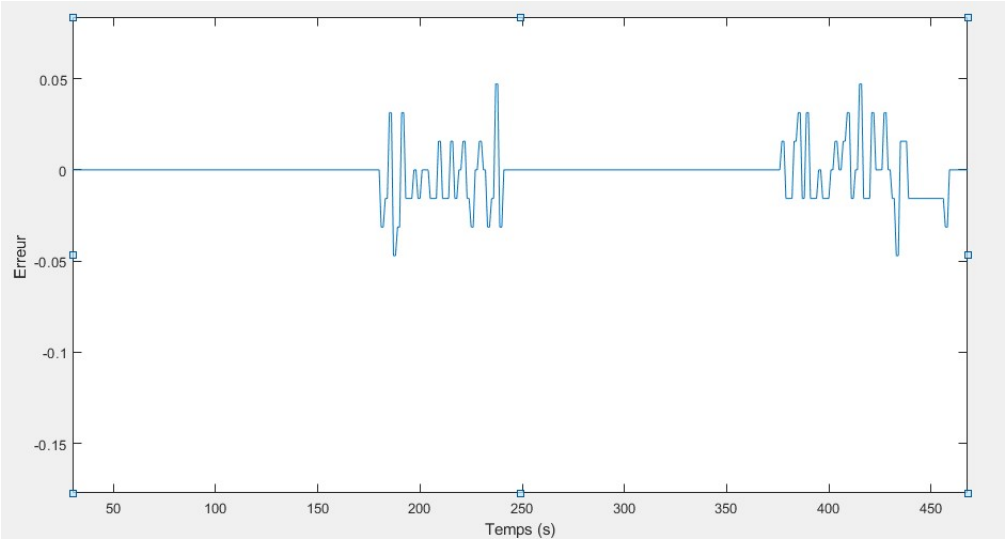

FIGURE 3.7 – Graphique de l'Erreur de Position

Analyse des Résultats

L'erreur de position est généralement faible, avec des variations autour de zéro. Les pics d'erreur sont dus à des changements rapides de la position du moteur maître, mais ils sont rapidement corrigés. Cela démontre que la commande neuro-floue est efficace pour compenser les retards fixes de transmission et maintenir une synchronisation précise entre les moteurs.

3.4 Conclusion

L'application d'une stratégie de commande neuro-floue à une architecture de téléopération position-position a montré des résultats très prometteurs. Le système a réussi à maintenir une faible erreur de suivi entre les moteurs maître et esclave, malgré le retard fixe de transmission. La combinaison des réseaux neuronaux et de la logique floue offre une solution robuste et adaptable pour améliorer les performances des systèmes de téléopération. Les résultats expérimentaux valident l'efficacité de cette approche, ouvrant la voie à des applications plus avancées et complexes dans le domaine de la téléopération.

Conclusion générale

La téléopération est une technologie essentielle qui permet le contrôle de systèmes robotiques à distance, jouant un rôle crucial dans des domaines variés tels que la médecine, l'industrie, et les opérations dans des environnements hostiles. Malgré ses nombreux avantages, la téléopération pose des défis importants, notamment en termes de gestion des retards de transmission. Ces retards, qu'ils soient dus à des distances géographiques, des contraintes de réseau ou des limitations techniques, peuvent grandement affecter la stabilité et la performance des systèmes de téléopération. Dans ce contexte, la recherche de solutions efficaces pour compenser ces retards est primordiale pour assurer la fiabilité et la précision des opérations de téléopération.

Ce mémoire s'est concentré sur l'implémentation d'une commande neuro-floue dans une architecture de téléopération position-position, en présence de retards fixes de transmission. La commande neuro-floue combine les capacités d'apprentissage des réseaux de neurones artificiels avec la flexibilité des systèmes flous, offrant ainsi une solution robuste pour gérer les incertitudes et les non-linéarités des systèmes complexes. L'objectif principal était de développer un contrôleur capable de s'adapter en temps réel aux variations des conditions de transmission, tout en maintenant une synchronisation précise entre les mouvements du maître et de l'esclave.

Le développement de ce contrôleur neuro-flou a nécessité une analyse approfondie des impacts des retards de transmission sur les performances des systèmes de téléopération. Une modélisation détaillée du système a été réalisée, suivie par la conception d'algorithmes d'apprentissage basés sur la méthode de descente du gradient pour l'ajustement en temps réel des paramètres du contrôleur. Cette méthode s'est révélée efficace pour minimiser l'erreur de position entre le maître et l'esclave, assurant une meilleure synchronisation des mouvements et une performance globale améliorée.

Un aspect clé de notre approche a été l'intégration du filtre de Kalman étendu, qui a permis d'estimer les termes de dérivée partielle nécessaires pour l'algorithme d'apprentissage du contrôleur neuro-flou. Cette intégration a renforcé la robustesse et la fiabilité du système de commande, permettant une adaptation plus précise et réactive aux variations de transmission.

Les résultats obtenus à travers des simulations et des expériences pratiques ont démontré la supériorité de notre approche par rapport aux méthodes traditionnelles de commande, particulièrement en termes de stabilité et de précision.

Les contributions de ce mémoire à l'état de l'art résident dans la proposition d'une solution novatrice pour la commande des systèmes de téléopération en présence de retards de transmission. En combinant les techniques d'intelligence artificielle et de contrôle automatique, cette approche offre une nouvelle perspective sur la gestion des défis liés aux retards de transmission dans les systèmes de téléopération. Les résultats obtenus ouvrent de nouvelles avenues pour des recherches futures, notamment l'application de techniques d'apprentissage profond qui pourraient offrir des améliorations supplémentaires en termes de précision et d'adaptabilité.

Les perspectives de ce travail sont vastes et prometteuses. Une direction de recherche future pourrait être l'extension de cette approche à des systèmes de téléopération avec plusieurs degrés de liberté, permettant de gérer des tâches plus complexes et variées. De plus, l'intégration de techniques de machine learning avancées, telles que l'apprentissage par renforcement, pourrait améliorer encore davantage les capacités d'adaptation et de performance des systèmes de téléopération. En outre, l'exploration de nouvelles architectures de réseau et de méthodes d'apprentissage pourrait conduire à des systèmes de commande encore plus robustes et efficaces.

En résumé, l'implémentation d'une commande neuro-floue pour les systèmes de téléopération représente une avancée prometteuse pour surmonter les défis posés par les retards de transmission. Cette approche, en combinant les capacités d'apprentissage des réseaux de neurones et la flexibilité des systèmes flous, offre une solution robuste et adaptable, améliorant ainsi la performance et la fiabilité des systèmes de téléopération. Les travaux présentés dans ce mémoire contribuent significativement à l'amélioration des technologies de téléopération et ouvrent de nouvelles perspectives pour des applications futures dans divers domaines. La capacité de ces systèmes à s'adapter en temps réel aux variations des conditions de transmission tout en maintenant une synchronisation précise des mouvements est un atout majeur qui pourrait révolutionner de nombreux secteurs d'application.

Bibliographie

[1] Sheridan, T. B. (1992). Telerobotics, Automation, and Human Supervisory Control. MIT Press.

[2] Funda, J.Taylor, R. H., end Eldridge, B. (1992). Comparison of position/position and rate/position teleoperation for a simple translational task. Proceedings of the 1992 IEEE International Conference on Robotics and Automation, 1, 119-124.

[3] Anderson, R. J., end Spong, M. W. (1990). Bilateral control of teleoperators with time delay. IEEE Transactions on Automatic Control, 34(5), 494-501.

[4] Niemeyer, G., end Slotine, J. J. E. (2003). Telemanipulation with time delays. The International Journal of Robotics Research, 23(9), 873-890.

[5] Jang, J.-S. R. (1993). ANFIS : adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics, 23(3), 665-685.

[6] Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338-353.

[7] Sadegh, N. (1993). A perceptron network for functional identification and control of nonlinear systems. IEEE Transactions on Neural Networks, 4(6), 982-988.

[8] Siciliano, B., Sciavicco, L., Villani, L., end Oriolo, G. (2009). Robotics : Modelling, Planning and Control. Springer.

[9] : Hocine KHATI, Commande d'une Architecture de Téléopération par la Carte FPGA , Thèse de Doctorat en Automatique à l' Université Mouloud MAMMERI de Tizi-Ouzou , Février 2020 .

[10] PF.Hokayem,andMW.Spong,"Bilateral teleoperation :An historical survy,"Automatica,Vol.242,no.12,pp.2035-2057,2006.

[11] J. Artigas,andG.Hirzinger,"A brief histiry of DLR's space telerobotics and force feedback teleoperation," Acta Polytechnica Hungarica,2016,13,pp.239-249.

[12] Pierre LETIER ,Bras Exosquelette Haptique : Conception et Contrôle, Thèse de Docteur en Sciences de l'Ingénieur à l'université Libre de Bruxelles ,Juillet 2010.

[13] G. D. Gersem, Kinaesthetic Feedback and Enhanced Sensitivity in Robotic Endoscopic Telesurgery, Thesis on the Katolic University of Leuven, Belgium, February 2005.

[14] W. Zerrad, Téléopération avec Retour d'Effort pour la Chirurgie Mini-Invasive, Thèse de Doctorat en Génie Informatique, Automatique et Traitement du Signal à l'université de Montpellier II, Décembre 2007.

[15] A. Peer and H. Bleuler, "A Novel Bilateral Position-Position Control Architecture for Teleoperation Systems," in IEEE Transactions on Robotics, vol. 22, no. 3, pp. 505-518, June 2006, doi : 10.1109/TRO.2006.874694.

[16] D. J. Lee, K. C. Kim, and J. W. Choi, "Stability analysis and control design of bilateral teleoperation systems with time-varying delay," in IEEE Transactions on Industrial Electronics, vol. 55, no. 11, pp. 4053-4060, Nov. 2008, doi : 10.1109/TIE.2008.927853.

[17] M. A. Peshkin et J. E. Colgate, "Cobots : Robots for Collaboration with Human Operators," in Journal of Intelligent and Robotic Systems, vol. 19, no. 3, pp. 155-176, 1997, doi : 10.1023/A :1007947913388.

[18] H. Li and M. Tomizuka, "Bilateral Control with Time Delay : A Summary of Existing Research," in Proceedings of the 2004 American Control Conference, Boston, MA, USA, June 30 - July 02, 2004, pp. 2550-2555, doi : 10.23919/ACC.2004.1384058.

[19] M. A. Peshkin and J. E. Colgate, "Delay Effects on Teleoperation : Causes and Cures," in Proceedings 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146), Leuven, Belgium, May 1998, pp. 880-885, doi : 10.1109/ROBOT.1998.677279.

[20] S. Kim and K. S. Kim, "Robust Tracking Control of Teleoperation Systems With Time-Varying Delay," in IEEE Transactions on Control Systems Technology, vol. 18, no. 4, pp. 866-876, July 2010, doi : 10.1109/TCST.2009.2026914.

[21] H. Lee, Y. Sunwoo and J. Kim, "Bilateral Control for Teleoperation Systems With Time Delay : A Predictive Approach," in IEEE Transactions on Industrial Electronics, vol. 58, no. 3, pp. 1069-1078, March 2011, doi : 10.1109/TIE.2010.2051223.

[22] A. Behal, M. Spong and M. W. Spong, "Neural Network-Based Control of Teleoperators With Time Delay," in IEEE Transactions on Neural Networks, vol. 14, no. 1, pp. 137-150, Jan. 2003, doi : 10.1109/TNN.2002.804265.

[23] M. A. Peshkin and J. E. Colgate, "Delay Effects on Teleoperation : Human Performance and Modeling," in Proceedings 2001 ICRA. IEEE International Conference

on Robotics and Automation (Cat. No.01CH37164), Seoul, South Korea, May 2001, pp. 3759-3764, doi : 10.1109/ROBOT.2001.932819

[24] N. Siddique, Intelligent Control : A Hybrid Approach Based on Fuzzy Logic, Neural Networks and Genetic Algorithms, Springer, 2014.

[25] https :www.ferdinandpiette.com/blog/2011/08/les-systemes-flous-le-fonctionnement/

[26] Li, Donghong, et al. "A Novel Fuzzy Logic Control Approach Based on an Improved Genetic Algorithm for a Permanent Magnet Synchronous Motor Drive System." IEEE Access, vol. 9, 2021, pp. 66098-66113, doi : 10.1109/ACCESS.2021.3076470.

[27] McCulloch, Warren S., and Walter Pitts. "A Logical Calculus of Ideas Immanent in Nervous Activity." The Bulletin of Mathematical Biophysics, vol. 5, no. 4, 1943, pp. 115-133, doi : 10.1007/BF02478259

[28] Haykin, Simon. "Neural Networks and Learning Machines." Pearson Education, 2009.

[29] Zhang, X., Wu, L., and Zhang, Y. (2020). A Comprehensive Survey on Recurrent Neural Networks. arXiv preprint arXiv :2001.00109

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

[31] Jang, Jyh-Shing Roger. "Neuro-Fuzzy and Soft Computing : A Computational Approach to Learning and Machine Intelligence." Prentice Hall, 1997.

[32] Jang, J.-S. R., Sun, C.-T., Mizutani, E. (1997). "Neuro-Fuzzy and Soft Computing : A Computational Approach to Learning and Machine Intelligence." Prentice Hall.

[33] Arduino(n.d.) "Arduino Due." Retrieved from https :www.arduino.cc/en/Main/ArduinoBoardDue

Résumé

La téléopération est une technique avancée permettant à un opérateur humain de contrôler à distance un système robotique. Le système se compose d'un robot maître qui envoie des commandes à un robot esclave. Ce dernier, situé à distance, est chargé d'exécuter les tâches spécifiées par l'opérateur. Ce projet a pour objectif de développer un contrôleur basé sur une carte Arduino afin de garantir un contrôle précis de la position du robot esclave. Ce contrôle précis doit être maintenu malgré le délai fixe de transmission des données entre le robot maître et le robot esclave. L'utilisation de la carte Arduino permet d'améliorer la précision et la fiabilité du système de téléopération, en compensant les effets du retard de transmission et en assurant une performance optimale du robot esclave.

Abstract

Teleoperation is an advanced technique that allows a human operator to remotely control a robotic system. The system consists of a master robot that sends commands to a slave robot, which is located at a distance and tasked with executing the specified tasks. This project aims to develop a controller based on an Arduino board to ensure precise position control of the slave robot, despite the fixed data transmission delay between the robots. The use of the Arduino board enhances the precision and reliability of the teleoperation system by compensating for the effects of transmission delay and ensuring optimal performance of the slave robot.