RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ MOULOUD MAMMERI DE TIZI-OUZOU

FACULTÉ DE GENIE ELECTRIQUE ET DE L'INFORMATIQUE

DÉPARTEMENT D'AUTOMATIQUE

THÈSE

Présentée pour obtenir le diplôme de

DOCTORAT $3^{ème}$ CYCLE LMD

Spécialité : AUTOMATIQUE

Par :

Hocine KHATI

THÈME

Commande d'une Architecture de Téléopération par la Carte FPGA

DEVANT LE JURY :

Président	Redouane KARA	*Professeur, Université Mouloud MAMMERI de Tizi-Ouzou*
Rapporteur	Rabah MELLAH	*Professeur, Université Mouloud MAMMERI de Tizi-Ouzou*
Examinateur	Mehammed DAOUI	*Professeur, Université Mouloud MAMMERI de Tizi-Ouzou*
Examinateur	Farid FERGUENE	*Professeur, Université Houari BOUMEDIENE d'Alger*
Invité	Ali BILEK	*Professeur, Université Mouloud MAMMERI de Tizi-Ouzou*

ANNÉE : 2020

Remerciements

Je remercie d'abord DIEU le tout puissant pour la santé, le courage et la foi qu'il m'a donnés pour arriver à ce jour.

Je tiens, tout d'abord, à exprimer ma gratitude, ma plus grande reconnaissance et mon profond respect à mon directeur de thèse, Monsieur Rabah MELLAH, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, qui a dirigé mes travaux de recherche et qui m'a apporté son savoir scientifique indéniable, sa confiance, ses judicieux conseils et sa disponibilité, tout au long de l'élaboration de ce travail de thèse. Qu'il trouve ici l'expression de ma profonde reconnaissance.

Je tiens particulièrement à remercier le chef de département d'Automatique, Monsieur Mohand Achour TOUAT, Maître de conférences à l'université Mouloud MAMMERI de Tizi-Ouzou, pour sa disponibilité, son soutien permanent, ses encouragements et ses conseils précieux. Je lui en suis très reconnaissant.

Ma reconnaissance va également à Monsieur Ali BILEK, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, qui m'a beaucoup aidé pour l'élaboration de mon travail expérimental.

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

Je souhaite ensuite remercier Monsieur Redouane KARA, Professeur à l'université Mouloud MAMMERI de Tizi-Ouzou, qui m'a fait l'honneur de présider le jury de cette thèse. Ses enseignements et ses expertises m'ont permis de progresser et d'orienter mes activités de recherche. J'ai été honoré de l'avoir dans mon jury de thèse.

Je tiens à remercier Monsieur Mehammed DAOUI, Professeur à l'université Mouloud

MAMMERI de Tizi-Ouzou, d'avoir accepté de juger et d'évaluer mon travail de thèse.

Je tiens à remercier également Monsieur Farid FERGUENE, Professeur à l'université des sciences et des technologies Houari Boumediene d'Alger, de m'avoir fait l'honneur d'examiner mon travail.

Mes sincères remerciements vont aussi mes très chers amis T. Hand et Y. Ouali pour leurs soutiens et leur patience durant toute la période de préparation de cette thèse. Je remercie également mes amis : H. Amirouche, N. Amine, H. Ammar, I. Menad, K. Ali, D. Lynda, H. Sadjia, B. Melissa, R. Djamila, K. Sarah, B. Lounas, K. Mohamed, K. Tarik et H. Hocine pour leurs encouragements quotidiens.

Mes remerciements les plus chaleureux vont à mes chers parents et toute ma famille pour leurs encouragements, leur patience, et leur grand soutien durant toutes ces années de préparation de ce travail de thèse.

Je remercie tous ceux qui ont contribué, de près ou de loin, à la réalisation de cette thèse.

Notations

Acronymes

FPGA :	Field Programmable Gate Array
PID :	Proportional Integral Derivative
ANFIS :	Adaptive Neuro-Fuzzy Inference System
ASIC :	Application-Specific Integrated Circuit
HDL :	Hardware Description Languag
CLB :	Configurable Logic Block
IOB :	Input Output Block
VHSIC :	Very High Speed Integrated Circuit
SRAM :	Static Random Access Memory
EPROM :	Erasable Programmable Read-Only Memory
EEPROM :	Electically Erasable Programmable Read-Only Memory
ROM :	Read-Only Memory
DSP :	Digital Signal Processor
SoC :	System on Chips
DCM :	Digital Clock Manager
USB :	Universal Serial Bus
CAN :	Controller Area Network
PCI :	Peripheral Component Interconnect
SPI :	Serial Peripheral Interface
I2C :	Inter-Integrated Circuit
LUT :	Look Up Table
PLL :	Phase-Locked Loop
CPLD :	Complex Programmable Logic Device
ARM :	Advanced RISC Machines
RISC :	Reduced Instruction Set Computing
PS :	Processor System
PL :	Programmable Logic

SDRAM :	Synchronous Dynamic Random Access Memory
DDR :	Double Data Rate
AC :	Alternating Current
DC :	Direct Current
JTAG :	Joint Test Action Group
USB OTG :	Universal Serial Bus On-The-Go
UART :	Universal Asynchronous Receiver Transmitter
GTX :	Gigabit Transceiver
LPC :	Low Pin Count
FMC :	FPGA Mezzanine Card
AMS :	Analog and Mixed Systems
HDMI :	High Definition Multimedia Interface
VGA :	Video Graphics Array
OLED :	Organic Light-Emitting Diodes
FF :	Flip-Flop
QSPI :	Quad Serial Peripheral Interface
FIFO :	First In First Out
SCSI :	Small Computer System Interface
SATA :	Serial Advanced Technology Attachment
XADC :	Xilinx Analog-to-Digital Converter
RTL :	Register Transfer Level
HLS :	High-Level Synthesis
IP :	Intellectual Property
RNA :	Réseau de Neurones Artificiels
ADC :	Analog-to-Digital Converter
PWM :	Pulse Width Modulation
CC :	Courant Continu

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La manipulation des produits dangereux a été à l'origine de la téléopération [1]. Dans les années 1940 à 1950, Raymond C. Goertz a créé des systèmes permettant aux humains de manipuler des matières radioactives [2]. Les premiers systèmes de téléopération étaient électriques, utilisés pour activer des moteurs et des axes grâce à un ensemble d'interrupteurs. En raison que ces manipulateurs étaient lents et difficiles à exploiter, Goertz est amené à construire des robots maître-esclave liés mécaniquement, néanmoins ces systèmes nécessitaient l'utilisation de dispositifs cinématiquement identiques avec une distance limitée entre l'opérateur et l'environnement [3]. Par la suite, Goertz a développé d'autres manipulateurs bilatéraux modernes couplés électriquement avec retour d'effort [2, 4]. A partir des années 1960, les effets du retard de transmission sur les systèmes de téléopération ont été devenus un sujet de recherche [1, 2]. Dans les années 1990, l'augmentation des capacités de calcul des ordinateurs a permis l'intégration de la réalité virtuelle à la téléopération. Aujourd'hui, grâce à l'évolution de l'informatique et de la télécommunication numérique, les systèmes de téléopération sont largement développés dans divers domaines tels que le domaine médical et l'exploration spatiale [5].

Un système de téléopération bilatérale est un dispositif électromécanique composé d'un robot maître et d'un robot esclave, où les informations de position et de force sont échangées entre eux via un canal de communication, de telle sorte à ce que le robot esclave suive les mouvements du maître, qui est manipulé par un opérateur humain [6–8]. Idéalement, un système de téléopération bilatérale doit être complètement transparent, afin que l'esclave reproduise les mouvements du maître avec fidélité, et que l'opérateur humain ait l'impression comme s'il réalisait la tâche à la place de l'esclave. Néanmoins, afin de

garantir une telle transparence, les contrôleurs relatifs aux maître et l'esclave doivent être très performants, car les robots maître et esclave présentent des systèmes non linéaires complexes, avec des incertitudes dynamiques particulièrement élevées lorsque l'esclave est en contact avec l'environnement distant, de plus les performances de suivi sont difficiles à obtenir lorsque la position de l'esclave et la force de réflexion sur le maître sont contrôlées simultanément [9], d'autre part l'opérateur humain et l'environnement interagissent respectivement avec le maître et l'esclave, et leurs modèles dynamiques ne peuvent pas être disponibles en raison du comportement imprévisible de l'opérateur humain. Pour cela, les contrôleurs doivent être robustes et adaptatifs pour faire face à ces contraintes de contrôle. Le contrôle adaptatif en utilisant les techniques de l'intelligence artificielle (logique floue, réseaux de neurones, ...,etc) a fait objet de plusieurs travaux de recherche dans le domaine de la téléopération [10–13]. La combinaison de la logique floue avec les réseaux de neurones a été reconnue comme une solution prometteuse pour compenser les incertitudes dynamiques des systèmes robotiques. L'utilisation de la logique floue pour le contrôle des systèmes non linéaires permet de fournir de bonnes performances grâce à sa capacité descriptive élevée par des variables linguistiques. Néanmoins, aucune méthode n'existe pour déterminer la base des règles et les fonctions d'appartenance [14]. En outre, les réseaux de neurones sont largement utilisés dans la résolution des problèmes du contrôle non linéaire, en raison de leur structure flexible, leur parallélisme de calcul ainsi que leur capacité d'apprentissage. En revanche, ils présentent des difficultés pour la détermination de la structure du réseau tels que les nombres de neurones et de couches cachées [15]. Par conséquent, la combinaison de la logique floue avec les réseaux de neurones en une seule structure permet de construire des contrôleurs neuro-flous plus efficaces, et cela en tirant profit de la capacité d'inférence du raisonnement flou et du parallélisme de calcul des réseaux neuronaux, où ces derniers sont utilisés pour ajuster en temps réel les paramètres des règles floues et des fonctions d'appartenance, en utilisant un algorithme d'apprentissage. Parmi les systèmes neuro-flous adaptatifs les plus populaires et performants, on trouve la structure ANFIS (Adaptive Neuro-Fuzzy Inference System), proposée par Jang [16], qui est capable d'optimiser et d'ajuster en ligne les paramètres de la pré-

misse et de la conséquence du réseau neuro-flou à l'aide d'un algorithme d'apprentissage basé sur la méthode de la descente de gradient et du filtre de Kalman étendu. Néanmoins, l'utilisation d'une telle structure adaptative requière un temps de calcul très élevé pour assurer une bonne précision, et son implémentation en pratique nécessite alors des cartes de contrôle très puissantes afin de minimiser le temps de calcul de l'algorithme en vue de converger très rapidement vers les performances désirées.

Avec le grand développement des semi-conducteurs et de la micro-électronique, plusieurs solutions matérielles pour l'implémentation des algorithmes de contrôle sont apparues telles que les FPGAs (Field-Programmable Gate Array), les ASICS (Application-Specific Integrated Circuit) et les DSPs (Digital signal processor). Ces solutions offrent de grandes capacités de traitement de données, ce qui permet de réduire le temps de calcul des algorithmes à implémenter. Bien que les DSPs et les ASICs peuvent résoudre le problème de complexité de calcul, ils présentent l'inconvénient d'être très couteux avec également une grande consommation d'énergie. Parmi les nouvelles solutions matérielles les plus efficaces utilisées actuellement dans le domaine de contrôle et de traitement de signal, on cite les circuits FPGA. Ces derniers offrent un traitement parallèle de données et une vitesse de calcul élevée avec une faible consommation d'énergie [17, 18]. Les FPGAs fournissent des temps d'échantillonnage précis par rapport aux microprocesseurs [19], de plus, il peuvent exécuter plusieurs processus à la fois avec une fréquence de fonctionnement élevée, contrairement aux microprocesseurs ordinaires, qui ne peuvent exécuter qu'un seul processus à la fois avec une fréquence inférieure [20]. Le FPGA est donc un circuit approprié pour effectuer un contrôle de téléopération bilatérale [21], car il permet d'accélérer l'algorithme de contrôle ce qui peut augmenter les performances de la téléopération. Cependant, la méthodologie de conception de l'algorithme de contrôle peut affecter négativement les performances du circuit logique généré en terme de surface occupée par la conception sur le FPGA, car certains algorithmes complexes nécessitent plusieurs ressources matérielles [17]. La consommation des ressources matérielles est plus optimale dans les conceptions à virgule fixe que celles à virgule flottante [22], cependant, la spécification optimale des longueurs de mots binaires reste une tâche complexe qui peut

affecter la précision lors des calculs effectués par l'algorithme de contrôle [17]. Pour cela, la société MathWorks offre des outils de conception et de développement de haut niveau qui prennent en charge une large gamme de cartes FPGA, particulièrement ceux de la famille Zynq-7000 du constructeur Xilinx (utilisés dans le cadre de ce travail). En outre, le logiciel MATLAB permet à l'utilisateur de développer et d'implémenter des conceptions FPGA, sans avoir recours au langage de programmation VHDL (VHSIC : Very High Speed Integrated Circuit). L'environnement Simulink de MATLAB permet, à l'aide de l'outil " Fixed-Point Tool " d'automatiser la spécification du type de données à virgule fixe à partir d'un modèle Simulink à virgule flottante [23], en respectant les règles de l'arithmétique en virgule fixe tout en optimisant les longueurs des mots binaires, par la suite, l'outil " HDL Coder " permet de générer le code VHDL optimisé en termes de ressources matérielles et de temps d'exécution [23], qui sera enfin implémenté sur la carte FPGA directement via l'environnement Simulink. L'utilisation d'une telle méthodologie permet de réduire de façon considérable le temps de conception des algorithmes à implémenter, tout en offrant une flexibilité de développement des conceptions.

Problématique

Le contrôle des systèmes de téléopération fait actuellement l'objet de plusieurs travaux de recherche [17, 19, 24–29]. Les robots maître et esclave étant des systèmes non linéaires, avec une dynamique parfois inconnue en raison du retards de communication entre les deux stations, ou particulièrement lorsque l'esclave et en contact avec l'environnement distant, les contrôleurs doivent être adaptatifs et robustes afin de faire face à ces contraintes, et atteindre une téléopération à haute performance en termes de transparence et de stabilité, néanmoins l'utilisation de tels contrôleurs en pratique nécessite des cartes de commande très puissantes en raison du temps de calcul important que ces contrôleurs utilisent pour converger rapidement vers les performances désirées, tout en garantissant une meilleure précision.

Contribution

Dans ce travail de thèse, nous proposons un contrôle adaptatif neuro-flou pour commander un système de téléopération bilatérale à un degré de liberté, en utilisant une carte FPGA. L'utilisation d'un tel contrôle permettra d'obtenir de bonnes performances de transparence et de stabilité, et cela en ajustant les paramètres de la conséquence du réseau neuro-flou en ligne (en temps réel), tout en tirant profit des avantages du FPGA en termes de parallélisme de calcul et de la fréquence de fonctionnement élevée. Pour ce faire, nous allons opter pour le logiciel MATLAB pour le développement, la conception et l'implémentation de l'algorithme de commande proposé, en utilisant les fonctionnalités de l'environnement Simulink à savoir les outils "Fixed-Point Tool" et "HDL Coder". Une telle méthodologie nous permettra de réduire considérablement le temps de conception et d'obtenir un algorithme de contrôle précis et optimisé en terme de ressources matérielles consommées par le FPGA.

Organisation de la thèse

Le présent manuscrit est organisé comme suit.
L'introduction générale décrit l'objectif et les motivations de la thèse, et donne un aperçu général sur le travail effectué.
Dans le premier chapitre, nous avons présenté de façon générale les systèmes de téléopération, ainsi que le formalisme mathématique qui les décrit. Par la suite, nous avons présenté les trois architectures de contrôle les plus utilisées en pratique et leurs critères de performances. Dans la dernière partie du chapitre, nous avons abordé la méthode de l'analyse de stabilité d'un système de téléopération par la passivité, et nous avons donné les critères qui définissent sa transparence idéale.
Le deuxième chapitre est consacré à la description des circuits FPGA ainsi que les outils permettant leur programmation. Dans cette partie de la thèse, nous avons cité d'abord les technologies de programmation des FPGAs, par la suite nous avons présenté les FPGAs du constructeur Xilinx, plus particulièrement ceux de la famille Zynq-7000, utilisés dans le

cadre de ce travail de thèse. La dernière partie du chapitre est consacrée à la présentation des outils de développement et d'implémentation des conceptions FPGA ainsi que leurs avantages et leurs insuffisances.

Dans le chapitre trois, nous avons présenté le système de téléopération réalisé dans le cadre de ce travail de thèse, ainsi que les outils de l'environnement MATLAB utilisés pour l'identification des modèles des actionneurs afin de synthétiser deux contrôleurs PID, à savoir les outils "System Identification Toolbox" pour l'identification des modèles, et "PID Tuner" pour la synthèse des contrôleurs PID.

Dans le quatrième chapitre, la méthode de contrôle adaptatif proposée pour commander le système de téléopération à un degré de liberté est présentée. Cette méthode consiste à concevoir un contrôleur neuro-flou de type ANFIS pour chaque système (maître et esclave), selon la stratégie de commande adoptée (position-position et quatre canaux). Les correcteurs développés ajustent en ligne les paramètres de la conséquence du réseau neuro-flou afin de s'adapter aux variations dynamiques des systèmes maître et esclave, et cela en tirant profit du parallélisme de calcul du FPGA et sa fréquence de fonctionnement élevée.

Le chapitre cinq est dédié à l'implémentation des contrôleurs proposés sur la carte FPGA Zedboard à l'aide du logiciel MATLAB, via les outils "Fixed-Point Tool" et "HDL Coder" de l'environnement Simulink. La dernière partie du chapitre présente les résultats expérimentaux obtenus en utilisant les contrôleurs classiques PID et les contrôleurs adaptatifs ANFIS, selon chaque stratégie de commande adoptée (position-position et quatre canaux).

Enfin, ce travail de thèse est clôturé par une conclusion et quelques perspectives.

Chapitre 1

Présentation des différentes architectures de téléopération

1.1 Introduction

Un système de téléopération est un dispositif qui permet à un opérateur humain de manipuler des objets à distance [30] tout en l'éloignant de l'environnement de travail et des machines qu'il manipule. La téléopération élimine ainsi les risques liés aux environnements dangereux et à la manipulation des substances toxiques [7, 31].

Un système de téléopération comprend un robot maître, qui reçoit les commandes de l'opérateur humain, un robot esclave qui est chargé de réaliser la tâche et un canal de communication pour communiquer entre les deux robots [30]. Son fonctionnement repose sur l'échange d'informations de position et de force entre le maître et l'esclave [31]. Ce système est dit bilatéral si la force de réaction de l'environnement sur l'esclave est transmise au maître [8, 32, 33]. Ce retour d'effort permet à l'opérateur humain d'évaluer la situation du robot esclave qu'il manipule [24].

Dans une téléopération bilatérale, il est souhaitable que le système soit stable est transparent [34] afin que l'esclave reproduise les mouvements du maître avec fidélité et que l'opérateur se sente directement en contact avec l'environnement distant, néanmoins la stabilité et la transparence restent deux objectifs concurrents en raison du retard de trans-

mission existant entre le poste maître et celui de l'esclave.
Les architectures de contrôle de base des systèmes de téléopération comprennent deux ou quatre canaux [35]. Elles sont classées généralement par rapport au nombre et au type d'informations échangées entre le maître et l'esclave [31]. Leurs performances sont alors évaluées selon cet échange d'informations. Dans ce chapitre, nous allons présenter une description générale sur les systèmes de téléopération, leur analyse à travers l'évaluation des deux critères de performances (stabilité et transparence), ainsi que les architectures de contrôle les plus utilisées en pratique et leurs performances.

1.2 Définitions

1.2.1 Téléopération

La téléopération désigne l'ensemble des principes et techniques qui permettent à un opérateur humain d'exécuter une tâche à distance, au moyen d'un système robotique d'intervention, contrôlé à partir d'une station de commande, via un canal de communication.

1.2.2 Système de téléopération

C'est un système commandé à distance. Il permet à un opérateur humain de contrôler un robot situé à une distance lointaine à travers un canal de communication. Son fonctionnement repose sur l'échange d'informations de position et de force entre le maître, qui est la télécommande de l'utilisateur, et l'esclave qui est le robot réalisant effectivement la tâche. Un système de téléopération est dit unilatéral si les informations de position et de force sont uniquement transmises du site maître au site esclave. Il est dit bilatéral si l'environnement distant fournit une information d'effort en retour à l'opérateur à travers une interface haptique. Ce retour d'information permet à l'opérateur humain d'évaluer la situation du robot à manipuler [35]. L'opérateur humain est en contact direct avec le dispositif maître, tandis que le dispositif esclave est en contact direct avec l'environnement. L'opérateur humain applique une force sur le maître (un joystick par exemple), générant ainsi une position désirée, qui est envoyée par le canal de communication au

dispositif esclave. Le robot esclave exécute la commande reçue. De même, les forces d'interaction de l'esclave avec l'environnement sont renvoyées par le canal de communication vers l'interface haptique (un joystick avec retour de force par exemple), ou par un retour visuel ou sonore, vers l'opérateur humain. Ce retour d'information permet à l'opérateur humain d'évaluer la situation du dispositif esclave et donc de modifier sa force appliquée. La Figure 1.1 illustre un système de téléopération bilatérale.

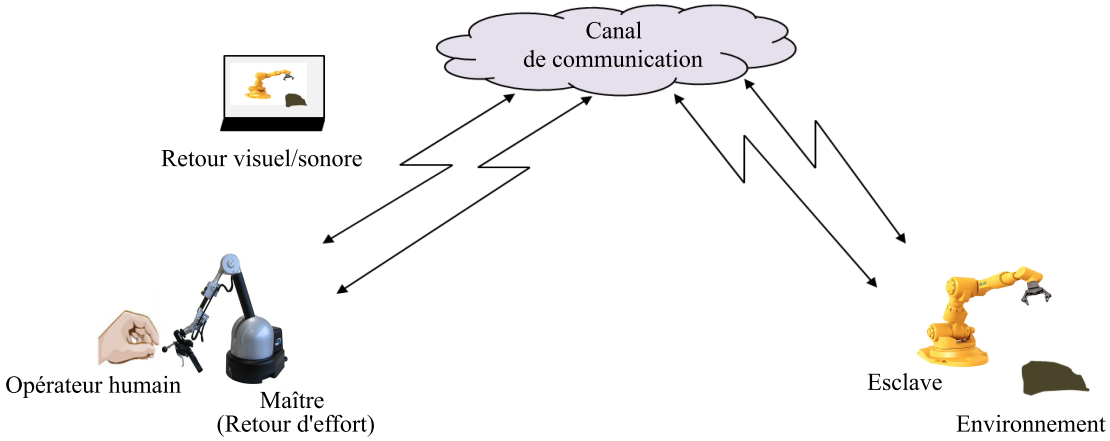


FIGURE 1.1: Représentation générale d'un système de téléopération bilatérale.

1.3 Structure générale d'un système de téléopération

Un système de téléopération (Figure 1.2) est constitué de [36, 37] :

Opérateur humain : Il interagit avec le système maître à travers l'interface haptique. Ses actions sont transportées vers le robot esclave. En téléopération bilatérale, l'opérateur reçoit un retour d'effort, qui représente les interactions du robot avec l'environnement distant, et éventuellement un retour visuel.

Interface maître : C'est l'appareil physique avec lequel interagit l'opérateur humain. Cet appareil peut acquérir toutes les données nécessaires sur les actions de l'opérateur, qui seront exécutées par l'esclave. De plus, cette interface permet à l'opérateur, à travers un bloc haptique ou un moniteur, de recevoir toutes les informations appropriées sur le site distant.

Manipulateur esclave : C'est le robot qui se charge d'interagir physiquement avec l'environnement distant. Il a pour mission d'effectuer la tâche requise en reproduisant les mouvements de l'opérateur sur le dispositif maître, en plus il doit transmettre, au moyen de divers capteurs qu'il possède, toutes les informations nécessaires à l'opérateur.

Canal de transmission : Il représente le support qui permet la transmission de données (signaux de position et de force) entre le site maître et le site esclave. L'échange d'informations peut s'effectuer au moyen de câbles, radios ou satellites, selon l'emplacement des deux robots et les exigences de la tâche à effectuer. Ce support est caractérisé par un retard de transmission dû au traitement parallèle de données transmises des deux postes du système de téléopération, et peut être aussi une source de perte d'informations due aux bruits de communication.

Contrôleurs : Ce sont les lois de commande appliquées aux robots maître et esclave selon différentes architectures de contrôle. Elles permettent de calculer les couples nécessaires pour les différents actionneurs, afin d'assurer la poursuite des mouvements du maître par l'esclave, et les forces de l'environnement distant par le maître. Ces lois de commande définissent généralement une grande partie des performances d'un système de téléopération, elles doivent être donc les plus robustes possibles face aux retards de transmission causés par le canal de communications, ainsi qu'aux incertitudes cinématiques et dynamiques liées au système de téléopération, particulièrement lorsque le robot esclave interagit avec l'environnement inconnu.

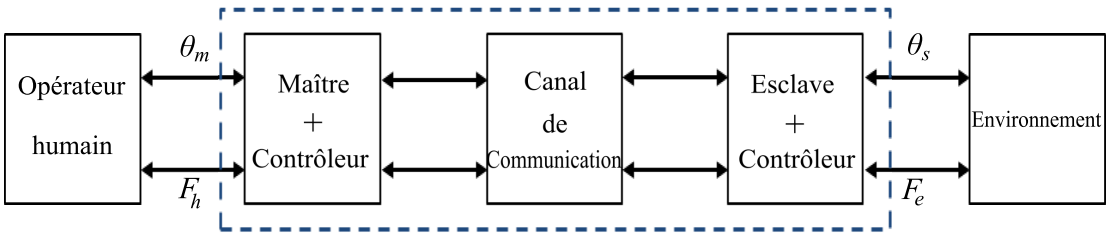


FIGURE 1.2: Représentation d'un système de téléopération sous forme d'un réseau.

Environnement distant : Il est composé de différents objets qui sont manipulés physiquement par le manipulateur esclave suivant les ordres de l'opérateur. Ces objets peuvent

être structurés ou complètement inconnus.

1.4 Représentation mathématique d'un système de téléopération

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

Les systèmes de téléopération sont généralement constitués de dispositifs avec plusieurs degrés de liberté [38] néanmoins, dans la littérature, un modèle a été défini pour un système avec un degré de liberté afin de mettre le problème simple à étudier.
Mathématiquement, on considère un système de téléopération à un degré de liberté avec deux sous-systèmes maître et esclave, échangeant les signaux de vitesses, de forces (couples) et d'accélérations. La dynamique du dispositif maître et du dispositif esclave est donnée par les équations suivantes [31, 35, 38, 39]

$$\begin{cases} M_m\ddot{\theta}_m + B_m\dot{\theta}_m = F_m + F_h \\ M_s\ddot{\theta}_s + B_s\dot{\theta}_s = F_s - F_e \end{cases} \tag{1.1}$$

Où M_m et B_m, M_s et B_s sont respectivement les masses et les frottements visqueux des robots maître et esclave. θ_m est la position du maître et θ_s est celle de l'esclave, F_h est la force (couple) appliquée par l'opérateur sur le dispositif maître, F_e est la force (couple) exercée par l'environnement sur l'esclave. F_m et F_s sont les forces (couples) délivrées par les actionneurs pour commander respectivement 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 dynamique suivante [38] :

$$F_s = M_w\ddot{\theta}_s + B_w\dot{\theta}_s + C_w\theta_s \tag{1.2}$$

Où M_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 également par l'équation suivante [38] :

$$F_{op} - F_m = M_{op}\ddot{\theta}_m + B_{op}\dot{\theta}_m + C_{op}\theta_m \tag{1.3}$$

Où M_{op}, B_{op} et C_{op} représentent respectivement la masse, les frottements visqueux et la rigidité de l'opérateur, alors que que F_{op} désigne la force générée par les muscles de l'opérateur.

1.4.2 Représentation hybride

Un système de téléopération peut être modélisé par un réseau à deux ports, dérivé des réseaux électriques [40], reliant la force (couple) de l'opérateur F_h et sa position θ_m aux variables de force F_e et de position θ_s du manipulateur esclave.

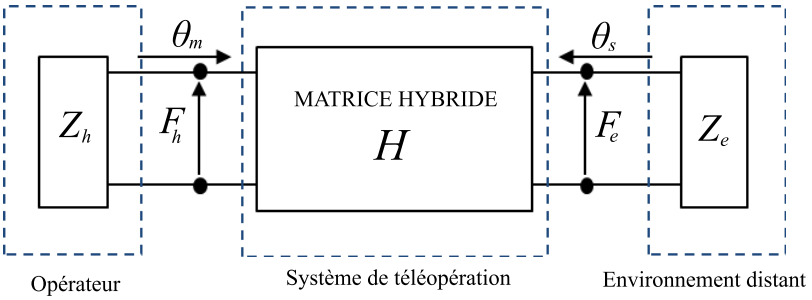


FIGURE 1.3: Représentation d'un système de téléopération par un modèle à deux ports.

La Figure 1.3 illustre un système de téléopération présenté par un réseau à deux ports sous une forme hybride, reliant l'opérateur humain caractérisé par une impédance Z_h à l'environnement distant d'impédance Z_e. Ce réseau est décrit par la relation suivante, proposée par Hannaford [40] :

$$\begin{bmatrix} F_h \\ \theta_s \end{bmatrix} = \underbrace{\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}}_{H} \begin{bmatrix} \theta_m \\ F_e \end{bmatrix} \tag{1.4}$$

La matrice H est appelée matrice d'admittance hybride, qui caractérise complètement le système de téléopération à savoir la dynamique du maître et de l'esclave, les contrôleurs et les communications.

Les paramètres h_{ij} de la matrice hybride H définissent les quatre critères de performances, qui évaluent la transparence d'un système de téléopération. Ces critères sont définis comme suit [31, 37, 40, 41] :

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 égal à zéro.

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

Pour $\theta_s = 0$ (Contact rigide) :

• $-\frac{F_e}{F_h} = F_T = \frac{h_{21}}{h_{12}h_{21} - h_{11}h_{22}}$: Représente le tracking en force lors d'un contact rigide.

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

1.5 Caractéristiques

Un système de téléopération bilatérale peut être caractérisé par deux critères essentiels : la stabilité et la transparence [37].

1.5.1 Stabilité

La stabilité est un critère primordial pour un fonctionnement utile et sûr de la téléopération [42]. Elle se définit par le fait qu'un déplacement/effort borné effectué par l'opérateur sur l'interface maître, entraine un déplacement/effort borné réalisé par le robot esclave au niveau de l'environnement distant [37]. Son étude est un problème délicat en raison, d'une part de l'interaction avec un opérateur humain dont le comportement est incertain, et avec un environnement caractérisé par de fortes non-linéarités, et d'autre part

en raison des délais de transmission causés par le canal de communication, qui mènent à l'instabilité du système. Les lois de commande du système de téléopération doivent être donc robustes face à ces incertitudes dynamiques du système maître-esclave et des retards de transmission, pour assurer la stabilité.

1.5.2 Transparence

L'objectif pour un système de téléopération est d'atteindre la transparence, et cela en reflétant fidèlement les interactions entre l'esclave et l'environnement sur l'opérateur. Ce dernier doit être capable de ressentir les efforts comme s'il réalisait directement la tâche à la place de l'esclave [31].
Dans [38], les auteurs définissent la transparence parfaite par l'égalité entre les réponses de position et de force du robot maître et celles du robot esclave, autrement dit par la capacité du système à assurer des suivis parfaits de forces et de positions [31]. Cela revient à écrire :

$$\begin{cases} F_h(t) = -F_e(t) \\ \theta_s(t) = \theta_m(t) \end{cases} \tag{1.5}$$

Dans ce cas, la matrice hybride idéale est définie comme suit [40] :

$$H_{idéale} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{1.6}$$

1.6 Architectures de contrôle des systèmes de téléopération

Dans cette section, nous présentons les différentes architectures de contrôle bilatéral rapportées dans la littérature. Elles sont classées par rapport au nombre et au type d'informations transmises entre le site maître et le site esclave à travers le canal de communication [41, 43]. Les stratégies de commande implémentant deux canaux sont basées sur l'échange d'un seul signal (position ou force) dans chaque sens, alors que les stratégies implémentant quarte canaux reposent sur l'échange de deux signaux dans chaque sens,

permettant ainsi d'atteindre une meilleure transparence [31, 36, 37].

Dans la suite de cette section, Z_m et Z_s représentent respectivement les impédances du maître et de l'esclave, C_m est le contrôleur du maître et C_s est le contrôleur de l'esclave. Les architectures de contrôle de base des systèmes de téléopération bilatérale comprenant deux ou quatre canaux sont [37, 41] :

- Architecture de contrôle position-position
- Architecture de contrôle force-position
- Architecture de contrôle à quatre canaux

1.6.1 Architecture de contrôle position-position

L'architecture position-position a été la première stratégie de commande bilatérale implémentée sur un système de téléopération [4]. Son fonctionnement repose sur l'échange de signaux de position seulement entre le maître et l'esclave à travers deux canaux de communication C_1 et C_4. 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 (C_m) et de l'esclave (C_s) 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 1.4.

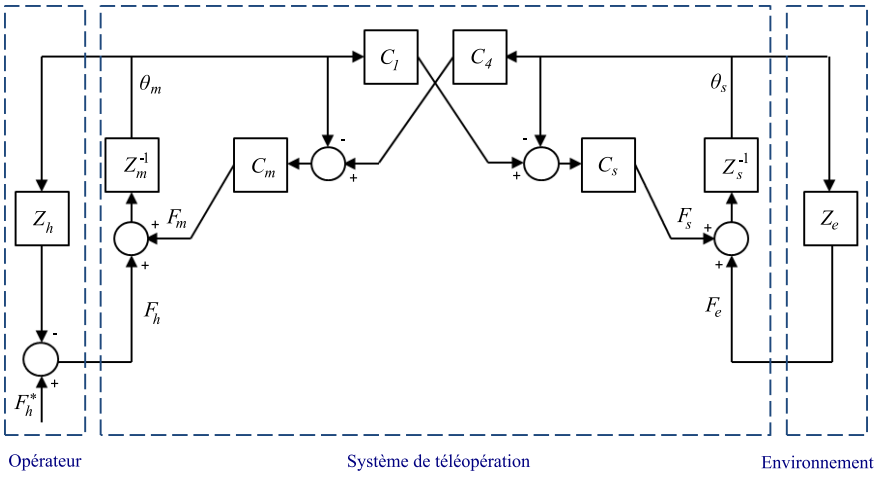


FIGURE 1.4: Schéma de contrôle position-position.

Les quatre critères de performances qui caractérisent cette méthode de contrôle sont définis comme suit [31, 41] :

$$h_{11} = Z_{\min} = Z_m + \frac{Z_s C_m}{Z_s + C_s} \approx Z_m + \frac{C_m}{C_s} Z_s \tag{1.7}$$

$$h_{21} = X_T = \frac{C_s}{Z_s + C_s} \approx 1 \tag{1.8}$$

$$F_T = \frac{Z_m + C_m}{C_s} \approx \frac{C_m}{C_s} \tag{1.9}$$

$$Z_{\max} = Z_m + C_m \approx C_m \tag{1.10}$$

L'avantage principal de cette stratégie de contrôle est sa facilité d'implémentation puisque le retour d'effort est généré seulement par les capteurs de position, aucun capteur de force donc n'est nécessaire. Néanmoins, l'opérateur peut ressentir l'impédance du maître plus celle de l'esclave en mouvement libre [31], comme le montre le premier critère de performances. Cette méthode présente alors l'inconvénient de ne pas atteindre la transparence parfaite [43], cependant, elle assure une bonne stabilité par rapport aux autres architectures de contrôle à deux canaux [31].

1.6.2 Architecture de contrôle force-position

Cette stratégie de contrôle a été largement utilisée dans les systèmes de téléopération bilatérale. Son fonctionnement repose sur l'échange des signaux de position et de force à travers les canaux C_1 et C_2 respectivement, un capteur d'effort est donc nécessaire sur le site distant pour mesurer les forces d'interaction de l'esclave avec l'environnement. Dans cette méthode, la position du maître est transmise au robot esclave, tandis que ce dernier transmit à l'opérateur les efforts qu'il applique sur l'environnement [37]. Le schéma de base de l'architecture de contrôle force-position est présenté dans la Figure 1.5.
Les critères de performances caractérisant cette stratégie de contrôle sont [31, 41] :

$$h_{11} = Z_{\min} = Z_m \tag{1.11}$$

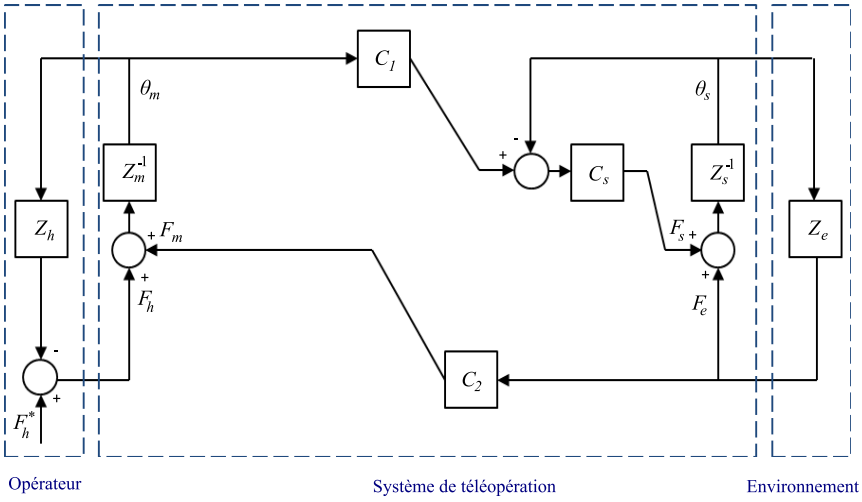


FIGURE 1.5: Schéma de contrôle force-position.

$$h_{21} = X_T = \frac{C_1 C_s}{Z_s + C_s} \tag{1.12}$$

$$F_T = \frac{-C_1 C_s}{Z_m + C_1 C_2 C_s} \tag{1.13}$$

$$Z_{\max} = Z_m + C_1 C_2 C_s \tag{1.14}$$

Dans cette architecture de contrôle, l'opérateur ressent les effets dynamiques du maître. Donc pour avoir de meilleures performances, il faut d'une part utiliser une interface maître avec des caractéristiques dynamiques faibles [31], et d'autre part utiliser un contrôleur de position robuste pour l'esclave afin de maintenir la stabilité du système [41].

1.6.3 Architecture de contrôle à quatre canaux

Ce schéma de contrôle a été proposé par Lawrence [44] afin de garantir une transparence parfaite et une stabilité satisfaisante [37]. Son fonctionnement repose sur l'échange mutuel d'informations de position et de force entre les deux sites à travers les quatre canaux C_1, C_2, C_3 et C_4. La présence d'un capteur de position et un autre de force est donc nécessaire dans chacun des robots maître et esclave. Le schéma de base de Lawrence est présenté dans la Figure 1.6.

En considérant un système de téléopération avec des robots maître et esclave identiques, les quatre critères de performances qui caractérisent cette stratégie de commande sont

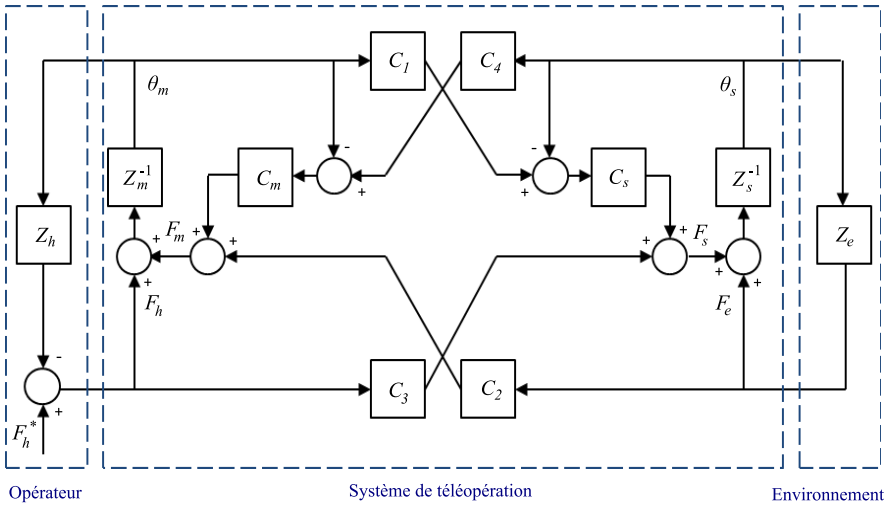


FIGURE 1.6: Schéma de contrôle à quatre canaux.

donnés comme suit [31] :

$$h_{11} = Z_{\min} = Z_m \tag{1.15}$$

$$h_{21} = X_T = 1 \tag{1.16}$$

$$F_T = 1 \tag{1.17}$$

$$Z_{\max} = \infty \tag{1.18}$$

Dans le contrôle à quatre canaux, la transparence idéale est vérifiée en ce qui concerne le tracking en force, le tracking en position et l'impédance maximum transmissible par le système lors d'un contact, néanmoins l'impédance minimum ressentie en mouvement libre présente un défaut, où l'opérateur ressent les effets dynamiques de l'interface maître [31]. Dans la suite de ce travail de thèse, nous allons proposer un contrôleur adaptatif neuro-flou pour commander un système de téléopération à un degré de liberté, en utilisant les deux architectures de contrôle position-position et quatre canaux.

1.7 Étude des systèmes de téléopération

Dans une téléopération bilatérale, il est souhaitable que le système soit complètement transparent de telle sorte à ce que l'opérateur se sente directement en contact avec l'envi-

ronnement éloigné. Cependant, la présence des retards de transmission fait en sorte que la transparence et la stabilité soient deux objectifs concurrents. L'étude d'un système de téléopération se fait alors par l'évaluation de ces deux critères essentiels : La stabilité et la transparence [37].

1.7.1 Stabilité

La stabilité d'un système de téléopération est une contrainte essentielle pour assurer la qualité haptique et la sécurité [31], néanmoins, la présence de délais de transmission liés au canal de transmission mène souvent à l'instabilité. L'opérateur ainsi que l'environnement distant apparaissent comme des systèmes incertains et variables [37], de plus les robots maître et esclave présentent des systèmes non linéaires et complexes, avec une dynamique qui peut être non disponible en raison des incertitudes du modèle [27,32], particulièrement lorsque l'esclave est en contact avec l'environnement inconnu [25]. Les lois de commande doivent être alors robustes face à ces incertitudes dynamiques et cinématiques afin de garantir la stabilité de la téléopération bilatérale.
Pour assurer la stabilité d'un système de téléopération à retour d'effort, la méthode d'analyse de la passivité est généralement utilisée dans la littérature. Dans cette technique qui est issue des réseaux électriques, on cherche à ce que le port d'interaction entre le robot et l'environnement soit passif, ça veut dire ne crée pas d'énergie. Si la passivité de ce port est vérifiée, le système interagira via ce port de façon stable, avec n'importe quel environnement passif. La stabilité du système est donc déduite de sa passivité.

Stabilité par la passivité

La théorie de la passivité est une propriété d'analyse des systèmes dynamiques. Elle est issue des réseaux électriques et repose principalement sur l'échange d'énergie entre les systèmes inter-connectés. Un système est dit passif s'il n'absorbe plus d'énergie qu'il n'en produit [5].
Un système de téléopération peut être modélisé par un ensemble de blocs en cascade,

représentant le maître et l'esclave inter-connectés par un réseau de communication représenté par un quadripôle à deux ports (Figure 1.7), avec une matrice hybride [45, 46].

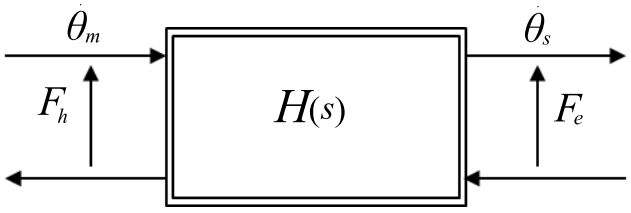


FIGURE 1.7: Représentation d'un système de téléopération sous forme d'un quadripôle.

La représentation hybride du système dans le domaine de Laplace est :

$$\begin{bmatrix} F_h(s) \\ -\dot{\theta}_s(s) \end{bmatrix} = \begin{bmatrix} h_{11}(s) & h_{12}(s) \\ h_{21}(s) & h_{22}(s) \end{bmatrix} \begin{bmatrix} \dot{\theta}_m(s) \\ F_e(s) \end{bmatrix} \tag{1.19}$$

Considérons le réseau à deux ports (Figure 1.7), avec $\dot{\theta}(t)$ le vecteur de vitesse et $F(t)$ le vecteur de force. La puissance totale du réseau est définie comme suit [47] :

$$P(t) = F^T(t)\,\dot{\theta}(t) \tag{1.20}$$

Étant donné que le stockage d'énergie est nul à $t = 0$, le réseau est dit passif s'il satisfait la condition suivante [5, 46, 48] :

$$\int_0^\infty P(t)dt \geq 0 \tag{1.21}$$

C'est-à-dire :

$$\int_0^\infty F^T(t)\dot{\theta}(t)\,dt \geq 0 \tag{1.22}$$

Généralement, l'analyse de la stabilité par la passivité d'un système de téléopération représenté sous forme d'un réseau à deux ports se fait par l'analyse de sa matrice d'impédance [35], ou à travers sa matrice de dispersion.

Dans [46], la théorie de la dispersion (diffusion) a été introduite. Elle repose sur l'étude des signaux d'entrée et de sortie d'un réseau à deux ports. Ce réseau est alors dit passif si l'énergie des signaux d'entrée est totalement diffusée et ne peut pas être supérieure à celle des signaux de sortie [37].

Le système à deux ports illustré dans la Figure 1.7 est décrit par les équations suivantes [46] :

$$F(s) - \dot{\theta}(s) = S(s)\left(F(s) + \dot{\theta}(s)\right) \tag{1.23}$$

Avec : $F(s) = \begin{bmatrix} F_h(s) \\ F_e(s) \end{bmatrix}$ et $\dot{\theta}(s) = \begin{bmatrix} \dot{\theta_m}(s) \\ \dot{\theta_s}(s) \end{bmatrix}$

s représente la variable de Laplace et $S(s)$ la matrice de dispersion, définie par [46] :

$$S(s) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} (H(s) - I)(H(s) + I)^{-1} \tag{1.24}$$

Afin de garantir la passivité du réseau, les signaux diffusés ne peuvent pas avoir une énergie supérieure à celle des signaux incidents ; la règle de la stabilité par la passivité est décrite par le théorème suivant :

Théorème (Anderson et Spong) [46]

Un système à n ports est dit passif si seulement si $\|S(j\omega)\| \leq 1$.

Pour un système de téléopération avec un retard de communication T , la matrice de dispersion devient [35] :

$$S(s) = \begin{bmatrix} 0 & e^{-sT} \\ e^{-sT} & 0 \end{bmatrix} \tag{1.25}$$

Dans [35], l'auteur a montré que la passivité du système est garantie car $\|S(j\omega)\| = 1$.

Par conséquent, le concept de la stabilité par la passivité est largement utilisé dans de nombreuses recherches sur les systèmes de téléopération qui présentent des délais de transmission [38, 44].

1.7.2 Transparence

La transparence est un critère de performances primordial dans un système de téléopération bilatérale. Il s'agit de transmettre le plus fidèlement possible les signaux de position, de vitesse et de force entre le maître et l'esclave, de telle sorte à ce que l'opéra-

teur humain ait l'impression comme s'il réalisait directement les opérations à la place de l'esclave [31].

La Figure 1.8 représente un système de téléopération sous forme d'un réseau.

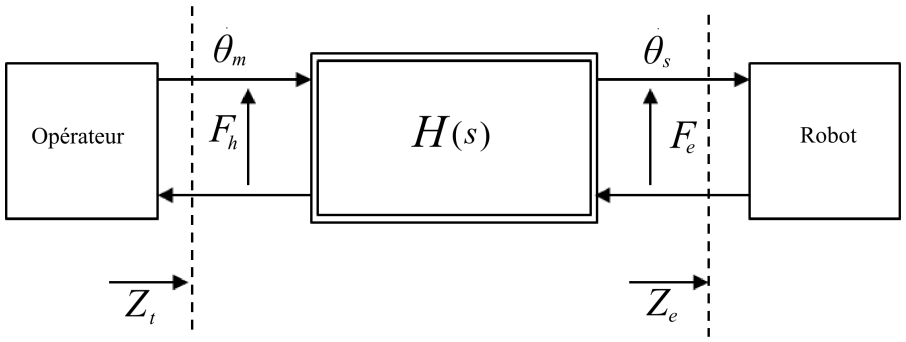


FIGURE 1.8: Représentation d'un système de téléopération sous forme d'un réseau.

Dans [44], Lawrence définit la transparence idéale par l'égalité entre la relation qui lie la force F_h appliquée par l'opérateur sur le maitre avec la vitesse du robot maître $\dot{\theta}_m$, et la relation qui lie la force F_e exercée par l'esclave sur l'environnement avec la vitesse du robot esclave $\dot{\theta}_s$. Cela veut dire que si l'opérateur humain a l'impression comme s'il agissait directement sur place lorsque le manipulateur esclave est en contact avec l'environnement distant, alors pour les mêmes forces F_h et F_e, on veut obtenir les mêmes vitesses $\dot{\theta}_m$ et $\dot{\theta}_s$, ceci revient à écrire :

$$Z_e\left(\dot{\theta}_s\right) = Z_t\left(\dot{\theta}_m\right) \tag{1.26}$$

Avec :

$$\begin{cases} F_e = Z_e\left(\dot{\theta}_s\right) \\ F_h = Z_t\left(\dot{\theta}_m\right) \end{cases} \tag{1.27}$$

Où : Z_t est l'impédance ressentie par l'opérateur humain.

Or, pour que la relation (1.26) soit vraie (i.e. $F_e = F_h$ et $\dot{\theta}_s = \dot{\theta}_m$), l'impédance Z_t ressentie par l'opérateur doit être la plus proche possible de l'impédance Z_e transmise par l'environnement sur l'esclave.

Donc, pour que la transparence du système de téléopération soit idéale, le rapport Z_t/Z_e doit tendre vers 1 ($(Z_t/Z_e) \rightarrow 1$) [44].

1.8 Compromis stabilité / transparence

Dans un système de téléopération, il est souhaitable que le système soit stable et transparent afin d'obtenir une téléopération sécurisée et de haute performance, néanmoins la stabilité et la transparence restent deux objectifs concurrents, plus précisément lorsqu'un retard de transmission existe. Plusieurs études ont été menées dans le but d'avoir le meilleur rapport stabilité / transparence. Dans [40], l'auteur a appliqué un modèle hybride à deux ports pour un système de téléopération et a montré qu'une bonne transparence peut être obtenue. L'auteur dans [44] a proposé une structure de commande basée sur quatre canaux de transmission et des contrôleurs dynamiques afin de garantir une transparence idéale, cependant la stabilité n'est pas toujours garantie. Dans [38], un schéma de commande avec des modèles dynamiques précis des robots maître et esclave a été introduit afin d'assurer une transparence parfaite mais avec une robustesse plus ou moins bonne.

1.9 Conclusion

Dans ce chapitre, nous avons donné un aperçu général sur les systèmes de téléopération ainsi que le modèle mathématique qui les définit, de même nous avons donné les critères de performances qui permettent d'évaluer leur transparence.
Les trois architectures de contrôle des systèmes de téléopération, les plus utilisées en pratique ont été présentées et comparées par rapport aux critères de performances. Les architectures position-position et force-position sont les plus faciles à implémenter en pratique en termes de matériels (capteurs) et de contrôle, cependant elles ne permettent pas d'atteindre de très bonnes performances, contrairement à l'architecture de contrôle à quatre canaux, qui nécessite l'utilisation de plus de capteurs, ce qui rend la mise en œuvre du contrôle plus compliquée, néanmoins cette méthode permet d'atteindre une transparence idéale en théorie et d'avoir d'excellentes performances en pratique.
L'analyse des systèmes de téléopération se fait par l'évaluation de la stabilité et de la transparence. Nous avons présenté la méthode de l'analyse de la passivité en ce qui concerne

la stabilité, ainsi nous avons donné les critères qui définissent la transparence idéale.
Notre travail de thèse consiste en l'implémentation d'un régulateur adaptatif neuro-flou sur un circuit programmable qui est le FPGA, pour commander un système de téléopération à un degré de liberté. L'utilisation d'un tel circuit programmable va rendre la mise en œuvre du contrôle très compliquée, néanmoins, elle nous permettra d'obtenir un système complètement transparent et d'atteindre de très bonnes performances grâce à son parallélisme de calcul et sa fréquence de fonctionnement élevée. Cependant, son utilisation nécessite une bonne connaissance de ses caractéristiques et une très bonne maîtrise des outils d'implémentation, ce qui sera détaillé dans le chapitre suivant.

Chapitre 2

Description de la carte FPGA

2.1 Introduction

La technologie FPGA permet à l'utilisateur de construire des systèmes configurables et reconfigurables afin d'atteindre les exigences d'une conception. Cette technologie est utilisée dans plusieurs domaines tels que le contrôle des systèmes et le traitement de signal, et cela grâce à ses multiples avantages, particulièrement la fréquence de fonctionnement élevée et le parallélisme de calcul. les FPGAs permettent une grande capacité de traitement, où chaque tâche est traitée de façon indépendante par une section dédiée sur la puce FPGA sans influencer sur les autres blocs logiques de la puce. Les circuits FPGA sont programmés et configurés à l'aide des outils basés sur le langage de description matérielle (HDL) tels que VHDL et Verilog, ou à l'aide des outils de développement de haut niveau tel que le logiciel MATLAB.

Dans ce chapitre, nous allons présenter de façon générale les architectures des FPGAs ainsi que leurs technologies de programmation, par la suite, nous allons nous focaliser sur les FPGAs du constructeur Xilinx, plus précisément ceux de la famille Zynq-7000 utilisés dans ce travail de thèse, ainsi que les outils de conception et de développement proposés par le constructeur pour la mise en œuvre des conceptions.

2.2 Définition d'un FPGA

Un FPGA (*Field Programmable Gate Array*) est un circuit programmable basé sur des semi-conducteurs, composé de blocs logiques configurables, interconnectés entre eux par des interconnexions programmables [49]. Ces blocs permettent de réaliser des fonctions combinatoires de base, à savoir les portes logiques élémentaires, ainsi que des fonctions logiques complexes. Un FPGA est constitué également de blocs mémoires permettant de configurer le circuit [50].
Les FPGAs ont la particularité d'être reconfigurables même au cours du fonctionnement (reconfiguration dynamique), les blocs logiques ainsi que les interconnexions peuvent être donc reconfigurés au cours du fonctionnement dans le but d'atteindre les exigences d'une application.

2.3 Technologies de programmation

Les technologies de programmation des FPGAs différent d'un constructeur à un autre. Chacune des technologies possède des caractéristiques différentes qui définissent l'architecture de la logique programmable dans le circuit [51], permettant ainsi d'assurer une grande flexibilité de conception [52]. Les technologies de programmation des FPGAs les plus courantes sont [21, 51] :

2.3.1 Technologie de programmation SRAM

Dans ce cas, les connexions entre les blocs logiques configurables (CLBs) et les blocs d'entrées/sorties (IOBs) sont réalisées en rendant les transistors passants [52]. Cette technologie de programmation est la plus utilisée dans les circuits FPGA puisqu'elle présente l'avantage d'avoir une vitesse de reprogrammabilité rapide avec une faible consommation d'énergie [51], cependant son principal inconvénient réside dans la surface utilisée car les SRAMs nécessitent beaucoup de transistors [53], ce qui rend cette technologie couteuse, de plus les SRAMs ont l'inconvénient d'être volatiles, de ce fait, un périphérique externe est toujours nécessaire pour stocker les données de configuration [51].

2.3.2 Technologie de programmation flash

Cette technologie présente un avantage puisqu'elle n'est pas volatile [51,53], un FPGA programmé avec cette technologie peut être utilisé même après son redémarrage , néanmoins, elle présente des inconvénients car les FPGAs basés sur cette technologie sont limités en terme du nombre de reconfigurations [51], de plus, cette dernière prend un temps plus important par rapport aux technologies SRAM [52].

2.3.3 Technologie de programmation anti-fusible

Dans cette technologie, les connexions sont programmables une seule fois par la destruction des fusibles. Le principal avantage de cette technologie de programmation réside dans sa surface d'occupation qui est basse par rapport à la technologie SRAM [51,53], de plus cette technologie est non volatile néanmoins, elle présente un grand inconvénient, car les appareils basés sur la technologie anti-fusible ne peuvent pas être reconfigurés [51].

2.3.4 Technologie de programmation EPROM

Cette technologie utilise des transistors de type EPROM (*Erasable Programmable Read-Only Memory*). Son principal inconvénient réside dans la reconfiguration qui nécessite une source ultra-violet [52, 54].

2.3.5 Technologie de programmation EEPROM

Dans ce cas, les transistors EEPROM (*Electrically Erasable Programmable Read-Only Memory*) sont utilisés. Cette technologie de programmation présente un avantage par rapport à la technologie EPROM, puisque sa configuration est réalisée électriquement [52, 54].

2.4 Architecture générale d'un circuit FPGA

L'architecture d'un FPGA consiste essentiellement en trois éléments à savoir les blocs logiques configurables, les blocs d'entrées/sorties et les ressources d'interconnexion [49]. Cependant, l'architecture interne du circuit diffère d'un constructeur à un autre, définissant ainsi la façon avec laquelle la mise en œuvre matérielle est réalisée via sa configuration. L'architecture de base d'un FPGA du constructeur Xilinx, que nous avons utilisé dans la suite de ce travail de thèse, est caractérisée par deux couches différentes [55] :
La première couche est constituée d'une matrice de blocs logiques configurables (CLBs) et des blocs d'entrées/sorties (IOBs). Les CLBs permettent de réaliser les différentes fonctions combinatoires et séquentielles tandis que les IOBs réalisent l'interface entre l'architecture interne et l'environnement extérieur [21].
La deuxième couche contient des mémoires SRAM, qui permettent de mémoriser la programmation du FPGA. Le processus de programmation consiste à réaliser les connexions entre les CLBs et les IOBs en rendant les transistors passants afin de réaliser les fonctions souhaitées. Ces blocs SRAM sont volatils [21, 51], c'est-à-dire que la configuration est perdue à chaque nouveau démarrage du dispositif. Une mémoire ROM est donc utilisée pour mémoriser la configuration qui sera chargée dans les SRAMs à chaque redémarrage du FPGA [52].

2.5 Architecture interne

La structure interne des FPGAs diffère d'un constructeur à un autre. La structure de base est constituée de trois éléments essentiels : les blocs logiques configurables (CLBs), les blocs d'entrées/sorties (IOBs) et un réseau d'interconnexions programmables (Figure 2.1). Afin de faciliter la mise en œuvre des conceptions, les dispositifs FPGA récents sont équipés de blocs supplémentaires tels que les DSPs, les mémoires, les gestionnaires d'horloge numérique (DCM) et des processeurs [56], ainsi que des blocs de communication qui prennent en charge différents protocoles de communication tels que les protocoles USB, Ethernet, CAN, PCI, SPI et I2C [21].

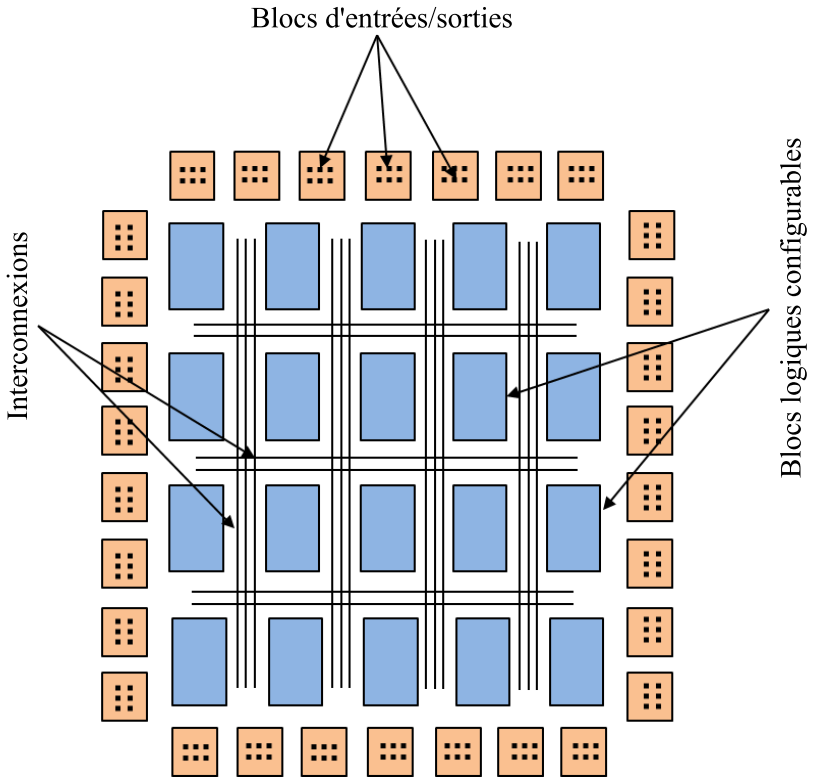


FIGURE 2.1: Architecture interne d'un FPGA.

L'architecture interne d'un FPGA du constructeur Xilinx se présente sous forme de deux blocs :

2.5.1 Blocs élémentaires

Blocs logiques configurables (CLBs) : Les blocs logiques configurables représentent l'unité de base d'un FPGA, qui permet la mise en œuvre des circuits combinatoires et séquentiels. Un CLB est composé de générateurs de fonctions avec des tables de correspondances (LUT) et des bascules D pour la mémorisation et la synchronisation. Chaque CLB est connecté à une matrice de commutation pour accéder à la matrice de routage générale [57]. Il existe plusieurs types de CLBs en fonction du type du FPGA utilisé selon le constructeur [55]. Pour Xilinx, un CLB est constitué d'éléments appelés SLICEs [58]. Chaque SLICE est composé d'un nombre donné de LUTs selon la famille du FPGA, combiné avec des bascules D et des multiplexeurs (Figure 2.2).

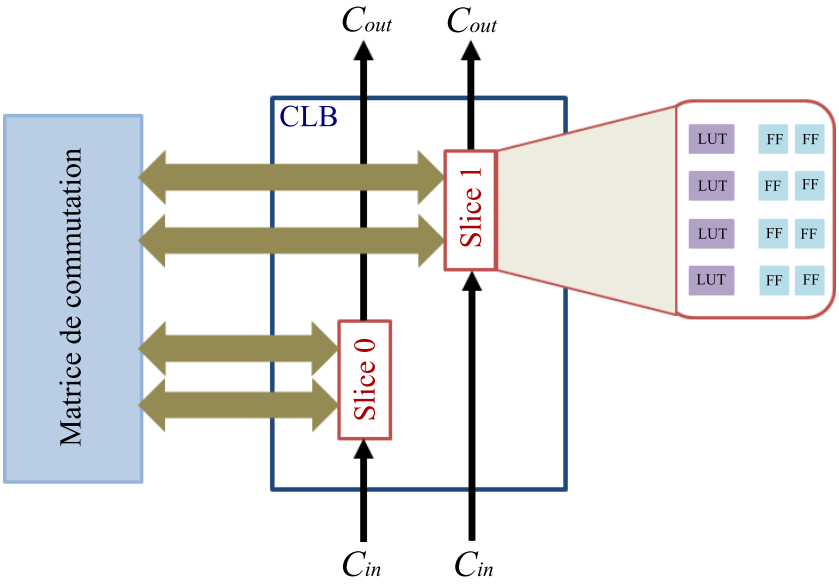


FIGURE 2.2: Architecture d'un bloc logique configurable.

Blocs d'entrées/sorties (IOBs) : Ces blocs reconfigurables permettent l'interfaçage entre la logique interne développée par l'utilisateur et les modules extérieurs [21]. Les blocs d'entrées/sorties ont leur propre mémoire de configuration, et ils peuvent être configurés en entrées, en sorties ou en signaux bidirectionnels [55].

Interconnexions : Le rôle des interconnexions consiste à relier les blocs logiques avec les blocs d'entrées/sorties afin de mettre en œuvre tout circuit défini par l'utilisateur [51]. Les ressources d'interconnexion présentent un réseau programmable qui occupe environs 90 % de la surface d'un composant FPGA, la flexibilité de ce dernier dépend alors principalement du réseau programmable [59]. Le réseau d'interconnexion est constitué de pistes de routage horizontales et verticales interconnectées via des blocs de commutation [59], qui sont configurées à l'aide de la technologie programmable adoptée [51].

2.5.2 Blocs supplémentaires

Blocs mémoires : Ces blocs comprennent des mémoires ROM, RAM et Flash RAM. Ils permettent de stocker des ensembles de données [55] et ils sont intégrés sur le FPGA afin d'optimiser les ressources matérielles consommées pendant le processus [21].

DSPs : Ce sont des blocs arithmétiques comprenant des multiplicateurs, des addition-

neurs et des accumulateurs [21]. Ils sont conçus pour effectuer des calculs en temps réel avec des opérations simples ou des opérations en traitement de signal, sur des nombres en virgule fixe ou en virgule flottante.

Gestionnaires d'horloge numérique (DCMs) : Les DCMs sont généralement basés sur des PLLs (Phase-Locked Loops). Leur fonction consiste à générer d'autres fréquences d'horloge en effectuant la multiplication ou la division d'une fréquence d'horloge interne ou externe, tout en compensant les temps de propagation avec correction des déphasages [21].

Processeurs : Parmi les ajouts les plus importants des constructeurs sur les FPGAs, on trouve les processeurs embarqués. L'utilisation d'une carte FPGA dotée d'un processeur permet de simplifier le processus de conception et d'intégration des algorithmes complexes tout en réduisant les ressources matérielles consommées [55]. Plusieurs constructeurs proposent des cartes de développement FPGA avec des cœurs de processeurs, comme on le trouve chez la société Xilinx qui propose des plateformes de développement de type Zynq-7000 (utilisées dans ce travail de thèse) avec des cœurs de processeurs de type ARM.

2.6 FPGAs de la société Xilinx

Xilinx est une entreprise américaine fondée en 1984, spécialisée dans la fabrication des circuits intégrés et des dispositifs à semi-conducteurs [60]. Elle est la mère des FPGAs dont la création et la commercialisation pour la première fois étaient en 1985.

Depuis sa fondation, la société a élargi sa gamme de produits. Xilinx vend une large gamme des FPGAs, de dispositifs de logique programmable (CPLDs) ainsi que des outils de conception. Elle propose des FPGAs très puissants en utilisant des technologies très avancées telles que les puces 3D [60]. Ces technologies ont permis à la société de combiner des composants séparés auparavant dans une seule puce en combinant tout d'abord un FPGA avec des émetteurs-récepteurs afin d'augmenter la capacité de la bande passante. Par la suite, Xilinx introduit une nouvelle gamme de dispositifs qui est la SoC Zynq-7000 de 28 nm, qui combine un cœur ARM avec un FPGA [61].

Après avoir introduit Xilinx 7 en 2010, la société a migré vers trois grandes familles des FPGAs : la Virtex haut de gamme, la Kintex au milieu de gamme et la Artix à faible coût. En 2012, la société a lancé le logiciel Vivado Design Suite qui est un environnement de haut niveau pour la conception des systèmes numériques. Ce logiciel permet de développer des conceptions très puissantes et très robustes dans diverses applications tout en tirant profits des avantages des FPGAs des nouvelles gammes.

2.7 Présentation de la carte de développement Zedboard de type Zynq-7020

ZedBoard est une carte d'évaluation et de développement basée sur la plateforme de traitement extensible Xilinx Zynq-7000, conçue pour créer des conceptions basées sur Linux, Androïd ou Windows. La combinaison robuste sur la Zedboard d'un système de traitement (PS) basé sur un double processeur ARM de type Cortex-A9 avec une logique programmable (PL) de 85 000 cellules logiques de la technologie série 7 de Xilinx, permet de créer des conceptions très puissantes dans diverses applications (contrôle de procédés, traitement de signal,. . . .) [61].

La carte de développement Zedboard comprend (Figure 2.3) [61–63] :

• Une puce FPGA de type Zynq-7000 AP SoC XC7Z020-CLG484-1 d'une horloge de 100 MHz.

• Deux processeurs ARM de type Cortex-A9 d'une horloge de 33.333 MHz.

• Alimentation : Un switch d'alimentation et un régulateur de tension AC-DC 12V-5A.

• Mémoires : Une mémoire SDRAM DDR3 512 MB et une mémoire QSPI flash 256 Mb.

• Interfaces d'entrées/sorties : Port USB-JTAG pour programmation, port Ethernet 10/100/1G, port USB OTG 2.0, carte mémoire 4GB avec port d'emplacement, port USB 2.0 FS avec un pont USB-UART, 5 ports d'entrées/sorties dont 4 pour le circuit logique et 1 pour les processeurs, deux boutons de réinitialisation (1 pour PS et 1 pour PL), 7 boutons poussoirs (2 pour PS et 5 pour PL), 8 switchs (PL), 9 leds pour utilisateur (7 pour PL et 1 pour PS), une led de signalisation (PL), un connecteur LPC FMC et un connecteur

AMS (signal analogique mixte).

• Interface d'affichage/audio : Sortie HDMI, connecteur VGA, écran OLED, entrées/sorties audio, connecteurs pour casque et microphone.

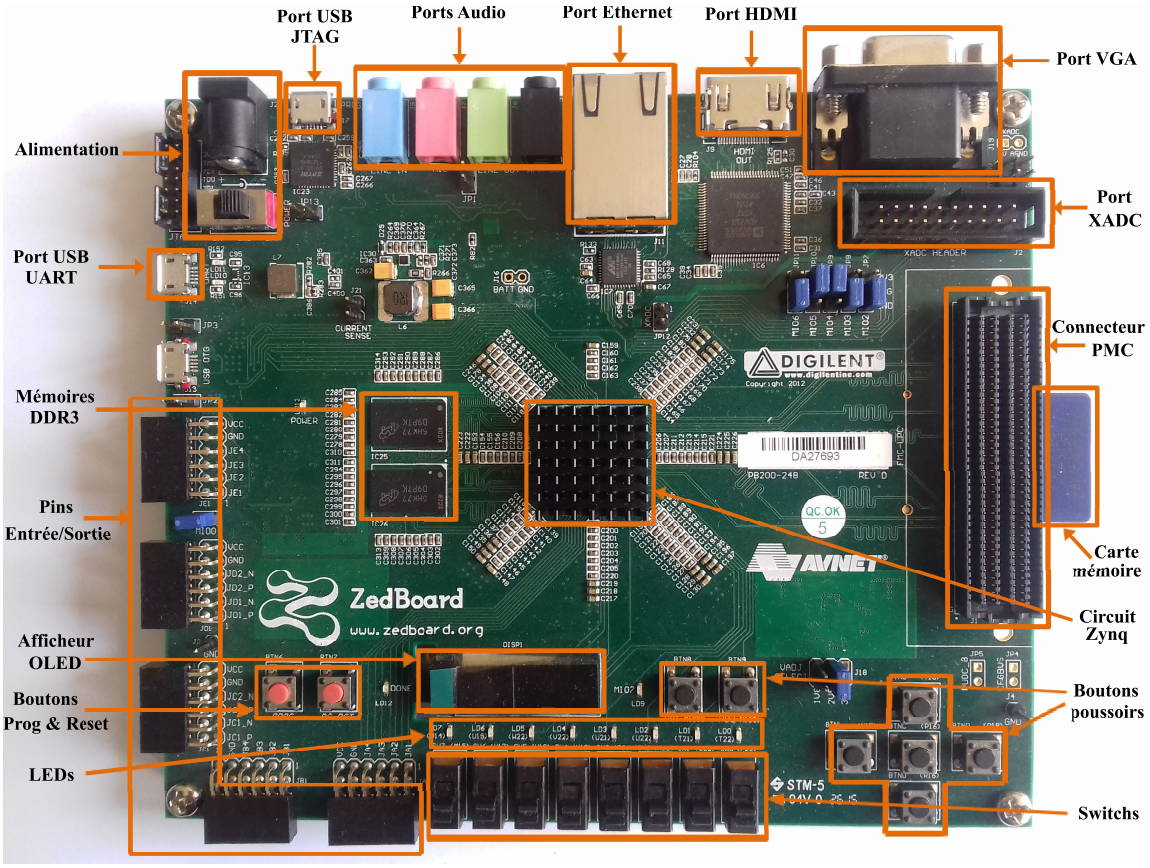


FIGURE 2.3: Composition de la carte Zedboard.

La carte Zedboard dispose d'un nombre important de blocs logiques à usage général, de blocs RAMs et DSP48E1, qui lui permettent une flexibilité et une grande vitesse de calcul. Le Tableau 2.1 décrit les caractéristiques de l'architecture interne de la Zedboard [64,65].

Les éléments de l'architecture interne de la puce Zynq sont décrits comme suit (Figure 2.4) [64] :

Ressources logiques : Ce sont les éléments principaux de la Zynq, qui sont composés de :

• *Blocs logiques configurables (CLBs) :* Les CLBs sont des éléments logiques regroupés sur

Processeur	Dual core ARM Cortex-A9
Nombre de Flips-Flops (FF)	106400
Nombre de LUTs 6 entrées (Look Up Table)	53200
Nombre de mémoires LUT	17400
Nombre de blocs RAM 36 Kb	140
Nombre de DSP48 (Digital Signal Processing)	220
Nombre de blocs entrée/sortie (I/O)	200
Nombre de buffers (BUFG)	32
Nombre de gestionnaires d'horloge numérique (MMCM)	4

TABLE 2.1: Éléments de la carte Zedboard.

la logique combinatoire (PL) et connectés à d'autres ressources similaires via des interconnexions programmables. Chaque CLB est placé à coté d'une matrice de commutation et contient deux Slices logiques.

• *Slice :* C'est l'unité interne principale du CLB. Elle contient des ressources pour la mise en œuvre des circuits logiques combinatoires et séquentiels. Pour le dispositif Zynq, un Slice est composé de 4 LUTs, 8 Flips-flops et d'autres blocs logiques.

• *LUT :* C'est une ressource logique flexible capable de mettre en œuvre : une fonction logique de six entrées maximum, une ROM, une RAM et un registre à décalage.

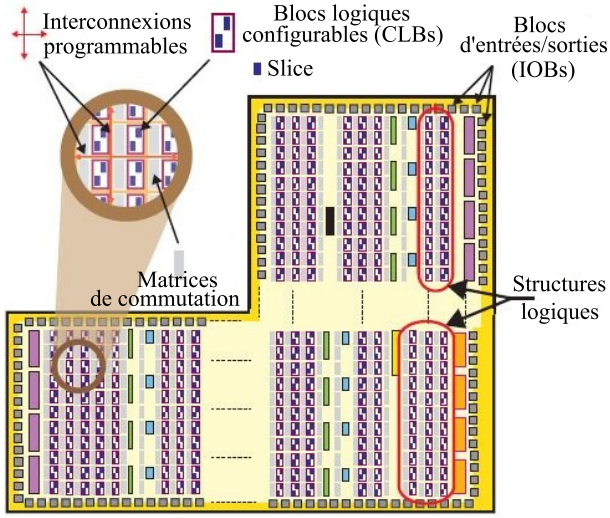


FIGURE 2.4: Architecture interne d'un FPGA Zynq-7000.

• *Blocs Flips-flops (FFs) :* Un Flip-flop est un circuit séquentiel qui implémente un registre à 1 bit avec une réinitialisation.

• *Matrices de commutation :* Une matrice de commutation se trouve à coté d'un CLB et fournit une fonction de routage flexible pour établir des connexions entre les éléments internes du CLB, et d'un CLB et une autre ressource dans la logique programmable.

• *Blocs entrées/sorties (IOBs) :* Généralement situés autour du périmètre de l'appareil, les blocs entrées/sorties sont des ressources qui assurent l'interfaçage entre la logique programmable et les périphériques logiques qui sont utilisés pour la connexion des circuits externes.

Ressources spéciales : La Zynq dispose de deux composants à usage spécifique : Les blocs RAM pour les besoins mémoire, et les blocs DSP48E1 pour l'arithmétique à grande vitesse (Figure 2.5).

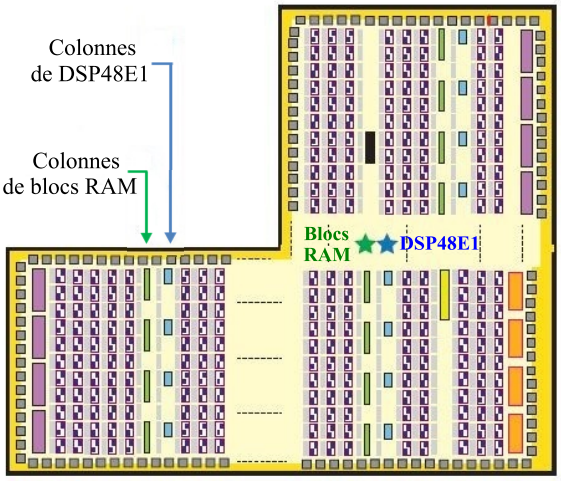


FIGURE 2.5: Blocs RAM et DSP48E1.

• *Blocs RAM :* Les blocs RAM peuvent implémenter une mémoire vive (RAM), une mémoire morte (ROM) et les tampons FIFO. La taille des mots par défaut est de 18 bits, chaque RAM comprend 2048 éléments mémoires, et peut stocker jusqu'à 36 Ko d'informations, et peut être aussi configurée comme une RAM de 36 Ko ou deux RAMs indépendantes de 18 Ko.

• *Blocs DSP48E1 :* Les blocs DSP48E1 sont des Slices spéciaux pour la mise en œuvre de l'arithmétique à grande vitesse pour des signaux avec des longueurs de mots moyennes ou longues. Ce sont des ressources en Silicium qui comprennent principalement un pré-additionneur/soustracteur, un multiplicateur et un post-additionneur/soustracteur avec

une unité logique (Figure 2.6).

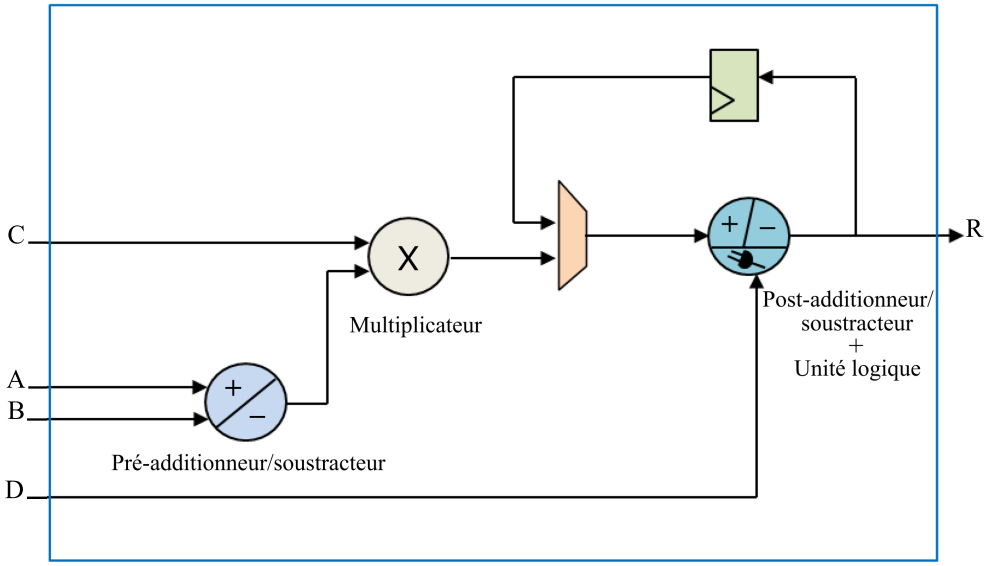


FIGURE 2.6: Structure du Bloc DSP48E1.

Interfaces de communication : Le dispositif Zynq comprend des blocs transmetteurs GTX, et des blocs d'interface de communication à grande vitesse, intégrés sur la structure logique. Ce sont des blocs en Silicium capables de prendre en charge un certain nombre d'interfaces standard à savoir le PCI express, communication série rapide, le SCSI et le SATA.

Autres interfaces externes logiques programmables

• *Blocs de conversion analogique-numérique :* La logique programmable inclut un bloc XADC pour conversion de signaux analogiques-numériques mixtes, qui comprend deux convertisseurs analogique-numérique distincts de 12 bits.

• *Horloges :* La logique programmable de la Zynq dispose de quatre entrées d'horloges différentes provenant du PS, et dispose de fonctionnalités qui lui permettent de générer et de distribuer ses propres signaux d'horloge.

• *Programmation/débogue :* Le périphérique Zynq dispose de ports JTAG pour faciliter la configuration et le débogage de la logique programmable. Les fonctionnalités offertes via JTAG prennent en charge le débogage avec les outils ARM et Xilinx.

2.8 Type de données

Dans les circuits numériques, les nombres sont stockés dans des mots binaires. Un mot binaire est une séquence de bits (1 et 0) de longueur fixe. La façon dont les composants matériels interprètent cette séquence de bits est définie par le type de données [22].
Les nombres binaires sont présentés sous forme de deux types de données : virgule fixe et virgule flottante.

2.8.1 Virgule fixe

La virgule fixe est le système de représentation des nombres réels le plus simple. Le type de données à virgule fixe est caractérisé par : la longueur du mot en bits, le signe (signé ou non signé) et la position du point binaire. Cette dernière définit le moyen avec lequel les valeurs sont interprétées [22].
Un nombre réel R représenté en virgule fixe est constitué de deux parties [66], une partie entière I_R sur n_I bits, une partie fractionnaire F_R sur n_F bits et un bit de signe S_R, où 0 désigne un nombre positif tandis que 1 désigne un nombre négatif (Figure 2.7).

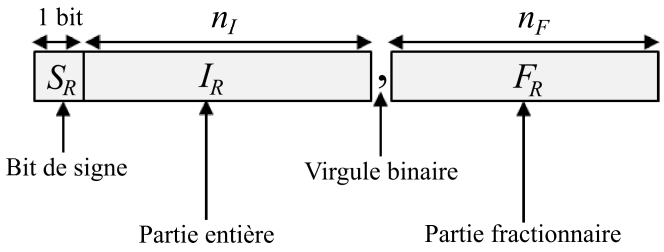


FIGURE 2.7: Représentation d'un nombre réel en virgule fixe.

Le Nombre réel R peut s'écrire [67] :

$$R = \overline{I_R F_R} * 2^{n_F} \tag{2.1}$$

Exemple : Représentation de $(6.375)_{10}$ en virgule fixe, avec n_I=3 bits et n_F=3 bits.

$$R = 110.011 = \overline{110011} * 2^{-3} = 51 * 2^{-3} = (6.375)_{10} \tag{2.2}$$

2.8.2 Virgule flottante

Dans ce type de données, le terme flottant indique qu'il n'y a pas de nombre fixe de chiffres avant et après le point décimal. Ce dernier peut alors flotter [68].
Un nombre réel R représenté en virgule flottante est défini par la relation suivante [67] :

$$R = (-1)^{S_R} .M.2^{E} \tag{2.3}$$

S_R représente le signe de R où 0 désigne un nombre positif tandis que 1 désigne un nombre négatif, M est la mantisse et E est l'exposant.
La norme IEEE 754 est la plus utilisée actuellement dans les systèmes numériques. Elle représente une spécification unique de la représentation binaire en virgule flottante. L'objectif de cette norme est d'améliorer la compatibilité entre les diverses architectures [67].

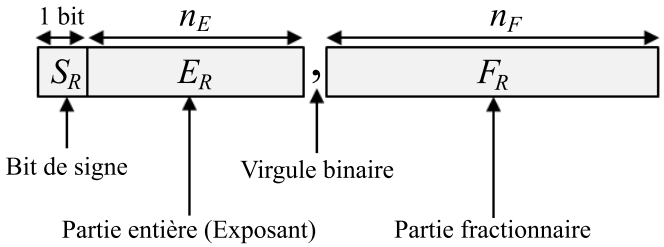


FIGURE 2.8: Représentation d'un nombre réel en virgule flottante.

Un nombre réel F représenté en virgule flottante par la norme IEEE 754 (Figure 2.8) est défini par la relation suivante :

$$R = (-1)^{S_R} .M.2^{E_R - E_0} \tag{2.4}$$

Avec $M = \overline{1, F_R}$ et $E_0 = 2^{n_E - 1}$, où E_0 représente le biais.
La norme IEEE 754 définit deux formats [67, 69] :
Simple précision : Représenté sur 32 bits (1 bit pour le signe, 8 bits pour la partie entière ($n_E = 8$) et 23 bits pour la partie fractionnaire ($n_F = 23$)).
Double précision : Représenté sur 64 bits (1 bit pour le signe, 11 bits pour la partie

entière ($n_E = 11$) et 52 bits pour la partie fractionnaire ($n_F = 52$)).

Tous les FPGAs prennent en charge le type de données à virgule flottante double précision et le type de données à virgule fixe, néanmoins, les conceptions à virgule flottante nécessitent des ressources matérielles plus importantes que leurs équivalentes en virgule fixe. Cette utilisation importante de ressources entraine une consommation d'énergie plus élevée augmentant ainsi le coût global de la mise en œuvre de la conception [22], contrairement aux implémentations à virgule fixe, elles sont toujours plus efficaces que leurs homologues en virgule flottante car elles consomment moins de ressources matérielles et moins de puissance [23] avec un temps de calcul plus court.
Dans ce travail de thèse, nous avons opté pour le type de données à virgule fixe pour la conception de nos algorithmes de contrôle, en utilisant l'outil " Fixed-Point Tool " de l'environnement Simulink de MATLAB (Voir chapitre 5).

2.9 Outils de conception et de développement FPGA de Xilinx

Le VHDL (VHSIC) est un langage de description matérielle apparu dans les années 80, qui permet de concevoir des circuits intégrés [70]. Au début, ce langage n'était pas destiné à la synthèse des circuits numériques jusqu'à ce qu'il a été développé avec des outils de synthèse associés, et à partir de là, il est devenu le principal mécanisme de développement des conceptions FPGA [71].
Les circuits FPGA ont été toujours considérés comme des dispositifs difficiles à programmer puisqu'ils présentent une technologie matérielle [71]. Pour cette raison, les outils de synthèse ont enregistré un important développement afin de rendre cette technologie plus facile à programmer. Pour cela, les constructeurs FPGA ont lancé plusieurs outils. Xilinx propose le logiciel ISE pour le développement des conceptions basées sur les langages de programmation VHDL et Verilog, et propose également des outils de synthèse de haut niveau tels que le logiciel Vivado. Les FPGAs du constructeurs Xilinx sont pris en charge

aussi par des outils de haut niveau développés par la société MathWorks à savoir les outils System Generator et HDL Coder , permettant ainsi de réduire le temps des conceptions FPGA et de rendre les implémentations plus faciles.

2.9.1 Xilinx ISE

Xilinx ISE est un environnement de conception conçu par la société Xilinx. Il consiste en un ensemble d'outils logiciels qui permettent de créer et de simuler des conceptions numériques dans les différentes puces fournies par le constructeur Xilinx. Ce logiciel offre à l'utilisateur un outil de développement intégré et une interface graphique pour la conception et la synthèse de circuits logiques numériques [72], qui peuvent être implémentés sur une carte FPGA.
Xilinx ISE offre un environnement de conception extrêmement efficace avec tous les outils nécessaires pour la conception, la simulation et l'implémentation, il intègre un éditeur de texte qui sert à saisir le programme (VHDL/Verilog) de l'utilisateur et d'un simulateur intégré pour vérifier la validité du code avant d'être implémenté sur le matériel.

2.9.2 Vivado Design Suite

Vivado Design Suite est un outil logiciel développé par la société Xilinx, conçu pour augmenter l'intégration et la mise en œuvre des systèmes utilisant des périphériques Xilinx [73]. Ce logiciel permet d'accélérer la mise en œuvre des conceptions grâce à divers outils d'optimisation, de routage et de synthèse [73] tel que l'extension intégrée Vivado HLS, qui est capable de convertir des conceptions C (C, C++, système C) en des fichiers RTL (VHDL/Verilog) pour la mise en œuvre sur des composants Xilinx [64, 74]. Vivado HLS permet de tester à la fois les conceptions d'origine en C et les fichiers RTL générés permettant ainsi de corriger d'éventuelles erreurs (Figure 2.9). Vivado dispose de tous les outils nécessaires pour le développement et l'implémentation des conceptions à savoir des bibliothèques C optimisées qui prennent en charge le type de données à virgule fixe, des fonctions mathématiques, des fonctions d'algèbre linéaire, des fonctions DSP, des fonctions vidéo et une bibliothèque IP [74]. Toutes ces fonctionnalités que dispose Vivado

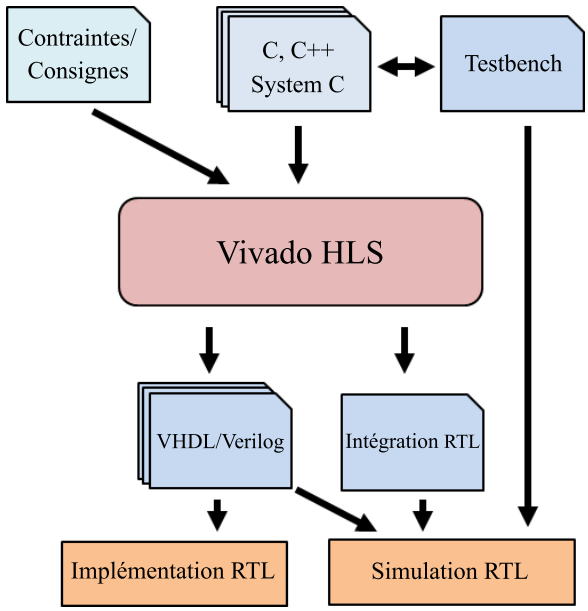


FIGURE 2.9: Conception avec Vivado HLS.

permettent à l'utilisateur de créer plusieurs implémentations ciblées sur une zone réduite avec une grande vitesse, lui permettant ainsi de réduire le temps de développement des conceptions.

2.9.3 System Generator

System Generator est un outil de MathWorks qui utilise une plateforme sur l'environnement Simulink de MATLAB pour la création de conceptions matérielles sur des systèmes FPGA [64]. Il est utilisé généralement pour des conceptions DSP, car il fournit plusieurs blocs allant des opérations mathématiques simples à des opérations DSP complexes ainsi que des blocs de contrôle et de communication qui prennent en charge plusieurs types de données [69]. Il permet de générer des fichiers HDL (VHDL ou Verilog) à partir des modèles conçus par l'utilisateur à base de ces blocs. L'utilisateur peut avoir accès au code HDL généré, néanmoins ce dernier n'est pas toujours lisible, ce qui peut rendre certaines conceptions difficiles à suivre [64]. En outre, System Generator est également intégré au logiciel de conception et de développement Vivado, ce qui permet de créer des blocs IP intégrés directement à partir de ce logiciel.

2.9.4 HDL Coder de l'environnement MATLAB

HDL Coder est un outil développé par la société MathWorks, qui permet la génération de codes HDL (VHDL/Verilog) à partir de modèles Simulink et de fonctions MATLAB (Figure 2.10) [64, 75, 76]. Il fournit un flux de travail qui permet d'analyser des modèles synthétisés en virgule flottante dans l'environnement MATLAB/Simulink, puis les convertir en des modèles en virgule fixe implémentables sur FPGA [64] sans la nécessité d'avoir de connaissances préalables sur le langage de programmation VHDL.

L'outil HDL Coder permet également la vérification du code HDL généré et de le tester auprès du modèle MATLAB/Simulink original, de plus, lors de la génération du code HDL, HDL Coder permet d'effectuer plusieurs types d'optimisation en vue de partager avec efficacité les ressources FPGA afin de réduire la zone occupée par la conception et d'améliorer la fréquence d'horloge [23].

Bien que HDL Coder permet de réduire considérablement le temps des conceptions FPGA à partir des modèles MATLAB/Simulink, il présente des inconvénients car certaines cartes FPGA ne sont pas prises en charge par cet outil, et toutes les fonctions MATLAB et blocs Simulink ne prennent pas en charge la génération HDL [64]. A cet effet, ces blocs peuvent

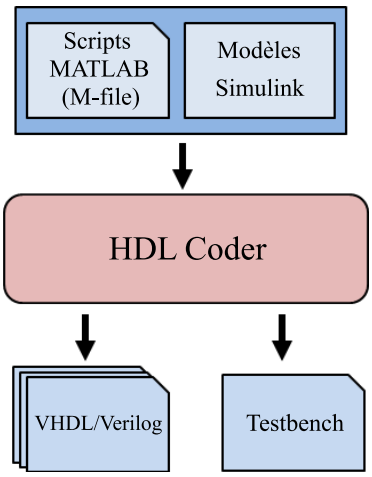


FIGURE 2.10: Conception avec HDL Coder.

être obtenus avec la combinaison des blocs pris en charge et certaines fonctions peuvent être approximées en utilisant les méthodes d'analyse numérique telle que l'interpolation polynomiale.

Dans notre travail, nous avons opté pour l'outil HDL Coder pour la conception et le développement de nos algorithmes de contrôle pour deux raisons différentes. D'une part, Vu la complexité des algorithmes proposés, la spécification optimale des longueurs des mots binaires en virgule fixe reste une tâche complexe qui pourra affecter la précision de l'algorithme pendant les calculs, en conséquence la dégradation de ses performances. Pour cela, l'outil Fixed-Point Tool de Simulink est utilisé car il permet d'automatiser le processus de conversion du modèle synthétisé en virgule flottante vers un modèle en virgule fixe implémentable sur FPGA, tout en optimisant la longueur des mots binaires [23]. D'autre part, la génération du code VHDL et l'implémentation sur la carte FPGA se feront via l'outil HDL Coder, sans avoir recours aux outils de synthèse VHDL puisque l'outil HDL Coder prend en charge la carte Zedboard, et fournit directement une interface pour la mise en œuvre sur la carte FPGA directement via l'environnement Simulink de MATLAB.

2.10 Conclusion

Ce chapitre a été une description générale des circuits FPGA, de leurs technologies de programmation ainsi que de leurs architectures internes. Nous avons présenté les FPGAs du constructeur Xilinx, plus particulièrement ceux de la famille Zynq-7000, utilisés dans ce travail de thèse. Ce type de circuits programmables, issu de la technologie serie 7 de Xilinx, combine de façon robuste un système de traitement basé sur un double processeur de type ARM avec une logique programmable d'un nombre important de cellules logiques en une seule puce, ce qui lui donne une grande flexibilité de conception ainsi qu'une grande vitesse de calcul.

Les constructeurs FPGA proposent différents outils et logiciels pour la mise en œuvre des conceptions. Xilinx propose le logiciel ISE pour des conceptions basées sur le langage VHDL ainsi que des outils de synthèse de haut niveau tel que le logiciel Vivado. De plus, les FPGAs du constructeur Xilinx sont également pris en charge par des outils de haut niveau, développés par la société MathWorks tels que les outils System Generator et HDL Coder, qui permettent de faciliter les mises en œuvre et de réduire de façon considérable

le temps des conceptions.

Dans le chapitre qui suit, nous allons décrire l'architecture de téléopération que nous avons réalisée dans le cadre de ce travail de thèse. Par la suite, cette architecture sera commandée par une carte de développement de type Zedboard de la famille Zynq-7000, en utilisant l'outil HDL Coder du logiciel MATLAB.

Chapitre 3

Réalisation d'un système de téléopération à un degré de liberté

3.1 Introduction

Dans ce chapitre, nous allons présenter le système de téléopération à un degré de liberté, réalisé dans le cadre de cette thèse, ainsi que la démarche suivie en utilisant l'environnement MATLAB, pour l'identification du modèle des actionneurs, pour la synthèse de deux contrôleurs PID. Nous allons d'abord décrire le matériel utilisé (actionneurs, capteurs) pour la réalisation du banc de test, par la suite nous allons expliquer les techniques utilisées pour identifier les modèles de position et de couple des moteurs à courant continu à l'aide de l'outil "System Identification Toolbox", et synthétiser les correcteurs PID en utilisant l'outil et "PID Tuner" de MATLAB. Les performances des contrôleurs PID seront comparées avec celles des contrôleurs neuro-flous proposés dans le chapitre 4.

3.2 Description du banc de test maître-esclave

La Figure 3.1 illustre le banc de test utilisé, qui représente les différents composants du système de téléopération à un degré de liberté, que nous avons réalisé. Il est constitué d'un dispositif maître et d'un autre esclave, chacun est composé d'un moteur à courant

continu de 12V avec un bras rotatif, doté d'un encodeur et d'un capteur de couple. Ces deux dispositifs maître et esclave sont reliés à une carte FPGA (Zedboard Zynq-7000), qui permettra l'acquisition et le traitement des signaux de position et de couple, selon le type de contrôle implémenté, ainsi que de délivrer les signaux de commande adéquats pour actionner les moteurs. La manipulation des fonctionnalités de la carte FPGA est assurée par une interface utilisateur, en utilisant le logiciel MATLAB-Simulink. Cette interface permet à l'utilisateur de choisir le type d'implémentation sur la carte, de définir les différents paramètres du contrôle adopté ainsi que de tracer les différentes courbes représentant les résultats de la commande réalisée.

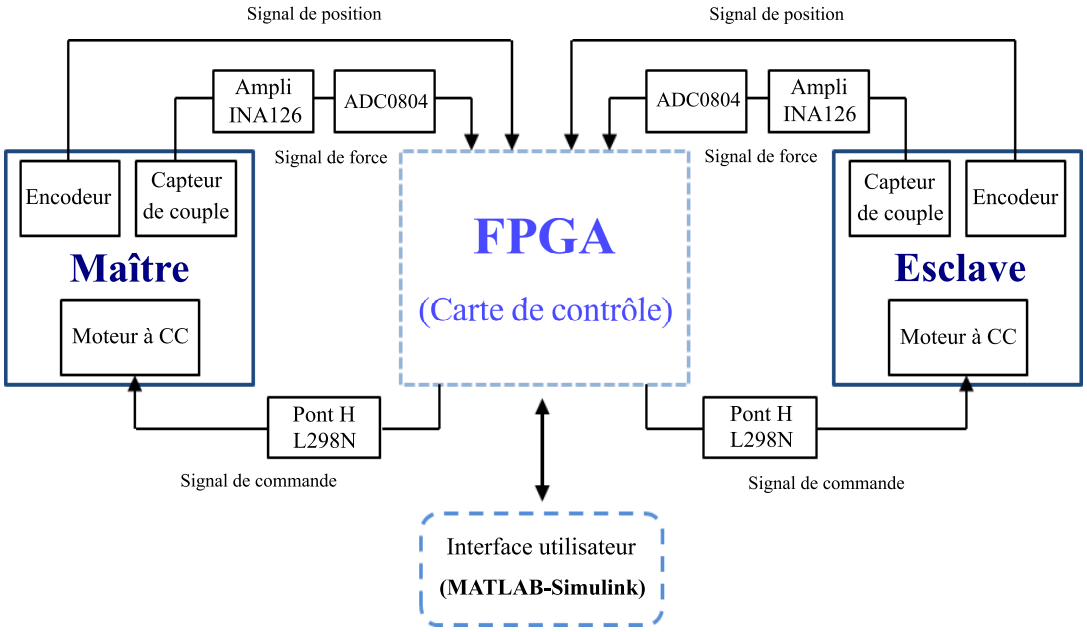


FIGURE 3.1: Banc de test maître-esclave à un degré de liberté.

3.3 Matériel utilisé

Le système de téléopération à un degré de liberté réalisé dans le cadre de ce travail comporte : Deux actionneurs (moteurs à courant continu) chacun avec un bras rotatif, un capteur de position (encodeur) ainsi qu'un capteur de couple. Tous ces composants sont montés de façon adéquate afin de pouvoir réaliser une téléopération bilatérale.

3.3.1 Moteurs

Le système de téléopération utilisé est composé essentiellement de deux appareils identiques (maître et esclave), dont chacun est constitué principalement d'un bras actionné par un moteur à courant continu de 12V (modèle DFRobot 28PA51G)(Figure 3.2). Chaque moteur est équipé d'un encodeur à effet Hall intégré et une carte d'adaptation pour l'encodeur (FIT0324), qui permet de récupérer le signal de position de l'arbre du moteur. L'encodeur est capable de délivrer 675 impulsions par tour, avec une vitesse maximale de 143 tours par minute.

FIGURE 3.2: Moteur à courant continu et carte d'encodeur.

3.3.2 Capteurs

L'implémentation des schémas de commande sur la carte de contrôle (FPGA), afin de faire fonctionner notre système nécessite la présence de deux signaux : La position et la force (couple). Chaque signal est délivré par un capteur adéquat monté sur chacun des dispositifs maître et esclave.

Encodeurs

La position du maître (de l'esclave) est mesurée par un encodeur à effet Hall, intégré sur chaque moteur. L'encodeur dispose d'un disque magnétique multipolaire à faible iner-

tie et deux sondes à effet Hall pour générer deux signaux de sortie A et B déphasés de 90^o. En cas de rotation dans le sens horaire, le signal A précède le signal B. En revanche, la rotation s'effectue dans le sens anti-horaire si le signal B précède le signal A.

Afin de traduire les signaux A et B délivrés par l'encodeur en position angulaire, un compteur est utilisé. Sur la base de ses différentes entrées, ce compteur émet une valeur qui représente le nombre de fronts comptés, il compte alors les événements enregistrés sur ses entrées, et en fonction des états (haut/bas) des signaux A et B, il incrémente ou décrémente le comptage.

Le processus par lequel les valeurs du compteur sont converties en position dépend du type du codage utilisé. Il existe trois types de codages selon le nombre d'incrémentations/décrémentations dans une période : Codages X1, X2 et X4. Dans notre travail, nous avons utilisé le type de codage X4. Dans ce cas, chaque période comporte quatre incrémentations ou décrémentations du compteur. Ce dernier s'incrémente ou se décrémente de manière similaire sur chaque front des canaux A et B, où l'incrémentation ou la décrémentation du compteur dépend du canal qui mène l'autre. La Figure 3.3 illustre le type de codage X4 utilisé dans notre travail.

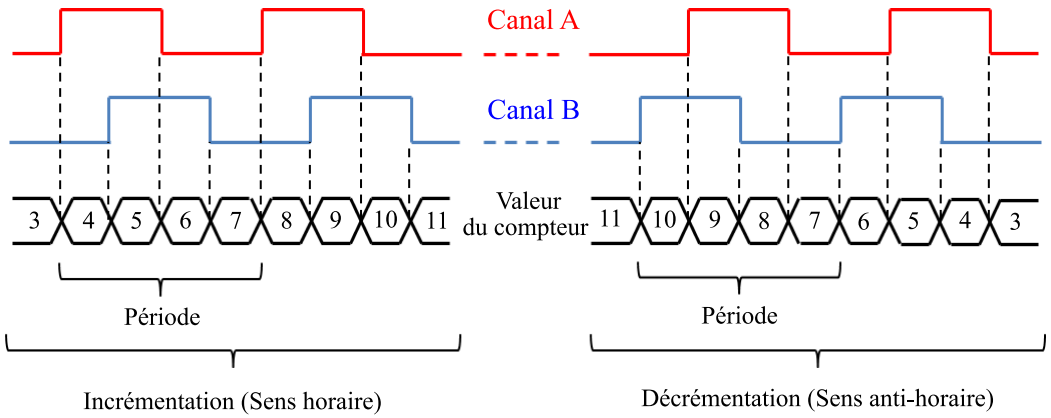


FIGURE 3.3: Codage type X4.

La conversion de la valeur du compteur en position angulaire est donnée par l'équation suivante :

$$Position\,(rad) = \frac{Val_{Compt}}{x.N}\,(2\pi) \tag{3.1}$$

Où :

Val_{Compt} : La valeur délivrée par le compteur.

x : Le type de codage (nombre d'incrémentations/décrémentations par période).

N : Le nombre d'impulsions par tour délivrées par l'encodeur (675 impulsions par tour).

Jauges de contrainte

Afin de mesurer les couples d'interaction avec l'opérateur (au poste maître) ou avec l'environnement (au niveau du poste esclave), une jauge de contrainte a été utilisée.
Une jauge de contrainte est un capteur conçu à base de semi-conducteurs, dont la résistance change de façon marquée lors de la déformation ou la dilatation du matériau sur lequel elle est montée. Cette jauge est généralement placée dans un pont de Wheatstone sous différentes configurations afin de mesurer les changements de résistance du dispositif (Pour plus de détails, voir Annexe A).
Dans notre travail, la mesure de couple est assurée par une jauge de contrainte placée sur chaque bras du moteur (Figure 3.4). Les forces appliquées sur le bras provoquent une flexion qui est mesurée par cette jauge.

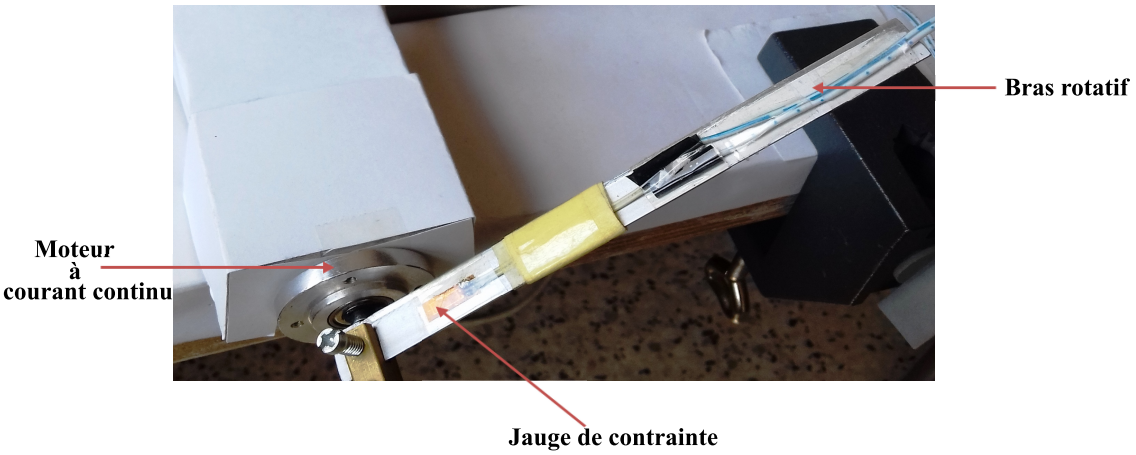


FIGURE 3.4: Jauge de contrainte montée sur le bras du moteur à CC.

Afin de mesurer les changements de résistance de la jauge causés par la force exercée sur le bras, la jauge de contrainte a été montée dans un pont de Wheatstone avec une confi-

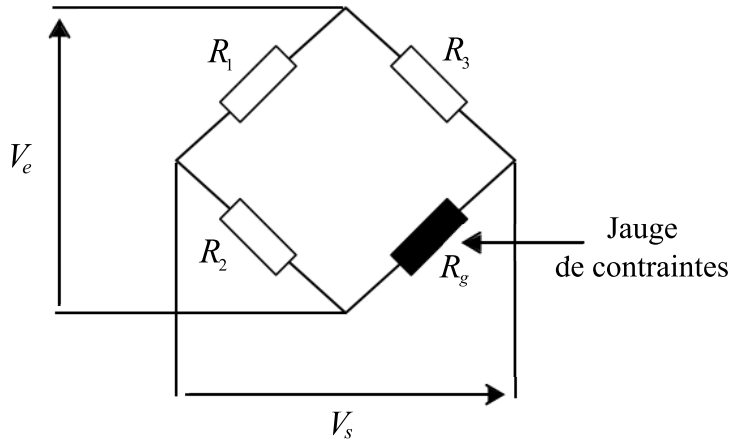


FIGURE 3.5: Montage d'une jauge de contrainte en un quart de pont.

guration en un 1/4 de pont (Figure 3.5). La tension de sortie V_s du pont de Wheatstone est donnée par la relation suivante [77] :

$$V_s = V_e \left(\frac{R_g}{R_g + R_3} - \frac{R_2}{R_2 + R_1} \right) \tag{3.2}$$

Avec :

R_g : La résistance de la jauge.

R_1, R_2, R_3 : Résistances connues.

V_e : La tension d'entrée du pont.

La mesure du couple est basée principalement sur la mesure d'une force F, puisque le couple T peut être considéré comme une force F appliquée sur un bras de levier de longueur l :

$$T = F.l \tag{3.3}$$

L'étalonnage de la jauge a été effectué en fonction des forces appliquées sur le bras du moteur pour calculer le couple correspondant.

3.3.3 Carte de contrôle

La carte de contrôle utilisée est une plateforme de développement FPGA de type Zedboard de la famille Zynq-7000 du constructeur Xilinx. La carte permet l'acquisition des

signaux de position et de force délivrés respectivement par les encodeurs à effet Hall et les convertisseurs analogique/numérique (ADC0804), de plus, elle permet l'implémentation des algorithmes de contrôle (ANFIS et PID) pour calculer la commande pour chaque système (maître et esclave) en fonction de la stratégie de commande adoptée (position-positon et quatre canaux). La Zedboard dispose d'un module d'interface série de type UART, qui assure la communication par liaison série entre la carte FPGA et l'ordinateur, ainsi que deux ports Ethernet et USB-JTAG permettant sa programmation.

FIGURE 3.6: Système de commande en utilisant la carte FPGA Zedboard.

La particularité de cette carte de développement est qu'elle dispose de deux technologies sur une seule puce : Une technologie FPGA avec un nombre très important de cellules logiques, et une technologie microprocesseur avec deux processeurs ARM de type Cortex-A9. Une telle disposition nous permet une flexibilité lors de la conception des algorithmes de contrôle sur le logiciel MATLAB, une facilité d'implémentation ainsi que l'obtention des différentes courbes représentant les résultats directement sur l'environnement Simulink du logiciel MATLAB. La Figure 3.6 représente le système de commande avec la carte FPGA Zedboard en utilisant MATLAB-Simulink.

3.4 Identification du modèle des moteurs

Dans la suite de notre travail, nous allons implémenter sur la carte FPGA Zedboard un contrôleur classique PID et un contrôleur neuro-flou ANFIS, pour commander le système de téléopération à un degré de liberté décrit précédemment, selon deux stratégies de contrôle (position-position et quatre canaux). Néanmoins, le développement des contrô-

leurs PID de position et de couple nécessite l'identification des moteurs à courant continu utilisés, et cela afin d'élaborer un modèle mathématique, à base duquel les paramètres P, I et D seront réglés. De plus, la préparation des contrôleurs (PID et ANFIS) pour l'implémentation sur la carte FPGA nécessite la simulation du système de commande (contrôleurs+modèles) afin de pouvoir convertir les signaux issus de la simulation du modèle synthétisé sur Simulink en virgule flottante, en des signaux à virgule fixe.

L'identification est une étape primordiale dans la définition d'un modèle pour les systèmes physiques. Elle peut être réalisée en deux étapes : La première étape consiste à choisir le type du modèle (linéaire, non linéaire) et son ordre, alors que la deuxième étape est consacrée à l'adaptation des paramètres du modèle choisi, afin d'avoir un comportement équivalent à celui du système physique. Pour ce faire, différentes méthodes sont utilisées pour l'identification de ces paramètres telles que : Les méthodes de Strejc, de Broïda et des moindres carrés [78]. L'environnement MATLAB dispose d'un outil qui permet la réalisation de ces deux étapes, nommé "System Identification Toolbox" (Figure 3.7).

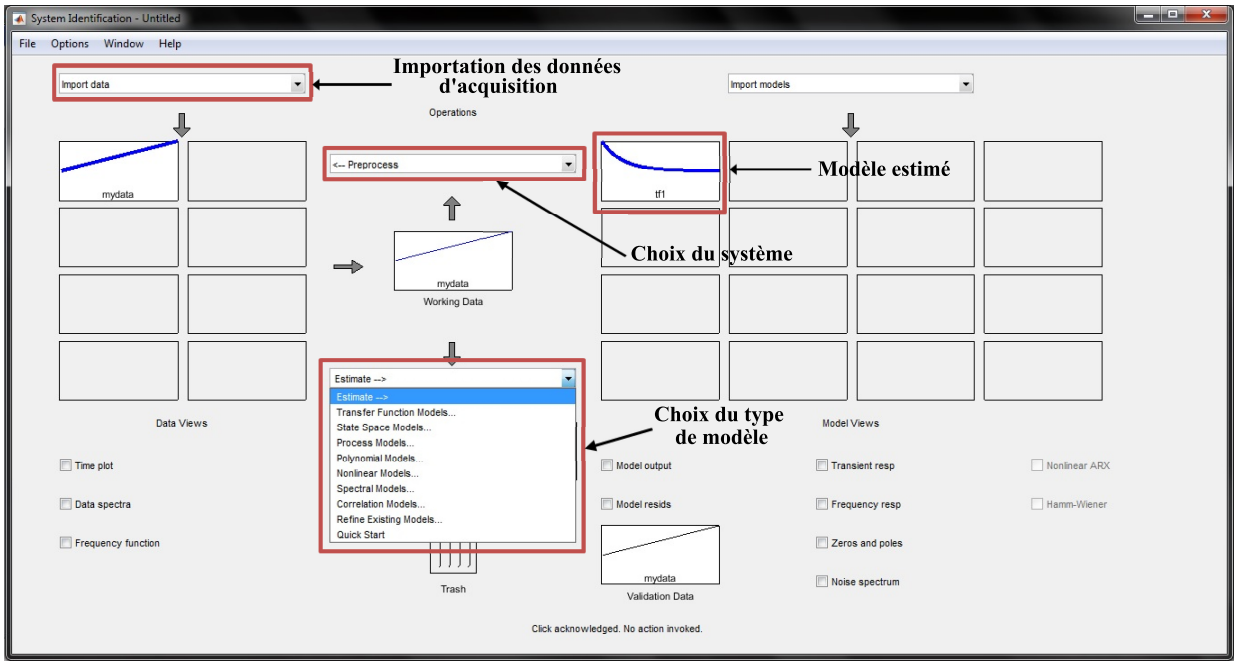


FIGURE 3.7: Interface de System Identification Toolbox.

System Identification Toolbox est un outil intégré sur MATLAB, destiné au développement des modèles mathématiques, pour des systèmes dynamiques à partir des données de mesure entrée-sortie. Cet outil dispose de plusieurs fonctionnalités permettant d'identifier, à partir des ces données mesurées dans les domaines temporel ou fréquentiel, des modèles linaires, non linéaires et des représentations d'état à temps continu ou discret [79]. Dans ce travail de thèse, le développement des modèles de position et de couple pour les moteurs, en utilisant System Identification Toolbox, est réalisé selon les étapes suivantes :

• Importation des données de mesure entrée-sortie sur l'interface de System Identification Toolbox, issues de la carte d'acquisition FPGA Zedboard.

Le processus d'acquisition des données entrée-sortie du moteur consiste à définir un programme sur l'environnement Simulink, qui est ensuite implémenté sur la carte FPGA à l'aide du câble JTAG ou Ethernet. L'excitation du moteur à courant continu est réalisée via l'interface de Simulink, le signal de commande est alors envoyé vers la carte Zedboard via le câble de communication UART, qui sera traité par le programme développé et par la suite injecté au moteur à travers le pilote L298N.

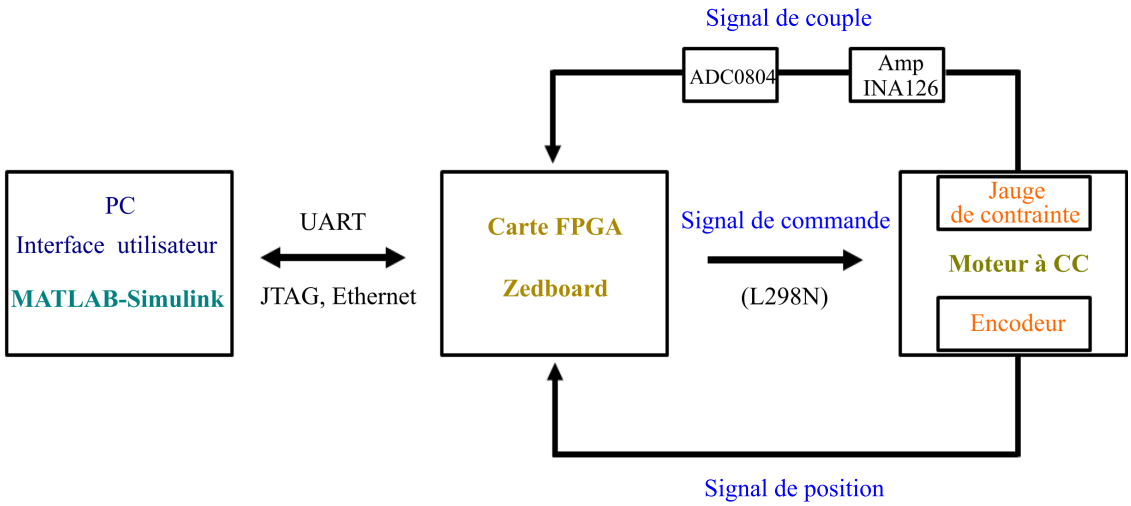


FIGURE 3.8: Acquisition de données par la carte Zedboard en utilisant MATLAB-Simulink.

Les signaux de position et de couple (données de mesure) délivrés respectivement par les capteurs de position (encodeur) et de couple (jauge de contrainte) sont transmis de la carte FPGA au Logiciel MATLAB via le câble UART. Le schéma de La Figure 3.8

illustre le processus d'acquisition des données de mesure par la carte FPGA Zedboard en utilisant le logiciel MATLAB-Simulink.

• Traitement des données d'acquisition (filtrage).

• Choix du type de modèle (fonction de transfert dans notre cas), et estimation de son ordre.

• Estimation des paramètres du modèle choisi.

• Validation du modèle estimé.

Les modèles de position et de couple obtenus sont donnés comme suit : ($T_e = 0.001s$)

Modèle de position :

$$G_1(z) = \frac{8.468.10^{-6}z + 8.466.10^{-6}}{z^2 - 1.999z + 0.9993}$$

Modèle de couple :

$$G_2(z) = \frac{1.002.10^{-6}z + 1.001.10^{-6}}{z^2 - 1.998z + 0.998}$$

3.5 Synthèse des contrôleurs PID

Dans cette section, nous allons synthétiser deux contrôleurs PID de position et/ou de couple en fonction de chaque stratégie de commande adoptée (position-position et quatre canaux). Pour ce faire, nous allons utiliser l'outil "PID Tuner" de l'environnement MATLAB (Figure 3.9).

PID Tuner est un outil intégré sur MATLAB, qui assure le processus de réglage des gains proportionnel (P), intégral (I) et dérivé (D) d'un contrôleur PID, afin d'atteindre les performances souhaitées et répondre aux exigences des conceptions [80].

PID Tuner offre également la possibilité de [80] :

• Identifier le modèle du système à partir des données de mesure entrée-sortie.

• Régler automatiquement les paramètres des contrôleurs : P, PI, PD et PID, sous leurs différentes structures (série ou parallèle), et dans les domaines temporel ou fréquentiel, continu ou discret.

• Régler les performances désirées du système commandé (temps de réponse et dépassement).

• Ajouter des perturbations en entrée et en sortie au système.

• Tracer les différentes courbes représentant le comportement du système (réponse indicielle, qualité du rejet de perturbations et digrammes de Bode et de Nyquist).

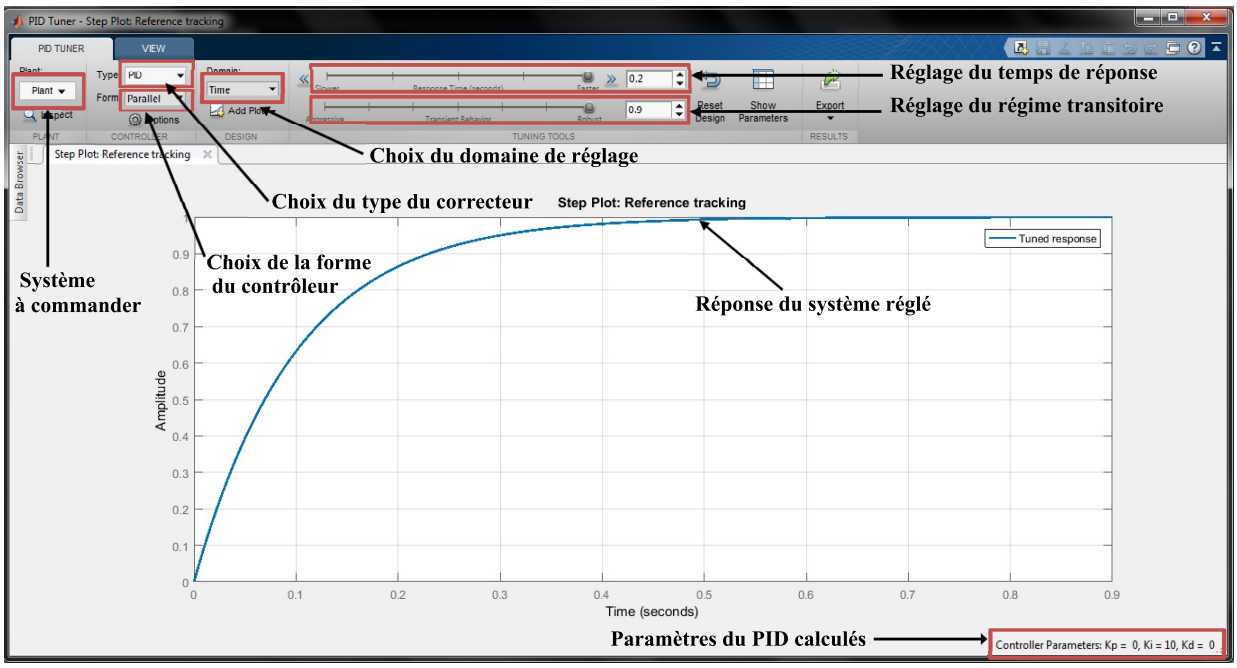


FIGURE 3.9: Interface de PID Tuner.

Dans notre travail, nous avons synthétisé, en utilisant PID Tuner, deux contrôleurs PID discrets de position pour chacun des systèmes maître et esclave en utilisant l'architecture de commande position-position, et un contrôleur de position pour le système esclave et un autre de couple pour le système maître en utilisant l'architecture de contrôle à quatre canaux. Les paramètres des contrôleurs PID obtenus pour chacune des stratégies de commande sont illustrés dans les Tableaux 3.1 et 3.2.

Paramètre	Contrôleurs de position (Maître/Esclave)
P	20
I	2
D	0.1

TABLE 3.1: Paramètres des contrôleurs PID pour l'architecture de contrôle position-position.

Paramètre	Contrôleur de couple (Maître)	Contrôleur de position (Esclave)
P	22	20
I	0.1	2
D	1.5	0.1

TABLE 3.2: Paramètres des contrôleurs PID pour l'architecture de contrôle à quatre canaux.

3.6 Conclusion

Dans ce chapitre, Nous avons présenté le système maître-esclave réalisé dans le cadre de ce travail de thèse, ainsi que les différents composants utilisés pour assurer son fonctionnement avec la carte de contrôle Zedboard. Nous avons expliqué également le fonctionnement des capteurs de position intégrés sur les moteurs (encodeurs), ainsi que celui des capteurs de couple (jauges de contrainte). Par la suite, nous avons abordé les outils de l'environnement MATLAB utilisés pour l'identification des modèles des moteurs en utilisant "System Identification Toolbox", afin de pouvoir synthétiser les contrôleurs PID en utilisant "PID Tuner".

Dans le chapitre qui suit, nous allons proposer un contrôleur adaptatif neuro-flou de type ANFIS, qui sera par la suite implémenté sur la carte Zedboard afin de contrôler le système maître-esclave, présenté dans ce chapitre.

Chapitre 4

Commande neuro-floue du système de téléopération

4.1 Introduction

Les techniques de l'intelligence artificielle telles que la logique floue, les réseaux de neurones et les réseaux neuro-flous sont utilisées pour résoudre plusieurs problèmes dans le domaine de l'industrie et de la robotique. La plupart des systèmes robotiques comme les systèmes de téléopération, présentent des modèles dynamiques non linéaires et complexes, ou parfois inconnus, ce qui rend leur commande très délicate. L'utilisation de l'intelligence artificielle est alors menée pour remédier à ces contraintes de contrôle, notamment la nécessité de disposer d'un modèle mathématique précis, sachant que les erreurs de modélisation affectent négativement les lois de commande, et contribuent à dégrader les performances des contrôleurs.

La logique floue, qui est une approche sans modèle, peut être utilisée pour contrôler un système non linéaire à plusieurs entrées-sorties, en utilisant des règles floues basées sur un raisonnement humain, défini par différentes fonctions d'appartenance [14]. Cette stratégie de contrôle peut offrir de bonnes performances, mais, il est difficile de déterminer les fonctions d'appartenance pour chaque entrée-sortie [15]. L'utilisation des réseaux de neurones artificiels est alors une solution adéquate. En effet, ces réseaux permettent d'ajuster

automatiquement les paramètres des entrées et des sorties du système flou [12], et ceci, en utilisant leur grande puissance de calcul grâce à leur capacité d'apprentissage, à l'aide d'un algorithme d'optimisation non linéaire tel que la rétro-propagation [15, 81]. La combinaison de la logique floue et des réseaux de neurones artificiels permet de tirer profit des capacités d'inférence de la logique floue avec la capacité d'apprentissage des réseaux de neurones [82], afin de former des réseaux neuro-flous puissants. L'utilisation des réseaux neuro-flous pour le contrôle bilatéral des systèmes de téléopération est alors appropriée puisque ces systèmes disposent d'une dynamique non linéaire et des incertitudes dues aux retards de transmission et au contact avec l'environnement inconnu.
Dans la première partie de ce chapitre, nous allons présenter les principes de la logique floue et la commande floue. La deuxième partie sera consacrée à la description des réseaux de neurones artificiels et leurs différentes architectures. Par la suite, dans la troisième partie du chapitre, nous allons aborder les techniques neuro-floues ainsi que les différentes combinaisons les plus utilisées dans le contrôle des systèmes. Enfin, la dernière partie sera consacrée à la synthèse d'un contrôleur neuro-flou de type ANFIS pour commander le système de téléopération à un degré de liberté, réalisé dans le cadre de ce travail, en utilisant un algorithme d'apprentissage basé sur le filtre de Kalman étendu.

4.2 Généralités sur la logique floue

4.2.1 Logique floue

La logique est l'étude du raisonnement, où ce dernier signifie d'obtenir 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 vaut 1 ou 0. La logique classique est largement utilisée dans le monde scientifique, néanmoins plusieurs problèmes liés à son utilisation et à son application existent du fait qu'elle présente des valeurs de vérité absolue. La logique floue est alors apparue, elle représente une transition de la vérité absolue à la vérité partielle présentée par un nombre compris dans l'intervalle [0 1] [83]. Cette logique permet de déduire des conclusions imprécises et floues à partir de prémisses

imprécises. Zadeh [84], qui est le fondateur de la logique floue en 1965, stipule que cette logique tente de modéliser de la façon la plus proche possible le comportement humain du fait que celui là est capable d'une part de raisonner et de prendre des décisions rationnelles concernant l'incertitude et l'imprécision, et d'autre part du fait qu'il est capable d'exécuter plusieurs tâches mentales et physiques sans mesures ni calculs.
La logique floue est la technique de modélisation et de manipulation des concepts flous modélisés par des ensembles flous, elle permet d'associer à des variables des degrés d'appartenance à des sous-ensembles flous, avec des valeurs comprises entre 0 et 1 en prenant ainsi en considération toutes les incertitudes sur la variable [83].

4.2.2 Sous-ensemble flou

Un sous-ensemble classique A d'un ensemble U est défini par sa fonction d'appartenance $\mu_A(x)$ qui caractérise chaque élément x appartenant à U [85].

$$\mu_A(x) = \begin{cases} 1 \; si \; x \in A \\ 0 \; si \; x \notin A \end{cases} \tag{4.1}$$

Cette fonction d'appartenance ne peut prendre que deux valeurs possibles : la valeur 1 si x appartient à A ou la valeur 0 si x n'appartient pas à A.
Un sous-ensemble flou A d'un ensemble U défini par une fonction d'appartenance $\mu_A(x)$ peut prendre plusieurs valeurs comprises entre 0 et 1. Le degré d'appartenance de l'élément x au sous-ensemble flou A appartient à l'intervalle [0 1] [85].

$$\mu_A(x) \in [0 \; 1] \tag{4.2}$$

4.2.3 Variable linguistique

Une variable linguistique décrit une situation imprécise par des mots ou des expressions tels que : (négatif, zéro, positif) ou (petit, très petit, moyen, grand, très grand).
Une variable linguistique prend une valeur linguistique qui est un ensemble flou défini sur l'univers de discours [86].

4.2.4 Fonction d'appartenance

Les variables linguistiques sont représentées par des fonctions d'appartenance, de telle manière à ce que chaque variable floue x d'un sous-ensemble flou A soit associée à une fonction d'appartenance qui représente le degré d'appartenance de x à A [87].
Les fonctions d'appartenance peuvent être représentées sous plusieurs formes selon l'application. Les formes les plus couramment utilisées dans la théorie du contrôle flou sont les fonctions triangulaires, trapézoïdales, sigmoïdes et gaussiennes [87].

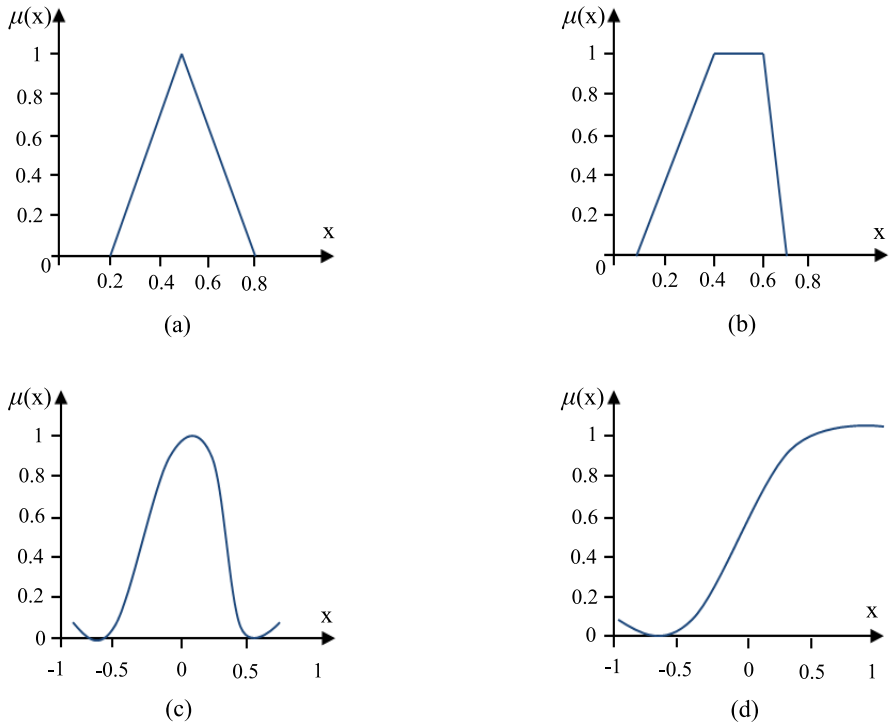


FIGURE 4.1: Fonctions d'appartenance usuelles.

Fonction triangulaire : Elle est caractérisée par trois paramètres (a,b et c) qui définissent les coordonnées des trois sommets (Figure 4.1(a)).

$$\mu(x) = max\left(min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right) \tag{4.3}$$

Fonction trapézoïdale : Cette fonction est définie par quatre paramètres (a,b,c et d) qui déterminent les coordonnées des quatre sommets (Figure 4.1(b)).

$$\mu(x) = max\left(min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}\right), 0\right) \tag{4.4}$$

Fonction gaussienne : Elle est caractérisée par deux paramètres (σ et m) (Figure 4.1(c)).

$$\mu(x) = exp\left(-\frac{(x-m)^2}{2\sigma^2}\right) \tag{4.5}$$

Fonction sigmoïde : La fonction sigmoïde est définie par deux paramètres (a et b) (Figure 4.1(d)).

$$\mu(x) = \left(\frac{1}{1 + exp\left(-a\left(x-c\right)\right)}\right) \tag{4.6}$$

4.2.5 Univers de discours

L'univers de discours représente le domaine de variation de la variable linguistique. Pratiquement, il représente le domaine du fonctionnement du processus à commander qu'il faut prendre en considération lors du réglage du régulateur.

4.2.6 Règle d'inférence

Les règles d'inférence sont l'ensemble des règles floues qui permettent de relier les variables floues d'entrée aux variables floues de sortie d'un système, au moyen de différents opérateurs flous [88].
Ces règles sont décrites sous la forme suivante :

Si condition 1 ET/OU condition 2 (ET/OU...) alors action sur les sorties OU
Si condition 3 ET/OU condition 4 (ET/OU...) alors action sur les sorties OU
...
Si condition n ET/OU condition $n+1$ (ET/OU...) alors action sur les sorties.

4.2.7 Opérations sur les ensembles flous

Dans les ensembles classiques, les variables sont reliées à l'aide d'opérateurs logiques tels que : ET (Intersection), OU (Union) et NON (complément à 1). Dans la logique floue, ces opérateurs sont interprétés respectivement par : Minimum, Maximum et Complément à 1, et sont définis par leurs fonctions d'appartenance [87].
La description de ces opérateurs dans les deux logiques classique et floue est illustrée dans le Tableau 4.1.

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

TABLE 4.1: Opérateurs dans les logiques classique et floue.

4.2.8 Commande floue

Les techniques de contrôle classique présentent beaucoup de limitations. Il est souvent difficile de modéliser le comportement du système par un ensemble d'équations mathématiques lorsque le système est non linéaire, partiellement inconnu ou avec des incertitudes dynamiques imprévisibles. Par conséquent, la logique floue est utilisée, où tous ces comportements imprévisibles peuvent être modélisés à l'aide de l'approche linguistique de Zadeh, selon un modèle proche du raisonnement humain [83].
L'utilisation de la logique floue dans le contrôle des systèmes représente un sujet de recherche d'actualité puisqu'elle permet d'obtenir une loi de commande considérablement efficace pour des systèmes complexes et fortement non linéaires sans avoir recours à un modèle mathématique bien défini, et cela en utilisant des inférences avec des règles floues basées sur des variables linguistiques. Dans ce qui suit, nous allons présenter les composants essentiels d'un régulateur flou (Figure 4.2) [83].

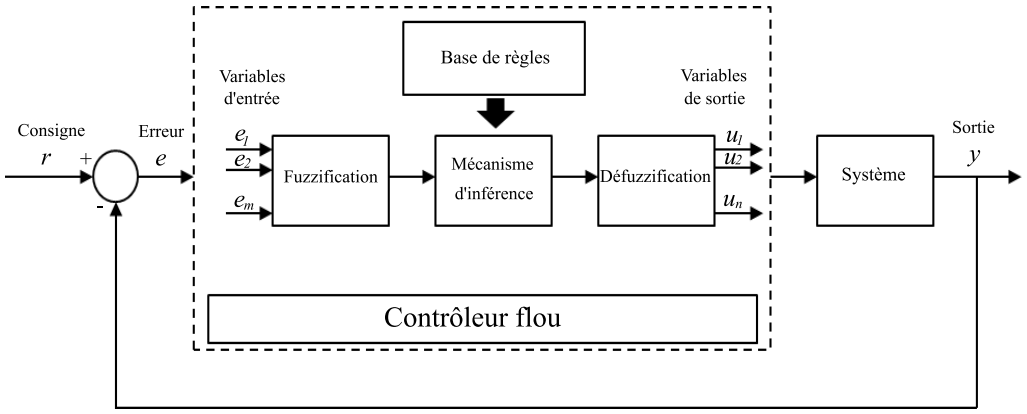


FIGURE 4.2: Schéma fonctionnel d'un régulateur flou.

Fuzzification

La fuzzification consiste à transformer des grandeurs numériques générées par un capteur en des variables linguistiques. Elle caractérise le degré avec lequel ces grandeurs numériques appartiennent à un sous-ensemble flou, en utilisant différentes formes de fonctions d'appartenance dont le choix du nombre et de la forme est réalisé par l'expert selon le domaine d'application et la facilité de calculs [83].

Considérons une entrée e (erreur) définie par l'ensemble des variables linguistiques : N=Négatif, Z= Zéro, P=Positif. L'univers de discours associé est [-1 1] (Figure 4.3).

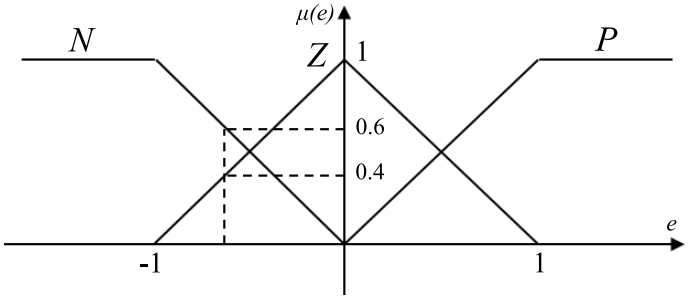


FIGURE 4.3: Exemple de fuzzification.

Les degrés d'appartenance de l'entrée e aux sous-ensembles sont :

$\mu_N(e) = 0.6$.

$\mu_Z(e) = 1 - 0.6 = 0.4$.

$\mu_P(e) = 0$.

Base de règles

La base de règles d'un système flou décrit en termes qualitatifs le comportement d'une sortie lorsqu'elle est soumise à diverses entrées. Ce comportement est décrit par les règles floues sous la forme [83] :

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.

Dans ce cas, les entrées peuvent être une erreur, une variation de l'erreur ou une somme d'erreurs, et la sortie peut être l'action de contrôle ou l'action de variation de contrôle.

Mécanisme d'inférence

Cette étape consiste à transformer la partie floue obtenue par la fuzzification en une nouvelle partie floue [83]. Elle comprend trois composants : une base de règles contenant une sélection de règles floues, une base de données qui définit les fonctions d'appartenance utilisées dans les règles floues et un mécanisme de raisonnement qui exécute la procédure d'inférence sur les règles pour obtenir une conclusion raisonnable. Si la sortie de la conséquence est exprimée par une variable linguistique, le régulateur flou est de type Mamdani, et si la sortie est une valeur numérique ou une fonction mathématique, le régulateur est de type Takagi-Sugeno [83].
L'inférence floue nécessite l'utilisation d'un opérateur flou pour chaque règle. Plusieurs méthodes s'appliquent pour la réalisation de ces opérateurs, néanmoins les méthodes d'inférence floue MAX-MIN de Mamdani et somme-produit de Sugeno sont les plus utilisées.

Défuzzification

Cette étape représente l'inverse de la fuzzification. Elle consiste à transformer la partie floue provenant de l'inférence en une valeur numérique pour former un signal de contrôle pour le système à commander [83,89]. Plusieurs méthodes de défuzzification existent dans la littérature telles que la méthode du centre de gravité, la moyenne des maximas et le centre des maximas [83] néanmoins, la méthode la plus utilisée dans le contrôle flou est la défuzzification par centre de gravité, qui consiste à calculer l'abscisse du centre de gravité de la surface générée par la fonction d'appartenance résultante.

4.3 Généralités sur les réseaux de neurones artificiels

4.3.1 Réseaux de neurones

Les réseaux de neurones artificiels (RNAs) sont considérés comme des modèles mathématiques simplifiés du cerveau humain [90]. Ce sont des systèmes cellulaires physiques capables d'acquérir, de stocker et d'utiliser des connaissances expérimentales, à l'aide d'un algorithme d'apprentissage [91].

Les réseaux de neurones artificiels sont composés essentiellement d'éléments de traitement appelés neurones artificiels, et des connexions entre eux avec des coefficients de pondération liés à ces connexions ainsi qu'un algorithme d'apprentissage [92].

Parmi les principaux avantages des réseaux de neurones artificiels, on citera la grande capacité de calcul ainsi que la capacité d'apprentissage de manière itérative. Lors de la phase d'apprentissage, les poids de connexions du réseau varient de telle sorte à ce que ce dernier apprenne à reproduire les sorties désirées pour des entrées connues, et si de nouvelles entrées sont fournies au réseau, les meilleurs résultats sont obtenus en fonctions de ces nouvelles entrées [92].

Les modèles des réseaux de neurones artificiels sont classés par rapports à leurs méthodes d'apprentissage (supervisé et non supervisé), leurs architectures (bouclés et non bouclés) ou leurs types de sortie (binaire ou continue) [92].

4.3.2 Neurone biologique

Le neurone est la cellule fondamentale du système nerveux central. Son rôle est de conduire des impulsions électriques issues des réactions physico-chimiques dans certaines conditions de fonctionnement [93]. Le neurone est composé de quatre parties principales [93] : Les dendrites, le corps cellulaire, l'axone et les synapses (Figure 4.4).

Les dendrites permettent d'acquérir en continu les signaux issus des autres neurones.

Le corps cellulaire est l'unité responsable du traitement de toutes les informations provenant des dendrites.

L'axone a pour mission de guider les signaux vers d'autres neurones connectés.

Les synapses sont des connexions qui permettent le transfert des signaux électriques des axones d'un neurone particulier aux dendrites des autres neurones.

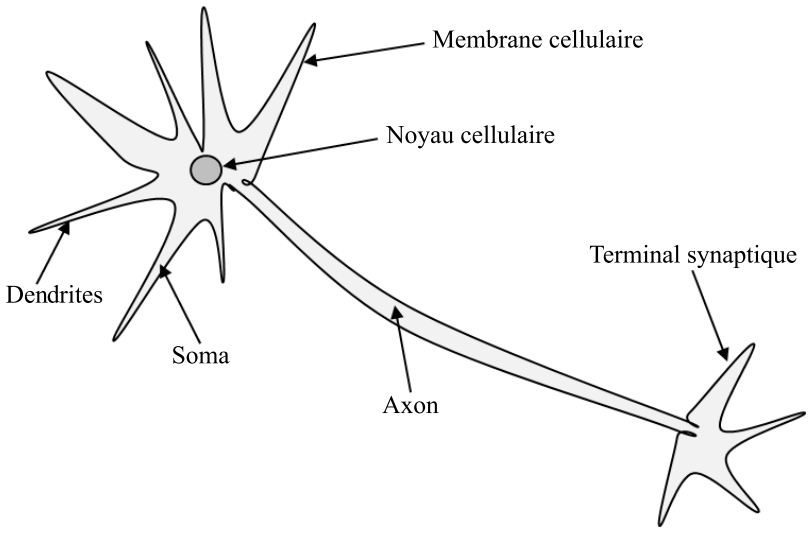


FIGURE 4.4: Neurone biologique.

4.3.3 Neurone formel

Le neurone formel représente un modèle simplifié du neurone biologique. Ce modèle a été proposé par McCulloch et Pitts en 1943, il comprend les principales caractéristiques d'un neurone biologique [93].

La Figure 4.5 représente un modèle d'un neurone artificiel, où l'ensemble des entrées $\{x_1, x_2, ..., x_n\}$ est analogue aux impulsions électriques rassemblées par les dendrites des

poids du neurone biologique, les jonctions synaptiques sont représentées par des poids synaptiques $\{w_1, w_2, ..., w_n\}$, et z est la sortie du corps cellulaire artificiel qui est la somme pondérée de ses entrées, définie par [93] :

$$z = \sum_{i=1}^{n} w_i x_i - b \tag{4.7}$$

Où b représente le biais (seuil d'activation).
La sortie du neurone y est définie par :

$$y = f(z) \tag{4.8}$$

Où f est la fonction d'activation, qui peut prendre plusieurs formes à savoir : Fonction signe, tout ou rien, fonction affine, plus au moins à seuil, saturation, fonction gaussienne et fonction arc-tangente.

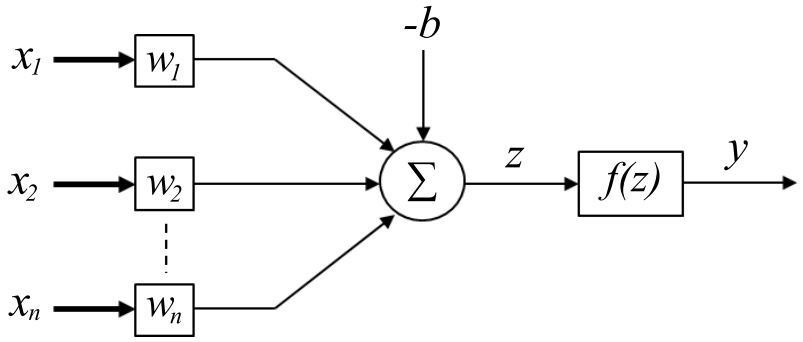


FIGURE 4.5: Modèle général d'un neurone formel.

4.3.4 Architectures des réseaux de neurones artificiels

Un RNA est composé d'un nombre important de neurones artificiels et d'un très grand nombre d'interconnexions entre eux. En fonction de ces interconnexions, différents types d'architectures de RNAs sont définis [94].

Architecture non-bouclée à une seule couche

Ce type d'architectures de RNAs ne comporte qu'une seule couche d'entrée et une seule couche de neurones, qui représente la couche de sortie, où l'information circule toujours de la couche d'entrée à la couche de sortie dans une seule direction (Figure 4.6) [93]. Ces réseaux sont généralement utilisés dans les problèmes de classification et de filtrage linéaire.

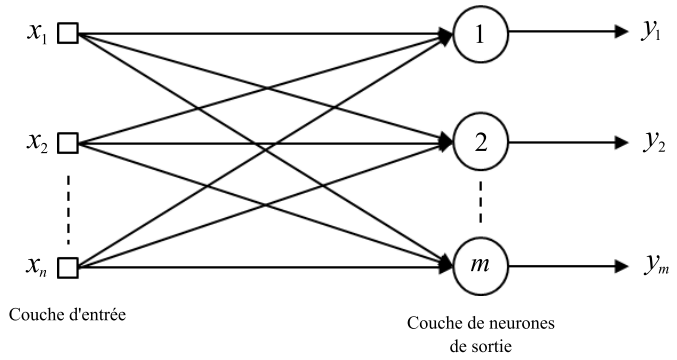


FIGURE 4.6: Réseau de neurones artificiels non bouclé mono-couche.

Parmi les principaux types de réseaux non-bouclés à une seule couche, on cite le Perceptron et l'ADALINE, qui représentent deux architectures prédictives dont les algorithmes d'apprentissage utilisés reposent respectivement sur la règle de Hebb et la règle de Delta [93].

Architecture non-bouclée à multi-couches

Ce type d'architectures de RNAs comporte plusieurs couches : une couche d'entrée, une couche de neurones de sortie et une ou plusieurs couches cachées (Figure 4.7). Dans ces réseaux non-bouclés, la sortie d'une couche n'affecte pas la même couche [94], et le nombre de couches cachées et leur nombre de neurones respectifs dépend de la nature et de la complexité du problème à résoudre ainsi que de la qualité et de la quantité des données disponibles sur ce problème [93].

Ce type de RNAs est utilisé pour résoudre divers problèmes tels que la classification des formes, l'approximation des fonctions, l'optimisation, l'identification des systèmes et le contrôle des procédés.

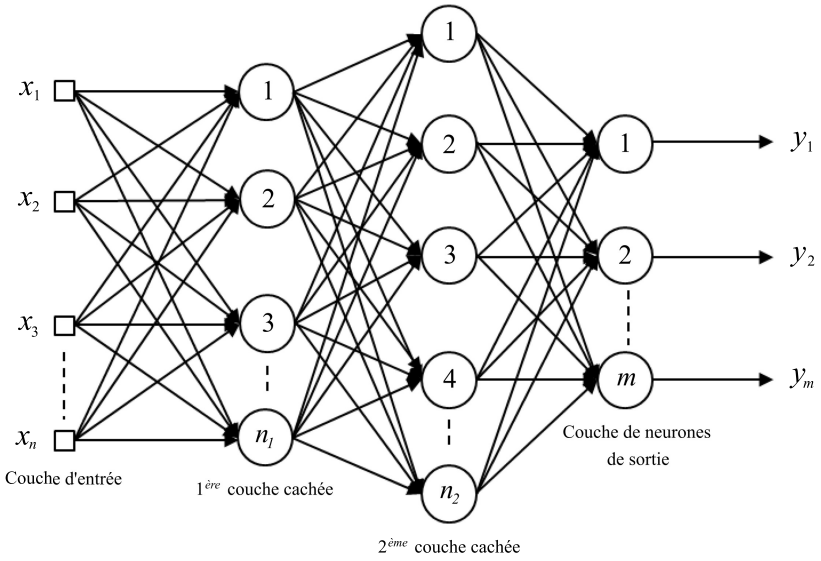


FIGURE 4.7: Réseau de neurones artificiels non bouclé multi-couche.

Parmi les principaux réseaux non-bouclés multi-couches, on trouve le Perceptron multi-couches et la fonction de base radiale [93].

Architecture bouclée (récurrente)

Dans ce type d'architectures, les sorties des neurones (couches) sont utilisées comme des entrées de rétroaction pour d'autres neurones (couches), permettant ainsi aux signaux de circuler dans les deux sens (Figure 4.8).

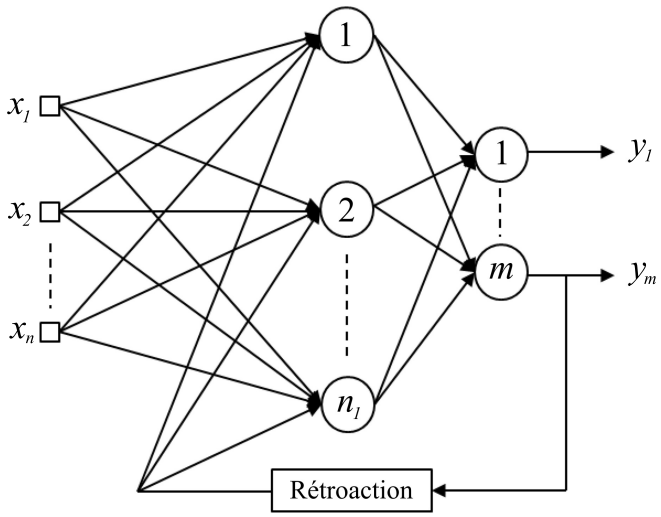


FIGURE 4.8: Réseau de neurones artificiels bouclé.

Ces réseaux sont très puissants, leur dynamique leur permettent de changer leurs états continuellement jusqu'à atteindre un point d'équilibre [94]. Ce type de réseaux est utilisé généralement pour la prédiction temporelle, l'identification, l'optimisation et le contrôle des processus. Parmi les principaux réseaux de neurones bouclés, on cite les réseaux de Hopfield et de Perceptron [93].

4.4 Processus d'apprentissage

L'une des principales caractéristiques des RNAs est leur capacité d'apprentissage. Un réseau de neurone artificiel peut apprendre et trouver la relation entre les entrées et les sorties pour généraliser une solution et produire une sortie proche de la sortie désirée [93]. Le processus d'apprentissage consiste à appliquer plusieurs étapes de façon ordonnée afin d'ajuster les poids et les seuils synaptiques des neurones artificiels pour généraliser des solutions produites par les sorties du réseau. L'ensemble de ces étapes ordonnées s'appelle algorithme d'apprentissage [93].
Le processus d'apprentissage peut s'effectuer selon plusieurs règles et différentes manières. A cet effet, on distingue trois types d'apprentissages : Supervisé, non supervisé et par renforcement.

4.4.1 Apprentissage supervisé

Dans l'apprentissage supervisé, les entrées sont fournies, et c'est au réseau de calculer les sorties correspondantes. Le réseau compare ensuite les sorties obtenues aux sorties désirées pour déterminer l'erreur qui sera utilisée pour modifier le réseau afin d'atteindre les performances souhaitées (Figure 4.9) [94].
Parmi les algorithmes d'apprentissage supervisé les plus connus, on cite l'algorithme de rétro-propagation de l'erreur.

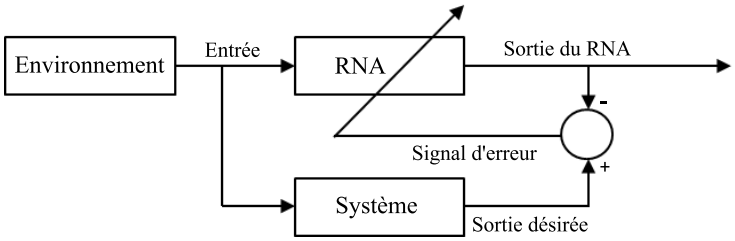


FIGURE 4.9: Apprentissage supervisé.

Algorithme de rétro-propagation

La méthode de rétro-propagation de l'erreur est une technique d'apprentissage supervisé. Elle permet d'ajuster les poids et les biais du RNA afin de minimiser l'erreur quadratique entre la sortie calculée du réseau et sa sortie réelle. Pour chaque couple entrée/sortie, une erreur est calculée. Si la réponse du réseau est différente de la réponse réelle, les poids et les biais sont modifiés en ligne sur le réseau de manière à tendre l'erreur vers zéro.

Considérons le réseau de neurones à trois couches (n,m,p) illustré dans la Figure 4.10, et soit x^q le vecteur d'entrée du réseau et y^q le vecteur des sorties désirées.

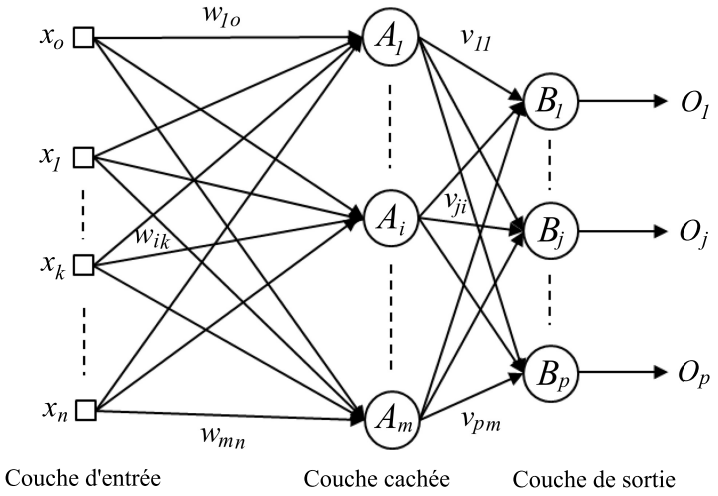


FIGURE 4.10: Réseau de neurones à trois couches.

Considérons le problème de minimisation de la fonction objectif, définie par l'erreur qua-

dratique E^q entre la sortie calculée du réseau O^q et la sortie désirée y^q.

$$E^q = \frac{1}{2}\sum_j \left(O_j^q - y_j^q\right)^2 \tag{4.9}$$

Soit v_{ji} les poids de connexion du neurone caché i au neurone de sortie j, et w_{ik} les poids de connexion du nœud d'entrée k au neurone caché i.

L'objectif consiste à calculer les poids v_{ji} et w_{ik} afin de minimiser E^q.

Les équations des mises à jour des poids v_{ji} et w_{ik} sont définies par :

$$v_{ji} \longrightarrow v_{ji} + \Delta v_{ji} \tag{4.10}$$

$$w_{ik} \longrightarrow w_{ik} + \Delta w_{ik} \tag{4.11}$$

Premièrement, les poids v_{ji} sont ajustés en utilisant la méthode de la descente du gradient. Pour un neurone de sortie j et un neurone caché i, les poids v_{ji} sont mis à jour par [87] :

$$\Delta v_{ji} = \sum_{j=1}^{N} \Delta^q v_{ji} = -\eta \sum_{j=1}^{N} \frac{\partial E^q}{\partial v_{ji}} = \sum_{j=1}^{N} \left(-\eta \delta_j^q z_i^q\right) \tag{4.12}$$

Où z_i^q est l'entrée du neurone caché i, définie par :

$$z_i^q = f_i\left(\sum_{k=0}^{n} w_{ik} x_k^q\right) \tag{4.13}$$

Et :

$$\delta_j^q = \left(O_j^q - y_j^q\right) f_i^{'}\left(\sum_{k=1}^{m} v_{jk} z_k^q\right) \tag{4.14}$$

Où f est la fonction d'activation.

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

$$\Delta^q w_{ik} = -\eta \frac{\partial E^q}{\partial w_{ik}} \tag{4.15}$$

Avec :

$$\frac{\partial E^q}{\partial w_{ik}} = \frac{\partial E^q}{\partial O_i^q}\frac{\partial O_i^q}{\partial w_{ik}} \tag{4.16}$$

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

$$O_i^q = f_i\left(\sum_{k=0}^{n} w_{il}x_l^q\right) = z_i^q \tag{4.17}$$

Nous avons :

$$\frac{\partial O_i^q}{\partial w_{ik}} = f_i^{'}\left(\sum_{l=0}^{n} w_{ik}x_l^q\right)x_k^q \tag{4.18}$$

Soit $\delta_i^q = \frac{\partial E^q}{\partial O_i^q}$ dans la couche cachée, défini par :

$$\delta_i^q = \frac{\partial E^q}{\partial O_i^q} = \sum_{j=1}^{p}\frac{\partial E^q}{\partial O_j^q}\frac{\partial O_j^q}{\partial O_i^q} \tag{4.19}$$

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

$$O_i^q = f_j^{'}\left(\sum_{l=1}^{m} v_{il}z_l^q\right)v_{ji} \tag{4.20}$$

Donc, δ_i^q pour le neurone caché i, est calculé à partir des valeurs déjà connues de δ_j^q pour tout j de la couche de sortie.

Cet algorithme est appelé algorithme de rétro-propagation parce qu'il commence à transmettre les modèles d'entrée x^q afin d'atteindre la couche de sortie, puis calcule les δ_j^q pour tout neurone de sortie j, ensuite propage les δ_j^q en arrière vers la couche inférieure (la couche cachée), afin de calculer les δ_i^q pour tous les neurones i de cette couche.

4.4.2 Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, un algorithme basé sur l'apprentissage non supervisé ne nécessite aucune connaissance sur les sorties désirées (Figure 4.11) [93]. Le réseau doit alors s'auto-organiser et s'adapter aux changements d'entrées pour déterminer sa sortie, et cela en ajustant les poids et les seuils synaptiques des neurones artificiels [94].

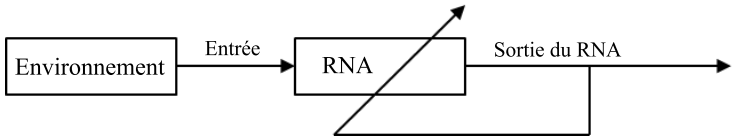


FIGURE 4.11: Apprentissage non supervisé

4.4.3 Apprentissage par renforcement

Les algorithmes d'apprentissage basés sur cette méthode ajustent les paramètres du RNA en fonction des informations acquises lors de l'interaction avec l'environnement, afin d'évaluer les performances d'apprentissage [93].
Le processus d'apprentissage se fait par essais et erreurs, la réponse du réseau pour une entrée est alors évaluée si elle est satisfaite ou non. Si la sortie est satisfaite, les poids et les seuils sont progressivement incrémentés [93].

4.5 Réseaux neuro-flous

4.5.1 Systèmes neuro-flous

D'une part, les systèmes flous présentent l'avantage d'être construits à partir des connaissances humaines, et cela en modélisant le cerveau humain par son mode de fonctionnement, de plus, grâce à l'utilisation des variables linguistiques, ces systèmes possèdent une capacité descriptive élevée, néanmoins, aucune méthode formelle n'existe permettant de définir les fonctions d'appartenance ou de construire la base de règles pour les systèmes. D'autre part, parmi les caractéristiques principales des réseaux de neurones artificiels est leurs structures flexibles ainsi que leurs capacités d'apprentissage à partir de données expérimentales. En raison de ces avantages, les réseaux de neurones sont utilisés dans de nombreuses applications telles que la reconnaissance de formes, la modélisation et le contrôle des procédés. Par conséquent, la combinaison des réseaux de neurones et de la logique floue, dans laquelle le raisonnement flou est réalisé dans un réseau de neurones,

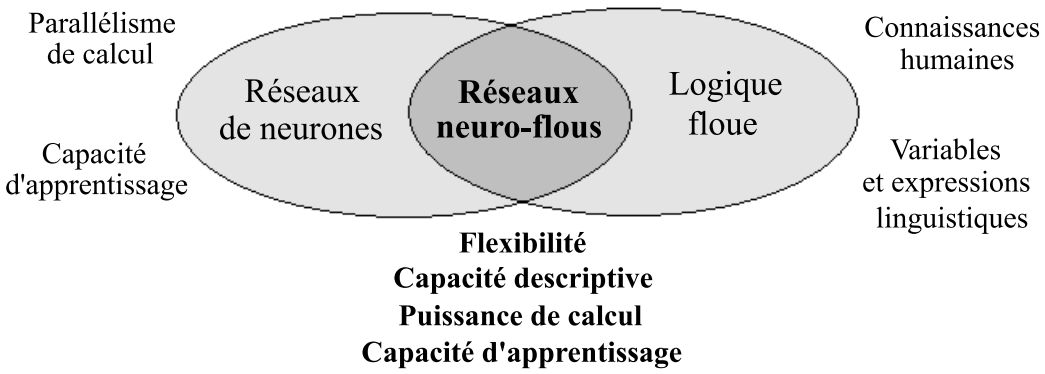


FIGURE 4.12: Réseaux neuro-flous.

permet de construire des systèmes neuro-flous plus efficaces, en combinant les capacités des réseaux de neurones artificiels avec la capacité d'inférence de la logique floue [82] (Figure 4.12). Une telle combinaison permet d'adapter les règles linguistiques et les fonctions d'appartenance ou optimiser les règles existantes. Le processus d'apprentissage fonctionne sur des informations locales et ne provoque que des modifications locales sur le système neuro-flou [83].

Parmi les avantages de la commande par les réseaux neuro-flous est qu'elle n'exige pas de connaissances sur le modèle mathématique du système à commander, et comme les systèmes neuro-flous sont considérés comme des approximateurs universels, la commande à base de ces réseaux est utilisée pour résoudre plusieurs problèmes de commande dont le modèle mathématique est inconnu ou présente des incertitudes dynamiques.

Dans notre travail, nous allons utiliser un contrôleur neuro-flou pour commander un système de téléopération, sans avoir recours aux modèles mathématiques précis du maître et de l'esclave. L'utilisation d'un tel contrôleur a pour objectif de s'adapter aux variations de la dynamique des systèmes maître et esclave particulièrement lorsque ce dernier est en contact avec l'environnement inconnu.

4.5.2 Type de combinaisons neuro-floues

Selon leurs fonctionnements et leurs architectures, il existe trois grandes familles de combinaisons de réseaux de neurones avec la logique floue [91] :

Système flou neuronal

Dans les système flous neuronaux, les réseaux de neurones sont utilisés pour augmenter le traitement numérique des ensembles flous, tel que l'élicitation des fonctions d'appartenance. Ce type de systèmes est utilisé dans les applications du contrôle des procédés.

Système neuronal flou

Les systèmes neuronaux flous conservent les propriétés et les architectures de base des réseaux de neurones et fuzzifient certains de leurs éléments. Dans ce type de réseaux, un neurone peut devenir flou et sa réponse peut être de type relation floue plutôt que de type sigmoïde. Les systèmes neuronaux flous sont utilisés principalement dans les applications de reconnaissance de formes.

Système neuro-flou hybride

Dans ce type de combinaisons, les techniques floues et les réseaux de neurones jouent un rôle clé. En utilisant leurs avantages individuels, ces deux techniques s'intègrent et se complètent pour atteindre un objectif commun. Les systèmes neuro-flous hybrides exploitent la puissance des réseaux de neurones pour leurs capacités de traitement et les avantages de la logique floue pour sa capacité de raisonnement flexible pour la prise de décision. Ce type de systèmes est utilisé dans les applications de contrôle des procédés et la reconnaissance de formes.

4.5.3 Types d'implémentation des réseaux neuro-flous

Système d'inférence neuro-floue

Un système d'inférence neuro-floue est réalisé sous forme d'un réseau de neurones, où les paramètres de la logique floue sont représentés par les poids synaptiques du réseau de neurones. Par conséquent, la structure neuronale permet d'ajuster automatiquement et optimiser les fonctions d'appartenance, et cela en modifiant les poids des connexions du RN via un algorithme d'apprentissage approprié.

Système d'inférence neuro-floue adaptatif (ANFIS)

Les systèmes d'inférence neuro-floue adaptatifs appartiennent à la famille des systèmes neuro-flous hybrides, qui combinent les avantages de la logique floue avec ceux des réseaux de neurones dans un seul réseau.
La structure ANFIS a été proposée par Jang en 1993 [16]. Elle décrit le fonctionnement d'un système flou, avec des règles qui peuvent être de type Takagi-Sugeno ou Mamdani, sous forme d'un réseau de neurones, entraîné à l'aide d'un algorithme d'apprentissage de rétro-propagation, qui ajuste et optimise les paramètres de la prémisse et de la conséquence du réseau [91]. Le système ANFIS comporte cinq couches, où les nœuds adaptatifs contenant des paramètres sont situés dans la première et la quatrième couche, alors que les autres nœuds sans paramètres sont fixes.
Les modèles ANFIS sont utilisés dans plusieurs applications de traitement de signal, de filtrage adaptatif et de contrôle des systèmes, et présentent de bonnes performances notamment dans les applications de contrôle. Dans notre travail, nous allons utiliser un contrôleur basé sur ce type d'architectures pour contrôler le système bilatéral maître-esclave à une degré de liberté, réalisé dans le cadre de ce travail de thèse.

4.5.4 Méthodes d'apprentissage des réseaux neuro-flous

Les algorithmes de calcul des réseaux de neurones artificiels et des réseaux neuro-flous, utilisés plus particulièrement en robotique fournissent des systèmes intelligents capables de s'adapter aux environnements dynamiques inconnus, grâce à des mécanismes d'apprentissage en ligne ou hors ligne [95].

La méthode d'apprentissage en ligne consiste à ajuster les paramètres du réseau neuro-flou en temps réel au cours du processus, simultanément avec le fonctionnement du système à contrôler, en utilisant des algorithmes d'optimisations locale et globale. Dans ce cas, les données sont présentes de manière séquentielle par rapport au temps, et l'algorithme estime et prédit les paramètres du réseau en fonction des actions passées et finies. Par contre, dans le cas d'un apprentissage hors ligne, les paramètres du réseau sont mis à jour une seule fois, dans ce cas, ces paramètres sont alors fixes peu importe l'évolution du système à contrôler dans le temps.

Dans notre travail, nous allons commander le système de téléopération bilatérale présenté précédemment avec un contrôleur neuro-flou de type ANFIS, en utilisant une carte FPGA. Nous avons opté pour la méthode d'apprentissage en ligne pour ajuster les paramètres du régulateur développé, et cela en tirant profit des avantages de la carte FPGA utilisée en termes de vitesse et de parallélisme de calcul ainsi que de la fréquence de fonctionnement élevée.

4.5.5 Structure ANFIS

La combinaison des réseaux de neurones avec les systèmes flous dans un seul réseau permet au réseau ANFIS de tirer profit des avantages de chacune des deux approches simultanément.

La structure du réseau neuro-flou ANFIS est composée de deux parties : Une partie prémisse et une partie conséquence, connectées entre elles par une base de règles floues sous forme d'un réseau [96]. Cette structure consiste à adapter ces règles floues aux changements de conditions. De ce fait, les systèmes ANFIS peuvent optimiser et ajuster automatiquement les fonctions d'appartenance [87] à l'aide d'un algorithme d'apprentissage.

Un réseau neuro-flou avec un modèle d'inférence floue du premier ordre de Sugeno, comprenant deux entrées x et y, et une seule sortie z, est représenté dans la Figure 4.13. Le réseau comporte 5 couches dont la fonction de chacune est décrite dans ce qui suit.

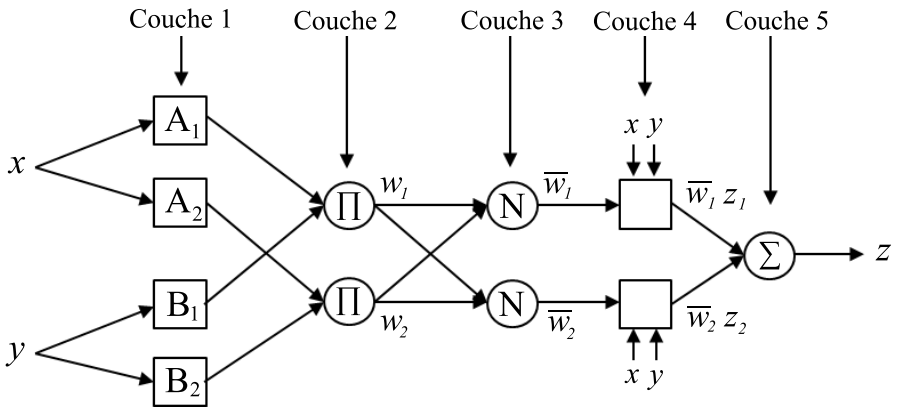


FIGURE 4.13: Architecture d'un réseau ANFIS.

Les deux règles floues sont définies comme suit [97] :

$$\text{Règle } 1: \text{ si } x \text{ est } A_1 \text{ et } y \text{ est } B_1 \text{ alors } z_1 = p_1 x + q_1 y + r_1 \tag{4.21}$$

$$\text{Règle } 2: \text{ si } x \text{ est } A_2 \text{ et } y \text{ est } B_2 \text{ alors } z_2 = p_2 x + q_2 y + r_2 \tag{4.22}$$

Couche 1 : Fuzzification

Chaque nœud i de cette couche est un nœud adaptatif doté d'une fonction d'appartenance avec des paramètres ajustables :

$$O_{1,j} = \mu_{A_j}(x), \quad pour\ j = 1, 2 \tag{4.23}$$

$$O_{1,j} = \mu_{B_{j-2}}(y), \quad pour\ j = 3, 4 \tag{4.24}$$

$O_{1,j}$ 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.

$$O_{2,j} = w_j = \mu_{A_j}(x)\mu_{B_j}(y), \quad j = 1, 2. \tag{4.25}$$

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^{ime} règle.

$$O_{3,j} = \overline{w}_j = \frac{w_j}{w_1 + w_2}, \quad j = 1, 2. \tag{4.26}$$

Couche 4 : Calcul des sorties des règles

Chaque nœud de cette couche est un nœud adaptatif dont la fonction est :

$$O_{4,j} = \overline{w}_j z_j = \overline{w}_j(p_j x + q_j y + r_j) \tag{4.27}$$

Où $\overline{w}_j$ est la sortie de la couche 3 qui représente le degré d'activation normalisé de la règle et (p_j, q_j et 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 :

$$O_{5,j} = z = \sum_j \overline{w}_j z_j = \frac{\sum\limits_j w_j z_j}{\sum\limits_j w_j} \tag{4.28}$$

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.

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

Afin de réaliser une téléopération bilatérale à hautes performances, les algorithmes de contrôle doivent être stables, robustes et rapides [98]. En terme de transparence, l'architecture de commande à quatre canaux de Lawrence est la plus efficace, car elle suppose une parfaite connaissance des impédances du maître et de l'esclave [32]. Cependant, les dispositifs maître et esclave représentent des systèmes non linéaires et complexes, avec une dynamique qui ne peut pas parfois être disponible en raison des incertitudes du modèle [27,28] car l'esclave interagit souvent avec des environnements inconnus [25]. De plus, lorsque les positions et les forces de contact de l'esclave avec l'environnement distant sont simultanément contrôlées, il est difficile d'obtenir de bonnes performances de suivi [9] et de garantir la stabilité de contact de l'esclave avec l'environnement. D'une autre part, en terme de mise en œuvre matérielle, afin d'obtenir une téléopération avec de bonnes performances, il est nécessaire de garantir un taux de contrôle élevé [17], en particulier lorsque les algorithmes de contrôle sont complexes. L'utilisation des contrôleurs adaptatifs ainsi que les techniques de l'intelligence artificielle est alors une solution adéquate pour remédier à ces contraintes. Plusieurs travaux ont été menés dans ce domaine. L'auteur dans [10] a proposé un contrôleur approché par un modèle de Takagi-Sugeno pour un système de téléopération bilatérale non linéaire. Dans [11], un contrôle adaptatif basé sur des réseaux de neurones a été adopté, en considérant un système de téléopération avec des incertitudes dynamiques et des perturbations externes. L'utilisation des techniques neuro-floues dans le contrôle de systèmes de téléopération a été présentée dans [12–15,81]. Ces travaux avaient pour objectif d'améliorer la stabilité et les performances du suivi de la force et de la position en compensant les incertitudes liées à la dynamique du système maître-esclave. Les auteurs ont également proposé des contrôleurs adaptatifs pour des systèmes de téléopération avec une dynamique inconnue [27] ou en considérant des systèmes avec des incertitudes dynamiques et cinématiques [26] et des retards de communication [25, 29]. L'auteur dans [28] a proposé un contrôleur adaptatif pour le robot maître avec des incer-

titudes dynamiques et un autre pour l'esclave avec des incertitudes dynamiques et une zone morte.

Dans ce travail de thèse, nous proposons un contrôle adaptatif neuro-flou (ANFIS) pour le système de téléopération réalisé précédemment, en utilisant une carte FPGA Zedboard [99]. Ce type de contrôle ne nécessite aucune connaissance de la dynamique du maître et de l'esclave, ni de celle de l'opérateur humain et de l'environnement. Les contrôleurs ANFIS qui sont une approche sans modèle sont utilisés pour s'adapter aux changements de la dynamique du système lorsque l'esclave est en contact avec un environnement distant, en ajustant les paramètres de la conséquence du réseau neuro-flou dans un temps très court avec un algorithme d'apprentissage basé sur le filtre de Kalman étendu, et cela en tirant profit du calcul parallèle du FPGA et de sa fréquence d'échantillonnage élevée [98], ce qui accélère l'algorithme de contrôle afin de le faire converger vers les performances désirées.

Dans cette section, nous développons un contrôleur neuro-flou adaptatif de type ANFIS, pour commander le système maître-esclave. Le système sera commandé en utilisant deux architectures de contrôle : Contrôle position-position [100] et contrôle à quatre canaux.

Dans ce qui suit, nous présentons la structure du régulateur ANFIS proposé pour les systèmes maître et esclave, pour chaque architecture de contrôle.

4.6.1 Contrôle position-position

L'architecture de contrôle position-position est basée sur l'échange de signaux de position entre le poste maître et celui de l'esclave, où la position du maître est retournée vers l'esclave et vice versa. Dans cette partie, nous allons commander en position le maître et l'esclave en utilisant deux contrôleurs ANFIS.

Dans le schéma de commande illustré sur la Figure 4.14, θ_m et θ_s sont respectivement les positions du maître et de l'esclave, T_e est le couple d'interaction de l'esclave avec l'environnement et T_h est le couple exercé par l'opérateur sur le maître. u_m et u_s sont les signaux de commande délivrés par les actionneurs pour contrôler respectivement les mouvements du maître et de l'esclave.

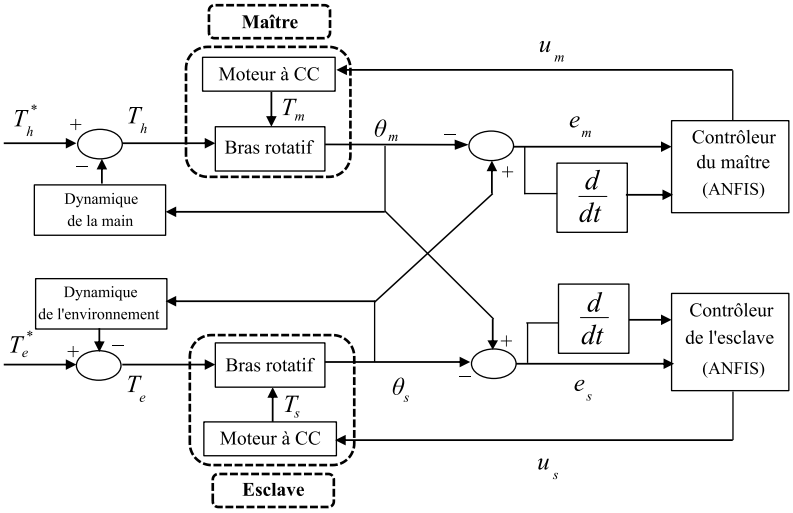


FIGURE 4.14: Schéma de commande position-position.

4.6.2 Contrôle à quatre canaux

Dans ce cas de contrôle, les signaux de force et de position sont échangés simultanément entre le maître et l'esclave, où la force d'interaction de l'esclave avec l'environnement est mesurée avec un capteur de force, et retournée vers le maître. Ici, l'esclave est contrôlé en position alors que le maître est commandé en force, en utilisant deux contrôleurs neuro-flous ANFIS (Figure 4.15), dont le développement sera présenté dans la section qui suit.

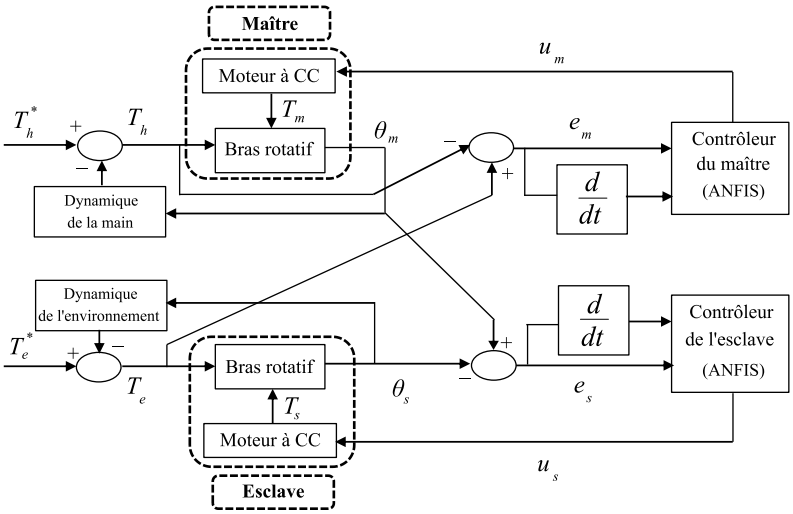


FIGURE 4.15: Schéma de commande à quatre canaux.

4.6.3 Algorithme de commande ANFIS

Dans cette section, nous développons le contrôleur ANFIS pour le système maître (Position) en adoptant la stratégie de commande Position-position. Ce développement est également le même pour le contrôleur du système esclave (position), ainsi que pour les contrôleurs du maître (couple) et celui de l'esclave (position) pour la stratégie de commande à quatre canaux.

Considérons un réseau neuro-flou de premier ordre avec un système d'inférence floue de type Takagi-Sugeno, comportant deux entrée x_1 et x_2 et une seule sortie f (Figure 4.16).

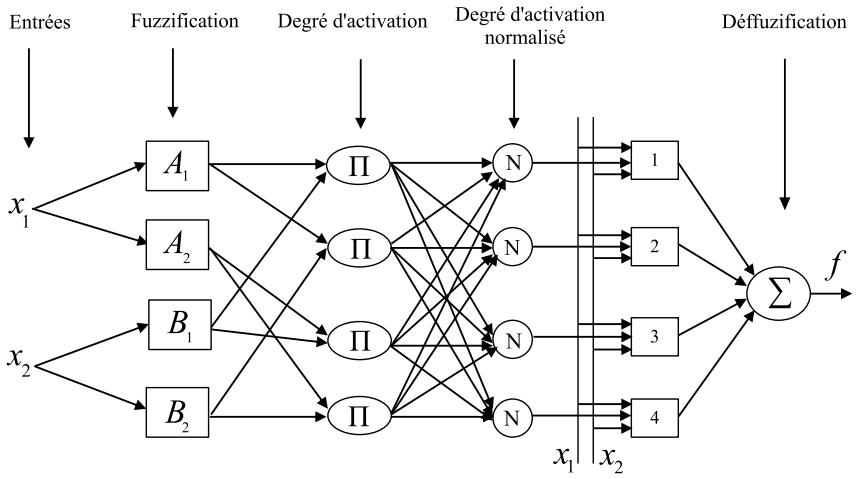


FIGURE 4.16: Structure du régulateur ANFIS.

Les règles floues sont définies comme suit :

$$\text{Règle } 1: \text{ si } x_1 \text{ est } A_1 \text{ et } x_2 \text{ est } B_1 \text{ alors } f_1 = p_1x_1 + q_1x_2 + r_1 \tag{4.29}$$

$$\text{Règle } 2: \text{ si } x_1 \text{ est } A_1 \text{ et } x_2 \text{ est } B_2 \text{ alors } f_2 = p_2x_1 + q_2x_2 + r_2 \tag{4.30}$$

$$\text{Règle } 3: \text{ si } x_1 \text{ est } A_2 \text{ et } x_2 \text{ est } B_1 \text{ alors } f_3 = p_3x_1 + q_3x_2 + r_3 \tag{4.31}$$

$$\text{Règle } 4: \text{ si } x_1 \text{ est } A_2 \text{ et } x_2 \text{ est } B_2 \text{ alors } f_4 = p_4x_1 + q_4x_2 + r_4 \tag{4.32}$$

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.

Nous associons deux ensembles flous pour chacune des entrées x_1 et x_2 nommés $N(Negative)$ et $P(Positive)$. μ_P et μ_N sont les degrés d'appartenance des variables x_i aux sous ensembles flous A_i et B_i, définis par les fonctions d'appartenance suivantes (Figure 4.17) : Pour i=1,2 :

$$\mu_P(x_i) = \begin{cases} 0, & \text{si} \quad x_i < -1, \\ 0.5x_i + 0.5, & \text{si} \quad -1 < x_i < 1, \\ 1, & \text{si} \quad x_i > 1. \end{cases} \tag{4.33}$$

$$\mu_N(x_i) = \begin{cases} 1, & \text{si} \quad x_i < -1, \\ -0.5x_i + 0.5, & \text{si} \quad -1 < x_i < 1, \\ 0, & \text{si} \quad x_i > 1. \end{cases} \tag{4.34}$$

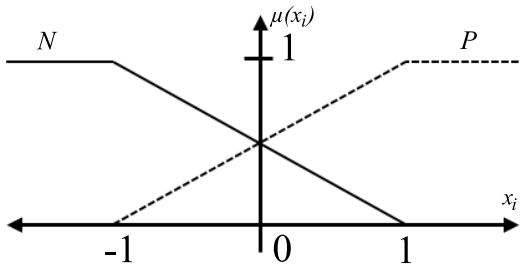


FIGURE 4.17: Fonctions d'appartenance.

En utilisant la méthode de déffuzification par centre de gravité, la valeur numérique de la sortie u est donnée par [97] :

$$u = \frac{\sum_{j=1}^{4} f_j w_j}{\sum_{j=1}^{4} w_j} \tag{4.35}$$

Où :

$$\begin{cases} w_1 = \mu_P(x_1).\mu_P(x_2) = \mu_{A_1}(x_1).\mu_{B_1}(x_2) \\ w_2 = \mu_P(x_1).\mu_N(x_2) = \mu_{A_1}(x_1).\mu_{B_2}(x_2) \\ w_3 = \mu_N(x_1).\mu_P(x_2) = \mu_{A_2}(x_1).\mu_{B_1}(x_2) \\ w_4 = \mu_N(x_1).\mu_N(x_2) = \mu_{A_2}(x_1).\mu_{B_2}(x_2) \end{cases} \tag{4.36}$$

4.6.4 Algorithme d'apprentissage

Le processus d'apprentissage consiste à identifier les paramètres de la prémisse et de la conséquence. Dans cette section, nous développons l'algorithme d'apprentissage pour le système maître, et le développement est le même pour le système esclave. Ainsi, posons y_d et y respectivement les sorties désirée et réelle du système maître (respectivement θ_s et θ_m). Dans ce cas, nous considérons que les paramètres de la prémisse sont fixes, alors que ceux de la conséquence sont ajustés en minimisant la fonction objectif suivante [14] :

$$J = \frac{1}{2}e^2 \tag{4.37}$$

Où $e = y_d - y$.

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 [101] comme suit :

$$\Phi_j(k+1) = \Phi_j(k) - \alpha(k)\frac{\partial J}{\partial \Phi_j} \tag{4.38}$$

Nous avons :

$$\frac{\partial J}{\partial \Phi_j} = -e\frac{\partial y}{\partial \Phi_j} = -e\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j} \tag{4.39}$$

A partir des équations (4.38) et (4.39), nous aurons :

$$\Phi_j(k+1) = \Phi_j(k) + \alpha(k)\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j}e \tag{4.40}$$

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 (4.40) peut être écrite comme suit :

$$\Phi_j(k+1) = \Phi_j(k) + K_j^{'} \left(\Psi_j\right)^T e \tag{4.41}$$

Où :

$$\left(\Psi_j\right)^T = \frac{\partial u}{\partial \Phi_j} = \begin{bmatrix} \frac{\partial u}{\partial p_j} \\ \frac{\partial u}{\partial q_j} \\ \frac{\partial u}{\partial r_j} \end{bmatrix} \tag{4.42}$$

$$K_j^{'} = \alpha(k) \frac{\partial y}{\partial u} \tag{4.43}$$

L'équation (4.41) peut être identifiée à l'équation du filtre de Kalman étendu [102] :

$$\Phi_j(k+1) = \Phi_j(k) + K\left(k\right) e \tag{4.44}$$

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

$$K\left(k\right) = \frac{P\left(k\right) H^T\left(k\right)}{H\left(k\right) P\left(k\right) H^T\left(k\right) + R\left(k\right)} \tag{4.45}$$

Avec $H\left(k\right)$ est la matrice jacobienne (la matrice d'observation de système), $P\left(k\right)$ est la matrice d'estimation de covariance de l'erreur et $R\left(k\right)$ est la matrice de covariance du bruit de processus.

En prenant $H^T\left(k\right) = \left(\Psi_j\right)^T$, $P\left(k\right) = \lambda_1$ et $R\left(k\right) = \lambda_2$, le gain $K\left(k\right)$ peut être écrit sous la forme :

$$K\left(k\right) = \frac{\lambda_1}{\left(\Psi_j\right) \lambda_1 \left(\Psi_j\right)^T + \lambda_2} \left(\Psi_j\right)^T \tag{4.46}$$

Donc l'équation (4.44) devient :

$$\Phi_j(k+1) = \Phi_j\left(k\right) + \frac{\lambda_1}{\left(\Psi_j\right) \lambda_1 \left(\Psi_j\right)^T + \lambda_2} \left(\Psi_j\right)^T e \tag{4.47}$$

Par identification entre les équations (4.41) et (4.47), nous obtenons :

$$K_j^{'} = \frac{\lambda_1}{(\Psi_j)\,\lambda_1\,(\Psi_j)^T + \lambda_2} \tag{4.48}$$

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

$$\Phi_j(k+1) = \Phi_j\,(k) + K_j^{'}\,(\Psi_j)^T\,e \tag{4.49}$$

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

$$\dot{\Phi}_j = \frac{\Phi_j\,(k+1) - \Phi_j\,(k)}{T_e} = \frac{K_j^{'}\,(\Psi_j)^T\,e}{T_e} = K_1\,(\Psi_j)^T\,e \tag{4.50}$$

Où $K_1 = \dfrac{K_j^{'}}{T_e}$.

Donc :

$$\dot{\Phi}_j = K_1\,(\Psi_j)^T\,e = (\Psi_j)^T\,e_u \tag{4.51}$$

Où $e_u = K_1 e$.

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

Ce qui implique :

$$\dot{\widetilde{\Phi}}_j = \dot{\Phi}_{jd} - \dot{\Phi}_j = -\,(\Psi_j)^T\,e_u \tag{4.52}$$

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 :

$$e_u = u_d - u = \sum_{j=1}^{4}\left((\Psi_j)\,\Phi_{jd} - (\Psi_j)\,\Phi_j\right) = \sum_{j=1}^{4}\left((\Psi_j)\,(\Phi_{jd} - \Phi_j)\right) = \sum_{j=1}^{4}\left((\Psi_j)\,\widetilde{\Phi}_j\right) \tag{4.53}$$

4.6.5 Stabilité du système de contrôle

Considérons la fonction de Lyapunov suivante [103] :

$$V_j = \frac{1}{2}\sum_{j=1}^{4}\left(\left(\widetilde{\Phi}_j\right)^T\left(\widetilde{\Phi}_j\right)\right) \tag{4.54}$$

En dérivant V_j par rapport au temps, nous aurons :

$$\dot{V}_j = \sum_{j=1}^{4}\left(\left(\dot{\widetilde{\Phi}}_j\right)^T\left(\widetilde{\Phi}_j\right)\right) \tag{4.55}$$

A partir des équations (4.52), (4.53) et (4.55), nous obtenons :

$$\dot{V}_j = \sum_{j=1}^{4}\left(\left(-\left(\Psi_j\right)^T e_u\right)^T\left(\widetilde{\Phi}_j\right)\right) = -\left(e_u\right)^T\sum_{j=1}^{4}\left(\left(\Psi_j\right)\left(\widetilde{\Phi}_j\right)\right) = -\left(e_u\right)^T\left(e_u\right) \tag{4.56}$$

Par conséquent :

$$\dot{V}_j = -\left(e_u\right)^T\left(e_u\right) \leq 0 \tag{4.57}$$

A partir de l'équation (4.57), nous obtenons $\dot{V}_j \leq 0$, par conséquent, nous pouvons conclure que le système est asymptotiquement stable au sens de Lyapunov [14, 104].

4.7 Conclusion

Dans ce chapitre, nous avons présenté les concepts généraux de la logique floue, des réseaux de neurones et des réseaux neuro-flous. Nous avons d'abord décrit les notions de base de la logique floue ainsi que la structure de la commande floue. Ensuite, nous avons abordé le concept des réseaux de neurones artificiels et leur apprentissage par l'algorithme de rétro-propagation. Par la suite, nous avons présenté les structures de base des combinaisons de la logique flou avec les réseaux de neurones, plus particulièrement la structure ANFIS, ainsi que l'algorithme d'apprentissage utilisé dans notre travail. La dernière partie du chapitre a été consacrée pour le développement des contrôleurs ANFIS proposés. Ces contrôleurs sont utilisés pour commander le système de téléopération à un degré de

liberté, décrit précédemment, suivant deux stratégies de commande (position-position et quatre canaux).

Dans le chapitre qui suit, nous allons implémenter les contrôleurs proposés sur la carte FPGA Zedboard en utilisant l'environnement Simulink de MATLAB.

Chapitre 5

Implémentation et résultats

5.1 Introduction

Dans ce chapitre, nous allons présenter les techniques utilisées pour concevoir et implémenter sur la carte Zedboard les contrôleur PID et ANFIS développés précédemment, en utilisant l'environnement Simulink de MATLAB, afin de commander notre système de téléopération. Ce dernier sera contrôlé suivant les deux architectures de commande position-position et à quatre canaux. Les deux schémas de commande (PID et ANFIS) en utilisant les deux stratégies de contrôle seront implémentés sur une carte de développement FPGA de type Zedboard de la famille Zynq-7000, en utilisant les outils Fixed-Point Tool et HDL Coder de l'environnement Simulink de MATLAB. La dernière partie du chapitre sera consacrée à la présentation des résultats expérimentaux obtenus.

5.2 Implémentation des algorithmes de contrôle

De nos jours, les FPGAs sont utilisés dans de nombreux domaines [21]. Ils permettent une vitesse de calcul élevée et un traitement de données parallèle à faible consommation d'énergie [17, 18]. Cette technologie offre des temps d'échantillonnage précis par rapport aux implémentations basées sur un microprocesseur [19]. Le FPGA est alors un circuit approprié pour effectuer un contrôle de téléopération [21] puisqu'il permet d'accélérer l'algorithme de contrôle, ce qui peut augmenter les performances du système. Cependant,

la méthodologie de conception de l'algorithme de contrôle peut affecter les performances du circuit logique généré en terme de surface occupée par la conception sur le FPGA, car certains algorithmes complexes nécessitent plusieurs ressources matérielles [17].

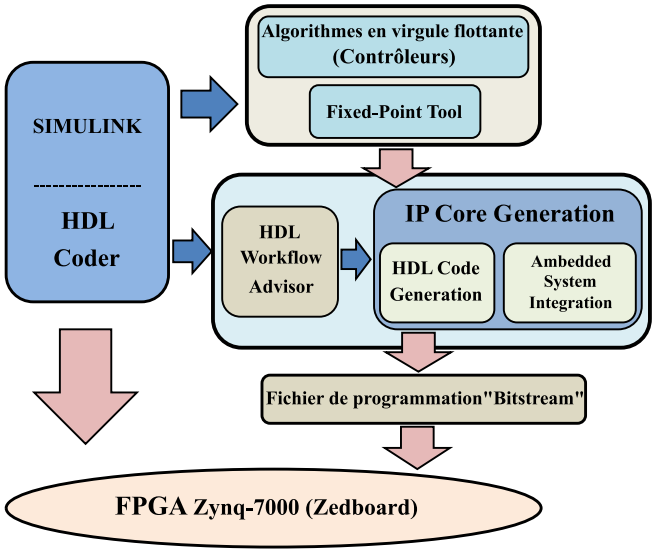


FIGURE 5.1: Méthodologie de conception et d'implémentation de l'algorithme du contrôle sur la carte FPGA Zedboard avec MATLAB-Simulink.

Les composants FPGA étant composés de ressources fixes, la longueur des mots de données doit être gérée de manière à répondre aux exigences de performances et à utiliser efficacement les ressources. Dans le cas où les algorithmes de contrôle à implémenter sont complexes, la spécification de la longueur optimale des mots de données est une tâche très complexe pouvant affecter la précision de l'algorithme lors de ses calculs [17], donc la dégradation des performances du système de téléopération. Des études sur les implémentations des algorithmes non linéaires sur FPGA pour les systèmes de téléopération tels que les modes glissants ont été présentées dans [17, 19, 24], en utilisant une méthodologie de conception basée sur LabVIEW pour le prototypage rapide. Dans ce travail de thèse, l'implémentation des algorithmes de contrôle proposés précédemment (PID et ANFIS) sur la carte FPGA Zedboard est réalisée selon une méthodologie de conception basée sur l'utilisation de l'environnement Simulink de MATLAB (Figure 5.1). Une telle méthodologie permet de réduire considérablement le temps de conception des algorithmes et d'obtenir un programme VHDL précis et optimal en terme de ressources consommées.

Le logiciel MATLAB-Simulink permet à travers l'outil Fixed-Point Tool d'automatiser la spécification du type de données en virgule fixe à partir d'un modèle Simulink en virgule flottante [23], tout en respectant les règles de l'arithmétique en virgule fixe et en optimisant les longueurs des mots binaires, par la suite HDL Coder permettra de générer le code VHDL qui sera implémenté sur la carte FPGA directement via l'environnement Simulink.

5.2.1 Conversion en virgule fixe en utilisant Fixed-Point Tool

Les contrôleurs PID et neuro-flous (ANFIS) ont été développés dans l'environnement Simulink de MATLAB, en utilisant les blocs pris en charge par HDL Coder. Ces blocs se présentent par défaut sous forme de données à virgule flottante à double précision. Par la suite, l'outil Fixed-Point Tool permet de gérer le processus de conversion des modèles (contrôleurs) synthétisés en virgule flottante en des modèles en virgule fixe (Figure 5.2).

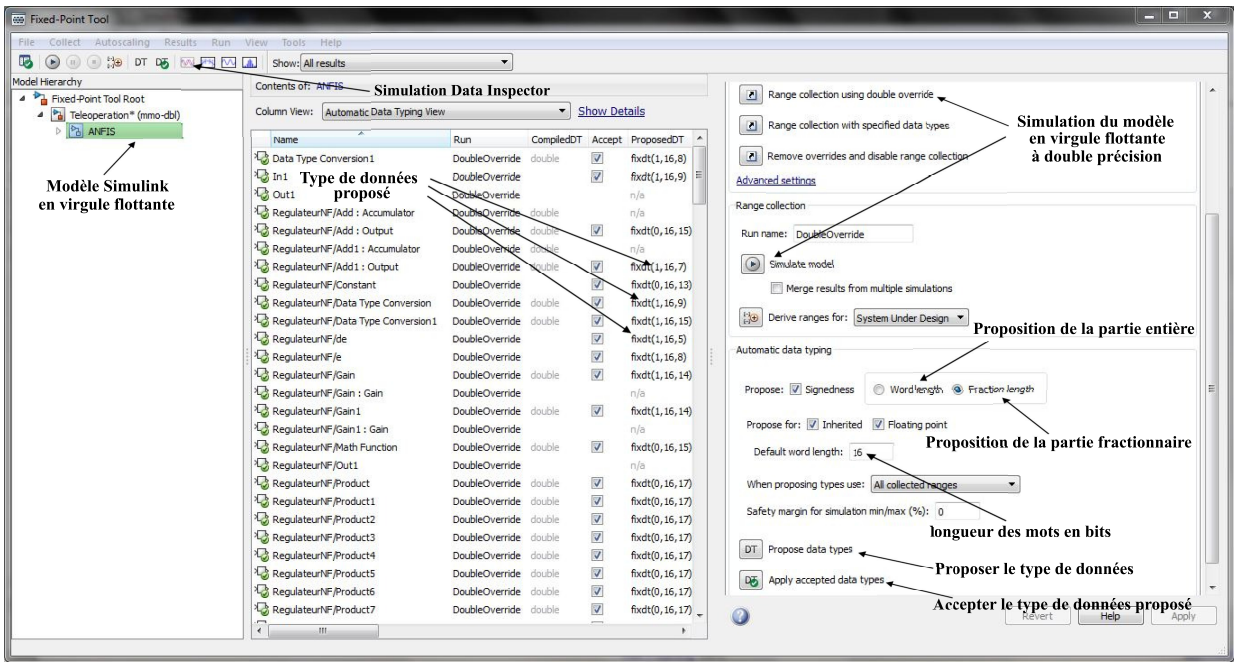


FIGURE 5.2: L'interface de Fixed-Point Tool sur Simulink.

Cet outil utilise les données de simulation pour proposer les longueurs des mots binaires ou celles des parties fractionnaires, selon le choix de l'utilisateur, en respectant les exigences de précision pour chaque opération de la conception tout en offrant à l'utilisateur la

possibilité de valider les données proposées ou les remplacer par ses propres données. Fixed-Point Tool permet également via l'outil " Simulation Data Inspector " de vérifier et de comparer les résultats obtenus en virgule fixe avec ceux obtenus avec les modèle initiaux (en virgule flottante) développés sur Simulink, avant d'être validés pour être implémentés sur FPGA.

Les contrôleurs du maître et de l'esclave (PID et ANFIS), les encodeurs et les PWMs ont été programmés avec des blocs de Simulink pris en charge par HDL Coder. Les sorties des encodeurs ont été programmées sur 16 bits, alors que les signaux de force (couple) et les entrées des PWMs sur 8 bits. Les sorties des contrôleurs ont été calculées avec l'outil Fixed-Point Tool en combinant sur 16 bits les parties entière et fractionnaire pour assurer la précision.

5.2.2 Implémentation sur la carte FPGA via HDL Coder

Les contrôleurs proposés ont été implémentés sur la carte Zedboard via l'outil HDL Coder en utilisant l'interface " Workflow Advisor " (Figures 5.3, 5.4). Cette interface dis-

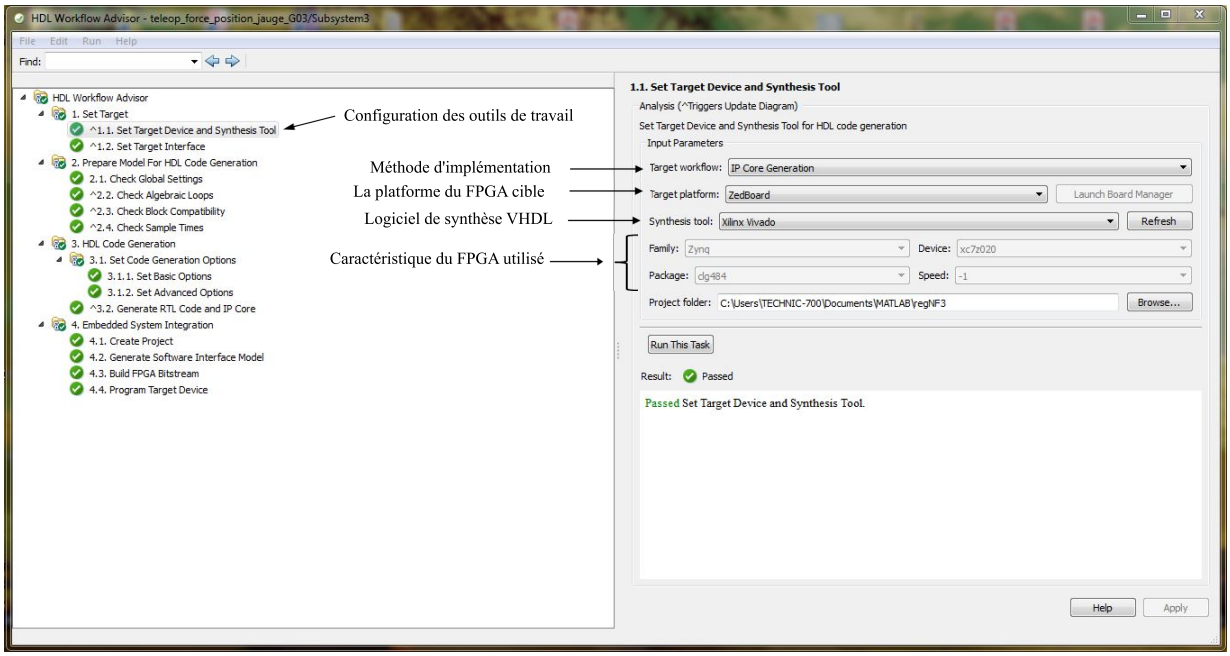


FIGURE 5.3: Configuration de la plateforme FPGA sur l'interface de HDL Coder.

pose de plusieurs fonctionnalités pour la mise en œuvre des conceptions sur FPGA telles que la méthode d'implémentation choisie, la plateforme cible, le logiciel de synthèse HDL choisi, la spécification des pins entrée-sortie, ainsi que des outils d'optimisation et de simulation. Dans notre travail, nous avons implémenté les contrôleurs via le flux de travail " IP Core Generation " par l'intermédiaire du logiciel de synthèse Vivado de Xilinx, où un bloc IP est généré à partir des modèles développés sur l'environnement Simulink.

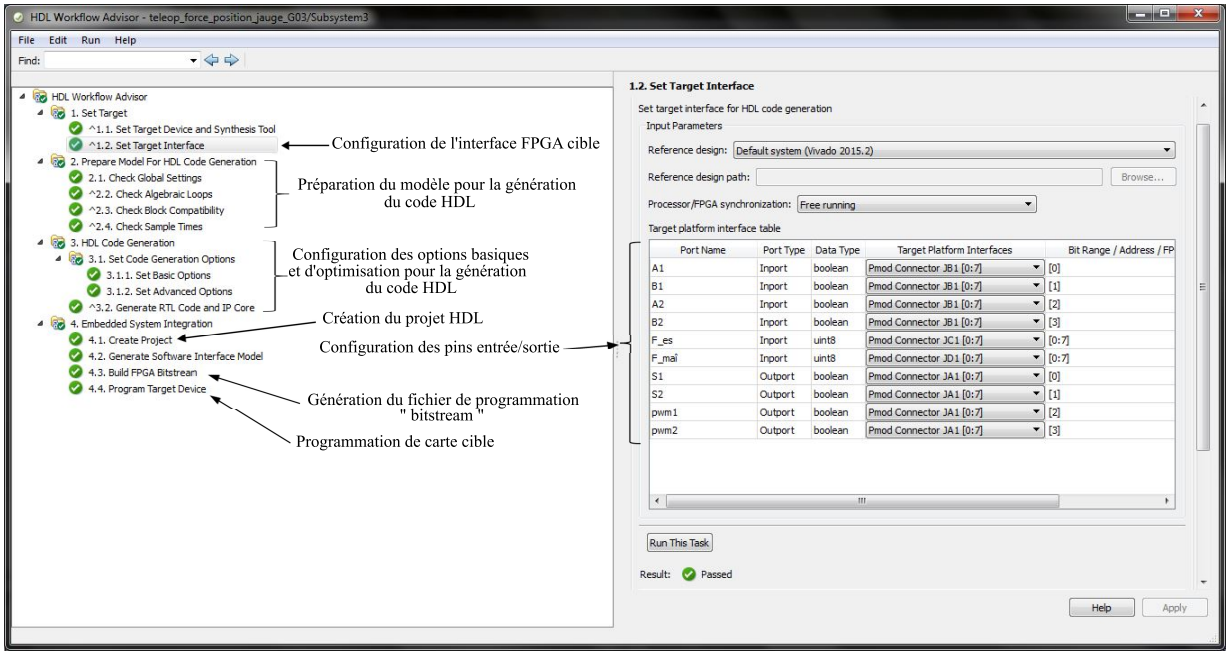


FIGURE 5.4: Implémentation du programme sur la carte FPGA Zedboard via l'interface HDL Coder.

Le processus de génération du code VHDL comporte des phases d'optimisation pour les ressources matérielles consommées ainsi que les temps de propagation des signaux, permettant ainsi d'obtenir un code VHDL optimisé et efficace. Afin de pouvoir communiquer avec la carte Zedboard via le logiciel MATLAB, une fenêtre est générée sur Simulink par l'outil " Generate Software Interface Mode ", où l'utilisateur a la possibilité de modifier les paramètres de la conception ainsi que de récupérer les différentes courbes en temps réel pendant le fonctionnement du système commandé, et cela en utilisant les deux technologies FPGA et microprocesseurs ARM simultanément. Le fichier de programmation " bitstream " est généré par l'outil " Build FPGA Bitsream " et ensuite chargé vers la carte

FPGA par la fonction " Program Target Device " en utilisant un câble de programmation JTAG ou un câble Ethernet.

5.2.3 Ressources matérielles consommées

Lors de la génération de code VHDL à partir des modèles Simulink, HDL Coder permet plusieurs types d'optimisation permettant de partager les ressources matérielles sur le FPGA afin de réduire la zone occupée par la conception et d'améliorer la fréquence d'horloge, tout en optimisant le temps d'exécution [23].
La méthodologie de conception adoptée dans ce travail assure d'une part une bonne précision de calcul, ce qui garantit de bonnes performances pour le système de téléopération, et d'autre part, une consommation optimale des ressources matérielles grâce aux techniques d'optimisation utilisées par " HDL Coder ".
Les Tableaux 5.1 et 5.2 illustrent les ressources matérielles consommées sur le FPGA lors de l'implémentation des algorithmes de contrôle PID et Neuro-flou en utilisant la stratégie de contrôle position-position, avec un taux d'échantillonnage de 1 MHz, alors que les Tableaux 5.3 et 5.4 illustrent les ressources matérielles utilisées lors de l'implémentation des algorithmes de contrôle PID et Neuro-flou en utilisant la stratégie de contrôle à quatre canaux. A partir de ces tableaux, on peut constater que les ressources matérielles consommées lors de la mise en œuvre des deux algorithmes sont considérablement optimisées en raison de la méthodologie de conception adoptée.

Ressources	Utilisation	Disponible	Utilisation (%)
FF (Flip-Flop)	726	106400	0.68
LUT (Look Up Table)	898	53200	1.69
Mémoires LUT	64	17400	0.37
I/O (Input/Output)	16	200	8.00
DSP48 (Digital Signal Processor)	10	220	4.55
BUFG (Global Buffer)	3	32	9.38
MMCM (Mixed-Mode Clock Manager)	1	4	25.00

TABLE 5.1: Ressources matérielles consommées sur le FPGA avec le contrôle position-position en utilisant les contrôleurs PID.

Ressources	Utilisation	Disponible	Utilisation (%)
FF (Flip-Flop)	1064	106400	1.00
LUT (Look Up Table)	5064	53200	9.52
Mémoires LUT	64	17400	0.37
I/O (Input/Output)	16	200	8.00
DSP48 (Digital Signal Processor)	134	220	60.91
BUFG (Global Buffer)	3	32	9.38
MMCM (Mixed-Mode Clock Manager)	1	4	25.00

TABLE 5.2: Ressources matérielles consommées sur le FPGA avec le contrôle position-position en utilisant les contrôleurs ANFIS.

Ressources	Utilisation	Disponible	Utilisation (%)
FF (Flip-Flop)	726	106400	0.68
LUT (Look Up Table)	831	53200	1.56
Mémoires LUT	64	17400	0.37
I/O (Input/Output)	32	200	16.00
DSP48 (Digital Signal Processor)	11	220	5.00
BUFG (Global Buffer)	3	32	9.38
MMCM (Mixed-Mode Clock Manager)	1	4	25.00

TABLE 5.3: Ressources matérielles consommées sur le FPGA avec le contrôle à quatre canaux en utilisant les contrôleurs PID.

Ressources	Utilisation	Disponible	Utilisation (%)
FF (Flip-Flop)	1055	106400	0.99
LUT (Look Up Table)	4780	53200	8.98
Mémoires LUT	64	17400	0.37
I/O (Input/Output)	32	200	16.00
DSP48 (Digital Signal Processor)	133	220	60.45
BUFG (Global Buffer)	3	32	9.38
MMCM (Mixed-Mode Clock Manager)	1	4	25.00

TABLE 5.4: Ressources matérielles consommées sur le FPGA avec le contrôle à quatre canaux en utilisant les contrôleurs ANFIS.

5.3 Mise en œuvre expérimentale

L'opérateur humain contrôle à distance le dispositif esclave pour suivre les mouvements de position du maître, tout en ressentant le couple de réaction de l'environnement

sur l'esclave.

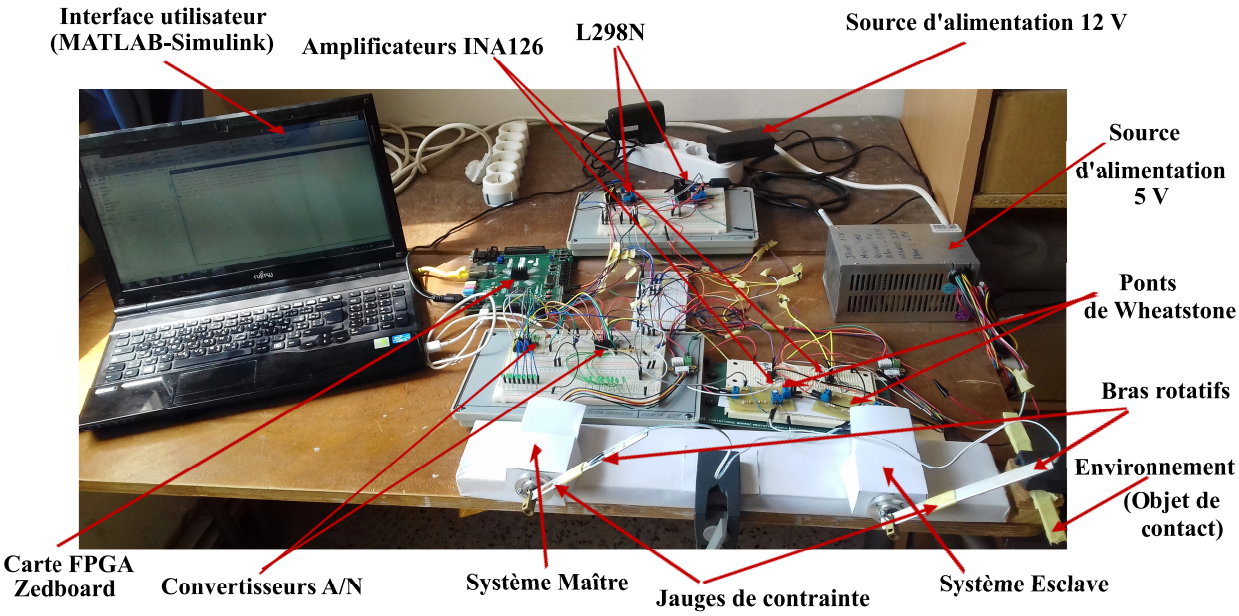


FIGURE 5.5: Banc de test expérimental à un degré de liberté.

La Figure 5.5 illustre le système de téléopération utilisé. Ce système a été commandé par une carte FPGA en utilisant deux schémas de commande (position-position et quatre canaux). Ces deux schémas de contrôle suivant deux stratégies de commande PID et neuro-floue ont été implémentés via le logiciel MATLAB-Simulink sur une carte de développement Zedboard avec un FPGA de type Zynq-7000 AP-SoC-XC7Z020-CLG484 [61], en utilisant une fréquence d'échantillonnage de 1 MHz. Les positions des robots maître (θ_m) et esclave (θ_s) sont mesurées à l'aide d'encodeurs à effet Hall intégrés dans les moteurs à courant continu, tandis que le couple d'interaction de l'esclave avec l'environnement (T_e) et le couple appliqué par l'opérateur sur le maître (T_h) sont mesurés par des jauges de contrainte. Les signaux analogiques délivrés par les ponts de Wheatstone sont amplifiés par des amplificateurs d'instrumentation de type INA 126 (Voir Annexe B), puis convertis en utilisant des convertisseurs analogique-numérique 8 bits de type ADC0804 (Voir Annexe C). La carte FPGA permet l'acquisition des signaux de couple délivrés par les convertisseurs, et des signaux A et B des encodeurs afin de calculer les positions des deux robots. Les signaux des erreurs de position et de couple sont injectés dans les contrôleurs

afin de calculer les signaux de commande PWM et identifier les sens de rotation S1 et S2 de chaque moteur. Ces derniers sont contrôlés par des pilotes de type L298N, dont chacun est composé d'un double pont H, conçus pour contrôler des moteurs à courant continu (Voir Annexe D). La Figure 5.6 représente le circuit de commande à quatre canaux en utilisant la carte FPGA.

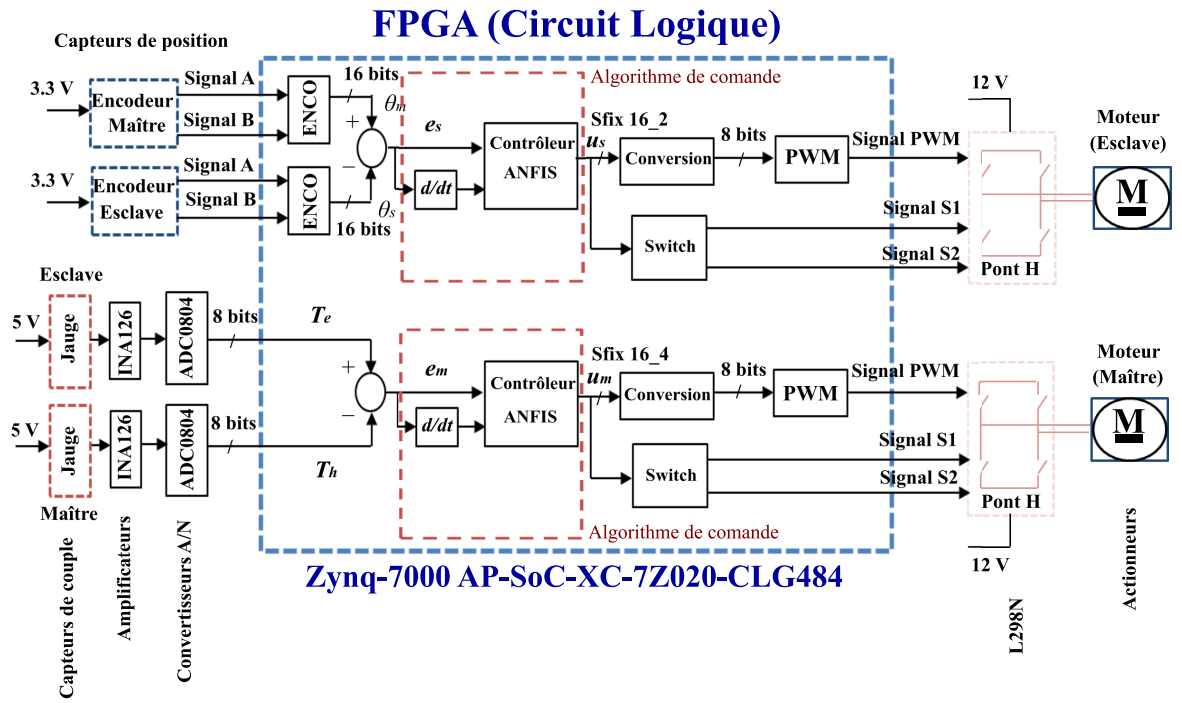


FIGURE 5.6: Schéma représentant le circuit de commande pour l'architecture à quatre canaux avec la carte FPGA.

5.4 Résultats expérimentaux

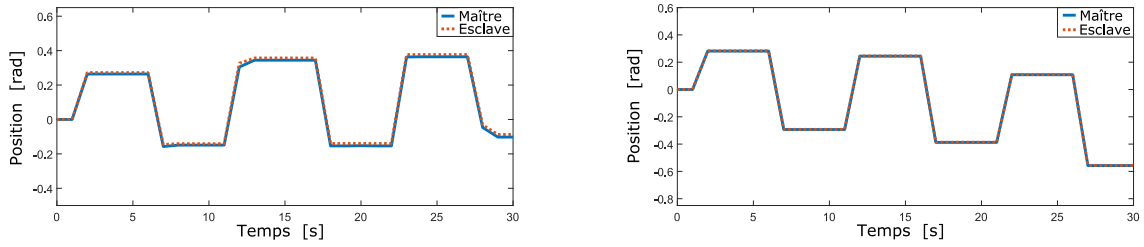
Les schémas de contrôle position-position et à quatre canaux ont été testés expérimentalement sur le système de téléopération à un degré de liberté, décrit précédemment. La structure de contrôle position-position a été testée en mouvement libre (pas de contact avec l'environnement distant), alors que la structure de commande à quatre canaux a été testée dans deux environnements distincts : Environnement souple et environnement rigide.

Les résultats expérimentaux ont été menés pour vérifier l'efficacité et les performances des

contrôleurs ANFIS proposés. Afin de confirmer la robustesse de ces derniers, nous avons comparé dans des conditions identiques les performances des contrôleurs neuro-flous à celles des contrôleurs classiques PID.

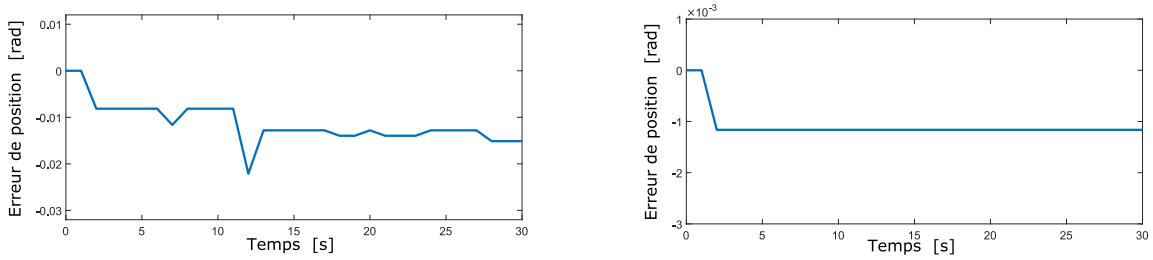
5.4.1 Contrôle position-position

Premièrement, les contrôleurs de position PID sont implémentés avec les paramètres définis dans le Tableau 3.1 (Chapitre 3). Les résultats expérimentaux montrant le suivi de position en mouvement libre en utilisant les contrôleurs PID sont illustrés dans la Figure



(a) (b)

FIGURE 5.7: Comportements du suivi de position avec : (a) Les contrôleurs PID, (b) Les contrôleurs ANFIS.



(a) (b)

FIGURE 5.8: Comportements des erreurs de position avec : (a) Les contrôleurs PID, (b) Les contrôleurs ANFIS.

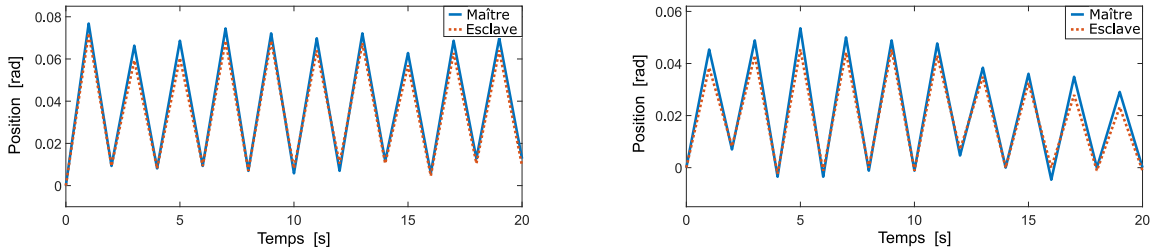
5.7(a). A partir de cette figure, nous pouvons constater que la trajectoire de la position de l'esclave suit les variations de la position du maître ; Cependant, une petite erreur de

suivi peut être remarquée, comme indiqué dans la Figure 5.8(a).
Deuxièmement, les contrôleurs neuro-flous sont implémentés. La Figure 5.7(b) représente l'évolution temporelle de la position du maître et de celle de l'esclave en utilisant les contrôleurs ANFIS. Sur cette figure, nous pouvons déduire que le robot esclave suit correctement le robot maître avec une très petite erreur, comme indiqué dans la Figure 5.8(b). A partir de ces résultats, nous pouvons constater que les contrôleurs ANFIS offrent de très bonnes performances de suivi et de robustesse par rapport aux contrôleurs conventionnels PID. Les contrôleurs neuro-flous proposés assurent une meilleure transparence car l'esclave reproduit les mouvements de position du maître avec fidélité, et cela grâce à l'algorithme d'apprentissage qui ajuste en ligne les paramètres de la conséquence du réseau neuro-flou, en utilisant le parallélisme de calcul du FPGA et sa fréquence d'échantillonnage élevée afin de converger très rapidement vers les performances désirées.

5.4.2 Contrôle à quatre canaux

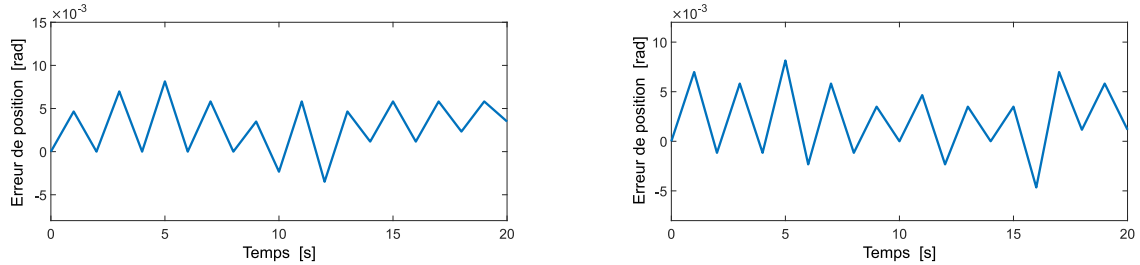
Afin de tester la robustesses des contrôleurs ANFIS proposés, nous avons introduit deux objets de contact avec des rigidités différentes. Les deux expériences suivantes ont été alors réalisées : Contact avec un environnement souple (tige en aluminium flexible) et contact avec un environnement rigide (tige en aluminium dure).
Premièrement, les contrôleurs PID avec les paramètres définis dans le Tableau 3.2 (Chapitre 3) ont été implémentés dans les deux cas cités ci-dessus.

Les résultats expérimentaux relatifs aux suivis en position et en couple dans le cas où l'esclave est en contact avec un environnement souple sont illustrés respectivement dans les Figures 5.9(a) et 5.11(a), alors que ceux relatifs aux suivis en position et en couple dans le cas où l'esclave est en contact avec un environnement rigide sont illustrés respectivement dans les Figures 5.9(b) et 5.11(b). Les Figures 5.10(a,b) et 5.12(a,b) représentent respectivement l'évolution temporelle des erreurs de position et de couple.
Ces figures montrent qu'un bon suivi de position est obtenu avec une petite erreur dans le cas où l'esclave est en contact avec un environnement souple et qu'un bon suivi de couple



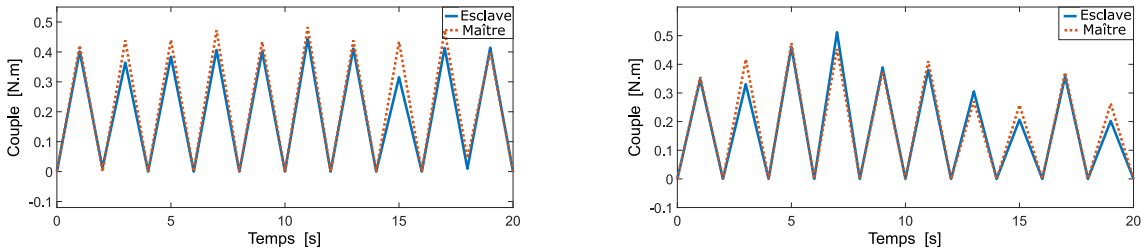
(a) (b)

FIGURE 5.9: Comportements du suivi de position avec les contrôleurs PID. (a) Contact avec un environnement souple, (b) Contact avec un environnement rigide.



(a) (b)

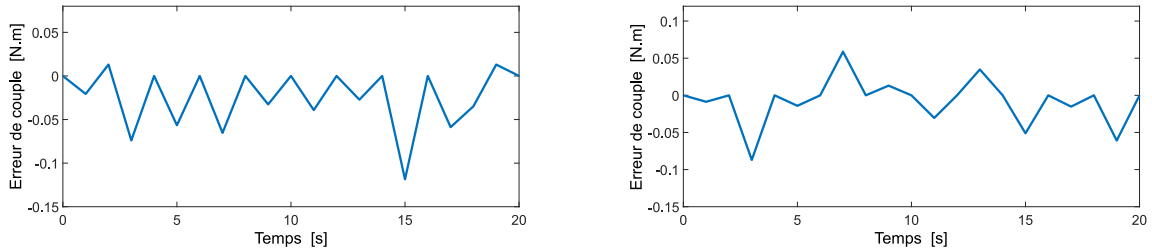
FIGURE 5.10: Comportements des erreurs de position avec les contrôleurs PID. (a) Contact avec un environnement souple, (b) Contact avec un environnement rigide.



(a) (b)

FIGURE 5.11: Comportements du suivi de couple avec les contrôleurs PID. (a) Contact avec un environnement souple, (b) Contact avec un environnement rigide.

est obtenu. Cependant, lorsque l'esclave est en contact avec un environnement rigide, une détérioration de la poursuite en position est observée, alors que les performances de



(a) (b)

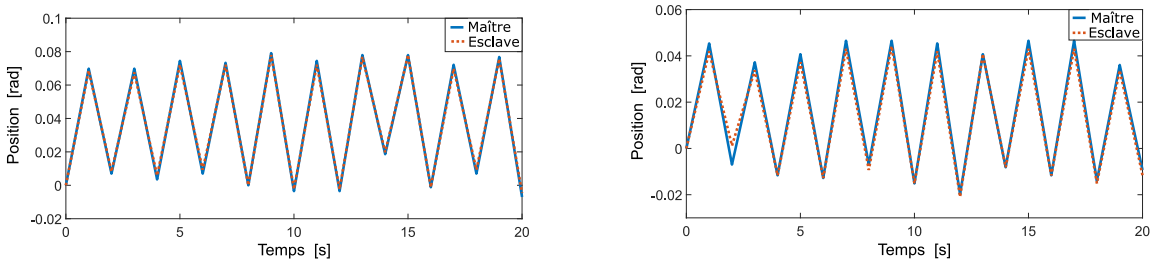
FIGURE 5.12: Comportements des erreurs de couple avec les contrôleurs PID. (a) Contact avec un environnement souple, (b) Contact avec un environnement rigide.

poursuite en couple sont acceptables. Ces résultats peuvent être expliqués en raison que les contrôleurs PID ne peuvent pas s'adapter aux variations dynamiques du maître et de l'esclave.

Deuxièmement, les contrôleurs ANFIS ont été également adoptés dans les deux cas cités précédemment.

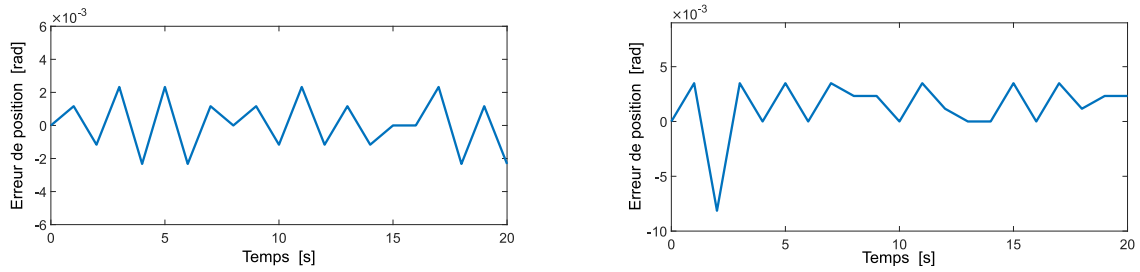
Les résultats expérimentaux montrant les performances de poursuite en position et en couple dans le cas où l'esclave est en contact avec un environnement souple sont représentés respectivement dans les Figures 5.13(a) et 5.15(a), alors que ceux montrant les performances de poursuite en position et en couple dans le cas où l'esclave est en contact avec un environnement rigide sont illustrés respectivement dans les Figures 5.13(b) et 5.15(b). Les Figures 5.14(a,b) et 5.16(a,b) représentent respectivement l'évolution temporelle des erreurs de poursuite en position et en couple. Ces figures montrent qu'un excellent suivi en position a été obtenu avec une très petite erreur dans le cas où l'esclave est en contact avec un environnement souple, ainsi qu'un bon suivi en couple a été obtenu.

Les Figures 5.13(a) et 5.15(a) montrent que les résultats de suivi en position et en couple obtenus avec les contrôleurs ANFIS lors d'un contact avec un objet souple sont meilleurs que ceux obtenus avec les contrôleurs PID (Figure 5.9(a), 5.11(a)). Lorsque l'esclave est en contact avec un objet rigide, nous pouvons constater que les résultats du suivi en position obtenus avec les contrôleurs ANFIS sont également meilleurs comparativement aux résultats obtenus par les contrôleurs PID dans le même cas, car l'esclave suit les positions



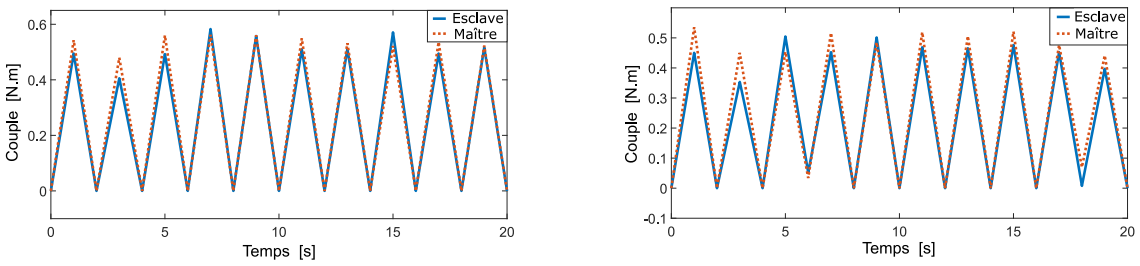
(a) (b)

FIGURE 5.13: Comportements du suivi de position avec les contrôleurs ANFIS. (a) Contact avec un environnement souple, (b) Contact avec un environnement rigide.



(a) (b)

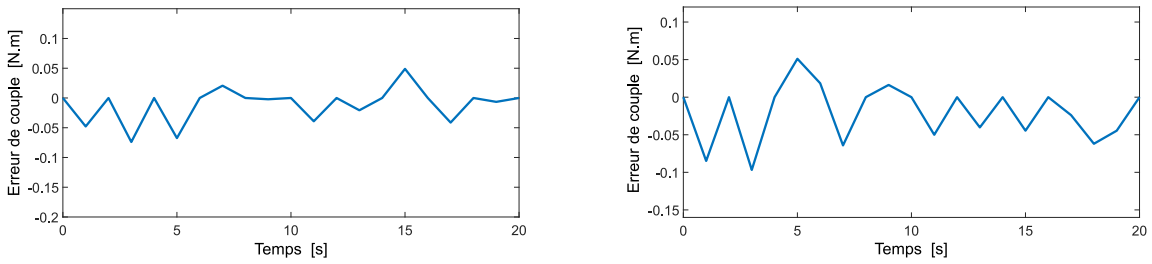
FIGURE 5.14: Comportements des erreurs de position avec les contrôleurs ANFIS. (a) Contact avec un environnement souple, (b) Contact avec un environnement rigide.



(a) (b)

FIGURE 5.15: Comportements du suivi de couple avec les contrôleurs ANFIS. (a) Contact avec un environnement souple, (b) Contact avec un environnement rigide.

du maître avec précision, traduite par une très petite erreur, et le suivi en couple était bon, comme le montrent les Figures 5.13(b) et 5.15(b).



(a) (b)

FIGURE 5.16: Comportements des erreurs de couple avec les contrôleurs ANFIS. (a) Contact avec un environnement souple, (b) Contact avec un environnement rigide.

A partir de tous ces résultats graphiques, on peut constater que les contrôleurs ANFIS proposés dans ce travail offrent une meilleure transparence par rapport aux contrôleurs PID classiques, où l'esclave suit les mouvements du maître avec fidélité tout en permettant à l'opérateur de ressentir la force de réaction de l'environnement sur l'esclave. Les contrôleurs ANFIS garantissent une stabilité de contact avec l'environnement, de plus, les efforts ressentis par l'opérateur via le robot maître sont très proches de la raideur de l'environnement, et l'opérateur humain a alors l'impression d'être en contact direct avec l'environnement distant qui change de rigidité.

Ces résultats peuvent être expliqués par les deux contrôleurs ANFIS de position et de couple, qui appliquent un algorithme de contrôle intelligent qui compense les non-linéarités du système de téléopération, et qui s'adapte rapidement aux variations dynamiques du système lorsque l'esclave est en contact avec l'environnement inconnu, et cela en ajustant rapidement les paramètres de la conséquence du réseau neuro-flou en temps réel grâce à la puissance de calcul du FPGA et sa fréquence d'échantillonnage élevée. De plus, la méthodologie de conception des contrôleurs, utilisée dans ce travail, a permis d'obtenir de bons résultats en termes de précision et de stabilité. L'environnement MATLAB-Simulink, avec ses fonctionnalités Fixed-Point Tool et HDL Coder, a permis de concevoir un algorithme robuste avec une grande précision tout en optimisant le temps d'exécution, les fréquences de la conception et les ressources matérielles consommées par le FPGA.

5.5 Conclusion

Dans ce chapitre, nous avons présenté la méthodologie de conception et d'implémentation des contrôleurs proposés sur la carte FPGA Zedboard, ainsi que les résultats expérimentaux obtenus lors de la commande du système de téléopération réalisé, selon chaque stratégie de commande (position-position et quatre canaux).
Dans la première partie, nous avons expliqué la méthodologie adoptée à l'aide de MATLAB-Simulink pour concevoir et implémenter sur FPGA les contrôleurs proposés, en utilisant l'outil Fixed-Point Tool pour la conversion du type de données à virgule flottante en virgule fixe, et l'outil HDL Coder pour charger les programmes sur la carte Zedboard. Une telle méthodologie nous a permis de réduire le temps de conception des algorithmes et d'obtenir des contrôleurs précis et robustes, avec un code VHDL optimal en terme de ressources matérielles consommées sur le FPGA. La dernière partie du chapitre a été consacrée à la présentation des résultats expérimentaux obtenus par les commandes PID et neuro-floue suivant les deux stratégies de commande utilisées. Ces résultats montrent l'efficacité de la méthode proposée comparativement aux méthodes conventionnelles (PID).

Conclusion générale

Le travail présenté dans le cadre de cette thèse a pour objectif d'aborder les techniques de l'intelligence artificielle, plus particulièrement les techniques neuro-floues appropriées à la commande des systèmes de téléopération. Ces méthodes constituent une alternative aux méthodes classiques, et sont utilisées pour surmonter les contraintes liées à la commande bilatérale des systèmes de téléopération afin de répondre aux exigences de performance et de robustesse. Les systèmes maître et esclave étant non linéaires et complexes, avec une dynamique qui parfois ne peut pas être disponible en raison du retard de communication entre les deux postes, les contrôleurs doivent être adaptatifs afin d'assurer la stabilité et la transparence. Dans ce travail, nous avons proposé un contrôleur neuro-flou adaptatif de type ANFIS pour commander un système de téléopération à un degré de liberté, en utilisant une carte FPGA. L'utilisation d'un tel circuit programmable a permis de minimiser le temps de calcul des algorithmes proposés, et d'éliminer le retard de communication entre le maître et l'esclave en transmettant le plus fidèlement possible les informations de position et de force échangées entre les deux postes.

Le présent travail a été réparti sur cinq parties.

La première partie a été consacrée à la présentation des modèles mathématiques décrivant les systèmes de téléopération couramment utilisés dans la littérature, ainsi que les trois architectures de contrôle de base les plus répondues en pratique et leurs critères de performances.

Dans la deuxième partie, nous avons présenté l'architecture des circuits FPGA du constructeur Xilinx, particulièrement ceux de la famille Zynq-7000 utilisés dans ce travail. Nous avons donné également les particularités de la carte de développement Zedboard que nous

avons utilisée, ainsi que les différents logiciels et outils permettant sa programmation.
Dans la troisième partie, nous avons présenté le système de téléopération à un degré de liberté, réalisé dans le cadre de ce travail de thèse. Nous avons expliqué aussi le fonctionnement des capteurs utilisés (encodeurs et jauges de contrainte), ainsi que les techniques utilisées sur l'environnement MATLAB-Simulink pour l'identification des modèles des actionneurs (System Identification Toolbox) et la synthèse des contrôleurs PID (PID Tuner).
La quatrième partie de ce travail a été dédiée à la présentation des techniques de l'intelligence artificielle. Nous avons d'abord donné quelques notions de base sur la logique floue, les réseaux de neurones ainsi que les réseaux neuro-flous, par la suite, nous avons développé deux régulateurs neuro-flous adaptatifs de type ANFIS pour les système maître et esclave : Un régulateur de position pour chaque système en utilisant l'architecture de contrôle position-position, et un régulateur de couple pour le maître et un autre de position pour l'esclave en utilisant l'architecture de contrôle à quatre canaux. La combinaison des avantages de la logique floue avec ceux des réseaux de neurones en un seul réseau a permis d'obtenir un contrôleur très performant, en tirant profit de la capacité d'apprentissage des réseaux de neurones pour l'adaptation des paramètres de la conséquence du réseau neuro-flou en ligne, en utilisant un algorithme d'apprentissage basé sur le filtre de Kalman étendu, ce qui a permis de converger vers les performances désirées en un temps très court grâce la capacité de calcul du FPGA et sa fréquence d'échantillonnage élevée.
Dans la dernière partie, nous avons présenté les techniques utilisées pour la mise en œuvre des contrôleurs proposés sur FPGA. La conception et l'implémentation des algorithmes de contrôle sur la carte Zedboard ont été effectuées en utilisant l'environnement Simulink de MATLAB. La préparation pour l'implémentation des modèles développés sur Simulink a été faite en utilisant l'outil Fixed-Point Tool, où les modèles ont été convertis en virgule fixe avant d'être implémentés sur la carte FPGA via l'outil HDL Coder. Une telle méthodologie de conception et d'implémentation a permis de réduire considérablement le temps de conception, et d'obtenir un algorithme rapide et précis avec un code VHDL optimal en terme de ressources matérielles consommées sur le FPGA. Les deux expériences suivantes ont été effectuées pour les deux stratégies de contrôle : Contact avec deux environnements

différents (souple et rigide) dans le cas du contrôle à quatre canaux, et fonctionnement en mouvement libre (pas de contact) dans le cas du contrôle position-position. Les résultats expérimentaux ont démontré l'efficacité des contrôleurs neuro-flous proposés. En effet, ces derniers ont donné de meilleurs résultats en termes de poursuite, de stabilité et de robustesse face aux variations dynamiques du système, par rapport aux contrôleurs classiques PID. Ceci s'explique par le fait que les régulateurs ANFIS permettent de s'adapter aux variations dynamiques des systèmes maître et esclave ainsi qu'aux changements de rigidité de l'environnement distant, et cela grâce d'une part, à l'utilisation d'un apprentissage en ligne pour le réseau neuro-flou, et d'autre part à l'utilisation du FPGA qui accélère le temps de calcul pour converger rapidement vers les performances désirées.

Comme perspectives à ce travail, il serait intéressant de prendre en considération des retards de communication constants et variables entre le maître et l'esclave, en utilisant des contrôleurs adaptatifs basés sur des approches prédictives (réseaux de neurones prédicteurs, estimateurs par le filtre de Kalman étendu), pour améliorer la transparence et la stabilité.

Bibliographie

Bibliographie

[1] A. N. Rodriguez, *ASSET : Une Architecture Générale pour la Télérobotique* , Thèse de Doctorat en informatique à l'université Paul Sabatier de Toulouse (sciences), Janvier 2003.

[2] B. Siciliano, O. Khatib, *Springer Handbook of Robotics* , Springer, 2008.

[3] R. C. Goertz, *Fundamentals of General-Purpose Remote Manipulators*, Nucleonics, pp. 36-45, November 1952.

[4] R. Goertz, *Mechanical Master-Slave Manipulator*, Nucleonics, tome 12, No. 11, 1954.

[5] Z. Li, Y. Xia, C-Y. Su, *Intelligent Networked Teleoperation Control*, Springer-Verlag Berlin Heidelberg, 2015.

[6] Y. Liv, N. Chopra, *Control of Semi-Autonomous Teleoperation System With Time Delays*, Automatica, Vol. 49, pp. 1553-1567, 2013.

[7] A. Haddadi, K. Hashtrudi-Zaad, *Bounded-Impedance Absolute Stability of Bilateral Teleoperation Control Systems*, IEEE Transactions on Haptics, Vol. 3, No. 1, pp. 15-27, 2010.

[8] F. Hashemzadeh, I. Hassanzadeh, M. Tavakoli, *Teleoperation in the Presence of Varying Time Delays and Sandwich Linearity in Actuators*, Automatica, Vol. 49, No. 9, pp. 2813-2821, September 2013.

[9] H. Gao, J. Liu, Y. Li, K. Hong, Y. Zhang, *Dual-Layer Fuzzy Control Architecture for the CAS Rover Arm*, International Journal of Control, Automation, and Systems, Vol. 13, No. 5, pp. 1262-1271, 2015.

[10] U. Farooq, M. El-Hawary, MU. Asad, *Fuzzy Model Based Bilateral Control Design of Nonlinear Tele-operation System Using Method of State Convergence*, IEEE Access, Vol. 4, pp. 4119-4135, 2016.

[11] Y. Yang, C. Ge, H. Wang, X. Li, C. Huaa, *Adaptive Neural Network Based Prescribed Performance Control for Teleoperation System Under Input Saturation*, Journal of the Franklin Institute, Vol. 352, pp. 1850-1866, 2015.

[12] W. Po-Ngaen, *Adaptive Four-Channel Neuro-Fuzzy Control of a Master-Slave Robot*, International Journal of Advanced Robotic Systems, Vol. 10, pp. 1-8, 2013.

[13] H. Shao, K. Nonami, T. Wojtara, R. Yuasa, S. Amano, D. Waterman, *Neuro-Fuzzy Position Control of Demining Teleoperation System Based on RNN Modeling*, Robotics and Computer Integrated Manufacturing, Vol. 22, pp. 25-32, 2006.

[14] R. Mellah, S. Guermah, R. Toumi, *Adaptive Control of Bilateral Teleoperation System with Compensatory Neural-Fuzzy Controllers*, International Journal of Control, Automation and Systems, Vol. 15, pp. 1-11, 2017.

[15] R. Mellah, R. Toumi, *Compensatory Neuro-Fuzzy Control of Bilateral Teleoperation System*, Proc. of 20th International Conference on Methods and Models in Automation and Robotic ; 2015 Aug 24-27 ; Miedzyzdroje, Poland, pp. 382-387, 2015.

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

[17] A. Hace, M. Franc, *FPGA Implementation of Sliding-Mode-Control Algorithm for Scaled Bilateral Teleoperation*, IEEE Transactions on Industrial Informatics, Vol. 9, No. 3, pp. 1291-1300, 2013.

[18] H. Tanaka, K. Ohnishi, H. Nishi, T. Kawai, Y. Morikawa, S. Ozawa, T. Furukawa, *Implementation of Bilateral Control System Based on Acceleration Control Using FPGA for Multi-DOF Haptic Endoscopic Surgery Robot*, IEEE transaction on Industrial Electronics, Vol. 56, No. 3, pp. 618-627, 2009.

[19] A. Hace, M. Franc, *Enhanced Pseudo-Sensorless Bilateral Teleoperation by PLL α β-Tracker and FPGA*, Automatika, Vol. 55, No. 3, pp. 265-275, 2014.

[20] E. Ishii, H. Nishi, K. Ohnishi, *Improvement of Performances in Bilateral Teleoperation by Using FPGA*, IEEE Transactions on Industrial Electronics, pp. 317–322, 2017.

[21] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A.Tisan, M. W. Naouar, *FPGAs in Industrial Control Applications*, IEEE Transactions on Industrial Informatics, 2011.

[22] A. Finnerty, H. Ratigner, *Reduce Power and Cost by Converting from Floating Point to Fixed Point*, Xilinx All Programmable, WP491, (v. 1.0), pp. 1-14, 2017.

[23] *HDL Coder User's Guide*, MathWorks. MATLAB and SIMULINK, r2018a, 2018.

[24] M. Franc, A. Hace, *A Study on the FPGA Implementation of the Bilateral Control Algorithm Towards Haptic Teleoperation*, Journal for Control, Measurement, Electronics, Computing and Communications, Vol. 54, No. 1, pp. 49-61, 2013.

[25] Y. Lia, Y. Yin, S. Zhang, J.Dong, R. Johansson, *Composite Adaptive Control for Bilateral Teleoperation Systems without Persistency of Excitation*, Journal of The Franklin Institute, pp. 1-21, April 2018.

[26] X. Liu, R. Tao, M. Tavakoli, *Adaptive Control of Uncertain Nonlinear Teleoperation Systems*, Mechatronics, Vol. 24, pp. 66-78, 2014.

[27] S. Liu, X. Zhang, W. Zheng, B. Yang, *Adaptive Control for Time-Delay Teleoperation Systems with Uncertain Dynamics*, Journal of Physics : Conf. Series 887, pp. 1-8, 2017.

[28] X. Liu, M. Tavakoli, *Bilateral Adaptive Control of Nonlinear Teleoperation Systems with Uncertain Dynamics and Dead-Zone*, Journal of Dynamic Systems, Measurement, and Control, Vol. 140, pp. 1-10, 2018.

[29] M. Motaharifar, A. Bataleblu, H.D. Taghirad, *Adaptive Control of Dual User Teleoperation with Time Delay and Dynamic Uncertainty*, 24th Iranian Conference on Electrical Engineering (ICEE), pp. 1318–1323, 2016.

[30] C. Pacchierotti, *Cutaneous Haptic Feedback in Robotic Teleoperation*, Springer Series on Touch and Haptic Systems, 2015.

[31] P. Letier, *Bras Exosquelette Haptique : Conception et Contrôle*, thèse de Doctorat en sciences appliquées à l'université Libre de Bruxelles, Juillet 2010.

[32] X. Liu, M. Tavakoli, *Inverse Dynamics-Based Adaptive Control of Nonlinear Bilateral Teleoperation Systems*, IEEE International Conference on Robotics and Automation, Shanghai, China, May 9-13, pp. 1323-1328, 2011.

[33] M. Q. Le, M. T. Pham, M. Tavakoli, R. Moreau, *Sliding Mode Control of a Pneumatic Haptic Teleoperation System with On/Off Solenoïd Valves*, IEEE International Conference on Robotics and Automation, Shanghai, China, May 9-13, 2011.

[34] A. Alfi, M. Farrokhi, *A Simple Structure for Bilateral Transparent Teleoperation Systems with Time Delay*, Journal of Dynamic Systems, Measurement, and Control, Vol. 37, pp. 1-9, 2008.

[35] P. F. Hokayem, M. W. Spong, *Bilateral Teleoperation : An Historical Survey*, Automatica, Vol. 42, No. 12, pp. 2035-2057, June 2006.

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

[37] 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.

[38] Y. Yokokohji, T. Yoshikawa, *Bilateral Control of Master-Slave Manipulators for Ideal Kinesthetic Coupling-Formulation and Experiment*, IEEE Transactions on Robotics and Automation, Vol. 10, No. 5, pp. 605–620, October 1994.

[39] H. Arioui, *Commande de Dispositifs à Retour Haptique en Présence de Retards de Transmission*, Thèse de Doctorat en Robotique à l'université d'Evry Val d'Essonne, Décembre 2002.

[40] B. Hannaford, *A Design Framework for Teleoperators with Kinesthetic Feedback*, IEEE Transactions on Robotics and Automation, Vol. 5, No. 4, pp. 426–434, August 1989.

[41] I. Aliaga, A. Rubio, E. Sanchez, *Experimental Quantitative Comparison of Different Control Architectures for Master-Slave Teleoperation*, IEEE Transactions on Control Systems Technology, Vol. 12, No. 1, pp. 2–11, January 2004.

[42] G. A. V. Christiansson, *Hard Master, Soft Slave Haptic Teleoperation*, Doctorate Thesis at Chalmers University of Technology, sweden, october 2007.

[43] L. Barbé, *Télé-opération avec Retour d'Efforts pour les Interventions Percutanées*, Automatique / Robotique, Université Louis Pasteur - Strasbourg I, 2007.

[44] D. A. Lawrence, *Stability and Transparency in Bilateral Téléopération*, IEEE Transactions on Robotics and Automation, Vol. 9, No. 5, pp.624–637, October 1993.

[45] F. Courreges, *Détermination d'une Architecture de Téléopération Bilatérale en Présence de Retards*, Laboratoire Vision et Robotique.

[46] R. J. Anderson, M. W. Spong, *Bilateral Control of Teleoperators With Time Delay*, Proceeding of the 27th Conference on Decision and Control, Austin, Texas, December 1988.

[47] J-y. Lee, S. Payandeh, *Haptic Teleoperation Systems Signal Processing Perspective*, Springer International Publishing Switzerland, 2015.

[48] G. Niemyer, J-J. E. Slotine, *Stable Adaptative Teleoperation*, IEEE Journal of Oceanic Engineering, Vol. 16, No. 1, January 1991.

[49] E. Monmasson, LM. N. Cirstea, *FPGA Design Methodology for Industrial Control Systems-A Review*, IEEE Transactions on Industrial Electronics, 2007.

[50] A. Blanchardon, *Synthèse d'Architectures de Circuits FPGA Tolérants aux défauts*, Technologies Émergentes. Université Pierre et Marie Curie - Paris VI, 2015.

[51] U. Farooq, Z. Marrakchi, H. Mehrez, *Tree-Based Heterogeneous FPGA Architectures Application Specific Exploration and Optimization*, Springer Science and Business Media, 2012.

[52] M. W. Nouar, *Commande Numérique à Base de Composants FPGA d'une Machine Synchrone, Algorithme de Contrôle du Courant*, École Nationale d'Ingénieurs de Tunis et Université de Cergy Pontoise, 2007.

[53] I. Kuon, R. Tessier, J. Rose, *FPGA Architecture : Survey and Challenges*, Foundations and Trends in Electronic Design Automation, Vol. 2, No. 2, pp. 135–253, 2008.

[54] J. Rose, A. El Gamal, A. Sangiovani-Vincentelli, *Architecture of Field-Programmable Gate Arrays*, Procedings of the IEEE, 1993.

[55] N. Marques, *Méthodologie et Architecture Adaptative pour le Placement Efficace de Tâches Matérielles de Tailles Variables sur des Partitions Reconfigurables*, Thèse de Doctorat à l'Université de Lorraine, 2012.

[56] D. Chen, J. Cong, P. Pan, *FPGA Design Automation : A Survey*, Foundations and Trends in Electronic Design Automation, Vol. 1, No. 3, pp. 195–330, 2006.

[57] A. Palchaudhuri, R.S. Chakraborty, *High Performance Integer Arithmetic Circuit Design on FPGA*, Springer Series in Advanced Microelectronics 51, 2016.

[58] T. Ahmed, P.D. Kundarewich, J. H. Anderson. B. L. Taylor, R. Aggarwal, *Architecture-Specific Packing for Virtex-5 FPGAs*, Proceedings of the ACM/SIGDA 16th International Symposium on Field Programmable Gate Arrays, FPGA 2008, Monterey, California, USA, February 24-26, 2008.

[59] H. Parvez, H. Mehrez, *Application-Specific Mesh-based Heterogeneous FPGA Architectures*, Springer Science and Business Media, 2011.

[60] *Corporate Responsibility Report FY2014* , Xilinx All Programmable, 2014.

[61] *ZedBoard, (Zynq Evaluation and Development)*, Hardware User's Guide. Version 2.2, pp. 1-37, 2014.

[62] *ZedBoard Getting Started Guide*, Version 7.0, Avnet, Inc, 2017.

[63] *ZedBoard Booting and Configuration Guide*, Avnet Electronics Marketing, 2012.

[64] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, R. W. Stewart, *The Zynq Book, Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000*, All Programmable SoC, Department of Electronic and Electrical Engineering University of Strathclyde Glasgow, Scotland, UK, 2014.

[65] *Zynq-7000 All Programmable SoCs Product Tables and Product Selection Guide*, Xilinx Inc, 2015.

[66] *HDL Coder User's Guide*, MathWorks. MATLAB and SIMULINK, r2017b, 2017.

[67] J. DETREY, *Arithmétiques Réelles sur FPGA : Virgule Fixe, Virgule Flottante et Système Logarithmique*, École Normale Supérieure de Lyon, 2007.

[68] N. Ghatte, S. Patil, D. Bhoir, *Floating Point Engine Using VHDL*, International Journal of Engineering Trends and Technology (IJETT), Vol. 8, No. 4, pp. 198-203, 2014.

[69] A. Arockia Bazil Raj, *FPGA-Based Embedded System Developer's Guide*, CRC Press, 2018.

[70] *HDL Synthesis for FPGAs Design Guide*, Xilinx Inc, 1995.

[71] S. Churiwala, *Designing with Xilinx FPGAs Using Vivado*, Springer, 2017.

[72] E. O. Hwang, *Digital Logic and Microprocessor Design with Interfacing*, Cengage Learning, 2016.

[73] *Vivado Design Suite User Guide Getting Started*, Xilinx Inc, 2018.

[74] R. Woods, J. McAllister, G. Lightbody, Y. Yi, *FPGA-based Implementation of Signal Processing Systems*, Wiley-Blackwell, 2018.

[75] *MATLAB & SIMULINK R2018a : HDL Coder Getting Started Guide*, The MathWorks Inc, 2018.

[76] Y. Altman, *Accelerating MATLAB Performance 1001 tips to Speed Up MATLAB Programs*, Chapman and Hall/CRC, 2015.

[77] T. Hallböök, B. Månsson, R.A Nilsén, *A Strain-Gauge Pletysmograph with Electrical Calibration*, Scandinavian Journal of Clinical and Laboratory Investigation, Vol. 25, No. 4, pp. 413-418, 1970.

[78] P. Borne, G. Dauphin-Tanguy, J. P. Richard, F. Rotella, I. Zambettakis *Automatique : Modélisation et Identification des Processus, tome 2*, Editions Technip, 1992.

[79] L. Ljung, *MATLAB & SIMULINK R2019b : A System Identification Toolbox, Getting Started Guide*, MathWorks, 2019.

[80] *MATLAB & SIMULINK R2019b : Control System Toolbox, Getting Started Guide*, MathWorks, 2019.

[81] R. Mellah, R. Toumi, *Control Bilateral Teleoperation by Compensatory ANFIS*, Advanced Mechatronics Solutions, Vol. 393, pp. 167-172, 2016.

[82] W.E. Kelly, R. Challoo, R. Mclauchlan, S.I. Omar, *Neuro-fuzzy Control of a Robotic Arm*, Proceedings of the Artificial Neural Networks In Engineering Conference, pp. 837-842, 1996.

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

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

[85] H. T. Nguyen, C. L. Walker, E. A. Walker *A First Course in Fuzzy Logic, Fourth Edition* , Chapman and Hall/CRC, 2018.

[86] J. M.Keller, D. Liu, D. B. Fogel, *Fundamentals of Computational Intelligence : Neural Networks, Fuzzy Systems, and Evolutionary Computation*, Wiley-Blackwell, 2016.

[87] H.T. Nguyen, N.R. Prasad, C.L. Walker, E.A. Walker, *A First Course in Fuzzy and Neural Control*, Chapman and Hall/CRC, 2002.

[88] R. Mellah, *Contribution à la Commande Adaptative Neuro-Floue. Application à la Robotique*, Thèse de Doctorat à l'Université des Sciences et de la Technologie Houari Boumediene, Alger, Mai 2006.

[89] O. Castillo, P. Melin, J. Kacprzyk, *Fuzzy Logic Augmentation of Neural and Optimization Algorithms : Theoretical Aspects and Real Applications*, Springer, 2018.

[90] R. Fuller, *Introduction to Neuro-Fuzzy Systems*, Springer, 2000.

[91] P. Stavroulakis, *Neuro-Fuzzy and Fuzzy-Neural Applications in Telecommunications*, Springer, 2004.

[92] M. B. Gorzalczany, *Computational Intelligence Systems and Applications, Neuro-Fuzzy and Fuzzy Neural Synergisms*, Physica-Verlag Heidelberg, 2002.

[93] I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, S. F. D. R. Alves, *Artificial Neural Networks, A Practical Course*, Springer, 2017.

[94] S. Chakravertys, S. Mall, *Artificial Neural Networks for Engineers and Scientists, Solving Ordinary Differential Equations*, CRC Press, 2017.

[95] T. Kuremoto, Y. Yamano, L. Feng, K. Kobayashi, M. Obayashi, *A Neuro-fuzzy Network with Reinforcement Learning Algorithms for Swarm Learning*, Future Wireless Networks and Information Systems, Springer-Verlag Berlin Heidelberg, pp. 101-108, 2012.

[96] M. A. Shoorehdeli, M. Teshnehlab, A. K. Sedigh, M. A. Khanesar, *Identification Using ANFIS with Intelligent Hybrid Stable Learning Algorithm Approaches and Stability Analysis of Training Methods*, Applied Soft Computing, ELSEVIER, Vol. 9, No. 2, pp.833-850, 2009.

[97] JSR. Jang, CT. Sun, E. Mizutani, *Neuro-Fuzzy and Soft Computing : A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, 1997.

[98] M. Franc, A. Hace, *FPGA Implementation of the Bilateral Control Algorithm for a High Performance Haptic Teleoperation*, The 12th IEEE International Workshop on Advanced Motion Control ; 2012 Mar 25-27 ; Sarajevo, Bosnia and Herzegovina, pp. 1-6, 2012.

[99] H. Khati, H. Talem, R. Mellah, A. Bilek *Neuro-Fuzzy Control of Bilateral Teleoperation System Using FPGA*, Iranian Journal of Fuzzy Systems, pp. 1-16, 2019.

[100] H. Khati, R. Mellah, H. Talem, *Neuro-Fuzzy Control of a Position-Position Teleoperation System Using FPGA*, Proc. of 24th International Conference on Methods and Models in Automation and Robotic ; 2019, Aug 26-29 ; Miedzyzdroje, Poland, pp. 64-69, 2019.

[101] S. Ruder, *An Overview of Gradient Descent Optimization Algorithms*, Insight Centre for Data Analytics, NUI Galway Aylien Ltd, Dublin, pp. 1-14, 2016.

[102] S. Hayki, *Kalman Filtering and Neural Networks*, John Wiley and Sons, Inc, 2001.

[103] L. Peng, PY. Woo, *Neural-Fuzzy Control System for Robotic Manipulators*, IEEE Control Systems Magazing, pp. 53-63, 2002.

[104] FL. Lewis, CT. Abdallah, DM. Dawson, *Control of Robot Manipulators*, Macmillan, New York, 1993.

Annexes

Annexe A : Jauge de contrainte

Une jauge de contrainte est un capteur qui répond à la dilatation ou à la contraction d'un matériau, ou à sa déformation. Elle consiste en une longue pièce en métal mince qui se replie sur elle-même ou en zigzags sur le capteur (Figure 5.17). Lorsque le matériau se dilate ou se contracte, la longue pièce de métal mince s'allonge ou se raccourcit, ce qui modifie la résistance du métal. La tension de sortie du capteur correspond au changement de résistance.

FIGURE 5.17: Jauge de contrainte.

Une jauge de contrainte est un métal ou un semi-conducteur dont la résistance change de façon marquée lors de la déformation. Cette dernière est généralement considérée comme une mesure de la contrainte, et donc de la force appliquée à une structure. La résistance d'un échantillon de matériau de longueur L et de section A est donnée par :

$$R = \rho \frac{L}{A} \tag{5.1}$$

Un changement de longueur ou de zone sous contrainte produit un changement de résistance d'un élément particulier.

Si la résistance d'un élément particulier est R_0 sans contrainte, alors le facteur de jauge de contrainte G est donné par :

$$G = \frac{\Delta R}{R_0} \frac{L}{\Delta L} \tag{5.2}$$

avec $\frac{\Delta R}{R_0}$ le changement relatif de la résistance et $\frac{L}{\Delta L}$ le changement relatif de la longueur.

Les jauges de contrainte ne transportent généralement qu'un faible courant (15 - 100 mA) pour éviter les modifications auto-échauffantes de la résistance et les erreurs de dilatation thermique.

Une jauge de contrainte est généralement montée en un 1/4 de pont de Wheatstone, qui est utilisé pour mesurer les changements de résistance du dispositif.

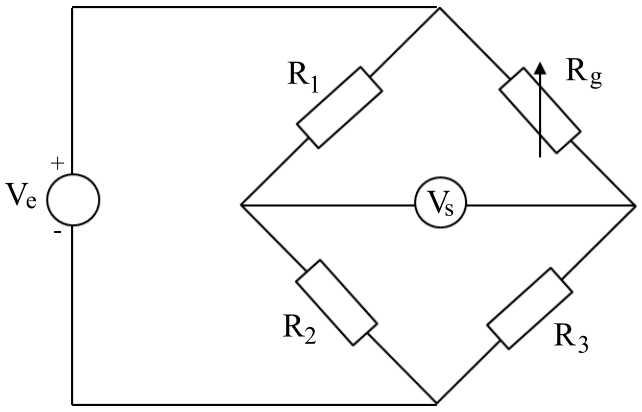


FIGURE 5.18: Jauge de contrainte montée sur un pont de Wheatstone.

Dans le cas où la jauge de contrainte est placée en une configuration en un 1/4 de pont (Figure 5.18), La tension de sortie V_s est donnée par :

$$V_s = \left[\frac{R_3}{R_3 + R_g} - \frac{R_2}{R_1 + R_2} \right] V_e \tag{5.3}$$

V_e est la tension d'alimentation du Pont de Wheatstone, et R_g est la résistance de la jauge.

A l'équilibre, $R_1 R_3 = R_2 R_g$ et $V_s = 0$.

Lorsque la résistance R_g change, la tension de sortie du pont V_s change également.

La variation de R_g dépend du facteur de jauge G et de la déformation ε telle que :

$$\Delta R = \varepsilon R_g G \tag{5.4}$$

En prenant $R_1 = R_2$ et $R_3 = R_g$, la tension de sortie V_s pour un changement ΔR dans R_g est donnée par :

$$V_s = \frac{1}{2}\left[\frac{1}{1+\varepsilon G/2} - 1\right] V_e \tag{5.5}$$

$$V_s = -\frac{\varepsilon G}{4}\frac{1}{1+\varepsilon G/2} V_e \tag{5.6}$$

Avant de mesurer la contrainte, le pont doit être être à l'équilibre (la tension de sortie $V_s = 0$). Lorsqu'une contrainte est appliquée, la tension de sortie est mesurée.
Les circuits de jauge de contrainte utilisent une tension d'excitation de 5 à 10 V, qui doit être extrêmement stable. Le signal de sortie se situe dans la plage des mV.

Annexe B : Amplificateur INA126

Le circuit INA126 est un amplificateur d'instrumentation de précision conçu pour l'ac-

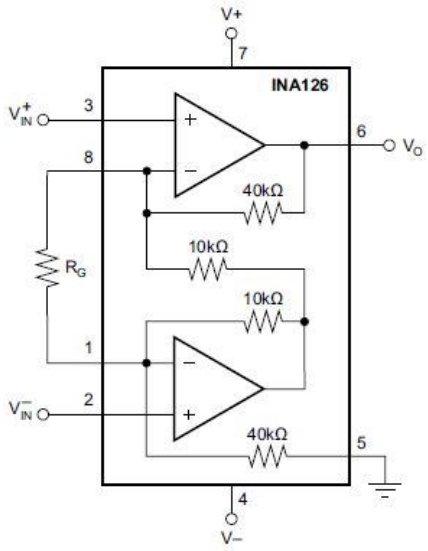


FIGURE 5.19: Schéma simplifié du INA126.

quisition de signaux différentiels précis et à faible bruit (Figure 5.19). Sa conception avec

deux amplificateurs opérationnels offre d'excellentes performances avec un faible courant de repos (175 μA / canal). Combiné à une large plage de tensions de fonctionnement de ± 1,35 V à ± 18 V, l' INAx126 est idéal pour l'instrumentation portable et les systèmes d'acquisition de données.

La tension de sortie V_0 est :

$$V_0 = (V_{IN}^{+} - V_{IN}^{-})G \tag{5.7}$$

G est le gain d'amplification, défini par :

$$G = 5 + \frac{80K\Omega}{R_G} \tag{5.8}$$

Le gain peut être réglé entre 5 et 10 000 avec une seule résistance externe. Les circuits d'entrée ajustés au laser fournissent une tension de décalage faible (maximum 250 μV), deux faibles tensions de dérive (maximum 3 μV /^{o} C) et une excellente réjection en mode commun.

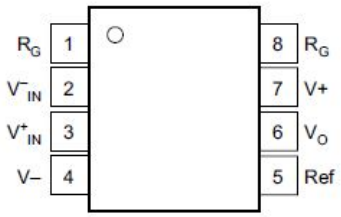


FIGURE 5.20: Composant INA126.

Le Tableau 5.5 présente les fonctions des pins du composant INA126 (Figure 5.20).

Pin	E/S	Description
R_G	-	Pin de réglage de gain
V_{IN}^{-}	E	Entrée négative
V_{IN}^{+}	E	Entrée positive
$V-$	-	Alimentation négative
Ref	E	Entrée de référence
V_O	S	Sortie
$V+$	-	Alimentation positive

TABLE 5.5: Fonctions des pins du INA126.

Annexe C : Convertisseur analogique/numérique 8 bits (ADC0804)

Le circuit ADC0804 est un convertisseur à approximations successives à 8 bits utilisant une échelle potentiométrique différentielle (Figure 5.21). Ce convertisseur est conçu

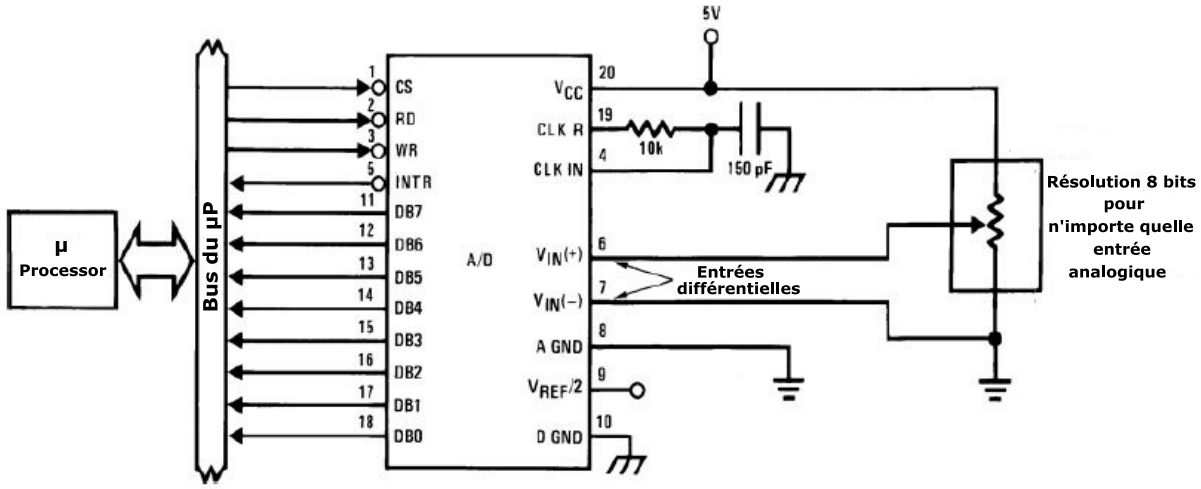


FIGURE 5.21: Schéma simplifié du ADC0804.

pour fonctionner avec un bus de contrôle avec des verrous de sortie à trois états pilotant directement le bus de données. Le ADC0804 apparait comme un emplacement de mémoire ou d'un port d'entrée / sortie sur le microprocesseur et aucune logique d'interface n'est nécessaire. Les entrées de tension analogiques différentielles permettent d'augmenter la réjection en mode commun et de décaler la valeur de tension d'entrée analogique nulle. De plus, l'entrée de référence de tension peut être ajustée pour permettre le codage de toute plage de tension analogique plus petite à la résolution complète de 8 bits.
Le Tableau 5.6 présente les fonctions des pins du composant ADC0804 (Figure 5.22).

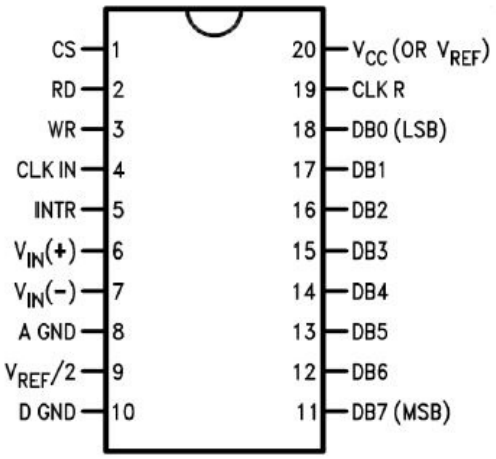


FIGURE 5.22: Composant ADC0804.

Pin	E/S	Description
CS	E	Entrée de sélection
RD	E	Lecture
WR	E	Écriture
CLK IN	E	Entrée d'horloge externe
INTR	S	Interruption
$V_{IN}(+)$	E	Entrée analogique différentielle positive
V_{IN}(-)	E	Entrée analogique différentielle négative
A GND	E	Pin de masse analogique
Vref/2	E	Entrée de tension de référence pour l'ajustement
D GND	E	Pin de masse numérique
DB7 (MSB)	S	Bit de données 7
DB6	S	Bit de données 6
Db5	S	Bit de données 5
Db4	S	Bit de données 4
Db3	S	Bit de données 3
Db2	S	Bit de données 2
Db1	S	Bit de données 1
Db0 (LSB)	S	Bit de données 0
CLK R	E	Broche d'entrée de la résistance RC pour l'horloge interne
Vcc (Vref)	E	Tension d'alimentation + 5V

TABLE 5.6: Fonctions des pins du ADC0804.

Annexe D : Circuit L298N

Le module L298N est un circuit électronique basé sur le circuit intégré L298. Ce dernier dispose d'un double pont H complet à haute tension et à haute intensité, conçu

pour piloter des charges inductives telles que des relais, des moteurs à courant continu et des moteurs pas à pas. Deux entrées d'activation sont fournies pour activer ou désactiver l'appareil indépendamment des signaux d'entrée. Les émetteurs des transistors inférieurs de chaque pont sont connectés ensemble et le terminal externe correspondant peut être utilisé pour la connexion d'une résistance de détection externe. Une entrée d'alimentation supplémentaire est fournie afin que la logique fonctionne à une tension inférieure.

La Figure 5.23 illustre le schéma du circuit électronique du L298N.

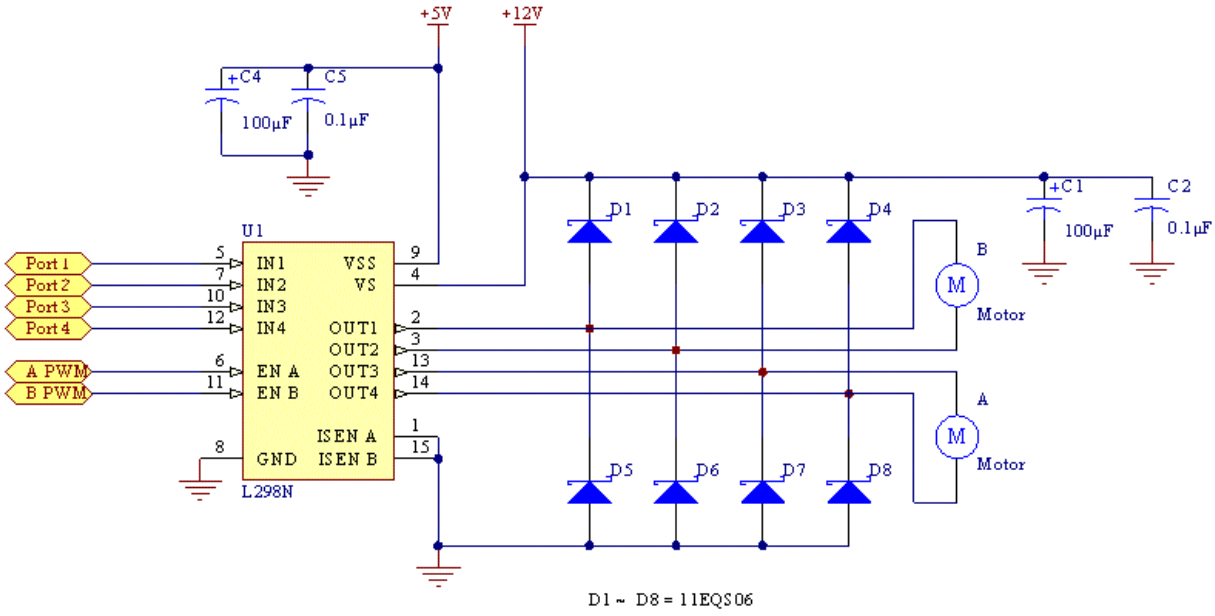


FIGURE 5.23: Schéma du circuit électronique du L298N.

Caractéristiques

- Tension de fonctionnement 4V à 46V.
- Courant total (DC) jusqu'à 4 A.
- Basse tension de saturation.
- Protection contre la surchauffe.
- Régulateur embarqué 5V à faible chute de tension, accessible par l'utilisateur.
- Diodes de protection.
- Alimentation logique : 5V.
- Puissance maximale : 25W.

- Deux voyants de direction du moteur.
- Température de stockage : -25 à +135.

La Figure 5.24 présente le module L298N.

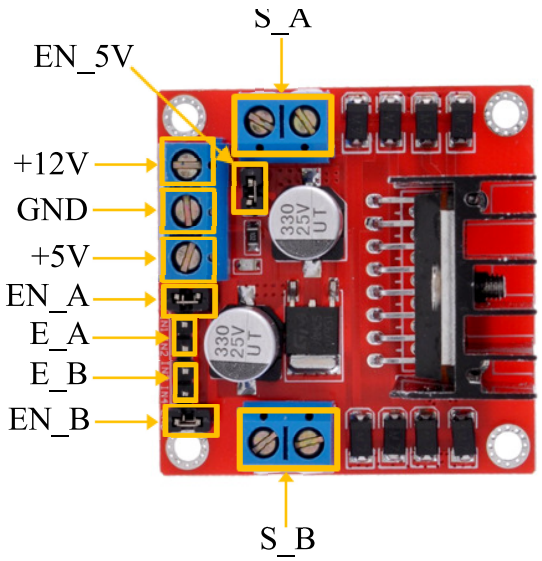


FIGURE 5.24: Module L298N.

Le Tableau 5.7 présente les fonctions des pins du circuit L298N.

Nom de la pin	Description
S_A	Moteur A
S_B	Moteur B
+12V	Entrée d'alimentation 12 V
GND	Terre
+5V	Sortie 5V
E_A	2 Entrées logiques pour le moteur A
A_B	2 Entrées logiques pour le moteur B
EN_A	Contrôle du moteur A
EN_B	Contrôle du moteur B
EN_5V	Activation 5V

TABLE 5.7: Fonctions des pins du L298N.

Résumé

Résumé : Dans ce travail de thèse, nous avons élaboré un nouveau schéma de commande bilatérale basé sur FPGA pour un système de téléopération à un degré de liberté. Un contrôleur neuro-flou adaptatif est développé pour les systèmes maître et esclave afin de contrôler la position et le couple en fonction de chaque stratégie de commande adoptée (position-position ou quatre canaux). L'apprentissage du réseau neuro-flou est effectué en ligne, en ajustant les paramètres de la conséquence des règles floues à l'aide d'un algorithme d'apprentissage basé sur les méthodes de la descente du gradient et du filtre de Kalman étendu, et cela en tirant profit du parallélisme de calcul du FPGA et sa fréquence d'échantillonnage élevée. Les contrôleurs proposés sont développés sur l'environnement Simulink de MATLAB et implémentés en utilisant les outils " Fixed-Point Tool " et " HDL Coder ". Une telle méthodologie de conception nous a permis d'obtenir un algorithme précis avec un code VHDL optimal en terme de ressources matérielles consommées, tout en réduisant le temps de conception de l'algorithme. Les résultats expérimentaux obtenus ont démontré l'efficacité des contrôleurs proposés comparant aux contrôleurs classiques PID.

Mots-clés : Téléopération, FPGA, HDL Coder, ANFIS, Commande neuro-floue, Filtre de Kalman étendu, Fixed-Point Tool.

Abstract : In this thesis work, we have developed a new FPGA-based bilateral control scheme for a teleoperation system with one degree of freedom. An adaptive neuro-fuzzy controller is developed for the master and the slave systems in order to control position and torque according to each adopted control strategy (position-position or four-channel). The learning of the neuro-fuzzy network is carried out online, by adjusting the consequence parameters of the fuzzy rules using a learning algorithm based on the methods of the gradient descent and the extended Kalman filter, and this by taking advantage of the FPGA's parallel computing and its high sampling frequency. The proposed controllers are developed on the Simulink environment of MATLAB and implemented using "Fixed-Point Tool" and "HDL Coder". A such design methodology has resulted in an accurate algorithm with optimal VHDL code in terms of consumed hardware resources while reducing the design time of the algorithm. The experimental results have demonstrated the efficiency of the proposed controllers compared to conventional controllers PID.

Keywords : Teleoperation, FPGA, HDL Coder, ANFIS, Neuro-fuzzy control, Extended Kalman filter, Fixed-Point Tool.