Iranian Journal of Fuzzy Systems

Volume **, Number *, (****), pp. **

University of Sistan and Baluchestan

Neuro-fuzzy Control of Bilateral Teleoperation System Using FPGA

H. Khati[1], H. Talem[2], R. Mellah[3] and A. Bilek[4]

[1,2,3] *Design and Drive of Production Systems Laboratory, Faculty of Electrical and Computing Engineering, University Mouloud Mammeri of Tizi-Ouzou, Tizi-Ouzou, Algeria*
[4] *Laboratory of Mechanical, Structure and Energetic, Faculty of Engineering Construction, University Mouloud Mammeri of Tizi-Ouzou, Tizi-Ouzou, Algeria*

hoc.khati@gmail.com, talemhand2015@gmail.com, mellah.rab@gmail.com, alibilek2000@yahoo.fr

Abstract

This paper presents an adaptive neuro-fuzzy controller ANFIS (Adaptive neuro-fuzzy inference system) for a bilateral teleoperation system based on FPGA (Field Programmable Gate Array). The proposed controller combines the learning capabilities of neural networks with the inference capabilities of fuzzy logic, to adapt with dynamic variations in master and slave robots and to guarantee good practical robustness against the disturbances, by adjusting neuro-fuzzy network output parameters in a short time, thanks to the computing power of FPGA and its high sampling frequency. The design methodology adopted to design the control algorithm aims to minimize the hardware resources used by the FPGA in order to optimize the execution and design times, and this by using the Fixed-Point Tool and HDL Coder features of MATLAB-Simulink. The proposed controllers were experimentally validated on a teleoperation system comprising a pair of one degree of freedom. The experimental results clearly show that the proposed ANFIS control algorithm significantly outperformed the conventional control methods (PID).

Keywords: ANFIS, Fixed-Point Tool, FPGA, HDL Coder, Neuro-fuzzy, Teleoperation.

1 Introduction

The manipulation of dangerous products was at the origin of teleoperation [12]. A teleoperation system allows an operator to perform a remote task [28] while moving it away from the work environment and the machines he manipulates. Teleoperation thus eliminates risks associated with hazardous works such as the manipulation of toxic substances and space exploration [33]. A teleoperation system consists of a master robot, a slave robot, a controller, a human operator, a communication channel and the remote environment [3, 41]. The master device is in contact with the human operator while the slave device, which reproduces the movements of the operator on the master robot, interacts with the remote environment [11]. Its operation is essentially based on the exchange of position and force informations between the master, which is controlled by the operator, and the slave which performs the task [22]. If this information is transmitted only from the master station to the slave one, the teleoperation system is called unilateral. If the reaction force of the environment on the slave is also returned to the master, the teleoperation system is said to be bilateral [17]. This feedback allows the human operator to evaluate the situation of the robot to manipulate [7]. In a bilateral teleoperation, it is desired that the system be stable and transparent [1] so that the slave reproduces the movements of the master with fidelity, and that the operator feels directly in contact with the distant environment. The basic architectures of teleoperation systems, with two or four channels, were presented in [17].

In order to achieve a high-performance bilateral teleoperation, the control algorithms must be stable, robust and fast [8]. In terms of transparency, the four-channel control architecture of Lawrence is the most efficient because it assumes a perfect knowledge of the master and the slave impedances [27], However, master and slave devices represent nonlinear and complex systems, with a dynamic that may not be available due to the uncertainties of the model [26, 28] since the

Corresponding Author: H. Khati
Received: **; Revised: **; Accepted: **.

slave often interacts with unknown environments [24]. Moreover, when the positions and the contact forces of the slave with the remote environment are simultaneously controlled, it is difficult to obtain good tracking performance [9] and to guarantee the contact stability of the slave with the environment. Moreover, in terms of hardware implementation, in order to achieve good performances for a bilateral teleoperation, it is necessary to ensure a high control rate [11] especially when the control algorithms are complex. Therefore, this paper proposes an adaptive control that can adapt to the dynamic variations in master and slave and compensate for dynamic uncertainties, using the FPGA, which allows high computing power and parallel data processing [11, 39], and ensures short sampling periods which increase the performances of speed, stability, and robustness of the teleoperation system.

Artificial neural networks and fuzzy logic controllers have known rapid growth in robotics and industrial applications. Fuzzy logic, which is a model-free approach, can be used to control a multi-inputs, multi-outputs nonlinear system, using fuzzy rules based on human reasoning, defined by different membership functions [29]. This control strategy can provide good performance; however, there are many difficulties to determinate the membership functions for each input-output [30]. The use of neural networks is then an adequate solution. They allow adjusting automatically the parameters of the inputs and outputs of the fuzzy system [36], and this by using the computing power of neural systems such as learning ability, through a nonlinear optimization algorithm such as back propagation [30, 31]. These techniques were used in teleoperation, the author in [4] proposed a bilateral state convergence controller for a nonlinear teleoperation system approximated by a Takagi-Sugeno model. In [40], an adaptive control based on neuron networks was adopted considering teleoperation with dynamic uncertainties and external disturbances. The combination of fuzzy logic and artificial neural networks makes it possible to take advantage of the inference capabilities of fuzzy logic with the learning ability of neural networks [21]. The use of neuro-fuzzy networks for the bilateral control of teleoperation systems is then adequate since these systems have a non-linear dynamic and uncertainties due to transmission delays and contact with the environment. Several studies have been conducted in the literature in this domain. The use of neuro-fuzzy techniques in the control of teleoperation systems was presented in [29, 30, 31, 36, 38], the purpose of these works was to improve the stability and performance of force and position tracking by compensating for uncertainties about the dynamics of the master-slave system. Authors have also proposed adaptive controllers for teleoperation systems with unknown dynamics [28] or considering systems with dynamic and kinematic uncertainties [25] and communication delays [24, 33]. The author in [26] proposed an adaptive controller for the master robot with dynamic uncertainties and another for the slave with dynamic uncertainties and a dead zone. The study of optimal control has also been developed in [3], where a solution to the reduced-order differential equations of the teleoperation system has been proposed. This paper proposes an adaptive neuro-fuzzy control (ANFIS) for a bilateral teleoperation system that requires neither knowledge of the dynamics of the master and the slave, nor those of the human operator and the environment. ANFIS controllers that are a modeless approach are used to adapt to changes in system dynamics when the slave is in contact with a remote environment by adjusting the parameters of the consequences of the neuro-fuzzy network in a very short time with a learning algorithm based on the extended Kalman filter, and this by taking the advantage of the FPGA parallel computation and its high sampling frequency [8], which can accelerate the ANFIS control algorithm to converge towards the desired performances.

Nowadays, FPGAs are used in many areas [32]. They allow high computing speed and parallel data processing with low power consumption [11, 39]. FPGAs provide accurate sampling times compared to implementations based on microprocessor [10], moreover, it can run multiple processes at the same time with a high operating frequency, contrary to ordinary processors that can only run one process at a time, with a lower frequency [18], the FPGA is then a suitable circuit to perform a control of teleoperation [32] since it allows to accelerate the control algorithm, which can increase system performance due to its high sampling rate. However, the design methodology of the control algorithm can affect the performances of the generated logic circuit in terms of the area occupied by the design on the FPGA, because some complex algorithms require multiple hardware resources [11]. Hardware resource consumption is more optimal in fixed-point type designs than floating-point type designs [5]; however, the optimal specification of the binary words lengths remains a complex task which can affect precision during the control algorithm calculations [11]. Further studies on FPGA implementations of non-linear algorithms to teleoperation systems such as sliding modes were presented in [7, 10, 11], using a LabVIEW-based design methodology for rapid prototyping. In this paper, an FPGA implementation of the ANFIS algorithm on a bilateral teleoperation system was studied, using a MATLAB-Simulink based design methodology. The MATLAB-Simulink environment allows through its functionality Fixed-Point Tool to automate the specification of the fixed-point data type from a floating-point Simulink model [16], by respecting the rules of the fixed-point arithmetic, and optimizing the binary words lengths, then, HDL Coder allows generating the optimized VHDL code in terms of hardware resources and execution time [16].

This paper proposes an implementation of neuro-fuzzy control ANFIS for a teleoperation system with force feedback using a Zynq7000 FPGA development board. ANFIS Controllers have been developed in the MATLAB-Simulink environment. The design methodology chosen in this work allows through the Simulink functionalities to obtain a highly optimized VHDL code in FPGA in terms of hardware resources, design time and execution time [16]. In addition, this design methodology provides very short control periods; therefore, a high-performance teleoperation is obtained. The implementation has been validated by a master-slave device.

The structure of the paper is organized as follows: In Section 2, the adopted control architecture is presented. Section 3 gives the mathematical formalism of the proposed ANFIS controller, and the control stability. The control method of the teleoperation system based on FPGA through the implementation of two controllers ANFIS and PID is explained in section 4. Section 5 shows the realization of our teleoperation system. The efficiency of the proposed controller is justified by experimental results in Section 6. Finally, section 7 concludes this paper.

2 Force-position control

The force-position control method relies on the exchange of position and force signals simultaneously between the master site and the slave site. It is based on measuring the interaction between the slave and the environment by a force sensor [22]. In this structure, the master is controlled in force while the slave is controlled in position, where θ_m and θ_s are the positions of the master and the slave respectively, T_e is the interaction torque of the slave with the environment and T_h is the torque produced by the user on the master. u_m and u_s are the torques delivered by the actuators, to respectively create the force feedback on the user and to control the movement of the slave.

In this work, we will control the master in force and the slave in position using two controllers ANFIS (Figure 1).

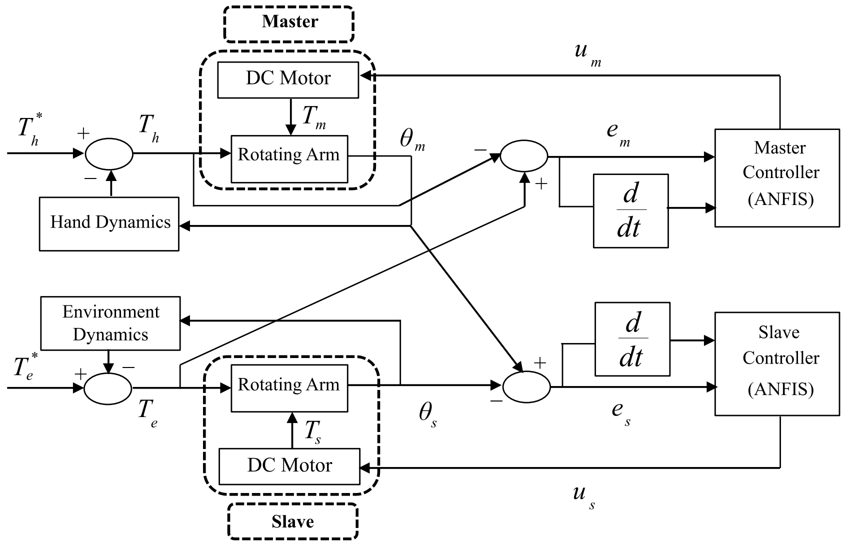


Figure 1: Bilateral control architecture.

3 Adaptive neural fuzzy controller

The combination of neural networks and fuzzy logic, in which fuzzy reasoning is realized in a neural network, allows constructing more efficient neuro-fuzzy controllers, by combining the capacities of artificial neural networks with the inference capacity of fuzzy logic [21]. The reason for representing a fuzzy system in the form of a neural network is to use the learning ability of neural networks to improve performance, such as the adaptation of fuzzy systems. Among the most popular neuro-fuzzy systems, there is the adaptive neuro-fuzzy inference system algorithm (ANFIS), which adjusts the parameters of the premises and the consequences of the neuro-fuzzy network using a hybrid learning algorithm [19, 20, 34].

3.1 Adaptive neuro-fuzzy inference system (ANFIS)

The ANFIS structure was proposed by Jang [19]. It consists of adapting fuzzy rules to condition changes. This structure combines the advantages of fuzzy reasoning with those of neural networks in a single network. As a result, neuro-fuzzy systems can automatically optimize and adjust membership functions [34] using a learning algorithm. In this work, we consider a neuro-fuzzy network with a first-order fuzzy inference model of Sugeno, including two inputs x_1 and x_2, and a single output f (Figure 2). In this subsection, the ANFIS controller is developed for the master system (Torque), and it is the same development for the slave system (Position) .

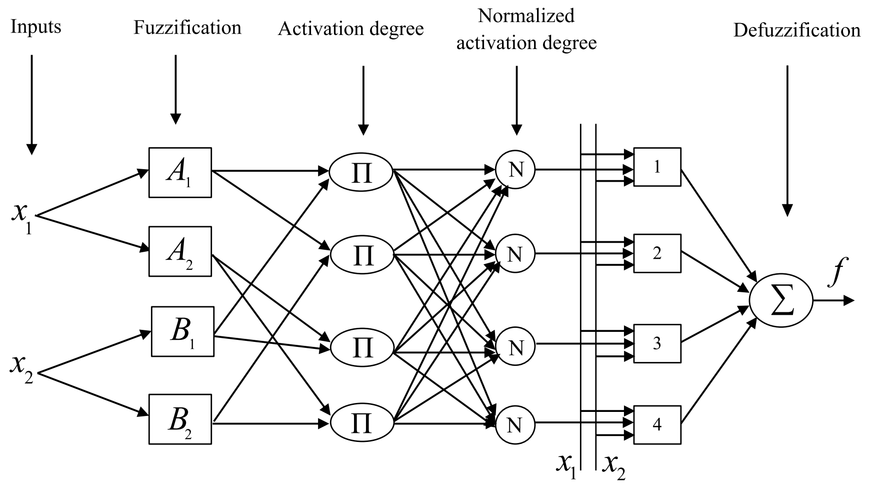


Figure 2: ANFIS controller architecture [20, 34].

Fuzzy rules are defined as follows:

$$\text{Rule}\quad 1: \text{ if } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } B_1 \text{ then } f_1 = p_1x_1 + q_1x_2 + r_1 \tag{1}$$

$$\text{Rule}\quad 2: \text{ if } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } B_2 \text{ then } f_2 = p_2x_1 + q_2x_2 + r_2 \tag{2}$$

$$\text{Rule}\quad 3: \text{ if } x_1 \text{ is } A_2 \text{ and } x_2 \text{ is } B_1 \text{ then } f_3 = p_3x_1 + q_3x_2 + r_3 \tag{3}$$

$$\text{Rule}\quad 4: \text{ if } x_1 \text{ is } A_2 \text{ and } x_2 \text{ is } B_2 \text{ then } f_4 = p_4x_1 + q_4x_2 + r_4 \tag{4}$$

Where $f_j = p_jx_1 + q_jx_2 + r_j$ for $j = 1, 2, ..., 4$, x_1 and x_2 are the torque error and its derivative: $[x_1, x_2] = [e, \Delta e]$. A_i and B_i are fuzzy subsets, p_j, q_j and r_j are the consequence parameters of the rule j determined in the learning process.

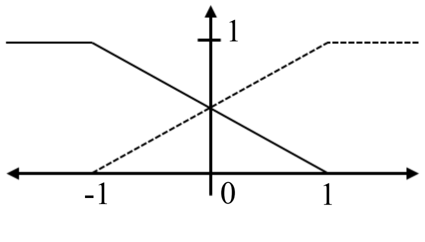


Figure 3: Membership functions.

We associate two fuzzy sets for each of the inputs x_1 and x_2 namely $N(Negative)$ and $P(Positive)$. μ_P and μ_N are the degrees of membership of the variables x_i to the fuzzy subsets A_i and B_i, defined by the following membership functions (Figure 3):

For i=1,2 :

$$\mu_P(x_i) = \begin{cases} 0, \quad \text{if} \quad x_i < -1, \\ 0.5x_i + 0.5, \quad \text{if} \quad -1 < x_i < 1, \\ 1, \quad \text{if} \quad x_i > 1. \end{cases} \tag{5}$$

$$\mu_N(x_i) = \begin{cases} 1, \quad \text{if} \quad x_i < -1, \\ -0.5x_i + 0.5, \quad \text{if} \quad -1 < x_i < 1, \\ 0, \quad \text{if} \quad x_i > 1. \end{cases} \tag{6}$$

Using the defuzzification method by the center of gravity, the numerical value of the output u is given by [20]:

$$u = \frac{\sum\limits_{j=1}^{4} f_j w_j}{\sum\limits_{j=1}^{4} w_j} \tag{7}$$

Where:

$$\begin{cases} w_1 = \mu_P\left(x_1\right).\mu_P\left(x_2\right) = \mu_{A_1}\left(x_1\right).\mu_{B_1}\left(x_2\right) \\ w_2 = \mu_P\left(x_1\right).\mu_N\left(x_2\right) = \mu_{A_1}\left(x_1\right).\mu_{B_2}\left(x_2\right) \\ w_3 = \mu_N\left(x_1\right).\mu_P\left(x_2\right) = \mu_{A_2}\left(x_1\right).\mu_{B_1}\left(x_2\right) \\ w_4 = \mu_N\left(x_1\right).\mu_N\left(x_2\right) = \mu_{A_2}\left(x_1\right).\mu_{B_2}\left(x_2\right) \end{cases} \tag{8}$$

3.2 Learning algorithm

The learning process consists of identifying the parameters of the premises and consequences. In this subsection, we develop the learning algorithm for the master system (Torque). The development is the same for the slave system (Position). Thus, let us assume y_d and y are respectively the desired and actual outputs of the master system (respectively T_e and T_h). In this work, we consider that the parameters of the premises are fixed, whereas those of the consequences are adjusted by the minimization of the following objective function [29]:

$$J = \frac{1}{2}e^2 \tag{9}$$

Where $e = y_d - y$.

Let be Φ_j the vector of the parameters to be adjusted. The objective is to find the parameters p_j, q_j and r_j of the vector Φ_j using the gradient descent method [37] as follows:

$$\Phi_j(k+1) = \Phi_j(k) - \alpha(k)\frac{\partial J}{\partial \Phi_j} \tag{10}$$

we have:

$$\frac{\partial J}{\partial \Phi_j} = -e\frac{\partial y}{\partial \Phi_j} = -e\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j} \tag{11}$$

From equations 10 and 11, it follows:

$$\Phi_j(k+1) = \Phi_j(k) + \alpha(k)\frac{\partial y}{\partial u}\frac{\partial u}{\partial \Phi_j}e \tag{12}$$

In our case, $\frac{\partial y}{\partial u}$ cannot be evaluated, but can be estimated using the extended Kalman filter equations.

Equation 12 can be written as:

$$\Phi_j(k+1) = \Phi_j(k) + K_j^{'}\left(\Psi_j\right)^T e \tag{13}$$

where:

$$(\Psi_j)^T = \frac{\partial u}{\partial \Phi_j} = \begin{bmatrix} \dfrac{\partial u}{\partial p_j} \\ \dfrac{\partial u}{\partial q_j} \\ \dfrac{\partial u}{\partial r_j} \end{bmatrix} \tag{14}$$

$$K_j^{'} = \alpha(k)\frac{\partial y}{\partial u} \tag{15}$$

The equation 13 can be identified to the extended Kalman filter equation [14]:

$$\Phi_j(k+1) = \Phi_j(k) + K\left(k\right)e \tag{16}$$

Where $K\left(k\right)$ is the Kalman gain defined as follows:

$$K\left(k\right) = \frac{P\left(k\right)H^T\left(k\right)}{H\left(k\right)P\left(k\right)H^T\left(k\right) + R\left(k\right)} \tag{17}$$

Where $H\left(k\right)$ is the Jacobian matrix (observation matrix of the system), $P\left(k\right)$ is the covariance estimation matrix of the error and $R\left(k\right)$ is the covariance matrix of the process noise.

Taking $H^T\left(k\right) = \left(\Psi_j\right)^T$,$P\left(k\right) = \lambda_1$ and $R\left(k\right) = \lambda_2$, the gain $K\left(k\right)$ can be written:

$$K\left(k\right) = \frac{\lambda_1}{\left(\Psi_j\right)\lambda_1\left(\Psi_j\right)^T + \lambda_2}\left(\Psi_j\right)^T \tag{18}$$

Hence equation 16 reduces:

$$\Phi_j(k+1) = \Phi_j\left(k\right) + \frac{\lambda_1}{\left(\Psi_j\right)\lambda_1\left(\Psi_j\right)^T + \lambda_2}\left(\Psi_j\right)^T e \tag{19}$$

By identification between equations 13 and 19, we have:

$$K_j^{'} = \frac{\lambda_1}{\left(\Psi_j\right)\lambda_1\left(\Psi_j\right)^T + \lambda_2} \tag{20}$$

Consequently, the vector of parameters Φ_j can be estimated by the following formula:

$$\Phi_j(k+1) = \Phi_j\left(k\right) + K_j^{'}\left(\Psi_j\right)^T e \tag{21}$$

For a very short sampling time T_e, we have:

$$\dot{\Phi}_j = \frac{\Phi_j\left(k+1\right) - \Phi_j\left(k\right)}{T_e} = \frac{K_j^{'}\left(\Psi_j\right)^T e}{T_e} = K_1\left(\Psi_j\right)^T e \tag{22}$$

Where $K_1 = \dfrac{K_j^{'}}{T_e}$.

Hence:

$$\dot{\Phi}_j = K_1\left(\Psi_j\right)^T e = \left(\Psi_j\right)^T e_u \tag{23}$$

Where $e_u = K_1 e$.

Let be $\widetilde{\Phi}_j = \Phi_{jd} - \Phi_j$, where Φ_j is the vector of the parameters and Φ_{jd} the vector of the desired parameters.

That implies:

$$\dot{\widetilde{\Phi}}_j = \dot{\Phi}_{jd} - \dot{\Phi}_j = -\left(\Psi_j\right)^T e_u \tag{24}$$

Where $e_u = u_d - u$ is the error between the controller's desired output u_d and actual output u. For linear variation, it is defined by:

$$e_u = u_d - u = \sum_{j=1}^{4}\left(\left(\Psi_j\right)\Phi_{jd} - \left(\Psi_j\right)\Phi_j\right) = \sum_{j=1}^{4}\left(\left(\Psi_j\right)\left(\Phi_{jd} - \Phi_j\right)\right) = \sum_{j=1}^{4}\left(\left(\Psi_j\right)\widetilde{\Phi}_j\right) \tag{25}$$

3.3 Stability analysis of the control system

Consider the following Lyapunov function [35]:

$$V_j = \frac{1}{2}\sum_{j=1}^{4}\left(\left(\widetilde{\Phi}_j\right)^T\left(\widetilde{\Phi}_j\right)\right) \tag{26}$$

Differentiating V_j with respect to time yields, we obtain:

$$\dot{V}_j = \sum_{j=1}^{4}\left(\left(\dot{\widetilde{\Phi}}_j\right)^T\left(\widetilde{\Phi}_j\right)\right) \tag{27}$$

From equations 24 , 25 and 27 , we obtain:

$$\dot{V}_j = \sum_{j=1}^{4}\left(\left(-\left(\Psi_j\right)^T e_u\right)^T\left(\widetilde{\Phi}_j\right)\right) = -\left(e_u\right)^T\sum_{j=1}^{4}\left(\left(\Psi_j\right)\left(\widetilde{\Phi}_j\right)\right) = -\left(e_u\right)^T\left(e_u\right) \tag{28}$$

Consequently:

$$\dot{V}_j = -\left(e_u\right)^T\left(e_u\right) \leq 0 \tag{29}$$

From equation 29 , we find that $\dot{V}_j \leq 0$, so we conclude that the system is asymptotically stable in the sense of Lyapunov according to the LaSalle theorem [23, 29].

4 FPGA implementation

4.1 Fixed-point data type

All FPGAs support floating point double-precision data type, and fixed-point data type [5]. Floating-point designs require larger amounts of FPGA resources than an equivalent fixed-point solutions; this high resource use leads to higher energy consumption, thus increasing the overall cost of implementing design [5]. Unlike fixed-point implementations, they are always more efficient than their floating point counterparts because they consume fewer FPGA resources and less power [16] with shorter computation time.

Real numbers represented as a fixed point consist of two parts [6], an integer part, a fractional part and a sign bit where 0 designates a positive number while 1 designates a negative number [2]. For example, $(6.5)_{10}$ (in the base 10) is written 0110.1000 with the fixed point binary representation as shown in Figure 4 [2].

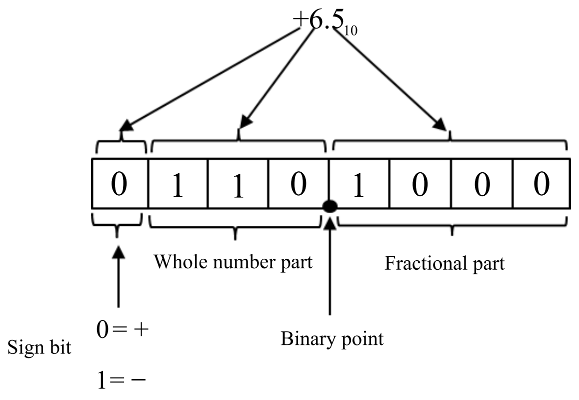


Figure 4: Binary fixed-point representation.

4.2 Design methodology of the control algorithm

Since FPGA components are composed of fixed resources, the length of the data words must be managed to meet the performance requirements and effectively use the resources. In the case where the control algorithms to be implemented are complex, specifying the optimal length of the data words is a very complex task which can affect the precision of the algorithm during its computations [11], thus the degradation of the performances. For this, the Fixed-Point Tool of Simulink is used in this work. It automates the process of converting of models synthesized in double-precision floating point to fixed-point models while optimizing the length of words [16]. Then, HDL Coder from MATLAB generates VHDL code from the Simulink models and MATLAB functions. The generated HDL code can be used to program the FPGA [23]. HDL Coder allows with its Workflow Advisor function to automate the programming of FPGAs [15] while allowing the user to control the architecture and implementation HDL, and to generate estimates of the use of hardware resources.

In this paper, the design methodology of the control algorithm, using the Fixed-Point Tool and HDL Coder of Simulink, is summarized in the diagram of Figure 5. Such design can reduce the design time compared to the low-level programming language VHDL.

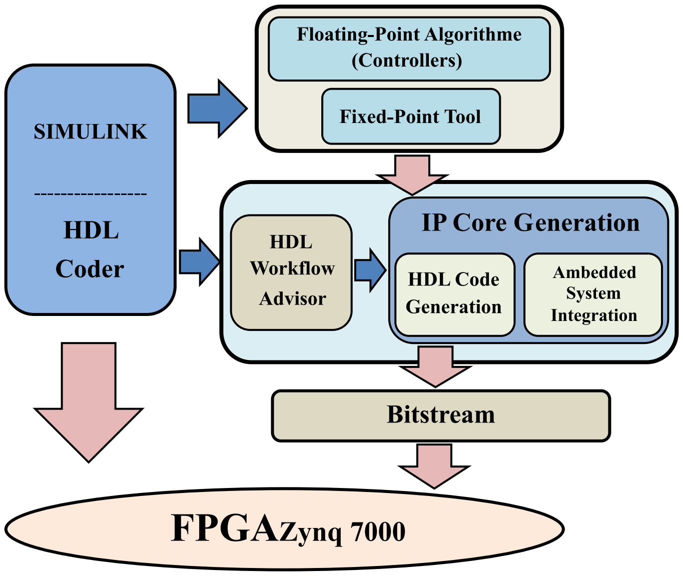


Figure 5: Algorithm design methodology on FPGA with MATLAB-Simulink.

The position and force controllers of master and slave have been developed in the MATLAB-Simulink environment using HDL Coder supported Simulink blocks, which are in the form of double precision floating point data. Then, Fixed-Point Tool automates and manages the process of converting double precision floating point models (controllers) into fixed-point models [6]. The new fixed-point models are then implemented on the FPGA via the Simulink Workflow Advisor tool, where HDL Coder Generation generates the appropriate VHDL code to program the FPGA. The final step is to create the project via Workflow Advisor's Embedded System Integration tool through the Xilinx Vivado software to create the bitstream program file that will finally be loaded onto the FPGA via the JTAG cable.

4.3 Resources consumption

When generating VHDL code from Simulink models, HDL Coder enables several types of optimizations that allow to share FPGA resources to reduce the area occupied by the design and to improve the clock frequency, while optimizing the execution time [16].

The proposed design methodology assures a good calculation precision on the one hand, which ensures good performance of the teleoperation system, and on the other hand, provides optimal hardware resources thanks to the optimization techniques used by HDL Coder.

Table 1 and Table 2 describe the FPGA hardware resources consumed when implementing PID and Neuro-Fuzzy control algorithms, with a sampling rate of 1 MHz. From these tables, it can be noted that the hardware resources consumed during the implementation of the two algorithms are considerably optimized thanks to the adopted design methodology.

Resources	Utilization	Available	Utilization (%)
FF (Flip-Flop)	726	106400	0.68
LUT (Look Up Table)	831	53200	1.56
Memory 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 1: FPGA resources usage with PID controllers.

Resources	Utilization	Available	Utilization (%)
FF (Flip-Flop)	1055	106400	0.99
LUT (Look Up Table)	4780	53200	8.98
Memory 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 2: FPGA resources usage with ANFIS controllers.

5 Experiment

The system consists of two identical devices, master and slave. Each device consists of an arm actuated by a DC motor 12 V (DFRobot model 28PA51G). These motors are equipped with integrated hall encoders and encoder adapter cards (FIT0324) which allow recovering the position signal of the motor shaft. The hall encoder produces 2700 pulses per turn and can detect 0.54 degrees rotation of the shaft.

To measure the interaction torques with the operator (at the master station) or the environment (at the slave station), a strain gauge has been placed on each arm of the engine. The forces applied on the arm cause a flexion which is measured by a strain gauge. This gauge has been mounted in a Wheatstone bridge with a 1/4 bridge configuration (Figure 6). The output voltage of the Wheatstone bridge is given by the following relation [13]:

$$V_{out} = V_{in}\left(\frac{R_g}{R_g + R_3} - \frac{R_2}{R_2 + R_1}\right) \tag{30}$$

With R_g is the gauge resistance.

The measurement of the torque is based mainly on the measurement of a force F since the torque T can be considered as a force F applied on a lever arm of length l:

$$T = F.l \tag{31}$$

The gauge calibration was done according to the forces applied on the motor arm to calculate the corresponding torque.

An instrumentation amplifier (INA126) is used to amplify the voltage delivered by the gauge that will be converted using an analog-to-digital converter 8 bits (ADC 0804) to be processed by the acquisition card (FPGA).

The human operator remotely controls the slave device to follow the movements of master's positions, while feeling the reaction torque of the environment on the slave. In order to confirm the robustness of the neuro-fuzzy controller (ANFIS) developed previously, we compared its performances with those of a conventional PID controller. Therefore, master and slave controllers for each PID and ANFIS control strategy, encoders and PWMs have been programmed with blocks of Simulink supported by HDL Coder. The outputs of the encoders were programmed on 16 bits, while

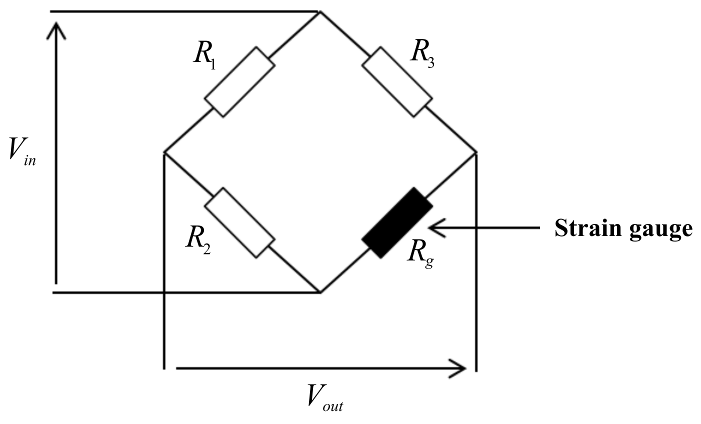


Figure 6: Strain gauge mounted on a Wheatstone bridge (1/4 bridge configuration).

torque signals and PWM inputs on 8 bits. The outputs of the controllers were calculated with Fixed-Point Tool, with a 16-bit integer part and an adequate fractional part to ensure precision. The two controllers were implemented via the Workflow Advisor from Simulink on a Zedboard development board with a FPGA of Xilinx family type Zynq-7000 AP-SoC-XC7Z020-CLG484 [42], by considering sampling frequency of 1 MHz (Figure 7).

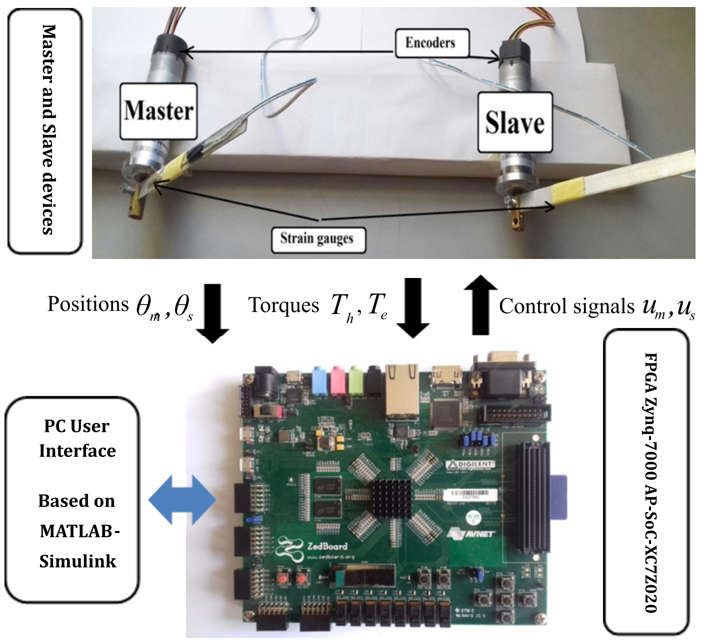


Figure 7: The master-slave experimental system.

5.1 Electronic circuit of the system

The positions of the master and slave robots are measured through hall effect encoders integrated into DC motors, while the interaction torque of the slave with the environment and the torque applied by the operator on the master are measured by the strain gauges. Signals delivered by the Wheatstone bridge are amplified by instrumentation amplifiers (INA126) and then converted by analog-to-digital converters (ADC 0804). The FPGA allows the acquisition of the torque signals delivered by the converters, and the encoder signals A and B in order to calculate the positions of the two robots. These signals are used by the controllers to calculate the PWM control signals and identify the directions of rotation S1 and S2 of each motor. These master and slave motors are controlled by drivers (type L298N), each consisting of a double H bridge, designed to control DC motors. The electronic circuit of the teleoperation system is illustrated in Figure 8 .

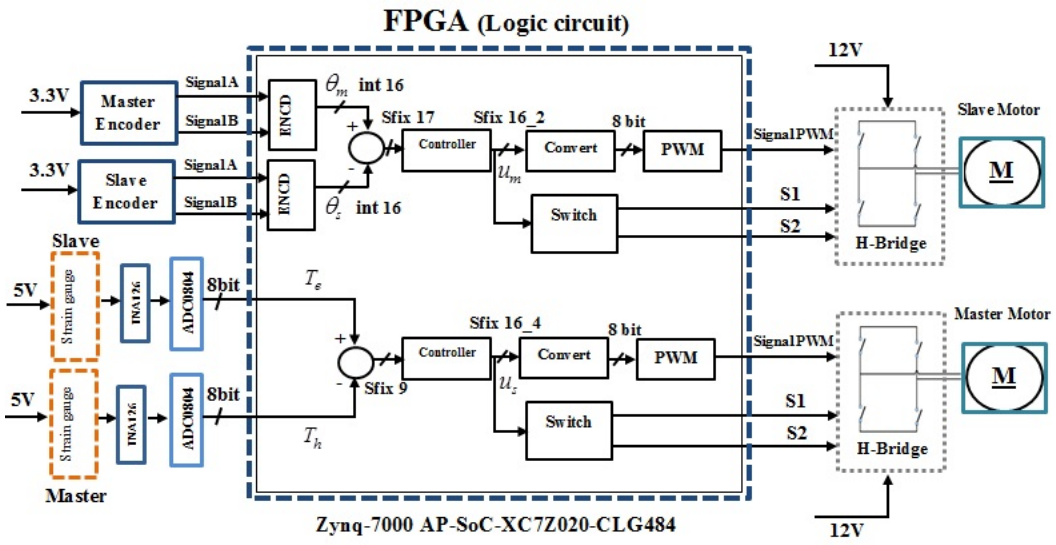


Figure 8: Electronic circuit of the teleoperation system.

6 Experimental results

In this section, experimental results conducted to verify the efficiency and performance of the proposed approach for one degree of freedom teleoperation system. The following two experiments were carried out: Contact with a soft environment (flexible aluminum rod) and contact with a hard environment (hard aluminum rod).

In order to confirm the ANFIS controller's robustness, we compared in identical conditions the performances of the neuro-fuzzy controllers with those of the PID controllers.

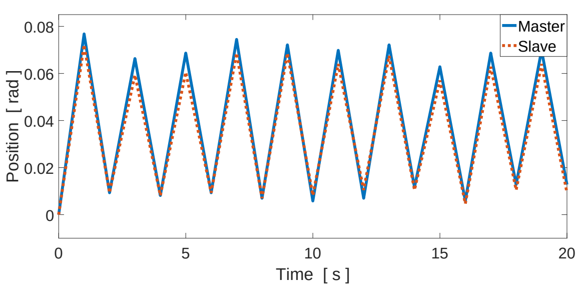
	Torque controller (Master)	Position controller (Slave)
P	22	20
I	0.1	2
D	1.5	0.1

Table 3: Parameters of PID controllers.

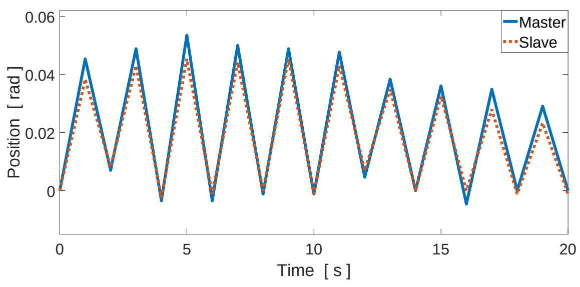
Firstly, the PID controllers with the parameters listed in Table 3 were adopted in two different cases: Contact with a soft environment and contact with a hard environment.

Experimental results showing torque and position trackings in the case where the slave is in contact with a soft environment are illustrated in Figures 9(a) and 11(a), while those showing torque and position trackings in the case where the slave is in contact with a hard environment are illustrated in Figures 9(b) and 11(b). Figures 10(a,b) and 12(a,b) show respectively the time evolution of the position errors and the torque errors. From these figures, it can be seen that a good position tracking has been achieved with a small error in the case where the slave was in contact with a soft environment, and good torque tracking was obtained. However, when the slave was in contact with a hard environment, deterioration in position tracking was observed, while the results of the torque tracking are acceptable. These results can be explained by the PID controllers which cannot adapt to the dynamic variations in master and slave.

Secondly, the ANFIS controllers were adopted in the two cases mentioned above.

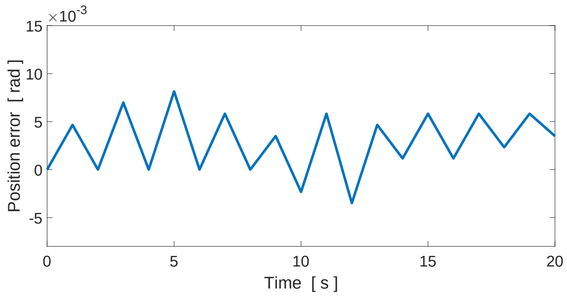


(a)

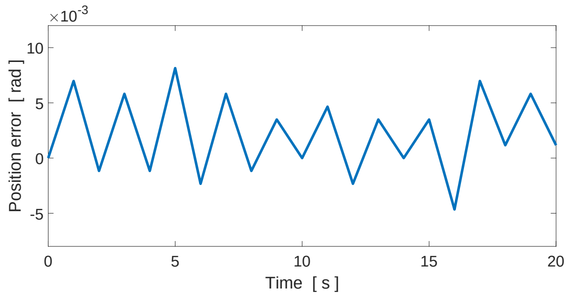


(b)

Figure 9: Motion tracking behavior with PID controllers. (a) Contact with a soft environment, (b) Contact with a hard environment.

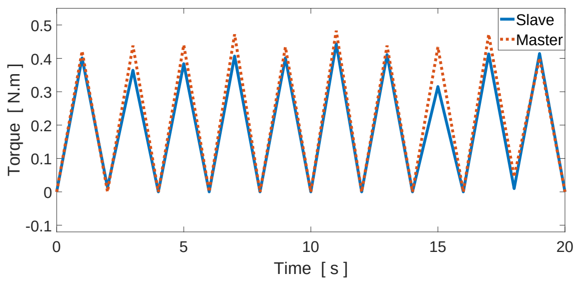


(a)

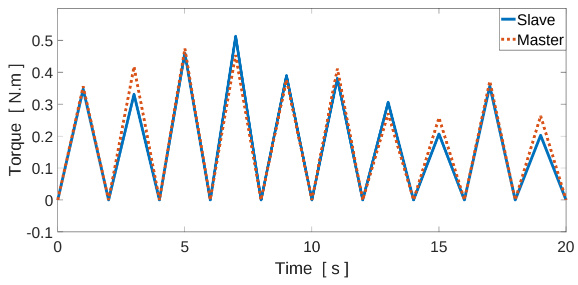


(b)

Figure 10: Position error behavior with PID controllers. (a) Contact with a soft environment, (b) Contact with a hard environment.



(a)



(b)

Figure 11: Torque tracking behavior with PID controllers. (a) Contact with a soft environment, (b) Contact with a hard environment.

Experimental results showing torque and position trackings in the case where the slave is in contact with a soft environment are illustrated in Figures 13(a) and 15(a), while those showing torque and position trackings in the case where the slave is in contact with a hard environment are illustrated in Figures 13(b) and 15(b). Figures 14(a,b) and 16(a,b) show respectively the time evolution of the position errors and torque errors. From these figures, it can be seen that an excellent position tracking was obtained with a very small error in the case where the slave was in contact with a soft environment and that good torque tracking was obtained.

Figures 13(a) and 15(a) show that the position and torque tracking results obtained with the ANFIS controllers during a contact with a soft object were much better compared to those obtained with the PID controllers (Figures 9(a), 11(a)). When the slave was in contact with a hard environment, we can see that the results of the position

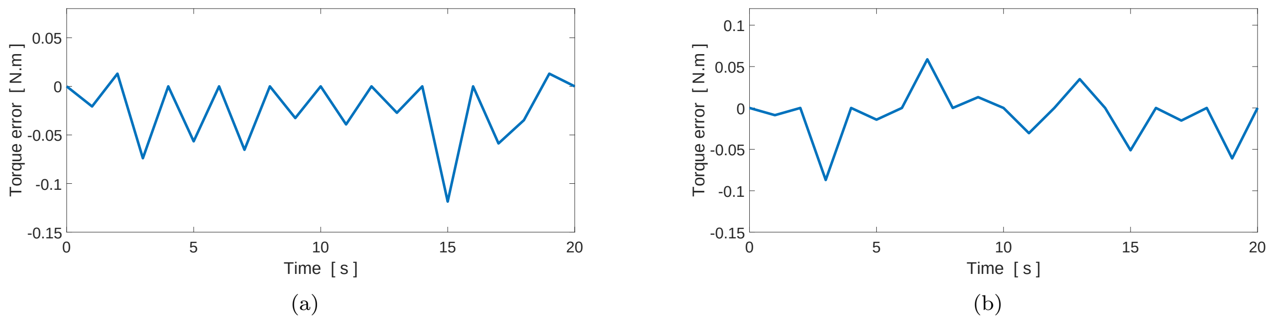


Figure 12: Torque error behavior with PID controllers. (a) Contact with a soft environment, (b) Contact with a hard environment.

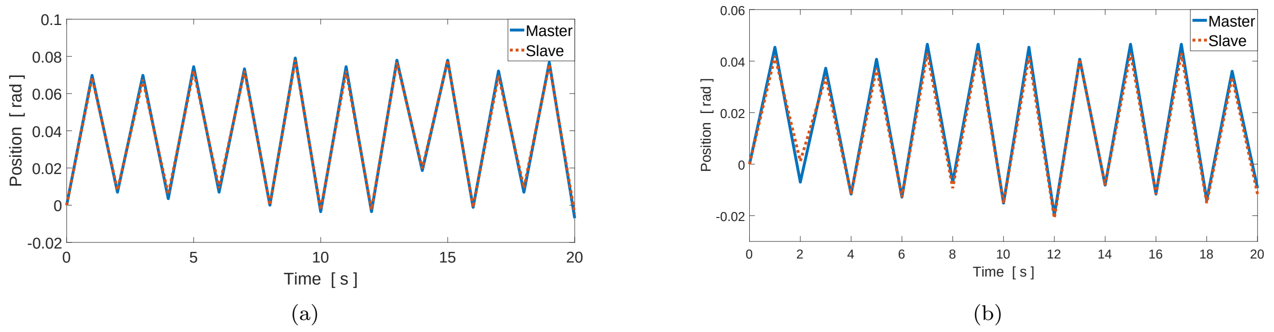


Figure 13: Motion tracking behavior with ANFIS controllers. (a) Contact with a soft environment, (b) Contact with a hard environment.

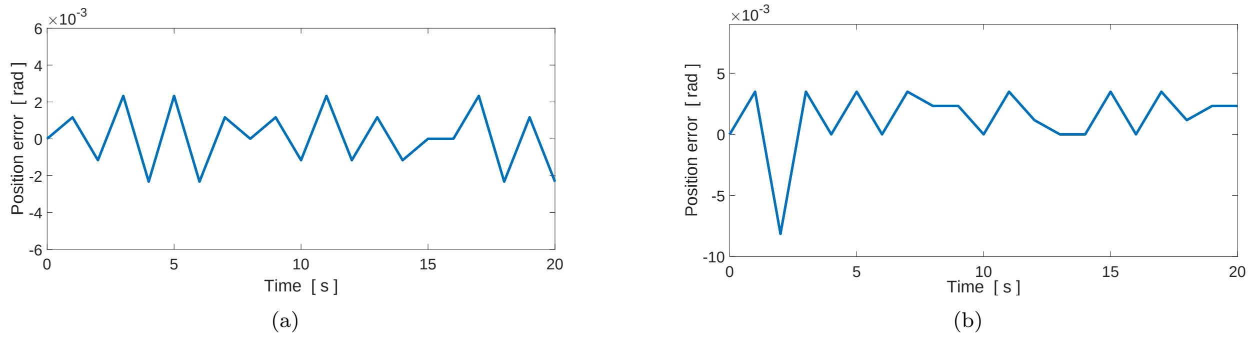
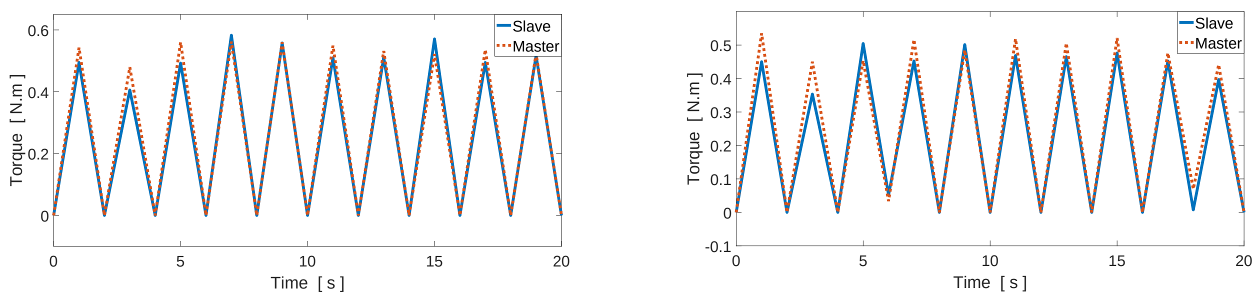


Figure 14: Position error behavior with ANFIS controllers. (a) Contact with a soft environment, (b) Contact with a hard environment.

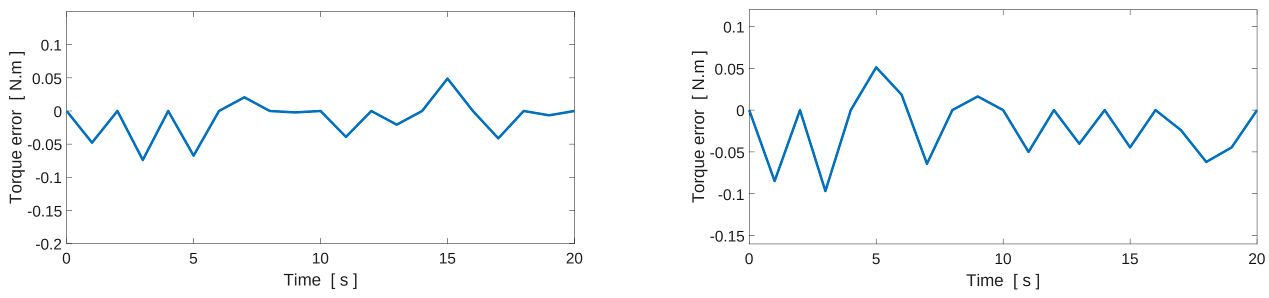
tracking obtained with the ANFIS controllers in the case where the slave was in contact with a hard object were better compared to the results obtained by the PID controllers in the same case, the slave tracked the master positions with fidelity with a very small error, and the torque tracking was good, as shown in Figures 13(b) and 15(b).

From all these graphs, it can be seen that ANFIS controllers proposed in this work provide better transparency compared to conventional PID controllers, where the slave follows the movements of the master with fidelity while allowing the operator to feel the reaction force of the environment on the slave. The ANFIS controllers ensure the stability of contact with the environment, moreover, the efforts felt by the operator through the master robot are very close to the stiffness of the environment, and the human operator then feels as if he is in direct contact with the remote environment that changes stiffness. These results can be explained by the two ANFIS controllers of the position and



(a) (b)

Figure 15: Torque tracking behavior with ANFIS controllers. (a) Contact with a soft environment, (b) Contact with a hard environment.



(a) (b)

Figure 16: Torque error behavior with ANFIS controllers. (a) Contact with a soft environment, (b) Contact with a hard environment.

the torque, which apply an intelligent control algorithm that compensates for the nonlinearities of the teleoperation and quickly adapt to the changing system dynamics when the slave was in contact with the environment, and this by updating in a short time the consequence parameters of the neuro-fuzzy network thanks to the high computing power of the FPGA and its short sampling periods. In addition, the design methodology of the controllers used in this paper allowed obtaining good results in our teleoperation system in terms of precision and stability. The MATLAB-Simulink environment with its Fixed-Point Tool and HDL Coder functionalities allowed designing a robust algorithm with great precision by optimizing the execution time, the design frequencies, and the hardware resources used in the FPGA.

7 Conclusion

In this paper, an adaptive neuro-fuzzy control (ANFIS) for a bilateral teleoperation system with one degree of freedom using a FPGA was presented. The basic idea was to design two adaptive controllers (ANFIS) for a nonlinear teleoperation system whose dynamics are unknown. These controllers have guaranteed excellent transparency and fidelity, they were able to adapt to the nonlinear dynamic variations of the master and slave robots. The ANFIS controllers were able to reduce position and torque tracking errors with the proposed learning algorithm while maintaining a stable contact of the slave robot with the unknown environment, and this by combining the inference capabilities of fuzzy logic with the learning capabilities of neural networks. The ANFIS algorithms developed in this paper allowed updating the consequence parameters of the neuro-fuzzy network in a very short time using a learning algorithm, and this by the use of the FPGA, which offers a data-parallel computation and a high sampling frequency.

The design methodology of the control algorithm adopted in this work and the use of the fixed-point data type allowed to have an optimal VHDL code in terms of hardware resources and execution time, with low consumption of energy. The MATLAB-Simulink environment has allowed with its Fixed-Point Tool and HDL Coder features to reduce design time and obtain a faster and more accurate algorithm.

The experiments performed on the teleoperation system have shown that the results of the position and torque trackings were better using the ANFIS controllers compared to the results obtained by the conventional PID controllers. ANFIS controllers were able to reduce the position and torque tracking errors with the proposed learning algorithm, which adjust the output parameters of the neuro-fuzzy network in a very short time, thanks to the advantages of the FPGA in terms of computing speed and the sampling frequency.

References

[1] A. Alfi, M. Farrokhi, *A simple structure for bilateral transparent teleoperation systems with time delay*, Journal of Dynamic Systems, Measurement, and Control, **37** (2008), 1–9.

[2] G. Dubey, *Fixed-point design in MATLAB and SIMULINK*, Mathworks, (2013), 1–28.

[3] A. Erfani, S. Rezaei, M. Pourseifi, H. Derili, *Optimal control in teleoperation systems with time delay: A singular perturbation approach*, Journal of Computational and Applied Mathematics, **333** (2018), 168–184.

[4] 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, **4** (2016), 4119–4135.

[5] A. Finnerty, H. Ratigner, *Reduce power and cost by converting from floating point to fixed point*, XILINX All Programmable, WP491(v1.0), (2017), 1–14.

[6] *Fixed-point designer user's guide*, Mathworks, MATLAB r2017b, (2017).

[7] 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, **54**(1) (2013), 49–61.

[8] 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, 1–6.

[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, **13**(5) (2015), 1–10.

[10] A. Hace, M. Franc, *Enhanced pseudo-sensorless bilateral teleoperation by PLL-tracker and FPGA*, Automatika, **55**(3) (2014), 265–275.

[11] A. Hace, M. Franc, *FPGA implementation of sliding-mode-control algorithm for scaled bilateral teleoperation*, IEEE Transactions on Industrial Informatics, **9**(3) (2013), 1291–1300.

[12] A. Haddadi, K. Hashtrudi-Zaad, *Bounded-impedance absolute stability of bilateral teleoperation control systems*, IEEE Transactions on Haptics, **3**(1) (2010), 15–27.

[13] T. Hallböök, B. Månsson, R. A. Nilsén, *A strain-gauge pletysmograph with electrical calibration*, Scandinavian Journal of Clinical and Laboratory Investigation, **25**(4) (1970), 413–418.

[14] S. Hayki, *Kalman filtering and neural networks*, John Wiley and Sons, Inc, 2001.

[15] *HDL Coder getting started guide*, Mathworks. MATLAB and SIMULINK, r2018a, 2018.

[16] *HDL Coder user's guide*, Mathworks. MATLAB and SIMULINK, r2018a, 2018.

[17] P. F. Hokayem, M. W. Spong, *Bilateral teleoperation: An historical survey*, Automatica, **242**(12) (2006), 2035–2057.

[18] E. Ishii, H. Nishi, K. Ohnishi, *Improvement of performances in bilateral teleoperation by using FPGA*, IEEE Transactions on Industrial Electronics, (2007), 317–322.

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

[20] J. S. R. Jang, C. T. Sun, E. Mizutani, *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*, Prentice Hall, 1997.

[21] 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, (1996), 837–842.

[22] P. Letier, *Bras exosqulette haptique: Conception et contrôle*, Doctorat en Sciences de l'Ingénieur. Université Libre de Bruxelles (ULB), 2010. (In French).

[23] FL. Lewis, CT. Abdallah, DM. Dawson, *Control of robot manipulators*, Macmillan, New York, 1993.

[24] Y. Li, Y. Yin, S. Zhang, J. Dong, R. Johansson, *Composite adaptive control for bilateral teleoperation systems without persistency of excitation*, Journal of the Franklin Institute, (2018), 1–21.

[25] X. Liu, R. Tao, M. Tavakoli, *Adaptive control of uncertain nonlinear teleoperation systems*, Mechatronics, **24** (2014), 66–78.

[26] X. Liu, M. Tavakoli, *Bilateral adaptive control of nonlinear teleoperation systems with uncertain dynamics and dead-zone*, Journal of Dynamic Systems, Measurement, and Control, **140** (2018), 1–10.

[27] X. Liu, M. Tavakoli, *Inverse dynamics-based adaptive control of nonlinear bilateral teleoperation systems*, IEEE International Conference on Robotics and Automation,Shanghai International Conference Center, (2011), 1323–1328.

[28] S. Liu, X. Zhang, W. Zheng, B. Yang, *Adaptive control for time-delay teleoperation systems with uncertain dynamics*, Journal of Physics: Conf. Series 887, (2017), 1–8.

[29] R. Mellah, S. Guermah, R. Toumi, *Adaptive control of bilateral teleoperation system with compensatory neural-fuzzy controllers*, International Journal of Control, Automation and Systems, **15** (2017), 1–11.

[30] 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, (2015), 382–387.

[31] R. Mellah, R. Toumi, *Control bilateral teleoperation by compensatory ANFIS*, Advanced Mechatronics Solutions, **393** (2016), 167–172.

[32] E. Monmasson, L. Idkhajine, MN. Cirstea, I. Bahri, A. Tisan, MW. Naouar, *FPGAs in industrial control applications*, IEEE Transactions on Industrial Informatics, **7**(2) (2011), 224–243.

[33] 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), (2016), 1318–1323.

[34] 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.

[35] L. Peng, P. Y. Woo, *Neural-fuzzy control system for robotic manipulators*, IEEE Control Systems Magazing, (2002), 53–63.

[36] W. Po-Ngaen, *Adaptive four-channel neuro-fuzzy control of a master-slave robot*, International Journal of Advanced Robotic Systems, **10** (2013), 1–8.

[37] S. Ruder, *An overview of gradient descent optimization algorithms*, Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin, (2016), 1–14.

[38] 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, **22** (2006), 25–32.

[39] 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, **56**(3) (2009), 618–627.

[40] 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, **352** (2015), 1850–1866.

[41] T. Yang, J. Hu, W. Geng, Y. Fu, M. Tavakoli, *FPAA-based control of bilateral teleoperation systems for enhanced user task performance*, Presence: Teleoperators and Virtual Environments, **26**(2) (2017), 210–227.

[42] *ZedBoard (Zynq evaluation and development)*, Hardware user's guide. Version 2.2, (2014), 1–37.