PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

FERHAT ABBAS SETIF-1 UNIVERSITY

FACULTY OF TECHNOLOGY

THESIS

submitted to the Electronics Department

To obtain a diploma of

DOCTOR IN SCIENCES

By

REFOUFI Salim

THEME

Control of autonomous mobile robot by quantitative inference

Defended the 23/12/2018 with the committee members:

CHEMALI Hemimi	**Professor**	**Univ. F. Abbas Setif 1**	**President**
BENMAHAMMED Khier	**Professor**	**Univ. F. Abbas Setif 1**	**supervisor**
BERTIL Arres	**Professor**	**Univ. F. Abbas Setif 1**	**examiner**
ABDESSEMED Foudil	**Professor**	**Univ. Hadj Lakhdar Batna**	**examiner**
BOUMEHRAZ Mohamed	**Professor**	**Univ. M.Khieder Biskra**	**examiner**
TAIBI Mahmoud	**Professor**	**Univ. Badji Mokhtar Annaba**	**examiner**

Thanks

Praise to ALLAH the Almighty, the All Merciful and the Most Merciful, to have given us the patience, willingness and the necessary force to realize this project.

I expressed my profound thanks to mythesis director Prof. K Benmahammed for the responsibility ensuring and for encouraging and supporting me in our project, I also express my sincere gratitude for the trust who testified to me throughout this work.

I extend my sincere thanks to the jury members, Mr.CHEMALI Hemimi Professor at the University of F. Abbas SETIF-1 for accepting preside this work, Mr. BERTIL Arres Professor at the University of F. Abbas SETIF-1, Mr.ABDESSEMED Foudil Professor at the University of Hadj. Lakhdar Batna, Mr.BOUMEHRAZ Mohamed Professor at the University of M.Khieder Biskra, Mr.TAIBI Mahmoud Professor at the University of BadjiMokhtar Annaba,for agreeing to judge this work.

I wanted to thanks greatly

• The staff of the electronics department.

• My friends from the LSI laboratory for their friendships and support.

Summary

Summary

List of Figures

List of tables

Lists of Acronyms and Symbols

Acronymes

d.o.f	Degree of freedom
ANN	Artificial neural networks
FNNs	Fuzzy neural networks
FLS	Fuzzy Logic System
GAs	Genetic algorithms
PID	Proportional/integral/derivate
MFs	Membership functions
E-L	Euler-Lagrange
D-H	Denavit-Hartenberg
w.r.t	With respect to
s.t	Such that
LTI	Linear time invariant
LHP	Left-half plane
e.s	Exponentially stable
DFL	Dynamic Feedback Linearization

Symboles

$M(q)$:	Inertia matrix.
$H(q, \dot{q})$:	Coriolis/centrifugal vector.
$G(q)$:	Gravity vector.
$F(q, \dot{q})$:	Friction torque
m_1:	Body mass 1.
m_2 :	Body mass 2.
m_p	Load mass.
G_1 :	Center of body mass 1.
G_2 :	Center of body mass 2.
L_1 :	Body length 1.
L_2 :	Body length 2.
L_{c1} :	Center of mass position G_1 compared to O_1.
L_{c2} :	Center of mass position G_2 compared to O_2.
m :	The mobile robot mass.
I :	The moment of inertia of the robot about its center of mass.
d :	The distance between the center of mass and the center of the wheel axle.
r :	The wheel radius.
R:	Half distance between the two wheels.
τ_l, τ_r:	The motor torques on the wheels.
λ:	The constraint force.
q :	Vector joint positions.
$\dot{q}$:	Vector joint velocities.
$\ddot{q}$:	Vector joint accelerations.
τ :	The generalized joint torque vector.

GENERAL INTRODUCTION

1.1. State of the art

Faced with the delicate problems of modeling and control of the complex systems such as the robots, the tools used are becoming increasingly cheaper and more powerful. The PID has proven its effectiveness in many problems of the industrial regulation. However, if the precision is required, it becomes inapplicable because of its linearity. In the recent decades, many research works have shown that the nonlinearities generate specific phenomena and these had to be taken into account to study the system in a satisfactory manner. Indeed, either these phenomena are undesirable, and then the knowledge of their mechanisms is essential to avoid them, or it is inevitable and intrinsic to the system itself, then it must be understood and modeled, or it must beintroducedfor the purpose of well-defined applications. Therefore, one needs the introduction of other algorithms to develop new powerful controllers [1,2]. An approach having experienced significant developments these last years is the fuzzy controller relating to the systems containing knowledge: “IF (conditions) THEN (Action)" [3]. In spite of the significant number of applications developed by the fuzzy control, it always lacks tools that make it possible to analyze these controllers [4,5]. To avoid the problem of the expertise, the researchers tried to replace the expert by several methods such as the neural networks and the genetic algorithms.

Neural networks are based on the mechanisms of operations of the human brain, in the form of layers [6,7,8]. The genetic algorithms are also very much used for the optimization of fuzzy regulators. They are based on the theory of evaluation of the most adapted elements of a generation towards the other, so that the best adapted ones would last in time, while the other should disappear. The proposed adaptive neuro-fuzzy optimized by *GAs*control scheme has some advantages over classical fuzzy control and neural networks control schemes [9,10]. For example, in a classical fuzzy controller design, some controller parameters must be tuned by trial and error. Such parameters include scalars (or gains) for its inputs and outputs data, the number of membership functions, the width of a single membership function, and the number of control rules [11,12,13,14,15,16,17,18,19,20,21]. On the contrary, the adaptive neuro-fuzzy controller based on the dynamic model estimation would allow the parameters to be tuned automatically [22]. In addition, the performance of the neuro-fuzzy controller optimized by *GAs* is also considered to be excellent.

The neuro-fuzzy-genetic controller does not require knowledge of the robot parameter values. It needs, neither system dynamic model nor control experts for the robot control problem.

The application of neuro-fuzzy to the manipulator robot has motivated considerable research in this area in recent years. In [23], an adaptive control using multiple incremental fuzzy neural networks (*FNNs*) was presented. The controller is combined by a feedback controller and several *FNNs* to learn inverse dynamics of the manipulator robot for different tasks. The several *FNNs* are obtained dynamically using an incremental hyper-plane-based fuzzy clustering algorithm to compensate the unknown disturbances of the system. In [24], a decentralized intelligent control method for a robust control of underwater manipulator was developed. The controller is based on

a neuro-fuzzy approach where the feedback gains are adapted using fuzzy logic, whereas a neural network is used to add to the controller a feed-forward compensation input. The neural network is trained using the back-propagation approach to estimate the system dynamics from the input-output data. Therefore, the proposed method does not depend on modeling and dynamics estimation system. The same authors in [25] proposed an intelligent controller used a neuro-fuzzy approach for precise control, robust and energy efficient. The decentralized controller is a PD type, with fuzzy setting feedback gains modified using modified fuzzy membership functions. The magnitude of the gains and thus, the energy expenditure is reduced by the neural network based on the identification of the system dynamics and hydrodynamic disturbances. In [26], a comparative study of several hybrid neuro-fuzzy adaptive control systems to control the six degrees of freedom (Puma 560) robot arm, with uncertainties were presented. The proposed controller combines ANFIS based controllers, with some well established traditional controllers (computed torque control, feed forward inverse dynamics, and critical damping inverse dynamics control).

In [11], both orthogonalization and passivity properties are taken into account to design a neuro-fuzzy system (NFS) like a PID controller to ensure tracking by exploring a new energy re-shaping of the closed loop system through a self-optimization of the dissipation rate gain (DRG) without any knowledge of the robot dynamics. In [12], Fuzzy Logic System (FLS) was presented to generate the rules using neuro-fuzzy methods. In [27], a single hidden layer in a fuzzy recurrent wavelet neural network (SLFRWNN) was developed and used for the function approximation and the identification of dynamic systems. And in [28], a supervised training algorithm based on sliding mode theory that implements fuzzy reasoning on a spiking neural networks, for the trajectory control problem of a robotic manipulator was developed and tested. In [29], a new neuro-fuzzy classifier which draws its inspirations from the concepts of the linguistic hedges was used.

1.2. Problem studied

For a manipulator robot which carried loads, control planning becomes a problem even more difficult to solve than in the case where the robot contains parameters poorly modeled. The robot is subjected to new constraints generated by its contact interaction with the environment, and can not be neglected in a drive generation process. In this work we used a neuro-fuzzy-genetic controller to drive the robots. The neural networks are proposed for the automatic extraction of the fuzzy rules, using a Kalman filter as a training means. To optimize the location of the fuzzy subsets, the *GAs* are used, and since there are several forms of the fuzzy subsets such as triangular, Gaussian…, the choice of the appropriate form becomes a problem. Thus the variation of the load carried by the robot causes the dynamics of the manipulator to lose the tracking.

For the mobile robot from a control viewpoint, the peculiar nature of nonholonomic kinematics makes that feedback control at a given trajectory cannot be achieved via smooth time-invariant control. This indicates that the problem is really nonlinear, linear control is ineffective, even locally, and innovative design techniques are needed.

1.3. Theproposed approach

The work presented in this thesis is describes the displacementscontrol of a robot carrying a load. To well learn dynamics of the manipulator robot, which represent an extension of the idea given by [25] and [29] where they used a triangular membership functions (MFs), with linguistic hedges to modify the shape of the MFs. In this work, we used a neuro-fuzzy-genetic controller. The neural networks are proposed for the automatic extraction of the fuzzy rules, using a Kalman filter as a training means. To optimize the location of the fuzzy subsets, we used the *GAs*, and since there are several forms of fuzzy subsets such as triangular, Gaussian, etc…, and as a contribution, we used an interpolation function with three points, and then with five points, for each fuzzy subset, and we optimized the curves by the *GAs*, to find the most suitable shape of a fuzzy subset, which is going to be adapted according to the load. The same idea applied for the control of a mobile robot using dynamic feedback linearization as linear control and we replace the PD compensator by a Neuro-fuzzy-Genetic controller.

1.4. Thesis organization

This thesis proposes a hybrid type controller that combines a fuzzy regulator adapted by neural networks, and optimized by genetic algorithms.

It is organized as follows:

The first chapter is devoted to a geometric, kinematic and dynamic modeling of a robot arm, and a mobile robot of differential speed, as well as the generation of trajectories, and their control techniques.

The second chapter treats the general description of the genetic algorithms functioning.

The third chapter presents the theoretical basis of fuzzy logic, generalities on neural networks, and a combination of the two to obtain a neuro-fuzzy controller.

In the fourth chapter, we presente the different fuzzy control methods, Neuro-fuzzy, Neuro-fuzzy optimized by genetic algorithms, for the control of a manipulator arm and a mobile robot.

Chapter 01

MODELING AND CONTROL OF ROBOTS

1.1. Introduction

For several decades, robots have been used in the industry, the majority of which are manipulator type and are used to perform repetitive tasks in a production cycle. Self-guided vehicles (also called mobile robots) are also used in the manufacturing industry.

However, research in the field of robotics is still very active and a very large part of this activity is directed towards the development of new applications for robots [11,23,24,25,26]. Work focuses on service robots, i.e, robots (manipulators or mobiles) working in automatic or semi-automatic mode, and which are not used in a context of industrial production. These robots perform tasks that are useful to people, to accomplish certain duties. For exemple, the robot can replace a person working in hazardous conditions, such as nuclear plant inspection robots, robots specializing in the detection and manipulation of explosives (anti-personnel mines), planetary exploration robots and submersible self-guided vehicles or those operating in underground mines are examples of these service robots.

1.2. Manipulator robots

1.2.1. Dynamic model

A mechanical system can be represented as a dynamic model to facilitate its study through the differential equations, which exist between the state variables of the mechanism, their derivatives and the external forces acting on each body. The most general form is [30]:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{1.1}$$

The dynamic model (1.1) expresses the driving torque (or force) of the actuators of the various manipulator robot arms as a function of the positions q, speeds $\dot{q}$, and accelerations $\ddot{q}$, articular forces and friction forces F to exert on the end effector. It expresses the balance between drive and braking torque due to the inertia, centrifugal and Coriolis forces as well as gravitational forces. This model is also called inverse dynamic model.

$$\tau = g(q,\dot{q},\ddot{q},F) \tag{1.2}$$

Where $\tau \in \Re^n$: Vectors of motor torques whose dimension is equal to the number of degrees of freedom of the manipulator robot.

1.2.1.1. **Lagrange Formalism**

The dynamic model (1.2) can be obtained by several methods, the most used is that of Lagrange which describes the dynamic behavior of a system in terms of work and energy.

The dynamic model is obtained by the following of Euler-Lagrange (E-L) equations [30]:

$$\frac{d}{dt}\left\{\frac{\delta L}{\delta \dot{q}_i}\right\}-\frac{\delta L}{\delta q}=\tau_i \tag{1.3}$$

With L denotes the Lagrange function given by the equation

$$L(q,\dot{q})=K(q,\dot{q})-U(q). \tag{1.4}$$

Where $K(q,\dot{q})$ is the kinetic energy and $U(q)$ the potential energy.

1.2.1.2. **Model of the robot with two degrees of freedom (*d.o.f*)**

We will model a robot with two degrees of freedom that is to say two rotational joints (Figure 1.1), moving in the vertical plane.

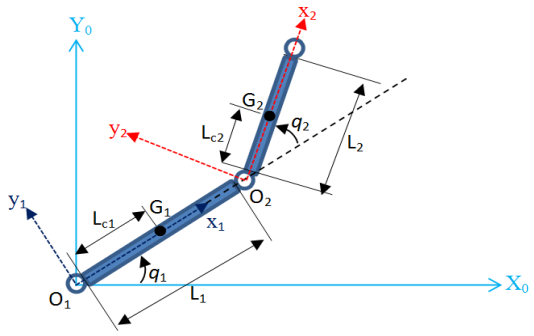


Figure 1.1 Representation of the robot with two rotational axes.

a. ***Robot dynamic model with two degrees of freedom***

According to the preceding paragraphs, the matrices and the gravities terms vector of the manipulator arm dynamic model are expressed as follows:

Inertia matrix:

$$M(q)=\begin{bmatrix} j_1(\theta_1)+j_2(\theta_2)+m_2L_1^2+2(m_2L_1L_{c2}+m_pL_1L_2)\cos(q_2) & j_2(\theta_2)+(m_2L_1L_{c2}+m_pL_1L_2)\cos(q_2) \\ j_2(\theta_2)+(m_2L_1L_{c2}+m_pL_1L_2)\cos(q_2) & j_2(\theta_2) \end{bmatrix} \tag{1.5}$$

Centrifugal and Coriolis forces Matrix

$$C(q,\dot{q})=\begin{bmatrix} -(m_2L_1L_{c2}+m_pL_1L_2)\dot{q}_2\sin(q_2) & -(m_2L_1L_{c2}+m_pL_1L_2)(\dot{q}_1+\dot{q}_2)\sin(q_2) \\ (m_2L_1L_{c2}+m_pL_1L_2)\dot{q}_1\sin(q_2) & 0 \end{bmatrix} \tag{1.6}$$

Gravitational efforts vector.

$$G(q)=\begin{bmatrix} g(m_1L_{c1}+m_2L_1+m_pL_1)\cos(q_1)+g(m_2L_{c2}+m_pL_2)\cos(q_1+q_2) \\ g(m_2L_{c2}+m_pL_2)\cos(q_1+q_2) \end{bmatrix} \tag{1.7}$$

$$\begin{cases} j_1(\theta_1) = \int (x^2 + y^2) dm_1 \\ j_2(\theta_2) = \int (x^2 + y^2) dm_2 \end{cases}$$

1.3. Trajectory generation

Robot dynamics require imposing achievable trajectories, so continuity in position, speed and acceleration gives the robot the ability to continue the trajectory with achievable controls.

The decomposition of the task into several intermediate points requires continuity of the first and second order.

The imposition of a constant final position for example, requires the use of a third degree polynomial allowing continuity in position and speed.

This task results through the passage from one equilibrium state to another, these two states will define the boundary conditions of the interpolation polynomial.

$$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 . \begin{cases} p(0) = \theta_0 & \dot{p}(0) = \dot{\theta}_0 \\ p(t_f) = \theta_f & \dot{p}(t_f) = \dot{\theta}_f \end{cases} \quad (1.8)$$

$$\begin{cases} a_0 = \theta_0 \\ a_1 = \dot{\theta}_0 \\ a_2 = \dfrac{3}{t_f^3}(\theta_f - \theta_0) - \dfrac{2}{t_f}\dot{\theta}_0 - \dfrac{1}{t_f}\dot{\theta}_f \\ a_3 = -\dfrac{2}{t_f^3}(\theta_f - \theta_0) + \dfrac{1}{t_f^2}(\dot{\theta}_f + \dot{\theta}_0) \end{cases}$$

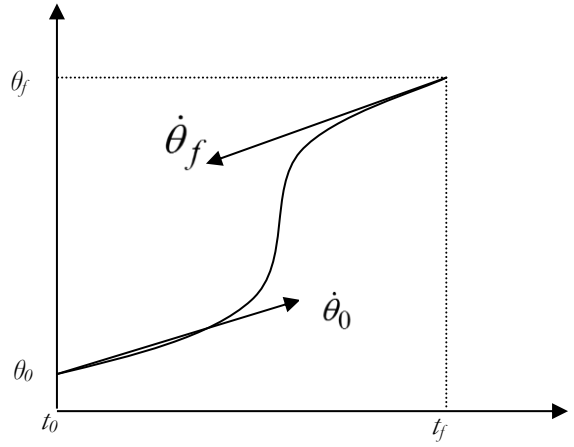


Figure 1.2 Generation of an interpolation polynomial.

From the boundary conditions (equation (1.8)), the different coefficients of the polynomial $p(t)$ are calculated.

1.4. Control techniques of manipulator robots

1.4.1. Classic control

The classical control is the set of *PID* type linear laws with constant gains. Therefore, to develop a *PID* control, it is necessary to consider each joint of the robot as an independent mechanism and can be linearized in an operation zone. The drive therefore takes a local character.

The figure shows the diagram of a classic *PID* control.

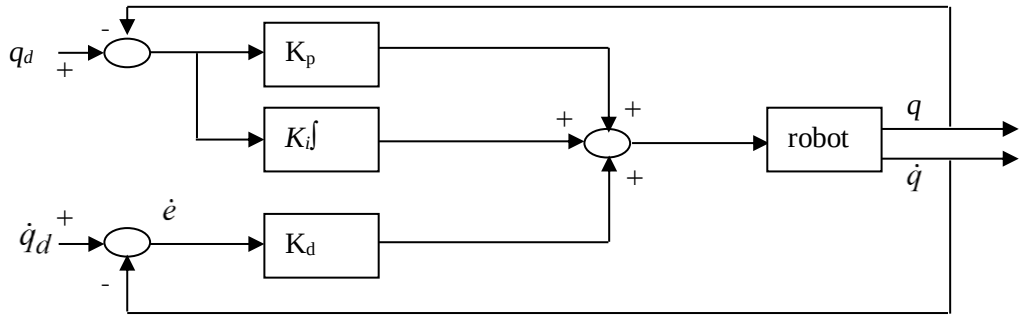


Figure 1.3 Classic diagram of a PID control.

The classical control holds the monopoly in the industrial field but it presents certain disadvantages and certain advantages.

Advantages are**:**

- Implantation efficiency.
- Low cost (implementation, computation time).

The disadvantages are:

- This drive, based on a linear model of the manipulator robot, is not acceptable for large displacements performed at high speeds and requiring good tracking.
- The integral term is indispensable to eliminate the position static error due to the forces of gravity. However, for a ramp type input, the static error remains.

The dynamics of the manipulator varying with its configuration, it will not be possible to maintain the performance of the system for all accessible configurations if the coefficients of the corrector are constant.

Drive Laws:

If the gravitational forces are compensated mechanically or otherwise, the control law chosen is of the *PD* type:

$$\tau = K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})$$

In the case where the gravitational forces are not compensated, a *PID* control is necessary and the corresponding law is of the form:

$$\tau = K_P(q_d - q) + K_D(\dot{q}_d - \dot{q}) + K_I \int_0^t (q_d - q)dt \qquad (1.9)$$

Where q_d is the desired position, $q$the real position, $\dot{q}_d$ desired velocity and $\dot{q}$ the actual velocity and K_P, K_D, K_I are the (n×n) diagonal matrices containing the gains K_{Pi}, K_{Di}, K_{Ii}.

Implementing the *PID* control requires knowledge of the gains K_{Pi}, K_{Di}, K_{Ii} of each joint**.**

For this, one must linearize the model in assuming that the dynamic equations of the joints are decoupled by neglecting the interferences between the joints, and neglecting the centrifugal and Coriolis forces as well as the gravity forces and friction.

The corresponding equation of each articulation takes the form:

$$\tau_i = J_i \ddot{q}_i \tag{1.10}$$

Where J_i represents the fixed part (or maximum in other cases) of the element m_{ij} of the inertia matrix $M(q)$.

The model is even more realistic as the reduction ratio is important, the velocities are low and the gains in position and velocity are high.

Equating the equation (1.10) with the system equation (1.9) we obtain:

$$K_{Pi}(q_d - q) + K_{Di}(\dot{q}_d - \dot{q}) + K_{Ii}\int_0^t (q_d - q)dt = J_i \ddot{q}_i$$

The closed-loop transfer function between q_i and q_{di} is as follows:

$$\frac{q_i}{q_{di}} = \frac{K_{Di}s^2 + K_{Pi}s + K_{Ii}}{J_i s^3 + K_{Di}s^2 + K_{Pi}s + K_{Ii}} \tag{1.11}$$

In robotics, the most common practice is to choose the gains so as to obtain as dominant poles a real double negative pole, in order to obtain afast response without oscillations, the other pole is chosen real negative but far from the other two.

The characteristic equation of the transfer function (1.11) is written in the form:

$$E_i(s) = J_i S^3 + K_{Di}s^2 + K_{Pi}s + K_{Ii}$$

If one puts $K'_{Pi} = \frac{K_{Pi}}{J_i}$, $K'_{Di} = \frac{K_{Di}}{J_i}$, $K'_{Ii} = \frac{K_{Ii}}{J_i}$

$$\text{So } E_i(s) = S^3 + K'_{Di}s^2 + K'_{Pi}s + K'_{Ii} \tag{1.12}$$

The chosen poles are therefore as follows:

$$K'_{Pi} = (2K+1)\beta^2, \quad K'_{Di} = (2+K)\beta, K'_{Ii} = K\beta^2 \tag{1.13}$$

Whence $E_i(s) = (s+\beta)^2(s+K\beta)$ avec $K>0$ and $\beta>0$

We notice that the gains K_p, K_D, K_I are functions of J_i supposed constant, but in reality J_i varies according to the situation of the whole robot, so the damping is really critical only for the value of J_i chosen .

1.4.2. Improving the tracking by the control

In most cases, the only measuring means present on a robot are the proprioceptive sensors measuring the quantities related to the joints. The control of an industrial robot is therefore performed using only this information. In addition, the control of a robot being in most cases a joint control.

The search for precision and speed imposes a dynamic control, that is, the actuators must be controlled by the torque. We will review the main techniques currently developed.

1.4.2.1. *Non-linear decoupling control*

The possibilities of the application of dynamic control come after the great development of

information and methods of identification. The nonlinear decoupling method is proposed by Freund [31].

This last control consists in transforming by state feedback the control problem of a linear system, which then makes it possible to apply the classical techniques of the theory of the linear control.

The diagram of this control is shown in Figure 1.4

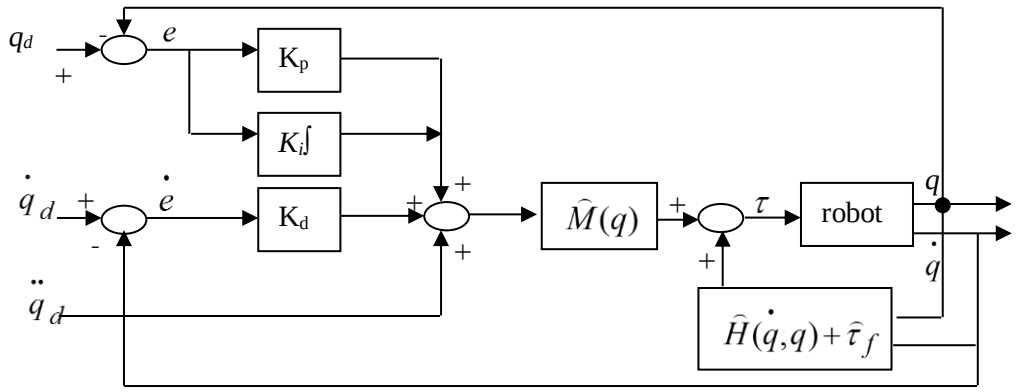


Figure 1.4 Diagram of a dynamic control by nonlinear decoupling.

If the model equation is as follows:

$$\tau = M(q)\ddot{q} + H(q,\dot{q}) + \tau_f(q)$$

The equation of the control law will be given by:

$$\tau = \hat{M}(q)u + \hat{H}(q,\dot{q}) + \hat{\tau}_f(q) \tag{1.14}$$

Where
$$u = \ddot{q}_d + K_D(\dot{q}_d - \dot{q}) + K_P(q_d - q) + K_I\int_0^t (q_d - q)dt \tag{1.15}$$

And $\hat{M}$, $\hat{H}$, et $\hat{\tau}_f$ the estimated values of *M*, *H*, et τ_f respectively, *M* the inertia matrix, *H* the matrix of Coriolis, centrifugal and gravity terms and τ_f the friction torque.

In the case where the dynamic model is exact, equation (1.15) gives us the equation of error $e = (q_d - q)$.

$$\ddot{e} + K_D\dot{e} + K_P e + K_I\int_0^t e(x)dx = 0 \tag{1.16}$$

The error equation is decoupled and linear.

The right choice of constants K_p, K_D and K_I makes the error asymptotically close to zero.

From equation (1.16) we deduce the transfer function between the desired position and the actual measured position:

$$\frac{q_i}{q_{di}} = \frac{s^3 + K_{Di}s^2 + K_{Pi}s + K_{Ii}}{s^3 + K_{Di}s^2 + K_{Pi}s + K_{Ii}} \tag{1.17}$$

The transfer function (1.17) is unitary, so the trajectory of the robot must exactly follow the

modeling error trajectory.

The computation of the dynamic control depends on the task to be performed

- If the load is known, the identification is done offline.
- If the load is not known online identification is needed.

The control vector u is obtained by the anticipation of the desired acceleration and a decentralized PD corrector is added to reduce residual errors related to the modeling errors.

$$u = \ddot{q}_d + K_d \dot{e} + K_p e \tag{1.18}$$

By using the fact that $u = \ddot{q}$ in the perfect case, the behavior of the error is then characterized by the following equation:

$$\ddot{e} + K_d \dot{e} + K_p e = 0 \tag{1.19}$$

In this case, the error behaves like a second-order system. The proper pulsation and the damping are then regulated by the gain of the correctors:

$$\begin{cases} K_p = \omega^2 \\ K_d = 2\xi\omega \end{cases} \tag{1.20}$$

The presence of an integral gain is theoretically useless since the control system behaves like a double integrator. However, in practice, integral gain is used to reduce the modeling errors effects, since the calculated torque control giving by the non-linear decoupling control also tends to be weak compared to the modeling errors like the uncertainties of the inertial parameters and of the unknown loads or of the friction, in this case a PID corrector is necessary specially when the modeling errors are important.

The equation of the error will be given by the following relation:

$$\hat{M}(q)u + \hat{H}(q,\dot{q}) + \hat{\tau}_f = \hat{M}(q)\ddot{q} + \hat{H}(q,\dot{q}) + \tau_f$$

And

$$\ddot{e} + k_d \dot{e} + k_p e + k_i \int_0^t e(x)dx = \hat{M}^{-1}\left[(M - \hat{M})\ddot{q} + H(q,\dot{q}) - \hat{H}(q,\dot{q}) + \tau_f - \hat{\tau}_f\right]$$

Where

$$\ddot{e} + k_d \dot{e} + k_p e + k_i \int_0^t e(x)dx = \hat{M}^{-1}\tau_{prt} \tag{1.21}$$

τ_{prt} is the perturbation torque.

One deduces that the modeling error constitutes an excitation for the equation of the error '*e*'.

To remedy this problem is to increase the gains k_p, k_d and k_i.

1.4.3. Predictive dynamic control

This is the case of the dynamic control with non-linear decoupling, where during the computation, the prediction consists of replacingthe values of the measured velocity and the positions $(q, \dot{q})$, by their desired corresponding values (q_d, $\dot{q}_d$) when calculating the compensation terms $\hat{M}$, $\hat{H}$, $\hat{\tau}_f$. From the Figure 1.5.

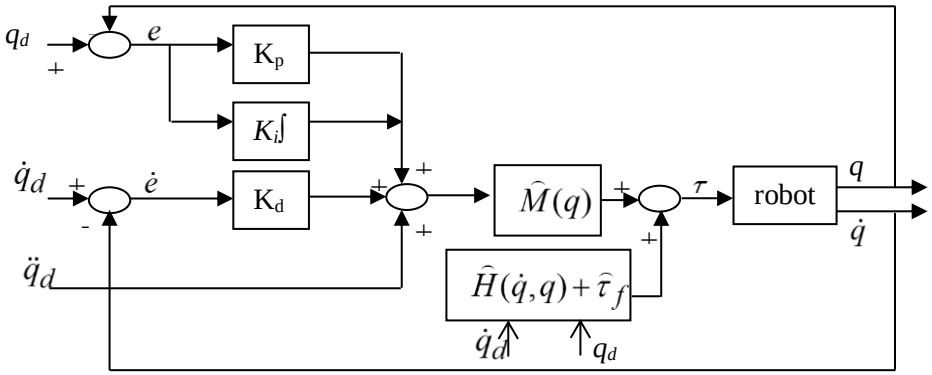


Figure 1.5 Predictive dynamic control.

The equation of the control law is given by:

$$\tau = \hat{M}(q)u + \hat{H}(q,\dot{q}) + \hat{\tau}_f(q) \tag{1.22}$$

Where $$u = \ddot{q}_d + K_D(\dot{q}_d - \dot{q}) + K_P(q_d - q) + K_I\int_0^t (q_d - q)dt \tag{1.23}$$

And the equation of error is of the form

$$\ddot{e} + K_D\dot{e} + K_P e + K_I\int_0^t e(x)dx = 0 \tag{1.24}$$

The error equation is linear and decoupled.

The advantage of this control resides in the possibility of computing off-line compensation terms $\hat{M}$, $\hat{H}$, $\hat{\tau}_f$ and store them, This greatly reduces online computing time.

Predictive control is an offline control technique so it is only useful when the trajectory to be followed is repeated often.

1.4.4. Adaptive control

The adaptive control automatically adjusts the regulator parameters in real time to achieve a certain level of performance when the dynamic behavior of the process is unknown, poorly known, and / or variable in time.

The closed-loop control of a system by an adaptive control is generally formed by two loops, a normal internal loop containing the regulator and the process, an external loop which adjusts the parameters of the regulators in order to minimize the chosen criterion.

1.4.4.1. *Different adaptive controls*

Although there are several types of adaptive controls, we present the most used ones:

a. ***Gain-scheduling***

It is possible sometimes to find auxiliary variables that have a great correlation with the change of the dynamic parameters. It is therefore possible to reduce the effects of the parameters adjusting the regulator based on these auxiliary variables.

Gain-scheduling regulation problem is the determination of the auxiliary parameters, which requires knowledge of the physics of the system to be controlled.

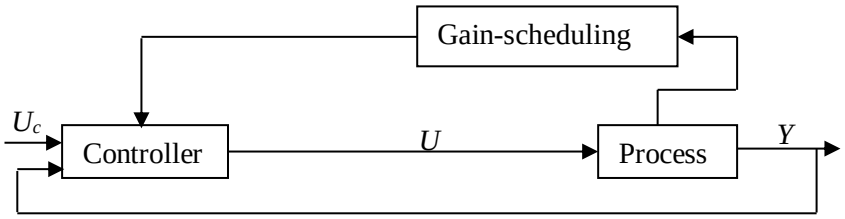


Figure 1.6 Block diagram of the gain-scheduling control.

b. ***Adaptive Control by Reference Model* (*MRAS*)**

The first *MRAS* was proposed by Whitaker in 1958 to solve the problem of servomechanisms. The original scheme proposed by Whitaker is shown in Figure 1.7.

The desired performances are formulated by the reference model which gives the desired response to the control signal.

The entire control system has a first ordinary loop containing the reference model and the regulator parameter adjustment mechanism.

The two new ideas brought by the *MRAS* are: the performances are fixed by the choice of a reference model and the adjustment of the parameters of the regulator is based on the error

$e = y_m - y$. Among the approaches used as adjustment mechanisms, we note:

- The rules MIT (gradient).
- The function of Lyapunov.
- The hyperstability of Popov.

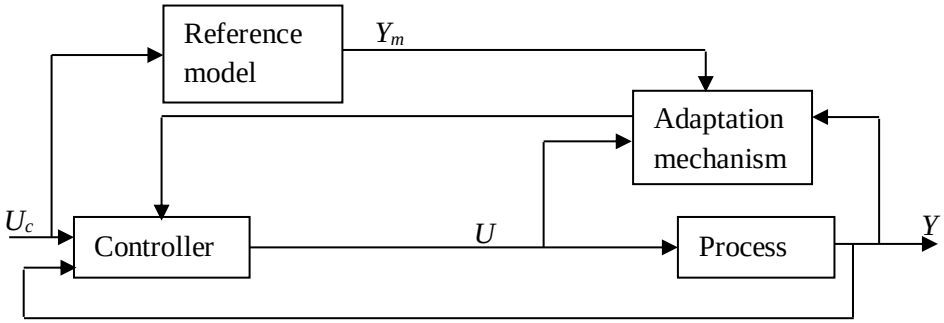


Figure 1.7 Synoptic diagram of an adaptive reference model system.

c. ***Self-adaptive regulator:***

In this case of adaptive systems, it is assumed that the regulator parameters are adjusted all the time and follow the process changes. However, it is difficult to analyze the convergence and stability of such systems.

The self-adaptive regulator is based on the idea of separating between the unknown parameter estimator and the adjustment procedure.

Figure 1.8 shows that unknown parameters are estimated online using recursive estimation methods.

The estimated parameters are assumed to take the actual values and the uncertainties of the

estimate are not considered.

Among the methods for estimating parameters are:

- Stochastic approximations.
- The least squares.
- Least squares extended and generalized.
- The instrumental variable.
- The maximum of Likelihood.

The methods of adjusting the parameters used are:

- The minimum variance.
- Thepoles placement.
- Model tracking.

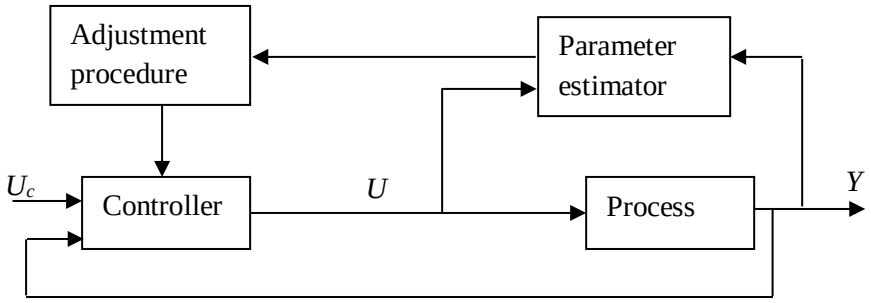


Figure 1.8 Block diagram of a self-adaptive regulator.

1.4.4.2. ***Approaches for manipulator arm analysis***

The construction of an adaptive reference model system consists of the:

1. Choice of a reference model that definesthe desired performance.
2. Choicee of the control law.
3. Determination of the adaptation law.

If the choice of the reference model and the control law is related to the desired performances, the adaptation law must ensure the stability of the system and the tendency of the error towards zero.

Among the approaches used for the determination of the adaptation law are the gradient approach, the Lyapunov function and the Popov hyperstability.

a. ***Gradient approach***

The gradient method is first used by Whitaker in his original work. The development of this approach towards the *MRAS* returns to using the rules of *MIT*.

b. ***MIT Rule***

Let the error be between y_m and y ($e=y_m-y$) and θ be the vector of the parameters to be adjusted. A criterion to be minimized is proposed as:

$$J(\theta)=\frac{1}{2}e^2 \tag{1.25}$$

Therefore, for *J* to be small it is reasonable to change the parameters in the negative direction of the *J* gradient i.e :

$$\frac{d\theta}{dt} = -\gamma \frac{\delta J}{\delta \theta} \tag{1.26}$$

$$\frac{d\theta}{dt} = -\gamma . e . \frac{\delta e}{\delta \theta}$$

If it is assumed that the parameters θ change more slowly than the other variables of the system, then one can compute $\frac{\delta e}{\delta \theta}$ in this case assuming constant θ. $\frac{\delta e}{\delta \theta}$ represents the sensitivity of the system and γ determines the speed of parameters adaptation.

The diagram in Figure 1.9 represents the error model:

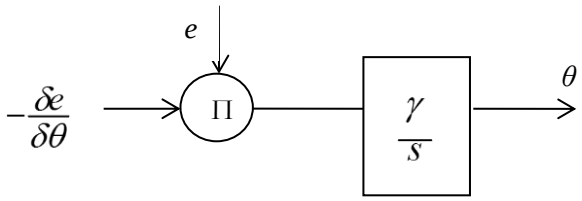


Figure 1.9 Error model

The choice of the criterion is arbitrary, if one poses:

$$J = e \tag{1.27}$$

$$\frac{\delta e}{\delta \theta} = -\gamma \frac{\delta e}{\delta \theta} sign(e) \tag{1.28}$$

The rule of *MIT* is powerful if γ is chosen small, but its value may depend on the amplitude of the signal and the gain of the process. As a result, it is not possible to give γ a limit that ensures the overall stability of the system. The system may be stable for certain values and not for others.

A way to overcome the previous problem is to use the modified MIT rules given by the formula.

$$\frac{d\theta}{dt} = -\gamma \frac{e \frac{\delta e}{\delta \theta}}{\alpha + (\frac{\delta e}{\delta \theta})^T (\frac{\delta e}{\delta \theta})} \tag{1.29}$$

Another way to limit the speed of convergence is to introduce the saturation function:

$$\frac{d\theta}{dt} = -\gamma . sat\left[\frac{e \frac{\delta e}{\delta \theta}}{\alpha + (\frac{\delta e}{\delta \theta})^T (\frac{\delta e}{\delta \theta})} . \beta \right] \tag{1.30}$$

Sometimes the choice of wide values for *y* causes instability.

For all these reasons, the choice of an approach that is based on stability seems better.

c. ***Lyapunov function***

The extensive research conducted to find a stability analysis method, which ensures the tendency of the error to zero could give the Lyapunov function.

The method proposed by Lyapunov is valid for nonlinear systems and whose idea is illustrated in Figure 1.10.

The statement of the method is as follows: the equilibrium is stable if one finds a real function of the state space, its curve envelopes the state of equilibrium and the derivative of the state variable points to the inside the curve.

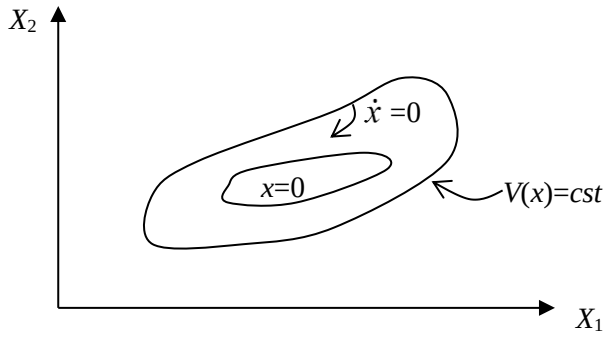


Figure 1.10 Illustration of Lyapunov's method

To formally state the results, we consider the following differential equation:

$$\dot{x} = f(x,t) \ , \qquad f(0,t)=0 \tag{1.31}$$

Where*x*is a state vector of *n*dimension.

assumed the function $V : R^{n+1} \to R$ Satisfying the following conditions:

- $V(0, t) = 0$ for all $t \in R$.
- *V* is differentiable at*x*and*t*.
- *V* is positive definite.

The sufficient condition for uniform asymptotic stability for the (1.31) system is:

$$\dot{V}(x,t) = F^T(x,t)grad(V) + \frac{\delta V}{\delta t} \ (\ 0 \text{ for } x \neq 0) \tag{1.32}$$

So $\dot{V}$ must be defined negative:

Condition (1.32) will therefore fix the adaptation law thus ensuring the stability and tendency of the error towards zero.

1.5. Mobile robots

1.5.1. Definition of a mobile robot

A mobile robot is a mechanical, electronic and computer system acting physically on its environment in order to achieve a goal that has been assigned to it. This machine is versatile and able to adapt to certain variations of its operating conditions. It has perception, decision and action functions. Thus, the robot should be able to perform various tasks, in several ways, and perform its task correctly, even if it encounters new unexpected situations.

Currently, the most sophisticated mobile robots are primarily oriented towards applications in variable or uncertain environments, often with obstacles, requiring adaptability to the task.

1.5.2. Wheeled mobile robots

Wheel mobility is the most used mechanical structure. This type of robot provides a displacement with a fast movement but requires relatively flat ground. We distinguish several

classes of wheeled robots, mainly by the position and the number of wheels used. We will mention here the four main classes of wheeled robots [32].

1.5.3. Classification of wheeled mobile robots

1.5.3.1. ***Classification according to the type of wheels***

In the field of mobile robotics, we find four types of wheels as shown in the Figure 1.11[33]:

- Fixed wheels whose axis of rotation passes through the center of the wheels.

- Orientable center wheels, whose axis of orientation passes through the center of the wheels.

- Foolish wheels whose orientation axis does not pass through the center of the wheel.

- Swedish wheels for which the zero component of the sliding speed at the point of contact is not in the plane of the wheel.

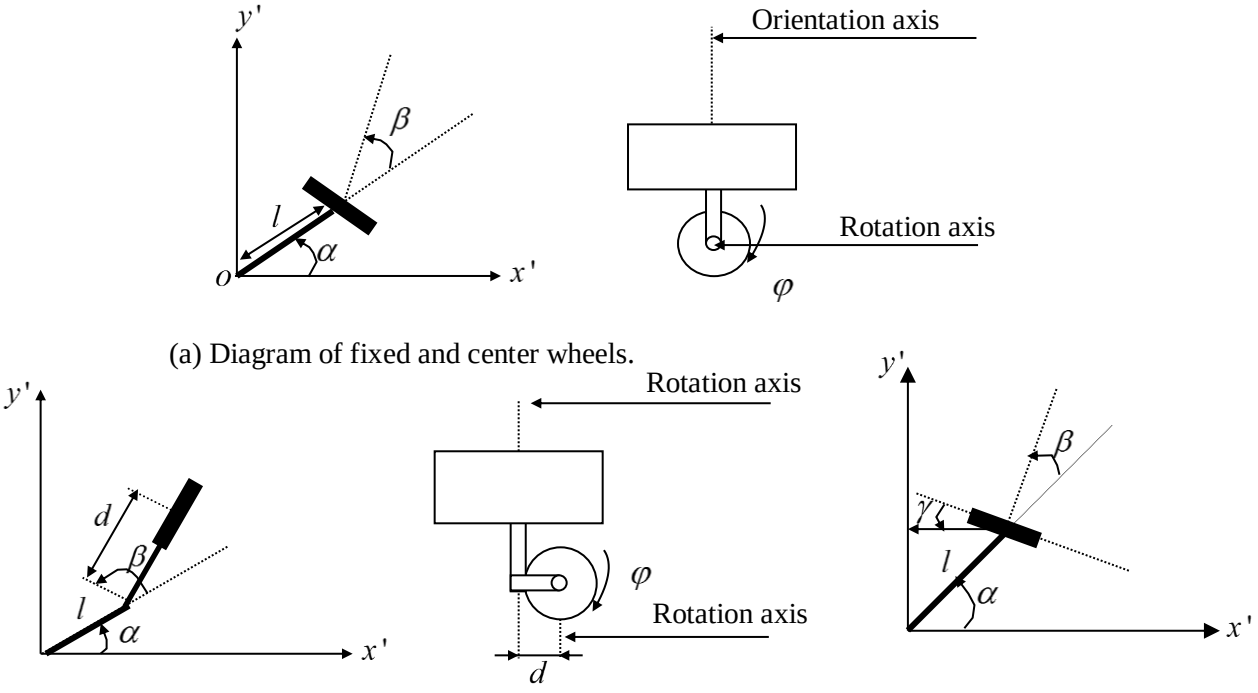


Figure 1.11 Diagram of the different wheels types of the mobile platforms

1.5.4. Kinematic modeling

1.5.4.1. ***Wheel model***

Kinematic modeling is the study of the motion of a mechanical system regardless of the forces that influence its motion. At this stage, we are interested only in velocity vectors.

1.5.4.2. ***Modeling hypothesis***

The problem of the control of mobile robots being too vast to be presented exhaustively, in this part a number of simplifying hypotheses:

- The robots are considered rigid and evolving on a plane,

- The robots are equipped with conventional wheels: the point of contact between the wheel and the ground is reduced to a point and the wheel is subjected to rolling constraint without sliding.

1.5.4.3. ***Rolling without slip and non-holonomy***

Many mechanical systems are subject to position and/or velocity constraints, i.e., several relationships between the positions and/or velocities of different points of the system must be satisfied throughout the motion. These constraints are said to be holonomic if it is possible to integrate them and they lead to algebraic relations linking the configuration parameters. These relationships can be eliminated by appropriate change of variables and the system is said to be holonomic. In other words, the robot ability to move from a position to any direction is called holonomic. In the case of non-integrable constraints, the elimination is no longer possible and the system is said to be non-holonomic.

In the case of wheeled robots, these kinematic constraints result from the assumption of rolling without sliding. Consider a vertical wheel that rolls without sliding on a flat ground, the rolling without sliding translates to the zero velocity of point *I* of the wheel in contact with the ground [34].

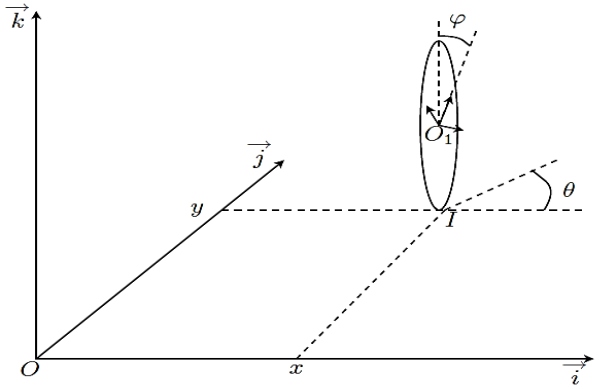


Figure 1.12 Description of a wheel

With the notation in Figure 1.12, we obtain:

$$\vec{v}(I/R_0) = \dot{x}\vec{i} + \dot{y}\vec{j} + (\dot{\theta}\vec{k} + \dot{\varphi}(-\sin\theta\vec{i} + \cos\theta\vec{j}))(-r\vec{k})$$
$$= (\dot{x} - r\dot{\varphi}\cos\theta)\vec{i} + (\dot{y} - r\dot{\varphi}\sin\theta)\vec{j} = 0$$

Where r is the radius of the wheel and (x, y) is the coordinate of the point o_1 in the fixed reference $R_0 = (o, \vec{i}, \vec{j}, \vec{k})$. We deduce two constraints:

$$\begin{cases} \dot{x} - r\dot{\varphi}\cos\theta = 0 \\ \dot{y} - r\dot{\varphi}\sin\theta = 0 \end{cases} \tag{1.33}$$

The model (1.33) can be transformed to make appear the components of the velocities in the planes of the wheel and perpendicular to the wheel, the following kinematic constraints are then obtained:

$$\begin{cases} \dot{x}\cos\theta + \dot{y}\sin\theta = r\dot{\varphi} \\ -\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \end{cases}$$

It is interesting to note that by introducing $v = r\dot{\varphi}$ the rolling speed of the wheel and $w = \dot{\theta}$ its speed of rotation around the axis $\vec{k}$, we form the following model:

$$\dot{q} = S(q)u \Rightarrow \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}, \quad u = \begin{bmatrix} v \\ w \end{bmatrix} \tag{1.34}$$

1.5.4.4. ***Kinematic model of unicycle wheel robots (differential)***

The kinematic model (1.34) of the wheel subjected to rolling without sliding constraints have certain characteristics. One of the characteristics of these non-holonomic systems is that the linearized system is not controllable whereas the real system is.

We consider the mobile robot type unicycle schematically in Figure 1.13, this robot is equipped with two independently controlled fixed-wheel drive and foolish wheels ensuring its stability [34].

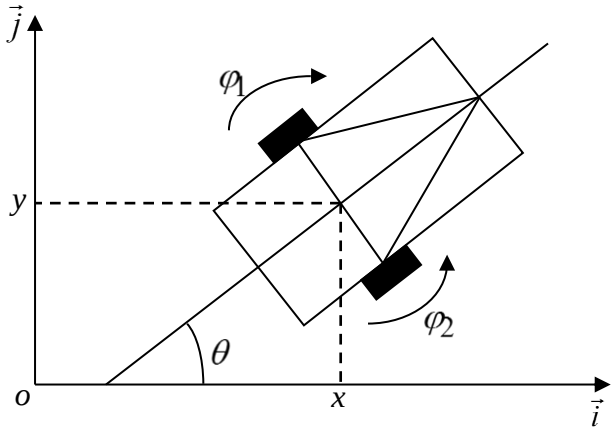


Figure 1.13 UnicycleRobot type

Let the abscissa and the ordinate (x, y) of the middle of the axis of the two driving wheels, θ the orientation of the robot, r the radius of the wheels and $2R$ the distance between the two drive wheels. By taking again the equation (1.33) and one can easily show that the constraints of rolling without sliding of each of the driving wheels are written:

– For the left wheel

$$\dot{x} - R\dot{\theta}\cos\theta - r\dot{\varphi}_1\cos\theta = 0 \tag{1.35}$$

$$\dot{y} - R\dot{\theta}\sin\theta - r\dot{\varphi}_1\sin\theta = 0 \tag{1.36}$$

– For the right wheel

$$\dot{x} - R\dot{\theta}\cos\theta - r\dot{\varphi}_2\cos\theta = 0 \tag{1.37}$$

$$\dot{y} - R\dot{\theta}\sin\theta - r\dot{\varphi}_2\sin\theta = 0 \tag{1.38}$$

These four constraints are not independent since the difference of the left-hand side of equalities (1.35) and (1.37) is proportional to that associated with (1.36) and (1.38). For example, the last constraint can be omitted. In addition, the constraint is completely integrable. Indeed, we have:

$$\begin{cases} \dot{x}\cos\theta + \dot{y}\sin\theta - R\dot{\theta} = r\dot{\varphi}_1 \\ \dot{x}\cos\theta + \dot{y}\sin\theta + R\dot{\theta} = r\dot{\varphi}_2 \end{cases}$$

So, do one gets:

$$2R\dot{\theta} = r(\dot{\varphi}_2 - \dot{\varphi}_1)$$

Which implies:

$$2R\theta = r(\varphi_2 - \varphi_1) + \text{constant}$$

Consequently, there remain only two independent constraints, (1.35) and (1.36), that are written as:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = w \end{cases} \tag{1.39}$$

Where

– The linear speed of the robot is

$$v = \frac{r}{2}(\dot{\varphi}_1 - \dot{\varphi}_2) \tag{1.40}$$

– The angular velocity of the robot is

$$w = \frac{r}{2R}(\dot{\varphi}_2 - \dot{\varphi}_1) \tag{1.41}$$

The fact that this system is the same as the model (1.34) obtained for a single wheel (unicycle), which justifies the qualifier term unicycle often used in the literature.

1.5.5. Dynamic model of unicycle wheeled robots

By ignoring the mass of the wheels, the equations of motion can be derived using Euler-Lagrange method as

$$M(q)\ddot{q} + \overline{C}(q,\dot{q})\dot{q} = E(q)\tau - C^T(q)\lambda \tag{1.42}$$

Where

$$M = \begin{bmatrix} m & 0 & -dm\sin\theta \\ 0 & m & dm\cos\theta \\ -dm\sin\theta & dm\cos\theta & d^2m + I \end{bmatrix},\ \tau = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}$$

$$\overline{C} = \begin{bmatrix} 0 & 0 & -dm\dot{\theta}\cos\theta \\ 0 & 0 & -dm\dot{\theta}\sin\theta \\ 0 & 0 & 0 \end{bmatrix},\quad E = \frac{1}{r}\begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ b & -b \end{bmatrix}$$

Where m is the robot mass, I the moment of inertia of the robot about its center of mass,

d the distance between the center of mass and the center of the wheel axle, *r* the wheel radius *R* half distance between the two wheels, τ_l, τ_r the motor torques on the wheels and λ the constraint force.

By differentiating (1.34), one gets $\ddot{q} = \dot{S}u + S\dot{u}$. Then by substituting $\ddot{q}$ into (1.42), pre-multiplying by S^T, and using the property $S^T C^T = 0$, one can have

$$\dot{u} = -(S^T MS)^{-1} S^T (M\dot{S} + \overline{C}S)u + (S^T MS)^{-1} S^T E\tau \tag{1.43}$$

1.6. Properties and Control of a mobile robot

1.6.1. Definitions, Properties and Linearization

Definition 1.1 : *Lie Derivative* [35]

Let $h : D \subset \Re^n \to \Re$, $f : D \subset \Re^n \to \Re^n$ then the Lie derivative of *h*, with respect to the vector field *f* is given as

$$L_f h(x) = \frac{\partial h}{\partial x} f(x)$$

Used earlier to find $\dot{V}(x)$

$$L_g L_f h(x) = \frac{\partial L_f h(x)}{\partial x} g(x)$$

$$L_f L_f h(x) = L_f^2 h(x) = \frac{\partial L_f h(x)}{\partial x} f(x)$$

Example

$$h(x) = \frac{1}{2}(x_1^2 + x_2^2)\ , \quad f(x) = \begin{bmatrix} x_2 \\ -x_1 - \mu x_2 (1 - x_1^2) \end{bmatrix}, \quad g(x) = \begin{bmatrix} -x_1 - x_1 x_2^2 \\ -x_2 + x_1^2 x_2 \end{bmatrix}$$

$$L_f h(x) = \frac{\partial h}{\partial x} f(x) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} x_2 \\ -x_1 - \mu x_2 (1 - x_1^2) \end{bmatrix} = -\mu x_2^2 (1 - x_1^2)$$

$$L_g h(x) = \frac{\partial h}{\partial x} g(x) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} -x_1 - x_1 x_2^2 \\ -x_2 + x_1^2 x_2 \end{bmatrix} = -(x_1^2 + x_2^2)$$

Second-order Derivatives

$$L_g h(x) = -(x_1^2 + x_2^2)$$

$$L_g^2 h(x) = \frac{\partial L_g h(x)}{\partial x} g(x) = -2 \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} -x_1 - x_1 x_2^2 \\ -x_2 + x_2 x_1^2 \end{bmatrix} = 2(x_1^2 + x_2^2)$$

$$L_f L_g h(x) = \frac{\partial L_g h(x)}{\partial x} f(x) = -2 \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} x_2 \\ -x_1 - \mu x_2 (1 - x_1^2) \end{bmatrix} = 2\mu x_2^2 (1 - x_1^2)$$

Definition 1.2 : *Lie Bracket* [35]

Let the vectors fields $f, g : D \subset \Re^n \to \Re^n$ then the Lie Bracket of *f* and *g* is found to be

$$[f, g](x) = \frac{\partial g}{\partial x} f(x) - \frac{\partial f}{\partial x} g(x)$$

Example

$$f(x) = \begin{bmatrix} -x_2 \\ -x_1 - \mu x_2 (1 - x_1^2) \end{bmatrix}, \quad g(x) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$[f, g](x) = \frac{\partial g}{\partial x} f(x) - \frac{\partial f}{\partial x} g(x) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -x_2 \\ -x_1 - \mu x_2 (1 - x_1^2) \end{bmatrix} - \begin{bmatrix} 0 & -1 \\ -1 + 2\mu x_1 x_2 & -\mu(1 - x_1^2) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ -2\mu x_1^2 x_2 \end{bmatrix}$$

Definition 1.3 : *Diffeomorphism* [35]

Let $f : D \subset \Re^n \to \Re^n$, is said to bea diffeomorphism on *D* (local diffeomorphism)if:

1- *f* is continuously differentiable on *D*

2- f^{-1} (inverse) existand continuously differentiable, such that

$$f^{-1}(f(x)) = x, \forall x \in D$$

The function *f* is said to be a global diffeomorphism if in addition

1 - $D = \Re^n$,

2 - $\ell im_{\|x\| \to \infty} \|f(x)\| = \infty$

Lemma

Let $f : D \subset \Re^n \to \Re^n$

•Be continuously differentiable on *D*, and

• If the Jacobian matrix $Df = \nabla f$ is nonsingular at a point $x_0 \in D$.

Then *f* is a diffeomorphism in subset $\Omega \subset D$.

Definition 1.4 : *Distribution* (*Differential Geometry*) [35]

A finite set of vectors $S = \{x_1, x_2, ...x_p\}$ in $\Re^n$ is said to be linearly dependent if there exist a corresponding set of real numbers $\{\lambda_i\}$, not all zero, such that

$$\sum_i \lambda_i x_i = \lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_p x_p = 0$$

On the other hand, if $\sum_i \lambda_i x_i = 0$ implies that $\lambda_i = 0$ for each *i*, then the set $\{x_i\}$ is said to be linearly independent.

Given a set of vectors $S = \{x_1, x_2, ...x_p\}$ in $\Re^n$, a linear combination of those vector defines a new vector $x \in \Re^n$, that is, given real numbers $\lambda_1, \lambda_2, ...\lambda_p$.

$$x = \lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_p x_p \in \quad \Re^n$$

The set of *all* linear combinations of vectors in *S* generates a subspace *M* of $\Re^n$ known as the span of *S* and denoted by $span(S) = span\{x_1, x_2, ...x_p\}$ i.e.

$$span\{x_1, x_2, ...x_p\} = \{x \in \Re^n : x = \lambda_1 x_1 + \lambda_2 x_2 + ... + \lambda_p x_p, \quad \lambda_i \in \Re\}$$

The concept of distribution is somewhat related to this concept.

Now consider a differentiable function $f : D \subset \Re^n \to \Re^n$ .As we well know, this function can be interpreted as a vector field that assigns the *n*-dimensional vector *f* (*x*) to each point $x \in D$ .Now consider "*p*" vector fields $f_1, f_2, ... f_p$ on $D \subset \Re^n$. At any fixed point $x \in D$ the functions f_i generate vectors $f_1(x), f_2(x), ... f_p(x) \in \Re^n$ and thus

$$\Delta(x) = span\{f_1(x), f_2(x), ... f_p(x)\}$$

Definition 1.5 : *Nonsingular Distribution* [35]

A distribution Δ defined on $D \subset \Re^n$ is said to be nonsingular if there exists an integer *d* such that

$$\dim(\Delta(x)) = d \ \forall x \in D.$$

If this condition is not satisfied, then Δ is said to be of variable dimension.

Definition 1.6 : *InvolutiveDistribution* [35]

A distribution Δ is said to be involutive if: $\forall g_i \in \Delta, i = 1,2 \Rightarrow [g_1, g_2] \in \Delta$

It then follows that $\Delta(x) = span\{f_i(x), i = 1,....., p\}$ is involutive if and only if

$$rank\lfloor f_1(x), f_2(x), ... f_p(x) \rfloor = rank\lfloor f_1(x), f_2(x), ... f_p(x), \lfloor f_i(x), f_j(x) \rfloor \rfloor \forall x \text{ and all } i, j$$

Example

Let $D = \Re^3, \quad \Delta(x) = span\{f_1(x), f_2(x)\}, \quad$ with $f_1(x) = \begin{bmatrix} 1 \\ 0 \\ x_1^2 \end{bmatrix}, \quad f_2(x) = \begin{bmatrix} 0 \\ x_1 \\ 1 \end{bmatrix}$ then

$$[f_1, f_2] = \frac{\partial f_2}{\partial x} f_1(x) - \frac{\partial f_1}{\partial x} f_2(x) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \text{ and } rank[f_1, f_2] = 2$$

$$rank\lfloor f_1, f_2, \lfloor f_i(x), f_j(x) \rfloor \rfloor = 3, \forall x, i\ j \ \Delta(x) \text{ is not involutive}$$

Definition 1.7 :*Complete Integrability* [35]

A linearly independent set of vector fields $f_1, f_2, ... f_p$ on $D \subset \Re^n$ is said to be completely integrable if for each $x_0 \in D$ there exists a neighborhood *N* of x_o and *n-p* real-valued smooth functions $h_1(x), h_2(x), ...h_{n-p}(x)$ satisfying the partial differentiable equation.

$$\frac{\partial h_j}{\partial x} f_i(x) = 0, \quad 1 \le i \le p, \quad 1 \le j \le n - p$$

And the gradients ∇h_i are linearly independent.

1.6.1.1. ***Frobenius Theorem*** [35]

A set of linearly independent vector fields $\{f_i(x), i = 1,....., p\}$ be a set of linearly independent vector fields, is completely integrable if and only if it is involutive.

Example

Let the partial differential equations be

$$2x_3 \frac{\partial h}{\partial x_1} - \frac{\partial h}{\partial x_2} = 0$$

$$-x_1 \frac{\partial h}{\partial x_1} - 2x_2 \frac{\partial h}{\partial x_2} + x_3 \frac{\partial h}{\partial x_3} = 0$$

Which can be rewrite as

$$\begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} & \frac{\partial h}{\partial x_3} \end{bmatrix} \begin{bmatrix} 2x_3 & -x_1 \\ -1 & -2x_2 \\ 0 & x_3 \end{bmatrix}$$

Or $\nabla h [f_1 \quad f_2], \qquad f_1(x) = \begin{bmatrix} 2x_3 \\ -1 \\ 0 \end{bmatrix}, \quad f_2(x) = \begin{bmatrix} -x_1 \\ -2x_2 \\ x_3 \end{bmatrix}$

The Lie derivatives :

$$L_{fi} h(x) = \frac{\partial h}{\partial x} f_i(x) = 0, \quad i = 1,2$$

$$\begin{bmatrix} \frac{\partial h}{\partial x_1} & \frac{\partial h}{\partial x_2} & \frac{\partial h}{\partial x_3} \end{bmatrix} \begin{bmatrix} 2x_3 & -x_1 \\ -1 & -2x_2 \\ 0 & x_3 \end{bmatrix}$$

To determine whether the set of partial differential equations is solvable or, equivalently, whether $[f_1, f_2]$ is completely integrable, we consider the distribution Δ defined as follows:

$$\Delta(x) = span\{f_1(x), f_2(x)\}$$

It can be checked that Δ has dimension 2 everywhere on the set D defined by $D = \{x \in \Re^3 : x_1^2 + x_2^2 \neq 0\}$. Computing the Lie bracket $[f_1, f_2]$, we obtain

$$[f_1, f_2] = \frac{\partial f_2}{\partial x} f_1(x) - \frac{\partial f_1}{\partial x} f_2(x)$$

$$\frac{\partial f_2}{\partial x}=\begin{bmatrix}-1 & 0 & 0\\ 0 & -2 & 0\\ 0 & 0 & 1\end{bmatrix},\quad \frac{\partial f_1}{\partial x}=\begin{bmatrix}0 & 0 & 2\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$$

$$[f_1,\ f_2]=\begin{bmatrix}-4x_3\\ 0\\ 0\end{bmatrix}$$

And thus $$[f_1(x),\ f_2(x),[f_1(x),\ f_2(x)]]=\begin{bmatrix}2x_3 & -x_1 & -4x_3\\ -1 & -2x_2 & 2\\ 0 & x_3 & 0\end{bmatrix}$$

It follows that the distribution is involutiveif $\forall x \in D$

$$rank[f_1(x),\ f_2(x)\ ]=rank[f_1(x),\ f_2(x),[f_1(x),\ f_2(x)]]$$

$$rank\begin{bmatrix}2x_3 & -x_1\\ -1 & -2x_2\\ 0 & x_3\end{bmatrix}=2,\quad \forall x\in D,\qquad rank\begin{bmatrix}2x_3 & -x_1 & -4x_3\\ -1 & -2x_2 & 2\\ 0 & x_3 & 0\end{bmatrix}=2,\quad \forall x\in D$$

Which has rank 2 for all $x \in \Re^3$.And thus it is completely integrable on D, by the Frobenius Theorem.

Definition 1.8: *linearization about an equilibrium point* (u_0, x_0, y_0) [36]

$$\dot{x}=f(x)+g(x)u$$
$$y=h(x)$$

Linearization about an equilibrium point (u_0, x_0, y_0)

$$\dot{x}=\left[\frac{\partial f(x_0)}{\partial x}+\frac{\partial g(x_0)}{\partial x}u_0\right](x-x_0)+g(x_0)(u-u_0)$$
$$y-y_0=\frac{\partial h(x_0)}{\partial x}(x-x_0)$$

1.6.1.2. ***Feedback Linearization*** [37]

The main idea is to transform algebraically a nonlinear system dynamics into a (totally or partially) linear dynamic, so that linear control techniques can be applied.

In its simplest form, feedback linearization cancels the nonlinearities in a nonlinear system, so closed-loop dynamics is linear.

Example 1: Consider the control of the level h of fluid in a tank to a specified level h_0.

The control input is the flow u into the tank, and the initial level is h_0.

Consider the dynamics model of the tank

$$A(h)\dot{h}(t)=u-a\sqrt{2gh}$$

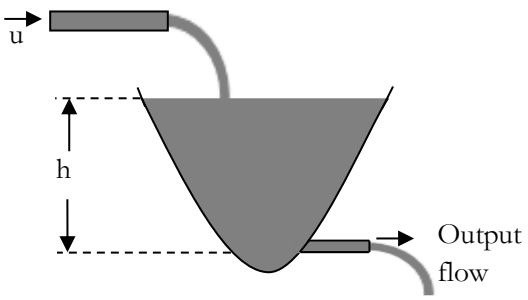


Figure 1.14 Fluid control in a tank

Where $A(h)$ is the cross section of the tank and a is the cross section of the outlet pipe.

Choose $u = a\sqrt{2gh} + A(h)v$ *where* $\dot{h} = v$ and choose the equivalent input $v : v = -\alpha\tilde{h}$ *where* $\tilde{h} = h(t) - h_d$ is error level, α a positive constant. The resulting closed-loop dynamics: $\dot{h} + \alpha\tilde{h} = 0 \Rightarrow \tilde{h} \to 0 \; as \; t \to \infty$

The actual input flow: $u = a\sqrt{2gh} - A(h)\alpha\tilde{h}$

The first term provides an output flow $a\sqrt{2gh}$ and the second term controls the fluid level according to the desired linear dynamics

If h_d is time-varying $v = \dot{h}_d(t) - \alpha\tilde{h}$, $\tilde{h} \to 0 \; as \; t \to \infty$. The idea of feedback linearization, *i.e.,* of canceling the nonlinearities and imposing a desired linear dynamics, can be simply applied to a class of nonlinear systems described by the so-called *companionform,* or *controllability canonical form.*

A system is said to be in companion form if its dynamics is represented by:

$$x^n(t) = f(x) + b(x)u$$

Where u is the scalar control input, x is the scalar output of interest, $x = \left[x, \dot{x}, ..., x^{(n-1)}\right]$, is the state vector, and $f(x)$ and $b(x)$ are nonlinear functions of the states. This form is unique in the fact that, although derivatives of x appear in this equation, no derivative of the input u is present.

Note that, in state-space latter representation, can be written

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} x_2 \\ \vdots \\ x_n \\ f(x) + b(x)u \end{bmatrix}$$

For systems which can be expressed in the controllability canonical form, using the control input $b \neq 0$, the control input: $u = \frac{1}{b}\left[v - f\right]$

The control law: $v = -k_0 x - k_1 \dot{x} - \cdots - k_{n-1} x^{(n-1)}$

K_i is chosen such thatthe roots of $s^n + k_{n-1}s^{n-1} + \ldots + k_0$ are strictly in left-half plane (*LHP*). Thus : $x^n + k_{n-1}x^{n-1} + \ldots + k_0 = 0$ is Exponentially stable (e.s).

For tracking desired output x_d, the control law is:

$$v = x_d^{(n)} - k_0 e - k_1 \dot{e} - \cdots - k_{n-1} e^{(n-1)}$$

Exponentially convergent tracking, $e = x - x_d \to 0$

This method is extendable when the scalar x was replaced by a vector and the scalar b by an invertible square matrix.

When u is replaced by an invertible function $g(u)$, $u = g^{-1}(\frac{1}{b}[v - f])$

Example 2: A two-link robot [37]:

Objective: the joint positions q_1 and q_2 follow the desired position $q_{d1}(t)$ and $q_{d2}(t)$.

Using the Lagrangian equations in classical dynamics, one can easily show that the dynamic equations of the robot is:

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \ddot{q} + \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \dot{q} + \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

Control law for tracking, (computed torque):

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \dot{q} + \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$$

Where $v = \ddot{q}_d - 2\lambda\tilde{\dot{q}} - \lambda^2\tilde{q}$, $\tilde{q} = q - q_d$ is the position tracking error, λ a positive constant.

$\ddot{\tilde{q}} + 2\lambda\dot{\tilde{q}} + \lambda^2\tilde{q} = 0$, where $\tilde{q}$ converge to zero exponentially.

1.6.1.3. ***Input-state Linearization*** [37]

When the nonlinear dynamics is not in a controllability canonical form, one uses algebraic transformations.

Consider the SISO system

$$\dot{x} = f(x) + G(x)u$$

The technique of input-state linearization solves this problem in two steps

1. Finds a state transformation $z = T(x)$ and an input transformation $u=u(x, v)$ so that the nonlinear system dynamics is transformed into an equivalent *linear time-invariant* dynamics, in the familiar form $\dot{z} = Az + Bv$

2. One uses standard linear techniques (such as pole placement) to design v

$$\dot{z} = Az + Bv$$
$$v = W(x)\{u - \phi(x)\}$$

Let us illustrate the approach on a simple second-order example. Consider the system

$$\dot{x}_1 = -2x_1 + ax_2 + \sin x_1$$
$$\dot{x}_2 = -x_2 \cos x_1 + u\cos(2x_1)$$

Even though linear control design can stabilize the system in a small region around the equilibrium point (0, 0), it is not obvious at all what controller can stabilize it in a larger region.

A specific difficulty is the nonlinearity in the first equation, which cannot be directly

canceled by the control input *u*.

However, if we consider the new set of state variables

$$z_1 = x_1$$
$$z_2 = ax_2 + \sin x_1$$

Then, the new state equations are

$$\dot{z}_1 = -2z_1 + z_2$$
$$\dot{z}_2 = \frac{-z_2 \cos z_1 + \cos z_1 \sin z_1 + au\cos(2z_1)}{a}$$

Note that the new state equations also have an equilibrium point at (0, 0). Now we see that the nonlinearities can be canceled by the control law of the form

$$u = \frac{1}{a\cos(2z_1)}(av - \cos z_1 \sin z_1 + z_2 \cos z_1)$$

Where*v* is an equivalent input to be designed (equivalent in the sense that determining *v* amounts to determining *u* and vice versa), leading to a linear input-state relation

$$\dot{z}_1 = -2z_1 + z_2$$
$$\dot{z}_2 = v$$

Since the new dynamics is linear and controllable, it is well known that the linear state feedback control law

$$v = -k_1 z_1 - k_2 z_2$$

When both z_1 and z_2 converge to zero, so the original state *x* converges to zero.

1.6.1.4. ***Input-Output Linearization*** [37]

Consider the system

$$\dot{x} = f(x,u)$$
$$y = h(x)$$

The objective is the tracking of a desired trajectory $y_d(t)$, while keeping the whole state bounded, $y_d(t)$ and its time derivatives up to a sufficiently high order are known and bounded.

The difficulty is that the output *y* is only *indirectly* related to the input *u*, it is not easy to see how the input *u* can be designed to control the tracking behavior of the output *y*.

Input-output linearization approach:

1. Generating a linear input-output relation
2. Formulating a controller based on linear control

Example

$$\dot{x}_1 = \sin x_2 + (x_2 + 1)x_3$$
$$\dot{x}_2 = x_1^5 + x_3$$
$$\dot{x}_3 = x_1^2 + u$$
$$y = x_1$$

To generate a direct relationship between the output *y* and the input *u*, differentiate the output

$$\dot{y} = \dot{x}_1 = \sin x_2 + (x_2 + 1)x_3$$

Since $\dot{y}$ is still not directly related to the input *u*, let us differentiate again. We now obtain:

$\ddot{y} = (x_2 + 1)u + f(x)$, where

$$f(x) = (x_1^5 + x_3)(x_3 + \cos x_2) + (x_2 + 1)x_1^2$$

Control input law

$$u = \frac{1}{x_2 + 1}(v - f)$$

Choose $v = \ddot{y}_d - k_1 e - k_2 \dot{e}$, where $e = y - y_d$ the tracking error and, k_1 and k_2 are constants positive.The closed-loop system: $\ddot{e} + k_2 \dot{e} + k_1 e = 0$

The control law is defined everywhere except at singularity points such that $x_2 = -1$.

To implement the control law, the complete state measurement is necessary, because the computations of both the derivative *y* and the input transformation require the value of *x*.

If the output of a system should be differentiated *r* times to generate an explicit relation between *y* and *u*, the system is said to have *relative degree r*.

For any controllable system of order *n*, by taking at most *n* differentiations, the control input will appear to any output, i.e., $r \leq n$.

If the control input never appears after more than *n* differentiations, the system would not be controllable.

1.6.1.5. ***Feedback Linearization* (*Internal dynamics*)** [37]

A part of the system dynamics (described by one state component) has been rendered "unobservable" in the input-output linearization. This part of the dynamics will be called the *internal dynamics*, because it cannot be seen from the external input output relationship ($\ddot{y}$). For the above example, the internal state can be chosen to be x_3 (because x_3, y, and $\dot{y}$ constitute a new set of states), and the internal dynamics is represented by the equation

$$\dot{x}_3 = x_1^2 + \frac{1}{x_2 + 1}(\ddot{y}_d(t) - k_1 e - k_2 \dot{e} - f)$$

The desired performance of the control based on the reduced order model depends on the stability of the internal dynamics.

Example: Consider the system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2^3 + u \\ u \end{bmatrix}$$

$$y = x_1$$

Assume that the control objective is to make *y* track $y_d(t)$. Differentiation of *y* simply leads to the first state equation. Thus, choosing the control lawyields exponential convergence of *e* to zero $\dot{e} + e = 0$

The same control input is also applied to the second dynamic equation, leading to the internal dynamics: $\dot{x}_2 + x_2^3 = \dot{y}_d - e$

Since *e* and $\dot{y}_d$ are bounded $(\dot{y}_d(t) - e \leq D)$, x_2 is ultimately bounded.

- I/O linearization can also be applied to stabilization ($y_d(t) \equiv 0$):

- For previous example the objective will be y and $\dot{y}$ driven to zero and stable internal dynamics guarantee stability of the whole system.
- No restriction to choose physically meaningful $h(x)$ in $y = h(x)$.
- Different choices of output function lead to different internal dynamics which some of them may be unstable.
- When the relative degree of a system is the same as its order:
  -There is no internal dynamics.
  - The problem will be input-state linearization.

1.6.1.6. ***Controllability about a Trajectory***

The nonlinear model (1.34) possesses, from the point of control view, two fundamental properties [34].

Property 1.1: The linearized of this system around a point of equilibrium is not controllable.

Proof

The linear approximation of the system (1.34) around a state of equilibrium ($q(x_0, y_0, \theta_0)=0$, $u=0$), is given by:

$$\dot{\tilde{q}} = \begin{bmatrix} \cos\theta_0 & 0 \\ \sin\theta_0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}, \quad \tilde{q} = q - q_0$$

The rank of the controllability matrix being smaller than the size of the system state (1.34), the criterion of Kalman's controllability is not verified and the linearized is therefore not controllable.

Property 1.2: The nonlinear system (1.34) is controllable.

Proof

The system (1.34) can be written in the form:

$$\dot{q} = g_1(q)v + g_2(q)w \tag{1.44}$$

With $q = [x, y, \theta]^T$, $g_1(q) = [\cos\theta, \sin\theta, 0]^T$ and $g_2(q) = [0, 0, 1]^T$. The following result then gives a sufficient condition for the system (1.44) to be controllable.

Theorem 1.3 : Consider the system

$$\dot{q} = \sum_{j=1}^{m} g_j(q)u_j \tag{1.45}$$

Where $q \in \Re^n$, $u_j\,(j = 1,.....,m)$ are the control variables and $g_j(k)(j = 1,.....,m)$ are differentiable vector fields on $\Re^n$.

Either the accessibility algebra noted $Lie(g_1, \ldots, g_m)$ defined as the smallest subalgebra of analytic vector fields defined on $\Re^n$ which contains the vector fields generated by $\{g_1, \ldots, g_m\}$:

$$Lie(g_1,.....,g_m)=\left\{\left[h_k,\left[h_{k-1},\left[....,\left[h_2,h_1\right]\right]\right]\right],\ h_j\in\{g_1,.....,g_m\},\ j=1...k,\ k=0...\infty\right\}$$

Where $\left[h_i,h_j\right]=\frac{\partial h_j}{\partial q}h_i-\frac{\partial h_i}{\partial q}h_j$. For $q=\Re^n$, accessibility distribution A_c is the distribution generated by $Lie(g_1,.....,g_m)$:

$$A_c(q)=vect\{h(x),\ h\in Lie(g_1,.....,g_m)\}$$

The system (1.45) is controllable if:

$$\dim(A_c(q))=n,\forall q\in\Re^n$$

Property 1.2 is an immediate corollary of this theorem. Indeed, it is enough to check that in every point *q*, the vectors $g_1(q)$, $g_2(q)$, and $[g_1, g_2](q)$ form a basis of $\Re^3$.

For the system (1.34), we have *n*=3, vector fields

$$rank\{g_1,g_2,[g_1,g_2]\}=\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ \sin\theta & 0 & -\cos\theta\\ 0 & 1 & 0\end{bmatrix}=3$$

For a desired Cartesian motion for the unicycle, it can be practical to generate a corresponding state trajectory $q_d(t) = (x_d(t), y_d(t), \theta_d(t))$. To be feasible, the latter must satisfy the non-holonomic constraint the movement of the vehicle or, equivalently, be compatible with (1.44).

Setting the status tracking error as $\tilde{q}=q-q_d$, and the input variations as $\tilde{v}=v-v_d$, and $\tilde{w}=w-w_d$, the tangent linearization of system (1.44) about the reference trajectory is [38]

$$\dot{\tilde{q}}=\begin{bmatrix}0 & 0 & -v_d\sin\theta_d\\ 0 & 0 & v_d\cos\theta_d\\ 0 & 0 & 0\end{bmatrix}\tilde{q}+\begin{bmatrix}\cos\theta_d & 0\\ \sin\theta_d & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\tilde{v}\\ \tilde{w}\end{bmatrix}=A(t)\tilde{q}+B(t)\begin{bmatrix}\tilde{v}\\ \tilde{w}\end{bmatrix} \tag{1.46}$$

Since the linearized system is *time-varying*, a necessary and sufficient controllability condition is that the controllability Gramian is nonsingular. However, a simpler analysis can be conducted by defining the state tracking error through a rotation matrix as

$$\tilde{q}_R=\begin{bmatrix}\cos\theta_d & \sin\theta_d & 0\\ -\sin\theta_d & \cos\theta_d & 0\\ 0 & 0 & 1\end{bmatrix}\tilde{q} \tag{1.47}$$

Using (1.46), we obtain

$$\dot{\tilde{q}}_R=\begin{bmatrix}0 & w_d & 0\\ -w_d & 0 & v_d\\ 0 & 0 & 0\end{bmatrix}\tilde{q}_R+\begin{bmatrix}1 & 0\\ 0 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\tilde{v}\\ \tilde{w}\end{bmatrix} \tag{1.48}$$

When v_d and w_d are constant, the above linear system becomes time-invariant and controllable, since matrix

$$C = \begin{bmatrix} B & AB & A^2B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & -w_d^2 & v_d w_d \\ 0 & 0 & -w_d & v_d & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.49}$$

Have rank 3 provided that either v_d or ω_d are nonzero. Therefore, we conclude that the kinematic system (1.34) can be locally stabilized by linear feedback about trajectories which consist of linear or circular paths, executed with constant velocity.

1.6.1.7. ***Static Feedback Linearizability* (*Chained Forms*)** [38]

The existence of canonical forms for kinematic models of non-holonomic robots allows a general and systematic development of control strategies in open and closed loop. The most useful canonical structure is the chained form, which in the case of two-input systems is

$$\begin{aligned} \dot{z}_1 &= u_1 \\ \dot{z}_2 &= u_2 \\ \dot{z}_3 &= z_2 u_1 \\ &\vdots \\ \dot{z}_n &= z_{n-1} u_1 \end{aligned}$$

The nonholonomic kinematic model (1.34) cannot be transformed into a linear controllable system using *static* (i.e., time-invariant) state feedback. In fact, the controllability condition (*Property* 1.2) means that the distribution generated by vector fields g_1 and g_2 is not involutive, thus violating the necessary condition for full state feedback linearizability [39].

However, system equations can be transformed via feedback into simple integrators (input–output linearization and decoupling). The choice of the linearizing outputs is not unique. An interesting example is the following. For the kinematic model (1.34), the globally defined coordinate transformation

$$\begin{aligned} z_1 &= \theta \\ z_2 &= x\cos\theta + y\sin\theta \\ z_3 &= x\sin\theta - y\cos\theta \end{aligned}$$

The static state feedback

$$\begin{aligned} v &= u_2 + z_3 u_1 \\ w &= u_1 \end{aligned}$$

Lead to the so-called *chained form*

$$\begin{aligned} \dot{z}_1 &= u_1 \\ \dot{z}_2 &= u_2 \\ \dot{z}_3 &= z_2 u_1 \end{aligned} \tag{1.50}$$

With z_1 and z_2 as linearizing outputs. Note that (z_2, z_3) is the position of the unicycle in a rotating left-handed frame having the z_2 axis aligned with the vehicle orientation. We note also that the transformation in chained form is not unique.

1.6.2. Trajectory Tracking

1.6.2.1. *Feedforward Control Generation*

Assume that the point (x, y) of the unicycle must follow the Cartesian trajectory $(x_d(t), y_d(t))$, with $t \in [0, T]$ (possibly, $T \to \infty$). From the kinematic model (1.34) one has [38].

$$\theta = ATAN2(\dot{y}, \dot{x}) + k\pi \qquad k = 0, 1 \tag{1.51}$$

Therefore, the nominal feedforward controls are

$$v_d(t) = \pm\sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \tag{1.52}$$

$$w_d(t) = \frac{\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \tag{1.53}$$

The derivative of (1.51) with respect to time in order to calculate w_d. The chosen sign for $v_d(t)$ will determine the forward or backward motion of the vehicle. We note that, in order to be exactly reproducible using $v_d(t)$ and $w_d(t)$, the desired Cartesian motion $(x_d(t), y_d(t))$ should be twice differentiable in $[0, T]$.

1.6.2.2. *Linear Control Design*

Consider again the linearization procedure of the unicycle around the trajectory. Define the state tracking error e as [38]

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix} \tag{1.54}$$

Using the following nonlinear transformation of velocity inputs

$$\begin{aligned} v &= v_d \cos e_3 - u_1 \\ w &= w_d - u_2 \end{aligned} \tag{1.55}$$

The error dynamics becomes

$$\dot{e} = \begin{bmatrix} 0 & w_d & 0 \\ -w_d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 0 \\ \sin e_3 \\ 0 \end{bmatrix} v_d + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{1.56}$$

Linearizing (1.56) around the reference trajectory, we obtain the same linear time-varying equations (1.48), now with state e and input (u_1, u_2). Define the linear feedback law.

$$\begin{aligned} u_1 &= -k_1 e_1 \\ u_2 &= -k_2 sign(v_d(t)) e_2 - k_3 e_3 \end{aligned} \tag{1.57}$$

A desired closed-loop polynomial

$$(\lambda + 2\zeta a)(\lambda^2 + 2\zeta a\lambda + a^2) \qquad \zeta, a > 0$$

We having *constant* eigenvalues (one negative real at $-2\zeta a$ and a complex pair with natural angular frequency $a > 0$ and damping coefficient $\zeta \in (0, 1)$) can be obtained by choosing the gains in (1.57) as

$$k_1 = k_3 = 2\zeta a \qquad k_2 = \frac{a^2 - w_d(t)^2}{|v_d(t)|}$$

However, k_2 will go to infinity (i.e., an infinite control effort would be required for the same transient performance) as $v_d \to 0$. Therefore, a convenient *gain scheduling* is achieved by letting $a = a(t) = \sqrt{w_d^2(t) + bv_d^2(t)}$ so that

$$k_1 = k_3 = 2\zeta\sqrt{w_d^2(t) + bv_d^2(t)} \qquad k_2 = b|v_d(t)| \tag{1.58}$$

In terms of the original control inputs, this design leads to the *non linear time-varying* controller

$$\begin{aligned} v &= v_d \cos(\theta_d - \theta) + k_1[\cos\theta(x_d - x) + \sin\theta(y_d - y)] \\ w &= w_d + k_2 sign(v_d)[\cos\theta(y_d - y) - \sin\theta(x_d - x)] + k_3(\theta_d - \theta) \end{aligned} \tag{1.59}$$

1.6.2.3. ***Nonlinear Control Design***

We present now a nonlinear design for trajectory tracking. Consider again (1.56). Define [38]

$$\begin{aligned} u_1 &= -k_1(v_d(t), w_d(t))e_1 \\ u_2 &= -\bar{k}_2 v_d(t)\frac{\sin e_3}{e_3}e_2 - k_3(v_d(t), w_d(t))e_3 \end{aligned} \tag{1.60}$$

Constant $\bar{k}_2 > 0$, $k1(\cdot, \cdot)$ and $k3(\cdot, \cdot)$ continuous gain functions.

Theorem 5.1. *Assuming that v_d and ω_d are bounded with bounded derivatives, and that $v_d(t) \not\to 0$ or $w_d(t) \not\to 0$ when $t \to \infty$, the control law* (1.60) *globally asymptotically stabilises the origin* $e = 0$.

Proof: It is based on the use of the Lyapunov function

$$V = \frac{\bar{k}_2}{2}(e_1^2 + e_2^2) + \frac{e_3^2}{2}$$

Whose time derivative along closed-loop system solutions has not been increasing since

$$\dot{V} = -k_1\bar{k}_2 e_1^2 - k_3 e_3^2 \le 0$$

Therefore, $\|e(t)\|$ is bounded, $\dot{V}(t)$ is uniformly continuous, and $V(t)$ tends to some limit value. Using Barbalat lemma, $\dot{V}(t)$ tends to zero. From this and by analyzing the equations of the system, we can show that $(v_d^2 + w_d^2)e_i^2 \quad (i = 1, 2, 3)$ tends to zero so that, from the persistency of the (state) trajectory, the result follows.

Merging (1.54), (1.55), and (1.60), the resulting control law is

$$v = v_d \cos(\theta_d - \theta) + k_1(v_d, w_d)\left[\cos\theta(x_d - x) + \sin\theta(y_d - y)\right]$$
$$w = w_d + \bar{k}_2 v_d \frac{\sin(\theta_d - \theta)}{\theta_d - \theta}\left[\cos\theta(y_d - y) - \sin\theta(x_d - x)\right] + k_3(v_d, w_d)(\theta_d - \theta) \quad (1.61)$$

Taking advantage of the previous linear analysis, we can choose the gain functions k_1 and k_3 and the constant gain $\bar{k}_2$ as

$$k_1(v_d(t), w_d(t)) = k_3(v_d(t), w_d(t)) = 2\zeta\sqrt{w_d^2(t) + b v_d^2(t)} \qquad \bar{k}_2 = b$$

With $b > 0$ and $\zeta \in (0, 1)$.

1.6.2.4. ***Dynamic Feedback Linearization***

The dynamic feedback linearization problem consists in finding, if possible, a dynamic state feedback compensator of the form [38]

$$\dot{\zeta} = a(q, \zeta) + b(q, \zeta)u$$
$$w = c(q, \zeta) + d(q, \zeta)u \quad (1.62)$$

With v dimensional state ξ and m dimensional external input u, such that the closed-loop system (1.34), (1.62) is equivalent, under a state transformation $z = T(q, \xi)$, to a linear controllable system.

Only necessary or sufficient conditions exist for the solution of the dynamic feedback linearization problem. Constructive algorithms, which are essentially based on input-output decoupling.

The starting point is the definition of an appropriate m-dimensional system output $\eta = h(q)$, at which a desired behavior can be assigned (in our case, follow a desired trajectory). One then proceeds by successively differentiating the output until the input appears in a non-singular way.

At some phase, adding integrators to a subset of the input channels may be necessary to avoid subsequent differentiation of the original inputs. This *dynamic extension* algorithm builds the state ζ of dynamics compensator (1.62). The algorithm ends after a finite number of differentiations each time the system is invertible with respect to the chosen output. If the sum of the output differentiation orders is equal to the $n + v$ of the extended dimension state space, a complete input-state-output linearization is also obtained. The closed-loop system is then equivalent to a set of decoupled input-output chains of integrators from u_i to η_i ($i = 1, ..., m$).

We illustrate this exact linearization procedure for the unicycle model (1.44).

Define the linearization output vector as $\eta = (x, y)$. Differentiation w.r.t. time then returns

$$\dot{\eta} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix}$$

Showing that only v affects $\dot{\eta}$, while the angular velocity w can not be recovered, from this first-order differential information. In order to proceed, we must to add an integrator (whose state is noted ζ) on the linear velocity input.

$$v = \zeta \qquad \dot{\zeta} = a \quad \Rightarrow \quad \dot{\eta} = \zeta \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

Being the new input *a* the *linear acceleration* of the unicycle. Differentiating further

$$\ddot{\eta} = \dot{\zeta} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} + \zeta\dot{\theta} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\zeta\sin\theta \\ \sin\theta & \zeta\cos\theta \end{bmatrix} \begin{bmatrix} a \\ w \end{bmatrix}$$

And the matrix multiplying the modified input (*a*, *w*) is nonsingular provided, that $\zeta \neq 0$. Under this assumption, one can define

$$\begin{bmatrix} a \\ w \end{bmatrix} = \begin{bmatrix} \cos\theta & -\zeta\sin\theta \\ \sin\theta & \zeta\cos\theta \end{bmatrix}^{-1} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

In order to obtain

$$\ddot{\eta} = \begin{bmatrix} \ddot{\eta}_1 \\ \ddot{\eta}_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = u \tag{1.63}$$

The resulting dynamic compensator is

$$\begin{aligned} \dot{\zeta} &= u_1\cos\theta + u_2\sin\theta \\ v &= \zeta \\ w &= \frac{u_2\cos\theta - u_1\sin\theta}{\zeta} \end{aligned} \tag{1.64}$$

Since the dynamic compensator is one-dimensional, we have $n+v = 3+1 = 4$, equal to the total number of output differentiations in (1.63). Therefore, in the new coordinates

$$\begin{aligned} z_1 &= x \\ z_2 &= y \\ z_3 &= \dot{x} = \zeta\cos\theta \\ z_4 &= \dot{y} = \zeta\sin\theta \end{aligned} \tag{1.65}$$

The extended system is fully linearized in a controllable form and described by the two chains of second-order input-output integrators given by (1.63), rewritten as

$$\begin{aligned} \ddot{z}_1 &= u_1 \\ \ddot{z}_2 &= u_2 \end{aligned} \tag{1.66}$$

Note that the dynamic feedback linearization controller (1.64) has a potential singularity at $\zeta = v = 0$, that is, when the unicycle is not rolling. The appearance of such singularity in the dynamic extension process has proved to be structural for non-holonomic systems. This difficulty must be taken into account when designing linear model laws on the equivalent linear model.

Assume the robot must follow a smooth output trajectory ($x_d(t)$, $y_d(t)$) which is persistent, that is, such that the nominal control input $v_d = (\dot{x}_d^2 + \dot{y}_d^2)^{1/2}$ along the trajectory does never go to zero. On the equivalent linear and decoupled system (1.66), it is straightforward to design a

globally exponentially stabilizing feedback for the desired trajectory (with linear Cartesian transients) as

$$\begin{aligned} u_1 &= \ddot{x}_d(t) + k_{p1}(x_d(t) - x) + k_{d1}(\dot{x}_d(t) - \dot{x}) \\ u_2 &= \ddot{y}_d(t) + k_{p2}(y_d(t) - y) + k_{d2}(\dot{y}_d(t) - \dot{y}) \end{aligned} \tag{1.67}$$

With PD gains chosen as $k_{pi}>0$, $k_{di}>0$, for $i = 1, 2$.

We must not that the state of the dynamic compensator must be correctly initialized to the value $\zeta(0) = v_d(0)$. This ensures an exact tracking of the trajectory for an initial state of the corresponding robot. In this case, the control law ((1.64), (1.67) is reduced to the pure feedforward action.

1.7. Conclusion

In this chapter, we have given an overview of manipulators and mobile robots, which for the coming years will be an extremely attractive field of work, as much in its speculative research component, an ideal paradigm for robotics. third generation, only by that of applications that concern a very wide spectrum ranging from the machine for flexible workshop to the exploration vehicle for mission on Mars. These twolines create a synergy that induces rapid domain development in both core research laboratories and application centers.

Chapter 02

THE GENETIC ALGORITHMS (*GAS*).

2.1. Introduction

Among the problems encountered by the researcher and the engineer, the problems of optimization which occupy in our time a place of choice, and many researches, both practical and theoretical, are devoted to him. If we put aside the problems of discrete or multicriterion optimization, then the theory of optimization can be broken down into two main branches: local optimization and global optimization. If we can consider that the first is almost "fully known", the second is still partially known and research is at its peak, as confirmed by recent literature.

The main task of global optimization is the search for the solution that will minimize a given cost criterion, called the "global optimum". Global optimization therefore aims to seek not only a local minimum, but especially the smallest of these local minima.

There are two main approaches to global optimization. One is called deterministic whose search algorithms always use the same path to arrive at the solution, where we can "determine" in advance the steps of the research. The other is random [40]: for given initial conditions, the algorithm will not follow the same path to go to the solution found, and may even propose different solutions. It is towards this second branch, called the random global search, that our work will focus, and more particularly towards a specific type of random search algorithm, the *GAs*.

2.2. Genetic algorithms

2.2.1. history

Genetics, the study of heredity, is a discipline in biology. Indeed, while the notion of the inherited passage of biological information between generations dates back to the 17th century, the identification of DNA as a support for heredity dates back only to 1944, and its structure was only clarified in 1953. Advances in genetic engineering will allow a considerable development of knowledge about genes, their functioning and their regulation. The techniques quickly reach a high level of performance allowing today the study of the human genome.

Modern genetics can be traced back to Mendel's work in the 1860s, which first established the laws of heredity (Mendel, 1866). Mendel understands that a hereditary character can exist in different versions - aisles - some dominant, others recessive. It states the laws of transmission of certain hereditary traits. The responsible cellular element is the gene. Genes are transmitted between generations and function independently.

In 1900, three botanists, De Vries, Correns and Tschermak, "rediscover" the laws of Mendel's hybridization. In less than a decade, a new science is based on this foundation, Bateson calls it "genetic." In the mid-1910s, it was Morgan's work on Drosophila that led to the development of the chromosomal theory of heredity. The genes are then located on the chromosomes, and the work of Sturtevant, can order the genes along the chromosomes, constituting the first genetic

maps.

The material support of heredity (deoxyribonucleic acid), or DNA, was identified as such in 1944. The DNA double helix structure is elucidated by Watson and Crick in 1953. The golden years of biology are the 1950s. They culminated with the discovery by Jacob and Monod in the early 1960s, (Jacob and Monod, 1961), of the regulatory model of protein production in bacteria and viruses: messenger RNA, transcript of DNA into messenger RNA, translation of the messenger RNA into polypeptide sequences.

2.2.2. Definition of a *GAs*

Genetic algorithms are an abstraction of the theory of evolution. The basic idea is simple: if evolution has optimized biological processes, the use of the paradigm of evolution to find optimal solutions in the context of computer processing has a meaning.

The use of the theory of evolution as a computer model to find an optimal solution can be justified by the fact that the theory of evolution makes it possible to look for the solution among a very large number of possibilities within a reasonable time (the evolution process takes place in parallel). Thus, genetic algorithms are systems that rely on Darwin's principles of selection and on gene combination methods introduced by Mendel to address optimization problems.

As mentioned earlier, genetic algorithms focus on simulating the process of natural selection in a hostile environment related to the problem to be solved.

We will talk about an individual in a population and often the individual will be considered asa single chromosome. Chromosomes themselves are made up of genes that contain the hereditary characteristics of the individual. We will also find the fundamental principles of natural evolution, namely the principles of selection, crossing, mutation,...

In the context of optimization, each individual represents a point in the state space with which the value of the criterion to be optimized is associated. A random population of individuals is then generated for which the *GAs* focuses on selecting the best individuals while ensuring an efficient exploration of the state space. *GAs* differs from conventional optimization and search algorithms essentially in four fundamental points [41]:

1. They use a coding of the elements of the search space and not the elements themselves.
2. They search a solution from a population of points and not from a single point.
3. They impose no regularity on the studied function (continuity, differentiability, convexity ...). This is one of the great assets of *GAs*.
4. They are not deterministic, they use probabilistic transition rules.

2.2.3. Operating principle of *GAs*

GAs work on strings of fixed size. A population of individuals is initially defined, each of them has a particular character string (coding its chromosome) generally defined randomly.

Individuals will be evaluated according to an objective function, so those selected, will reproduce and submit to mutations. It is an iterative process that usually ends when the population no longer evolves between two periods.

An abstract description of the basic *GAs* can be done according to the following steps [4]:

1. Initialize randomly a chromosome population (individuals).
2. Evaluate each chromosome in the population. Each chromosome is associated with a function cost or fitness function determining its rank in the population. This function is the final

arbiter deciding the life or death of each individual.

3. Create new chromosomes, applying the operators of selection and reproduction.

4. Evaluate the new chromosomes (descend) and insert them into the population to build a new generation.

This process is repeated until satisfying the criteria for stopping *GAs* which is generally specified by a maximum number of generations.

The following flowchart gives a general description of the standard *GAs*.

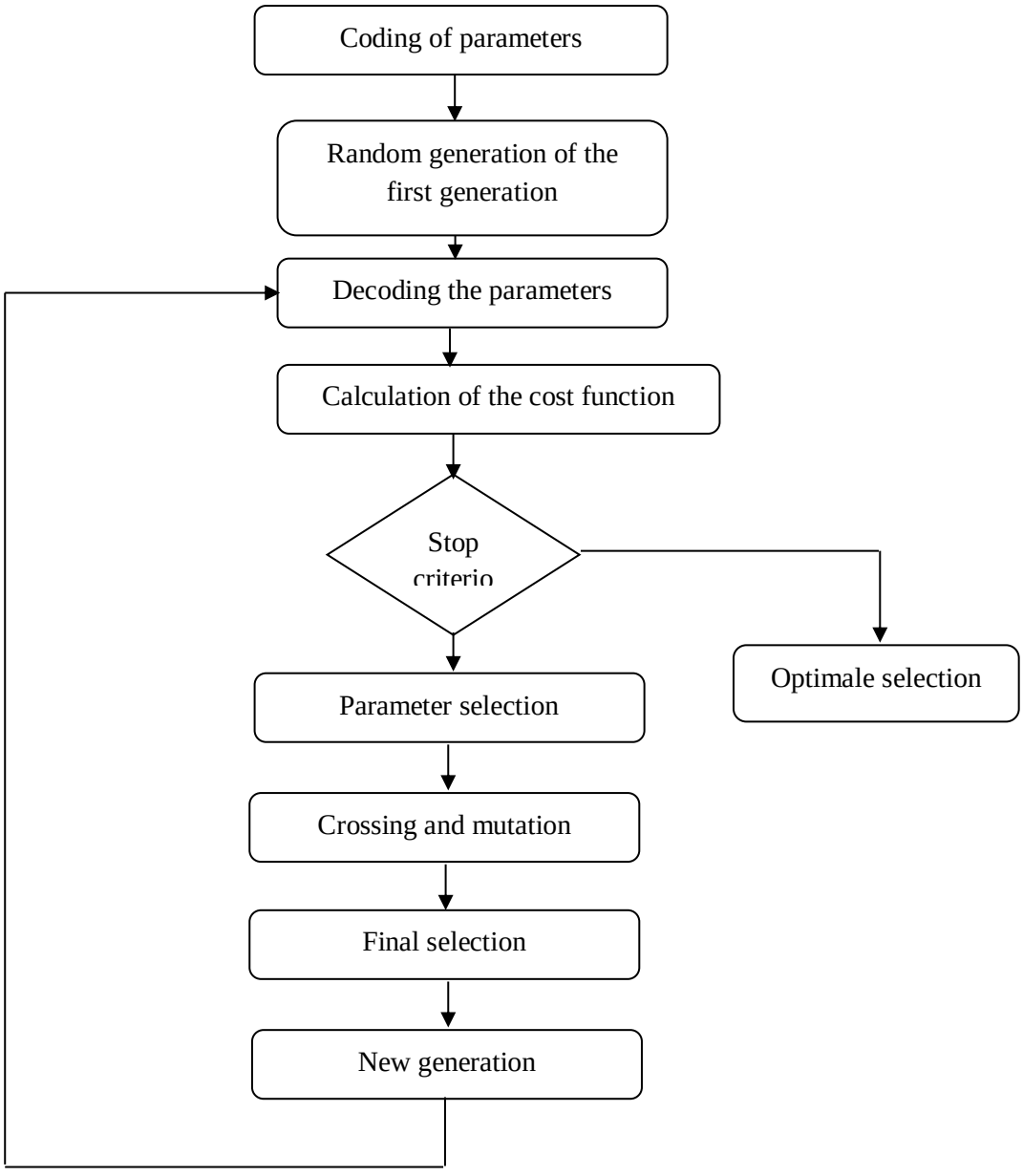


Figure 2.1 Flowchart of the *GAs*.

GAs are original systems, inspired by the presumed functioning of the living. The method used is very different from conventional optimization algorithms. There are four main points:

• *GAs* use a coding of parameters and not parameters themselves.

• *GAs* use only the values of the function being studied (instead of its derivative or other auxiliary knowledge).

• *GAs* work on a population of points instead of a single point.

• *GAs* use stochastic operators and not deterministic rules.

It is important to understand that the operation of such an algorithm does not guarantee success. We are in the presence of a stochastic system and the probability exists that a genetic pool is too far from the solution, or, for example, that a too rapid convergence blocks the process of evolution. These algorithms are nonetheless extremely efficient, their use is developing in fields as diverse as the stock market forecasting, the scheduling of production systems or the programming of assembly robots in the automotive industry.

GAs are characterized by:

1- A special method for coding solutions (chromosomes).

2- A fitness function that determines the rank of a chromosome.

3- A set of operations to manipulate the chromosomes of a generation to produce a new generation.

4- A method of initialization of chromosomes.

2.3. Fitness Technique

GAs requires having a cost function or an evaluation function that is minimized indirectly, its value is elaborate to an aptitude value that must be maximized.

However, each chromosome in a population may represent a different idea about solving the problem, thus satisfying the evaluation function. The decision of coding the chromosomes representing the possible solutions of a particular problem leads to the requirement of a suitable choice of the evaluation function. An evaluation function must be able to interpret the continuous data in the chromosomes and decide if the resulting solution is optimal. The evaluation function of a *GA* takes a chromosome and returns an aptitude value associated with it. The cost function $g(.)$ is calculated and this value must be transformed into an aptitude value to be suitable for a *GAs*.

Because of its analogy with the theory of natural evolution, *GAs* is naturally formulated in terms of maximization. Given a real function f with one or more variables, the optimization problem on the search space E is written as:

$$\max_{x \in E} \{f(x)\} \tag{2.1}$$

Moreover, the function to be optimized by a *GAs* must have positive values over the entire domain E. Otherwise, it is necessary to add to the values of f a positive constant F_{min} in accordance with the equivalence of (2.1).

$$\max_{x \in E} \{f(x)\} + F_{\min} \tag{2.2}$$

In many situations, the objective is expressed as the minimization of a performance function g.

$$\min_{x \in E} \{g(x)\} \tag{2.3}$$

The transition from the minimization problem to the maximization problem is obtained by the transformation of the function g.

2.3.1. Direct transformation

This technique transforms the function $g(-)$ to be minimized into an aptitude value f_i which

must be maximized by the *GAs*. The most used is:

$$\max_{x \in E} \{h(x)\} \tag{2.4}$$

With:

$$h(x) = \begin{cases} G_{\max} - g(x) & G_{\max} \geq g(x) \\ 0 & otherwise \end{cases} \tag{2.5}$$

The choice of the function *h* is not unique because any composition of the function *g* by any function decreasing and rising on the domain *E* will lead to a problem of maximization equivalent to the equation (2.3). In particular, we find in the literature the transformation function *h* given by.

$$h(x) = \frac{1}{1 + g(x)} \tag{2.6}$$

2.3.2. Windowing

A minimum or no fitness value is assigned to the wrong chromosome. Then each element of population is credited with an increasing aptitude, proportional to the quality with which the cost of the bad case exceeds its cost [4] as shown in the following figure:

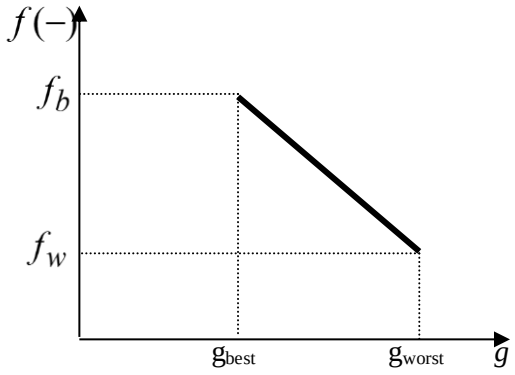


Figure 2.2 Relationship between ability and cost.

With*m* and *b* are constants, calculated according to f_b, f_w, g_{best}, g_{worst} in each generation. f_b and f_w, are the upper and lower limits respectively, of the windowing and they are set by the user. g_{best} and g_{worst} are the cost values associated with the best and the wrong chromosomes, respectively.

$$m = \frac{f_b - f_w}{f_{best} - f_{worst}} \tag{2.7}$$

$$b = f_b - m g_{best}$$

2.3.3. Linear scale change

First, a brutefitness is calculated using a direct transformation, then a linear function transforms the brutefitnessso that the average cost of the population is transformed into average fitness and zero fitness or a minimal amount is given to the chromosome with the maximal cost.

Direct transformation is the simplest and direct method of transforming cost values into

fitness values, but as soon as research develops, the population becomes more uniform and the difference in fitness between good and bad chromosomes becomes small. Since this difference governs the survival and decadence of chromosomes, the performance of the algorithm decreases. Windowing and linear scale change produce two alternatives to overcome this difficulty.

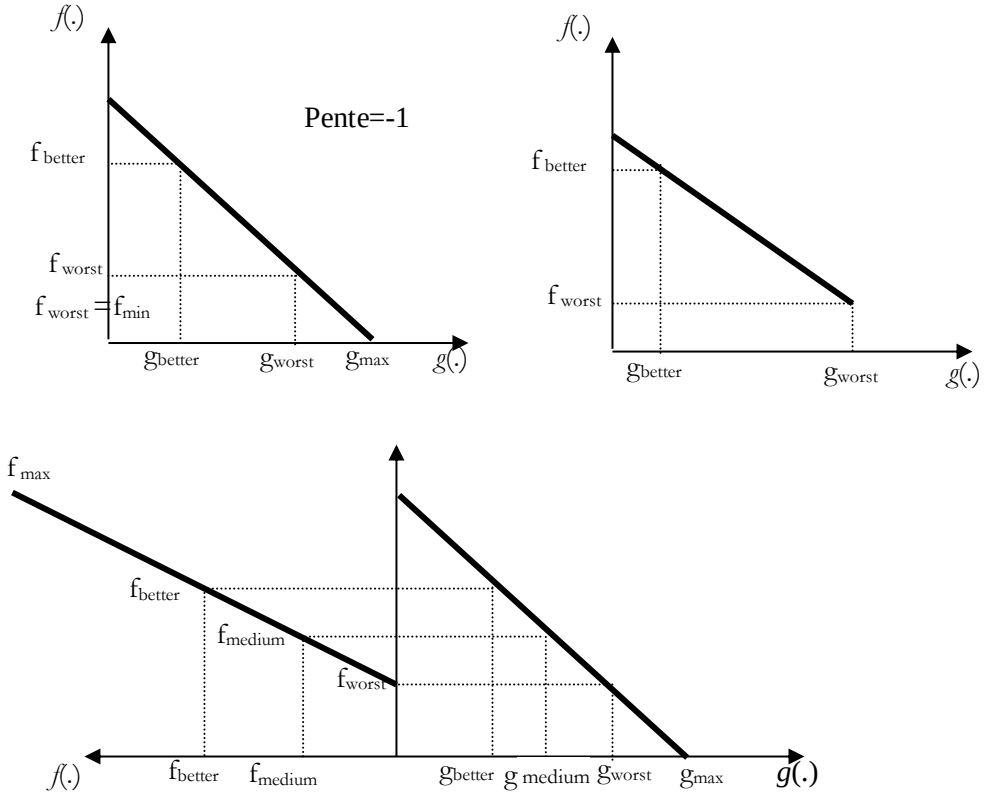


Figure 2.3 Fitness function, a) direct transformation,b) windowing, c) linear change of scale.

2.4. Operating mechanism of a *GAs*

2.4.1. Initial population

The choice of the initial population of individuals strongly conditions the speed and efficiency of the algorithm. By individuals, we do not necessarily speak of "physical" individuals, but of objects used to work on the algorithm. For example, if we want to find the global minimum of a function, an individual will be a number. If the position of the optimum in the state space is totally unknown, it is natural to randomly generate individuals by making uniform draws in each of the domains associated with the state space components, ensuring that the individuals produced respect the constraints. It is a generation of this type that we will use in our program, because we do not know initially what type of animal will be able to survive in the middle entered by the user. If, on the other hand, a priori information on the problem is available, it obviously seems natural to generate the individuals in a particular sub-domain in order to accelerate convergence.

2.4.2. Coding and decoding of parameters

2.4.2.1. *Coding*

In nature, the gene structures are coded in base 4, whose "digits" are the four nitrogenous bases: adenine (A), thymine (T), cytosine (C) and guanine (G). In the context of genetic algorithms, this type of coding is very difficult to use and is therefore not retained. Historically the coding used by genetic algorithms was represented as bit strings containing all the information needed to describe a point in the state space. This type of coding is useful for creating simple crossover and mutation operators (by bit inversion, for example).

The first step is to define and properly code the problem. At each optimization variable x_i (at each parameter of the device), one matches a gene. One calls chromosome a set of genes. Each device is represented by an individual with a genotype consisting of one or more chromosomes. One calls population a set of *N* individuals that to make it evolve.

In binary coding, each gene is a long integer (32 bits) for example. A chromosome is a table of genes (Figure 2.5 and Figure 2.6). An individual is an array of chromosomes. The population is an array of individuals. Note that one could also use other forms of coding (real, Gray coding ...).

The result is a structure with five levels of organization (Figure 2.4), resulting in the complex behavior of *GAs*:

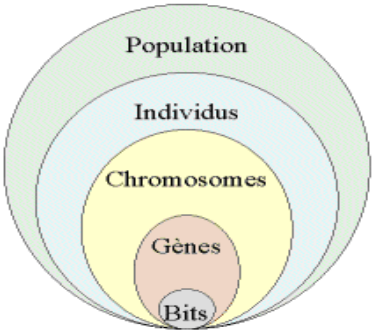


Figure 2.4 The five levels of organization of a genetic algorithm.

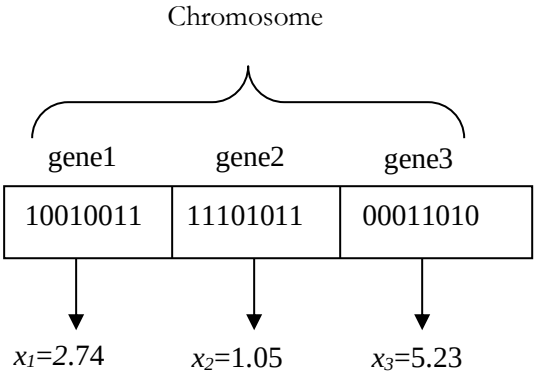


Figure 2.5 Schematic illustration of the coding of the optimization variables x_i.

Recall that in this study the *n* variables are assumed to be real. We consider a finite search space:

$$x_{i\min} \le x_i \le x_{i\max} \qquad \forall i \in [1; n]$$

In order to code our real variables in binary, we discretize the search space. Thus a 32-bit coding involves a discretization of the intervals in $g_{max} = 2^{32}-1 = 4\ 294\ 967\ 295$ discrete values.

Note that if this discretization is finer than that of the physical model used, the function is comparable to a staircase function, if we consider it on a sufficiently small scale.

Long Integer (32 bits) $g_i = \sum_{j=0}^{31} b_j 2^j$

Figure 2.6 Each gene (each parameter of the device) is encoded by a long integer (32 bits).

2.4.2.2. ***decoding***

The transformation of a binary string into a real number *x* can then be executed in two steps:

1. Conversion (base 2 to base 10): $x' = \sum_{i=1}^{S} 2^{i-1}$ (2.8)

2. Search for the corresponding real number: $x = x_{\max} - x' \frac{x_{\max} - x_{\min}}{2^S - 1}$ (2.9)

Or what amounts to the same directly in one step by:

$$x = x_{\max} - \sum_{i=1}^{S} \frac{2^{i-1} ld}{2^S - 1}, \; ld = x_{\max} - x_{\min} \quad (2.10)$$

2.4.3. Genetic operators

The genetic algorithm realizes the optimization by the manipulation of a population of chromosomes with each generation, the *GAs* creates a set of new chromosomes with various operations called genetic operators:

- The selection.
- The crossover.
- The mutation.

2.4.3.1. ***The selection***

Unlike other optimization techniques, *GAs* does not need to know the derivative of the objective function, which makes their field of application more extensive.

Selection, as the name implies, allows the best individuals in a population to be identified statistically and to partially eliminate the bad ones. Nevertheless, as in Nature, we should not confuse selection and elitism: it is not because an individual is good that he will necessarily survive (the hazards of life) and so it is not because it is bad that it must disappear (luck also helps to survive). Indeed, very often, a "well adapted" species can descend from an individual decreed "bad". There are different principles of selection, including the most classic ones.

a. ***Methods of selection***

A) The proportional selection "Wheel selection"

The proportional selection "wheel selection" (or selection by wheel casino): it consists in associating with each individual a segment whose length is proportional to his fitness. These

segments are then concatenated on a graduated axis that is normalized between 0 and 1. One then draws a random number of uniform distribution between 0 and 1, then one looks at what is the selected segment, and one reproduces the corresponding individual. With this technique, the good individuals will be more often selected than the bad ones, and the same individual will be able to be selected several times with this method. Nevertheless, on small populations, it is difficult to obtain exactly the mathematical expectation of selection because of the small number of draws (the ideal case of application of this method is obviously that where the population is large.). So one will has a selection bias more or less strong depending on the size of the population.

Individual	fitness	relative fitness
1	15	0.1
2	30	0.2
3	45	0.3
4	22.5	0.15
5	7.5	0.05
6	30	0.2

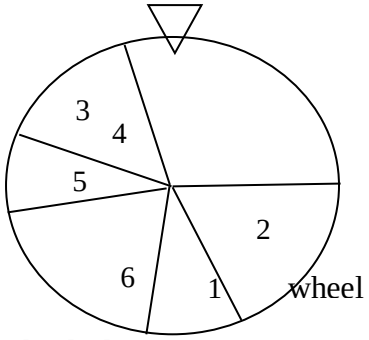


Figure 2.7 Exemple application of casino wheelselection.

B) Selection with stochastic remainder

The "stochastic remainder without replacement selection": in this case one canavoid this kind of problem, because a part of the population is selected in a purely deterministic way. One associates with each individual the ratio f_i of his fitness on the average of the fitness then one takes his whole part $E(f_i)$ which indicates the number of times to reproduce the individual i. This ensures an exact number of representatives for the next generation, eliminating bias. However, weak individuals (lower fitness than average fitness) are invariably eliminated with this method, which is bad because these occupy positions in the state space that associated with others can bring us closer to the under-domain containing the optimum. One therefore associates with the deterministic selection a random selection principle based on a " wheel selection" executed on all the individuals affected by new finesses f_i-$E(f_i)$.

Individual	fitness	r_i	$E\|r_i\|$	$r_i - E\|r_i\|$
1	15	0.6	0	0.6
2	30	1.2	1	0.2
3	45	1.8	1	0.8
4	22.5	0.9	0	0.9
5	7.5	0.3	0	0.3
6	30	1.2	1	0.2

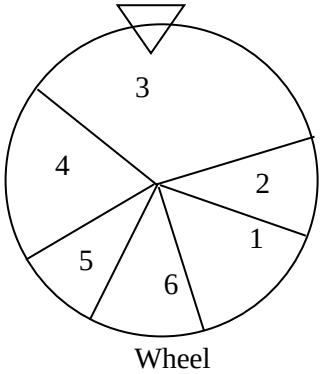


Figure 2.8 Application of the « stochastic remainder without replacement selection » to theprevious example.

C) *Selection of individuals of the new generation*

This step involves selecting individuals from a population that will be kept for the next generation. Therefore, following the generation of the new individuals (child), one will determine which individuals of the current population will be replaced by them. For this there are 2 alternative methods mainly used: stationary replacement and elitist replacement.

C.1) *Stationary selection*

For each generation, a small number of individuals (children) are generated, usually less than or equal to the number of individuals in the population, to form the new population. In this case, the children automatically replace the parents regardless of their respective performance. The choice of parents who will be replaced is done randomly or by replacing the less successful individuals in the population.

This type of replacement generates a population with a large variation and to be favored the genetic drift which manifests all the more as the population is small. Moreover, in many cases, since even a child with poor performance necessarily replaces a parent, the best solution is not reached, but only approaches it. We can also say that in the stationary replacement the performance function is not necessarily monotonous increasing.

C.2) *Elitist selection*

In the elitist strategy, one keeps at least the individual with the best performances from one generation to the next. In general, it can be assumed that a new individual (child) takes place in the population if he fulfills the criterion of being more efficient than the least efficient of the individuals of the preceding population. So the children of a generation will not necessarily replace their parents as in the stationary replacement. The performance function of the best individual will therefore be monotonous increasing from generation to generation [42].

This type of strategy improves the performance of evolutionary algorithms in some cases. But also has a disadvantage because it increases the rate of premature convergence.

2.4.3.2. ***Reproduction***

The reproduction of the latter ensures the chance for the best solution to follow generations by applying the following genetic operators

- Crossover.
- Mutation.

a. ***Crossover***

A) Crossover operation

The crossing operator uses 2 individuals (parents) to create 1 or 2 children. There are currently no specific rules for this operation. This step simply consists in determining the child's genetic structure according to that of the parents.

During this operation, 2 genetically close individuals will generate children close to them. Otherwise, if 2 individuals are very distant genetically, the children will be too far away.

A crossover can lead to low performance children, which means they will not be designated to generate other individuals. In this case, it is said to be a lethal crossover. One method to avoid this is to form pairs of individuals on the basis of their genetic components (scheme). Specifically, individuals who have a close search space.

Crossover is a random operation, there are methods like uniform crossingor crossing at 1-point or two-point.

A.1) One-point crossover

In the 1-Point cross, a cut in the genotype of the individual is randomly determined. Then one applies this cut on both parents to create the children. The principle is simple, a child inherits the genotype formed from the genes of the first parent to the cutoff point and genes of the second parent from this cutoff point to the last gene. The second child is complementary to the first (inversion of first and second parent numbering) [42].

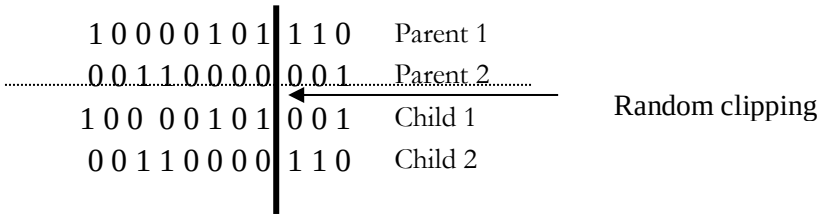


Figure 2.9 One-point crossover.

A.2) Two-point crossover

This technique is similar to the previous one, but in this case two positions are selected and the sub-chains between these two positions are exchanged. This method has the same properties already described. But it can combine some scheme that the one-point version can not [42].

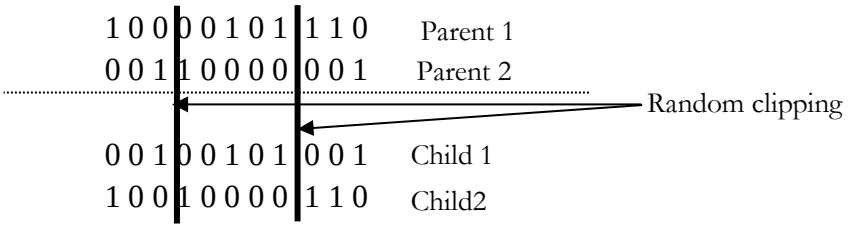


Figure 2.10 Two-point intersection.

A.3) Uniform crossover

The uniform crossing uses what is called a crossover mask. The latter calculates randomly using a uniform distribution. A mask is used for a single pair.

A gene at 1 in the mask indicates that the child inherits from the first parent, otherwise he inherits from the second parent. The second child is complementary to the first [42].

1 1 0 1 0 1 0 0 0 1 0	Crossovermask
1 0 0 1 1 1 0 1 1 0 0	Parent 1
0 0 1 0 0 1 1 0 1 0 1	Parent 2
1 0 1 1 0 1 1 0 1 0 1	Child 1
0 0 0 0 1 1 0 1 1 0 0	Child 2

Figure 2.11 Uniform crossover.

A.4) Arithmetic Crossover

This technique is introduced by Michalewicz. If this operator is applied on both parents C^1 and C^2, two children (off springs) $H^k = (h_1^k, h_2^k, ..., h_i^k, ..., h_L^k)$, $k = 1,2.$ are generated, such as

$$h_i^1 = \lambda_i c_i^1 + (1-\lambda_i)c_i^2 \text{ and } h_i^2 = \lambda_i c_i^2 + (1-\lambda_i)c_i^1 \tag{2.11}$$

If λ_i is a randomly generated value belonging to the interval [0, 1], so one speaks of a non-uniform arithmetic crossing, and if λ_i is a constant chosen by the user, one speaks of uniform arithmetic crossing. In addition if $\lambda_i = 1/2$ this is known as a guaranteed average crossover.

This technique guarantees that $h_i^1, h_i^2 \in [a_i, b_i]$, for dominions of bounded variation since $\lambda_i \in [0,1]$ and $C_i^1, C_i^2 \in [a_i, b_i]$

A.5) Crossover BLX-α (*Blend crossover*)

Developed by (Eshelman and Scahffer, 1993) [43]. A child is generated $H = (h_1, h_2, ..., h_i, ..., h_L)$ such as h_i is a random value chosen in the interval $[c_{\min} - I\alpha, c_{\max} + I\alpha]$ where

$$c_{\max} = \max(c_i^1, c_i^2), c_{\min} = \min(c_i^1, c_i^2) \quad \text{and} \quad I = c_{\max} - c_{\min} \tag{2.12}$$

The parameter α is generally taken equal to 0.5 to have a balanced relationship between exploration and exploitation (convergence).

One note that the domain of gene values of the generated chromosome no longer belongs to the domain [a_i, b_i] of the parents. As one goes along, this area will expand or shrink. The explanation is as follows:

Knowing that $c_{\min} \in [a_i, b_i]$ and $c_{\max} \in [a_i, b_i]$ then $I \in [a_i, b_i]$.

On the other hand, the interval whose values obtained from the product $I.\alpha$ is defined according to the value of α. To remain in the search domain [a_i, b_i], then certain conditions must be respected to determine a α parameter with good properties. However, for α = 0, we have the special case of the BLX-0.0 crossover or the flat crossover. For which the problem of the bounded exploration domain does not arise.

In summary, this type of crossover ($\alpha \neq 0$) expands the field of research, hence a very

extensive exploration, this makes it difficult to converge towards possible solutions.

A.6) Linear crossover

The linear crossing generates three progeny, whose genes h_i^k k = 1, 2, 3; are calculated by

$$h_i^1 = 1/2(c_i^1 + c_i^2) \qquad h_i^2 = \tfrac{3}{2}c_i^1 - \tfrac{1}{2}c_i^2 \qquad h_i^3 = -\tfrac{1}{2}c_i^1 + \tfrac{3}{2}c_i^2 \tag{2.13}$$

When selecting, the new generation will be made up of the two best of these children. Onenotices that the exploration domain is bounded.

A.7) Discret crossover

The values of the $h_i$genes of the children generated H are equal to the values of the genes c_i^1 of the parent C^1 or to the values of the genes c_i^2 of the parent C^2.

The choice of parent is made in a random way with a uniform distribution. The exploration interval is bounded.

A.8) Heuristic crossover

This technique was developed by (Wright, 1990) [44], it is unique in that: (1) uses the values of the objective function $o(C^i)$ to determine the search direction, (2) it produces a single descendant, and (3) it may not produce a descendant at all. The operator generates a single child (offspring) as

$$h_i^1 = r(c_i^1 - c_i^2) + c_i^1 \tag{2.14}$$

Where r is a random number between 0 and 1, and C^1 is the most suitable parent, i.e, $o(C^1) \geq o(C^2)$ for a maximization problem and $o(C^1) \leq o(C^2)$ for a minimization problem.

It is clear that this operator can generate genes outside the search domain. In this case another random number r is generated and another descendant is created. And if after w trials the constraint is not satisfied (search domain overflowed), the process is broken and does not produce a descendant.

Finally, it seems that this operator contributes to the accuracy of the solution found; his major responsibilities are (1) to improve local research, and (2) to carry out research in the most promising direction.

A.9) *Extended crossover*

The h_i gene of the H chromosome generated by this operator is defined as follows

$$h_i^1 = c_i^1 + \alpha(c_i^2 - c_i^1) \tag{2.15}$$

Such as α is randomly chosen having a uniform distribution of the interval $[-d, 1+d]$]. In this type of crossing, d is taken equal to 0. In the opposite case this type of cross is called intermediate crossing. The typical value for d in this case is 0.25.

One notes that the gene values of the chromosomes generated no longer belong to the interval [a_i, b_i]. This makes the field of exploration quite large (same remark as for the crossover BLX-α).

Crossing is the key to *GAs* power. It is directly related to the ability of a population of

individuals to explore their research space and to combine the best results. It allows the *GAs* to concentrate its research in the most promising parts of the solution space, by combining chains containing partial solutions. However this operator is controlled by an important parameter which is the probability of crossing p_c. This probability varies between 0.6 and 1. It indicates whether the crossing between two individuals is with pairing (gene exchange) or without pairing.

For example for two individuals (parents) C^1 and C^2 chosen for a cross if $p_c = 0.6$. Then a random number r is generated in the interval [0, 1], if $r \leq p_c$ one has a crossing with pairing, otherwise one will has a crossing without matching and the parents are thus copied in the new generation.

b. ***Mutation operator***

One defines a mutation as the inversion of a bit in a chromosome (Figure 2.12). This is like randomly changing the value of a device parameter. Mutations play the role of noise and prevent the evolution from freezing. They make it possible to ensure a global as well as a local search, according to the weight and the number of mutated bits. Moreover, they mathematically guarantee that the global optimum can be achieved.

Unlike crossover, the mutation rate is generally low and ranges between 0.5% and 1% of the total population. This low rate makes it possible to avoid a random dispersion of the population and involves only a few modifications on a limited number of individuals. Mutation is becoming more and more important in *GAs*, while only a few years ago its role was still considered as an accessory.

1 0 1 1 0 1 1 0 1 0 1 Child
1 0 1 1 0 1 0 0 1 0 1 Mutated child

Figure 2.12 Mutation.

To avoid altering the performance of the *GAs*, Goldberg advises to take a mutation frequency every 1000 bits. Some researchers recommend taking the following value.

$$P_m = \frac{1}{L} \tag{2.16}$$

Where L is the length of the chromosome. Other studies have led to an empirical formula that expresses the optimal mutation rate in terms of chromosome length L and population size N.

$$P_m = \frac{1}{N\sqrt{L}} \tag{2.17}$$

2.5. Convergence

If the *GAs* has been correctly implemented, the population evolves step by step over successive generations and one finds that the average score of the population evolves globally towards the optimum of the function to be maximized. *Convergence* is the progression towards this optimality. A gene will be said to have converged if 95% of the population has this characteristic. The population converged when all the genes converged.

2.6. Advantage of *GAs*

Unlike operational research, the *GAs* requires no knowledge of how to solve the problem. It is only necessary to be able to evaluate the quality of the solution. Also, in the case of the search for the optimum of the analytic function, one does not need any differentiability or continuity.

2.7. Limitation of *GAs*

As mentioned earlier, *GAs* are effective tools for a very large class of problems. In addition, they make it possible to deal with problems in which the function to be optimized does not have any property of continuity or differentiability, but *GAs* have certain limitations [42]:

1. They are less efficient than a specific deterministic algorithm (where one exists) dedicated to a given problem.

2. The many parameters that control them are delicate to regulate (probabilities of crossing and mutation in particular), as well as chromosome coding that can radically change the speed of convergence.

3. In order to guarantee the robustness of evolutionary algorithms, the calculation of a very large number of fitness (sometimes of the order of several hundreds of thousands) is generally necessary before obtaining a good solution. This large number of calculations can be problematic when the cost of computing (system or temporal resources) of the fitness is important, when working in large scale on functions of high complexity for example.

4. They may have difficulty handling many complex constraints because the penalty technique integrated in the cost function uses a posteriori constraints in a passive way.

2.8. Conclusion:

Genetic algorithms can be used as a combinatorial research method, including properties based on parallelism and exploration, interesting search heuristics based on principles of self-organization.

Genetic algorithms may be an interesting alternative when traditional optimization methods (climber method, analytical methods such as least-squares) fail to provide reliable results efficiently.

However, as one has often pointed out in this chapter, most of the common implementations of *GAs* require an environment (process or process model) that is tolerant enough to allow the algorithm the "right to error "

This implies that the environment must permit the testing and evaluation of poor condidate solutions.

The genetic algorithm used alone is a search mechanism that is often "too blind" so it can be made more effective when combined with traditional, usually more local, search methods (conventional regulator, fuzzy regulator or neural control system).

Chapter 03

FUZZY LOGIC, NEURAL NETWORKS, NEURO-FUZZY.

3.1. Introduction

The knowledge of the universe in which one evolves is generally imperfect, in the measure where they can suffer from *uncertaintiesand/or inaccuracies.*One can see that man naturally integrates these imperfections in everyday life, especially in terms of reasoning and decision.

The idea of Zadeh, in 1965, through the new concept of gradual set membership of an element to a set, was to define a multivalued logic allowing to model these imperfections i.e., to take into account the intermediate states between the whole and the nothing.

Regarding the control of any process, fuzzy logic allows an innovative approach compared to the conventional control system. In the latter, one in general tries to model the process through a certain number of differential equations. This modeling is made difficult as the complexity of the processes to be controlled increases. In a radically opposite way, a controller will not describe the process but the way to control it, as would a human expert through rules that naturally integrate imprecision and uncertainty. In this approach, one will therefore speak of fuzzy expert systems based on production rules of the form "if premise then conclusion". These systems are extensions of classical expert systems in the measured that they incorporate imperfect knowledge.A number of applications using fuzzy logic have emerged these last years. The most publicized are certainly the achievements of researchers, since the 80s, were interested in particular control / process control. These applications are based on the work of Mamdani, who was certainly the first to see the potential of fuzzy subset theory in this field.

3.2. Fuzzy logic principle

A temperature of 10 ^{o}C, for a human, is generally considered cold, a temperature of 40 ^{o}C is qualified as hot. If each of these values belongs to a well defined category (set), what about intermediate values? An intuitive answer is to say that they belong to one or two of the preceding categories with different levels (normalized i.e., defined on [0,1]). This avoids rigid transitions between different categories, as is the case in binary logic (Figure 3.1). It seems in effect surprising to consider that a temperature of 40^{o}C is hot, while a temperature of 39.9^{o}C is not.

This example illustrates the fact that a classical binary logic is, in some cases, too restrictive. It is necessary to use a multivalued logic that will be seen as an extension of the previous one.

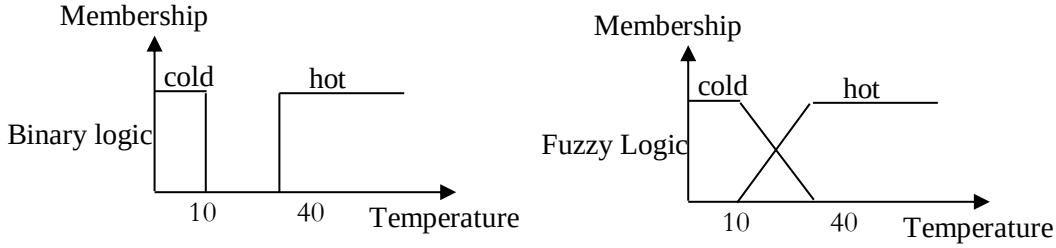


Figure 3.1 Example of sets definition on a universe of discourse in binary logic and fuzzy logic.

3.2.1. Fuzzy sets

In a classical set, an element belongs to a set if its binary membership function is 1.

In a fuzzy set, the membership function takes its values in [0, 1]. Thus, an element belongs more or less to the "concept "that represents the whole. A fuzzy set is thus defined by the set of values of the membership function on the definition domain (discrete or continuous).

Ex: The concept of " Cold " can be defined by the discrete fuzzy set.

A = {1/10, 0.8 / 15, 0.6 / 20, 0.4 / 30, 0.1 / 38, 0/40, 0/50}.

Ex: The concept "hot" can be defined as the set

$$C = \{x \in \Re / \ f_c(x) = x\}$$

3.2.1.1. *Fuzzy subset*

One has just seen what is meant by subset or, from a formal point of view. A fuzzy subset *A* on a universe of discourse *U*, is represented as in Figure 4.2through its characteristic function μ_A . It can also be described by a number of characteristics such as:

a. ***Support***

The support is defined as the set of values of the domain *X* for which the membership function is not null.

$$Supp\langle A\rangle = \{x \in X / \ f_A(x) \neq 0\} \tag{3.1}$$

b. ***Height***

The height of a fuzzy set *A* is the max value of the membership function on domain *X*.

$$h(A) = Max_{x \in X} \ f_A(x) \tag{3.2}$$

c. ***Kernel***

The kernel of a fuzzy set *A* is the set of values for which the membership function is equal to 1.

$$Kern(A) = \{x \in X / \ f_A(x) = 1\} \tag{3.3}$$

d. ***Cardinal***

The cardinal of a fuzzy set *A* is the sum of the membership functions.

$$|A| = Card(A) = \sum_{x \in X} J_A(x) \tag{3.4}$$

Given a membership function $\mu_A(x)$ of trapezoidal shape, one can define its height, its support and its kernel as follows:

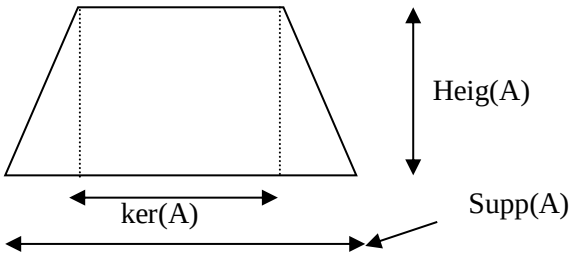


Figure 3.2 **S**upport, kernel, and height of a trapezoidal function.

e. ***Equality***

Two fuzzy sets *A* and *B* are equal if their equivalence functions are equal for all values of domain *X*.

$$\forall x \in X: \qquad f_A(x) = f_B(x) \tag{3.5}$$

f. ***Inclusion***

A fuzzy set *A* is included in a fuzzy set *B* if all its membership function values are smaller than those of *B* over the entire *X* domain.

$$\forall x \in X: \qquad f_A(x) \leq f_B(x) \tag{3.6}$$

g. ***Intersection***

The intersection *C* of 2 fuzzy sets *A* and *B* is defined as the fuzzy set whose membership function is the *min* of the membership functions of sets *A* and *B*.

$$\forall x \in X: \qquad f_C(x) = Min\{f_A(x), f_B(x)\} \tag{3.7}$$

h. ***Union***

The union *C* of 2 fuzzy sets *A* and *B* is defined as the fuzzy set whose membership function is the *max* of the membership functions of sets *A* and *B*.

$$\forall x \in X: \qquad f_C(x) = Max\{f_A(x), f_B(x)\} \tag{3.8}$$

i. ***Complement***

The complement A^C of a fuzzy set *A* is defined as:

$$\forall x \in X: \qquad f_A^C(x) = 1 - f_A(x) \tag{3.9}$$

A subset makes it possible to represent different notions, in particular the specificity and the precision. These different notions appear in Figure 3.3.

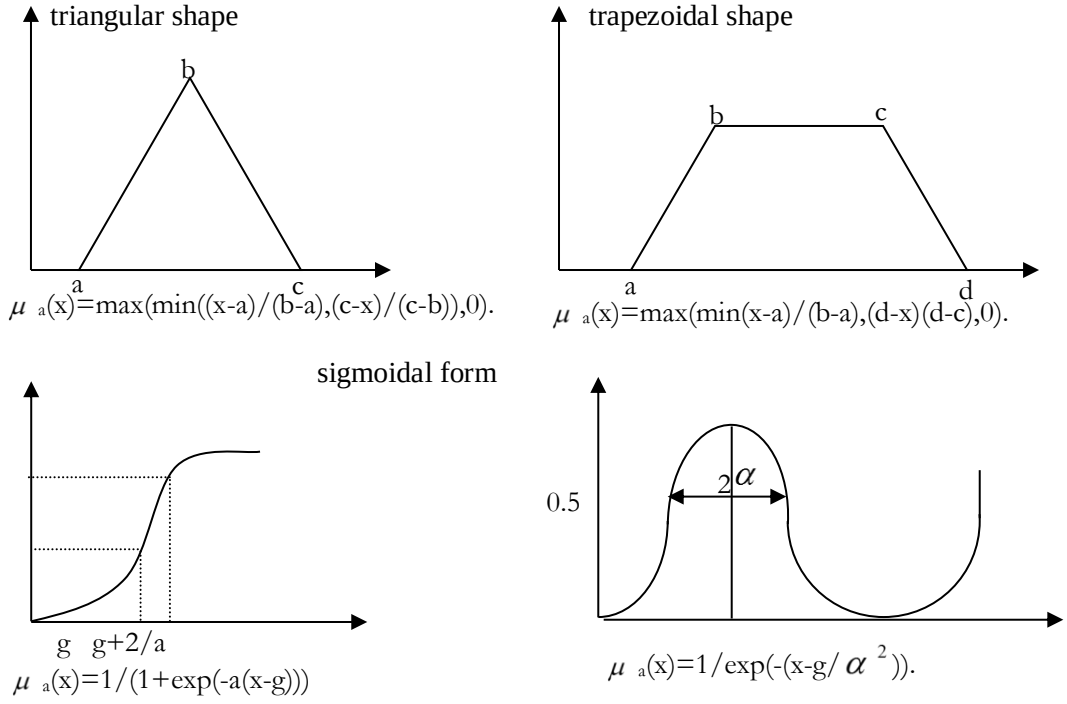


Figure 3.3 Form of membership functions.

3.2.2. Linguistic variable

A linguistic variable is defined by a triplet (V,U,T_V) where V represents a classical variable (age, temperature, ...) defined on the universe of discourse$U$. T_V is the set of possible instantiations of the variable V : it is about fuzzy subsets identified by their label A_i : one writes thus $T_V = \{A_1, A_2, ..., A_3\}$ [45].

Graphically, a linguistic variable can be represented as in Figure 3.4.

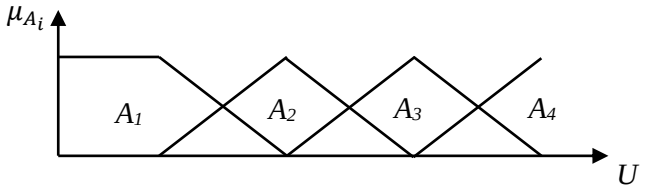


Figure 3.4 Representation of a linguistic variable defined as $\{V,U,T_V\} = \{A_1, A_2, A_3, A_4\}$

3.2.3. Fuzzy characterization

A fuzzy characterization of a linguistic variable (V, U, T_V) is a label A_i of subset or membership to T_V. Thereafter, this name will qualify indifferently a fuzzy subset or his label.

3.2.4. Proposals and fuzzy rule

An elementary fuzzy proposition is defined from a linguistic variable (V, U, T_V) by the qualification "V is A", with Amembership to T_V. For example, "Paul's size is AVERAGE" is an elementary proposition defined from the linguistic variable (Paul's size, {sizes}, {..., AVERAGE, ...}) [45].

The truth value of an elementary proposition "V is A" is equal to $A(v)$ where v is the exact numerical value of V.

A general fuzzy proposition is defined from elementary propositions and binary (and, or, implied) or unary (no) logical operators. There are several methods for calculating the truth value of such propositions. One gives here only the most commonly used.

- conjunction : (V_1 is A_1) and (V_2 is A_2)

$Min(\mu_{A1}(v1), \mu_{A2}(v2))$ (Zadehlogic)

$Max(\mu_{A1}(v1) + \mu_{A2}(v2) \Leftrightarrow 1, 0)$ (Lukasiewicz logic)

$\mu_{A1}(v1) . \mu_{A2}(v2)$ (probabiliste logic)

- disjunction : (V_1 is A_1) or (V_2 is A_2)

$Max(\mu_{A1}(v1), \mu_{A2}(v2))$ (Zadeh logic)

$Min(\mu_{A1}(v1) + \mu_{A2}(v2), 1)$ (Lukasiewicz logic)

$\mu_{A1}(v1) + \mu_{A2}(v2) \Leftrightarrow \mu_{A1}(v1) . \mu_{A2}(v2)$ (probabiliste logic)

- implication : (V_1 is A_1) **implies** (V_2 is A_2)

$Min(1 \Leftrightarrow \mu_{A1}(v1) + \mu_{A2}(v2), 1)$ (Lukasiewicz logic)

$Min(\mu_{A1}(v1), \mu_{A2}(v2))$ (Mamdani logic)

$\mu_{A1}(v1) \cdot \mu_{A2}(v2)$ (Larsen logic)

- complementation : **not** (V is A)

$1 \Leftrightarrow \mu_A(v)$

In the above, v, v_1 and v_2 correspond to numerical instantiations actual variables V, V_1 and V_2.

A fuzzy rule is a general fuzzy proposition using an implication between two fuzzy propositions. For example:

(V_1 is A_1) and (V_2 is A_2) **implies** (V_3 is A_3)

Or in a more linguistic form:

If (V_1 is A_1) et (V_2 is A_2) **then** (V_3 is A_3) Is a fuzzy rule. The part (V_1 is A_1) **and** (V_2 is A_2) is called premise of the rule and (V_3 is A_3) is the conclusion.

3.2.4.1. ***Activation of a fuzzy rule***

A rule can only be activated (i.e. intervene in the inference process) when the truth value of the fuzzy proposal constituting its premise is not zero.

3.2.5. Classic architecture of a fuzzy controller (FLC)

The classical architecture of a controller (*FLC*), proposed by Mamdani, is illustrated in Figure 3.5[45].

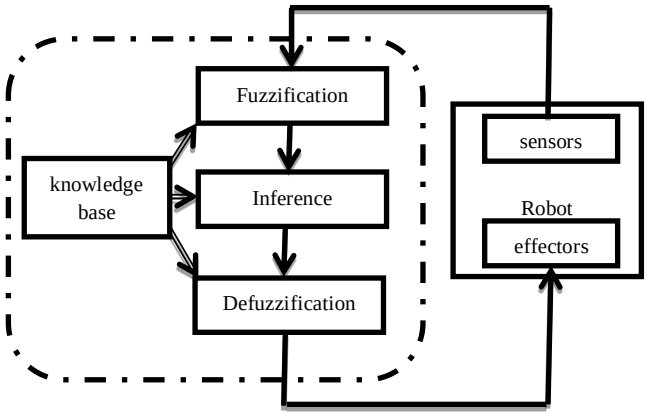


Figure 3.5 Architecture of a fuzzy controller.

As one can see in Figure 3.5, a controller or is composed of three parts:

3.2.5.1. ***Fuzzification***

The objective of fuzzification is to transform the input deterministic variables into fuzzy variables, that is to say into linguistic variables, by defining membership functions for these different input variables.

Membership functions can have different forms, but the trapezoidal, triangular or mathematical derived forms are the most commonly used.

The physical input quantities X are reduced to normalized quantities x in a range of variation, often called the universe of discourse, which may be either discrete or continuous. Very often, this universe of discourse is limited, by applying a limitation on the numerical value of $|x| \leq 1$, to overcome the problem of large variations of X. Normalization gains characterize scale factors

between x and X.

In the case of a continuous universe of discourse, the number of linguistic values (very negative, negative, zero, positive, very positive ...) represented by membership functions for a variable x_1 can vary (for example three, five or seven). An example of continuous fuzzification is shown (Figure 3.4) for a single variable of x, with trapezoidal membership functions, the corresponding linguistic values are characterized by symbols such as N for negative, Z for zero, and P for positive. So x_1=0.2.

After fuzzification becomes the fuzzy subset x_1 = (0, 0.8, 0.2).

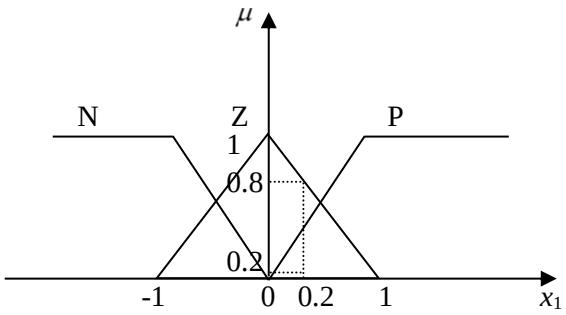


Figure 3.6 Continuous fuzzification with three membership functions.

In general, the membership functions take their value in the interval [0, 1].

The next section discusses the second step of establishing the rules from these fuzzy variables and evaluating them.

3.2.5.2. ***Inference:***

The adjustment strategy depends essentially on the inferences adopted. They link the measured quantities, which are the input variables (transformed into linguistic variables using fuzzification). To the output variable.The latter is also expressed as a linguistic variable. In the case of fuzzy logic adjustment. These are inferences with several rules.

a. ***Fuzzy rules - rules inference matrix***

The fuzzy rules represent the heart of the controller, and allow to express in linguistic form the input variables of the controller to the control variables of the system.

One type of rule may be for example:

If x_1 is "positive" and x_2 is "zero" then u is "negative"

Wherex_1 and x_2 represent two input variables of the controller such as: the deviation and its variation and u the control. Experience in developing these rules plays an important role.

However, the literature gives us some methods that can be used to collect these rules, which can be classified into two types of studies.

In the case of a conventional system, where the inputs and outputs of the regulator are well defined, it is useless to go through a knowledge extraction step. But in general, the number of rules is small, from four to twenty.

In the case of a complex system, these methods are based on the know-how of the human

operator, who himself describes control strategies.

A graphical representation of all the rules, called the inference matrix or rule matrix, commonly used in the literature, makes it possible to synthesize the heart of the fuzzy regulator.

Table 3.1 shows an example of an inference matrix for the two input linguistic variables x_1 and x_2 and the fuzzy output variable u used in the previous paragraph. The *VP* and *VN*symbols represent Very Negative and Very Positive respectively.

u		x_2		
		N	Z	P
x_1	N	VP	P	Z
	Z	P	Z	N
	P	Z	N	VN

Table 3.1 Inference matrix.

Box (3, 2) of the Table (3.1) represents the rule:

If x_1 is "positive" and x_2 is "zero" then u is "negative"

When all the boxes in the inference matrix are filled (as in the example in Table 3.1), they are complete inference rules, otherwise incomplete inference rules. However, this graphical representation becomes more complex if there are more than three input linguistic variables.

b. ***Linguistic description of the adjustment inferences***

For the presentation of the different possibilities express the inferences, a two-input system (linguistic variables after fuzzification) is chosen.

According to the general indications in the previous paragraph, inferences can be written explicitly using a number of rules [46].

If (x_1 is big negative AND x_2 is about zero)

So (x_R equal big positive), OR

If (x_1 is bignegative AND x_2 is average positive)

So ($x_R$equal mean positive), OR

.

.

.

If (x_{N-1} is big positive AND x_N is big positive)

So (x_R equal big positive), OR

In this case, the inference is composed of twenty five rules. Each fuzzy variable x_1 and x_2 is subdivided into five sets.

c. ***Symbolic description of the inferences of settings***

The linguistic description of the adjustment inferences is generally quite heavy.A certain simplification is achieved by the symbolic description.

If (x_1 is BN AND x_2 is AZ).Then (x_R = BP), OR.

If (x_1 is BN AND x_2 is AP).Then (x_R = MP), OR.

.
.
.

If ($x_{N-1}$is BP AND x_N is BP).
Then (x_R = BP), OR.

d. ***Numerical processing of inferences***

The purpose of the fuzzy reasoning is to determine, according to the input linguistic variables *x*, resulting from the fuzzification of the real input variables *X* and the fuzzy control rules, a fuzzy set of possible values for the output linguistic variable *u*.

An inference rule is activated when the membership factor related to the condition of this rule is not zero.

-For fuzzy logic adjustment. One of the following methods is generally used.

- MAX-MIN inference method.
- MAX-PROD inference method.
- SUM-PROD inference method.

3.2.5.3. ***Defuzzification***

As one has seen, inference methods provide a function for the resulting membership variable u_{Res} (x_R), so this is fuzzy information. Since the controller requires a precise control signal *u* at its input, it is necessary to providea transformation of this fuzzy information into a specific information, this transformation is called defuzzification.

a. ***Defuzzication by center of gravity***

The most commonly used defuzzification method is the determination of the center of gravity of the resulting membership function $u_{Res}(x_R)$ [46].

$$x_R = \frac{\int_{-1}^{1} x_R \mu_{Res}(x_R) dx_R}{\int_{-1}^{1} \mu_{Res}(x_R) dx_R} \tag{3.10}$$

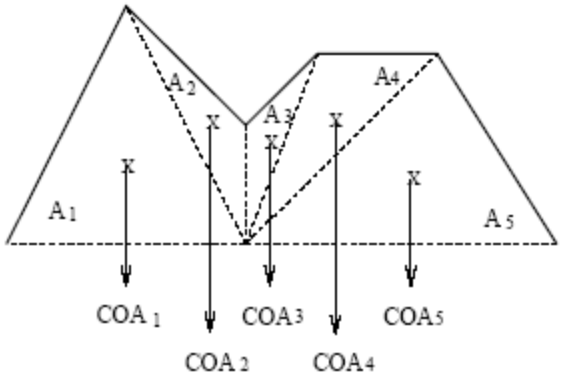


Figure 3.7 Calculation of the center of gravity in the case of simple membership functions.

b. ***Method of maximum***

It consists in taking the abscissa corresponding to the maximum value of the resulting membership function as the output value of the controller. In the case where there are several corresponding abscissas, one then takes their averages.

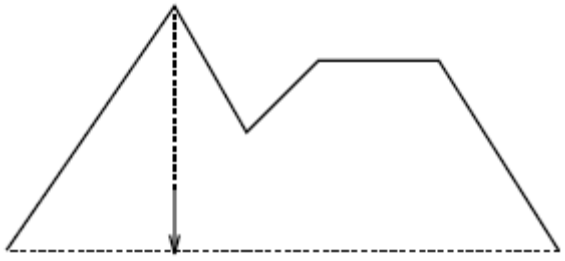


Figure 3.8 Defuzzification by the method of the maximum.

The recent theoretical progress and the growing number of industrial applications (more than 1,500 industrial applications identified in 1994) of fuzzy control seem to give a respectable place to this approach.

Indeed, fuzzy control applications now exist in a wide variety of fields, such as robotics, machine tools, vehicles, railways, domestic applications (air conditioning, shower, cameras, household appliances). . .

It seems that the realization of a fuzzy controller is particularly recommended when the process is poorly known or difficult to describe, due to its complexity. In this case, the ease of implementation depends of course on the application, the difficulty residing in obtaining the expertise related to the application.

There are still some difficult points, particularly with regard to the problem of choosing the techniques specific to fuzzy controllers to be used for a given type of process.This problem is compensated by the ease of implementation of fuzzy controllers and their ability to implement human expertise in the form of rules.

That is why researchers, trying to combine fuzzy logic with other methods to improve fuzzy control, among these methods one has neural networks.

3.3. Neural networks

The principle of artificial neural networks was born in the 1940s from an analogy with the human nervous system. The term today refers to a very large number of models, many of which have not morebig thing to be seen with the functioning of biological neurons, and must therefore be taken as a metaphor. These different models have in common the use of automata, called neurons or units, each capable of performing very simple processing and exchanging information between them.

ANNs are generally associated with a "learning" algorithm that allows the processing carried out to be modified more or less automatically in order to perform a given task.One will only discuss *ANNs* here with the vision that an engineer might have, forgetting aspects related to cognitive science or neurobiology.

3.3.1. Definition of a biological neuron

The brain is made up of about 10^{12} neurons (1000 billion), with 1000 to 10000 synapses (connections) per neuron. The neuron is a cell composed of a cell body and a nucleus. The cellular body branches to form what are called dendrites. These are sometimes so numerous that one speak then of dendritic hair or dendritic arborization. It is through dendrites that information is sent from the outside to the soma, the body of the neuron.

The information processed by the neuron then travels along the (unique) axon to be transmitted to the other neurons. Transmission between two neurons is not direct. In fact, there is an intercellular space of a few tens of Angstroms (10^{-9} m) between the axon of the afferent neuron and the dendrites of the efferent neuron. The junction between two neurons is called the synapse Figure 3.9.

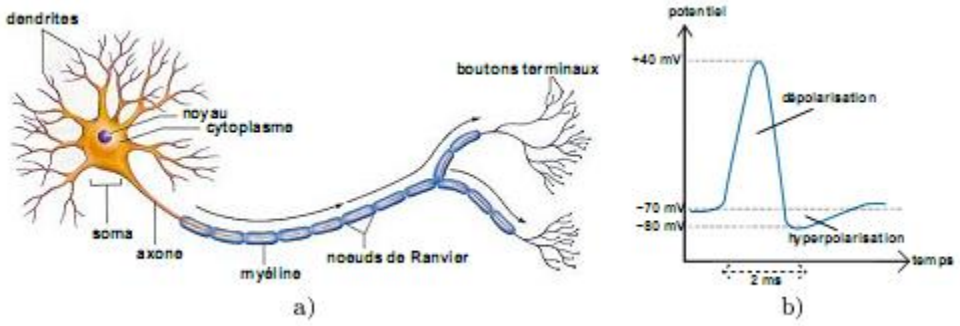


Figure 3.9 A neuron with its dendritic arborization

a) Structure of the neuron, b) Action potential of a neuron.

3.3.2. The artificial neuron:

Figure 3.10 shows the structure of an artificial neuron. Each artificial neuron is an elementary processor. It receives a variable number of inputs from upstream neurons. Each of these inputs is associated with a weight *w* representing the strength of the connection.

Each elementary processor has a unique output, which then branches to feed a variable number of downstream neurons. Each connection is associated with a weight.

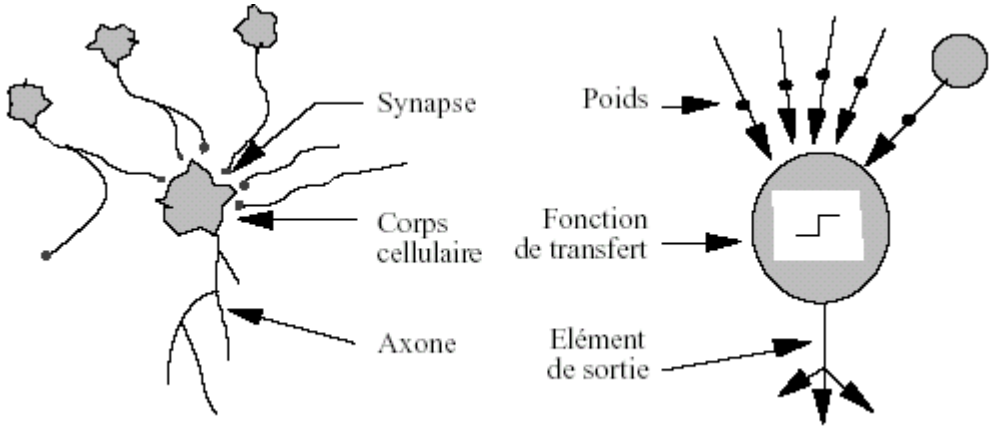


Figure 3.10 Mapping biological neuron to artificial neuron.

3.3.3. Function of the neuron

There are two phases. The first is usually the calculation of the weighted sum of inputs (a) according to the following expression:

$$a = \sum (w_i e_i) \tag{3.11}$$

From this value, a threshold function calculates the value of the state of the neuron. There are many possible forms for the threshold function. The most common are shown in Figure 3.11 Offering an infinity of possible values include in the interval [0, +1] (or [-1, +1]).

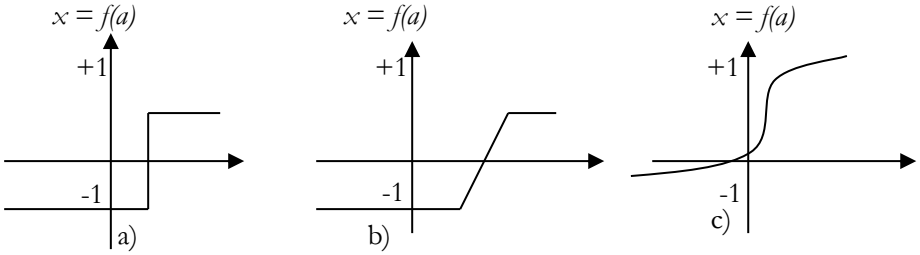


Figure 3.11 Different types of threshold functions for the artificial neuron.

a) Threshold function, b) Linear by pieces, c) Sigmoid.

3.3.4. Formal neuron

3.3.4.1. *Model of Mc Culloch and Pitts* [47]

The formal neuron is a model with approximately all the characteristics of the biological neuron. His first modeling was in 1943 by MAC Culloch and Pitts.

It makes a weighted sum of the potential actions that reach him. Then activate it according to the value found.

If this value exceeds a certain threshold, the neuron is activated and transmits a signal, otherwise it does not transmit it, it is said that the neuron has an activity or nothing. One can represent a formal neuron by the following figure:

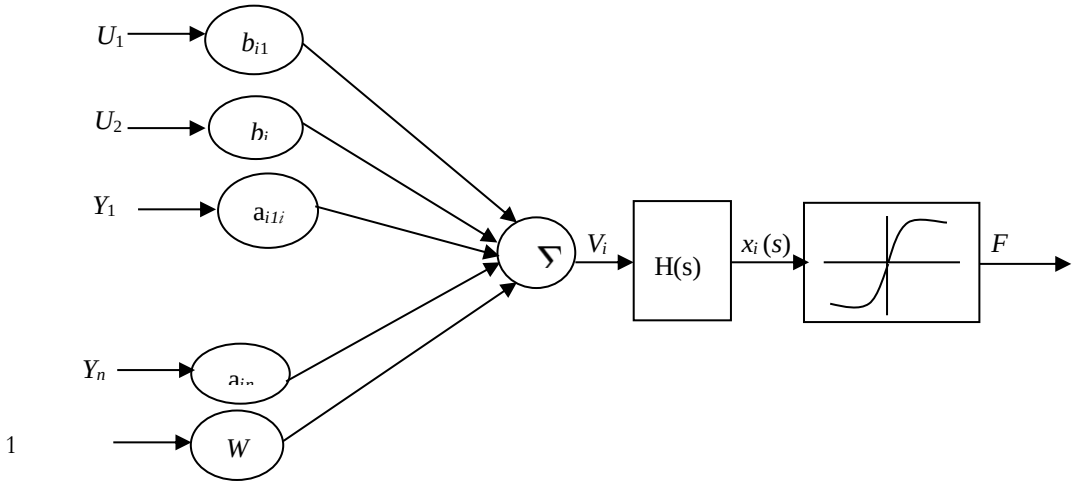


Figure 3.12 General models of neurons.

- The somator is realizes the weighted sum of the inputs of the neuron and is governed by the equation

$$V_i(t) = \sum_{j=1}^{N} a_{ij} Y_j(t) + \sum_{k=1}^{M} b_{ik} U_K(t) + W_i \qquad (3.12)$$

Where I is the index of the neuron in question, V_i (t) the output of the weighting summator of neuron i at time (t), Y_j (t)(j = 1 ... ..N) the outputs of the N neurons connected to the neuron I (where a neuron can have a return on itself), a_{ij} the synaptic weight connecting neuron j to neuron I, $U_k(t)$, k = 1 ... ..M the network inputs, b_{ik} the synaptic weight connecting input K to neuron I and W_i the weight terms bias (its input is 1), this term is added for reasons of convergence.

3.3.5. Some models of ANNs

One presents in this section several models of important networks for the following of our work. One limits this presentation to non-recurring networks [48].

3.3.5.1. *Multilayer ANNs*

It is an extension of the previous one, with one or more layers hidden between the input and the output. Each neuron in a layer is connected to all the neurons of the previous layer and the next layer (except for the input and output layers) and there are no connections between the cells of the same layer. The activation functions used in this type of network are mainly threshold or sigmoid functions. It can solve non-linearly separable problems and more complicated logical problems, including the famous problem of *XOR*. He also follows a supervised learning according to the error correction rule.

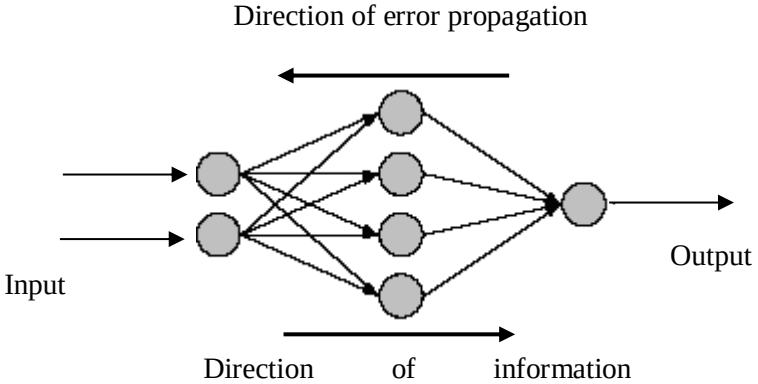


Figure 3.13 A multilayer network with 2 input neurons, 4 hidden neurons and an output neuron.

a. ***Supervised learning of multilayered ANNs***

The classical method for supervised learning consists in giving a set of examples, i.e a finite set of vector pairs (x_i, y_i). In such a pair, x_1 refers to the network input and y_1 the desired output for this input. The function calculated by the network is then written in a parametric form: $f(x, w)$ designates the output of the network when the vector x is presented as its input and it

uses the synaptic weights contained in the vector w. One finally givesFor himself a distance on the output vector space, i.e a means to measure the error committed at a point by the network. One then tries to find the value of w which minimizes the sum of the total error committed by the network, that is to say the sum of the distances between the outputs obtained and the desired outputs, i.e the sum of the $d(f(x_i, w), y_i)$. This error is a function of w and one can therefore use the classical techniques of function optimization to find its minimum.

A.1) Back propagation in the case of a PMC

The sigmoid function of parameter $k > 0$ is defined by:

$$\sigma_i(x) = \frac{e^{Kx}}{e^{Kx}+1} = \frac{1}{1+e^{-Kx}} \tag{3.13}$$

This function is an approximation indefinitely differentiable of the Heaviside threshold function, the better that k is large. One will take $k = 1$ in the following:

$$\sigma_i(x) = \frac{e^{x}}{e^{x}+1} = \frac{1}{1+e^{-x}} \tag{3.14}$$

One can notice that the derivative of the function σ is simple to calculate:

$$\sigma'(x) = \frac{e^{x}}{\left(1+e^{x}\right)^2} = \sigma(x)(1-\sigma(x)) \tag{3.15}$$

It is essential that this calculation is simple because the derivative of this function will be used in the weight update rule by the gradient back-propagation algorithm. For this reason, one sometimes uses the function $th(x)$ which is an approximation of the Heaviside function whose derivative is equal to $1-th^2(x)$. One can now define the networks considered in the following:

Definition

An elementary cell with n real inputs $\vec{x} = (x_1, ..., x_n)$ is defined by the actual synaptic weights $\vec{w} = (w_1, ..., w_n)$ and the output o is calculated by the following formula:

$$o(\vec{x}) = \frac{1}{1+e^{-y}} \quad \text{where} \quad y = \vec{x}\,\vec{w} = \sum_{i=1}^{n} w_i x_i \tag{3.16}$$

A.2) Introduction of the algorithm

The principle of the algorithm is, as in the case of the linear perceptron, to minimize an error function. It is then necessary to calculate the contribution to this error of each of the synaptic weights. It is this step that is difficult. Indeed, each of the weights influences the corresponding neuron, but the modification for this neuron will influence all the neurons of the following layers. This problem is sometimes referred to as the "Credit Assignment Problem".

Either a *PMC* defined by an architecture with n inputs and p outputs, or $\vec{w}$ the vector of the synaptic weights associated with all the links of the network. The error of the *PMC* on anlearning

sample *S* of examples $(\vec{x}^S, \vec{c}^S)$ is defined by:

$$E(\vec{w}) = \frac{1}{2} \sum_{(\vec{x}^s, \vec{c}^s) \in S} \sum_{K=1}^{p} (c_K^s - o_K^s)^2 \tag{3.17}$$

Where o_K^s is the *k-th* component of the output vector $\vec{o}^S$ computed by the *PMC* on the input $\vec{x}^S$. The error therefore measures the difference between the expected and calculated outputs over the complete sample. One assumes*S* fixed, the problem is to determine a vector $\vec{w}$ which minimizes $E(\vec{w})$. However, in the same way as for the perceptron with the Widrow-Hoff rule, rather than trying to minimize the overall error on the entire sample, one tries to minimize the error on each individual example presentation. The error for an example is:

$$E_{(\vec{x}, \vec{c})}(\vec{w}) = \frac{1}{2} \sum_{K=1}^{p} (c_K - o_K)^2 \tag{3.18}$$

One denotes *E* the function $E_{(x\rightarrow, c\rightarrow)}$, *E* is a function of synaptic weights, to apply the gradient method, one must evaluate the partial derivatives of this function *E* with respect to synaptic weights. The following calculations are easy. The only complication comes from the complexity of the notations and indices used, acomplication due to the structure of the *PMC*.

- Each cell is defined by an index.
- The network has *p* output cells.
- If*i* is the index of an output cell, c_i is the expected output for this cell on the input $\vec{x}$.
- w_{ij} is the synaptic weight associated with the link between cell *j* to cell *i*, which implies that they are on two successive layers by definition of the architecture.
- x_{ij} is the input associated with the link between cell *j* to cell *i*.
- $pred(i)$ is the set of cells whose output is an input of cell *i*, this implies that the cell is not an input cell and that all elements of $pred(i)$ belong to the previous layer of that to which cell *i* belongs,
- y_i the total input of cell *i*, either $y_i = \sum_{j \in pred(i)} w_{ij} x_{ij}$,
- o_i is the output of cell *i*, either $o_i = \sigma(y_i)$,
- *Succ*(*i*) is the set of cells that take as input the output of cell i, this implies that the cell is not an output cell and that all elements of *Succ*(i) belong to the next layer of the one to which cell *i* belongs.

One now has to evaluate $\partial E(\vec{w}) / \partial w_{ij}$ which one will note

$\partial E / \partial w_{ij}$. First of all, w_{ij} can only influence the output of the network through the calculation

of the quantity y_i, which authorizes to write that:

$$\frac{\delta E}{\delta w_{ij}} = \frac{\delta E}{\delta y_i}\frac{\delta y_i}{\delta w_{ij}} = \frac{\delta E}{\delta y_i}x_{ij} \tag{3.19}$$

So one needs only to compute $\partial E / \partial y_i$, for that one will distinguish two cases: the case where cell *i* is an output cell and the case where it is an internal cell.

Cell *i* is an output cell:

In this case, the quantity y_i can only influence the output of the network by calculation of o_i. So one has:

$$\frac{\delta E}{\delta y_i} = \frac{\delta E}{\delta o_i}\frac{\delta o_i}{\delta y_i} \tag{3.20}$$

One will now calculate each of the two partial derivatives appearing in equation (3.20). For the first of these two derivatives one has:

$$\frac{\delta E}{\delta o_i} = \frac{\delta}{\delta o_i}\frac{1}{2}\sum_{K=1}^{p}(c_i - o_i)^2 \tag{3.21}$$

Only the term corresponding to *k*=*i*has a non-zero derivative, which finally gives us:

$$\frac{\delta E}{\delta o_i} = \frac{\delta}{\delta o_i}\frac{1}{2}(c_i - o_i)^2 = -(c_i - o_i) \tag{3.22}$$

For the second of the two derivatives of equation (3.20), using the definition of the calculation of the output of an elementary cell and the formula for calculating the derivative of the sigmoid function given in(3.21), one has :

$$\frac{\delta o_i}{\delta y_i} = \frac{\delta \sigma(y_i)}{\delta y_i} = \sigma(y_i)(1-\sigma(y_i)) = o_i(1-o_i) \tag{3.23}$$

By substituting the results obtained by equations (3.22) and (3.23) in equation (3.14), one obtains:

$$\frac{\delta E}{\delta y_i} = -(c_i - o_i)o_i(1-o_i) \tag{3.24}$$

Cell *i* is an internal cell

In this case, the quantity y_i will influence the network by all the calculations of the cells of the set $Succ(i)$. One then has:

$$\frac{\delta E}{\delta y_i} = \sum_{K\in Succ(i)}\frac{\delta E}{\delta E_K}\frac{\delta y_K}{\delta y_i} = \sum_{K\in Succ(i)}\frac{\delta E}{\delta y_K}\frac{\delta y_K}{\delta o_i}\frac{\delta o_i}{\delta y_i} = \sum_{K\in Succ(ii)}\frac{\delta E}{\delta y_K}.w_{Ki}.o_i.(1-o_i)$$

Either still:
$$\frac{\delta E}{\delta y_i} = o_i(1-o_i)\sum_{K\in Succ(i)}\frac{\delta E}{\delta y_i}.w_{Ki} \tag{3.25}$$

By studying these two cases, one obtained two equations (3.24) and (3.25), which allow us to compute the partial derivatives $\partial E / \partial y_i$ for any cell *i*. The calculation should be done for the

output cells and then the cells of the before latest layer to the cells of the first layer. This is why one talks about "back propagation". Thanks to the equation (3.19). One can calculate all the partial derivatives $\partial E(\vec{w})/\partial w_{ij}$. Finally, to deduce the modification to be made on the synaptic weights, one just has to remember that the gradient method tells us that:

$$\Delta w_{ij} = -\varepsilon \frac{\delta E(\vec{w})}{\delta w_{ij}} \tag{3.26}$$

All the elements are therefore in place to allow us to define the algorithm of back-propagation of the gradient.

A.3) The learning step

Back propagation is a gradient descent technique that must converge to the solution, provided that the error surface is continuous. Usually, the step is changed over time (intuitively, it is to reproduce a phenomenon found in psychology on learning subjects, where when the subjects are "new" they tend to absorb the information with very little of critical thinking, while more experienced subjects are more reluctant to acquire new concepts challenging their " culture "), by taking values near to 1 at the beginning of the learning, and by making this coefficient tend towards 0.

3.3.5.2. ***ANN with radial basic functions***

These are the networks that are also calls *RBF* ("Radial Basic Functions"). The architecture is the same as for the *PMC* showever, the basic functions used here are Gaussian functions.

As in the previous model, one finds an organization with an input layer, a hidden layer and an output layer. The main difference is that each hidden neuron only reacts to a small part of the input space (its area of influence).

For a network with *n* inputs and *m* hidden units, the activation of hidden neurons is given by a Gaussian type function (the input and activation functions are confused):

$$a_i = \exp\left(-\tfrac{1}{2} \sum_{K=1}^{n} \frac{\left(e_k - c_{k,i}\right)^2}{\sigma_{k,i}^2} \right) = \prod_{k=1}^{n} \exp\left(-\tfrac{1}{2} \frac{\left(e_k - c_{k,i}\right)^2}{\sigma_{k,i}^2} \right) \tag{3.27}$$

Where*i* denotes the index of the neuron, *k* traverses the set of inputs denoted e_k, and $c_{k,i}$ and $\sigma_{k,i}^2$ are parameters called respectively centers and variances of the Gaussians.

Figure 3.14shows the shape of this activation function for a neuron with a single input.

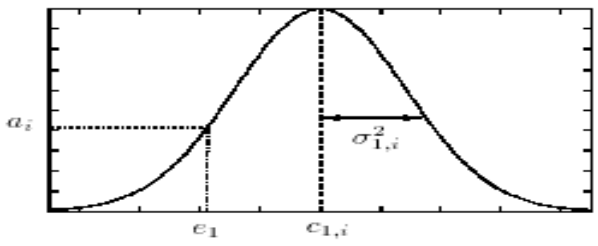


Figure 3.14 Activation function of a hidden neuron with a single input.

Each of these neurons is therefore activated significantly, only for input values relatively close to Gaussian centers.

Connections from input neurons are not weighted. The activation of an output neuron of index *i* is given by:

$$a_i = \frac{\sum_{j=1}^{m} w_{ij} a_j}{\sum_{j=1}^{m} a_j} \tag{3.28}$$

Where*j* goes through all the indices of hidden neurons. Neurons of this type therefore realize a weighted sum of the activation values of the hidden neurons.

The term $\sum_{j=1}^{m} a_j$ called normalization factor is not mandatory. One speaks of a standardized network when it is used.

a. ***Learning***

Learning takes place in these networks by modifying the weights of connections between hidden and output neurons, and the centers and variances of the Gaussians. As before, a gradient descent is performed in order to minimize the quadratic error, the expression of which is given by equation (3.17). The modifications of the different parameters are given by the following rules [48].

$$\Delta w_{ij} = -\eta \frac{\partial Q}{\partial w_{ij}}, \quad \Delta c_{i,k} = -\eta \frac{\partial Q}{\partial c_{i,k}} \quad and \quad \Delta \sigma_{i,k} = -\eta \frac{\partial Q}{\partial \sigma_{i,k}} \tag{3.29}$$

It is also possible, as in the case of equation(3.29), to introduce a term proportional to the last modification of these parameters.

By posing $R = \sum_{j=1}^{m} a_j$ (normalization factor) one gets:

$$\frac{\partial Q}{\partial w_{ij}} = \frac{\partial Q}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} = (a_i - o_i) \frac{a_j}{R} \tag{3.30}$$

Where*i* is the index of an output neuron and *j* is the index of a hidden neuron.

$$\frac{\partial Q}{\partial c_{j,k}} = \frac{\partial Q}{\partial a_j} \frac{\partial a_j}{\partial c_{j,k}} \quad and \quad \frac{\partial Q}{\partial \sigma_{j,k}} = \frac{\partial Q}{\partial a_j} \frac{\partial a_j}{\partial \sigma_{j,k}}$$

Where *j* is the index of a hidden neuron.

$$\frac{\partial Q}{\partial a_j} = \sum_i \frac{\partial Q}{\partial a_i} \frac{\partial a_i}{\partial a_j} = \sum_i (a_i - o_i) \frac{w_{ij} R - \sum_k w_{ik} a_k}{R^2} \tag{3.31}$$

Where *i* goes through the indices of the neurons to which the neuron *j* sends connections (output neurons).

$$\frac{\partial a_j}{\partial c_{j,k}} = a_j \frac{e_k - c_{j,k}}{\sigma_{j,k}^2} \tag{3.32}$$

$$\frac{\partial a_j}{\partial \sigma_{j,k}} = a_j \frac{\left(e_k - c_{j,k}\right)^2}{\sigma_{j,k}^3} \tag{3.33}$$

It is frequent to use only learning on the weights w_{ij} of the connections between the hidden layer and the output layer. One can note that the network output is linear with respect to these parameters, which reduces the risk of blocking in a local minimum of the error function during learning. In this case, the variances and centers of Gaussians are in this case chosen in such a way as to uniformly cover the desired input domain.

3.3.5.3. ***CMAC-type ANN*** **[49]**

The *CMAC* type networks (for Cerebellar Model Articulation Control), introduced by J. S. Albus.

a. ***Structure***

The functioning of this model is very simple and relatively far from that of the other models [48].

It is based on several different discretizations of the input space in the form of grids slightly offset from each other (see Figure 3.15).

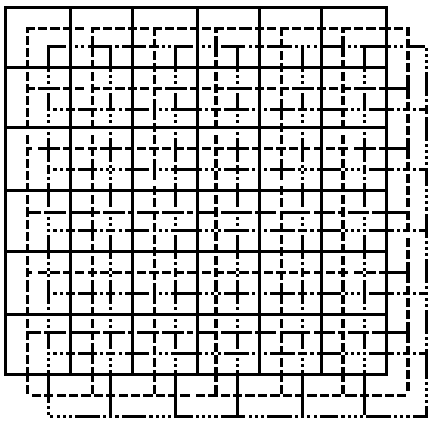


Figure 3.15 Discretization of a two-dimensional input space by a CMAC-type network.

Each of these "divisions" defines a set of cells (also called receptor fields) to which the input neurons of the network are associated (each cell is associated with a neuron).

Each of these neurons has a binary activation degree (i.e. 0 or 1) and is activated when the input value presented to the network is is located in the corresponding cell.

The network is also composed of a set of output neurons that receive weighted connections from each input unit, and average the values transmitted by those connections.

The learning is simply done in this network, by modifying the weights of the connections by a quantity proportional to the output error.

b. ***Properties***

The organization of the discretizations is such that the sets of cells activated by two adjacent input values have a large number of common elements. The network outputscorresponding to these two inputs are therefore also quite close since a part of the weights intervener in their calculation is the same in both cases.

On the other hand, the neurons activated by two very dissimilar inputs are all different, and the two corresponding output values can therefore be very far apart.

This characteristic is obtained thanks to the successive shift of the grids (see Figure 3.16). This topology gives the network a property of local generalization identical to that of RBF networks type. The modification of the weight associated with a connection originating from a given neuron only affects the output calculated by the network for the input values that activate thisneuron.

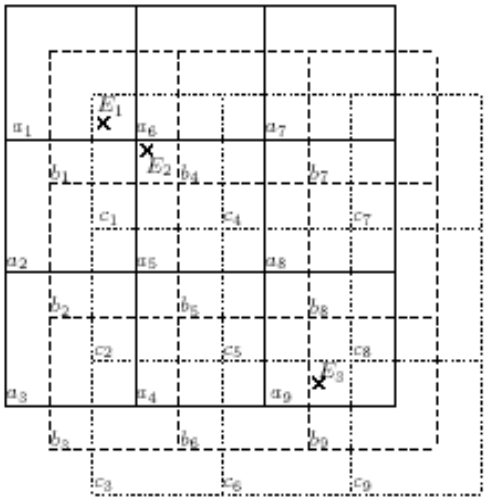


Figure 3.16 Local generalization property of a CMAC network.

The calculations of the values corresponding to the inputs E_1 and E_2 both of which are involved the weights associated with the cells b_1 and c_1 but different by the use of the weights associated with the cells a_1 and a_5. On the other hand, the inputs E_1 and E_3 do not activate any common cells.

One can note that the *ANNs* formalism is not at all essential for this type of network whose operation can be seen as the application of a simple interpolation mechanism between different memory tables.

3.3.6. ANN training by Kalman Filter

One ends this part by Kalman filter, which is the set of mathematical equations, is considered as one of the important discoveries in the control theory principles. E. Kalman's article [50] was published in the year 1960. Its most immediate applications were in control of complex dynamic systems, such as manufacturing processes, aircrafts, ships or spaceships (it was part of the Apollo onboard guidance system). It was and still is frequently used not only in automation, but also in the graphical and economical applications, etc.

However, the Extended Kalman Filter started to appear in the neural network training applications only relatively recently, which was caused by the progress of computer systems development.

Kalman filter has two prinsipal phases predict and update. In the prediction phase, the old state will be modified according to the physical laws (the dynamic or "state transition" model). Next, in the update phase, a measurement of the state is taken. Along with this measurement comes some amount of uncertainty, and its covariance relative to that of the prediction from the previous phase determines how much the new measurement will affect the updated prediction.

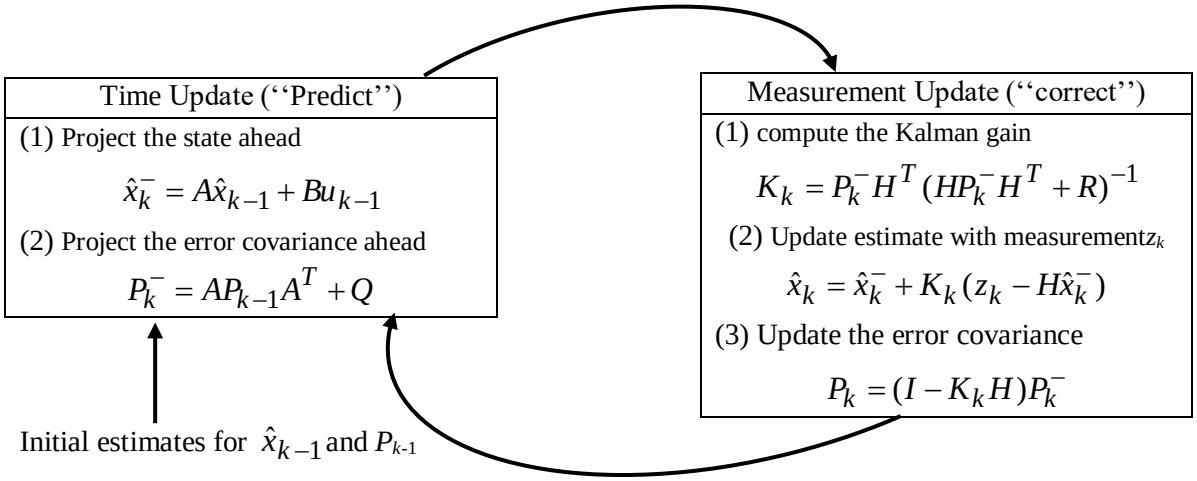


Figure 3.17 Kalman filter phases

Where $\hat{x}_k^-$ is the a priori state estimate, $\hat{x}_k$ is the posteriori state estimate, z_k the measurement state, P_k the error covariance estimation, K_k the Kalman gain, Q, R the process and measurement noise covariance, A the$n \times n$ matrix relating the state K at the previous time step K-1, B the optional control input *matrix, and H* the measurement equation.

When the model is *nonlinear*, which is the case of neural networks, one has to extend Kalman filter using linearization procedure. The resulting filter is then called extended Kalman filter (EKF). A neural network is a nonlinear dynamic system that can by described by the following equations:

$$x_k = x_{k-1} + q_{k-1} \quad and \quad y_k = h(x_k, u_k, v_{k-1}) + r_k$$

The process equation expresses the state of neural network as a stationary process corrupted with the process noise q_k, where the state of the network x consists of network weights.

Measurement equation expresses the desired output of the network as a nonlinear function of the input vector u_k, of the weight vector x_k and for recurrent networks also of the activations of recurrent neurons from previous step v_{k-1}.

3.4. Fuzzy logic and neural networks

Neural networks can deal with imprecise data, and fuzzy logic provides higher-level cognitive features such as approximate reasoning. Since 1988, various associations of these two methods

were developed and are generally used to solve problems of classification, random processes as well as for the identification and control of complex systems.

Systems based on neural networks and fuzzy logic make good function approximations from sample data.

But they provide one more thing compared to the statistical approach: they do not need a priori knowledge about the internal functioning of the system. Fuzzy and neural systems do not need a mathematical model. They are pure estimators.

Neural and fuzzy systems are digital estimators and dynamic systems. Neural theory drowns in the mathematical semantics of dynamic systems, adaptive control and statistics. Fuzzy theory overlaps with these semantics and in addition with probabilities, mathematical logic and measurement theory.

In general, neural and fuzzy systems are used to improve the performance of real systems [51].

3.4.1. Multi-layered neuro-fuzzy networks

The combined use of neural networks and fuzzy logic makes it possible to take advantage of both methods. The learning abilities of the first, the readability and flexibility of the second [52] make their joint use suitable for our purpose.

The use of a multi-layer network for which each layer corresponds to the realization of a step of a fuzzy inference system. A classical architecture can be described as follows (see Figure 3.18):

The input layer receives the values of the input variables.

The units of the first hidden layers calculate by their activation function the degree of truth of the different fuzzy subsets (fuzzification).

- The next layer corresponds to the calculation of the degree of truth of the premises of the different rules.

The last layer or layers of the network perform the defuzzification phase and provide the output values.

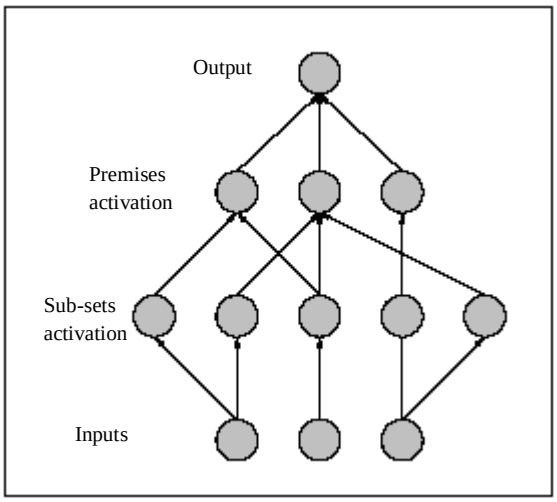


Figure 3.18 An example of a neuro-fuzzy network architecture.

Several achievements of these different steps are possible. It is important to note that the use of a gradient descent learning algorithm (e.g gradient back-propagation) requires the use of derivable activation functions for all network units.

3.4.1.1. ***Coding of fuzzy subsets***

The first layer of an architecture of this type has as many neurons as there are fuzzy subsets in the inference system represented. Each unit calculates the degree of truth of a particular subset by its threshold function. The only restriction on the choice of this function concerns its derivability. One generally finds it in the literature, the use of Gaussian functions similar to those of RBF networks [75]. This is particularly the case in the ANFIS architecture of J-S. R. Jang [53] or in the work of P-Y. Glorennec [54].

In order to keep the formalism of the *ANNs*, the modifiable parameters (i.e., Gaussian centre and slope) are coded by the weights of connections coming from a neuron whose output remains fixed at 1.

Some authors also choose to use multiple units spread over two layers. The membership function is in this case coded by the minimum or the difference of two sigmoids. It always has the shape of a Gaussian but has the advantage of being able to use two different average slopes to the left or right of the center.

3.4.1.2. ***Calculation of the degree of activation of the premises***

The neurons of the hidden second layer each represent the premise of a rule. They receive as input the degree of truth of the different fuzzy subsets composing this premise and are in charge of calculating its own degree of truth. The activation functions used for these neurons depend on the operators present in the rules (conjunction or disjunction) and the logic used:

- Zadeh logic: the minimum and maximum operators are not differentiable. However, they can be approached by derivable functions, called Softmin and Softmax:

$$SOFTMIN(x,y) = \frac{x\exp(-Kx) + y\exp(-Ky)}{\exp(-Kx) + \exp(-Ky)} \tag{3.34}$$

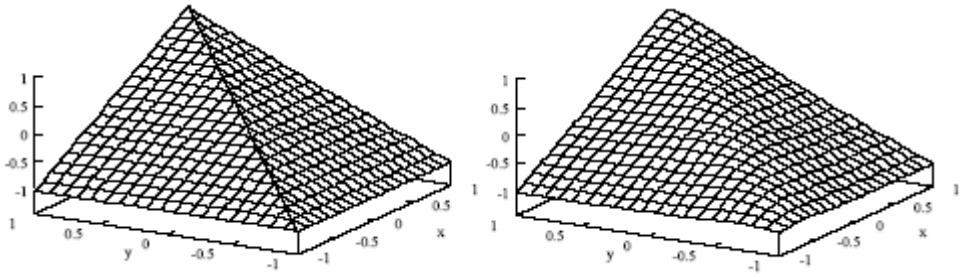
$$SOFTMAX(x,y) = \frac{x\exp(Kx) + y\exp(Ky)}{\exp(Kx) + \exp(Ky)} \tag{3.35}$$

In these two equations, K is a constant that must be chosen as large as possible. Figure 3.19 compares the minimum (x, y) and *softmin* (x, y) functions for K = 10,

- Lukasiewicz logic: the conjunction operator can be obtained using the approximation of the maximum function (0, x) by a sigmoid (see Figure 3.20):

$$Maximum(0,x) \approx \frac{1}{1+\exp(\frac{-(x-0.5)}{0.227})} \tag{3.36}$$

The disjunction operator can be obtained by a similar method.



Minimum softmin

Figure 3.19 Minimum (*x*, *y*) and softmin (*x*, *y*) functions.

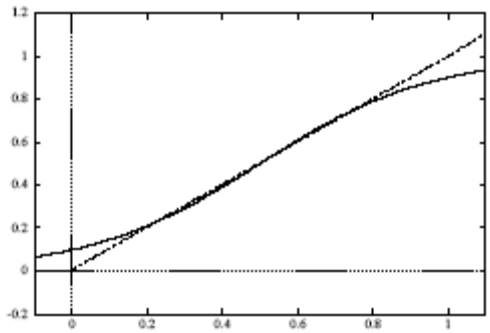


Figure 3.20 Approximation of the function Maximum (0, *x*) (dashed) by a sigmoid (in a continuous line).

Probabilistic logic: the operators of probabilistic logic are derivable and therefore do not pose problems.

3.4.1.3. ***Inference and defuzzification***

These two phases are generally realized by the units of the last layer (or layers) of the network. The threshold functions used depend on the type of rules chosen:

- Mamdani type rules: the many defuzzification methods proposed for rules of this type can not generally be expressed by a derivable expression. H. R. Berenji and P. Khedar [55] provide a particular solution in the case where the membership functions of the fuzzy subsets appearing in the conclusions are triangular functions. A first layer of neurons calculates a numerical value for each rule using the following expression:

$$output = S + \frac{1}{2}(L_d - L_g)(1 - \mu_{premise}) \tag{3.37}$$

Where S is the abscissa of the summit of the triangular membership function, L_d and L_g are respectively the lengths of the subset portions to the right and left of S, and $\mu_{premise}$ is the degree of truth of the premise of the rule.

This value corresponds to the average of the area for which the fuzzy subset has a maximum value (see Figure 3.21).

A last layer of neurons (the output layer) produces a normalized sum of the values thus obtained for the different rules. This defuzzification method is called local average maxima.

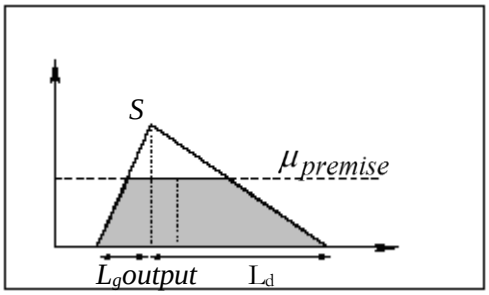


Figure 3.21 Defuzzification by local average maxima. The fuzzy subset corresponding to the result of the rule is shown in gray.

3.4.1.4. ***Examples of the multilayer network***

a. ***Example of Mamdani method***

It is the simplest in its rule form, but it is tedious to put in the form of nonlinear adaptive network (*NAN*).

Description

As shown in Figure 3.22, there are five layers. Each layer represents a part of the fuzzy inference system.

Layer 1: It consists of fixed nodes. Nodes have the same function.

$O_{li} = f_i(f_{li}) = x_{li}$

O_{li} is the output of the nodes (i) of the layer 1, x_{li} is the input of the nodes (i) of the layer 1, and f_i function of the nodes of the layer 1.

This layer distributes the information of the input.

Layer 2: It consists of nodes with adjustable parameters. These nodes represent the membership functions associated with the fuzzy values taken by the linguistic variables, which appear in the premises. Each node has a function.

$$y = f_1(x, m, \sigma) = e^{-\left\{\frac{x-m}{\sigma}\right\}^2} \tag{3.38}$$

Layer 3: The nodes of this layer are fixed, and perform functions simulating the inference engine. Each node has the function.

$$y = f_3(x_1, x_2, ...x_n) = \min(x_1, x_2, ...x_n) \tag{3.39}$$

Layer 4: This layer is a little special, because it works in two modes:

Mode 1: It performs the aggregation, so each node has the function.

$$y = f_4(x_1, x_2, ...x_n) = \min\left(1, \sum_{i=1}^{n} x_i\right) \tag{3.40}$$

Mode 2: It is used for learning, and it receives its inputs from layer 5.

Layer 5: There are two types of nodes in this layer:
Type 1: Nodes of this type perform defuzzification and have the function:

$$y = f_5((x_1, x_2,.....,x_n),(m_1, m_2,.....,m_n),(\sigma_1,\sigma_2,.....,\sigma_n)) = \frac{\sum_{i=1}^{n} x_i \sigma_i x_i}{\sum_{i=1}^{n} m_i \sigma_i} \quad (3.41)$$

Type 2: Nodes of this type are used for learning. They receive the desired ones and transmit them to layer 4.

They have the function.

$$y = f_5(f_d) = y_d \quad (3.42)$$

Where $y$is the output of the node, x_1, x_2,....., x_n are the node inputs, y_d the desired output, and m,σ,m_1, σ1,...,m_n, σn the parameters.

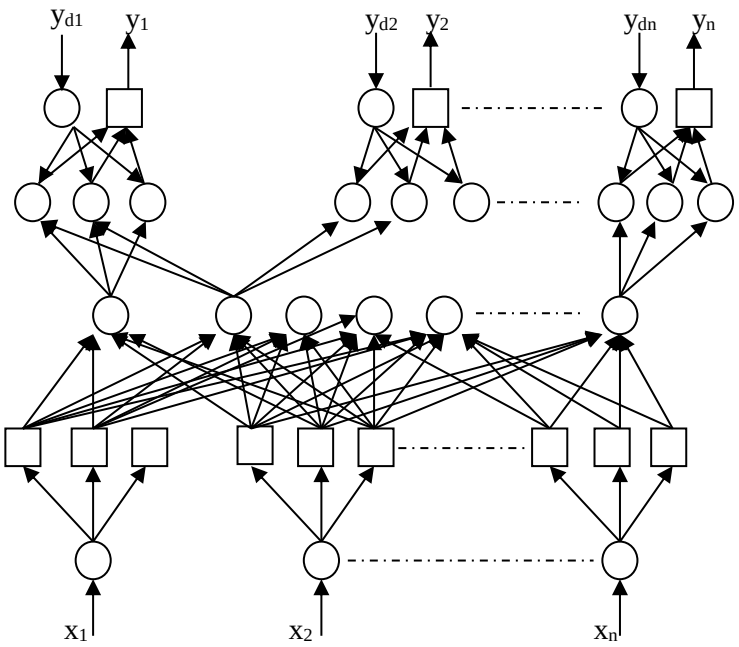


Figure 3.22 Fuzzy rule system in the form of a *NAN* (Mamdani).

A.1) Learning

After the classification of the learning data, the back-propagation algorithm is used, which consists in minimizing the following quadratic error:

$$E = \frac{1}{2}(y_d - y)^2 \quad (3.43)$$

Where y_d is the desired output and $y$the real output.

b. ***Example of sugeno method***

Obtaining fuzzy rules in the form of sugeno is difficult. But, the representation in the form of an *NAN* is very simple.

These rules are the most frequently encountered in neuro-fuzzy applications. Since the output is only a real constant, it is sufficient to code its value by the weight of the link between the neuron calculating the degree of truth of the premise and a neuron calculating the output value.

The latter computes, once again, a normalized sum of the values of the different rules as described by the equation (3.28).

- Sugeno type rules of higher order: when the output of the rules used is a function of the inputs, it is sufficient to add a layer of neurons calculating this function.

B.1) Description

The network is made up of five layers. The shape of such a network is shown by the example of Figure 3.13. This example presents the two fuzzy rules, in the form of sugeno, as follows:

IF ((*x is* A_1) *and* (y *is* B_1)) *So* ($z=p_1x+q_1y+r_1$)
IF ((*x is* A_2) *and* (y *is* B_2)) *So* ($z=p_2x+q_2y+r_2$)

Layer 1:

Each node of this layer has adjustable parameters. The function of the node is identical to the membership function to a fuzzy subset of the universe ofdiscourse of the inputs.

$$O_i^1 = f_i^1(x) = \mu_{Ai}(x) = w_i^1 \tag{3.44}$$

Where x the input of the node and A_i the linguistic value associated with μ_{Ai}

We often take

$$\mu_{Ai}(x) = \exp\left\{-\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}\right\} \tag{3.45}$$

Or

$$\mu_{Ai}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}} \tag{3.46}$$

With $\{a_i, b_i, c_i\}$ the node parameters for (1, *i*).

Layer 2: The nodes of this layer are fixed, and they have the functions.

$$O_i^2 = f_i^2(O_i^1, O_{i+2}^1) = f_i^2(w_i^1, w_{i+2}^1) = w_i^1.w_{i+2}^1 = w_i^2 \tag{3.47}$$

Layer 3: The nodes of this layer are fixed, they possess the functions.

$$\delta_i^3 = f_i^3(O_1^2, O_2^2) = f_i^3(w_1^2, w_2^2) = \frac{w_i^2}{w_1^2 + w_2^2} = w_i^3 \tag{3.48}$$

Layer 4: Each node of this layer is adjustable, and has a function.

$$\delta_i^4 = f_i^4(O_1^3, O_2^3; x, y) = f_i^4(w_1^3, w_2^3; x, y) = w_i^3(p_i x + q_i y + r_i) = w_i^4 \tag{3.49}$$

With $\{p_i, q_i, r_i\}$ the adjustable parameters.

Layer 5: It is a single node, fixed and serves as summing.

$$O_i^5 = f_i^5(O_1^4, O_2^4) = f_i^5(w_1^4, w_2^4) = w_1^4 + w_2^4 = w_1^5 = z \tag{3.50}$$

z : the output.

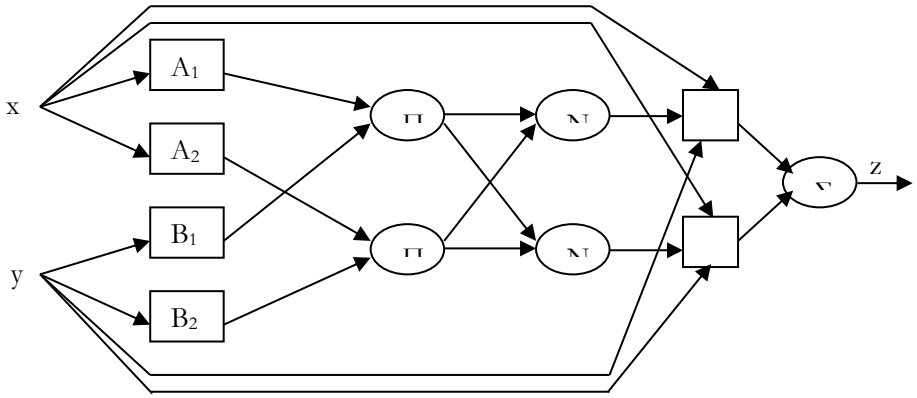


Figure 3.23 System of fuzzy rules in the form of an *NAN* (Sugeno).

Learning

Suppose one has an *NAN* with L layers, and n_k nodes in layer k. Let the function and also the output of the node (k, i). The output of a node depends on the inputs of this node as well as its parameters.

$$O_i^K = f_i^K(O_1^{K-1}, \ldots, O_{n_{K-1}}^{K-1}; a, b, \ldots, c) \tag{3.51}$$

a, b,..., c the node parameter (K, i).

The learning is done by the algorithm of back-propagation by minimizing the error E.

$$E = \sum_{P=1}^{n_i} E_P \tag{3.52}$$

$$E_P = \sum_{m=1}^{n_i} E_P(T_{m.P} - O_{m.P}^L), \qquad 1 \le p \le P \tag{3.53}$$

Where

P :the number of examples to be learned.

$T_{m.p}$:the component of the desired node output (L, m).

$O^L_{m.p}$:the component of the actual output of the node (L, m).

3.5. Conclusion

Fuzzy logic is a very powerful tool that makes it possible to obtain results and generate responses from vague, ambiguous, incomplete and imprecise information, where the mathematical models of the system are unknown or difficult to extract. Thus, the introduction of fuzzy logic into a process is intended to add a dimension to it. Intelligence, inspired by human thought which is a superposition of intuition.

The knowledge acquisition phase presents a very delicate task, especially in the absence of an expert who can provide his *know-how* on the problem to be addressed. For this reason, extensive research has led to the development of systematic and optimal methods for the design of fuzzy controllers. Among these techniques are neuro-fuzzy, which are very powerful competitors in thefield of optimization and learning.

Chapter 04

RESULTSAND DISCUSSIONS

4.1. Introduction

This chapter talks about the implementation of the Neuro-Fuzz-Genetic control of a manipulator arm and a mobile robot. The traditional approach to fuzzy design is essentially based on the knowledge acquired by expert operators formulated as rules. Operators may not be able to transcribe their knowledge and experience into a fuzzy logic controller. Based on these findings, several optimal and systematic methods for designing fuzzy controllers have been developed.

Among these methods one has adopted the connectionist approach of combining artificial neural networks and fuzzy systems to form neuro-fuzzy systems, genetic algorithms are used especially for optimizationof fuzzy subsets.

4.2. The first application

(control of The manipulator robot ROBAI Cyton Gamma 1500)

The ROBAI Cyton Gamma 1500 is a humanoid robot arms, with seven degrees of freedom. It can reach around obstacles and by pass-gaps, reconfigure for strength, and manipulate objects with a clever fluid movement. In this study, we kept only the first three joints with the end effector, see Figure 4.1. The three degrees of freedom are controlled by three DC servo motors, dynamixel MX-64.

Figure 4.1 The ROBAI Cyton arm.

The most general form is [56]

$$M(q)\ddot{q} + H(q,\dot{q}) + G(q) + F(q,\dot{q}) = \tau \qquad (4.1)$$

Where $M(q)$ the 2×2 inertia matrix, $H(q,\dot{q})$ 2×1 Coriolis/centrifugal vector, $G(q)$ 2×1 gravity vector, $F(q,\dot{q})$ 2×1 friction torque, $\dot{q},\ddot{q}$ 2×1 velocity and acceleration vectors and τ 2×1 the generalized joint torque vector.

4.2.1. Trajectory generation

The robot dynamics require from us to impose realizable trajectories. Continuity in position, speed and acceleration makes it possible for the robot to continue the trajectory with realizable controls, as in Figure 4.14.

One can ensure this continuity by a Third-order polynomial trajectories, which is given by the equation

$$q_d(t) = a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3 \qquad t_0 \le t \le t_f \tag{4.2}$$

$$\mathrm{a}_0 = \mathrm{q}_0\,, \quad a_1 = \dot{q}_0, \quad a_2 = \frac{-3(q_0 - q_f) - (2\dot{q}_0 + \dot{q}_f)(t_f - t_0)}{(t_f - t_0)^2},$$

$$a_3 = \frac{2(q_0 - q_f) + (\dot{q}_0 + \dot{q}_f)(t_f - t_0)}{(t_f - t_0)^3}$$

With $t_f$the movement final time, and q_d the reference trajectory, $q_0, \dot{q}_0$ and $q_f, \dot{q}_f$ the initial and final positions and velocities.

4.2.2. Application of the fuzzy control

First one applies a fuzzy control with a Mamdani-type regulator on a robot with two degrees of freedom. The control algorithm is as follows:

1. Login values x_1 and x_2 (Controller inputs)
2. Calculate $mf_1[i]$ and $mf_2[i]$ for all *i*, *j*. (Find the values of all membership functions for input values x_1 and x_2).
3. Calculate *prem*[*i*, *j*]=*min* [$mf_1[i]$, $mf_2[j]$] for all *i* and *j*. (Find the values of the premises of the membership functions x_1 and x_2 given using the *min* operator).
4. Calculate *imps*[*i*, *j*]=*surfimp*[*rule*[i, j], *prem*[*i*, *j*]] for all *i* and *j*. (Find the surfaces below the membership functions for all fuzzy sets involved).
5. Initialisation *num*=0, *den*=0. (Initialize the numerator and denominator values of the gravity center *CG*).
6. *For i* = 0 to *n*, *for j* = 0 to *n*, (Sweeping all surfaces to find the gravity center *CG*).
num = *num* + *imps*[*i*, *j*]* center [*rule*[*i*, *j*]] (Calculate the numerator for the *CG*).
den = *den*+*imps*[*i*, *j*] (Calculate the denominator for the *CG*)
7. Next *i*, Next *j*
8. Output *u*=*num*/*den*. (Output signal for the fuzzy controller).
9. Go to step 1.

4.2.2.1. *Fuzzy controller with three membership functions*

In this step, one uses a PD-type fuzzy controller that contains three membership functions for the error and the variation of the error.

The rules basics.(N: negative, Z: zero, P: positive).

ΔE / E	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

Table 4.1 Rules base

a. ***Membership functions:***

The form of the membership functions associated with the error, the variation of the error, for the two regulators is given in Figure 4.2.

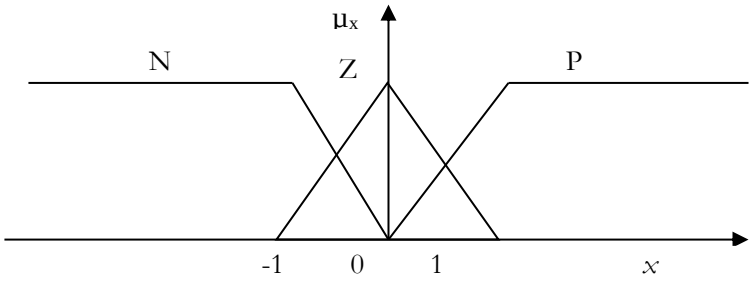


Figure 4.2 Membership functions, FLC of three membership functions.

The graphs in Figure 4.3 represent the two joints of the robot, during the trajectories tracking.

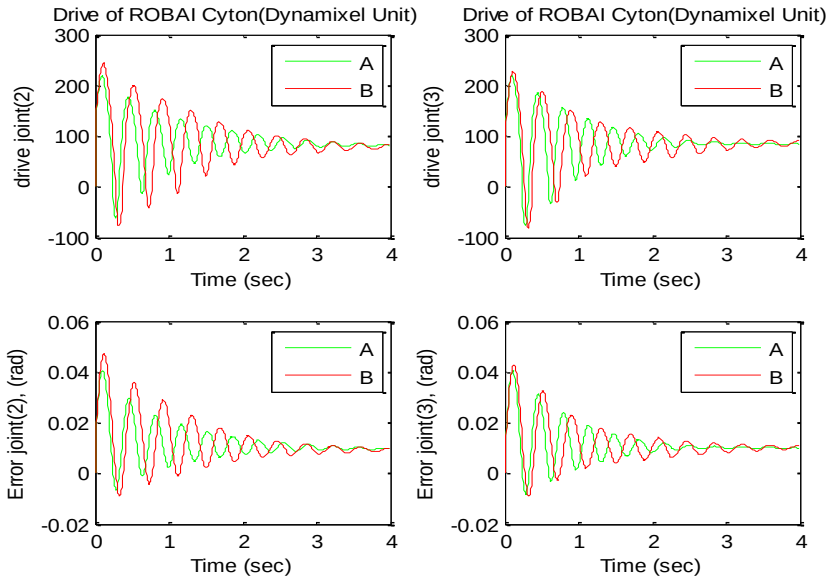


Figure 4.3 Response of the system.(Fuzzy- regulatorof 3 membership functions).
A. without load. B. with load.

From the simulation results obtained one can notice that:

- The joints of the robot follow the desired trajectories, with a maximum error that does not reach 5×10^{-2} radians.
- A quasi-linear relationship between the error and the control.
- The oscillations at the start of the robot are due to the inertia of the arm.

In order to test the robustness of the controller onecarries out the following tests:

b. ***Load test:***

In this case the FLC (3 × 3) was tested for a sudden change in the load carried by the robot. This is achieved by letting load of 0.18 and 0.36 Kg at 3.6 seconds. It is concluded from Figure 4.4 that the two regulators provide a higher torque to compensate the effect of the additional load. The fall of the latter during the movement does not affect the system, because of the response speed of the regulators.

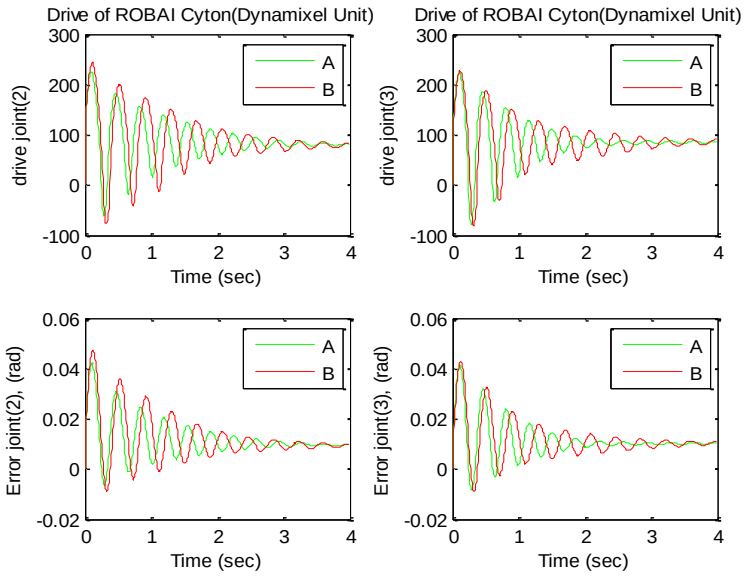


Figure 4.4 Response of the system.(Fuzzy- regulatorof 3 membership functions). A. letting load of 0.18 Kg, B. letting load of 0.36 Kg.

c. ***Drive rupture:***

This test consists of simulating a failure of one of the two regulators controlling the robot during its movement. One introduces a fault on regulator 1 at t = 2 seconds.

The simulation results obtained in Figure 4.5 show that despite the divergence of the first joint caused by the failure introduced on the first regulator, the second follows its reference trajectory with an acceptable error. This illustrates the advantage of decentralized control over centralized control.

To represent the influence of the number of membership functions, and the rules on the control law oneincreases the number of membership functions.

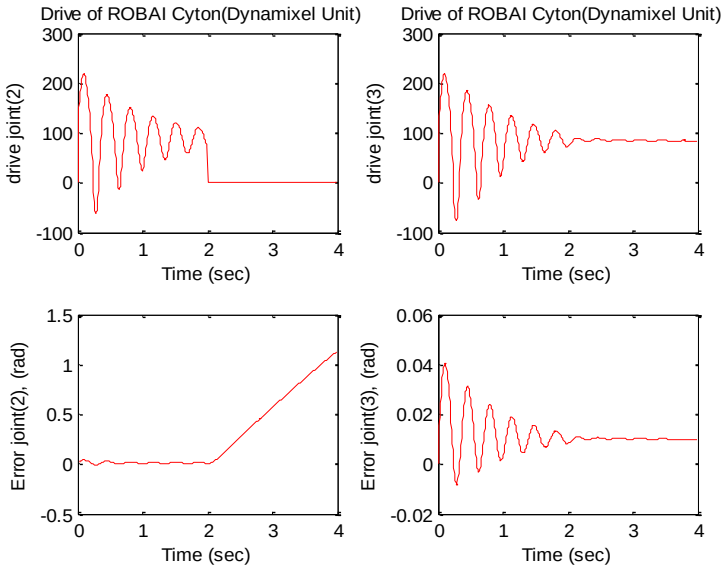


Figure 4.5 Response of the system.(Fuzzy- regulatorof 3 membership functions). Rupture drive.

4.2.2.2. ***Fuzzy controller with five membership functions***

In this case one uses 5 membership functions, with 25 rules in the knowledge base.

The forms of the membership function and the normalization gains are the same as those of the 3 membership functions regulator.

The rules basics.

ΔE / E	NB	NS	ZE	PS	PB
NB	NB	NB	NB	NS	ZE
NS	NB	NB	NS	ZE	PS
ZE	NB	NS	ZE	PS	PB
PS	NS	NE	PS	PB	PB
PB	ZE	PS	PB	PB	PB

Table 4.2 Basic rules, FLC (5 × 5).

Membership functions:

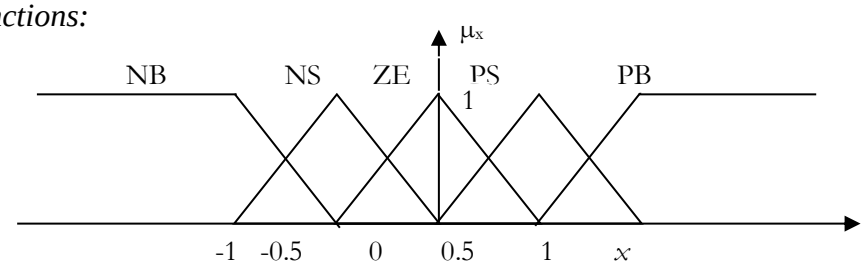


Figure 4.6 Membership functions, FLC of 5 membership functions.

To be able to compare the performance of the synthesized controls, one does the same performance tests for the three membership functions fuzzy controller.

The simulation results obtained are represented in Figure 4.7, Figure 4.8 where no significant improvement in performance is observed.

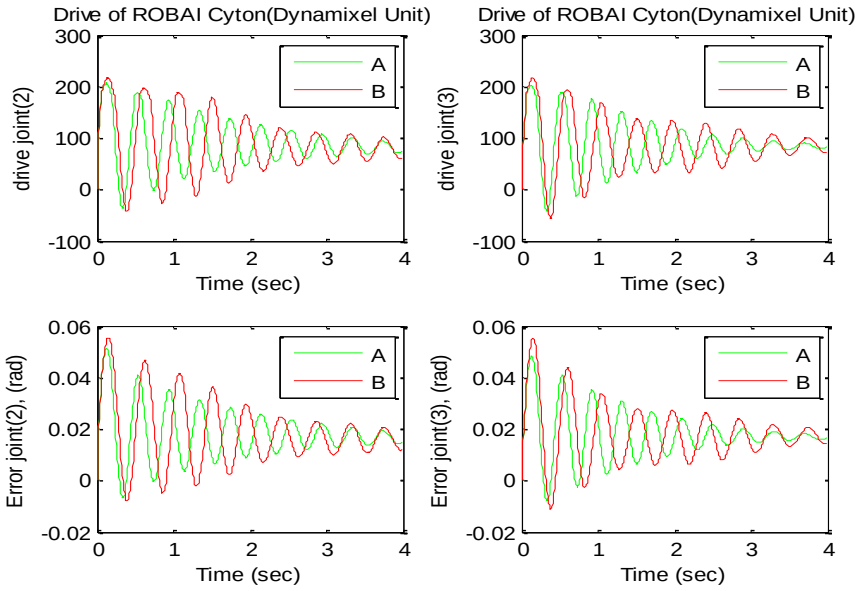


Figure 4.7 Response of the system.(Fuzzy- regulatorof 5 membership functions). A. without load.B.with load.

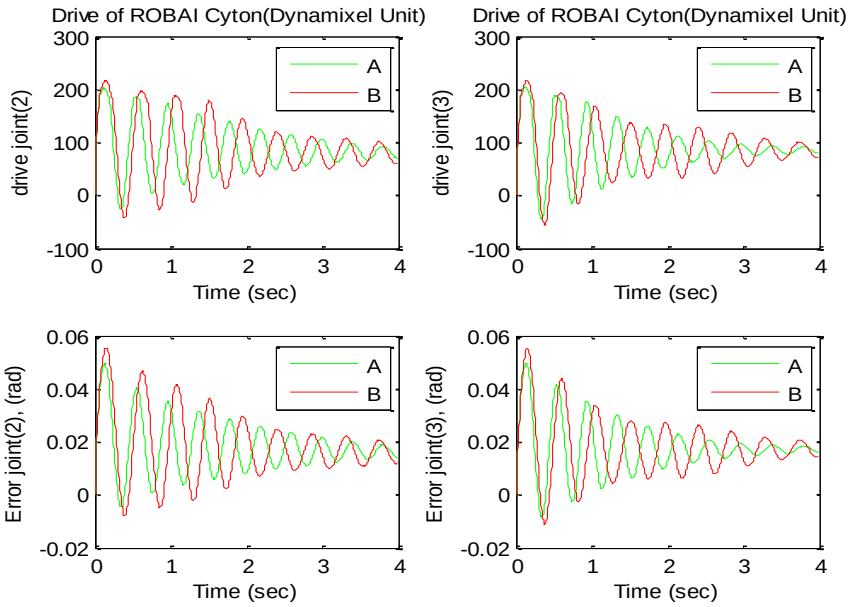


Figure 4.8 Response of the system.(Fuzzy- regulatorof 5 membership functions).
A. letting load of 0.18 Kg, B. letting load of 0.36 Kg.

4.2.2.3. *Fuzzy controller with eleven membership functions*

In this case one has synthesized a fuzzy regulator that has 121 rules in the knowledge base. One does the same thing as the 3 and 5 membership functions regulators.

The rules basics.

ΔE / E	NB	NT	NC	ND	NA	ZE	PA	PD	PC	PT	PB
NB	NB	NB	NB	NB	NB	NB	NT	NC	ND	NA	ZE
NT	NB	NB	NB	NB	NB	NT	NC	ND	NA	ZE	PA
NC	NB	NB	NB	NB	NT	NC	ND	NA	ZE	PA	PD
ND	NB	NB	NB	NT	NC	ND	NA	ZE	PA	PD	PC
NA	NB	NB	NT	NC	ND	NA	ZE	PA	PD	PC	PT
ZE	NB	NT	NC	ND	NA	ZE	PA	PD	PC	PT	PB
PA	NT	NC	ND	NA	ZE	PA	PD	PC	PT	PB	PB
PD	NC	ND	NA	ZE	PA	PD	PC	PT	PB	PB	PB
PC	ND	NA	ZE	PA	PD	PC	PT	PB	PB	PB	PB
PT	NA	ZE	PA	PD	PC	PT	PB	PB	PB	PB	PB
PB	ZE	PA	PD	PC	PT	PB	PB	PB	PB	PB	PB

Table 4.3 Basic rules, FLC (11 × 11).

Membership functions:

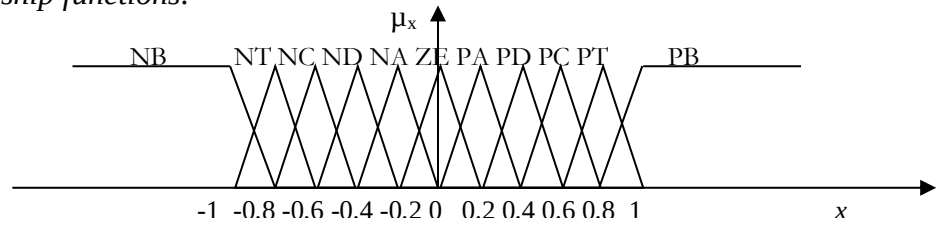


Figure 4.9 Membership functions, FLC of 11membership functions.

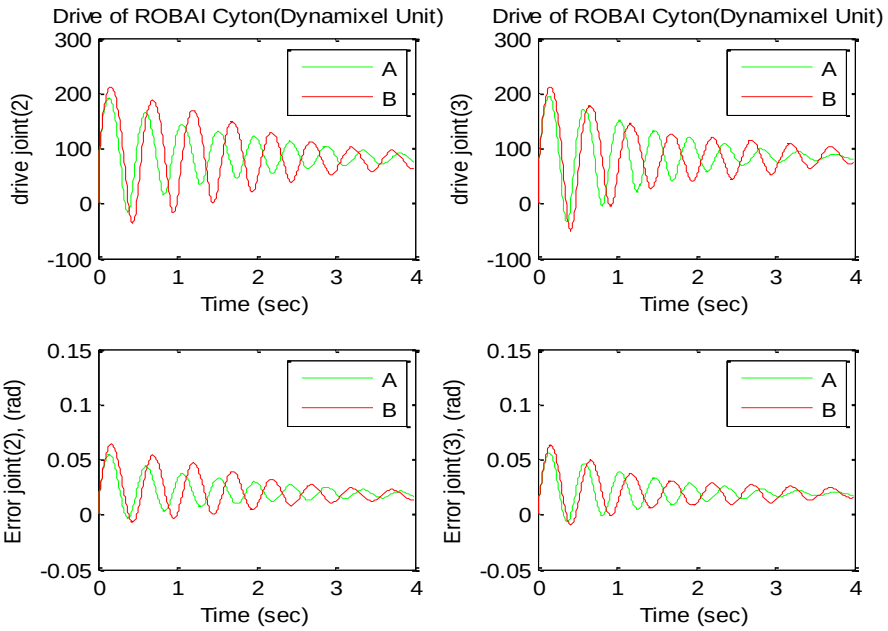


Figure 4.10 Response of the system.(Fuzzy- regulatorof 11 membership functions). A. without load.B.with load.

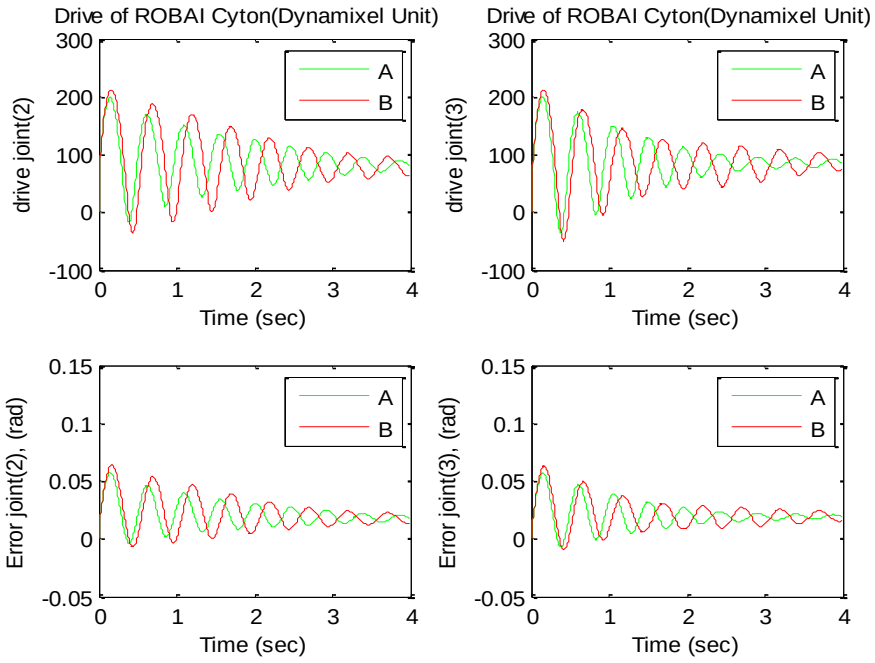


Figure 4.11 Response of the system.(Fuzzy- regulatorof 11 membership functions). A. letting load of 0.18 Kg, B. letting load of 0.36 Kg.

4.2.2.4. ***Comparative analysis***

Looking at the results already presented in Table 4.4, it can be seen that:

The regulator with 3 membership functions give better results than those given by the regulators with 5, and with 11 membership functions.

The drive is always in a quasi-linear relation with the error.

Table 4.4 The tracking errors at the end of trajectory of the two joints.

	joint 2		joint 3	
	Simulation results		Simulation results	
	errors at the end of trajectory (rad)	$\sum\|Drive_2\|$ (Dynamixel unit)	errors at the end of trajectory (rad)	$\sum\|Drive_3\|$ (Dynamixel unit)
a) Fuzzy-regulatorof 3 membership functions, without load	0.0096	5515.6	0.0100	5519.3
b) Fuzzy-regulatorof 3 membership functions, with load	0.0099	5631.6	0.0110	5514.8
c) Fuzzy-regulatorof 5 membership functions, without load	0.0148	5458.3	0.0166	5408.8
d) Fuzzy-regulatorof 5 membership functions, with load	0.0120	5523.4	0.0144	5451.4
e) Fuzzy-regulatorof 11 membership functions, without load	0.0164	5409.1	0.0176	5380.9
f) Fuzzy-regulatorof 11 membership functions, with load	0.0131	5469.5	0.0155	5406.3

4.2.2.5. ***Justification***

• The low value of the error is ensured by the normalization gains at the inputs and the output of the regulator.

• The quasi-linear relationship between error and error variation and control is justified by the fact that oneuses a fuzzy PD controller. This type of controller behaves in the neighborhood of the 0 error and the variation of the error as being a linear regulator having the proportional and derivative actions.

4.2.3. Neuro-Fuzzy Regulator (2×2)

The structure of the regulator in the form of an adaptive network is presented in Figure 4.12.

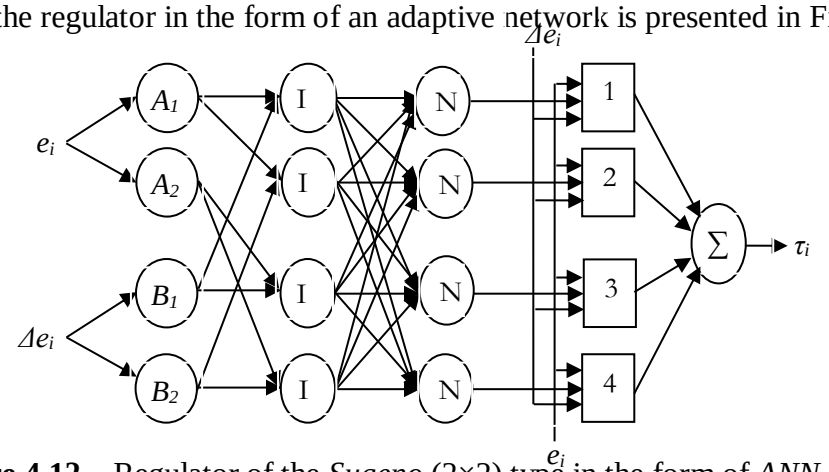


Figure 4.12 Regulator of the *Sugeno* (2×2) type in the form of *ANN*.

It contains, as inputs, the tracking error e_i and the error variation Δe_i for the second and the third joint and, as an output, the drive joint τ_i for each joint.

4.2.4. Control Algorithm

1) *Layer1*:

The output of each node in this layer

$O_{1,n}^{i} = \mu_{A_n}^{i}(e), for\ n = 1,2.$ (4.3)

$O_{1,n}^{i} = \mu_{B_n}^{i}(\Delta e), for\ n = 3,4.$(4.4)

Such as $\mu_A^i$and $\mu_B^i$are related to the membership function form, of fixed parameters associated with the premises.

2) Layer2: The nodes outputs of this layer are expressed by

$$\begin{cases} O_{2,1}^{i} = O_{1,1}^{i}.O_{1,3}^{i} = w_1^i \\ O_{2,2}^{i} = O_{1,1}^{i}.O_{1,4}^{i} = w_2^i \\ O_{2,3}^{i} = O_{1,2}^{i}.O_{1,3}^{i} = w_3^i \\ O_{2,4}^{i} = O_{1,2}^{i}.O_{1,4}^{i} = w_4^i \end{cases} \quad (4.5)$$

3) Layer 3: The node output is given by

$$O_{3,n}^{i} = \overline{w_n^i} = \frac{w_n^i}{\sum_{m=1}^{4} w_m^i} n = 1 \dots 4 \quad (4.6)$$

4) Layer 4: The nodes of this layer have adjustable parameters, their functions are

$$O_{4,n}^{i} = \overline{w_n^i}\left(r_n^i + f_n^i e_i + l_n^i \Delta e_i\right) \quad (4.7)$$

Where r_n^i, f_n^i, l_n^i are the adjustable parameters.

5) Layer 5: This layer contains only one node which carries out the sum of the nodes outputs of the preceding layer

$$O_5^i = \sum_{n=1}^{4} O_{4,n}^{i} \quad (4.8)$$

The parameters of the premises are supposed to be fixed, therefore O_5^i can be written in the form

$$O_5^i = \sum_{n=1}^{4} (\overline{w_n^i}) r_n^i + (\overline{w_n^i} e_i) f_n^i + (\overline{w_n^i} \Delta e_i) l_n^i \quad (4.9)$$

4.2.5. Training Algorithm

The extended Kalman filter cannot only estimate the states of nonlinear dynamical systems from noisy measurements, but can also be used to estimate the parameters of a nonlinear system.

Among the applications of the parameter estimation is the parameter estimation of the neural networks, so one uses the extended Kalman filter as a training algorithm [57,58].

The regulator is characterized by a parameter vector η. The objective is to find the values of the vector η by minimizing the following error [59]:

$$e_i(k) = q_{d_i}(k) - q_i(k) \quad (4.10)$$

The approach of the extended Kalman filter consists at any moment to linearize the output q_i around the estimated vector $\hat{\eta}^i$. The well known form of the solution is

$$\hat{\eta}_n^i(k) = \hat{\eta}_n^i(k-1) + p_n^i(k)\psi_n^i(k)e_i(k) \quad (4.11)$$

Where

$$\psi_n^i(k) = \frac{\partial o_5^i}{\partial \eta^i} \tag{4.12}$$

$$p_n^i(k) = \frac{\alpha_1}{\alpha_2 + \psi_n^{iT} \psi_n^i} \tag{4.13}$$

Where α_1 , α_2 is the adaptation gains for varying the convergence rate.

4.2.6. Neuro-fuzzy control optimized by *GAs*

In this second application, we have applied the neuro-fuzzy system of Sugeno type, similar to the preceding one, to control the second and the third joint of the robot ROBAI Cyton, but we have optimized the various inputs/outputs, which represent the position and the shape of the different membership functions of inputs, and the gains of outputs [41,60,61,62,63].

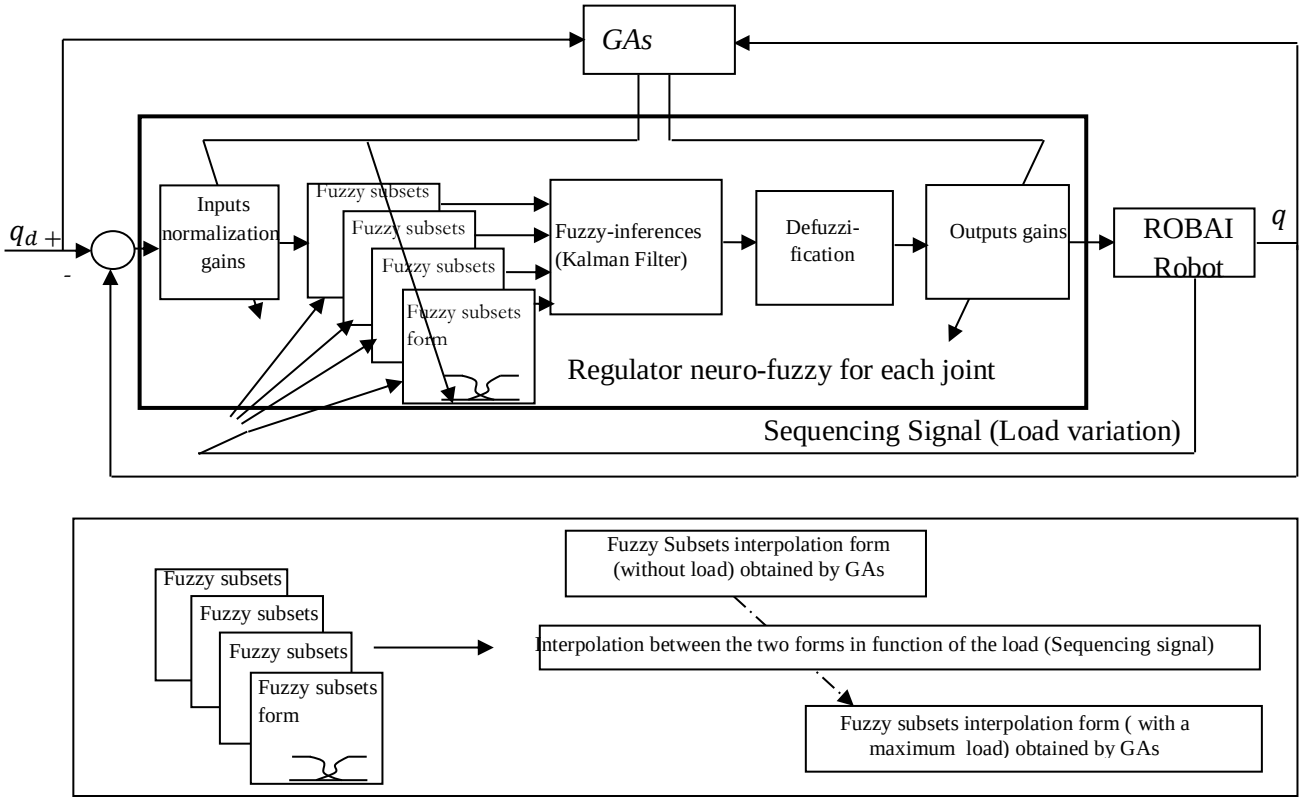


Figure 4.13 Neuro-Fuzzy Subsets Form-Genetic Control algorithm (N-FSF-GC).

In this case, we have at the beginning 6 parameters to be optimized with *GAs*, which are two inputs normalization gains (a_i, b_i) of fuzzification, equation (4.14), see Figure 4.16 B), (where(-a_i, a_i) or (-b_i, b_i) represent the limits of straight line membership) and one output gain of defuzzification for each decentralized regulator. So, one has taken as initial values the absolute values of a_i and$b_i$for the case of a standard regulator neuro-fuzzy *Sugeno* type. The type of the used coding is the binary coding. The objective function is the absolute sum of the two errors on the entire trajectory. Tuned neuro-fuzzy control is shown in Figure 4.13, where the Table 4.6presents the parameters of *GAs*.

And after one takes the inputs normalization gains optimized previously (a_i, b_i) for the error e_i and the variation error (Δe_i) ($i = 2, 3$), and interpolated each straight line (-a_i, a_i),(-b_i, b_i) of the fuzzy subsets by an interpolation function:

-First,with three points, where the two points on the limits are (-a_i, a_i) for the error, and

($-b_i$, b_i) for the variation error, and one point in the middle of the curve which is optimized by the *GAs* of Figure 4.16.C, (c_{1i}, d_{1i}, equation (4.15)).

- Second, with five interpolation points, where the three points inside the curve are optimized by the *GAs* for only the error subsets Figure 4.18, ($c_{ai}, d_{ai}, f_{ai}, f_{bi}, d_{bi}, c_{bi}$ equation (4.16)). For the errors variation form inputs ($\mu(\Delta e_2), \mu(\Delta e_3)$, of Figure 4.18) one keeps the same form as a three points interpolation (equation (4.16)) to reduce the number of optimized points, because one finds that they do not have an influence on the tracking.One always chooses an odd point number to ensure that there is a point in the middle, because the error is around zero [59].

$$\begin{cases} \begin{cases}\mu_{ei1}(e_i)=1\\ \mu_{ei2}(e_i)=0\end{cases} \text{if } e_i \le -a_i, \begin{cases}\mu_{ei1}(e_i)=0\\ \mu_{ei2}(e_i)=1\end{cases} \text{if } e_i \ge a_i, \text{and thus:} \begin{cases}\mu_{ei1}(e_i)=\frac{e_i-a_i}{-2a_i}, \text{if-} a_i \le e_i \le a_i,\\ \mu_{ei2}(e_i)=\frac{e_i+a_i}{2a_i}, \text{if-} a_i \le e_i \le a_i,\end{cases} i=2,3 \\ \begin{cases}\mu_{\Delta ei1}(\Delta e_i)=1\\ \mu_{\Delta ei2}(\Delta e_i)=0\end{cases} \text{if } \Delta e_i \le -b_i, \begin{cases}\mu_{\Delta ei1}(\Delta e_i)=0\\ \mu_{\Delta ei2}(\Delta e_i)=1\end{cases} \text{if } \Delta e_i \ge b_i, \text{and thus:} \begin{cases}\mu_{\Delta ei1}(\Delta e_i)=\frac{\Delta e_i-b_i}{-2b_i}, \text{if-} b_i \le \Delta e_i \le b_i\\ \mu_{\Delta ei2}(\Delta e_i)=\frac{\Delta e_i+b_i}{2b_i}, \text{if-} b_i \le \Delta e_i \le b_i\end{cases} i=2,3 \end{cases} \quad (4.14)$$

$$\begin{cases} \begin{cases}\mu_{ei1}(e_i)=1\\ \mu_{ei2}(e_i)=0\end{cases} \text{if } e_i \le -a_i, \begin{cases}\mu_{ei1}(e_i)=0\\ \mu_{ei2}(e_i)=1\end{cases} \text{if } e_i \ge a_i, \text{and thus:} \begin{cases}\mu_{ei1}(e_i)=\frac{e_i(e_i-a_i)}{2{a_i}^2}+c_{1i}\frac{(e_i+a_i)(e_i-a_i)}{-{a_i}^2}, \text{if-} a_i \le e_i \le a_i,\\ \mu_{ei2}(e_i)=\frac{e_i(e_i+a_i)}{2{a_i}^2}+c_{1i}\frac{(e_i+a_i)(e_i-a_i)}{-{a_i}^2}, \text{if-} a_i \le e_i \le a_i,\end{cases} i=2,3 \\ \begin{cases}\mu_{\Delta ei1}(\Delta e_i)=1\\ \mu_{\Delta ei2}(\Delta e_i)=0\end{cases} \text{if } \Delta e_i \le -b_i, \begin{cases}\mu_{\Delta ei1}(\Delta e_i)=0\\ \mu_{\Delta ei2}(\Delta e_i)=1\end{cases} \text{if } \Delta e_i \ge b_i, \text{and thus:} \begin{cases}\mu_{\Delta ei1}(\Delta e_i)=\frac{\Delta e_i(\Delta e_i-b_i)}{2{b_i}^2}+d_{1i}\frac{(\Delta e_i+b_i)(\Delta e_i-b_i)}{-{b_i}^2}, \text{if-} b_i \le \Delta e_i \le b_i\\ \mu_{\Delta ei2}(\Delta e_i)=\frac{\Delta e_i(\Delta e_i+b_i)}{2{b_i}^2}+d_{1i}\frac{(\Delta e_i+b_i)(\Delta e_i-b_i)}{-{b_i}^2}, \text{if-} b_i \le \Delta e_i \le b_i\end{cases} i=2,3 \end{cases}$$

(4.15)

$$\begin{cases} \begin{cases}\mu_{ei1}(e_i)=1\\ \mu_{ei2}(e_i)=0\end{cases} if\ e_i \le -a_i, \begin{cases}\mu_{ei1}(e_i)=0\\ \mu_{ei2}(e_i)=1\end{cases} if\ e_i \ge a_i, \text{and thus:} \\ \mu_{ei1}(e_i)=\frac{e_i\left(e_i+\frac{a_i}{3}\right)\left(e_i-\frac{a_i}{3}\right)(e_i-a_i)}{2{a_i}^2\left(-a_i-\frac{a_i}{3}\right)\left(-a_i+\frac{a_i}{3}\right)}+c_{ai}\frac{e_i\left(e_i-\frac{a_i}{3}\right)(e_i-a_i)(e_i+a_i)}{2\left(\frac{a_i}{3}\right)^2\left(-a_i-\frac{a_i}{3}\right)\left(a_i-\frac{a_i}{3}\right)}+d_{ai}\frac{(e_i-a_i)(e_i-a_i/3)(e_i+a_i/3)(e_i+a_i)}{{a_i}^2\left(\frac{a_i}{3}\right)^2}+f_{ai}\frac{e_i(e_i-a_i)\left(e_i+\frac{a_i}{3}\right)(e_i+a_i)}{2\left(\frac{a_i}{3}\right)^2\left(-a_i+\frac{a_i}{3}\right)\left(a_i+\frac{a_i}{3}\right)}, if-a_i \le e_i \le a_i, \\ \mu_{ei2}(e_i)=\frac{e_i\left(e_i+\frac{a_i}{3}\right)\left(e_i-\frac{a_i}{3}\right)(e_i+a_i)}{2{a_i}^2\left(a_i-\frac{a_i}{3}\right)\left(a_i+\frac{a_i}{3}\right)}+f_{bi}\frac{e_i\left(e_i-\frac{a_i}{3}\right)(e_i-a_i)(e_i+a_i)}{2\left(\frac{a_i}{3}\right)^2\left(-a_i-\frac{a_i}{3}\right)\left(a_i-\frac{a_i}{3}\right)}+d_{bi}\frac{(e_i-a_i)(e_i-a_i/3)(e_i+a_i/3)(e_i+a_i)}{{a_i}^2\left(\frac{a_i}{3}\right)^2}+c_{bi}\frac{e_i(e_i-a_i)\left(e_i+\frac{a_i}{3}\right)(e_i+a_i)}{2\left(\frac{a_i}{3}\right)^2\left(-a_i+\frac{a_i}{3}\right)\left(a_i+\frac{a_i}{3}\right)}, if-a_i \le e_i \le a_i, \\ \begin{cases}\mu_{\Delta ei1}(\Delta e_i)=1\\ \mu_{\Delta ei2}(\Delta e_i)=0\end{cases} if\ \Delta e_i \le -b_i, \begin{cases}\mu_{\Delta ei1}(\Delta e_i)=0\\ \mu_{\Delta ei2}(\Delta e_i)=1\end{cases} if\ \Delta e_i \ge b_i, \text{and thus:} \begin{cases}\mu_{\Delta ei1}(\Delta e_i)=\frac{\Delta e_i(\Delta e_i-b_i)}{2{b_i}^2}+d_{1i}\frac{(\Delta e_i+b_i)(\Delta e_i-b_i)}{-{b_i}^2}, if-b_i \le \Delta e_i \le b_i\\ \mu_{\Delta ei2}(\Delta e_i)=\frac{\Delta e_i(\Delta e_i+b_i)}{2{b_i}^2}+d_{1i}\frac{(\Delta e_i+b_i)(\Delta e_i-b_i)}{-{b_i}^2}, if-b_i \le \Delta e_i \le b_i\end{cases} i=2,3 \end{cases}$$

(4.16)

4.2.7. Simulation results

The simulation time is4sec, the desired joints trajectory is chosen to be a third-order polynomial function. Joints move simultaneously from $q_{2,3} = 0.3491$ radian to $q_{2,3} = 2.7053$ radian to sweeping the maximum operating range applied on the second and third joint of the robot ROBAI Cyton (see Table 4.5) [59].

Designed control system is highly ineffective when the unstructured and structured uncertainties of the manipulator robot (e.g., external disturbance and changing payload) are present.

From the exactly known behaviour of the linguistic variables, the fuzzy sets are easily characterized by the triangular membership functions.

By applying the control law, the desired joint trajectory is presented in Figure 4.14, the position tracking errors without load and the joints torques are shown in Figure 4.15 A, where the neuro-fuzzy controller is not optimized by *GAs*. From the simulation results provided above (see Table 4.7, a), one draws the conclusion that the proposed controller is quite insensitive to various

uncertainties of the robot dynamics. However, if the uncertainties become unreasonably large the controller will collapse. This can be explained from the fact that under such situations the torques are computed from the trained feedforward neuro-fuzzy and consequently do not hold. The site of the fuzzy subsets before optimization are presented in Figure 4.16 A.

In order to optimize the input gains(a_i, b_i), which represent the normalization gains of fuzzification, and the output gains of deffizification of each neuro-fuzzy-genetic control applied to drive the manipulator robot (Figure 4.13), one uses the *GAs* which are the basis of the optimization stochastic algorithms, where the (Table 4.6) presents the parameters of *GAs*. Thus, Figure 4.15 B illustrates the position tracking errors without load in order to test the performances of the proposed control scheme. Hence, oneconcludes that proposed neuro-fuzzy-genetic controller with robust design yields a better result in comparison to neuro-fuzzy approache (Table 4.7,b).

The neuro-fuzzy- genetic controller has the best structure for the tracking control of the robot with uncertainties. The site of the fuzzy subsets after optimization is presented in Figure 4.16 B.

Figure 4.15 C presents the tracking errors without load after the interpolation of the lines fuzzy subsets by a three points interpolation functions, where the limits of the straight line are (($-a_i$, a_i) ,($-b_i$, b_i)) and just the point in the middle who is optimized by *GAs*. The fuzzy subsets form obtained are shown in Figure 4.16 C.

Figure 4.17 A provide the same information as above without load, but with five interpolation points for the error, where two points always (($-a_i$, a_i) ,($-b_i$, b_i)) and the remainder three points inside the curve are optimized by *GAs*. One can notice from the previous figures that the tracking errors, especially the errors at the end of trajectories, are improved (Table 4.7,d) for each case.

In Figure 4.17 A, one was able to obtain a drive which is perfectly smooth without a mass load because of the fuzzy subsets forms of the errors inputs $(\mu(e_2), \mu(e_3)$,Figure 4.18), but for the errors variation form inputs $(\mu(\Delta e_2), \mu(\Delta e_3)$, Figure 4.18) one finds that they have not an influence, so one keeps the same form as a three points interpolation (equation (4.16)) to reduce the number of optimized points.

When one changes the mass load to 0.36 *Kg*, one finds a small lose of performances appearance, in the case of five points interpolation (Table 4.7,e), because the system is changed. On the other hand, the subsets form obtained with0*Kg* follow exactly the dynamic of 0 *Kg* system, so onetries to find another fuzzy subsets for 0.36 Kg load, for the dominate fuzzy subsets which represent just one between the two fuzzy subsets form for the error input for the first regulator and the same for the second $(\mu(e_2), \mu(e_3)$, Figure 4.18 blue color). Just the point in the middle of the five points interpolation which change because the error around the zero, d_{ai} equation (4.16), the performance becomes better (Table 4.7,f).

One can note from Table 4.7 that the energy dissipated is almost the same for the two joints in the four first cases, which explains that the control curve, whenever it can be linearized around the last case (the curve of five points interpolation Figure 4.17 A), i.e., the controller overcomes the system dynamics. Thus, one can linearize the control with a high performance tracking error.

When one compares this work with that of [25,29], one finds that they used a triangular membership functions (*MFs*), with linguistic hedges to modify the shape of the *MFs*.

The advantage of the method [25,29] is that the number of optimized parameters is reduced and is the same number with the proposed method when oneuses three points interpolation. However the membership function form obtained is not the most appropriate one. The advantage of the

proposed method is that the resulting form would be the most appropriate form with five points interpolation, though the number of optimized parameters is very important and to remedy this latter problem one uses only two fuzzy subsets for each input e_i and Δe_i.

After one has made a generalization, i.e., for each value of load between 0 *Kg* and 0.36 *Kg*, one had a fuzzy subsets forms, between the two values of load 0 *Kg*and 0.36 *Kg* by adjusting the point in the middle of the curve $\mu(e_2), \mu(e_3)$Figure 4.18 figure 4.18 (point around zero of error axis, d_{ai} equation (4.16)), by an interpolation function. One supposes that the value of 0 *Kg* and 0.36 *Kg* are the minimum and the maximum of this function, then tries to find the other values to get the fuzzy subsets related to the load value.

Figure 4.19 presents response of the system after five interpolation points with fall of load from 0.36 *Kg* to 0 *Kg* at 3.6 second (practically at the end). One can see that the errors at the end of the trajectories have been well compensated (Table 4.7,h), this is because one changes the form of the fuzzy subsets following the change of the load; otherwise, one has well understood the dynamics of the manipulator robot with the load variation, comparatively with (Table 4.7,g) when one keeps the same membership functions of 0.36 *Kg*.

Table 4.5 Parameters of ROBAI Cyton.

Parameters Link/Joint	**1**	**2**	**3**
Mass(kg)		0.364	0.1387
Length(m)		0.12583	0.105
Stall Torque 6(N.m) with external gear ratio 5		6×5	6×5

Table 4.6 Parameters of *GAs*.

GAs	**parameters**
Population size	200
Crossover fraction	0.8
Mutation	0.1

Table 4.7 The tracking errors at the end of trajectory of the two joints.

	joint 2				joint 3			
	Simulation results		Practical results		Simulation results		Practical results	
	errors at the end of trajectory (rad)	$\sum\|Drive_2\|$ (Dynamixel unit)	errors at the end of trajectory (rad)	$\sum\|Drive_2\|$(Dynamixel unit)	errors at the end of trajectory (rad)	$\sum\|Drive_3\|$ (Dynamixel unit)	errors at the end of trajectory (rad)	$\sum\|Drive_3\|$ (Dynamixel unit)
a) Before optimization (Fig. 4.15 A)	0.0033	53806	0.0059	46901	0.0015	53442	0.0013	52122
b) After optimization (Fig. 4.15 B)	0.0005	54322	0.0017	48681	0.0012	53157	0.0002	50746
c) After 3 interpolation points (Fig.	0.0004	54778	0.0013	48573	0.0007	53648	0.0002	51270

4.15 C)								
d) After 5 interpolation points (Fig. 4.17 A, without load, membership functions 0 Kg)	0.0002	53862	0.0002	48580	0.0007	53189	0.0002	51932
e) After 5 interpolation points (with load, but membership functions of 0 Kg)	0.0009	53780	0.0017	50148	0.0033	53544	0.0002	51511
f) After 5 interpolation points (Fig. 4.17 B with load, membership functions of 0.36 Kg)	0.0008	53952	0.0002	49614	0.0032	53390	0.0002	51164
g) After 5 interpolation points with load variation at 3.6 *sec* from 0.36 Kg to 0 Kg and one keeps the same membership functions of 0.36 Kg (Fig. 4.18, Fig. 4.19 A)	0.0011	54135	0.0033	50537	0.0026	53349	0.0013	51703
h) After 5 interpolation points with load variation at 3.6 sec from 0.36 Kg to 0 Kg but one changes the membership functions from 0.36 Kg to 0 Kg (Fig. 4.18, Fig. 4.19 B)	0.0007	54153	0.0002	49997	0.0022	53337	0.0013	50843

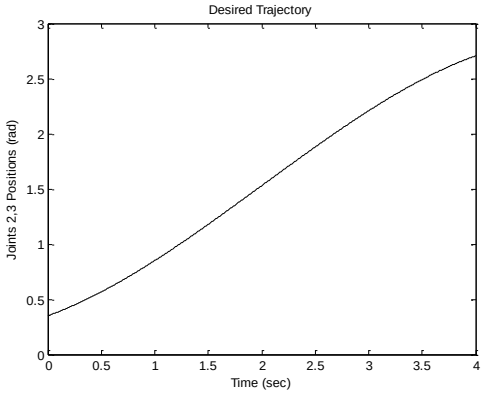


Figure 4.14 The trajectory of the two joints.

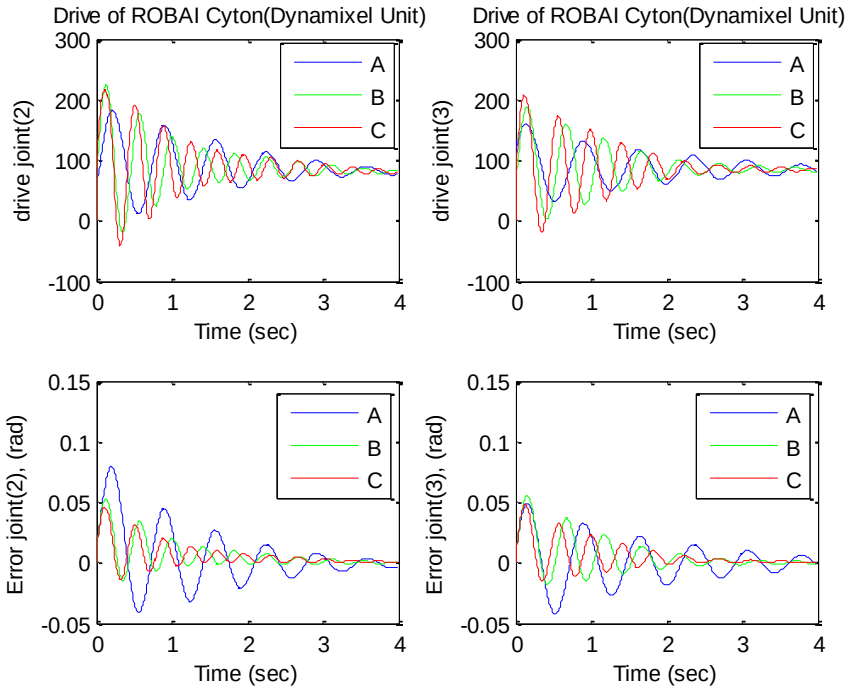


Figure 4.15 Response of the system for a tracking,without load. A. Neuro-fuzzy regulator of the (2×2)Sugeno type, without optimization B. Neuro-fuzzy regulator optimized by genetic of (2×2) Sugenotype, C. Neuro-fuzzy-genetic regulator of (2×2) Sugenotype with three interpolation points, where the point in the middle optimized by *GAs*.

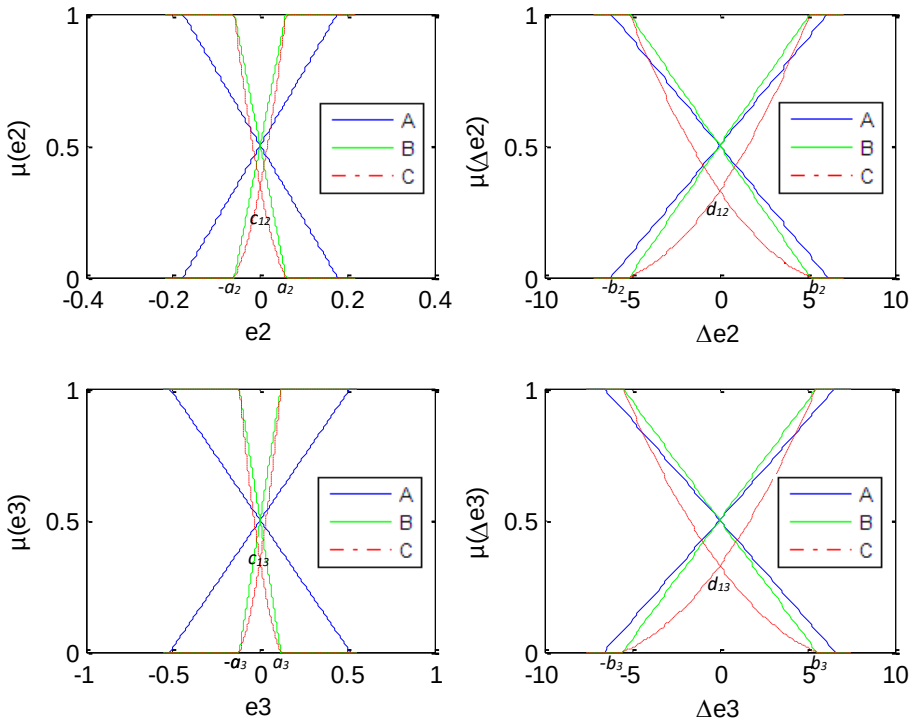


Figure 4.16 The membership functions for the two regulators (e : error, Δ*e* : error variation), A. Before optimization, B. After optimization of the sub-set fuzzy (normalization gains of fuzzification), C. After three interpolation points, where the point in the middle optimized by *GAs*.

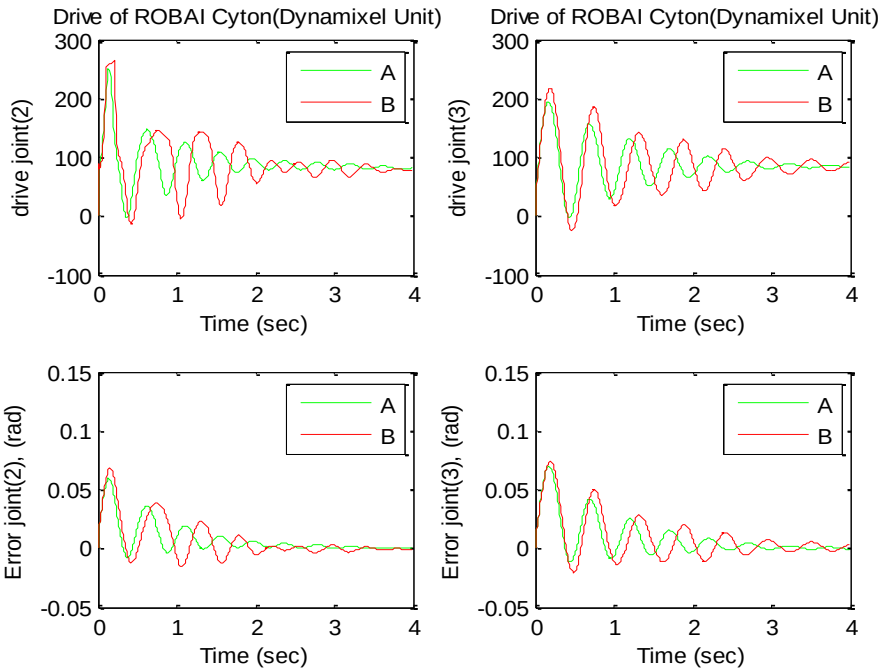


Figure 4.17 Response of the system.(Neuro-fuzzy-genetic regulator of (2×2) Sugenotype), after five interpolation point. A. without load.B.with load.

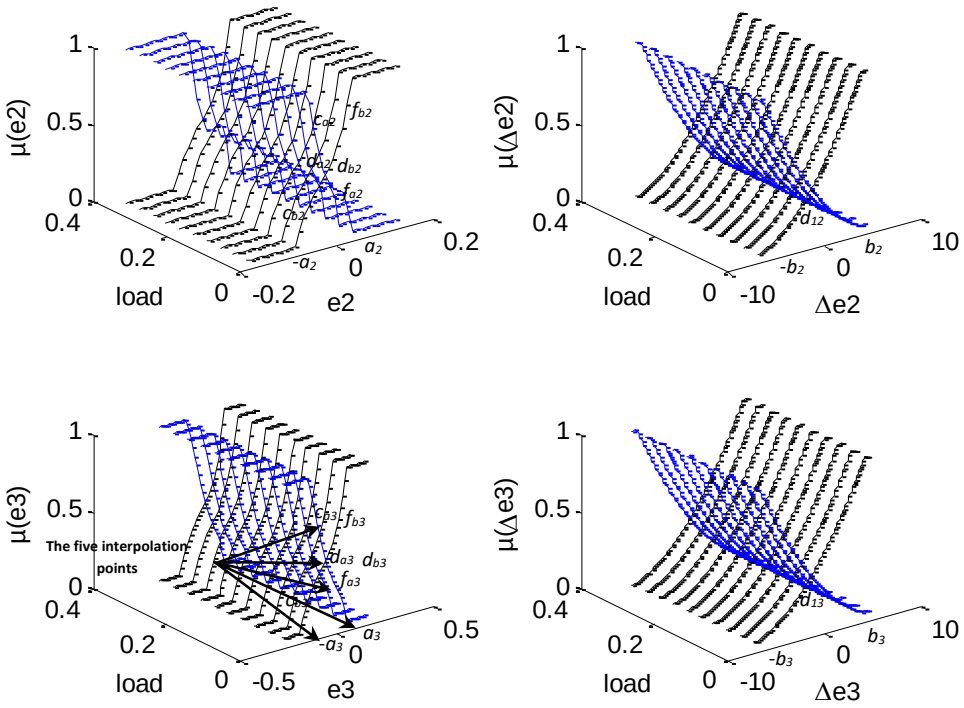


Figure 4.18 The membership functions for the two regulators (e : error, Δ*e* : errorvariation) after five interpolation points, where the three points inside the curve optimized by *GAs*.

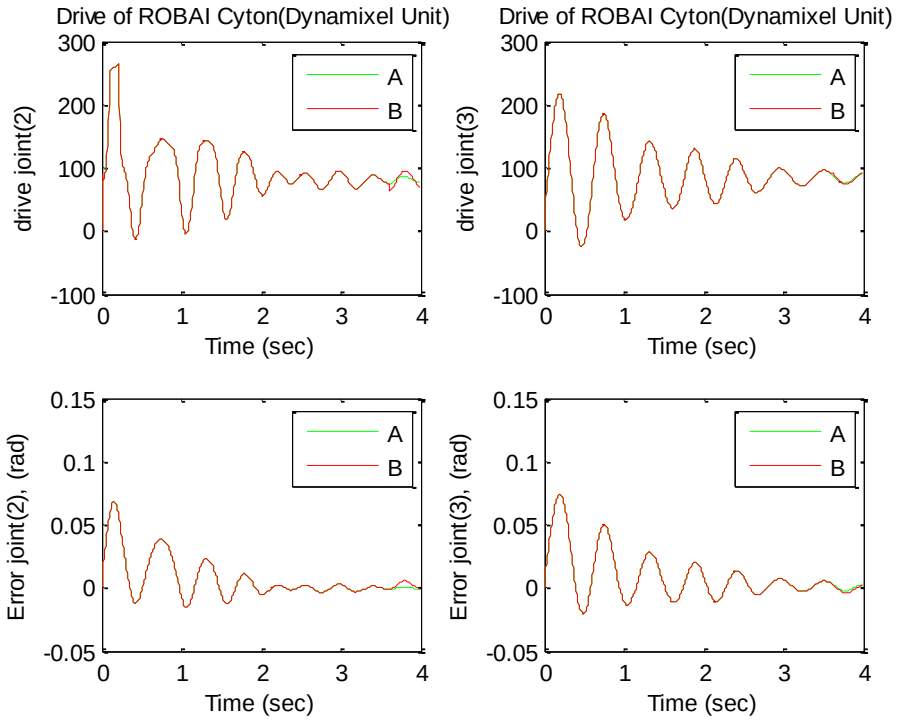


Figure 4.19 Response of the system.(Neuro-fuzzy-genetic regulator of (2×2) Sugenotype), after five interpolation points with fall of load from 0.36 *Kg* to 0 *Kg* at 3.6 sec time. A. one keeps the same membership functions of 0.36 *Kg*. B. one changes the membership functions from 0.36 *Kg* to 0 *kg* at 3.6 sec time.

4.2.8. Practical results

To show the effectiveness of the above mentioned controller, it was tested for the control of two freedom degrees of ROBAI Cyton manipulators actuated by direct current servo motors dynamixel MX-64, where the value of the torque is transformed to a value varying between zero and a maximum (30 *Nm* because of the external gear ratio) corresponding to 1023 unit.

These experimental results, shown in Figure 4.20, Figure 4.21, and Figure 4.22, are compared with the simulated results, depicted in Figure 4.15, Figure 4.17, and Figure 4.19, using identical trajectories and the same optimized gains for both controllers. Very similar results are obtained in both cases with perfectly smooth driving torques.

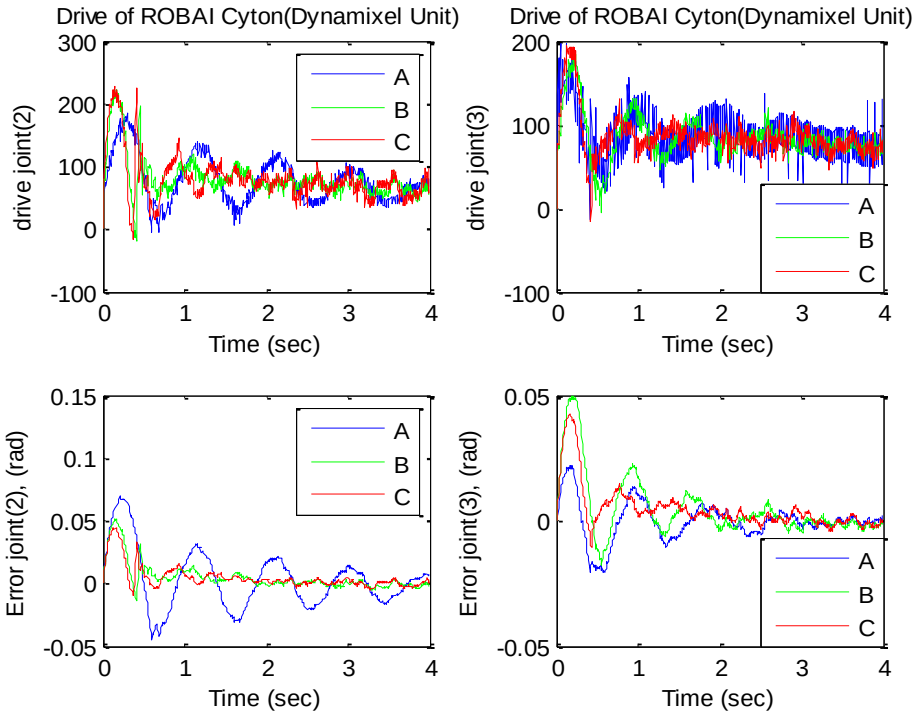


Figure 4.20 Same practical results as in Figure 4.15.

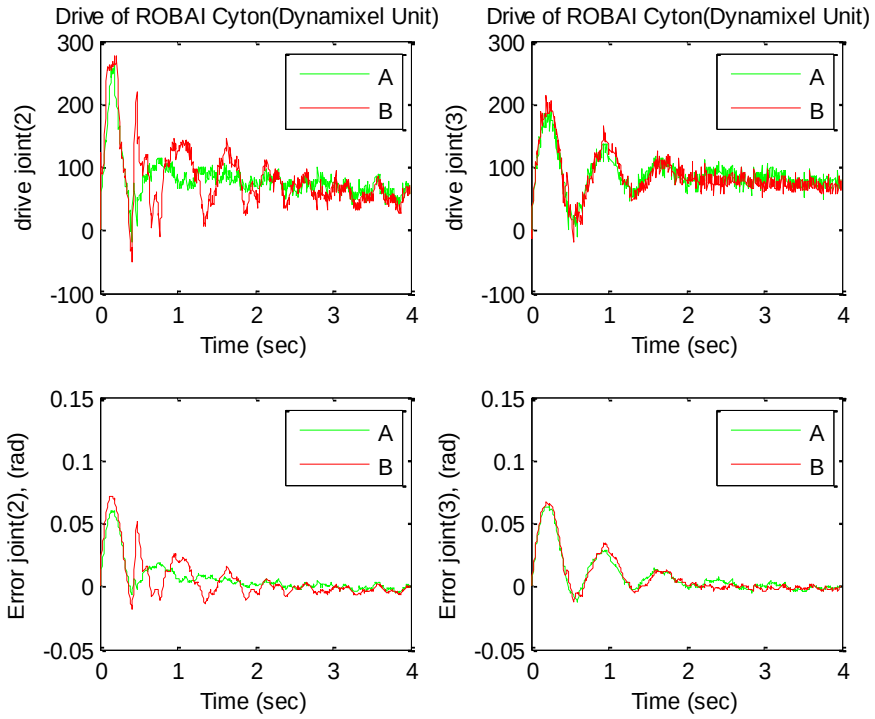


Figure 4.21 Same practical results as in Figure 4.17.

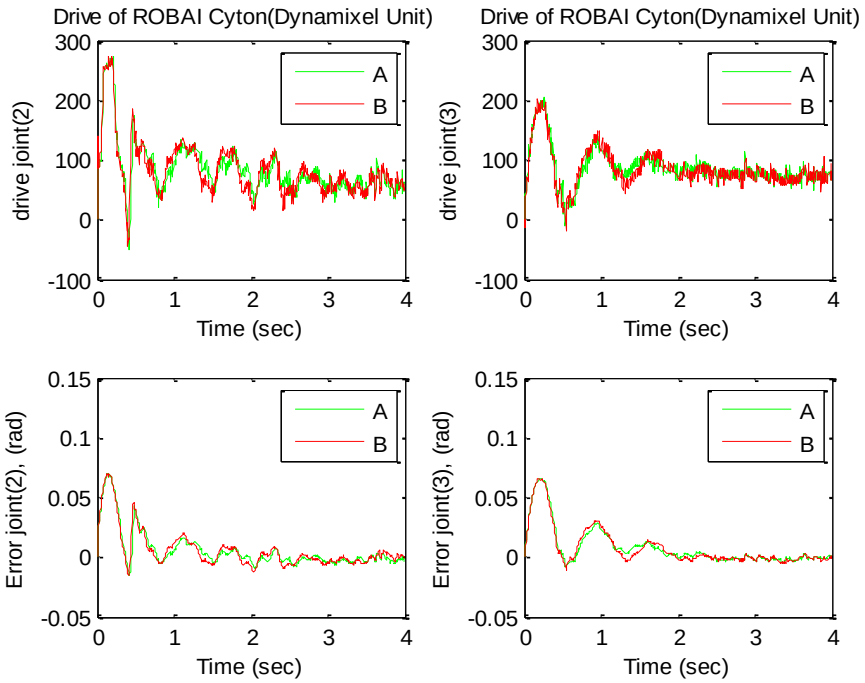


Figure 4.22 Same practical results as in Figure 4.19.

4.3. Second application

(Control of the mobile robot Pioneer P3-DX)

4.3.1. Kinematics modeling

Consider the unicycle-type mobile robot shown in Figure 4.23. This robot consists of three degrees of freedom, i.e., *x*, *y* and θ. For simplicity, the robot is represented by its projection on a plane parallel to the ground. The unicycle is equipped with two independently operated wheels [64] [65]. To model this system oneconsiders the point located in the middle of the wheels axle. A frame $R' = (O', \overrightarrow{x'}, \overrightarrow{y'}, \overrightarrow{z'})$ is associated with this point, as shown in Figure 4.23. An inertial frame $R = (O, \vec{x}, \vec{y}, \vec{z})$ is fixed to the ground, whose axis is vertical. A configuration vector of the unicycle is given by: $q = (x, y, \theta) \in Q = \mathbb{R}^2 \times \mathbb{S}^1$.

Where (x, y)represent the coordinates of the point situated in the middle of the axis of the rear wheels, θ the angle (the orientation angle of the robot), $\varphi_r$the angle of rotation of the right wheeland $\varphi_l$the angle of rotation of the left wheel.

Let θ be the radius of the wheels and the L distance between the point O' and the midpoint of each wheel.

The robot in the absolute coordinate system *OXY* is specified by the vector *q*:

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{4.17}$$

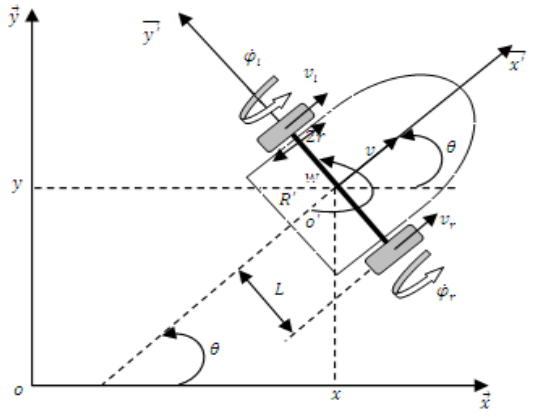


Figure 4.23 Kinematic representation of a mobile robot unicycle type.

The contact between the wheels of the mobile robot and the ground, satisfies the non-holonomic conditions of pure rolling and non-slipping device [65] [66].

$$\dot{x}sin\theta - \dot{y}cos\theta = [sin\theta \quad -cos\theta \quad 0]\begin{bmatrix}\dot{x}\\ \dot{y}\\ 0\end{bmatrix} \tag{4.18}$$

To characterize the configuration of the unicycle robot. An associated kinematic model is given by:

$$\dot{q} = J\zeta \Leftrightarrow \begin{cases}\dot{x} = vcos\theta\\ \dot{y} = vsin\theta\\ \dot{\theta} = \omega\end{cases}, \dot{q} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix}, \zeta = \begin{bmatrix}v_1\\ v_2\end{bmatrix} = \begin{bmatrix}v\\ \omega\end{bmatrix}, J = \begin{bmatrix}cos\theta & 0\\ sin\theta & 0\\ 0 & 1\end{bmatrix} \tag{4.19}$$

Where $v_1 = v = \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l)$ the longitudinal or advancing speed of the robot and

$v_2 = \dot{\theta} = \omega = \frac{r}{2L}(\dot{\varphi}_r - \dot{\varphi}_l)$ the rotation speed about the axis perpendicular to the rolling plane.

In the proposed case, we used the MobileSim to simulate the robot Pionner 3DX, which works with MATLAB.

4.3.2. The trajectory tracking control

Trajectory tracking control of differential speed drives is giving for feedforward and feedback control law as suggested in Kanayama et al. [67], where in the proposed study, we introduced a control law as an adaptation of the control strategy presented by A. De Luca and al. [68].

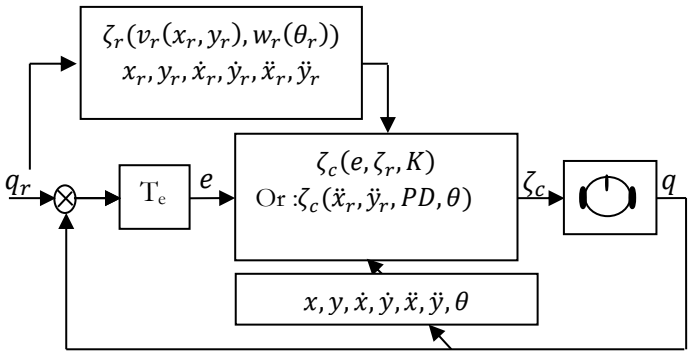


Figure 4.24 The trajectory tracking control of the mobile robot.

Wetracked the eight-shaped trajectory represented by the following equations:

$$x_r(t) = sin\frac{t}{10}, y_r(t) = cos\frac{t}{10}, \quad t \in [0 \quad T] \tag{4.20}$$

With $$q_r = [x_r \quad y_r \quad \theta_r]^T , \zeta_c = [v_r \quad w_r]^T$$

A simple design of a trajectory tracking control is based on tangent linearization along the reference trajectory, where the error posture $T_e(q_r - q)$, which represent the difference between the current posture and the reference posture is giving by:

$$e = T_e(q_r - q), \qquad \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{4.21}$$

The auxiliary linear velocity and the angular velocity terms ζ_c ensure the control of the mobile robot.

4.3.3. Linear Control Design

Using the nonlinear transformation of velocity inputs [67], [69]

$$\zeta_c = \begin{bmatrix} v_r cose_3 + k_1 e_1 \\ w_r + k_2 sign\big(v_t(t)\big)e_2 + k_3 e_3 \end{bmatrix} \tag{4.22}$$

Where k_1, k_2, k_3 are positive constants parameters

$$k_1 = k_3 = 2\varsigma\sqrt{w_r^2(t) + g v_r^2(t)}, \qquad k_2 = g|v_r(t)|, \qquad g > 0$$

In terms of the original control inputs, this design leads to the *non linear time-varying* controller

$$\begin{cases} v = v_r\, cos(\theta_r - \theta) + K_1[cos\theta(x_r - x) + sin\theta(y_r - y)] \\ w = w_r + K_2 sign(v_r)[cos\theta(y_r - y) - sin\theta(x_r - x)] + K_3(\theta_r - \theta) \end{cases} \tag{4.23}$$

4.3.4. Nonlinear Control Design

A nonlinear design for trajectory tracking is done

$$\zeta_c = \begin{bmatrix} v_r cose_3 + k_1(v_r(t), w_r(t))e_1 \\ w_r + \bar{k}_2 v_r(t)\frac{sine_3}{e_3}e_2 + k_3(v_r(t), w_r(t))e_3 \end{bmatrix} \tag{4.24}$$

$$k_1\big(v_r(t), w_r(t)\big) = k_3\big(v_r(t), w_r(t)\big) = 2\varsigma\sqrt{w_r^2(t) + g v_r^2(t)}, \qquad \overline{k}_2 = g,\ \ g > 0,\ \ \varsigma \in (0,1)$$

4.3.5. Dynamic Feedback Linearization

Define the linearizing output vector as $\eta = (x, y)$, so the differentiation gives [39]

$$\dot{\eta} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ 0 & sin\theta \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{4.25}$$

Showing that only *v* affects $\dot{\eta}$, whereas the angular velocity *w* can not be recovered from this first-order differential information. To do this, an integrator (whose state is designated by ξ) must be added to the linear velocity input

$$v = \xi, \quad \dot{\xi} = a \Longrightarrow \ \dot{\eta} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} \tag{4.26}$$

Oneassume another input *a* which represents the linear acceleration, by the differentiation of the linear velocity.

$$\ddot{\eta} = \dot{\xi}\begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} + \xi\dot{\theta}\begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\xi\sin\theta \\ \sin\theta & \xi\cos\theta \end{bmatrix}\begin{bmatrix} a \\ w \end{bmatrix} \tag{4.27}$$

In order that the matrix which multiplies (a, w) is not singular, it is necessary that $\xi \neq 0$.

According to this hypothesis, one can define.

$$\begin{bmatrix} a \\ w \end{bmatrix} = \begin{bmatrix} \cos\theta & -\xi\sin\theta \\ \sin\theta & \xi\cos\theta \end{bmatrix}^{-1}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.28}$$

In order to obtain:

$$\ddot{\eta} = \begin{bmatrix} \ddot{\eta}_1 \\ \ddot{\eta}_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = u$$

The resulting dynamic compensator is

$$\begin{array}{c} \dot{\xi} = u_1\cos\theta + u_2\sin\theta \\ v = \xi \\ w = \frac{u_2\cos\theta - u_1\sin\theta}{\xi} \end{array} \Longrightarrow \zeta_c = \begin{bmatrix} \xi \\ \frac{u_2\cos\theta - u_1\sin\theta}{\xi} \end{bmatrix} \tag{4.29}$$

Where

$$\begin{aligned} u_1 &= \ddot{x}_r(t) + k_{p1}(x_r(t) - x) + k_{d1}(\dot{x}_r(t) - \dot{x}) \\ u_2 &= \ddot{y}_r(t) + k_{p2}(y_r(t) - y) + k_{d2}(\dot{y}_r(t) - \dot{y}) \end{aligned} \tag{4.30}$$

k_{pi}, k_{di} are positive constants parameters.

4.3.6. Dynamic Feedback Linearization with Neuro-fuzzy-Genetic controller

In this section, we used the same controller presented in equation (4.29), but replaced the PD compensator of equation (4.30) by a neuro-fuzzy-genetic presented in section (4.2.3).

4.3.7. Simulation results

We used the MobileSim simulator which works with MATLAB to simulate the Pioneer P3-DX robot, the inputs of the robot are the linear and the angular velocity(v, w), and the outputs are the position (x, y) giving by the odometric which requires the filtering of the signal and the orientation (θ), we choosed the same trajectory eight-shaped reference proposed by [39], see Figure 4.25 for tests, where the trajectory starts from the origin with $\theta_r(0) = \pi/6\, rad$ (the starting configuration robot is coordinated with the reference trajectory), Figure 4.26give the desired trajectory tracking, after the tests of some control techniques, we noticed that the LCD and NCD controls give almost the same results for the same $\varsigma\, and\, g$ (equations (4.22) and (4.24)) which are presented in Figure 4.27.

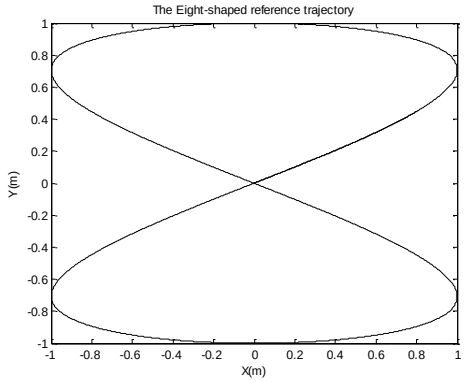


Figure 4.25 The Eight-shaped reference trajectory.

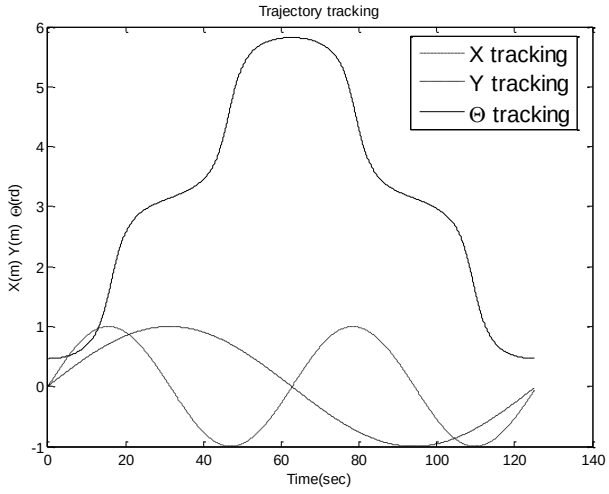


Figure 4.26 Trajectory tracking.

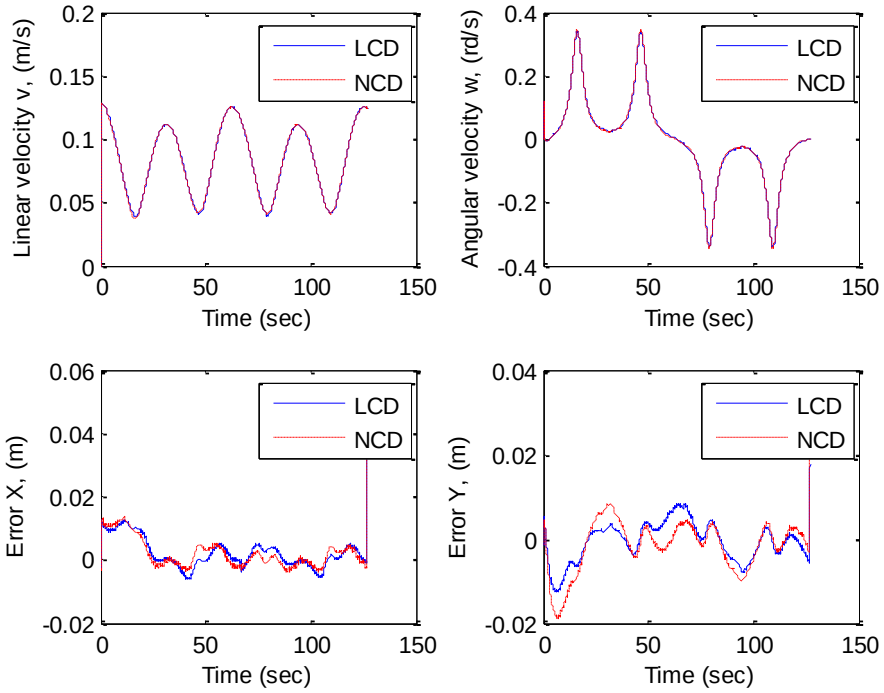


Figure 4.27 Trajectory tracking, LCD.Linear Control Design, NCD.Nonlinear Control Design.

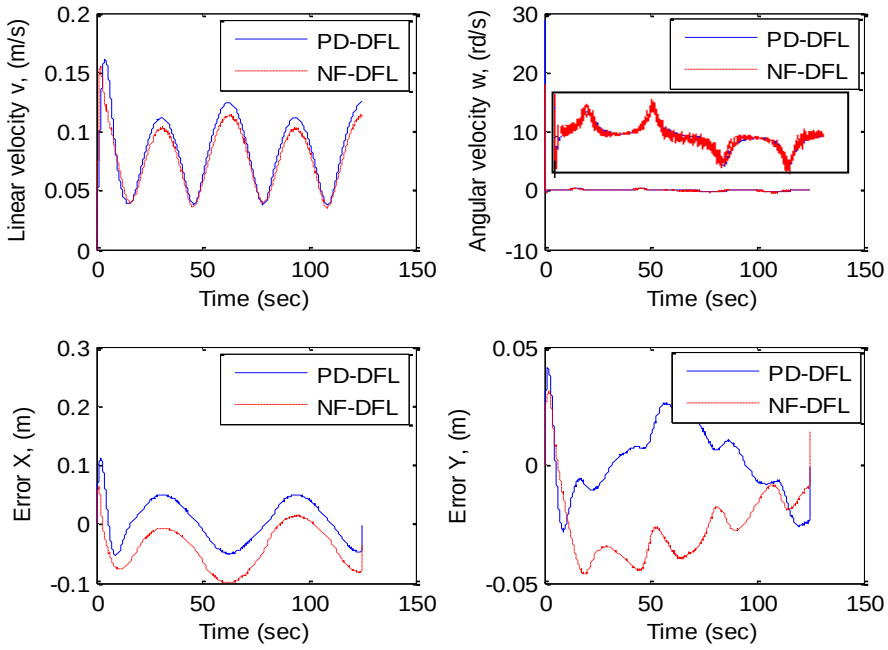


Figure 4.28 Trajectory tracking, PD-DFL.Dynamic feedback linearization, NF-DFL.Neuro-fuzzy Dynamic feedback linearization before optimization.

To improve the DFL control, the idea is to replace the classical PD compensator presented in the equation (4.30) by a Neuro-fuzzy controller, when the results of the two control laws are compared, see Figure 4.28 before optimization, and Figure 4.29 after optimization, we found that the two controllers have almost the same performance.

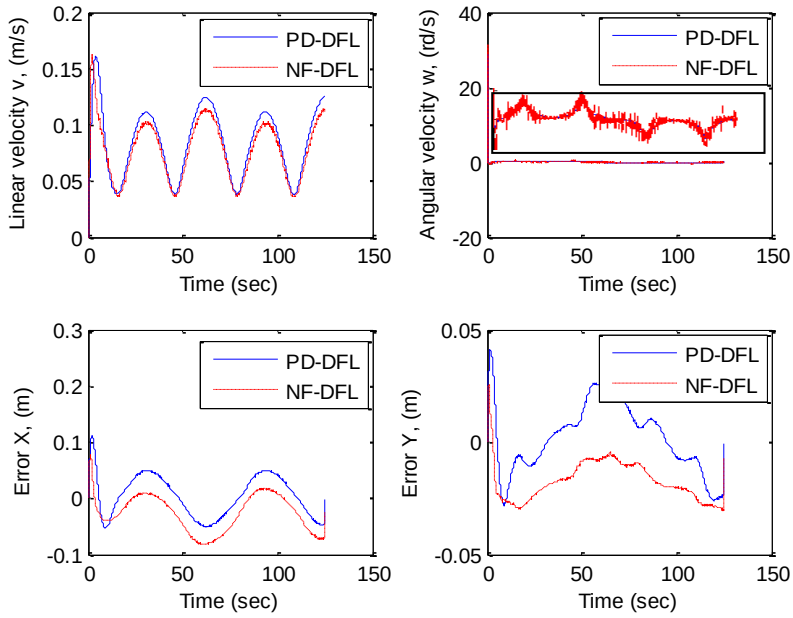


Figure 4.29 Trajectory tracking, PD-DFL.Dynamic feedback linearization, NF-DFL.Neuro-fuzzy Dynamic feedback linearization after optimization.

Another set of experiments was performed, with an initial position of the eight-shaped reference trajectory $q_r = (0.2,\ -0.3)\ (m, m)$, and $\big(x(0), y(0), \theta(0)\big) = (0,0,0)$ Figure 4.30, We have noticed that the tracking is better for the DFL conventional controller, where the tracking error is minimal and the tracking shape is better. The Figure 4.31shows the Cartesien motion $(x,\ y)\ \ (m)$.

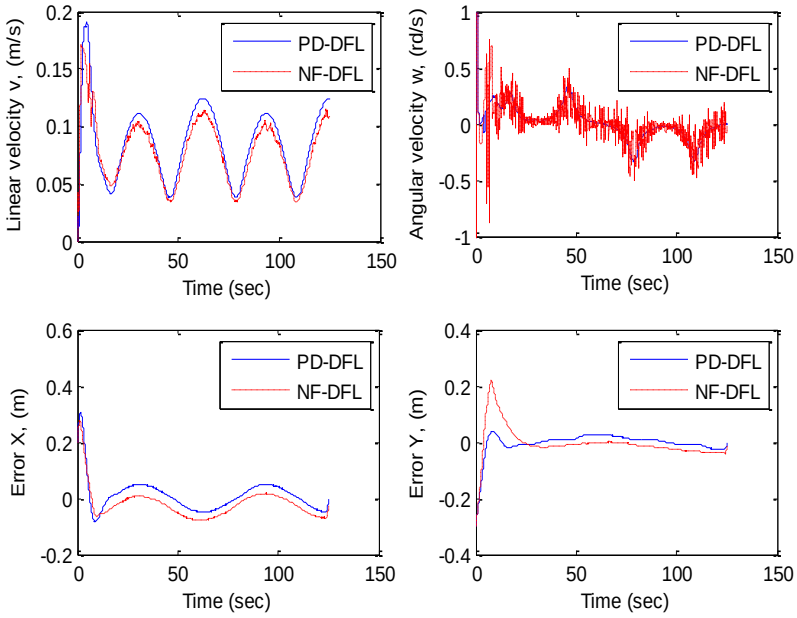


Figure 4.30 Trajectory tracking with initial error, PD-DFL.Dynamic feedback linearization, NF-DFL.Neuro-fuzzy Dynamic feedback linearization after optimization.

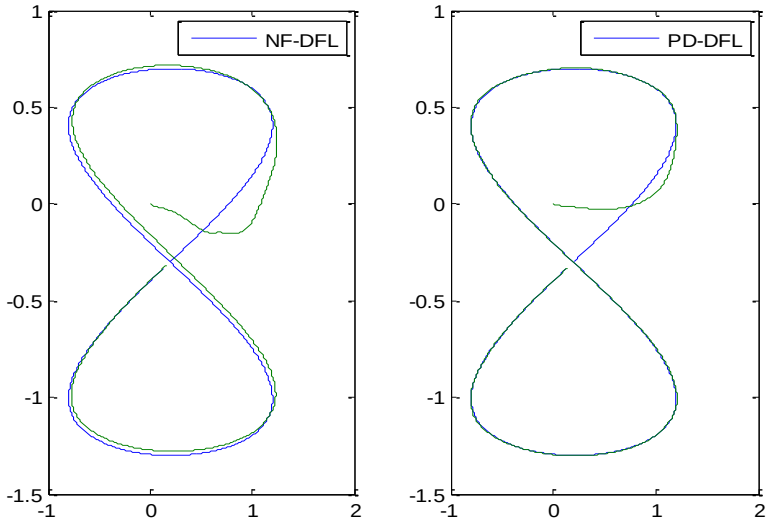


Figure 4.31 Cartesien motion (x, y) (m) with an initial error.

4.3.8. Practical results

The Differential mobile robot Pioneer P3-DX

The Pioneer P3-DX of figure 4.32 is a mobile robot designed for research.It is built by Adept Mobile Robots.Because of its reliability and durability, the Pioneer is a reference platform for research in robotics, fully programmable and equipped with sensors necessary for good navigation. It comes fully assembled with a built-in controller, motors with 500-head encoders, 19 cm wheels, a strong aluminium body, 8 ultrasonic sensors facing forward (sonar), 8 optional sensors, 1, 2 or 3 hot-swappable batteries and a complete software development kit, and an optional internal computer or laptop. Additional sensors, manipulators and accessories are available to customize the operation of Pioneer for specific and complex tasks.

For the development of applications with the "Pioneer P3-DX" mobile robot, five toolsare available:

1- A library of C ++ codes, "with a simple extension in Matlab" well supplied, under Linux and Windows, with ARIA.

2- MobileEyes software to control and visualize the position of your robot remotely.

3- MobileSim tool to simulate all possible configurations.

4- Mapper 3 Perform all your robot mapping operations.

5-SONARNL to ensure the location and the control of your robot through the information obtained by the sonar integrated into your robot.

Figure 4.32 The mobile robot PioneerP3-DX

To show the effectiveness of the above controllers, the developed algorithmwas applied to drive areal mobile robot Pioneer P3-DX. These experimental results are compared to simulation results, using identical trajectory tracking, and controller gains.

The comparison of the practical results with those of the simulations, reveals very small differences.

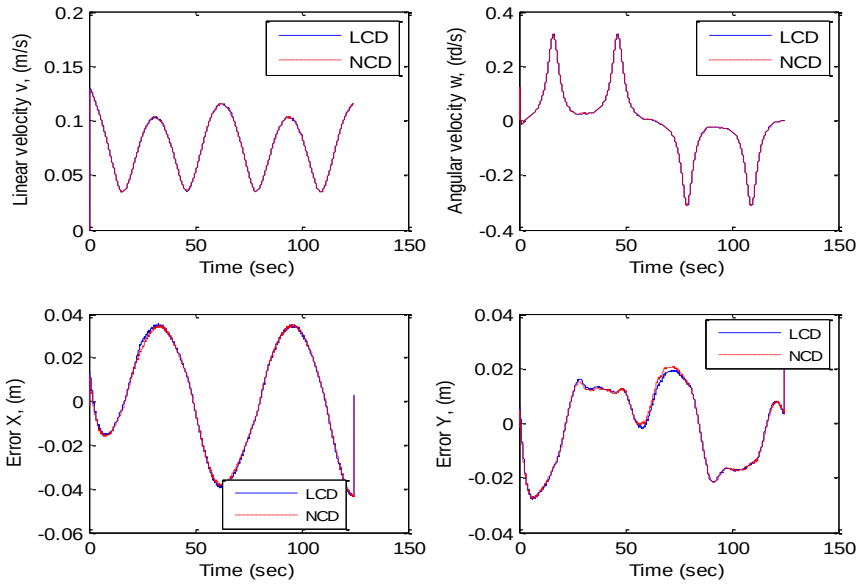


Figure 4.33 Same practical results asFigure 4.27.

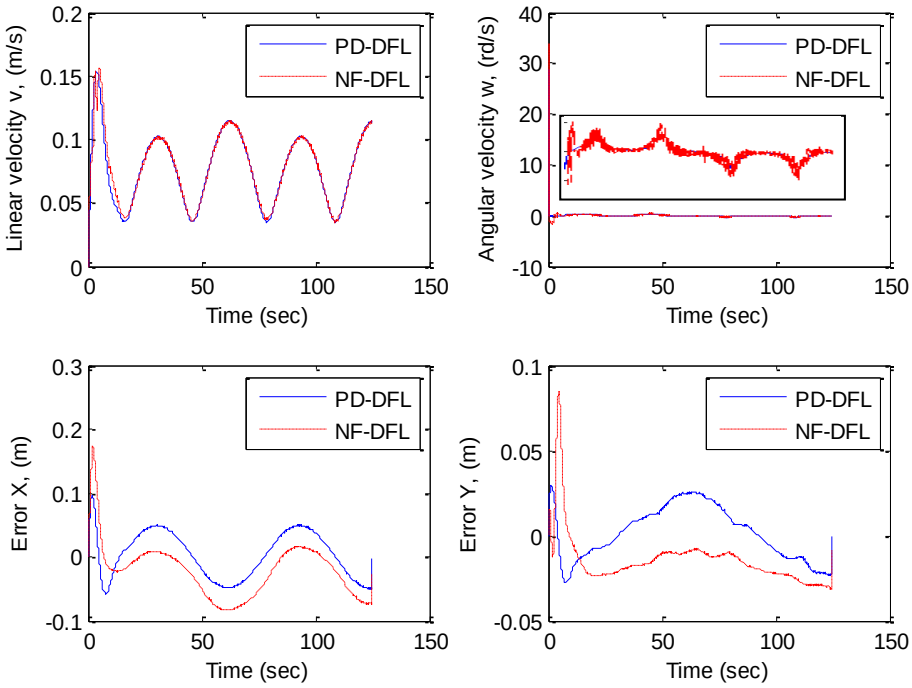


Figure 4.34 Similar practical results as in fFigure 4.29.

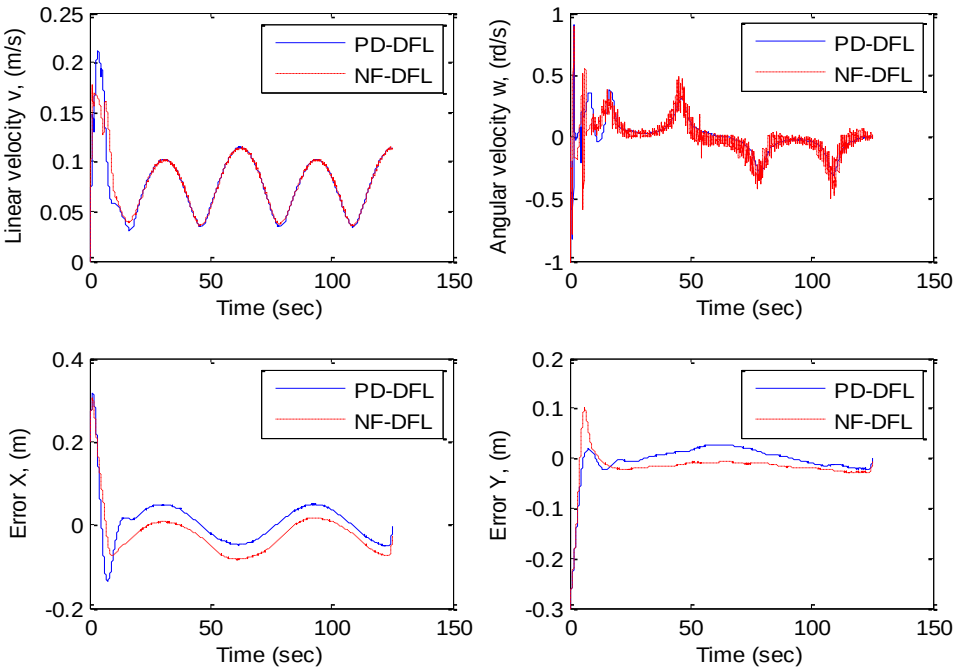


Figure 4.35 Similar practical results as in fFigure 4.30**.**

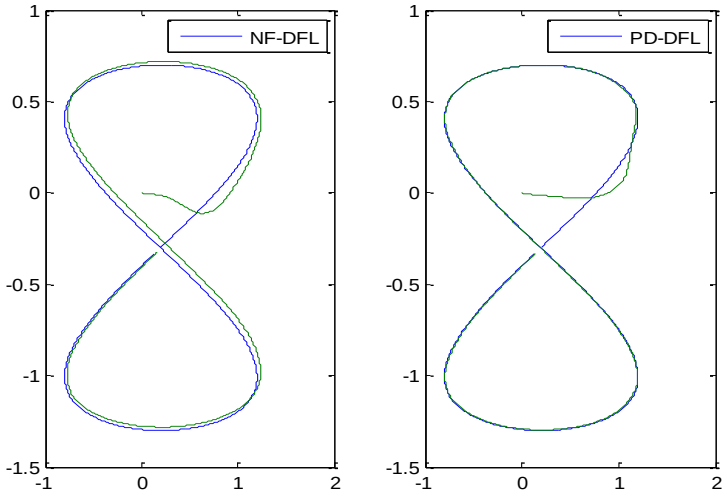


Figure 4.36 Similar practical results as in fFigure 4.31.

4.4. Conclusion

This chapter focuses on the simulation and practical results of fuzzy and neuro-fuzz control with decentralized structure, optimized by *GAs*, applied to a manipulator arm and a mobile robot.

The approach followed is first to synthesize the Mamdani regulator type with several membership functions. However, given the fixed structure of this regulator, the tracking performance is degraded in the presence of parametric variations.

To overcome this problem, we considered the structural projection of the fuzzy inference system in a multilayer network as well as the Sugeno-type regulator. In order to adjust the widths of the universes of discourse and the weights of the connections, by alearning algorithm based on the extended Kalman filter which consists in linearizing at any moment the joint control around the estimated vector. *GAs* are used to optimize the location and the shape of fuzzy subsets. The simplicity relative to the structure of the controlers as well as the results obtained would allow one to envisage an adjustment of the parameters on very complex systems.

GENERAL CONCLUSION AND PERSPECTIVES

The work presented in this thesis deals with the controls in the articular space of rigid manipulator robots, and the control of mobile robots.

The proposed approach is the modeling of the manipulator arm by exploiting the Euler - Lagrange formalism. Once the dynamic model of the robot is obtained, one was interested in the planning of an adequate trajectory which ensures continuity in position, speed and acceleration to realize the tracking of the position trajectory, through a feasible drive.

For the mobile robot from a control viewpoint, the peculiar nature of nonholonomic kinematics makes that feedback stabilization at a given position cannot be achieved via smooth time-invariant control. This indicates that the problem is really nonlinear and linear control is ineffective, even locally, and innovative design techniques are needed.

Three control structures have been proposed namely fuzzy control, neuro-fuzzy control, and finally neuro-fuzzy control optimized by genetic algorithms.

The neuro-fuzzy control optimized by *GAs* for a robot. With this aim in mind, we proceeded to model the robot. Weused a regulator, which is a combination of fuzzy logic, and neural networks in the form of a multi-layer network, for the automatic extraction of the rules, where the human expertise is generally needed.The main advantages of the neuro-fuzzycontroller were the learning capacity, the processing speed, and the adaptability.The neuro-fuzzy controller does not require knowledge of the parameter values of the robot and it does not need neither dynamic models of the system nor control experts for the robot control problem. But in robotics, it is important to optimize the work plan and the robot motions by using an evaluation function. In the case of complex tasks, there are many factors such as energy, time and distance, to evaluate the motions of robot.

To avoid the problem of fuzzy subsets positions and their shapes, we used trapezoidal fuzzy subsetswhere each straight lines is supposed to be an interpolation functions of three points, and then of five points. By optimizing these functions using the genetic algorithms (*GAs*), and minimizing a performance criterion based on the tracking error, we found the most suitable position and form of the fuzzy subsets which ensure a good tracking, whatever the assigned task was. The obtained results show the effectiveness of the suggested approach.

The obtained results show that the combination of both algorithms, namely online training Kalman filter for neural networks to extract fuzzy rules and *GAs* for the fuzzy subsets, improves significantly the fuzzy controller performances.

Finally,we can say that the Neuro-fuzzy approach optimized by the genetic algorithms proposed decentralized structure is a very useful tool for the design of controllers, even if the number of elementary regulators is higher.

As perspectives for this work, it would be interesting to apply this control technique to a robot arms that requires high precision such as surgical medical robots, or robots that make integrated circuits, and to study the effects of other parameters on the robot dynamics for the choice and the variation of the membership functions.

Bibliographic References

[1] P. Hušek , O. Cerman .Fuzzy model reference control with adaptation of input fuzzy sets. elsevier. Knowledge-Based Systems 49 (2013) 116–122.

[2] H. P. Huy Anh, K. K. Ahn .Hybrid control of a pneumatic artificial muscle (PAM) robot arm using an inverse NARX fuzzy model. elsevier. Engineering Applications of Artificial Intelligence 24 (2011) 697–716.

[3] L. Hu, H.D. Cheng, M. Zhang. A high performance edge detector based on fuzzy inference rules. ScienceDirect, Information Sciences 177 (2007) 4768–4784.

[4] A Soukou. conception d'un contrôleur Neural-flou par les algorithmes génétique. Magister,Ferhat Abbas Setif University, electronic departement 1998/1999.

[5] Y. Tansel Iç, M. Yurdakul, and B. Dengiz .Development of a decision support system for robot selection. elsevier. Robotics and Computer-Integrated Manufacturing 29 (2013) 142–157.

[6] Y. Zuo, Y. Wang , X. Liu, S.X. Yang, L. Huang, X. Wu, Z. Wang .Neural network robust H_∞ tracking control strategy for robot manipulators. elsevier. Applied Mathematical Modelling 34 (2010) 1823–1838.

[7] D. Zhao, W. Ni, Q. Zhu .A framework of neural network s based consensus control for multiple robotic manipulators. elsevier. Neurocomputing 140 (2014) 8–18.

[8] T. D. Le, H-J. Kang .An adaptive tracking controller for parallel robotic manipulators based on fully tuned radial basic function networks. elsevier. Neurocomputing 137 (2014) 12–23.

[9] T. Lianfang , C. Curtis .Adaptive Neuro-fuzzycontrol of a flexible manipulator. Elsevier Mechatronics., (2005) 1305-1320.

[10] J. Botzheim, P. Földesi .Novel calculation of fuzzy exponent in the sigmoid functions for fuzzy neural networks. elsevier. Neurocomputing 129 (2014) 458–466.

[11] P-V.Vicente, G-R.Rodolfo, A.Jorge .Neurofuzzy self-tuning of the dissipation rate gain for model-free force-position exponential tracking of robots., (2016) ELSEVIER Neurocomputing., 171 (2016) 209–219.

[12] F-G.Juan, O-R. Cynthia, A-G.José .Rule generation of fuzzy logic systems using a self-organized fuzzy neural network., (2015) ELSEVIER Neurocomputing., 151 (2015) 955–962.

[13] G-R. Rodolfo, P-V Vicente .Normal and tangent force neuro-fuzzy control of a soft-tip robot with unknown kinematics. ScienceDirect, Engineering Applications of Artificial Intelligence. 65 (2017) 43-50.

[14] M. Vijay, D. Jena.PSO based neuro fuzzy sliding mode control for a robot manipulator. ScienceDirect, Journal of Electrical Systems and Information Technology xxx (2016) xxx–xxx.

[15] J. K. Pothala, D. R. Parhia .Navigation of Multiple Mobile Robots in a highly clutter terrains using Adaptive Neuro-Fuzzy Inference system. ScienceDirect, Robotics and Autonomous Systems 72, 2015, 48-58.

[16] R. Al-Jarrah, A. Shahzad, H. Roth .Path Planning and Motion Coordination for Multi-Robots System Using Probabilistic Neuro-Fuzzy. ScienceDirect, IFAC-PapersOnLine 48-10 (2015) 046–051.

[17] J. Armendariz, V. Parra-Vega, R. García-Rodríguez, S. Rosales .Neuro-fuzzy self-tuning of PID control for semiglobal exponential tracking of robot arms. ScienceDirect, Applied Soft Computing 25 (2014) 139–148.

[18] I. Baturonea,1, A. Gersnoviez b, Á. l Barriga .Neuro-fuzzy techniques to optimize an FPGA embedded controller for robot navigation. ScienceDirect, Applied Soft Computing 21 (2014) 95–106.

[19] H. Erdem .Application of Neuro-Fuzzy Controller for Sumo Robot control. ScienceDirect, Expert Systems with Applications 38 (2011) 9752–9760.

[20] K. Demirli, M. Khoshnejad .Autonomous parallel parking of a car-like mobile robot by a neuro-fuzzy sensor-based controller. ScienceDirect, Fuzzy Sets and Systems 160 (2009) 2876 – 2891.

[21] C-H Lee, M-H Chiu .Recurrent neuro fuzzy control design for tracking of mobile robots via hybrid algorithm. ScienceDirect, Expert Systems with Applications 36 (2009) 8993–8999.

[22] J. C. Figueroa-García, C. M. Ochoa-Rey, J. A. Avellaneda-González .Rule generation of fuzzy logic systems using a self-organized fuzzy neural network. elsevier. Neurocomputing 151 (2015) 955–962.

[23] C.H. Kim, J.H Seok, B.S. Choi, J.J. Lee .Multiple Incremental Fuzzy Neuro-Adaptive Control of Robot Manipulators. The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. St. Louis, USA, (2009).

[24] S.R. Pandian, N. Sakagami . A Neuro-Fuzzy Controller for Underwater Robot Manipulators. 2010 ,11th International Conference on Control Automation Robotics & Vision, 7-10 Dec. 2010, Publisher : IEEE

[25] B. Xu, S. R. Pandian, N. Sakagami, F. Petry .Neuro-fuzzy control of underwater vehicle-manipulator systems. Journal of the Franklin Institute 349 (2012) 1125 – 1138.

[26] S.Alavandar, M.J.Nigam .New hybrid adaptive neuro-fuzzy algorithms for manipulator control with uncertainties – Comparative study. Elsevier. ISA Transactions 48 (2009) 497–502.

[27] S.Ganjefar, M.Tofighi .Single-hidden-layer fuzzy recurrent wavelet neural network: Applications to function approximation and system identification., (2015)Elsevier Information Sciences., 294(2015) 269–285.

[28] Y.Oniz, O.Kaynak, Control of a direct drive robot using fuzzy spiking neural networks with variable structure systems-based learning algorithm., (2015) Elsevier Neurocomputing., 149 (2015) 690–699.

[29] A. Chatterjee, P. Siarry ,A PSO-aided neuro-fuzzy classifier employing linguistic hedge concepts. elsevier. Expert Systems with Applications 33 (2007) 1097–1109.

[30] Jacques Gangloff. Asservissement visuels rapide d'un robot manipulateur à 6 degrés de liberté. Thèse de doctorat, université deStrasbourg.france, 1999.

[31] E. Freund.The structure of decoupled nonlinear systems. International Journal of Control 21(3):443-450 · March 1975

[32] B. Bayle, .Robotique mobile : note de cours. Ecole Nationale Supérieure de Physique de Strasbourg, 2009.

[33] B. Bayle, .Robotique Mobile. thèse de doctorat, Ecole Nationale Supérieure de Physique de Strasbourg, Université de Strasbourg 2008.

[34] M. Defoort. contributions a la planification et a la commande pour les robots mobiles cooperatifs. Thèse de Doctorat, l'Ecole Centrale de Lille 2007.

[35] H. J. Marquez **.**Nonlinear Control Systems Analysis and Design. Book. WILEY INTERSCIENCE A JOHN WILEY & SONS, INC., PUBLICATION Copyright © 2003

[36] M.A Henson, D.E.Seborg .Nonlinear process control. Book.Prentice-Hall, Inc. Upper Saddle River, NJ, USA ©1997

[37] J.J.E.Slotine,W.LiAppliednonlinear control. Book. by Prentice-Hall, Inc. A Division of Simon & Schuster Englewood Cliffs, New Jersey 07632, 1991.

[38] A. De Luca, G. Oriolo, and M. Vendittelli .Control of Wheeled Mobile Robots: AnExperimental Overview.Lecture Notes in Control and Information Sciences book series (LNCIS, volume 270), pp. 181−226 Springer-Verlag Berlin Heidelberg, 2001.

[39] G. Oriolo, A. De Luca, M. Vendittelli.WMR Control Via Dynamic Feedback Linearization. Design, Implementation, and Experimental Validation. IEEE transactions on control systems technology, vol. 10, no. 6, November 2002.

[40] R Köker .A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. Elsevier. Information Sciences 222 (2013) 528–543.

[41] R. Martínez, O. Castillo, L. T. Aguilar .Optimization of interval type-2 fuzzy logic controllers for a perturbed autonomous wheeled mobile robot using genetic algorithms. Elsevier. Information Sciences 179 (2009) 2158–2174.

[42] D. Gonzalez. Les algorithmes génétiques. Edition ellipse, Février 2004.

[43] L. J. Eshelman, D. J. Schaffer.Real-coded Genetic Algorithms and Interval-Schemata. In Whitley, D. L. (Ed.), Foundations of Genetic Algorithms 2 (pp. 187-202). California: Morgan Kaufmann, 1993.

[44] A. H. Wright .Genetic Algorithms for Real Parameter Qptimization.Proceedings of the First Workshop on Foundations of Genetic Algorithms. Bloomington Campus, Indiana, USA, July 15-18 1990.

[45] P. Garnier .Contrôle d'exécution réactif de mouvements de véhicules en environnement dynamique structuré . Doctorat à l'institut national polytechnique de grenoble, Spécialité : informatique, 1995.

[46] H.Bûhler. Réglage par logique floue . Presses polytechniques et universités Romandes,1994.

[47] S. Warren, S.McCulloch. W.H.Pitts .A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics, Vol. 5, 1943,p.115-133.

[48] E.Gauthier .Utilisation des Réseaux de Neurones Articiels pour la Commande d'un Véhicule Autonome. Doctorat à l'institut national polytechnique de grenoble, Spécialité : informatique, et systèmes et communications, 1999.

[49] J. S. Albus .A New Approach to Manipulator Control: the Cerebellar Model Articulation Controller (CMAC). Trans. ASME, Series G. Journal of Dynamic Systems, Measurement and Control, 97, pp. 220-233, 1975.

[50] R. E. KALMAN.A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME–Journal of Basic Engineering, 82 (Series D): 35-45. Copyright © 1960 by ASME.

[51] N.Kharmandar, A.A. Khayyat, Force Impedance Control of a Robot Manipulator Using a Neuro-Fuzzy Controller. 2011 International Conference on Mechatronic Science, Electric Engineering and Computer., Jilin, China, (2011) 978-1-61284-722-1/11/$26.00 ©2011 IEEE.

[52] M. Porticia, C. Oscar.intelligent system of a stepping motor drive using an adaptive neuro-fuzzy inference system. Elsevier information sciences., (2005) pp.133-151,170.

[53] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man, and Cybernetics . Vol.23 , Issue: 3 , May/June 1993.

[54] P.Y.Glorennec .Neuro-Fuzzy control using reinforcement learning. Proceedings of IEEE Systems Man and Cybernetics Conference - SMC. Le Touquet, France,17-20 Oct. 1993.

[55] H. R. Berenji and P. Khedar, Learning and Tuning Fuzzy Logic Controllers Through Reinforcements. IEEE Trans.On neurona networks, Vol. 3, No. 5, september 1992.

[56] A. Otsuka, F Nagata, Application of Genetic Algorithms to Fine-Gain Tuning of Improved the Resolved Acceleration Controller. ScienceDirect, Procedia Computer Science 22 (2013) 50 – 59.

[57] M. Eftekhari, S.D. Katebi, Extracting compact fuzzy rules for nonlinear system modeling using subtractive clustering , GA and unscented filter. elsevier. Applied Mathematical Modelling 32 (2008) 2634–2651.

[58] M. A. Shoorehdeli , M. Teshnehlab, and A. K. Sedigh .Training ANFIS as an identifier with intelligent hybrid stable learning algorithm based on particle swarm optimization and extended Kalman filter. elsevier. Fuzzy Sets and Systems 160 (2009) 922 – 948.

[59] S. Refoufi, K. Benmahammed. Control of a manipulator robot by neuro-fuzzy subsets form approach control

optimized by the genetic algorithms. Science Direct, elsevier. ISA Transactions 77(2018)133-145.

[60] W. Ye, B-spline-based Neuro-Fuzzy Velocity Field Navigation and Control for a Nonholonomic Mobile Robot. Proceeding of the 31st Chinesse Control. Hefei, China2012.

[61] S-M. Chen , C-Y. Wang, Fuzzy decision making systems based on interval type-2 fuzzy sets. Elsevier. Information Sciences 242 (2013) 1–21.

[62] R. Martínez-Soto, O. Castillo, L. T. Aguilar, Type-1 and Type-2 fuzzy logic controller design using a Hybrid PSO–GA optimization method. Elsevier. Information Sciences 285 (2014) 35–49.

[63] B. M. Al-Hadithi, A. Jiménez, F. Matía. A new approach to fuzzy estimation of Takagi–Sugeno model and its applications to optimal control for nonlinear systems. elsevier. Applied Soft Computing 12 (2012) 280–290.

[64] R.Solea, D.C.Cernega .Obstacle Avoidance for Trajectory TrackingControl of Wheeled Mobile Robots. Proceedings of the 14th IFAC Symposium on Information Control Problems in Manufacturing Bucharest, Romania, May 23-25, 2012.

[65] J-Ku Lee, Ji-B Park, Y-H Choi. Tracking Control of Nonholonomic Wheeled Mobile Robot Based on New Sliding Surface with Approach Angle. 3rd IFAC Symposium on Telematics Applications The International Federation of Automatic Control November 11-13, 2013. Seoul, Korea.

[66] Y. Yamomoto, X. Yun. Coordinating Locomotion and Manipulation of a Mobile Manipulator. In Y. F. Zheng(Ed.) Recent Trends in Mobile Robots, World Scientific, Singapore, 1993, pp. 157-181.

[67] Y.Kanayama, Y Kimura, F.Miyazaki, T.Noguchi. A Stable Tracking Control Method for an Autonomous Mobile Roobt. Proceedings of IEEE International Conference on Robotics and Automation,13-18 May 1990.

[68] A. De Luca, G. Oriolo, M. Vendittelli .Control of Wheeled Mobile Robots: AnExperimental Overview. S. Nicosia et al. (Eds.): RAMSETE, LNCIS 270, pp. 181−226, 2001.© Springer-Verlag Berlin Heidelberg 2001.

[69] Y.Kanayama, Y Kimura, F.Miyazaki, T.Noguchi. A Stable Tracking Control Method for a Non-Holonomic Mobile Robot. Proceedings of IROS '91, IEEE/RSJ International Workshop on Intelligent Robots and Systems '91,3-5 Nov. 1991.

[70] R.Mellah, S.Guermah, and R.Toumi .Adaptive Control of Bilateral Teleoperation System with Compensatory Neural-fuzzy Controllers. International Journal of Control, Automation and Systems 15(4) (2017) 1949-1959.

[71] R.Mellah, R.Toumi, Control Bilateral Teleoperation By Compensatory ANFIS. Springer International Publishing Switzerland 2016R.

[72] G. Klancar,I.Skrjanc, Influence of sample period variation to MPC trajectory tracking of mobile robot. ScienceDirect. IFAC-PapersOnLine 48-1 (2015) 669–670.

[73] S.A. Ahmed, M.G.Petrov, Trajectory Control of Mobile Robots using Type-2 Fuzzy-Neural PID Controller.Science Direct IFAC-Papers On Line 48-24 (2015) 138–143.

[74] R. Eder , J. Gerstmayr .Special Genetic Identification Algorithm with smoothing in the frequency domain. elsevier. Advances in Engineering Software 70 (2014) 113–122.

[75] H. R. Hassanzadeh, M-R. Akbarzadeh-T, A. Akbarzadeh, A. Rezaei .An interval-valued fuzzy controller for complex dynamical systems with application to a 3-PSP parallel robot. elsevier. Fuzzy Sets and Systems 235 (2014) 83–100.

[76] H-J. Rong, S. Han, J-M. Bai, Y-Q. Liang .Improved adaptive control for wing rock via fuzzy neural network with randomly assigned fuzzy membership function parameters. elsevier. Aerospace Science and Technology 39 (2014) 614–627.

[77] J. Armendariz, V. Parra-Vega, R. García-Rodríguez, S. Rosales .Neuro-fuzzy self-tuning of PID control for semiglobal exponential tracking of robot arms. elsevier. Applied Soft Computing 25 (2014) 139–148.

[78] A. M. Gonzalez, H.Werner .LPV Formation Control of Non-Holonomic Multi-Agent Systems. Proceedings of the 19th World Congress The International Federation of Automatic Control Cape Town, South Africa. August 24-29, 2014.

[79] E. S. El youssef, N.A. Martins, E. R. De Pieri, U. F. Moreno.PD-Super-Twisting Second Order Sliding Mode Tracking Control for a Nonholonomic Wheeled Mobile Robot. Proceedings of the 19th World Congress The International Federation of Automatic Control Cape Town, South Africa. August 24-29, 2014.

[80] C.Z.Resende, R. Carelli, S-F. Mário .A nonlinear trajectory tracking controller for mobile robots with velocity limitation via fuzzy gains. Control Engineering Practice 21 (2013) 1302–1309.

[81] D Hidalgo, P Melin, O Castillo .An optimization method for designing type-2 fuzzy inference systems based on the footprint of uncertainty using genetic algorithms. Elsevier. Expert Systems with Applications 39 (2012) 4590–4598.

[82] S. S. Roy, D. K. Pratihar .Soft computing-based expert systems to predict energy consumption and stability margin in turning gaits of six-legged robots. Elsevier. Expert Systems with Applications 39 (2012) 5460–5469.

[83] Z. Bingül , O. Karahan .A Fuzzy Logic Controller tuned with PSO for 2 DOF robot trajectory control. elsevier. Expert Systems with Applications 38 (2011) 1017–1031.

[84] C. Z. Resende, F. Espinosa, I. Bravo, M. Sarcinelli-Filho, T F. Bastos-Filho .A Trajectory Tracking Controller With Dynamic Gainsfor Mobile Robots. IEEE/RSJ International Conference on Intelligent Robots and Systems September 25-30, 2011. San Francisco, CA, USA.

[85] M. A. Vélez, O. Sanchez, S. Romero, J. M. Andujar .A new methodology to improve interpretability in neuro-fuzzy TSK models. elsevier. Applied Soft Computing 10 (2010) 578–591.

[86] M.S. Alam , M.O. Tokhi .Hybrid fuzzy logic control with genetic optimisation for asingle-link flexible manipulator. Engineering Applications of Artificial Intelligence 21 (2008) 858 – 873.

[87] S.Alavandar, M.J.Nigam .Neuro-Fuzzy based Approach for Inverse Kinematics Solution of Industrial Robot Manipulators. Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. III (2008), No. 3, pp. 224-234

[88] A. Filipescu, A.L. Stancu, S.Filipescu, G. Stamatescu .real-time sliding-mode adaptive control of a mobile platform wheeled mobile robot. Proceedings of the 17th World Congress The International Federation of Automatic Control Seoul, Korea, July 6-11, 2008.

[89] F.N.Martins, W.C.Celeste, R Carelli, M. Sarcinelli-Filho, T.F. Bastos-Filho .An adaptive dynamic controller for autonomous mobile robot trajectory tracking. Science Direct, Control Engineering Practice 16 (2008) 1354– 1363.

[90] Z.P. Wang, SS Ge, TH Lee, XC Lai. Adaptive smart neural network tracking control of wheeled mobile robots. 9th International Conference on Control, Automation, Robotics and Vision,5-8 Dec. 2006,Singapore.

[91] C.-S. chiu. Robust adaptive control of uncertain MIMO non-linear systems-feed forward Takagi-sugeno fuzzy approximation based approach. IEE proc.-control Theory Appl., vol. 125, N, (2005) pp 157-164.

[92] T, Vallée, M, Yildizoglu. Présentation des algorithmes génétiques et de leurs applications en économie .Université de Nantes, 7 septembre 2001.

[93] J.Laumond . Robot motion planning and Control. Springer 1998.

[94] A. Hait .Algorithmes pour la planication de trajectoires robustes d'un robot mobile autonome sur un terrain accidentée. thèse de doctorat à l'Université Paul Sabatier de Toulouse Spécialité: Robotique 1998.

[95] M. Fernando .Planification de mouvements avec prise en compte explicite des incertitudes géometriques. thèse de doctorat de l'institut national polytechnique de grenoble, 1996.

[96] C.Samson . Control of chained systems, application to path following and time varying point - stabilization of mobile robots. IEEE Trans, On Automatic control, Vol 40, N° 01 PP 64-77, 1995

[97] M. Fliess, J. Lévine, P. Martin, and P. Rouchon. Design of trajectory stabilizing feedback for driftless flat systems. inProc. 3rd EuropeanControl Conf., Rome, Italy, 1995, pp. 1882–1887.

[98] A. De Luca, D.Marika. D.Benedetto .Control of nonholonomic systemsvia dynamic compensation.kybernetika — Volume *29* (1993) , number 6 , pages 593 – 608.

[99] B. Armstrong, O. Khatib, J. Burdick .The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm. Proceedings ofthe IEEE International Conference on Robotics and Automation,, San Francisco, USA, April 7-10, 1986.

[100] R.W. Brockett, R.S. Millman, H.J. Sussmann . Differential geometric control theory. proceedings of the conference held at Michigan Technological University, June 28-July 2, 1982(Vol. 27), Birkhauser.

ملخص:
في هذه الأطروحة ، وصفنا شكلاً جديدًا من التحكم الغامض العصبي الجيني لنظام غير خطي مشتق من روبوت ذراع آلي وروبوت متحرك.تجمع الطريقة المقترحة بين المنطق الغامض والشبكات العصبية والتي تثير اهتمامًا متزايدًا في مجال الروبوتات ، فلا يتطلبالمتحكم الغامض العصبي معرفة القيم المميزة للربوت .علاوة على ذلك, تتطلب الخوارزميات الجينية لتطبيق الحركة المعقدة للربوتات وظيفة تقييم تأخذ في الاعتبار عدة عوامل. يتم تطبيق خوارزمية تحسين تستند إلى الخوارزميات الجينية من أجل توفير الشكل الأكثر ملاءمة للمجموعات الجزئية الغامضة والتي تعتبر كعلاقات ربط. يوفر المنهج المقترح تعلم جيد لديناميات الربوت مهما كانت المهمة المسندة إليه. وتوضح المحاكاة والنتائج العملية فعالية الإستراتجية المقترحة. إضافة إلى مناقشة مزايا الأسلوب المقترح وإمكانية التحسينات الإضافية.

كلمات مفتاحيـــة :متحكم متكيف, شكل المجموعات الجزئية الغامضة, الشبكات العصبية, الخوارزميات الجينية,الغامض العصبي الجيني, روبوت.

Résumé :

Dans cette thèse, nous décrivons une nouvelle forme de conception de contrôleur neuro-flou-génétique pour un système non linéaire dérivé d'un robot manipulateur et d'un robot mobile. La méthode proposée combine la logique floue et les réseaux de neurones qui sont de plus en plus utilisés en robotique, le contrôleur neuro-flou ne nécessite pas la connaissance des valeurs des paramètres du robot. En outre, les algorithmes génétiques pour la planification de mouvements complexes de robots nécessitent une fonction d'évaluation qui prend en compte plusieurs facteurs. Un algorithme d'optimisation basé sur les algorithmes génétiques est appliqué afin de fournir la forme la plus adéquate des sous-ensembles flous considérés comme des fonctions d'interpolation. L'approche proposée fournit un bon apprentissage de la dynamique du robot quelle que soit la tâche assignée. les résultats de simulation et pratiques illustrent l'efficacité de la stratégie proposée. Les avantages de la méthode proposée et les possibilités d'améliorations supplémentaires sont discutés.

Mots Clés **:** Contrôle adaptatif, forme de sous-ensembles flous, réseaux de neurones, algorithmes génétiques, Neuro-flou-génétique, robot.

Abstract**:**
In this thesis, we describe a new form of neuro-fuzzy-genetic controller design for nonlinear system derived from a manipulator robot and a mobile robot. The proposed method combines fuzzy logic and neuronal networks which are of growing interest in robotics, the neuro-fuzzy controller does not require the knowledge of the robot parameters values. Furthermore, the genetic algorithms for complex motion planning of robots require an evaluation function which takes into account multiple factors. An optimizing algorithm based on the genetic algorithms is applied in order to provide the most adequate shape of the fuzzy subsets that are considered as interpolation functions. The proposed approach provides a good learning technique of the robot dynamics whatever the assigned task. Simulation and practical results illustrate the effectiveness of the proposed strategy. The advantages of the proposed method and the possibilities of further improvements are discussed.

Key Words: Adaptive control, fuzzy subsets form, Neural Networks, genetic algorithms, Neuro-fuzzy-Genetic, robot.