PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF AMAR TELIDJI LAGHOUAT

Faculty of Technology - Department of Electronics

MASTER THESIS

Option: Control Systems And Industrial Computing

Theme

Adaptive Neural Network Nonlinear Control Of A Flexible Robot Manipulator

---

Supervised by
PR. BELKHIRI Mohammed

Presented by
TOUHAMI Imane
BENSAYAH Fatma

Before the jury composed of
Dr. HADJ AISSA Boubakeur
Mr. ABOUCHABANA Nabil

Academic Year 2019-2020

DEDICATION

I dedicate this modest work to my father and my mother. Words fail me to qualify their sacrifices. May God keep them for us.

My dear sisters who never ceases to encourage me and who was always there to help me.

To all my friends, may they find here the expression of my sincere friendships.

Imane

First I would thank GOD

I dedicate my dissertation work to my Mother and father whose words of encouragement and push me for tenacity ring in my ears.Also my friends. A special feeling of gratitude My brother and my sisters have never left my side

Fatma

ACKNOWLEDGEMENTS

First of all, we would like to thank “ALLAH” for giving us enough courage, strength, the good health and well-being to develop and present this humble work.
We wish to express sincere, deep gratitude to our advisor Pr. BELKHIRI Mohammed for being extremely supportive throughout our graduate studies. He has taught us much more than control system theory, and we consider ourselves very fortunate to have had the opportunity to work with him.
We would like to express our sincere gratitude to the members of the jury for taking the time to read and evaluate this work.

ABSTRACT

In this project, we present the design of an adaptive output feedback control methodology for the flexible robot benchmark problem. Its model is presented with some proposed solutions for the uncertainties due to unmodeled dynamics and parameteric uncertainties/ Our objective is to design a controller that forces the system output (the angular joint positions) to track a smooth reference trajectory with bounded errors. The proposed method is based on a feedback linearization controller associated to a feed-forward neural network and it has proven to be a very effective way to design controllers based on approximate knowledge of the system dynamics.

Key words: flexible robot, benchmark, feedback linearisation, neural networks

RÉSUMÉ

Dans ce projet, nous présentons la synthèse d'une loi de commande adaptative par bouclage non linéaire des sorties pour le problème de contrôle des robots flexibles. Son modèle est présenté avec quelques solutions proposées pour les incertitudes dues à la dynamique non modélisée et aux incertitudes paramétriques.

Notre objectif est de concevoir un contrôleur qui force la sortie du système (les positions des joints angulaires) à suivre une trajectoire de référence lisse avec des erreurs limitées.

La méthode proposée est basée sur un contrôleur de linéarisation à rétroaction associé à un réseau de neurones FFNN et s'est avérée être un moyen très efficace de concevoir des contrôleurs basés sur une connaissance approximative de la dynamique du système.

Mots clés : robot flexible, benchmark, linéarisation par bouclage nonlinéaire, réseaux de neurones

ملخص

في هذا المشروع ، نقدم تصميمًا لمنهجية التحكم باستخدام التغذية الراجعة التكيفية للروبوت المرن مرجعي. يتم تقديم نموذجها مع بعض الحلول المقترحة لحالات عدم اليقين الناتجة عن الديناميكيات غير النموذجية وأوجه عدم اليقين البارامترية / هدفنا هو تصميم وحدة تحكم تفرض إخراج النظام)مواضع المفصل الزاوي(لتتبع مسار مرجعي سلس مع أخطاء محدودة. تعتمد الطريقة المقترحة على وحدة تحكم خطية التغذية المرتدة المرتبطة بشبكات العصبية وقد أثبتت أنها طريقة فعالة للغاية لتصميم وحدات التحكم بناءً على المعرفة التقريبية لديناميكيات النظام.

الكلمات المفتاحية: روبوت مرن ، قياس الأداء ، خطية التغذية الراجعة ، الشبكات العصبية.

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

ABBREVIATIONS

CTC Computed torque controller

DOF Degree Of Freedom

$FFNN$ Feed-forward Neural Network

MLP Multi-Layer Perceptron

NN Neural Network

PB Back-Propagation

SHL Single Hidden Layer

INTRODUCTION

Industrial robots, are often designed to perform repetitive tasks that aren't easy to accomplish by human beings. A robot can be remotely controlled by a human operator to do hard and dangerous tasks. These tasks are sometimes to be realized from far distances and that causes increasing demand to improve movement speed, accuracy, trajectory tracking and eliminating uncertainties. [9]

Lately, trajectory following control of flexible-link manipulator system has been an important area of research. A non-rigid link bears resemblance to a flexible beam often used as a starting point in modelling the dynamics of a flexible link.

The trajectory control of a flexible robot can be divided into two parts. Tracking the desired trajectory on the dynamic phase of the movement and positioning the tip of the link on the final phase of the movement. Most of the existing researches on the control of flexible link manipulators focus on the positioning phase of the movement, while very few of them deal with the dynamic phase of the movement.[9]

The joint flexibility is fundamental in many robotic applications, on the other hand, it is well known that ignoring the modelling of elastic coupling between the actuators can result in oscillatory problems and the robot joints can cause instability, high frequency vibrations and reduced performance in some severe cases, joint flexibility in robots can be caused by naturally inherent of used material in structure of robots or human made that is applied intentionally. When harmonic drives, belts or long shafts are used as motion transmission elements, a dynamic displacement is introduced between the position of the driving actuators and that of the driven links, which is the output to be controlled.[16]

It is possible to consider the dynamic behaviour of a flexible manipulator as a combination of rigid-body and flexible dynamics. Accordingly, control strategies devised for such systems are to take account of both rigid-body motion and flexible motion control.

When it comes to the development of modern robot manipulators, it is necessary for the robot controller to have certain characteristics. the robot controller should have the following properties: the capability to overcome unmodeled dynamics, variable payloads, friction torques, torque disturbances, parameter variations and measurement noises which can be often presented in the practical environment. Industrial projects are expected to have accuracy, repeatability and simplicity in the realization of the control law, that is why using most excellent controllers such as nonlinear and intelligent classes are justified.[13]

The early nonlinear techniques such as computed torques, which is based essentially on input-output linearization via static feedback. When the linearized output is chosen to be the tip position, non-minimum phase behavior is introduced special inversion techniques has been presented.

There exists a gap between control theory and practise, i.e, many control methods suggested by researchers are seldom implemented in real systems and on the other hand many important industrial control problems are not studied in the academic research. In order to fill this gap we are about to present an industrial robot benchmark problem with the intention to apply advanced control algorithms in the area of adaptive control for flexible industrial manipulators using neural networks .

This master project is inspired from the research paper [12] where a benchmark problem is presented as a benchmark problem from an industrial robot constructed by ABB and the authors encourage researchers to apply advanced adaptive and robust control techniques to this real problem. A simple feedback control law was proposed for a two flexible link manipulator as a starting point for researchers to compete to achieve better performance. Our objective is to achieve a better performance by working on this benchmark problem using different controller design method and in our case we opted for the neural network as an advanced control technique that can be used to approximate uncertain terms.

The main theme of the research presented in this report is controlling a flexible robot manipulator using adaptive neural networks. The thesis is organized as follows: The first chapter gives an introduction to robotics in general, and the modeling of robot manipulators besides an overview of the benchmark problem

The second chapter is devoted for the basic notions related to nonlinear control theory, beginning from the feedback linearization to the neural network theory, where some basic definitions are given, including learning processes for neural networks

The third chapter, illustrates the main control strategy based on the augmentation of feedback linearization by neural networks.

The last chapter introduces a method consisted of applying a nonlinear controller to stabilize the output tracking error, whereas the unmatched uncertainty was canceled by choosing an appropriate reference signal. Furthermore, NNs have been applied for solving problems of trajectory tracking in the presence of parametric variation and unmodeled dynamics by employing their property of function approximation.

Finally, we will end up this study with a general conclusion in which we will summarize the main findings and results followed by presenting some perspectives for further and future research.

Chapter 1

Benchmark Robot Presentation and Modeling

1.1 Introduction

Despite the fact that there exist various types of industrial manipulators, the ones with six serially mounted links are the commonly used, their links are all controlled by electrical motors via gears. We had chosen the IRB6600 from ABB equipped with a spot welding gun as model for our control study, the dynamic of manipulator change rapidly when the robot links move fast within the manipulator workspace and dynamic coupling between the links are in general strong. Besides, the structure is elastic and the gears have nonlinearities such as hysteresis, backlash, friction and nonlinear elasticity [12].

Figure 1.1: IRB6600 from ABB equipped with a spot welding gun

The purpose of the motion control is, however, orientation control and the position of the tool when moving the tool along a certain desired path. In this chapter, we provide a brief introduction to robot dynamics together with some of its properties.

The ABB IRB 6600 robotics system is very flexible with bend-over-backwards capability. Available in five versions with a handling capacity of up to 225 kg, reach of up to 3200 mm, and a wrist torque of up to 1320 Nm. The IRB 6600 S4C+ comes with a built-in Service Information System, allowing for it to easier plan service and maintenance [1]

1.2 Rigid robots

Rigid manipulators have increased tensile strength with more precision due construction material such as steel or aluminum frames but have high cost and weight.

We are interested here in the general dynamic model of robot manipulators with n elastic joints of finite, but constant stiffness. The model contains the dynamics of $2n$ rigid bodies (n links and n actuators), coupled through the elastic joints. Let $q \in R_n \; and \; \theta \in R_n$ be, respectively, the generalized coordinates of the driven links and of the driving actuators.

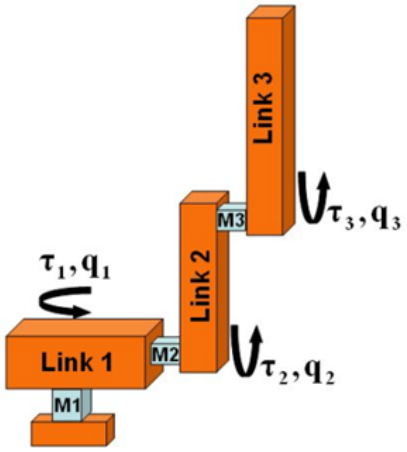


Figure 1.2: A rigid dynamic model with 3 DOF.

1.2.1 The Rigid Robot Dynamic Model

There are several methods for obtaining a rigid dynamic model. The two most common approaches are the Lagrange formulation and the Newton-Euler formulation. A third method is Kane's. All methods are based on classical mechanics.

All methods produce the same result even though the equations may differ in computational efficiency and structure.

The dynamics of n-link rigid robot manipulator is driven by the Euler-Lagrange equations as :

$$M(q)\ddot{q} + N(q, \dot{q}) + K(q - \theta) = 0 \tag{1.1}$$

$$B\ddot{\theta} + K(q - \theta) = \tau \tag{1.2}$$

Where $M(q)$ is the inertia matrix of the robot links, vector $N(q, \dot{q})$ contains the centrifugal, Coriolis and gravity forces, $K = diag\{k_1, ..., k_n\} > 0$ is the joint stiffness matrix. $B = diag\{b_1, ..., b_n\}$ is the inertia matrix of the actuators, and $\tau \in R_n$ are the motor torques.

1.3 Flexible robots

T he flexible robot system consists of a single lightweight flexible arm counterbalanced with a rigid appendage. The arm is actuated by DC motor located at the base. A servo amplifier is used to control the DC motor; this amplifier accepts control inputs from the computer through the DAS20 D/A converter in the range [-5,5] Volts. An optical encoder attached to the shaft of the DC motor is used to measure the angular position of the shaft at each sampling instant. The optical encoder's outputs are passed through a signal conditioning circuit before being acquired by the computer.[8]

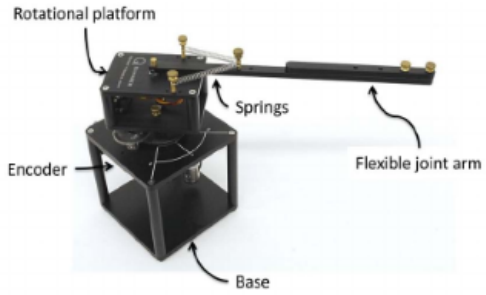


Figure 1.3: Flexible joint robot arm by Quanser.

Every mechanical system is inherently flexible, due to the elasticity of the material which composes its elements. Therefore, the rigidity assumption remains an approximation in all cases. In practice, neglecting the flexibilities of a system amounts to considering that:
- Its deformation under a maximum static load (depending on the application) is negligible.
- Its vibrations, caused by its own movements or by external stresses, give rise to negligible effects (accelerations, displacements, noise, and material fatigue)
Classic industrial robots are very massive and very rigid; their flexibility is therefore neglected in their normal conditions of use.

The flexibility of conventional industrial robots becomes significant under the effect of extreme mechanical loads (handling heavy loads) or external vibration loads (robotized machining applications, for example).[17]

1.3.1 Flexible link robot

Generally, if the segments have a significant stiffness compared to that of the connections, we can calculate equivalent joint stiffness which gives a simple model, but still representative of the real dynamics of the robot.

This approach has the advantage of introducing only one additional degree of freedom per articulation, which limits the dimension of the model obtained (number of variables and dimension of the matrices) and generally makes it possible to meet all the needs, including simulation and control.

However, this model is too limited to be able to represent complex behavior of a robot with very flexible segments.

The model of flexible joints is generally kept to represent the behavior of connections, but it must be associated with a flexible segment behavior model

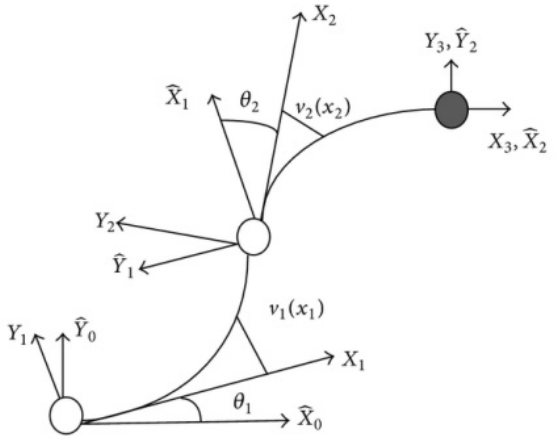


Figure 1.4: two flexile-link manipulator.

The figure 1.4 shows a two-flexible-link manipulator. This system moves in the horizontal plane and consists of two motors that generate two torques, two flexible links with mass m_i length L_i linear density ρ_i and rigidity EI_i i= (1,2), and a payload that has a mass m_pThe first link is attached to the first motor, and the second link is clamped to the rotor of the second motor. The flexible links are supposed uniform and are modeled as Euler-Bernoulli beams, and the deformations are assumed to be small.

1.3.2 Flexible Link Models

If the elasticity of the links cannot be neglected a distributed elasticity model can be used to increase the accuracy of the model. These models are described by partial differential equations with infinite dimension. One way to reduce the model to a finite-dimensional model is to use the assumed modes method. The link deflections are then described as an infinite series of separable modes that is truncated to a finite number of modes.[7]

Using Lagrange equations, the dynamical model of an n DOF flexible manipulator is given by [8]

$$M(q)\ddot{q} + H(q,\dot{q}) + D\dot{q} + K(q) = \tau \tag{1.3}$$

Where M is the inertia and mass matrix , $H(q,\dot{q}))$ is the Coriolis and centrifugal forces vector, D is the friction matrix, and K is the rigidity matrix. q Represents the vector of the generalized coordinates and τ is the vector of the applied torques. For the n rigid coordinates and n flexible links, the deformation of the i_{th} flexible link is given by the following equation:

$$V_i(x,t) = \sum_{i=1}^{Z_i} \phi_{ij}(x) q_{ij}(t) \tag{1.4}$$

$$V_i(x,t) = \sum_{i=1}^{Z_i} \phi_{ij}(x) q_{ij}(t), i = 1, ..., n.$$

Where q_{ij} is the j_{th} generalized flexible coordinate, $\phi(x)$ is its j_{th} shape function, and Z_i is the number of the retained flexible modes of the i_{th}flexible link. We may notice that the total number of the flexible modes is $Z = \sum_{i=1}^{n} Z_i$ and the number of the rigid modes is n.

1.3.3 Flexible joint robot

Elastic gear transmissions and rigid links can be described by the so called simplified flexible joint model where the inertial coupling between the links and the motors are neglected. This approximation is valid for a reasonable high gear ratio. Furthermore, the viscous damping is also neglected in the simplified model.[11]

The equations of motion are derived by computing the linear and angular momentum and their time derivatives. By using Kane's method (Kane and Levinson, 1985; Lesser, 2000) the projected equations of motion are derived to yield a system of ordinary differential equations (ODE) with minimum number of DOF.

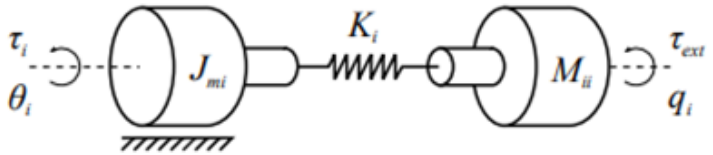


Figure 1.5: Schematic representation of a flexible joint.

1.3.4 The Flexible Joint Dynamic Model

This robot can be modeled by the so called flexible joint model which is illustrated in Figure 1.5
The model equations can be described as a system of second order ODE's [11]

$$M_a(q_a)\ddot{q}_a + n(q_a, \dot{q}_a) + K(q_a - q_m) = 0 \tag{1.5}$$

$$M_m\ddot{q}_m + K(q_m - q_a) = u \tag{1.6}$$

Where $\dot{x}$ Denotes dx/dt. $M_a(q_a) \in R^{NxN}$ is the inertia matrix for the links.

$$n(q_a, \dot{q}_a) = c(q_a, \dot{q}_a) + g(q_a) \tag{1.7}$$

where $c(q_a, \dot{q}_a) \in R^N \; and \; g(q_a) \in R^N$ describes the Coriolis, centrifugal and gravity torques.
$M_m \in R^{NxN}N$ is the diagonal inertia matrix of the motors.
The link and motor angular positions are denoted $q_a \in R^N$ and $q_m \in R^N$ respectively.
Note that the gear ratio matrix, r, is not explicitly shown in the equations. All equations are expressed on the link side and $M_m = r^T {M_m}^m r$, where ${M_m}^m$ is the motor inertia matrix on the motor side.

1.4 The benchmark problem

We are focusing only on the regulator problem, a method is presented where a feedback controller should be designed such that the tool position is closed to the desired reference, in the presence of motor torque disturbance. In order to get a powerful dynamic coupling, just the second and third links of the manipulator are taken in the benchmark model. The nonlinear rigid body dynamics related with the change of configuration (link positions) as well as gravity, centripetal and Coriolis torques are included in the model.[12]

1.5 Mathematical models

The figure below illustrates the planar model, all movements are constrained to the x, z plane. The links are indicated as link 1 and link 2 each of the two has the following rigid body attributes:

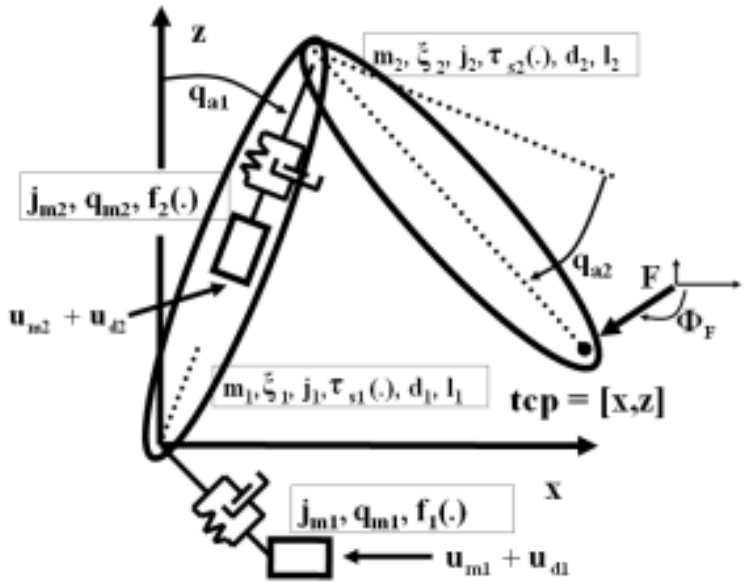


Figure 1.6: Two link robot model

- mass m_1 and m_2
- link length l_1 and l_2
- center of mass ε_1 and ε_2 (distances from the centers of rotation)
- inertia w.r.t. center of mass j_1 and j_2

The links are actuated by electrical motors, connected to the links via elastic joints. The nonlinear spring torque $\tau_s(q)$ describes the joints (gear transmissions)
D is the linear damping matrix, f($\dot{q}$) the friction torque n_1 and n_2 the gear ratios, jm_1 and jm_2 motors inertias There are two degrees of freedom (DOF) for each axis described by the motor angular positions qm1, qm2 and link angular positions qa1, qa2.

The motor torques um_1 and um_2 represents the control signals, which are subject to saturation. The motor torque control is modeled as a gain uncertainty γ Time delay Td_1, Td_2 Two sources of disturbance are acting on the system. A force F is applied at the tool center point (TCP) at angle ϕF and a motor torque disturbance is applied as input disturbance signals ud_1 and ud_2. The angular positions and the model inputs are described by :

$$q = \begin{pmatrix} qa_1 \\ qa_2 \\ qm_1/n_1 \\ qm_2/n_2 \end{pmatrix} \tag{1.8}$$

$$u = \begin{pmatrix} ua_1 \\ ua_2 \\ um_1 + ud_1/n_1 \\ um_2 + ud_2/n_2 \end{pmatrix} \tag{1.9}$$

1.6 Dynamic robot models

Dynamic robot models describe the relations between the motions of the robot and the forces that cause the motions. [7] To obtain the dynamic model, we can use several formalism: - Newton-Euler formalism. - The Alembert formalism which uses the principle of virtual powers. - The formalism of Lagrange equations. -The general robot dynamics equation is shown as

$$u = M(q)(\ddot{q}) + C(q, \dot{q}) + G(q) + D\dot{q} + \tau_s(q) + f(\dot{q}) \tag{1.10}$$

Since the gear ratio is high we neglect the inertial coupling between the motor and link rotation. The inertia matrix M, gravity vector G and vector of speed dependent torques (Coriolis and centripetal) C, can then easily be described as

$$M(q) = \begin{pmatrix} j_{11} & j_{12} & 0 & 0 \\ j_{21} & j_{22} & 0 & 0 \\ 0 & 0 & jm_1 n_2^2 & 0 \\ 0 & 0 & 0 & jm_2 n_2^2 \end{pmatrix} \tag{1.11}$$

$$j_{11}(q) = j_1 + m_1 {\xi_1}^2 + j_2 + m_2({l_1}^2 + {\xi_2}^2 - 2l_1\xi_2 \sin q_{a2}) \tag{1.12}$$

$$j_{12}(q) = j_2 1(q) = j_2 + m_2(\xi_2^2 - l_1\xi_2 \sin q_{a2}) \tag{1.13}$$

,

$$j_{22}(q) = j_2 + m_2 {\xi_2}^2 \tag{1.14}$$

$$G(q) = [g_1(q)\ g_2(q)\ 0\ 0]^T \tag{1.15}$$

$$g_1(q) = -g(m_1\xi_1(sinq_a 1) + m_2(l_1 \sin(q_a 1) + \xi_2 \cos q_a 1 + q_a 2))) \tag{1.16}$$

$$g_2(q) = -m_2\xi_2 g \cos(q_a 1 + q_{a2}) \tag{1.17}$$

,

$$C(q,\dot{q}) = \begin{pmatrix} -m_2 l_1 \xi_2 g \cos(q_{a2})(2\dot{q}_{a1}\dot{q}_{a2} + \dot{q}_{a2}^2) \\ -m_2 l_1 \xi_2 g \cos(q_{a2})(\dot{q}_{a2}^2) \\ 0 \\ 0 \end{pmatrix} \tag{1.18}$$

Where g is the gravitational constant

The nonlinear torque is given by

$$\Gamma_s(q) = \begin{pmatrix} \tau_{s1}(\Delta_{q1}) \\ \tau_{s2}(\Delta_{q2}) \\ \tau_{s1}(-\Delta_{q1}) \\ \tau_{s2}(-\Delta_{q2}) \end{pmatrix} \tag{1.19}$$

$$\Delta_{q_i} = q_{ai} - q_{mi}/n_i i = 1,2 \tag{1.20}$$

With

$$\tau_{si} = K_{i1}\Delta_{qi} + K_{i3}{\Delta_{qi}}^3, |\Delta_{qi}| \leq \psi_i \tag{1.21}$$

$$\tau_{si} = sign(\Delta_{qi})(m_{i0} + m_{i1}(|\Delta_{qi}| - \psi_i)) > \psi_i \tag{1.22}$$

$$k_{i1} = {k_i}^{low} \tag{1.23}$$

$$k_{i3} = ({k_i}^{high} - {k_i}^{low})/(3{\psi_i}^2) \tag{1.24}$$

$$m_{i0} = K_{i1}\psi_i + K_{i3}{\psi_i}, ^3 \tag{1.25}$$

$$m_{i1} = {k_i}^{high} \tag{1.26}$$

$$D = \begin{pmatrix} d_1 & 0 & -d_1 & 0 \\ 0 & d_2 & 0 & -d_2 \\ -d_1 & 0 & d_1 & 0 \\ 0 & -d_2 & 0 & d_2 \end{pmatrix} \tag{1.27}$$

$$f(\dot{q}) = [0\ 0\ f_1(\dot{q})\ f_2(\dot{q})]^T \tag{1.28}$$

$$f_i(\dot{q}) = n_i(f_{di}\dot{q}_{mi} + (f_{ci}(\mu_{ki} + (1-\mu_{ki})\cosh^{-1}(\beta_i\dot{q}_{mi}))\tanh(\alpha_i\dot{q}_{mi})), i = 1,2 \tag{1.29}$$

If we take into consideration the absence of friction and disturbances, the dynamics of manipulator can be generalized in the following form

$$u = M_m(q)(\ddot{q}) + C_m(q,\dot{q}) + G_m(q) + D\dot{q} + \tau_s(q) \tag{1.30}$$

with

$$q = \begin{pmatrix} q_{m1}/n_1 \\ q_{m2}/n_2 \end{pmatrix},\ u = \begin{pmatrix} (u_{m1} + u_{d1})n_1 \\ (u_{m2} + u_{d2})n_2 \end{pmatrix} \tag{1.31}$$

$$M_m(q) = \begin{pmatrix} J_{11}(q) & J_{12}(q) \\ J_{12}(q) & J_{22}(q) \end{pmatrix},\ C_m = \begin{pmatrix} -m_2 l_1 \xi_2 g \cos(q_{a2})(2\dot{q}_{a1}\dot{q}_{a2} + \dot{q}_{a2}^2) \\ -m_2 l_1 \xi_2 g \cos(q_{a2})(\dot{q}_{a2}^2) \end{pmatrix} \tag{1.32}$$

$$G_m = \begin{pmatrix} -g(m_1\xi_1(sinq_a1) + m_2(l_1 \sin(q_a1) + \xi_2 \cos q_a1 + q_a2))) \\ -m_2\xi_2 g \cos(q_a1 + q_{a2}) \end{pmatrix} \tag{1.33}$$

1.6.1 Dynamic model properties

We will present the properties of the different terms constituting the equation of the dynamic model of a flexible manipulator robot.

These properties can be used for the analysis of the behavior of the system and the calculation of the control law.

Properties of the inertia matrix

The inertia matrix M(q) is a symmetric positive definite matrix which checks:

$$\mu_1 I_N < M(q) < \mu_2 I_N \tag{1.34}$$

μ_1, μ_2 are two strictly positive values and I_N the identity matrix Likewise, we have the property :

$$\frac{1}{\mu_1} I_N < M(q) < \frac{1}{\mu_2} I_N \tag{1.35}$$

This property is a very well-known result in mechanics; it determines the borders of the Inertia matrix $M(q)$ in the case of rotoid joints.

If the joints are prismatic, the above property can be stated as follows:

$$M_1 < M(q) < M_2 \tag{1.36}$$

Properties of the vector of the Coriolis and centrifugal forces

The matrix C(q, $\dot{q}$) is characterized by three properties:

Property 1 :
The matrix

$$N(q, \dot{q}) = M(q) + 2C(q, \dot{q}) \tag{1.37}$$

is an skew-symmetric matrix.
It means $q^T N(q, \dot{q}) = 0\ q \in R^{n+1}$.

This equation means that the sum of the internal interaction forces of the mass of the manipulator robot is zero.

Property 2 :
Whatever the two vectors x, y then:

$$C(q,x)y = C(q,y)x(x,y) \in R^N x R^N \tag{1.38}$$

Property 3 :
The vector C(q,$\dot{q}$)$\dot{q}$ represents the torques due to the Coriolis force and centrifugal.
It verifies:

$$C(q,\dot{q})\dot{q} \leqslant C_0, \|\dot{q}\|^2 \tag{1.39}$$

For a certain constant $C_0 > 0$. The constant C_0 , can be interpreted as being the total torque generated by the centrifugal forces of the manipulator robot.

Properties of the vector of the forces of gravity

The vector G(q) checks:

$$G(q) \leqslant g_m$$

Or g_m is a scalar function, constant in the case of only rotoid joints. It depends on q in the case of only prismatic joints.
Based on their make, robotic manipulators can be categorized as rigid and flexible.

1.7 Conclusion

The industrial manipulator control problem is a challenging task. The control of such manipulators can be described and classified in many ways. According to the type of mechanical arm to control, the type of drive system, control of Robot Manipulators, type of model used for the (model-based) control etc…

Flexible joint and flexible link manipulators control is a large research area with various publications, and almost every possible control method ever invented has been suggested for dealing with these systems.

In this thesis we will try to solve the tracking problem to a flexible robot manipulator using adaptive neural network controland the adpative law is derived and ensured to converge by Lyaponov's second method.

Chapter 2

Nonlinear Flexible Robot Control

2.1 Introduction

A countless control strategies for manipulators have been the focus of several studies in recent years, among these strategies, controlling robots with flexible arms in order to reach and maintain a desired position is a challenge. Many control methods with efficient controllers can be created if the parameters are fixed and known, but the variation in system parameters due to changes in working conditions requires control laws that use variable gains.

If we are treating a non-linear system where the conditions on the precision and other dynamic characteristics are precise, this kind of controller can prove to be very ineffective. In this case it is necessary to develop control laws which are insensitive to disturbances and non-linearities. A common problem to many systems, for which there are many solutions, non-linear control laws are the formula to these obstruction.

The objective of this chapter is to present an overview on advanced nonlinear control techniques for robot control and the neural networks methods in control.

2.2 Nonlinear robot control

The aim of a robot controller is to calculate the torque that must be developed by actuators to make it perform the desired task. In the development of modern robot manipulators, it is required that the robot controller has the capacity to overcome unmodeled dynamics, variable payloads, friction torques, torque disturbances, parameter variations, measurement noises which can be often presented in the practical environment.

In nonlinear control field, a common strategy is called model based control, which can be derived from the mathematical model of the system.

However, in case of robot manipulator, it is weakened by inaccuracies presented in the robot model, where the performance of the control algorithm is not guaranteed. These inaccuracies can be defined as parametric uncertainties, unmodeled dynamics, and unknown external disturbances. To overcome the uncertainties drawback, robust nonlinear control can be a solution.

The goal of robust control is to maintain performance in terms of stability, tracking error, or other specifications despite inaccuracies present in the system.

2.3 PD controller

We are going to present the most common controller applied to manipulator robots, which we can integrate into our mechanism. First this command deals with the regulation problem. The PD (Proportional and Derivative) command is the simplest command among the dynamic commands, it ensures the overall asymptotic stability of the robot's dynamics in a closed loop but using large gains.

It is only effective in the case where the parameters of the system to be controlled (robot) are well defined, i.e. no uncertainties, no friction and no disturbances. Otherwise, it is recommended to use the so-called robust control laws.

2.4 PD type control law

As a first step we apply a PD command on the manipulator robot, neglecting terms of friction and external disturbances, the equation of the robot becomes

$$(q)q(\ddot{q}) + C(q, \dot{q}) + G(q) = \tau \tag{2.1}$$

The tracking objective is to maintain the position of the robot joint position q constant around a constant desired position denoted q_d. The derived proportional control (PD) has the following basic form

$$\tau = K_p(q_d - q) - K_v\dot{q} \tag{2.2}$$

Let's take $\widetilde{q}$ = q - q_d position error then we have $\dot{\widetilde{q}} = \dot{q}$ and $\ddot{\widetilde{q}} = \ddot{q}$
since $\dot{q_d}$ = 0 by applying This control law on the dynamics of the robot, we obtain the dynamics of the closed loop system

$$M(q)q(\ddot{q}) + C(q, \dot{q}) + G(q) + K_v\dot{q} + K_pq = 0 \tag{2.3}$$

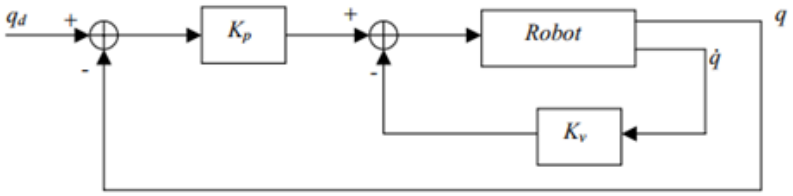


Figure 2.1: Block diagram of the PD command

2.5 Feedback linearization

The basic idea of simplifying the form of a system by choosing a different state representation is not completely unfamiliar; rather it is similar to the choice of reference frames or coordinate systems in mechanics. [7]

It is known as a way of transforming original system models into equivalent linear models of a simpler form.

2.5.1 Input/Output Linearization

A control technique where the output y of the dynamic system is differentiated until the physical input u appears in the $r - th$ derivative of y. Then u is chosen to yield a transfer function from the "synthetic input",v, to the output y which is:

$$\frac{Y(s)}{v(s)} = \frac{1}{s^r}$$

If r, the relative degree, is less than n, the order of the system, then there will be internal dynamics. If $r = n$, then I/O and I/S linearizations are the same.

2.5.2 Input/State Linearization

A control technique where some new output $y_{new} = h_{new}(x)$ is chosen so that with respect to ynew, the relative degree of the system is n. Then the design procedure using this new output ynew is the same as for I/O linearization.

2.5.3 Input/State Linearization

A control technique where some new output $y_{new} = h_{new}(x)$ is chosen so that with respect to ynew, the relative degree of the system is n. Then the design procedure using this new output y_{new} is the same as for I/O linearization.

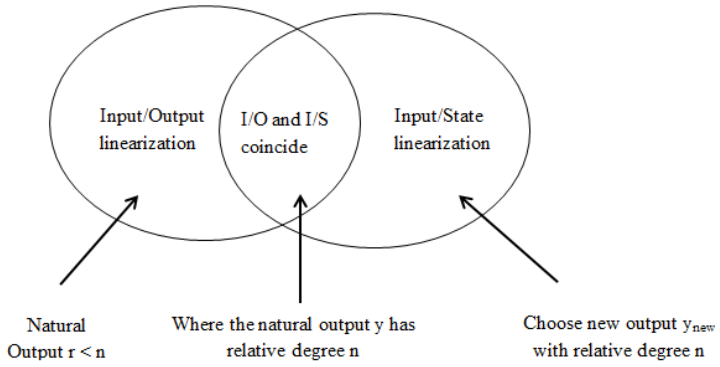


Figure 2.2: Input/State Linearization

2.5.4 Feedback Linearization for MIMO Nonlinear Systems

Consider a "square" system (where the number of inputs is equal to the number of outputs = n)

$$\begin{cases} \dot{x} = f(x) + \sum_{i=1}^{m} g_i u_i \\ y = [h_1, h_2, ...h_m]^T \end{cases} \tag{2.4}$$

$$\dot{y_k} = L_f(h_k) + \sum_{i=1}^{m} L_{g_i} h_k u_i \tag{2.5}$$

Let r_k, the relative degree, be defined as the relative degree of each output, i.e. For some I,$L_{g_i}({L_f}^{r^-1_k}(h_k)) \neq 0$. Let J(x) be an mxm matrix such that:

$$\begin{pmatrix} L_{g_1}({L_f}^{r^-1_1}(h_1) & ... & L_{g_m}({L_f}^{r^-1_1}(h_1) \\ ... & ... & ... \\ L_{g_1}({L_f}^{r^-1_m}(h_m) & ... & L_{g_m}({L_f}^{r^-1_m}(h_m) \end{pmatrix} \tag{2.6}$$

$J(x)$ is called the invertibility or decoupling matrix. We will assume that $J(x)$ is non-singular.

Let:

$$y^r = \begin{pmatrix} \frac{d^{r_1} y_1}{dt^{r_1}} \\ \dots \\ \frac{d^{r_m} y_1}{dt^{r_m}} \end{pmatrix} \tag{2.7}$$

where y^r is an mx1 vector

$$y^r = \begin{pmatrix} {L_f}^{r^-1_1}(h_1) \\ \dots \\ {L_f}^{r^-1_m}(h_m) \end{pmatrix} \tag{2.8}$$

Then we have

$$y^r \equiv l(x) + J(x).u \equiv v$$

We obtain a decoupled set of equations :

$$\begin{cases} \frac{d^{r_1}}{dt^{r_1}} = v_1 \dots\dots \frac{d^{r_m}}{dt^{r_m}} = v_m \end{cases} \tag{2.9}$$

so $y \Leftrightarrow v$
Design v any way you want to using linear techniques

$$u = J^{-1}(v - l) \tag{2.10}$$

Applying feedback linearization on physical systems (e.g., robot manipulators) is called computed torque control. Computed torque controller (CTC) is a powerful nonlinear controller used in control of robot manipulator. It is based on feedback linearization and computes the required arm torques using the nonlinear feedback control law.

It works very well when all dynamic and physical parameters are known but when the robot manipulator has variation in dynamic parameters the controller has no acceptable performance.

In practice, real physical systems have unmodelled dynamics and uncertainties and in this project we are trying to use Neural Networks as a tool that estimates uncertainties combined with existing controllers.

2.6 Neural networks

2.6.1 History of Neural Networks

Neural networks are composed of a massive connection of simple elements called neurons. The philosophy behind these neural networks is to imitate the human brain and understand how it works to build artificial intelligent systems.

The first appearance of neural networks was in 1943 when Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neurons might work. They employed electrical circuits to model a simple neural network. [3]

After that, Donald Hebb affirmed the last approach by writing The Organization of Behavior in 1949, a book which indicates that neural pathways are strengthened each time that they are used.

The evolution of computers in 1950s allowed the possibility to begin modeling the rudiments of these theories concerning human thought and the first simulation of the neural network was made by Nathanial Rochester from the IBM research laboratories even but that first attempt has failed.

In 1959, Bernard Widrow and Marcian Hoff of Stanford developed models they called ADALINE and MADALINE. These models were named for their use of Multiple ADAptive LINear Elements. MADALINE was the first neural network to be applied to a real world problem. It is an adaptive filter which eliminates echoes on phone lines. This neural network is still in commercial use. [3]

In 1982 several events caused a renewed interest. John Hopfield of Caltech presented a paper to the national Academy of Sciences. Hopfield's approach was not to simply model brains but to create useful devices. With clarity and mathematical analysis, he showed how such networks could work and what they could do. Yet, Hopfield's biggest asset was his charisma. He was articulate, likeable, and a champion of a dormant technology.[3]

Today, neural networks discussions are occurring everywhere. Their promise seems very bright as nature itself is the proof that this kind of thing works. Yet, its future, indeed the very key to the whole technology, lies in hardware development. Currently most neural network development is simply proving that the principal works. This research is developing neural networks that, due to processing limitations, take weeks to learn. To take these prototypes out of the lab and put them into use requires specialized chips. Companies are working on three types of neuro chips - digital, analog, and optical. Some companies are working on creating a "silicon compiler" to generate a neural network Application Specific Integrated Circuit (ASIC). These ASICs and neuron-like digital chips appear to be the wave of the near future. Ultimately, optical chips look very promising. Yet, it may be years before optical chips see the light of day in commercial applications. [3]

2.6.2 Biological Neural Network

A neuron (or nerve cell) is an information-processing biological organ. This consists of one cell nucleus, or soma, and two types of tree-like out-reaching branches: the axon and the dendrites.

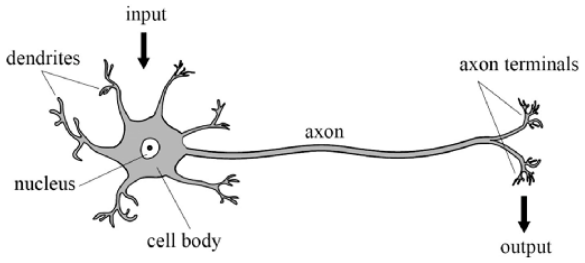


Figure 2.3: The-biological-neuron

The cell body has a nucleus containing information on generic traits and plasma which holds the molecular equipment required by neuron to produce material.

A neuron, through its dendrites (receivers) receives signals (impulse from other neurons and transmits signals produced by its cell body along the axon (transmitter), which gradually branches into strands and sub strands. A synapse is a simple structure and functional unit between two neurons (one axon strand of one neuron and the other dendrite).[2]

Once the impulse hits the terminal of the synapse, certain chemicals called neurotransmitters are produced. The neurotransmitters spread through the synaptic distance to boost or suppress the receptor neuron's own ability to release electrical impulses, depending on the form of synapse.

The strength of the synapses can be modified by the signals that pass through it, so that the synapses can benefit from the behaviors they engage in. This dependence on history serves as a memory which could be responsible for human memory.[2]

2.6.3 Artificial (formal) Neuron

The first model of the formal neuron dates from the forties. It was presented by Mc Culloch and Pitts[1]. Inspired by their work on biological neurons they proposed the following model:

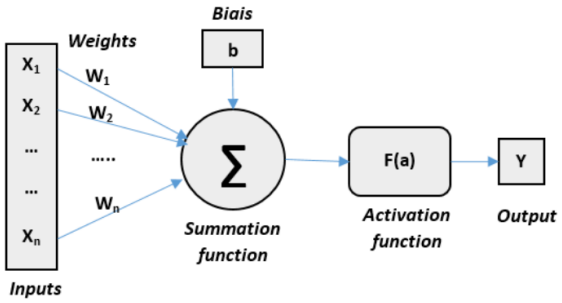


Figure 2.4: The formal neuron model

An artificial neuron is a mathematical function conceived as a crude model, or abstraction of biological neurons.

Artificial neurons are the constitutive units in an artificial neural network. Depending on the specific model used, it can receive different names, such as semi-linear unit, Nv neuron, binary neuron, linear threshold function or McCulloch-Pitts (MCP) neuron.

The artificial neuron receives one or more inputs (representing the one or more dendrites) and sums them to produce an output (representing a biological neuron's axon). Usually the sums of each node are weighted and the sum is passed through a non-linear function known as an activation function or transfer function. The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions, or step functions. [2]

More generally, a formal neuron is a processing element (operator mathematical) with n inputs (which are the external neurons or the outputs of others neurons), and only one output.

This model is described mathematically by the following equations:

$$S = \sum_{i}^{n+1} w_i x_i \tag{2.11}$$

$$y = f(x) \tag{2.12}$$

Where w_i, x_i, f, y are respectively, the synaptic weights (parameters), the inputs, the activation function and neuron output.

We can summarize the difference between artificial and biological neuron in the figure 2.5

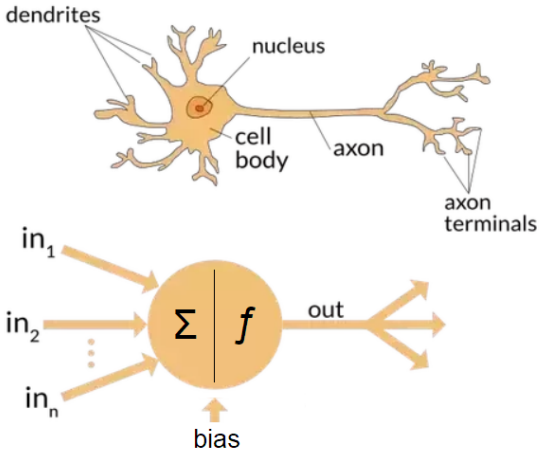


Figure 2.5: Artificial and biological neuron

2.6.4 The activation function

Activation functions generally represent certain forms of no linearity. One of the simplest forms of non-linearity, which is suitable for networks discrete, is the sign function 2.6a.

Another variant of this type of nonlinearity is the step function 2.6b. For the majority of learning algorithms it is necessary to use differentiable sigmoid functions, such as the sigmoid function unipolar, Figure 2.6c and the bipolar sigmoid function 2.6d. [1]

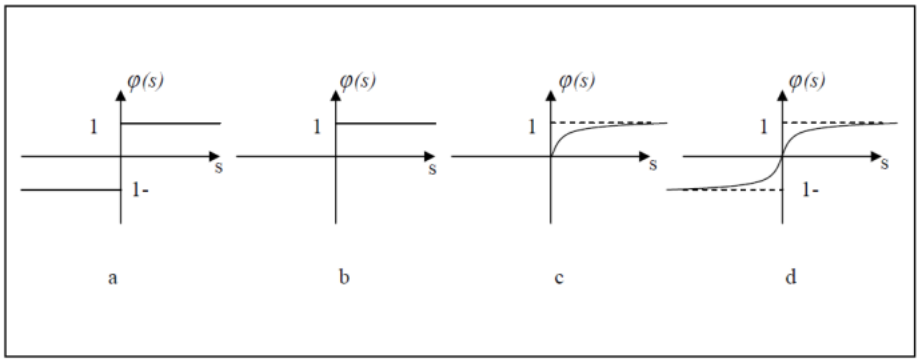


Figure 2.6: activation function types

2.6.5 Selecting the activation function

The selection of activation function bears an important effect on the neural network results. It constitutes a mathematical transformation of the summarized weighted input units to generate the output of the neuron. the most used activation functions, in the field of modeling and control of nonlinear systems is the bipolar sigmoid function.

Technically speaking an activation function comprises of two parts; a combination function that factors in all the input units into a single value (Weighted sum of inputs) and a transfer function which applies a nonlinear transformation to these Summarized units to trigger an output unit. Usually, the data pattern determines the forms of the transfer functions.

Most studies resort towards the sigmoid transfer functions. The benefits of logistic or sigmoid function is that they are continuous functions that monotonically rise or fall, saturate towards the minimum and maximum values, approximate the step function very well and are differentiable on the whole domain and can thereby dramatically reduce the computation burden for training It is important to note that NN where the hidden neurons have sigmoid activation function and the output neurons the sigmoidal or identity function are called Multi-Layer Perceptrons (MLP).[2]

2.6.6 Multi-Layer Perceptron

We present here one of the most used network architectures. It is a class of feedforward artificial neural network that has at least three layers of nodes.

It generates a set of outputs $\{y_1, y_2, ...y_m\}$ a set of inputs $\{x_1, x_2, ...x_n\}$. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. The spread of information is thus proceeds in one direction from the input layer to the output layer. The activation function used for neurons can be any function increasing and differentiable.

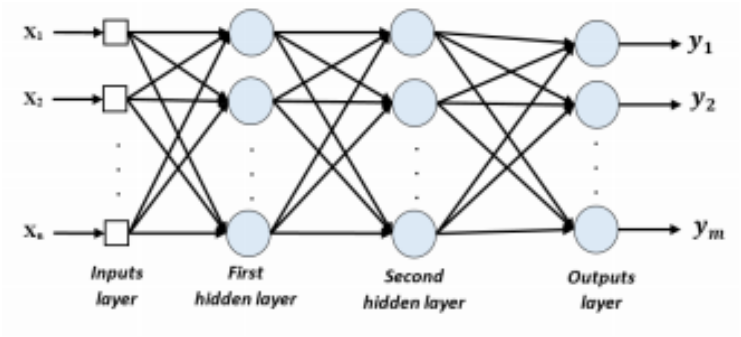


Figure 2.7: MLP model

A neural network is trained with input and target pair patterns with the ability of learning. MLP can separate data that is not linearly distinguishable. It is especially trained using a supervised learning technique called back-propagation (PB) algorithm, which aims at minimizing the global error measured at the output layer by the relation bellow [15]

$$e(t) = y_d(t) - y_m(t) \tag{2.13}$$

Where $y_d(t)$ denotes the desired output, and $y_m(t)$ the measured output of the neuron.

The BP algorithm uses an iterative supervised learning procedure, where the MLP is trained with a set of predefined inputs and outputs. The global error $E_d(t)$ calculated by the equation below, this error can be minimized by the gradient descent technique.[11]

$$E_d(t) = \frac{1}{2}\sum_{i=1}^{n}(y_{d,i}(t) - y_{m,i}(t))^2 \tag{2.14}$$

2.6.7 Network Architecture

Neural networks composed of only two layers, the input and output layer is the simplest that means no hidden layers. This is often called the skip layer which basically constitutes a conventional linear regression modeling in a NN design whereby the input and output layers are connected directly, hence bypassing the hidden layer.

Like any other networks, this form of NN depend on Weight as the connection between an input and the output; the weight representing the relative Significance of a specific input in the computation of the output.

It is important to bear in mind that the Output generated will heavily depend on the activation function type we are going to use. However, since the hidden layer confers strong learning ability to the NN, in practical applications, a three and over three NN architecture is used..

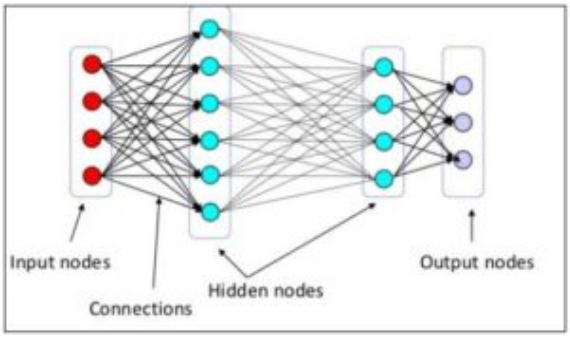


Figure 2.8: Neural networks layers

The figure 2.8 shows the neural networks layers, it is important to differentiate between two classes of weights in NN; there are those who connect the inputs to the hidden layer and then those weights that join the hidden layer with the output layer. Otherwise there are two classes of activation functions; one is in the hidden layer and the other in the output layer.

An infinite number of ways prevail as to the construction of a NN so there are two terms used to describe the way in which a NN is organized. First the neurodynamics (fundamentally gives the properties of an individual neuron such as its transfer function and the combination of the inputs) and architecture (defines the structure of NN including the number of neurons in each layer And the number of types of interconnections) The following elements mast be taken into consideration when building up a network. - Best starting values (weight initialization)
- Number of hidden layers
- Number of neurons in each hidden layer
- Number of input variables or combination of input variables
- Learning rate
- Momentum rate
- Training time or amount of training (i.e., the number of iterations to employ)
- Type of activation function to use in the hidden and output layers

2.6.8 Neural networks Learning

The need for learning arises when the a priori information is incomplete, and its type depends on the degree of completeness of this information, Like the information that can acquire a neural network is represented in the weights of the connections between neurons, learning therefore consists in adjusting these weights in such a way that the network exhibits certain desired behaviors.[1] Mathematically learning is defined by:

$$\frac{\partial w}{\partial t} \neq 0 \tag{2.15}$$

Where w is weight matrix.
We can distinguish three types of learning, supervised learning, unsupervised learning and reinforcement learning.

Supervised Learning

Network is provided with a correct answer (output) for every input pattern(presence of a master who provides the desired response).

Besides weights are determined to allow the network to produce answers as close as possible to the known correct answers. The back-propagation algorithm belongs into this category.[2]

This procedure is repeated until a performance criterion is satisfied. Once the learning procedure is completed, the synaptic coefficients take optimal values with regard to the stored configurations and the network can be operational

Unsupervised Learning

This type of learning does not require a correct answer associated with each input pattern in the training set and the learning procedure is based only on input values. It explores the underlying structure in the data, or correlations between patterns in the data and organizes patterns into categories from these correlations.

The network is self-organized in such a way as to optimize a certain cost function, without being given the answer desired. This property is called self-organization. The Kohonen algorithm belongs into this category.[1][2]

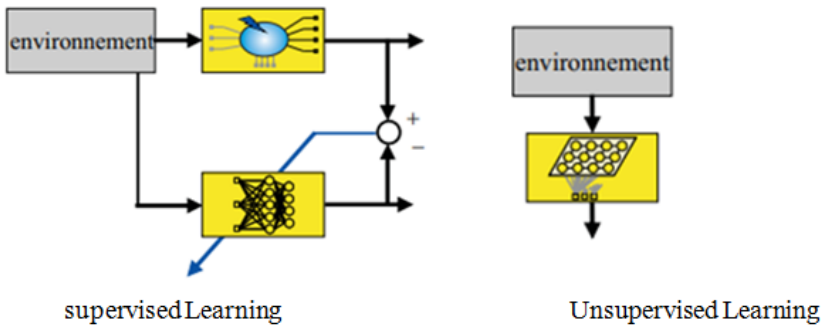


Figure 2.9: Principle of supervised and unsupervised learning

Reinforced Learning

Combines supervised and unsupervised learning; Part of the weights are determined through supervised learning and the others are obtained through a unsupervised learning.

The basic idea of reinforcement learning is inspired by the mechanisms learning in animals. In this type of learning we assume that there is no master (supervisor) who can provide the correct answer, but the system to be trained is indirectly informed about the effect of their chosen action. This action is reinforced if it leads to an improvement in the performance of the driven system, and the elements which contribute in the generation of this action are either rewarded or punished

2.7 Robotics and neural networks

There are historical parallels between the study of neural networks and robots. Both are biologically inspired in their origin, and both have acquired important engineering applications in their own right.[18]

Robots are physical devices that substitute for (or replicate) human actions which involve interaction with the world such as manipulation or locomotion. Robotics aim to create systems that can seamlessly combine visual inputs with motor responses, including in the face of novel stimuli and environmental shifts. Living systms ' capacity to learn and adapt offers the quality by which robotic systems are assessed.[18]

A variety of scientists have attempted to develop robot controllers that are based on known processes in the brain and musculoskeletal system to mimic those abilities. These theoretical models have two purposes: they form the basis for biologically inspired approaches to robot control design while at the same time providing some insight into the brain mechanical conduct.[18]

2.8 Identification of nonlinear systems by neural networks

The modeling of nonlinear systems by neural networks was the subject of a lot of research over the past ten years because of the capacity learning, approximation and generalization that these networks have.[14] [6]

In indeed, this new approach provides an effective solution through which large classes nonlinear systems can be modeled without a precise mathematical description.

Identification is the operation of determining the dynamic model of a system from input / output measurements. Often the measured output of systems is tainted with noise. This is due either to the effect of disturbances acting at different places in the process, or measurement noises. These disturbances introduce errors in the identification of model parameters.[1]

The use of neural networks to model non-linear systems naturally follows their approximation and generalization skills. The establishment of neuronal identification model of a system generally involves the following steps:

- Acquisition of training and test data,
- Choice of model structure.
- Estimating the model parameters.
- Validation of the identified model.

The first step provides the input / output data likely to allow extracting a significant process model, the second step is to choose the structure of the model likely to represent the dynamics of the system, the architecture of the neural network and its inputs. Static multilayer networks are the most used because of the simplicity of their learning algorithms and their approximation skills and generalization. There are no general methods for choosing the number of neurons on each hidden layer as well as the number of these. However, a hidden single-layer network is in most cases sufficient. In reference to the theory of linear systems, several nonlinear models have been proposed. If identifying a system allows a model to reproduce its behavior, then the connectionist models are one solution among

many. The use of networks of neurons does not, however, lead to an analytical formulation, or even to know the values of the parameters. The use of a connectionist model allows by against getting a ”mapping” between the input and output space of the system. Her accuracy depends on the number of neurons involved and the inherent complexity of the network structure. [1]

Connectionist models are not parametric models at the conventional meaning. They have their own representation and their own parameters (weights, activation functions). To an input stimulus, they ”respond” with the output that they have learned to associate with it. As they have generalization capabilities, at an entrance that never been presented they return a response which is sort of an average of the outputs associated with the nearest entries.[19]

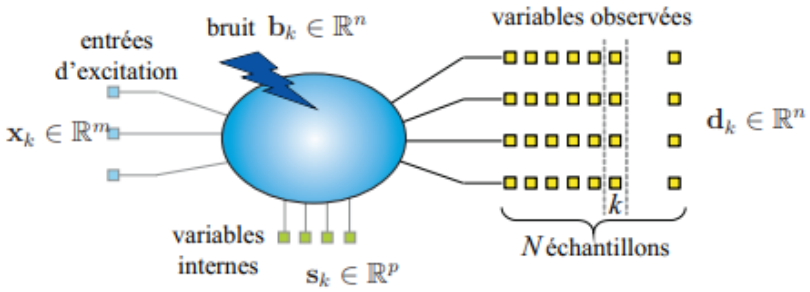


Figure 2.10: learning diagram to reproduce the behavior of a process

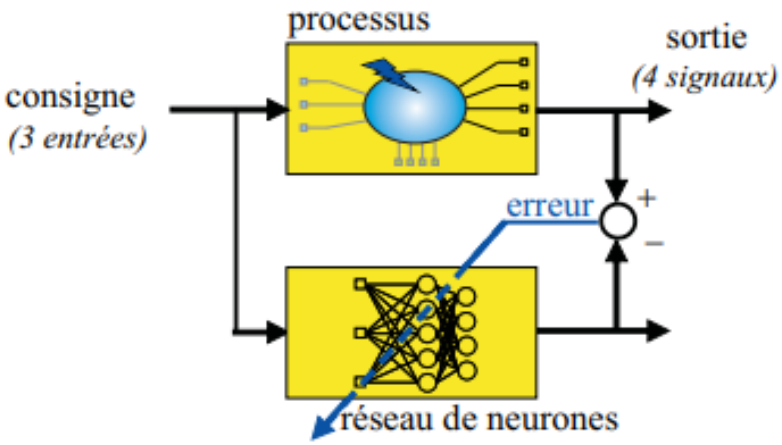


Figure 2.11: learning diagram to reproduce the behavior of a process

Learning a process generally involves supervised learning, less for a system defined by inputs and outputs.

This principle is illustrated by the Figure 2.11. The system then imposes its dimensions on the neural network that is used to reproduce his behavior. After learning, for a vector of inputs to a given time, the neural network delivers an output vector.[1]

as close as possible to the system one. The error between its own output and the system's is used to adjust its weights so the estimate will be even better at the next iteration. To

explicitly take into account the dynamic nature of a system with a Connectionist model, several techniques can be used.

A first method relies on the use of a recurrent neural network or any other network able to capture, mimic and reproduce the dynamic effect of the system. Another solution is to provide signals to the neural network representative of the dynamic nature of the system. This is usually done using delay lines; their number depends on the dynamics.[1]

2.9 Conclusion

This chapter allowed us to outline some basic notions of nonlinear control and neural networks, then had briefly summarized the general aspect of Identification of nonlinear systems by neural networks

The objective of the next chapter will be working on the design of control laws of a flexible manipulator arm using neural networks.

Chapter 3

Control Of SISO System Using Feedback Linearization Augmented By NN

3.1 Introduction

We consider single-input/ single-output nonlinear systems, whose output has known and full relative degree. Systems with both parameter uncertainty and unmodeled dynamics can be included as well.

The aim of this chapter is based on the ordinary Feedback linearisation assocoiated with NN augmentation will be applied for a class of nonlinear systems with uncertainty

3.2 Problem formulation

The system below is an observable and stabilizable nonlinear single-input-single output (SISO) system :

$$\dot{x} = f(x, u) \tag{3.1}$$

$$y = g(x) \tag{3.2}$$

where $x \in \Omega \subset R^n$ is the state of system, u, y $\in$ R are the system input (control) and output (measurement) signals,respectively.
f and g are smooth partially known functions, and n is not necessarily known.

3.3 Controller design and tracking error dynamics

3.3.1 Feedback Linearization and Model Inversion Error

Consider the non-affine system with well defined relative degree in the normal form

$$\dot{\mathcal{X}} = f_0(\xi, \mathcal{X})$$

$$\dot{\xi}_i = \xi_{i+1} \quad i = 1, ..., r-1$$

$$\dot{\xi}_r = h(\xi, \mathcal{X}, u)$$

$$\xi_1 = y \tag{3.3}$$

where $h(\xi, \mathcal{X}, u) = L^{(r}{}_f g, \xi = [\xi_1 \ \dots \ \xi_r]^T$, and $\mathcal{X}$ are the states associated with the internal dynamics, and f_0 is globally Lipshitz continuous in its arguments.
A linearizing feedback control law is approximated by introducing the following signal:

$$u = \hat{h}^{-1}(y, v) \tag{3.4}$$

where v is defined as:

$$v = \hat{h}(y, u) \tag{3.5}$$

and it is commonly referred to as pseudocontrol.The function $\hat{h}(y, u)$ represents any available approximation of $h(\xi, \chi, u)$ that is invertible with respect to its second argument.

It may be constructed from approximate linear models. For now, it is enough to assume its invertibility With this definition of pseudocontrol the output dynamics can be expressed as

$$y^{(r)} = v + \Delta \tag{3.6}$$

where

$$\begin{aligned} \Delta(x,u) &= \Delta(\xi,\chi,u) \\ &= h(\xi,\chi,\hat{h}^{-1}(y,v)) - \hat{h}(y,\hat{h}^{-1}(y,v)) \end{aligned} \tag{3.7}$$

is the difference between the possibly unknown function $h(\xi,\chi,u)$ and its approximation $\hat{h}(y,u)$, usually referred to as modeling error.[10]

The pseudocontrol is chosen to have the form $v = y_c^{(r)} + v_{dc} - v_{ad}$ where $y_c^{(r)}$ is the r^{th} derivative of the input signal, generated using an r^{th} (or higher) order stable reference model forced by an external input v_{dc} is the output of a linear output feedback dynamic compensator, and v_{ad} is the adaptive control signal designed to cancel Δ.[10]

the output dynamics is reduced to

$$y^{(r)} = y(r)_c + v_{dc} - v_{ad} + \Delta. \tag{3.8}$$

From 3.7, notice that Δ depends on v_{ad} through v, whereas v_{ad} has to be designed to cancel Δ. Therefore, the following assumption is introduced to guarantee the existence and uniqueness of a solution for v_{ad}.

Assumption: The mapping v_{ad} is a contraction over the entire input domain of interest. A contraction is defined by the following condition:

$$\left|\frac{\partial\Delta}{\partial v_{ad}}\right| < 1 \tag{3.9}$$

Using 3.7 implies

$$\left|\frac{\partial\Delta}{\partial v_{ad}}\right| = \frac{\partial h - \hat{h}}{\partial u}\frac{\partial u}{\partial v}\frac{\partial v}{\partial v_{ad}} = \frac{\partial(h-\hat{h})}{\partial u}(\frac{\partial\hat{h}}{\partial u})^{-1} < 1$$

which can be rewritten in the following way:

$$\left|\frac{\frac{\partial h}{\partial u}}{\frac{\partial\hat{h}}{\partial u}} - 1\right| < 1 \tag{3.10}$$

Condition 3.10 is equivalent

$$sgn\left(\frac{\partial h}{\partial u}\right) = sgn\left(\frac{\partial\hat{h}}{\partial u}\right) \tag{3.11}$$

$$\left|\frac{\partial h}{\partial u}\right| > \frac{|\frac{\partial h}{\partial u}|}{2} > 0. \tag{3.12}$$

The first condition states that unmodeled control reversal is not permissible. The second condition places a lower bound on our estimate of the control effectiveness in 3.4. [10]

3.4 Dynamic Compensator and Error Dynamics design

Define the output tracking error

$$\tilde{y} = y_c - y. \tag{3.13}$$

With this definition, the dynamics in 3.10 can be rewritten

$$\tilde{y}^{(r)} = -v_{dc} + v_{ad} - \Delta. \tag{3.14}$$

For the case $\delta = 0$, the adaptive term v_{ad} in 3.9 is not required, and the error dynamics in 3.14 reduce to

$$\tilde{y}^{(r)} = -v_{dc}. \tag{3.15}$$

The following linear compensator is introduced to stabilize the dynamics in 3.15:

$$\dot{\eta} = A_c\eta + b_c\tilde{y},\ v_{dc} = c_c\eta + d_c\tilde{y}\ \eta \in R^{r-1}. \tag{3.16}$$

Note that η needs to be at least of dimension (r-1). This follows from the fact that 3.15 corresponds to error dynamics that have r poles at the origin. One could elect to design a compensator of dimension $\geq$ r as well. In the future, we will assume that the minimum dimension is chosen. Returning to 3.14, notice that the vector $e = [\ \tilde{y}\ \ \dot{\tilde{y}}\ ...\ \tilde{y}(r-1)\ ]^T$ together with the compensator state η will obey the following dynamics, hereafter (with a slight abuse of language) referred to as tracking error dynamics:

$$\begin{bmatrix}\dot{e}\\ \dot{\eta}\end{bmatrix} = \begin{bmatrix}A - d_cbc & -bc_c\\ bc_c & A_c\end{bmatrix}\begin{bmatrix}e\\ \eta\end{bmatrix} + \begin{bmatrix}b\\ 0\end{bmatrix}\begin{bmatrix}v_{ad} - \Delta\end{bmatrix} \tag{3.17}$$

$$z \triangleq [\tilde{y}\ \eta^T\ ]^T \tag{3.18}$$

where

$$A = \begin{bmatrix}0 & 1 & 0 & \dots & 0\\ 0 & 0 & 1 & & 0\\ \vdots & \vdots & & \ddots & \\ 0 & 0 & & & 1\\ 0 & 0 & 0 & \dots & 0\end{bmatrix},\ b = \begin{bmatrix}0\\ 0\\ \vdots\\ 0\\ 1\end{bmatrix},\ c = \begin{bmatrix}1\\ 0\\ \dots\\ 0\\ 0\end{bmatrix}^T,$$

and z is the vector of available measurements. For ease of notation, define the following matrices:

$$\bar{A} = \begin{bmatrix}A - d_cbc & -bc_c\\ bc_c & A_c\end{bmatrix}, \bar{b} = \begin{bmatrix}b\\ 0\end{bmatrix}, \bar{C} = \begin{bmatrix}c & 0\\ 0 & I\end{bmatrix} \tag{3.19}$$

and a new vector

$$E \triangleq \begin{bmatrix}e\\ \eta\end{bmatrix} \tag{3.20}$$

With these definitions, the error dynamics in 3.18 can be rewritten

$$\dot{E} = \bar{A}E + \bar{b}\ [v_{ad} - \Delta\ ]$$

$$z = \bar{C}E \tag{3.21}$$

Note that A_c, b_c, C_c and d_c in 4.18 should be designed such that $\bar{A}$ is Hurwitz. Naturally, there might be multiple approaches for doing this in the most general case.[10]

3.5 Design and analysis of an observer for the error dynamics

consider the following linear observer for the tracking error dynamics in 3.21:

$$\dot{\hat{E}} = \bar{A}\hat{E} + K(z - \hat{z}) \tag{3.22}$$

$$\hat{z} = \bar{C}\hat{E} \tag{3.23}$$

where K is a gain matrix and should be chosen such that $\bar{A} - K\bar{C}$ is asymptotically stable, and z is defined in 3.21.
Let

$$\tilde{A} \triangleq \bar{A} - K\bar{C}, \tilde{E} = \hat{E} - E. \tag{3.24}$$

then, the observer error dynamics can be written

$$\dot{\tilde{E}} = \tilde{A}\tilde{E} - \bar{b}\,[v_{ad} - \Delta\,]. \tag{3.25}$$

3.6 Single hidden layer neural network approximation of the inversion error

An SHL NN has an output given by

$$y_i = \sum_{j=1}^{N_2} [m_{ij}\sigma(\sum_{K=1}^{N_1} n_{jk}x_k + \theta_{nj}) + \theta_{mi}\,] \tag{3.26}$$

This implies that a general function f(x) $\in \mathcal{C}, x \in \mathcal{D} \subset \mathcal{R}^n$ can be written as

$$f(x) = M^T\sigma(N^Tx) + \epsilon(x) \tag{3.27}$$

where $\epsilon(x)$ is the function reconstruction error. In general, given a constant real number $\epsilon^* > 0, f(x)$ is within ϵ^* range of the NN 3.26, if there exist constant weights M and N , such that for all x $\in \mathcal{D} \in \mathcal{R}^n$, the representation in 3.27 holds with $\| \epsilon \| < \epsilon^*$. The following theorem extends these results to map the unknown dynamics of an observable system from available input–output history. [10]
Theorem 1: Given $\epsilon^* > 0$, there exists a set of bounded weights M and N , such that $\Delta(x, u)$ in 3.7 can be approximated over a compact set $\mathcal{D} \subset \Omega x\mathcal{R}$ by an SHL NN

$$\Delta(x, u) = M^T\sigma(N^T\mu) + \epsilon(d, \mu), |\epsilon| < \epsilon^* \tag{3.28}$$

using the input vector

$$\mu(t) = [1\ \bar{v}_d^T(t)\ \bar{y}_d^T(t)]^T,\ \| \mu \| \leq \mu^*, \mu^* > 0 \tag{3.29}$$

where

$$\bar{v}_d^T(t) = [v(t)\ v(t-d)\ \ldots\ v(t-(n_1-r-1)d]^T \tag{3.30}$$

$$\bar{y}_d^T(t) = [y(t)\ y(t-d)\ \ldots\ y(t-(n_1-r-1)d]^T \tag{3.31}$$

with $n_1 \geq n, d > 0$ denoting time-delay and μ^* being a uniform bound for all $(x, u) \in \mathcal{D}$. In [5], explicit upper bounds and rate of convergence for this approximation are derivedIt has been shown that when $d \to 0$, then the error bound of such an approximation converges to Barron's original result. [4]
Remark 1: To be consistent with 3.7, the input vector 3.29 to the NN must be defined using the actual control signal and not the pseudocontrol. The use of $v(t)$ in place of $u(t)$ is justified by the relationship in 3.4, which represents a static map between these two signals. The first component in 3.29 is introduced to approximate a nonzero offset of Δ.
Remark 2:The input–output history of the original nonlinear system is needed to map in systems with zero dynamics, because for such systems the unobservable subspace is not estimated by 3.24. If the system has full relative degree, the observer in 3.24 provides all the estimates needed for the reconstruction of Δ , and no input–output history is required. [10]

3.6.1 Adaptive Control

The adaptive signal is chosen to be the output of an SHL NN

$$v_{ad} \triangleq \hat{M}^T \sigma(\hat{N}^T \mu) \tag{3.32}$$

where $\hat{M}$ and $\hat{N}$ are estimates of and that are updated according to the following adaptation laws:

$$\dot{\hat{N}} = -G\big[2\mu \hat{E}^T\ P\bar{b}\hat{M}^T\hat{\sigma}' + k(\hat{N} - N_0)\big] \tag{3.33}$$

$$\dot{\hat{M}} = -F\big[2(\hat{\sigma} - \hat{\sigma}'\hat{N}^T\mu)\hat{E}^T P\bar{b} + k(\hat{M} - M_0)\big] \tag{3.34}$$

in which M_0 and N_0 are initial values (or guess) of the NN weights, $\hat{\sigma} = \sigma(\hat{N}\mu)$, denotes the Jacobian matrix, P is the solution of the Lyapunov equation

$$\bar{A}^T P + P\bar{A} = -Q \tag{3.35}$$

Using 3.28 and 3.32, the error dynamics in 3.23 can be expressed as

$$\dot{E} = \bar{A}E + \bar{b}\big[\hat{M}^T\sigma(\hat{N}^T\mu) - M^T\sigma(N^T\mu) - \epsilon\big] \tag{3.36}$$

$$z = \bar{C}E \tag{3.37}$$

Define

$$\tilde{M} = \hat{M} - M, \tilde{N} = \hat{N} - N, \tilde{Z} = \begin{bmatrix} \tilde{M} & 0 \\ 0 & \tilde{N} \end{bmatrix} \tag{3.38}$$

and note that

$$\| \hat{M} \| < \| \tilde{M} \| + M^*, \| \hat{N} \|_F < \| \tilde{\|}_F + N^* \tag{3.39}$$

where M^* and N^* are the upper bounds for the weights in 3.28

$$\| M \| < M^*,\ \| N \|_F < N^* \tag{3.40}$$

the subscript denoting the Frobenius norm. With 3.39, the representation

$$v_{ad} - \Delta = \hat{M}^T \sigma(\hat{N}^T \mu) - M^T \sigma(N^T \mu) - \epsilon \tag{3.41}$$

We are about to illustrate the previous study by presenting a second order system with internal states. using neural networks to solve the problem of uncertainties.

3.7 Van der Pol Example

The Van Der Pol example is probably the best choice to apply our control technique on, in order to prove that the approach is applicable on systems with both parametric uncertainty and unmodeled dynamics.

we first consider the following nonlinear system

$\dot{x_1} = x_2$
$\dot{x_2} = x_1 - 0.2(x_1^2 - 1)x_2 - 0.2x_3 + \frac{3u}{\sqrt{|u|+0.1}}$
$\dot{x_3} = x_4$
$\dot{x_4} = -0.2x_4 - x_3 + x_1$
$y = x_1 + 0.2x_3$

the initial conditions are: $x_1(0) = 0.5$, $x_2(0) = 1.5$, and $x_3(0) = 0$. The output y has a relative degree of r = 2.

From a practical perspective, the system can be thought of as a second-order nonlinear system model, whose realization consists of states x_1 and x_2, in which the output is modeled as $y = x_1$.

However, the system also contains a very lightly damped unmodeled mode, with a natural frequency equal to that of the linearized system. This mode is excited by the system displacement state x_1 and coupled to the measurement. The output y does not have full relative degree in the presence of the unmodeled mode.

The low natural frequency of this mode is encompassed by the bandwidth of the controlled system. We treat this example as if even the nonlinear portion of the model is unknown to the designer, and only the fact that r = 2 is given, i.e., approximate feedback linearization in 3.6 is performed using $\hat{h} = u$. The following dynamic compensator:

$$\dot{\eta} = 5\eta + \tilde{y}$$
$$v_{dc} = -35\eta + 8.2\tilde{y}\eta \in R$$

places the poles of the closed-loop error dynamics in 3.18 at -3, -1$\pm$ j. The observer dynamics in 3.22 were designed so that its poles are four times faster than those of the error dynamics. We implemented ten hidden neurons and the following sigmoidal basis function

$$\dot{\sigma}(x) = \frac{1}{1 + e^{-ax}} \tag{3.42}$$

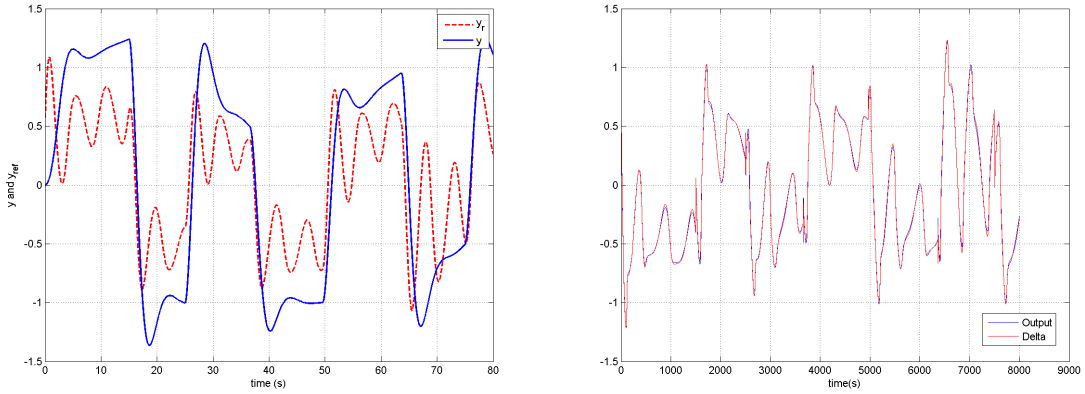
A second-order command filter was implemented so that

$$y_c = \frac{w_n^2}{s^2 + 2\zeta w_{n^2} + w_n^2} y_{ref} \tag{3.43}$$

with w_n = 1 and ζ = 0.5, for y_{ref} defined to be a square wave of amplitude 1

3.7.1 Simulation results

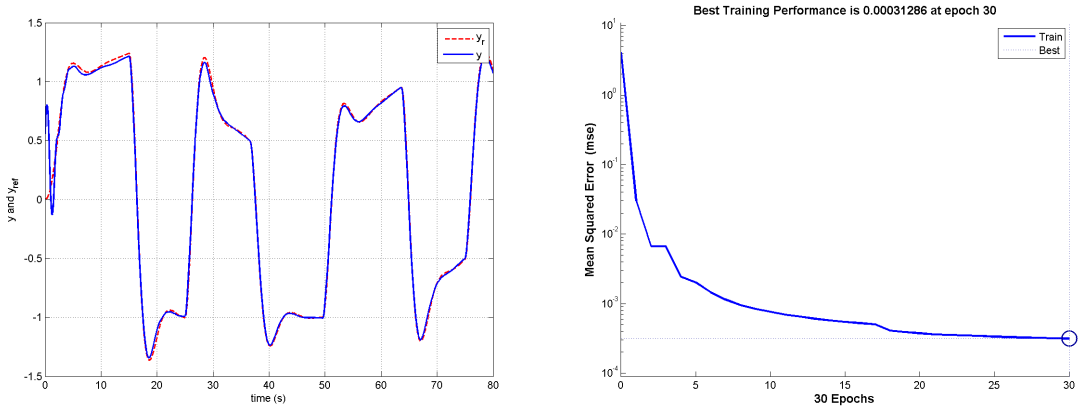
The results below presents the tracking errors before and after the NN augmentation.



(a) Tracking performance. (b) Control effort.

Figure 3.1: Simulation without NN augmentation.

In Figure 3.1, we show the tracking performance and the control efforts without NN augmentation,



(a) Tracking performance. (b) Training performance with SHL NN.

Figure 3.2: Simulation with NN augmentation.

while in Figure 3.2, we show it with NN augmentation.

3.8 Conclusion

This chapter was illustrated by a technique for controlling nonlinear systems augmented by neural networks.

The next chapter will be build up on the idea of extending the previous concept, otherwise to design adaptive neural network that estimates unknown functions and force the robot to track smooth reference trajectory

Chapter 4

Adaptive Nonlinear Control For Flexible Manipulator Using NN

4.1 Introduction

In the previous chapters, we addressed modeling of all kind of robotic systems, we have also discussed the problem of controlling uncertain nonlinear systems by feedback linearization with the augmentation by neural networks.

After having seen those technical descriptions and different concepts, in order to validate the theoretical study of the command, we carried out simulation analyzes on Matlab.

The control of uncertain systems is generally accomplished using either an adaptive control or a philosophy of robust control. The basic idea of the adaptive approach is to design a command that tries to know the uncertain parameters of a system, and, if it is properly designed, it will subsequently be a better command for uncertain system. In general, the adaptive approach is applicable to the largest range of uncertainties

In what follows we are interested with the benchmark robot trying to include an adaptive neural network control to eliminate uncertainties and disturbances

4.2 Problem Formulation

Consider the two flexible link manipulator. According to what we have seen in the first chapter, the dynamic model of multi-flexible link robotic manipulator can be described as

$$u = M(q)(\ddot{q}) + C(q,\dot{q}) + G(q) + D\dot{q} + \tau_s(q) + f(\dot{q}) \tag{4.1}$$

To simplify the study we we suppose $f(\dot{q}) = 0$, then we divide the equation below into two subsystems

$$M_m(q)(\ddot{q}) + C_m(q,\dot{q}) + G_m(q) + D_m\dot{q} + \tau_{sm}(q) = 0 \tag{4.2}$$

$$M_j + D_j(\dot{q}) + \tau_{sj}(q) = u \tag{4.3}$$

where

$$\begin{bmatrix} j_{11}(q) & j_{12}(q) \\ j_{12}(q) & j_{22}(q) \end{bmatrix} \begin{bmatrix} \ddot{q}_{a1} \\ \ddot{q}_{a2} \end{bmatrix} + \begin{bmatrix} -m_2 l_1 \xi_2 g \cos q_{a2}(2\dot{q}_{a1}\dot{q}_{a2} + (q_{a2})^2 \\ -m_2 l_1 \xi_2 g \cos q_{a2}(\dot{q}_{a2})^2 \end{bmatrix}$$
$$+ \begin{bmatrix} -g(m_1\xi_1(\sin q_{a1}) + m_2(l_1 \sin q_{a1} + \xi_2 \cos q_{a1} + q_{a2} \\ -m_2\xi_2 g \cos q_{a1} + q_{a2} \end{bmatrix}$$
$$+ \begin{bmatrix} d_1(\dot{q}_{a1} - \dot{q}_{m1}/n_1) \\ d_2(\dot{q}_{a2} - \dot{q}_{m2}/n_2) \end{bmatrix} + \begin{bmatrix} (K_1(q_{a1} - q_{m1}/n_1) \\ K_2(q_{a2} - q_{m2}/n_2) \end{bmatrix} = 0$$

$$\begin{bmatrix} jm_1 & 0 \\ 0 & jm_2 \end{bmatrix} \begin{bmatrix} \ddot{q}_{a1} \\ \ddot{q}_{a2} \end{bmatrix} \begin{bmatrix} d_1/n_1(-\dot{q}_{a1} + \dot{q}_{m1}/n_1) \\ d_2/n_2(-\dot{q}_{a2} + \dot{q}_{m2}/n_2) \end{bmatrix} + \begin{bmatrix} K_1/n1(q_{m1}/n_1 - q_{a1}) \\ K_2/n2(q_{m2}/n_2 - q_{a2}) \end{bmatrix} = \begin{bmatrix} u_{m1} + u_{d1} \\ u_{m2} + u_{d2} \end{bmatrix} \tag{4.4}$$

Using the state vector :

$$X = \begin{bmatrix} q_{a1} \\ q_{a2} \\ q_{m1} \\ q_{m2} \\ \dot{q}_{a1} \\ \dot{q}_{a2} \\ \dot{q}_{m1} \\ \dot{q}_{m2} \end{bmatrix} \tag{4.5}$$

$$\dot{X} = \begin{bmatrix} \dot{q}_{a1} \\ \dot{q}_{a2} \\ \dot{q}_{m1} \\ \dot{q}_{m2} \\ \ddot{q}_{a1} \\ \ddot{q}_{a2} \\ \ddot{q}_{m1} \\ \ddot{q}_{m2} \end{bmatrix} \tag{4.6}$$

$$\dot{X} = \begin{bmatrix} x_5 \\ x_6 \\ x_7 \\ x_8 \\ \begin{bmatrix} j_{11}(q) & j_{12}(q) \\ j_{12}(q) & j_{22}(q) \end{bmatrix}^{-1} \Big(- C_a - G_a - \begin{bmatrix} d_1(\dot{q}_{a1} - \frac{\dot{q}_{m1}}{n_1}) \\ d_2(\dot{q}_{a2} - \frac{\dot{q}_{m2}}{n_2}) \end{bmatrix} - \begin{bmatrix} K_1(q_{a1} - q_{m1}/n_1) \\ K_2(q_{a2} - q_{m2}/n_2 \end{bmatrix} \Big) \\ \begin{bmatrix} j_{m1} & 0 \\ 0 & j_{m2} \end{bmatrix}^{-1} (u_m + \begin{bmatrix} (d_1/n_1(\dot{q}_{a1} - \frac{\dot{q_{m1}}}{n_1} \\ d_2/n_2(\dot{q}_{a2} - \frac{q_{m2}}{n_2}) \end{bmatrix} - \begin{bmatrix} K_1/n_1(q_{a1} - q_{m1}/n_1) \\ K_2/n_2(q_{a2} - q_{m2}/n_2) \end{bmatrix} + \begin{bmatrix} f_{fric1}(x) \\ f_{fric2}(x) \end{bmatrix} \end{bmatrix} \tag{4.7}$$

The outputs of the system are chosen to be the measured angular position of the motors $\begin{pmatrix} q_{m1} \\ q_{m2} \end{pmatrix}$ Normally at steady state q_{mi}/n_i tends to q_{ai}

$$\Delta(x) = \begin{bmatrix} j_{m1} & 0 \\ 0 & j_{m2}^{-1} \end{bmatrix}^{-1} \Big(\begin{bmatrix} d_1/n_1(\dot{q}_{a1} - \dot{q}_{m1}/n_1) \\ d_2/n_2(\dot{q}_{a2} - \dot{q}_{m2}/n_2) \end{bmatrix} - \begin{bmatrix} (K_1/n_1(q_{a1} - q_{m1}/n1) \\ K_2/n_2(q_{a2} - q_{m2}/n_2) \end{bmatrix} + \begin{bmatrix} f_{fric1}(x) \\ f_{fric2}(x) \end{bmatrix} \Big) \tag{4.8}$$

4.2.1 Control Design

The control objective is that the tool position (the end effector position) follows a given reference trajectory that can be reached by a given reference $q_{mi}^{ref} = n_i q_{ai}^{ref}$ The error vector is

$$e = y - y^{ref} = q_m - q_m^{ref} = \begin{bmatrix} (x_3 - q_{m1})^{ref} \\ (x_4 - q_{m2})^{ref} \end{bmatrix} \tag{4.9}$$

The output feedback linearization suggests the derivation of the output signal until the appearance of the control signal:

$$\dot{e} = \dot{y} - \dot{y}^{ref} = \dot{q}_m - \dot{q}^{ref} = \begin{bmatrix} x_7 - \dot{q}_{m1}^{ref} \\ x_8 - \dot{q}_{m2}^{ref} \end{bmatrix} \tag{4.10}$$

$$\ddot{e} = \ddot{y} - \ddot{y}^{ref} = \ddot{q}_m - \ddot{q}_m^{ref} = \begin{bmatrix} (x_7 \\ x_8) \end{bmatrix} = \begin{bmatrix} [j_{m1} & 0 \\ 0 & j_{m2}] \end{bmatrix}^{-1} u_m + \Delta(x) - \ddot{q}_m^{ref} \tag{4.11}$$

The control signal appears so the pseudo control signal is chosen as

$$v = \begin{bmatrix} j_{m1} & 0 \\ 0 & j_{m2} \end{bmatrix}^{-1} u_m \tag{4.12}$$

$$\ddot{e} = v + \Delta(x) - \ddot{q}_m^{ref} \tag{4.13}$$

The Pseudo Control Signal is chosen as

$$v = v_l - v_{ad} + \ddot{q}_m^{ref} \tag{4.14}$$

so v is called the pseudo control v_{ad} is an adaptive control neural network signal designed to cancel the effect of all neglected unmodeled dynamics and external disturbances represented by $\Delta(x)$

It means that the dynamics of the system's output has become linear, the only step that is left here is to calculate the pseudo control so that these dynamics is fully stabilized. This method is well known as the output feedback linearization technique.[5]

4.3 Linear Control Signal

The Linear Control signal is either the output of a PID/PD controller from classical control or equivalently a dynamic compensator designed to achieve acceptable performance for the nominal robot system.

If we used a PID Controller

$$V_l = K_i \int (y - y^{ref}) + K_p(y - y^{ref}) + K_d(\dot{y} - \dot{y}^{ref}) \tag{4.15}$$

it can be represented using the error vector

$$e_1 = \int (y - y^{ref}) e_2 = y - y^{ref} e_3 = \dot{y} - \dot{y}^{ref} \tag{4.16}$$

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} K_i & K_p & k_d \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (\Delta(x) - V_{ad}) \tag{4.17}$$

$$\dot{E} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ K_i & K_p & K_d \end{bmatrix} E + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (\Delta(x) - V_{ad}) \tag{4.18}$$

So if the adaptive control signal v_{ad} is well designed to cancel $\Delta(x)$ Then the closed loop error dynamics can be stabilized and converge by the right choice of the PID gains :

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ K_i & K_p & K_d \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} \tag{4.19}$$

According to Lyapunov indirect method for A_{cl} Hurwitz we can find a Symetric Positive definite Matrix P such that the Lyapunov equation is satisfied for a given Positive definite Matrix Q

$$A_{cl}^T P + P A_{cl} = -Q \tag{4.20}$$

Where the matrices P, Q have the same structure as A, given as P = blockdiag $[P_1 \dots P_n]$ and the matrix Q = blockdiag $[Q_1 \dots Q_n]$

4.4 The Neural Network uncertain term Approximator

The essential part of this control method is the uncertainty approximation. There exists a set of fixed bounded weights W and V , such that the uncertainty term can be approximated over a compact set by a Single Hidden Layer Neural Network as follows:

$$\Delta(x) = W\Phi(V\mu) + e \tag{4.21}$$

In the above expressions W and V indicate constant parameter, these values are not necessarily unique.

Using the input vector composed of the delayed input

$$u_d^T = \begin{bmatrix} u(t) & u(t-d) & \dots & u(t-n_1 d) \end{bmatrix} \tag{4.22}$$

and the delayed measured vector a subset of the state vector

$$z_d^T = \begin{bmatrix} z(t) & z(t-d) & \dots & z(t-n_2 d) \end{bmatrix} \tag{4.23}$$

$$\mu = \begin{bmatrix} 1 & u_d^T & z_d^T \end{bmatrix}^T \tag{4.24}$$

where $n_1 \geq 1$ and $n_2 \geq 1$, $d > 0$ denoting time-delay. The adaptive signal is chosen to be the output of an SHL NN

$$v_{ad} = \hat{W}\Phi(\hat{V}\mu) \tag{4.25}$$

where $\hat{W}$ and $\hat{V}$ are the estimates of W and V that are updated according to the following adaptation laws:

$$\dot{\hat{V}} = -G(2\mu E^T P b \hat{W}\hat{\phi} + k(\hat{V} - V_0)) \tag{4.26}$$

$$\dot{\hat{W}} = -F(2(\hat{\phi} - \hat{\phi}\hat{V}\mu)E^T P b + k(\hat{W} - W_0)) \tag{4.27}$$

In which V_0 and W_0 are initial values (or guess) of the NN weights, $\hat{\phi}'$ denotes the Jacobian matrix Using 4.18 and 4.21, the error dynamics can be expressed as

$$\dot{E} = A_{cl}E + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} (W\Phi(V\mu) + e - \hat{W}\phi(\hat{V}\mu)) \tag{4.28}$$

4.5 Application

We applied the MLP NN technique on a two link flexible robot. The dynamic model of this kind of robots is discussed widely in the first chapter.

We present the following specifications The PID gains are chosen as
$K_{p1} = 45$, $K_{d1} = 1.5$, $K_{i1} = 0.95$. $K_{p2} = 15$, $K_{d2} = 0.5$, $K_{i2} = 0.95$.

And the initial weight values are chosen as random values.

For a simplified study we fixed one of the fixed bounded weights witch is the V.

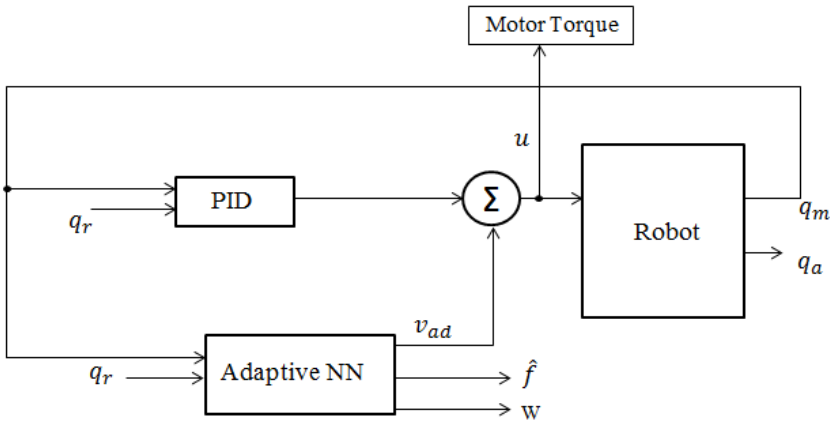


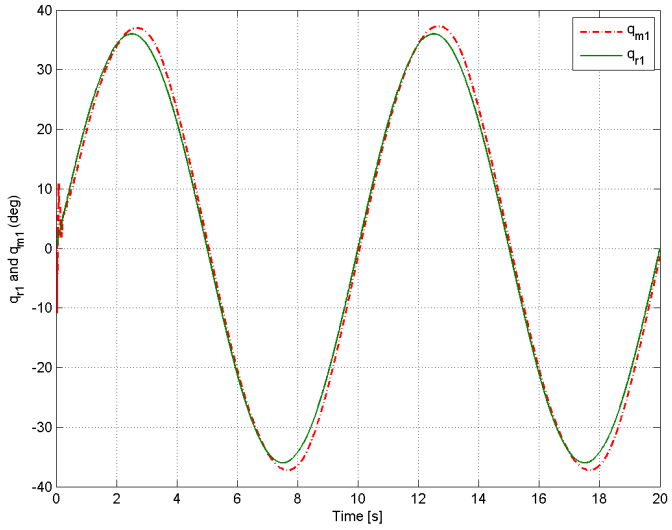
Figure 4.1: adaptive NN controller architecture.

The robot parameters are given in the appendix

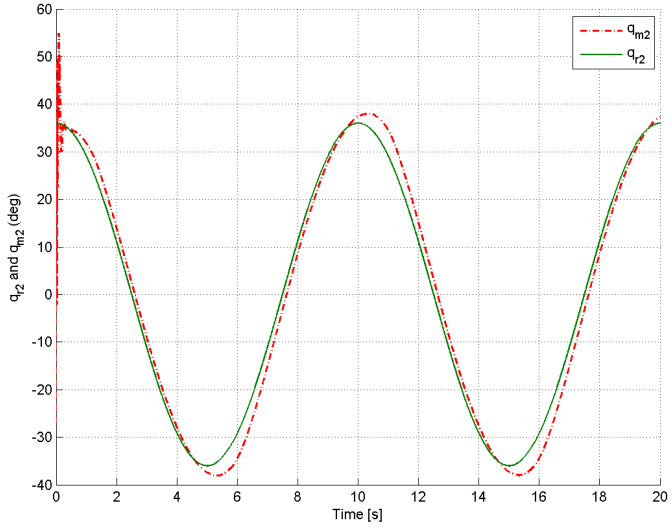
4.5.1 Simulation Results

In this study we have implemented the simulations using Matlab/Simulink for the benchmark robot and the objective is to make the robot joints follw a given sinusoidal reference trajectories.

The error tracking of the motor angular positions illustrated in Figure 4.2 Denotes a static error, which is due only to the boundedness of the error signals, not asymptotic stability of the controller.



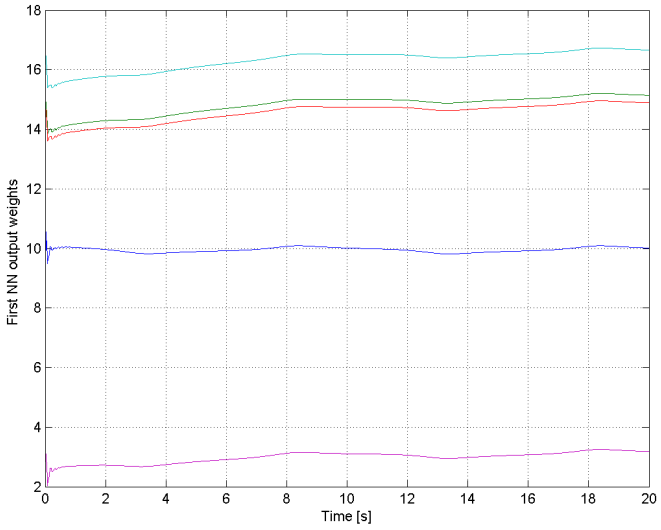
(a)



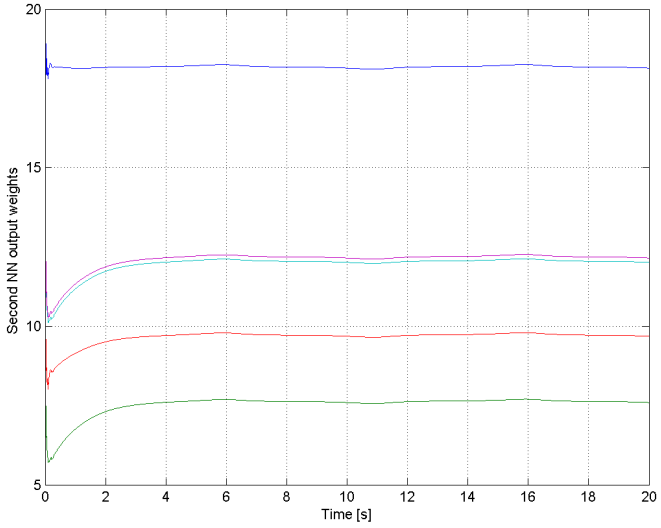
(b)

Figure 4.2: Tracking performance of the joint angles

The next two Figures 4.3a and 4.3b show respectively the time evolution of NN weights for the first output and the second one



(a) 1^{st} output



(b) 2^{nd} output

Figure 4.3: Weights history

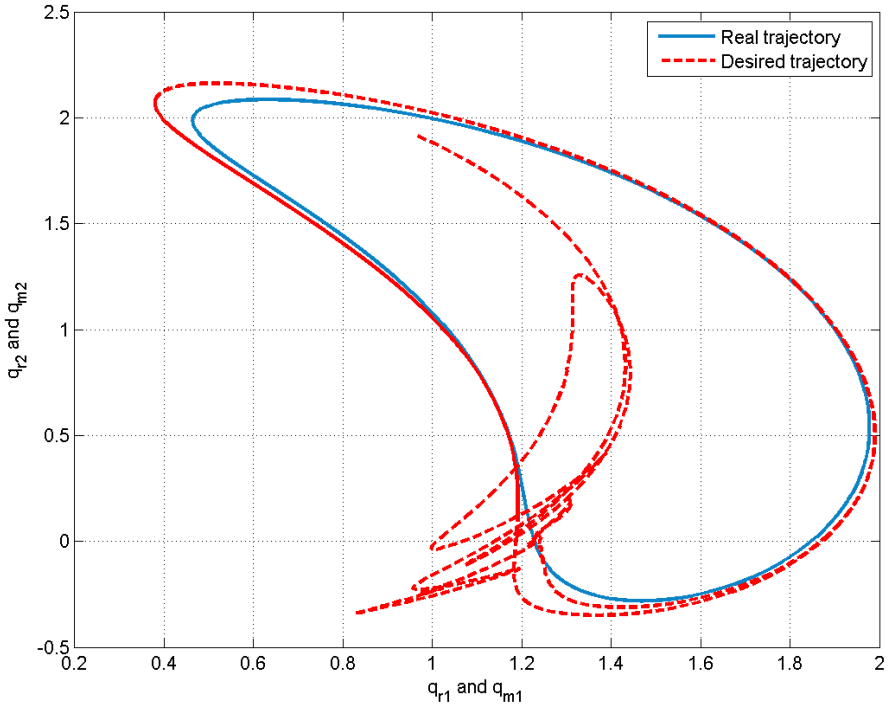
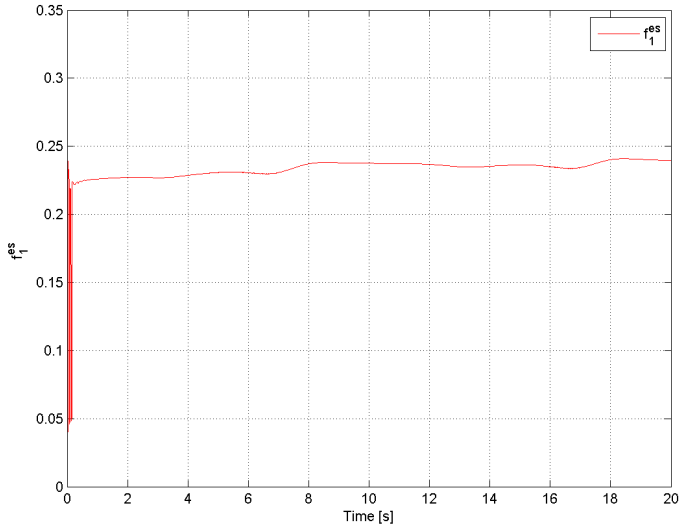


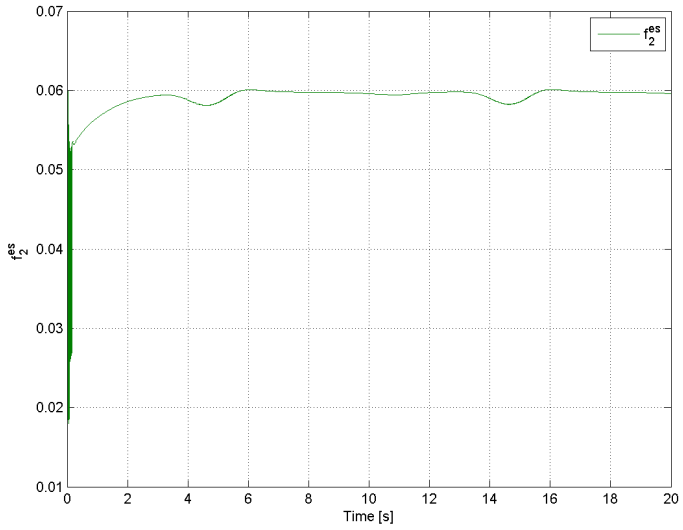
Figure 4.4: Trajecory tracking

The result of trajectory tracking presented in Figure 4.4 shows that the NN approximation performance is quite good in eliminating the vibrations, and parametric uncertainties on trajectory tracking, this result confirms that the proposed controller has accomplish the control objectives.

Finally, Figures 4.5a, and 4.5b present the two estimated functions by NN



(a)



(b)

Figure 4.5: NN estimated functions

4.6 Conclusion

In this chapter, the main contributions of the thesis are addressed, where adaptive output feedback control of uncertain nonlinear systems using NNs is proposed to control flexible link manipulator; Under the assumption that the system is feedback linearizable, an SHL NN is introduced to cancel the inversion error, the approach investigates trajectory tracking for the joint angles in the presence of link flexibilities.

A simple linear observer is introduced to estimate the derivatives of the tracking error. From what we have seen previously, we note that the MLP NN technique would be a suitable way to solve the problem of uncertainties and trajectory tracking .

CONCLUSION

In this thesis we present modern nonlinear adaptive control established on geometric and Lyapunov based control strategies and intelligent function approximators applied on flexible robot manipulators. The work dealt basically with improving classical nonlinear control techniques (feedback linearization) using intelligent neural networks to solve the problem of uncertainties and also reduce the complexity of controllers.

The goal here is to design a control techniques that improve the performance of flexible manipulators on the dynamic phase of the trajectory and reduce the computational burden.

The controller is elaborated by approximating the robot's dynamic equation of motion with an artificial neural network and adding a proportional, integral and derivative term. The aim of this controller is to provide a stable and fast control, based on the dynamic model of the system. The adaptive neural network strategy ensure that the real trajectory matches the desired one by compensating errors due to structured and unstructured uncertainties, increasing the precision of the control.

The main advantage of neural networks control techniques among others is that they use nonlinear regression algorithms that can model high dimensional systems with extreme flexibility due to their learning ability

Simulation study on an industrial manipulator involving an adaptive NNs controller were presented to verify the effectiveness of the proposed approach.

Firstly, we gave a brief presentation of the robot manipulators in general, after that we mentioned the benchmark problem adopted. As a second step a nonlinear techniques were introduced in order to be used in our study.

Before the main application we have illustrated our work by a Van Der Pol example in the purpose to prove that the approach is applicable to systems with both parametric uncertainty and unmodeled dynamics.

Finally, the developed methodology was extended to a MIMO system ; an MLP Neural Network Control was applied on a benchmark problem, Despite its complexities, it was possible to implement a model for intelligent control simulation under the Matlab environment. The simulation results show that the controller we developed is able to successfully converge to the stable state and track the desired references. At the end of this work, by all counts and with proven results our control strategy offered a great performance in terms of tracking positional accuracy. In the other hand we did our best to balance between theory and practice with the intention to increase the relevance of academic research to engineering practice

After all, this study remains a good experience in terms of learning and discovery on the pedagogical, scientific and personal level.

We hope that through this study we have been able to present first of all a useful document for the future use of the benchmark , with more illustrations. It is also hoped that the applications outlined in the last chapter and the problems arising from them will form a substrate for further investigation.

According to the problem that the tracking accuracy is not high enough in trajectory tracking control of robot manipulators, a model reference adaptive control scheme based

on neural networks technique could be adopted to achieve robot manipulator model identification and robot motion control In the onther hand despite of obtaining an efficient control structure, current effort does not include a stability analysis of the overall control scheme. Future work includes exploiting sliding mode control to improve stability and performance of the control strategy proposed.

Bibliography

[1] *ABB IRB 6600*. url: https://www.robots.com/robots/abb-irb-6600. (accessed: 21.03.2020).

[2] *Artificial Neural Networks*. url: http://freakysquare.com/artificial-neural-network-simplified. (accessed: 27.03.2020).

[3] *Artificial Neural Networks Technology*. url: http://www2.psych.utoronto.ca/users/reingold/courses/ai/cache/neural4.html. (accessed: 25.03.2020).

[4] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

[5] Anthony J Calise. *Neural Network Based Adaptive Control of Uncertain and Unknown Nonlinear Systems*. Tech. rep. GEORGIA INST OF TECH ATLANTA, 2004.

[6] SABS Chen and SA Billings. "Neural networks for nonlinear dynamic system modelling and identification". In: *International journal of control* 56.2 (1992), pp. 319–346.

[7] Habib Esfandiar and Saeed Daneshmand. "Complete dynamic modeling and approximate state space equations of the flexible link manipulator". In: *Journal of mechanical science and technology* 26.9 (2012), pp. 2845–2856.

[8] *FLEXIBLE LINK ROBOT*. url: http://www2.ece.ohio-state.edu/~passino/flr.html. (accessed: 21.03.2020).

[9] Anthony Green and Jurek Z Sasiadek. "Dynamics and trajectory tracking control of a two-link robot manipulator". In: *Journal of Vibration and Control* 10.10 (2004), pp. 1415–1440.

[10] Naira Hovakimyan et al. "Adaptive output feedback control of uncertain nonlinear systems using single-hidden-layer neural networks". In: *IEEE Transactions on Neural Networks* 13.6 (2002), pp. 1420–1431.

[11] Stig Moberg and Sven Hanssen. "On feedback linearization for robust tracking control of flexible joint robots". In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 12218–12223.

[12] Stig Moberg, Jonas Öhr, and Svante Gunnarsson. "A benchmark problem for robust control of a multivariable nonlinear flexible manipulator". In: *IFAC Proceedings Volumes* 41.2 (2008), pp. 1206–1211.

[13] Alireza Mohammadi et al. “Nonlinear disturbance observer design for robotic manipulators”. In: *Control Engineering Practice* 21.3 (2013), pp. 253–267.

[14] Kumpati S Narendra and Lingji Chen. “Identification and control of a nonlinear dynamical system/spl Sigma/based on its linearization/spl Sigma//sub L/. I”. In: *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*. Vol. 3. IEEE. 1998, pp. 2977–2982.

[15] Mohammed Saber et al. “Artificial neural networks, support vector machine and energy detection for spectrum sensing based on real signals”. In: *International Journal of Communication Networks and Information Security* 11.1 (2019), pp. 52–60.

[16] Alaa Shawky et al. “Modeling and nonlinear control of a flexible-link manipulator”. In: *Applied Mathematical Modelling* 37.23 (2013), pp. 9591–9602.

[17] Thomas Solatges. “Modélisation, conception et commande de robots manipulateurs flexibles. Application au lancement et à la récupération de drones à voilure fixe depuis un navire faisant route”. PhD thesis. Toulouse, ISAE, 2018.

[18] Dominique Valentin et al. “Principal component and neural network analyses of face images: What can be generalized in gender classification?” In: *Journal of Mathematical Psychology* 41.4 (1997), pp. 398–413.

[19] Patrice Wira. “Approches neuromimétiques pour l’identification et la commande”. PhD thesis. 2009.

Appendices

Appendix A

Table A.1: Nominal and Uncertain Parameters

Parameter	Value	Unit	Uncertainty
j_{m2}	0.004	$kg.m^2$	
j_{m2}	0.001	$kg.m^2$	
j_1	5	$kg.m^2$	
j_2	50	$kg.m^2$	
m_1	50	kg	$\pm 10\%$
m_2	150	kg	$\pm 10\%$
l_1	1.0	m	
l_2	1.5	m	
ξ_1	0.5	m	
ξ_2	0.8	m	
η	200		
k_1^{high}	$0.5.10^6$	Nm/rad	$\pm 20\%$
k_1^{low}	1.5. 10^6	Nm/rad	$\pm 20\%$
ψ_1	2	$arcmin$	$\pm 20\%$
k_2^{high}	0.2 . 10^6	Nm/rad	$\pm 20\%$
k_2^{low}	0.6 . 106	Nm/rad	$\pm 20\%$
ψ_2	3	arcmin	$\pm 20\%$
d_1	600	Nm . s/rad	$\pm 20\%$
d_2	200	Nm . s/rad	$\pm 20\%$
f_{v1}	0.006	Nm . s/rad	$\pm 80\%$
f_{c1}	1.5	Nm	$\pm 80\%$
f_{v2}	0.003	Nm · s/rad	$\pm 80\%$
f_{v2}	1.0	Nm	$\pm 80\%$
μ	0.5		$\pm 50\%$
β	0.4		$\pm 50\%$
α	5		
g	9.81	m/s^2	
γ	1		$\pm 10\%$
P_N	3.10^{-12}		
F	500	N	$\pm 10\%$
ϕ_F	π	rad	$\pm\pi$
t_1	10	s	
t_2	10.5	s	
A_{c1}	1	Nm	
A_{c2}	-1	Nm	
t_3	0.5	s	$\pm 0.5\%$
t_4	8	s	
f_{cs}	0 Or 15	Hz	random choice
f_{ce}	0 Or15	Hz	random choice
T_s	$0.25.10^{-3}$	s	
T_d	$0.25.10^{-3}$	s	
u_d	50	N.m	
u_{m1}^{max}	35	Nm	
u_{m2}^{max}	20	Nm	
k_i^1	30		
k_p^1	45		
k_d^1	1.5		
K_p^2	15		
K_i^2	10		
K_d^2	0.5		
Z_p	0.95		

Appendix B

B.1 PID controller

The benchmark system,has a discrete-time diagonal PID controller with derivative filter described by the transfer function.

$$G_{pid}(Z) = \begin{bmatrix} g_{11}(z) & 0 \\ 0 & g_{22}(z) \end{bmatrix}$$

where

$$g_{ii}(z) = k_p^i + k_d^i \frac{z - i(1 - z_p^i)z}{T_s z(z - z_p^i)} + \frac{k_i^i T_s z}{1 - z}$$

The benchmark system consists of the manipulator model P described in chapter 1 and a feedback controller G as illustrated in Figure A.1

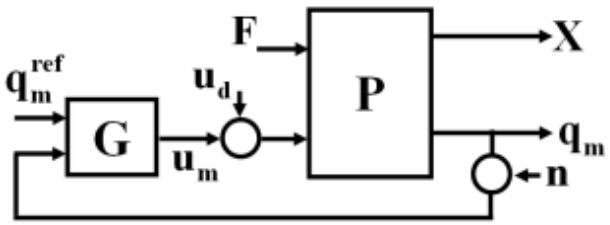


Figure B.1: A block diagram of the benchmark system.