INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

DIRECT ADAPTIVE CONTROL FOR UNDERACTUATED MECHATRONIC SYSTEMS USING FUZZY SYSTEMS AND NEURAL NETWORKS: A PENDUBOT CASE

Murad Musa Al-Shibli

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

September 2002

ABSTRACT

DIRECT ADAPTIVE CONTROL FOR UNDERACTUATED MECHATRONIC SYSTEMS USING FUZZY SYSTEMS AND NEURAL NETWORKS

A PENDUBOT CASE

Murad Musa Al-Shibli

This thesis describes the implementation of a vertical motion and position control scheme for a mechatronic system, specifically the Pendubot robot. The Pendubot is a non-linear, underactuated and unstable two-link planar robot arm that is frequently used as a benchmark in research studies involving nonlinear control theory and underactuated systems. Control of the Pendubot poses two challenging tasks: (i) to swing the two links from their stable hanging position to unstable vertical equilibrium positions, and (ii) to balance the links about the desired equilibrium positions. PD fuzzy controller is formulated and employed to meet challenges associated with swing-up control. Vertical balance control employs fuzzy systems and radial Gaussian neural networks. As such, an adaptive neural network and fuzzy controller is further analyzed, where the balance stability depends on a controller weight that is determined using Lyapunov theory. This approach is proven to be globally stable, with errors converging to a neighbourhood of zero. Then, the proposed swing-up and the balancing controllers are coupled together to achieve the motion objective in a stable manner, while resisting the external disturbances. The simulation results show that both the swing-up and balancing control schemes can be realized using 25 and 5 If-Then-rules, respectively. The simulation results confirm the results attained from the theoretical analysis.

ACKNOWLEDGEMENT

I dedicate this thesis to my supervisor Prof. Chun-Yi Su for his great guidance, knowledge and support that has made this work successful. I look forward to his worthy guidance in my future education and research.

I also would like to acknowledge all faculty members and staff of the Mechanical and Industrial Engineering Department at Concordia, not to forget the School of Graduate Studies especially the thesis office for their assistance and support. Special thanks to the Graduate Program Assistant: Charlene Wald for her distinguished kindness, help and support during my study.

To my lovely and great parents, my gentle brothers and sweet sisters and their families who supported, encouraged me at all times and sacrificed their entire life to teach me and let me succeed. The words will not help me enough to express thanks for every thing.

As the space does not allow me to mention all their names, I should not forget to dedicate this work to all my special and great friends for their recognized support and encouragement. All of them show a very great example of friendship and brotherhood. For me, they are like a golden treasure that can not be lost and forgotten easily.

I would like also to thank Mrs. Xiao Qing Ma and Mr. Zhen Cai for their valuable help during this research. Finally I dedicate this thesis to my best friend Ann.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

$D(q)$	Inertia matrix
$C(q,\dot{q})$	Coroilis / centripetal vector
$g(q)$	Gravity vector
d_{11}	Inertia matrix parameter
d_{12}	Inertia matrix parameter
d_{21}	Inertia matrix parameter
d_{22}	Inertia matrix parameter
h	Coroilis / centripetal vector parameter
ϕ_1	Gravity vector parameter
ϕ_2	Gravity vector parameter
m_1	Total mass of link one.
l_1	Length of link one
l_{c1}	Distance to the center of mass of link 1
I_1	Moment of inertia of link one about its centroid
m_2	Total mass of link two
l_2	Length of link two
l_{c2}	Distance to the center of mass of link 2
I_2	Moment of inertia of link one about its centroid

g	Acceleration of gravity
θ_1	Pendubot system dynamics parameter
θ_2	Pendubot system dynamics parameter
θ_3	Pendubot system dynamics parameter
θ_4	Pendubot system dynamics parameter
θ_5	Pendubot system dynamics parameter
$V(q)$	Potential energy due to gravity
E	Total energy
$W(q,\dot{q})$	Kinetic energy
t	Time
T	Limit time
$\bar{\tau}$	Constant torque
τ	Input torque
Co	Controllability matrix
A	Linearized state space matrix
B	Linearized input torque matrix
q_1	Link one angle
q_2	Link two angle
$\dot{q}_1$	Link one angular velocity
$\dot{q}_2$	Link two angular velocity
$\ddot{q}_1$	Link one angular acceleration
$\ddot{q}_2$	Link two angular acceleration

x_1	State-space variable for link one angle
x_2	State-space variable for link one angular velocity
x_3	State-space variable for link two angle
x_4	State-space variable for link two angular velocity
X	Fuzzy set universe of discourse
x	Fuzzy set variable
A	Fuzzy set
B	Fuzzy set
$\mu_A(x)$	Membership function for fuzzy set A
$\mu_B(x)$	Membership function for fuzzy set B
MF	Membership function
C	Fuzzy set
$\mu_C(x)$	Membership function for fuzzy set C
c	Gaussian membership center
σ	Gaussian membership function width
$T(x)$	Set of fuzzy set linguistic values or linguistic terms
u_i	Crisp input value
y_i	Crisp output value
U_i	Crisp input set
Y_i	Crisp output set
$\tilde{y}_i$	Fuzzy output linguistic variable
$\tilde{u}_i$	Fuzzy input linguistic variable

$\widetilde{A}_i^l$	Linguistic value of the fuzzy input linguistic variable
$\widetilde{B}_i^l$	Linguistic value of the fuzzy output linguistic variable
U_i^*	Set of all possible fuzzy sets defined on universe of discourse
F	Fuzzification operator
$\hat{A}_i^{fuz}$	Fuzzification fuzzy set
y_q^{crisp}	Fuzzification output membership function
X	Fuzzy input vector
$\widetilde{y}$	Fuzzy system output
$\widetilde{F}_h^a$	Linguistic value associated with linguistic variable
$\widetilde{x}_h$	Linguistic variable describes crisp input x_h
c_q	Output consequences for each q th fuzzy rule
μ_i	Membership function
$\widetilde{f}$	Fuzzy system output
$\theta_k(X)$	Fuzzy set of Lipschitz continuous functions
$g_q(X)$	Consequence of the q th rule
ζ_i	Area of the implied membership function associated with ith rule
z	Fuzzy vector associated with fuzzy variables
$f(x)$	Continuous function
$y(W,x)$	Approximation function
W	Weight vector
ε	Acceptable small error

MLP	Multi-layer percepton network
RBF	Radial basis function network
HONN	Higher order neural network
$a(z)$	Neural network activation function vectors
v_{ij}	First-to-second layer interconnection weight
w_{jk}	Second-to-third layer interconnection weights
z_i	Input to the activation function
y_i	Output of the neural network
W	Vector of the second-to-third layer interconnection weights
V	Vector of the first-to-second layer interconnection weight
I	Identity matrix
$R_i(x)$	Receptive unit output is denoted
$\bar{y}_i$	Strength (or a weight) radial basis neural network
M	Number of receptive units
R	Number of rules
b_i	Output membership function centers
f^*	Fuzzy logic approximation function
$d_\infty(f^*, f_2)$	Distance sub-metric function
$\tilde{z}$	Output of the first hidden layer produces a vector of functions
$b_{i,0}$	Basis for the i th node
y_m	Desired output trajectory

u_p	Plant input
$\dot{\mathrm{X}}$	State-space vector
y_p	Plant output
r	Relative degree of the plant
$\dot{\xi}_r$	State-space variable of plant degree r
$L_g^r h(\mathrm{X})$	Lie derivative of function of $h(\mathrm{X})$ with respect to g
$\alpha_k(t)$	Known components of the dynamics of the plant
$\beta_k(t)$	Known components of the dynamics of the control input of the plant
β_0	Bounding value associated to the control input
$\dot{\pi}$	Zero dynamics of the plant
$\beta(\mathrm{X})$	Unknown control input dynamics
n	Plant order
γ_1	Positive constant
B	Positive function
ν_1	Free parameter function
γ_2	Positive constant
γ_3	Positive constant
γ_4	Positive constant
$\beta(\mathrm{X})$	Controller gain
β_1	Bounded constant for controller gain
$u*$	Feedback linearization free parameter ideal controller

$\alpha(\mathrm{X})$	Approximated system dynamics
u_k	Known part of the controller
$d_u(\mathrm{X})$	Approximation error
$Z_{t'}^T$	Input vector membership function
A_u^*	Ideal direct control parameters matrix
$\hat{u}$	Fuzzy approximation of the desired control
A_u	Updated parameters matrix
$\Phi_u(t)$	Parameter error matrix for the direct adaptive controller
u_{bd}	Bounding control term
u_{sd}	Sliding-mode control term
e_s	Output signal error
e_o	Output error
k	Constant vector
η	Scalar constant
v_{bd}	Bounding control Lyabunov candidate function
ε_M	Bounding fixed parameter
M_e	Bounding fixed parameter
$\prod(t)$	Bounding function associated to the bounding control term
$\hat{G}_t(s)$	Bounding error transfer function
V_d	Adaptation control Lyabunov candidate function
$q(t)$	Function term to improve fuzzy adaptation

Q_u^{-1}	Identity diagonal matrix for adaptation
S_x	Compact set
u_{sd}	Sliding control
$k_{sd}(t)$	Function associated with sliding control
$\bar{d}_{11}$	Pendubot linearized system parameter
$\bar{c}_{11}$	Pendubot linearized system parameter
$\bar{c}_{12}$	Pendubot linearized system parameter
$\bar{\phi}_1$	Pendubot linearized system parameter
K_p	Proportional derivative positive gain
K_d	Proportional derivative positive gain positive gains
q_1^d	Link one angular desired trajectory
z_1	Link one zero dynamics
z_2	Link one zero dynamics
η_1	Link one state space variable in Pendubot linearized model
η_2	Link two state space variable in Pendubot linearized model
PD	Proportional derivative
NB	Negative big
NS	Negative small
ZE	Zero
PS	Positive small
PB	Positive big

A	Width error membership function of PD fuzzy controller
B	Width change in error membership function of PD fuzzy controller
D	Width output membership function of PD fuzzy controller
e	PD fuzzy error
$\dot{e}$	PD fuzzy change-in-error
u	Single control input for the Pendubot
x_r	Equilibrium values of the states
u_r	Equilibrium values control respectively
$f_a(x_r, u_r)$	Pendubot at equilibrium positions

CHAPTER 1

INTRODUCTION

1.1 Overview

A system is said to be underactuated when the number of actuators is less than the number of degrees of freedom of the system [1-6]. This class of systems presents challenging control problems and has recently gained an upswing in research attention. They often exhibit feedforward nonlinearities, nonholonomic constraints and nonminimum phase characteristics, which make them difficult to control. They arise in applications as underactuated marine vehicles, space robots [19], flexible robots, in mobile robot systems when a manipulator arm is attached to a mobile platform, walking and gymnastic robots [7] such as the Pendubot [8-13] [22-23] and Acrobot [14-15] [20-21].

Underactuated robots are those robots with both passive and active joints [6]. For example, when a joint motor in a fully-actuated robot fails, that joint becomes passive. It is important in such cases to design control techniques to still control the robot, if possible. Another example is a hyper-redundant snake-like robot with several degrees-of-freedom. In this case one would like to see if control is possible when not all joints are actuated. In space applications, a space platform may also benefit from the study of underactuated robots. The kinematics, dynamics and control of free-floating space robotic

system satellite can be found in [16-19]. In such publications, the free-floating space robotic system can be modeled as a 6 degrees-of-freedom passive mechanism. The overall system (satellite + manipulator) can be considered as a combination of actuated and passive subsystems, i.e., as an underactuated system.

Uunderactuted robotic systems have received special interest in the last decade because of its wide range of applicability in different control research areas such as: nonlinear control [20], linear control [6], sliding mode control [2], optimal control, learning control [25], robust and adaptive control [1], fuzzy logic control [22-23], neural networks control, reference tracking control, intelligent control [15], hybrid and switching control, gain scheduling, and other control paradigms.

Acrobot is one case study of underactuated two-link planar robot. It mimics the human acrobat who hangs from a bar and tries to swing up to a perfectly balanced upside-down position with his/her hands still on the bar. Acrobot system just has one actuator at the elbow. Plenty of researchers worked on Acrobot and its control. Swing up controller of Acrobot was proposed in [14]. Pseudolinearization using spline functions with application to Acrobot was the interest of [21]. Nonlinear control of Acrobot can be found in [20]. Intelligent and learning control is proposed in [15].

Pendubot, which is the case study of this research, has been under valuable researchers interest [8-13] [22-23]. Pendubot, as shown in Figure 1.1 short name for **pendu**lum ro**bot,** is a nonlinear underactuated mechatronic system [6], consisting of unstable two

link planar robot arm [8-10]. One can program the Pendubot for swing up control, balancing, regulation and tracking, system identification, disturbance rejection, and friction compensation to name just a few of the applications [9-11].

Using the Pendubot one generally can also investigate equilibria of underactuated mechanical systems which depend on both their kinematic and dynamic parameters. If the Pendubot is mounted so that joint axes are perpendicular to gravity, then there will be a continuum of equilibrium configurations, each corresponding to a constant value of the input torque [11]. Indeed, its nonlinear dynamics have forced researchers to employ two very different varieties of controllers, one for the swing up and another for balancing as shown in Figure 1.2. Typically, a heuristic strategy is used for swing-up, where the goal is to force the Pendubot to reach its vertical upright position with near zero velocity of both links. Then, when the links are close to the inverted position, a balancing controller is switched on and used to maintain the Pendubot in the inverted position.

Many controllers for the Pendubot have been proposed. Non-adaptive control techniques to control Pendubot can be found in [4] [12-13]. The authors in [12] investigated how to control the Pendubot by energy based method. A passivity based control of the Pendubot is proposed in [13] as well. In [4] the control of underactuated systems using switching and saturation is discussed. In [9-11] researchers focused on the development of conventional controllers for the Pendubot. Their work serves an introduction to the Pendubot, its dynamics, equilibrium manifold, identification and controllability. A partial feedback linearization controller is designed to swing up the Pendubot from its free stable

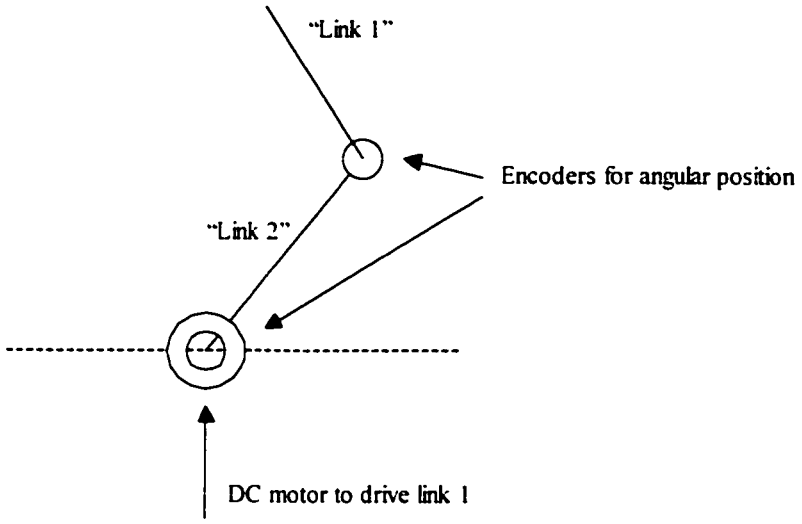

Figure 1.1: Pendubot

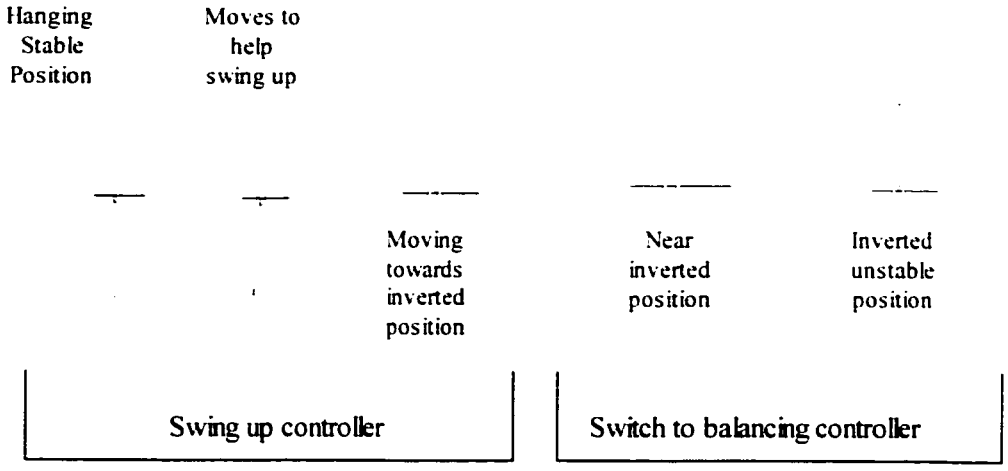

Figure 1.2: Pendubot stages of swing up and balancing

hanging position [10]. A linear quadratic regulator (LQR) balancing controller is developed to catch the pendubot in the vertical position and keep it there [11].

Fuzzy Logic Controllers (FLC) use fuzzy logic as process of mapping from a given input (crisp numerical value e) to an output (signal control u). This process has a basic structure that involves a fuzzifier, an inference engine, a knowledge base (rule data base), and a deffuzzifier which transforms fuzzy sets into real numbers to provide control signals.

Applying non-linear and fuzzy logic on underactuated mechatronic systems attracts many researchers. Two relevant works [22-23] are done at Concordia University by Prof C.-Y. Su and his students, who focused on developing of nonlinear control theory on robotic manipulators, proposed and implemented different controllers. In [22] a simplified Tsukamoto's reasoning method and quasi-linear-mean aggregating fuzzy operators proposed and implemented on Pendubot. While in [23] a reference fuzzy trajectory control is proposed and implemented on Pendubot.

Adaptive control is well known in its ability to learn the system parameters. Because Adaptive control of underactuated systems has proven its value in many publications. Su and others [2] pioneered the research on the sliding model control of nonholomonic mechanical system- underactuated manipulator case. Su et al. also proposed [1] an adaptive variable structure set-point control of underacatauted robots. Adaptive control of space robot systems free-floating satellite has been approached in [19].

1.2 Research Objectives

The objective of this research as well as mentioned before is to design stable controllers for two challenging robotics control problems associated with the Pendubot: swing-up and balancing [9-11] with only one torque input located at the shoulder.

For the swing up control, the goal is to design a controller by using proportional derivative (PD) fuzzy controller with zero dynamics and minimum number of linguistic rules If-Then rules.

Then the other objective related to the balancing control is to design a stable direct adaptive fuzzy balancing controller. A class of continuous time single-input single-output systems (SISO) with simple neural network or fuzzy rules are used to achieve this goal. The direct adaptive scheme allows for the inclusion of a priori knowledge about the system in terms of exact mathematical equations or linguistics. We prove that with such knowledge the adaptive schemes can "learn" how to control the plant and achieve asymptotically stable zero reference inputs. This adaptive fuzzy controller consists of three sub-controllers: fuzzy controller, sliding mode controller, bounding mode controller in addition to an adaptation mechanism to guarantee stable performance adaptation.

1.3 Contribution

An extensive use of fuzzy control as well as neural network has been made in this research. Three different controllers have been proposed and implemented on the Pendubot. One is for the swing up by using PD fuzzy controller with zero dynamics [4], and another for balancing controller: by using adaptive neural fuzzy control .

a) This thesis research presents the first application and results, which has not been reported to the best of author's knowledge on the use of adaptive fuzzy control and for the Pendubot.
b) The sability of the system has been proved by using Lyapunov stability.
c) The Pendubot was able to swing up from its free position to the vertical position by using 25 If-Then-Rules rules.
d) The adaptive fuzzy controller was able to catch the pendubot around the vertical position and maintain this postion using only 5 If-Then-Rules.
e) The boundness of the states, control gain, and error is proved.
f) The data attained for the Pendubot is used to provide the learning of fuzzy controller on a continuing basis, which ensures that the performance objectives are met while uncertainties of the system are reasonably well controlled.

1.4 Thesis Organization

This theis is composed of 9 chapters. It is organized as follows. In chapter 2, a definition of the Pendubot system, description of the Pendubot hardware and hardware setup are explained.

Chapter 3 goes through the derivation of the mathematical model of the Pendubot, identification of its parameters, the equilibrium manifold and controllability. The ordinary differential equations presented in this chapter are the basis for the controller designs to be used later.

Chapter 4 describes the aspects of the fuzzy logic and fuzzy control, fuzziness and probability, fuzzy logic terminologies, fuzy control systems: rules-base, fuzzification, inference mechanism, defuzzification method, and standard fuzzy system.

Chapter 5 defines neural networks. It introduces neural networks, neural network approximations, radial basis function (RBF) network and its properties.

Chapter 6 first discusses the equivalence between the fuzzy systems and neural networks from approximation point of view. A direct adaptive fuzzy or neural network controller is proposed to balance the Pendubot later. This chapter covers the derivation of the control law, bounding control, adaptation algorithm, sliding mode and stability properties.

Chapter 7 covers the control parts of the Pendubot. It describes the both swing up and balancing controllers used to control the Pendubot. For the swing up controller a PD fuzzy controller with zero dynamics is derived. Then an adaptive fuzzy is used to balance the Prndubot.

In chapter 8 the simulation results of the two different controllers are recorded and discussed.

At the end the in chapter 9 conclusions have been presented and future work is proposed.

CHAPTER 2

PENDUBOT SYSTEM

2.1 What is the Pendubot

Def 2.1 Underactuated Robotic System

A manipulator system in which the number of actuators is less than the number of joints or degrees of freedom.

Def 2.1 Pendubot

The Pendubot, short for PENDUlum robot [9-11], is an unederactuated electro-mechanical (or mechatronic) system consisting of two rigid links interconnected by revolute joints Figure 2.1.

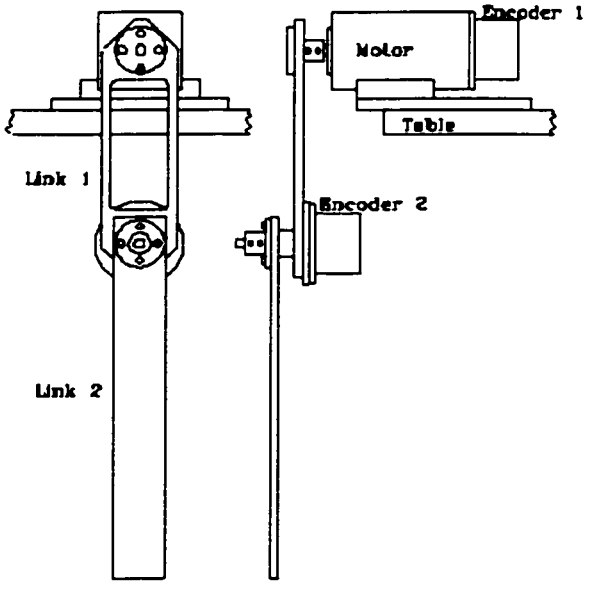

Figure 2.1: Front and side perspective drawings of the Pendubot.

This system has only one actuator at the shoulder that is why it has been called underactauted system in which the number of actuators is less than the number of degrees of freedom or joints. The first joint is actuated by a DC-motor and the second joint is unactuated. Thus the second link may be thought of as a simple pendulum whose motion can be controlled by actuation of the first link. The Pendubot is similar in spirit to the classical inverted pendulum on a cart or the more recent rotational inverted pendulum. However, because of the nonlinear dynamic coupling between the two links, the Pendubot possesses some unique features and challenges for control research and education not found in other devices.

2.2 Description of the Hardware

The Pendubot consists of two links with ¼ inch (0.635 cm) thick length aluminium [10]. Link 1 is 6 inches (15.24 cm) long and is directly coupled to the shaft of a 90V permanent magnet DC motor mounted on the supporting base. Link 2 is 9 inches (22.86 cm) long and contains a coupling that attaches to the shaft of joint two (Figure 2.2).

The Pendubot is shown schematically in Figure 2.3. The actuated joint is driven by a high torque 10VDC permanent magnet motor without gear reduction. To give joint one direct drive control, the Pendubot was designed to hang off the side of a table coupling link one directly to the shaft of the motor. The mount and bearings of the motor are then the support for the entire system. Link one also includes the bearing housing which allows

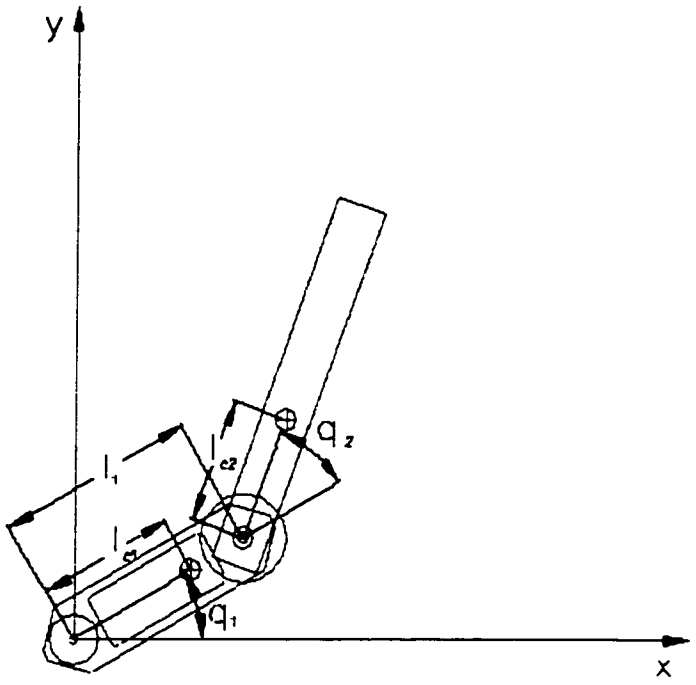

Figure 2.2: Coordinate description of the Pendubot

for the motion of joint two. Needle roller bearings riding on a ground shaft were used to construct this revolute joint for joint two. The shaft extends out both directions of the housing allowing coupling to both link two and an optical encoder mounted on link one. This optical encoder produces the position feedback of link two. The design gives both links full 360° of motion. Link one, however, cannot continuously rotate due to the encoder cable for link two. Link has no constraint on continuous revolutions.

Figure 2.3: Pictorial of the Pendubot's interface with its controller

Two Dynamics Research Corporation 1250 counts/rev resolution optical encoders, one attached at the elbow joint and the other attached to the motor, and are used as the feedback mechanism for the joint angles. An Advanced Motion Control's 25A8 PWM servo amplifier is used to drive the motor. In the control algorithm this amplifier can be thought of as just a gain. In the case of the Pendubot we setup the amplifier in torque mode and adjusted it for a gain of 1V=1.2Amps. The final component of the Pendubot's hardware is its controller. See Figure 2.2 for a pictorial description of the interface between the Pendubot and the controller.

In an attempt to simplify the controller for the Pendubot and minimize its cost, Mechatronic Systems Inc. implemented control algorithm using only the microprocessor in PC instead of purchasing an additional DSP card. A 486DX2/50 IBM compatible PC with a D/A card and an encoder interface card are used. The CIO-DAC-02 card from Computer Boards, Inc. is used for digital to analog conversion and a Dynamics Research

Corporation optical encoder card is used to interface with the optical encoders. Controller timing is provided by a timer board of Computer Boards, Inc. that utilizes the 9513 timer chip. Using the standard software library routines supplied with the interface cards together with our own drivers we are able to program control algorithms directly in Microsoft C 7.0.

2.3 Hardware Setup

The Pendubot system is comprised of the following hardware subsystems[10].

- A base unit consisting of a mounting plate, case, power supply, amplifier, and all the necessary connectors.
- A mechanical linkage (two rigid aluminum links that are coupled through a revolute joint that allows for 360 degrees of rotation).
- Two Dynamics Research Corporation 1250 counts/rev optical encoders (one encoder is included in the base unit and the second encoder is attached to the revolute joint of the mechanical linkage).
- A handheld amplifier inhibit switch that must be depressed for control effort to be applied to the Pendubot.
- The Servo-To-Go STG S8 motion control board is used for data acquisition.

If the hardware is setup as indicated by the instructions given in the Pendubot user's manual, the following connections are automatically incorporated

- D/A channel 0 powers the servo motor that drives the first link of the Pendubot.

- Encoder channel 0 reads position counts from the first link of the Pendubot.
- Encoder channel 1 reads position counts from the second link of the Pendubot.

CHAPTER 3

PENDUBOT SYSTEM DYNAMICS

3.1 Pendubot Model

The equation of motion for the Pendubot can be found using Lagrangian dynamics [62]. This provides a mathematical model that can be used to derive various controllers and to simulate the response. Since our device is a two-link robot (with only one actuator), its dynamic equations can easily be derived using the so-called Euler-Lagrange equations and can be found in numerous robotics textbooks. In additional, the inertia parameters of the Pendubot have been provided by the Mechatronic Systems, Inc [10-11].

For the purposes of control design, we assume that the robotic manipulator dynamics with n revolute rigid-link serially connected direct-drive can be given by

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{3-A}$$

Where $D(q)$ is the inertia matrix, $C(q,\dot{q})$ is the Coroilis / centripetal vector, $g(q)$ is the gravity vector.

Equations of motion for the Pendubot can be simply written in matrix [10]:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \begin{bmatrix} \tau \\ 0 \end{bmatrix} \tag{3.1}$$

where, $D(q)$ is symmetric and positive definite, and

Figure 3.1: Coordinate description of the Pendubot.

$$D(q) = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}$$

$$d_{11} = m_1 l_{c1}^2 + m_2 (l_1^2 + l_{c2}^2 + 2 l_1 l_{c2} \cos q_2) + I_1 + I_2 \tag{3.2}$$

$$d_{12} = d_{21} = m_2 (l_{c2}^2 + l_1 l_{c2} \cos q_2) + I_2$$

$$d_{22} = m_2 l_{c2}^2 + I_2$$

$$C(q, \dot{q}) = \begin{bmatrix} h\dot{q}_2 & h\dot{q}_2 + h\dot{q}_1 \\ -h\dot{q}_1 & 0 \end{bmatrix}$$

$$h = -m_2 l_1 l_{c2} \sin q_2 \tag{3.3}$$

Note the 0 in the vector on the right side of equation 3.1, indicating the absence of an actuator at the first joint where τ is the vector of torque applied to the links and q is the vector of joint angle positions.

Other,

$$g(q) = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}$$

$$\phi_1 = (m_1 l_{c1} + m_2 l_1) g \cos q_1 + m_2 l_{c2} g \cos(q_1 + q_2) \tag{3.4}$$

$$\phi_2 = m_2 g l_{c2} \cos(q_1 + q_2)$$

m_1 : the total mass of link one.

l_1 : the length of link one (See Figure 3.1).

l_{c1} : the distance to the center of mass of link 1 (See Figure 3.1).

I_1 : the moment of inertia of link one about its centroid.

m_2 : the total mass of link two.

l_2 : the length of link two (See Figure 3.1).

l_{c2} : the distance to the center of mass of link 2 (See Figure 3.1).

I_2 : the moment of inertia of link one about its centroid.

g : the acceleration of gravity.

From the above equations it is observed that the seven dynamic parameters can be grouped into the following five parameters equations

$$\theta_1 = m_1 l_{c1}^2 + m_2 l_1^2 + I_1$$

$$\theta_2 = m_2 l_{c2}^2 + I_2$$

$$\theta_3 = m_2 l_1 l_{c2} \tag{3.5}$$

$$\theta_4 = m_1 l_{c1} + m_2 l_1$$

$$\theta_5 = m_2 l_{c2}$$

Substituting these parameters (3.5) into the equations (3.2)-(3.4) leaves the following matrices

$$D(q) = \begin{bmatrix} \theta_1 + \theta_2 + 2\theta_3 \cos q_2 & \theta_2 + \theta_3 \cos q_2 \\ \theta_2 + \theta_3 \cos q_2 & \theta_2 \end{bmatrix}. \tag{3.6}$$

$$C(q, \dot{q}) = \begin{bmatrix} -\theta_3 \sin(q_2)\dot{q}_2 & -\theta_3 \sin(q_2)\dot{q}_2 - \theta_3 \sin(q_2)\dot{q}_1 \\ \theta_3 \sin(q_2)\dot{q}_1 & 0 \end{bmatrix}. \tag{3.7}$$

$$g(q) = \begin{bmatrix} \theta_4 g \cos q_1 + \theta_5 g \cos(q_1 + q_2) \\ \theta_5 g \cos(q_1 + q_2) \end{bmatrix}. \tag{3.8}$$

Finally, using the invertible property of the mass matrix $D(q)$, the state equations are given by

$$\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = D(q)^{-1}\tau - D(q)^{-1}C(q, \dot{q})\dot{q} - D(q)^{-1}g(q) \tag{3.9}$$

$$x_1 = q_1, \quad x_2 = \dot{q}_1, \quad x_3 = q_2, \quad x_4 = \dot{q}_2$$

$$\dot{x}_1 = x_2 \tag{3.10}$$

$$\ddot{x}_2 = \ddot{q}_1$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = \ddot{q}_2$$

3.2 Identification of Pendubot Parameters

For a control design that neglects friction, these five parameters are all needed [11]. There is no reason to go a step further and find the individual parameters since the control equations can be written with only the five parameters. Mechatronic Systems, Inc. performed on-line identification experiments using the Hamiltonian based energy equation method. The first step in this approach is to write the total energy as the sum of the kinetic energy and potential energy as

$$E = \frac{1}{2}\dot{q}^T D(q)\dot{q} + V(q) \tag{3.11}$$

where $D(q)$ is the inertia matrix defined in (3.1) and $V(q)$ represents the potential energy due to gravity. The total energy E is linear in the inertia parameters and so may be written as

$$E = W(q,\dot{q})\theta \tag{3.12}$$

where θ is the parametrization defined in (3.5). Using the well-known passivity or skew-symmetry property

$$\dot{E} = \dot{q}^T \tau \tag{3.13}$$

we have that, between any two times $t = T$ and $t = T + dT$

$$dE = E(T+dT) - E(T) = \int_{T}^{T=dT} \dot{q}^T(u)\tau(u)du = \{W(T+dT) - W(T)\}\theta \tag{3.14}$$

Equation (3.14) can be used with a standard least squares algorithm to identify the parameter vector θ by applying an open loop signal $t \rightarrow \tau(t)$ and recording $\tau, q, \dot{q}$ over a given time interval. Carrying out this identification procedure using a step input to excite the Pendubot resulted in the following parameter values.

$$\theta_1 = 0.0308 \qquad Volts.\sec^2.$$

$$\theta_2 = 0.0106 \qquad Volts.\sec^2.$$

$$\theta_3 = 0.0095 \qquad Volts.\sec^2.$$

$$\theta_4 = 0.2087 \qquad Volts.\sec^2/m.$$

$$\theta_5 = 0.0630 \qquad Volts.\sec^2/m.$$

3.3 The Equilibrium Manifold

Under-actuated mechanical systems generally have equilibria that depend on both their kinematic and dynamic parameters; see, for example, the Pendubot [11]. If the Pendubot is mounted so that the joint axes are perpendicular to gravity, then there will be a continuum of equilibrium configurations, as shown in Figure 3.2, each corresponds to a constant value, $\bar{\tau}$, of the input torque τ. Examining the equation (3.1) we see the equilibrium configurations are determined by

$$\theta_4 g\cos(q_1) + \theta_5 g\cos(q_1 + q_2) = \bar{\tau} \tag{3.15}$$

$$\theta_5 g\cos(q_1 + q_2) = 0 \tag{3.16}$$

It is easily seen that, applying a constant torque $\bar{\tau}$, the Pendubot will be balanced at a configuration (q_1, q_2) such that

$$q_1 = \cos^{-1}(\frac{\bar{\tau}}{\theta_4 g}) \tag{3.17}$$

$$q_2 = n\frac{\pi}{2} - q_1; \quad n = 1,3,5,... \qquad \text{provided} \frac{\bar{\tau}}{\theta_4 g} \le 1. \tag{3.18}$$

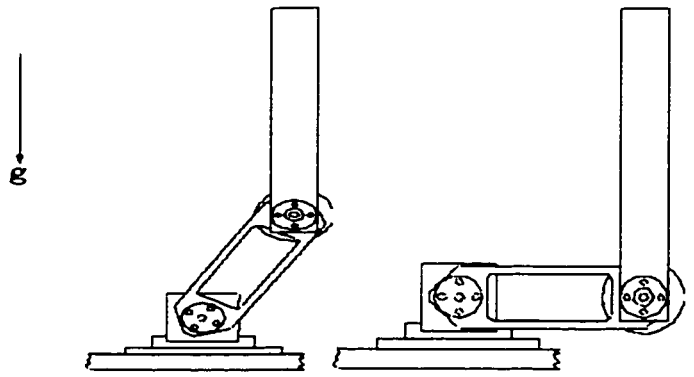

Figure 3.2: Possible equilibrium positions

3.4 Controllability

There are four uncontrollable positions, $q_1 = 0$, $q_2 = \pi/2$ or $-\pi/2$ and $q_1 = -\pi$, $q_2 = \pi/2$ or $-\pi/2$. However, we can also easily understand physically how the linearized system becomes uncontrollable at $q_1 = 0, \pm\pi$ as illustrated in Figure 3.3. Note that the zero or reference position for q_1 is horizontal. As the Pendubot approaches these unstable and uncontrollable configurations, the controllability matrix of the linearized approximation becomes increasingly ill-conditioned.

Define the controllability matrix as follows:

$$Co = [B \quad AB \quad A^2B \quad A^3B] \tag{3.19}$$

where both matrices A and B are defined in Appendix A, Equations A.4 and A.9, respectively. After substituting the system parameters and the configurations, $q_1 = 0$, $q_2 = \pi/2$ or $-\pi/2$ and $q_1 = -\pi$, $q_2 = \pi/2$ or $-\pi/2$.

$$A = \begin{bmatrix} 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 58.3047 & 0 & 58.3047 & 0 \end{bmatrix} \tag{3.20}$$

$$B = \begin{bmatrix} 0 \\ 32.4675 \\ 0 \\ -32.4675 \end{bmatrix} \tag{3.21}$$

The controllability matrix obtained was

$$Co = \begin{bmatrix} 0 & 32.4675 & 0 & 0 \\ 32.4675 & 0 & 0 & 0 \\ 0 & -32.4675 & 0 & 0 \\ -32.4675 & 0 & 0 & 0 \end{bmatrix} \tag{3.22}$$

Since the rank of Co is $2 < 4$, then the system is said to be uncontrollable at the given configurations.

Figure. 3.3: Uncontrollable configurations at $q_1 = 0$, $q_2 = \pi / 2$

CHAPTER 4

FUZZY LOGIC AND CONTROL

4.1. Introduction: Fuzziness versus Probability

Logical paradoxes as Crete's liars and the Heisenberg uncertainty principle led to the development of multi valued or "fuzzy" logic [35-48] in the 1920s and 1930s. Quantum theorists allowed for indeterminacy by including a third or middle truth value in the bivalent logical framework. The next step allowed degrees of indeterminacy, viewing TRUE and FALSE as the two limiting cases of the spectrum of indeterminacy.

But, is certainty the same as randomness? If we are not sure about something, is it only up to change? Do the notions of likelihood and probability exhaust our notions of uncertainty? Many people, trained in probability and statistics, believe so:

"*Any method of inference in which we represent degrees of plausibility by real numbers, is necessarily either equivalent to Laplace's probability, or inconsistent*".

"*Probability is the only sensible description of uncertainty and is adequate for all problems involving uncertainty. All other methods are inadequate*".

How important is to be exact right when a rough answer will do? Related to the importance of imprecision some people have said:

"*So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality*". Albert Einstein

"*As complexity rises, precise statements lose meaning and meaningful statements lose precision*". Zada

In 1965, Lofti Zadeh formally developed multivalued set theory, and introduced the term *fuzzy* into the technical literature. Nowadays, the recent emergence of fuzzy commercial products, as well as a new theory, has generated a new interest in multivalued systems. Yet already engineers have successfully applied fuzzy systems in many commercial areas: intelligent subways automation, emergency breakers, cement mixers, Kanji characters recognition, air conditioners, automatic washing machines, guide of robot-arm manipulators, and so on [36-40].

Fuzzy systems store banks of fuzzy associations or common-sense "rules" such as "*IF traffic is heavy in this direction, THEN keep the light green longer*" that might be articulated by a human expert. Some traffic configurations are *heavier* than others and some green-light duration are *longer* than others, so that, the single fuzzy association (HEAVY, LONGER) encodes all these combinations. That is to say, fuzzy systems directly encode the structured knowledge but in a numerical framework: by entering the fuzzy association (HEAVY, LONGER) as a single entry in a rule database to define an input-output transformation.

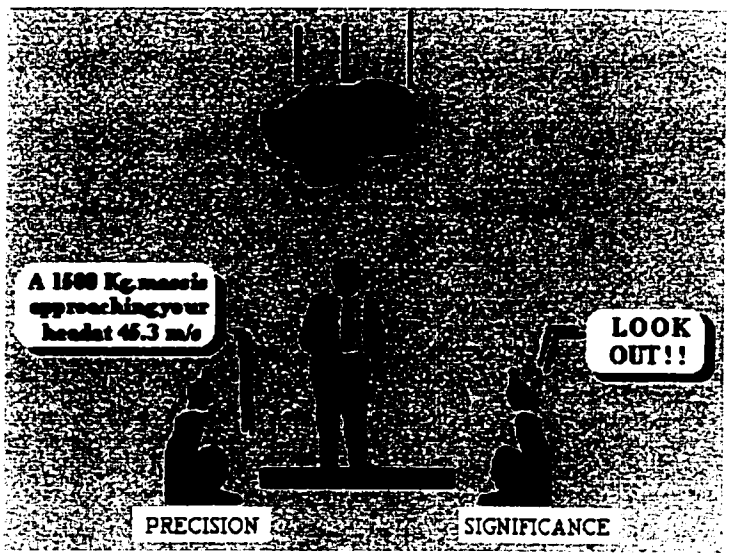

Figure 4.1: Precision and significance

Fuzzy logic sometimes appears exotic or intimidating, but it seems almost surprising when some becomes acquainted with it. In this sense, fuzzy logic is both old and new because, although the modern and methodical science of fuzzy logic is still young, the concepts of fuzzy logic reach down to our bones.

Claim: Probability theory is the only correct way of dealing with uncertainty and that anything can be done with fuzzy logic can be done equally well through the use of probability-based methods. And so fuzzy sets are unnecessary for representing and reasoning about uncertainty and vagueness - probability theory is all that is required.

"Close examination shows that the fuzzy approaches have exactly the same representation as the corresponding probabilistic approach and include similar calculi."

[Cheeseman, 1986].

Objection: Classical probability theory is not sufficient to express uncertainty encountered in expert systems. The main limitation is that it is based on two-valued logic. An event either occurs or does not occur; there is nothing between them. Another limitation is that in reality events are not known with sufficient precision to be represented as real numbers. As an example consider a case in which we have been given an information: An urn contains 20 balls of various sizes, several of which are large.

"One cannot express this within the framework of classical theory or, if it can be done, it cannot be done simply" (Zadeh to Cheeseman, in same book).

Fuzzy Logic has emerged as a profitable tool for the controlling of subway systems and complex industrial processes, as well as for household and entertainment electronics, and diagnosis systems. *Fuzzy* has become a keyword for marketing. Electronic articles without Fuzzy-component gradually turn out to be dead stock. As a gag, that shows the popularity of Fuzzy Logic, there even exists a toilet paper with "Fuzzy Logic" printed on it.

Fuzzy Logic is basically a multivalued logic that allows intermediate values to be defined between conventional evaluations like *yes/no, true/false, black/white,* etc. Notions like *rather warm* or *pretty cold* can be formulated mathematically and processed by computers. In this way an attempt is made to apply a more human-like way of thinking in the programming of computers.

4.2 Fuzzy Logic Terminologies

Let X be a space of objects and x be a generic element of X. A classical set A, $A \subseteq X$, is defined as a collection of elements or objects $x \in X$, such that each can either belong or not belong to the set A. By defining a characteristic function for each element x in X, the classical set A can be represented by a set of ordered pairs $(x,0)$ or $(x,1)$, which indicates $x \notin A$ or $x \in A$, respectively.

A Classical set is a set with crisp boundaries. For example: a classical set A of real numbers greater than 6 can be expressed as

$A = \{x \mid x > 6\}$,

where there is clear unambiguous boundary 6 such that if x is greater than this number, then x belongs to the set A; otherwise x does not belong to the set. This boundary is called a crisp boundary. In contrast a fuzzy set, as the name implies, is a set without a crisp boundary. That is, the transition from "belong to a set" to "not belong to a set" is gradual, and this smooth transition is characterized by what is called a membership function that gives the fuzzy sets the flexibility in modeling commonly used linguistic expressions.

Definition 4.1 Fuzzy sets and membership functions

If X is a collection of objects denoted generically by x, then a fuzzy set A in is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

where $\mu_A(x)$ is called the membership function (or MF for short) for the fuzzy set A. The MF maps each element of X to a membership grade or membership value between 0 and 1. Usually X is referred to as the universe of discourse, or simply the universe.

Definition 4.2 Support

The support of a fuzzy set A is the set of all points x in X such that $\mu_A(x) > 0$:

$$\text{support}(A) = \{x \mid \mu_A(x) > 0\}$$

Definition 4.3 Fuzzy singleton

A fuzzy set A whose support is a single point in X with $\mu_A(x) = 1$ is called a fuzzy singleton.

Definition 4.4 Containment or subset

Fuzzy set A is contained in fuzzy set B (or equivalently, A is a subset of B, or A is smaller than or equal to B) if only and if $\mu_A(x) \leq \mu_B(x)$ for all x. In symbols

$$A \subseteq B \leftrightarrow \mu_A(x) \leq \mu_B(x)$$

Definition 4.5 Union (disjunction)

A new set generated from two given sets A and B is called *unification of A and B*, if the new set contains all elements that are contained in A or in B or in both.

Then by symbols, union of two fuzzy sets A and B is a fuzzy set C, written as $C = A \cup B$ or $C = A$ OR B, whose MF is related to those of A and B by

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x)$$

Definition 4.6 Intersection (conjunction)

A new set generated from two given sets A and B is called *intersection of A and B*, if the new set contains exactly those elements that are contained in A and in B. Then by symbols, intersection of two fuzzy sets A and B is a fuzzy set C, written as $C = A \cap B$ or $C = A$ AND B, whose MF is related to those of A and B by

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x)$$

Definition 4.7 Complement (negation)

A new set containing all elements which are in the universe of discourse but not in the set A is called the *negation of A*. The complement of fuzzy set A, denoted by $\overline{A}$ ($-A$, NOT A), is defined as

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x)$$

Definition 4.8 Gaussian membership function

A MF is a curve that defines how each point in the universe of discourse is mapped to a value between 0 and 1. This value is called membership value or degree of membership. Then a Gausssian MF is specified by two parameters $\{c, \sigma\}$:

$$\text{gaussian}(x; c, \sigma) = \exp\left(-0.5\left(\frac{x-c}{\sigma}\right)^2\right)$$

A Gaussian MF is determined completely by c and σ: c represents the MFs center and σ determines the membership function width.

Definition 4.9 Linguistic variables

A linguistic variable is characterized by a quintuple $(x, T(x), X)$ in which x is the name of the variable; $T(x)$ is the term set of x-that is, the set of its linguistic values or linguistic terms; X is the universe of discourse.

In order to clarify this, the few examples follow explain the terms which are defined above.

Example 4.1:

If age x is interpreted as a linguistic variable, then its term set $T(age)$ could be

$T(age)$ = {young, not young; very young, Not very young, ...

middle aged, not middle aged, ...

old, not old, very old, more or less old, not very old, ...

not very young and not very old, ...},

where each term in $T(age)$ is characterized by a fuzzy set of a universe of discourse $X = [0,100]$.

Example 4.2:

First consider a set X of all real numbers between 0 and 10, which we call the universe of discourse. Now, let's define a subset A of X of all real numbers in the range between 5 and 8. $A = [5,8]$

We now show the set A by its characteristic function, i.e. this function assigns a number 1 or 0 to each element in X, depending on whether the element is in the subset A or not. The result is shown in the following figure:

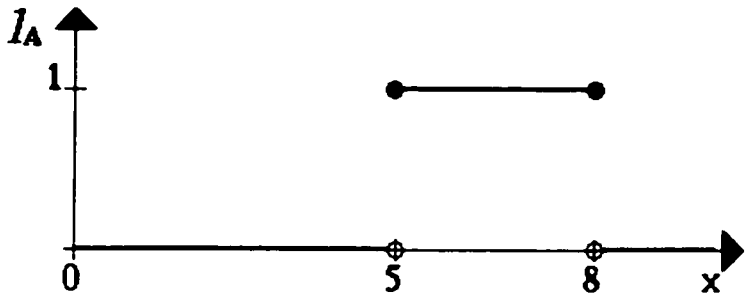

Figure 4.2: Universe of discourse between 0 and 10 and fuzzy set between 5 and 8

We can interpret the elements which have assigned the number 1 as *The elements are in the set A* and the elements which have assigned the number 0 as *The elements are not in the set A.*

This concept is sufficient for many areas of applications. But we can easily find situations where it lacks in flexibility. In order to show this consider the following example. In this example we want to describe a set of young people. More formally we can denote

$B = \{set\ of\ young\ people\}$

Since - in general - age starts at 0 the lower range of this set ought to be clear. The upper range, on the other hand, is rather hard to define. As a first attempt we set the upper range to, say, 20 years. Therefore we get B as a crisp interval, namely: $B = [0,20]$

Now the question arises: why is somebody on his 20th birthday *young* and right on the next day *not young*? Obviously, this is a structural problem, for if we move the upper bound of the range from 20 to an arbitrary point we can pose the same question.

A more natural way to construct the set B would be to relax the strict separation between *young* and *not young*. We will do this by allowing not only the (crisp) decision *YES he/she is in the set of young people* or *NO he/she is not in the set of young people* but more flexible phrases like *Well, he/she belongs a little bit more to the set of young people* or *NO, he/she belongs nearly not to the set of young people.*

The next shows how a fuzzy set allows us to define such a notion as *s/he is a little young.* As stated in the introduction we want to use fuzzy sets to make computers smarter, we now have to code the above idea more formally. In our first example we coded all the elements of the Universe of Discourse with 0 or 1. A straight way to generalize this concept is to allow more values between 0 and 1. In fact, we even allow infinite many alternatives between 0 and 1, namely the unit interval $I=[0, 1]$.

The interpretation of the numbers now assigned to all elements of the Universe of Discourse is much more difficult. Of course, again the number 1 assigned to an element means that the element is in the set B and 0 means that the element is definitely not in the set B. All other values mean a gradual membership to the set B.

To be more concrete we now show the set of young people similar to our first example graphically by its characteristic function.

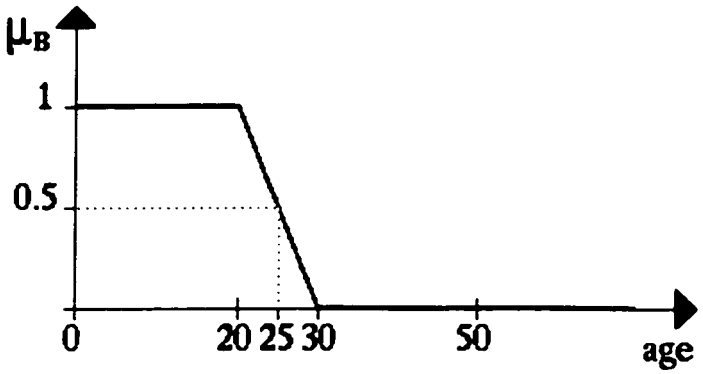

Figure 4.3: Fuzzy set and gradual membership for young people

This way a 25 years old would still be *young* to a degree of 50 percent. Now you know what a fuzzy set is. But what can you do with it?

Example 4.3:

Let *A* be a fuzzy interval *between* 5 *and* 8 and *B* be a fuzzy number *about* 4. The corresponding figures are shown below.

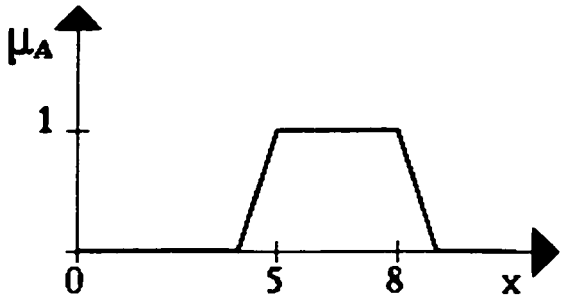

Figure 4.4: Fuzzy set between 5 and 8

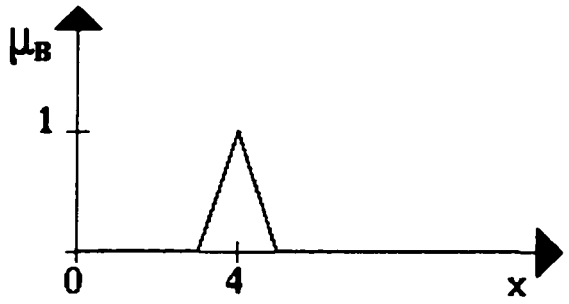

Figure 4.5: Fuzzy set about 4

Example 4.5:

The following figure shows the fuzzy set *between* 5 *and* 8 AND *about* 4.

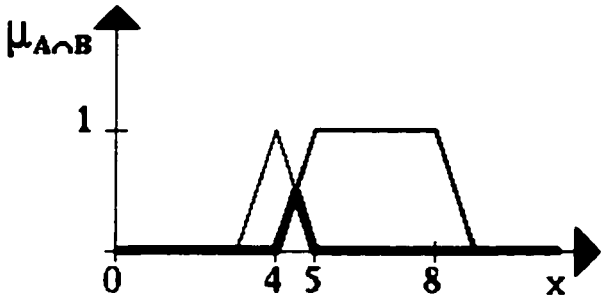

Figure 4.6: Fuzzy set between (5 and 8) and about 4

Example 4.6:

The Fuzzy set *between* 5 *and* 8 OR *about* 4 is shown in the next figure.

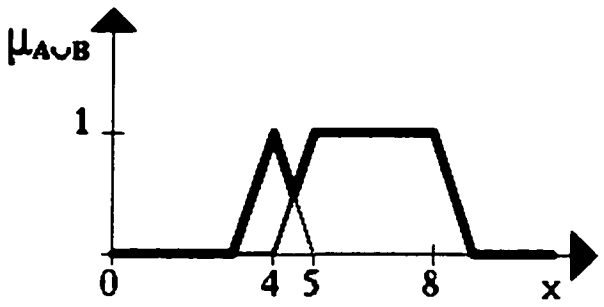

Figure 4.7: Fuzzy set 5 *and* 8 *or* about 4

Example 4.7:

This figure gives an example for a negation. The thick line is the negation of the fuzzy set *A*.

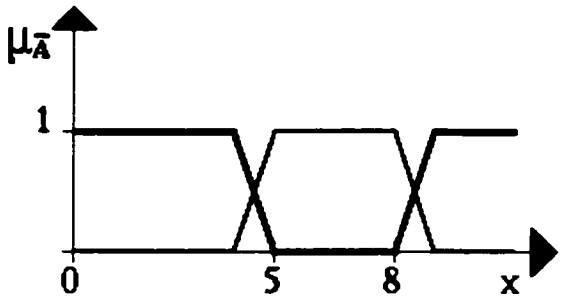

Figure 4.8: Negation of fuzzy set between 5 and 8

4.3 Fuzzy Control System

A fuzzy system is a static nonlinear mapping between its inputs and outputs. It is assumed that the fuzzy system has inputs $u_i \in U_i$ where $i = 1,2,\cdots,n$ and outputs $y_i \in Y_i$ where $i = 1,2,\cdots,n$ as shown in Figure 4.9. The inputs and outputs are crisp-that is, they are real numbers, not fuzzy sets. The ordinary crisp sets U_i and Y_i are called the universe of discourse for u_i and y_i, respectively (in other words, they are their domain). In practical applications, most often the universes of discourse are simply the set of real numbers or some intervals or subset of real numbers.

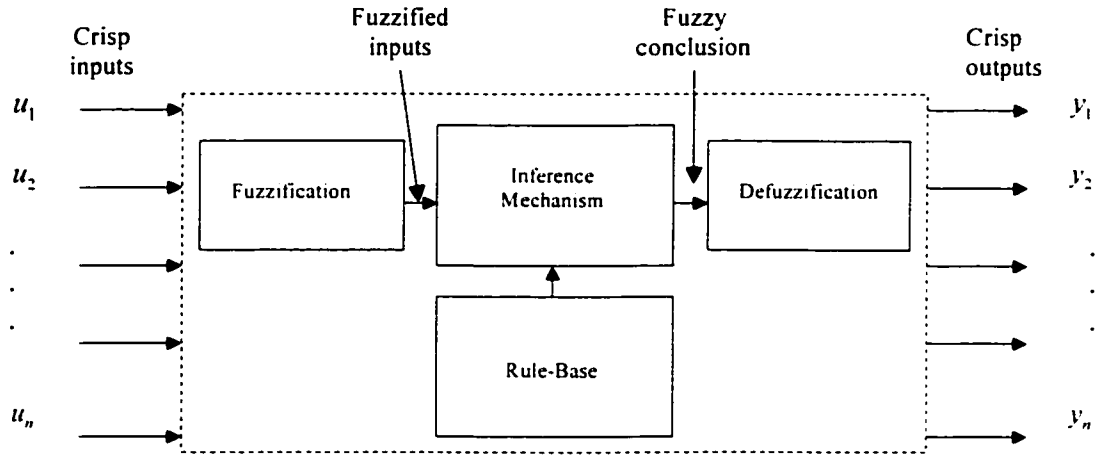

Figure 4.9: Fuzzy controller general structure

The block diagram in Figure 4.9 shows the general structure of the fuzzy control system. This fuzzy controller is composed of the following four elements: a rule-base unit, an inference mechanism (also called inference engine or fuzzy inference) uint , a fuzzification interface unit and finally a defuzzification interface unit.

The fuzzification block converts the crisp inputs to fuzzy sets, the rule-base contains all If-Then rules, the inference mechanism uses the fuzzy rules in the rule base to produce fuzzy conclusions, and the defuzzification block converts these fuzzy sets into crisp outputs.

To specify rules for the rule base, linguistic expressions are needed to describe the inputs and outputs and the characteristics of the inputs and outputs. Here linguistic variables denoted by $\tilde{u}_i$ are used to describe the inputs u_i. Similarly, linguistic variables denoted by $\tilde{y}_i$ are used to describe outputs y_i.

Example

An input to a fuzzy system may be described as $\tilde{u}_1$ = position error, or $\tilde{u}_2$ =velocity error, and output from the fuzzy system may be $\tilde{y}_1$ =voltage of the motor.

4.3.1 Linguistic values

Just as u_i and y_i take on values over each universe of U_i and Y_i, respectively, lingustic variables $\tilde{u}_i$ and $\tilde{y}_i$ take on linguistic values that are used to describe characterstics of the variables. Let $\tilde{A}_i^j$ denote j^{th} the linguistic value of the linguistic variable $\tilde{u}_i$ defined over the universe of discourse U_i. Then the set of all linguistic values is given by

$$\tilde{A}_i = \left\{\tilde{A}_i^j : j = 1,2,\ldots,N_i\right\}$$

And similarly, let $\tilde{B}_i$ denote j^{th} the linguistic variable $\tilde{y}_i$ defined over the universe of discourse Y_i. The linguistic variable takes on elements from the set of linguistic values denoted by

$$\tilde{B}_i = \left\{\tilde{B}_i^j : j = 1,2,\ldots,N_i\right\}$$

Linguistic values are generally descriptive terms such as positive large, zero, and negative big. Assume that $\tilde{u}_1$ denotes the linguistic variable speed then we may assign $\tilde{A}_1^1$ = slow, $\tilde{A}_1^2$ =medium, $\tilde{A}_1^3$ =fast so that $\tilde{u}_1$ has a value from $\tilde{A}_1 = \left\{\tilde{A}_1^1, \tilde{A}_1^2, \tilde{A}_1^3\right\}$.

4.3.2 Rule-Base and Linguistic Rules

A set of If-Then rules which contains a fuzzy logic quantification of the linguistic description of how to achieve good control. The mapping of the inputs to the outputs for a fuzzy system is in part characterized by a set of *condition*→*action* rules, or in modus ponens (If-Then) form. A fuzzy If-Then rule (also known as fuzzy rule, fuzzy implication or fuzzy conditional statement) assumes the form

If premise **Then** consequent

or

If x is A **then** y B

where A and B are linguistic values defined by fuzzy sets on the universe of discourse X and Y, respectively. Often "x is A " is called the antecedent or premise, while "y is B" is called the consequence or conclusion. Examples of fuzzy If-Then rules are widespread, such as:

- If the pressure is high then volume is small
- If the speed is high then apply then brake a little

Usually the inputs of the fuzy systems are associated with the premise, and the outputs are associated with the consequent. These If-Then rules can be represented in many forms. Two standard forms, multi-input multi-output (MIMO) and multi-input single-output (MISO). The MISO form of a linguistic rule is

If $\tilde{u}_1$ is $\tilde{A}_1^j$ **and** $\tilde{u}_2$ is $\tilde{A}_2^k$ **and**, ..., **and** $\tilde{u}_n$ is $\tilde{A}_n^l$ **then** $\tilde{y}_q$ is $\tilde{B}_q^p$

For instance, the MIMO rule with n inputs and m= 2 outputs is given by

If $\widetilde{u}_1$ is $\widetilde{A}_1^j$ **and** $\widetilde{u}_2$ is $\widetilde{A}_2^k$ **and**, ... **and** $\widetilde{u}_n$ is $\widetilde{A}_n^l$ **then** $\widetilde{y}_1$ is $\widetilde{B}_1^r$ **and** $\widetilde{y}_2$ is $\widetilde{B}_2^s$

4.3.3 Fuzzy Quantification of Rules: Fuzzy implications

Fuzzy implication is to quantify the linguistic elements in the premise and consequent of the linguistic If-Then rule with fuzzy sets. For example, suppose we are given If-Then rule in MISO form, then the fuzzy sets can be defined as follows:

$$A_1^j = \left\{\left(u_1, \mu_{A_1^j}(u_1)\right) : u_1 \in U_1\right\}$$

$$A_2^k = \left\{\left(u_2, \mu_{A_2^k}(u_2)\right) : u_2 \in U_2\right\}$$

$$\vdots$$

$$A_n^l = \left\{\left(u_n, \mu_{A_n^l}(u_n)\right) : u_n \in U_n\right\}$$

$$B_q^p = \left\{\left(y_q, \mu_{B_q^p}(y_q)\right) : y_q \in Y_q\right\}$$

These fuzzy sets quantify the terms in the premise and consequent of the given If-Then rule to make a fuzzy implication which is a fuzzy relation

If A_1^j and A_2^k and, ..., and A_n^l Then B_q^p

4.3.4 Fuzzification

Fuzzy sets used to quantify the information in the rule-base, and the inference mechanism operates on fuzzy sets to produce fuzzy sets, then how the fuzzy system will convert its numeric inputs $u_i \in U_i$ into fuzzy sets, this process is called fuzzification so that they can be used by the fuzzy system.

Let U_i^* denote the set of all possible fuzzy sets that can be defined on U_i. Given $u_i \in U_i$ fuzzification transforms u_i to a fuzzy set denoted by $\hat{A}_i^{fuz}$ defined on the universe of discourse U_i. This transformation is produced by the fuzzification operator F defined by

$$F : U_i \rightarrow U_i^*$$

where

$$F(u_i) = \hat{A}_i^{fuz}$$

quite often singleton fuzification is used, which produces a fuzzy set $\hat{A}_i^{fuz} \in U_i^*$ with a membership function defined by

$$\mu_{\hat{A}_i^{fuz}}(x) = \begin{cases} 1 & x = u_i \\ 0 & otherwise \end{cases}$$

Any fuzzy set with this form is called singleton.

4.3.5 Inference mechanism

The inference mechanism has two tasks: first, to determine the extent to which each rule is relevant to the current situation as characterized by the inputs $u_i, i = 1,2,\cdots,n$ which is called matching; and second, to draw conclusions using the current inputs u_i and the information in the rule base which is called inference step.

Matching

Suppose that at some time the inputs $u_i, i = 1,2,\cdots,n$ are obtained, and fuzzification produces

$\hat{A}_1^{fuz}$, $\hat{A}_2^{fuz}$, $\hat{A}_n^{fuz}$

which are the fuzzy sets representing the inputs. Then there are two basic steps to do matching

Step1: combine inputs with rule premises

The first step in matching involves finding fuzzy sets A_1', A_2^k, A_n' with membership functions

$$\mu_{\hat{A}_1'}(u_1) = \mu_{A_1'}(u_1) * \mu_{A_1^{fuz}}(u_1)$$

$$\mu_{\hat{A}_2^k}(u_2) = \mu_{A_2^k}(u_2) * \mu_{A_2^{fuz}}(u_2)$$

$$\vdots$$

$$\mu_{\hat{A}_n'}(u_n) = \mu_{A_n'}(u_n) * \mu_{A_n^{fuz}}(u_n)$$

for all j, k, ..., l that combine the fuzzy sets from fuzzification with fuzzy sets used in each of the terms in the premises of the rules. If singleton fuzzification is used, then each of these fuzy sets is a singleton that is scaled by the premise mebership function. That is, with singleton fuification $\mu_{\tilde{A}_i'}(u_i) = 1$ for all $i = 1,2,\cdots,n$ for the given inputs so that

$$\mu_{\tilde{A}_1'}(u_1) = \mu_{A_1'}(u_1)$$

$$\mu_{\tilde{A}_2'}(u_2) = \mu_{A_2'}(u_2)$$

$$\vdots$$

$$\mu_{\tilde{A}_n'}(u_n) = \mu_{A_n'}(u_n)$$

Step 2: Determine which rules are on:

In second step, the membership function values are formed for the rule's premise that represent the certainty that each premise holds for the given inputs. Define

$$\mu_i(u_1, u_2, \cdots, u_n) = \mu_{\tilde{A}_1'}(u_1) * \mu_{\tilde{A}_2'}(u_2) * \cdots * \mu_{\tilde{A}_n'}(u_n)$$

Which is simply a function of the inputs u_i . When singelton fuzzification is used, then

$$\mu_i(u_1, u_2, \cdots, u_n) = \mu_{A_1'}(u_1) * \mu_{A_2'}(u_2) * \cdots * \mu_{A_n'}(u_n)$$

$\mu_i(u_1, u_2, \cdots, u_n)$ is used to represent the certainty that the premise of rule i that matches the input information when singelton fuzification is used.

Inference step

The inference step is taken by computing for the i^{th} rule for $(j,k,\cdots,l)_t$, the implied fuzzy set $\hat{B}_q^t$ with membership function

$$\mu_{\hat{B}_q^t}(y_q) = \mu_t(u_1,u_2,\cdots,u_n) * \mu_{A_q^t}(y_q)$$

The implied fuzzy set $\hat{B}_q^t$ specifies the certainty level that the output should be specific crisp output with in the universe of discourse Y_q, taking into consideration only rule *i*.

4.3.6 Defuzification

Defuzification converts the conclustions of the inference mechanism into actual inputs for the process by using Center-average defuzzification: which means a crisp output is chosen using centers of each output membership functions given by

$$y_q^{crisp} = \frac{\sum_{t=1}^{R} b_t^q \mu_t(u_1,u_2,\cdots,u_n)}{\sum_{t=1}^{R} \mu_t(u_1,u_2,\cdots,u_n)}$$

where $\sum_{t=1}^{R} \mu_t(u_t,u_2,\cdots,u_n) \neq 0$ for all u_t.

4.4 Standard Fuzzy Logic Systems

The Sugeno fuzzy model (also known as Takagi-Sugeno-Kang fuzzy model) [44,46,48] developed a systematic approach to generating fuzzy rules from a given input-output data set. A typical fuzzy rule in a Sugeno fuzzy model has the form

If x is A and y is B then z=f(x,y)

Where A and B are fuzzy sets in the antecedent, while *z=f(x,y)* is a crisp function in the consequent. Usually *f(x,y)* is polynomial function within the input variables x and y, but it can be any function as long as it can appropriately describe the output of the system within the fuzzy region specified by the antecedent of the rule. When *f(x,y)* is a first-order polynomial, the resulting fuzzy inference system is called a first-order Sugeno fuzzy model, which was originally in [44,46].

It should be pointed out that the output of a zero-order Sugeno model is a smooth function in its variable as long as the neighboring MF's if the premise have enough overlap. In other words, the overlap of MF's in the consequent does not have a decisive effect on the smoothness of the interpolation; it is the overlap of the MF's in the premise that determines the smoothness of the resulting input-output behavior.

A multiple-input single-output (MISO) fuzzy system is a nonlinear mapping from an input vector $X = [x_1, x_2, ..., x_n]^T \in \Re^n$ (T denotes transpose) to an output $\tilde{y} = \tilde{f}(X) \in \Re$

(note that we use X as a general input vector to the fuzzy system; it may not be the same as "state" that is used in all the later sections). Using the Takagi-Sugeno model [29], the fuzzy system is characterized by a set of p **if-then** rules stored in the rule-base and expressed as

$$
\begin{aligned}
&R_1 : \text{If} \quad (\tilde{x}_1 \text{ is } \tilde{F}_1^l \text{ and} \ldots \text{and } \tilde{x}_n \text{ is } \tilde{F}_n^l) \\
&\quad \text{Then} \quad c_1 = g_1(X) \\
&\vdots \qquad\qquad \vdots \\
&R_n : \text{If} \quad (\tilde{x}_1 \text{ is } \tilde{F}_1^k \text{ and} \ldots \text{and } \tilde{x}_n \text{ is } \tilde{F}_n^l) \\
&\quad \text{Then} \quad c_p = g_p(X).
\end{aligned}
\tag{4.1}
$$

Here, $\tilde{F}_h^a$ is the a th linguistic value associated with linguistic variable $\tilde{x}_h$ that describes input x_h ,and $c_q = g_q(X)$ is the consequence of the q th rule and $g_q : \Re^n \to \Re$.

Using fuzzy set theory, the rule-base is expressed as

$$
\begin{aligned}
&R_1 : \text{If} \quad (F_1^l \text{ and} \ldots \text{and } F_n^l) \\
&\quad \text{Then} \quad c_1 = g_1(X) \\
&\vdots \qquad\qquad \vdots \\
&R_n : \text{If} \quad (F_1^k \text{ and} \ldots \text{and } F_n^l) \\
&\quad \text{Then} \quad c_p = g_p(X).
\end{aligned}
\tag{4.2}
$$

where F_h^a is a fuzzy set defined by

$$F_h^a := \{(x_h, \mu_{F_h^a}(x_h)) : x_h \in \Re\}.$$

The membership function $\mu_{F_h^a} \in [0,1]$ quantifies how well the linguistic variable $\tilde{x}_h$ represents x_h is described by the linguistic value $\tilde{F}_h^a$. There are many ways to define membership functions [29]. For instance, Table 4.3 specifies triangular membership functions with "center" c and "width" w, and it specifies Gaussian membership functions with "center" c and "width" σ.

Table 4.1 Types of membership functions

	Triangular	Gaussian
Left	$\mu(x) = \begin{cases} 1 & \text{if } x \le c \\ \max(0, 1 + \frac{c-x}{w}) & \text{otherwise} \end{cases}$	$\mu(x) = \begin{cases} 1 & \text{if } x \le c \\ \exp(-(\frac{x-c}{\sigma})^2) & \text{otherwise} \end{cases}$
Centers	$\mu(x) = \begin{cases} \max(0, 1 + \frac{x-c}{w} & \text{if } x \le c \\ \max(0, 1 + \frac{c-x}{w} & \text{otherwise} \end{cases}$	$\mu(x) = \exp(-(\frac{x-c}{\sigma})^2)$
Right	$\mu(x) = \begin{cases} \max(0, 1 + \frac{x-c}{w} & \text{if } x \le c \\ 1 & \text{otherwise} \end{cases}$	$\mu(x) = \begin{cases} \exp(-(\frac{x-c}{\sigma})^2) & \text{if } x \le c \\ 1 & \text{otherwise} \end{cases}$

The antecedent fuzzy set $F_1 \times F_2 \times \cdots \times F_n$ (fuzzy Cartesian product), of each rule is quantified by the "t-norm" [30] which may be defined by, for example, the min-operator or the product-operator

$$\mu_{F_1 \times \cdots \times F_n}(x_1, \cdots, x_n)$$

$$:= \min\{\mu_{F_1}(x_1), \cdots, \mu_{F_n}(x_n)\} \tag{4.3}$$

or

$$\mu_{F_1 \times \cdots \times F_n}(x_1, \cdots, x_n)$$

$$:= \mu_{F_1}(x_1) \cdots \mu_{F_n}(x_n) \tag{4.4}$$

Respectively (notice that for convenience, we have removed the superscripts from F_h^u). Using singleton fuzzification, defuzzification may be defined using

$$\tilde{y} = f(X) = \frac{\sum_{i=1}^{p} c_i \mu_i}{\sum_{i=1}^{p} \mu_i} \tag{4.5}$$

where $\mu_i := \mu_{F_1 \times \cdots \times F_n}(x_1, \cdots, x_n)$ is the value that the membership function [defined via (4.2) or (4.3)] for the antecedent of the ith rule takes on at $X = [x_1, \cdots, x_n]^T$. It is assumed that the fuzzy system is defined so that all $X \in \Re^n$, where $\sum_{i=1}^{p} \mu_i \neq 0$ .We may express (4.5) equivalently as

$$\tilde{y} = c^T \zeta = \tilde{f}(X) \tag{4.6}$$

where $c^T := [c_1 \cdots c_p]$ and $\zeta^T := [\mu_1 \cdots \mu_P] / [\sum_{i=1}^{p} \mu_i]$.

It is assumed that $\tilde{f}$, the mapping produced by fuzzy system, is Lipschitz continuous [29].

In this thesis, the output consequences for each rule are taken as linear combination of a set of Lipschitz continuous functions $\theta_k(X) \in \Re, k = 1,2, \cdots, m-1$, so that

$$c_i = g_i(X)$$

$$:= a_{i,0} + a_{i,1}\theta_1(X) + \cdots + a_{i,m-2}\theta_{m-2}(X) + a_{i,m-1}\theta_{m-1}(X) \tag{4.7}$$

$i = 1, \cdots, p.$

Define the following

$$z := \begin{bmatrix} 1 \\ \theta_1(X) \\ \vdots \\ \theta_{m-1}(X) \end{bmatrix} \in \Re^m \tag{4.8}$$

$$A^T := \begin{bmatrix} a_{1,0} & a_{1,1} & \cdots & a_{1,m-1} \\ a_{2,0} & a_{2,1} & \cdots & a_{2,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p,0} & a_{p,1} & \cdots & a_{p,m-1} \end{bmatrix} \tag{4.9}$$

The consequence vector associated with the fuzzy rules is now given by $c = A^T z$, so that the output of the fuzzy system may now be expressed as

$$\tilde{y} = z^T A \zeta \tag{4.10}$$

Clearly, (4.10) is a special form of Takagi-Sugino fuzzy system.

Standard fuzzy systems [29] naturally allow for the inclusion of Heuristics into controller design. In standard fuzzy control, the output of a fuzzy system may be found using the center of gravity operation, which for a wide class of fuzzy systems is expressed as

$$\tilde{y} = \frac{\sum_{i=1}^{p} c_i \xi_i}{\sum_{i=1}^{p} \xi_i} \tag{4.11}$$

Where c_i is the center of the output membership function associated with the ith rule, and ζ_i is the area of the implied membership function associated with ith rule (i.e. ζ_i is the area of the output membership function that is modified via the fuzzy implication that represents the ith rule). This fits the form of (4.10) with $z = [1], A = [c_1 \cdots c_p]$, and

$\xi_i = \xi_i / \sum_{i=1}^{p} \xi_i$ so that this fuzzy system is a special case of the Takagi-Sugeno fuzzy system defined by (4.10). Other standard fuzzy systems such as those that use centroid defuzzification will also fit the form of (4.10).

CHAPTER 5

NEURAL NETWORKS AND ROBOTICS

5.1 Introduction

The study of neural networks [28] [47-58] was started by the publication of Mc Culloch and Pitts [1943]. The single-layer networks, with threshold activation functions, were introduced by Rosenblatt [48] [1962]. These types of networks were called *perceptrons*. In the 1960s it was experimentally shown that perceptrons could solve many problems, but many problems which did not seem to be more difficult could not be solved. These limitations of one-layer perceptron were mathematically shown by Minsky and Papert in their book *Perceptron* [48]. The result of this publication was the subject neural networks lost their interest for almost two decades. In the mid-1980's, back-propagation algorithm was reported by Rumelhart, Hinton, and Williams [1986], which revived the study of neural networks. The significance of this new algorithm was that multilayer networks which will be dicussed in the next section could be trained by using it.

Neural network makes an attempt to simulate the human brain. The simulation is based on the present knowledge of brain function, and this knowledge is even at its best primitive. So, it is not absolutely wrong to claim that artificial neural networks probably have no close relationship to the operation of human brains. The operation of the brain is

believed to be based on simple basic elements called neurons which are connected to each other with transmission lines called axons and receptive lines called dendrites; as shown in Figure 5.1. The learning may be based on two mechanisms: the creation of new connections, and the modification of connections. Each neuron has an activation level which, in contrast to Boolean logic, ranges between some minimum and maximum value.

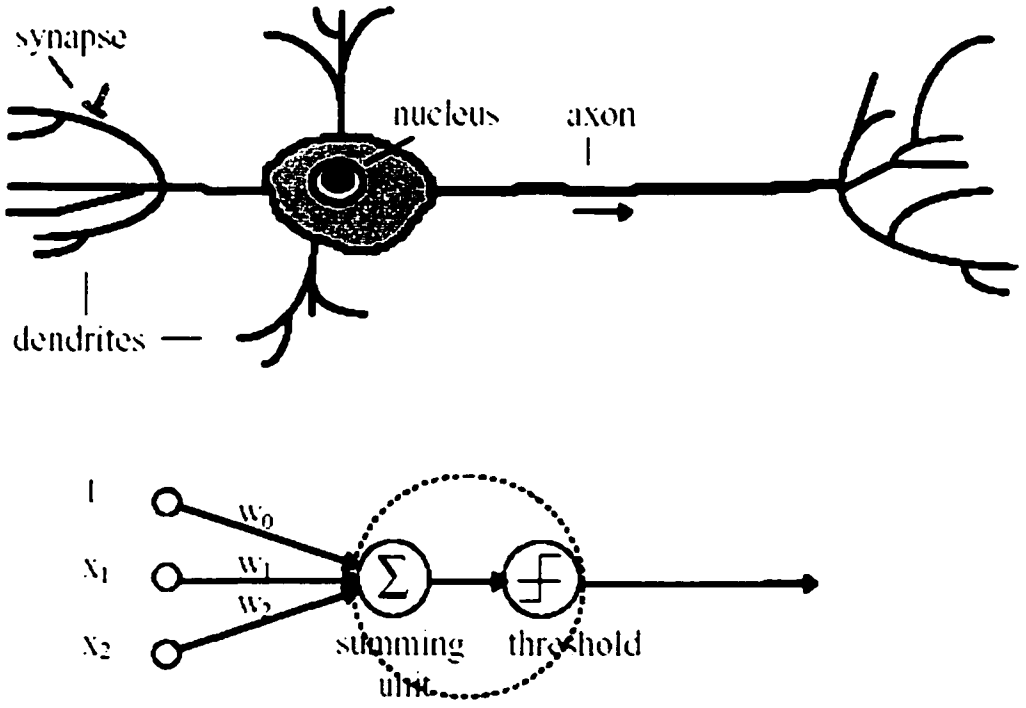

Figure 5.1: Simple illustration of biological and artificial neuron (percepton)

In artificial neural networks the inputs of the neuron are combined in a linear way with different weights [47-58]. The result of this combination is then fed into a non-linear

activation unit (activation function), which can in its simplest form be a threshold unit Figure 5.1.

Neural networks offer nonlinearity, input-output mapping, adaptivity and fault tolerance. Nonlinearity is a desired property if the generator of input signal is inherently nonlinear [Haykin, 1994]. The high connectivity of the network ensures that the influence of errors in a few terms will be minor, which ideally gives a high fault tolerance.

The dynamics models of robots based on the Langrange-Euler formation are very simple in its representation but are very difficult, time consuming and error prone to obtain except for simple cases [28]. The modeling process, in general, starts from the assignments of the inertial reference co-ordinate system and the moving relative co-ordinate system, formulates the transformation matrices and Lagrangrian functions, and concludes at the derivation of the Lagrange-Euler equations.

The majority of neural network [47-58] only have a static mapping capability. However, when they are used as controllers, they must be able to realize the dynamics. Thus, dynamic neural networks may be needed to fully model the dynamics of the system. Despite the fact that neural networks are powerful in learning complicated dynamics, the size of the networks (in terms of nodes, or weights) can be very large which subsequently leads to the need of powerful computational facilities for learning the adjustable parameters.

While it is true that the neural networks [28] have capability and can be used to approximate any dynamic function to any desired accuracy as long as the size of the networks are large enough, we should not abuse their capabilities and let them learn all the characteristics of the systems without any discrimination. Allowing a neural network to have many degrees of freedom (or flexibility) means that it will not only learn the desired dynamics, but also the inevitable noise leading to poor generalization. Otherwise, a construction algorithm can be used to minimize the number of degrees of freedom of the network. This is particularly important if network parameters are linear such as a Radial Basis Function (RBF) network is employed, as it suffers from the curse of dimensionality. It has been shown that, by fully exploiting the structural properties of a particular system, the full dynamics of the system can be approximated by static neural networks of predetermined structures. As a result, the size of the neural network model of robots becomes smaller than the usual dynamical ones. This is very desirable for actual implementation in terms of less computational power required, and accordingly, less costly.

The dynamics of robots can be expressed in task or Cartesian space. Since the task specifications are usually given relative to the end-effecter, it is to attempt to derive control algorithms directly in task space rather than in joint space. Dynamic neural networks modeling of robots controller in task space can be easily carried out without the exact knowledge of the regressor of the dynamics parameters of the robot.

Although neural networks are well-known universal approximators if the size of the network is sufficiently large, the underlying approximation nature of neural networks introduces approximation errors which must be taken into account. No matter how large the size of the neural network is, the modeling and generalization error always exists. Furthermore, the design problems like the number of layers; the choice of network architectures and basis functions; the number of nodes and others, may be difficult to determine for specific systems, unless some off-line construction algorithm employing non-linear optimization is used. Another concept of structural network modeling using the system's own functions known as parametric network modeling has been introduced to eliminate the approximation nature of neural networks, where the maximum size of the parametric networks is fixed.

5.2 Neural Network Approximations

Neural networks consist of a large number of simple processing elements called nodes. The nodes are introduced by weighted links where weights are the network's adjustable parameters. The arrangement of the nodes and the interconnections define the architecture of the neural networks [28]. However, the capabilities of different networks vary with different kinds of applications. Thus, a careful choice in the structure of the neural network has to be made for specific application.

In control engineering, a neural network is usually used to generate input/output maps using the property that a multi-layer neural network can approximate any function with any desired accuracy. The approximation problem [28] can be stated formally as follows:

Definition 5.1 (Function Approximation). If $f(x):\Re^n \to \Re^m$ is a continuous function defined on a compact set Ω, and $y(W,x):\Re^r \times \Re^n \to \Re^m$ is an approximation function that depends continuously on weight vector W and x, then, the approximation problem is to determine the optimal weight vector W^*, for some metric (or distance function) d, such that

$$d(y(W^*,x),f(x)) \le \varepsilon \tag{5.1}$$

for an acceptable small ε.

As a function emulator, firstly, an approximating function $y(W,x)$ for $f(x)$ is chosen, then, an algorithm is used to adjust the neural network weights based on the output error, by a training set. Hence there are two distinct problems in function approximation;

namely, the representation problem which deals with the selection of the approximation function $y(W,x)$ and the learning problem which is to find the training method to ensure the optimal parameters W^* for a given choice of $y(W,x)$ are obtained.

In the literature, quite a number of neural networks function approximation have been used, such as the multi-layer perceptron (MLP) networks, radial basis function (RBF) networks, and higher order neural networks (HONNs).

The MLP network is the most widely studied neural network structure. It is a feed-forward network where the input signals propogating forward through several processing layers before the network output is calculated. Figure 5.2 shows the schematic diagram of a MLP network with two hidden layers. Each node of the MLP network has input connections with the nodes of the previous layer only, and the output of each node is in turn applied to the next layer. In other words, all the nodes of one layer are connected to each of the nodes of the next layer, thus MLP network is a fully connected network. The hidden nodes and output does contain an appropriate activation function or transfer function which calculates the node's output from the weighted input signals. Sigmoidal and hyperbolic tangent functions are the typical choices for the activation functions in the hidden nodes. Whereas in the output nodes, a linear function is normally employed. It has been established that any continuous non-linear function is uniformly approximated to within an arbitrary accuracy by a MLP network is very flexible and can be employed in a wide variety of modeling and control tasks.

However, the MLP network suffers from several limitations. Firstly, the so-called back-propogation algorithm which is commonly used for the weight adjustment, also referred to as training in the neural network community, can not guarantee to converge and may take an undesirable long period of time to converge. This difficulty in training is not desirable for any type of on-line or real-time approximation. Secondly, although the MLP network has universal approximation capabilities, no method exists to choose the network structure necessary to achieve the desired approximation accuracy. Thirdly, the MLP network is often referred to as a non-linearly parameterized network, which means that network output is related to the adjustable weights in a non-linear fashion.

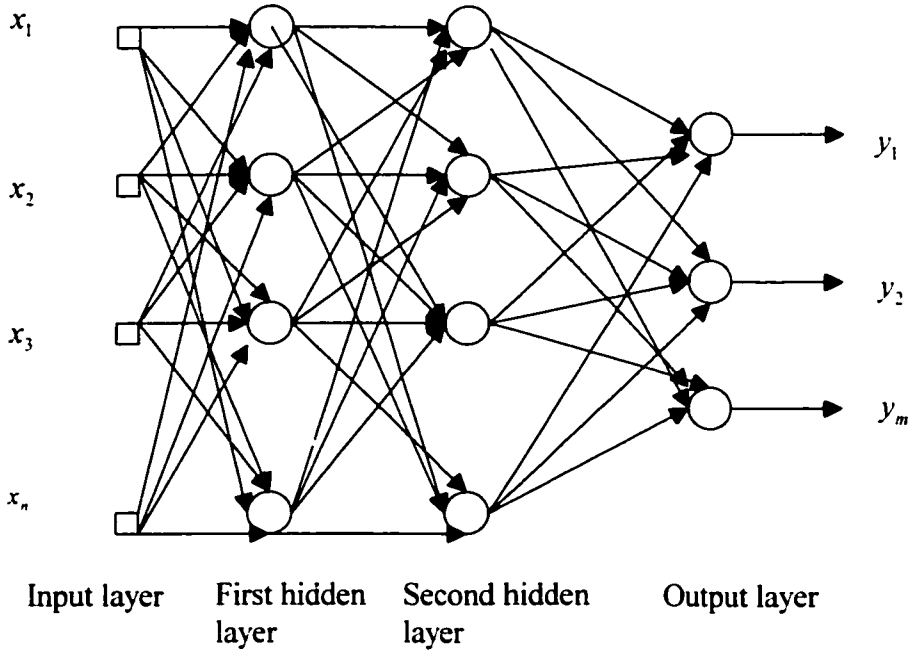

Figure 5.2: A multi-layer percepton (MLP) network

This property often makes analysis of systems containing MLP network difficult or impossible. Finally, MLP network tends to forget old information quickly, i.e. when presented with new data, small adjustment to the weights often cause bad approximation of data on which the network was previously trained.

A three-layer feed forward neural network of a different structure is shown schematically in Figure 5.3, where $x \in \Re^n$, $y \in \Re^m$, $a \in \Re^l$ are the input, the output, and the activation function vectors respectively, with v_{ij} as the first-to-second layer interconnection weights, and w_{jk} as the second-to-third layer interconnection weights. The input to the activation function z_i, and the output of the neural network are given by

$$z_i = \sum_{j=1}^{n} v_{ij} x_j, i = 1,2,\cdots,l \tag{5.2}$$

$$y_j = \sum_{j=1}^{n} w_{jk} a_k(z), j = 1,2,\cdots,m \tag{5.3}$$

Where $z = [z_1 \cdots z_k]^T$. Note that the bias (threshold offset) can be incorporated easily by augmenting the input variable by 1. Any tuning of the neural network weights would then include the tuning of the bias as well.

The neural network equations (5.2)-(5.3) can be conveniently expressed in a matrix format as

$$z(x) = V^T x$$

$$y(z) = W^T a(z)$$

where $W^T = [w_{jk}]^T, V^T = [v_{ij}]^T$. Thus, the output of the network can be simply expressed as

$$y(x) = W^T a(V^T x) \tag{5.4}$$

A general function $f(x) \in \Re^m$ can be approximated as

$$f(x) = W^T a(V^T x) + \varepsilon(x) \tag{5.5}$$

Where $\varepsilon(x)$ is the neural network functional reconstruction error. If there exist integers l

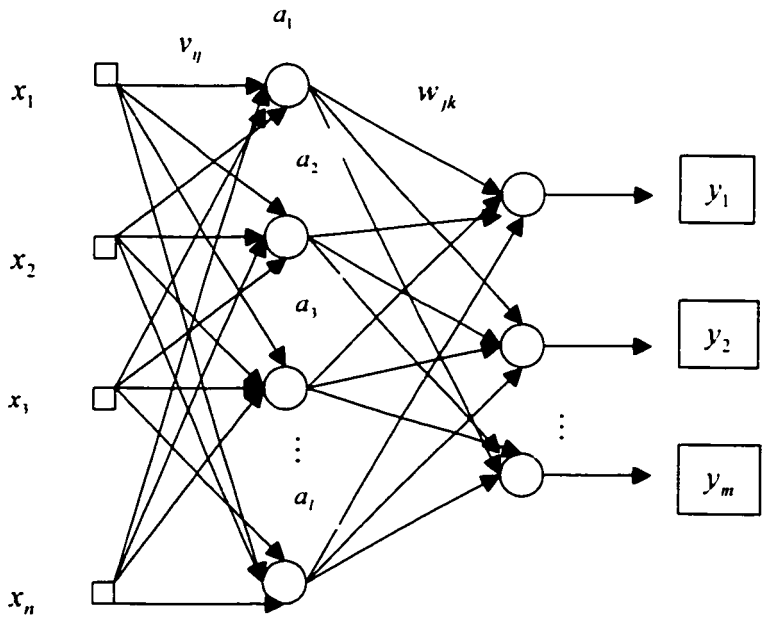

Figure 5.3: Three layer neural network (RBF network if $V = I$)

and m, and constants W and V, I is the identity matrix so that $\varepsilon = 0$, $f(x)$ is said in the functional range of the neural network. It is well known that any sufficiently smooth function can be approximated by a suitably large network using various activation functions, $a(.)$, to any desired accuracy over a compact set based on the Stone-Weiersrtass theorem. Typical choices include the sigmoid functions the hyperbolic tangent, radial basis functions (RBF), etc. whilst it is true that multi-layer neural networks can approximate systems with fairly severe non-linearities, it is difficult and complicated to carry out stability analysis, on-line tuning and actual implementation. In most cases, the results are not as good as linear in the parameters networks such as the radial basis function networks and high order networks.

The RBF network can be represented in a three-layer structure. The first layer is the input layer which consists of the source nodes. The second layer is a hidden layer, composed of computation nodes that are connected directly to all of the nodes in the input layer. Each hidden node contains a parameter vector called a center. The node calculates the Euclidean distance between the center and the network input vector, and passes the results through a non-linear activation function. The third layer is the output layer which performs the summation of the weighted outputs from the hidden layers.

The RBF networks have some useful properties which render them suitable for on-line non-linear adaptive modeling and control. First, they belong to a class of linearly parameterized networks where the network output is related to the adjustable network weights in a linear manner, assuming that the basis function centers and variances are

fixed a priori. Thus on-line learning rules can be used to update the weights and convergence results can be derived. Second, the activation functions of the RBF network are localized, thus theses networks store information locally in a transparent fashion. The adaptation in one part of the input space does not affect knowledge stored in a different area, i.e., they have spatially localized learning capability. This implies that they have better memory than MLP networks. Third, it exhibits a fast initial rate of learning convergence, this is because the network output is linearly dependent on the adjustable weight vector.

When $V = I$, the three feedward neural network in Figure 5.3 becomes a RBF network. The output of the network can be expressed as

$$y_k = \sum_{j=1}^{l} w_{jk} a_j \left(\left\| x - \mu_j \right\|^2 \right) \tag{5.6}$$

where l is the number of hidden nodes, $x \in \Re^n$ is the network input vector, $\|\cdot\|$ denotes the standard Euclidean norm, $a_j(.)$ is the activation function, $\mu_j \in \Re^n$ is the center vector, and $w_{ij} \in \Re$ is the network weight. This can be conveniently expressed in a matrix form as

$$y(x) = W^T a(x) \tag{5.7}$$

A list of function can be used as radial basis functions, such as Gaussian, Hardy's multiquadratic and Inverse Hardy's multiquadratic

(i) Gaussian RBFs

$$a(x) = \exp\left[\frac{-(x-\mu_i)^T(x-\mu_i)}{\sigma_i^2}\right] \tag{5.8}$$

(ii) Harady's multiquadratic RBFs

$$a_i(x) = \sqrt{\sigma_i^2 + (x-\mu_i)^T(x-\mu_i)} \tag{5.9}$$

(iii) Invers Harady's multiquadratic RBFs

$$a_i(x) = \frac{1}{\sqrt{\sigma_i^2 + (x-\mu_i)^T(x-\mu_i)}} \tag{5.10}$$

Gaussian radial basis functions have some attractive properties: (i) they are bounded, strictly positive and absolutely integrable on $\Re^n$, (ii) their Fourier transforms are their own, and (iii) they are time-frequency localized. Because of these nice properties, it has been widely used for the control of non-linear systems.

Gaussian radial functions do possess a localized response, they are not strictly compact, but they are compact in the frequency domain. A major disadvantage of Gaussian for RBFs is that the membership functions do not produce a partition of unity, and hence the normalization of the membership function is liable to change their shape producing unexpected results such that non-local weights. Neural fuzzy networks are not only compact, automatically form a partition of unity, but also have a linguistic interpolation for fuzzy rules. As with other linear in the adjustable weight networks and neurofuzzy networks are well conditioned, have provable learning characteristics, and are temporally stable, i.e. do not unlearn past knowledge.

CHAPTER 6

DIRECT ADAPTIVE CONTROL USING FUZZY SYSTEMS AND NEURAL NETWORKS

6.1 Equivalence Between the Neural Networks And Fuzzy Logic Systems

Recall a radial basis function neural network shown in Figure 6.1. There, the inputs are $x_i, i = 1,2,\cdots,n$, where n is the number of the inputs. Let $x = [x_1, x_2, \cdots, x_n]^T$ and the outputs is $y = f(x)$ where f represents the processing by the entire radial basis function neural network [35] [74]. The input to the i^{th} receptive field unit is x, and its output is denoted with $R_i(x)$. It has what is called a strength (or a weight) which we denote by $\bar{y}_i$. Assume that there are M receptive field units. Hence, from Figure 6.1,

$$y = f(x) = \sum_{i=1}^{M} \bar{y}_i R_i(x) \tag{6.1}$$

is the output of the radial basis function neural network.

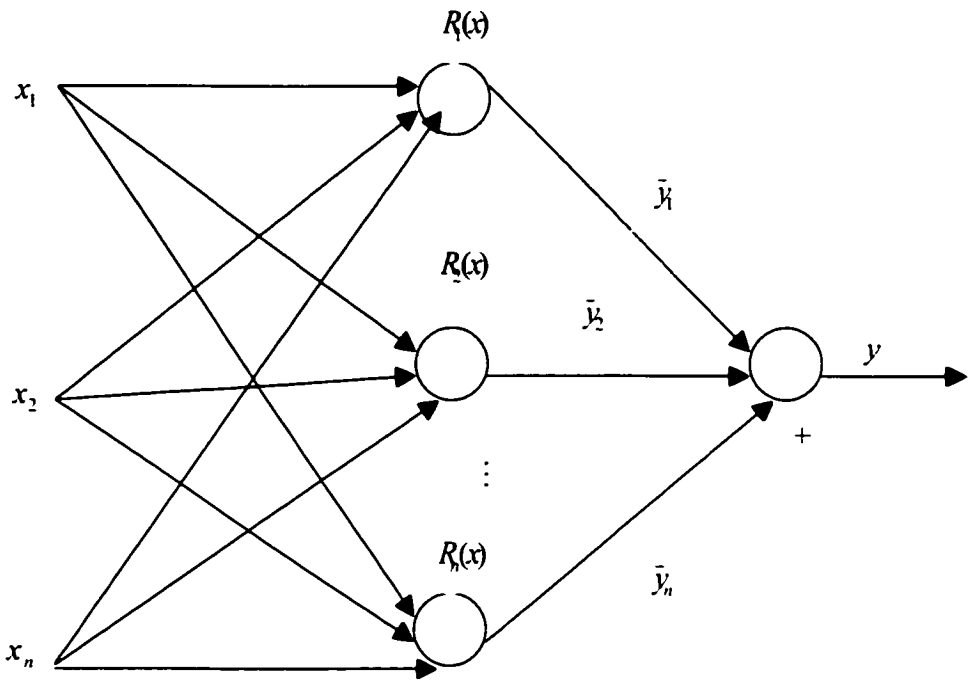

Figure 6.1: Radial basis function neural network model

There are several possible choices for the receptive field units $R_i(x)$:

Because of its attractive properties the Gaussian baiss function could be choosen as discussed before in section 5.2,

$$R_i(x) = \exp\left(-\frac{|x - \underline{c}_i|}{\sigma_i^2}\right)$$

where $\underline{c}_i = [c_1^i, c_2^i, \cdots, c_n^i]$, σ_i is scalar, and if z is a vector then $|z| = \sqrt{z^T z}$.

There are also alternatives to how to compute the output of the radial basis function neural network. For instance, rather than computing the simple sum as in Equation (6.1) one could compute a weighted average

$$y = f(x) = \frac{\sum_{i=1}^{M} \bar{y}_i R_i(x)}{\sum_{i=1}^{M} R_i(x)} \tag{6.2}$$

From Equation (6.2) it seems some radial basis function neural networks are equivalent to some standard fuzzy systems in the sense that they are functionally equivalent (i.e., given the same inputs, they will produce the same outputs). To show this, suppose that in Equation (6.2) the number of receptive field units equal to the number of rules, that is $M = R$, the receptive field unit strengths equal to the output membership function centers, that is $\bar{y}_i = b_i$ and choose the receptive field units as

$$R_i(x) = \mu_i(x) \tag{6.3}$$

(i.e., choose the receptive field units to be the same as the premise membership functions). In this case we see that the radial basis function neural network is identical to a certain fuzzy system that uses center-average defuzzification. This fuzzy system is then given by

$$y = f(x) = \frac{\sum_{i=1}^{M} \bar{y}_i R_i(x)}{\sum_{i=1}^{M} R_i(x)} = \frac{\sum_{i=1}^{R} b_i \mu_i(x)}{\sum_{i=1}^{R} \mu_i(x)} \tag{6.4}$$

it is also interesting to note that the functional fuzzy system (the more general version of the Takagi-Sugeno fuzzy system) is equivalent to a class of two-layer neural networks.

The equivalence between this type of fuzzy system and radial basis function neural network shows that all the techniques work in the same way for both of them. Due to the

above relationship between fuzzy systems and neural networks, some would like to view fuzzy systems and neural network as identical subject areas.

Regardless of the differences, it is important to note that many control methods (i.e., when we use a neural network for the control of a system) are quite similar to those in adaptive fuzzy control. For instance, since fuzzy system and radial basis function neural network can be linearly parameterized, we can use them as identifier structures in direct or indirect adaptive control schemes and use gradient or least squares methods to update the parameters.

Since the most important advantage of neural networks is that they have the capability to approximate nonlinear mappings as given in section 5.2. The following theorem which has bee proved in [74] shows that the fuzzy logic system (FLS) Equation (4.5) with Gaussian membership function in Table 4.3 is also capable of uniformly approximating any real continuous nonlinear function over X the universe of discourse to any degree of accuracy if X is compact. Let Y be the set of all fuzzy basis function expansions Equation (4.5) with Gaussian membership function, and $d_\infty(f_1, f_2) = \sup_{x \in X} |f_1(x) - f_2(x)|$ be the sup-metric: then (Y, d_∞) is a metric space.

Theorem 1: (Universal Approximation Theorem [74]): For any given real continious function f on a compact set $X \subset \Re^n$ and arbitrary $\varepsilon > 0$, there exists a FLS f^* in the form of Equation (6.2) with Gaussian membership function such that $d_\infty(f^*, f_2) = \sup_{x \in X} |f^*(x \mid \varphi) - f_2(x)| < \varepsilon$.

This theorem, theoretically establishes the equivalence between the fuzzy systems and neural networks, allows us for the use of neural networks in which a single hidden layer of radial-basis functions are used or if a special form of two hidden layers used. Figures 6.1 and 6.2 demonstrate these two cases. With a single hidden layer of the radial basis functions the output of neural network is given by

$$\tilde{y} = c^T \zeta \tag{6.5}$$

where $\zeta \in \Re^P$ are (possibly normalized) radial-based functions (e.g., squashing functions characterized By Guassian functions [29-30]) and C is a vector of adapting weights. This type of system may be described by (4.9) with $Z = [1]$ and $A = c^T$. As it is well recognized in the literature, this is exactly the same representation as used with standard fuzzy systems.

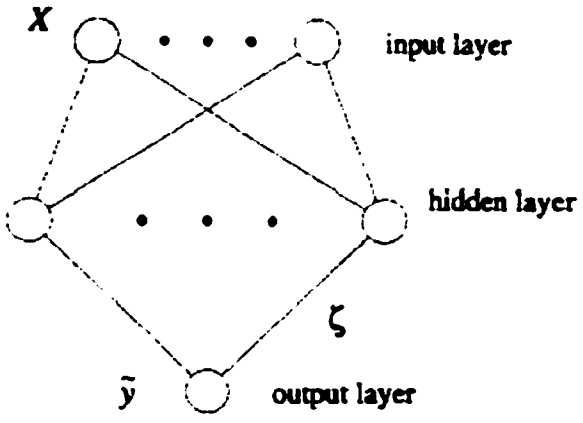

Figure 6.2: Simple neural network

Example 6.1: Consider a two-link robot arm where the purpose is to design a neural contoller. If a simple neural network is considered then

$$Z=[1],\ A=c^T=\begin{bmatrix} a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ a_{4,0} & a_{4,1} & a_{4,2} & a_{4,3} \end{bmatrix},\ \zeta^T=[\mu_1 \quad \mu_2 \quad \mu_3 \quad \mu_4]/\sum_{i=1}^{4}\mu_i$$

and $\mu_i(x)=\exp\left[\dfrac{-(x_i-c_i)^T(x_i-c_i)}{\sigma_i^2}\right],\ i=1,\ldots,4$ where x_1 is link 1 angle, x_2 is link 1 angular velocity, x_3 is link 2 angle, x_4 is link 2 angular velocity and c_i is the center of gravity of the Guassian RBF. The output is then given by $\tilde{y}=c^T\zeta$

A second type of neural network considered in this work is one in which there are two hidden layers with the second hidden layer of a special form. The output of the first hidden layer produces a vector of functions

$$\tilde{z}=[\theta_1 \cdots \theta_m]. \tag{6.6}$$

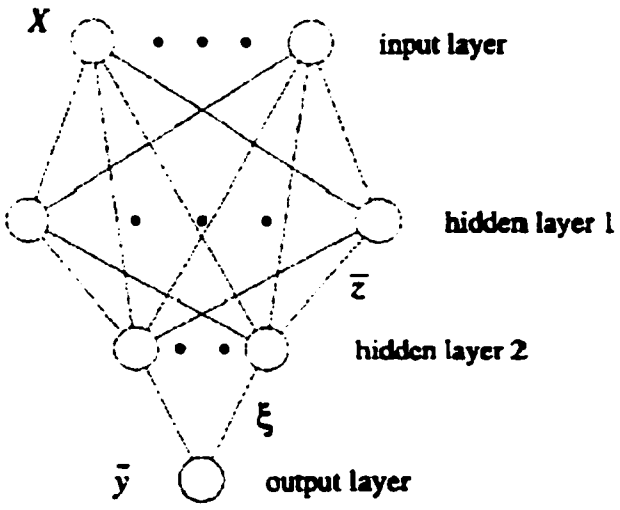

Figure 6.3: Neural networks with 2 hidden layers

The nodes which make up the first hidden layer may be normalized radial-basis function [29-30]. Here, we allow both the output of the first hidden layer and the original: input to be passed to the second hidden layer (Figure 6.3). The output of the ith node of the second hidden layer is given by

$$\xi_i = \zeta_i(\tilde{z}, X)\left(b_{i,0} + \sum_{j=1}^{m} b_{i,j}\theta_j + \sum_{j=1}^{n} b_{i,j+m}x_j\right) \tag{6.7}$$

where $\zeta_i(\tilde{z}, X)$ are squashing functions or radial-basis functions (which may be normalized) and $b_{i,0}$ is the basis for the ith node. The output of the neural network is taken as a linear combination of the outputs of the second hidden layer, that is

$$\tilde{y} = \sum_{j=1}^{p} c_i\xi_i \tag{6.8}$$

We may combine (6.7) and (6.8) to obtain

$$\tilde{y} = \zeta_i(z, X)\left(a_{i,0} + \sum_{j=1}^{m} a_{i,j}\theta_j + \sum_{j=1}^{n} a_{i,j+m}x_j\right) \tag{6.9}$$

Which may be expressed in the form of (6.8) with $z = [1 \quad \theta_1 \quad \cdots \quad \theta_m x_1 \cdots \quad x_n \quad]^T$ and $A = [a_{i,j}]$ with $a_{i,j} = c_i b_{i,j}$ note that z may or may not include any θ_i or x_i.

Example 6.2: For the two link robot arm if we consider two layers neuro network then we can take $z = [1 \quad x_1 \quad x_2 \quad x_3 \quad x_4]^T$, $A = \begin{bmatrix} c_1b_{1,0} & c_1b_{1,1} & c_1b_{1,2} & c_1b_{1,3} \\ c_2b_{2,0} & c_2b_{2,1} & c_2b_{2,2} & c_2b_{2,3} \\ c_3b_{3,0} & c_3b_{3,1} & c_3b_{3,2} & c_3b_{3,3} \\ c_4b_{4,0} & c_4b_{4,1} & c_4b_{4,2} & c_4b_{4,3} \end{bmatrix}$, $\theta_i = 1$.

$$\zeta^T = [\mu_1 \quad \mu_2 \quad \mu_3 \quad \mu_4]/\sum_{i=1}^{4}\mu_i \text{ .and} \qquad \mu_i(x) = \exp\left[\frac{-(x_i - c_i)^T(x_i - c_i)}{\sigma_i^2}\right], \quad i = 1,...,4$$

where x_1 is link 1 angle, x_2 is link 1 angular velocity, x_3 is link 2 angle, x_4 is link 2 angular velocity and c_i is the center of gravity of the Guassian RBF. The output is then given by $\tilde{y} = \zeta_i(z, X)\left(a_{i,0} + \sum_{j=1}^{4} a_{i,j}\theta_j + \sum_{j=1}^{4} a_{i,j+m}x_j\right)$.

6.2 Derivation of Adaptive Control Law

The objective is to design a control system which will cause the output of a relative degree r plant, y_p, to track a desired output trajectory, y_m, (a relative degree r plant is one in which the plant input appears in the output dynamics after r differentiation of the output) [29-34]. This controller will be used later to control the Pendubot in the vertical position. The desired output trajectory may be a signal to the control system so that first derivative of y_m may be measured, or by a reference model, with relative degree greater than or equal to r which characterize the desired performance as shown in Figure (6.4). With these considerations, we make the following assumption about the signal [let $y^{(r)}_m$ denote the r th derivative of y_m with respect to time].

R1) Reference Input Assumption: The desired output trajectory and its derivative $y_m, \ldots, y^{(r)}_m$ are measurable and bounded.

Here, we consider the SISO plant

$$\dot{X} = f(X) + g(X)u_p \tag{6.10}$$

$$y_p = h(X) \tag{6.11}$$

where $X \in \Re^n$ is the state vector, $u_p \in \Re$ is the input, $y_p \in \Re$ is the output of the plant and functions $f(X)$, $g(X) \in \Re^n$ and $h(X) \in \Re$ are smooth. If the system has "strong relative degree" r then

$$\dot{\xi}_1 = \xi_2 = L_f h(\mathrm{X})$$

$$\vdots$$

$$\dot{\xi}_{r-1} = \xi_r = L_f^{r-1} h(\mathrm{X}) \tag{6.12}$$

$$\dot{\xi}_r = L_f^r h(\mathrm{X}) + L_g L_f^{r-1} h(\mathrm{X}) u_p$$

with $\xi_1 = y_p$ which may be rewritten as

$$y_p^{(r)} = [\alpha_k(t) + \alpha(\mathrm{X})] + [\beta_k(t) + \beta(\mathrm{X})] u_p \tag{6.13}$$

where $L_g^r h(\mathrm{X})$ is the rth Lie derivative of $h(\mathrm{X})$ with respect to g

$\{ L_g h(\mathrm{X}) = (\partial h / \partial \mathrm{X}) g(\mathrm{X})$ and $L_g^2 h(\mathrm{X}) = L_g [L_g h(\mathrm{X})] \}$: and it is assumed that for some $\beta_0 > 0$, we have $|\beta_k(t) + \beta(\mathrm{X})| \geq \beta_0$ so that it is bounded away from zero (for convenience we assume $\beta_k(t) + \beta(\mathrm{X}) > 0$), however , the following analysis may easily be modified for systems which are defined with $\beta_k(t) + \beta(\mathrm{X}) < 0$). We will also assume that $\alpha_k(t)$ and $\beta_k(t)$ are known components of the dynamics of the plant (that may depend on the state) or known exogenous time dependent signals and which represent nonlinear dynamics of the plant that are unknown. It is reasonable to assume that if X is a bounded state vector, then α_k and $\beta_k(t)$ are bounded signals. Throughout the analysis to follow, both $\alpha_k(t)$ and $\beta_k(t)$ may be set to zero for all $t \geq 0$.

The functions $\alpha_k(t)$ and $\beta_k(t)$ shall be approximated with fuzzy systems or neural network based on the equivalence established in section 6.1.

$$\widetilde{y}_\alpha = \widetilde{f}_\alpha(\mathrm{X}) = z_\alpha^T A_\alpha \xi_\alpha \tag{6.14}$$

The dynamics for a relative degree plant described by (6.12) may be written in normal form as

$$\dot{\xi}_1 = \xi_2$$
$$\vdots \qquad \vdots$$
$$\dot{\xi}_{r-1} = \xi_r$$
$$\dot{\xi}_r = \alpha(\xi, \pi) + \beta(\xi, \pi) u_p \tag{6.15a}$$
$$\dot{\pi} = \psi(\xi, \pi)$$

With $\pi \in \Re^{n-r}$ and $y_p = \xi_1$.

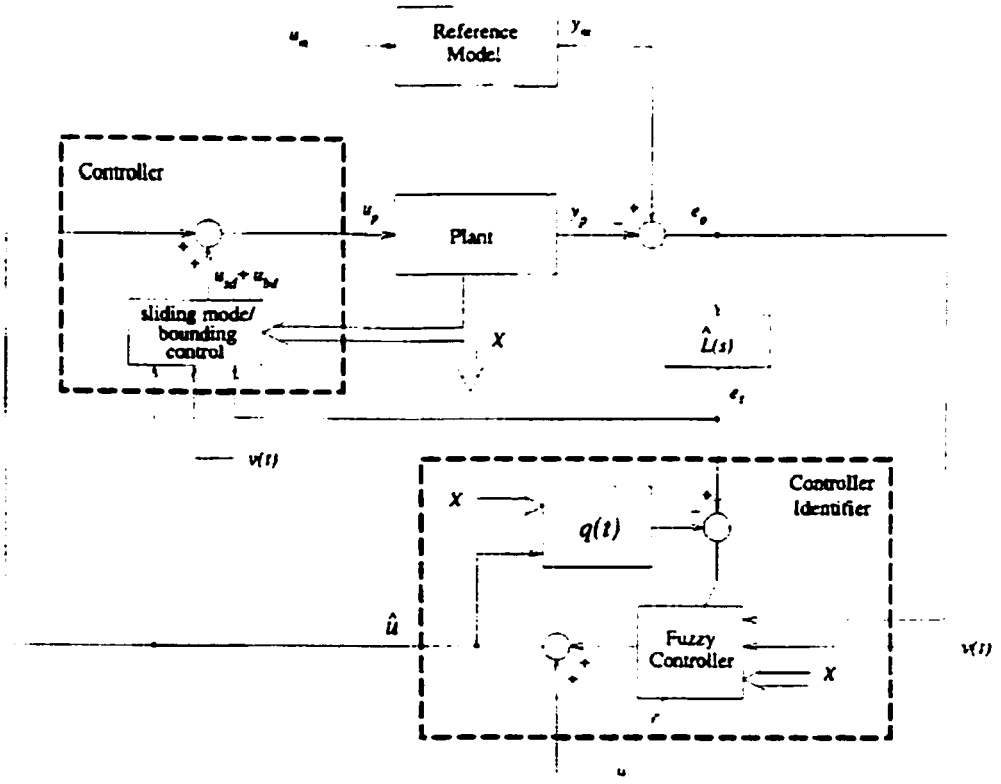

Figure 6.4: Adaptive neural fuzzy control

The zero dynamics of the system are given as

$$\dot{\pi} = \psi(0, \pi). \tag{6.15b}$$

Now consider the adaptive control of plants without zero dynamics, or plants which have exponential attractively of the zero dynamics (i.e., plants where (6.15b) is exponentially stable when the states move outside a ball $|\pi| > \beta$). The two plant types are characterized by the following assumptions.

P1) Plant Assumption: The plant is relative degree $r = n$ (i.e., no zero dynamics) such that

$$\frac{d}{dt} x_i = x_{i+1} \qquad i = 1, \cdots, n-1 \tag{6.16}$$

$$\frac{d}{dt} x_n = [\alpha_k(t) + \alpha(\mathrm{X})] + [\beta_k(t) + \beta(\mathrm{X})] u_p \tag{6.17}$$

Where $y_p = x_1$ with $\alpha_k(t)$ and $\beta_k(t)$ known functions. Here, it is assumed that there exists $\beta_0 > 0$ such that $\beta_k(t) + \beta(\mathrm{X}) \geq \beta_0$ and that $x_1, \cdots, x_n$ are measurable.

P2) Plant Assumption: The plant is of relative degree r, $1 \leq r < n$ with the zero dynamics exponentially attractive and there exists $\beta_0 > 0$ such that $\beta_k(t) + \beta(\mathrm{X}) \geq \beta_0$. The outputs $y_p, \cdots, y^{(r-1)}{}_p$ are measurable.

Clearly plants satisfying P1) have bounded states if the reference input y_m and its derivatives are bounded with the error e_o and its derivatives bounded. By using Lipschitz properties of $\psi(\xi, \pi)$ to see the plants satisfying P2) have bounded states if the output is

bounded in the following manner [30]. For some positive constants $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ and function B and function v_1 we have

$$\gamma_1|\pi|^2 \le v_1(\pi) \le \gamma_2|\pi|^2 \tag{6.18}$$

$$\frac{dv_1}{d\pi}\Psi(0,\pi) \le -\gamma_3|\pi|^2 \text{ , if } |\pi| > B \tag{6.19}$$

$$\left|\frac{dv_1}{d\pi}\right| \le \gamma_4|\pi| \tag{6.20}$$

if the zero dynamics are exponentially attractive. Since reference signals are bounded, by R1, $|\xi| \le k_1$ where k_1 is some positive constant. Using (6.20), we have

$$\dot{v}_1 = \frac{dv_1}{d\pi}\psi(\xi,\pi) \tag{6.21}$$

$$\le -\gamma_3|\pi|^2 + \frac{dv_1}{d\pi}[\psi(\xi,\pi) - \psi(0,\pi)] \quad \text{if } |\pi| > B \tag{6.22}$$

If $\Psi(\xi,\pi)$ is Lipschitz in ξ then $|\psi(\xi,\pi)-\psi(0,\pi)| \le k_2|\xi|$ some positive k_2. Using this, if $|\pi| > B$, then

$$\dot{v}_1 \le -\gamma_3|\pi|^2 + \left|\frac{dv_1}{d\pi}\right|[|\psi(\xi,\pi) - \psi(0,\pi)| \tag{6.23}$$

$$\le -\gamma_3|\pi|^2 + \gamma_4 k_2|\xi||\pi|$$

$$\le -\gamma_3|\pi|^2 + \gamma_4 k_2 k_1|\pi| \text{ .} \tag{6.24}$$

Therefore $\dot{v}_1 \le 0$, if $|\pi| \ge \max(B, \gamma_4 k_1 k_2 / \gamma_3)$. This ensures the boundedness of ξ and π therefore the system states are bounded.

P3) Plant Assumption:

Given $y_p^{(r)} = [\alpha_k(t) + \alpha(\mathrm{X})] + [\beta_k(t) + \beta(\mathrm{X})]u_p$ (6.25)

It is required that $\beta_k(t) = 0, t \geq 0$ and there exists positive constants β_0 and β_1 such that $0 < \beta_0 \leq \beta(\mathrm{X}) \leq \beta_1 < \infty$ and some $\beta(\mathrm{X}) \geq 0$ function such that $\left|\dot{\beta}(\mathrm{X})\right| = \left|(\partial\beta / \partial \mathrm{X})\dot{\mathrm{X}}\right| \leq B(\mathrm{X})$ for all $\mathrm{X} \in S_Y$. Here, as earlier, $\alpha_K(t)$ is a known time-dependent signal.

The first part of P3) introduces a new requirement that the controller gain $\beta(\mathrm{X})$ be bounded from the above by a constant β_1. In general, this will not pose a large restriction upon $y_p^{(r)}$ the class of plants since situations in which finite input will cause an infinitely large effect upon rarely occur in physical plants. The second restriction with in P3) requires that $\left|\dot{\beta}(\mathrm{X})\right| \leq B(\mathrm{X})$ for some. It is known that $\left|\dot{\beta}(\mathrm{X})\right| \leq \left\|\partial\beta(\mathrm{X}) / \partial \mathrm{X}\right\|\left\|\dot{\mathrm{X}}\right\|$ thus if $\left\|\partial\beta(\mathrm{X}) / \partial \mathrm{X}\right\|$ and $\left\|\dot{\mathrm{X}}\right\|$ are bounded, then some $B(\mathrm{X})$ may be found. Once again if one considers physical plants with finite controller gain, then $\left\|\partial\beta(\mathrm{X}) / \partial \mathrm{X}\right\|$ will be bounded if $y_p^{(i)}, i = 0, \cdots, r$ is bounded, then plants with no zero dynamics are ensured that $\left\|\dot{\mathrm{X}}\right\|$ is bounded since the sates may be written in terms of the outputs, $y_p^{(i)}, i = 0, \cdots, r-1$ .If a plant has zero dynamics, but $\beta(\mathrm{X})$ is not dependent upon the zero dynamics, then once again we have $\left|\dot{\beta}(\mathrm{X})\right|$ bounded.

Using feedback linearization [29], we know that there exists some ideal controller

$$u^* = \frac{1}{\beta(\mathrm{X})}[-\alpha(\mathrm{X}) + v(t)] \tag{6.26}$$

where $v(t)$ is a free parameter. u^* may be expressed in terms of a Takagi-Sugeno fuzzy model , so that

$$u^* = Z_{t'}^T A_u^* \xi_u + u_k + d_u(\mathrm{X}) \tag{6.27}$$

where u_k is a known part of the controller (possibly a fuzzy, proportional integral derivative (PID), or some other type of controller). Define the ideal direct control parameters

$A_u^* \in \Re^{m_u \times p_u}$

$$A_u^* := \arg \min_{A_t \in \Omega_t} \left[\sup_{X \in S_1, v \in S_2} \left| Z_u^T A_u \xi_u - (u^* - u_k) \right| \right] \tag{6.28}$$

so that $d_u(\mathrm{X})$ is an approximation error which arises when u^* is represented by fuzzy system. Assume that $D_u(\mathrm{X}) \geq |d_u(\mathrm{X})|$, where $D_u(\mathrm{X})$ is a known bound on the error in representing the ideal controller with a fuzzy system. So if $|d_u(\mathrm{X})|$ is to be small, then our fuzzy controller will require X and v to be available, either through the input membership function or through $Z_{t'}^T$ .The fuzzy approximation of the desired control is

$$\hat{u} = Z_{t'}^T A_u^* \xi_u + u_k \tag{6.29}$$

where the matrix A_u is updated online. The parameter error matrix for the direct adaptive controller

$$\Phi_u(t) = A_u - A_u^* \tag{6.30}$$

is used to define the difference between the parameters of the current controller and the desire controller.

Consider the control law

$$u_p = \hat{u} + u_{sd} + u_{bd} \tag{6.31}$$

The direct adaptive control law is comprised of a bounding control term, u_{bd}, a sliding-mode control term u_{sd}, and an adaptive control term, $\hat{u}$. Here, define

$$v(t) := y_m^{(r)} + \eta e_s + \bar{e}_s - \alpha_k(t) \tag{6.32}$$

with e_s defined as $e_s := k^T e$ where $e := [e_o \quad \dot{e}_o \quad \cdots \quad e_o^{(r-1)}]^T$, $k := [k_1 \quad \cdots \quad k_{r-2} \quad 1]^T$ and $e_o := y_m - y_p$ $\bar{e}_s$ so that $e_s = [e_o \cdots e_o^{(r-1)}][k_o \cdots k_{r-1}]^T$, thus, and $\bar{e}_s = [\dot{e}_o \cdots e_o^{(r-1)}][k_o \cdots k_{r-2} 1]^T$. We pick the elements of k such that $\hat{L}(s) := s^{r-1} + k_{r-2}s^{r-2} + \cdots + k_1 s + k_0$ has its roots in the open left-half plane. The goal of the adaptive algorithm is to learn how to control the plant to drive the e_s to zero. Thus, e_s is a measure of the tracking error.

Using the control (6.31), the r th derivative of the output error becomes

$$e_o^{(r)} = y_m^{(r)} - \alpha(\mathrm{X}) - \alpha_k(t) - \beta(\mathrm{X})(\hat{u} + u_{sd} + u_{bd}). \tag{6.33}$$

Using the definition of u^* (6.27) we may arrange (6.33) so that

$$\begin{aligned} e_o^{(r)} &= y_m^{(r)} - \alpha(\mathrm{X}) - \alpha_k(t) - \beta(\mathrm{X})u^* - \beta(X)(\hat{u} - u^*) - \beta(X)(u_{sd} + u_{bd}) \\ &= -\eta e_s - \bar{e}_s - \beta(X)(\hat{u} - u^*) - \beta(X)(u_{sd} + u_{bd}). \end{aligned} \tag{6.34}$$

Alternatively (6.34) can be expressed as

$$\dot{e}_s + \eta e_s = -\beta(X)(\hat{u} - u^*) - \beta(X)(u_{sd} + u_{bd}) \tag{6.35}$$

in the next section the bounding control term u_{bd} for the direct adaptive controller will be defined.

6.2.1 Bounding Control

The bounding control for the direct adaptive controller is determined by considering

$$v_{bd} = \frac{1}{2} e_s^2 \tag{6.36}$$

Differentiate (6.36) and use (6.36) to obtain

$$\dot{v}_{bd} = -\eta e_s^2 - e_s \left[\beta(X)(\hat{u} - u^*) + \beta(X)(u_{sd} + u_{bd}) \right] \tag{6.37}$$

$$\leq -\eta e_s^2 - |e_s| \left[\beta(X)(|\hat{u}| - |u^*|) + \beta(X)|u_{sd}| \right] - \beta(X) u_{bd} e_s \tag{6.38}$$

u^* is not explicitly known, however, so the bounding controller will be implemented using $\alpha_1(X) \geq |\alpha(x)|$. Now choose

$$u_{bd} = \prod(t) k_{bd}(t) \operatorname{sgn}(e_s) \tag{6.39}$$

where $\prod(t)$ is as defined as

Let ε_M and M_e be fixed parameters such that $0 < \varepsilon_M \leq M_e$.

$$\prod(t) = \begin{cases} 1, & if \quad M_e \leq e_s \\ \dfrac{|e_s| + \varepsilon_m - M_e}{\varepsilon_M}, & if \quad M_e - \varepsilon_M \leq |e_s| < M_e \\ 0, & otherwise \end{cases} \tag{6.40}$$

and

$$\operatorname{sgn}(x) := \begin{cases} 1, & x > 0 \\ -1, & x < 0 \end{cases} \tag{6.41}$$

$$k_{bd}(t) = |\hat{u}| + |u_{sd}| + \frac{\alpha_1(X) + |v|}{\beta_0} \tag{6.42}$$

he bounding control is continuous and defined so that it is always used when $e_s \geq M_e$. Where the parameter M_e defines a bounded, closed subset of the error-state space within which the error is guaranteed to stay. It is required that there are known bounds $\beta_1(X) \geq |\beta(X)|$ and $\alpha_1(X) \geq |\alpha(X)|$ when $|e_s| \geq M_e$ with $\alpha_1(X)$ and $\beta_1(X)$ continuous in x. Using these state dependent bounds. Using Equations (6.36) with (6.48), then we get

$$\dot{v}_{bd} \leq -\eta e_s^2, \; |e_s| \geq M_e.$$

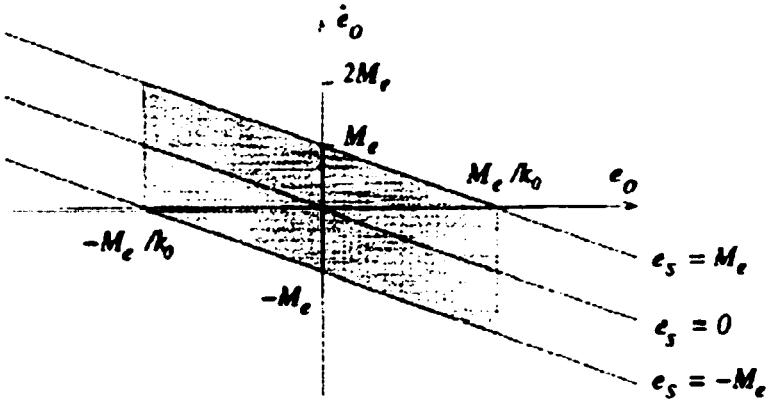

Figure 6.5: Boundedness around manifold $e_s = \dot{e}_0 + k_0 e_0 = 0$.

(Note that $|u^*| \le [\alpha_1(X) + |v|]/\beta_0$). With this definition of u_{bd}, if once again guarantee that the initial condition are such that $|e_1(0)| \le M_e$. Then plant assumption P1 or P2 are then required so that state boundness is guaranteed.

Thus, it is ensured that if there exists a time t' such that $|e_1(t')| > M_e$, then for $t > t'$, will decrease exponentially until $|e_1| \le M_e$.

At this point, it is convenient to define transfer functions

$$\hat{G}_i(s) := \frac{s^i}{\hat{L}(s)},\ i = 0, \cdots, r-1 \tag{6.43}$$

Which each are stable since $\hat{L}(s)$ has its poles in the open left half plane. Since $e_o^{(i)} = \hat{G}_i(s)e_1$ with e_1 bounded, then, $e_o^{(i)} \in \lambda_\infty$ $(\lambda_\infty = \{z(t) : \sup_t |z(t)| < \infty\}$. This is shown for the case $e_1 = \dot{e}_o + k_o e_o$ in Figure 6.5 where if $|e_1| \ge M_e$ then $\dot{e}_o$ and e_o stay in the shaded region (i.e., $|e_1| \le M_e / k_o$ and $|e_1| \le 2M_e$). This may be extended to higher dimensional systems as

$$\left|e_o^{(r-1)}\right| \le M_e \left\|\hat{G}_i(s)\right\|_1,\ i = 0, \cdots, r-2 \tag{6.44}$$

and since $e_o^{(r-1)} = e_1 \sum_{i=0}^{r-2} k_i e_o^{(i)}$ the triangular inequality may be used to show that

$$\left|e_o^{(r-1)}\right| \le M_e + M_e \sum_{i=0}^{r-2} k_i \left\|\hat{G}_i(s)\right\|_1 \tag{6.45}$$

for all time if $|e_1| \le M_e$ and $\left|e_o^{(i)}(0)\right| \le M_e \left\|\hat{G}_i(s)\right\|_1$, $i = 0, \cdots, r-2$.

The transfer function 1-norm is defined as $\left\|\hat{G}_t(s)\right\|_1 := \int_{-\infty}^{\infty} |g_t(t)| d\tau$. where $g_t(t)$ is the impulse response of $\hat{G}_t(s)$.Using for example of $e_v = \dot{e}_o + k_o e_o$. we obtain $\hat{G}_t(s) = \dfrac{1}{s + k_0}$ which has impulse response function of $g_0(t) = e^{-k_0 t}$ with the 1-norm $\left\|\hat{G}_t(s)\right\|_1 = 1/k_0$. using (6.44) and (6.45). then the bounds $|e_v| \le M_e / k_0$ and $|\dot{e}_o| \le 2M_e$ as shown in Figure 6.5. Over all Equations (6.44) and (6.45) provide explicit bounds on the output error when the bounding control is used u_{bd}.

6.2.2 Adaptation Algorithm

Consider the following Lyapunov equation candidate

$$V_d = \frac{1}{2\beta(X)} e_s^2 + \frac{1}{2} tr(\Phi_u^T Q_u \Phi_u) \tag{6.46}$$

where $Q_u \in \Re^{m_u \times m_u}$ is positive definite and diagonal. Since $0 < \beta_0 \le \beta(X) \le \beta_1 < \infty$, V_d is radially unbounded. The Lyapunov candidate V_d is used to describe both the error in tracking and the error between the desired controller and current controller if $V_d \to 0$, then both the tracking and learning objectives have been fulfilled. Taking the derivative of (6.46) yields

$$\dot{V}_d = \frac{e_s}{\beta(X)}[\dot{e}_s] + tr(\Phi_u^T Q_u \dot{\Phi}_u) - \frac{\dot{\beta}(X) e_s^2}{2\beta^2(X)} \tag{6.47}$$

Substituting $\dot{e}_s$, as defined in (6.35), yields

$$\dot{V}_d = \frac{e_s}{\beta(X)}\left[-\eta e_s - \beta(X)(\hat{u} - u^*)\right] - \beta(X)(u_{sd} + u_{bd}) + tr(\Phi_u^T Q_u \dot{\Phi}_u) - \frac{\dot{\beta}(X) e_s^2}{2\beta^2(X)} \tag{6.48}$$

Now consider the following fuzzy controller update law

$$\dot{A}_u(t) = Q_u^{-1} z_u \zeta_u^T [e_s - q(t)] \tag{6.49}$$

where $q(t)$ is a function yet to be defined. Since $\dot{\Phi}_u = \dot{A}_u$

$$\dot{V}_d = -\frac{\eta}{\beta(X)} e_s^2 - [z_u^T \Phi_u \zeta_u - d_u + u_{sd} + u_{bd}] e_s + tr(Z_u^T \Phi_u \xi_u)[e_s - q(t)] - \frac{\dot{\beta}(X) e_s^2}{2\beta^2(X)} \tag{6.50}$$

Equation (6.50) may be equivalently be expressed as

$$\dot{V}_d = -\frac{\eta}{\beta(X)} e_s^2 - q(t) z_u^T \Phi_u \zeta_u - [\frac{\dot{\beta}(X) e_s^2}{2\beta^2(X)} - d_u] e_s - e_s(u_{sd} + u_{bd}) \tag{6.51}$$

Typically, we will choose $q(t) = 0$, for all $t \geq 0$, however, we will later show how to incorporate information about the plant inverse dynamics so that $[q(t)] = \text{sgn}(z_u^T \Phi_u \zeta_u)$ to improve adaptation.

Using the fuzzy adaptation law defined by (6.49), we are not guaranteed that $A_u \in \Omega_u$. Once again we use a projection algorithm. The parameter space is defined so that the parameters are bounded by $A_u \in [A_u^{\min}, A_u^{\max}]$. Define to be the i th and j th element of the parameter matrix is updated according to

$$\dot{A}_u(t) = Q_u^{-1} \hat{A} \tag{6.52}$$

where the elements of $\hat{A}_u(t)$ are defined by

$$\hat{a}_{ui,j} = \begin{Bmatrix} 0, & if \quad a_{ui,j} \notin (A_u^{\min}, A_u^{\max}) and \quad \bar{a}_{ui,j}(a_{ui,j} - a_{ui,j}^c) > 0 \\ \bar{a}_{ui,j}, & otherwise \end{Bmatrix} \tag{6.53}$$

with $A_u^c \in (A_u^{\min}, A_u^{\max})$ .using this modified update law will ensure that the parameter matrices stay within the feasible parameter space and that

$$\dot{V}_d = -\frac{\eta}{\beta(X)} e_s^2 - q(t) z_u^T \Phi_u \zeta_u - [\frac{\dot{\beta}(X) e_s}{2\beta^2(X)} - d_u] e_s + e_s (u_{sd} + u_{bd}) \tag{6.54}$$

Since the modified adaptation law guides the searching algorithm toward the optimal parameters A_u^*.

6.2.3 Sliding Mode Control Term

Once again define a sliding-mode control term to compensate for the approximation error in modeling u^* by a fuzzy system or neural network. If $q(t) = 0$, for all $t \geq 0$ or $\text{sgn}[q(t)] = \text{sgn}(z_u^T \Phi_u \zeta_u)$ and u_{bd} is as defined in (6.39), then

$$\dot{V}_d \leq -\frac{\eta}{\beta(X)} e_s^2 - [\frac{\dot{\beta}(X) e_s}{2\beta^2(X)} - d_u] e_s - e_s u_{sd} \tag{6.55}$$

$$\leq -\frac{\eta}{\beta(X)} e_s^2 + [\frac{|\dot{\beta}(X)||e_s|}{2\beta^2(X)} + |d_u|] |e_s| - e_s u_{sd} \tag{6.56}$$

Now define the sliding-mode control term for the direct adaptive controller as

$$u_{sd} = k_{sd}(t)\,\text{sgn}(e_s) \tag{6.57}$$

Where

$$k_{sd}(t) = \frac{B(X)|e_s|}{2\beta_0^2} + D_u(X) \tag{6.58}$$

Which ensures $\dot{V}_d \leq -\eta e_s^2 / \beta_1$ that as long as we choose $q(t) = 0$, for all $t \geq 0$ or $\text{sgn}[q(t)] = \text{sgn}(z_u^T \Phi_u \zeta_u)$.

6.2.4 Stability Properties

The controller assumption for the direct control scheme is given as follows:

Control Assumption: The fuzzy systems (neural networks) are defined such that $D_u(X) \in \lambda_\infty$, for $X \in S_x \subseteq \Re^n$ and there are some known continuous functions $\alpha_1(X)$ and $\beta_1(X)$ such that $\alpha_1(X) \geq |\alpha(X)|$ and $\beta_1(X) \geq |\beta(X)|$. The function $q(t) = 0$ for all $t \geq 0$ or $\text{sgn}[q(t)] = \text{sgn}(z_u^T \Phi_u \zeta_u)$.

If reference input assumption R1) holds, either plant assumption P1) or P2) holds, plant assumption P3) holds and the plant control law is defined by Equation (6.35) with the control assumption. Then the following holds

a) The plant output and its derivatives $y_p, \cdots, y_p^{(r-1)}$ are bounded.

b) The control signals are bounded, i.e., $u_{bd}, u_{sd}, \hat{u} \in \lambda_\infty$.

c) The magnitude of the output error $|e_0|$ decreases at least asymptotically to zero, i.e., $\lim_{t \to \infty} |e_o| = 0$.

Part 1) Equation (6.34) and (6.35) guarantee that $|e_o^{(i)}| \in \lambda_\infty, i = 0, \cdots, r-1$, since $|e_o^{(i)}|$. By definition, $e_o^{(i)} = y_m^{(i)} - y_p^{(i)}, \forall i = 0, \cdots, r-1$, with y_m and $e_o^{(i)}$ bounded; therefore $y_p^{(i)}, \forall i = 0, \cdots, r-1$ is bounded.

Part 2) with $y_p, \cdots, y_p^{(r-1)} \in \lambda_\infty$ the plant states are bounded using plant assumption P1 and P2 .This implies that $\alpha(X), \alpha_k(t), \beta_k(t) \in \lambda_\infty$.the projection algorithm ensures that $\beta_k(t) + \hat{\beta}(X)$ is bounded away from zero and that $\hat{\alpha}(X)$ is bounded. Thus,

$u_{sd} \in \lambda_\infty$. With $\alpha_1(X), \beta_1(X) \in \lambda_\infty$ establish $u_{hd_1} \in \lambda_\infty$ that. Since the fuzzy systems are defined appropriately so that $D_\alpha(X), D_\beta(X) \in \lambda_\infty$ then $u_{sdi} \in \lambda_\infty$.

Part 3) to show asymptotic stability of the output, we would like to find a bound on $\int_0^\infty e_s^2 dt$ using then

$$\int_0^\infty \eta e_s^2 dt \le -\int_0^\infty \dot{V}_t dt \tag{6.59}$$

$$= V_t(t) - V_t(\infty) \tag{6.60}$$

This establishes that $e_s \in \lambda_2$, $(\lambda_2 = \{z(t) : \int_0^\infty z^2 dt < \infty\})$ since $V_t(t), V_t(\infty) \in \lambda_\infty$ then $e_s \in \lambda_\infty$ by the definition of $V_t(t)$. In addition, we know that $\left|e_o^{(i)}\right| \in \lambda_\infty, i = 0, \cdots, r-1$ since $e_s \in \lambda\infty$ and $e_o^{(i)} = G_i(s) e_s$ with all the poles of $G_i(s), i = 0, \cdots, r-1$ in the open half-left plane. If $\alpha(X), \hat{\alpha}(X), \beta(X), \hat{\beta}(X), u_{sdi}, u_{hdi} \in \lambda_\infty$, then $\dot{e}_s \in \lambda_\infty$ from Since $e_s \in \lambda_2, \lambda_\infty$ and $\dot{e}_s \in \lambda_\infty$ by Barbalat's lemma we have asymptotic stability of e_s (i.e., $\lim_{t\to\infty}|e_o| = 0$), which implies asymptotic stability of e_o (i.e., $\lim_{t\to\infty}|e_o| = 0$).

CHAPTER 7

SWING UP AND BALANCING CONTROL OF THE PENDUBOT

7.1 Swing up Control

As stated earlier the goal of the Pendubot controller is to swing the links from their stable hanging position to unstable equilibrium positions and then balance the links about the equilibrium. This control is divided into two parts; swing up control. and balancing control. Both joints must approach the inverted position with nearly zero velocity so that they may be caught by a balancing controller [8-13].

The swing up control uses the method of partial feedback linearization [10-11]. Partial feedback linearization needs position feedback from both link one and link two but takes into account the nonlinear effects of the linkage.

We will now derive the partial feedback control for the pendubot. The equations of motion of the Pendubot are given by equation (3.1). Equation (3.1) can be simplified in the following form [10]

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + c_{11}\dot{q}_1 + c_{12}\dot{q}_2 + \phi_1 = \tau_1 \tag{7.1}$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + c_{21}\dot{q}_1 + \phi_2 = 0 \tag{7.2}$$

where q_1, q_2 are the joint angles , d_{11}, d_{22}, d_{12}, d_{22}, ϕ_1, ϕ_2 are as defined above, and

$$h_1 = -m_2 l_1 l_{c2} \sin(q_2)\dot{q}_2^2 - 2m_2 l_1 l_{c2} \sin(q_2)\dot{q}_2\dot{q}_1$$

$$h_1 = m_2 l_1 l_{c2} \sin(q_2)\dot{q}_1^2$$

The important distinction then between the system (7.1) and (7.2) and a standard two-link robot (3.A) is, of course, the absence of a control input torque to the second Equation (7.2).

Based on the fact that the second link is not actuated, the dynamics of both degrees of freedom can not linearized. "However, linearizing one of the degrees of freedom facilitates the design of an outer loop control that will track a given trajectory for the linearized degree of freedom. In case of the Pendubot we chose to linearize [10] about the collocated degree of freedom q_1. Equation (7.2) was solved for the angular accelaration of link two

$$\ddot{q}_2 = (-d_{21}\ddot{q}_1 - c_{21}\dot{q}_1 + \phi_2)/d_{22} \tag{7.3}$$

This was then substituted into equation (7.1) and written as

$$\bar{d}_{11}\ddot{q}_1 + \bar{c}_{11}\dot{q}_1 + \bar{c}_{12}\dot{q}_2 + \bar{\phi}_1 = \tau_1 \tag{7.4}$$

with

$$\bar{d}_{11} = d_{11} - \frac{d_{12}d_{21}}{d_{22}}$$

$$\bar{c}_{11} = c_{11} - \frac{d_{12}c_{21}}{d_{22}} \tag{7.5}$$

$$\bar{c}_{12} = c_{12}$$

$$\bar{\phi}_1 = \phi_1 - \frac{d_{12}\phi_2}{d_{22}}$$

Then just as with full linearization method the inner loop control that linearize q_1 can be defined as

$$\tau_1 = \bar{d}_{11} v_1 + \bar{c}_{11}\dot{q}_1 + \bar{c}_{12}\dot{q}_2 + \bar{\phi}_1 \tag{7.6}$$

This result in the system

$$\ddot{q}_1 = v_1 \tag{7.7}$$

$$d_{22}\ddot{q}_2 + c_{21}\dot{q}_1 + \phi_2 = -d_{21} v_1 \tag{7.8}$$

Since equation (7.7) is now linear, an outer loop control can be designed to track a given trajectory for link one. Equation (7.8) represents internal dynamics with respect to an output q_1.

The additional control term v_1 may now be chosen as [10]

$$v_1 = K_p(q_1^d - q_1) - K_d\dot{q}_1 \tag{7.9}$$

where K_p and K_d are positive gains ,so that q_1 tracks the reference angle q_1^d. with state variables

$$z_1 = q_1 - q_1^d \qquad z_2 = \dot{q}_1 \tag{7.10}$$

$$\eta_1 = q_2 \qquad \eta_2 = \dot{q}_2 \tag{7.11}$$

The closed loop system may be written as

$$\dot{z}_1 = z_2 \tag{7.12}$$

$$\dot{z}_2 = -K_p z_1 - K_d z_2 \tag{7.13}$$

$$\dot{\eta}_1 = \eta_2 \tag{7.14}$$

$$\dot{\eta}_2 = -\frac{1}{d_{22}}(h_2 + \phi_2) - \frac{d_{12}}{d_{22}} v_1 \tag{7.15}$$

We see from the above that the surface $z = 0$ in state space defines an invariant manifold for the system. Since $A = \begin{bmatrix} 0 & I \\ -k_p & -k_d \end{bmatrix}$ is Hurwitz for the positive values of K_p and K_d, this invariant manifold is globally attractive. The dynamics on this manifold are given by

$$\eta_1 = \omega(0, \eta) \tag{7.16}$$

with

$$\omega(0,\eta) = \begin{pmatrix} \eta_2 \\ -\dfrac{1}{d_{12}(\eta_1)}\left(h_1(0,\eta_1,\eta_2) + \phi_1(q_1^d, \eta_1)\right) \end{pmatrix} \tag{7.17}$$

It is interesting to note that the same result can be obtained by simply choosing an output equation

$$y = q_1 \tag{7.18}$$

for the original system (3.1)-(3.4), differentiating the output y until the input appears, and then choosing the control input to linearize the resulting equation. The system therefore has relative degree 2 with respect to output q_1.

The above reduced order system (7.16) therefore represents the zero dynamics of the system with respect to the output $y = q_1$. The zero dynamics are computed by specifying that the output y identically track the reference q_1^d. Therefore, substituting $z_1 = 0 = z_2$ into the equation yields

$$\dot{\eta}_1 = \eta_2 \tag{7.19}$$

$$\dot{\eta}_2 = -\frac{1}{d_{22}(q_2)}\phi_2(q_1^d, q_2) \tag{7.20}$$

Writing the above system in the original second order form yields the expression for the zero dynamics as a second order, autonomous nonlinear system

$$d_{22}(q_2)\ddot{q}_2 + \phi_2 = 0 \tag{7.21}$$

7.1.1 Swing up PD Fuzzy Controller

The goal of the outer loop control then is to track a given trajectory for link one and at the same time excite the internal dynamics to swing link two to a balancing position. For the swing up part a PD fuzzy controller has been used with following 5 rules [34].

There are many different types of fuzzy controllers we could examine for MISO case [34]. Here we will constrain ourselves to the two input "proportional-derivative fuzzy controller". This controller is similar to SISO fuzzy controller (proportional) with addition of the second input, $\dot{e}$ rate-change of error e. In fact, the membership functions on the universes of discourses and linguistic values NB, NS, ZE, PS, and PB for e and u are "negative big", "negative small", and so on. Assuming that there are seven membership functions on each input universe of discourse, there are 25 possible rules that can be put in the rule-base. A typical rule will take on the form

If e is NB and $\dot{e}$ is NB Then u is NB

The complete set of rules is shown in tabulated form in Table 7.1. In Table 7.1 the premises for the input $\dot{e}$ are presented by linguistic values found in the top row, the premises for the input e are presented in the linguistic values found in the left-most column, and the linguistic values representing the consequents for each of the 5 rules and 25 consequences can be founded the intersections of the row and column of the appropriate premises.

Table 7.1: PD Fuzzy controller set of rules

"Output" u		**"Change in error"** $\dot{e}$				
		NB	NS	ZE	PS	PB
"Error" e	NB	NB	PS	PB	PB	PB
	NS	NB	NS	PS	PB	PB
	ZE	NB	NS	ZE	PB	PB
	PS	NB	NS	NS	PS	PB
	PB	NB	NB	NB	NS	PB

where:

NB: NEGATIVE BIG

NS: NEGATIVE SMALL

ZE: ZERO

PS: POSITIVE SAMLL

PB: POSITIVE BIG

The membership functions for the premises and consequents of the rules are all are chosen as Gaussian membership functions. The width of each membership function is parameterized by A, B and D, respectively. The fuzzy controller will be adjusted by changing the values of A, B and D. The fuzzy inference mechanism operates by using

the product to combine the conjunctions in the premise of the rules and in the representation of the fuzzy implication. Singleton fuzzification is used, and defuzzification is performed using center-average method.

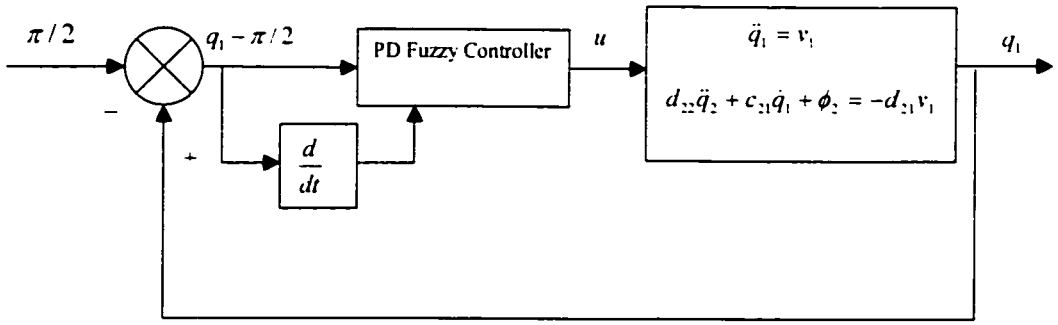

Figure 7.1 PD fuzzy controller diagram

7.2 Balancing Controller

Balancing the Pendubot refers to maintaining the Pendubot in the inverted position when it starts close to this position [8-13]. To balance the Pendubot in the inverted position, one must design a controller which stabilizes the behaviour of the system in some region about an equilibrium point. The Pendubot has a single stable equilibrium point corresponding to both links hanging vertically beneath joint one. Rather than a single inverted equilibrium point, however, the Pendubot dynamics result in a manifold of inverted equilibrium positions.

Physically, the Pendubot is in an inverted equilibrium position whenever the system center of mass is directly above joint one [10]. Each equilibrium position is associated with a unique constant torque input [11]; thus, only the completely vertical inverted position results in a zero torque input. Several of the infinitely many inverted equilibrium positions along the equilibrium manifold are shown in Figure 3.1. In this section we will design controllers to balance the Pendubot only in the completely vertical position.

The control for the balancing the Pendubot is very similar to the inverted cart-pole inverted pendulum problem. To design the controller we linearized the Pendubot's nonlinear equations of motion (3.9) [11]. The Taylor series approximation

$$f_a(x,u) = f_a(x_r,u_r) + \frac{\partial f}{dx}|_{x_r,u_r}\ (x-x_r) + \frac{\partial f}{dx}|_{x_r,u_r}\ (u-u_r) \qquad (7.22)$$

was used to linearize the plant. x is the vector of states given in equation (3.9) u is the single control input for the Pendubot. x_r and u_r are the equilibrium values of the states

and control respectively. Since we are only interested in controlling the Pendubot at equilibrium positions, $f_a(x_r, u_r)$ will always be zero.

All that is needed then to find the partial derivative matrices and evaluate them at the equilibrium points. Studying equation (3.6) through (3.9) it is observed that the Pendubot's equilibrium points can be defined by

$$u_r = \theta_4 g \cos(x_{r1}) \tag{7.23}$$

$$x_{r1} + x_{r3} = \pi / 2 \tag{7.24}$$

Differentiating equation (3.9) with respect to the states leaves the A matrix in the linearized model

$$\frac{df}{dx} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{df_2}{dx_1} & 0 & \frac{df_2}{dx_3} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{df_4}{dx_1} & 0 & \frac{df_4}{dx_3} & 0 \end{bmatrix} \tag{7.25}$$

The B matrix is found by the partial derivative with respect to the control input

$$\frac{df}{du} = \begin{bmatrix} 0 \\ \frac{df_2}{du} \\ 0 \\ \frac{df_4}{du} \end{bmatrix} \tag{7.26}$$

Refer to Appendix A for a full derivation of these partial derivative terms.

We define the top balancing position as the upright position with $x_{r1} = \pi / 2$, $x_{r3} = 0$ and $u_r = 0$.

Using these equilibrium values and parameters identified by the energy equation method

A linear models for the top is as follows

$$\dot{X} = AX + Bu \tag{7.27}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 51.9265 & 0 & -13.9704 & 0 \\ 0 & 0 & 0 & 1 \\ -52.8402 & 0 & 68.4210 & 0 \end{bmatrix} \tag{7.28}$$

$$B = \begin{bmatrix} 0 \\ 15.9549 \\ 0 \\ -29.3596 \end{bmatrix} \tag{7.29}$$

7.2.1 Adaptive Neural Fuzzy Control (Design Using Feedback Linearization as a Known Controller)

Now we turn our attention to the direct adaptive fuzzy control method derived in chapter 6 for the Pendubot. Here, the approach is rather to search for an unknown control law that provides (at least) asymptotically stable tracking and is able to compensate for disturbances and maintain stability. The direct adaptive fuzzy control methodology allows the designer to use previous knowledge or experience with the plant in various ways. It Is beneficial for the Pendubot application to include the known dynamics because it will increase the robustness of the design. Direct adaptive fuzzy control provides the designer with a method to incorporate a best guess of what the controller should be (below we will call this the "known controller," denoted by u_k). The algorithm then adaptively tunes a fuzzy controller to compensate for inaccuracies in our choice of this known controller.

The Pendubot has a relative degree of two. The input-output equation of the Pendubot can thus be rewritten as

$$\ddot{q}_2 = [\alpha_k(t) + \alpha(\mathrm{X})] + [\beta_k(t) + \beta(\mathrm{X})]u \tag{7.30}$$

where, $\alpha_k(t) \equiv 0$ and $\beta_k(t) \equiv 0$ ($\alpha_k(t)$ and $\beta_k(t)$ are known measurable parts of the system dynamics. Substituting the numerical values of the parameters we obtain

$$\alpha(\mathrm{X}) \approx -52.8402\dot{q}_1 + 68.4210\dot{q}_2 \tag{7.31}$$

$$\beta(\mathrm{X}) \approx -29.3596\,. \tag{7.32}$$

in these equations, approximately equal signs are used because the numerical parameters of the equations are not expected to represent the Pendubot's input-output dynamics exactly, rather, the right-hand side of Equations (7.31) and (7.32) are simply our best known approximations to α (X)and β(X), respectively. Note that $\beta(X) < 0$, so there exists $\beta_0 < 0$. For instance, $\beta_0 = -50$ for all $t \geq 0$; thus, β (X)is bounded away from zero, a condition is needed to ensure stability.

Direct adaptive fuzzy control is a somewhat more restrictive technique than its indirect counterpart since, it is also required that the system input–output (7.30) is such that $\beta_k(t) = 0$ for $t \geq 0$; further, it is assumed that β (X) is bounded by two finite constants, β_0 and β_1. For the Pendubot, this assumption holds since $-\infty < \beta_0 \leq \beta(X) \leq \beta_1 < 0$, where, for instance, β_0= -50 and $\beta_1 = 0$. The last plant assumption needed in Equations 7.30 and 7.32 is that for some β (X) ≥ 0, $|\dot{\beta}(X)| \leq B(X)$. Since is expected to be a constant, we can safely set $B(X) = 0$, and the assumption holds.

Note that the control Equations 6.26 and 6.27 are based on the premise that β (X)is positive, but it is stated there that the laws can be modified to allow for the negative case. Thus, the equations used here will be slightly modified versions of those in 7.26 and 7.27 as required by the characteristics of the Pendubot; specifically, the adaptation differential equation and the sliding-mode control term will each have a small but crucial sign change.

Let u^* be an unknown ideal controller that derived in Equation 6.27 to be approximated. This ideal controller is assumed to be a feedback linearizing law of the form

$$u^* = 1/\beta(\mathrm{X})[-\alpha(\mathrm{X}) + v_u(t)] \tag{7.33}$$

In general, it is possible to express u^* in terms of a Takagi–Sugeno fuzzy system, as $u^* = z_u^T A_u^* \zeta_u + u_k + d_u(\mathrm{X})$ where u_k is some known controller term, which we will use in this section and set equal to zero, and $d_u(\mathrm{X})$ is the error between the fuzzy representation and u^*. It is assumed that $D_u(\mathrm{X}) \geq |d_u(\mathrm{X})|$, where $D_u(\mathrm{X})$ is a known bound for the error. In practice, it is often hard to have a concrete idea about the magnitude of $D_u(\mathrm{X})$, because the relation between u^* and its fuzzy representation might be difficult to characterize; however, it is much easier to begin with a rough, intuitive idea about this bound, and then iterate the design process and adjust it, until the performance of the controller indicates that one is close to the right value. These bounds are both relatively small, which indicates that the fuzzy system we used, although a simple one, could represent the ideal controller with sufficient accuracy. The fuzzy control approximator which is going to be used is

$$\hat{u} = z^T A_u(t)\zeta_u + u_k \tag{7.34}$$

where $\zeta_u \in \Re^5$ is used with the fuzzy sets of Figure 7.2 The matrix $A_u(t) \in^{5\times 5}$ is adaptively updated on-line, and the function vector z is taken as $z = [1, q_1, \dot{q}_1, q_2, \dot{q}_2]$. The fuzzy system again uses only five rules each of the form

IF q_2 is F_i then $c_i = f_i(z)$, $i = 1, \cdots, 5$ (7.35)

where $f_i(z)$ is, respectively, a row of the matrix $z^T A_u(t)$. The fuzzy system is initialized with $A_u(t) = 0^{5\times 5}$. The input fuzzy sets F_i are as described in Figure 7.2.

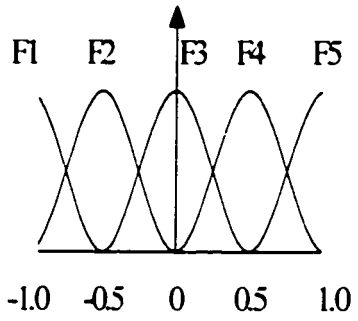

Figure 7.2: Input membership function

The direct adaptive fuzzy control law is given by $u_d = \hat{u} + u_{sd} + u_{bd}$. It is formed by three terms: the fuzzy approximation to the optimum controller (7.34), and sliding and bounding control terms. To approximate a feedback linearizing controller we will define v_a as in law (7.33) as follows: take the signals $e_o = y_m - y$, $e_s = k_0 e_o + \dot{e}_o$ and $v_a(t) = \ddot{y}_m + \eta e_s + k_0 \dot{e}_o$, where $\eta = 1$ and $k_0 = 2$. Since (as noted above) $B(\mathrm{X}) = 0$, the sliding-mode term is given by $u_{sd} = -D_u(\mathrm{X})\mathrm{sgn}(e_s)$. Note the minus sign which is a result of the fact that $\beta(\mathrm{X}) < 0$.

The bounding-control term needs the assumption that $\alpha(\mathrm{X})$ is bounded, with $|\alpha(\mathrm{X})| \le \alpha_1(\mathrm{X})$. Take $\alpha(\mathrm{X}) = 75\dot{q}_1 + 100\dot{q}_2$; then, Consider Figure 6.5, if $e_s > M_\varepsilon$,

$u_{hd} = \{|\hat{u}| + |u_{cd}| + [\alpha(\mathrm{X}) + |\nu_a(t)|]/\beta_0\}\mathrm{sgn}(e_a)$ and $u_{hd} = 0$ otherwise. For simulation, we used M_e=0.9.

The last part of the direct adaptive fuzzy control mechanism is the adaptation law, which is chosen in such a way that the output error converges asymptotically to zero, and the parameter error remains at least bounded. This law is given, in general by

$$\dot{\mathrm{A}}_u(t) = Q^{-1}\zeta_u^T[-e_s - q(t)] \tag{7.36}$$

Again, note the minus sign for e_s. The parameter $q(t)$ can be chosen nonzero to potentially improve adaptation mechanism given by Equation 7.33, but here we took $q(t)$=0 for $t \geq 0$. For simulation, $Q = 0.9I_5$ is used. With these choices the algorithm was able to adapt and estimate the control law $\hat{u}$ fast enough to perform well and compensate for disturbances.

Figures 8.1-8.6 show the simulation results with this controller. It has a behavior typical of feedback linearizing controllers on this plant: the error is effectively decreased to zero. Again, the advantages given by the adaptive capability of this algorithm appear most distinctively in the presence of strong disturbances: the controller is quite successful with both. The Pendubot is kept balanced, and the control input remains within small bounds around zero. Thus, this design proved to be robust and reliable.

CHAPTER 8

SIMULATION RESULTS AND DISCSSION

8.1 Simulation Results

This chapter displays the simulation results found when simulating the Pendubot with Matlab. Then a discussion of these results has been made at the end of this chapter. The summary of the simulation parameters is as follows:

8.1.1 Adaptive Neural Fuzzy

Figures 8.1, 8.2, 8.3, 8.4, 8.5 and 8.6 show a swing up, catch and balance of the Pendubot in the top position adaptive neural fuzzy control using the following parameters:

- Number of neural network layers: 2. (Equation 6.8)
- Number of If-Then rules: 5. (Equations 6.8)
- Number of fuzzy outputs sequences: 25 (Equation 6.8)
- Type of membership function: Centered-Gaussian (RBF). (Table 4.3)
- σ =25. (Table 4.3)
- β_0= -50 and $\beta_1 = 0$ (boundness). (Plant Assumption P3, page 78)
- $\alpha_k(t) \equiv 0$. (Equations 6.25 and 7.30)

- $\beta_k(t) \equiv 0$ (Equations 6.25 and 7.30)
- $D_u(\mathrm{X}) = 0.001$. (Page 79)
- $k_v = 2$. (Page 101)
- $\eta = 1$. (Page 101)
- $A_u(t) \in^{5\times 5}$ (Equation 6.28 and 7.36)
- $M_c = 0.9$. (Page 101-102)
- $q(t) = 0$ for $t \geq 0$ (adaptation mechanism) (Equation 6.28 and 7.36)
- $\mathrm{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 51.9265 & 0 & -13.9704 & 0 \\ 0 & 0 & 0 & 1 \\ -52.8402 & 0 & 68.4210 & 0 \end{bmatrix}$ (Equation 7.25, 7.28)
- $\mathrm{B} = \begin{bmatrix} 0 \\ 15.9549 \\ 0 \\ -29.3596 \end{bmatrix}$ (Equation 7.26, 7.29)

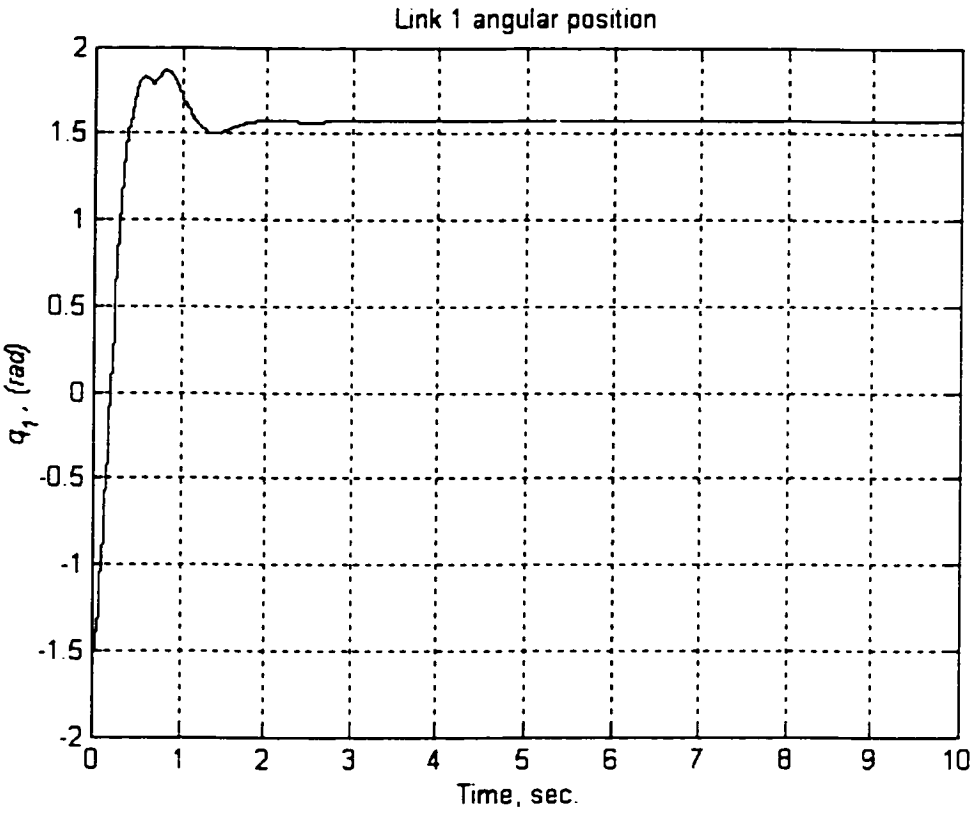

Figure: 8.1 link 1 angular position with adaptive neural fuzzy controller

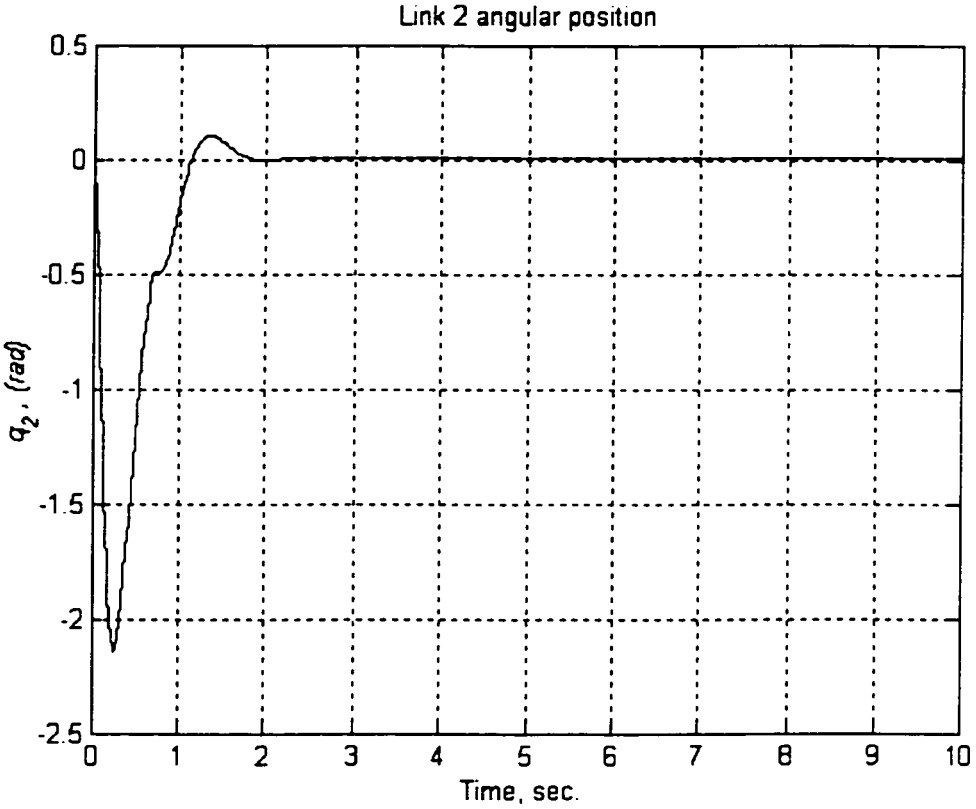

Figure: 8.2 link 2 angular position with adaptive neural fuzzy controller

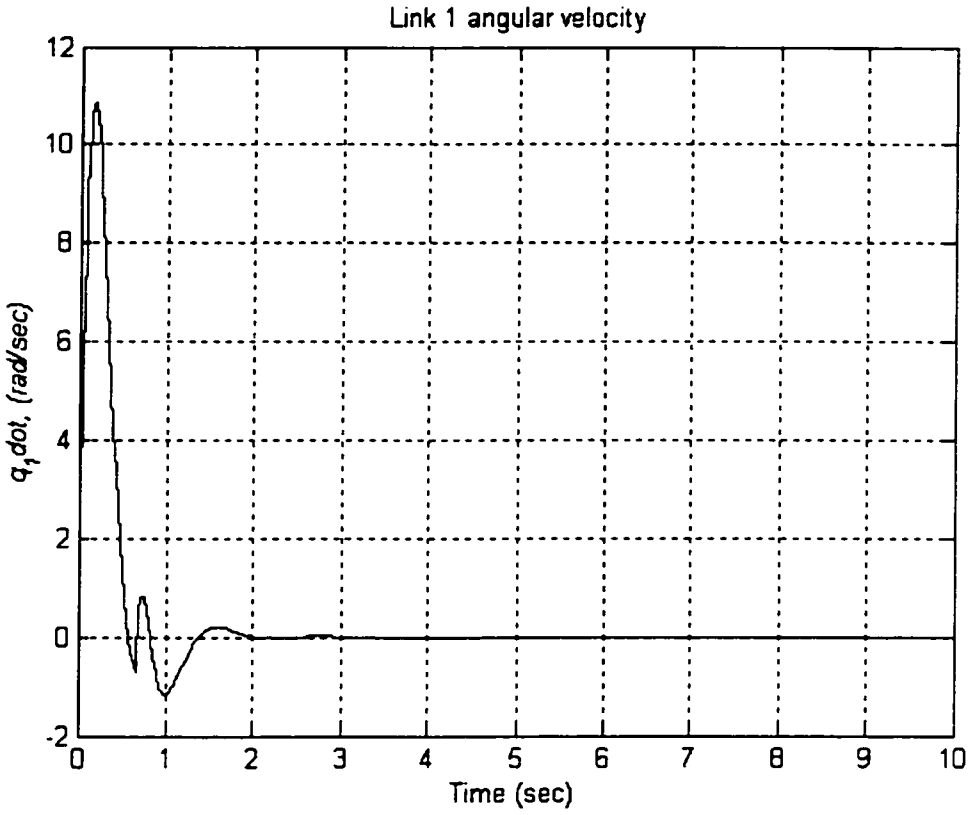

Figure: 8.3 link 1 angular velocity with adaptive neural fuzzy controller

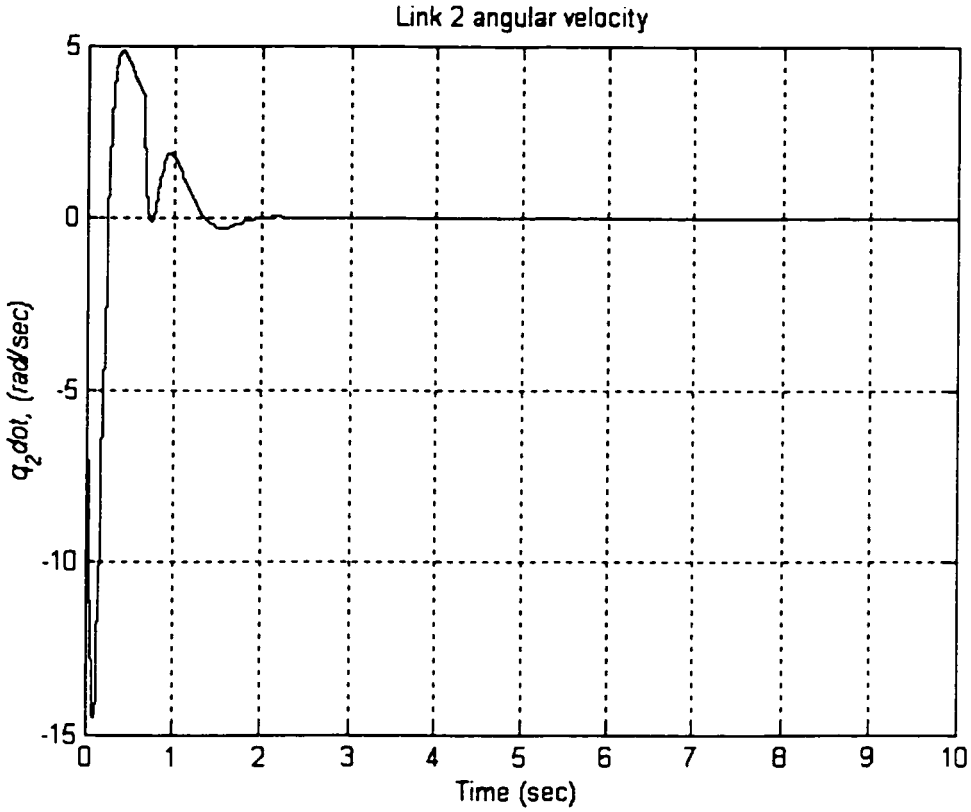

Figure: 8.4 link 2 angular velocity with adaptive neural fuzzy controller

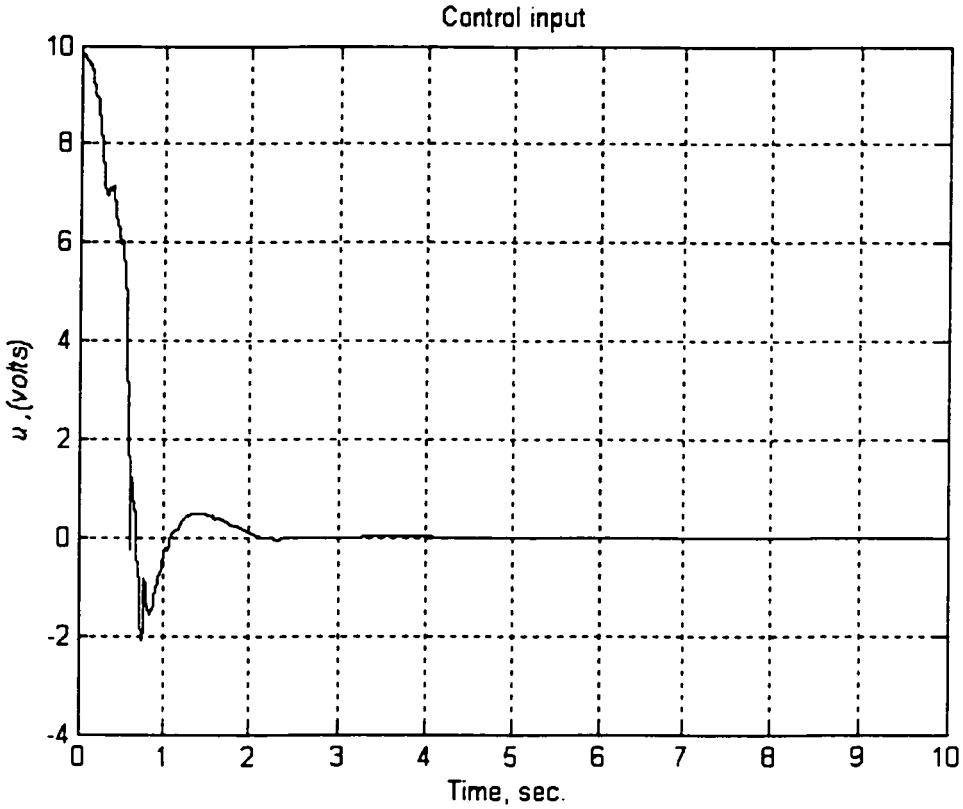

Figure: 8.5 Control input by using adaptive neural fuzzy

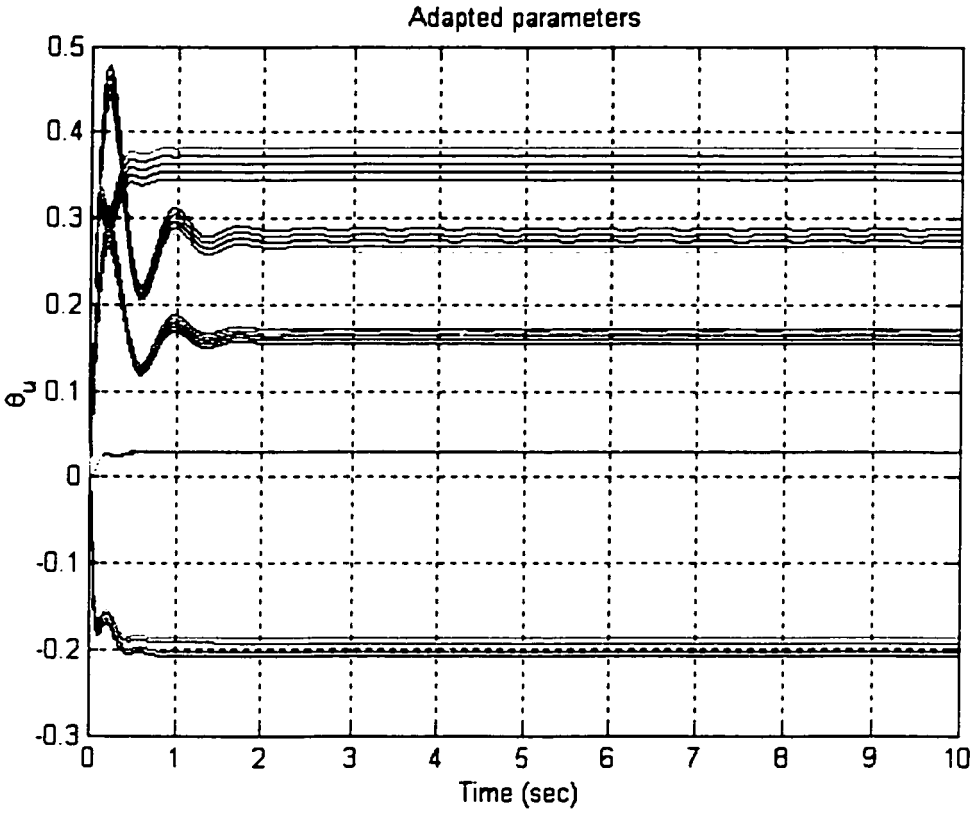

Figure: 8.6 Adaptive parameters with adaptive neural fuzzy controller

8.1.2 PD Fuzzy Controller

While figures 8.7, 8.8, 8.9, 8.10 and 8.11 show a swing up, catch and balance of the Pendubot in the top position using PD fuzzy controller when using the following parameters

- Number of If-Then rules: 5. (Table 7.1)
- Number of sequences: 25. (Table 7.1)
- Type of membership function: Centered-Gaussian (RBF). (Table 4.3)
- σ=10. (Table 6.1)
- Effective universe of discourse $[-1,1]$.
- $k_p = 50$, $k_d = 8.1818$ (Static normalizing gains). (Equation 7.13)

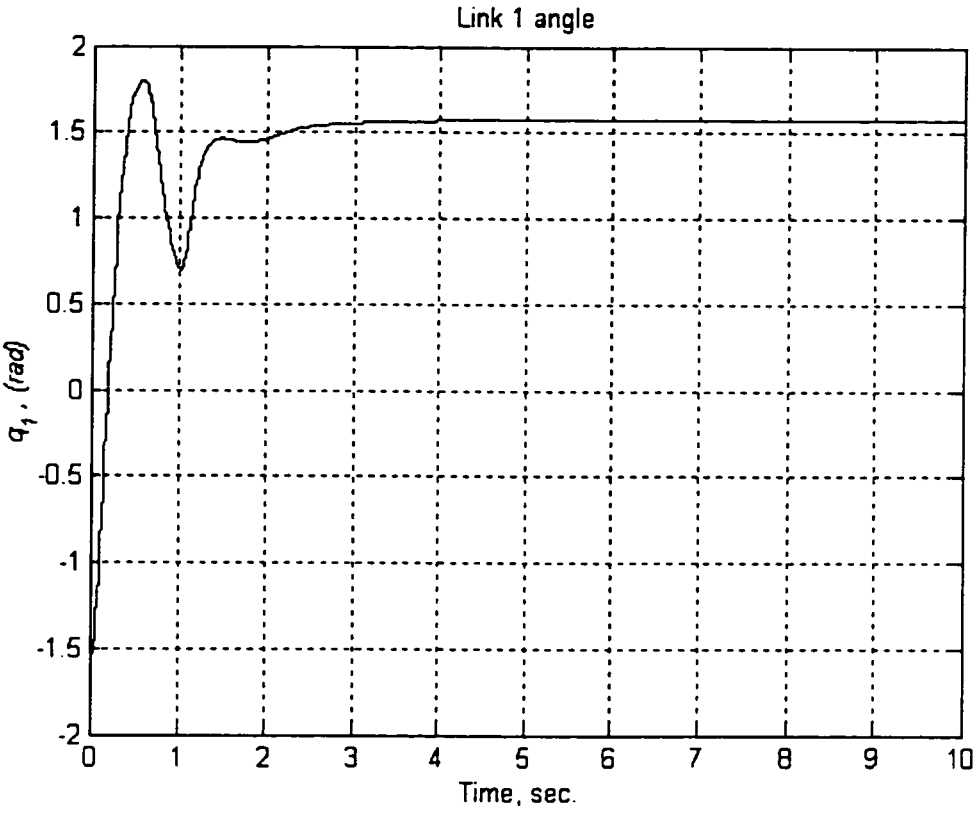

Figure: 8.7 link 1 angular position with PD fuzzy controller

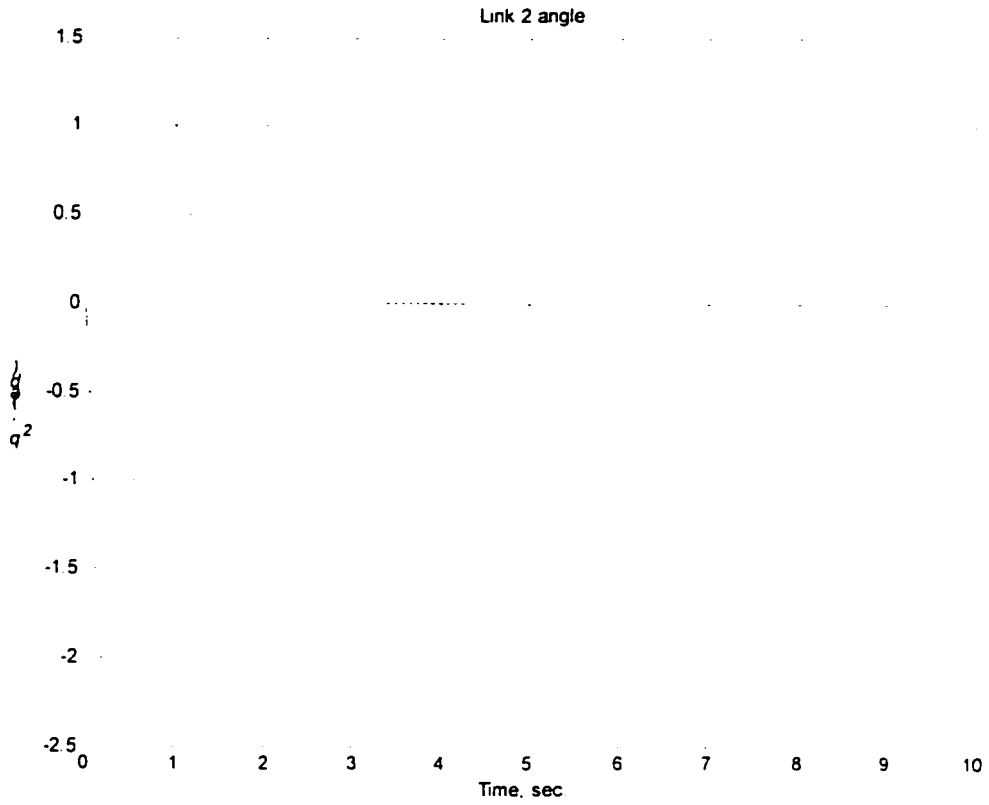

Figure: 8.8 link 2 angular position with PD fuzzy controller

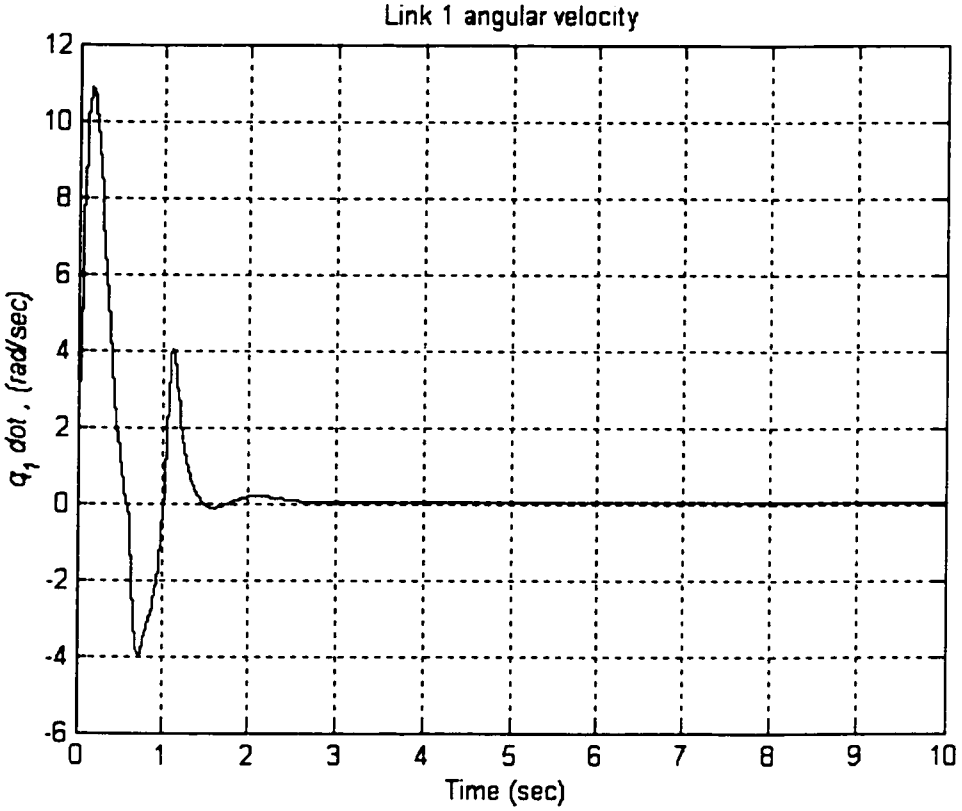

Figure: 8.9 link 1 angular velocity with PD fuzzy controller

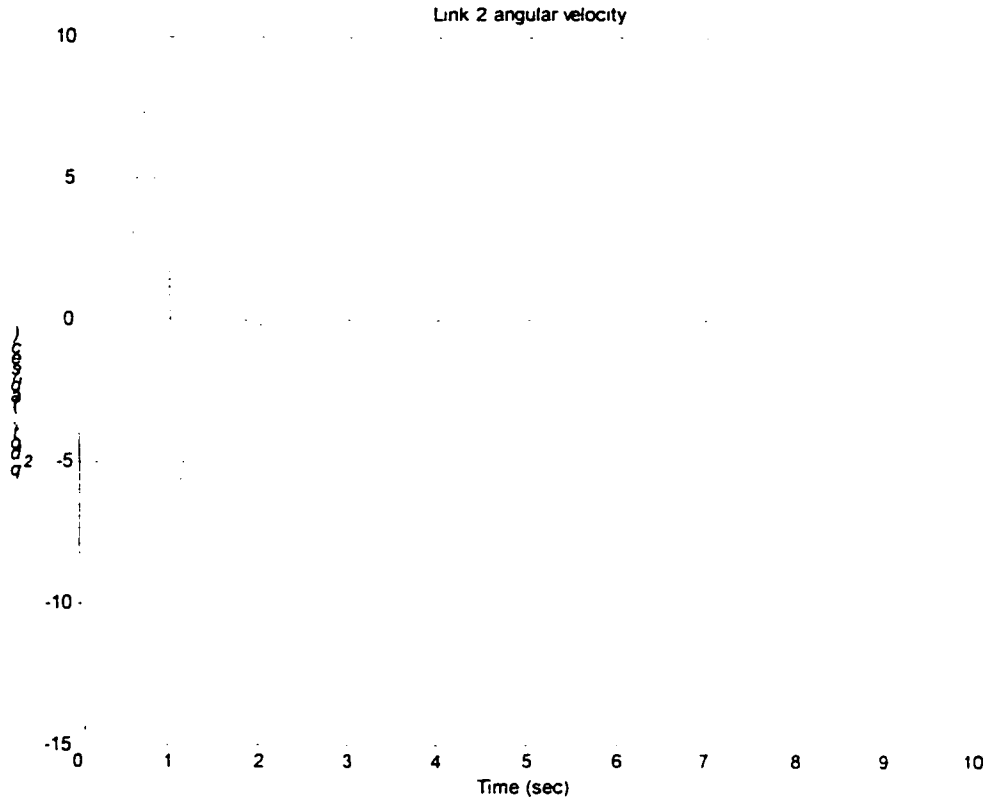

Figure: 8.10 link 2 angular velocity with PD fuzzy controller

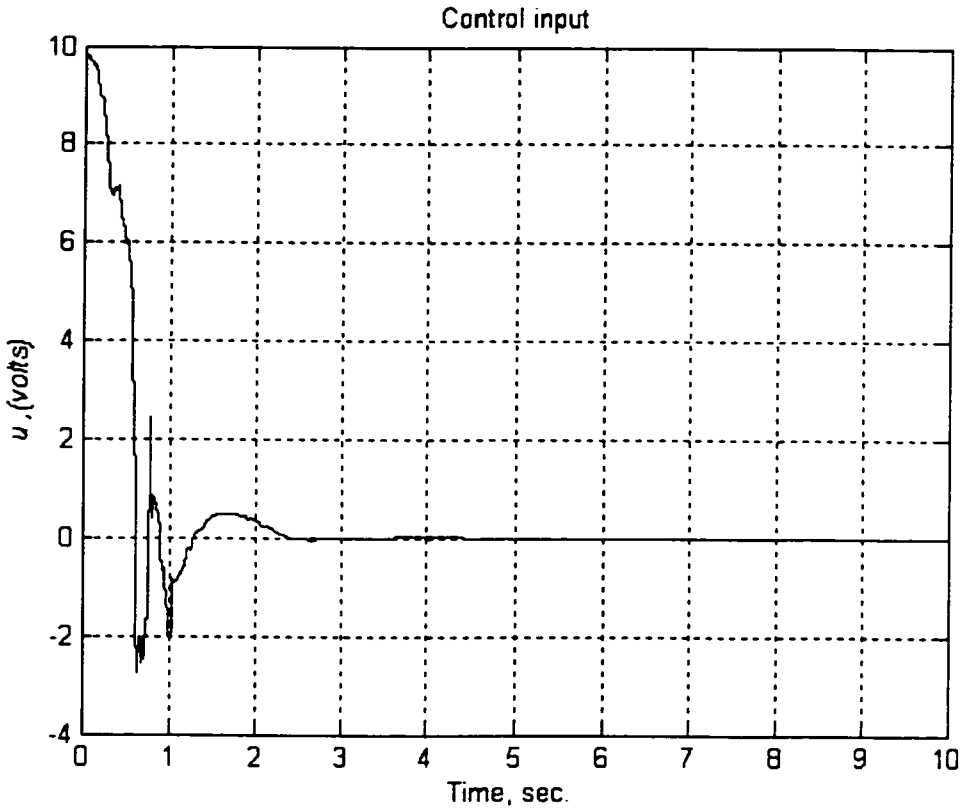

Figure: 8.11 Control input by using adaptive neural fuzzy

8.2 Discussion

The position response of the Pendubot for direct adaptive controller using fuzzy systems and neural networks is shown in figures 8.1 and 8.2, while the angular velocity response is shown in figures 8.3 and 8.4. The initial positions for link one was $-\pi/2$, and 0 for link two. Note that the zero or reference position for q_1 is horizontal, and the zero reference for link two is angle measured between link two and the longitudinal axis of link one. After the system was commanded to swing up, link one reached around the vertical position ($\pi/2$) within 0.5 seconds. But it is noticed that link one kept moving around this position and slowing down its speed from the overshoot to zero at around 2 seconds. To balance link 2, as a result of the excitation of link one, link two gets close to its set-position at around 1 second and kept slowing down the speed to zero after 2 seconds. Note that link one position response shows a larger overshoot than link 2. The control input which was measured in volts is recorded in figure 8.5. The maximum input was recorded at the initial time of swing up and was 10 volts, then starts decreasing to –2 volts at the maximum overshoot, then at around 2 seconds it was zero which proved the point view of boundness. The boundness of the adaptation parameters was recorded in figure 8.6 as well.

In the case of the PD fuzzy controller, the position response for link one and link two is recorded in figures 8.7, and 8.8. Angular velocity response curves are recorded in figures 8.9 and 8.10. Link one has an the maximum overshoot at around 0.5 seconds then it started to fall down, but the swing up controller brought it up at around 1 second and kept

trying to get the vertical position and slowing down the velocity to zero at 2.5 seconds. For link two which had been excited by link one, it started falling down at 0.5 seconds then it moved to the vertical position after a time of 1 second by then its velocity decreased down to zero after 2.5 seconds. The input to this controller was at its peak at the beginning of the swing up, then went down to about -2.2 volts at the overshoot time of the first link, when both links were at rest the input was zero volt. The control input was bounded between 10 volts and –2.2 volts.

Comparing both controllers it easily to recognize that the direct adaptive controller was faster the PD fuzzy controller and more robust.

CHAPTER 9

CONCLUSTIONS AND FUTURE WORK

9.1 Conclusions

This thesis presented the control of a two link underactuated planar revolute robot, named the Pendubot. Its actuated joint located at the shoulder and the elbow joint is unactuated and allowed to swing free. Two controllers were designed for the Pendubot. PD fuzzy controller with zero dynamics and 25 If-then fuzzy rules was used to design the control that swung the links from their hanging stable position to unstable equilibrium position. This controller shows the behavior of PD conventional controller.

Then to catch and balance the second link at the unstable equilibrium, a direct adaptive control using fuzzy systems and neural networks was designed using 5 If-Then rules. In addition to the fact that both fuzzy systems and neural networks have the capability to approximate the dynamics of the systems, some radial basis function neural networks are equivalent to certain standard fuzzy systems in the sense that they are functionally equivalent. In other words, if the number of receptive field units equal to the number of rules, the receptive field unit strengths equal to the output membership function centers, and the receptive field units to be the same as the premise membership functions then they will produce the same outputs.

For the direct adaptive fuzzy system or neural network controller if the the desired output trajectory and its derivative are measurable and bounded. if the plant is strong relative degree. and if controller gain is bounded then:

a) The asymptotic stability of the system output in the vertical position can been proved by Layponuv.
b) The plant output and its derivatives are bounded.
c) The control signals are bounded.
d) The magnitude of the output error decreases at least asymptotically to zero.

The results which were presented in this thesis demonstrate the performance of the Pendubot with these controllers.

9.2 Future Work

As this thesis has dealt with control of single-input single-output underactuated mechatronic system with two degrees of freedom. a further future research work on direct adaptive control for nonlinear underactuated robotic system is proposed to investigate other topics:

a) A direct adaptive control for underactuated robotic systems with multi-degrees of freedom using fuzzy control systems and neural networks.
b) A direct adaptive control using fuzzy control systems and neural networks can also be developed for a class of continuous time multi-input multi-output nonlinear underactuated systems with poorly understood dynamics.

REFERENCES

[1] Su, C.-Y., and Stepanenko, Y., "Adaptive variable structure set-point control of underactuated robots," *IEEE Transactions on Automatic Control*, vol. 44, no. 11,1999.

[2] Su, C.-Y., Leung, T.-P and Stepanenko, Y. , "Sliding Model Control of Nonholomonic Mechanical System: Under-actuated Manipulator Case," *Nonlinear Control System Design Symposiums* , IFAC Preprint , Tehac City, pp. 609-613,2000.

[3] Spong, M.W., "Underactuated Mechanical Systems," *Control Problems in Robotics and Automation*, B. Siciliano and K.P. Valavanis (Eds), *Lecture Notes in Control and Information Sciences 230* Spinger-Verlag, London, UK, 1997.

[4] Spong, M.W., and Praly, L., "Control of Underactuated Mechanical Systems Using Switching and Saturation" *Proc. of the Block Island Workshop on Control Using Logic Based Switching* , Springer-Verlag, 1996.

[5] Spong, M.W., "The Control of Underactuated Mechanical System", *Plenary Address at The First International Conference on Mechatronics*, Mexico City, Jan. 26--29, 1994.

[6] Spong, M.W., "Partial Feedback Linearization of Underactuated Mechanical Systems", *IROS'94* , Munich, Germany, pp. 314-321, Sept. 1994.

[7] Takashima, S.,"Control of a Gymnast on a High Bar," IEEE Int'l Workshop on Intelligent Robots and Systems, IROS'91, pp1424-1429, Osaka, Japan, Nov.1991.

[8] Spong, M. W., and Block, D.J. "The pendubot: A mechatronic system for control research and education," In *IEEE Conference on Decision & Control*, pages 555-556, 1995.

[9] Block, D.J., and Spong, M.W., "Mechanical Design & Control of the Pendubot," *SAE Earthmoving Industry Conference* , Peoria, IL, April 4-5, 1995.

[10] Mechatronic System ,Inc. , "Pendubot Model P-2 User's Manual", *Mech. Systems, Inc.*, 1998

[11] Block, D.J., "Mechanical Design and Control of Pendubot", M.S. Thesis, Department of General Engineering, *University Of Illinois at Urbana-Champian*,1996

[12] Fantoni, I., Lozano, R., and Spong, M.W. "Energy based control of the Pendubot," *IEEE Transactions on Automatic Control*, AC-45, No. 4 , pp. 725 -729, April 2000.

[13] Fantoni, I., Lozano, R., and Spong, M.W., "Passivity Based Control of the Pendubot," *American Control Conference*, San Diego, June, 1999.

[14] Spong, M.W., "The swing up control problem for the acrobot", *IEEE Control Systems Magazine*, 15(1):49-55, February 1995.

[15] Brown S.C., Passino K.M., "Intelligent Control for an Acrobot," *Journal of Intelligent and Robotic Systems*, Vol. 18, pp. 209-248, 1997.

[16] Papadopouls, E., and Dubowsky, S.,"Dynamic Singularityies in Free-Floating Space Manipulators," *ASME J. Dynamical Systems , Measurement, and Control*, Vol. 115, March,1993.

[17] Dubowsky , S., and Papadopoulos, E., "The Kinenmatics, Dynamics, and Control of Free-Flying and Free-Floating Space Robotic Systems," *IEEE Trans. On Robotics and Automation*, Vol. 9, No. 4, pp. 411-422, Aug. 1993.

[18] Papadopouls, E., and Dubowsky, S.,"Coordinated Manipulator/Spacexcraft Motion Control for Space Robotic Systems," Proc. *IEEE Int. Conf. On Robotics and Automation*, pp. 731-737, Sacramento, CA, May, 1991.

[19] Gu, Y-L, and Xu , Y.," A Normal Form Augmentation Approach to Adaptive Control of Space robot Systems," Proc. *IEEE Int. Conf. On Robotics and Automation*, pp. 731-737, Atlanta, GA, May, 1993.

[20] Hauser, J. and Murray , R. M.. Nonlinear contollers for non-integrable systems: the acrobot example. *In Proc. American Control Conference*, 1990.

[21] Bortoff, S.A., Pseudolinearization using Spline Functions with Application to the Acrobot, Ph.D. Thesis, Dep. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 1992.

[22] Xiao Qing, M., " Fuzzy Control of an Under-actuated Robotic Manipulator: Pendubot ", M.S. Thesis, Department of Mechanical and Industrial Engineering, *Concordia University*,2002.

[23] Zhen, C., "Trajectory tracking control of Pendubot using the Takagi-Sugeno fuzzy system", M.S. Thesis, Department of Mechanical and Industrial Engineering, *Concordia University*,2002.

[24] Su, C.-Y. and Y. Stepanenko, Adaptive control of a class of nonlinear systems with fuzzy logic, *IEEE Transactions on Fuzzy Systems*, vol. 2, no. 4, 285-294, 1994

[25] Han, H., Su, C.-Y. and Y. Stepanenko, Adaptive control of a class of nonlinear systems with nonlinearly parameterized fuzzy approximators, *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 2, pp. 315-323, 2001

[26] Su, C.-Y. and Stepanenko, Y., "Adaptive sliding mode control of robot manipulators: General sliding manifold case", *Automatica*, vol. 30, no. 9, 1497-1500, 1994

[27] Su, C.-Y., Leung, T.-P and Stepanenko, Y. "Real-time implementation of regressor based sliding mode control scheme for robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 40, no. 1, 71-79, 1993 (invited paper)

[28] Ge, S.S., Lee, T.H, Harris, C.J. Adaptive Neural Network Control of Robotic Manipulators, *World Scientific*, 1998.

[29] Ordóñez, R., Zumberge J., Spooner, J.T., Passino, K.M., "Adaptive Fuzzy Control: Experiments and Comparative Analyses," *IEEE Trans. on Fuzzy Systems*, Vol. 5, No. 2, 167-188, 1997.

[30] Spooner, J.T., Passino, K.M., "Stable Adaptive Control Using Fuzzy Systems and Neural Networks," *IEEE Trans. on Fuzzy Systems*, Vol. 4, No. 3, pp. 339-359, Aug. 1996.

[31] Ordóñez, R., Passino, K.M., "Stable Multiple-Input Multiple-Output Adaptive Fuzzy/Neural Control," *IEEE Trans. on Fuzzy Systems*, Vol. 7, No. 3, pp. 345-353, 1999.

[32] Jenkins, D., Passino, K.M., "An Introduction to Nonlinear Analysis of Fuzzy Control Systems," *Journal of Intelligent and Fuzzy Systems*, Vol. 7, No. 1, pp. 75-103, 1999.

[33] Zumberge, J., Passino, K.M., "A Case Study in Intelligent vs. Conventional Control for a Process Control Experiment," *Journal of Control Engineering Practice*, Vol. 6, No. 9, pp. 1055-1075, 1998.

[34] Passino, K.M., Stephen, Y., Fuzzy Control, *Addison-Wesley*, 1997.

[35] Mamdani, E. H., Assiliian, S. "An experiment in linguistic synthesis with fuzzy logic controller," *International journal of Man-Machine Studies*, 7(1): 1-13, 1975.

[36] Zadeh, L. A., "Outline of a new approach to the analysis of complex systems and decistion processes," *IEEE Trnsaction on Systems, Man, Cybernetics*, 3(1):28-44, January1973.

[37] Zadeh, L. A., Desoer, C.A., Linear system theory: the state space approach. McGraw-Hill. New York, N.Y., 1963.

[38] Zadeh, L. A., "Fuzzy sets", *Information and control*, 8:338-353, 1965.

[39] Zadeh, L. A., "Quantitative fuzzy semantics," *Information Sciences*, 3:159-176, 1971.

[40] Zadeh, L. A., "The concept of a linguistic variable and its application to approximate reasoning," *Parts 1, 2, and 3. Information Science*, 8:199-249, 8:301-357, 9:43-80, 1975.

[41] Kosko, B., Neural networks and fuzzy systems: a dynamical system approach. Prentice Hall, Upper Saddle River, NJ, 1991.

[42] Lee, C.-C, "Fuzzy logic in control systems: fuzzy logic controller-part 1," *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2): 419-435, 1990.

[43] Jang, J.R., Sun, C.-T, "Neuro-Fuzzy and Control,". *IEEE*, March 1997.

[44] Sugeno, M. International applications of fuzzy control. *Elsevier Science*, 1985.

[45] Sugeno, M., Kang, G.T., "Structure identification of fuzzy model," *fuzzy sets and systems*, 28:15-33, 1998.

[46] Takagi, T., Sugeno, M., "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. On Systems, Man, and Cybernetics*, 15:116-132, 1985.

[47] Jang, J.R., Sun, C.-T, Mizutani, E. Neuro-Fuzzy and Soft Computing. *Printice-Hall International.* Upper Saddle River, 1997.

[48] Rosenblat, F. Priciples of neurodynamics: perceptions and theory of brain mechanism. Spartan, New York, 1962.

[49] Psaltis, D., Sideris, A., Yamamaura, A., "A multilayred neural network controller," *IEEE Control systems Magazine*, 8(4): 17-21, April 1998.

[50] Kung, S.Y., Hwang, J.-N., "Neural Networks Architecture for Robotics Applications," *IEEE Transactions on Robotics and Automation*, 5(5), 541-557,1889.

[51] Lippmann, R.P., "An introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, 4, April, 4-22, 1987.

[52] Lin, L.-J., "Programming Robots Using Reinforcement Learning, Planning and Teaching," *Machine Learning*, 8, 293-321, 1992.

[53] Hetch-Nielson, R., Neurocomputing, Reading, Massachusetts, Addison-Wesley, Reading, Massachusetts, 1990.

[54] Arbib, M.A., "Neural Computing: The Challenge on the Sixth Generation," *Reprint from the EDUCOM Bulletien*, 23(1), 2-12, 1988.

[55] Barto, A.G., Sutton, R.S., Anderson, C.W.,"Neurolike Adaptive Element that Can Solve Difficult Control Problems," *IEEE Transactions on Systems, Man and Cybernetics*, 13, 834-846, 1983.

[56] Brooks, R., " New approaches to Robotics," *Science*, 253(13),1227-1232,1991.

[57] Brooks, R., "Intelligent without Representation," *Artificial Intelligence*, 47,139-159,1991.

[58] Hertz, J., Krogh, A., Palmar, R.G., Intorduction to the theory of Neural Computation, *Addison,-Wesley*, Reading, Massachusetts, 1991.

[59] Nise Norman S., Control System Engineering, *The Benjamin/Cummings Publishing Company, Inc.*, 1999.

[60] Su, C.-Y. and Stepanenko, Y., "Redesign of hybrid adaptive/robust motion control of rigid-link electrically-driven robot manipulators," *IEEE Transactions on Robotics and Automation*, vol.14, no. 4, pp. 651-655, 1998

[61] Su, C.-Y. and Stepanenko, Y., "Hybrid adaptive/robust motion control of rigid-link electrically-driven robot manipulators," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 426-432, 1995

[62] Spong, M.W., and Vidyasagar, M., Robot Dynamics and Control, John Wiley & Sons, Inc., New York, 1989.

[63] Wiklund, M., Kristenson, A., and Astrom, K.J. "A New Strategy for Swinging up an Inverted Pendulum," *Proc. IFAC Symposium*, Sydney, Australia, 1993.

[64]Gautier, M., and Khalil, W., "On the identification of the inertial parameters of robots," *Proceedings of 27th IEEE CDC*, pages 2264-2269, 1988

[65] Passino, K.M., Michel, A.N., Antsaklis, P.J., "Lyapunov Stability of Discrete Event Systems", *Avomaticka i Telemekhanika*, (*Int. Journal of Automation and Remote Control*) No. 8, pp. 3-18, 1992 (Invited paper, published in Russian; also appears in English published by Plenum Press).

[66] Garcia-Benitez, E., Yurkovich, S., Passino, K.M., "Rule-Based Supervisory Control of a Two-Link Flexible Manipulator", *Journal of Intelligent and Robotic Systems*, Vol. 7, No. 2, pp. 195-213, April 1993.

[67] Layne, J.R., Passino, K.M., Yurkovich, S., "Fuzzy Learning Control for Anti-Skid Braking Systems", *IEEE Trans. on Control Systems Technology*, Vol. 1, No. 2, pp. 122-129, June 1993.

[68] Passino, K.M., Michel, A.N., Antsaklis, P.J., "Lyapunov Stability of a Class of Discrete Event Systems", *IEEE Transactions on Automatic Control*, Vol. 39, No. 2, pp. 269-279, Feb. 1994.

[69] Moudgal, V.G., Kwong, W.A., Passino, K.M., and Yurkovich S., "Fuzzy Learning Control for a Flexible-Link Robot", *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 2, pp. 199-210, May 1995.

[70] Spooner, J.T., Passino, K.M., "Adaptive Control of a Class of Decentralized Nonlinear Systems," IEEE Trans. on Automatic Control, Vol. 41, No. 2, pp. 280-284, Feb. 1996.

[71] Layne, J.R., Passino, K.M., "Fuzzy Model Reference Learning Control," Journal of Intelligent and Fuzzy Systems, Vol. 4, No. 1, pp. 33-47, 1996.

[72] Ordóñez, R., Passino, K.M., "Indirect Adaptive Control for a Class of Time-Varying Nonlinear Systems," International Journal of Control, Vol. 74, No. 7, pp. 701-717, 2001.

[73] Diao, Y., Passino, K.M., "Fault Tolerant Stable Adaptive Fuzzy/Neural Control for a Turbine Engine," IEEE Trans. on Control Systems Technology, Vol. 9, No. 3, pp. 494-509, May 2001.

[74] Lee, H., Tomizuka, M., "Robust Adaptive Control Using a Universal Approximator for SISO Nonlinear Systems," *IEEE* Transaction on Fuzzy Systems, Vol. 8, No 1, February 2000.

APPENDIX A

LINEARIZED EQUATIONS

$$f_a(x,u) = f_a(x_r,u_r) + \frac{\partial f}{\partial x}|_{x_r,u_r}(x - x_r) + \frac{\partial f}{\partial u}|_{x_r,u_r}(u - u_r) \tag{A.1}$$

$$u_r = \theta_4 g \cos(x_{r1}) \tag{A.2}$$

$$x_{r1} + x_{r3} = \pi/2 \tag{A.3}$$

$$A = \frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{\partial f_2}{\partial x_1} & 0 & \frac{\partial f_2}{\partial x_3} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{\partial f_4}{\partial x_1} & 0 & \frac{\partial f_4}{\partial x_3} & 0 \end{bmatrix} \tag{A.4}$$

with

$$\frac{\partial f_2}{\partial x_1} = \frac{g[2\theta_2\theta_4 \sin(x_{r1}) - \theta_3\theta_5 \sin(x_{r1}) - \theta_3\theta_5 \sin(x_{r1} + 2x_{r3})]}{2\theta_1\theta_2 - \theta_3^2 - \theta_3^2 \cos(2x_{r3})} \tag{A.5}$$

$$\frac{\partial f_2}{\partial x_3} = \frac{-\theta_3^2 \sin(2x_{r3})\left[\theta_2 u_r - g\theta_2\theta_4 \cos(x_{r1}) + \frac{g\theta_3\theta_5 \cos(x_{r1})}{2}\right] - \theta_3^2 \sin(2x_{r3})\left[\frac{g\theta_3\theta_5 \cos(x_{r1}+2x_{r3})}{2}\right]}{\left[\theta_1\theta_2 - \frac{\theta_3^2}{2} - \frac{\theta_3^2\cos(2x_{r3})}{2}\right]^2}$$

$$+ \frac{2g\theta_3\theta_5 \sin(x_{r1}+2x_{r3})}{-2\theta_1\theta_2 + \theta_3^2 + \theta_3^2\cos(2x_{r3})}$$

(A.6)

$$\frac{\partial f_4}{\partial x_1} = \frac{g\theta_5 \sin(x_{r1}+x_{r3})[\theta_1+\theta_2+2\theta_3\cos(x_{r3})] - g[\theta_2+\theta_3\cos(x_{r3})] \quad [\theta_4\sin(x_{r1})+\theta_5\sin(x_{r1}+x_{r3})]}{\left[\theta_1\theta_2 - \frac{\theta_3^2}{2} - \frac{\theta_3^2\cos(2x_{r3})}{2}\right]}$$

(A.7)

$$\frac{\partial f_4}{\partial x_3} = \frac{\theta_3^2 \sin(2x_{r3})[g\theta_5\cos(x_{r1}+x_{r3})[\theta_1+\theta_2+2\theta_3\cos(x_{r3})]]}{\left[\theta_1\theta_2 - \frac{\theta_3^2}{2} - \frac{\theta_3^2\cos(2x_{r3})}{2}\right]^2}$$

$$- \frac{\theta_3^2 \sin(2x_{r3})[\theta_2+\theta_3\cos(x_{r3})][-u_r + g\theta_4\cos(x_{r1}) + g\theta_5\cos(x_{r1}+x_{r3})]}{\left[\theta_1\theta_2 - \frac{\theta_3^2}{2} - \frac{\theta_3^2\cos(2x_{r3})}{2}\right]^2}$$

+

$$\frac{\left[\frac{g\theta_3\theta_4\sin(x_{r1}-x_{r3})}{2}\right] + \theta_3 u_r \sin(x_{r3}) - \left[\frac{g\theta_3\theta_4\sin(x_{r1}+x_{r3})}{2}\right] + g\theta_1\theta_5\sin(x_{r1}+x_{r3}) + g\theta_3\theta_5\sin(x_{r1}+2x_{r3})}{\left[\theta_1\theta_2 - \frac{\theta_3^2}{2} - \frac{\theta_3^2\cos(2x_{r3})}{2}\right]}$$

(A.8)

$$B = \frac{\partial f}{\partial u} = \begin{bmatrix} 0 \\ \dfrac{\partial f_2}{\partial u} \\ 0 \\ \dfrac{\partial f_4}{\partial u} \end{bmatrix} \tag{A.9}$$

with
$$\frac{\partial f_2}{\partial u} = \frac{\theta_2}{\theta_1\theta_2 - \dfrac{\theta_3^2}{2} - \dfrac{\theta_3^2\cos(2x_{r3})}{2}} \tag{A.10}$$

$$\frac{\partial f_4}{\partial u} = \frac{-\theta_2 - \theta_3\cos(x_{r3})}{\theta_1\theta_2 - \dfrac{\theta_3^2}{2} - \dfrac{\theta_3^2\cos(2x_{r3})}{2}} \tag{A.11}$$

APPENDIX B

SAFETY INSTRUCTION

Section 2.3 and Section 2.4 in the Pendubot User's Manual [10] contains vital information about safety issues associated with the system.

All users must read and understand the safety and operation guidelines in Section 2.3 and Section 2.4 of the Pendubot User's Manual prior to operating the system.

Caution: the user must hang the base plate far enough off the edge of the table (approximately 5 cm) so that the encoder on link 1 does not hit the table when it swings.

If any material is unclear, the user must contact Mechatronic Systems for clarification before operating the system.

In the event of an emergency, the control effort should be immediately discontinued by releasing the button found on the handheld amplifier inhibit switch, and/or moving the toggle switch found on the back of the base unit to the off position.