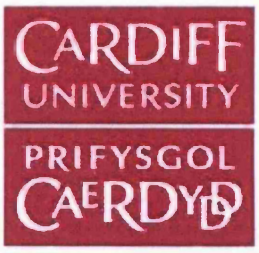
BINDING SERVICES
Tel +44 (0)29 2087 4949
Fax +44 (0)29 20371921
e-mail bindery@cardiff.ac.uk



NEURO-FUZZY MODELLING AND CONTROL OF ROBOTIC MANIPULATORS

A thesis submitted to the University of Wales, Cardiff

In candidature for the degree of

Doctor of Philosophy

By

A. A. FAHMY, B.Sc., M.Sc.

Intelligent Robotic Systems Laboratory

Cardiff School of Engineering

University of Wales, Cardiff

United Kingdom

2005

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

In the name of Allah,

The Most Gracious, The Most Merciful

To my family

ACKNOWLEDGEMENTS

I would like to express my special gratitude to the supervisor of my studies, Professor D. T. Pham, for his encouragement, invaluable guidance and strong support throughout my studies. I consider myself very lucky to have him as my study supervisor.

I wish to express my sincere thanks to the University of Wales Cardiff, especially the Intelligent Robotic Systems Laboratory and Manufacturing Engineering Centre, Cardiff School of Engineering for the use of the facilities to pursue this research work.

Grateful acknowledgement for my funding and support must be made to my home country Egypt and the Egyptian Ministry of Higher Education. Also my sincere thanks must go to their representative in UK, the Egyptian Educational and Cultural Bureau.

My hearty thanks go to my family, Mother, Father, and my valuable wife for their encouragement and support over the past years.

SYNOPSIS

The work reported in this thesis aims to design and develop a new neuro-fuzzy control system for robotic manipulators using Machine Learning Techniques, Fuzzy Logic Controllers, and Fuzzy Neural Networks. The main idea is to integrate these intelligent techniques to develop an adaptive position controller for robotic manipulators. This will finally lead to utilising one or two coordinated manipulators to perform upper-limb rehabilitation. The main target is to benefit from these intelligent techniques in a systematic way that leads to an efficient control and coordination system. The suggested control system possesses self-learning features so that it can maintain acceptable performance in the presence of uncertain loads. Simulation and modelling stages were performed using dynamical virtual reality programs to demonstrate the ideas of the control and coordination techniques.

The first part of the thesis focuses on the development of neuro-fuzzy models that meet the above requirement of mimicking both kinematics and dynamics behaviour of the manipulator. For this purpose, an initial stage for data collection from the motion of the manipulator along random trajectories was performed. These data were then compacted with the help of inductive learning techniques into two sets of if-then rules that form approximation for both of the inverse kinematics and inverse dynamics of the manipulator. These rules were then used in fuzzy neural networks with differentiation characteristics to achieve online tuning of the network adjustable parameters.

The second part of the thesis introduces the proposed adaptive neuro-fuzzy joint-based controller. To achieve this target, a feedback Fuzzy-Proportional-Integral-Derivative incremental controller was developed. This controller was then applied as a joint servo-controller for each robot link in addition to the main neuro-fuzzy feedforward controller used to compensate for the dynamics interactions between robot links. A feedback error learning scheme was applied to tune the feedforward neuro-fuzzy controller online using the error back-propagation algorithm.

The third part of the thesis presents a neuro-fuzzy Cartesian internal model control system for robotic manipulators. The neuro-fuzzy inverse kinematics model of the manipulator was used in addition to the joint-based controller proposed and the forward mathematical model of the manipulator in an adaptive internal model controller structure. Feedback-error learning scheme was extended to tune both of the joint-based neuro-fuzzy controller and the neuro-fuzzy internal model controller online.

The fourth part of the thesis suggests a simple fuzzy hysteresis coordination scheme for two position-controlled robot manipulators. The coordination scheme is based on maintaining certain kinematic relationships between the two manipulators using reference motion synchronisation without explicitly involving the hybrid position/force control or modifying the existing controller structure for either of the manipulators. The key to the success of the new method is to ensure that each manipulator is capable of tracking its own desired trajectory using its own position controller, while synchronizing its motion with the other manipulator motion so that the differential position error between the two manipulators is reduced to zero or kept within acceptable limits. A simplified test-bench emulating upper-limb rehabilitation was used to test the proposed coordination technique experimentally.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ADC	Analog to Digital Converter.
AFNC	Adaptive Fuzzy Neural Controller.
ANFIS	Adaptive Network-Based Fuzzy Inference System.
ARMA	Auto Regressive with Moving Average.
BP	Back-Propagation.
CAD	Computer-Aided Design.
COA	Centre Of Area.
CRS	Cooperating Robot System.
CSTR	Continuously Stirred Tank Reactor.
D	Derivative.
DC	Direct Current.
EC	European Commission.
EU	European Union.
FAM	Fuzzy Associative Memory.
FCE	Fuzzy Control Element.
FCM	Fuzzy C-Mean clustering.
FDCE	Fuzzy Derivative Control Element.
FEL	Feedback Error Learning.
FFNN	Feedforward Fuzzy Neural Network.
FICE	Fuzzy Integral Control Element.
FLC	Fuzzy Logic Controller.
FLS	Fuzzy Logic System.
FNN	Fuzzy Neural Network.
FPCE	Fuzzy Proportional Control Element.
FPID	Fuzzy Proportional-Integral-Derivative Controller.
GMP	Generalised Modus Ponens.
GMT	Generalised Modus Tolens.

HPFC	Hybrid Position/Force Control.
I	Integral.
IMC	Internal Model Control.
KBS	Knowledge Base System.
MF	Membership Function.
MIMO	Multi-Input Multi-Output.
MLT	Machine Learning Techniques.
MOM	Mean Of Maxima.
NB	Negative Big.
NL	Negative Large.
NN	Neural Network.
NNC	Neural-Network Controller.
NS	Negative Small.
P	Proportional.
PB	Positive Big.
PD	Proportional-Derivative.
PID	Proportional-Integral-Derivative.
PL	Positive Large.
PS	Positive Small.
RC	Regularity Criterion.
RFNN	Recurrent Fuzzy Neural Network.
SE	Seed Example.
SISO	Single-Input Single-Output.
SLS	Systematic Laser Sintering.
SOC	Self-Organising Controller.
SOFNN	Self-Organising Fuzzy Neural Network.
TDOF	Two-Degree-Of-Freedom.
TS-model	Takagi-Sugeno's Model.
ZE	Zero Equal.

NOMENCLATURE

CHAPTER (2)

A_i, B_i, and C_i	Linguistic terms of the linguistic variables.
A', B', and C'	Specific linguistic terms of the linguistic variables.
a_i, b_i, and c_i	Parameter set of the i^{th} node at the first layer of ANFIS network.
E	Error function.
f and a	Net input of a node and the node activation function.
F	End-effector equivalent force vector.
$F(\theta, \dot{\theta})$	An $n \times 1$ vector of friction of robot manipulator.
$G(\theta)$	An $n \times 1$ vector of gravity terms of robot manipulator.
$J(\theta)$	Jacobean matrix of robot manipulator.
$J^{-1}(\theta)$	Inverse Jacobean matrix of robot manipulator.
$J^{T}(\theta)$	Transposed Jacobean matrix of robot manipulator.
K_p	Position controller gain.
K_v	Velocity controller gain.
$M(\theta)$	An $n \times n$ inertia matrix of robot manipulator.
$M^{j}_{X_i}$	Membership function of the j^{th} term at the term set describing the i^{th} fuzzy variable.
m_{ij} and σ_{ij}	Centre (or mean) and width (or variance) of Gaussian function.
O_i^n	Activation function of the i^{th} node at the n^{th} layer of ANFIS network.

p_i, q_i, and r_i	Parameter set of consequent linear equation of TS-model based FLS.
R_i	i^{th} Fuzzy rule.
[S]	Control mode diagonal matrix with ones and zeros.
T_i	i^{th} Joint torque of robot manipulator.
τ_d	An $n \times 1$ vector of unknown terms that represents joint load torques of robot manipulator.
U_i	i^{th} Maximum point at the universe U.
U, V, and W	Universes of discourse corresponding to the linguistic variables.
u_i^n	Input of the n^{th} layer node corresponding to the i^{th} fuzzy variable.
u and σ	Centre (or mean) and width (or variance) of Gaussian function.
$V(\theta,\dot{\theta})$	An $n \times 1$ vector of centrifugal and coriolis terms of a manipulator.
w	Adjustable free parameter.
w_j	The point in the j^{th} quantisation level in a universe W at which $\mu(w)$ achieves its maximum value.
w_o	A crisp output of a fuzzy system.
w_{ij}^n	Link weight at the n^{th} layer.
Δw	Change in weight w.
x, y, and z	Linguistic variables.
x_o, y_o, and z_o	Crisp inputs and crisp output.
X_d	Desired Cartesian position vector.
δX	Cartesian position error vector.

$y(t)$ and $y_{net}(t)$	Desired output and the current network output.
α	Choice parameter.
α_i	i^{th} Rule firing strength.
$\alpha_i^{\bullet}$	i^{th} Rule firing strength in case of using the product compositional operator.
$\alpha_i^{\wedge}$	i^{th} Rule firing strength in case of using the min compositional operator.
$\theta_i(t)$	i^{th} Joint displacement of robot manipulator.
$\dot{\theta}_i(t)$	i^{th} Joint velocity of robot manipulator.
$\ddot{\theta}_i(t)$	i^{th} Joint acceleration of robot manipulator.
$\delta\theta$	Joint position error vector.
$\mu_A(x)$	Membership degree of variable x to fuzzy membership function A.
$\mu_{Ri} \equiv \mu_{(Ai \text{ and } Bi \rightarrow Ci)}$	Fuzzy implication (relation).
$\mu(w)$	Membership function defined over the universe W.
η	Learning rate.
$\rightarrow$	Fuzzy implication function.
o	Compositional operator.
*	Operator that represents a triangular norm.
+	Operator that represents a triangular co-norm.
X	Cartesian product.
^	Minimum operator.

CHAPTER (3)

$A^1, A^2, \ldots, A^m$	m^{th} Attribute (equivalent to linguistic variables).

a_i	i^{th} Joint axis normal distance from previous joint axis.
a_i and $\bar{a}_i$	i^{th} Argument of *softmin* and *softmax* functions and its complement.
Cdt_i	i^{th} condition on i^{th} attribute.
C_E	Class (output) value.
C_{rule}	Rule class value.
$D_{example1\ \&example2}$	Measure for the distance between any two examples.
d_i	i^{th} Joint coordinates normal distance from previous joint coordinates.
E	Example data (input record).
E	Error function.
F_{SE}	Gaussian output membership function of seed example.
f_{ij}^2, a_{ij}^2, and w_{ij}^2	Aggregation function, activation function and weight link of the j^{th} node of the i^{th} group of nodes at layer two.
f_r^3 and a_r^3	Aggregation function and activation function of the r^{th} rule node at layer three.
f_{ij}^4, u_{ij}^4, and a_{ij}^4	Aggregation function, net input and activation function of the j^{th} node of the i^{th} group of nodes at layer four.
f_{ni}^5, u_{ni}^5, and a_{ni}^5	Aggregation function, net input and activation function of the i^{th} numerator node at layer five.
f_{di}^5, u_{di}^5, and a_{di}^5	Aggregation function, net input and activation function of the i^{th} denominator node at layer five.
f_i^6, y_i, and a_i^6	Aggregation function, crisp output and activation function of the i^{th} output node at layer six.
$f^k(.)$ and $a^k(.)$	Aggregation function and activation function of a node at the k^{th} layer.

$J(\theta)$	Jacobean matrix of robot manipulator.
L	Number of output variables.
M1,M2,…, M6	Robot manipulator link mass.
m_{ij} and σ_{ij}	Centre (or mean) and width (or variance) of the Gaussian membership function.
n	Number of input linguistic variables.
o_i	i^{th} Node output.
P	Number of rules sharing the same consequent.
q	Number of inputs for a particular rule.
$\dot{r}(t)$	End-effector displacement in Cartesian coordinates.
Softmin	Soft minimum function.
Sofmax	Soft maximum function.
T_i^k	i^{th} Joint torque of robot manipulator at time interval k.
u_i	i^{th} Node input.
$u(k)$	System input at time interval k.
v_i^k	i^{th} Joint speed of robot manipulator at time interval k.
V_{SE}^{output}	Output class value of seed example.
$V_{\min}^i$ and $V_{\max}^i$	i^{th} Attribute range of values (equivalent to linguistic membership functions).
W	Adjustable free parameter of the proposed neuro-fuzzy network.
Δw	Change in weight w.
w_{nij}^5	Link weight that connects the i^{th} output numerator node to the output term nodes.

w_{dij}^5	Link weight that connects the i^{th} output denominator node to the output term nodes.
w_{ni}^6 and w_{di}^6	Link weights associated with each output variable node at layer six.
x_i, f_i^1, u_i^1, a_i^1, and w_i^1	The i^{th} input, aggregation function, net input, activation function and weight link of the i^{th} node at layer one of the proposed neuro-fuzzy network.
x, y, and z	End-effector Cartesian position.
$y(k)$	System output at time interval k.
$y(t)$ and $y_{net}(t)$	Desired output and current network output.
α_i	i^{th} joint axis rotation from previous joint axis.
β_{ij}	Characteristic value for the sigmoidal function.
θ_i^k	i^{th} Joint angle of robot manipulator at time interval k.
δ_{ni}^6 and δ_{di}^6	Propagated error from layer six to numerator and denominator nodes at layer five.
δ_{ij}^5	Propagated error from layer five to the j^{th} node at the i^{th} term set at layer four.
δ_{ri}^4	Propagated error from a node at the i^{th} output term set at layer four to the r^{th} rule node at layer three.
δ_r^4	Total propagated error from layer four to the r^{th} rule node at layer three.
δ_{ijm}^3	Propagated error from the m^{th} rule node of the rule nodes that share the same j^{th} term node at the i^{th} input term set at layer two.
δ_{ij}^3	Total propagated error from layer three to the j^{th} term node at the i^{th} term set at layer two.
δ_{ij}^2	Propagated error from the j^{th} term node at the i^{th} term set at layer two to the i^{th} input node at layer one.

δ_i^2	Total propagated error from layer two to the i^{th} input node at layer one.
μ_{A_i}	Membership degree to fuzzy membership A_i.
η	Learning rate.
ζ	Constant that determines the degree of softening and the accuracy of the approximation of *fmin* and *fmax* functions.

CHAPTER (4)

Cop_{Ri}	FPCE output membership function centre value for i^{th} rule.
Cod_{Ri}	FDCE output membership function centre value for i^{th} rule.
Coi_{Ri}	FICE output membership function centre value for i^{th} rule.
$E_{tot}(w_k)$	Total error at the k^{th} iteration.
e_1 and e_2	Two normalized input error variables of the FPID controller.
e_1^i and e_2^i	Current and past position errors at link *i.*
err_1 and err_2	Two input error variables of the FPID controller.
e(kt) and *e(kt-t)*	Present and previous time sample error values.
$e(kt) = e_1$	Discrete form of normalized current input error variables of the FPID controller.
$e(kt-t) = e_2$	Discrete form of normalized past input error variables of the FPID controller.
$err(kt) = err_1$	Discrete form of current input error variables of the FPID controller.
$err(kt-t) = err_2$	Discrete form of past input error variables of the FPID controller.

fmin	Two argument *softmin* function.
fmax	Two argument *softmax* function.
f_P, f_D, and f_I	Proportional, Derivative, and Incremental Integral functions to be implemented using the FPID controller.
h	Membership degree.
h_{e1} and h_{e2}	Two arguments of *fmin* function.
h_{p1} and h_{p2}	Two arguments of *fmax* function.
K_P, K_D, and K_I	Proportional, Derivative and Integral gains of discrete form PID controller.
K_{NP}, K_{ND}, and K_{NI}	Equivalent nonlinear Proportional, Derivative and Integral gains that can be defined according to input condition.
$K_{e_2} = K_{e_1} = K_e$	Scaling factors corresponding to the two input variables.
K_U	Output scaling factor of the FPID controller.
L	Distance between the centres of each two consecutive membership functions of the input universes of the FPID controller.
L_P, L_D, and L_I	Distance between the centres of each two consecutive membership functions of the Proportional, Derivative and Integral fuzzy control elements of the FPID controller.
l	The total link numbers of the robot.
Mf	Membership function.
[PL, PS, ZE, NS, NL]	Term sets of the normalised input variables and normalised output variables of the FPID controller.
T	Sampling time.

T_{FF}^{i}	Feedforward controller torque at robot link *i.*
T_{FB}^{i}	Feedback controller torque at robot link *i.*
T_{tot}^{i}	Total torque at robot link *i.*
U_I	Output of the integral element of the FPID controller.
$U_I(kt\text{-}t)$	The past output of the integral controller element.
$\Delta U_I(kt)$	Incremental value of integral controller element.
$U_P(kt), U_D(kt)$, and $U_I(kt)$	Proportional, Derivative and Integral control actions of discrete form PID controller.
$U_{PID}(kt)$	Discrete form of the FPID controller total output.
w_k	Vector of weight values after the k^{th} iteration.
Δw_k	Change in weight vector.
$\mu_{Ri}(e_1)$	Membership degree of the present error to i^{th} rule.
$\mu_{Ri}(e_2)$	Membership degree of the past error to i^{th} rule.
η	Learning rate.

CHAPTER (5)

D	External disturbances acting at the plant inputs.
F	Low-pass pre-filter.
R	Vectors of reference inputs to the system.
U	Vectors of control inputs to the plant.
Y	Vectors of the system outputs.
X_d	Desired Cartesian position vector.
X_m	Measured Cartesian position vector.

Φ_c	Controller equivalent transfer function.
Φ^D	Internal model controller overall disturbance transfer function.
Φ^{Φ_p}	Existing joint-based controller disturbance transfer function.
Φ_{FF}	Forward path controller equivalent transfer function.
Φ_{FB}	Feedback path controller equivalent transfer function.
Φ_{IK}	Inverse kinematics neuro-fuzzy network equivalent transfer function.
Φ_K	Forward kinematics equations equivalent transfer function.
Φ_m	Plant model.
Φ_p	Plant.
$\Phi_p^{\Phi_p}$	Existing joint-based controller equivalent transfer function.
$\Delta\Phi_p$	Unmodelled dynamics and/or parameters additive uncertainties of the plant.
$\delta\Phi_p$	Unmodelled dynamics and/or parameters multiplicative uncertainties of the plant.
Φ^R	Overall input/output transfer function.
θ_m	Measured joints position vector.
θ_d	Desired joints position vector.
$\xi_{\partial\Phi_p}^{\Phi_p}$	Existing joint-based controller closed loop multiplicative sensitivity function.
$\xi_{\partial\Phi_p}^{\Phi^R}$	Internal model controller overall closed loop multiplicative sensitivity function.
$\xi_{\Delta\Phi_p}^{\Phi_p}$	Existing joint-based controller closed loop additive sensitivity function.

$\xi_{\Delta\Phi_p}^{\Phi^R}$	Internal model controller overall closed loop additive sensitivity function.

CHAPTER (6)

A	Constant vector.
$e_i(t)$	Cartesian error vector of robot manipulator *i*.
$e_v(t)$	Connection-vector error.
$e_s(t)$	Synchronization error.
$f(x_i)$	Synchronization function.
ε	Tolerance bandwidth for the synchronization system.
$l_1(t)$ and $l_2(t)$	Displacement of the mobile robot two driving wheels.
$R_1(t)$ and $R_2(t)$	Radii of the desired curves that the mobile robot two driving wheels follow.
$r_{1m}(t)$ and $r_{2m}(t)$	Reference modification signal for the two cooperating robots.
$s^d(t)$	Reference connection-vector trajectories.
$s(t)$	Actual connection-vector trajectories.
$x_i(t)$	Cartesian coordinates of robot manipulator *i*.
$x_i^d(t)$	Desired Cartesian position trajectory of robot manipulator *i*.
$\mu_{e1}(k)$ and $\mu_{e2}(k)$	Membership degree of e_1 and e_2.
$\mu_{es}(k)$	Membership degree of e_s.
$\mu_{r1m}(k)$ and $\mu_{r2m}(k)$	Membership degree of r_{1m} and r_{2m}.

CHAPTER 1

Introduction

The complexity of industrial processes has increased dramatically in the last few decades. This in turn limits the use of conventional control design techniques because their success is based mainly upon knowledge of the process mathematical model. In fact, it is often impossible to obtain exact mathematical models even for the simplest dynamics processes. This is because most mathematical modelling techniques rely upon a linearization of the process dynamics around a certain operating point. On the other hand, experienced operators of an industrial process can efficiently control such a process to achieve the required performance. These operators know nothing about process mathematical models or complex control theories. Their strategy when controlling such a process is mainly based on heuristics which can be expressed as "*if* antecedent *then* consequent" rules. The antecedent and consequent are vague and involve fuzzy expressions such as faster, small, approximately, large, etc., with which traditional logical systems cannot deal. In 1968, Lotfi A. Zadeh introduced his pioneering fuzzy set theory which offers a mathematical framework that can deal with such vague information. In his subsequent work, [Zadeh, 1973] explained how vague logical statements can be employed within a computational method to enable inferences to be derived from vague data. It was realised that this method could be applied to model and control complex systems [Mamdani, 1974], such as robotic manipulator systems [Kazemian, 2002].

1.1. Motivation

Constructing an efficient rule base is the main problem in the development of fuzzy logic controllers. Due to difficulties in knowledge acquisition, it often requires great effort to construct a fuzzy rule base using a heuristic method. This has led to attempts to extract these rules from numerical observation data. One approach seeks to capture the operational knowledge of human experts during normal process operation. This approach records the states of the process under study and the human operator's control actions as input and output data pairs and extracts fuzzy control rules from the recorded data pairs. An alternative approach utilises a fuzzy model of the process. To construct such a model, the process is stimulated and both the stimulus and its response are recorded. Fuzzy rules are then extracted from the recorded data. An efficient method for extracting rules from the recorded data is via the use of machine learning techniques (MLT). A particularly interesting type of machine learning technique is symbolic inductive learning, because the models it creates have a structure similar to that employed in human reasoning (in the form of *if-then* rules). The generated rules can be regarded as fuzzy rules. In this way, fuzzy rules can be generated to form the inverse model of the system and then used as a controller.

Following the pioneering work of Zadeh, many applications of fuzzy control for industrial processes were presented. There has also been increasing interest in the application of neural networks (NN) for modelling and control of dynamics systems [Pham and Liu, 1993; 1995; and 1996]. NN possess interesting and attractive features such as online learning. A NN can learn a mapping between input-output spaces and

form an associative memory that retrieves the appropriate output when presented with an input. It can also estimate an output when activated with previously unseen inputs. Subsequent research has explored NN capabilities for function approximation and adaptive control. Recently, the application of the learning abilities of NN to automate the realisation of fuzzy logic systems (FLS) has become a very active research area [Takagi and Hayashi, 1991]. This integration brings the low-level learning and the computation power of NN to FLS and brings the high-level human-like thinking and reasoning of FLS to NN. This integration has developed new intelligent systems called fuzzy neural networks (FNN). FNN provide a new method for the realisation of intelligent control systems. The ability of such systems to learn or to adapt their control policy according to its past experience makes them an ideal solution for all those applications characterised by time-changing dynamics and unstructured operating conditions.

The use of robots in many industrial applications is becoming more commonplace due to the necessity to increase productivity and requirement to reduce cost. Robot manipulators are multi-input multi-output (MIMO) coupled dynamics systems. Mathematical modelling or mathematical model-based control techniques for such systems are very complex, and very difficult to be implemented [Appendix A]. With the help of new model-free techniques such as fuzzy and neuro-fuzzy mechanisms, modelling and efficient control of such complex systems can be achieved without the need for tedious mathematical formulation processes. Currently industrial robots are generally used for tasks that involve one-handed manipulation. Inevitably some applications, for example handling large, heavy, or awkwardly shaped objects, may

require two-handed manipulation. Utilisation of robotics for medical applications is also an increasing demand, especially in the area of rehabilitation in relation to the shortage of physiotherapists. A Physiotherapist usually uses both arms to perform upper-limb rehabilitation. A robotic cell would likewise require two robot arms to perform the task effectively. In this case, efficient control and coordination techniques have to be developed.

The use of a robotic cell incorporating intelligent identification and control techniques for upper-limb motion therapy for patients with neuro-motor impairments is particularly attractive for the following reasons:

- Patients lack appropriate personalised motion therapy because of the limited numbers and unavailability of physiotherapists.
- Current commercial rehabilitation robotic systems do not utilize the knowledge, dexterity and skill of the physiotherapist.
- Current commercial rehabilitation robotic systems are not adaptable for motion parameters.
- Physiotherapists may be unable to conduct motion therapy for a long period of time, with very slow speed and high torque requirements in some cases.
- Rehabilitation robotics has achieved little market success probably because research has resulted in custom-made equipments, which requires strong marketing both among manufacturers and users.

1.2. Research Objectives

The work reported in this thesis aimed to design and develop intelligent neuro-fuzzy control systems for robot manipulators using machine learning techniques, fuzzy logic controllers, and fuzzy neural networks in a systematic manner. The main target is to integrate these techniques to achieve a mathematical-model-free manipulator adaptive control system capable of adapting its parameters to cope with the variations in the dynamics characteristics of the load attached to the manipulator. The final objective is to utilise one or two coordinated robots as a working cell to perform upper-limb rehabilitation application. The control and coordination systems must be able to drive the robots to perform the basic actions that a physiotherapist usually carries out in these situations. In this respect, the specifications of the control system are:

- Efficiently perform both inverse kinematics and inverse dynamics calculations using neuro-fuzzy techniques from numerical input/output observation data.
- Possess self-learning and adaptive capabilities to achieve the required trajectories with unspecified loading conditions from the human-arm resistance to motion.
- Effectively coordinate between two robot manipulators to perform upper-limb rehabilitation within acceptable accuracy.

The overall targets of this research can be summarised as follows:

- To develop intelligent adaptive models that represent both the inverse kinematics and inverse dynamics of the robot manipulator.
- To develop a virtual dynamics model for the robot manipulator.

- To develop intelligent adaptive robotic control techniques.
- To develop an efficient synchronisation technique for two robot manipulators.
- To test the developed control and coordination technique on a simplified experimental test-bench.

To achieve the above targets, several components needed to be designed, developed and then integrated to form the proposed control system. For this purpose, an inductive fuzzy learning technique *dynafuzz* Bigot [2003] is modified and applied for fuzzy rule generation during the offline structure learning phase. A new differentiable fuzzy neural network termed *dynafuzznn* is developed to construct the feedforward robotic inverse control system. A modified Fuzzy-PID-like incremental feedback controller is also used as the servo-controller for each link of the robot. A new modified neuro-fuzzy Cartesian internal model control technique for robotic manipulators is developed to construct an adaptive Cartesian control of the robot arm. Also, a new simple motion synchronisation technique is developed to achieve motion coordination between two robot arms. The proposed control system is applied to a virtual dynamics model of the Puma 560® industrial robot arm for visual inspection of the proposed strategy. Finally, a simplified rapid prototype test-bench was constructed to experimentally investigate the proposed method.

1.3. Outline of the Thesis

The remainder of the thesis is organised as follows:

Chapter 2 presents background information on fuzzy logic systems and fuzzy neural networks and their applications in systems modelling and control of robotic manipulators. In this chapter, the basic components of FLS are described. Different techniques for the adaptive tuning of the free parameters of FLS are presented. Different types of FNN are also discussed. Finally, robot manipulator kinematics, dynamics, control, and coordination techniques using conventional, intelligent, fuzzy, neural, and neuro-fuzzy systems are discussed.

Chapter 3 first reviews different classes of existing neuro-fuzzy robot modelling techniques. Then, it proposes a method for virtual dynamics modelling and data collection from the motion of the robot arm along random trajectories. Next, it proposes a modelling technique based upon machine learning for automatic fuzzy rule generation from observation data. Then, it proposes the use of these rules in a full differentiable fuzzy neural network termed *dynafuzznn* to achieve online adaptation of the model. Finally, to investigate the proposed structure performance, the results for robot inverse kinematics and inverse dynamics FNN for Puma 560® are compared with the targeted outputs at the end of the chapter.

Chapter 4 first reviews different types of existing joint-based robot control techniques. Then, it introduces the structure of the proposed controller. The inverse dynamics FNN developed in chapter (3) is used as the feedforward controller to compensate for the

dynamics interaction in the robot structure in addition to a Fuzzy-PID-like incremental servo-controller for each robot link. Feedback-error learning scheme is used to provide an online adaptation mechanism for the proposed controller. The control system is used to drive the virtual model of the Puma 560® robot to follow certain joint trajectories. Finally, the obtained results compared with conventional-PID link controller's results are presented.

Chapter 5 first reviews different classes of existing manipulators Cartesian control techniques. Then, it introduces a brief analysis of conventional internal model control structures. Furthermore, it introduces a new modified neuro-fuzzy Cartesian internal model control technique for robotic manipulators. Next, it presents the simulation results when the control structure is used to drive the virtual model of the Puma 560® robot to follow Cartesian end-effector trajectories. Then, it discusses the idea of upper-limb rehabilitation using robotic manipulators. Finally, it introduces the obtained results when the proposed joint-based and Cartesian internal model controllers are used to control one robot manipulator while performing upper-limb rehabilitation.

Chapter 6 first reviews different classes of existing robot position coordination techniques. Then, it explains the synchronization function notation. Furthermore, it introduces the structure of the proposed coordination scheme for two position-controlled robot manipulators. The coordination scheme is based on maintaining certain kinematic relationships between the robot manipulators using motion synchronisation. Finally, the chapter presents a test for the proposed control and position coordination technique using a simplified rapid-prototype test-bench for the upper-limb

rehabilitation cell formed by two 2-link planar robots linked to a simplified upper-limb model. The experimental results for the actual trajectories are presented and compared with the targeted trajectories at the end of the chapter.

Finally, Chapter 7 summarises the conclusions and contributions of the research, and gives suggestions for further investigation.

CHAPTER 2

Overview of Fuzzy and Neuro-Fuzzy Techniques

The traditional approach to formal modelling, reasoning and computation is mostly deterministic and precise rather than uncertain or vague. In conventional logic, for instance, a statement can be true or false and nothing in between. In set theory, an element can either belong to a set or not, and in optimisation, a solution is either feasible or not. Real situations, on the other hand, are very often uncertain or vague in a number of ways. One familiar type of uncertainty is that, due to lack of information, the future state of the system might not be known completely. This category is called stochastic uncertainty, and has been treated appropriately in theory and statistics. Despite the ambiguity of the system state, in stochastic uncertainty it is assumed that the meaning of statements and events is clearly defined. There is however another type of vagueness concerning the description of the semantic meaning of events, phenomena or statements themselves, which can be called *fuzziness*. Fuzziness is found in many areas of daily life, particularly those in which human judgement, evaluation and decision are relevant. For example, there are fuzzy terms that are well-known in science and engineering such as linear approximation, small neighbourhood, and ill-conditioned matrix [Pasino and Yurkovich, 1998].

Applications of formal methods to describe real world phenomena may be limited to simple systems or at least viewed as an approximation of more complex situations. In

analytical modelling, for instance, based on classical set theory, there are two difficulties:

- The first is due to the excessive complexity of the situation being modelled so that either it is not possible to formulate the mathematical model, or the model is too complicated to be implemented in practice.
- The second inconvenience consists of the indeterminacy caused by the inability to differentiate events in real situations exactly and hence, inability to define system behaviour in a precise form.

Real situations are very often not "crisp" and they cannot be described precisely. An underlying philosophy of the theory of fuzzy sets is to provide a strict mathematical framework, where imprecise conceptual phenomena in modelling, decision-making, and control may be precisely and rigorously studied. In the particular field of application concerned with systems modelling and control, there are many difficulties that are commonly experienced by practicing engineers. For instance, it is generally difficult to accurately model a complex process using a mathematical model. Furthermore, it is common knowledge that the performance of some processes can be considerably improved through control actions (tuning actions in particular) provided by an experienced and skilled operator. Although some of these actions have been recently formulated using conventional control algorithms, it seems that the key elements in human thinking are not numbers, but labels of not crisp but fuzzy sets, that is, classes of objects in which transition from membership to non-membership is gradual rather than abrupt.

The fuzzy methodology of fuzzy-logic modelling and control, based on fuzzy set theory and fuzzy logic, appears promising when the phenomena are too complex for analysis by conventional quantitative techniques, and when the available sources of information are interpreted qualitatively, inexactly or uncertainly. Thus, fuzzy-logic modelling and control may be viewed as a step towards a rapprochement between the conventional, precise analytical approach and human-like decision making.

Artificial Neural Networks (NN) are made up of simple, highly interconnected processing elements called neurons. Each of these neurons performs two main functions: aggregation of its inputs from other neurons or the external environment and generation of an output from the aggregated inputs. The output from a neuron is fed to other neurons to which it is connected via weighted links. Through this simple structure, NN have been shown to be able to approximate most continuous functions to any degree of accuracy, by the choice of an appropriate neuron structure, activation functions, and learning algorithm [Tsoukalas and Uhrig, 1997].

Fuzzy Logic Systems (FLS) and Neural Networks (NN) have played an important role in the development of intelligent control systems. FLS have the ability to deal with system uncertainty using their logically oriented reasoning techniques. NN handle system complexity by employing their particular structure and learning methods. A promising approach to obtaining the benefits of both NN and FLS is to combine them into an integrated system termed a Fuzzy Neural Network (FNN) or neuro-fuzzy system. This integration brings the low-level learning and computation powers of NNs to FLS and the high-level human-like thinking and reasoning of FLS to NNs. Many

researchers have studied different techniques to employ FLS and FNN together with control theory to build high performance controllers for complex systems that involve imprecise data and nonlinear dynamics such as for the case of robot manipulators.

This chapter presents background information on FLS and FNN and their applications in robotic systems modelling and control. The basic components of FLS and their design parameters are described. Furthermore, different techniques for the adaptive tuning of FLS are discussed. Various types of FNN are presented based on different structures and learning algorithms. Finally, applications of FLS and FNN in robotic systems modelling and control are outlined.

The remainder of this chapter is organised as follows. Section 2.1 reviews the basic structure and design elements of FLS. Section 2.2 examines the basic structure and design elements of FNN. Section 2.3 describes applications of FLS and FNN in control. Section 2.4 describes applications of FLS and FNN in robots modelling. Section 2.5 describes the applications of FLS and FNN in robots control. Section 2.6 describes applications of FLS and FNN in robots coordination. Section 2.7 gives a summary of the chapter.

2.1. FLS Basic Structure and Design Elements

The basic structure of a FLS consists of four main components [Lee, 1990a]. These are the fuzzification process, knowledge base, decision-making logic, and defuzzification

process as shown in figure (2.1). In the following sections, the function and design parameters of each of these components are presented.

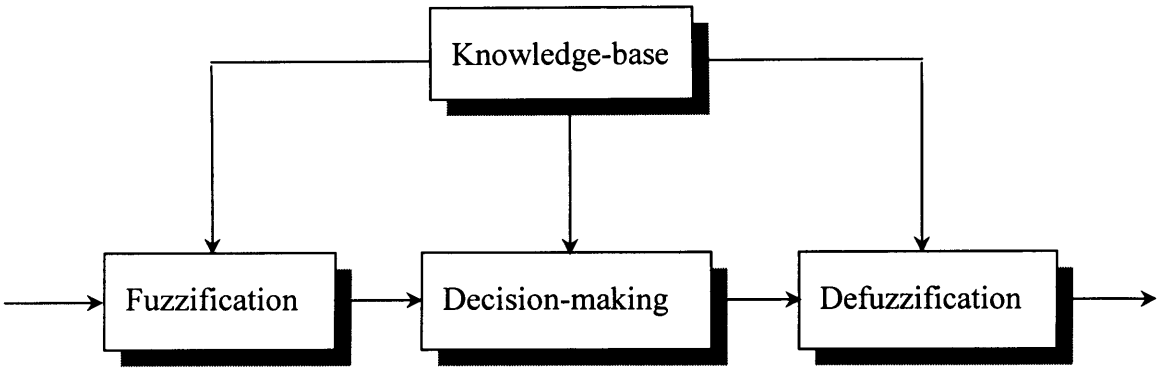


Figure (2.1). Basic configuration of a fuzzy logic system.

2.1.1. Fuzzification Process

Fuzzification is related to the vagueness and imprecision of natural languages. It is a mapping that transforms measurements into a linguistic value, and hence it could be defined as a mapping from an observed measurement space into a subjective feature space. In fuzzy control applications, the observed data is usually crisp. Since the processed data in FLS are based on fuzzy sets, fuzzification is necessary during the early stages to transform the observed crisp data into fuzzy sets. A commonly used fuzzification approach is to transform this crisp data into fuzzy singletons (fuzzy sets comprising a single element) within a certain universe of discourse. The transformation process begins with the normalisation or scaling of the crisp measurements to a certain bounded range say [-1, +1] using suitable scaling factors. The purpose of the normalisation process is to map the crisp input data into a universe of discourse with a

finite range. Subsequently, the fuzzification interface transforms the normalised crisp input x_o into a fuzzy set A in universe X with the membership function $\mu_A(x_o)$ equal to a value between zero and one according to the location of x_o with respect to the centre of the fuzzy set A in the universe of discourse. In general, the role of the fuzzification interface can be summarised as follows [Keller et. al., 1992]:

a) It observes the crisp input values to a FLS.

b) It performs a scale transformation (normalisation) from the measurement space into the corresponding universe of discourse.

c) It performs the fuzzification function that converts the scaled input data into fuzzy sets.

2.1.2. Knowledge Base

The knowledge base [Lee, 1990a] comprises knowledge concerning the application field and the desired control or modelling objectives. It consists of a database and a linguistic (fuzzy) rule base in the form of *if* antecedent *then* consequent. The database provides necessary definitions, which are employed to define linguistic rules and data manipulation in FLS.

2.1.2.1. Data Base

The definitions associated with the database are employed to characterise fuzzy rules and data manipulation in FLS. These definitions are subjective in nature, which reflects engineering experience and judgement. These definitions comprise the

normalisation/discretisation of a fuzzy universe of discourse, the partition of a fuzzy universe of discourse and the definition of membership functions associated with fuzzy sets. In what follows, important definitions relating to the construction of the database in FLS are discussed.

a. Normalisation/Discretisation of a fuzzy universe of discourse

The normalisation of a universe of discourse involves a priori knowledge of the input/output universe of measurements. The normalisation process is a scale transformation of the input/output universe of measurements into a normalised closed interval universe. For example, if the measured input data ranges from -8.0 to +4.5, the universe of the input measurements can be normalised by a scale transformation into a closed interval universe [-1, +1].

Discretisation of a universe of discourse is defined as the quantisation of this universe into a certain number of segments (quantisation levels). Each segment is labelled as a generic element of a discrete universe. A fuzzy set is then defined by assigning a grade of membership to each generic element of the universe.

b. Fuzzy partition of the input/output universe

A linguistic variable in the antecedent or consequent of a fuzzy rule forms a fuzzy input or output feature space respectively. The input or the output feature space of each input or output linguistic variable is defined over a certain universe of discourse. Each feature space is internally partitioned into a number of clusters or fuzzy sets that define the

term set of the input or output linguistic variables. Each fuzzy set is defined by a certain linguistic term, and usually has a meaning such as negative big (NB), negative small (NS), positive big (PB), etc. The number of partitions of the input and output feature spaces determines the maximum number of fuzzy rules that can be generated. Therefore the selection of the number of partitions influences the generated number of rules for FLS. In most applications of FLS, experience and engineering judgement are employed to choose the number of partitions of the fuzzy feature space. However, some applications follow heuristic methods for feature space partitioning as in [Abe and Lan, 1995]. Other applications employ a deterministic method, for example the Fuzzy C-Mean (FCM) method was employed to partition the fuzzy feature space in [Sugeno and Yasukawa, 1993; Wang and Langari, 1996; and Emami et. al., 1999].

c. Definition of the membership functions of fuzzy sets

There are two commonly used methods which define the membership functions of fuzzy sets depending on whether the universe of discourse is discrete or continuous [Lee, 1990b]. The first method is a numerical definition where the grade of membership in a fuzzy set is represented as a vector of numbers. The dimension of this vector depends on the number of discrete levels in the feature space. In this case, the membership function of each fuzzy set can be written as follows:

$$\mu_A(x) = [\mu_A(x_o) / x_o + \mu_A(x_1) / x_1 + \ldots\ldots\ldots\ldots + \mu_A(x_n) / x_n] \qquad (2.1)$$

where n is the number of supports of the discrete universe of discourse, x_n is the n^{th} support of the discrete universe of discourse, and $\mu_A(x_n)$ is the membership grade of the

n^{th} support in fuzzy set A as shown in figure(2.2a). The second method is a functional definition, which expresses the membership function of a fuzzy set in a functional form, typically a Gaussian, right/left sigmoidal, right/left saturation, trapezoid-shaped, or triangle-shaped function (figures (2.2b, c)). The functional definition of the Gaussian membership function, for example can be written as:

$$\mu_A(x_o) = \exp.[-(x_o - u)^2 / \sigma^2] \qquad (2.2)$$

where u and σ are respectively, the centre (or mean) and the width (or variance) of the Gaussian function as shown in figures (2.2b).

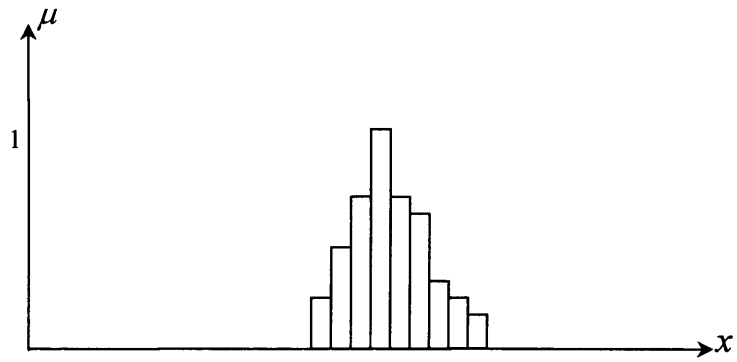


Figure (2.2a). Discretised membership function.

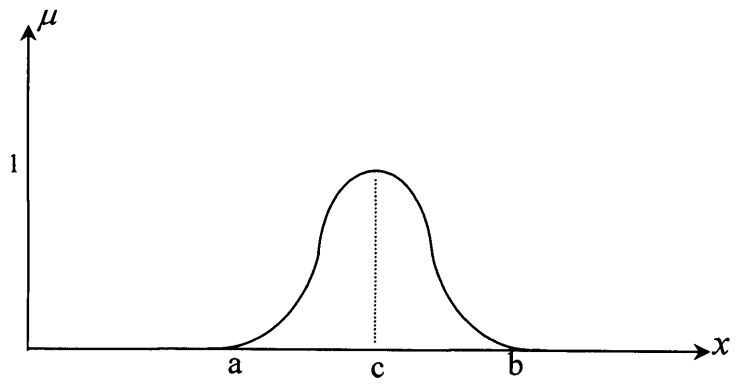


Figure (2.2b). Gaussian membership function.

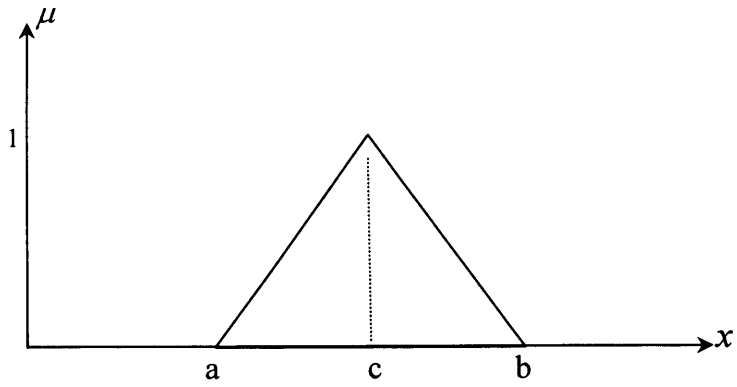


Figure (2.2c). Triangular membership function.

2.1.2.2. Rule Base

A FLS is characterised by a set of linguistic statements based on expert knowledge. The expert knowledge is usually in the form of *if - then* rules, which are easily implemented by conventional fuzzy statements in fuzzy logic. The collection of fuzzy rules that are expressed as fuzzy conditional statements forms the rule set or the rule base of a FLS. In this section, the following factors which influence the design and implementation of a fuzzy rule base are discussed: the choice of the FLS input/output variables, the approaches employed to generate fuzzy rules, and the functional implementation of fuzzy rules.

2.1.2.2.1. Choice of the FLS Input/output Variables

It is important to choose suitable input and output variables for FLS, because they influence the number of rules generated and the final performance of the system. In many applications of FLS, the selection of input/output variables relies on experience and control engineering. In some other applications, the selection is based on a deterministic method [Sugeno and Yasukawa, 1993]. In such applications the employed FLS is tested using only one of the available input variables at a time. A function of the

FLS output termed the Regularity Criterion (RC) is calculated for each variable. The variable that minimises the calculated function is chosen to be the first effective variable of the FLS. The employed FLS is tested further using the selected variable and only one of the remaining variables at each step. The second variable that minimises the criterion when employed in addition to the first selected variable is chosen to be the second effective variable. This process is continued to obtain the maximum number of effective variables, identified as when the value of RC starts to increase.

2.1.2.2.2. Derivation of the Fuzzy Rules

There are two common approaches to deriving fuzzy rules. These two approaches are not mutually exclusive, and it seems likely that a combination of them is necessary to construct an effective method of deriving fuzzy rules. The first approach is to generate fuzzy rules based on expert experience and control engineering knowledge. This approach is mainly suitable for generating fuzzy rules for diagnosis systems including fault diagnosis and medical diagnosis systems. It is also suitable for generating fuzzy control rules for Fuzzy Logic Controllers (FLC). This approach is a heuristic approach, in which the fuzzy rules are obtained mainly from human experience. A human expert has to interpret his experience as linguistic relations between the input and output variables of the FLS. This approach can be successful if the human expert can perform this interpretation. However, if the human expert cannot express his experience linguistically, then the second approach, based on the observed input/output data, can be employed. This approach can be used to generate fuzzy rules for FLC and for fuzzy process models. In the case of FLC, the fuzzy rules can be generated based on

observations of the human expert's control actions in terms of input/output data. In the case of fuzzy process models, the fuzzy rules are generated based on the process input/output data [Takagi and Sugeno, 1985; Wang and Mendel, 1992a and 1992b]. With this method, the input/output universes are partitioned into fuzzy regions, the size and shape of which are determined by experience. Then, based on the given input/output numerical data and the input/output fuzzy regions, the fuzzy rules are generated. Finally, a Fuzzy Associative Memory (FAM) bank is constructed using rules generated from the numerical data and rules obtained from expert experience as well. A disadvantage of this method is that the number of generated rules increases dramatically as the number of input variable increases.

Modern techniques based on data mining algorithms can also help in generating rules from numerical observation data. These algorithms are based on a range of technologies, from statistics to machine learning techniques, and include neural networks, genetic algorithms, inductive learning, association rules, etc. These algorithms allow the creation of different types of models that describe the patterns found in the data. The obtained model can be used as a predictive model.

One interesting type of data mining techniques, namely inductive learning algorithm, is very important, as its model structure (in the form of *if-then* rules) is similar to that employed during human reasoning. Because of this, inductive learning has become popular for classification problems. Consequently in the work of [Srinivasan et al., 1993], the ID3 inductive learning algorithm was employed to reduce the number of rules generated utilising the method proposed in [Wang and Mendel, 1992a and 1992b].

In these last examples, structure design was performed based on experience. In [Sugeno and Yasukawa, 1993], in contrast, structure identification was performed using deterministic methods. The number of necessary input/output variables was decided using the so-called Regularity Criterion (RC) and the number of clusters of fuzzy input/output variables was determined using the clustering technique known as Fuzzy C-Mean clustering (FCM). In addition, a performance index was employed to tune the free parameters of the membership functions (width and centre in the case of a Gaussian membership function). Whereas in the methods employed by Wang and Mendel, Srinivasan et al., and Sugeno and Yasukawa, the number of fuzzy regions is fixed, a method to generate fuzzy rules with variable fuzzy regions was presented in [Abe and Lan, 1995]. With this method, the fuzzy rules are extracted directly from numerical data by recursively resolving overlaps between each pair of classes. The numerical data employed to generate the fuzzy rules sometimes includes a large amount of noise and/or involuntary mistakes made by the operator. To address this, an intermediate Auto-Regressive-with-Moving-Average (ARMA) model was generated in [Zapata et al., 1999] as an alternative to directly extracting fuzzy rules from raw experimental data. This ARMA model was then employed to generate the required linguistic fuzzy model. Another rule generation method designed to deal with noisy data was introduced in [Li, 1999]. With this method, a confidence degree is given to each rule according to the frequency that it is generated during the presentation of the data. Based on this confidence degree, rules that are generated infrequently are considered less important than rules generated more often.

Using fuzzy logic theory in combination with clustering techniques such as C-mean clustering, fuzzy rule induction is proposed in order to handle noisy continuous outputs and inputs in [Bigot, 2003]. The presented algorithm allows the automatic creation of membership functions and produces accurate and compact fuzzy sets.

2.1.2.2.3. Functional Implementation of Fuzzy Rules

A rule base of a FLS consists of a set of fuzzy rules. For example, consider the following rules:

R_1: IF x is A_1 and y is B_1 THEN z is C_1

R_2: IF x is A_2 and y is B_2 THEN z is C_2

.....

.....

R_n: IF x is A_n and y is B_n THEN z is C_n

where x, y and z are linguistic variables and A_i, B_i and C_i are linguistic terms (fuzzy sets) of the linguistic variables x, y and z in the universes of discourse U, V and W respectively, with i = 1, 2,...., n. The i^{th} fuzzy rule is implemented by a fuzzy implication (fuzzy relation) R_i. This fuzzy relation is a fuzzy set in U×V×W and is defined for all u ∈ U, v ∈ V and w ∈ W as follows:

$$R_i = \{[(u,v,w) , \mu_{Ri}(u,v,w)] \mid (u,v,w) \in (U \times V \times W)\} \quad (2.3)$$

and its membership function is given by:

$$\mu_{Ri}(u, v, w) = \mu_{(Ai\ and\ Bi \rightarrow Ci)}(u, v, w) = [\mu_{Ai}(u)\ and\ \mu_{Bi}(v)] \rightarrow \mu_{Ci}(w) \qquad (2.4)$$

where "A_i *and* B_i" is a fuzzy set in the Cartesian product space U×V which can be defined based on the interpretation of the sentence connective "*and*" and R_i = (A_i *and* B_i) → C_i is a fuzzy implication (relation) in the Cartesian product space U×V×W which can be defined based on the interpretation of the sentence connective "*and*" and the definition of the fuzzy implication function→. Since a fuzzy rule represents a fuzzy relation, the overall behaviour of a FLS can be characterised by a single fuzzy relation that is the combination of the fuzzy relations in the rule base. This combination can be defined based on the definition of the sentence connective "*also*". In the following, the fuzzy implication function → and the sentence connectives "*and*" and "*also*" are defined.

Many implication functions have been proposed. In general, they can be classified into two categories. The first category is the fuzzy conjunction that is defined for all u ∈ U and v ∈ V as follows:

$$A \rightarrow B = \int_{U\times V} \mu_A(u) * \mu_B(v)/(u,v) \qquad (2.5)$$

where A and B are fuzzy sets in the universes of discourse U and V respectively, A → B is a fuzzy implication in the Cartesian product space U×V and * is an operator that

represents a triangular norm [Keller et al., 1992]. The second category is the fuzzy disjunction that is defined for all u ∈ U and v ∈ V as follows:

$$A \rightarrow B = \int_{U \times V} \mu_A(u) + \mu_B(v)/(u,v) \tag{2.6}$$

where A and B are fuzzy sets in the universes of discourse U and V respectively, A → B is a fuzzy implication in the Cartesian product space U×V and + is an operator that represents a triangular co-norm [Keller et al., 1992]. Based on these definitions, many fuzzy implication functions may be generated using different triangular norms and co-norms. In general, using the fuzzy conjunction along with the intersection and algebraic product triangular norms, the two commonly used fuzzy implication functions can be written as follows:

$$A \rightarrow B = \int \mu_A(u) \wedge \mu_B(v)/(u,v)$$

$$A \rightarrow B = \int_{U \times V} \mu_A(u) \wedge \mu_B(v)/(u,v) \tag{2.7}$$

where $\mu_A(u) \wedge \mu_B(v) = \min[\mu_A(u), \mu_B(v)]$ is the intersection triangular norm.

$$A \rightarrow B = \int_{UXV} \mu_A(u) \cdot \mu_B(v)/(u,v) \tag{2.8}$$

where $\mu_A(u) \cdot \mu_B(v) = \mu_A(u)\mu_B(v)$ is the algebraic product triangular norm.

In most existing FLS, the sentence connective "*and*" is usually implemented as a fuzzy conjunction in a Cartesian product space [Lee, 1990b]. As an illustration, for two fuzzy sets A and B in the universes of discourse U and V respectively, "A *and* B" is defined by a fuzzy set A×B in the Cartesian product space U×V. If the sentence connective "*and*" is interpreted using the intersection triangular norm, the membership function of this fuzzy set is expressed as:

$$\mu_{A\times B}(U\times V) = \min\left[\mu_A(u), \mu_B(v)\right] \tag{2.9}$$

Alternatively, if the sentence connective "*and*" is interpreted using the algebraic product triangular norm, the membership function of this fuzzy set is expressed as follows:

$$\mu_{A\times B}(U\times V) = \mu_A(u)\cdot\mu_B(v) \tag{2.10}$$

On the other hand, the interpretation of the sentence connective "*also*" is based on the fact that different orders of fuzzy rules in the rule base should not influence the overall behaviour of a FLS. This requires that the sentence connective "*also*" have the properties of commutatively and associativity. It has been reported in [Lee, 1990b] that the operators in triangular norms and co-norms (intersection, algebraic product, bounded product, union, algebraic sum, bounded sum, etc.) possess these properties and thus qualify as candidates for the interpretation of the connective "*also*". However, investigations in [Lee, 1990b], concerning FLS characteristics using different

interpretations of triangular norms and co-norms concluded that the interpretation of the connective "also" as the union operator Y yielded the best results. The union operator Y is a triangular co-norm defined using the max function [Lee, 1990b].

2.1.3. Decision Making Logic

FLS may be regarded as a means of emulating a skilled human operator through an inference engine. More generally, the FLS inference engine may be viewed as another step towards modelling the human decision making process within the conceptual framework of fuzzy logic and approximate reasoning. The function of the FLS inference engine is to infer recommended solutions from fuzzy rules relevant to given inputs based on the employed inference strategy and inference mechanism.

2.1.3.1. FLS Inference Strategies

Generally, there are two important inference strategies in approximate reasoning. They are generalised modus ponens (GMP) and generalised modus tollens (GMT). Specifically, consider the following rule:

IF x is A THEN y is B

where x and y are linguistic variables and A and B are linguistic terms of the linguistic variables x and y in the universes of discourse U and V respectively. The GMP strategy can be defined as "given x is A' and the fuzzy relation R of the fuzzy rule then infer y is B' ". This inference strategy is a data-driven or forward chaining strategy, which is

particularly useful in FLC. On the other hand the GMT strategy is defined as "given y is B' and the fuzzy relation R of the fuzzy rule then infer x is A' ". This inference strategy is a goal-driven or backward chaining strategy, which is commonly used in expert fault diagnosis systems.

2.1.3.2. FLS Inference Mechanisms

Consider the rule base of Subsection 2.1.2.2. Given x is A' and y is B', based on the GMP inference strategy, the role of the inference engine is to infer an output z = C'. In general, the compositional rule of inference [Zadeh, 1973] is employed to deduce the resultant output fuzzy set C' in the universe W as follows:

$$C' = (A', B') \text{ o } R = \{(w, \mu_{C'}(w)) \mid w \in W\} \tag{2.11}$$

And its membership function is given by:

$$\mu_{C'}(w) = (\mu_{A'}(u), \mu_{B'}(w)) o \mu_R(u, v, w) \tag{2.12}$$

where R is the fuzzy relation defined in Equation (2.3), and $\mu_R(u, v, w)$ is the membership function of the fuzzy relation defined in Equation (2.4) using the union interpretation of the connective "*also*" and o denotes a compositional operator.

In general, a compositional operator may be expressed as the sup-star composition where star represents an operator e.g. min, product, etc. In applications of FLS, the sup-

min and sup-product operators are the most frequently adopted compositional operators [Lee, 1990a].

However, the FLS inference in Equation (2.11) can be expressed in different forms using different implication functions, different interpretations of the connectives "*and*" and "*also*" and different compositional operators. For example, in the case of FLC, if the fuzzified inputs are fuzzy singletons, namely $A' = u_o$ and $B' = v_o$ and if the union interpretation of the connective "*also*" is employed, four commonly used inference mechanisms can be expressed. The first inference mechanism is achieved using the min interpretation of the connective "*and*" and the min implication function and can be expressed as follows:

$$C_1 = \bigcup_{i=1}^{n} \mu A_i(u_o) \wedge \mu B_i(v_o) \wedge \mu C_i(w) = \bigcup_{i=1}^{n} \alpha_i^{\wedge} \wedge \mu C_i(w) \qquad (2.13)$$

The second inference mechanism is achieved using the min interpretation of the connective "*and*" and the product implication function and can be expressed as follows:

$$C_2 = \bigcup_{i=1}^{n} \mu A_i(u_o) \wedge \mu B_i(v_o) \cdot \mu C_i(w) = \bigcup_{i=1}^{n} \alpha_i^{\wedge} \cdot \mu C_i(w) \qquad (2.14)$$

where $\alpha_i^{\wedge} = \mu A_i(u_o) \wedge \mu B_i(v_o)$ is the i^{th} rule firing strength when using the min interpretation of the connective "*and*".

The third inference mechanism is achieved using the algebraic product interpretation of the connective "*and*" and the min implication function and can be expressed as follows:

$$C_3 = \bigcup_{i=1}^{n} \mu A_i(u_o) \cdot \mu B_i(v_o) \wedge \mu C_i(w) = \bigcup_{i=1}^{n} \alpha_i^{\bullet} \wedge \mu C_i(w) \qquad (2.15)$$

The fourth inference mechanism is achieved using the algebraic product interpretation of the connective "*and*" and the product implication function and can be expressed as follows:

$$C_4 = \bigcup_{i=1}^{n} \mu A_i(u_o) \cdot \mu B_i(v_o) \cdot \mu C_i(w) = \bigcup_{i=1}^{n} \alpha_i^{\bullet} \cdot \mu C_i(w) \qquad (2.16)$$

where $\alpha_i^{\bullet} = \mu A_i(u_o) \cdot \mu B_i(v_o)$ is the i^{th} rule firing strength when using the algebraic product interpretation of the connective "*and*".

2.1.4. Defuzzification Strategies

Most practical control applications require crisp control actions to drive the controlled process. Moreover, the output of most modelling and prediction systems has to be crisp. Defuzzification is the mapping from the linguistic fuzzy output defined over an output universe into a crisp output space [Grzegorzewski, 2001]. There are three commonly used defuzzification strategies. The first strategy is the maximum criterion. The max criterion produces the point w_o in the output universe W that has the maximum degree

of membership in the output fuzzy set $\max_{w \in W} \mu(w) = \mu(w_o)$. A problem arises with this method when more than one element of W possesses this maximal value and thus w_o is not uniquely determined. The second strategy is the Mean of Maxima (MOM). If there is more than one element in the output universe W possess the maximal membership value, then the MOM method produces the average value of the maxima. MOM method does not consider rules fired below the maximum level [Saade, 1996]. The third and most commonly used strategy is the Centre of Area (COA) strategy. The COA method attempts to correct the drawback of MOM by considering rules that may be fired below the maximum level. COA generates the centre of gravity w_o of the possibility distribution of a control action as follows:

$$w_o = \frac{\sum_{j=1}^{n} \mu(w_j).w_j}{\sum_{j=1}^{n} \mu(w_j)} \qquad (2.17)$$

where n is the number of quantisation levels of a universe W, w_j is the point in the j^{th} quantisation level in a universe W at which $\mu(w)$ achieves its maximum value, and $\mu(w_j)$ is the inference membership degree of the output membership function $\mu(w)$. [Runkler, 1997] discusses in more details the advantages and disadvantages of MOM and COA in terms of their static and dynamic properties.

2.1.5. Models of FLS

In general, several models of FLS have been reported in the literature. These models can be distinguished because they employ different types of consequents and antecedents in the FLS rule base, different type of membership functions and different

type of inference mechanisms. The two most frequently used models are Mamdani's model [Lee, 1990b], and the TS-model [Takagi and Sugeno, 1985]. Mamdani's model employs linguistic terms in both the antecedents and the consequents of its rule base and adopt the min operation as the fuzzy implication function. The inference mechanism of this model employs the sup-star compositional rule of inference to infer the resultant output fuzzy set. The TS-model employs linguistic terms to represent the antecedents of the rules and uses a linear function of its input linguistic variables to represent the consequents. In this model, the rule base contains rules of the following form:

R_i : IF x is A_i and y is B_i THEN $z = f_i(x, y)$

where x and y are linguistic variables, A_i and B_i are linguistic terms of the linguistic variables x and y in the universes of discourse U and V respectively, with $i = 1,2,\ldots,n$ and $f_i(x, y)$ is a linear function of the linguistic variables x and y. Given the crisp inputs x_o, y_o, the crisp output z_o of this model can be written as follows:

$$z_o = \left(\sum_{i=1}^{n} \alpha_i f_i(x_o, y_o)\right) / \left(\sum_{i=1}^{n} \alpha_i\right) \qquad (2.18)$$

where $\alpha_i = \mu A_i(x_o) \cdot \mu B_i(y_o)$ is the i^{th} rule firing strength when the algebraic product interpretation of the connective "*and*" is used, or $\alpha_i = \mu A_i(x_o) \wedge \mu B_i(y_o)$ is the i^{th} rule firing strength when the min interpretation of the connective "*and*" is used.

2.2. Fuzzy Neural Networks (FNN)

NN and FLS are both numerical model-free estimators for dynamic systems. They share a common ability to deal with difficulties arising from uncertainty, imprecision, and noise in the natural environment. Both systems and their techniques have been successfully applied to various areas [Efe and Kaynak, 1999]. A promising approach to obtain the benefits of both NN and FLS is to combine them into an integrated system. The low-level learning and computation power of NN can enhance FLS, and the high-level human-like thinking and reasoning of FLS can improve NN. Several approaches have been presented for combining NN and FLS into so-called Fuzzy Neural Networks (FNN). These approaches can be categorised into three commonly used classes according to the neural network structure. These are the Feedforward Fuzzy Neural Network (FFNN), Recurrent Fuzzy Neural Network (RFNN), and the Self-Organising Fuzzy Neural Network (SOFNN). Each of these classes can be categorised into a number of sub-classes according to the employed fuzzy model, learning algorithm, learning technique, and type of processed data. In the following, background information is provided on the different types of fuzzy neural networks [Choi et. al., 1992; Lee et. al., 1993].

2.2.1. Feedforward Fuzzy Neural Networks (FFNN)

Several types of FFNN are described in the current literature, according to the fuzzy model and learning techniques employed. The two common fuzzy models that can be integrated within a FNN structure to form a FFNN are the Mamdani-model [Lee, 1990b] and the TS-model [Takagi and Sugeno, 1985]. There are two learning

techniques that can be employed for both of these models, namely supervised learning and reinforcement learning. In the following, some examples of FFNN that employ the two fuzzy models and these two learning techniques are described.

2.2.1.1. Mamdani-Model Based FFNN

Several examples of Mamdani-model based FFNN have been presented. For example, in [Lin and Lee, 1991 and 1992], a FFNN structure consisting of five layers was employed to represent a FLC as shown in figure (2.3). The inputs and outputs of this FFNN are numerical crisp inputs and outputs. The first layer is an input layer that simply transmits the input crisp values of the fuzzy variables to the second layer, which is the input term nodes layer. The output function of each node in the second layer is the membership degree corresponding to one linguistic term in the term set which describes each fuzzy variable. For example, a Gaussian function can be written as follows:

$$f = M_{x_i}^{j}(m_{ij}, \sigma_{ij}) = -\frac{(u_i^2 - m_{ij})^2}{\sigma_{ij}^2} \quad \text{and} \quad a = e^f \tag{2.19}$$

where f is the net input to the node, a is the node activation function, $M_{x_i}^{j}$ is the membership function of the j^{th} term in the term set describing the i^{th} fuzzy variable, m_{ij}, and σ_{ij} are respectively the corresponding centre (or mean) and the width (or variance) of the Gaussian function, and u_i^2 is the input to the second layer node corresponding to the i^{th} fuzzy variable (the superscript "2" corresponds to the layer number).

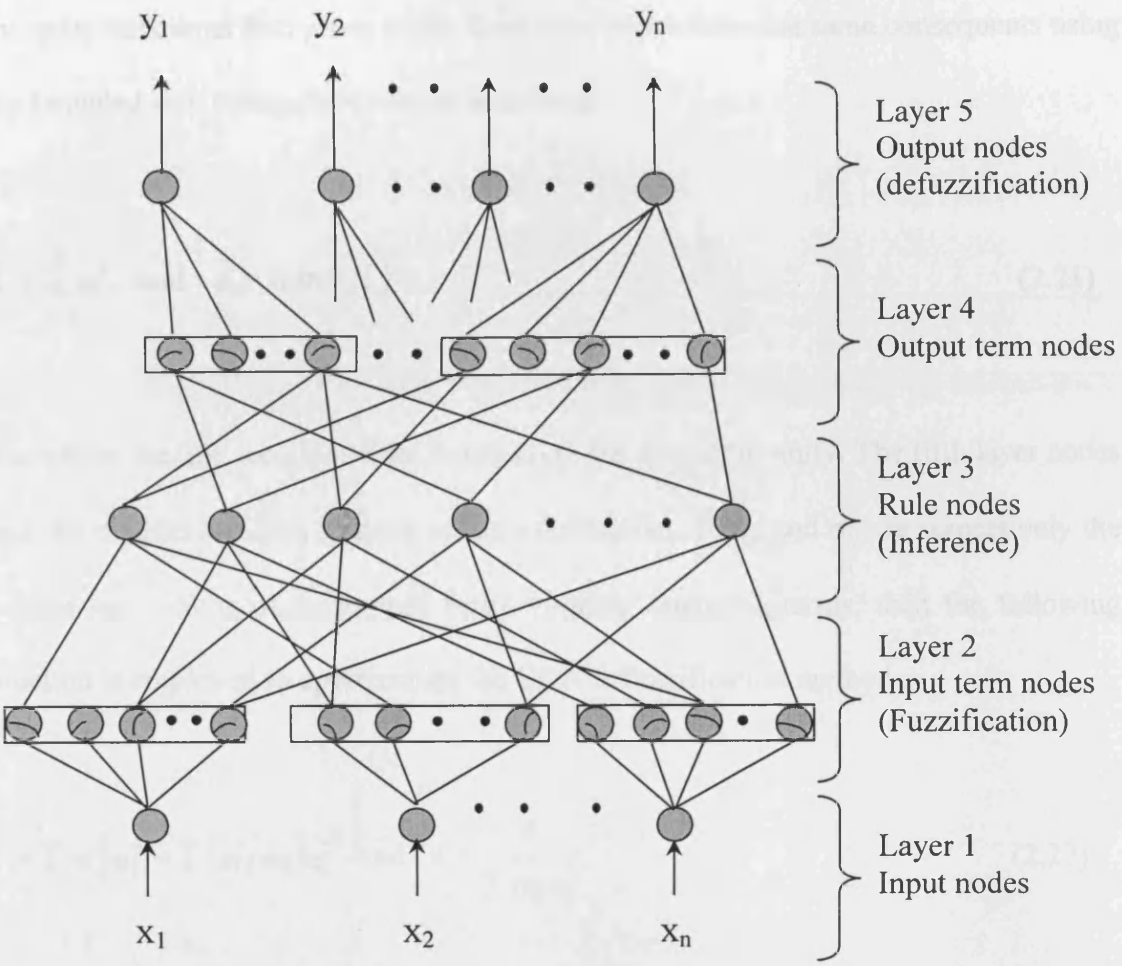


Figure (2.3). Structure of Mamdani-model based FFNN.

Hence, the link weight in layer two $\left(w_{ij}^{2}\right)$ can be interpreted as m_{ij}, which can be adaptively tuned through learning. The nodes in the third layer are rule nodes, each of which represents the antecedent of one rule in the rule base. Therefore, the nodes of the third layer are employed to perform precondition matching of fuzzy rules. Hence, the rule nodes perform the fuzzy min function as follows:

$$f = \min\left(u_1^3, u_2^3, \ldots\ldots, u_p^3\right) \text{ and } a = f \qquad (2.20)$$

Therefore, the link weights of this layer are set to unity. The nodes of the fourth layer integrate the output fuzzy sets of the fired rules which have the same consequents using the bounded sum triangular co-norm as follows:

$$f = \sum_{i=1}^{p} u_i^4 \quad \text{and} \quad a = \min(1, f) \tag{2.21}$$

Therefore, the link weights of the fourth layer are also set to unity. The fifth layer nodes and the weights attached to them act as a defuzzifier. If m_{ij} and σ_{ij} are respectively the centres and widths of the output fuzzy variable linguistic terms, then the following function is employed to approximate the COA defuzzification method:

$$f = \sum w_{ij}^5 u_i^5 = \sum (m_{ij}\sigma_{ij}) u_i^5 \quad \text{and} \quad a = \frac{f}{\sum \sigma_{ij} u_i^5} \tag{2.22}$$

where the i^{th} link weight in layer five $\left(w_{ij}^5\right)$ is given by $m_{ij}\sigma_{ij}$, which can be adaptively tuned through learning. Based on the above network structure, a hybrid learning algorithm is developed. The first learning phase is self-organised learning to obtain the network initial structure. The second learning phase is supervised learning to change the network adjustable weights using the Back-propagation (BP) learning algorithm. The structure and the fuzzy model that were employed in this network were as described in [Lin and Lu, 1995]. The advantage of using a Mamdani-model based FFNN is that the rule base is in the form of linguistic rules with linguistic antecedents as well as linguistic consequents, so that it is understandable by human users.

Moreover, the rule base can be constructed with numerical data and/or linguistic information from human experts. However, a disadvantage of this model is that it does not allow easy mathematical analysis due to the logical nature of its inference functions, e.g. min/max functions. Moreover, all the examples just reviewed employ BP as a learning algorithm, however the differentiation results of the algorithm are not accurate due to the non-differentiable min/max functions. Furthermore, [Estevez and Nakano, 1995] introduced an alternative for the logic-max and logic-min functions in the form of a differentiable function that could approximate either of these two functions with proper selection of parameters. Later, [Shankir, 2001] introduced both the *softmin* and *softmax* functions for use in the Mamdani-model based feedforward fuzzy neural networks as a direct fuzzy logic complement to each other.

2.2.1.2. TS-Model Based FFNN

Similarly, several TS-model based FFNN have been reported. For example, in [Jang, 1992 and 1993], a FFNN termed Adaptive Network-Based Fuzzy Inference System (ANFIS) was designed to represent a FLC as shown in figure (2.4). A TS fuzzy model was employed in ANFIS that can be written as follows:

IF x_1 is A_i and x_2 is B_i THEN y is $p_i x_1 + q_i x_2 + r_i$

where A_i and B_i are the fuzzy subsets describing the fuzzy variables x_1 and x_2 respectively, [p_i ,q_i and r_i] are the parameter set of the consequent linear equation. ANFIS is also a five-layer FFNN. In layer one, every node has a node representing the linguistic input term nodes.

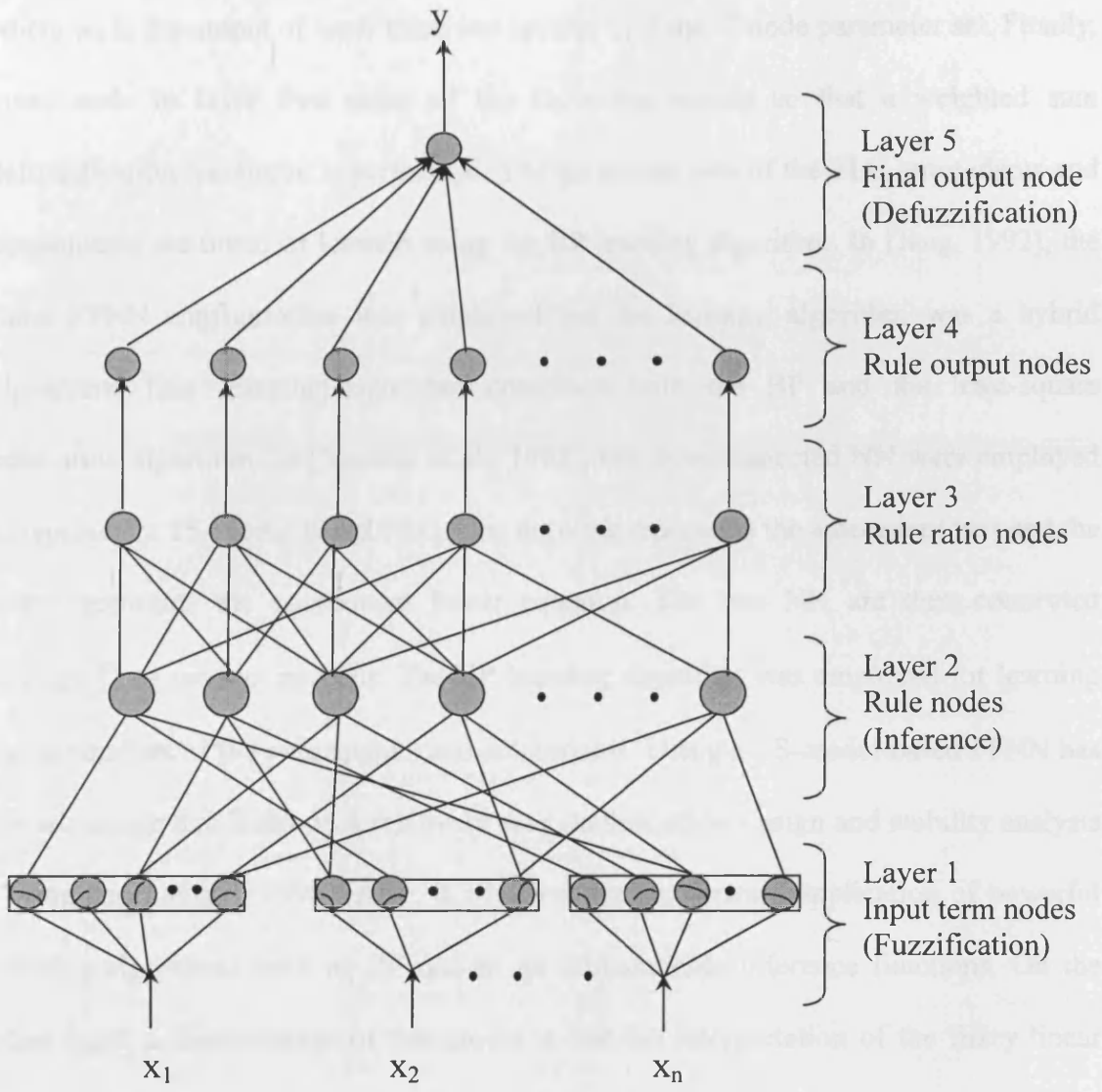


Figure (2.4). Structure of TS-model based FFNN.

In layer two, the number of nodes is equal to the total number of rules. The output of each node represents the firing strength of a rule. Normally, product triangular norm is used in this layer. In layer three, the i^{th} node calculates the ratio of the firing strength of the i^{th} rule to the sum of the firing strength of all fired rules. In layer four the output of each rule is computed, where the i^{th} node has a node function:

$$O_i^4 = w_n\left(p_i x_i + r_i\right) \tag{2.23}$$

where w_n is the output of layer three and [p_i and r_i] is the i^{th} node parameter set. Finally, every node in layer five sums all the incoming signals so that a weighted sum defuzzification technique is performed. The parameter sets of the FLC antecedents and consequents are tuned or learned using the BP learning algorithm. In [Jang, 1992], the same FFNN configuration was employed but the learning algorithm was a hybrid algorithm. This learning algorithm combined both the BP and the least-square estimation algorithm. In [Yaochu et al., 1995], two interconnected NN were employed to represent a TS-model based FLC. One network represents the antecedent part and the other represents the consequent linear equation. The two NN are then connected through $\prod$ or product neurons. The BP learning algorithm was employed for learning the parameters of the consequents and antecedents. Using a TS-model based FFNN has the advantage that it allows a relatively easy mathematical design and stability analysis [Wang and Langari, 1996]. Also, it allows a straight forward application of powerful learning algorithms such as BP due to its differentiable inference functions. On the other hand, a disadvantage of this model is that the interpretation of the fuzzy linear rules is difficult compared to that for linguistic rules. Also, the rule base of this model can only be constructed using only numerical input/output data and it is not possible to incorporate linguistic information from human experts to construct such a model.

2.2.2. Recurrent Fuzzy Neural Networks (RFNN)

RFNN can be considered a feedforward fuzzy neural network with some feedback connections from some neurons output to other neurons input in the same layer or in previous layers. In some cases, this feedback is made between the same neuron output

and inputs. There are many RFNN described in the current literature with different feedback connections structure, almost all of them belong to Mamdani-model fuzzy neural network while a few reported to be of TS-model type [Ballini et. al., 2001; Jeen-Shing and Lee, 2003].

2.2.3. Self-Organising Fuzzy Neural Networks (SOFNN)

Despite the successful applications of FFNN and RFNN in modelling and control, they suffer from a main problem with regard to the connectionist structure. The structure of the FFNN cannot be dynamically changed, it is fixed and reflects the designer's experience. If the structure needs to be changed due to poor performance, the designer has to repartition the input/output universes, regenerate the rule base and retrain the FFNN. Self-Organising Fuzzy Neural Networks (SOFNN) are characterized by being able to modify their structure dynamically. They use the competitive learning technique as their learning algorithm. Therefore, they have the ability to accommodate new data without destroying old information [Baraldi and Blonda, 1999a and 1999b].

2.2.4. Learning in FFNN

As already mentioned, there are two common learning strategies that can be employed for learning in FFNN namely supervised learning and reinforcement learning.

2.2.4.1. Supervised Learning

In supervised learning, a teacher provides the desired control objectives and necessary control actions to the learning system at each time step. The desired control objectives

are specified in the form of a desired output or a desired trajectory. The goal of the learning algorithm is to minimize the error between the output of a FFNN and a desired output as follows:

$$E = \frac{1}{2}(y(t) - y_{net}(t))^2 \tag{2.24}$$

where $y(t)$ is the desired output, and $y_{net}(t)$ is the current network output. If BP is employed as the learning algorithm, a backward pass is used to compute the rate of change of the error function with respect to the weights for all the hidden layers. Assuming that w is the adjustable weight, the general learning rule used is:

$$\Delta w = -\frac{\partial E}{\partial w} \tag{2.25}$$

$$w(t+1) = w(t) + \eta \Delta w \tag{2.26}$$

where η is the learning rate. Examples of FFNN that employ supervised learning can be found in [Lin and Lee, 1992; Lin and Lu, 1996].

2.2.4.2. Reinforcement Learning

In reinforcement learning, the feedback is not direct, immediate and informative as in supervised learning. If precise and immediate input/output training data are available, then supervised learning can be more efficient than reinforcement learning. However for some real-world applications, precise data for training can be difficult and

expensive if not impossible to obtain. In reinforcement learning, only an evaluative feedback signal (reinforcement signal) is available. The reinforcement signal, r(t), can take one of four different forms [Lin and Lu, 1995]. It can be a two-valued number, r(t) $\in$ [-1,0], such that r(t) = 0 means reward and r(t) = -1 means penalty. It can be a multi-valued discrete number in the range [-1,0], for example, r(t) $\in$ [-1,-0.75,-0.5,-0.25,0] which corresponds to different degrees of reward or penalty. Also it can be a real number, r(t) $\in$ [-1,0], which represents more detailed and continuous degrees of reward or penalty. Moreover, the reinforcement signal can be given in the form of fuzzy feedback information such as (good, very good, bad, very bad, etc.). Reinforcement-learning-based FFNN systems can be categorised in terms of type of the reinforcement signal into two main categories. In the first category, the reinforcement signal is numerical [Lee, 1991; Berenji and Khedkar, 1992]. In the second category, the reinforcement signal is a fuzzy reinforcement signal [Lin and Lu, 1995]. Reinforcement learning is sometimes called learning with a critic as opposed to learning with a teacher as in supervised learning.

2.3. Applications of FLS and FNN in Modelling and Control

When designing a FLC, certain controller parameters must normally be tuned by trial and error [Lai et. al., 1996; Li et. al., 1995; and Liaw and Wang, 1991]. Such parameters include scaling factors, and the width and centre of the membership functions [Costa Branco and Dente, 1998]. Moreover some rules have to be modified, deleted or added. It would be useful for control engineers to be able to automate the learning or tuning of the parameters and/or structure of FLC. FFNN is one of the tools

that can be applied for this purpose. FFNN can be interpreted as a FLC that can automatically tune its parameters using the learning capability of neurons [Pham and Oh, 1993]. In general, FFNN for control applications can be categorised as supervised-learning based FFNN controllers and reinforcement-learning based FFNN controllers. Several examples of supervised-learning-based FFNN controllers have been reported in literature [Delgado and Gonzalez, 1993; Lee et al., 1996]. Different fuzzy models were employed within the FFNN structure to perform the control function. In these examples, BP generally was employed as a learning algorithm to perform the parameter learning. For example, a TS-model based FFNN was employed for ship collision avoidance in [Hiraga et al., 1995], for the control of carbon monoxide concentration in [Tanaka et al., 1995], and for temperature control in [Lai and Lin, 1999; Lin and Chung, 1999]. A Mamdani-model based FFNN was employed for backing a truck to a loading dock in [Lin and Lin, 1997], and for welding process control in [Chen et al. 1997]. Moreover, a Mamdani-model based FFNN that employs compensatory neurons was employed to control a cart-pole balancing system in [Zhang and Kandel, 1998].

Fuzzy controllers have been suggested for motion control planning of mobile robots [Watanabe et al., 1996] and for intelligent control of complex robotic systems. For actuator-level applications, most research has focused on kinematics control. Calculation of the inverse kinematics of manipulators is computationally expensive, and consumes a large percentage of time in the real-time control of manipulators. Lack of the solutions for singularity configurations and existence of multiple solutions for redundant cases add further complexity to the problem. The idea of using human intuition and experience, by means of a fuzzy logic approach, to avoid complex

computation for inverse kinematics mapping has been investigated by several researchers.

2.4. Applications of FLS and FNN in Robotic Systems Modelling

The establishment of an input/output model for a process is a very important problem in systems engineering. Many deterministic and stochastic methods have been proposed to derive acceptable mathematical models for both continuous-time and discrete-time processes. However, in the modelling of complicated and/or ill-defined processes, precise mathematical models may fail to give satisfactory results. Also, the nonlinear behaviour of many practical systems and the uncertainty in these systems make analytical modelling and control of these systems by conventional methods very difficult. This is why FFNN have featured in several applications for systems modelling. For example, the Adaptive Network Fuzzy Inference System (ANFIS) [Jang, 1993] was employed for nonlinear function approximation.

In the robotics community, there is currently a growing interest in the use of intelligent neuro-fuzzy technology [Er et. al., 1997]. Almost all NN and FNN applications in robot control involve identifying the robot dynamics and/or inverse dynamics and incorporating this knowledge into the robot controller [Narendra and Parthasarathy, 1990; Pham and Oh, 1994; Pham and Yildirim, 1999, and Yildirim, 1998]. Sometimes, it is also required to incorporate the robot kinematics and/or inverse kinematics to achieve certain control objectives. Hence, for a controller design, a preliminary stage of robot modelling is essential. The basic idea is to employ a NN or FNN to learn

repeatedly the characteristics of the robot and then use this knowledge to generate control inputs. Forward kinematics of a robot manipulator is the relation between the end-effector Cartesian position and a given set of manipulator joint angles. In most cases, forward kinematics is straightforward and usually performed mathematically in the feedback path of the control system in the case of a Cartesian control system. Generally, for the Cartesian control of the robot, the inverse kinematics should be calculated exactly. Solving the inverse kinematics of a robot manipulator means finding the joint angles corresponding to a given end-effector Cartesian position. However, inverse kinematics calculation is complex and too much time consuming for a manipulator control system to calculate in real-time [Craig, 1996]. The calculation of the inverse kinematics can result in significant control delay. Fuzzy logic mapping and FNN have been used in literature to calculate approximate inverse kinematics for robot manipulators [Sang-Bae, 1997; Martinez et. al., 1996].

In [Nedungadi and Wenzel, 1991], a fuzzy associative memory bank (FAM) is used to relate the change in the Cartesian end-effector position to the change in the robot joint angles. In [Kim et. al., 1993], a fuzzy logic system is developed to solve the differential relationship between the joint space and Cartesian space of redundant manipulators. In this method, the inverse kinematics solution is performed with the help of the Jacobean matrix and a fuzzy system to calculate the differential change in the joint angles required to achieve a desired Cartesian displacement [Xu and Nechyba, 1993]. A similar idea was used in [Martinez et. al., 1996] for the configuration of a three-links revolving robot manipulator. In this method, by relating the polar position of the end-effector to its Cartesian position, a space variation for the joint angles can be drawn and

a fuzzy system relationship can be established relating the polar position to the joint angles. The degree of accuracy in these methods is strongly influenced by the selected membership function for the inputs and outputs, which have to be selected manually. Some of these methods require pre-calculation of the robot manipulator Jacobean matrix, resulting in additional computational burden. These methods also do not include any additional learning stage and are not generic or systematic. Robot manipulators have a dynamical model similar to that of the motion of rigid bodies of the form:

$$T = M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta) + F(\theta,\dot{\theta}) + \tau_d \qquad (2.27)$$

where $M(\theta)$ is the $n \times n$ inertia matrix of the manipulator, $V(\theta,\dot{\theta})$ is an $n \times 1$ vector of centrifugal and Coriolis terms, $G(\theta)$ is an $n \times 1$ vector of gravity terms, $F(\theta,\dot{\theta})$ is an $n \times 1$ vector of friction, and τ_d is an $n \times 1$ vector of unknown terms that represents joint load torques arising from un-modelled dynamics and external disturbances [Craig, 1996]. The parameters of this model are mostly obtained from CAD solid modelling or measured by the disembodied robot which results in inaccurate values. Forward and inverse dynamics of robotic manipulators are very complex and can almost never be accurately calculated due to parameter variation, nonlinearity, and backlash.

Neural networks have been used to identify or extract inverse dynamics for robot manipulators through learning. Some researchers used feedforward NN to learn the robot inverse model [Miyamoto et. al., 1988], others uses the dynamic structure of the recurrent neural network [Pham and Yildirim, 1999; Pham and Oh, 1994]. Other

researchers use self-organising neural networks to build the robot model, through online tuning of the network structure and network parameters, to reach the closest behaviour possible to the real system [Kishan and Jamshidi, 1997].

Regarding the relation between the neural network used and the investigated robot during learning, there are two main approaches to the identification of the inverse dynamics of the robot. The first approach can be called direct inverse learning, where a neural network is fed with the outputs from the robot and directly taught to generate the robot inputs that produced those outputs as shown in figure (2.5a). Errors between the desired and actual outputs of the network are used to adjust the network weights. The second approach can be called indirect inverse learning, where the learning is achieved by training the neural network to act as a controller to the robot as shown in figure (2.5b). Errors between the forward model outputs and actual outputs of the robot are back propagated through the robot forward model to adjust the weights of the inverse model network [Pham and Oh, 1999; Pham and Yildirim, 1999].

Most of the approaches used for robot forward and inverse dynamics model input variables selection are heuristic, and based on the method of using the model in the controller. These inputs may include present and past values of the robot joint positions, speeds, accelerations, position errors, and actuating torques according to the required approximation order of the model and the size of the network used. Generally, the higher the approximation order, the larger the size of the network, and the slower the learning and execution time. Also, different techniques for training and adaptation are used, with back-propagation algorithm being the most commonly used method

[Shin, 1994; Pham and Yildirim, 1999; Pham and Oh, 1999; and Pham and Sagiroglu, 2001].

Although most of these methods succeed in obtaining an approximate inverse dynamics model for the robot manipulators, they are poor in terms of the transparency to the dynamics behaviour of the robot and form a black box relationship between its inputs and outputs. Also, the learning in these techniques is performed online with the robot operating in control and with the network connected to it. The structure of the network is to be selected according to the experience of the user without direct relation to the robot behaviour, except for the self-organising neural network which is rarely used for robot inverse dynamics modelling.

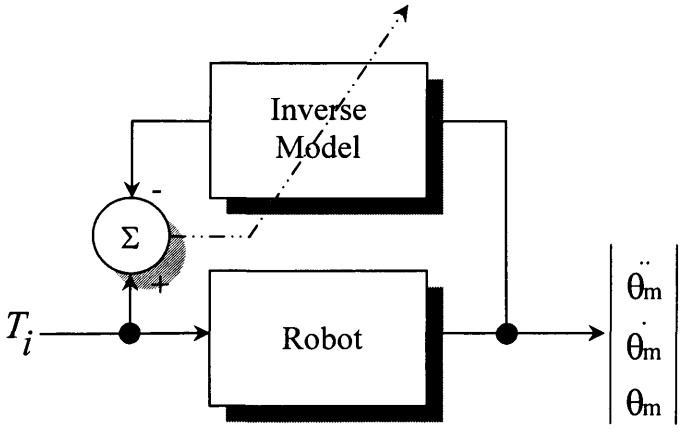


Figure (2.5a). Direct inverse learning [Pham and Oh, 1999].

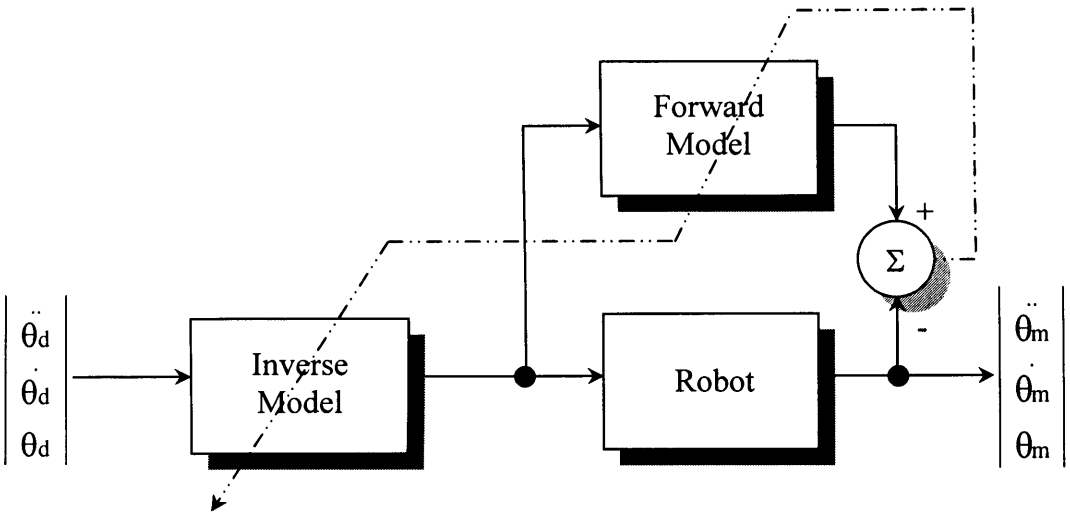


Figure (2.5b). Indirect inverse learning [Pham and Yildirim, 1999].

2.5. Applications of FLS and FNN in Robotic Systems Control

2.5.1. Conventional Control of Robotic Manipulators.

As explained in Appendix (A), nonlinearly, interactive dynamics, and other uncertainties in robotic systems prevent linear servo controllers from providing a satisfactory performance especially in transient and high-speed modes of operation. Although, conventional independent-PID joint controllers are used in many industrial robots, they limit the capability of the robot to pick and place operations and to slow motion applications. Many modifications have been added to the independent-PID joint controllers to include nonlinearity and coupling between joint dynamics. In some industrial robots, a gravity element has been added to the independent-PID joint controllers to compensate for the joint weights [Pan and Woo, 2000] as shown in figure (2.6).

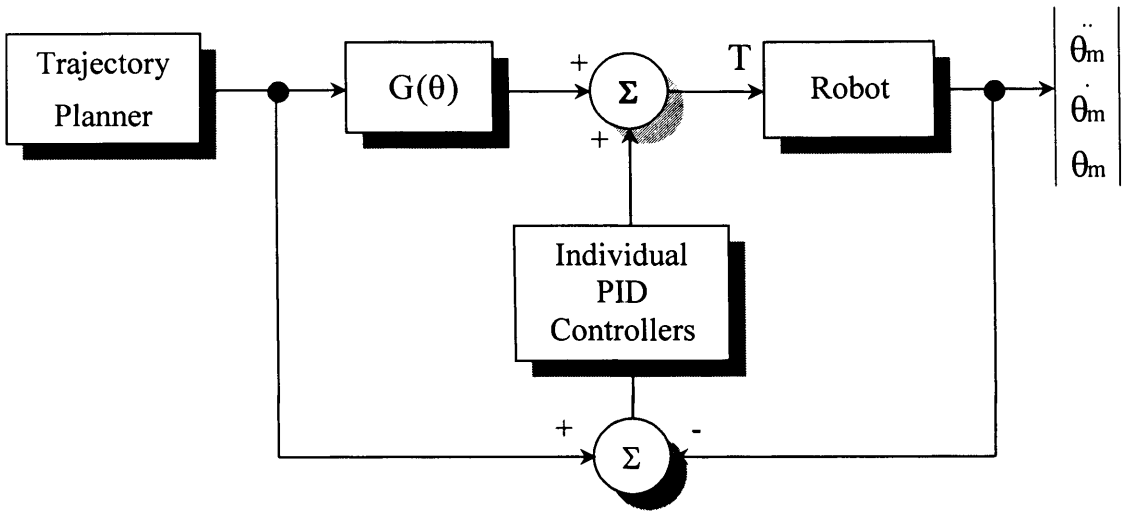


Figure (2.6). PID controllers with gravity compensator [Pan and Woo, 2000].

The problem of controlling a complicated nonlinear coupled system such as a robot manipulator can be handled by the partitioning of the controller into two parts, a *model-base part* and a *servo part*. The model-based part is affected by the manipulator model parameters and includes information about the system nonlinearity and dynamics coupling effects, whilst the servo part is independent of these parameters. Model-based control algorithms have been used as nonlinear feedback controllers to robotic manipulators under the name of the computed-torque control method [Craig, 1996]. This method was then modified to the control system shown in figure (2.7).

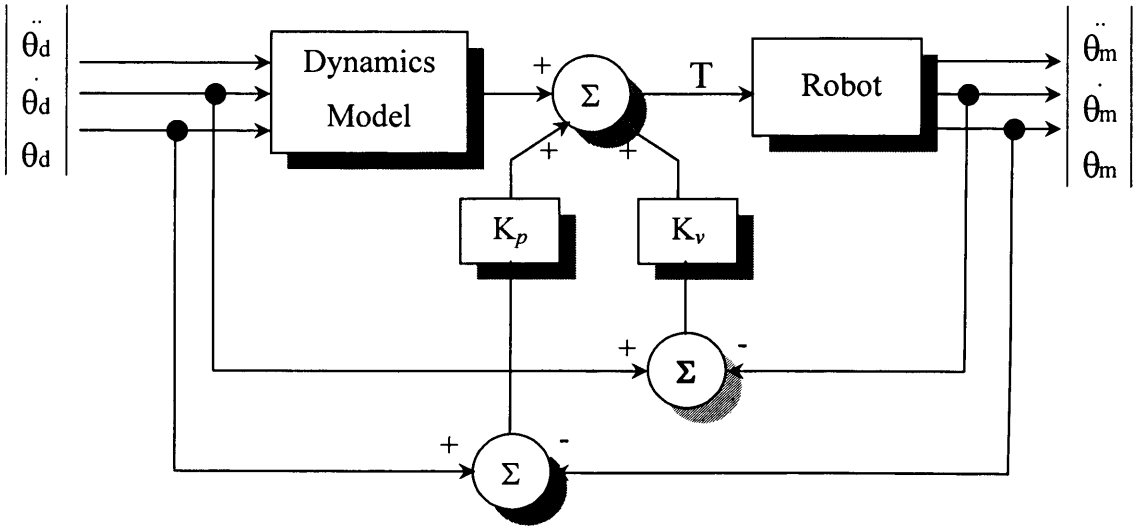


Figure (2.7). Modified computed torque control method [Craig, 1996].

In the control systems shown above, it was assumed that the desired trajectory was available in terms of joint position, velocity, and acceleration, so that what is called *joint-based control scheme* was built. Generally, the final target of the control system is to achieve certain trajectory for the robot end-effector, so that a trajectory conversion stage has to be performed first. The trajectory conversion process is quite difficult to undertake analytically. Inverse kinematics, inverse Jacobean, and inverse Jacobean differentiation have to be calculated which requires high computational resources [Hu et. al., 1996]. Usually, trajectory conversion is performed with the help of the inverse kinematics only for the end-effector position and successive numerical differentiation is used to obtain the joint speeds and accelerations from the resulting joint positions as shown in figure (2.8).

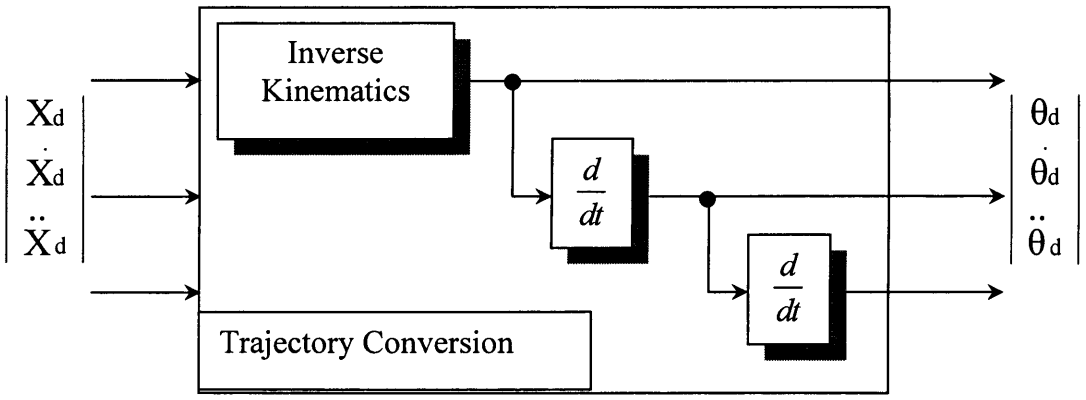


Figure (2.8). Trajectory conversion using inverse kinematics.

An alternative approach is shown in figure (2.9). Here, the sensed position of the manipulator is immediately transformed by means of forward kinematics equations into a Cartesian position of the end-effector. This Cartesian position is then compared to the desired one in order to form the error in Cartesian space. Control schemes that are

based on forming errors in Cartesian space are called *Cartesian-based control schemes* [Craig, 1996].

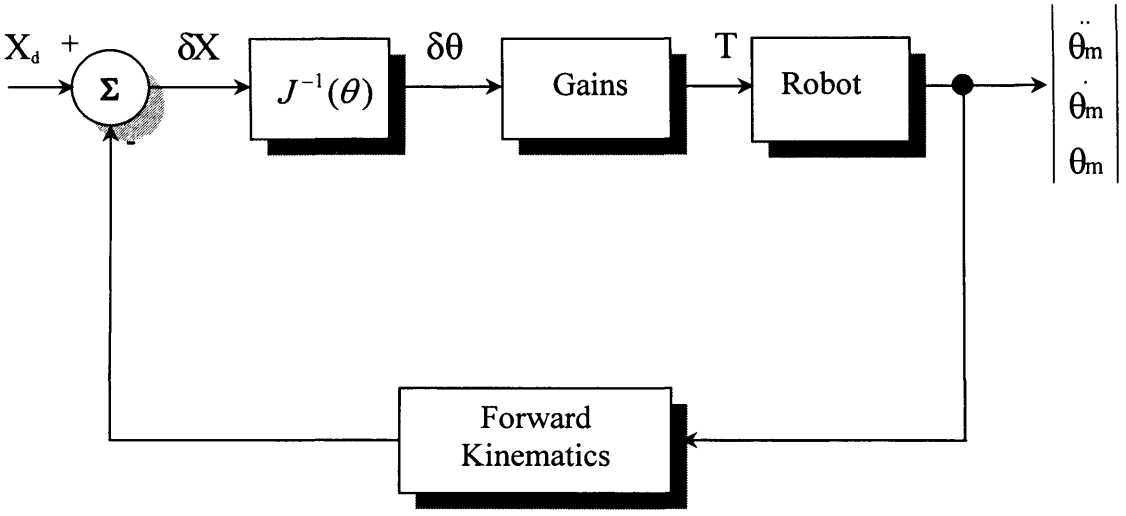


Figure (2.9). Inverse Jacobean Cartesian control scheme [Craig, 1996].

In figure (2.9), an inverse Jacobean has to be calculated to map the error in the Cartesian space to error in joint space. Finally, this latter is multiplied by a gain to compute the torques required to reduce the error.

Another scheme is shown in figure (2.10). Here, the Cartesian error vector is multiplied by a gain to compute a Cartesian force vector. This can be thought of a Cartesian force which, if applied to the end effector of the robot, would push the end-effector in a direction tending to reduce the Cartesian error. This Cartesian force vector is then mapped through the Jacobean transpose in order to compute the equivalent joint torques which would tend to reduce the Cartesian error [Craig, 1996].

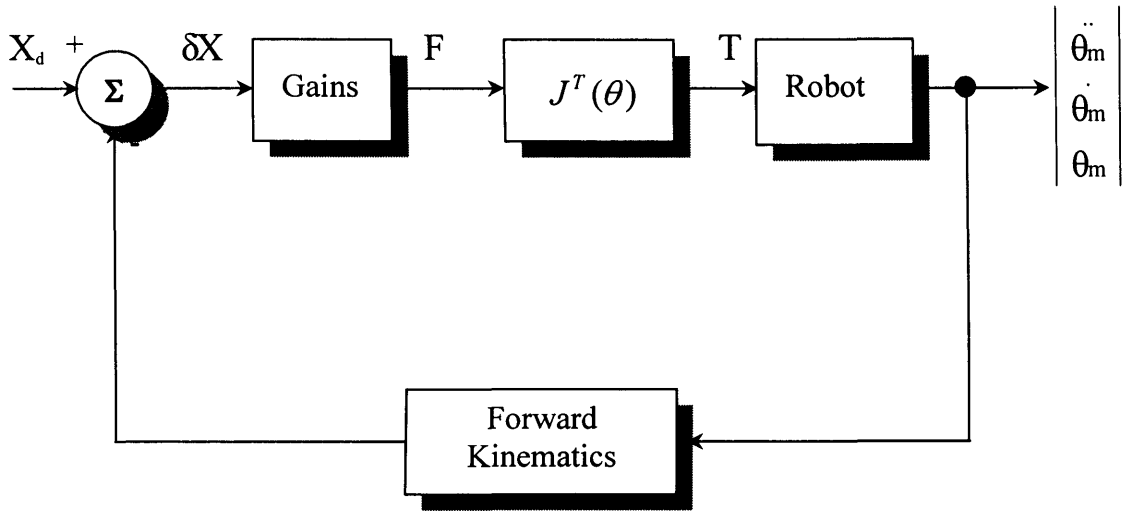


Figure (2.10). Transpose Jacobean Cartesian control scheme [Craig, 1996].

Robot control is more difficult when the robot has contact with external forces. When any contact is made between the end-effector and the manipulator's environment, position control may not suffice. In this case, control methods should generate high compliant or, in other words, low stiffness (where compliance is the inverse of stiffness) motion to balance external forces. Compliance is the tendency of a body to distort due to applied forces. When a robot manipulator is moving through free space, the natural constraints are all zero. If the end-effector is glued to a wall, the robot manipulator is subjected to position constraints. Position control schemes are designed to deal with the first situation, while the second situation does not occur in practice. Usually control systems consider force control in the context of partially constrained tasks in which some degrees of freedom are subjected to position control, while others are subjected to force control. Thus, in this case a Cartesian hybrid position/force control scheme is introduced as shown in figure (2.11).

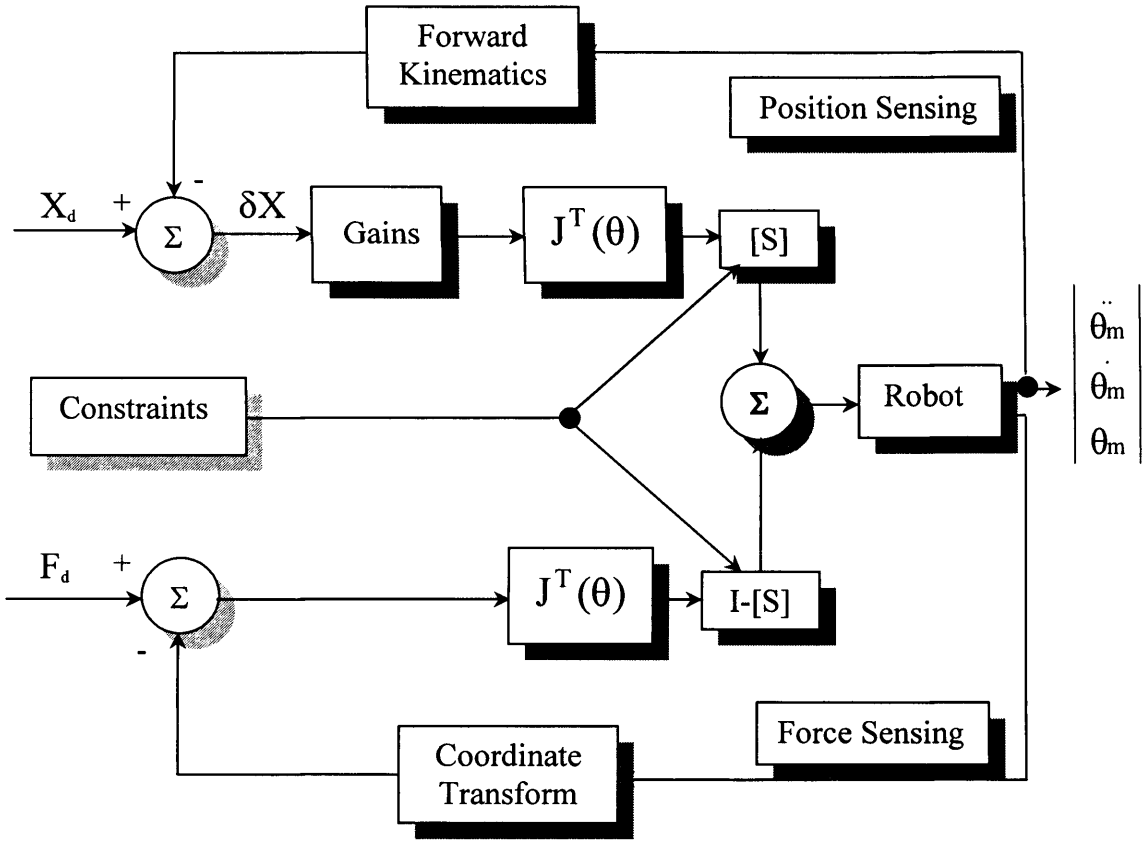


Figure (2.11). Cartesian hybrid position/force control scheme [Craig, 1996].

The hybrid position/force controller controls the manipulator by performing in three ways. First, position control is utilised along directions in which a natural force constraint exists. Second, force control is utilised along directions in which a natural position constraint exists. Third, these modes are mixed along the degrees of freedom of the robot manipulator [Wedel and Saridis, 1988].

In the control system shown in figure (2.11), both position controller and force controller are presented. The matrix S is used to select the control mode (position or force) of each joint. The S matrix is diagonal with ones and zeros. Hence, it is simply a switch which sets the control mode of each joint of the robot arm. In accordance with

the setting of S, there are always a number of components of the trajectory being controlled equal to the degrees of freedom of the robot, where the relative mix between position and force control is arbitrary according to the control mode. Hence when a certain degree of freedom is under force control, position errors on that degree of freedom are ignored [Craig, 1996].

2.5.2. Fuzzy Control of Robotic Manipulators.

Fuzzy control has been used extensively for independent direct feedback control of robot manipulators [Erbature et. al., 1995; Moudgal et. al., 1994 and 1995; Green and Sasiadek, 2001; Hitam, 2001; and Tang et. al., 2001]. During control, no adaptation of the rule base or membership functions is carried out, only system gains were modified in relation to link speeds and joint errors within predetermined design parameters. Few techniques considered the coupling dynamics between the manipulator links.

2.5.3. Adaptive Control of Robotic Manipulators.

A more recent research strategy for robot manipulator control is to incorporate a control scheme that is able to adapt (*adaptive control*) to uncertainties in the robot dynamics parameters. This allows the controller parameters to be modifies in real time until they converge to exact values. This has lead to the application of neural networks and FNN in robot manipulators control systems. The structure of the neural network or the FNN varies according to the order and input variables used in the approximation model [Tsai et. al., 1996]. Most of these techniques use different learning algorithms to tune the inverse dynamics model of the robot manipulator contained in the neural network using

error minimisation and online modification of the network link connection weights. Generally, a feedback controller is used along with the feedforward controller to improve the disturbance rejection capabilities of the control system. The feedback controller can be of any combination of a conventional P, I, D, or a fuzzy controller. The feedforward controller is normally a neural or a neuro-fuzzy network. As learning proceeds, the error signal will reduce and the role of the feedforward controller increases while that of the feedback controller decreases [Pham and Yildirim, 1999].

According to the learning signal used to train the inverse model, three main learning schemes are listed. The first is *the direct inverse learning scheme*, where the inverse model is connected in parallel with the robot. The error between the robot input and the model output is used to tune the model parameters. These parameters are then copied to the forward path controller as shown in figure (2.12) [Pham and Yildirim, 1999].

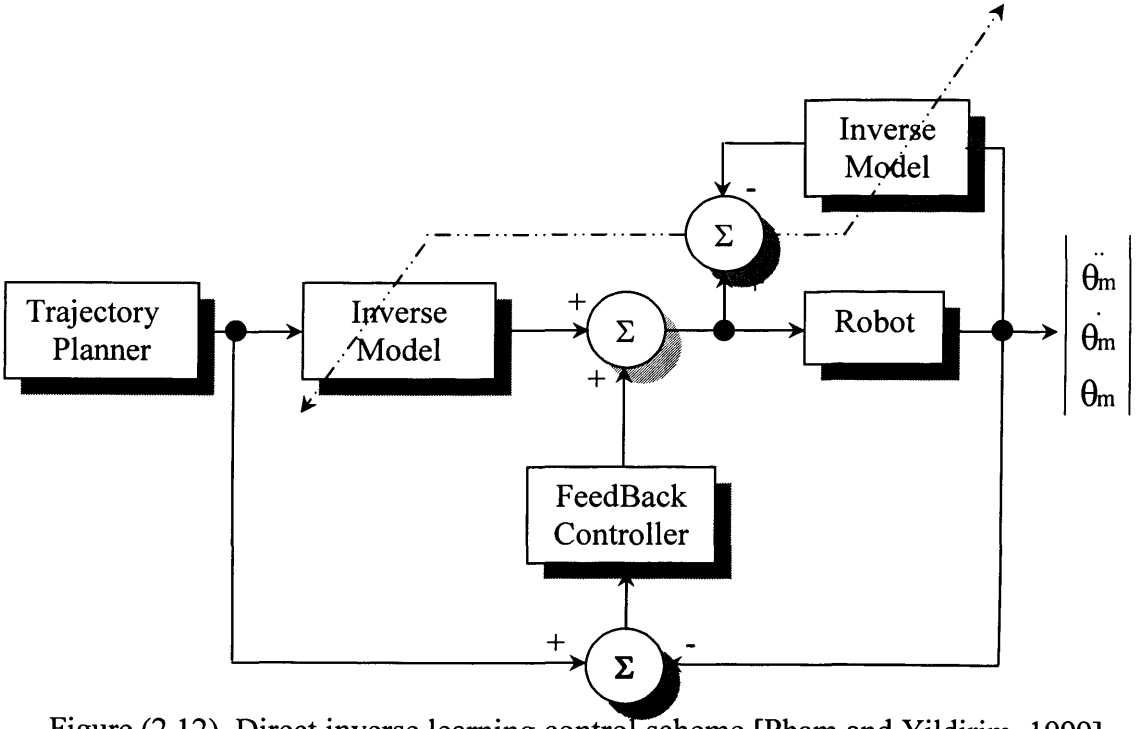


Figure (2.12). Direct inverse learning control scheme [Pham and Yildirim, 1999].

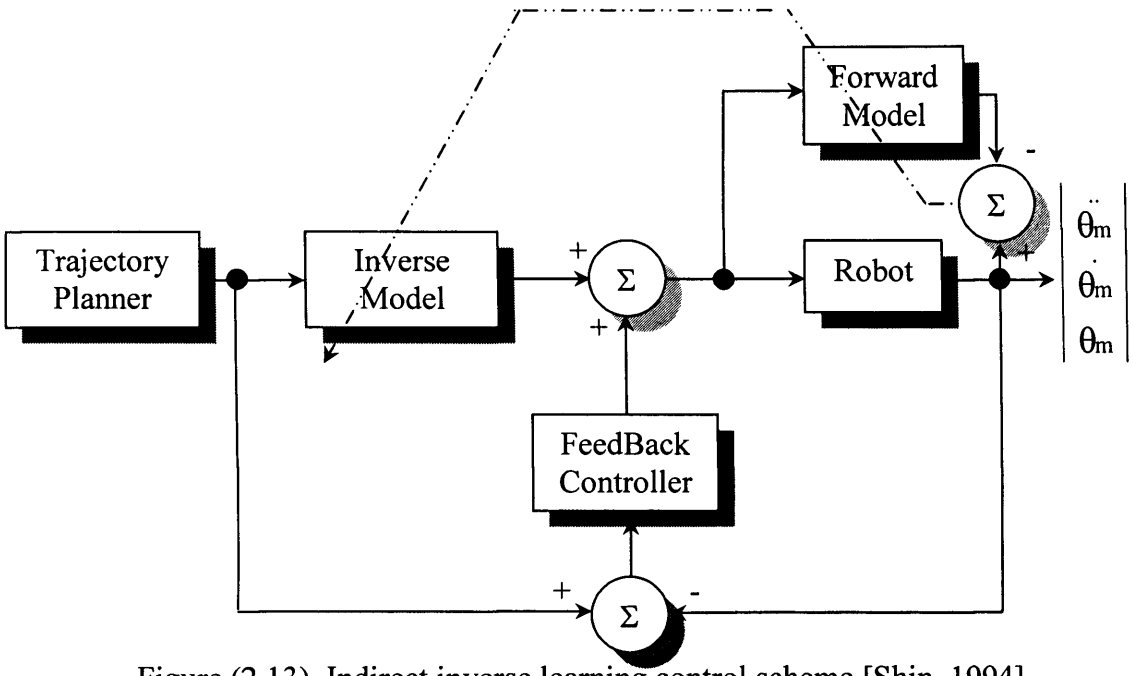


Figure (2.13). Indirect inverse learning control scheme [Shin, 1994].

The second scheme is *the indirect inverse learning scheme*, where the forward model is connected in parallel with the robot as shown in figure (2.13). The error between the robot and the forward model outputs is used to tune the parameters of the inverse model with information obtained for this purpose from the forward model [Shin, 1994]. The third scheme is *the feedback-error learning scheme*, where the feedback torque signal (from the *servo controller* portion) is used to tune the parameters of the inverse model [Kawafuki et. al., 1997] as shown in figure (2.14).

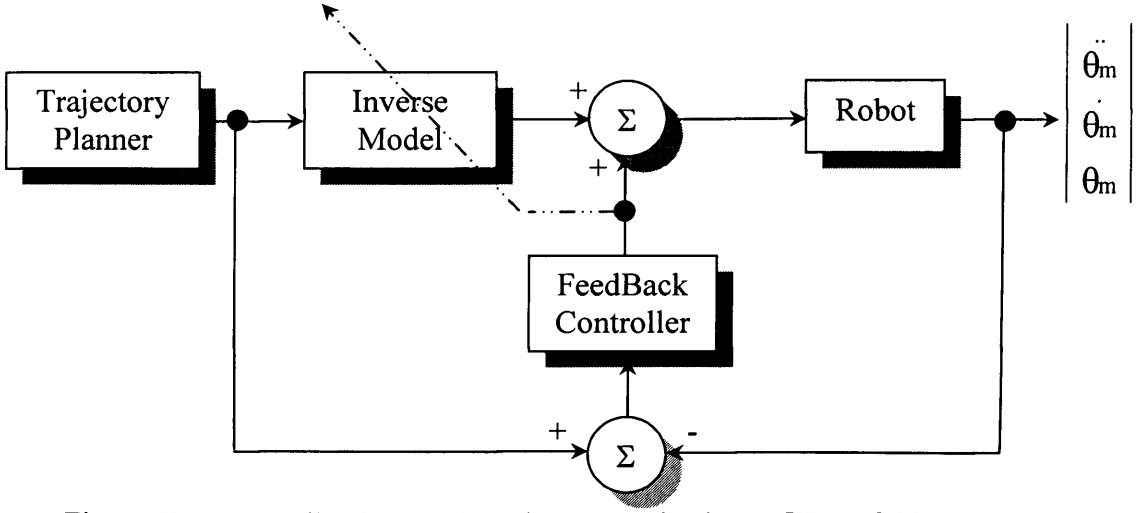


Figure (2.14). Feedback-error learning control scheme [Kawafuki et. al., 1997].

Although the total torque acting on the robot is simply the sum of the feedback torque and the feedforward torque, these two play entirely different roles in the robot control. The feedback is used for clumsy but robust control at an early stage of learning. The feedforward torque is necessary for smooth control and fast movement of the robot [Miyamoto et. al., 1988] [Emami et. al., 1996, 1998, 1999, and 2000].

2.5.4. Internal Model Control of Robotic Manipulators.

The application of internal model control (IMC) for robot manipulator has received much attention in the last decade (figure (2.15)). The IMC provides a direct method for the design of the nonlinear feedback controller, if a good model of the robot is available, the closed-loop system will achieve exact set point following despite unmeasured disturbances acting on the robot [Yildirim and Sukkar, 1996]. In [Li et. al., 1995] a back-propagation neural network is incorporated into a fixed standard structure internal model controller to achieve robot manipulator control.

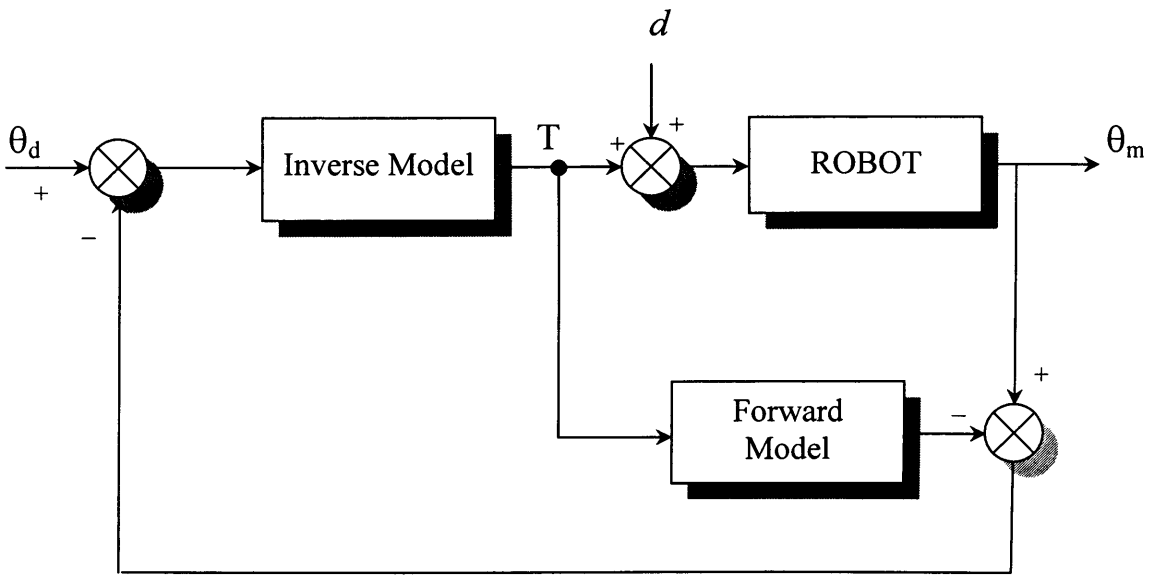


Figure (2.15). Internal model control structure [Yildirim and Sukkar, 1996].

2.6. Applications of FLS and FNN in Robotic Systems Coordination

Currently, industrial applications that utilise multiple manipulators involve the use of two or more robots, which although working simultaneously on the same task, are not manipulating the same object at the same time. Hence, the main area of concern is collision avoidance. The ability to manipulate the same object at the same time by two robots would enable the system to undertake the difficult two-handed manipulation tasks that humans are capable of performing [Akella and Hutchinson, 2002].

However, the formation of a cooperating robot system (CRS) causes control complications since an over constrained closed kinematic chain is generated. This means that a new control technique may be required to enable the CRS to perform handed manipulative tasks.

The techniques used for controlling a CRS can be broadly separated into two main categories:

- Position Control.
- Hybrid Position/Force Control.

In the position control scheme, the difference between the desired position and actual position of the robot is used to generate an appropriate control signal designed to minimise this error. For most industrial applications that use of single robot systems, a position-based scheme is satisfactory.

One approach for controlling a CRS uses a master/slave configuration, where the motion of the master robot is pre-planned according to the desired motion of the manipulated object and the motion of the slave robot is to follow the master. For further enhancement for the position control method, feedforward signals based on the object and robot dynamics can be incorporated into the controller to minimise trajectory errors. Sometimes a constant offset is specified to provide suitable following characteristics for the slave robot. That is, when the master is ready to begin motion along a pre-planned trajectory, it sends its destination to the slave robot, which performs a transformation of coordinates from the master's reference frame to its own reference frame. However, this transformation is dependent on the geometry of the manipulated object and has to be specified before motion begins [Tinos et. al., 2002].

In the position based control scheme outlined above, each robot in the CRS is controlled by minimising the position error of the end-effector along a common path. However, accurate knowledge of the robots kinematics properties is essential and also the scheme does not consider geometric errors. Geometric errors in the robot kinematics properties of a CRS can cause the target end-effector paths of each robot to be inconsistent. Since the robots are joined to form a closed chain, any inconsistency in the end-effector paths can result in forces being applied to the manipulated object. To solve this problem, flexible joints are introduces in the CRS system in [Osumi and Arai, 1994] between the robots and the manipulated object. In [Osumi et. al., 1997], free joint mechanisms are introduced to solve this problem.

In the hybrid position/force (HPFC) control scheme for a single robot, the force information is combined with position information to satisfy a set of position and force constraints as explained before. Methods for obtaining force information are: motor current measurements, motor output torque measurements, and wrist mounted force sensors. The HPFC method can be exploited for use in CRS [Sun and Liu, 2001] and, like all robot control strategies, can be implemented on either a centralised controller architecture (one controller for all robots) or a de-centralised controller architecture (independent controllers for each robot).

The HPFC can also form part of a master/slave CRS where the master robot is position controlled while the slave one is force controlled. For all HPFC schemes listed in literature, pre-planned force trajectory and appropriate force measurement at the end-effectors of the robots are required. This not only results in the need for force sensors of a suitable resolution, but also additional hardware and software to interpret and transform the data into usable format. Furthermore, to incorporate the force data, the computational complexity of the control scheme is increased. One further disadvantage of these control systems is that they are based on exact knowledge of the dynamics properties. However, in general, exact robot and load dynamics are difficult to drive due to the complex mechanical construction of the robot.

Adaptive control schemes explained before can be further extended for use in CRS systems, which then incorporate feedforward signals based on the robot and/or object dynamics. With the adaptive control scheme, the controller not only has to calculate the robot and object dynamics but also modify them to take into account parameter

inaccuracies. Few neuro and neuro-fuzzy adaptive control techniques have been reported for the coordination of robot manipulators in the literature [Gueaieb et. al., 2001]. A recurrent neural network is used to build a hybrid position/force controller for two SCARA type robots in [Yildirim, 2001]. In this case, both of the position and force controllers are built using the proposed recurrent neural network. A position reference model and force reference model are used to train the two neural controllers. Another method for separating interconnection variables in order to achieve a fully decomposed fuzzy model is introduced in [Rajasekharan and Kambhampati, 2001] for cooperative manipulators handling a common object. In [Jang, 2001], a neuro-controller is introduced to control a nonlinear two-robot MIMO system. The proposed neuro-controller consists of two linear controllers and a neural-network controller (NNC) to compensate for the nonlinearities and interactions between the two robots. The NNC is trained through a neural network identifier with an indirect learning scheme.

Even though most adaptive schemes provide suitable control for a CRS, many result in a control structure that is not suitable for application to conventional robot systems. Furthermore, in most cases, the suggested scheme is computationally expensive and requires complex mathematical techniques to ensure convergence of the system parameters to their exact values.

2.7. Summary

This chapter has outlined the basic concepts of FLS and FNN to provide background information concerning their structure and design parameters. A classification of FLS

according to their structure and design parameters has been presented. Similarly, a classification of FNN according to their structure, fuzzy model, and learning technique has been given. Finally, some applications of these techniques to the modelling, control, and coordination of robotic systems have been outlined.

CHAPTER 3

Neuro-Fuzzy Inverse Modelling of Robotic Manipulators

As a consequence of the rapid development in FLS and NN techniques in the 1980s, great progress in FNN design and implementation techniques was made. Since the early 1990s, FNN have attracted a great deal of interest because such systems are more efficient and more powerful than either NN or FLS alone. Different types of FNN have been presented in the literature. As mentioned in Chapter (2), these types can be identified based on the structure of the FNN, the fuzzy model employed and the learning algorithm adopted. On the one hand, according to the FNN structure and learning algorithm, the most commonly used and successful approach is the feedforward and recurrent structure model using the BP learning algorithm. On the other hand, according to the fuzzy model adopted, there are two types of fuzzy models that can be integrated with a neural network to form a FNN. These two models are the TS-model [Takagi and Sugeno, 1985] and the Mamdani-model [Lee, 1990a and 1990b] using either the sup-min or sup-product compositional operator. However, based on the review of the models of FNN in Chapter (2), Mamdani-model based FNN represent more transparent neuro-fuzzy systems compared with TS-model-based FFNN. The reason is that the rule base of the Mamdani-model is more understandable to human users. Also, it is more general in terms of how its rule base is created, because it can be constructed using human experience and/or numerical data. Also, it may be noted that feedforward neural networks are normally used for mapping of arbitrary static

functions while mapping of dynamic functions is normally performed using recurrent neural networks which is normally a feedforward NN with some feedback signals.

This chapter deals with the problem of both inverse kinematics and inverse dynamics modelling of robotic manipulators as a pre-control stage. The main target is to benefit from inductive learning techniques to develop rule sets for both inverse kinematics and inverse dynamics from data collected from the robot during random trajectories following. These rule sets will be then arranged in a Mamdani-type neuro-fuzzy set of networks for further tuning of the obtained fuzzy models.

"Pro/Engineer®" and *"Pro/Mechanica®"* visual dynamics simulation packages were used to simulate the robot manipulator. These packages allow simulation of dynamic systems by transferring the assembly CAD model from *"Pro/Engineer®"* to the associated virtual dynamics simulation program *"Pro/Mechanica®"*, which in turn allows the user to specify masses, loads, drive forces, torques, friction, etc., and many other dynamics parameters of the modeled assembly. This package generates the equations of motion, inertia parameters, orientation matrices, etc., for each body in the model and for the whole assembly from the geometry of the modeled system and the parameters specified by the user. Due to the nature of the research, no access to these equations of motion is required although *"Pro/Mechanica®"* allows for this if necessary. The most powerful part of this package is that it allows the user to interface to the modeled system with a user specified custom C++ subroutine. This subroutine allows the user to get information about the modeled system and to design a controller for the assembly and/or add any other feature to the modelling process, which may not

be present in the package programming tools. The Puma 560® industrial robot manipulator is used for the simulation process as the parameters of this robot are well published and can be checked against the parameters obtained from the virtual model.

The remainder of this chapter is organized as follows. Section 3.1 presents a review of current techniques used for inverse identification for robot manipulators. Section 3.2 describes the method used to virtually model the Puma 560® industrial robot manipulator under the *"Pro/Mechanica®"* environment. Section 3.3 explains the technique used to collect numerical data from the modeled robot and discusses the fuzzy rule generation method using the inductive learning technique *DynaFuzz* [Bigot, 2003]. Also, in the same section, the rules obtained for both inverse kinematics and inverse dynamics for the Puma 560® industrial robot are listed. Section 3.4 presents the structure of the proposed Mamdani-type neuro-fuzzy network used to formulate the fuzzy rules for online tuning of the fuzzy model. Also, in the same section the network decision-making mechanism using *softmin* and *softmax* differentiable activation functions [Estevez and Nakano, 1995] and the online adaptation mechanism are explained. Section 3.5 discusses the simulated performance of the proposed system when used to model the Puma 560® industrial robot manipulator for both inverse kinematics and inverse dynamics. Section 3.6 gives a summary for the chapter.

3.1. Inverse Model Identification of Robotic Manipulators

The forward kinematics of a robot manipulator produces the Cartesian position of the end-effector according to a given set of manipulator joint angles while the inverse

kinematics of a robot manipulator gives the joint angles according to a given end-effector Cartesian position. As stated in chapter (2), the inverse kinematics calculation is complex and consumes too much time to perform in real-time [Craig, 1996]. Moreover, singularities and multiple solutions exist in the inverse kinematics calculation. The inverse kinematics equations constitute a set of highly coupled nonlinear equations. The common method is to utilise the relationship between the joint speed and the end-effector speed to resolve the inverse kinematics problem. The differential motion relationship between end-effector Cartesian space and joint space is:

$$\dot{r}(t) = J(\theta)\dot{\theta}(t) \tag{3.1}$$

where r(t) is the end-effector displacement in Cartesian co-ordinates, $\theta(t)$ is joint displacement, and $J(\theta)$ is the Jacobean matrix from joint space to Cartesian space. From equation (3.1), the joint velocities can be obtained by calculating the inverse Jacobean and then the joint variables can be evaluated by numerical integration.

Many methods have been reported to circumvent the direct calculation of the inverse Jacobean using fuzzy logic mapping, NN, or the pseudo-inverse of the Jacobean matrix [Kim et. al., 1993; Xu and Nechyba, 1993; and Sang-Bae, 1997]. For example, in [Nedungadi and Wenzel, 1991], a fuzzy associative memory bank (FAM) is designed to relate the change in the Cartesian end-effector position to the change in the robot joint angles. In this method, the contribution of the joint angle variation to achieve the user requested move in the Cartesian coordination of the end-effector can be calculated through the developed FAM bank. Firstly, a total of 98 rules have been created for the

inverse kinematics solution of a single-link planar manipulator. Then, these rules were reduced to 28 rules due to symmetry in the FAM bank. This means that large numbers of rules for multi-link manipulators will be created. The FAM bank was developed based on the kinematics equations of the robot manipulator. In [Ming et. al., 2001], the same technique is used with a reduced number of fuzzy membership functions resulting in 25 rules for a two-link planar manipulator. A three-link three-dimension revolve manipulator was studied in [Martinez et. al., 1996]. In this method, the use of geometric relationships to relate the spherical coordinates of the end-effector to its Cartesian coordinates reduced the 3-dimentional positioning problem to a 2-dimentional problem where θ_1 is found trivially. Contours of constant angles θ_2 with varying θ_3 through its full range were drawn for the remaining two spherical coordinates. Contours of constant angles θ_3 with varying θ_2 through its full range were also drawn. By examining these contours, a total of 198 fuzzy rules were established relating the polar position to the joint angles. These rules were reduced to 82 due to the physical range constraints of the joints. As mentioned before, the degree of accuracy in these methods and the number of generated rules are strongly influenced by the selected membership function for both inputs and outputs, which are selected manually. Most researchers studied planar or simplified manipulators. Furthermore, these systems do not possess a further learning stage and are not generic or systematic.

The forward dynamics of a robot manipulator is the determination of the joint displacement variables (θ_i) according to a given set of manipulator joint torques (T_i), while the inverse dynamics of a robot manipulator is the determination of the joint torques required to cause certain joint displacements. As stated in chapter (2), forward

and inverse dynamics calculations are very complex, nonlinear multi-input multi-output problems, and can almost never be accurately calculated due to parameter variation, un-modelled friction, and backlash. The parameters of these models are mostly obtained from CAD solid modelling or measured by disembodied robot, which results in inaccurate values.

For a dynamic system with single input u and single output y, the system output at time interval k can be expressed in discrete form as:

$$y(k) = f(y(k-1), y(k-2),...., y(k-n), u(k-1), u(k-1),...., u(k-m)) \quad (3.2)$$

This equation can be used to represent any SISO dynamic system in discrete format and can be extended to represent MIMO dynamical systems as well. When input/output data are used, function f, and integer n and m define the dynamic system. If n and m are given, the only task is to find function f. f does not change with time for time-invariant systems. Feedforward neural networks can be employed to approximate f [Narendra and Parthasarathy, 1990]. Robot manipulators are assumed to be bounded-input bounded-output (BIBO) stable in presence of input, which means that equation (3.2) can be used to approximate robot manipulator dynamics.

Neural networks have been used extensively to identify the forward and inverse dynamics for robot manipulators. Most of the approaches used for network input variable selection are heuristic. These inputs may include present and past values of the robot joint positions, speeds, accelerations, position errors, and actuating torques. The number of the network input variables and the assumed order of the approximation

model affect the network size and learning capabilities. Furthermore, neural networks are poor in providing transparency to the dynamics of the robot and form a black box relationship between its inputs and outputs. In most cases, the structure of the network is to be selected according to the experience of the user [Miyamoto et al., 1988; Pham and Oh, 1999].

Dynamic system identification consists of first choosing an appropriate identification model structure and then adjusting the parameters of the model according to some adaptive law such that the response of the model to an input signal that approximates the response of the real system to the same input. In the following sections, an inductive learning technique is first used to automatically identify the inverse kinematics and inverse dynamics model structure from numerical observation data by generating fuzzy-type identification rules. Then, the obtained model structure will be arranged in a full-differentiable version of the Mamdani-type neural network for final tuning of the model parameters. This technique allows simple and direct creation of the neuro-fuzzy model, which will be used later as part of the control system of the robot.

3.2. Virtual Dynamics Model for Puma 560® Manipulator

The Puma 560® is the "guinea pig" of robotics research. It has been studied and used in countless experiments over many years and in many laboratories. The values of dynamics and kinematics parameters depend upon the choice of coordinate frames in which they are expressed. Figure (3.1) represents the commonly used definitions for the Puma 560® robot arm coordinates. Table (3.1) lists the relative values for these

coordinates in addition the actual joint ranges as it is stated in the manufacturer's manual. For the first three joints, because of their long link length for maximum reach and long travel distance between initial position and final position, the effects of transitional motion dominate the rotational motion. In contrast to the first three joints, the rotational effects dominate for the last three joints. This means that the first three joints can be considered responsible for reaching of a point, while the last three joints are responsible for the manipulation at this point.

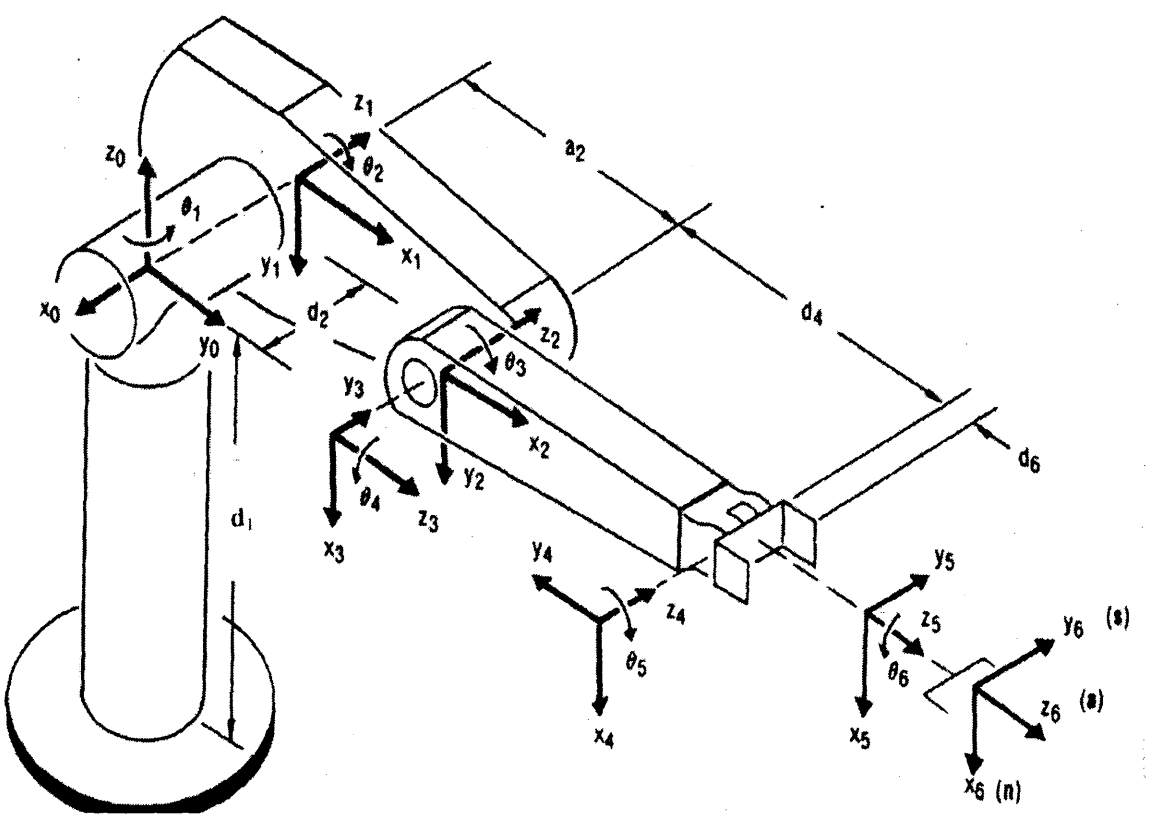


Figure (3.1). Coordinate definition for the Puma 560® robot arm.

Link	Joint Variable θ_i	α_i	d_i (mm)	a_i (mm)	Range
1	θ_1	±90	660.4	0	-160 to +160
2	θ_2	0	149.5	432	-225 to +45
3	θ_3	±90	0	0	-45 to +225
4	θ_4	±90	432	0	-110 to +170
5	θ_5	±90	0	0	-100 to +100
6	θ_6	0	56.5	0	-266 to +266

Table (3.1). Link coordinate system for the Puma 560® robot arm.
± Indicate left and right shoulder configuration, respectively.

With regard to model parameters for the Puma 560® robot arm, [Armstrong, 1988; Armstrong et. al., 1986; and Armstrong and Corke, 1994] represent the most commonly used references for these parameters. For the purpose of creating a virtual model of the Puma 560® robot arm, physical robot dimensions are first used to create a CAD solid model for each link under *"Pro/Engineer®"*. Then, these links are assembled together to form the robot using the pin-joint assembly feature in addition to assigning robot co-ordinates. The assembled robot is then passed to the associated virtual dynamics simulation program *"Pro/Mechanica®"* where the link mass, gravitational coefficient, and coefficients of friction between joints are defined according to the average values listed in [Armstrong and Khatib, 1986; Armstrong and Corke, 1994]. Table (3.2) lists the link mass parameters for each link, while table (3.3) lists the coefficients of friction for the first three links. All of the other parameters for the Puma 560 robot arm are listed in [Armstrong and Khatib, 1986; Corke and Good, 1992; Armstrong and Corke, 1994]. Figure (3.2) shows the obtained virtual dynamics model for Puma 560® robot.

M1	M2	M3	M4	M5	M6
13.00	17.4	4.8	1.18	0.35	0.13

Table (3.2). Link mass values [kg].

Link-1		Link-2		Link-3	
Static	Dynamic	Static	Dynamic	Static	Dynamic
8.43	3.45	7.67	12.77	5.57	3.27

Table (3.3). Coefficients of friction [Nm & Nm/rad.].

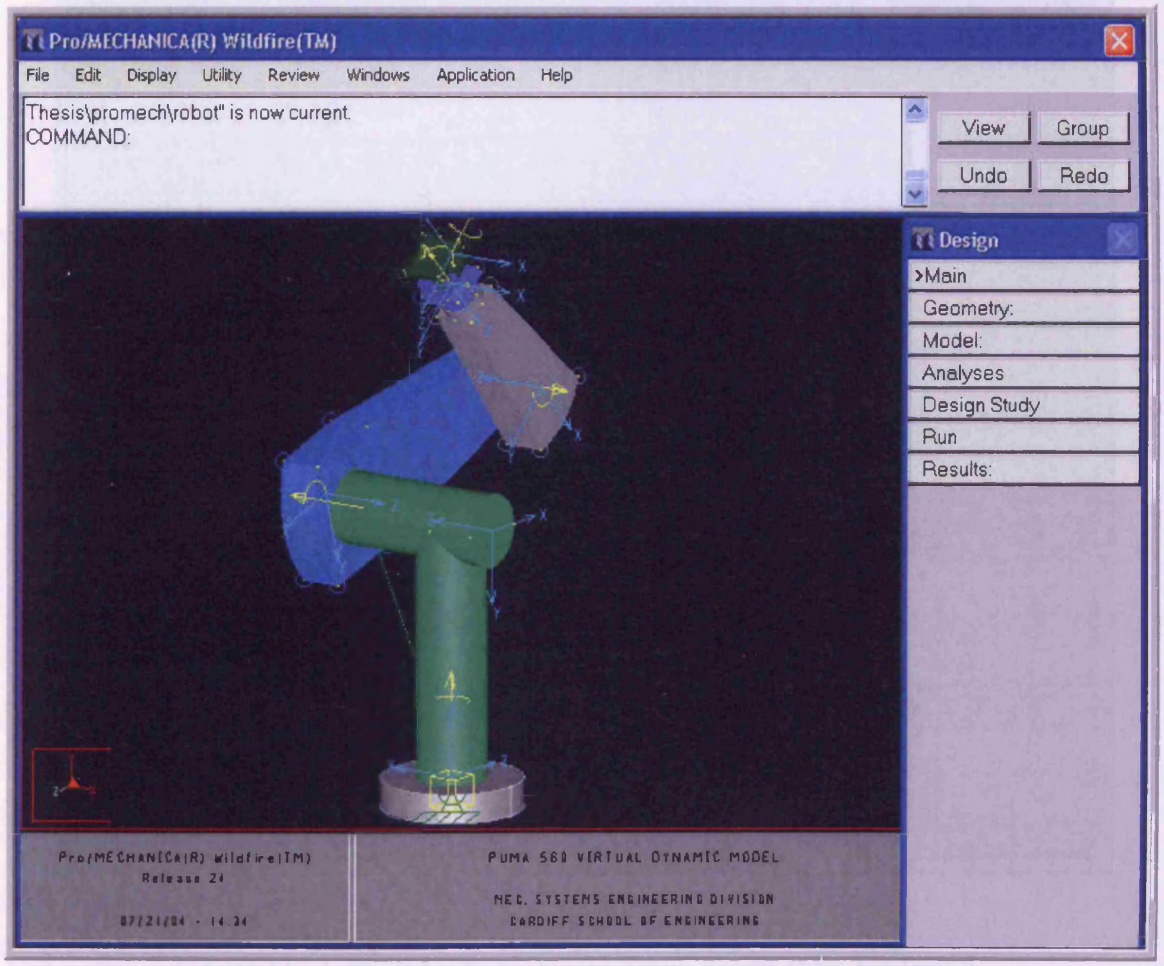


Figure (3.2). Virtual dynamics model for the Puma 560® robot arm.

"Pro/Mechanica®" calculates the remainder of the required parameters for the dynamics simulation. It also generates the equations of motion for each body in the model and for the whole assembly from the geometry of the modelled system and the parameters specified by the user. For example, figure (3.3) lists the inertia matrix parameters calculated by *"Pro/Mechanica®"* for link-1 of the robot around its centre of mass.

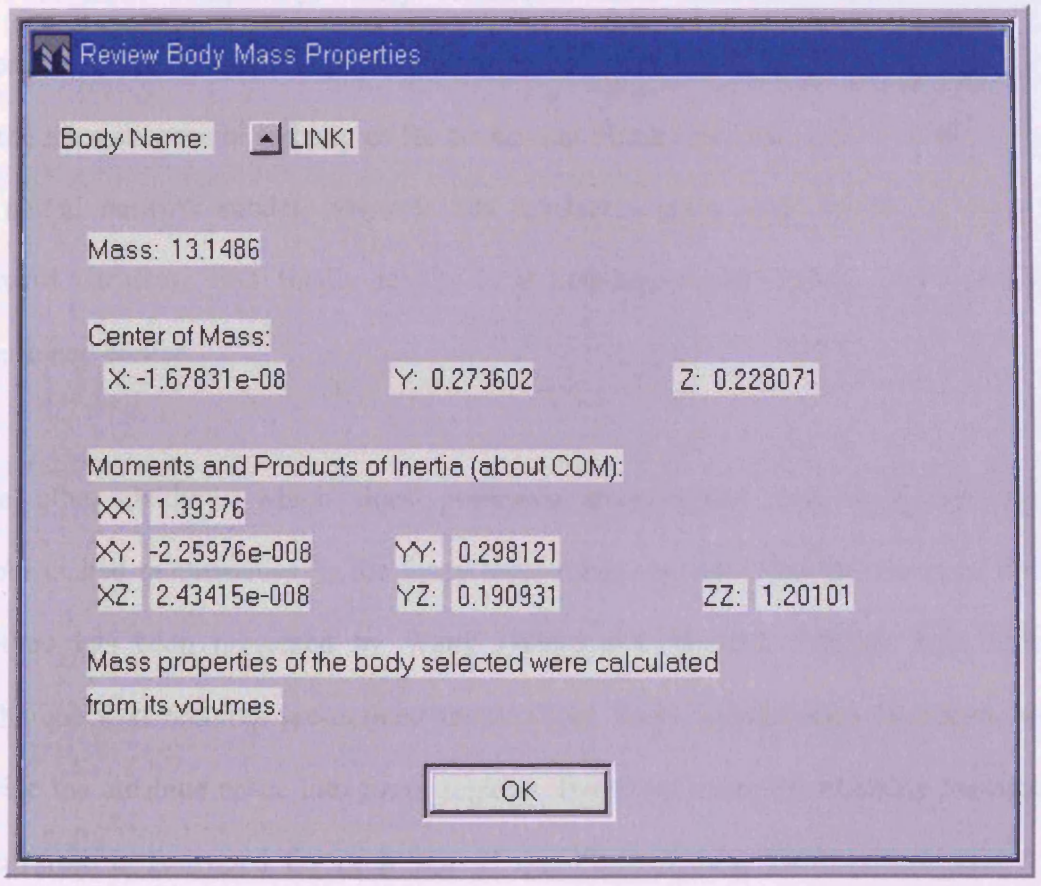


Figure (3.3). Pro/Mechanica calculated parameters for link-1.

3.3. Rule Generation from Observation Data

Many real world applications require the creation of approximate models because it is impossible or difficult to mathematically model the system otherwise. While the optimal solution would be the creation of an exact mathematical model, such model neither always exist nor can be derived for all complex systems, such as robot manipulators.

Due to such complexity, many techniques have been developed to generate approximate models from input/output numerical observation of complex systems. One of the most successful techniques for continuous numerical observations is the creation of neural network model, however this requires a great much effort to select the network structure and finally results in a non-transparent "Black Box" model as mentioned earlier.

The other method, which does possesses transparency and does not require sophisticated mathematics, is the fuzzy logic model system. Perhaps the most famous method has been presented by Wang [Wang and Mendel, 1992a]. This efficient technique first requires pre-defined input/output fuzzy membership functions, which divide the attribute space into fuzzy regions. Based on these membership functions, a fuzzy rule is generated for each pair of input/output data. Each rule is stored in a decision table, and in the case of conflict, a degree for each rule is assessed to select the best rule, and therefore to select the fuzzy membership function to be stored in the decision table. However, as mentioned earlier, there is a "growing memory" problem

when more and more training examples become available, and more and more rules are created, so that the selection of the best rules becomes difficult.

The use of a fuzzy rule for the classification of an example is in many ways similar to the use of rule created via inductive learning [Delgado and Gonzalez, 1993, Pham and Aksoy, 1995]. The main difference is that, due to the notion of fuzziness, a particular example will have a particular "degree of match" (μ rule(example)) with each rule [Srinivasan et. al., 1993]. This could be used for instance to evaluate how likely a rule is to classify an example properly. This "degree of match" is obtained by first assessing the membership degree of each example attribute value with regard to the corresponding fuzzy condition in the rule; (μ condition (Input_Attribute_Value)).

For example; assume the input attributes are Speed=14 km/h and Distance=57 m. For the rule IF Speed is zero AND Distance is long Then Brake is zero, the membership degrees are (μ speed_zero (14)) = 0.4 and (μ distance_long (57)) = 0.8. Using these membership degrees; the "degree of match" of an example to each rule can be assessed. Two main methods are available:

- The first selects the minimum membership degree and if an example contain N attributes, it is defined as:

(μ rule(example)) = min(μ condition_1(Input_Attribute_Value1),.....,

,....., μ condition_N(Input_Attribute_ValueN)).

- The second uses the product of all membership degrees and if an example contains N attributes, it is defined as:

$$(\mu \text{ rule(example)}) = \prod_{\text{for\_each\_attribute}} \mu_{\text{fuzzy\_condition}} \text{ (Input\_Attribute\_Value)}$$

It is important to note that if one membership degree is equal to zero, for both methods (μ rule(example)) will also be equal to zero; this represents the case when one example is not covered by a rule.

In order to obtain a single fuzzy output, one solution is to select the output membership function of the best covering rule (identified using a particular heuristic, for instance the "degree of match"). However a more appropriate method (because it fully takes in consideration the notion of fuzziness) is to combine the output membership function of each covering rule in order to obtain a new output fuzzy set. The obtained fuzzy sets can then be merged, by considering overlapping or fusion between fuzzy sets in order to obtain the output fuzzy set. The transformation of a fuzzy output set obtained through a fuzzy model into a single crisp continuous value is carried out by defuzzification.

3.3.1. Data Generation Technique

The first step in rule generation via observation data is the data collection strategy. For this purpose, random trajectories are applied to suitable gain P-controllers, as shown in figure (3.4). The resulting P-controller torques with random values and frequencies generated from the error signals are applied to the joints of the virtual dynamics model

of the robot arm as explained before. These torques allow the robot arm to move in all directions in the three-dimensional space. The data consists of the applied torque, their corresponding joint angles, joint velocities, and the three Cartesian coordinates (x, y and z) of the end effector main reference point, and are recorded for current and previous sampling intervals. The data collection test was performed for the first three joints only as they are the main joints responsible for the reaching process of the robot arm as mentioned earlier.

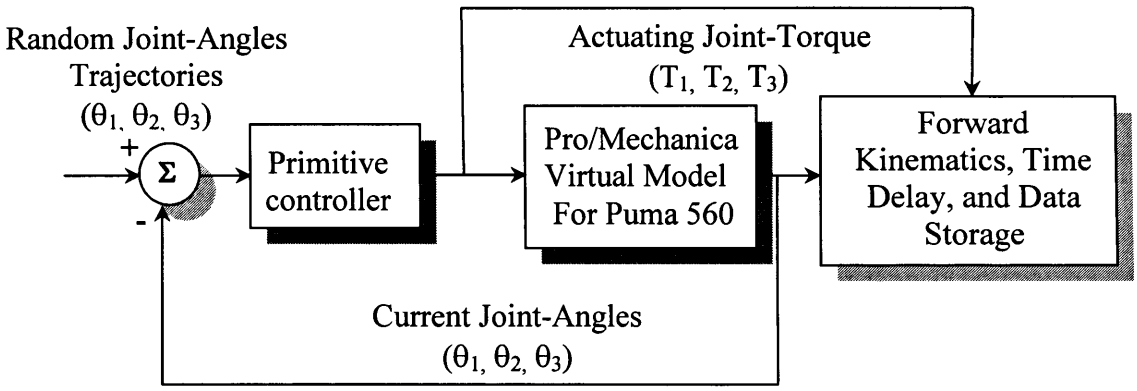


Figure (3.4). Data collection test for the "Puma 560®" robot arm.

The collected data will be used to construct the model of the inverse kinematics and inverse dynamics of the robot manipulator in two stages, a structure identification stage followed by parameters identification stage. Figure (3.5) illustrates these two stages.

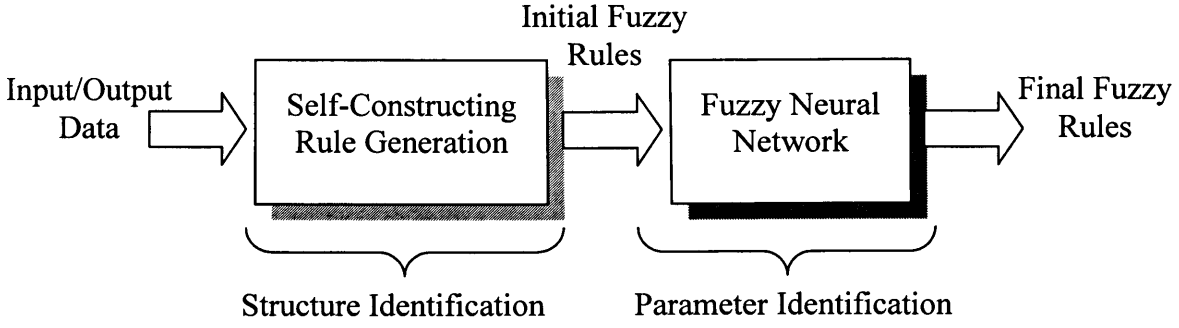


Figure (3.5). Steps in the proposed neuro-fuzzy modelling.

3.3.2. Inductive Learning Algorithm

The predefinition of the membership functions for both inputs and outputs used in Wang method [Wang and Mendel, 1992a] could be a difficult task. In fact, the problem of designing the membership function may be just as complex as designing the rules. The task of decomposition into membership functions can be seen as relatively similar to the task of discretisation in machine learning. One of the advantages of machine learning algorithms is that they permit the creation of compact models. In the next section, the fuzzy inductive learning algorithm *dynafuzz* [Bigot, 2003] is briefly explained including the suggested modifications. This method uses an automatic technique for input membership function creation during the rule forming process.

The algorithm is designed to extract fuzzy IF-THEN rules from a collection of examples (training set). Firstly, a manual step is performed to divide the output domain for continuous output examples to generate target classes. In the robot modelling case under investigation, the output is divided into equal, 50% overlapped Gaussian membership functions as shown in figure (3.6). The selection of the output membership functions shape, number, and degree of overlap is arbitrary. The Gaussian function is selected to allow for further tuning of the output membership function parameters in the Mamdani-type neuro-fuzzy network if required in the online learning stage. The number of the output membership functions can be regarded as the degree of precision prescribed for the model. The higher this number is; the higher should be the accuracy of the created rule set, however the number of rules will also be increased. This number

therefore gives the user a degree of control over the size and precision of the model to be created.

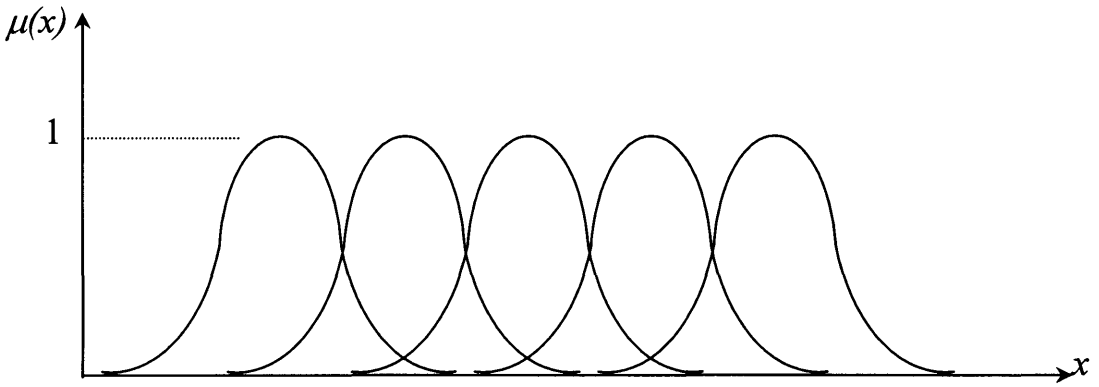


Figure (3.6). Selected output membership functions.

Each example (E) (input record) is described in terms of a fixed set of m (no. of inputs) attributes ($A^1, A^2, \ldots, A^m$) (equivalent to linguistic variables) and by a class (output) value (C_E). A range of values ($[V^i_{\min}, V^i_{\max}]$) (equivalent to linguistic membership functions) is assigned for the i^{th} attribute. Each created rule is composed of a number of conditions on each (or some of the) attribute(s) (Cdt_i) and by its class value (C_{rule}). Each rule can be represented as follows: $Cdt_1 \wedge Cdt_2 \wedge \ldots \wedge Cdt_m \rightarrow C_{rule}$. Each condition takes the form $[V^i_{\min} \leq A^i \leq V^i_{\max}]$ for continuous attributes i.

In order to create a rule set this algorithm incrementally employs a specific rule forming process until all examples are covered. Three particular steps of this process are of interest for the development of the fuzzy model.

The first step in this process is to select a seed example (SE), which is the first example in the list not covered by previously created rules. The second step consists of

employing a specific search process to create a consistent and general rule covering the (*SE*). The main feature of this search is that the conditions for continuous inputs are created automatically during the rule forming process. The continuous inputs are not pre-discretised (not divided into membership functions). The result is a rule where all continuous conditions will take the form $[V^i_{\min} < A^i < V^i_{\max}]$. These conditions might cover large areas in the example space. Thus, as the third and final step, the algorithm employs a post-processing technique that reduces the coverage of some continuous attribute conditions to the training data range only. This avoids the coverage of "unknown" areas and reduces the presence of overlapping rules.

3.3.2.1. Seed Example Selection

In this covering algorithm, a (SE) is selected from among the examples not covered by previously created rules. Then, the output class value (V^{output}_{SE}) (output fuzzy set) of (SE) is used as target class for the rule to be created. For instance, for the Gaussian membership function, the output membership function will be the fuzzy set (F_{SE} (a, b, c)) in which the membership degree will be maximum. In the particular case of 50% overlapping membership functions, where the membership degree is equal to 0.5 for two adjacent membership functions, only one of them is considered. Another problem occurs, with a fuzzy rule, because there are various degrees for the coverage of an example. In particular, one example might be covered and classified by a rule but its output value may have a low degree of belonging to the rule output membership function. This rule does not therefore properly represent the example. Thus, the "fuzzy algorithm" needs to create another rule for this example. In *dynafuzz*, the (SE) selected

is the first example in the training list that is not covered by at least one previously created rule where the degree of belonging of its output value to the rule output fuzzy set (F_{SE} (a,b,c)) is a maximum and ≥ 0.5.

3.3.2.2. Formation of a Rule

During the selection of (SE), the targeted fuzzy set has been identified (F_{SE}). Thus, for the discretisation of the example output values, it is proposed to classify as positive examples (belonging to the target class), those having an output value belonging to (F_{SE}) with (μ>0) and to classify the remaining examples as negative.

The search mechanism searches for rules that cover as many examples as possible from the target class and at the same time excludes examples belonging to other classes. The rule formation starts with a condition excluding the closest example not belonging to the target class. The assumption is that this also leads to the exclusion of the maximum number of other examples not belonging to the target class. To find the closest example, a measure is used to assess the distance between any two examples for the i^{th} continuous attribute as follows:

$$D_{example1\ \&example2} = \sqrt{\sum_{c}\left(\frac{Attr\_Value\_Ex.1 - Attr\_Value\_Ex.2}{Max\_Attr\_Value - Min\_Attr\_Value}\right)^2} \qquad (3.3)$$

where $\sum_{c}$ is the sum over all continuous attributes in the examples, Attr_Value_Ex.1 and Attr_Value_Ex.2 are the values of i^{th} continuous attribute in these two examples,

and Max_Attr_Value and Min_Attr_Value are the maximum and minimum known values for the i^{th} continuous attribute.

Applying this distance measure, the closest examples not belonging to the target class and covered by the rules formed so far can be found. Thus, in the next iteration the rule forming procedure considers appending only those conditions to the rule that exclude the closest examples.

For a particular uncovered example, the algorithm takes the closest example not belonging to the target class and creates candidate conditions to exclude it. These conditions are formed using attributes having different values for the considered two examples. The format of the formed condition will be [(attribute name) > or < (the attribute value of the closest example)]. For instance, if the attribute value of an example for which a rule is being created is V=5 and the attribute value of the closest example is V=10 then the resulted candidate condition will be [V < 10].

By applying this procedure, the algorithm handles continuous attributes generated from random operation of the robot manipulator arm. Thus, there is no need to pre-process the data in order to discretise the continuous input attribute data. The algorithm identifies splitting points for each continuous attribute range during the learning process.

At the end of the rule formation process, a rule is obtained belonging to the output class positive. Each condition will takes the form $[V_1^i < A^i < V_2^i]$ for continuous attributes, where V_1^i and V_2^i are continuous values included in the i^{th} continuous attribute range $[V_{min}^i, V_{max}^i]$.

3.3.2.3. Rule Post Processing

In the created rules, there is no need to account for overlapping and coverage of "unknown areas", as the fuzzy logic representation permits the handling of such uncertainties. In addition, it makes the transformation of condition ranges into membership functions more straightforward. Also, the use of the generated rules in the Mamdani-type neuro-fuzzy network will tune the obtained model to the most appropriate one.

After the rule forming process, the class of the rule (positive) is replaced by the targeted fuzzy set (F_{SE}) and each continuous conditions is transformed into a fuzzy condition using the following method in order to obtain the final fuzzy rule.

- For continuous attributes, considering the condition $[V_1^i < A^i < V_2^i]$ it is transformed into a membership function F(a, b, c):

- If V_1^i and V_2^i exist; a= V_1^i; b = V_2^i; and c = $(V_2^i + V_1^i)/2$.
- If V_1^i is equal to -∞; a= -∞; b=V_2^i; and c = V_{min}^i, which is the minimum known value of the attribute.

- If V_2^i is equal to +∞; a = V_1^i; b = +∞; and c = V_{max}^i, which is the maximum known value of the attribute.
- These values are then used to generate equivalent Gaussian and sigmoidal membership functions to be used in the Mamdani-type neuro-fuzzy network as shown in figure (3.7).

It can be noted that this algorithm allows the automatic creation of different membership functions for each continuous attribute in each created rule during the rule forming process.

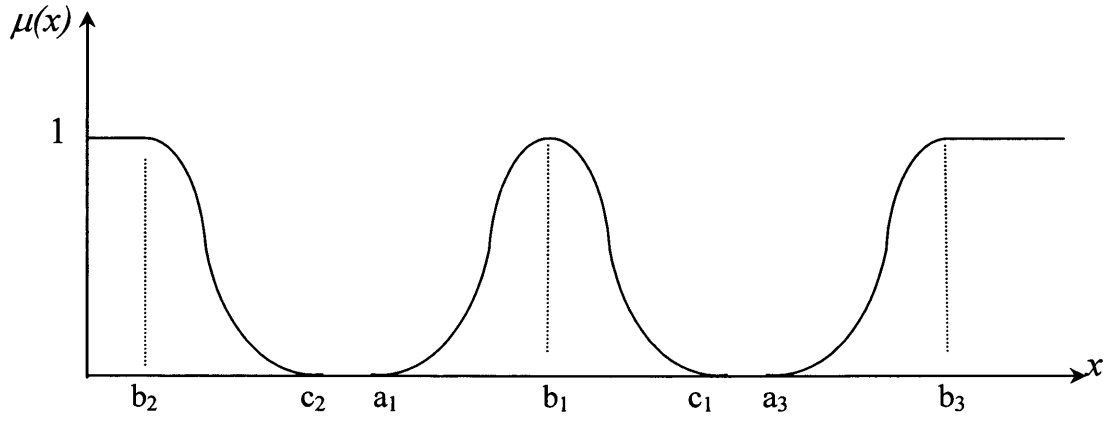


Figure (3.7). Types of generated input membership functions.

3.3.3. Inverse Kinematics and Inverse Dynamics Rules

To model the inverse kinematics of the robot arm in a Mamdani-type neuro-fuzzy network using the data recorded, a fuzzy rule-base that represents the inverse kinematics of the robot arm is generated first using the inductive learning rule

generation algorithm explained. Equation (3.4) expresses an approximate relation between the desired Cartesian position trajectory of the end-effector, the required joint angles trajectories to achieve this, and the current joint angles of the robot manipulator,

$$\theta_i^{k+1} \cong f(x^{k+1}, y^{k+1}, z^{k+1}, \theta_1^k, \theta_2^k, \ldots\ldots, \theta_n^k) \tag{3.4}$$

where k is the sampling interval, $i = (1,2,\ldots, n)$, n is the number of links, (x, y, z) are the end-effector Cartesian position, and θ is the joint angle.

Using equation (3.4) in addition to the data collected for full range operation of the joints from the virtual model for the first three links of the Puma 560® robot arm, an incremental-based model can be generated. Three sets of fuzzy rules can be generated representing the robot inverse kinematics. Each of these sets expresses a joint angle trajectory required to achieve the end-effector Cartesian trajectory as a function of this Cartesian trajectory and previous recorded values of the joint angles forming a 6-input single-output relationship. The entire training set is composed of 28,821 examples. The outputs have all been decomposed into 11 Gaussian membership functions. The resulting model is composed of 11 rules for the prediction of θ_1, 12 rules for the prediction of θ_2 and 16 rules for the prediction of θ_3, totaling 39 rules for performing the prediction of all outputs compared to more than eight hundred rules using Wang method [Wang and Mendel, 1992a]. Wherever an input variable is not mentioned in a rule, it means that this rule has no dependency on that particular input value. Each pair of margin values represent the generated membership function for that particular input as explained before.

Another rule-base representing the inverse dynamics of the robot arm is also generated. Equation (3.5) expresses an approximate relation between the desired joint angles trajectories of the robot arm, the required joint torques to achieve this, the current joint angles, and the current torques of the robot manipulator,

$$T_i^{k+1} \cong f(T_1^k,..., T_n^k, \theta_1^{k+1},..., \theta_n^{k+1}, \theta_1^k,..., \theta_n^k, v_1^{k+1},..., v_n^{k+1}, v_1^k,..., v_n^k) \quad (3.5)$$

where T is the joint torque and v is the joint velocity. Using equation (3.5) in addition to the data collected for full range operation of the joints from the virtual model for the first three links of the Puma 560® robot arm, three sets of fuzzy rules can be generated representing the robot inverse dynamics. Each set expresses a joint torque trajectory required to achieve the joint angle trajectories as a function of these trajectories and previous recorded values of these trajectories forming a 12-input single-output relationship. The entire training set is composed of 39,821 examples. The outputs have all been decomposed into 11 Gaussian membership functions. The resulting model is composed of 85 rules for the prediction of T_1, 92 rules for the prediction of T_2 and 51 rules for the prediction of T_3, totaling 228 rules performing the prediction of all outputs compared to more than fourteen hundred rules using Wang's method.

3.4. Proposed Neuro-Fuzzy Network (DYNAFUZZNN)

The proposed neuro-fuzzy network is a feedforward connectionist representation of a Mamdani-model based FLS so that the transparency of the fuzzy system generated so far is maintained. The neural network in fact employs time-delayed feedbacks from the

output layer to the input layer to represent the current state of the network output being feedback to the network input. Furthermore in order to achieve a suitable trade-off between the transparencies of the neurofuzzy system, the ease of mathematical analysis, and the effective application of back-propagation learning algorithm, the network has to employ differentiable alternatives for the logic-min and logic-max functions to implement its decision-making mechanism. For this purpose, a differentiable alternative of the logic-min function termed *softmin* and a differentiable alternative of the logic-max function termed *softmax* are presented [Estevez and Nakano, 1995; Shankir, 2001]. Using these two differentiable functions to implement the network decision-making mechanism allows a more accurate calculation of the partial derivatives, which are necessary for the back-propagation learning algorithm. In this way, online tuning for rule degree of confidence and for membership functions can be performed. The selected Gaussian and sigmoidal membership functions are differentiable and their parameters (a,b,c) can be tuned through back-propagation algorithm.

3.4.1. Softmin and Softmax Functions

Mamdani-model based FNN are the most commonly used FNN, but the parameter learning using BP through these networks is not accurate enough due to the chosen non-differentiable min/max functions. A few attempts have been presented to introduce analytical differentiable alternatives for the logic-min and logic-max functions [Yuan et al., 1992; Berenji and Khedkar, 1992]. In [Berenji and Khedkar, 1992] an analytical approximation of the logic min function termed *softmin*, is given by:

$$softmin\left(a_i, i=1,2,\ldots,n\right)=\frac{\sum_{i=1}^{n} a_i e^{-\zeta a_i}}{\sum_{i=1}^{n} e^{-\zeta a_i}} \tag{3.6}$$

where, a_i is the i^{th} argument and the parameter ζ controls the softness of the *softmin* function. As $\zeta \to \infty$, *softmin* function $\to$logic min. However, for a finite ζ, *softmin* becomes a multi-argument analytical approximation of the logic min function. [Estevez and Nakano, 1995] introduced the multi-argument *softmax* function used to approximate both the logic-max and logic-min function with a proper selection of parameters. Furthermore, based on De Morgan's law, which is valid for set theory and can be preserved for fuzzy sets, [Pedrycz, 1993; Shankir, 2001; and Zhang, 1996] presented a multi-argument alternative of the logic-max function termed *softmax* as a logic complement of the above mentioned *sofmin* function:

$$softmax(a_i, i=1,2,\ldots,n)=\left[1-\frac{\sum_{i=1}^{n} \overline{a_i} e^{-\zeta \overline{a_i}}}{\sum_{i=1}^{n} e^{-\zeta \overline{a_i}}}\right] \tag{3.7}$$

where $a_i = \mu_{A_i}$ and $\overline{a_i} = 1 - a_i$

These two differentiable functions will be utilized as the inference mechanisms within the neuro-fuzzy networks representing the model for both inverse kinematics and inverse dynamics of the Puma 560® robot manipulator.

3.4.2. DYNAFUZZNN Proposed Neuro-Fuzzy Network Structure

Figure (3.8) presents the structure of the proposed neuro-fuzzy network used for both inverse kinematics and inverse dynamics modelling. The network consists of a six-layer feedforward connectionist representation of a Mamdani-model based FLS, representing a Mamdani-model based RFNN. The network employs a full time-delayed feedback from output layer to input layer. This representation is due to the fact that the generated rules relating the output of the fuzzy system to the inputs which include the current state of the system, i.e. the previous time sample network output.

The network structure is similar to other Mamdani-model based FFNN with the first four layers have the same structure as the first four layers in Lin and Lee's FFNN [Lin and Lee, 1991] and in Berenji and Khedkar's FFNN [Berenji and Khedkar, 1992]. The difference is in the representation of the defuzzification function, which is represented using the last two layers (layer five and layer six). In Lin and Lee's FFNN and Berenji and Khedkar's FFNN the defuzzification function is represented using the last layer only (layer five). The reason for the chosen representation is to introduce adjustable scaling factors at the output layer in order to be able to tune the output membership functions during online adaptation.

In general, a node in any layer of the network has some finite fan-in of connections represented by weight values from other nodes and fan-out of connections to other nodes. Associated with the fan-in of a node is an aggregation function f that serves to combine information, activation, or evidence from other nodes.

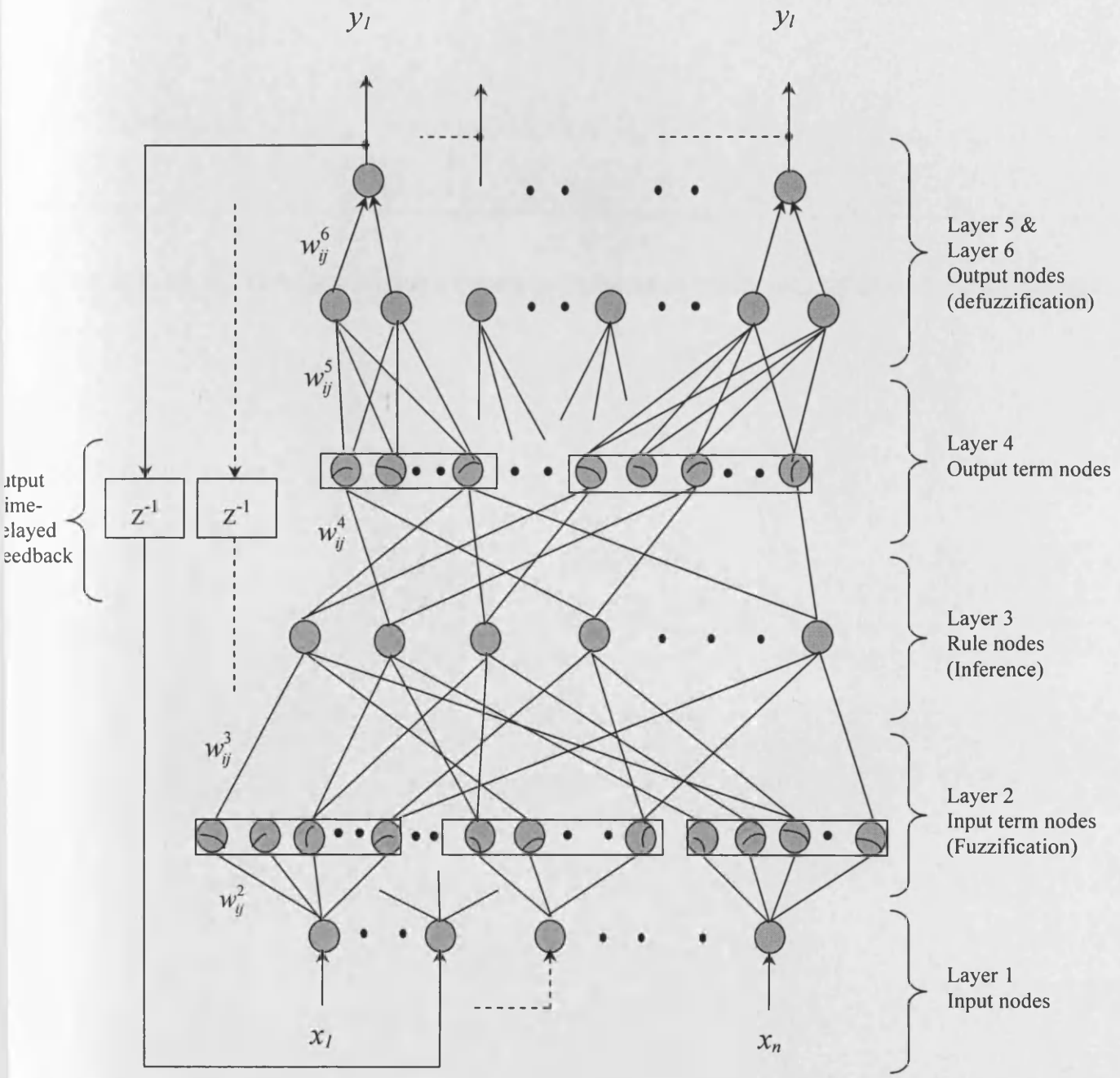


Figure (3.8). The structure of the proposed neuro-fuzzy network.

Using the same notation as in [Lin and Lee, 1991], the function provides the net input for such a node as follows:

$$input_{net} = f^k \begin{pmatrix} u_1^k, u_2^k, \ldots\ldots\ldots, u_p^k, \\ w_1^k, w_2^k, \ldots\ldots\ldots, w_p^k \end{pmatrix} \tag{3.8}$$

where p is the number of fan-ins of the node, w is the link weight associated with each fan-in, u is an output of a node in the preceding layer associated with the fan-in and the superscript k indicates the layer number. A second action of each node is to output an activation value as a function of its net-input,

$$output = o_i^k = a^k\left(f^k\right) \tag{3.9}$$

where $a^k(.)$ denotes the activation function in layer k. The functions of the nodes at each of the six layers of the proposed network are described next.

Layer 1: Nodes at layer one are input nodes, which represent input linguistic variables. Layer one contains n nodes ($n=6$ for inverse kinematics and $n=15$ for inverse dynamics), which receive a crisp input vector $X = (x_1, \ldots, x_n)$. The nodes in this layer simply transmit input values directly to the next layer. That is,

$$f_i^1 = u_i^1 = x_i \quad and \quad a_i^1 = f_i^1 \tag{3.10}$$

From the last equation, the link weights at layer one are fixed to unity.

Layer 2: Nodes at layer two are input term nodes which act as membership functions to represent the terms of the respective n input linguistic variables. An input linguistic variable x in a universe of discourse U is characterized by $A(x) = \{A_x^1, A_x^2, ..., A_x^m\}$, where $A(x)$ is the term set of x, that is the set of the generated membership functions for each input derived from inductive learning of the linguistic values of x, as explained previously. Layer two therefore accommodates n independent term sets, where each term set corresponds to an input x_i and is partitioned to m_i terms representing input membership functions. The function of each node j in a term set i is to calculate the degree of membership of the input x_i with respect to the membership function associated with the term set $A_j(x_i)$ according to the specific equation of this membership function:

$$f_{ij}^2 = \frac{\left(\left(w_{ij}^2 * a_i^1\right) - m_{ij}\right)^2}{\sigma_{ij}^2} \quad \text{and} \quad a_{ij}^2 = e^{-f_{ij}^2} \quad \textit{for Gaussian functions}$$

$$f_{ij}^2 = \frac{7*\left(\left(w_{ij}^2 * a_i^1\right) - \beta_{ij}\right)}{|\beta_{ij}|} \quad \text{and} \quad a_{ij}^2 = \frac{1}{1+e^{f_{ij}^2}} \quad \textit{for Left sigmoidals} \qquad (3.11)$$

$$f_{ij}^2 = \frac{7*\left(\beta_{ij} - \left(w_{ij}^2 * a_i^1\right)\right)}{|\beta_{ij}|} \quad \text{and} \quad a_{ij}^2 = \frac{1}{1+e^{f_{ij}^2}} \quad \textit{for Right sigmoidal functions}$$

where m_{ij} and σ_{ij} are, respectively, the centre (or mean) and the width (or variance) of the Gaussian function and β_{ij} is the characteristic value for the sigmoidal function. m_{ij}, σ_{ij}, and β_{ij} all calculated from the (a,b,c) parameters generated for each membership function from the offline inductive learning stage. Hence a link weight at layer two w_{ij}^2

can be interpreted as an adjustable free parameter of the input membership function. The tuning of this parameter (link weight) has the effect of tuning the membership function parameters (a,b,c).

Layer 3: The nodes at layer three are rule nodes which have been generated during the offline inductive learning stage explained previously; where each node associates one term node from each term set to form a condition part of one fuzzy rule if it is part of that rule. Hence, the rule nodes should perform the logic min operation if the min interpretation of the sentence connective "*and*" between the antecedents of a fuzzy rule is employed, or the algebraic product if the product interpretation of the sentence connective "*and*" is employed. In the proposed neuro-fuzzy network, the min interpretation is employed; consequently the logic-min function is replaced by the *softmin* function. Therefore the function of the r^{th} rule node using *softmin* can be written as follows:

$$f_r^3 = softmin(u_1^3, u_2^3, \ldots\ldots, u_q^3) = \frac{\sum_{i=1}^{q} u_i e^{-\zeta u_i}}{\sum_{i=1}^{q} e^{-\zeta u_i}} \qquad (3.12)$$

$$and \quad a_r^3 = f_r^3$$

where $r = 1,\ldots,R$, and R is the number of rules or rule nodes in layer three , q is the number of inputs for that particular rule, u_i is the i^{th} input to layer three, and ζ is an index representing the softness of the *softmin* function. However, in this layer, there are no link weights to be adjusted because all the link weights are fixed to unity to transmit only the membership degree of the linguistic input to the rule interpretation mechanism.

Layer 4: The nodes at layer four are output term nodes which act as membership functions to represent the output terms of the respective l linguistic output variables (in this case $l=3$). An output linguistic variable y in a universe of discourse W is characterized by $F(y) = \{F_y^1, F_y^2, ..., F_y^{11}\}$, where $F(y)$ is the term set of y which is equal to 11 in this case, that is the set of the class membership functions for each output, as explained previously, representing the linguistic values of y. Consequently layer four accommodates *three* independent term sets, where each term set corresponds to an output y_i and is partitioned to 11 terms representing output membership functions. The nodes in layer four should perform the logic-max operation to integrate the fired rules that have the same consequent. In the proposed neuro-fuzzy network the logic max function is replaced by the *softmax* function. Therefore, the function of each term node j in the output term set i, can be written as follows:

$$f_{ij}^4 = softmax(u_1^4, u_2^4, ..., u_p^4) = \left[1 - \frac{\sum_{i=1}^{n} \overline{u_i} e^{-\zeta \overline{u_i}}}{\sum_{i=1}^{n} e^{-\zeta \overline{u_i}}} \right] \qquad (3.13)$$

$$and \quad a_{ij}^4 = f_{ij}^4$$

where p is the number of rules sharing the same consequent (the same output term node), u_i is the ith input to layer four, and ζ is an index representing the softness of the *softmax* function. Hence the link weights at layer four are fixed to unity.

Layer 5: The number of nodes at layer five is $2l$, where l is the number of output variables, i.e. two nodes for each output variable. The function of these two nodes is to calculate the denominator and the numerator of an approximate form of Mean of

Maxima (MOM) defuzzification function [Saade, 1996; Runkler, 1997] for each output variable. The functions of the two nodes of the i^{th} output variable are described as:

$$f_{ni}^{5} = a_{ij}^{4} * m_{ij} \quad and \quad a_{ni}^{5} = f_{ni}^{5} \tag{3.14}$$

$$f_{di}^{5} = a_{ij}^{4} \quad and \quad a_{di}^{5} = f_{di}^{5} \tag{3.15}$$

where f_{ni}^{5} and f_{di}^{5} are respectively the node functions of the numerator and the denominator nodes of the i^{th} output variable. m_{ij} is the centre (or mean) of the Gaussian function of the j^{th} term of the i^{th} output linguistic variable y_i. Layer five employs *2l* weight vectors, with two weight vectors for each output variable. The first link weight vector connects the numerator node of the i^{th} output to the term nodes in its term set and its components are denoted by w_{nij}^{5}. Each component of this weight vector represents the centre (or mean) of the membership function of the j^{th} term of the term set of the i^{th} output variable. The second link weight vector connects the i^{th} output denominator node to the term nodes in its term set and its components are denoted by w_{dij}^{5}. Hence the link weights at layer five are fixed to unity.

Layer 6: The nodes at layer six are defuzzification nodes. The number of nodes in layer six equals the number of output linguistic variables. The function of the i^{th} node corresponding to the i^{th} output variable can be written as follows:

$$f_{i}^{6} = \frac{w_{ni}^{6} * a_{ni}^{5}}{w_{di}^{6} * a_{di}^{5}} \quad and \quad a_{i}^{6} = f_{i}^{6} \quad and \quad y_{i} = a_{i}^{6} \tag{3.16}$$

where w_{ni}^{6} and w_{di}^{6} are layer six link weights associated with each output variable node. These two link weights represent a scaling factor of an output variable.

3.4.3. Neuro-Fuzzy Network Parameters Tuning

Following the network construction phase, the network then enters the parameter learning phase to adjust its free parameters through online adaptation. The network adjustable free parameters were selected to be centres ($m_{ij}s$) of the output membership functions of the term nodes in layer four as well as the link weights at layers two and six. The supervised learning technique is employed along with the back-propagation learning algorithm to optimally tune these parameters. The problem for the supervised learning can be stated as: Given n input patterns $x_i(t)$, $i = 1,....,n$, and l desired output patterns $y_i(t)$, $i = 1,.....,l$, the fuzzy partitions, and the fuzzy rule base, adjust the network free parameters optimally. In the parameter learning phase, the network works in the feedforward manner, that is the goal is to minimize the following error function:

$$E = \frac{1}{2}\left(y(t) - y_{net}(t)\right)^2 \tag{3.17}$$

where $y(t)$ is the desired output, and $y_{net}(t)$ is the current network output. For each training data set, starting at the input nodes, a forward pass is followed to compute the activity levels of all the nodes in the network. Then, starting at the output nodes, a backward pass is followed to compute the rate of change of the error function with respect to the adjustable free parameters for all the hidden nodes. Assuming that (w) is the adjustable free parameter in a node, then the general learning rule can be written as follows:

$$\Delta w = -\frac{\partial E}{\partial w} \tag{3.18}$$

$$w(t+1) = w(t) + \eta \Delta w \tag{3.19}$$

where η is the learning rate, then using the chain rule, the partial derivative can be defined as follows:

$$\begin{aligned} \frac{\partial E}{\partial w} &= \frac{\partial E}{\partial (y(t) - y_{net}(t))} \frac{\partial (y(t) - y_{net}(t))}{\partial w} \\ &= \frac{\partial E}{\partial f} \frac{\partial f}{\partial w} = \frac{\partial E}{\partial a} \frac{\partial a}{\partial f} \frac{\partial f}{\partial w} \end{aligned} \tag{3.20}$$

Using the last learning rule, the calculations of the back-propagated errors as well as the updating of the free parameters can be described next starting at the output nodes:

Layer 6: Using Equation (3.20) and Equation (3.16), the adaptive rule to tune the weights of layer six is derived as follows:

$$\begin{aligned} \frac{\partial E}{\partial w^6_{ni}} &= \frac{\partial E}{\partial a^6_i} * \frac{\partial a^6_i}{\partial f^6_i} * \frac{\partial f^6_i}{\partial w^6_{ni}} = \\ &-(y(t) - y_{net}(t)) * \frac{a^5_{ni}}{w^6_{di} * a^5_{di}} \end{aligned} \tag{3.21.a}$$

$$w^6_{ni}(t+1) = w^6_{ni}(t) + \eta_6 \left(-\frac{\partial E}{\partial w^6_{ni}} \right) \tag{3.21.b}$$

$$\frac{\partial E}{\partial w_{di}^6} = \frac{\partial E}{\partial a_i^6} * \frac{\partial a_i^6}{\partial f_i^6} * \frac{\partial f_i^6}{\partial w_{di}} =$$

$$-\left(y(t) - y_{net}(t)\right) * \frac{-w_{ni}^6 * a_{ni}^5}{\left(w_{di}^6\right)^2 * a_{di}^5} \tag{3.22.a}$$

$$w_{di}^6(t+1) = w_{di}^6(t) + \eta_6\left(-\frac{\partial E}{\partial w_{di}^6}\right) \tag{3.22.b}$$

where η_6 is the learning rate of the link weights at layer six. The propagated error from layer six to the numerator and the denominator nodes at layer five are derived as follows:

$$\delta_{ni}^6 = \frac{\partial E}{\partial a_{ni}^5} = \frac{\partial E}{\partial a_i^6} * \frac{\partial a_i^6}{\partial f_i^6} * \frac{\partial f_i^6}{\partial a_{ni}^5} =$$

$$-\left(y(t) - y_{net}(t)\right) * \frac{w_{ni}^6}{w_{di}^6 * a_{di}^5} \tag{3.23.a}$$

$$\delta_{di}^6 = \frac{\partial E}{\partial a_{di}^5} = \frac{\partial E}{\partial a_i^6} * \frac{\partial a_i^6}{\partial f_i^6} * \frac{\partial f_i^6}{\partial a_{di}^5} =$$

$$-\left(y(t) - y_{net}(t)\right) * \frac{-w_{ni}^6 * a_{ni}^5}{w_{di}^6\left(a_{di}^5\right)^2} \tag{3.23.b}$$

Layer 5: At layer five, no adjustment is required for the link weights connected to the denominator nodes, while an adjustment is required for the link weights w_{nij}^5's which represent the centres m_{ij}'s of the output membership functions. Consequently, using

Equation (3.14) and Equation (3.20), the adaptive rule to tune the free parameters layer five is derived next. The adaptive rule to tune the centres of the output membership functions can be derived as follows:

$$\frac{\partial E}{\partial m_{ij}} = \frac{\partial E}{\partial a^5_{ni}} * \frac{\partial a^5_{ni}}{\partial f^5_{ni}} * \frac{\partial f^5_{ni}}{\partial m_{ij}} = \delta^6_{ni} * a^4_{ij} \tag{3.24.a}$$

$$m_{ij}(t+1) = m_{ij}(t) + \eta_5 \left(\frac{-\partial E}{\partial m_{ij}} \right) \tag{3.24.b}$$

where η_5 is the learning rate of the adjustable parameters (m_{ij}'s) at layer five. The propagated error from layer five to the j^{th} node in the i^{th} term set in layer four is derived as follows:

$$\begin{aligned} \delta^5_{ij} &= \left(\frac{\partial E}{\partial a^5_{ni}} * \frac{\partial a^5_{ni}}{\partial f^5_{ni}} * \frac{\partial f^5_{ni}}{\partial a^4_{ij}} \right) + \left(\frac{\partial E}{\partial a^5_{di}} * \frac{\partial a^5_{di}}{\partial f^5_{di}} * \frac{\partial f^5_{di}}{\partial a^4_{ij}} \right) \\ &= \left(\delta^6_{ni} * m_{ij} \right) + \left(\delta^6_{di} \right) \end{aligned} \tag{3.25}$$

Layer 4: No adjustment is required for the link weights of layer four. Only the error signals $\delta^4_r s$ need to be calculated and to be propagated to a rule node r in layer three. Each one of these error signals is a summation of L propagated error signals δ^4_{ri}, one error signal from a specific node j of each term set i, where $i = 1, \ldots, L$ and L is the

number of output variables (or term sets). Using Equation (3.20), the error signal δ_r^4 is calculated as follows:

$$\delta_r^4 = \sum_i \delta_{ri}^4 = \sum_i \left(\delta_{ij}^5 * \frac{\partial a_{ij}^4}{\partial f_{ij}^4} * \frac{\partial f_{ij}^4}{\partial a_r^3} \right) \tag{3.26}$$

Then from Equation (3.7) and Equation (3.13)

$$\frac{\partial a_{ij}^4}{\partial f_{ij}^4} = 1 \quad , \quad and$$

$$\frac{\partial f_{ij}^4}{\partial a_r^3} = \frac{-\left[\left(1 - \zeta \bar{a}_r^3\right) * e^{-\zeta \bar{a}_r^3} * \sum_{m=1}^{p} e^{-\zeta \left(\bar{u}_{ijm}^4 \right)} + \zeta * e^{-\zeta \bar{a}_r^3} * \sum_{m=1}^{p} \bar{u}_{ijm}^4 e^{-\zeta \left(\bar{u}_{ijm}^4 \right)} \right]}{\left(\sum_{m=1}^{p} e^{-\zeta \left(\bar{u}_{ijm}^4 \right)} \right)^2}$$

if the j^{th} term node at the i^{th} term set at layer four is connected to the r^{th} rule node at layer three, otherwise,

$$\frac{\partial f_{ij}^4}{\partial a_r^3} = 0$$

where p is the number of rules sharing the same j^{th} output term node, and $\bar{u}_{ijm}^4$ is the complement of the m^{th} input to the j^{th} output term node at the i^{th} term set at layer four.

Layer 3: Similarly to layer four, no adjustment is required for link weights at layer three. Only the error signals δ_{ij}^{3}'s need to be calculated and propagated from the r^{th} rule node at layer three to the j^{th} term node at the i^{th} term set at layer two. Each one of these error signals is a summation of p propagated error signals δ_{ijm}^{3} from layer three, where $m = 1,\ldots, p$, and p is the number of rules which share the same j^{th} term node at the same i^{th} input term set at layer two. Using Equation (3.20), the error signal δ_{ij}^{3} can be calculated as follows:

$$\delta_{ij}^{3} = \sum_{m} \delta_{ijm}^{3} = \sum_{m} \left(\delta_{m}^{4} * \frac{\partial a_{m}^{3}}{\partial f_{m}^{3}} * \frac{\partial f_{m}^{3}}{\partial a_{ij}^{2}} \right) \tag{3.27}$$

Then from Equation (3.6) and Equation (3.12),

$$\frac{\partial a_{m}^{3}}{\partial f_{m}^{3}} = 1 \quad , \quad and$$

$$\frac{\partial f_{m}^{3}}{\partial a_{ij}^{2}} = \frac{\left[\left(1 - \zeta a_{ij}^{2}\right) * e^{-\zeta a_{ij}^{2}} * \sum_{i=1}^{N} e^{-\zeta u_{mi}^{3}} + \zeta * e^{-\zeta a_{ij}^{2}} * \sum_{i=1}^{N} u_{mi}^{3} * e^{-\zeta u_{mi}^{3}} \right]}{\left(\sum_{i=1}^{N} e^{-\zeta u_{mi}^{3}} \right)^{2}}$$

if the j^{th} term node at the i^{th} input term set in layer two is connected to the rule node m at layer three, otherwise,

$$\frac{\partial f_m^3}{\partial a_{ij}^2} = 0$$

where N is the number of input term sets and u_{mi}^3 is the i^{th} input to the rule node m in layer three.

Layer 2: Using Equation (3.20) and Equation (3.11) the adaptive rule to tune the weights at layer two is derived as follows:

$$\frac{\partial E}{\partial w_{ij}^2} = \frac{\partial E}{\partial a_{ij}^2} * \frac{\partial a_{ij}^2}{\partial f_{ij}^2} * \frac{\partial f_{ij}^2}{\partial w_{ij}^2} = \delta_{ij}^3 * \frac{\partial a_{ij}^2}{\partial f_{ij}^2} * \frac{\partial f_{ij}^2}{\partial w_{ij}^2} \qquad (3.28.a)$$

$$w_{ij}^2(t+1) = w_{ij}^2(t) + \eta_2 \left(\frac{-\partial E}{\partial w_{ij}^2} \right) \qquad (3.28.b)$$

where $\frac{\partial a_{ij}^2}{\partial f_{ij}^2}$ is calculated as follows:

$= -e^{f_{ij}^2}$ *for Gaussian functions*

$= \frac{-e^{f_{ij}^2}}{\left(1+e^{f_{ij}^2}\right)^2}$ *for Left sigmoidal functions*

$= \frac{-e^{f_{ij}^2}}{\left(1+e^{f_{ij}^2}\right)^2}$ *for Right sigmoidal functions*

and $\frac{\partial f_{ij}^2}{\partial w_{ij}^2}$ is calculated as follows:

$$= \frac{2 * a_i^1 \left(\left(w_{ij}^2 * a_i^1 \right) - m_{ij} \right)}{\sigma_{ij}^2} \qquad \textit{for Gaussian functions}$$

$$= \frac{7 * a_i^1}{\left| \beta_{ij} \right|} \qquad \textit{for Left sigmoidal functions}$$

$$= \frac{-7 * a_i^1}{\left| \beta_{ij} \right|} \qquad \textit{for Right sigmoidal functions}$$

where η_2 is the learning rate of the link weights in layer two. The propagated error from layer two to the i^{th} input node at layer one is derived as follows:

$$\delta_i^2 = \sum_j \delta_{ij}^2 = \sum_j \left[\frac{\partial E}{\partial a_{ij}^2} * \frac{\partial a_{ij}^2}{\partial f_{ij}^2} * \frac{\partial f_{ij}^2}{\partial a_i^1} \right] = \sum_j \left[\delta_{ij}^3 * \frac{\partial a_{ij}^2}{\partial f_{ij}^2} * \frac{\partial f_{ij}^2}{\partial a_i^1} \right] \tag{3.29}$$

where $\frac{\partial f_{ij}^2}{\partial a_i^1}$ is calculated as follows:

$$= \frac{2 * w_{ij}^2 \left(\left(w_{ij}^2 * a_i^1 \right) - m_{ij} \right)}{\sigma_{ij}^2} \qquad \textit{for Gaussian functions}$$

$$= \frac{7 * w_{ij}^2}{\left| \beta_{ij} \right|} \qquad \textit{for Left sigmoidal functions}$$

$$= \frac{-7 * w_{ij}^2}{\left| \beta_{ij} \right|} \qquad \textit{for Right sigmoidal functions}$$

and $\frac{\partial a_{ij}^2}{\partial f_{ij}^2}$ is calculated as mentioned above.

Layer 1: The nodes in this layer just transmit input values to the next layer directly without any processing. So, the link weights at layer one are fixed to unity and no tuning is required in this layer.

Following the construction phase and the learning phase, an online tuning process is performed to obtain the optimum mapping for the inverse kinematics and inverse dynamics of the robot manipulator.

3.5. Puma 560® Manipulator Inverse Modelling Results

The modelling results for some random joint angle trajectories for the robot inverse kinematics compared with the real outputs and the filtered errors are shown in figure (3.9) through figure (3.14).

Also, the modelling results in per-unit (normalized) of the maximum joint torque for some random joint trajectories executed for the robot inverse dynamics compared with the real outputs and the filtered errors are shown in figure (3.15) through figure (3.20).

It can be seen from the modelling results that the suggested modelling method is very effective resulting in minor errors.

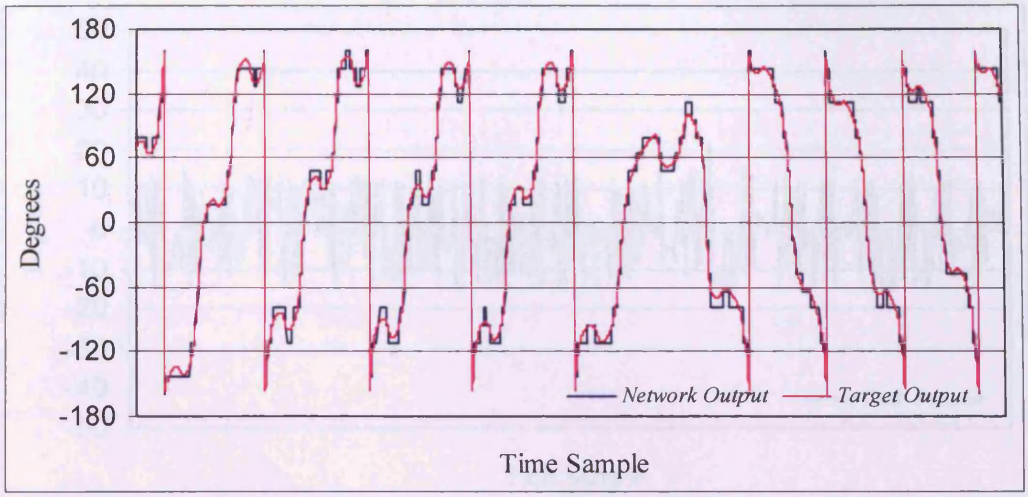


Figure (3.9). Results for link-1 angle prediction.

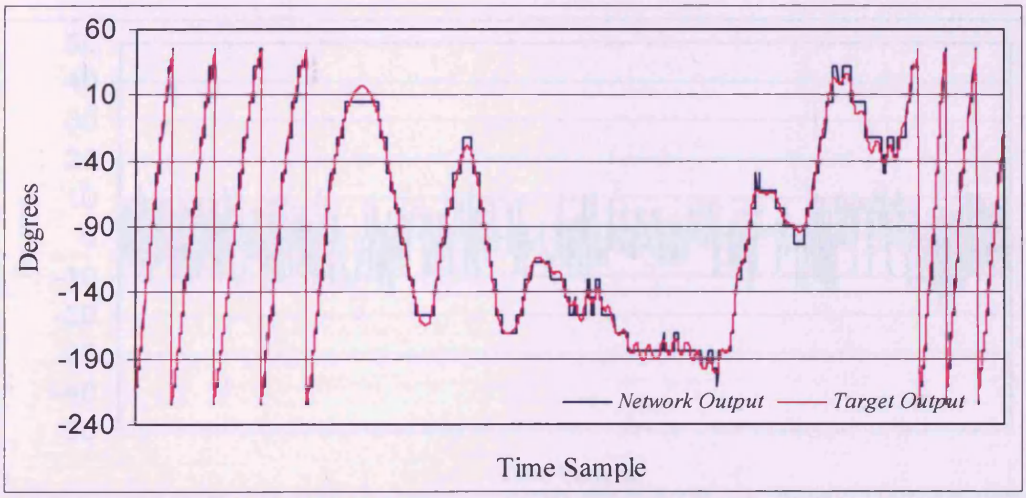


Figure (3.10). Results for link-2 angle prediction.

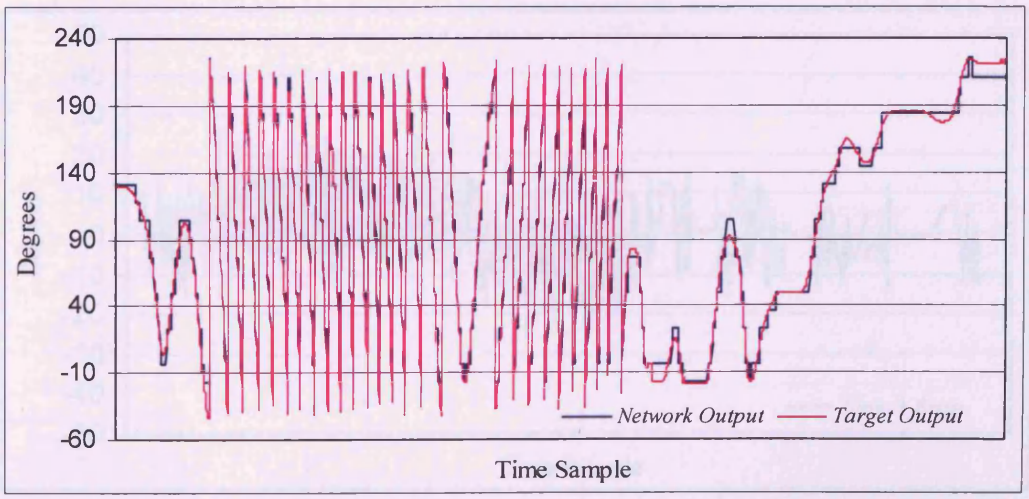


Figure (3.11). Results for link-3 angle prediction.

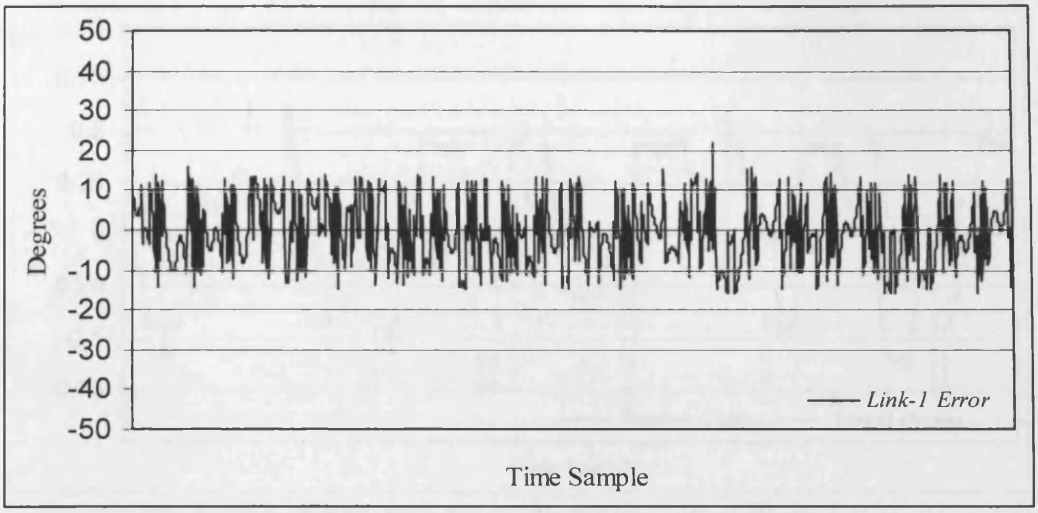


Figure (3.12). Results for link-1 angle error.

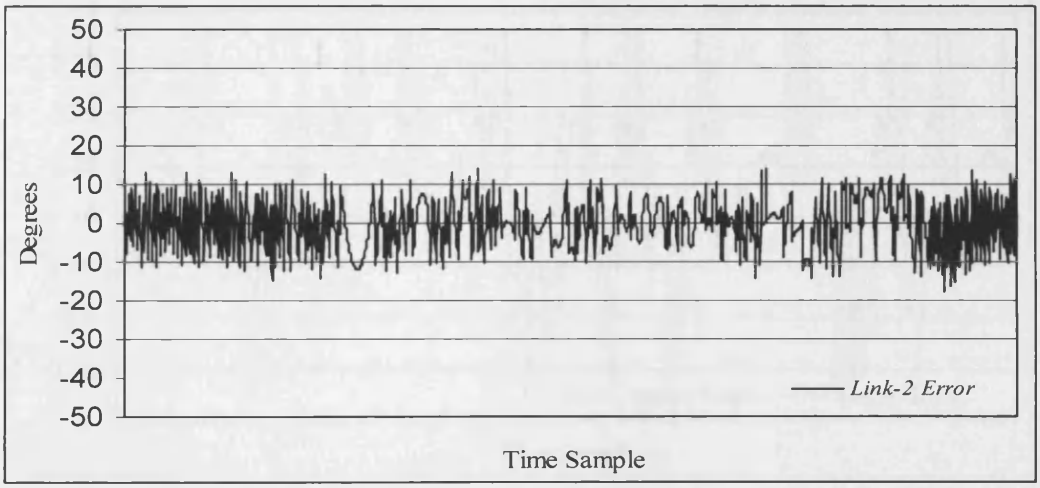


Figure (3.13). Results for link-2 angle error.

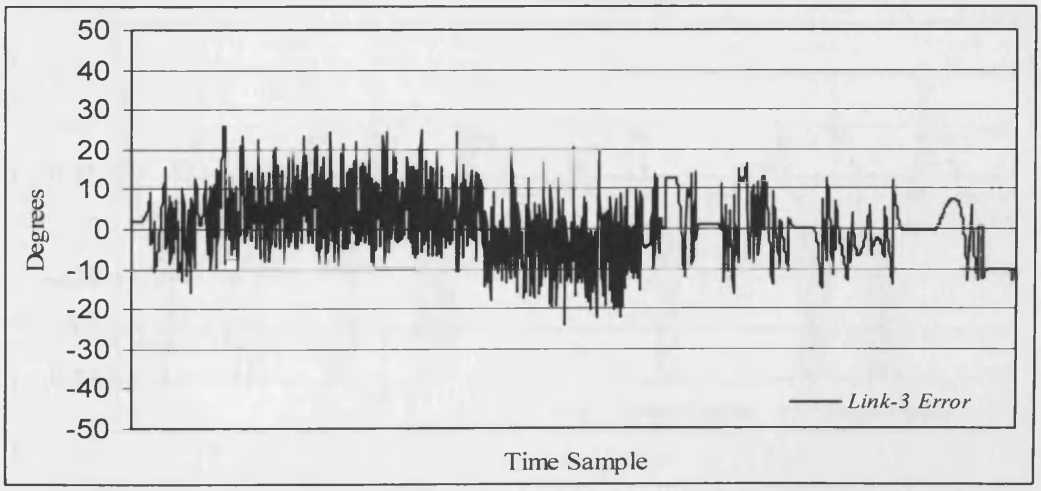


Figure (3.14). Results for link-3 angle error.

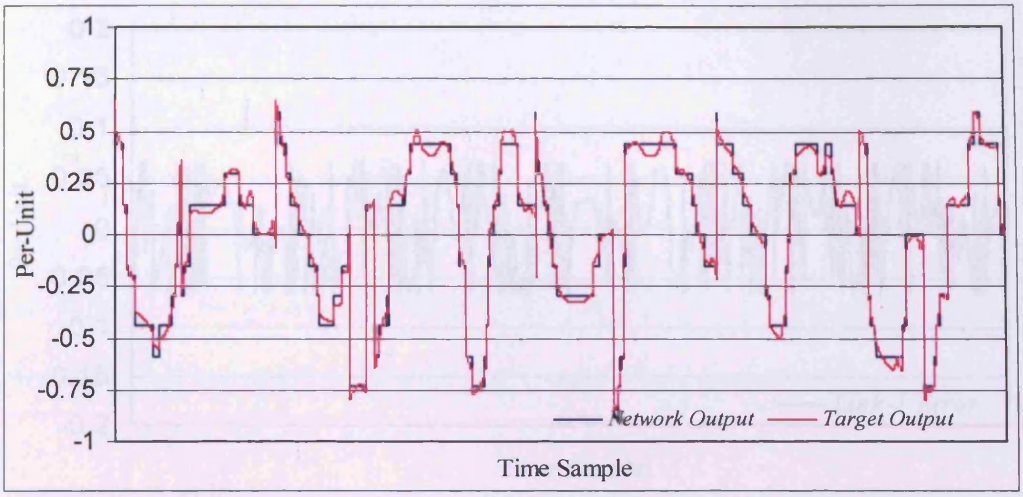


Figure (3.15). Results for link-1 torque prediction.

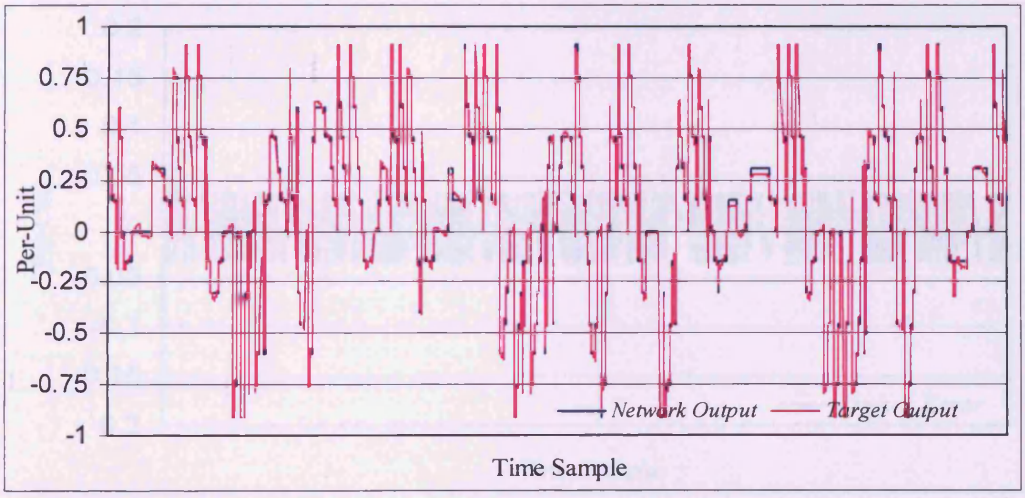


Figure (3.16). Results for link-2 torque prediction.

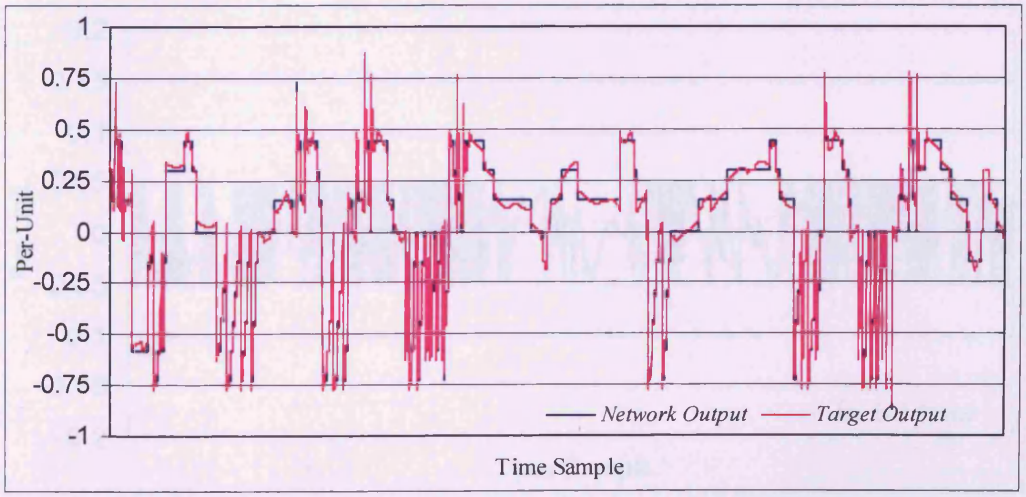


Figure (3.17). Results for link-3 torque prediction.

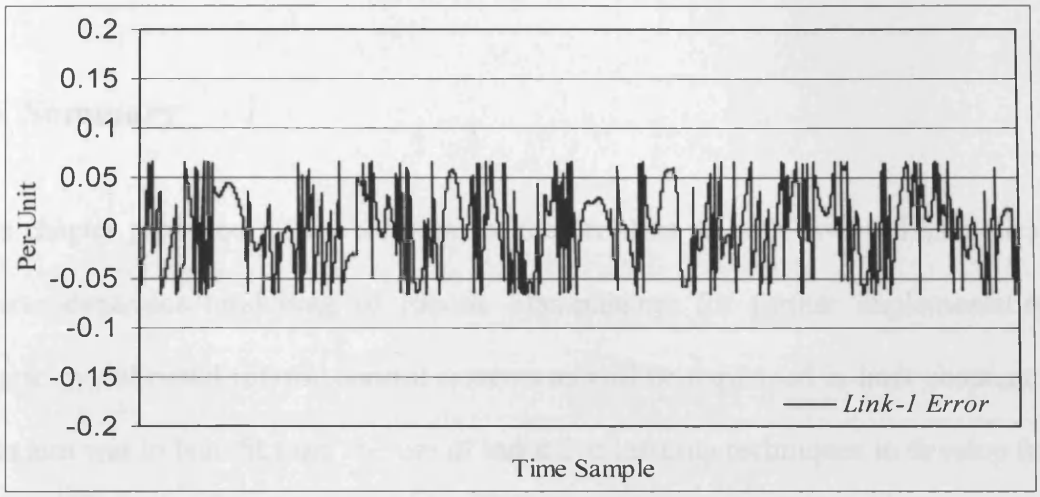


Figure (3.18). Results for link-1 torque error.

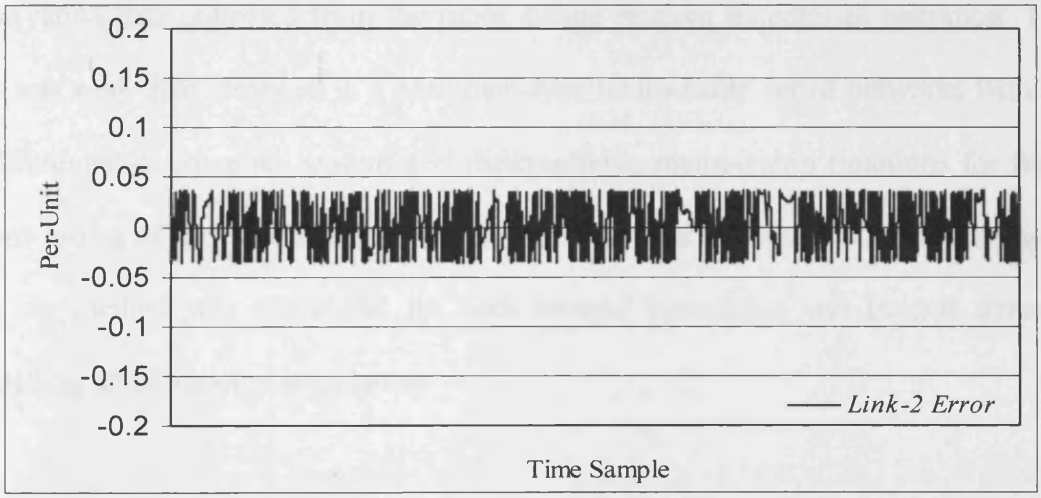


Figure (3.19). Results for link-2 torque error.

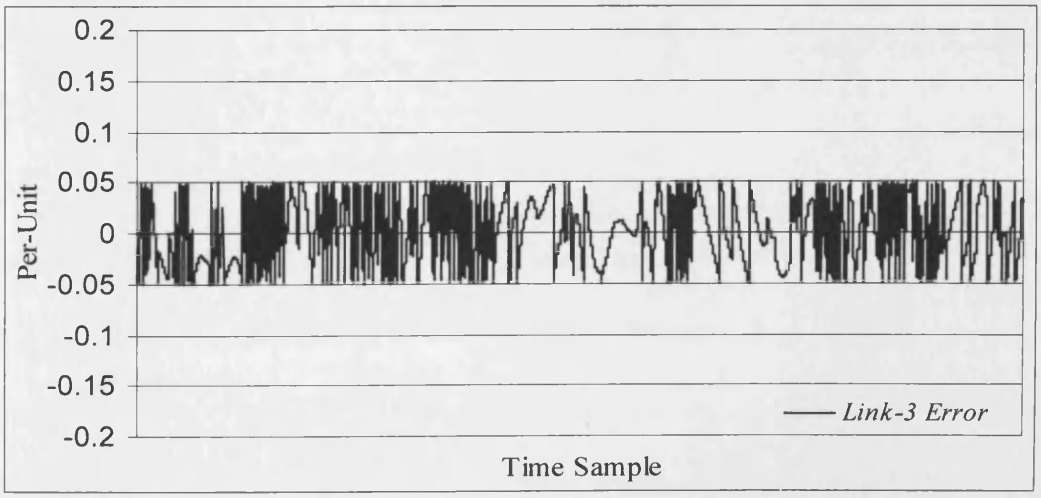


Figure (3.20). Results for link-3 torque error.

3.6. Summary

This chapter proposed a new solution for the problem of both inverse kinematics and inverse dynamics modelling of robotic manipulators for further implementation in inverse-model based robotic control systems as will be explained in later chapters. The main aim was to benefit from the use of inductive learning techniques to develop fuzzy-type rule sets for both inverse kinematics and inverse dynamics from numerical observation data collected from the robot during random trajectories operation. These rule sets were then arranged in a Mamdani-type neuro-fuzzy set of networks with both a differentiable inference system and differentiable membership functions for further online tuning of the obtained fuzzy models during inverse control. The results showed that the method was successful for both inverse kinematics and inverse dynamics modelling of the robot manipulators.

CHAPTER 4

Neuro-Fuzzy Joint-Based Control of Robotic Manipulators

Traditional proportional integral derivative (PID) controllers can be successful for systems that can be modelled relatively precisely by mathematical equations. Various combinations have been widely used for industrial processes due to their simplicity and effectiveness. PID controllers can be effectively used for first- and second-order linear systems, but usually cannot be employed for higher-order and nonlinear systems.

The control of a multi-input multi-output (MIMO) plant is a difficult problem when the plant is nonlinear and time varying and there are dynamic interactions between the plant variables. Robot manipulators, with two or more joints handling a changeable load, are of such type of systems. Conventional methods of designing controllers for a MIMO plant such as a multi-joint robot generally require, as a minimum, an accurate knowledge of the form of a mathematical model for the plant. In many cases, the values of the parameters of the model also need to be precisely known.

A model-based computed torque controller gives good control response if the dynamic model of the robot is available. For robot manipulators, it is almost impossible to identify precisely such a model and its parameters. Moreover, during operation, the dynamics of the robot may change significantly due to varying loading conditions. As a result, it is difficult to obtain an accurate mathematical model to allow computed torque

controllers or other model-based controllers to be accurately applied. This led to so called model-free control techniques.

The performance of model-free control techniques relies on incorporating a control scheme which is able to adapt (*adaptive control*) to uncertainties in the system dynamic parameters and to external disturbances. Neural Networks, which can learn the forward and inverse dynamics behaviour of complex plants, offer alternative methods of realising MIMO controllers capable of adapting to environmental changes. Neural Network controllers have been used extensively for adaptive robotic manipulator control. Most of the schemes utilizing Neural Networks use different learning techniques to adjust the inverse dynamic model of the robot manipulator contained in the Neural Network. In theory, the design of a Neural Network based control system should be relatively straightforward as it does not require any prior knowledge about the plant. However, practical problems regarding the Neural Network structure to be adopted, the number of input units and the training procedure, including training patterns, require investigation. This uncertainty regarding the appropriate network structure can result in large discrepancies between network output and desired output at the early stages of learning. This error can increase the learning time and convergence cannot be guaranteed.

In the last few decades, much research effort has been directed at the design of intelligent robotic controllers using fuzzy logic. These schemes provide nonlinear behaviour that is determined exclusively by the designer, lower sensitivity to plant parameter variations than Neural Network controllers, and simplicity of

implementation. Fuzzy control has been used for direct feedback control of robot manipulators [Erbature et al., 1995; Lin, 1993; Moudgal et al., 1995; and Tang et al., 2001]. In these examples, no adaptation for the rule base or membership functions of the fuzzy controller is carried out online, and only controller gain is modified in relation to link speed and joint errors within specific predetermined design parameters [Breedon et. al., 2002]. Although the idea of using fuzzy controllers for robotic manipulators was introduced in early 1990s, almost no systematic algorithm or detailed design procedure can be located in the literature. For example, the shape and location of the membership function for each fuzzy variable must be obtained using a heuristic (or trial-error) approach. Also, when the human expert cannot easily express his knowledge or experience in the form of linguistic "*IF-THEN*" control rules, it is not easy to construct the control rules.

A fuzzy logic system has the ability to express control rules as a linguistic fuzzy description but it has no learning capability. Neural Networks have the ability to generalize and can predict new output data from new input data, in real-time, without the need for a prior knowledge of the plant model. The fusion of these two approaches has the potential to produce a powerful intelligent control system having the features of adaptation and learning. Neural Networks are associated with the theory of polynomial function approximation, whereas fuzzy logic is based upon symbolic and linguistic processes expressed in an interactive rules base, with each rule fired with varying belief or support. The belief or confidence vector associated with a fuzzy logic rule base is equivalent to a weight in a Neural Network. [Er and Gao, 2003] presented a robust adaptive fuzzy neural controller (AFNC) suitable for motion control of multilink robot

manipulators. The proposed controller was of a self-organizing fuzzy Neural Network structure, where fuzzy control rules are generated or deleted automatically according to their significance to the control system and the complexity of the mapped system and no predefined fuzzy rules are used.

As mentioned before, in most inverse model control techniques, a feedback controller (servo controller) is used along with the Neural Network feedforward controller to improve the disturbance rejection capabilities of the control system. As learning proceeds, the error signal will reduce and the role of the feedforward neural controller increases while that of the feedback controller decreases [Pham and Yildirm, 1999]. Although the total torque acting on the robot is the sum of the feedback torque and the feedforward torque, these two play very different roles in the robot control. The feedback torque is used for clumsy but robust control at an early stage of learning, while the feedforward torque is necessary for smooth control and fast movement of the robot [Miyamoto et al., 1988]. Usually conventional P, PD, or PID controllers are used as the feedback controller in many reported works concerning Neural Network control of robot manipulators [Akbas and Esin, 2003]. The use of two feedback controllers in addition to the feedforward neuro-fuzzy controller is presented in [Peng and Woo, 2002]. The first feedback controller is a fuzzy-PD-like controller implemented in the form of a Neural Network with 15 rules derived from experience. The control strategy is to train this controller to approximate the optimum weights representing the optimum membership functions for the output torque. This is accomplished by using the data pairs collected from the system with a computed torque controller to train the Neural Network. This controller is trained offline and is kept fixed whilst online. The second

feedback controller is a sliding-mode controller calculated from the position and velocity errors of the robot joints. This controller is divided into two cascaded parts. The first part is a function of the position error and the velocity error, and the output of this function is used to train the feedforward neuro-fuzzy controller online. The second part is a sign function of the first part with a small constant gain representing the approximation error. The output of this part is used to make the tracking errors approach zero. The position and velocity references are chosen to be the input of the feedforward controller. [Peng and Woo, 2002] presents a complex control technique that depends on data collected from a computed-torque controller to train one of the feedback controllers. The structure of the main feedforward controller and the use of the computed-torque controller however are not clear.

This Chapter deals with the problem of the control of robot manipulators to track an arbitrary reference trajectory under the conditions of:

- A time-varying, nonlinear, multivariable, and coupled plant.
- An unknown plant and load model.

The main aim is to benefit from the approximate inverse dynamics neuro-fuzzy networks developed in Chapter (3) to achieve the required control. To do so, a nonlinear fuzzy-PID-like incremental controller is incorporated as a feedback servo-controller in addition to the developed network [Li, 1998; Mizumoto, 1995; and Yildirim et. al., 1996]. Incorporating *dynafuzznn* in the forward path controller gives the control system the proper structure and model parameters very close to those of the accurate robot inverse model. This in turns helps to reduce the convergence time of the

controller during online learning. For this purpose, two main parts are employed, the first is the neuro-fuzzy inverse dynamics of the robot manipulator developed at the offline stage, and the second is a fuzzy-PID-like incremental nonlinear controller. The direct inverse Neural Network controller is one of the several types of neuro-controllers which have been reported recently. It utilizes an inverse system model which can be directly cascaded with the controlled system. This approach relies on the fidelity of the inverse model used as the controller. Generally, serious problems arise due to the lack of robustness as a result of the absence of the feedback. This problem can be overcame to some extent by adjusting the parameters of the inverse model online, although the plant can still loose robustness during the control phase since it depends on the initial weight matrix of the Neural Network [Sasaki et. al., 1997]. Another approach to achieving an inverse Neural Network controller which aims to overcome this problem is known as feedback-error learning controller (specialized inverse learning controller). This scheme is based on using a workable traditional controller to stabilize the plant and on helping the Neural Network learn in order to provide precise control. A feedback-error learning control technique is used to form an efficient adaptive neuro-fuzzy controller. This technique differs from direct and indirect learning in that the controller no longer learns from input/output data pairs but from a direct evaluation of the network accuracy (during actual operation) with respect to the output of the plant. In this way, the feedforward controller will adapt its parameters (neuro-fuzzy network weights) to compensate for model changes during operation resulting from the attached load. The feedback controller response to the system error is used to tune the feedforward controller online. The suggested controller structure differs from previous work in two important aspects. The first is the use of the developed *dynafuzznn*

algorithm as the feedforward controller to achieve the proper controller structure. The second is the use of the fuzzy-PID-like incremental feedback controller to generate the nonlinear learning signal. Using a variable learning signal with variable feedback gain from a conventional PD servo controller has been reported to be successful in [Nascimento and McMichael, 1991]. This idea of variable feedback gain conventional PD servo controller motivated the idea of utilizing the fuzzy-PID-like incremental feedback controller to generate a nonlinear (variable gain) learning signal.

The remainder of this Chapter is organized as follows. Section 4.1 presents the proposed controller structure and the suggested fuzzy-PID-like incremental feedback controller. Section 4.2 explains the feedback-error learning scheme. Section 4.3 describes a comparative study of the simulation results for the developed controller while controlling the Puma 560® virtual model explained in chapter (3). Section 4.4 presents a summary for the Chapter.

4.1. Proposed Controller Structure

The structure of the proposed control system as shown in Figure (4.1) resembles the additive feedforward control that presented in [Craig, 1996] in some aspects. It consists of a feedforward path controller in addition to a feedback path controller. The net control action applied to the joints of the robot arm is the sum of the output from the feedforward controller and the output from the feedback controller. There are two differences in the proposed control system compared with that presented in [Craig, 1996]. The first is that a new nonlinear fuzzy-PID-like incremental controller is

adopted as the feedback servo-controller instead of the conventional linear PD controller. The second is that a new adaptive feedforward neuro-fuzzy network *dynafuzznn* developed in Chapter (3) is employed to incrementally approximate the inverse dynamics model of the robot arm instead of the linearised mathematical model. The feedback-error learning scheme described in Chapter (2) is used to tune the forward path neuro-fuzzy controller online. The suggested feedback fuzzy-PID-like incremental servo-controller provides a variable learning signal, which is necessary for robotic systems [Nascimento and McMichael, 1991].

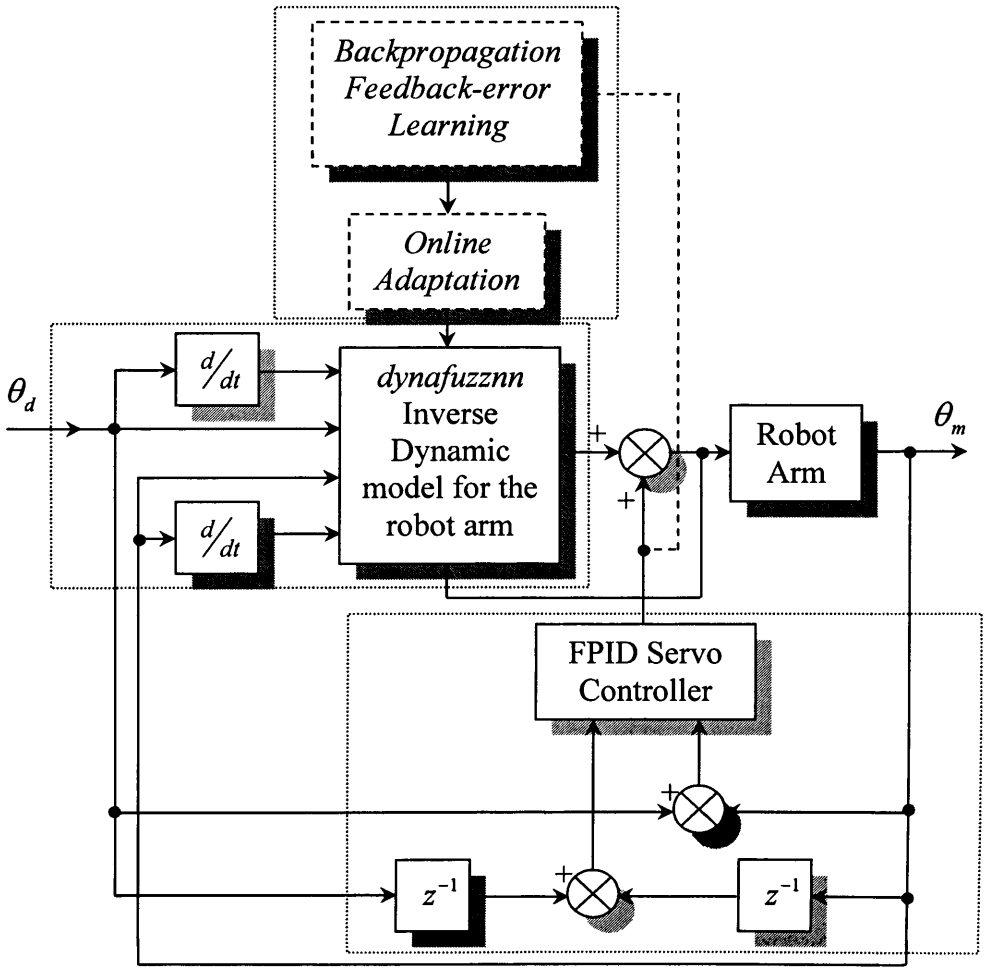


Figure (4.1). Proposed controller structure.

4.1.1. Forward Path Neuro-Fuzzy Controller

This path as shown in Figure (4.1) is a neuro-fuzzy approximate model of the inverse dynamics for the robot arm. The first step is to generate this model offline as explained in Chapter (3). The approximate inverse dynamic network has to be trained online to compensate for any external disturbances and/or reactions resulting from the attached load on the manipulator. This online training scheme utilizes the feedback controller response for adjusting the network link weights to adapt the network output so as to reduce the feedback controller response to zero.

4.1.2. Feedback Path Fuzzy-PID-like Incremental Servo Controller

This controller is mainly utilized to deal with the disturbances from the external load in early learning stages. The controller receives the error between the desired joint angles and the actual ones. It generates a control action, which is combined with the action from the feedforward controller to form the net torque (control action) applied to the joints of the robot.

Conventional PID controllers' output is proportional to an error, the time derivative of the error and the integral of the error. The controller employs a proportional control action to reduce the settling time and the rise time of the plant response, a derivative control action to reduce the overshoot and the oscillations of the plant response during transient conditions, and an integral control action to eliminate the steady state error during steady state conditions. This controller is easy to implement and sufficient tuning rules are available to cover a wide range of plant specifications. For example,

the well known Ziegler-Nichols [Ziegler and Nichols, 1942] tuning method can be applied to estimate the controller gains based on the transient response characteristics of a given system. Moreover, the available PID tuning heuristics are easy to understand and implement for simple practical control problems. This controller is more effective for linear plants than for nonlinear plants, due to its linear control policy. As explained before, FLC have been used successfully in nonlinear control applications. They generally provide nonlinear transfer elements for nonlinear control. The majority of FPID applications belong to the direct-action FPID type where the direct-action FPID is placed within the feedback control loop to compute the control actions through fuzzy inference. Several direct-action FPID structures have been reported using one, two or three inputs (error, rate of change of error and integral of error) [Mann et. al., 1999]. In all of these direct-action FPID controllers, the derivative and integral functions are performed quantitatively outside the FLC. They do not employ a FLS as a function approximator to perform a fuzzy integral or fuzzy derivative function. In these controllers, the FLS performs the nonlinear amplifications associated with the three PID control actions. For this work a new Fuzzy-PID controller [Shankir, 2001] is adapted with extended rules; this controller functionally performs fuzzy derivative and fuzzy integral functions, so that no calculations are required outside the FLC. The suggested fuzzy-PID-like incremental controller employs only two inputs (present and previous errors), so that the design procedure is simpler. Each element of the fuzzy-PID-like incremental controller can approximate the corresponding control function with separate nonlinear gain using five fuzzy set partitions (NL, NS, ZE, PS, and PL) for both input and output universes of discourses. The input universe of discourse of each input variable is uniformly partitioned using fuzzy sets defined by symmetrical

triangular membership functions with 50% overlap to allow continuous approximation of input signals as shown in figure (4.2). The left most and the right most membership functions of the input universe of discourse are saturated to unity membership value in the domain less than -2L and more than +2L respectively, where L is the distance between two consecutive membership functions centres. The output universe of discourse is uniformly partitioned using fuzzy sets defined by symmetrical triangular membership functions with 50% overlap as shown in figure (4.3). The left most and the right most membership functions of the output universe of discourse are both limited to the output minimum and maximum range of operation in the domain less than $-2L_i$ and more than $+2L_i$ respectively. These minimum and maximum ranges in addition to controller gains are related to the maximum permissible servo torque applied to the robot joints. L_i represents the distance between two consecutive output membership functions centres where, i is replaced by P, I, or D according to the proportional, integral, or derivative control element respectively.

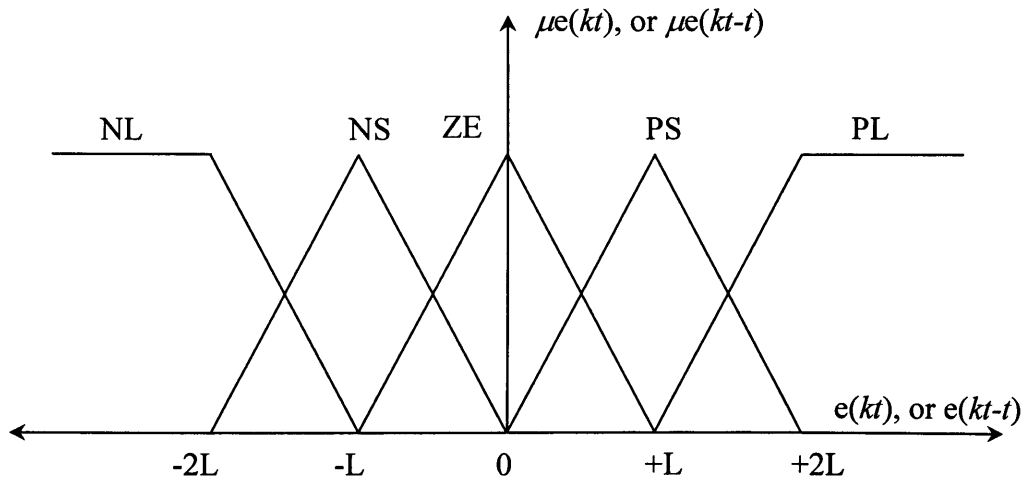


Figure (4.2). Input membership functions of fuzzy controller.

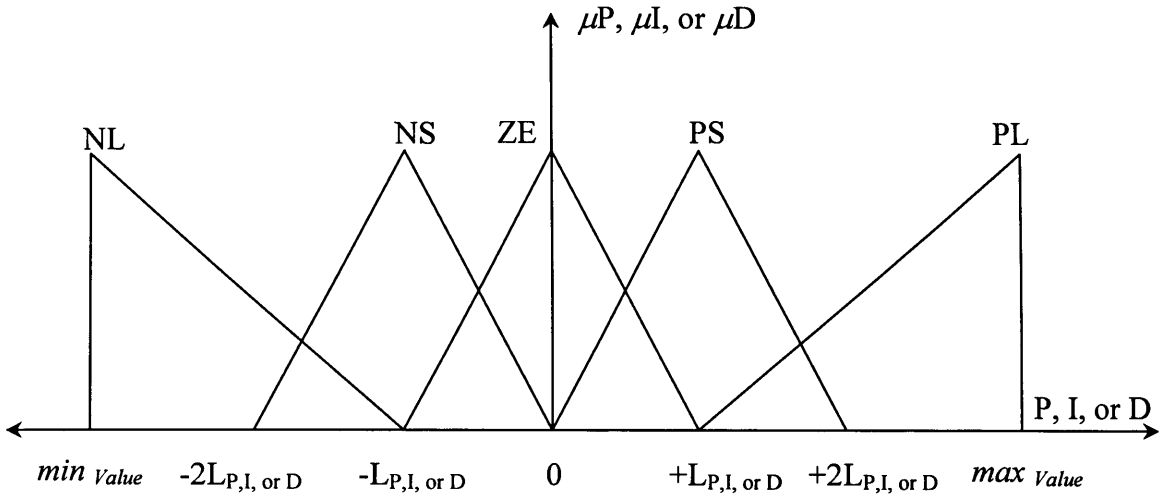


Figure (4.3). Output membership functions of fuzzy controller.

The proportional, derivative and incremental part of the integral control actions of a fuzzy-PID-like incremental controller are mainly functions of the two present and past error variables, $err(kt)$ and $err(kt-t)$, or their normalized variables, $e(kt)$ and $e(kt-t)$. Consequently,

$$U_{PID}(kt) = f_P(e(kt), e(kt\text{-}t)) + f_D(e(kt), e(kt\text{-}t)) + U_I(kt-t) + f_I(e(kt), e(kt\text{-}t)) \tag{4.1}$$

where the three functions f_P, f_D, and f_I are the proportional, derivative and incremental integral functions to be implemented using the fuzzy logic controller and $U_I(kt\text{-}t)$ is the past output of the integral controller element. It was proved in [Wang and Mendel, 1992] that fuzzy logic systems are universal approximators. Therefore, the three functions in equation (4.1) can be approximated using three two-input Fuzzy Control Elements (FCEs). Consequently, the outputs of the three FCEs are summed together to form the proposed fuzzy-PID-like incremental controller as shown in figure

(4.4). In the following sections, the design of the operation rules and implementation of the three functions in equation (4.1) in the form of three fuzzy control elements are explained.

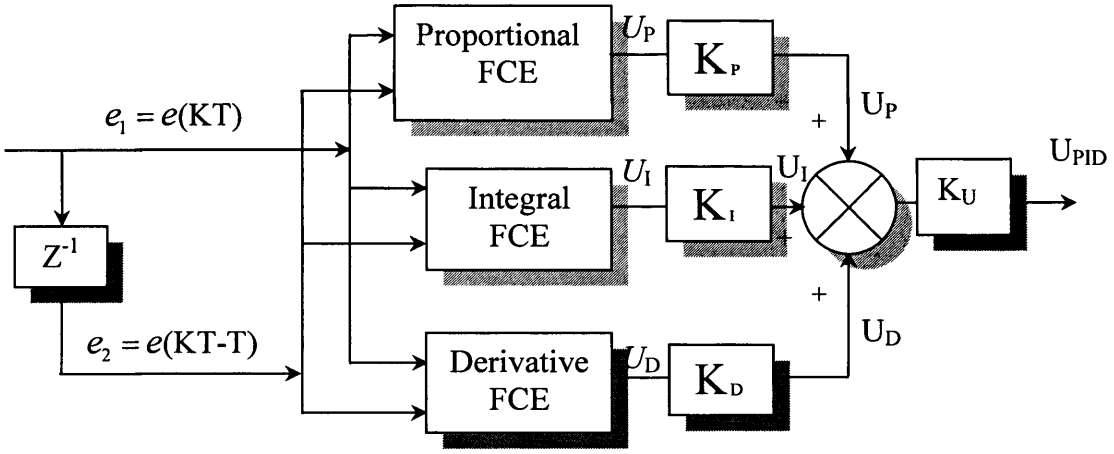


Figure (4.4). Structure of the fuzzy servo controller.

4.1.3. Design Procedures for fuzzy-PID-like Incremental Controller

Let the two error variables $err(kt)$ and $err(kt-t)$ be defined as err_1 and err_2 respectively. After normalizing the error variables, let the normalized error variables be given by $e(kt) = K_{e_1} err_1 = e_1$, and $e(kt-t) = K_{e_2} err_2 = e_2$, where K_{e_1} and K_{e_2} are the scaling factors corresponding to the two input variables. Because the two input variables are of the same nature, their scaling factors are equal, i.e. $K_{e_2} = K_{e_1} = K_e$, and their input universes are designed similarly as shown in figure (4.2). Each output function of the fuzzy PID-like incremental controller is of a different nature (proportional, derivative, or integral). Therefore the partition of the output universe of discourse is selected to be of the same membership function shape and degree of

overlapping but with different scaling factors to allow for different tuning of each control element. The design value for the membership functions adapted for the three output universe of discourses are (L=0.3 p.u), (L_P=0.2 p.u), (L_D=0.15 p.u), and (L_I=0.05 p.u) of the error domain range. These assigned values result in a small gain integral element compared to the other controller elements.

4.1.3.1. Fuzzy Proportional Control Element

To derive the general output of the Fuzzy Proportional Control Element (FPCE), the input universes of the normalized input variables e_1 and e_2 are partitioned into five fuzzy sets with five membership functions as shown in figure (4.2). A five-fuzzy-set output universe is considered for the normalized output $U_P(kt)$ as shown in figure (4.3). For the case of FPCE, the distance between the centres of any two adjacent membership functions is L_P. The fuzzy rules of the operation of the FPCE according to the suggested partitions are generated heuristically based on the intuitive concept that the proportional control action at any time step is directly proportionally to an error e_1 at the same time step regardless of the value of the error at the previous time step e_2. Therefore if the error variable e_1 is expressed linguistically as zero, positive small, positive large, negative small, or negative large, the proportional control action can be expressed linguistically as zero, positive small, positive large, negative small, or negative large respectively, regardless of the linguistic value of the error variable e_2. Consequently, the Fuzzy Associative Memory (FAM) rules according to this concept of the FPCE can be written as shown in table (4.1).

e_1 \ e_2	NL	NS	ZE	PS	PL
NL	NL	NL	NL	NL	NL
NS	NS	NS	NS	NS	NS
ZE	ZE	ZE	ZE	ZE	ZE
PS	PS	PS	PS	PS	PS
PL	PL	PL	PL	PL	PL

Table (4.1). Proportional element FAM bank.

where [NL, NS, ZE, PS, PL] are the term sets of the normalized input variables e_1 and e_2 and the normalized output variable $U_P(kt)$. From the above rule-base, it can be seen that although the consequent depends only on e_1, the rules employ two antecedent variables corresponding to the two variables e_1 and e_2. The reason for this is to unify the number of antecedent variables in the rules of the three FCEs (P, I, and D). Therefore, the rules in the three FCEs can be integrated into one rule base to represent the fuzzy-PID-like incremental controller. The rules in this integrated rule base have two-variable antecedents corresponding to the input variables (e_1 and e_2) and three-variable consequents corresponding to the three fuzzy-PID-like incremental controller elements (P, I, and D).

To infer the fuzzy output of the FPCE, Mamdani's min/max method using the bounded sum triangular co-norm is employed. In [Yuan et al., 1992], the *fmin* and *fmax* functions were introduced to approximate the logic min and logic max functions analytically. These two functions were formulated as follows:

$$fmin(h_{e1}, h_{e2}) = 0.5\left[(h_{e1} + h_{e2}) - \sqrt{(h_{e1} - h_{e2})^2 + (0.01)^2} + 0.01\right] \quad (4.2)$$

$$fmax(h_{p1}, h_{p2}) = 0.5\left[(h_{p1} + h_{p2}) + \sqrt{(h_{p1} - h_{p2})^2 + (0.01)^2} - 0.01\right] \quad (4.3)$$

where $(h_{e1} \text{ and } h_{e2})$ are defined as the fuzzy membership values of the input error variables (e_1 and e_2), while $(h_{p1} \text{ and } h_{p2})$ are defined as the fuzzy membership values of the same output membership function resulting from any two different rules at any time step. For generality, the *softmin* and *softmax* functions presented in Chapter (3) can replace equations (4.2) and (4.3). The centre average defuzzification method (Height method) [Ying et. al., 1990; Ying, 1993] is employed to calculate the crisp output of the FPCE. The use of these inference and defuzzification methods with overlapping triangular membership functions for both input and output variables defines the nonlinearity of the fuzzy controllers. Consequently, based on the defined membership functions, only four rules are triggered at a time. Therefore, the inference system produces four non-zero fuzzy outputs for the two crisp error inputs. The fuzzy output of a rule (output fuzzy sets after inference) is a fuzzy set with a trapezoid membership function whose height (h) equals the membership degree produced by the

min operator of equation (4.2). Based on the input errors condition, employed inference method, and defuzzification method used, a numerical example for the fuzzy output and the (h) value for each of the four rules for the FPCE are shown in figure (4.5).

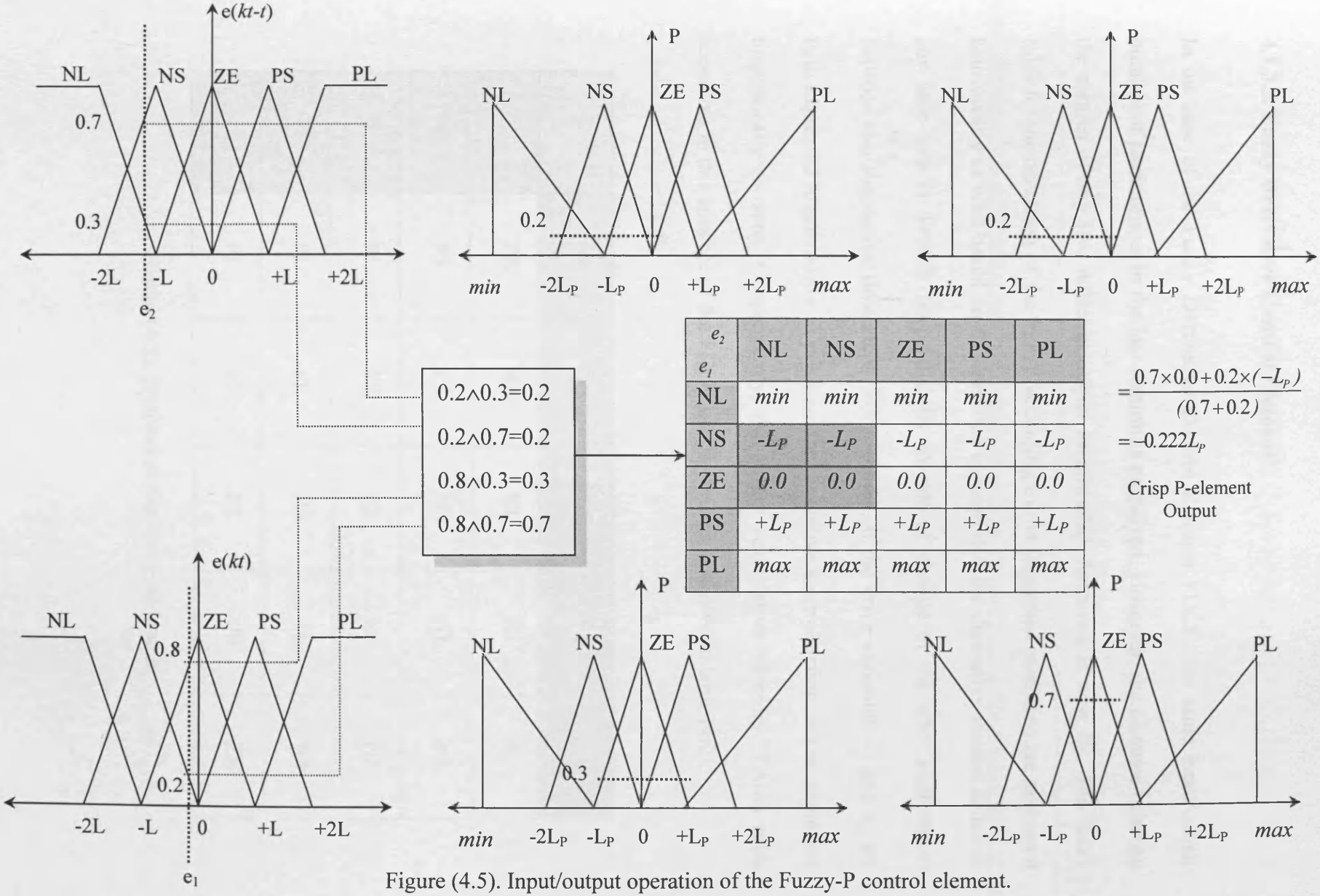
The output of the FPCE is calculated for any input condition using the centre average defuzzification method, assuming different membership output function for each rule inference, as follows:

$$FPCE_{output} = \frac{\sum_{i=1}^{4}\left[h\ value\ of\ the\ input\ Mf\ with\ min\ h\ *\ output\ Mf\ centre\right]_{Rule_i}}{\sum_{i=1}^{4}\left[h\ value\ of\ the\ input\ Mf\ with\ min\ h\right]_{Rule_i}} \qquad (4.4)$$

Using equations (4.2) and (4.4), the analytical solution of the proportional function of the FPCE $f_P(e_1, e_2)$ in Equation (4.1) can be expressed as follows:

$$FPCE_{output} = \frac{\sum_{i=1}^{4}\left[Cop_R\left((\mu_R(e_1)+\mu_R(e_2))-\sqrt{\left(\mu_R(e_1)-\mu_R(e_2)\right)^2+\left(0.01\right)^2}+0.01\right)\right]_{Rule_i}}{2*\sum_{i=1}^{4}\left[(\mu_R(e_1)+\mu_R(e_2))-\sqrt{\left(\mu_R(e_1)-\mu_R(e_2)\right)^2+\left(0.01\right)^2}+0.01\right]_{Rule_i}} \qquad (4.5)$$

where Cop_{Ri} is the FPCE output membership function centre value for rule i, $\mu_{Ri}(e_1)$ is the membership degree of the present error to the rule i, and $\mu_{Ri}(e_2)$ is the membership degree of the past error to the rule i.



e_2 / e_1	NL	NS	ZE	PS	PL
NL	*min*	*min*	*min*	*min*	*min*
NS	$-L_P$	$-L_P$	$-L_P$	$-L_P$	$-L_P$
ZE	*0.0*	*0.0*	*0.0*	*0.0*	*0.0*
PS	$+L_P$	$+L_P$	$+L_P$	$+L_P$	$+L_P$
PL	*max*	*max*	*max*	*max*	*max*

Figure (4.5). Input/output operation of the Fuzzy-P control element.

4.1.3.2. Fuzzy Derivative Control Element

In the case of the Fuzzy Derivative Control Element FDCE, the same input/output number of partitions as in the last section is employed. However, the distance between the centres of any two adjacent output membership functions is now L_D. The fuzzy rules for the operation of the FDCE according to the suggested partitions are generated heuristically as well based on the intuitive concept that the derivative control action at any time step is directly proportionally to rate of change of the error (difference) between two successive time steps. For example, if the error variables e_1 and e_2 are both expressed linguistically as positive, the derivative control action can be expressed linguistically as zero. Consequently, the Fuzzy Associative Memory (FAM) rules according to this concept of the FDCE can be written as shown in table (4.2).

e_1 \ e_2	NL	NS	ZE	PS	PL
NL	ZE	NS	NL	NL	NL
NS	PS	ZE	NS	NL	NL
ZE	PL	PS	ZE	NS	NL
PS	PL	PL	PS	ZE	NS
PL	PL	PL	PL	PS	ZE

Table (4.2). Derivative element FAM bank.

where [NL, NS, ZE, PS, PL] are the term sets of the normalized input variables e_1 and e_2 and the normalized output variable $U_D(kt)$.

Consequently, based on the defined membership functions, only four rules are triggered at a time. Therefore, the inference system generally produces four non-zero fuzzy outputs for the two crisp error inputs. The fuzzy output of a rule (output fuzzy sets after inference) is a fuzzy set with a trapezoid membership function whose height (h) equals the membership degree produced by the min operator of equation (4.2) during the fuzzy inference. Based on the input errors condition, employed inference method, and defuzzification method used, a numerical example for the fuzzy output and the (h) value for each of the four rules for the FDCE are shown in figure (4.6).

Using the same inference and defuzzification methods in the last section, the analytical solution of the FDCE function $f_D(e_1, e_2)$ in equation (4.1) can be written as follows:

$$FDCE_{output} = \frac{\sum_{i=1}^{4}\left[Cod_R\left((\mu_R(e_1)+\mu_R(e_2))-\sqrt{\left(\mu_R(e_1)-\mu_R(e_2)\right)^2+\left(0.01\right)^2}+0.01\right)\right]_{Rule_i}}{2*\sum_{i=1}^{4}\left[(\mu_R(e_1)+\mu_R(e_2))-\sqrt{\left(\mu_R(e_1)-\mu_R(e_2)\right)^2+\left(0.01\right)^2}+0.01\right]_{Rule_i}} \quad (4.6)$$

where Cod_{Ri} is the FDCE output membership function centre value for rule i, $\mu_{Ri}(e_1)$ is the membership degree of the present error for the rule i, and $\mu_{Ri}(e_2)$ is the membership degree of the past error for the rule

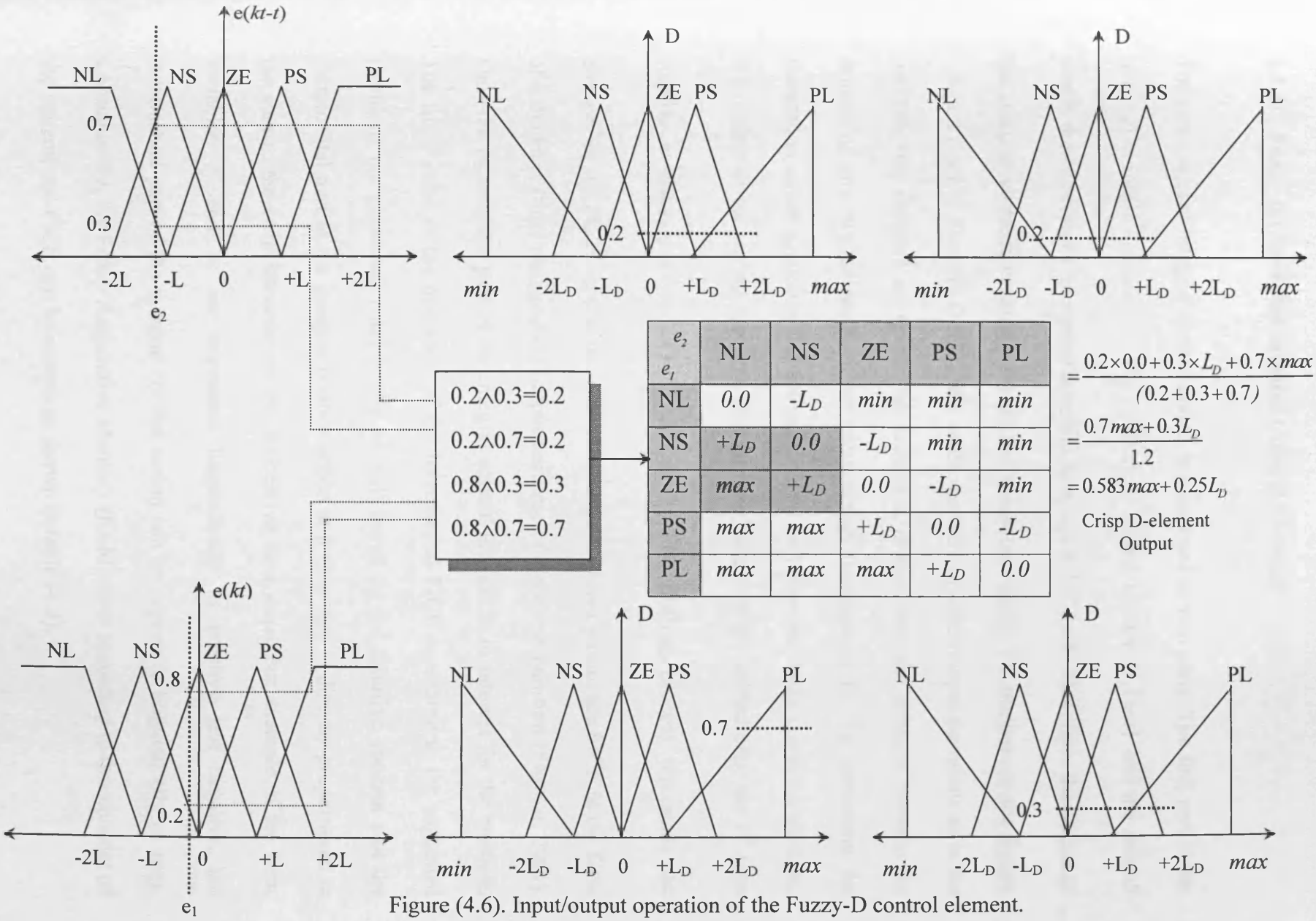


Figure (4.6). Input/output operation of the Fuzzy-D control element.

4.1.3.3. Fuzzy Incremental Integral Control Element

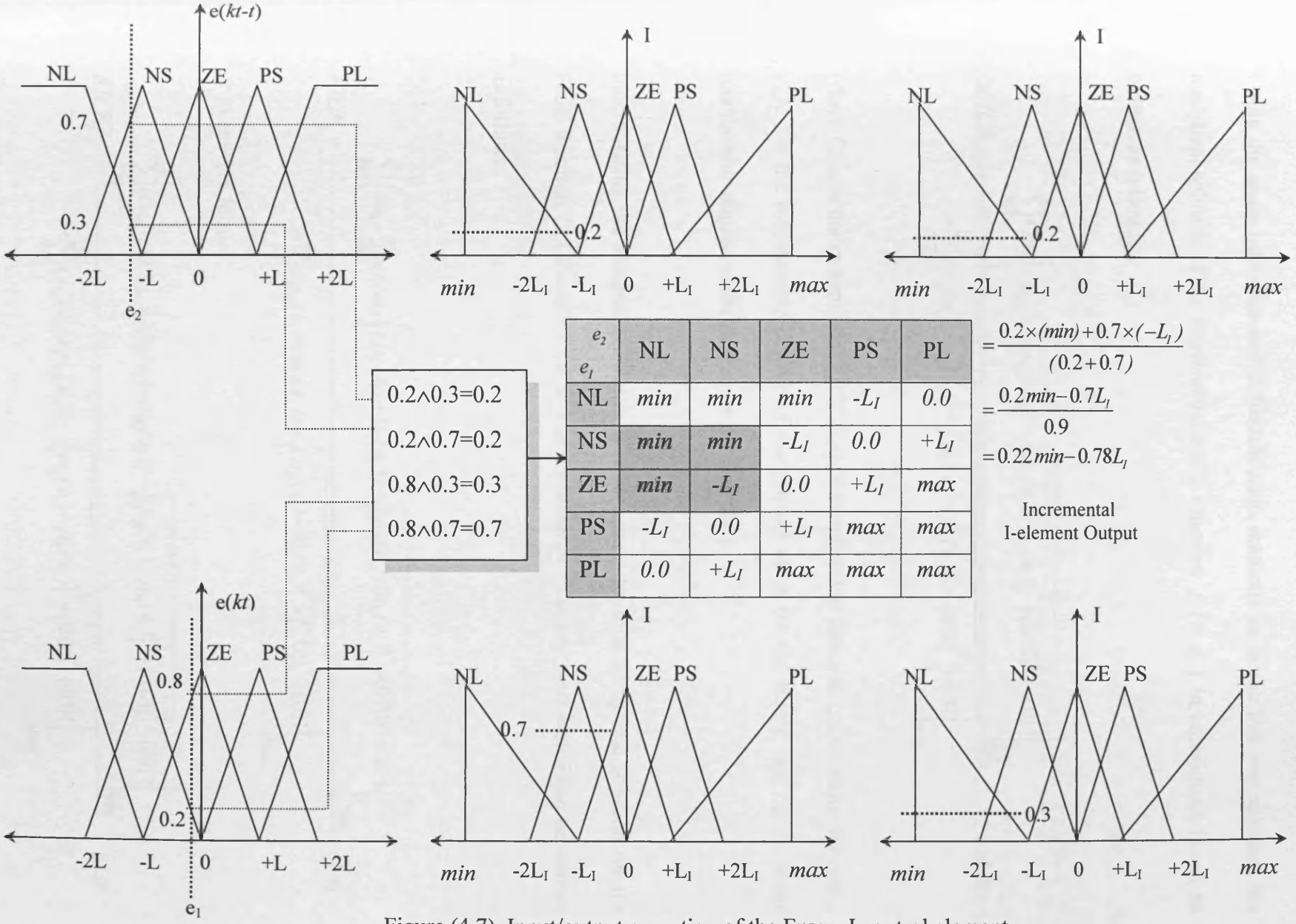
The conventional integral control action is composed of two parts. The first part is the integration initial condition or the controller's output history $U_I(kt\text{-}t)$ and the second part is the controller's incremental output $f_I(e_1, e_2) = \Delta U_I(kt)$. Therefore, the output of the integral element is composed of the same two parts. To implement the Fuzzy Integral Control Element (FICE), the same numbers of input/output partitions as in the previous two sections are employed. However, in this case, the distance between the centres of any two adjacent output membership functions is L_I. To implement the integration initial condition and the incremental part into one fuzzy controller element, the centres of the output universe membership functions are shifted after the k^{th} time step to a distance $U_I(kt-t) = \sum_{m=0}^{k-1} \Delta U_I(mt)$. The shifting process represents the memory of the FICE, so that the old information is stored within the FICE in the form of a dynamic (time changeable) output universe of discourse partition [Shankir, 2001]. Only the incremental part of the integral control element is of interest for the moment. The fuzzy rules of the operation of the incremental FICE according to the suggested partitions are generated heuristically as well based on the intuitive concept that the incremental part of the integral control action at a time step is directly proportional to the sum of the error variables at two successive time steps. For example, if the error variables e_1 and e_2 are expressed linguistically as positive and negative, the incremental part of the integral control action can be expressed linguistically as zero. Consequently, the Fuzzy Associative Memory (FAM) rules according to this concept of the incremental FICE can be written as shown in table (4.3).

e_1 \ e_2	NL	NS	ZE	PS	PL
NL	NL	NL	NL	NS	ZE
NS	NL	NL	NS	ZE	PS
ZE	NL	NS	ZE	PS	PL
PS	NS	ZE	PS	PL	PL
PL	ZE	PS	PL	PL	PL

Table (4.3). Integral incremental element FAM bank.

where [NL, NS, ZE, PS, PL] are the term sets of the normalized input variables e_1 and e_2 and the normalized output variable $\Delta U_I(KT)$.

To obtain the output of the incremental FICE, the same partitions, inference, and the same defuzzification method as in the last two sections are employed. Consequently, only four rules are triggered at a time. Therefore, the inference system generally produces four non-zero fuzzy outputs for the two crisp error inputs. The fuzzy output of a rule (output fuzzy sets after inference) is a fuzzy set with a trapezoid membership function whose height (h) equals the membership degree produced by the min operator of equation (4.2) during the fuzzy inference. Based on the input errors condition, employed inference method, and defuzzification method used, a numerical example for the fuzzy output and the (h) value for each of the four rules for the incremental FICE are shown in figure (4.7).



e_2 / e_1	NL	NS	ZE	PS	PL
NL	*min*	*min*	*min*	$-L_I$	*0.0*
NS	*min*	*min*	$-L_I$	*0.0*	$+L_I$
ZE	*min*	$-L_I$	*0.0*	$+L_I$	*max*
PS	$-L_I$	*0.0*	$+L_I$	*max*	*max*
PL	*0.0*	$+L_I$	*max*	*max*	*max*

$$= \frac{0.2\times(min)+0.7\times(-L_I)}{(0.2+0.7)}$$

$$= \frac{0.2min-0.7L_I}{0.9}$$

$$= 0.22min-0.78L_I$$

Incremental
I-element Output

Figure (4.7). Input/output operation of the Fuzzy-I control element.

Using the same inference and defuzzification methods as in the last two sections, the analytical solution of the incremental FICE function $f_I(e_1, e_2)$ in equation (4.1) can be written as follows:

$$\Delta FICE_{output} = \frac{\sum_{i=1}^{4}\left[Coi_R\left((\mu_R(e_1)+\mu_R(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right)\right]_{Rule_i}}{2*\sum_{i=1}^{4}\left[(\mu_R(e_1)+\mu_R(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right]_{Rule_i}} \quad (4.7)$$

where Coi_{Ri} is the incremental FICE output membership function centre value for rule i, $\mu_{Ri}(e_1)$ is the membership degree of the present error for the rule i, and $\mu_{Ri}(e_2)$ is the membership degree of the past error for the rule i.

Incorporating the integr$\sum_{i=1}^{4}\left[(Coi_R+(FICE)_{k-1})\left((\mu_R(e_1)+\mu_R(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right.\right.$

FICE function $U_I(kt-t)+f_I(e_1,e_2)$ in equation (4.1) at any time step k can be written as follows:

$$FICE_k = \frac{\sum_{i=1}^{4}\left[\left(Coi_R+\sum_{m=0}^{k-1}\Delta U_I(mt)\right)\left((\mu_R(e_1)+\mu_R(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01+0.01\right)\right]_{Rule_i}}{2*\sum_{i=1}^{4}\left[(\mu_R(e_1)+\mu_R(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right]_{Rule_i}} \quad (4.8)$$

or in another format:

$$FICE_k = \frac{\sum_{i=1}^{4}\left[(Coi_R+(FICE)_{k-1})\left((\mu_R(e_1)+\mu_R(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right)\right]_{Rule_i}}{2*\sum_{i=1}^{4}\left[(\mu_R(e_1)+\mu_R(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right]_{Rule_i}} \quad (4.9)$$

This finally leads to:

$$FICE_k = FICE_{k-1} + \frac{\sum_{i=1}^{4}\left[(Coi_{Ri})\left((\mu_R(e_1)+\mu_{Ri}(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right)\right]_{Rule_i}}{2*\sum_{i=1}^{4}\left[(\mu_R(e_1)+\mu_{Ri}(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right]_{Rule_i}} \quad (4.10)$$

From this last equation it can be seen that the integration using the shifting process in Fuzzy-PID as proposed by [Shankir, 2001] is similar from the analytical point of view to integration using a delay loop in the case of conventional discrete integral controller. A feedback-error learning scheme is utilized in the suggested robot control system. As mentioned earlier, this scheme ensures that online training will stop only when the feedback error is zero. This behavior resembles the integration action in a classical integral controller which will be achieved in this case by shifting the output membership functions centres of the proposed forward path network (*Dynafuzznn*), so that only the incremental part of the integral control element (equation (4.7)) is used for training of the neuro-fuzzy controller to ensure the learning signal reduces to zero automatically when the error reduces to zero and to guarantee that the control signal converge to that of the forward path controller only. Consequently, the rule base of the three incremental FCEs (P, D and I) can be combined together to form one rule base for the total fuzzy-PID-like incremental servo controller output as follows:

$$U_{PID} = \frac{k_u * \sum_{i=1}^{4}\left[(k_p Cop_{Ri} + k_d Cod_{Ri} + k_i Coi_{Ri})\left((\mu_R(e_1)+\mu_R(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right)\right]_{Rule_i}}{2*\sum_{i=1}^{4}\left[(\mu_R(e_1)+\mu_R(e_2))-\sqrt{(\mu_R(e_1)-\mu_R(e_2))^2+(0.01)^2}+0.01\right]_{Rule_i}} \quad (4.11)$$

where k_p, k_d, and k_i are the scaling factors corresponding to the three control actions, while k_u is an overall gain for the servo controller. A total of 25-rules with two inputs

and three outputs can represent the combined fuzzy-PID-like incremental controller as shown in table (4.4).

e_1	e_2	P-element	D-element	I-element
NL	NL	NL	ZE	NL
NS	NL	NS	PS	NL
ZE	NL	ZE	PL	NL
PS	NL	PS	PL	NS
PL	NL	PL	PL	ZE
NL	NS	NL	NS	NL
NS	NS	NS	ZE	NL
ZE	NS	ZE	PS	NS
PS	NS	PS	PL	ZE
PL	NS	PL	PL	PS
NL	ZE	NL	NL	NL
NS	ZE	NS	NS	NS
ZE	ZE	ZE	ZE	ZE
PS	ZE	PS	PS	PS
PL	ZE	PL	PL	PL
NL	PS	NL	NL	NS
NS	PS	NS	NL	ZE
ZE	PS	ZE	NS	PS
PS	PS	PS	ZE	PL
PL	PS	PL	PS	PL
NL	PL	NL	NL	ZE
NS	PL	NS	NL	PS
ZE	PL	ZE	NL	PL
PS	PL	PS	NS	PL
PL	PL	PL	ZE	PL

Table (4.4). Fuzzy servo controller combined FAM bank.

Finally, the total servo-controller output can be represented in the form:

$$U_{PID} = k_U\left[k_P k_{NP} e_1 + k_D k_{ND}(e_1 - e_2) + k_I k_{NI}(e_1 + e_2)\right] \tag{4.12}$$

where k_{NP} , k_{ND} , and k_{NI} are the equivalent nonlinear gains that can be defined according to the input condition (value of current and previous error) and values of the three partition values L_P, L_D, and L_I. The nonlinear gains provide the general nonlinear policy for the controller and the learning signal.

4.2. Feedback-Error Learning Scheme

Following the selection of the feedback controller, the total control torque acting on the robot manipulator is the sum of the feedforward torque and the feedback torque.

$$T_{tot}^i = T_{FB}^i + T_{FF}^i \tag{4.12}$$

Kawato and his group [Kawato et. al., 1988] proposed a novel architecture for control called the feedback error learning (FEL) method, which combines learning and control efficiently. It is essentially an adaptive two-degree-of-freedom (TDOF) control system with an inverse model in the feedforward path. In some sense, the method is closely related to the adaptive internal model control mentioned in Chapter (2). The novelty of the FEL method lies in its use of feedback error as a teaching signal for learning the inverse model, which is essentially new in control literature. The objective of control is to minimize the error between the command signal and the plant output. If the learning

part of the architecture is disregarded, then, if the inverse model of the plant exists and is stable, the tracking will be perfect. In [Miyamura and Kimura, 2002; Terashita and Kimura, 2001] a stability proof for the FEL algorithm for linear time-invariant systems is presented. Another important point, which was not investigated in Kawato's work, is the problem concerning non-invertibility of the plant, however this aspect was also proved to be stable in [Kimura and Miyamura, 2002].

The neuro-fuzzy forward path controller parameters are tuned online using the feedback controller response as the error signal. This control structure provides an internal teacher so that the control scheme works in an unsupervised manner as there is no external teacher in this case. The adjustment of the neuro-fuzzy network parameters during the control by feedback-error learning is more convenient than other learning structures. The network adjustable free parameters were selected to be centres ($m_{ij}s$) of the output membership functions of the term nodes in layer four as well as the link weights at layers two and six as mentioned before. Despite the effectiveness of the back-propagation, its speed of convergence can be painfully slow in online learning. The reasons for this have been discussed in details in [Jacobs, 1988]. Jacobs also presented an overview of heuristics that can be used to accelerate the convergence of the algorithm, suggesting that each weight should be given its own learning rate, and that learning rate be allowed to change over time during the learning process. He also suggested how the learning rate should be adjusted heuristically. [Fukuda et. al., 1990] proposed a variable learning method for robotic manipulators Neural Network controllers called "Fuzzy Turbo", which is based on fuzzy set theory to avoid stagnation during learning. In this method, a linear PID feedback controller is used

along with the Neural Network feedforward controller. In order to accelerate learning, they experimentally proposed a table relating the value of the learning rate to the fuzzy representation of the output error and the sum of weight changes at learning instant. In [Arabshahi et. al., 1992], fuzzy control of the learning rate η is suggested. The central idea behind fuzzy control of the back-propagation algorithm is the implementation of the heuristics used for faster convergence in terms of fuzzy IF... THEN rules. This is done by considering the error and the change in error to be fuzzy variables taking on the feedback controller output at each learning iteration n. A fuzzy variable is also considered for the change in learning rate $\Delta\eta_n$. The resulting rule set is suggested on the bases of changing the learning rate in the way to quickly drive the feedback controller output to minimum or zero in relation to the current output and the change in output. However, there is still no general guidance for the proper selection of the learning rate and one can say it is case dependent policy. In this study, the fuzzy PID-like feedback controller along with a fixed learning rate provides the general nonlinear policy of the controller and learning signal as well. The back-propagation learning algorithm explained in Chapter (3) is a gradient descent search in the space of the Neural Network weights and aims to minimize energy function which is normally defined as the sum of the squared errors, where each error is defined as the difference between target values and the actual values obtained during each iteration of the algorithm. Weight changes are performed at the k^{th} iteration according to:

$$E_{tot}(w_k) = \frac{1}{2}\sum_{i=1}^{l}\left[T_{tot}^{i} - T_{FF}^{i}\right]^2 = \frac{1}{2}\sum_{i=1}^{l}\left[T_{FB}^{i}\right]^2 \qquad (4.13)$$

$$\Delta w_k = -\eta \frac{\partial E_{tot}(w_k)}{\partial w_k} = (\eta T^i_{FB}) * \frac{\partial T^i_{FF}(w_k)}{\partial w_k} \tag{4.14}$$

$$\eta T^i_{FB} = \eta k_U \left[k_P k_{NP} e^i_1 + k_D k_{ND} \left(e^i_1 - e^i_2 \right) + k_I k_{NI} (e^i_1 + e^i_2) \right] \tag{4.15}$$

where $E_{tot}(w_k)$ is the total error at the k^{th} iteration, T^i_{tot} is the total acting torque at robot link i, T^i_{FF} is the feedforward controller torque at robot link i, T^i_{FB} is the feedback controller torque at robot link i, w_k is the vector of weight values after the k^{th} iteration, Δw_k is the change in these weights, l is the total link numbers of the robot, e^i_1 & e^i_2 are the current and past position errors at link i, and η is the learning rate. The chain rule is then applied to calculate the network output partial derivative with respect to the variables weights at each layer as explained in Chapter (3).

4.3. Comparison Study of the Results

The proposed control system is tested by applying it to control the first three links of a Puma 560® industrial robot. The controller algorithm was programmed in C++ and linked to the *"Pro/Mechanica®"* virtual model of the Puma 560® industrial robot as a subroutine as explained in Appendix (B). Figure (4.8) shows the user interface in the *"Pro/Mechanica®"* environment for the neuro-fuzzy controller developed. For comparison purposes, a conventional PID controller, tuned using the Ziegler-Nichols tuning rule [Ziegler and Nichols, 1942] and then fine tuned by trial-error, is also used to control the robot over the required pre-planned joint-trajectories while carrying a fixed payload of 7.0 kg. Figures (4.9) to (4.12) represent the results for the suggested neuro-

fuzzy controller, while figures (4.13) to (4.16) represent the results for the conventional PID controller. It can be observed from the results that the proposed neuro-fuzzy controller outperforms the conventional PID controller, both in terms of joint displacement and velocity tracking, as a result of the embedded knowledge of system dynamics in the neuro-fuzzy feedforward controller component.

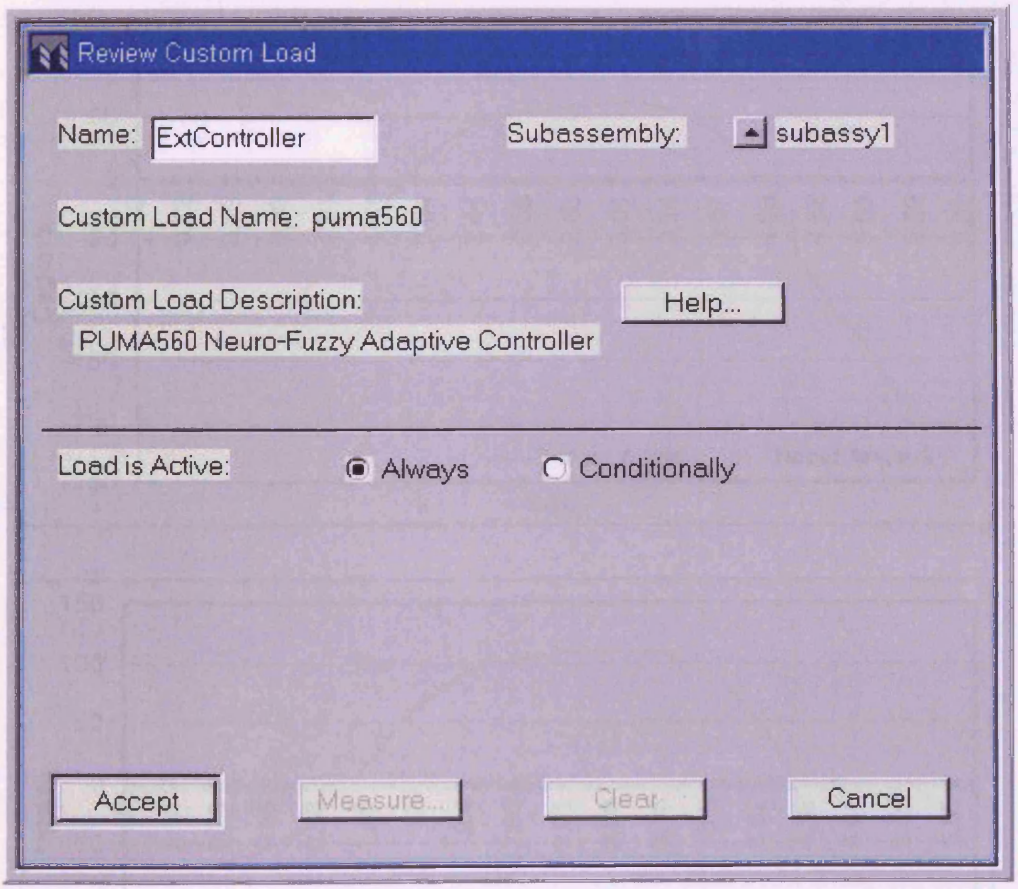


Figure (4.8). "Pro/Mechanica" user interface for neuro-fuzzy controller.

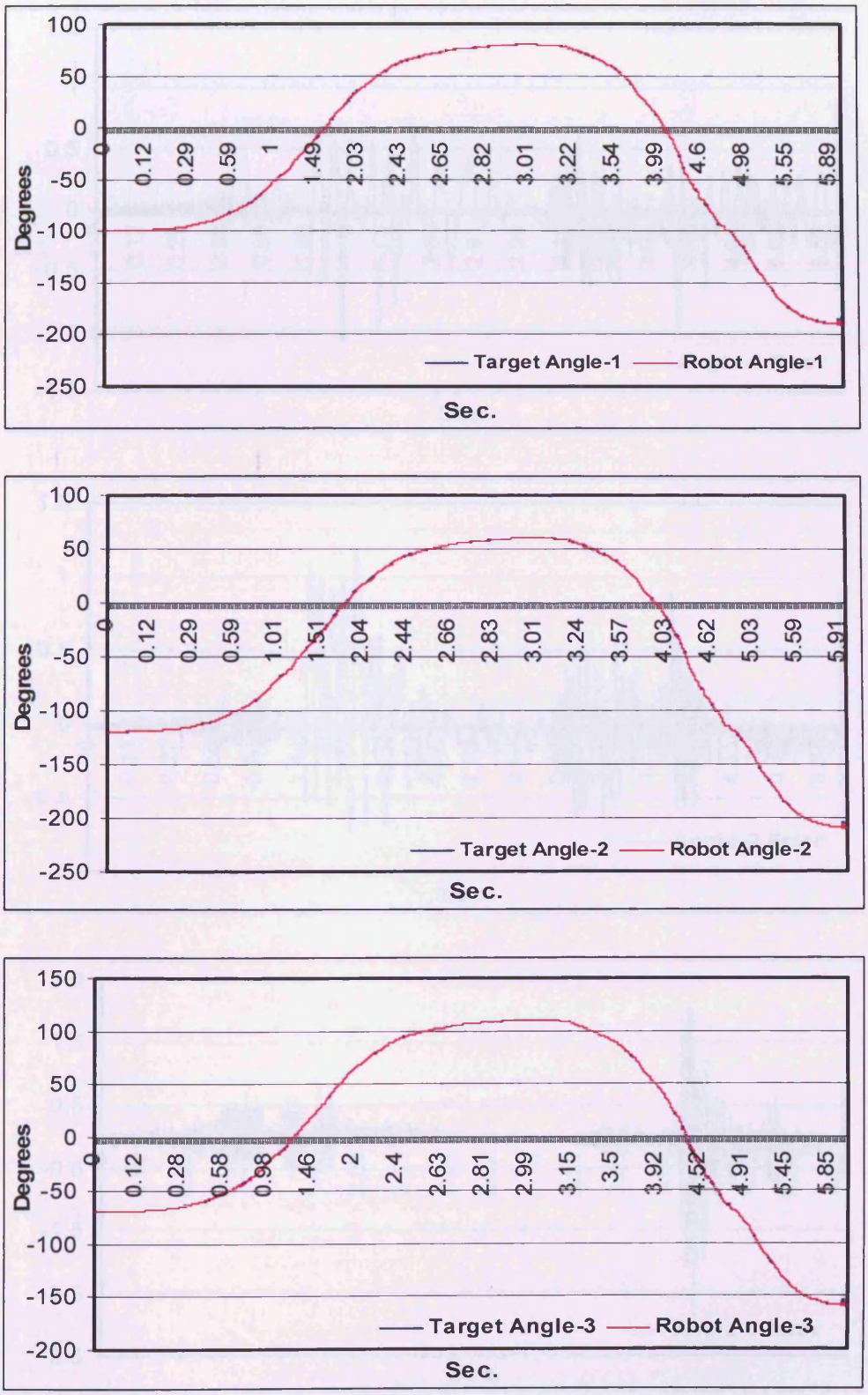


Figure (4.9). Neuro-fuzzy controller position trajectories tracking.

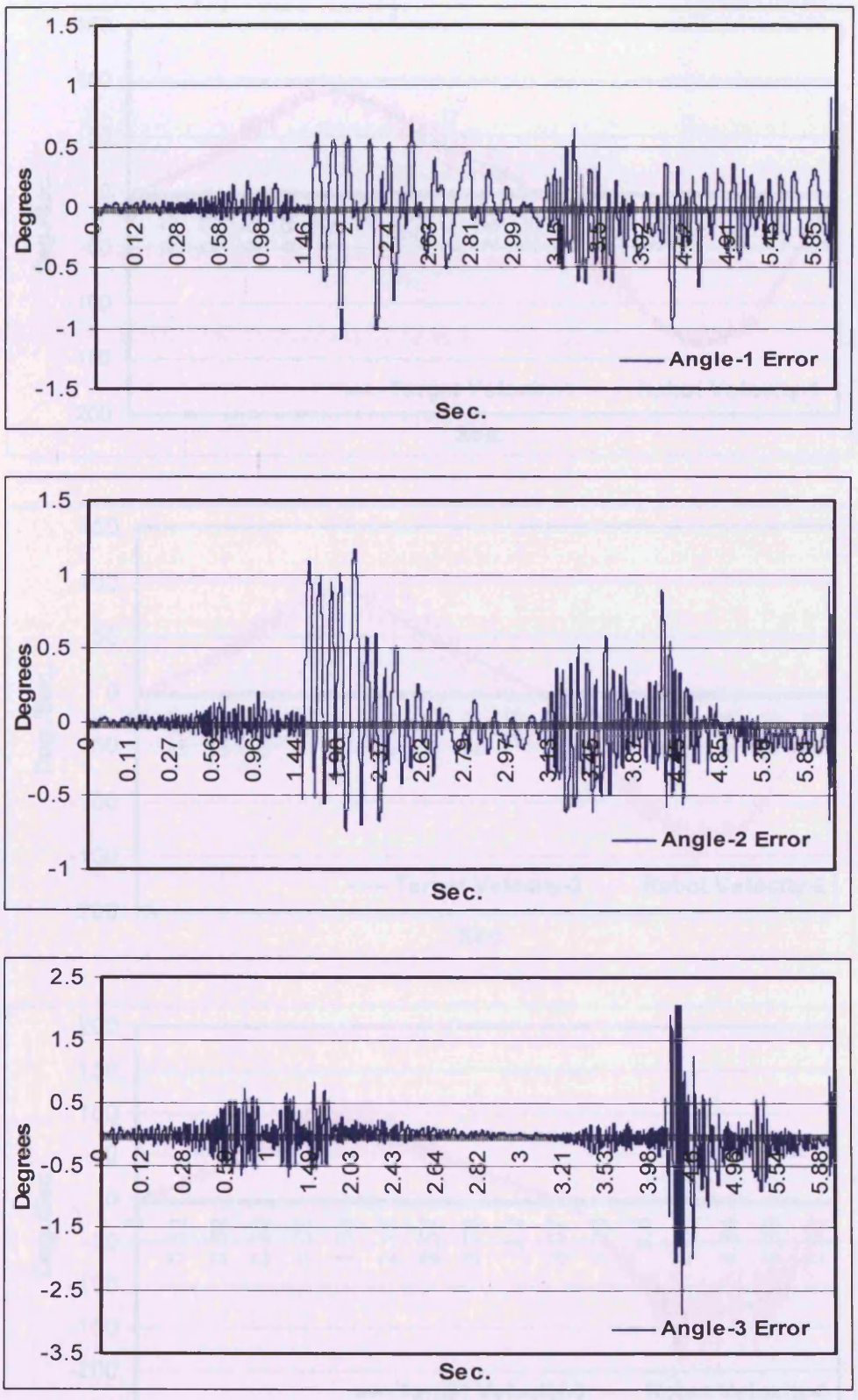
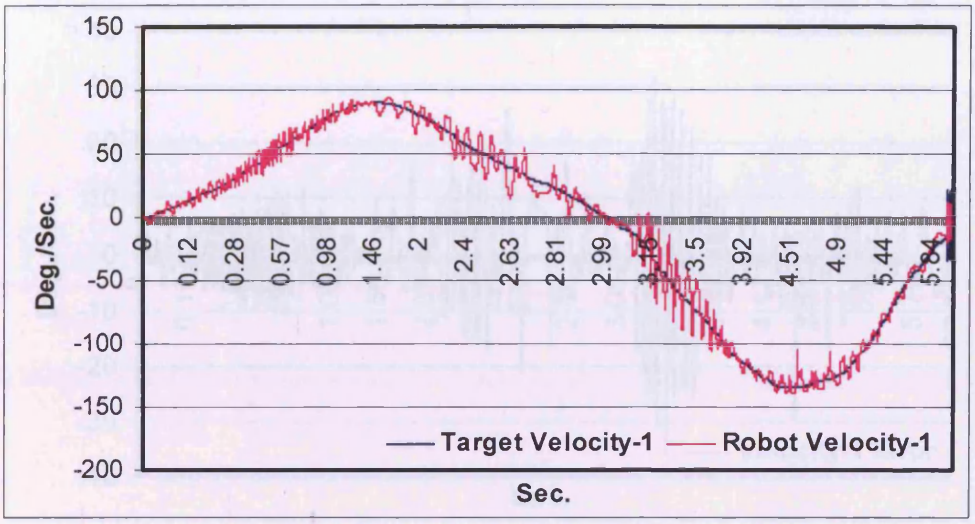
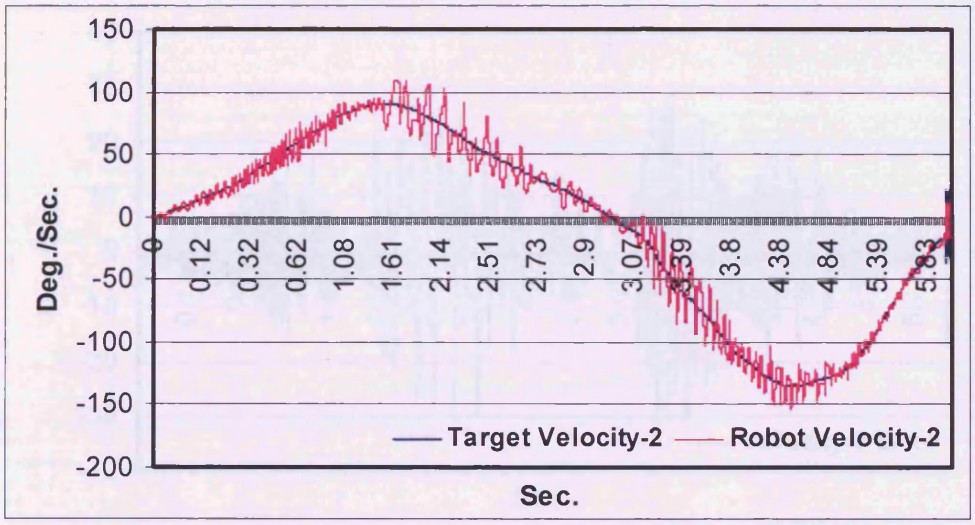


Figure (4.10). Neuro-fuzzy controller position tracking errors.





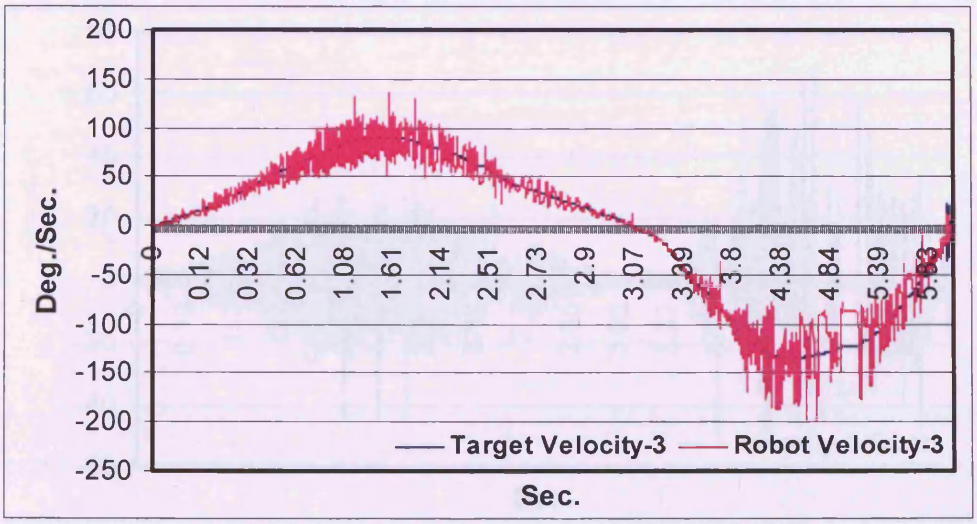
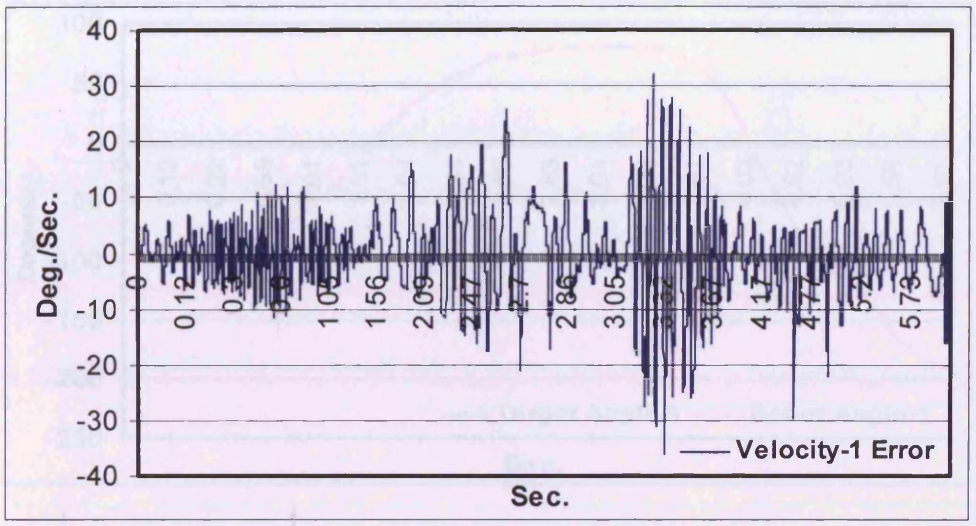
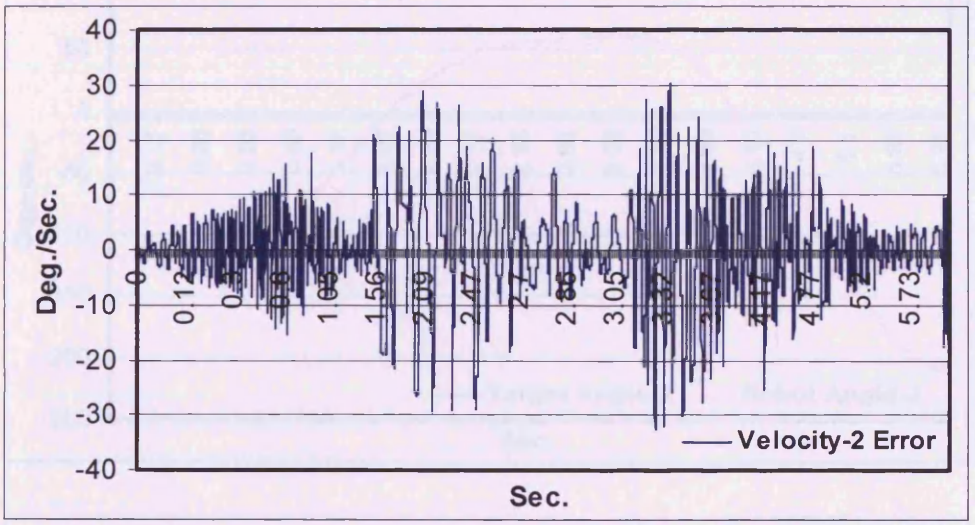


Figure (4.11). Neuro-fuzzy controller velocity trajectories tracking.





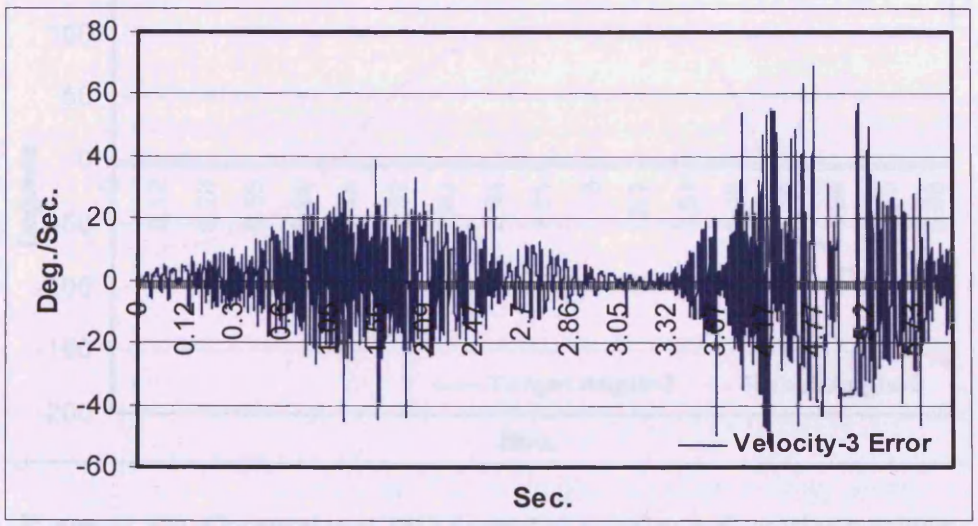


Figure (4.12). Neuro-fuzzy controller velocity tracking errors.

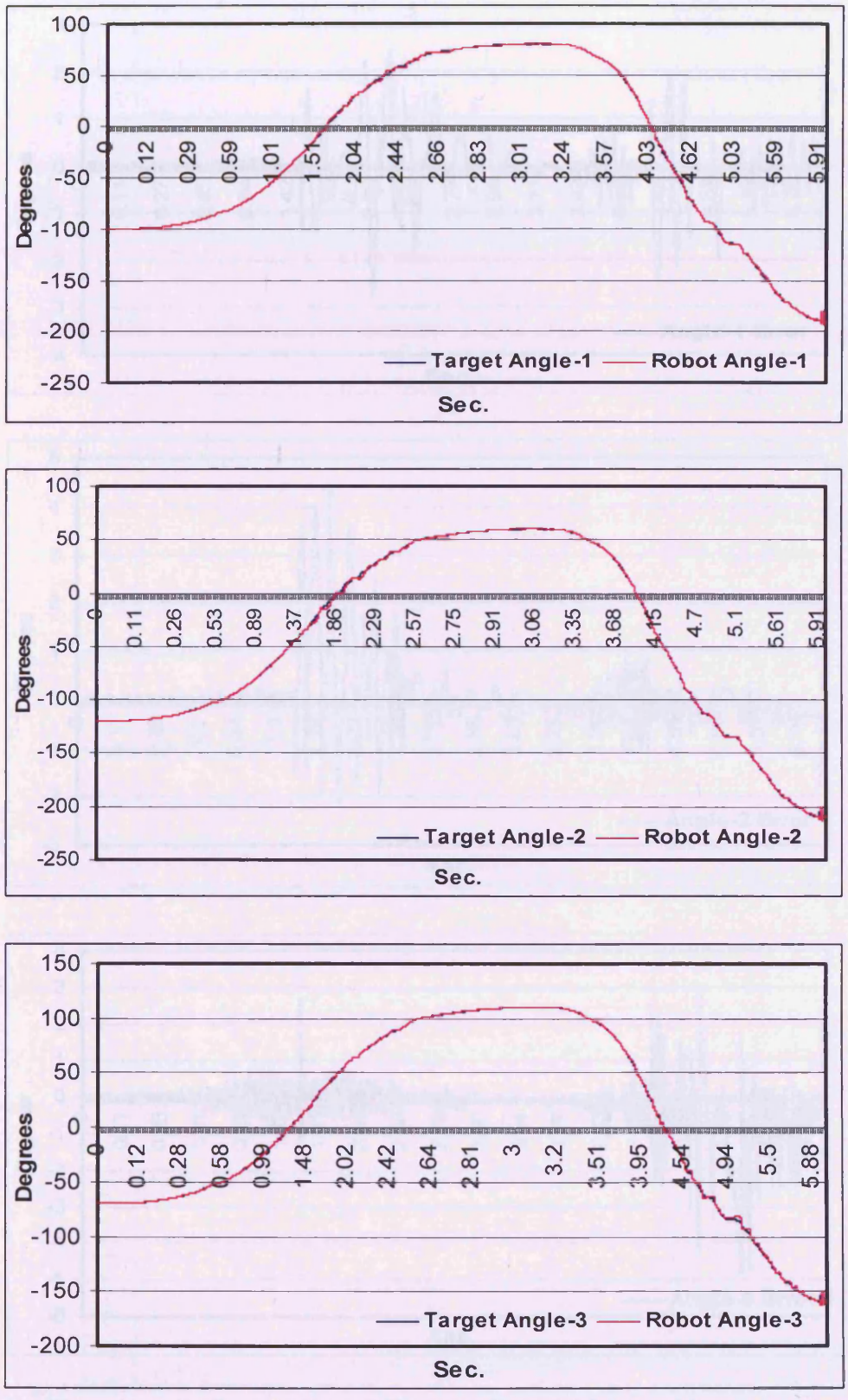


Figure (4.13). Conventional-PID controller position trajectories tracking.

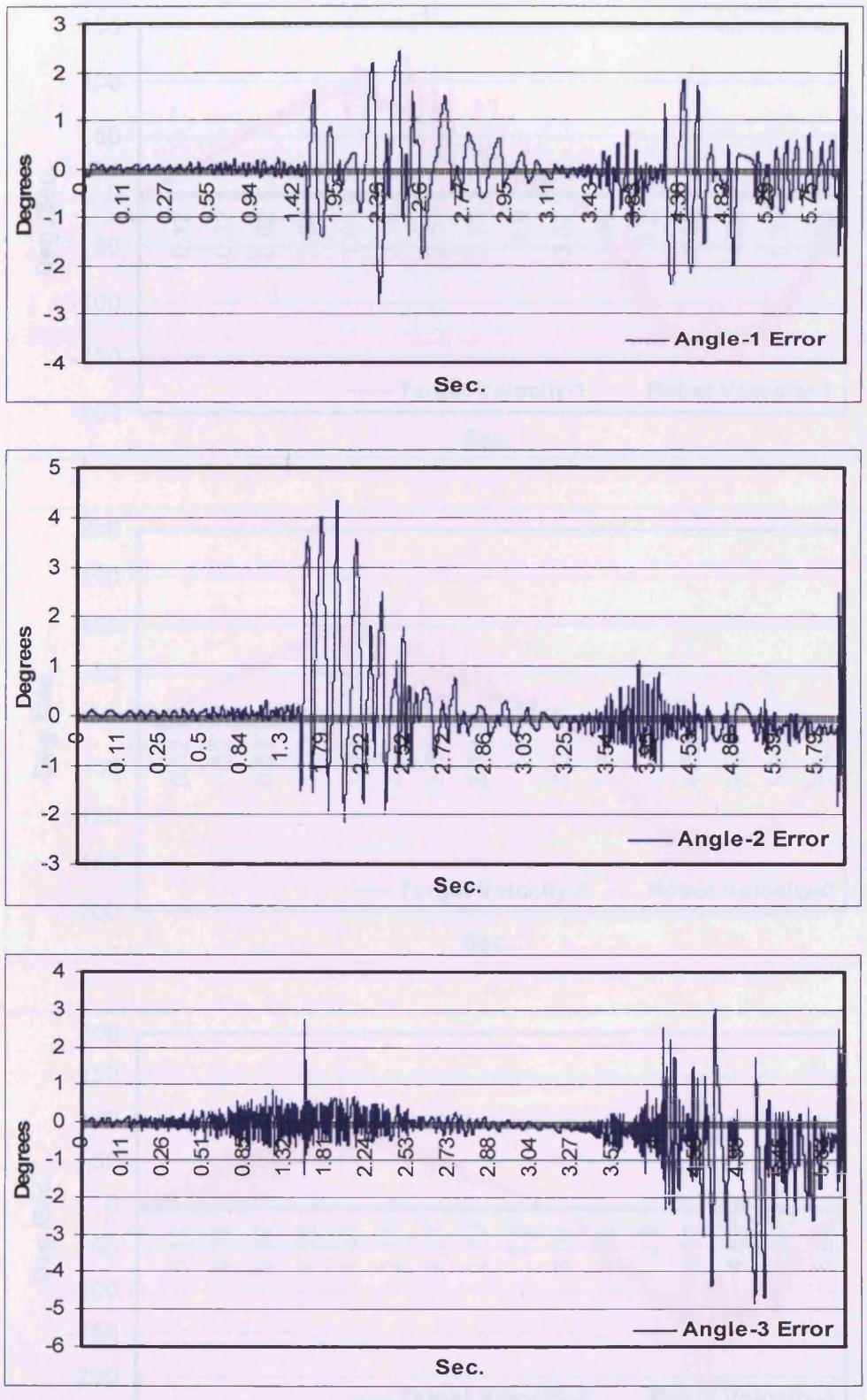
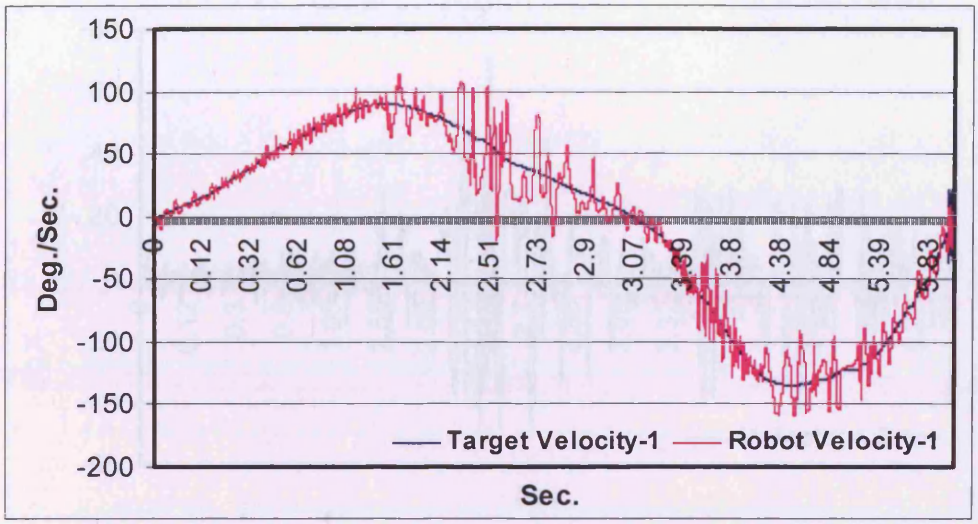
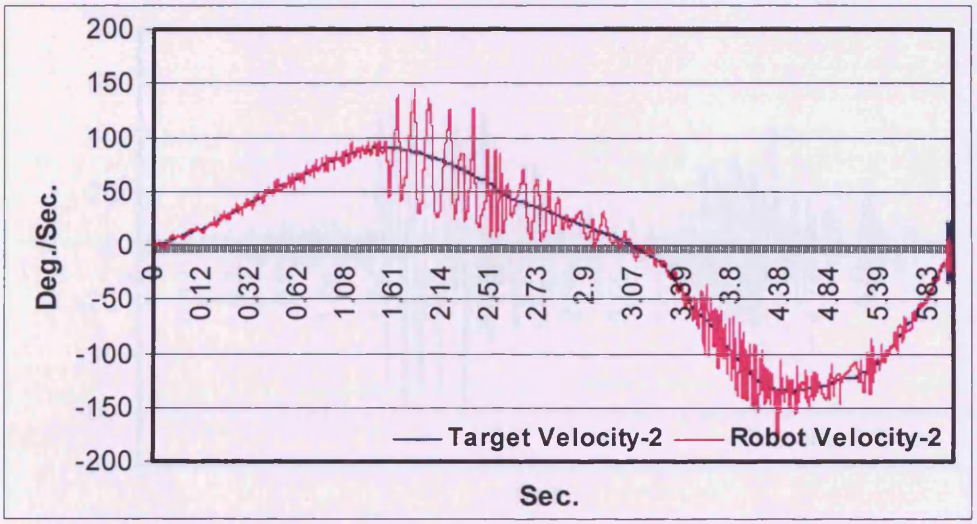


Figure (4.14). Conventional-PID controller position tracking errors.





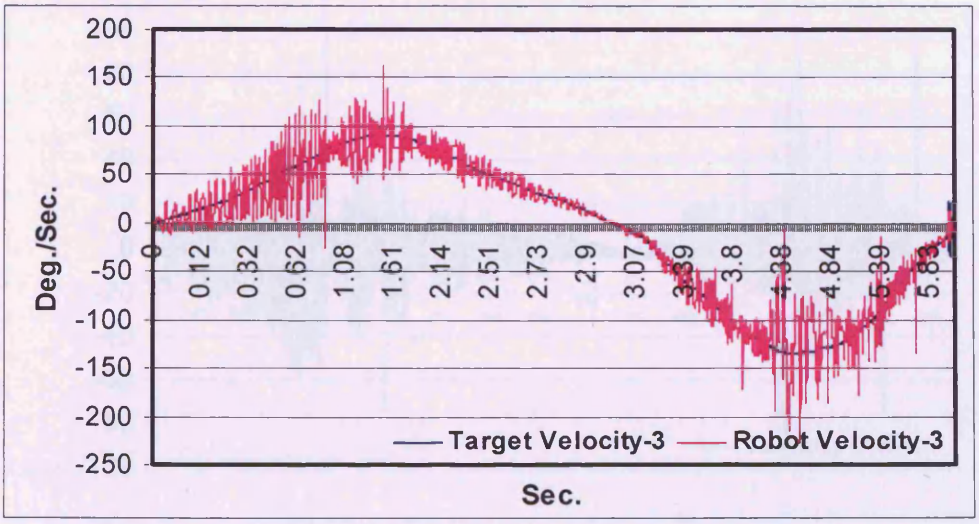
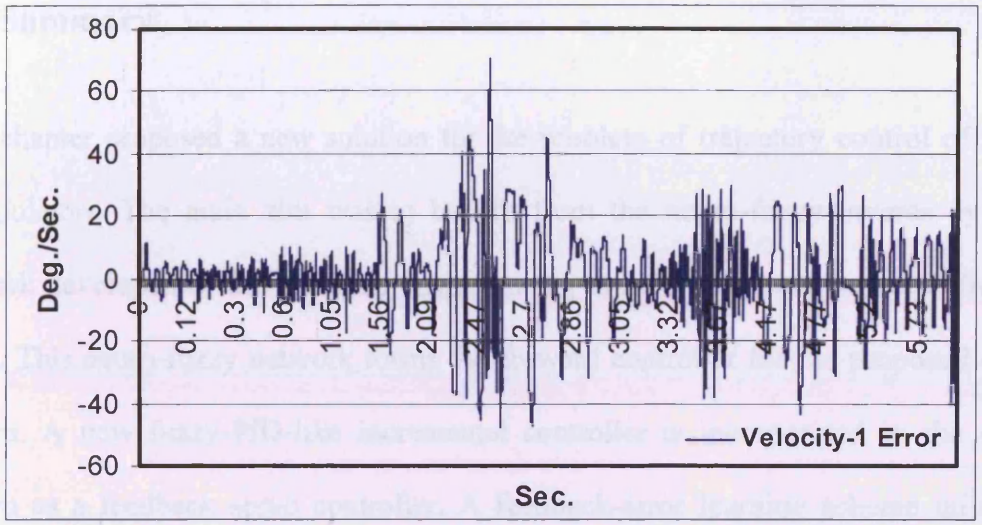
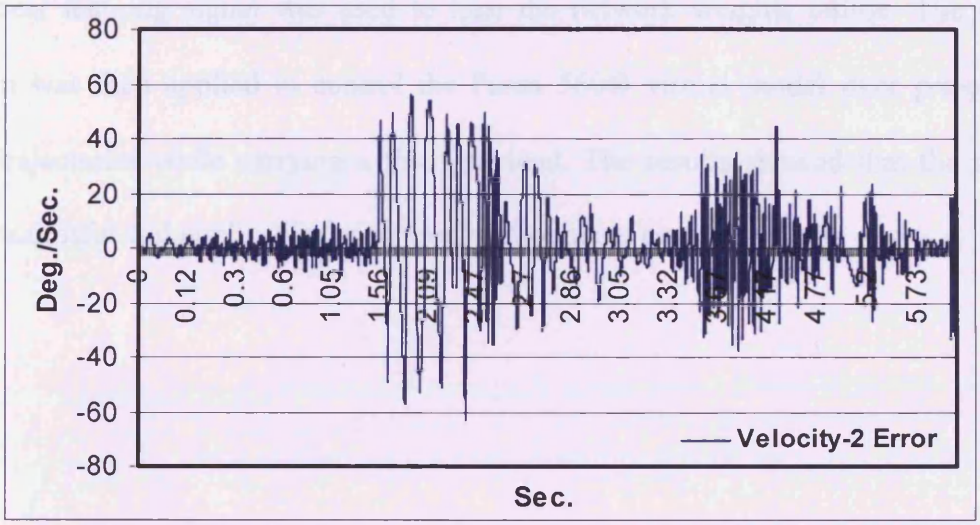


Figure (4.15). Conventional-PID controller velocity trajectories tracking.





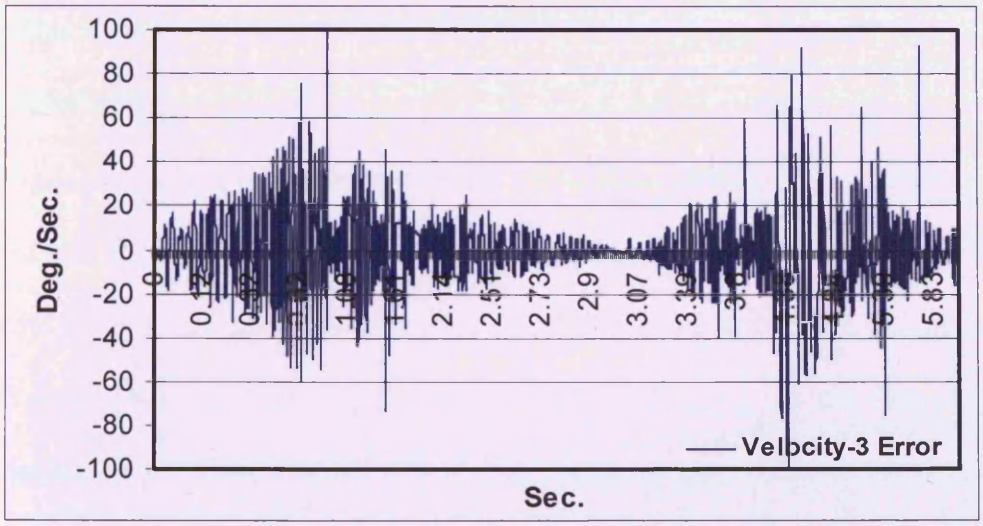


Figure (4.16). Conventional-PID controller velocity tracking errors.

4.4. Summary

This chapter proposed a new solution for the problem of trajectory control of robotic manipulators. The main aim was to benefit from the neuro-fuzzy inverse dynamics network developed in the previous chapter using input/output data collected from the robot. This neuro-fuzzy network forms the forward controller for the proposed control system. A new fuzzy-PID-like incremental controller is incorporated in the control system as a feedback servo controller. A feedback-error learning scheme utilizing a nonlinear learning signal was used to tune the network weights online. The control system was then applied to control the Puma 560® virtual model over pre-planned joint-trajectories while carrying a fixed payload. The results showed that the method was successful and applicable for robotic manipulators control.

CHAPTER 5

Neuro-Fuzzy Cartesian Control of Robotic Manipulators

Most recent developed manipulator control schemes require as inputs the desired position, velocity, and in some cases, acceleration of each joint of the manipulator. However, it is most likely for the control system to specify the desired trajectory in Cartesian coordinates as the task description is normally expressed in terms of a sequence of end-effector coordinates in Cartesian space. Generally, this information is transformed through inverse kinematics to a series of angular positions in the joint space, while the end-effector control is then accomplished indirectly by controlling the joint angles. Although end-effector control is the ultimate goal of any robot control system, direct control of the end-effector motion in Cartesian space has not attracted much attention. The transformation process from joint coordinates to Cartesian coordinates is a vector-valued non-linear function which can be obtained in a straightforward way from the geometry of the manipulator and is known as the forward kinematics method. However, the reverse process, the inverse kinematics may not be unique and is known not to exist in closed form for certain manipulators. To avoid the need to calculate the inverse kinematics, two techniques are used for Cartesian control of manipulators, the first technique transfer the sensed position of the manipulator immediately by means of forward kinematics equations into a Cartesian position of the end-effector. This Cartesian position is then compared to the desired one in order to form the error in Cartesian space. An inverse Jacobean matrix has to be calculated to map the error in the Cartesian space to error in joint space. Finally, this latter is

multiplied by a gain to compute the torques required to reduce the error. The second technique multiplies the Cartesian error vector by a gain to compute a Cartesian force vector. This can be thought of as a Cartesian force which, if applied to the end effector of the robot, would push the end-effector in a direction tending to reduce the Cartesian error. This Cartesian force vector is then mapped through the Jacobean transpose in order to compute the equivalent joint torques which would tend to reduce the Cartesian error. In all these cases, even though no direct calculation for inverse kinematics involved in the control loop, there still a need to calculate the Jacobean matrix or its transpose, which is not an easy task. Fuzzy systems and Neural Networks have been used in literature to approximate the inverse kinematics calculation for robot manipulators [Sang-Bae, 1997; Martinez et. al., 1996; and Kim et. al., 1993]. Most of these methods still require pre-calculation of the manipulator Jacobean matrix, resulting in additional computational burden. Also, these techniques are referenced as Cartesian control systems because the controller is implemented over the Cartesian error and cannot be applied to an existing joint-space control scheme found in all industrial manipulators. The operation of transforming the position component of a trajectory in Cartesian coordinates into a trajectory in joint coordinates which will be then used as inputs to an existing joint-space control system is called a *command generator* [Vaccaro and Hill, 1988], because it generates commands to the existing manipulator joint-space control system to move along the demanded Cartesian trajectory. [Jung and Hsia, 1995] proposed a new Neural Network control technique for non-model based PD control of robot manipulators. The main feature of this technique is that compensation of robot uncertainties is performed outside the control loop by modifying the desired input trajectory. By introducing the Neural Network outside the control loop, the

control algorithm was implemented at the command trajectory planning level external to an existing controller. Although the idea seems promising for Cartesian control of robot manipulators, it was implemented in [Jung and Hisa, 1996] for the joint-based control technique only.

The remainder of this chapter is organized as follows. Section 5.1 reviews the conventional internal model control structure. Section 5.2 presents a modified neuro-fuzzy internal model Cartesian control for robot manipulators. Section 5.3 presents the tuning method used to adapt the controller parameters. Section 5.4 introduces a robustness analysis for the proposed controller. Section 5.5 presents the simulation results obtained when the proposed controller is used to control the Puma 560® industrial robot virtual model developed in chapter (3) to follow both joint and Cartesian trajectories. Section 5.6 presents the application to upper-limb rehabilitation. Section 5.7 presents a summary for the chapter.

5.1. Internal Model Control

Both inverse control and internal model control have been recently used in non-linear control systems. Many of the control methods using neural/neuro-fuzzy networks are based on the principle of inverse control. Neural networks have been also used in non-linear internal model control lately. By studying the control principles of these schemes, it can be seen that the strengths of internal model control may compensate the disadvantages of inverse control. The principle of inverse control is the dynamics cancellation of the controlled plant. This is a special case of model reference

feedforward control in which the controller is cascaded with the plant. The block combining the controller and the plant is called the reference model. When this reference model is chosen to have no dynamics, the task of the controller is to achieve total cancellation of the dynamics of the plant. When they are combined, the two blocks disappear and reduce to an identity transfer function, so that the output from the system is exactly the input to the controller. This is the concept of perfect control. When there is dynamics in that model, the control system can also be viewed as a detuned inverse control system. When implementing a neural network in an inverse control scheme, usually the function approximation ability of the neural network is used as the controller to perform inverse mapping. Let U denote the input to the process and Y denote the process output. The task of the neural network is to produce U given Y. However, using Y alone as an input is not sufficient to generate U correctly. The most common remedy is to add other inputs, for example state feedback signals as explained in chapter (4). An inverse control scheme has a few serious problems. It is not possible to obtain an inverse model in some cases and the inverse controller is not robust. A control scheme is called robust when it remains stable under model uncertainty or inaccuracy.

The Internal Model Control (IMC) system was first introduced by [Garcia and Morari, 1985]. They designed an overall structure using a linear single-input single-output (SISO) discrete time process model. Then, they extended the SISO systems to multi-input multi-output (MIMO) systems. This control structure presents a model predictive process control algorithm. Actually, the name IMC came from the fact that the process model is explicitly an internal part of the controller. The IMC provides a

straightforward yet effective framework for analysis of control system performance, especially with respect to stability and robustness issues. The design of IMC is also simpler and more transparent than that of traditional control methods even when the goal is just a conventional PID feedback controller. IMC is composed of an inverse model connected in series with the plant and a forward model connected in parallel with the plant, this structure allows the error feedback to reflect the effect of disturbance and plant mismodelling resulting in a robust control loop. IMC is characterized by its fast smooth response to set-point changes and robustness against parametric changes. Also, if the match between the plant and the plant model is perfect, perfect control is achieved. However, perfect matching between plant and plant model is difficult to obtain and may lead to sensitivity problems. Normally, a pre-filter is introduced before the controller in the control loop forward path to reduce the gain of the feedback system in order to move away from the perfect controller and to introduce desirable robustness to the closed-loop system. Detailed study for IMC robustness and stability issues can be found in [Morari and Zafiriou, 1989]. The IMC structure is shown in figure (5.1). This structure consists of the plant Φ_p to be controlled, the model of the plant Φ_m, the inverse model of the plant Φ_c which represents the controller, and R, U, Y, and D the vectors of the reference inputs to the system, the control inputs to the plant, the system outputs and the external disturbances respectively.

For simplicity, all these quantities are assumed to be of dimension *n*. In general the IMC requires that both the plant Φ_p and the controller Φ_c be stable. In the case of an open-loop unstable plant, pre-stabilization for the plant by a conventional feedback loop is necessary before the standard IMC can be applied [Garcia and Morari, 1985].

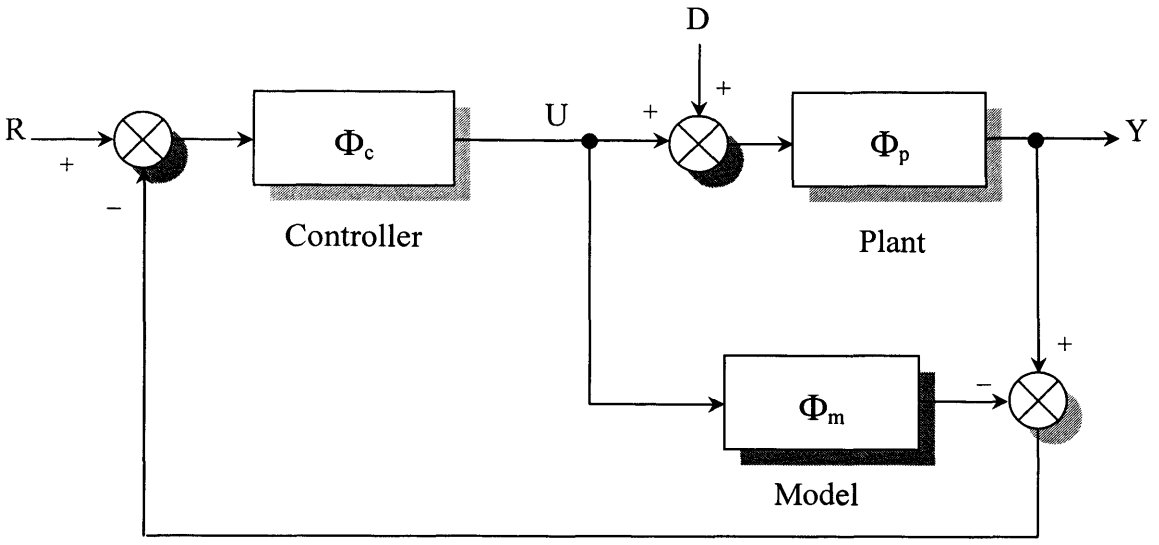


Figure (5.1). Standard internal model control structure.

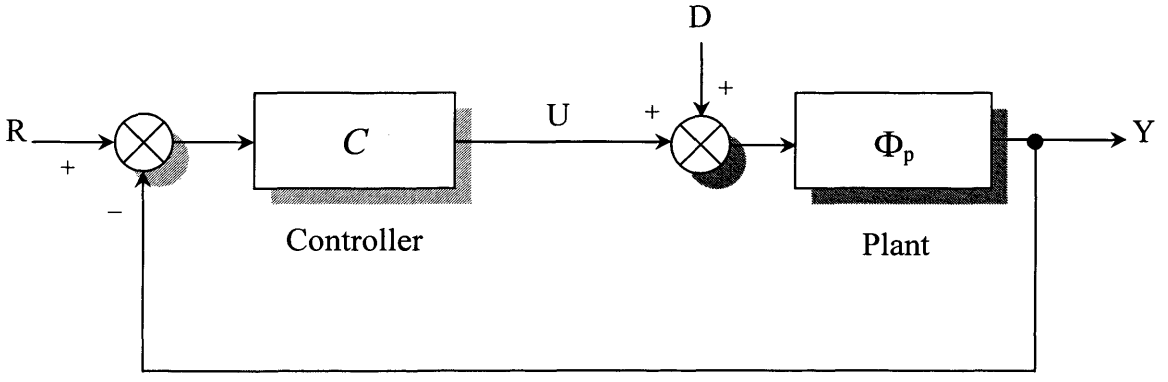


Figure (5.2). Classical feedback control structure.

The particular structure of IMC shown in Figure (5.1) can be proved to be equivalent to that of the conventional linear feedback control structure illustrated in Figure (5.2) regarding the following transformation:

$$C = (I - \Phi_c \Phi_m)^{-1} \Phi_c \qquad (5.1)$$

$$\Phi_c = (I + C\Phi_m)^{-1} C \tag{5.2}$$

The equivalence of these two control structures implies that whatever is possible employing a conventional linear control structure can be accomplished with the IMC structure and vice versa. However, it is more straightforward to design Φ_c instead of designing C. Furthermore, the IMC structure allows designers to include robustness as a design objective in a very intuitive manner [Garcia and Morari, 1985]. These can be illustrated by examining the IMC properties.

From the block diagram shown in figure (5.1), the input output (from R to Y) transfer function Φ^R, and the disturbance transfer function (from D to Y) Φ^D of the IMC system can be derived as:

$$\Phi^R = \frac{\Phi_p \Phi_c}{[I + \Phi_p \Phi_c - \Phi_m \Phi_c]} \tag{5.3}$$

$$\Phi^D = \frac{\Phi_p \left(I - \Phi_m \Phi_c\right)}{[I - \Phi_m \Phi_c + \Phi_p \Phi_c]} \tag{5.4}$$

Equations (5.3) and (5.4) can be rewritten as:

$$\Phi^R = [I + (\Phi_c^{-1} - \Phi_m)\Phi_p^{-1}]^{-1} \tag{5.5}$$

$$\Phi^{D} = [\Phi_{p}^{-1} + (\Phi_{c}^{-1} - \Phi_{m})^{-1}]^{-1} \tag{5.6}$$

The most important property of Equations (5.3) and (5.4) is that if the IMC controller is designed to be equal to the plant inverse model ($\Phi_c = \Phi_m^{-1}$), perfect reference tracking (Y = R) with asymptotically vanishing control error and disturbance rejection can be achieved despite any model/plant mismatch (i.e. $\Phi_p \neq \Phi_m^{-1}$). This can be seen from Equations (5.5) and (5.6) as for ($\Phi_c = \Phi_m^{-1}$), the input transfer function and the disturbance transfer function become $\Phi^{R} = I$ and $\Phi^{D} = 0$, respectively. If a low-pass pre-filter F is introduced in the control loop, Equation (5.5) can be rewritten as:

$$\Phi^{R} = [I + (\Phi_{c}^{-1} F^{-1} - \Phi_{m})\Phi_{p}^{-1}]^{-1} \tag{5.7}$$

In the ideal case, i.e. when the plant model is perfect and there is no disturbance, the above equation results in $\Phi^{R} = F$, which means that a desired closed-loop robust performance of the control system can be easily achieved by a proper design of the pre-filter. Furthermore, by choosing the pre-filter dynamics appropriately, the stability of the closed-loop system can be achieved for any degree of plant/model mismatch. In general, slower filters are required for large model errors. This can be interpreted as (*de-tuning*) of an ideal controller, while the procedure is more straightforward and intuitive than that of conventional linear controller.

Another important property of the IMC is that if both of the plant Φ_p and the IMC controller Φ_c are stable, the stability of the overall IMC system is achieved subjected to perfect plant modelling ($\Phi_p = \Phi_m$).This can be seen from Figure (5.1) as for($\Phi_p = \Phi_m$), the plant input control signal and plant output Y can be derived as follows:

$$U = \Phi_c(R - D\Phi_p) \tag{5.8}$$

$$Y = \Phi_p\Phi_c R + (I - \Phi_p\Phi_c)D\Phi_p \tag{5.9}$$

From Equations (5.8) and (5.9), the internal stability of Φ_p and Φ_c determines the stability of U and Y. Therefore, the overall IMC system in Figure (5.1) will be stable for stable Φ_p and Φ_c.

It is clear from the literature that the IMC approach has not been widely applied to the control of mechanical systems. The reason for this could be that the IMC scheme, in its original design form is applicable only to asymptotically stable systems, which is not the case for most mechanical systems. IMC is a powerful control strategy for linear systems, however its performance when applied to non-linear processes is not good enough. The development of a general non-linear extension of IMC faces the difficulty that non-linear systems are usually described by non-linear models while linear IMC is based on transfer function models in addition to the lack in powerful tools for design and analysis of robust non-linear controllers. However, several non-linear IMC based

controllers have been reported due to recent advances in intelligent modelling techniques. Actually, the key characteristics of the IMC described above also apply in the non-linear case. For example, a number of researchers have suggested Neural Networks to provide the non-linear plant models necessary for IMC from input/output data collected from the plant. Likewise, the application of neural networks to the inverse modelling of non-linear systems is common in the literature, particularly in the field of robotics control. This due to the fact that Neural Networks parallel processing architecture, adaptation and learning capabilities, and fast processing for large-scale dynamic systems provide solid base to represent the robot forward and inverse model within the IMC controller structure. Li et al. proposed compensations procedure for the robot dynamics, before the standard IMC scheme can be applied. This compensation procedure consists of two stages, namely pre-linearization using approximate inverse dynamic model and pre-stabilization using a conventional PD feedback loop [Li et al., 1995]. Li et al. proposed an adaptive algorithm based on Neural Networks to construct a joint-based IMC for robot manipulators. In this method, a Neural Network inverse model and a conventional PD feedback were used to pre-linearise and pre-stabilize the plant in a fixed structure IMC controller. The utilized Neural Network consists of an n sub-network structure, each sub-network operates independently based on each link angle, velocity, and acceleration to generate respective link actuating torque.

5.2. Modified Neuro-Fuzzy Internal Model Cartesian Control

In their subsequent work, Li et al. implemented the IMC structure to control robot manipulators in a comparison study with internal model control [Li et al., 1996]. In

their work, the computed-torque controller was used as the pre-compensation structure, which comprises a lineariser and a stabilizer, to modify the dynamics of the robot manipulator so that the standard IMC structure can be implemented without violating its original straightforward and intuitive design principle. In their proposed method, it can be seen that the computed-torque control is constructed in the same way as the design of the pre-compensation structure of the robot in the IMC system. Therefore, the overall robot IMC system can actually be considered as a framework combining the computed-torque-like control structure, i.e. the pre-compensation structure, as the inner loop, with the general IMC structure as the outer loop. On the other hand, from the viewpoint of a standard robot computed-torque control, the IMC configuration can be considered as an enhanced scheme of this control algorithm, because the outer loop structure in the IMC configuration, which includes the feedforward and the feedback component, can be considered as an additional compensator to the original computed-torque controller. Another IMC modification technique has been presented in [Liu and Yu, 2002]. In this work, a double control scheme, based on the PID control law and the internal model control strategy is used to control a continuously stirred tank reactor (CSTR) in which parameter uncertainty and system disturbance are considered. The plant model was constructed using generalized neuro-fuzzy network. Modelling errors due to input/output data of the plant result in mismatching between the inversion model and plant's practical characteristics. If the degree of model mismatching increases to some extent, the closed-loop response of the CSTR plant tends to be unstable because the internal model structure becomes invalid. So, the suggested modified control structure comprises neuro-fuzzy inverse model of the plant shunted by a conventional PID feedback controller and used in the forward path of the internal model control.

This has been done to improve the response performance and to extend the controllable range of the CSTR [Liu and Yu, 2002]. [Wen et. al., 2003] presented a general algorithm on internal model control based on fuzzy neural networks, where both the inverse and plant models are represented by FNN with online error back-propagation learning algorithm. These ideas motivated the implementation of the robot Cartesian controller system illustrated in figure (5.3). In the proposed control scheme, an approximate inverse kinematics model can satisfactorily act as a Cartesian controller. In this system, the original structure of the IMC is modified to make it suitable for practical Cartesian adaptive control of robotic manipulators. In this way, the IMC can be regarded as an adaptive form of *command generator* working for an existing joint-based robot controller. A pre-compensation structure, which comprises the neuro-fuzzy inverse dynamic neural network and the FPID servo controller explained in chapter (4), is used so that the IMC structure can be implemented using the inverse kinematics neuro-fuzzy network and the forward kinematics mathematical model, which is generally a group of trigonometric equations (see Appendix A), of the robot arm to achieve Cartesian control of the robot manipulator. The sensed joint displacements are transformed using the forward kinematics equations to the actual Cartesian displacement of the end effector, or this Cartesian displacement can be obtained by means of any ultrasonic measuring system. Note that in this case, the internal model (forward kinematics mathematical model) represents the model of the robot in addition to the existing joint-based controller cascaded by the Cartesian position calculation. From the block diagram shown in figure (5.4), the input/output (from X_d to X_m) relationship can be directly derived as:

$$\Phi^{R} = [I + (\Phi_{IK}^{-1} - \Phi_{K})\tilde{\Phi}_{p}^{-1}]^{-1} \tag{5.10}$$

where $\tilde{\Phi}_{p}$, the part surrounded by the dashed line, is the existing joint-based controller regarded as the pre-compensation structure of the robot dynamics, where Φ_{P} represents the robot arm dynamics, Φ_{FF} is the neuro-fuzzy inverse dynamic neural network regarded as the pre-lineariser, and Φ_{FB} is the FPID servo controller regarded as the stabilization element. The input/output relation of this part can be derived as:

$$\tilde{\Phi}_{p} = \frac{\Phi_{K}\Phi_{P}(\Phi_{FF} + \Phi_{FB})}{(I + \Phi_{P}\Phi_{FB})} = \Phi_{K} + \frac{\Phi_{K}(\Phi_{P} - \Phi_{FF}^{-1})}{\Phi_{FF}^{-1}[I + \Phi_{P}\Phi_{FB}]} \tag{5.11}$$

Consequently equation (5.10) can be rewritten as:

$$\Phi^{R} = \left[I + \frac{(\Phi_{IK}^{-1} - \Phi_{K})(I + \Phi_{p}\Phi_{FB})}{\Phi_{K}\Phi_{p}[\Phi_{FF} + \Phi_{FB}]} \right]^{-1} \tag{5.12}$$

Comparing the configuration shown in figures (4.1), (5.1), and (5.4), it can be seen that the neuro-fuzzy joint-based control is constructed in the same way as the design of the pre-compensation structure for the robot in the neuro-fuzzy Cartesian IMC system. Therefore, the overall Cartesian IMC can be considered as a framework combining the neuro-fuzzy joint-based control structure, i.e. the pre-compensation structure, as the inner loop controller, with the general IMC structure as the outer loop controller.

On the other hand, from the viewpoint of the neuro-fuzzy joint-based controller, the IMC configuration can be considered as an enhanced scheme of this control algorithm, because the outer loop structure in the IMC configuration, which includes the feedforward component Φ_{IK} (inverse kinematics neuro-fuzzy network) and the feedback component Φ_K (forward kinematics mathematical model), can be considered as an additional compensator for the original neuro-fuzzy joint-based controller. In another form, the IMC can be regarded as an adaptive form of a *command generator* for the existing neuro-fuzzy joint-based robot controller by introducing the Neuro-fuzzy inverse kinematics network outside the control loop achieving compensation for robot Cartesian uncertainties by modifying the desired input Cartesian trajectory.

5.3. Training Procedure

Both of the inverse kinematics and inverse dynamics neuro-fuzzy networks are constructed using the offline procedure explained in chapter (3). Adaptation for inverse dynamic neuro-fuzzy network parameters is performed online as explained in chapter (3) and chapter (4). Also, adaptation for the neuro-fuzzy inverse kinematics network parameters is performed online using the joint error signal calculated as an input for the FPID servo controller. This error signal is propagated through the inverse dynamic neuro-fuzzy network to the inverse kinematics neuro-fuzzy network to form adaptive Cartesian control through online parameters optimization as explained in chapter (3). It can be seen that by proper tuning for the parameters of the inverse kinematics neuro-fuzzy network, $\Phi_{IK}^{-1} \cong \Phi_K$ and Equation (5.12) can be reduced to $\Phi^R \cong I$ resulting in perfect tracking result over the Cartesian trajectory. Note that in the above analysis, a

well tuned existing joint-based controller or its model is not strictly necessary for perfect tracking, only a well trained neuro-fuzzy inverse kinematics model is enough for the control structure to follow the Cartesian trajectories.

From the above analysis, apparently there are no restrictions on the choice of the Φ_{IK}^{-1} and Φ_K transfer functions as far as they are the inverse of each other. However, it is most likely to select them to be the approximate inverse and forward model of the controlled plant. The reason of this can be explained through a modification of the control loop diagram of figure (5.4) to that of figure (5.5) and figure (5.6). Since Φ_{IK}^{-1} and Φ_K should ideally cancel each other out completely in the proposed control system, the positive feedback loop (1) appears to have the possibility of producing an infinite gain. However, if the controller Φ_{IK}^{-1} is an exact inverse of the controlled process (joint-based controller and plant up to Cartesian values), the signal in feedback loop (1) is balanced by that of feedback loop (2) effectively. The fundamental advantage of this scheme is that only an exact inverse neuro-fuzzy kinematic model could be enough for perfect Cartesian tracking, while a well tuned existing joint-based controller or its model is not strictly necessary.

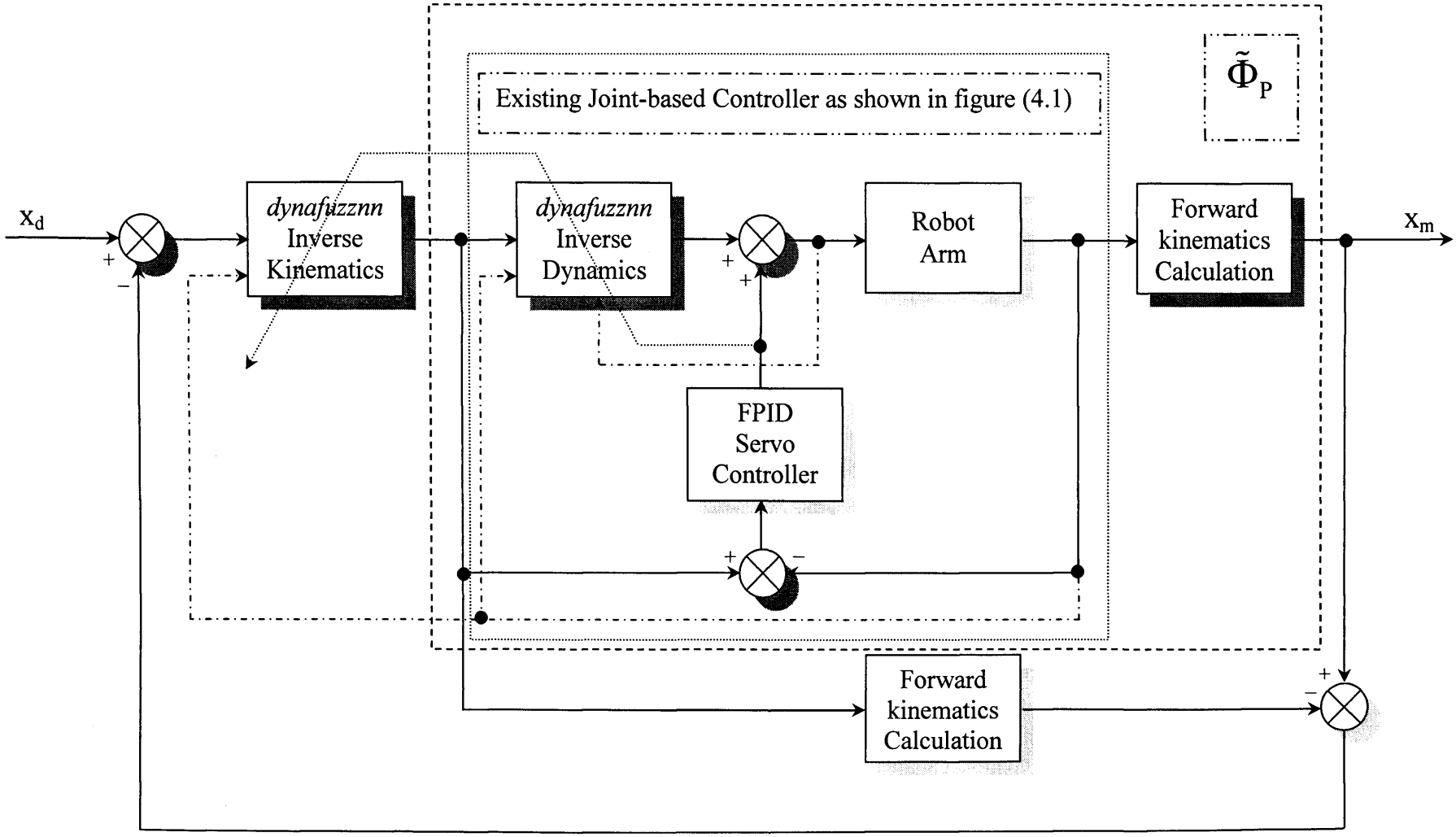


Figure (5.3). Modified neuro-fuzzy internal model controller

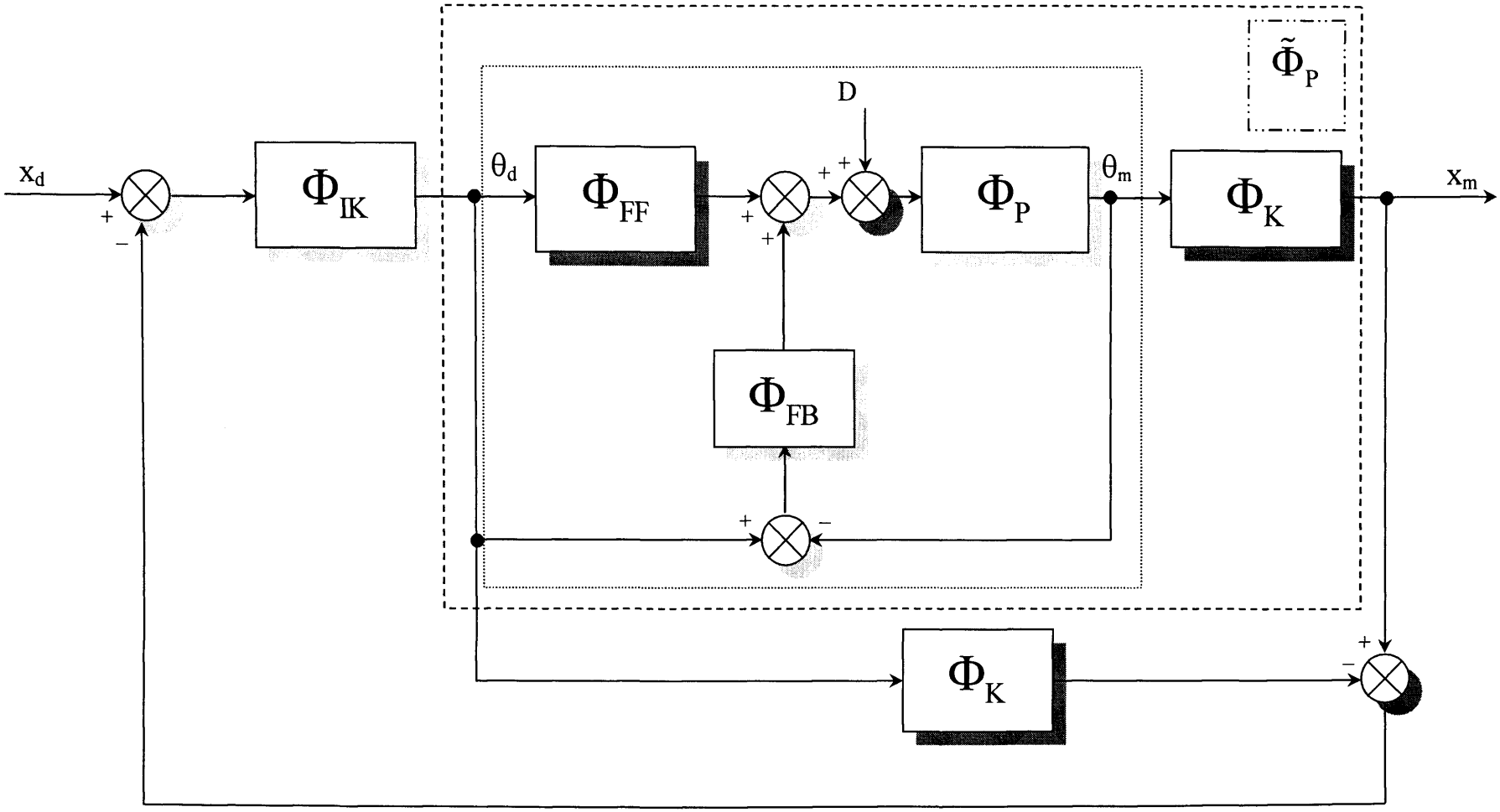


Figure (5.4). Internal model controller block diagram.

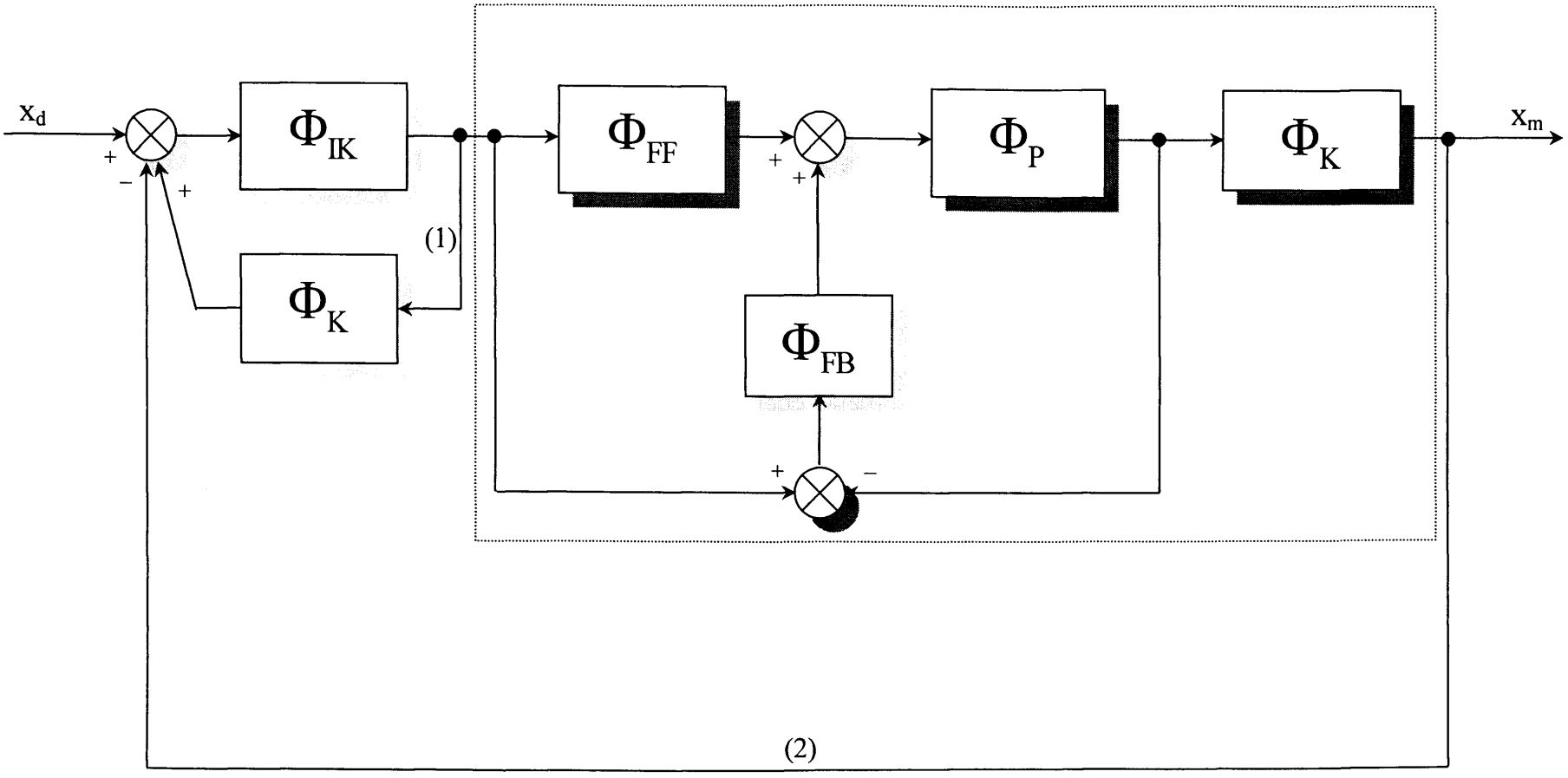


Figure (5.5). Simplified internal model controller block diagram

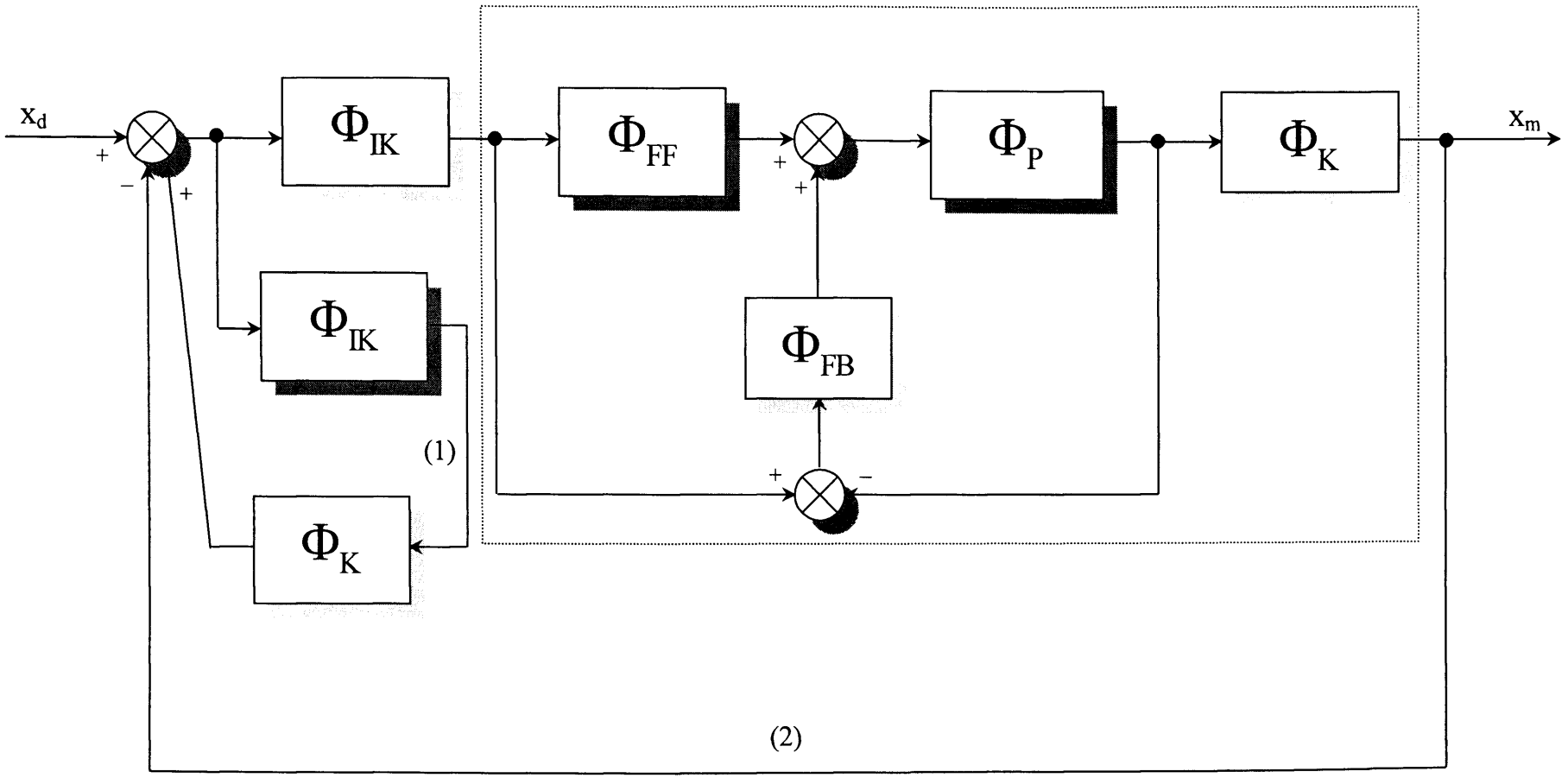


Figure (5.6). Modified internal model controller block diagram

5.4. Robustness Analysis

In this section, the proposed controller structure is analysed in terms of disturbance rejection and sensitivity to model uncertainties. The analysis is compared with the original joint-based controller response to disturbance and model uncertainties to highlight the added benefits from the new structure.

5.4.1. Disturbance Analysis

For robotic manipulators control, external disturbance are due to load torques acting at the joints as shown in figure (5.4). The disturbance transfer function for the neuro-fuzzy joint-based controller (from θ_d to X_m) can be directly derived as:

$$\tilde{\Phi}^D = \frac{\Phi_K \Phi_P}{I + \Phi_P \left(\Phi_{FB} \right)} \tag{5.13}$$

For the proposed neuro-fuzzy internal model Cartesian controller, the disturbance transfer function (from X_d to X_m) can be directly derived as:

$$\Phi^D = \frac{\Phi_K \Phi_P}{I + \Phi_P \left(\Phi_{FB} + \frac{\left(\Phi_{FF} + \Phi_{FB} \right)}{\left(\frac{I}{\Phi_K \Phi_{IK}} \right) - I} \right)} \tag{5.14}$$

Comparing equation (5.13) and (5.14), it can be seen that the effect of the external disturbances for the modified IMC has been changed over the joint-based controller by the term $\frac{(\Phi_{FF}+\Phi_{FB})}{\left(\frac{I}{\Phi_K\Phi_{IK}}\right)-I}$ in the denominator. This term appears to have the possibility of producing an infinite value driving the disturbance transfer function to zero, resulting in less sensitive control system to load disturbances compared to the original joint-based controller.

5.4.2. Sensitivity Analysis

Generally, in order to analyse the performance of any control system, it is a common practice to replace the plant by its modelled dynamics Φ_m and possible model uncertainties as follows:

$$\Phi_p = (I + \delta\Phi_p)\Phi_m + \Delta\Phi_p \qquad (5.15)$$

where $\delta\Phi_p$ and $\Delta\Phi_p$ are the unmodelled dynamics and/or parameters multiplicative and additive uncertainties of the plant respectively. Both kinds of uncertainties will be studied separately.

5.4.2.1. Sensitivity to Multiplicative Uncertainties

For the neuro-fuzzy joint-based controller alone, the closed loop multiplicative sensitivity function can be obtained as follows:

$$\xi_{\partial\Phi_p}^{\tilde{\Phi}_p} = \frac{\partial\tilde{\Phi}_p \Big/ \tilde{\Phi}_p}{\partial(\delta\Phi_p) \Big/ \delta\Phi_p} = \frac{\partial\tilde{\Phi}_p}{\partial\Phi_p}\frac{\partial\Phi_p}{\partial(\delta\Phi_p)}\frac{\delta\Phi_p}{\tilde{\Phi}_p}$$

$$= \frac{\Phi_K\left(\Phi_{FF}+\Phi_{FB}\right)}{\left(I+\Phi_P\Phi_{FB}\right)^2}\Phi_m\delta\Phi_p\frac{\left(I+\Phi_P\Phi_{FB}\right)}{\Phi_K\Phi_P\left(\Phi_{FF}+\Phi_{FB}\right)} \qquad (5.16)$$

$$= \frac{I}{I+\Phi_P\left(\Phi_{FB}\right)}\frac{\Phi_m}{\Phi_P}\delta\Phi_p$$

For the neuro-fuzzy Cartesian IMC controller, the closed loop multiplicative sensitivity function can be obtained as follows:

$$\xi_{\partial\Phi_p}^{\Phi^R} = \frac{\partial\Phi^R \Big/ \Phi^R}{\partial(\delta\Phi_p) \Big/ \delta\Phi_p} = \frac{\partial\Phi^R}{\partial\Phi_p}\frac{\partial\Phi_p}{\partial(\delta\Phi_p)}\frac{\delta\Phi_p}{\Phi^R}$$

$$= \frac{\Phi_K\left(\Phi_{IK}^{-1}-\Phi_K\right)\left(\Phi_{FF}+\Phi_{FB}\right)}{\left(\Phi_K\Phi_P\left(\Phi_{FF}+\Phi_{FB}\right)+\left(\Phi_{IK}^{-1}-\Phi_K\right)\left(I+\Phi_p\Phi_{FB}\right)\right)^2}\Phi_m\delta\Phi_p$$

$$\frac{\left(\Phi_K\Phi_P\left(\Phi_{FF}+\Phi_{FB}\right)+\left(\Phi_{IK}^{-1}-\Phi_K\right)\left(I+\Phi_p\Phi_{FB}\right)\right)}{\Phi_K\Phi_P\left(\Phi_{FF}+\Phi_{FB}\right)} \qquad (5.17)$$

$$= \frac{I}{I+\Phi_P\left(\Phi_{FB}+\dfrac{\left(\Phi_{FF}+\Phi_{FB}\right)}{\left(\dfrac{I}{\Phi_K\Phi_{IK}}\right)-I}\right)}\frac{\Phi_m}{\Phi_P}\delta\Phi_p$$

Comparing equation (5.16) and (5.17), again it can be seen that the multiplicative sensitivity for the modified IMC has been changed over the existing joint-based controller by the term $\frac{\left(\Phi_{FF}+\Phi_{FB}\right)}{\left(\frac{I}{\Phi_K\Phi_{IK}}\right)-I}$ in the denominator which appears to have the possibility of producing an infinite value resulting in less sensitive control system to multiplicative uncertainties compared to the original joint-based controller.

5.4.2.2. Sensitivity to Additive Uncertainties

For the neuro-fuzzy joint-based controller alone, the closed loop additive sensitivity function can be obtained as follows:

$$\begin{aligned}\xi_{\Delta\Phi_p}^{\tilde{\Phi}_p} &= \frac{\partial\tilde{\Phi}_p \Big/ \tilde{\Phi}_p}{\partial(\Delta\Phi_p) \Big/ \Delta\Phi_p} = \frac{\partial\tilde{\Phi}_p}{\partial\Phi_p}\frac{\partial\Phi_p}{\partial(\Delta\Phi_p)}\frac{\Delta\Phi_p}{\tilde{\Phi}_p} \\ &= \frac{\Phi_K\left(\Phi_{FF}+\Phi_{FB}\right)}{\left(I+\Phi_P\Phi_{FB}\right)^2}\Delta\Phi_p\frac{\left(I+\Phi_P\Phi_{FB}\right)}{\Phi_K\Phi_P\left(\Phi_{FF}+\Phi_{FB}\right)} \\ &= \frac{I}{I+\Phi_P\left(\Phi_{FB}\right)}\frac{\Delta\Phi_p}{\Phi_P}\end{aligned} \tag{5.18}$$

For the neuro-fuzzy Cartesian IMC controller, the closed loop additive sensitivity function can be obtained as follows:

$$
\begin{aligned}
\xi_{\Delta\Phi_p}^{\Phi^R} &= \frac{\partial\Phi^R \big/ \Phi^R}{\partial(\Delta\Phi_p) \big/ \Delta\Phi_p} = \frac{\partial\Phi^R}{\partial\Phi_p}\frac{\partial\Phi_p}{\partial(\Delta\Phi_p)}\frac{\Delta\Phi_p}{\Phi^R} \\
&= \frac{\Phi_K\left(\Phi_{IK}^{-1}-\Phi_K\right)\left(\Phi_{FF}+\Phi_{FB}\right)}{\left(\Phi_K\Phi_P\left(\Phi_{FF}+\Phi_{FB}\right)+\left(\Phi_{IK}^{-1}-\Phi_K\right)\left(I+\Phi_p\Phi_{FB}\right)\right)^2}\Delta\Phi_p \\
&\qquad \frac{\left(\Phi_K\Phi_P\left(\Phi_{FF}+\Phi_{FB}\right)+\left(\Phi_{IK}^{-1}-\Phi_K\right)\left(I+\Phi_p\Phi_{FB}\right)\right)}{\Phi_K\Phi_P\left(\Phi_{FF}+\Phi_{FB}\right)} \\
&= \frac{I}{I+\Phi_P\left(\Phi_{FB}+\dfrac{\left(\Phi_{FF}+\Phi_{FB}\right)}{\left(\dfrac{I}{\Phi_K\Phi_{IK}}\right)-I}\right)}\frac{\Delta\Phi_p}{\Phi_P}
\end{aligned}
\tag{5.19}
$$

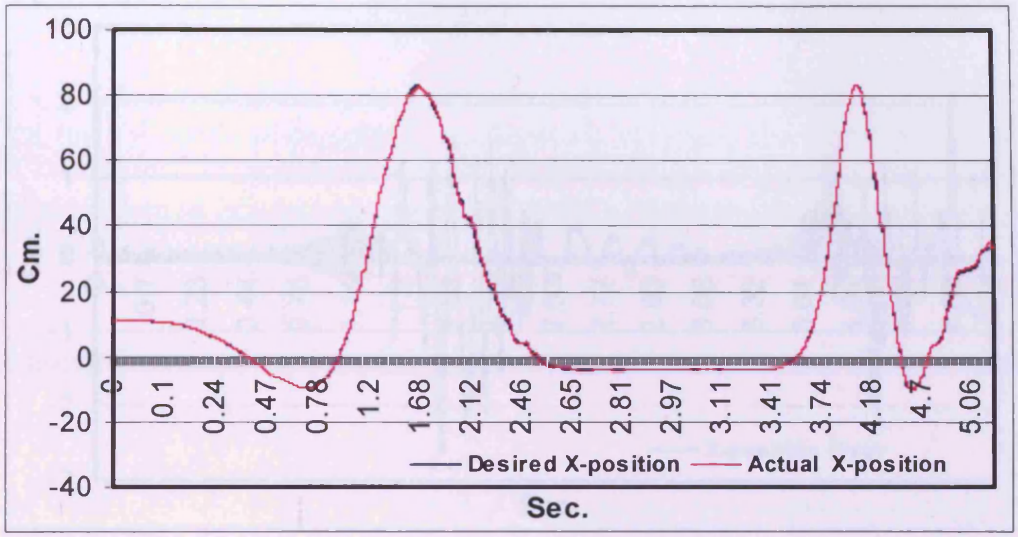
Comparing equations (5.18) and (5.19), again it can be seen that the additive sensitivity for the modified IMC has been changed over the existing joint-based controller by the term $\dfrac{\left(\Phi_{FF}+\Phi_{FB}\right)}{\left(\dfrac{I}{\Phi_K\Phi_{IK}}\right)-I}$ in the denominator. This term appears to have the possibility of producing an infinite value resulting in lower sensitivity to additive uncertainties compared to the original joint-based controller.

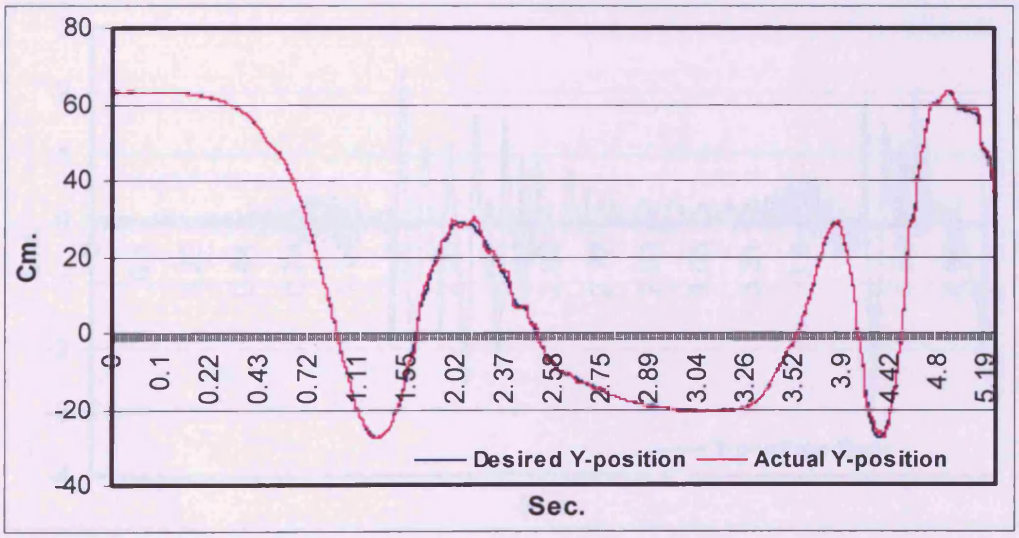
From previous analysis, it is clear that the overall performance of the system in the modified IMC structure is improved over the existing joint-based controller.

The control system developed can be regarded as an inner joint-based control loop that controls the joint angle of each link in addition to a Cartesian control loop which is closed around the joint control loop. The Cartesian controller adds an offset Cartesian position command, derived from the measured (calculated) Cartesian position of the end-effector, to the joint control loop. Thus, the purpose of the Cartesian controller is to minimize the measured (calculated) Cartesian end-effector position by modifying the commanded end-effector reference position which in turn modifies the joint angles references.

5.5. Simulation Results

In order to verify the effectiveness of the proposed Cartesian internal model control system, the proposed control system was tested by applying it to control the first three links of the Puma 560® industrial robot. The controller algorithm was programmed in C++ and linked to the *"Pro/Mechanica®"* virtual model of Puma 560® industrial robot as a subroutine as explained in Appendix (B). The joint coordinates trajectories are re-planned in Cartesian coordinates and then applied to the suggested control system as the reference Cartesian trajectory. The robot was tested again while carrying the same fixed payload of 7.0 kg. Figures (5.7) and (5.8) show the Cartesian position tracking results for the suggested neuro-fuzzy Cartesian internal model controller. The obtained results highly support the validity of the proposed control system.





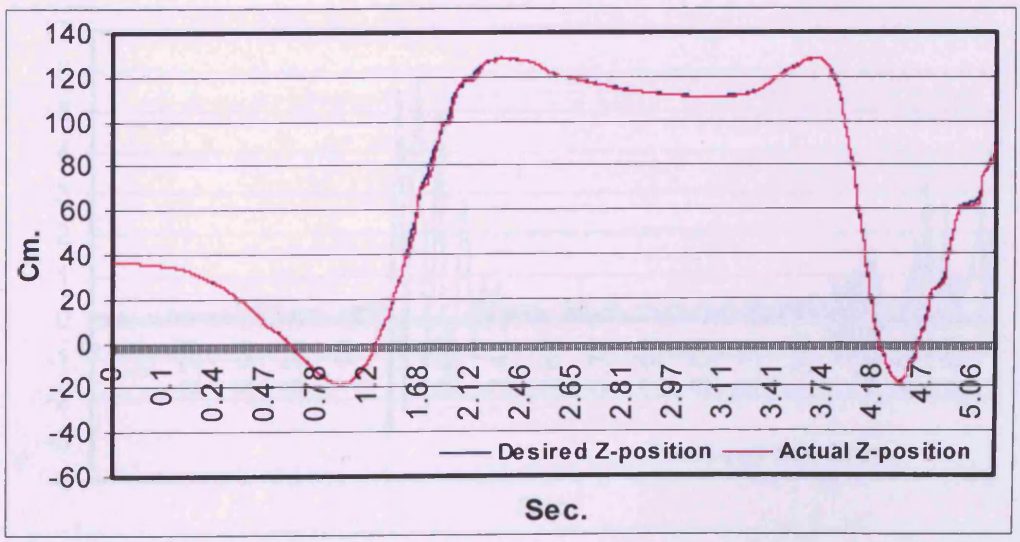
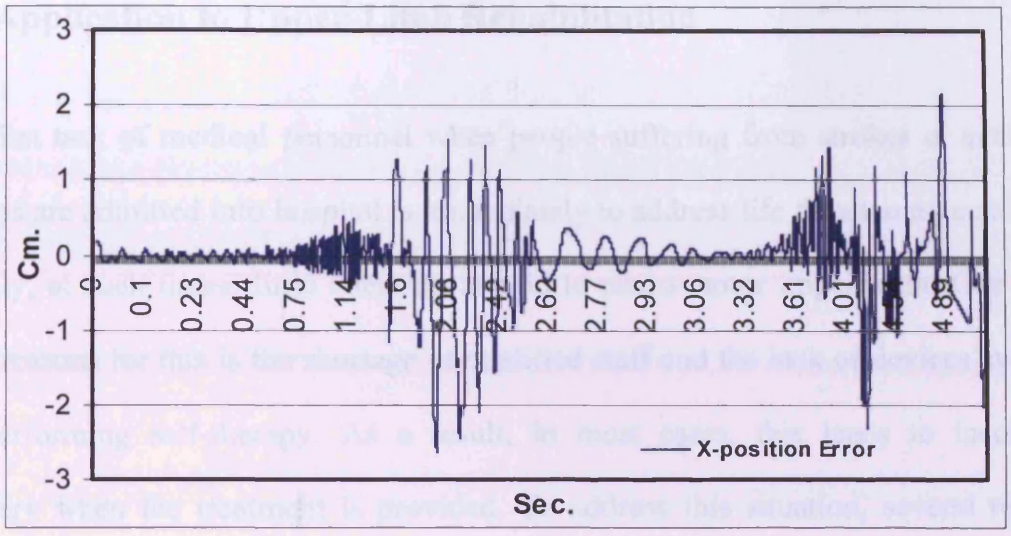
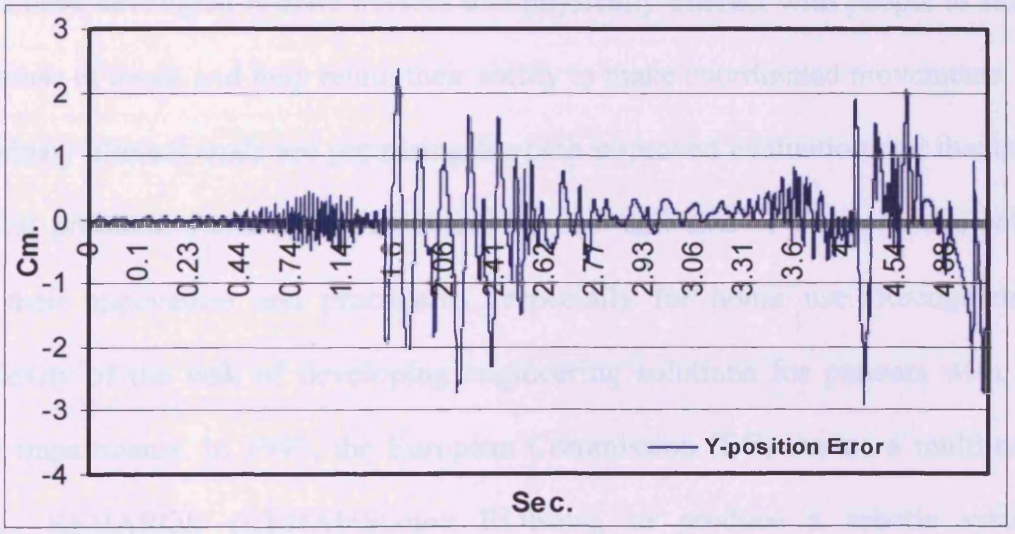


Figure (5.7). Cartesian trajectories tracking results





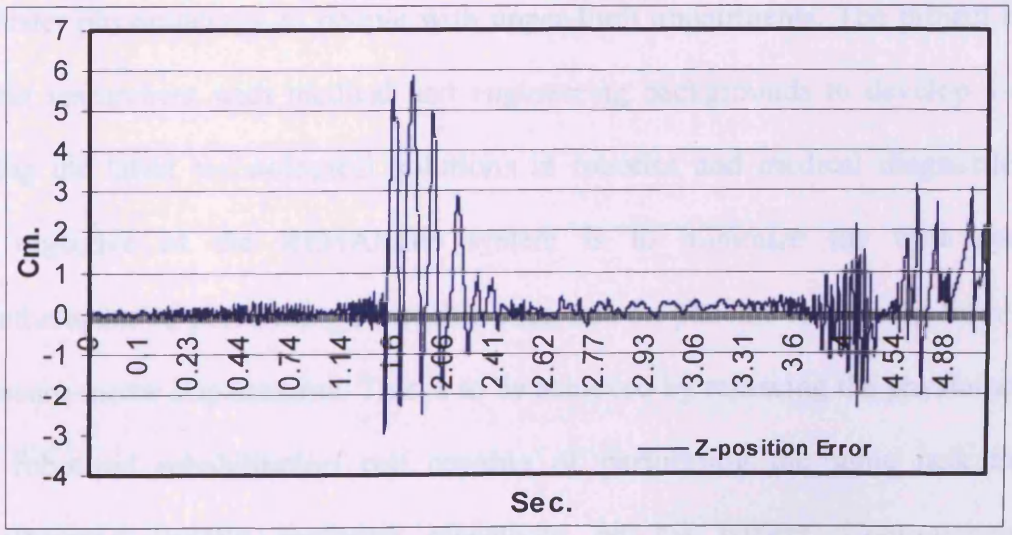


Figure (5.8). Cartesian trajectories tracking errors

5.6. Application to Upper-Limb Rehabilitation

The first task of medical personnel when people suffering from strokes or accidental injuries are admitted into hospital is immediately to address life threatening conditions. Usually, at such times, little attention is paid to neuro-motor impairment. One of the main reasons for this is the shortage of qualified staff and the lack of devices available for performing self-therapy. As a result, in most cases, this leads to incomplete recovery when the treatment is provided. To address this situation, several research groups have developed robotic devices that physically interact with people to stimulate their sense of touch and help retain their ability to make coordinated movements. While preliminary clinical trials are promising for both improved evaluation and therapy, key practical problems remain. In particular, the cost and size of the proposed solutions limit their application and practicality, especially for home use. Recognizing the complexity of the task of developing engineering solutions for patients with neuro-motor impairments, in 1999, the European Commission (EC) started a multi-national project, REHAROB (REHAbilitation ROBots), to produce a robotic system to administer physiotherapy to people with upper-limb impairments. The project brought together researchers with medical and engineering backgrounds to develop a system utilizing the latest technological solutions in robotics and medical diagnostics. The main objective of the REHAROB system is to minimize the time spent by physiotherapists in performing repetitive exercises on patients recovering from upper-limb neuro-motor impairments. This is to be achieved by replacing the physiotherapists by a robotized rehabilitation cell capable of performing the same task that the physiotherapist usually performs repeatedly on the patient. Consequently, the

introduction of the proposed system will allow more patients to be seen, assessed and rehabilitated by the physiotherapist by limiting his/her job to the diagnosis of the proper exercises, while leaving the muscular work to the robotized rehabilitation cell which can be regarded as a tireless physiotherapist.

5.6.1. Robotized Upper-Limb Rehabilitation

The proposed system will include in the final stage two industrial robots adapted for medical applications plus specialized teach-in and control modules. Information from sensors attached to a patient's arm together with data about the robot's angular position and velocity will be used to control and coordinate the movements of the robots to perform personalized sequences of exercises on the patient. There are two phases in performing the robotized physiotherapy with the proposed system. The first phase is the teach-in phase. During this phase, the physiotherapist performs a prescribed exercise on the patient's arm and at the same time one or two robots holding the arm freely follow its movements. Simultaneously, the motion trajectories for the next phase are generated by the robot controllers using data captured on the robot joint angles and velocities. The second phase is the play-back phase during which the robot/robots perform the taught exercises without the help of the physiotherapist. The selection of a suitable sequence of exercises in order to achieve a satisfactory rehabilitation result is a key to the successful implementation of the proposed system. The design and functionality of a knowledge-based system (KBS) is to assists physiotherapists in choosing the most appropriate sequence of exercises. One of the main problems is to simulate the muscle resistance torques, specific for a given type of rehabilitation procedure. The patient's

resistance torque values depend on a number of factors, such as the type of arm motion, the degree of motor impairments of the arm, the sequence of the given exercises, etc. This makes the load estimation an unsolvable task requiring the control system to be designed considering this situation.

A video library of 45 exercises has been created by the medical experts involved in REHAROB to include most of the exercises that are commonly performed by physiotherapists on patients with upper-limb neuro-motor problems. The KBS objective is to study the physiotherapist's decision-making process and develop a mechanism that proposes a particular sequence of exercises depending on the status of the patient. This intelligent mapping can be achieved by first encoding the exercises and the patient's data into formats suitable for further processing by the KBS [Pham et. al., 2001]. The rehabilitation exercises can be encoded in the form of the duration of the exercise cycle, movement range in each joint, degree of complexity, patient posture, and then finally categorized into three groups namely: Simple, Moderate, and Complex according to the number of joints from the human arm involved in the required motion. The KBS design itself is beyond the objectives of this thesis, as this categorization has no effect on the robot arm control system, where the movement is recorded in the form of robot end-effector position, link angles, and link velocities trajectories in the teach-in stage. It may affect the decision of utilizing only one robot manipulator or using two cooperating manipulators to perform the required exercise or group of exercises. Details of the KBS design can be found in [Pham et. al., 2001].

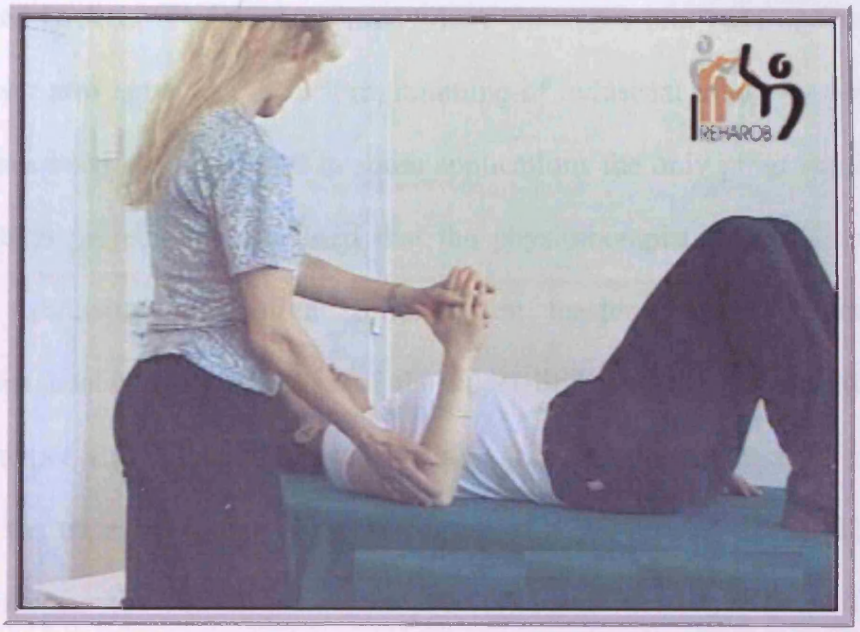


Figure (5.9.a). Simple exercise – Start position.

Figure (5.9.b). Simple exercise – End position.

Figure (5.9) illustrates one of the exercises categorized as a simple exercise. Here, with the patient lying on a couch, the arm is stretched until it points vertically upwards and

then lowered back to the initial position where the upper arm rests against the couch and the lower arm is held vertical. Programming of industrial robots by demonstration (teach-in) has been a popular and in some applications the only programming method. In REHAROB project, it is planned that the physiotherapist trains the robots while he/she is exercising the patient limb. When teaching mode is activated, the physiotherapist takes over the load of the upper-limb and the orthoses (devices that holds the upper and lower arms during therapy) from the robot(s) by grasping the handles of the outer shells covering the human arm [Kovacs et. al., 2001]. He or she can then exercise the patient while the robot(s) learns the trajectories. The teach-in stage is defined as performing the upper-limb rehabilitation exercises with the robot arm attached to the human arm and in idle condition (almost no actuating torques acting on the joint drivers and with links free to rotate with small forces) with the help of the physiotherapist expert as shown in figure (5.10). To achieve idle motion following (without exerting any forces) from the robot manipulators during teach-in stage, it is a requirement to control the grasping force subject to the end-effector to be zero. This requires implementation of incremental position based force control by introducing force sensors between the robots end-effectors and the attachment mechanism [Lange and Hirzinger, 1996]. The reference force trajectory in this case will be constant and equal to zero. Since it is aimed to use standard industrial robots controller with digital position control in REHAROB, therefore the teach-in control stage will be implemented by an inner-loop/outer-loop control architecture. The inner-loop represents the robot internal position control, while the outer-loop represents standard robots programming language based force controller [Kovacs et. al., 2001].

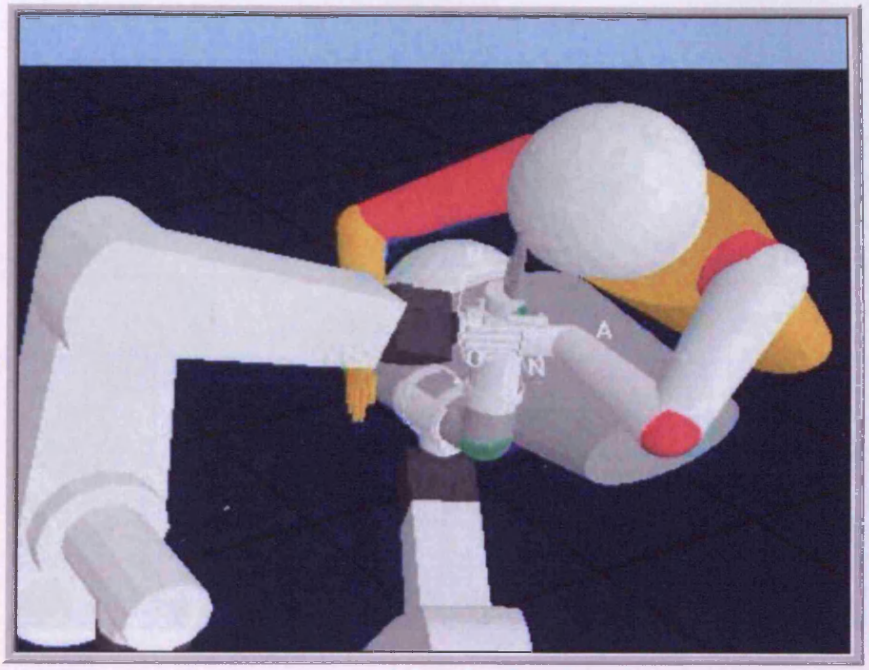


Figure (5.10). Representation of teach-in mode.

During this stage the position sensors of the robot arm records the resulting motion of each link in the robot in addition to the end-effector Cartesian position with respect to the reference global coordinates. These recorded trajectories are then used as the decoded exercises for the upper-limb rehabilitation application control system using one or two robot arms in the play-back mode as shown in figure (5.11). The use of the robots in the play-back mode requires implementation of incremental position based control system. As the weight of the patient arm and the degree of illness (resistance to motion) are very difficult to be pre-specified in accordance with the exercises trajectories, moreover they vary from patient to patient, the control system has to be capable of handling different patient states while achieving the main target trajectories by considering the patient resistance to motion as external disturbance. Of course the final control system must include some force sensing and safety devices to guarantee safe operation in addition to some emergency tripping devices from the patient himself.

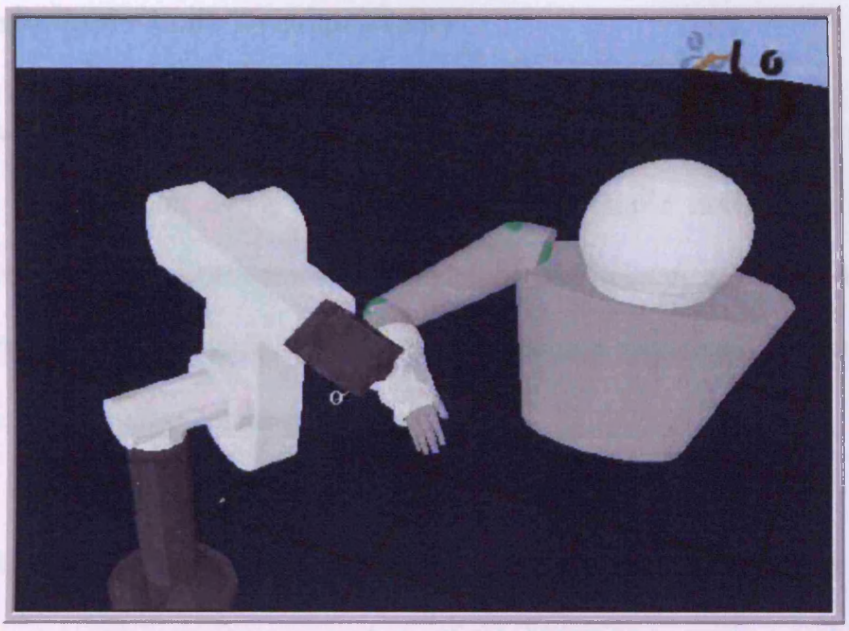


Figure (5.11.a). Representation of play-back mode using one robot.

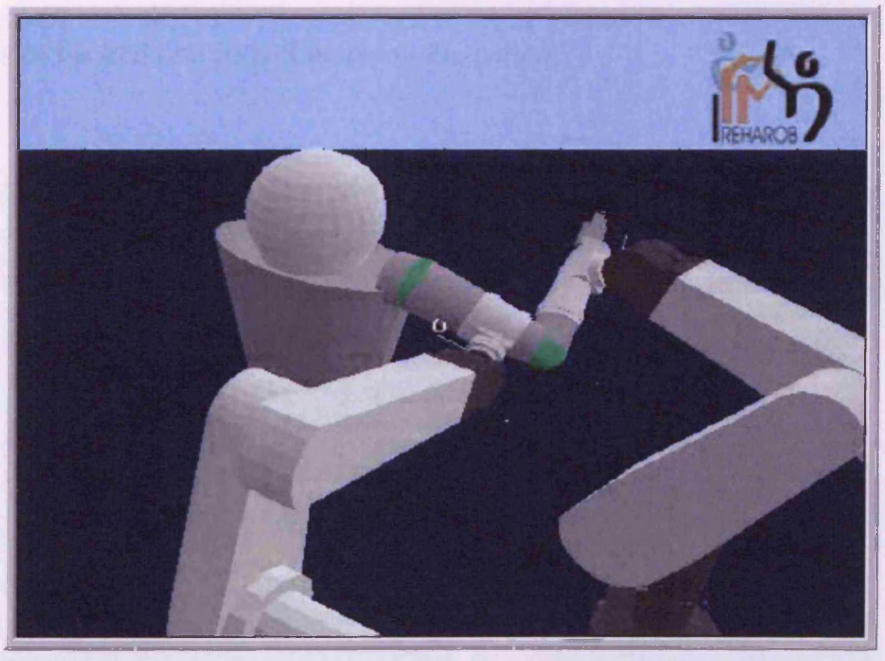


Figure (5.11.b). Representation of play-back mode using two robots.

5.6.2. Human Upper-Limb Dynamic Model

It is required to test the control system developed in previous chapters in performing upper-limb rehabilitation using one robot arm for a simple exercise as first stage. Firstly, a mechanical anthropomorphic model for the human arm was designed as shown in figure (5.12) to facilitate the required motion requested from the robot to perform the rehabilitation training exercises.

The main function of the model of the human upper limb is to simulate and investigate the patient arm movement for a given type of rehabilitation procedure and to generate the attachment point (attachment points of the robots to the patient arm) trajectories in global coordinates. It can be considered as an intermediate unit in between the learning procedures of the cell and its influence on the patient.

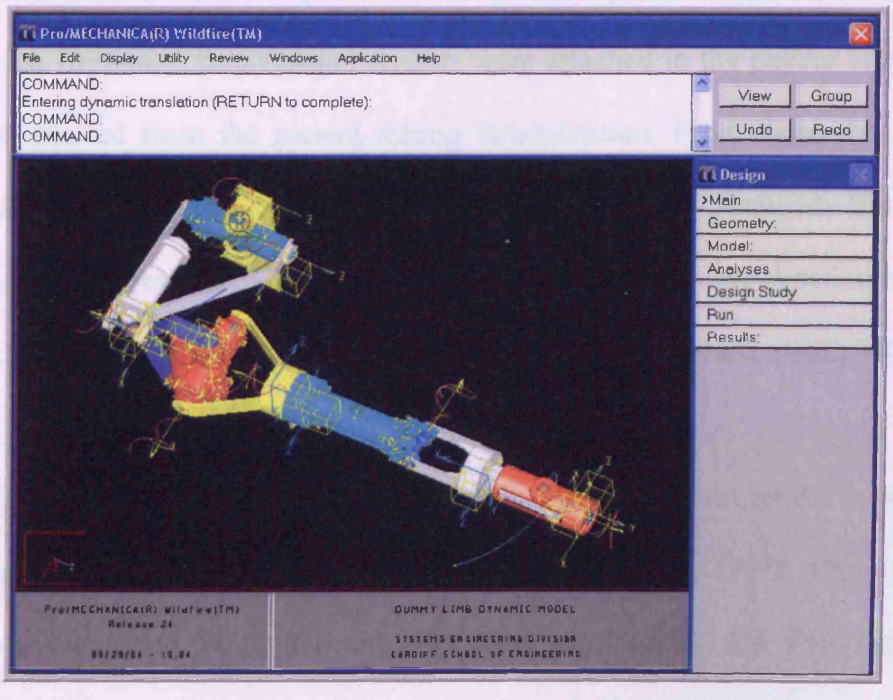


Figure (5.12). Virtual dynamic model of the limb.

From a modelling point of view, the human upper-limb can be considered as a system of rigid bodies (links) connected in a specific way by revolute joints. This is because the processes in the soft tissues, the blood movement and the muscle deformation do not influence the mass-inertia characteristics of the links. The creation of a model of the limb consists of determining the number of links (bones), their shape, and the types of the kinematic pairs (joints). Experimental medical investigations show that the coefficient of friction in the joints is very small, so that joint characteristics can be approximated to those of ideal kinematic pairs. In this study, the arm weights will be approximated by an average value of around 3.5Kg for the lower arm and 4.0Kg for the upper arm including the orthoses. The muscle resistance force/torque values depends on a number of factors, such as the type of arm motion, the degree of motor impairments of the arm, the sequence of the given cycle from the rehabilitation procedure, etc. An experiment to evaluate the range of the resistance torque was performed on a number of patients with different motor impairment for different types of rehabilitation exercises. During these exercises, force/torque sensors were attached to the patient arm to record the values exerted from the patient during rehabilitation. From these data, different profiles for the resistance torque of the patient arm were obtained. Based on all measurements, the resistance torques in the joints is calculated as a function of the joint angles. A detailed description of the process can be found in [Pham et. al., 2001].

To simplify the modelling of the control system, the human arm model is modeled as chain of rigid links connected by movable joints [Hsu, et. al, 1993]. This assumption allows us to formulate the human arm as a robot manipulator. The rigid links form a kinematic linkage, and their motions are constrained according to the degrees of

freedom (DOFs) of the various joints. Figure (5.13) shows the kinematic modelling of a human arm as a chain of two rigid links, upper and lower arm, connected together to other body parts by three joints, shoulder, elbow, and wrist. For the sake of simplicity, we model the shoulder and wrist joints as ball joints of 3 DOFs, and the elbow joint as butterfly (pin) joint of 1 DOF. So, the human arm model is approximated as a chain of two rigid links with a total of 7 DOFs. Using the same notation as in Table (3.1), the arm coordinate system can be represented as in Table (5.1). According to biomechanical modelling, the seven degrees of freedom human arm model contains twenty-nine spring-like muscles in the human arm [Byung-Ju and Freeman, 1995], seven muscles around the elbow joint, thirteen muscles around the wrist, and nine muscles around the shoulder. Dynamical modelling of such structure is very complicated. So, for the sake of simplicity, the arm muscles representation is limited to torsion springs of 2.0 N.m. torque constant at the arm joints and connection points to the robot manipulators, while limiting the arm model to planar motion only.

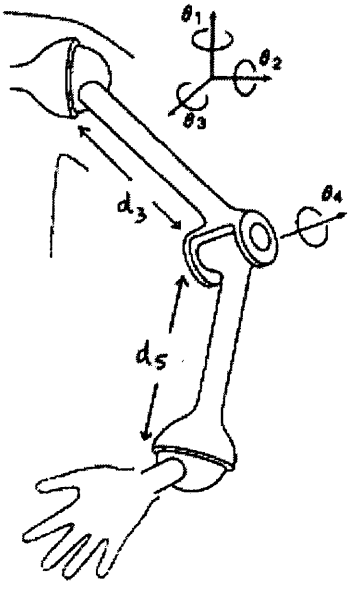


Figure (5.13). Kinematics model of the human arm.

Link	Angle θ	Distance d	Twist α	Length a
L_1	θ_1	0	90^o	0
L_2	θ_2	0	-90^o	0
L_3	θ_3	d_3	90^o	0
L_4	θ_4	0	-90^o	0
L_5	θ_5	d_5	90^o	0
L_6	θ_6	0	90^o	0
L_7	θ_7	0	0^o	0

Table (5.1). Human arm model coordinate system [Hsu, et. al, 1993].

This model was then simplified to ignore the wrist as the attachment points of the robot will be in the lower arm. Then, the lower and the upper arm model are attached to the Puma 560 virtual dynamic model developed in Chapter (3) to test control system functionality. Finally, the resulting combined model forms an upper-limb rehabilitation cell dynamic model. The trajectories required from the robot arm to perform certain exercise are pre-planned according to the simplified model. The aim is to test the ability of the proposed control system to follow any of these decoded exercises with different patient conditions within a safe operating range. Figure (5.14) shows the *"Pro/Mechanica®"* virtual model used to simulate the robot holding the simplified human arm model to perform upper-limb rehabilitation for a simple exercise from the library of exercises supplied by the physiotherapist.

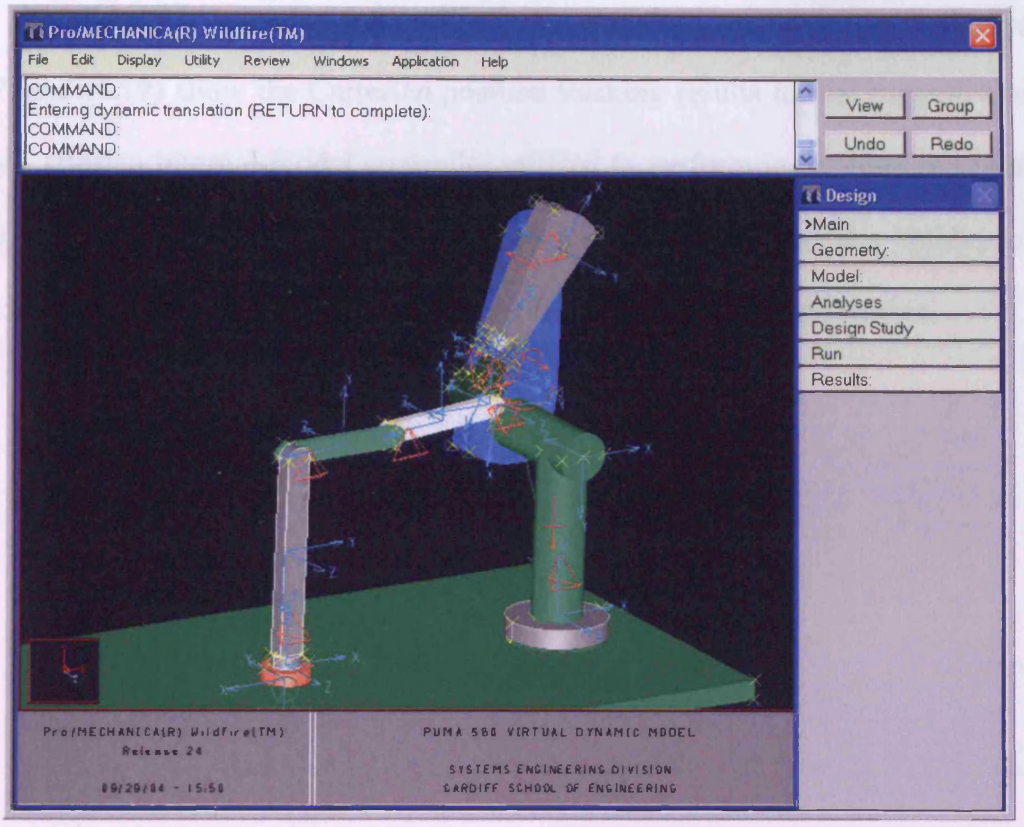


Figure (5.14). Simplified model for upper-limb rehabilitation using one robot.

5.6.3. Upper-Limb Rehabilitation Using One Robot Manipulator

The model shown in figure (5.14) was tested to follow the trajectories for one simple exercise from the library of the exercises provided after decoding this exercise into joint angles trajectories. The neuro-fuzzy controller described in chapter (4) was programmed using C++ and compiled as a custom load to be attached to the model. Figures (5.15) and (5.16) show the position tracking results for the suggested neuro-fuzzy controller utilized to perform the upper-limb rehabilitation using one robot manipulator. The obtained results support the validity of the proposed control system for upper-limb rehabilitation application. Also, the Cartesian neuro-fuzzy internal model controller developed was used to follow the trajectories for the same exercise

after decoding this exercise into end-effector position trajectories as well. Figures (5.17) and (5.18) show the Cartesian position tracking results for the suggested neuro-fuzzy Cartesian internal model controller utilized to perform upper-limb rehabilitation using one robot manipulator. The obtained results also support the validity of the proposed Cartesian control system for upper-limb rehabilitation application.

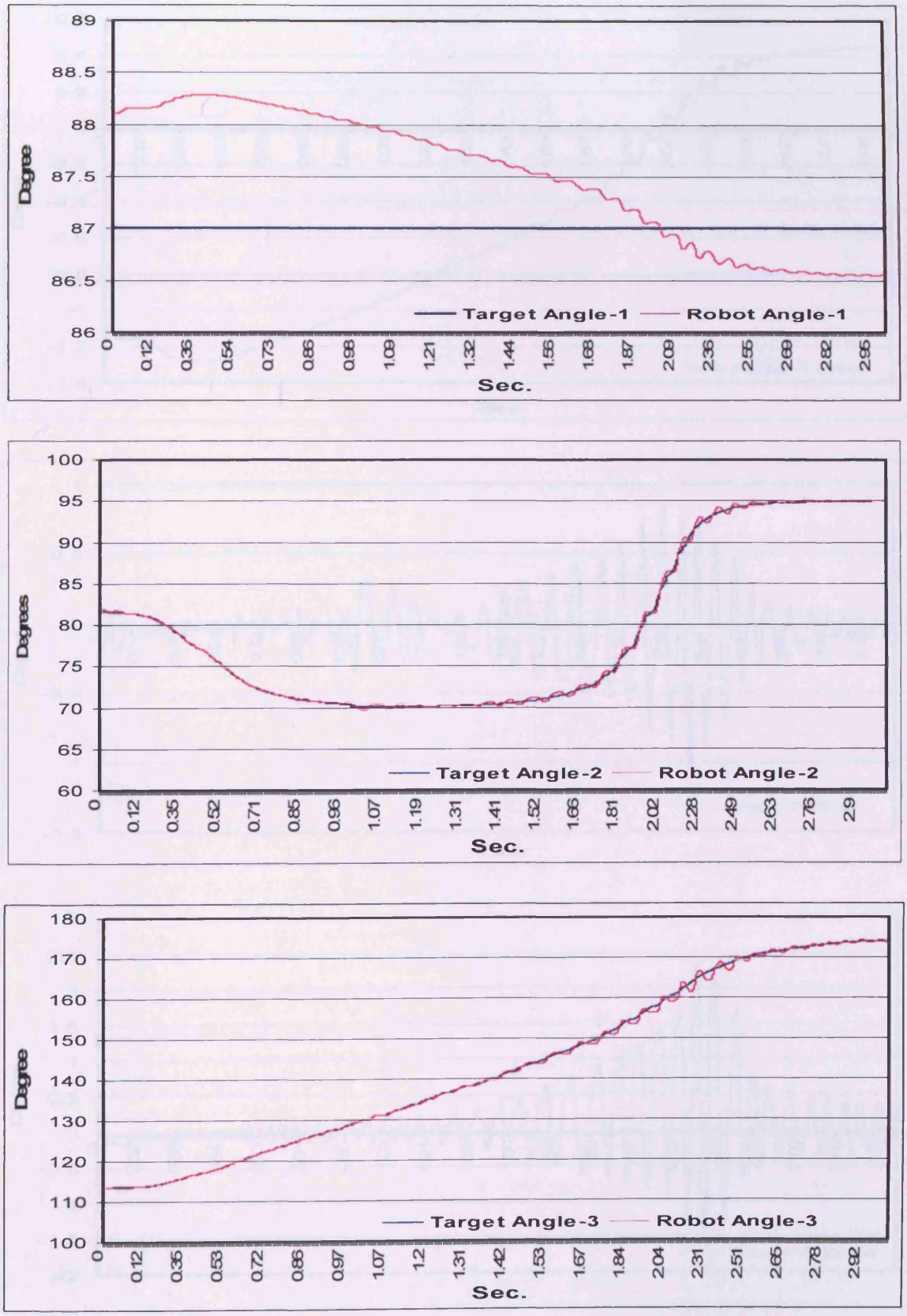


Figure (5.15). Upper-limb rehabilitation position trajectories tracking results.

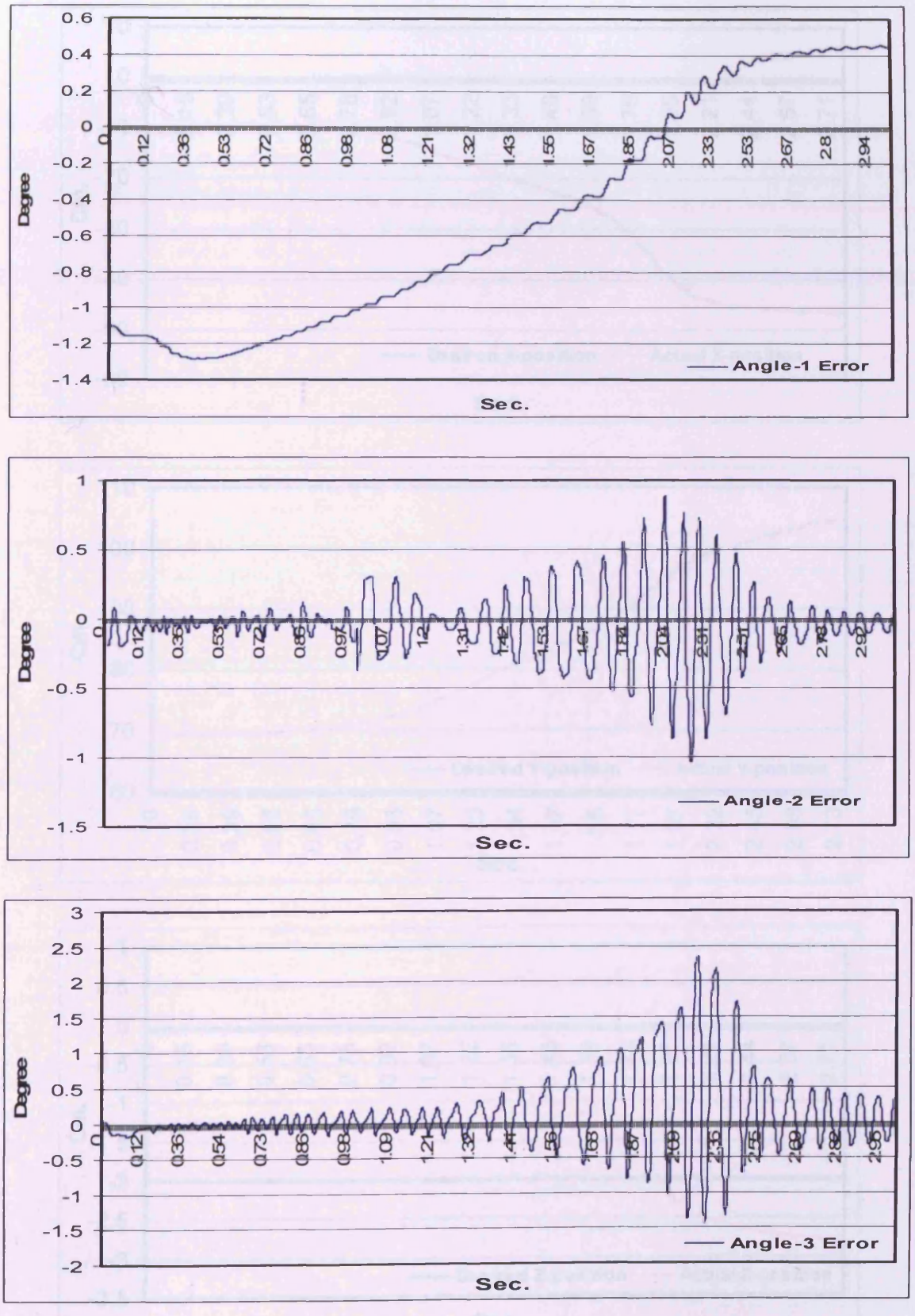
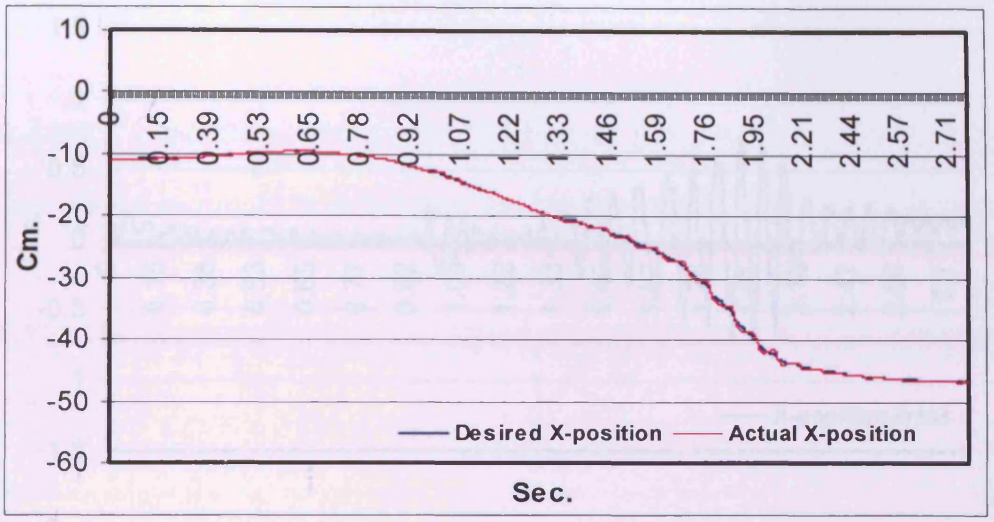
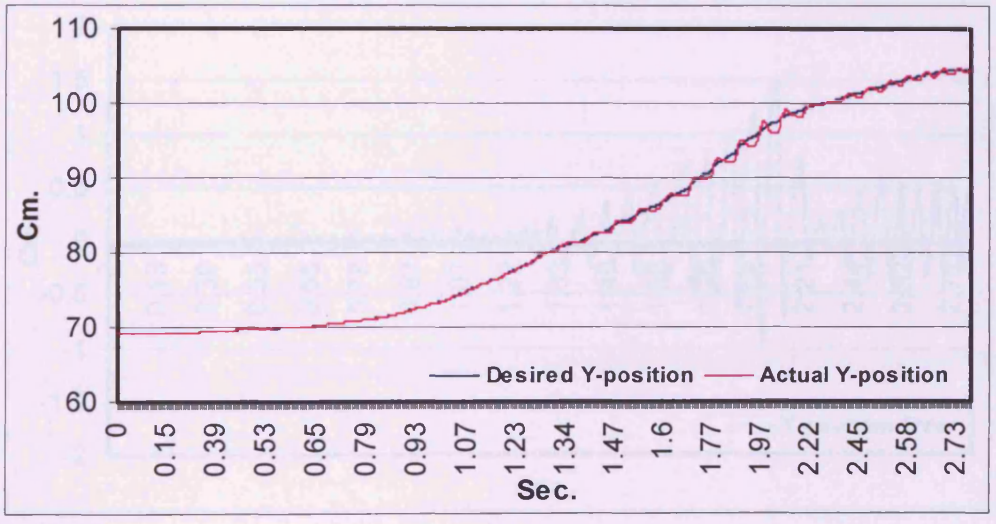


Figure (5.16). Upper-limb rehabilitation position trajectories tracking errors.





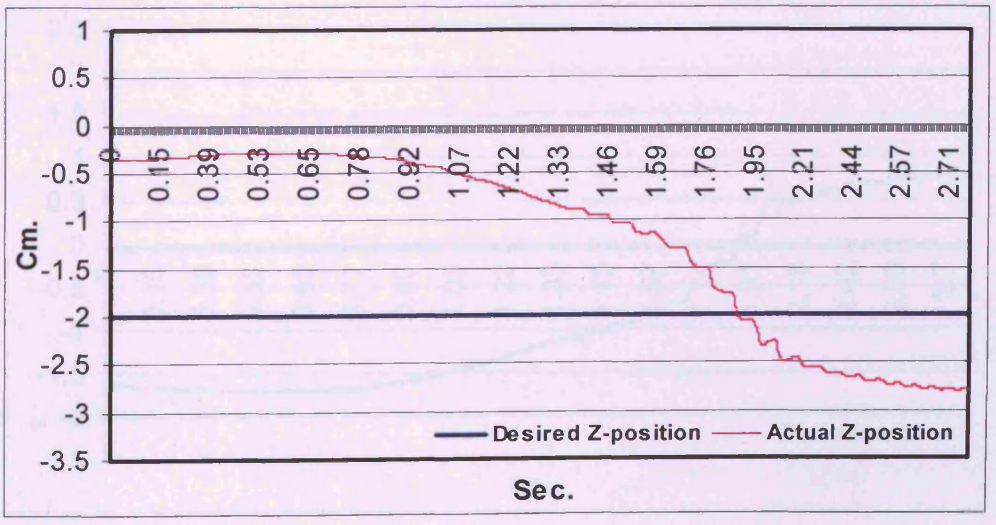


Figure (5.17). Upper-limb rehabilitation Cartesian trajectories tracking results

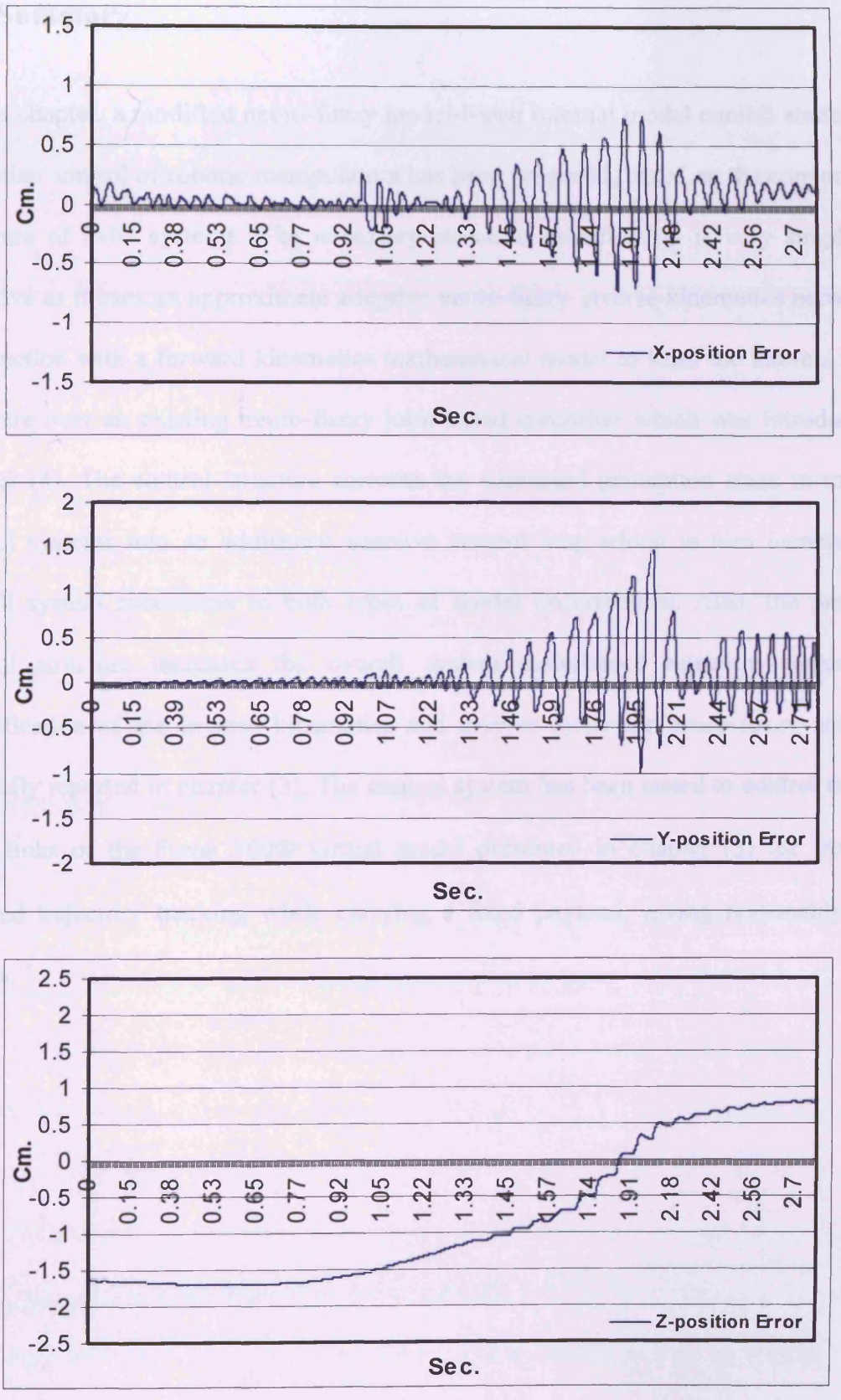


Figure (5.18). Upper-limb rehabilitation Cartesian trajectories tracking errors

5.7. Summary

In this chapter, a modified neuro-fuzzy model-based internal model control strategy for Cartesian control of robotic manipulators has been proposed, based on the conventional structure of IMC systems. The necessary structure modification is very simple and effective as it uses an approximate adaptive neuro-fuzzy inverse kinematics network in conjunction with a forward kinematics mathematical model to form the internal model structure over an existing neuro-fuzzy joint-based controller which was introduced in chapter (4). The control structure converts the command generation stage in robotics control systems into an additional adaptive control loop which in turn increases the overall system robustness to both types of model uncertainties. Also, the proposed control structure increases the overall system disturbance rejection capabilities. Identification of the inverse kinematics and inverse dynamics neuro-fuzzy networks was fully reported in chapter (3). The control system has been tested to control the first three links of the Puma 560® virtual model presented in chapter (3) for free pre-planned trajectory tracking while carrying a fixed payload, giving reasonably good results.

CHAPTER 6

Manipulators Position Coordination

Coordination of multi-robot systems has received extensive studies in the past decade. This is due to applications that require more than one robot manipulator to be performed like lifting heavy or awkwardly shaped object where independent manipulators controllers cannot be trusted to fulfill the task. Each robot controller will receive no information about the other, and any disturbance in one controller loop will cause an error that is corrected only by this controller loop, while the other controller loop will carry-on as before. This lack of coordination will cause an error in the overall task. Coordination between robot manipulators can be divided mainly into two groups, which are cooperation without interactions of forces between robots and cooperation with them [Osumi and Arai, 1994]. In the form of scheme categorization, there have been mainly three kinds of coordination schemes reported in the literature. The first scheme is the master/slave control where the motion of the master robot is pre-planned according to the desired motion of the manipulated object and the motion of the slave robot is to follow the master [Akella and Hutchinson, 2002]. Sometimes, the slave robot is position controlled with its desired trajectory is based on the actual position of the master robot and is modified in real time. To further enhance the master/slave position based scheme, relaxing of the grasp of the slave robot is used which basically results in its end-effector supporting the manipulated object rather than rigidly grasping it. Hence, any trajectory errors of either of the robots results in sliding of the manipulated object along the supporting end-effector of the slave robot if the object

support to the slave robot permits it. The hybrid position/force control method can also form part of a master/slave cooperative robotic system, where the master robot is position controlled and the slave robot is subject to compliant force control to maintain kinematic and force constraints. To further aid the HPFC method in minimizing trajectory following error, feedforward signals based on the object and the master robot position can be incorporated. HPFC schemes require appropriate force measurements at the end-effector of the robot. This not only results in the need for a force sensor of a suitable resolution to be attached between the robot and the load, but also additional hardware and software to interpret and transform the sensed value into a usable data format. Furthermore, incorporating the force data will increase the computational complexity. The second scheme utilizes centralized control architecture, in which robots and the grasped payload are considered as a closed kinematic chain. This method is designed based on a unified robot and payload dynamic model which is generally not easy to formulate. The third scheme is a decentralized control, in which each robot is controlled separately by its own local position controller, while installation of compliance devices, such as springs or free joints among robots is used to avoid excessive inner forces for the cooperative system [Osumi et al., 1997].

All of these coordination schemes considered the situation that the two manipulators are physically connected together, like grasping a common rigid payload, and employed complex setups of the hybrid position/force control architecture to overcome excessive inner forces between robots [Paljug and Yun, 1995; Subbarao et. al., 2001]. Actually, few of these coordination schemes can be applied to commercially available robots so far, this is due to the complex hardware and software setup of the control and

coordination strategy as explained. This situation gives another motivation for developing a simple coordination scheme specially to addressed the coordination problem when the robots are not kinematically constrained but perform a common task together such as one robot holding a payload while the other spreads adhesive on the edges, with both robots in motion simultaneously. In such cases coordination without interactions of forces is more realistic. Upper-limb rehabilitation, using two robot manipulators, can be viewed as an example for not kinematically constrained robots performing a common task (although it is forming a closed-chain kinematic system), where interacting forces between robots is not essential, due to the presence of the elbow joint, and can be disregarded in the controller design, while task planning is the most important issue while grasping the patient arm by the two robot manipulators. In this way, the upper-limb rehabilitation application using two-robots is similar to a closed-chain kinematic system with one free joint (elbow joint). Actually, position coordination between two-robots when there is a free joint between them, gives the designer a ready made solution that avoids excessive inner forces between the two robots [Osumi et. al., 1997; Tinos and Terra, 2002]. Also, for such an application the stability issue will not be so critical due to slow motion nature of the application.

In this chapter, a new coordination scheme for two position controlled manipulator system is developed by maintaining certain kinematic relationship between manipulators end-effectors using fuzzy motion synchronization to perform upper-limb rehabilitation application. The basic idea of the new coordination strategy is mainly to use the concept of motion synchronization. Since the problem of coordinating two manipulators is basically the problem of maintaining certain kinematic relationships

between them, it is of common practice to monitor and incorporate this kinematic relationship somehow in the control system [Sun and Mills, 2002]. The key to the success of the new method is to ensure that each manipulator tracks its desired trajectory while synchronizing its motion with the other manipulator motion so that the differential position error computed for the geometric connection-vector between the two manipulator end-effectors is reduced to zero or kept within low acceptable value. The proposed synchronization controller for each manipulator incorporates the cross-coupling technology into adaptive control architecture, by feeding back the differential geometric connection-vector position error in the control system. In a broad sense, cross-coupling control includes all control schemes that use feedback information from more than one control loop to control a composite error, which is normally calculated from individual loop errors, rather than individual loop error feedback. The use of the cross-coupling control in robotics was introduced by [Feng et al., 1993], where the differential velocity error of two driving wheels in a mobile vehicle was minimized through cross-coupled motion synchronization.

Implementation of this new coordination scheme is more straightforward and it is simple enough to synchronize any two kinematically constrained, physically not connected robots working together to perform certain task while the load on each robot is assumed to be within the capacity of these robots, which is typical the case of upper-limb rehabilitation application. The proposed control provides a unique advantage and opportunity for two-robot coordination by maintaining certain kinematic relationship without explicitly employing the hybrid position/force control amongst robots. Using this synchronization approach, manipulators are controlled in a synchronous manner so

that tracking errors and synchronization error converge to zero or to a very small value acceptable by the application nature.

The synchronization error is defined as a differential position error between the two manipulators end-effectors, and is used to evaluate the degree of coordination. The consideration of synchronization error in the proposed control design aimes to regulate robot trajectories in the transient stage which complies with the exercises execution nature of the rehabilitation application as a result of the sudden change in patient arm muscular resistance although the slow motion nature of the exercises. The significance of the proposed coordination scheme comes from:

- Incorporation of the differential position error into an adaptive architecture for two-robot control is relatively straightforward. There is no need to explicitly employ hybrid position/force control in the controller design.
- The controller being implemented using both adaptive neuro-fuzzy inverse kinematics and inverse dynamics robot controllers developed in previous chapters and is capable to sustain external force disturbances from patient arm.
- Position errors and synchronization error converging to zero or small value to be defined by the physiotherapist.

The remainder of this chapter is organized as follows. Section 6.1 presents the definition of the synchronization function. Section 6.2 presents the detailed structure of the proposed position coordinator for two robot manipulators. Section 6.3 explains the idea of implementing the proposed motion coordinator in the form of a fuzzy hysteresis

coordinator. Section 6.4 introduces the experimental set-up of the proposed coordination system over a simplified rapid prototype test-bench representing the upper-limb rehabilitation and presents the experimental results obtained. Finally, section 6.5 presents a summary for the chapter.

6.1. Synchronization Function

The problem of coordinating two robots is basically the problem of maintaining a kinematic relationship between them. Consider a robotic cell formed by two manipulators. Denote $x_i(t)$ as the Cartesian coordinates vector of robot manipulator *i*, where *i*=1 or 2. The position tracking error vector of the manipulator in following a desired position trajectory vector, $x_i^d(t)$, is given by:

$$e_i(t) = x_i(t) - x_i^d(t) \tag{6.1}$$

Consider that coordinated manipulators are subject to the following synchronization function, which defines the task supposed to be achieved:

$$f(x_i) = f(x_1(t), x_2(t)) = 0 \tag{6.2}$$

Assuming that the synchronization function is a linear function of variables $x_i(t)$ and is valid for all desired coordinates for the two robots.

$$f(x_i^d) = f(x_1^d(t), x_2^d(t)) = 0 \tag{6.3}$$

Using Taylor expansion, $f(x_i)$ can be expanded at the desired coordinates $x_i^d(t)$ as:

$$
\begin{aligned}
f(x_i) &= f(x_1(t), x_2(t)) \\
&= f(x_1^d(t), x_2^d(t)) + \frac{\partial f(x_i)}{\partial x_1}\Big|_{x_1^d}\left(x_1(t) - x_1^d(t)\right) \\
&\quad + \frac{\partial f(x_i)}{\partial x_2}\Big|_{x_2^d}\left(x_2(t) - x_2^d(t)\right) = 0
\end{aligned}
\tag{6.4}
$$

$$
\begin{aligned}
f(x_i) &= f(x_1(t), x_2(t)) \\
&= \frac{\partial f(x_i)}{\partial x_1}\Big|_{x_1^d}\left(e_1(t)\right) + \frac{\partial f(x_i)}{\partial x_2}\Big|_{x_2^d}\left(e_2(t)\right) = 0
\end{aligned}
\tag{6.5}
$$

Note that defining the synchronization function as a linear function of the variables $x_i(t)$ limits the order of the resulting Taylor expansion for it to be first order as all higher order derivatives of the series will equal to zero.

Example 1: Consider that a differential-drive mobile robot with two driving wheels tracks a curved path as shown in figure (6.1).

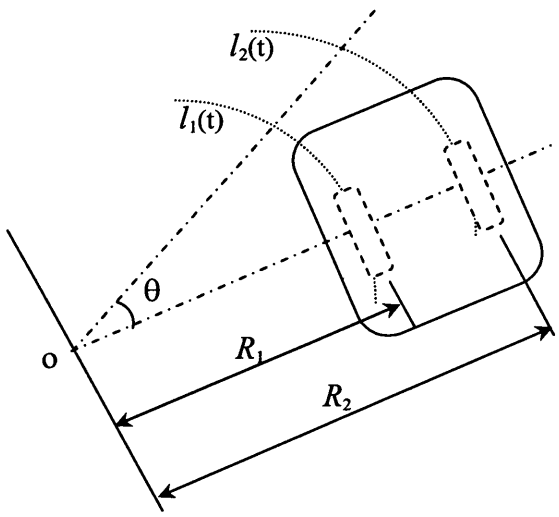


Figure (6.1). Mobile robot tracking a curved path.

Radii of the desired curves that the two driving wheels follow are denoted by $R_1(t)$ and $R_2(t)$, while the displacement of the two driving wheels denoted by $l_1(t)$ and $l_2(t)$, respectively. The two wheels displacements are subject to a synchronization function:

$$f(l_1(t), l_2(t)) = \frac{l_1(t)}{R_1(t)} - \frac{l_2(t)}{R_2(t)} = R_2(t) \times l_1(t) - R_1(t) \times l_2(t) = 0 \tag{6.6}$$

This synchronization function represents the condition which must be fulfilled to limit the orientation error to zero in order to sustain the desired curved path of the robot structure. According to equation (6.5), the above function is equivalent to causing the displacement errors $e_1(t)$ and $e_2(t)$ to satisfy:

$$\begin{aligned} f(l_i) = f(l_1(t), l_2(t)) &= \frac{\partial f(l_i)}{\partial l_1}\Big|_{l_1^d}(e_1(t)) + \frac{\partial f(l_i)}{\partial l_2}\Big|_{l_2^d}(e_2(t)) \\ &= R_2(t) \times e_1(t) - R_1(t) \times e_2(t) = 0 \end{aligned} \tag{6.7}$$

Example 2: Consider two robot manipulators holding a rigid object in a trajectory tracking task as shown in figure (6.2).

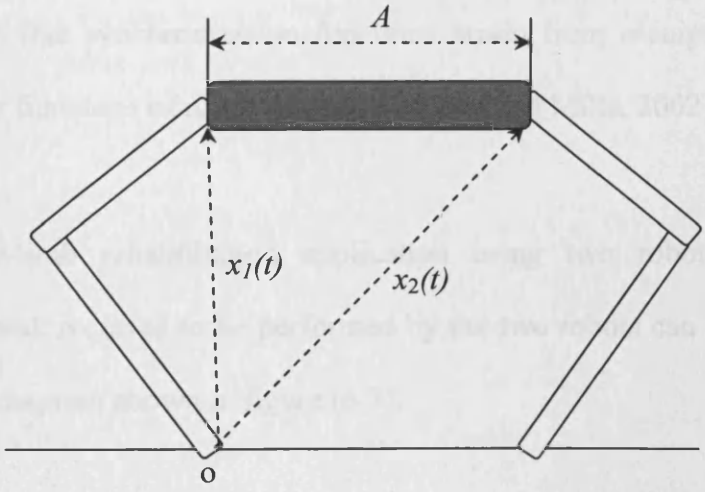


Figure (6.2). Two robot manipulators holding a rigid object.

Since it is a requirement that the difference between positions of the two end-effectors of the robots must remain constant in order not to damage the payload or robots, the position coordinates of the two manipulators end-effectors, denoted by $x_1(t)$ and $x_2(t)$, are subject to the synchronization function:

$$f(x_i) = f(x_1(t), x_2(t)) = x_1(t) - x_2(t) - A = 0 \tag{6.8}$$

where A is a constant vector of a magnitude equal to the effective rigid object length. According to equation (6.5), the above function is equivalent to causing position errors $e_1(t)$ and $e_2(t)$ to satisfy:

$$\begin{aligned} f(x_i) = f(x_1(t), x_2(t)) &= \frac{\partial f(x_i)}{\partial x_1}\Big|_{x_1^d}(e_1(t)) + \frac{\partial f(x_i)}{\partial x_2}\Big|_{x_2^d}(e_2(t)) \\ &= e_1(t) - e_2(t) = 0 \end{aligned} \tag{6.9}$$

Generally, synchronization functions may contain coordinate errors in the first order [i.e., $e_1(t)$ and $e_2(t)$] or of higher order [i.e., $e_1(t)\ e^T{}_1(t)$ and $e_2(t)\ e^T{}_2(t)$]. However, it is more common that synchronization functions arisen from manipulators coordination tasks are linear functions of robot coordinates [Sun and Mills, 2002].

For an upper-limb rehabilitation application using two robot manipulators, the rehabilitation task required to be performed by the two robots can be approximated by the schematic diagram shown in figure (6.3).

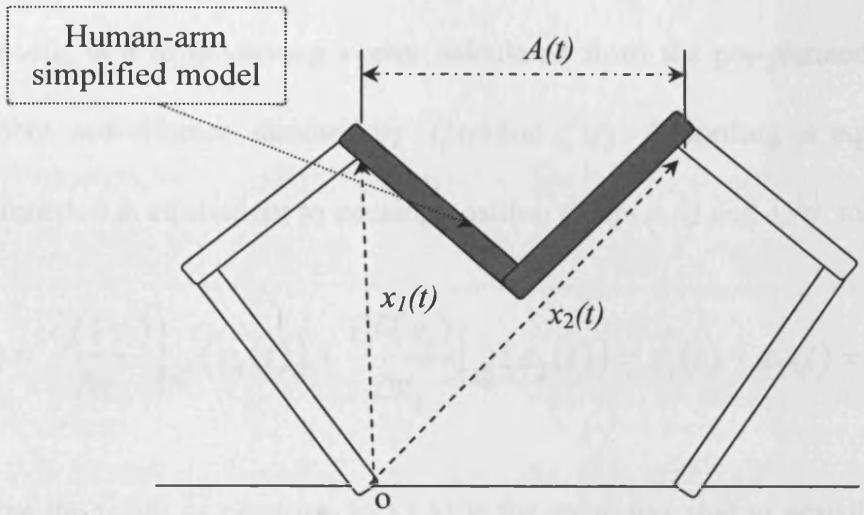


Figure (6.3). Two robot manipulators performing upper-limb manipulation.

The configuration of the arm is determined by the direction of the rotational axis for the elbow joint, the position of a point on that axis, and the angle between the two arm links. The position of the point can be computed by the joint angles of the manipulators if they are rigidly grasping the arm. As the motion is restricted to be in one plane, the direction of the rotational axis will remain fixed.

Since it is a requirement that the difference between position vectors of the two end-effectors of the robots to a common coordinates system must equal to the connection vector calculated from the pre-planned trajectories for each robot end-effector, the position coordinates of the two manipulators end-effectors, denoted by $x_1(t)$ and $x_2(t)$, are subject to the synchronization function:

$$f(x_i) = f(x_1(t), x_2(t)) = x_1(t) - x_2(t) - A(t) = 0 \qquad (6.10)$$

Where *A(t)* is a time-varying vector calculated from the pre-planned trajectories for each robot end-effector, denoted by $x_1^d(t)$ and $x_2^d(t)$. According to equation (6.5), the above function is equivalent to causing position errors $e_1(t)$ and $e_2(t)$ to satisfy:

$$f(x_i) = \frac{\partial f(x_i)}{\partial x_1}\Big|_{x_1^d}(e_1(t)) + \frac{\partial f(x_i)}{\partial x_2}\Big|_{x_2^d}(e_2(t)) = e_1(t) - e_2(t) = 0 \qquad (6.11.a)$$

Note that the result of equation (6.11.a) is the same like that of equation (6.9). This is due to the fact that the time-varying vector *A(t)* in equation (6.11.a) is calculated from the pre-planned robots end-effectors desired trajectories during the teach-in stage as explained in chapter (5), which do not depend on the current positions of the robots end-effectors, hence the partial derivative of vector *A(t)* with respect to either of the actual position coordinates of the two manipulators end-effectors equal to zero. Comparing this situation with the case of manipulating rigid object as in example 2, a small tolerance error ε above zero in equation (6.11.a) magnitude can be accepted as the manipulated object (human-arm) contains a free joint (elbow-joint) which prevents excessive forces from being transmitted from one robot to the other as indicated in equation (6.11.b). This idea results in accepting small error in the over all motion of the human arm due to the fact that the flexibility nature of the human arm tissues helps in absorbing such errors. The control of the synchronization error within this tolerance value aims to guarantee that no harmful twisting be applied to the human arm during trajectory execution.

$$|f(x_i)| = \left|\frac{\partial f(x_i)}{\partial x_1}\Big|_{x_1^d}(e_1(t)) + \frac{\partial f(x_i)}{\partial x_2}\Big|_{x_2^d}(e_2(t))\right| = |e_1(t) - e_2(t)| \le \varepsilon \qquad (6.11.b)$$

6.2. Proposed Coordinator Structure

The proposed motion synchronization controller can be regarded as comprising three main components. The first component is the joint-based controller, which controls the motion of each robot joints as explained in chapter (4). The second component is the Cartesian trajectory interpolator, which utilizes the inverse kinematics neuro-fuzzy network to generate the desired joint trajectories for a given desired Cartesian path for each robot as explained in chapter (5). The third component is the motion geometry controller (coordinator) which uses motion synchronization to coordinate the motion of the two robots. The motion synchronization system consists of:

- A hysteresis controller which is used to monitor the connection-vector between the two robots end-effectors and gives a signal whenever this vector violates a certain pre-defined tolerance value ε.

- Error mapping and decision-making logic which works to transform the error in the connection-vector into trajectory compensation signal to be fed to either or both of the two robots reference trajectories.

Actually, the compensation signal can be added to the robots local controllers output in the form of an additional control signal to the robot or in another way, it can be added to the robots local controllers inputs in the form of increased/decreased error (or increased/decreased reference input) as will be used here. The direct modification of the controllers' reference command is straightforward method which does not involve

changing the system configuration which is of great importance in our case to keep the internal model structure valid [Moore and Chen, 1995; Verdonck and Swevers, 2002]. The trajectory modification signal for each robot although depends on the synchronization error, it also depends on that of the other robot as will be explained in the following sections.

Figure (6.4) shows the general structure of the proposed control and synchronization system for the two robot arms. The neuro-fuzzy Cartesian controller explained in chapter (5) for each robot arm is minimized in one block to simplfiy the block diagram and to clarify the synchronization part of the overall controller.

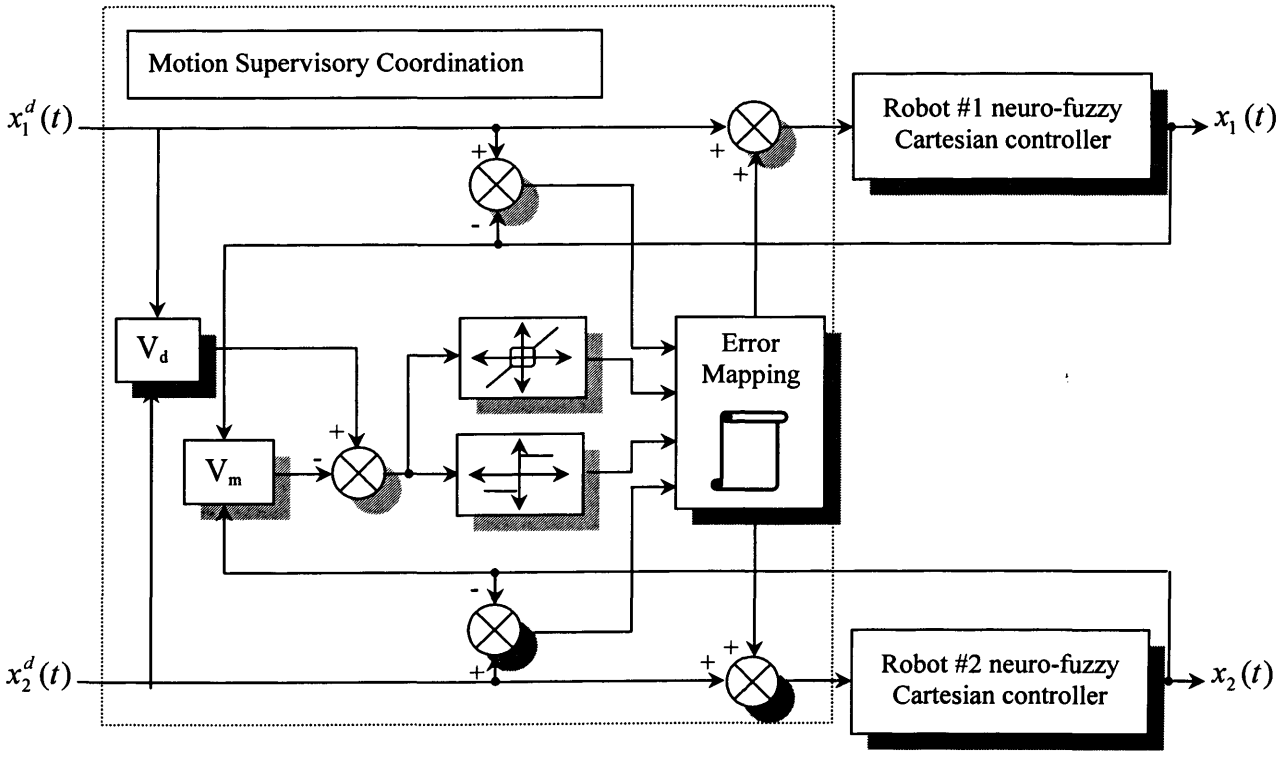


Figure (6.4). Structure of the proposed control and synchronization system.

6.2.1. Synchronization Error Controller

The design issue here is how to map the measured synchronization error vector to the demand position compensation vector of each robot so that the synchronization can be controlled as accurately as possible. The mapping rules between synchronization error and corrective actions are heuristically constructed from the commanded inputs and measured responses with the main objective of forcing the synchronization error to lie within the acceptable tolerance. The first component in the proposed coordinator is simply a sign generator which gives +1 for positive synchronization errors and a -1 for negative errors. The main component in the proposed coordinator is the hysteresis controller which is used to monitor the synchronization error and generates a switching signal for the error mapping mechanism to calculate the modifications required. The input-output characteristic of the hysteresis controller is as shown in figure (6.5). The width of the hysteresis loop, denoted by (ε), which represents the tolerance bandwidth for the synchronization system to interfere in the control system for modification. If the synchronization error is below this value, then there is no vital need for the synchronization system to interfere in any of the robots controllers and each controller is supposed to cover this error alone or even the system overall performance will not be affected by this error. This method ensures that the coordination controller is delayed behind each robot controller to allow the robot controller first to compensate for the generated error. In other words, the coordination controller is operating only when any of the robots controllers fails to quickly compensate for its own generated error and works to speed-up this compensation process.

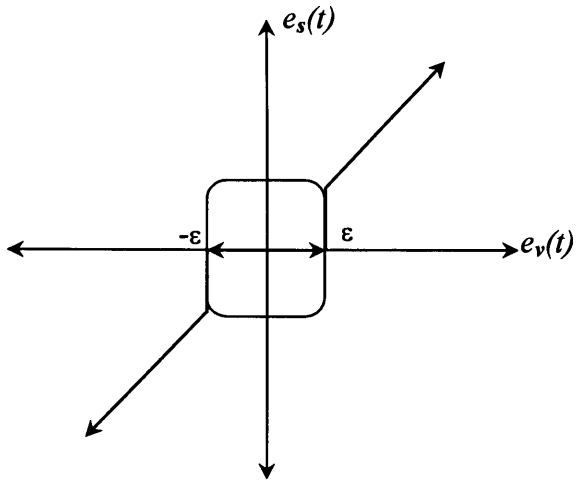


Figure (6.5). Hysteresis controller input/output characteristics.

The tolerance bandwidth of the hysteresis controller is to be designed according to the safety limits provided by the physiotherapist for the patient upper-limb allowable torsion and/or exercises trajectories violation allowance.

6.2.2. Error Mapping Look-up Table

This is the part responsible for transferring the synchronization error to a compensating signal to be added to either of the robot arms kinematics control part. The main strategy depends on the most significant error concept, which is defined as the error which has the largest impact on the motion accuracy at this moment.

The position tracking error $e_i(t)$ of the i^{th} manipulator in following position trajectory, $x_i^d(t)$, is given by equation (6.1), where i = 1 or 2. Although the independent adaptive control of each robot without synchronization ensures that position errors of each robot converge to zero and eventually the synchronization error converge to zero, the

proposed synchronization control aims to improve the transient performance of the system specially when a sudden or large change in the patient arm resistance occurs.

$$e_i(t) = x_i^d(t) - x_i(t) \tag{6.12}$$

where $x_i(t)$ is the end-effector Cartesian position of robot *i*, $x_i^d(t)$ is the reference Cartesian position of robot *i* end-effector. The reference connection-vector trajectories are calculated from the original values of both robots reference trajectories recorded during the teach-in process. Also, the actual connection-vector trajectories are calculated online from the forward kinematics positioning of the end-effector of each robot as shown in equations (6.13), (6.14) and (6.15).

$$s^d(t) = x_1^d(t) - x_2^d(t) \tag{6.13}$$

$$s(t) = x_1(t) - x_2(t) \tag{6.14}$$

$$e_s(t) = s^d(t) - s(t) \tag{6.15}$$

where $s^d(t)$ forms the reference connection-vector trajectories, $s(t)$ forms the actual connection-vector trajectories, and $e_s(t)$ is the synchronization error. The error mapping main function is to judge which robot is the one which is experiencing difficulty in following its predefined trajectories at this moment and affecting the overall motion of the manipulated object. By monitoring the present values of the synchronization error and both of the robots end-effectors Cartesian position errors, a decision on which robot to be helped to improve its response is taken according to table (6.1).

$e_s(t)$	$e_1(t)$	$e_2(t)$	$r_{1m}(t)$	$r_{2m}(t)$
+ve >ε	+ve	+ve	$e_s(t)$	zero
+ve >ε	+ve	zero	$e_s(t)$	zero
+ve >ε	-ve	-ve	zero	$-e_s(t)$
+ve >ε	zero	-ve	zero	$-e_s(t)$
-ve <-ε	+ve	+ve	zero	$e_s(t)$
-ve <-ε	-ve	-ve	$-e_s(t)$	zero
-ve <-ε	-ve	zero	$-e_s(t)$	zero
-ve <-ε	zero	+ve	zero	$e_s(t)$
+ve >ε	+ve	-ve	$e_s(t)$	$-e_s(t)$
-ve <-ε	-ve	+ve	$-e_s(t)$	$e_s(t)$

Table (6.1). Error mapping and corresponding compensating signals.

If one examines the first row in table (6.1), it implies that, *if* the synchronization error is positive *and* exceeded the tolerance value (ε), *and* both of the robots errors are positive *then*, robot #1 error forms the most significant error. In this case a positive torque or input reference compensation signal is to be added to robot #1 controller. Also, by examining the last row in table (6.1), it implies that, *if* the synchronization error is negative *and* exceeded the tolerance value (ε), *and* robot #1 error is negative, *while* robot #2 error is positive *then*, it is not certain which robot forms the most significant error. A solution for this case is to add a negative torque or reference compensation signal to robot #1 and a positive compensation signal to robot #2. By following this intuition, a total of 10 rules can be generated for the motion coordinator. In this way, a compensation signal will be added to the main controller of the robots to force the synchronization error to be maintained within the pre-specified tolerance value.

6.3. Fuzzy Hysteresis Coupling Coordinator

The above mentioned look-up table can be transformed to form a fuzzy hysteresis coordination system by assigning specific shape membership functions. Figure (6.6) illustrates the suggested input/output membership functions characteristics.

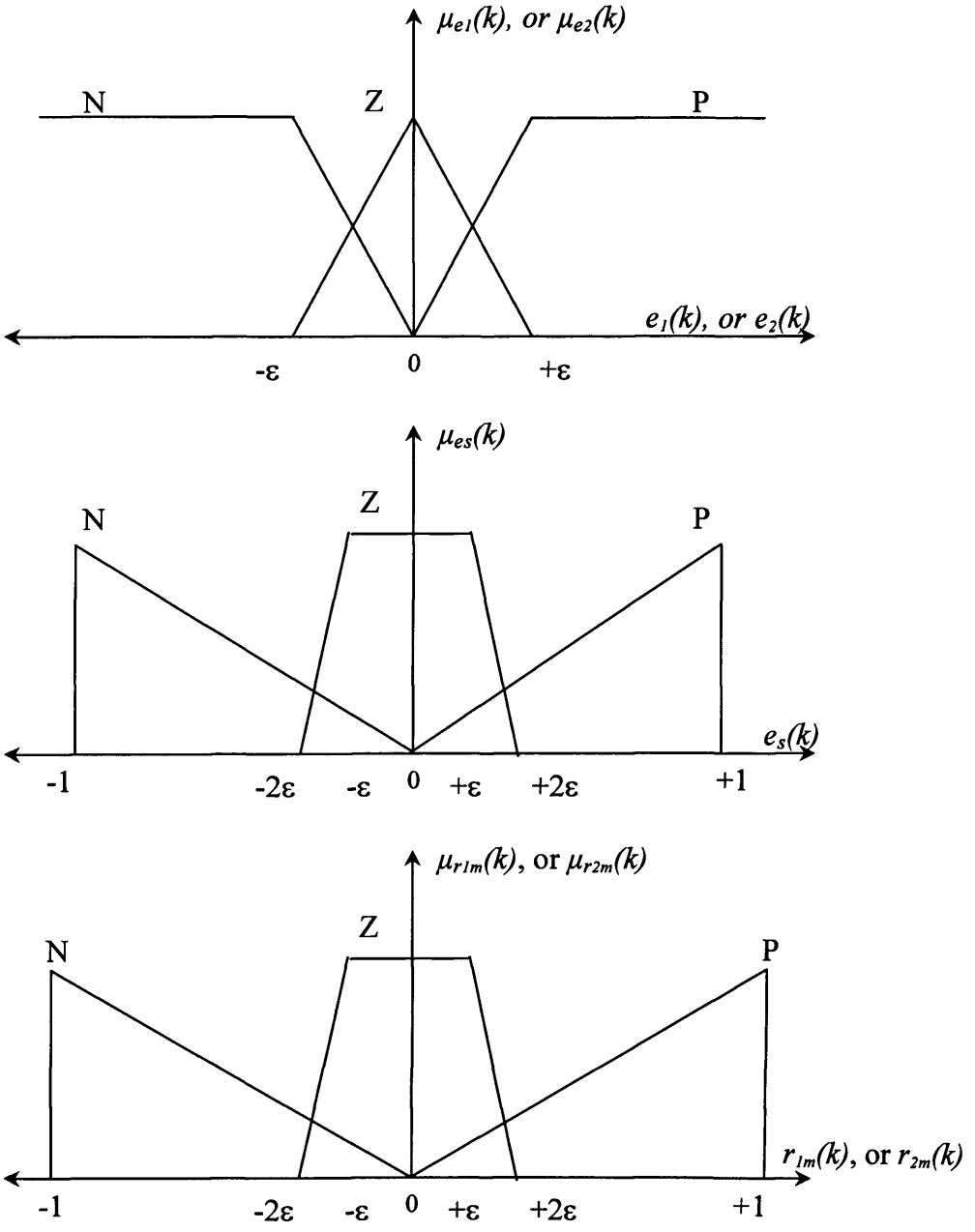


Figure (6.6). Input/output membership functions for the fuzzy hysteresis coordinator.

By assigning these membership functions, Table (6.1) can be re-written in the form of fuzzy hysteresis coordinator rules as listed in table (6.2).

No.	$e_s(t)$	$e_1(t)$	$e_2(t)$	$r_{1m}(t)$	$r_{2m}(t)$
1	P	P	P	P	Z
2	P	P	Z	P	Z
3	P	N	N	Z	N
4	P	Z	N	Z	N
5	N	P	P	Z	P
6	N	N	N	N	Z
7	N	N	Z	N	Z
8	N	Z	P	Z	P
9	P	P	N	P	N
10	N	N	P	N	P
11	Z	-	-	Z	Z

Table (6.2). Fuzzy hysteresis coupling rules.

Note that there is another rule added in this case (rule No.11) to represent the case when the synchronization error lies inside the tolerance band. Addition of this rule is essential and helps to smooth the output characteristics of the fuzzy coordinator. The COA defuzzification method [Runkler, 1997] is used to generate the crisp modification signals for each robot reference input from the fuzzy output in order to have smoothly varying modification signals while the synchronization error is in the range from $\pm\varepsilon$ to $\pm 2\varepsilon$ as specified in the input/output membership functions.

6.4. Experimental Coordination between Two SCARA® Type Robots

In order to test the coordination scheme, an experimental test for the proposed control and coordination strategy over a simplified real-time test-bench for the upper-limb rehabilitation cell formed by two 2-link similar planner robots linked to a simple upper-limb model developed in the intelligent systems laboratory as shown in figure (6.7).

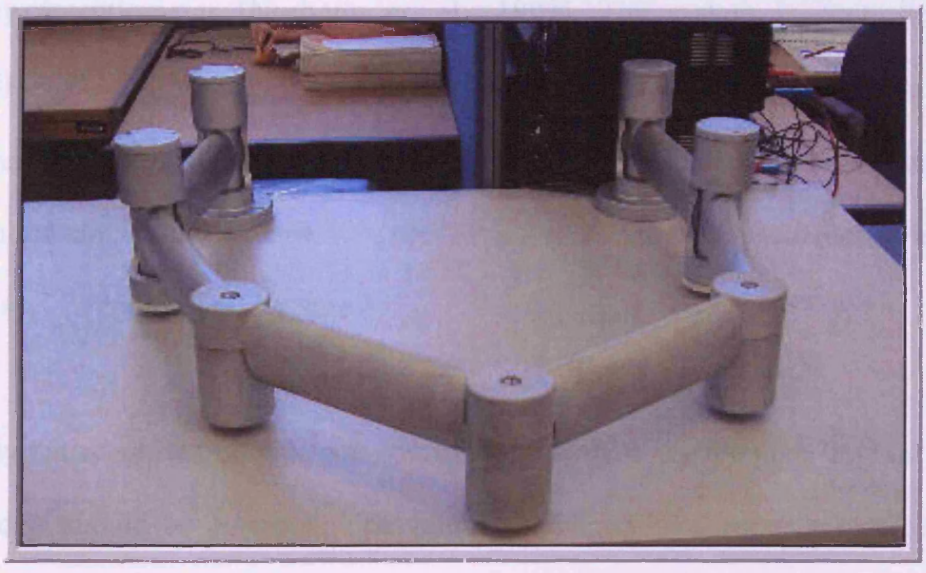


Figure (6.7). Experimental setup formed by two 2-link SCARA® type robots.

The mechanical structures of the robot's links and the arm simplified model were manufactured using Rapid-Prototyping facilities available in the Manufacturing Engineering Centre, Cardiff School of Engineering. This facility allows the transformation of the generated 3-D CAD model created for the parts (or even the assembly) by "*Pro/Engineer®*" to be rapidly manufactured from a selection of plastic or metallic powders using Selective Laser Sintering (SLS) machines. The manufactured parts possess acceptable tolerances in model final dimensions after cooling down.

6.4.1. Experimental Setup

Each robot link is then fitted with suitable bearing and powered by a high-torque compact frame D.C. motor with planetary reduction gear-head by which high-torque to weight and inertia ratios and virtually zero backlashes are achieved. Two different gear-ratio gear heads of 224 and 111 were used for link1 and link2 drive motors in each robot respectively. The angular displacement of each joint is measured by a high accuracy potentiometer [Norberto et. al., 1997]. The system is controlled by an ADLINK® DAQ/PXI-2501 PC general purpose interface card plugged in the host computer. Appendix (C) summarizes the technical specification for the joint motors, position sensors, interface card, ... , etc. Figure (6.8) shows the schematic view of the overall system control architecture.

To apply the suggested control and coordination system, an adequate amount of input-output data should be obtained. The validity, accuracy, and robustness of the model depend on the experiment and the input-output data extracted from it. The main target is to construct the inverse kinematics and inverse dynamic model for each of the 2-link robot arms. The same procedure described in chapter (3) was used. A simple feedback proportional controller was designed first for each link to perform stable behavior along different trajectories. A collection of input-output data was obtained through a wide range of joint movement. In order to ensure that the excitation during the identification experiment covers the entire applicable range of system input/output variables, the experiment should excite most of the modes of the system that may be excited when the

model is used. For robot manipulators, random joint trajectories that cover the desired range of input/output parameters are considered proper and sufficient input signal.

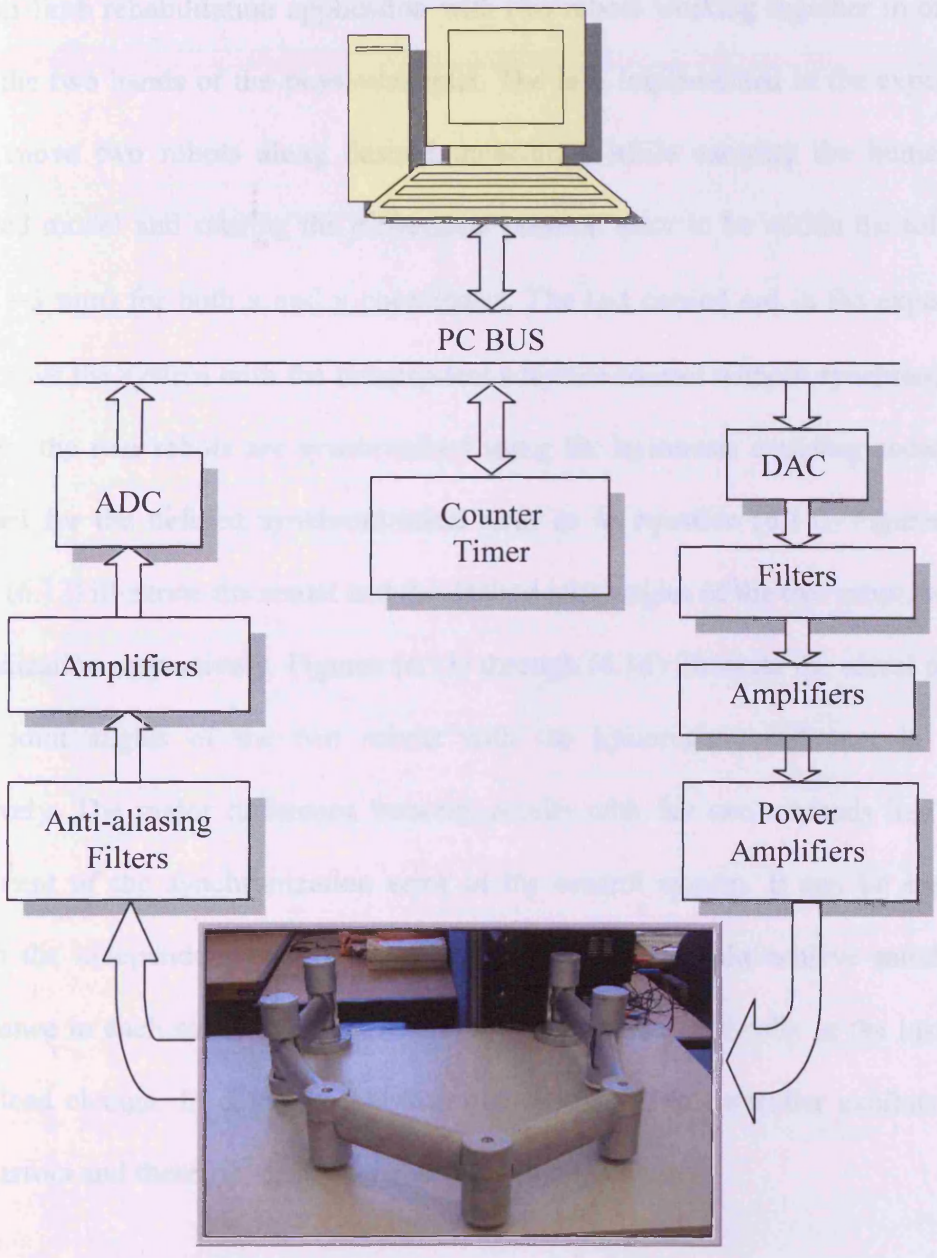
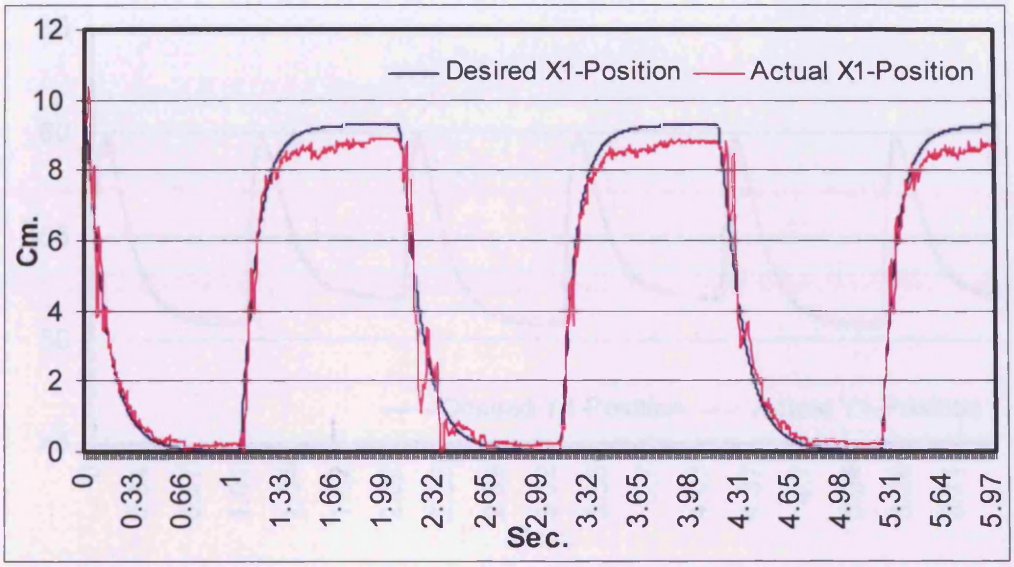


Figure (6.8). Experimental overall system control architecture.

6.4.2. Experimental Results

The neuro-fuzzy Cartesian control system presented in chapter (5) is used to control each robot in addition to the proposed synchronization system in order to implement the upper-limb rehabilitation application with two robots working together in order to imitate the two hands of the physiotherapist. The task implemented in the experiment was to move two robots along desired trajectories while carrying the human arm simplified model and causing the differential position error to be within the tolerance band (ε =3 mm) for both x and y coordinates. The test carried out in the experiment was to move the system with the independent adaptive control without synchronization first then, the two robots are synchronized using the hysteresis coupling coordinator developed for the defined synchronization error as in equation (6.15). Figures (6.9) through (6.12) illustrate the actual and the desired joint angles of the two robots without synchronization respectively. Figures (6.13) through (6.16) illustrate the actual and the desired joint angles of the two robots with the hysteresis coordinator in action respectively. The major difference between results with the two methods lies in the involvement of the synchronization error in the control system. It can be seen that although the independent control without synchronization could achieve satisfactory performance in each robot tracking, it exhibits large errors especially at the instant of sudden load change. In contrast, the proposed synchronized controller exhibits much smaller errors and therefore exhibits better coordination ability.



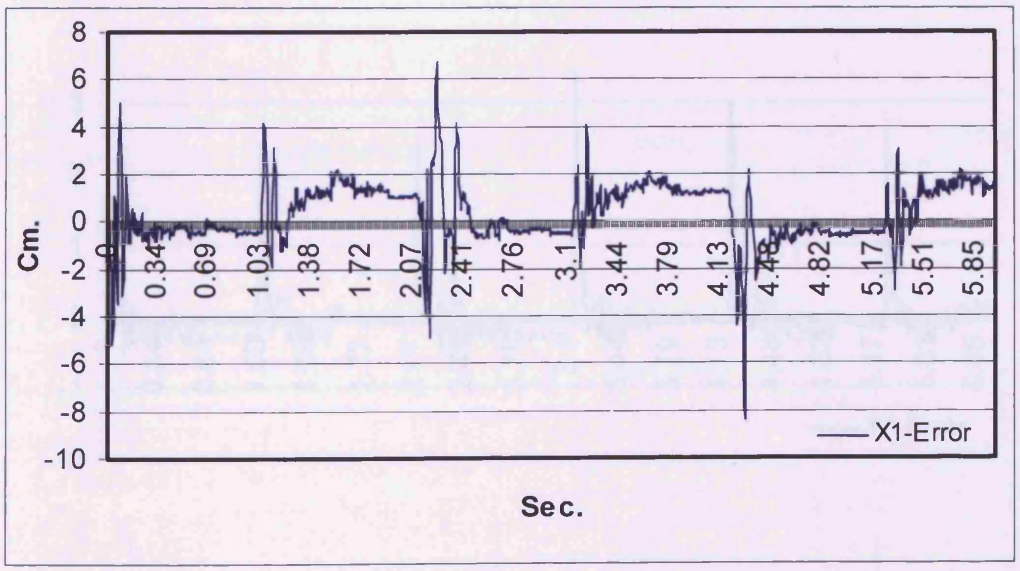
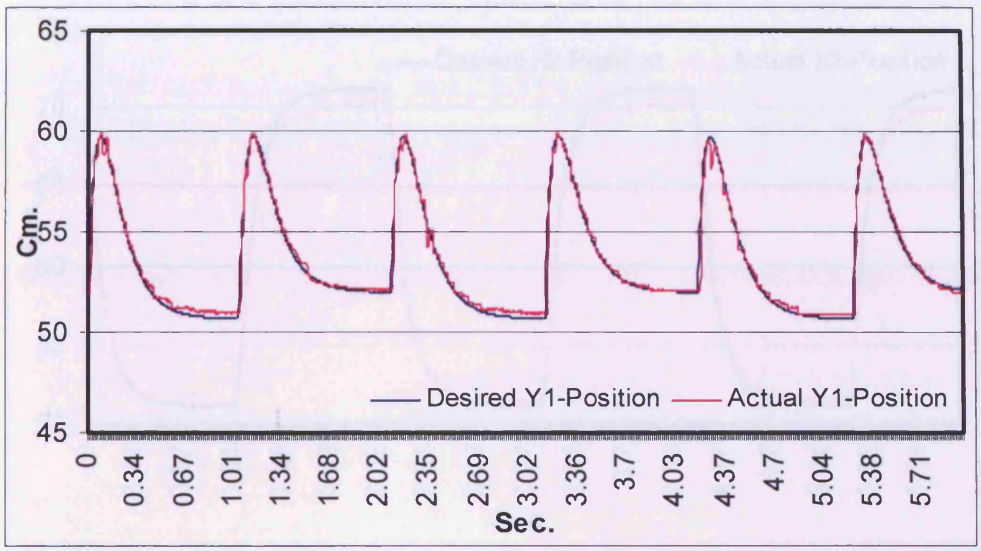


Figure (6.9). Robot#1 X-coordinate trajectory without coordination.



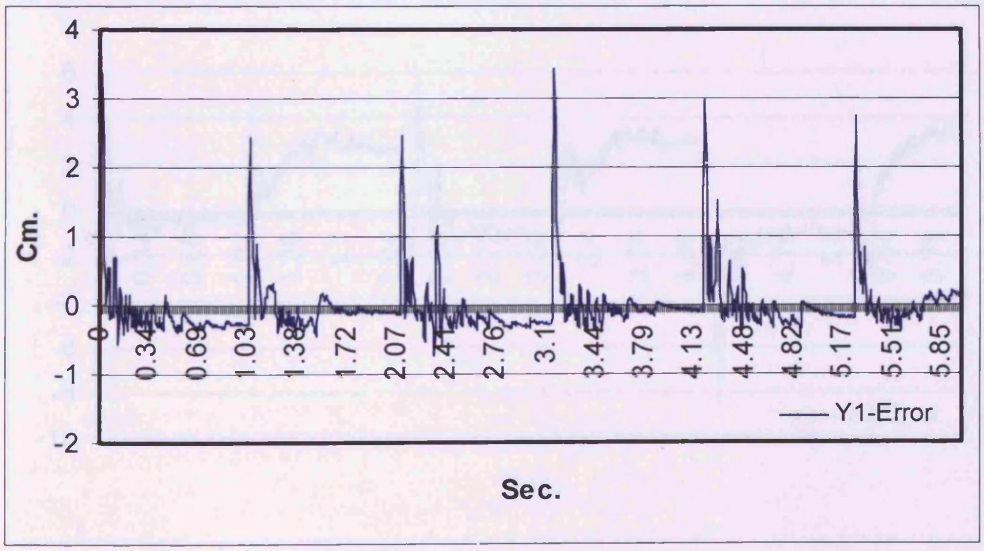
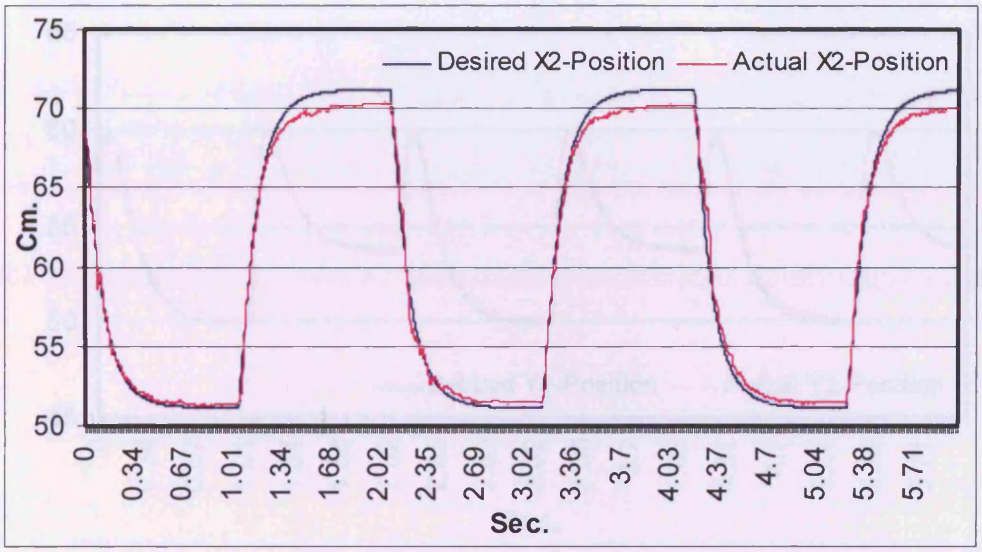


Figure (6.10). Robot#1 Y-coordinate trajectory without coordination.



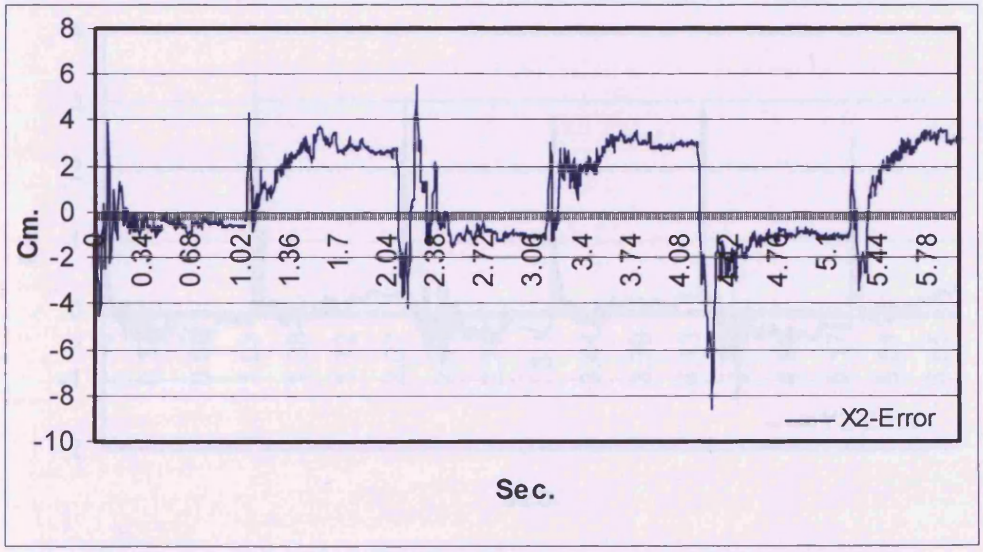
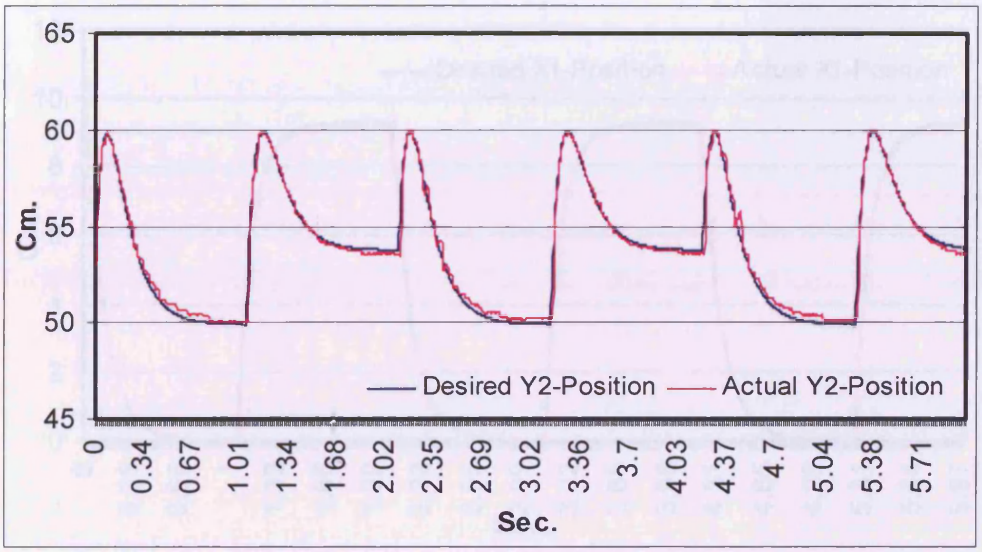


Figure (6.11). Robot#2 X-coordinate trajectory without coordination.



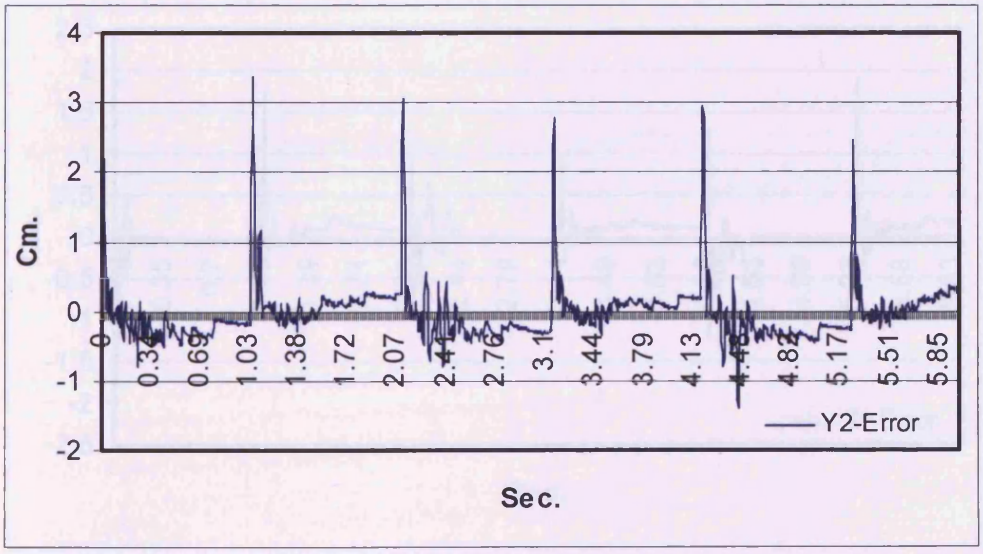
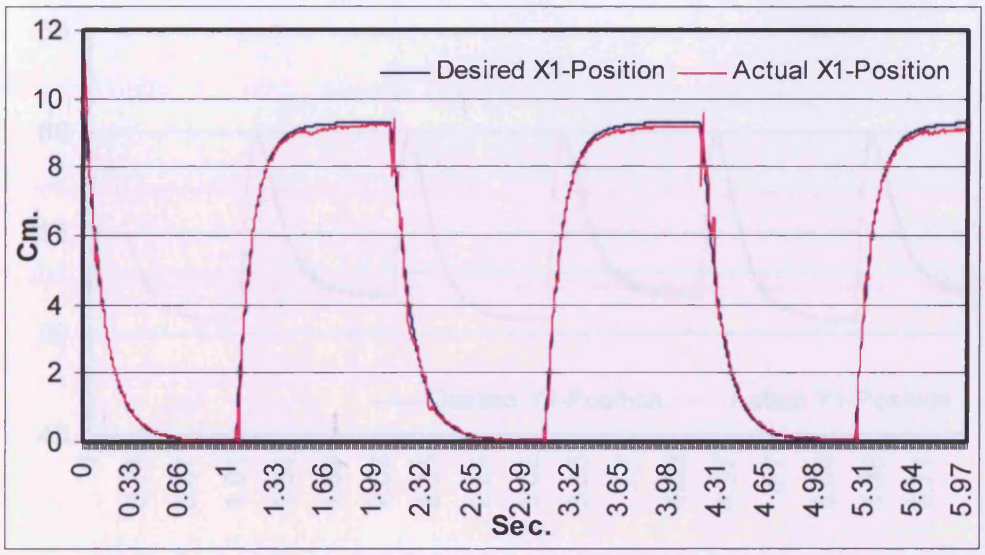


Figure (6.12). Robot#2 Y-coordinate trajectory without coordination.



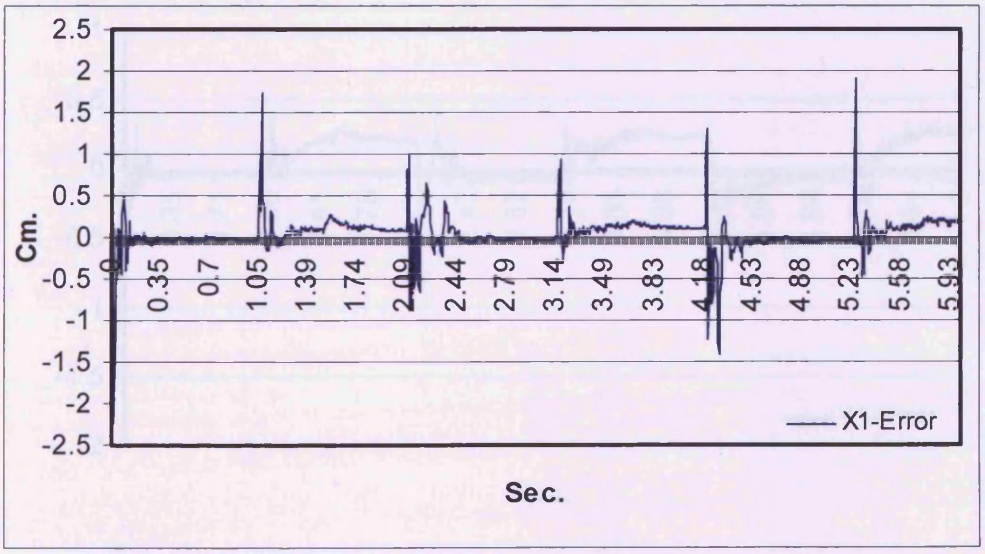
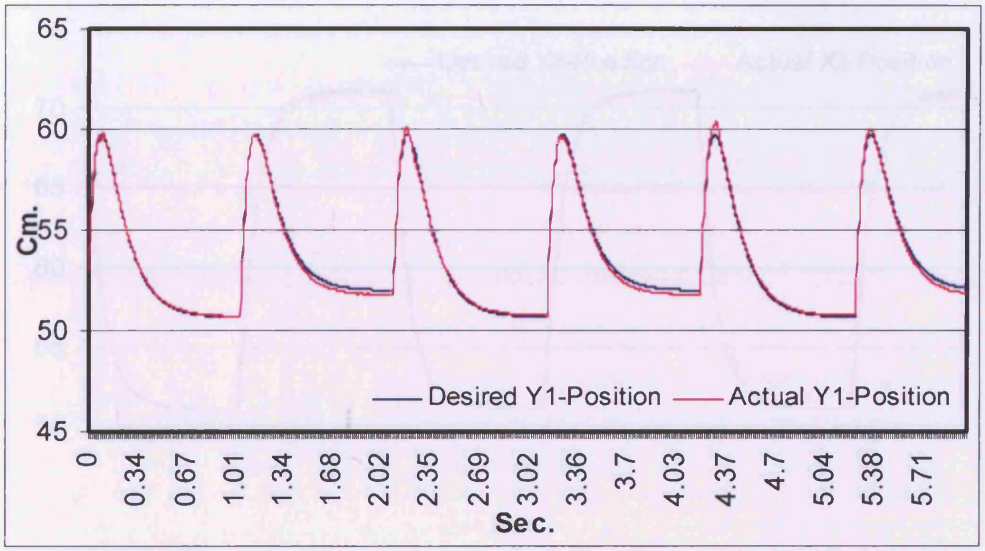


Figure (6.13). Robot#1 X-coordinate trajectory with coordination.



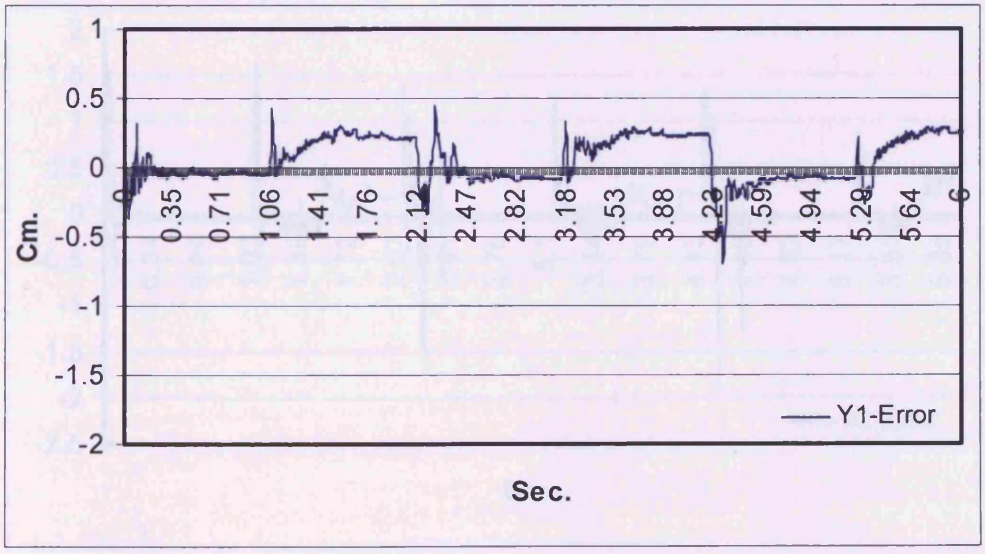
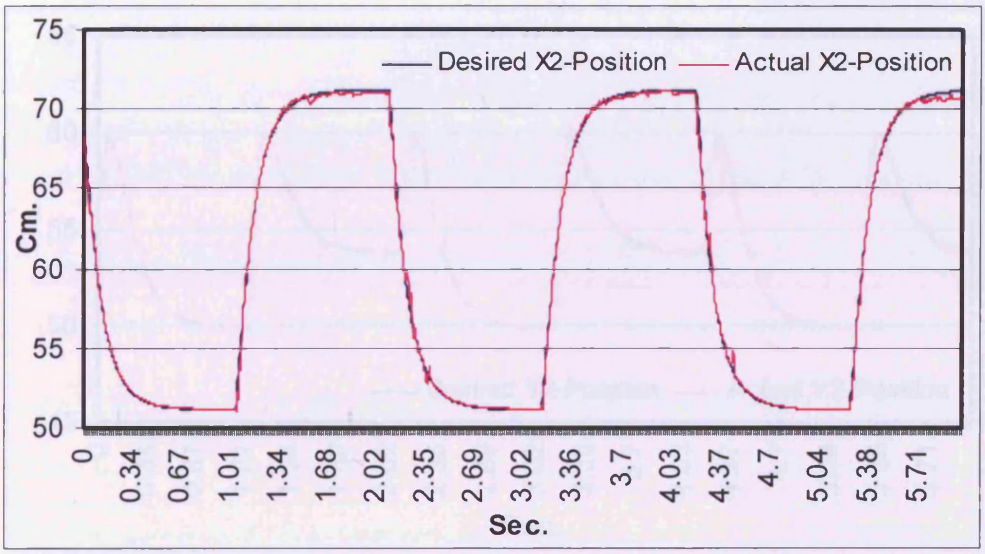


Figure (6.14). Robot#1 Y-coordinate trajectory with coordination.



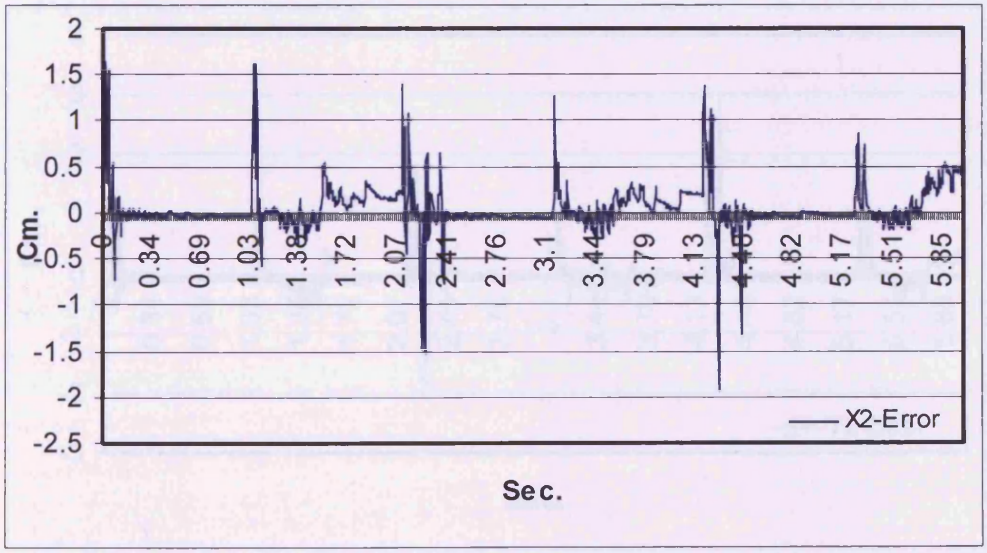


Figure (6.15). Robot#2 X-coordinate trajectory with coordination.

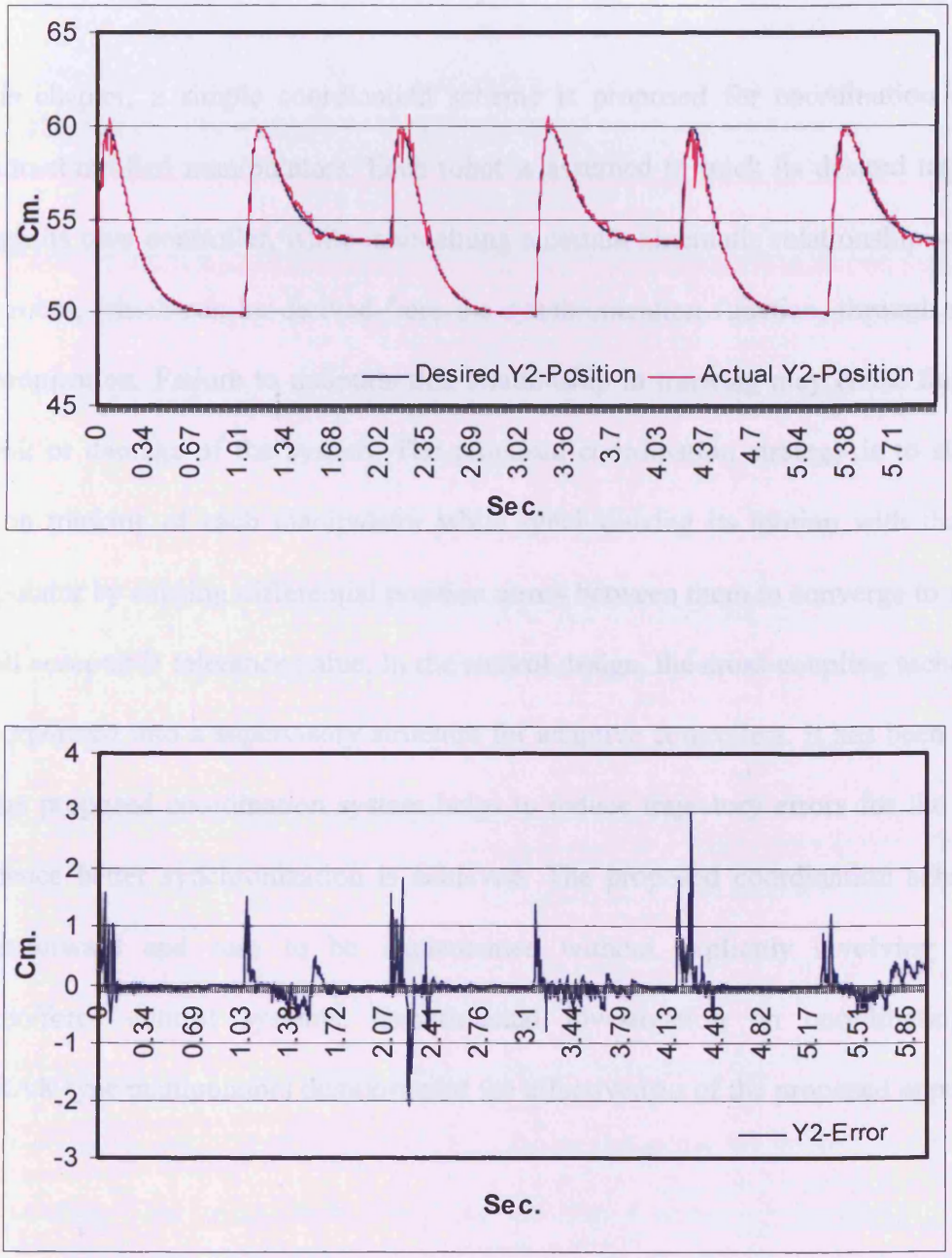


Figure (6.16). Robot#2 Y-coordinate trajectory with coordination.

6.5. Summary

In this chapter, a simple coordination scheme is proposed for coordination of two position-controlled manipulators. Each robot is assumed to track its desired trajectory through its own controller, while maintaining a certain kinematic relationship with the other robot, which can be derived from the synchronization function, through motion synchronization. Failure to maintain this relationship in tracking may cause failure of the task or damage of the system. The proposed coordination strategy is to stabilize position tracking of each manipulator while synchronizing its motion with the other manipulator by causing differential position errors between them to converge to zero or a small acceptable tolerance value. In the control design, the cross-coupling technology is incorporated into a supervisory structure for adaptive controllers. It has been shown that the proposed coordination system helps to reduce trajectory errors for the robots and hence better synchronization is achieved. The proposed coordination scheme is straightforward and easy to be implemented without explicitly involving hybrid position/force control systems. Experimental investigation on coordinating two SCARA® type manipulators demonstrated the effectiveness of the proposed approach.

CHAPTER 7

Conclusions and Future Work

The overall objective of this work is to design and develop intelligent neuro-fuzzy adaptive control systems for industrial robot manipulators using machine learning techniques (MLT), fuzzy logic controllers (FLC), and fuzzy neural networks (FNN). The main target is to integrate these techniques in a systematic manner to achieve adaptive robot manipulator control. This control system is coordinated for two robot manipulators to produce a work cell capable of performing upper-limb rehabilitation. The remainder of this chapter is organised as follows. Section 7.1 reviews the main Contributions of the thesis. Section 7.2 lists the conclusions of the thesis. Section 7.3 presents suggestions for future investigations.

7.1. Contributions

1. Development of a systematic robotic inverse dynamics and inverse kinematics modelling technique based on machine learning technique for automatic fuzzy rule generation from observation data. The developed technique introduces a fully differentiable fuzzy neural network termed *dynafuzznn* to achieve online adaptation of the developed models. The final result is a systematic neuro-fuzzy inductive learning algorithm that integrates the capabilities and performance of a good inductive learning algorithm with the ability to create accurate and compact neuro-fuzzy models.

2. Development of an adaptive neuro-fuzzy joint-based robotic control technique. This control technique uses the inverse dynamics FNN developed as a feedforward controller that compensates for the dynamics interactions of the robot structure in addition to a feedback Fuzzy-PID-like incremental servo-controller for each robot link. A feedback-error learning scheme is applied to provide an online adaptation mechanism for the proposed controller. This scheme ensures that online training will stop only when the feedback error reduces to zero. This behaviour resembles the integration action in a classical integral controller.

3. Development of an adaptive neuro-fuzzy Cartesian internal model control technique for robotic manipulators. The suggested control technique utilizes the neuro-fuzzy kinematic model of the robot arm in addition to the joint-based control structure proposed and the forward mathematical model of the robot arm in an adaptive internal model controller structure to achieve an adaptive form of robot Cartesian control. The suggested IMC structure can be regarded as an adaptive form of a *command generator* for the existing neuro-fuzzy joint-based robot controller by introducing the neuro-fuzzy inverse kinematics network outside the control loop which achieves compensation for robot Cartesian uncertainties by modifying the desired input Cartesian trajectory. The feedback error learning scheme is extended to include the IMC controller.

4. Development of a simple coordination scheme for two position-controlled robot manipulators. The coordination scheme is based on maintaining certain kinematic relationships between the two manipulators using reference motion synchronisation.

The coordination strategy is based on allowing each manipulator to follow its desired trajectory using its own controller while synchronising its motion with the other robot manipulator's motion so that the differential position error between the two manipulators end-effectors is reduced to zero or kept within acceptable limit.

7.2. Conclusions

1. Recent developments in intelligent algorithms such as machine learning techniques and neuro-fuzzy systems can result in a systematic modelling and control techniques which can be applied for complex systems such as robotic manipulators. Efficient application and integration of these algorithms results in compact and adaptable mathematical-model free control techniques capable of updating its parameters online to cop with the varying unstructured dynamics in robotic manipulators operating with unmodelled loads. By integrating these algorithms, a fuzzy neural network termed *dynafuzznn* is developed that can be used to replace any complex block in the control system effectively. The final result is an efficient and simple control system.

2. Fuzzy systems can be used to perform feedback control application effectively. A fuzzy-PID-like incremental servo-controller can be regarded as an online nonlinear stabilizer for a nonlinear plant. Also, it can be regarded as a nonlinear learning signal for an adaptive neuro-fuzzy feedforward control system by applying the feedback-error learning scheme. In this way, the learning signal will reduce to zero only when the feedforward controller outputs converge to the desired control

actions. Using nonlinear learning signals for control application of nonlinear systems is more realistic than using linear learning signals.

3. Cartesian control of robotic manipulators, although being the main target for any control application, is a tedious target when implemented using mathematical techniques as a result of computing inverse kinematics. Internal model control technique provides high disturbance rejection and low sensitivity to model uncertainties capabilities. Integration of intelligent techniques, which is capable of adapting its parameters to unstructured dynamic variations, with the internal model control technique results in an efficient model-free Cartesian control system for robotic manipulators. The internal model control structure can be applied as an adaptive form of a *command generator* for an adaptive neuro-fuzzy joint-based robot controller. This simplifies the implementation of Cartesian control for robotic manipulators.

4. Coordination between two position-controlled robot manipulators is not an easy task due to interactions between the manipulators. Generally, complex hybrid position/force control techniques are used which are very difficult to be implemented in practice. In some applications, interacting forces between robots are not important while task planning is the main problem. For such applications, by maintaining certain kinematic relationships between the two manipulators' end-effectors, coordination can be much simpler and effective, especially when the robots are controlled by intelligent adaptive controllers.

7.3. Further Work

1. Further research could be conducted to automate the creation of the output membership functions in the rule generation part of the modelling process to obtain the optimum number and shape of the output membership functions. This automation could be based on clustering techniques in order to reduce or enlarge the membership functions in areas of the target output space where more or less precision is required.

2. The selection of model variables from the available data is another problem to be investigated. The training data should reflect all the system dynamics during normal operation and cover the whole operation range. It is not certain how to detect which parts of the data satisfy these conditions. Therefore, further research could be conducted to automate the selection of input variables to be used in the model between all past and present values of position, speed, acceleration, and torque variables collected during the data collection test. Again this automation could be based on clustering or data-mining techniques in order to select the most dominant variables affecting the target output.

3. The proposed adaptation method tunes only the parameters of the neuro-fuzzy network online. No modification is carried out for the model structure, in other words, the rules generated in the offline stage are fixed during operation. Consequently, the offline method used for fuzzy rule extraction could be modified or integrated with other techniques so that the rule base could be also upgraded

online by generating new fuzzy rules or pruning the existing rules that are not frequently used during different control tests.

4. The model generated input membership functions which are close to each other, in regards to the membership function parameters, could be combined together into an approximate single membership function to reduce the size of the input domain to the neural network and to reduce the fuzzification calculation time.

5. The proposed FPID servo controller rules could be implemented in the same way as a fully differentiable neuro-fuzzy network including replacement of the membership functions by differentiable ones such as Gaussian membership functions, so that a pre-specified controller performance could be met using this controller only as a standing alone adaptive fuzzy PID feedback controller by online adaptation of the controller membership function and scaling parameters to produce control outputs that achieve the pre-specified system response.

REFERENCES

Abe S. and Lan M.S. (1995), Fuzzy Rules Extraction Directly from Numerical Data for Function Approximation, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, January 1995, Volume 25, Issue 1, Pages 119-129.*

Akbas E. and Esin E.M. (2003), Intelligent Stabilization of Direct Drive Manipulators, *EUROCON 2003, Computer as a Tool, The IEEE Region 8, 22-24 September 2003, Ljublijana, Slovenia, Volume 1, Pages 367-371.*

Akella S. and Hutchinson S. (2002), Coordinating the Motions of Multiple Robots with Specified Trajectories, *IEEE International Conference on Robotics and Automation, 11-15 May 2002 , Washington DC, USA, Volume 1, Pages 624-631.*

Arabshahi P., Choi J.J., Marks R.J., and Caudell T.P. (1992), Fuzzy Control of Backpropagation, *IEEE International Conference on Fuzzy Systems, 8-12 March 1992, San Diego, California, USA, Pages 967-972.*

Armstrong B. (1988), Dynamics for Robot Control Friction Modelling and Enduring Excitation during Parameter Identification, *Ph.D. thesis, Stanford University, USA.*

Armstrong B. and Corke P.I. (1994), A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot, *IEEE International Conference on Robotics and Automation, San Diego, California, USA, Volume 2, Pages 1608-1613.*

Armstrong B., Khatib O., and Burdick J. (1986), The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm, *IEEE International Conference on Robotics and Automation, April 1986, Washington, USA, Volume 1, Pages 510-518.*

Ballini R., Soares S., and Gomide F. (2001), A recurrent Neuro-Fuzzy Network Structure and Learning Procedure, *Proceeding of the Tenth IEEE International Conference on Fuzzy Systems, 2-5 December 2001, Melbourne, Vic. Australia, Volume 3, Pages 1408-1411.*

Baraldi A. and Blonda P. (1999a), A Survey of Fuzzy Clustering Algorithms for Pattern Recognition, Part-I, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, December 1999 , Part B, Volume 29, Issue 6, Pages 778-785.*

Baraldi A. and Blonda P. (1999b), A Survey of Fuzzy Clustering Algorithms for Pattern Recognition, Part-II, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, December 1999 , Part B, Volume 29, Issue 6, Pages 786-801.*

Berenji H.R. and Khedkar P. (1992), Learning and Tuning Fuzzy Logic Controllers Through Reinforcements, *IEEE Transactions on Neural Networks, September 1992, Volume 3, Issue 5, Pages 724-740.*

Bigot S. (2003), New Techniques for Continuous Values Handling in Inductive Learning, *Ph.D. Thesis, University of Wales, UK.*

Breedon P.J., Sivayoganathan K., Balendran V., and Al-Dabass D. (2002), Multi-Axis Fuzzy Control and Performance Analysis for an Industrial Robot, *Proceedings of the IEEE International Conference on Fuzzy Systems, 12-17 May 2002, Honolulu, HI, USA, Volume 1, Pages 500-505.*

Byung-Ju Y. and Freeman R.A. (1995), Feedforward Spring-Like Impedance Modulation in Human Arm Models, *IEEE International Conference on Robotics and Automation, 21-27 May 1995, Nagoya, Japan, Volume 3, Pages 3121-3128.*

Chen S.B., Wu L., and Wang Q.L. (1997), Self-Learning Fuzzy Neural Networks for Control of Uncertain Systems with Time Delays, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, Part B, Volume 27, Issue 1, Pages 142-148.*

Choi J.J., Arabshahi P., Marks R.J., and Caudell T.P. (1992), Fuzzy Parameter Adaptation in Neural Systems, *International Joint Conference on Neural Networks, 7-11 June 1992, Baltimore, MD, USA , Volume 1, Pages 23-238.*

Corke P.I. and Good M.C. (1992), Dynamic Effects in High-Performance Visual Servoing, *IEEE International Conference on Robotics and Automation, 12-14 May 1992, Nice, France, Volume 2, Pages 1838-1843.*

Costa Branco P.J. and Dente J.A. (1998), An Experiment in Automatic Modelling an Electrical Drive System Using Fuzzy Logic, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, Part C, Volume 28, Issue 2, Pages 254-262.*

Craig J.J. (1996), Introduction to ROBOTICS Mechanics and Control, *Addison-Wesley Publishing Company, 1996.*

Delgado M. and Gonzalez A. (1993), An Inductive Learning Procedure to Identify Fuzzy Systems, *Fuzzy Sets and Systems, April 1993, Volume 55, Pages 121-132.*

Efe M.O. and Kaynak O. (1999), A Comparative Study of Neural Network Structures in Identification of Non-linear Systems, *Mechatronics, Volume 9, Pages 287-300.*

Emami M.R., Turksen I.B., and Goldengerg A.A. (1996), An Improved Fuzzy Modelling Algorithm, Part-I: Interface Mechanism, *IEEE Fuzzy Information Processing Society, NAFIPS 1996, Biennial Conference of the North American, 19-22 June 1996, Berkeley, CA, USA , Pages 289-293.*

Emami M.R., Turksen I.B., and Goldengerg A.A. (1996), An Improved Fuzzy Modelling Algorithm, Part-II: System Identification, *IEEE Fuzzy Information Processing Society, NAFIPS 1996, Biennial Conference of the North American, 19-22 June 1996, Berkeley, CA, USA , Pages 294-298.*

Emami M.R., Turksen I.B., and Goldengerg A.A. (1998), Development of a Systematic Methodology of Fuzzy Logic Modelling, *IEEE Transactions on Fuzzy Systems, Volume 6, Issue 3, August 1998, Pages 346-361.*

Emami M.R., Turksen I.B., and Goldengerg A.A. (1998), Fuzzy-Logic Dynamics Modelling of Robot Manipulators, *IEEE International Conference Proceedings on Robotics and Automation, Leuven, Belgium, Volume 3, Pages 2512-2517.*

Emami M.R., Turksen I.B., and Goldengerg A.A. (1999), A Unified Parameterized Formulation of Reasoning in Fuzzy Modelling and Control, *Fuzzy Sets and Systems, Volume 108, Issue 1, 16 November 1999, Pages 59-81.*

Emami M.R., Turksen I.B., and Goldengerg A.A. (2000), Fuzzy-Logic Control of Dynamic Systems: from Modelling to Design, *Engineering Applications of Artificial Intelligence, 1 February 2000, Volume 13, Issue 1, Pages 47-69.*

Emami M.R., Turksen I.B., and Goldengerg A.A. (2000), Systematic Design and Analysis of Fuzzy-Logic Control and Application to Robotics, Part-I: Modelling, *Robotics and Automation Systems magazine, Volume 33, Issues 2-3, Pages 65-88.*

Emami M.R., Turksen I.B., and Goldengerg A.A. (2000), Systematic Design and Analysis of Fuzzy-Logic Control and Application to Robotics, Part-II: Control, *Robotics and Automation Systems Magazine, Volume 33, Issues 2-3, Pages 89-108.*

Er M.J., and Gao Y. (2003), Robust Adaptive Control of Robot Manipulators Using Generalized Fuzzy Neural Networks, *IEEE Transactions on Industrial Electronics, June 2003, Volume 50 , Issue 3 , Pages 620-628.*

Er M.J., Yap S.M., Yeaw C.W., and Luo F.L. (1997), A Review of Neural-Fuzzy Controllers for Robotic Manipulators, *Thirty-Second IAS Annual Meeting, IEEE Industry Applications Conference, 5-9 October 1997, New Orleans, Los Anglos, USA, Volume 2, Pages 812-819.*

Erbatur K., Kaynak O., and Rudas I. (1995), A Study of Fuzzy Schemes for Control of Robotic Manipulators, *IECON Twenty-First IEEE International Conference on Industrial Electronics, Control, and Instrumentation, 6-10 November 1995, Orlando, Florida, USA, Volume 1, Pages 63-68.*

Estevez P.A. and Nakano R. (1995), Hierarchical Mixture of Experts and Max-Min Propagation Neural Networks, *IEEE International Conference on Neural Networks, 27 November -1 December 1995, Perth, WA, Australia, Pages 651-565.*

Feng L., Koren Y., and Borenstein J. (1993), Cross-Coupling Motion Controller for Mobile Robots, *IEEE Control Systems Magazine, December 1993, Volume 13, Issue 6, Pages 35-43.*

Fukuda T., Shibata T., Tokita M., and Mitsuoka T. (1990), Adaptation and Learning for Robotic Manipulator by Neural Network, *Proceedings of The Twenty Ninth IEEE International Conference on Decision and Control, 5-7 December 1990, Honolulu, HI, USA, Volume 6, Pages 3283-3288.*

Garcia C.E. and Morari M. (1985), Internal Model Control - Multivariable Control Law Computation and Tuning, *Industry and Engineering Chemistry Design and Development, Volume 24, Pages 484-494.*

Green A. and Sasiadek J.Z. (2001), Fuzzy and Optimal Control of a Two-Link Flexible Manipulator, *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 8-12 July 2001, Como, Italy, Volume 2, Pages 1169-1174.*

Grzegorzewski P. (2001), Fuzzy Tests - Defuzzification and Randomisation, *Fuzzy Sets and Systems, Volume 118, Issue 3, 16 March 2001, Pages 437-446.*

Gueaieb W., Karray F., and Al-Sharhan S. (2001), An Adaptive Fuzzy Control Approach for Cooperative Manipulators, *IEEE International Symposium on Intelligent Control, 5-7 September 2001, Mexico City, Mexico, Pages 167-172.*

Hiraga I., Furuhashi T., Uchikawa Y., and Nakayama S. (1995), An Acquisition of Operator's Rules for Collision Avoidance Using Fuzzy Neural Networks, *IEEE Transactions on Fuzzy Systems, August 1995, Volume 3, Issue3, Pages 280-287.*

Hitam M.S. (2001), Fuzzy Logic Control of an Industrial Robot, *IFSA World Congress and 20th IEEE NAFIPS International Conference, 9th joint, 25-28 July 2001, Vancouver, BC, Canada , Volume 1, Pages 257-262.*

Hsu R., Kageyama M., Fukui H., Nakaya Y., and Harashima H. (1993), Human

Arm Modelling for Analysis/Synthesis Image Coding, *The Second IEEE International Workshop on Robot and Human Communication, Tokyo, Japan, Pages 352- 355.*

Hu J., Dawson D.M., and Qian Y. (1996), Position Tracking Control for Robot Manipulators Driven by Induction Motors Without Flux Measurements, *IEEE Transactions on Robotics and Automation, Volume 12, Issue 3, Pages 419-438.*

Jacobs R.A. (1988), Increased Rates of Convergence Through Learning Rate Adaptation, *Neural Networks, Volume 1, Issue 4, Pages 295-307.*

Jang J.O. (2001), Implementation of Indirect Neuro-Control for a Non-linear Two-Robot MIMO Systems, *Proceeding of Control Engineering Practice, January 2001, Volume 9, Issue 1, Pages 89-95.*

Jang J.-S.R. (1992), Self-Learning Fuzzy Controllers Based on Temporal Backpropagation, *IEEE Transactions on Neural Networks, Pages 714-723.*

Jang J.-S.R. (1993), ANFIS Adaptive-Network-Based Fuzzy Inference System, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, May-June 1993, Volume 23, Issue 3, Pages 665-685.*

Jeen-Shing W. and Lee C.S.G. (2003), Self-Adaptive Recurrent Neuro-Fuzzy Control of an Autonomous Underwater Vehicle, *IEEE Transactions on Robotics and Automation, April 2003, Volume 19, Issue 2, Pages283-295.*

Jung S. and Hsia T.C. (1995), New Neural Network Control Technique for Non-Model Based Robot Manipulator Control, *IEEE International Conference on Systems, Manufacturing, and Cybernetics, 'Intelligent Systems for the 21st Century', 22-25 October 1995, Vancouver, BC, Canada, Volume 3 , Pages2928 – 2933.*

Kawafuki M., Sasaki M., and Fujisawa F. (1997), Feedback-Error-Learning Neural Network for Trajectory Control of a Flexible Micro-Actuator, *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 16-20 June 1997, Tokyo, Japan, Page 73.*

Kawato M., Uno Y., Isobe M., and Suzuki R. (1988), Hierarchical Neural Network Model for Voluntary Movement with Application to Robotics, *IEEE Control Systems Magazine, Volume 8, Issue 2, April 1988, Pages 8-15.*

Kazemian H.B. (2002), The SOF-PID Controller for the Control of a MIMO Robot Arm, *IEEE Transactions on Fuzzy Systems, August 2002, Volume 10, Issue 4, Pages 523-532.*

Keller J.M., Yager R., and Tahani H. (1992), Neural Network Implementation of Fuzzy Logic, *Fuzzy Sets and Systems, 10 January 1992, Volume 45, Pages 1-12.*

Kim S.-W., Lee J.-J., and Sugisaka M. (1993), Inverse Kinematics Solution Based on Fuzzy Logic for Redundant Manipulators, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 1993, 26-30 July 1993, Yokohama, Japan , Volume 2, Pages 904-910.*

Kishan K.K. and Jamshidi M. (1997), Neural Network Based Identification of Robot Dynamics Used for Neuro-Fuzzy Controller, *IEEE International Conference on Robotics and Automation, 20-25 April 1997, Albuquerque, NM, USA, Volume 2, Pages 1118-1123.*

kovacs L.L., stepan G., Toth A., Arz G., and Magyar G. (2001), Industrial Robot in a Medical Application- Back to Walk-Through Programming, *17th National Conference on Manufacturing Research, Cardiff University, UK, Pages 479-485.*

Lai J.H. and Lin C.T. (1999), Application of Neural Fuzzy Network to Pyrometer Correction and Temperature Control in Rapid Thermal Processing, *IEEE Transactions on Fuzzy Systems, April 1999, Volume 7, Issue2, Pages 160-175.*

Lai M.F., Nakano M., and Hsieh G.C. (1996), Application of Fuzzy Logic in the Phase-Locked Loop Speed Control of Induction Motor Drive, *IEEE Transactions on Industrial Electronics, Volume 43, Issue 6, December 1996, Pages 630-639.*

Lange F. and Hirzinger G. (1996), Learning Force Control with Position Controlled Robots, *IEEE International Conference on Robotics and Automation, 22-28 April 1996, Minneapolis, MN, USA, Volume 3, Pages 2282–2288.*

Lee C.C. (1990a), Fuzzy Logic in Control Systems Fuzzy Logic controller Part I, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, March-April 1990, Volume 20, Issue 2, Pages 404-418.*

Lee C.C. (1990b), Fuzzy Logic in Control Systems Fuzzy Logic controller Part II, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, March-April 1990, Volume 20, Issue 2, Pages 419-435.*

Lee C.C. (1991), A Self Learning Rule-Based Controller Employing Approximate Reasoning and Neural Network Concepts, *International Journal of Intelligent Systems, Volume 6, Pages 71-93.*

Lee K.M., Kwak D.H., and Kwang H.L. (1996), Fuzzy Inference Neural Network for

Fuzzy Model Tuning, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, Part B, August 1996 , Volume 26, Issue 4, Pages 637-645.*

Lee M., Cho K.-B., Lee S.-Y., and Park C.H. (1993), A Feedforward/Feedback Neural Control Structure and its Application to a Robotic System, *Proceeding of the IEEE International Conference on Neural Networks, IJCNN 1993-Nagoya, South Korea, 25-29 October 1993, Volume 3, Pages 2757-2760.*

Li H.X. (1999), Approximate Model Reference Adaptive Mechanism for Nominal Gain Design of Fuzzy Control System, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, Part B, February 1999, Volume 29, Issue 1, Pages 43-48.*

Li H.X. and Gatland H.B. (1995), A New Methodology for Designing a Fuzzy Logic Controller, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, March 1995, Volume 25, Issue 3, Pages 505-514.*

Li Q., Poo A.N. and Lim C.M. (1996), Internal Model Structure in the Control of Robot Manipulators, *Mechatronics, August 1996, Volume 6, Issue 5, Pages 571-590.*

Li Q., Poo A.N., Lim C.M., and Ang M. (1995), Neuro-Based Adaptive Internal Model Control for Robot Manipulators, *Proceedings of the IEEE International Conference on Neural Network, 27 Novmeber-1 December 1995, Perth, WA, Australia, Volume 5, Pages 235-2357.*

Li W. (1998), Design of a Hybrid Fuzzy Logic Proportional plus Conventional Integral-Derivative Controller, *IEEE Transactions on Fuzzy Systems, November 1998, Volume 6, Issue 4, Pages 449-463.*

Liaw C.M. and Wang J.B. (1991), Design and Implementation of a Fuzzy Controller for a High Performance Induction Motor Drive, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, July-August 1991, Volume 21, Pages 921-929.*

Lin C.-T. and Lee C.S.G. (1991), Neural Network-Based Fuzzy Logic Control and Decision System, *IEEE Transactions on Computers, December 1991, Volume 40, Issue 12, Pages 1320-1336.*

Lin C.-T. and Lee C.S.G. (1992), Real-Time Supervised Structure Parameter Learning for Fuzzy Neural Network, *IEEE International Conference on Fuzzy Systems, 8-12 March 1992, San Diego, CA, USA, Pages 1289-1290.*

Lin C.-T. and Lu Y.C. (1995), A Neural Fuzzy System With Linguistic Teaching

Signals, *IEEE Transactions on Fuzzy Systems, Volume 3, Issue 2, Pages 169-188.*

Lin C.-T. and Lu Y.C. (1996), A Neural Fuzzy System with Fuzzy Supervised Learning, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, Part B, October 1996, Volume 26, Issue 5, Pages 745-763.*

Lin Y. J. and Lee T.S. (1993), An Investigation of Fuzzy Logic Control of Flexible Robots, *Robotics Magazine, Volume 11, Pages 363-371.*

Lin, C.J. and Lin C.-T. (1997), An ART-Based Fuzzy Adaptive Learning Control Network, *IEEE Transactions on Fuzzy Systems, Volume 5, Issue 4, Pages 477-496.*

Lin, C.-T. and Chung I.F. (1999), A Reinforcement Neuro-Fuzzy Combiner for Multiobjective Control, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, Part B, December 1999, Volume 29, Issue 6, Pages 726-744.*

Liu S.-R. and Yu J.-S. (2002), Robust Control Based on Neuro-Fuzzy Systems for a Continuous Stirred Tank Reactor, *IEEE International Conference Proceedings on Machine Learning and Cybernetics, China, Volume 3, Pages 1483-1488.*

Mamdani E.H. (1974), Application of Fuzzy Algorithms for Control of Simple Dynamic Plant, *IEEE Proceeding, Volume 121, Issue 12, pages 1585-1588.*

Mann G.K.I., Hu B.G., and Gosine R.G. (1999), Analysis of Direct Action Fuzzy PID Controller Structures, *IEEE Transactions on Systems, Manufacturing, and Cybernetics Part B, June 1999, Volume 29, Issue 3, Pages 371-388.*

Martinez J., Bowle J., and Mills P. (1996), A Fuzzy Logic Positioning System for an Articulated Robot Arm, *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, New Orleans, LA, USA, Volume 1, Pages 251-257.*

Ming Y., Guizhang L., and Jiangeng L. (2001), An Inverse Kinematics solution for Manipulators Based on Fuzzy Logic, *IEEE International conference on Info-tech and Info-net, 29 October-1 November 2001, Beijing, China, Volume 4, Pages 400-404.*

Miyamoto H., Kawato M., Setoyama T., and Suzuki R. (1988), Feedback-Error Learning Neural Network for Trajectory Control of Robotic Manipulator, *Neural Networks, Volume 1, Issue 3, Pages 251-265.*

Miyamura A. and Kimura H. (2002), Stability of Feedback Error Learning Scheme, *Systems & Control Letters, Volume 45, Issue 4, 5 April 2002, Pages 303-316.*

Mizumoto M. (1995), Realisation of PID Controls by Fuzzy Control Methods, *Fuzzy Sets and Systems, 20 March 1995, Volume 70, Issues 2-3, Pages 171-182.*

Moore P.R. and Chen C.M. (1995), Fuzzy Logic Coupling and Synchronised Control of Multiple Independent Servo-Drives, *Control Engineering Practice, Volume 3, Issue 12, December 1995, Pages 1697-1708.*

Morari M. and Zafiriou E. (1989), Robust Process Control, *Prentice Hall, Englewood Cliffs, New Jersey.*

Moudgal V.G., Kwong W.A., and Passino K.M. (1995), Fuzzy Learning Control for a Flexible-Link Robot, *IEEE Transactions on Fuzzy Systems, Volume 3, Issue 2, May 1995, Pages 199-210.*

Moudgal V.G., Passino K.M., and Yurkovich S. (1994), Rule-Based Control for a Flexible-Link Robot, *IEEE Transactions on Control Systems Technology, Volume 2, Issue 4, December 1994, Pages 392-405.*

Narendra K.S. and Parthasarathy K. (1990), Identification and Control of Dynamical Systems Using Neural Networks, *IEEE Transactions on Neural Networks, Volume 1, Issue 1, Pages 4-27.*

Nascimento C.L. and McMichael D.W. (1991), Robot Control Using the Feedback-Error-Learning Rule With Variable Feedback Gain, *IEEE Second International Conference on Artificial Neural Networks, Bournemouth, UK, Pages 139-143.*

Nedungadi A. and Wenzel D.J. (1991), A Novel Approach to Robot Control Using Fuzzy Logic, *IEEE International Conference on Systems, Manufacturing, and Cybernetics, Decision Aiding for Complex Systems Conference Proceedings, 13-16 October 1991, Charlottesville, VA, USA, Volume 3, Pages 1925-1930.*

Norberto Pires J., Sa da Costa J.M.G., (1997), Position Sensing and Motor Control in Industrial Robotics, *Proceedings of the IEEE International Symposium on Industrial Electronics, 7-11 July 1997, Guimaraes, Portugal, Volume 3, Pages 866-871.*

Osumi H. and Arai T. (1994), Cooperative Control Between Two Position-Controlled Manipulators, *IEEE International Conference on Robotics and Automation, 8-13 May 1994, San Diego, CA, USA, Volume 2, Pages 1509-1514.*

Osumi H., Ono M., Fujibayashi M., and Kagatani M. (1997), Cooperative System for Multiple Position-Controlled Robots with Free Joint Mechanisms, *IEEE International Conference on Robotics and Automation, 20-25 April 1997, Albuquerque, NM, USA, Volume 2, Pages 1484-1489.*

Paljug E. and Yun X. (1995), Experimental Study of Two Robot Arms Manipulating Large Objects, *IEEE Transactions on Control Systems, Volume 3, Issue 2, Pages 177-188.*

Pan L. and Woo P.Y. (2000), PD Manipulator Controller with Fuzzy Adaptive Gravity Compensation, *Journal of Robotic Systems, Volume 2, Pages 93-106.*

Pasino K.M. and Yurkovich S. (1998), Fuzzy Control, *Addison Wesley Publications Incorporation, 1998.*

Pedrycz W. (1993), Fuzzy Control and Fuzzy Systems, *John Wiley & Sons Publications Incorporation, Taunton, NewYork.*

Peng L. and Woo P.-Y. (2002), Neural-Fuzzy Control System for Robotic Manipulators, *IEEE Control Systems Magazine, Volume 22, Issue 1, Pages 53-63.*

Pham D.T. and Aksoy M.S. (1995), A New Algorithm for Inductive Learning, *Journal of Systems Engineering, Springer-Velarg Ltd., Volume 5, Pages 115-122.*

Pham D.T. and Liu X. (1993), Identification of Linear and Non-linear Dynamic Systems Using Recurrent Neural Networks, *Artificial Intelligence in Engineering, Volume 8, Issue 1, Pages 67-75.*

Pham D.T. and Liu X. (1995), Neural Networks for Identification, Prediction and Control, *Springler-Verlag Limited, London.*

Pham D.T. and Liu X. (1996), Training of Elman Networks and Dynamic System Modelling, *International Journal of Systems Science, Volume 27, Pages 221-226.*

Pham D.T. and Oh S.J. (1993), Adaptive Control of Dynamic Systems Using Neural Networks, *IEEE-SMC Conference on Systems Engineering in the Service of Human, 17-20 October 1993, Le Touquet, France, Pages 97-102.*

Pham D.T. and Oh S.J. (1994), Adaptive Control of a Robot Using Neural Networks, *Robotica, Issue 12, Pages 553-561.*

Pham D.T. and Oh S.J. (1999), Identification of Plant Inverse Dynamics Using Neural Networks, *Artificial Intelligence in Engineering, Pages 309-320.*

Pham D.T. and Sagiroglu S. (2001), Training Multilayer Preceptrons for Pattern Recognition a Comparative Study of Four Training Algorithms, *International Journal of Machine Tools & Manufacture 41, Pages 419-430.*

Pham D.T. and Yildirim S. (1999), Comparison of Four Methods of Robot Trajectory Control, *Artificial Intelligence in Engineering.*

Pham D.T. and Yildirim S. (1999), Control of the Trajectory of a Planar Robot Using Recurrent Hybrid Networks, *Machine Tools & Manufacture, March 1999, Volume 39, Issue 3, Pages 415-429.*

Pham D.T., Eldukhri E.E., Dimov S.S., Packianather M.S., Zlatov N.B., Fahmy A.A., and Shankir Y. (2001), A Knowledge-Based System for Selection of Exercises for Robotised Physiotherapy, *Eighth IEEE International Conference on Mechatronics and Machine Vision in Practice, 2001, Hong Kong, Pages.85-88.*

Pham D.T., Zlatov N.B., Eldukhri E.E., Dimov S.S., Bratanov D., Dobrev T., Packianather M.S., and Fahmy A.A. (2001), Modelling and Control of an 8-dof Mechatronic Limb, *Eighth IEEE International Conference on Mechatronics and Machine Vision in Practice, Hong Kong, 2001, Pages 314-317.*

PTC, Parametric Technology Corporation, Pro/Engineer & Pro/Mechanica User Manuals, *Release 23.3 (2002), www.ptc.com.*

Rajasekharan S. and Kambhampati C. (2001), Neuro-Fuzzy Modelling and Control of Cooperative Manipulators Handling a Common Object, *IEEE/IFSA World Congress and 20th NAFIPS International Conference, 9th joint, 25-28 July 2001, Vancouver, BC, Canada, Volume 3, Pages 1454-1459.*

Runkler T.A. (1997), Selection of Appropriate Defuzzification Methods Using Application Specific Properties, *IEEE Transactions on Fuzzy Systems, February 1997, Volume 5, Issue 1, Pages 72-79.*

Saade, J.J. (1996), A Unifying Approach to Defuzzification and Comparison of the Outputs of Fuzzy Controllers, *IEEE Transactions on Fuzzy Systems, August 1996, Volume 4, Issue 3, Pages 227-237.*

Sang-Bae L. (1997), Industrial Robotic Systems with Fuzzy Logic Controller and Neural Network, *IEEE First International Conference on Knowledge-Based Intelligent Electronic Systems, Adelaide, SA, Australia, Pages 599-606.*

Sasaki M., Kawafuku M., and Takahashi K. (1997), Comparison of Feedback Controllers for Feedback-Error-Learning Neural Network Control System with Application to a Flexible Micro-Actuator, *IEEE International Conference on Systems, Manufacturing and Cybernetics, Orlando, FL, USA, Pages 4035-4040.*

Shankir Y. (2001), Fuzzy Logic Systems and Fuzzy Neural Networks for Dynamic Systems Modelling and Control, *Ph.D. Thesis, University of Wales, Cardiff School of Engineering, Cardiff University, UK.*

Shin Y.C. (1994), Adaptive Control in Manufacturing, *Artificial Neural Networks for Intelligent Manufacturing, Pages 399-411.*

Srinivasan A., Batur C., and Chan C.C. (1993), Using Inductive Learning to Determine Fuzzy Rules for Dynamic Systems, *Engineering Applications of Artificial Intelligence, Volume 6, Issue 3, Pages 257-264.*

Subbarao K., Verma A., and Junkins J.L. (2001), Model Reference Adaptive Control of Constrained Cooperative Manipulators, *IEEE international Conference on Control Applications, 5-7 September 2001, Mexico City, Mexico, Pages 553-558.*

Sugeno M. and Yasukawa T. (1993), A Fuzzy-Logic-Based Approach to Qualitative Modelling, *IEEE Transactions on Fuzzy Systems, Volume 1, Issue 1, Pages 7-31.*

Sun D. and Liu Y.H. (2001), Position and Force Tracking of Two-Manipulator System Manipulating a Flexible Beam Payload, *IEEE International Conference on Robotics and Automation, 21-26 May, Seoul, Korea, Volume 4, Pages 3483-3488.*

Sun D. and Mills J.K. (2002), Adaptive Synchronized Control for Coordination of Two Robot Manipulators, *IEEE International Conference on Robotics and Automation, ICRA 2002, Washington DC, USA, Volume 1, Pages 976-981.*

Takagi H. and Hayashi I. (1991), NN-Driven Fuzzy Reasoning, *International Journal of Approximate Reasoning, Volume 5, Issue 3, Pages 191-212.*

Takagi T. and Sugeno M. (1985), Fuzzy Identification of Systems and Its Applications to Modelling and Control, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, Volume SMC-15, Issue 1, Pages 116-132.*

Tanaka, K., Sano M., and Watanabe H. (1995), Modelling and Control of Carbon Monoxide Concentration Using a Neuro-Fuzzy Technique, *IEEE Transactions on Fuzzy Systems, August 1995, Volume3, Issue3, Pages 271-279.*

Tang W., Chen G. and Lu R. (2001), A Modified Fuzzy PI Controller for a Flexible-Joint Robot Arm with Uncertainties, *Fuzzy Sets and Systems, February 2001, Volume 118, Issue 1, 16 Pages 109-119.*

Terashita J. and Kimura H. (2002), Robustness of Feedback Error Learning Method

with Time Delay, *Proceedings of the 41st SICE/IEEE Annual Conference , 5-7 August 2002, Tokyo, Japan, Volume 4, Pages 2240-2244.*

Tinos R. and Terra M.H. (2002), Control of Cooperative Manipulators with Passive Joints, *IEEE Proceeding of the American Control Conference, Anchorage, 8-10 May 2002, Sao Carlos, Brazil, Pages 1129-1134.*

Tinos R., Terra M.H., and Bergerman M. (2002), Fault Tolerance in Cooperative Manipulator, *IEEE International Conference on Robotics & Automation, 11-15 May 2002, Sao Carlos, Brazil, Pages 470-475.*

Tsai C.-H., Liu J.-S., and Lin W.-S. (1996), A Neuro-Fuzzy Logic Controller for Trajectory Tracking of Uncertain Robots, *IEEE International Conference on Robotics and Automation, April 1996, Minneapolis, MN, USA, Volume 2, Pages 1929-1934.*

Tsoukalas L.H. and Uhrig R.E. (1997), Fuzzy and Neural Approaches in Engineering, *John Wiley & Sons Incorporation, 1997.*

Vaccaro R.J. and Hill S.D. (1988), A Joint-Space Command Generator for Cartesian Control of Robotic Manipulators, *IEEE Transactions on Robotics and Automation, Volume 4 , Issue 1 , February 1988, Pages 70-76.*

Verdonck W. and Swevers J. (2002), Improving the Dynamic Accuracy of Industrial Robots by Trajectory Pre-Compensation, *IEEE/ICRA International Conference Proceedings on Robotics and Automation, Volume 4, 11-15 May 2002, Leuven, Belgium, Pages 3423-3428.*

Wang L. and Langari R. (1996), Complex Systems Modelling Via Fuzzy Logic, *IEEE Transactions on Systems, Manufacturing, and Cybernetics Part B, Volume 26, Issue 1, Pages 100-105.*

Wang L.X. and Mendel J.M. (1992a), Generating Fuzzy Rules by Learning from Examples, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, Volume 22, Issue 6, Pages 1414-1427.*

Wang L.X. and Mendel J.M. (1992b), Fuzzy basis functions, universal approximation, and orthogonal least-squares learning, *IEEE Transactions on Neural Networks, September 1992, Volume 3, Issue 5, Pages 807-814.*

Watanabe K., Tang J., Nakamura M., Koga S. and Fukuda T. (1996), A Fuzzy-Gaussian Neural Network and its Application to Mobile Robot Control, *IEEE Transactions on Control Systems Technology, Volume 4, Issue2, Pages 193-199.*

Wedel D.L. and Saridis G.N.(1988), An Experiment in Hybrid Position/Force Control of a Six DOF Revolute Manipulator, *IEEE International Conference on Robotics and Automation, 24-29 April 1988, Philadelphia, PA, USA, Pages 1638-1642.*

Wen X.-Y., Zhang J.-G., and Zhao Z.-C. (2003), Fuzzy Neural Network Internal Model Control, *IEEE International Conference on Machine Learning and Cybernetics, 2-5 November 2003, China, Volume 2, Pages 661-665.*

Xu Y. and Nechyba M. (1993), Fuzzy Inverse Kinematic Mapping: Rule Generation, Efficiency, and Implementation, *IEEE/RSJ International Conference on Intelligent Robots and Systems, 26-30 July 1993, Yokohama, Japan, Volume 2, Pages 911-918.*

Xu. X. and Chen Y. (2000), A Method for Trajectory Planning of Robot Manipulators in Cartesian Space, *IEEE Third world congress on Intelligent control and Automation, 28 June-2 July 2000, Hefei, China, Pages 1220-1225.*

Yaochu J., Jingping J., and Jing Z. (1995), Neural Network Based Fuzzy Identification and Its Application to Modelling and Control of Complex Systems, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, June 1995, Volume 25, Issue 6, Pages 990-997.*

Yildirim S. (1998), Robot Control Using Neural Networks, *Ph.D. Thesis, University of Wales, Cardiff School of Engineering, UK.*

Yildirim S. (2001), Neural Network Controller for Cooperating Robots, *Electronics Letters, 25 Oct 2001, Volume 37, Issue 22, Pages 1351-1352.*

Yildirim S. and Sukkar M.F.(1996), Internal Model Control of a Robot Using New Neural Networks, *IEEE International Conference on Systems, Manufacturing, and Cybernetics, October 1996, Volume 4, Pages 3095-3100.*

Yildirim S., Sukkar M.F., Demirci R., and Aslantas V. (1996), Design of Adaptive NNs-Robust-PID Controller for a Robot Control, *IEEE International Symposium on Intelligent Control, 15-18 September 1996, Dearborn, MI, USA, Pages 508-513.*

Ying H. (1993), The Simplest Fuzzy Controllers Using Different Inference Methods are Different Non-linear Proportional-Integral-Controllers with Variable Gains, *Automatica, Volume 29, Issue 6, Pages 1579-1589.*

Ying H., Siler W., and Buckley J.J. (1990), Fuzzy Control Theory: A Non-linear Case, *Automatica, Volume 26, Issue3, Pages 513-520.*

Yuan F., Feldkamp L.A., Davis L.I., Jr., and Puskorius G.V. (1992), Training a

Hybrid Neural-Fuzzy System, *IEEE/IJCNN International Joint Conference on Neural Networks, 7-11 June 1992, Baltimore, MD, USA, Volume 2, Pages 739-744.*

Zadeh L.A. (1973), Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, *IEEE Transactions on Fuzzy Systems, Volume SMC-3, Issue 1, pages 28-44.*

Zapata G.O.A., Galvao R.K.H., and Yoneyama T. (1999), Extracting Fuzzy Control Rules from Experimental Human Operator Data, *IEEE Transactions on Systems, Manufacturing, and Cybernetics, Part B, Volume 29, Issue 3, Pages 398-406.*

Zhang Y.Q. and Kandel A. (1998), Compensatory Neurofuzzy Systems with Fast Learning Algorithms, *IEEE Transactions on Neural Networks, January 1998, Volume 9, Issue 1, Pages 83-105.*

Zhang, X., Hang C.C., Tan S., and Wang P-Z. (1996), The Min-Max Function Differentiation and Training of Fuzzy Neural Networks, *IEEE Transactions on Neural Networks, September 1996, Volume 7, Issue 5, Pages 1139-1150.*

Ziegler J.G. and Nichols N.B. (1942), Optimum Setting for Automatic Controller, *Transactions on ASME, No. 64, pages 759-768.*

APPENDIX A

Mathematical Formulation

A.1. Kinematics Equations for Puma 560® Manipulator

The Kinematics function of the Puma 560® simulator returns a 4×4 transformation matrix representing the end-effector position and orientation with respect to the base frame of the manipulator as its output using a given set of joint angles and link parameters as input. The direct kinematics solution is a matter of calculating $T = A_0^6 = \prod_{j=1}^{i} A_{j-1}^{j}$ by chain multiplying the six A_{i-1}^{i} matrices and evaluating each element in the T matrix. The individual A_{i-1}^{i} matrices are given by:

$$A_{i-1}^{i} = \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(A.1)}$$

Using the link coordinate system shown in figure (3.1) and table (3.1), each of the A_{i-1}^{i} matrices for left and right arm orientation, respectively, can be expressed as follows:

$$A_0^1=\begin{bmatrix} C\theta_1 & 0 & \mp S\theta_1 & 0 \\ S\theta_1 & 0 & \pm C\theta_1 & 0 \\ 0 & \mp 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_1^2=\begin{bmatrix} C\theta_2 & 0 & -S\theta_1 & a_2C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & a_2S\theta_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_2^3=\begin{bmatrix} C\theta_3 & 0 & \pm S\theta_3 & 0 \\ S\theta_3 & 0 & \mp C\theta_3 & 0 \\ 0 & \pm 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} A_3^4=\begin{bmatrix} C\theta_4 & 0 & \mp S\theta_4 & 0 \\ S\theta_4 & 0 & \pm C\theta_4 & 0 \\ 0 & \mp 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_4^5=\begin{bmatrix} C\theta_5 & 0 & \pm S\theta_5 & 0 \\ S\theta_5 & 0 & \mp C\theta_5 & 0 \\ 0 & \pm 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_5^6=\begin{bmatrix} C\theta_6 & -S\theta_6 & 0 & 0 \\ S\theta_6 & C\theta_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.2)}$$

Where $S\theta_i=\sin(\theta_i)$ and $C\theta_i=\cos(\theta_i)$.

In this way the end-effector orientation and position with reference to the base co-ordinate system (frame 0) can be obtained from T as:

$$T=\begin{bmatrix} \mathrm{n} & \mathrm{s} & \mathrm{a} & \mathrm{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}=\begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.3)}$$

where:

$$n_x=C\theta_1[C(\theta_2+\theta_3)\{C\theta_4C\theta_5C\theta_6-S\theta_4S\theta_6\}-S(\theta_2+\theta_3)S\theta_5C\theta_6]-S\theta_1[S\theta_4C\theta_5C\theta_6+C\theta_4S\theta_6]$$

$$n_y=S\theta_1[C(\theta_2+\theta_3)\{C\theta_4C\theta_5C\theta_6-S\theta_4S\theta_6\}-S(\theta_2+\theta_3)S\theta_5C\theta_6]+C\theta_1[S\theta_4C\theta_5C\theta_6+C\theta_4S\theta_6]$$

$$n_z=\mp S(\theta_2+\theta_3)[C\theta_4C\theta_5C\theta_6-S\theta_4S\theta_6]\mp C(\theta_2+\theta_3)S\theta_5C\theta_6$$

$$s_x = C\theta_1[-C(\theta_2+\theta_3)\{C\theta_4C\theta_5S\theta_6 + S\theta_4C\theta_6\} + S(\theta_2+\theta_3)S\theta_5S\theta_6] - S\theta_1[-S\theta_4C\theta_5S\theta_6 + C\theta_4C\theta_6]$$

$$s_y = S\theta_1[-C(\theta_2+\theta_3)\{C\theta_4C\theta_5S\theta_6 + S\theta_4C\theta_6\} + S(\theta_2+\theta_3)S\theta_5S\theta_6] + C\theta_1[-S\theta_4C\theta_5S\theta_6 + C\theta_4C\theta_6]$$

$$s_z = \pm S(\theta_2+\theta_3)\{C\theta_4C\theta_5S\theta_6 + S\theta_4C\theta_6\} \pm C(\theta_2+\theta_3)S\theta_5S\theta_6$$

$$a_x = \pm C\theta_1[C(\theta_2+\theta_3)C\theta_4S\theta_5 + S(\theta_2+\theta_3)C\theta_5] \mp S\theta_1S\theta_4S\theta_5$$

$$a_y = \pm S\theta_1[C(\theta_2+\theta_3)C\theta_4S\theta_5 + S(\theta_2+\theta_3)C\theta_5] \pm C\theta_1S\theta_4S\theta_5$$

$$a_z = -S(\theta_2+\theta_3)C\theta_4S\theta_5 + C(\theta_2+\theta_3)C\theta_5$$

$$p_x = C\theta_1[\pm d_6\{C(\theta_2+\theta_3)C\theta_4S\theta_5 + S(\theta_2+\theta_3)C\theta_5\} \pm S(\theta_2+\theta_3)d_4 + a_2C\theta_2] \mp S\theta_1\{d_6S\theta_4S\theta_5 + d_2\}$$

$$p_y = S\theta_1[\pm d_6\{C(\theta_2+\theta_3)C\theta_4S\theta_5 + S(\theta_2+\theta_3)C\theta_5\} \pm S(\theta_2+\theta_3)d_4 + a_2C\theta_2] + C\theta_1\{d_6S\theta_4S\theta_5 + d_2\}$$

$$p_z = d_6\{C(\theta_2+\theta_3)C\theta_5 - S(\theta_2+\theta_3)C\theta_4S\theta_5\} + C(\theta_2+\theta_3)d_4 \mp a_2S\theta_2 + d_1$$

- $\pm$ Indicate left and right shoulder configuration, respectively.

Given the end-effector orientation and position as shown above, the inverse kinematics approach is used to obtain the joint angles θ_i of the robot arm as follows:

$$\theta_1 = Tan^{-1}\left[\frac{\pm p_y\sqrt{p_x^2+p_y^2-d_2^2} - d_2p_x}{\pm p_x\sqrt{p_x^2+p_y^2-d_2^2} + d_2p_y}\right], \tag{A.4}$$

$$\theta_2 = Tan^{-1}\left[\frac{-(p_z\{a_2 + d_4S\theta_3\} + \{d_4C\theta_3\}\{\pm\sqrt{p_x^2+p_y^2-d_2^2}\})}{p_z(d_4C\theta_3) - \{a_2 + d_4S\theta_3\}\{\pm\sqrt{p_x^2+p_y^2-d_2^2}\}}\right], \tag{A.5}$$

± Indicate left and right shoulder configuration, respectively.

$$\theta_3 = Tan^{-1}\left[\frac{p_x^2 + p_y^2 + p_z^2 - d_4^2 - a_2^2 - d_2^2}{\pm\sqrt{4d_4^2a_2^2 - (p_x^2 + p_y^2 + p_z^2 - d_4^2 - a_2^2 - d_2^2)^2}}\right], \quad \text{(A.6)}$$

- ± Indicate elbow-below-hand and elbow-above-hand configurations, respectively.

$$\theta_4 = Tan^{-1}\left[\frac{C\theta_1 a_y - S\theta_1 a_x}{C\theta_1 C(\theta_2 + \theta_3)a_x + S\theta_1 C(\theta_2 + \theta_3)a_y - S(\theta_2 + \theta_3)a_z}\right], \quad \text{(A.7)}$$

$$\theta_5 = Tan^{-1}\left[\frac{(C\theta_1 C(\theta_2 + \theta_3)C\theta_4 - S\theta_1 S\theta_4)a_x + (S\theta_1 C(\theta_2 + \theta_3)C\theta_4 + C\theta_1 S\theta_4)a_y - C\theta_4 S(\theta_2 + \theta_3)a_z}{C\theta_1 S(\theta_2 + \theta_3)a_x + S\theta_1 S(\theta_2 + \theta_3)a_y + C(\theta_2 + \theta_3)a_z}\right] \quad \text{(A.8)}$$

$$\theta_6 = Tan^{-1}\left[\frac{(-S\theta_1 C\theta_4 - C\theta_1 C(\theta_2 + \theta_3)S\theta_4)n_x + (C\theta_1 C\theta_4 - S\theta_1 C(\theta_2 + \theta_3)S\theta_4)n_y + (S\theta_4 S(\theta_2 + \theta_3))n_z}{(-S\theta_1 C\theta_4 - C\theta_1 C(\theta_2 + \theta_3)S\theta_4)s_x + (C\theta_1 C\theta_4 - S\theta_1 C(\theta_2 + \theta_3)S\theta_4)s_y + (S\theta_4 S(\theta_2 + \theta_3))s_z}\right] \quad \text{(A.9)}$$

- $-180^0 \le \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6 \le 180^0$
- The degenerate case (θ_5=0), i.e., when the axis of joint 6 is aligned with the approach vector $[a_x \quad a_y \quad a_z]^T$, results in ($\theta_4 + \theta_6$) = total angle required to align the orientation of the hand.
- For a given arm configuration, $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ is a set of solutions and
- $(\theta_1, \theta_2, \theta_3, \theta_4 + \pi, -\theta_5, \theta_6 + \pi)$ is another set of solutions.
- The joint angles θ_i are obtained in the following sequence $\theta_1, \theta_3, \theta_2, \theta_4, \theta_5, \theta_6$.

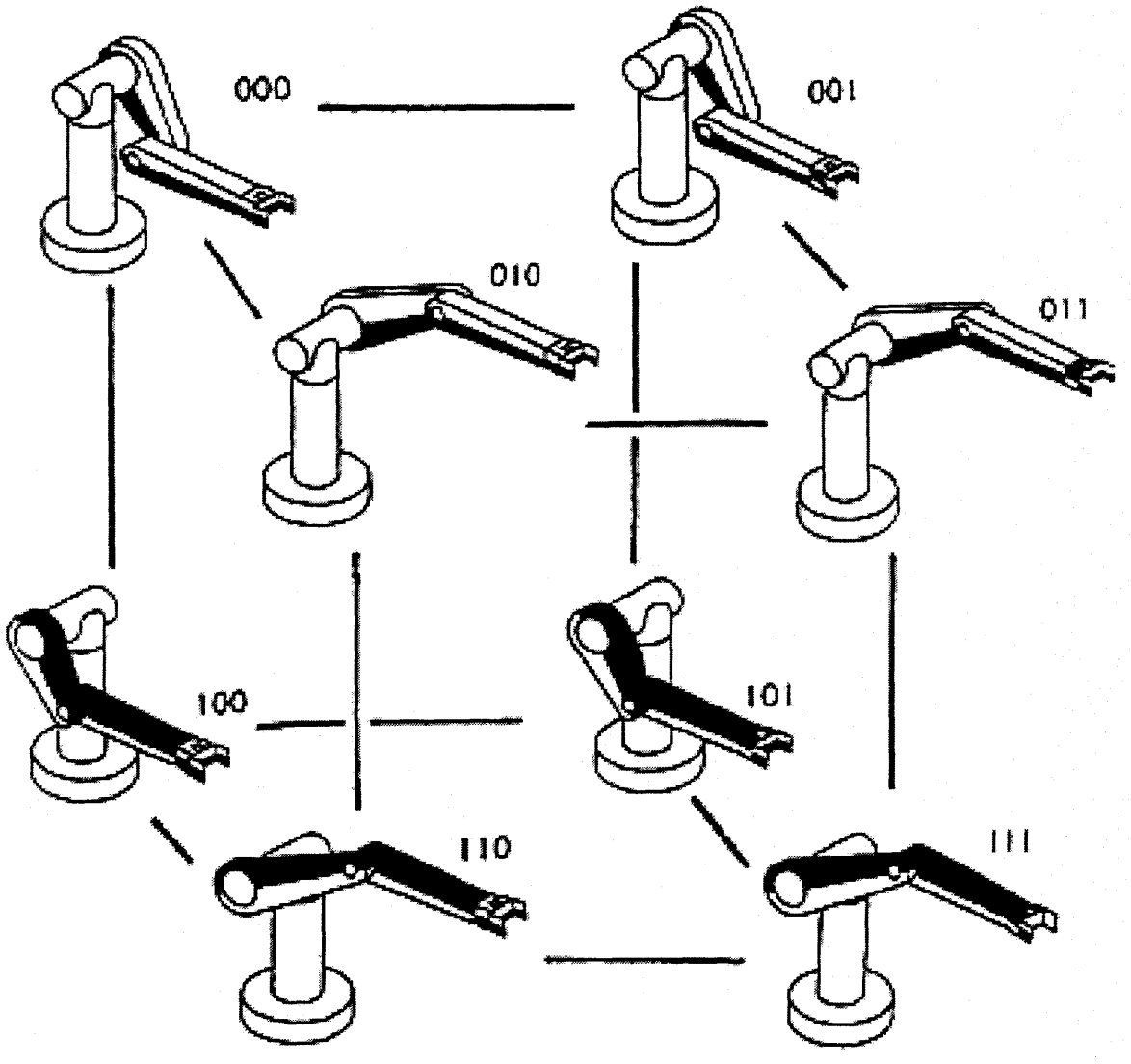


Figure (A.1). Definition of the Puma 560® robot arm position configuration.

A.2. D'Alembert Dynamic Equations for Puma 560® Manipulator

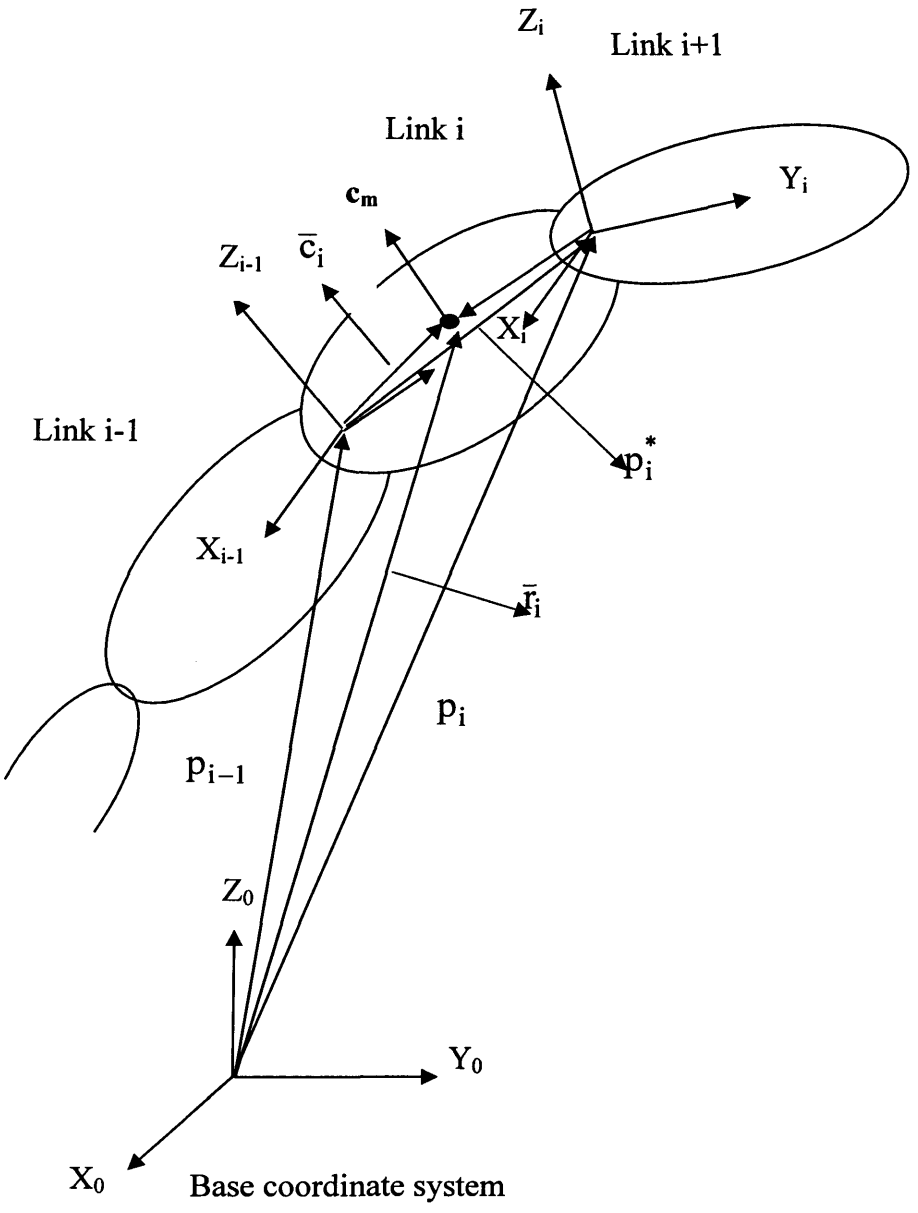


Figure (A.2). Vector definition for D'Alembert equations.

The dynamic equations of any open chain robot manipulator are expressed as:

$$\sum_{j=1}^{n} D_{ij}\ddot{\theta}_j + H_i^{Trans}(\theta,\dot{\theta}) + H_i^{Rot}(\theta,\dot{\theta}) + G_i = \tau_i$$

Where:

$$\begin{aligned} D_{ij} &= D_{ij}^{Rot} + D_{ij}^{Trans} \\ &= \sum_{s=j}^{n}\left\{\left[R_s^0 Z_{i-1}\right]^T I_s \left[R_s^0 Z_{j-1}\right]\right\} + \sum_{s=j}^{n}\left[m_s\left\{Z_{j-1}\times\left[\sum_{j=k}^{s-1} p_k^* + \bar{c}_s\right]\right\}\times\left\{Z_{i-1}\times\left[\bar{r}_s - p_{i-1}\right]\right\}\right] \\ &= \sum_{s=j}^{n}\left\{\left[R_s^0 Z_{i-1}\right]^T I_s \left[R_s^0 Z_{j-1}\right]\right\} \;;\; i \le j,\; i = 1,\ldots,n \\ &+ \sum_{s=j}^{n}\left[m_s\left\{Z_{j-1}\times\left[\bar{r}_s - p_{j-1}\right]\right\}\times\left\{Z_{i-1}\times\left[\bar{r}_s - p_{i-1}\right]\right\}\right] \end{aligned} \tag{A.10}$$

$$H_i^{Trans}(\theta,\dot{\theta}) =$$

$$\sum_{s=i}^{n}\left[m_s\left[\sum_{k=1}^{s-1}\left\{\left\{\left[\sum_{p=1}^{k}\dot{\theta}_p Z_{p-1}\right]\times\left[\left[\sum_{q=1}^{k}\dot{\theta}_q Z_{q-1}\right]\times p_k^*\right]\right\} + \left\{\sum_{p=2}^{k}\left[\left[\sum_{q=1}^{p-1}\dot{\theta}_q Z_{q-1}\right]\times\dot{\theta}_p Z_{p-1}\right]\times p_k^*\right\}\right\}\right]\times\left\{Z_{i-1}\times\left[\bar{r}_s - p_{i-1}\right]\right\}\right]$$

$$+\sum_{s=i}^{n}\left[m_s\left\{\begin{array}{l}\left\{\left[\sum_{p=1}^{s}\dot{\theta}_p Z_{p-1}\right]\times\left[\left[\sum_{q=1}^{s}\dot{\theta}_q Z_{q-1}\right]\times\bar{c}_s\right]\right\}+ \\ \left\{\sum_{p=2}^{s}\left[\left[\sum_{q=1}^{p-1}\dot{\theta}_q Z_{q-1}\right]\times\dot{\theta}_p Z_{p-1}\right]\times\bar{c}_s\right\}\end{array}\right\}\times\left\{Z_{i-1}\times\left[\bar{r}_s - p_{i-1}\right]\right\}\right],\; i = 1,\ldots,n \tag{A.11}$$

$$H_i^{Rot}(\theta,\dot{\theta}) = \sum_{s=i}^{n}\left[\begin{array}{l}\left[R_s^0 Z_{i-1}\right]^T I_s\left\{\sum_{j=1}^{s}\left[\dot{\theta}_j R_s^0 Z_{j-1}\times\left[\sum_{k=j+1}^{s}\dot{\theta}_k R_s^0 Z_{k-1}\right]\right]\right\}+ \\ \left\{R_s^0 Z_{i-1}\times\left[\sum_{p=1}^{s}\dot{\theta}_p R_s^0 Z_{p-1}\right]\right\}^T I_s\left[\sum_{q=1}^{s}\dot{\theta}_q Z_{q-1}\right]\end{array}\right];\; i = 1,\ldots,n \tag{A.12}$$

$$G_i = -g\left[Z_{i-1} \times \sum_{j=i}^{n} m_j \left[\bar{r}_j - p_{i-1} \right] \right];\ i = 1,\ldots,n \qquad \text{(A.13)}$$
$$g = (g_x, g_y, g_z)^T,\ |g| = 9.8062 m/s^2$$

where;

m_i = Mass of link i;

I_i = The inertia about the centre of mass of link i with respect to the base co-ordinate system;

τ_i = The applied torque exerted on link i;

$\bar{r}_s$ = The position vector to the centre of mass of link from the base co-ordinate system;

$\bar{c}_s$ = The position vector of the centre of mass of link s from the (s-1)th co-ordinate frame with reference to the base co-ordinate frame;

I_s = The inertia tensor matrix of link about its centre of mass expressed in the sth co-ordinate system;

R_s^0 = The rotation matrix with reference to the sth co-ordinate frame; $1 \le s \le n$;

Z_{j-1} = The axis of rotation of joint j with reference to the base co-ordinate frame;

The dynamic coefficients D_{ij} and G_i are functions of both the joint variables and inertial parameters of the manipulator, while H_i^{Trans} and H_i^{Rot} are functions of the joint variables, the joint velocities and inertial parameters of the manipulator. These coefficients have the following interpretations:

1. The elements of the D_{ij} matrix are related to the link's inertia of the manipulator. Equation (A.10) reveals the acceleration effects of joint *j* acting on joint *i* where the driving torque τ_i acts. The first term of equation (A.10) indicates the inertial effects of moving link j on joint *i* due to the rotational motion of link *j*, and vice versa. If *i* =*j*, it is the effective inertia felt at joint *i* due to the rotational motion of link *j*, while if $i \neq j$, it is the pseudo products of inertia of link j felt at joint *i* due to the rotational motion of link *j*. The second term has same physical meaning except it is due to the transitional motion of link *j* acting on joint *i*.

2. The $H_i^{Trans}(\theta, \dot{\theta})$ is related to the velocities of the joint variables. Equation (A.11) represents the combined centrifugal and Coriolis reaction torques felt at joint *i* due to the velocities of joints *p* and *q* resulted from the transitional motions of links p and q. The first and third terms of equation (A.11) constitute the centrifugal and Coriolis reaction forces from all the links below link *i* in the kinematic chain due to the transitional motion of the links. If *p*=*q*, then it represents the centrifugal reaction forces felt at joint *i*. If $p \neq q$, then it indicates the Coriolis forces acting on joint *i*. The second and fourth terms of equation (A.11) indicate the Coriolis reaction forces contributed from links below link *i* in the kinematic chain due to the transitional motion of the links.

3. The $H_i^{Rot}(\theta, \dot{\theta})$ is also related to the velocities of the joint variables. Similar to the $H_i^{Trans}(\theta, \dot{\theta})$, equation (A.12) reveals the combined centrifugal and Coriolis reaction torques felt at joint *i* due to the velocities of joints *p* and *q* resulted from the

rotational motion of link p and q. The first term of equation (A.12) indicates purely the Coriolis reaction forces of joints p and q acting on joint i due to the rotational motion of the links. The second term is the combined centrifugal and Coriolis reaction forces acting on joint i, while if $p \neq q$, then it represents the Coriolis forces acting on joint i due to the rotational motion of the links.

4. The coefficient G_i represents the gravity effects acting on joint i from the links above joint i.

For the Puma 560® robot arm, the elements of the D_{ij} matrix come from the transitional and rotational effects of the links. For the first three joints $(\theta_1, \theta_2, \theta_3)$, because of their usually long link length for maximum reach and long distance traveled between initial position and final position, the effects of transitional motion will dominate the rotational motion. In contrast to the first three joints, the rotational effects will dominate for the last three joints. Hence, one can simplify the computation of the D_{ij} matrix by considering only the transitional effects for the first three joints and the rotational effects for the last three joints. Similarly, one can evaluate the contribution of H_i^{Trans} and H_i^{Rot} and eliminate their computations if they are insignificant. The resulting simplified model retains the entire major interaction and coupling reaction forces at a reduced computation time and greatly aids the design of an appropriate control law for controlling the robot arm.

APPENDIX B

Pro/Mechanica Software Interface

B.1. Custom Load Definition

To apply intelligent neuro-fuzzy control to the virtual model created using *Pro/Mechanica®* software, there must be some way to exchange information with the mechanism. A custom load can create measures in C++ to provide useful information about the mechanism to the user and feed-back the driving torques and forces to the *Pro/Mechanica® Motion*'s engine. Proper design of custom load makes it appears as it is a built-in feature of *Pro/Mechanica® Motion*'s engine. When the custom load command is selected, a form appears similar to this one:

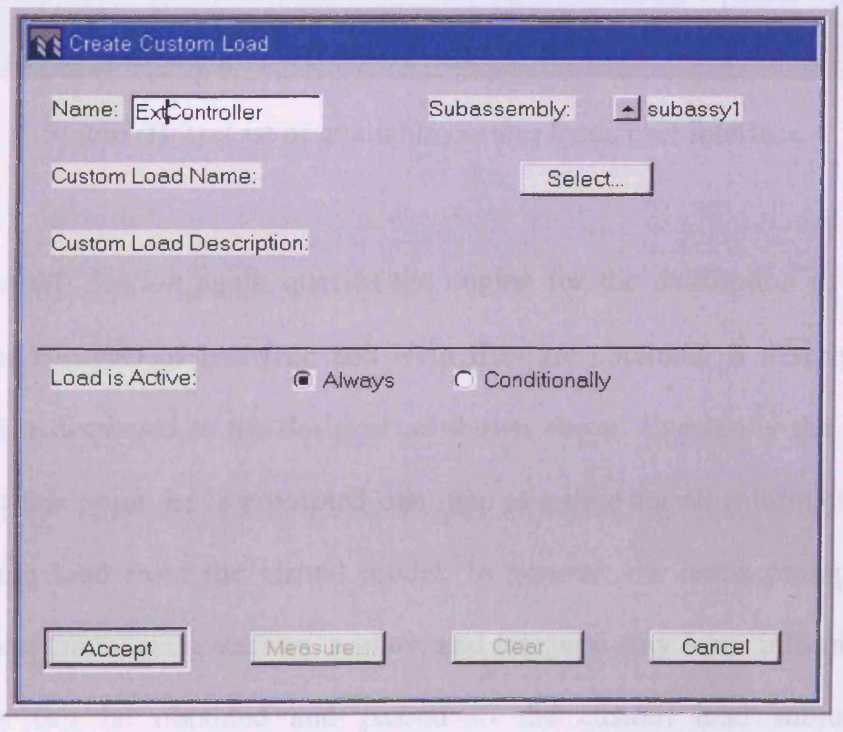


Figure (B.1). Custom load selection user interface

When the user chooses Select, *Pro/Mechanica® Motion* queries its engine for the names of available custom loads. A list of available custom loads appears and the user selects the proper custom load he needs.

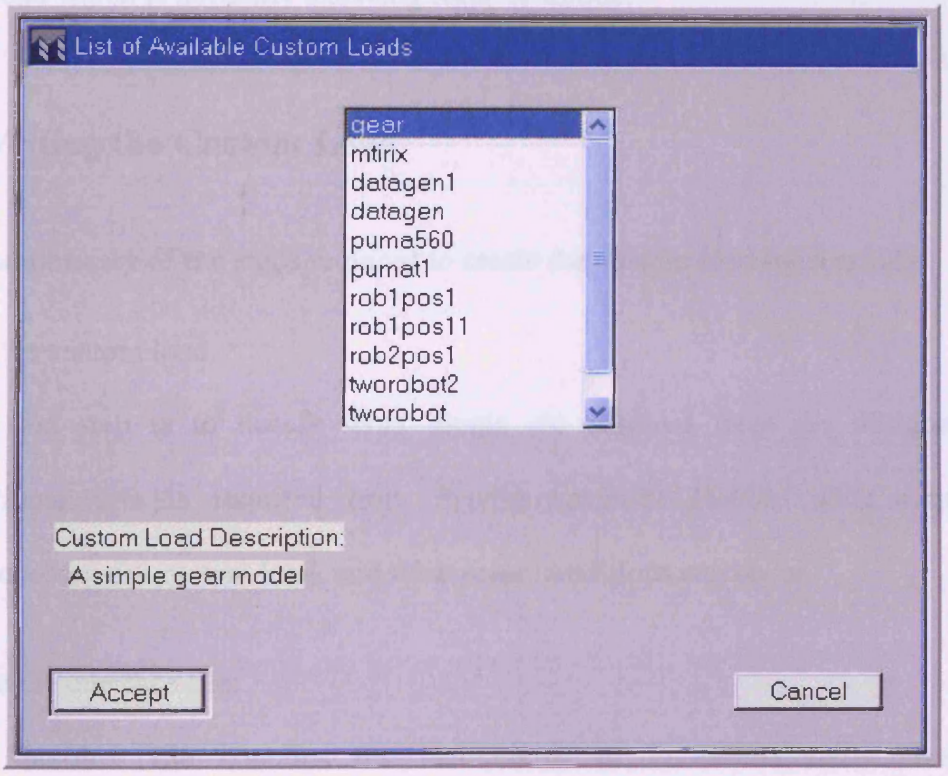


Figure (B.2). List of available custom loads user interface

Pro/Mechanica® Motion again queries the engine for the description of the custom load, and the contents of Interface and Help files are obtained. A description of the custom load is displayed to the designer as shown above. Eventually the user selects (Accept). At this point, he is prompted one item at a time for all information required by the custom load from the virtual model. In general, the items prompted for are bodies, points, joints, axes, vectors, scalars, and integers. Any other information about the prompts can be obtained and passed to the custom load subroutine from

Pro/Mechanica® Motion engine such as orientations, inertia matrices ...etc. In addition, if the custom load produces measures as output, the user is prompted to give names to these measures and one icon identical to the built-in loads icons is produced for each output load which is normally a driving force or torque.

B.2. Writing the Custom Load

This is a summary of the steps required to create the custom load successfully.

1- Plan the custom load.

The first step is to decide what inputs are required from the designer, what additional data is required from *Pro/Mechanica® Motion*, what outputs are produced by the custom load, and what error conditions can occur.

2- Write the interface file.

The Interface file contains the instructions to *Pro/Mechanica® Motion* for implementing the plan. For examples, it tells *Pro/Mechanica® Motion* what kind of data to get from the user and gives the prompts which should be used to request it, what kind of forces or torques coming to the model and where they should be applied, and what text should be used to report error conditions. This file is created using any text editor with the extension "*.ifc". The designer must compile this file (to produce "*.ifo" file) so that *Pro/Mechanica® Motion* can link it in to the model.

3- Write the custom load subroutine.

The Interface file specifies the inputs and outputs to the custom load, the subroutine actually accepts these inputs, processes them, and produces the outputs. The

designer creates a C++ subroutine with a predefined calling sequence. The designer must know the array length in advance. The designer must compile this subroutine (to produce "*.o" file) so that *Pro/Mechanica® Motion* can link it in to the model.

4- Write the help file.

The designer should document his custom load in sufficient details to allow successful use of it. This information can be put in a simple text ASCII file (in the form "*.hlp") that is accessible to *Pro/Mechanica® Motion* engine.

5- Installing the custom load.

Once the custom load is working, it can be installed with the *Pro/Mechanica® Motion* engine so that it is accessible to all users. All names of the files mentioned above should be the same with only the file name extension varying according to the file type.

B.3. Writing the Interface File

The interface file is a simple ASCII® file which can be created using any text editor. Once created, it must be placed under the subdirectory (\ \i486_nt) in the *Pro/Mechanica® Motion* directory and must be compiled from within this location using the command:

\ \i486_nt\ *mmifcc tworobot.ifc >tworobot.txt*

This command will generate the *tworobot.ifo* file from the *tworobot.ifc* file and also a summary file (*tworobot.txt*) containing the interfaced items description. The interface file used for the coordination of two Puma 560® robots is listed below.

Interface file (tworobot.ifc):

```
LOAD {
        DESCRIPTION     "2-PUMA 560 Neuro-Fuzzy Adaptive Synchronizing
                         Controller ";
        VERSION         "1.0";
        LANGUAGE        C;
        SUBROUTINE      tworobot;
PROMPTS {
        link0       "Select the ROBOT#1 Link-0 body"        BODY;
        link1       "Select the ROBOT#1 Link-1 body"        BODY;
        link2       "Select the ROBOT#1 Link-2 body"        BODY;
        link3       "Select the ROBOT#1 Link-3 body"        BODY;
        link4       "Select the ROBOT#1 Link-4 body"        BODY;
        link5       "Select the ROBOT#1 Link-5 body"        BODY;
        link6       "Select the ROBOT#1 Link-6 body"        BODY;
        link7       "Select the Connection Arm body"        BODY;
        link8       "Select the ROBOT#2 Link-0 body"        BODY;
        link9       "Select the ROBOT#2 Link-1 body"        BODY;
        link10      "Select the ROBOT#2 Link-2 body"        BODY;
        link11      "Select the ROBOT#2 Link-3 body"        BODY;
        link12      "Select the ROBOT#2 Link-4 body"        BODY;
        link13      "Select the ROBOT#2 Link-5 body"        BODY;
        link14      "Select the ROBOT#2 Link-6 body"        BODY;
        enefct1     "Select the end-effector#1 point"       POINT;
        enefct2     "Select the end-effector#2 point"       POINT;
        vecta1  "Select the ROB#1 Joint-1 angle vector"     VECTOR link1;
        vecta2  "Select the ROB#1 Joint-2 angle vector"     VECTOR link2;
        vecta3  "Select the ROB#1 Joint-3 angle vector"     VECTOR link3;
        vectj1  "Select the ROB#1 Joint-1 torque vector"    VECTOR link1;
        vectj2  "Select the ROB#1 Joint-2 torque vector"    VECTOR link2;
        vectj3  "Select the ROB#1 Joint-3 torque vector"    VECTOR link3;
        vecta4  "Select the ROB#2 Joint-1 angle vector"     VECTOR link9;
        vecta5  "Select the ROB#2 Joint-2 angle vector"     VECTOR link10;
        vecta6  "Select the ROB#2 Joint-3 angle vector"     VECTOR link11;
        vectj4  "Select the ROB#2 Joint-1 torque vector"    VECTOR link9;
        vectj5  "Select the ROB#2 Joint-2 torque vector"    VECTOR link10;
        vectj6  "Select the ROB#2 Joint-3 torque vector"    VECTOR link11;
        }
STATES {
        err1_int;
        err2_int;
```

```
        err3_int;
        err4_int;
        err5_int;
        err6_int;
        }
INPUTS {
        orint0 ORIENT link0;
        orint1 ORIENT link1;
        orint2 ORIENT link2;
        orint3 ORIENT link3;
        orint4 ORIENT link4;
        orint5 ORIENT link5;
        orint6 ORIENT link8;
        orint7 ORIENT link9;
        orint8 ORIENT link10;
        orint9 ORIENT link11;
        orint10 ORIENT link12;
        orint11 ORIENT link13;
        axis110 XFORM vecta1 link0;
        axis121 XFORM vecta2 link1;
        axis132 XFORM vecta3 link2;
        axis210 XFORM vecta4 link8;
        axis221 XFORM vecta5 link9;
        axis232 XFORM vecta6 link10;
        }
OUTPUTS {
        torq1 "torque on ROBOT#1 link1 joint" TORQUE link1 vectj1;
        torq2 "torque on ROBOT#1 link2 joint" TORQUE link2 vectj2;
        torq3 "torque on ROBOT#1 link3 joint" TORQUE link3 vectj3;
        torq4 "torque on ROBOT#2 link1 joint" TORQUE link9 vectj4;
        torq5 "torque on ROBOT#2 link2 joint" TORQUE link10 vectj5;
        torq6 "torque on ROBOT#2 link3 joint" TORQUE link11 vectj6;
        err1_der "derivative of ROBOT#1 error-1" DERIV  err1_int;
        err2_der "derivative of ROBOT#1 error-2" DERIV  err2_int;
        err3_der "derivative of ROBOT#1 error-3" DERIV  err3_int;
        err4_der "derivative of ROBOT#2 error-1" DERIV  err4_int;
        err5_der "derivative of ROBOT#2 error-2" DERIV  err5_int;
        err6_der "derivative of ROBOT#2 error-3" DERIV  err6_int;
        }
        }
```

Summary file (tworobot.txt):

Description : 2-PUMA 560 Neuro-Fuzzy Adaptive Synchronizing Controller
Load type : Ordinary
Language : C
Version : 1.0
Subroutine : tworobot

Prompts:

Name : link0
Type : BODY
Prompt : "Select the ROBOT#1 Link-0 body"
Index : 0

Name : link1
Type : BODY
Prompt : "Select the ROBOT#1 Link-1 body"
Index : 1

Name : link2
Type : BODY
Prompt : "Select the ROBOT#1 Link-2 body"
Index : 2

Name : link3
Type : BODY
Prompt : "Select the ROBOT#1 Link-3 body"
Index : 3

Name : link4
Type : BODY
Prompt : "Select the ROBOT#1 Link-4 body"
Index : 4

Name : link5
Type : BODY
Prompt : "Select the ROBOT#1 Link-5 body"
Index : 5

Name : link6
Type : BODY
Prompt : "Select the ROBOT#1 Link-6 body"
Index : 6

Name : link7
Type : BODY
Prompt : "Select the Connection Arm body"
Index : 7

Name : link8
Type : BODY
Prompt : "Select the ROBOT#2 Link-0 body"
Index : 8

Name : link9

Type : BODY
Prompt : "Select the ROBOT#2 Link-1 body"
Index : 9

Name : link10
Type : BODY
Prompt : "Select the ROBOT#2 Link-2 body"
Index : 10

Name : link11
Type : BODY
Prompt : "Select the ROBOT#2 Link-3 body"
Index : 11

Name : link12
Type : BODY
Prompt : "Select the ROBOT#2 Link-4 body"
Index : 12

Name : link13
Type : BODY
Prompt : "Select the ROBOT#2 Link-5 body"
Index : 13

Name : link14
Type : BODY
Prompt : "Select the ROBOT#2 Link-6 body"
Index : 14

Name : enefct1
Type : POINT
Prompt : "Select the end-effector#1 point"
Index : 15

Name : enefct2
Type : POINT
Prompt : "Select the end-effector#2 point"
Index : 16

Name : vecta1
Type : VECTOR
Prompt : "Select the ROB#1 Joint-1 angle vector"
Index : 17
Frame : link1
Default : 0 0 0

Name : vecta2

Type : VECTOR
Prompt : "Select the ROB#1 Joint-2 angle vector"
Index : 18
Frame : link2
Default : 0 0 0

Name : vecta3
Type : VECTOR
Prompt : "Select the ROB#1 Joint-3 angle vector"
Index : 19
Frame : link3
Default : 0 0 0

Name : vectj1
Type : VECTOR
Prompt : "Select the ROB#1 Joint-1 torque vector"
Index : 20
Frame : link1
Default : 0 0 0

Name : vectj2
Type : VECTOR
Prompt : "Select the ROB#1 Joint-2 torque vector"
Index : 21
Frame : link2
Default : 0 0 0

Name : vectj3
Type : VECTOR
Prompt : "Select the ROB#1 Joint-3 torque vector"
Index : 22
Frame : link3
Default : 0 0 0

Name : vecta4
Type : VECTOR
Prompt : "Select the ROB#2 Joint-1 angle vector"
Index : 23
Frame : link9
Default : 0 0 0

Name : vecta5
Type : VECTOR
Prompt : "Select the ROB#2 Joint-2 angle vector"
Index : 24
Frame : link10
Default : 0 0 0

Name : vecta6
Type : VECTOR
Prompt : "Select the ROB#2 Joint-3 angle vector"
Index : 25
Frame : link11
Default : 0 0 0

Name : vectj4
Type : VECTOR
Prompt : "Select the ROB#2 Joint-1 torque vector"
Index : 26
Frame : link9
Default : 0 0 0

Name : vectj5
Type : VECTOR
Prompt : "Select the ROB#2 Joint-2 torque vector"
Index : 27
Frame : link10
Default : 0 0 0

Name : vectj6
Type : VECTOR
Prompt : "Select the ROB#2 Joint-3 torque vector"
Index : 28
Frame : link11
Default : 0 0 0

States:
Name : err1_int
Index : 29

Name : err2_int
Index : 30

Name : err3_int
Index : 31

Name : err4_int
Index : 32

Name : err5_int
Index : 33

Name : err6_int
Index : 34

Inputs:

Name : orint0
Type : ORIENT
Index : 35
InputId : link0

Name : orint1
Type : ORIENT
Index : 36
InputId : link1

Name : orint2
Type : ORIENT
Index : 37
InputId : link2

Name : orint3
Type : ORIENT
Index : 38
InputId : link3

Name : orint4
Type : ORIENT
Index : 39
InputId : link4

Name : orint5
Type : ORIENT
Index : 40
InputId : link5

Name : orint6
Type : ORIENT
Index : 41
InputId : link8

Name : orint7
Type : ORIENT
Index : 42
InputId : link9

Name : orint8
Type : ORIENT
Index : 43
InputId : link10

Name : orint9
Type : ORIENT
Index : 44
InputId : link11

Name : orint10
Type : ORIENT
Index : 45
InputId : link12

Name : orint11
Type : ORIENT
Index : 46
InputId : link13

Name : axis110
Type : XFORM
Index : 47
InputId : vecta1
InputId : link0

Name : axis121
Type : XFORM
Index : 48
InputId : vecta2
InputId : link1

Name : axis132
Type : XFORM
Index : 49
InputId : vecta3
InputId : link2

Name : axis210
Type : XFORM
Index : 50
InputId : vecta4
InputId : link8

Name : axis221
Type : XFORM
Index : 51
InputId : vecta5
InputId : link9

Name : axis232
Type : XFORM

Index : 52
InputId : vecta6
InputId : link10

Outputs:
Name : torq1
Type : TORQUE
Desc : "torque on ROBOT#1 link1 joint"
Index : 0
InputId : link1
InputId : vectj1

Name : torq2
Type : TORQUE
Desc : "torque on ROBOT#1 link2 joint"
Index : 1
InputId : link2
InputId : vectj2

Name : torq3
Type : TORQUE
Desc : "torque on ROBOT#1 link3 joint"
Index : 2
InputId : link3
InputId : vectj3

Name : torq4
Type : TORQUE
Desc : "torque on ROBOT#2 link1 joint"
Index : 3
InputId : link9
InputId : vectj4

Name : torq5
Type : TORQUE
Desc : "torque on ROBOT#2 link2 joint"
Index : 4
InputId : link10
InputId : vectj5

Name : torq6
Type : TORQUE
Desc : "torque on ROBOT#2 link3 joint"
Index : 5
InputId : link11
InputId : vectj6

Name : err1_der
Type : DERIV
Desc : "derivative of ROBOT#1 error-1"
Index : 6
InputId : err1_int

Name : err2_der
Type : DERIV
Desc : "derivative of ROBOT#1 error-2"
Index : 7
InputId : err2_int

Name : err3_der
Type : DERIV
Desc : "derivative of ROBOT#1 error-3"
Index : 8
InputId : err3_int

Name : err4_der
Type : DERIV
Desc : "derivative of ROBOT#2 error-1"
Index : 9
InputId : err4_int

Name : err5_der
Type : DERIV
Desc : "derivative of ROBOT#2 error-2"
Index : 10
InputId : err5_int

Name : err6_der
Type : DERIV
Desc : "derivative of ROBOT#2 error-3"
Index : 11
InputId : err6_int

B.4. Writing the Custom Load Subroutine

The interface subroutine is a C++ file which can be created using any text editor. Once created, it must be placed under the subdirectory (\ \i486_nt) in the *Pro/Mechanica*

Motion directory and must be compiled using C++ compiler to produce the corresponding object file in the same location.

Inputs as seen from C++				
Input[i] is	**INTS[i][d]**	**REALS[i][d]**	**VECS[i][d][c]**	**Notes**
BODY	d=0: body tag	0: mass	0: COM loc 1: COM vel 2: angvel	(1) & (2)
ORIENT	0: body tag		0: orient1 1: orient2 2: orient3	(3)
INERTIA	0: body tag		0: xx xy xz 1: xy yy yz 2: xz yz zz	(4)
POINT	0: point tag 1: body tag		0: location 1: velocity	(5)
JOINT	0: joint tag 1: joint type 2: body1 tag 3: body2 tag			(10)
AXIS	0: joint tag 1: axis num 2: rot/trans	0: position 1: velocity	0: axis vec	(6)
VECTOR			0: vec	(7)
SCALAR		0: real		
INTEGER	0: int			
DISTANCE		0: dist 1: sep. vel.		(8)
XFORM			0: vec	(9)
STATE		0: real		

Table (B.1). Input variables arrays as seen from C++

Then the object file must be compiled from the first parent directory using:

\ \ *cl /J /c /Foi486_nt\tworobot.o tworobot.c*

This command will generate the file tworobot.o from the tworobot.c file. The interface subroutine exchanges data with Pro/Mechanica Motion engine in the form of arrays as shown in tables (B.1) and (B.2).

Outputs as seen from C++				
Output[i] is	**REALOUT[i]**	**VECOUT[i]**	**PTLOCS[i]**	**Notes**
FORCE	signed frc mag			(11)
TORQUE	signed trq mag			(11)
TWPOT	signed frc mag			(12)
GENFRC		frc vector	application point	(13)
GENTRQ		trq vector		(14)
AXIS	signed load			(15)
MEASURE	measure val			
DERIV	state deriv			
ICOND	state initial condition			(16)

Table (B.2). Output variables arrays as seen from C++

Notes:

1- COM loc (centre of mass location) is ground frame measure numbers of vector from ground origin to body COM.

2- Angular velocity is the angular velocity of the body with respect to ground, expressed in the ground frame.

3- Orientation relates body local frame to ground frame, that is, orient1 is the ground frame measure numbers of a vector currently aligned with the first local frame axis.

4- These are the moments and products of inertia about the body's local frame axes.

5- Point location and velocity are w.r.t. the ground origin and expressed in the ground frame. The axis type is 1 for rotational and 2 for translational.

6- The joint axis vector is a unit vector parallel to the axis and expressed in the ground frame.

7- The vector is a unit vector expressed in the ground frame.

8- Distance is non-negative, separation velocity is positive when points are separating, negative when approaching.

9- The transformed vector is expressed in the local frame of the specified body.

10- The following table shows the numerical value passed in for each joint type and the ordering of the joint axes (t=translational, r=rotational, -=part of ball joint).

Joint type	No.	1	2	3	4	5	6
pin	0	r					
slider	1	r					
U-joint	2	r	r				
gimbal	3	r	r	r			
cylinder	4	r	t				
planar	5	t	t	r			
ball	6	-	-	-			
free	7	t	t	t	-	-	-
sixdof	8	t	t	t	r	r	r
bearing	9	t	r	r	r		
weld	10						

Table (B.3). Numerical values representing joint type

11- For fixed or follower force or torque, Pro/Mechanica multiplies the returned quantity by the fixed or follower vector, then applies the result to the appropriate point or body.

12- For twopt (two points) force, Pro/Mechanica multiplies the returned quantity by a unit vector aligned with (pt2-pt1) and applies the result at pt2. It applies the negative of this value to pt1.

13- For the general force, Pro/Mechanica applies the force vector (expressed in the global frame) to the point of the specified body which is coincident with the application point (relative to the global origin and expressed in the global frame).

14- For general torque, Pro/Mechanica applies the torque vector (expressed in the global frame) to the specified body.

15- For an axis load, Pro/Mechanica applies the specified force or torque as a joint axis force or torque.

16- Pro/Mechanica reference ICOND outputs only after it calls with flag=1.

The interface subroutine used for the coordination of two Puma 560 robots is too big to be listed here, only the overall structure is outlined below.

Interface subroutine (tworobot.c):

```
/*    Custom load subroutine for PRO/MECHANICA "tworobot" custom load.      */
/*       2-PUMA 560 Neuro-Fuzzy Adaptive Synchronizing Controller           */
/*              - Copyright 2003 Cardiff School of Engineering              */

#define  NIN          52  /* Number of Input Variables from PRO/MECHANICA  */
#define  link0        0       /* Link-0  Body        ON ROBOT#1 ARM  Link-0  */
#define  link1        1       /* Link-1  Body        ON ROBOT#1 ARM  Link-1  */
#define  link2        2       /* Link-2  Body        ON ROBOT#1 ARM  Link-2  */
```

```
#define  link3       3     /* Link-3  Body      ON ROBOT#1 ARM  Link-3   */
#define  link4       4     /* Link-4  Body      ON ROBOT#1 ARM  Link-4   */
#define  link5       5     /* Link-5  Body      ON ROBOT#1 ARM  Link-5   */
#define  link6       6     /* Link-6  Body      ON ROBOT#1 ARM  Link-6   */
#define  link7       7     /* Link-7  Body      ON CONNECTION    ARM    */
#define  link8       8     /* Link-11 Body      ON ROBOT#2 ARM  Link-0   */
#define  link9       9     /* Link-12 Body      ON ROBOT#2 ARM  Link-1   */
#define  link10      10    /* Link-13 Body      ON ROBOT#2 ARM  Link-2   */
#define  link11      11    /* Link-14 Body      ON ROBOT#2 ARM  Link-3   */
#define  link12      12    /* Link-15 Body      ON ROBOT#2 ARM  Link-4   */
#define  link13      13    /* Link-16 Body      ON ROBOT#2 ARM  Link-5   */
#define  link14      14    /* Link-17 Body      ON ROBOT#2 ARM  Link-6   */
#define  enefct1     15   /* End-effector of ROBOT#1 Location in Ground Frame */
#define  enefct2     16   /* End-effector of ROBOT#2 Location in Ground Frame */
#define  vecta1      17    /* ROBOT#1 Joint-1 Angle Vector to Ground Frame    */
#define  vecta2      18    /* ROBOT#1 Joint-2 Angle Vector to Ground Frame    */
#define  vecta3      19    /* ROBOT#1 Joint-3 Angle Vector to Ground Frame    */
#define  vectj1      20    /* ROBOT#1 Joint-1 Torque Vector to Ground Frame   */
#define  vectj2      21    /* ROBOT#1 Joint-2 Torque Vector to Ground Frame   */
#define  vectj3      22    /* ROBOT#1 Joint-3 Torque Vector to Ground Frame   */
#define  vecta4      23    /* ROBOT#2 Joint-1 Angle Vector to Ground Frame    */
#define  vecta5      24    /* ROBOT#2 Joint-2 Angle Vector to Ground Frame    */
#define  vecta6      25    /* ROBOT#2 Joint-3 Angle Vector to Ground Frame    */
#define  vectj4      26    /* ROBOT#2 Joint-1 Torque Vector to Ground Frame   */
#define  vectj5      27    /* ROBOT#2 Joint-2 Torque Vector to Ground Frame   */
#define  vectj6      28    /* ROBOT#2 Joint-3 Torque Vector to Ground Frame   */
#define  err1_int    29     /* ROBOT1 Error-1 Integration (Integral of err1_der)*/
#define  err2_int    30     /* ROBOT1 Error-2 Integration (Integral of err2_der)*/
#define  err3_int    31     /* ROBOT1 Error-3 Integration (Integral of err3_der)*/
#define  err4_int    32     /* ROBOT2 Error-1 Integration (Integral of err4_der)*/
#define  err5_int    33     /* ROBOT2 Error-2 Integration (Integral of err5_der)*/
#define  err6_int    34     /* ROBOT2 Error-3 Integration (Integral of err6_der)*/
#define  orint0      35    /* ROBOT#1 Link-0 orientation matrix to ground frame*/
#define  orint1      36    /* ROBOT#1 Link-1 orientation matrix to ground frame*/
#define  orint2      37    /* ROBOT#1 Link-2 orientation matrix to ground frame*/
#define  orint3      38    /* ROBOT#1 Link-3 orientation matrix to ground frame*/
#define  orint4      39    /* ROBOT#1 Link-4 orientation matrix to ground frame*/
#define  orint5      40    /* ROBOT#1 Link-5 orientation matrix to ground frame*/
#define  orint6      41    /* ROBOT#2 Link-0 orientation matrix to ground frame*/
#define  orint7      42    /* ROBOT#2 Link-1 orientation matrix to ground frame*/
#define  orint8      43    /* ROBOT#2 Link-2 orientation matrix to ground frame*/
#define  orint9      44    /* ROBOT#2 Link-3 orientation matrix to ground frame*/
#define  orint10     45    /* ROBOT#2 Link-4 orientation matrix to ground frame*/
#define  orint11     46    /* ROBOT#2 Link-5 orientation matrix to ground frame*/
#define  axis110     47    /* ROBOT#1 Joint-1 Vector to Link-0 Local Frame    */
#define  axis121     48    /* ROBOT#1 Joint-2 Vector to Link-1 Local Frame    */
#define  axis132     49    /* ROBOT#1 Joint-3 Vector to Link-2 Local Frame    */
```

```
#define  axis210      50      /* ROBOT#2 Joint-1 Vector to Link-0 Local Frame    */
#define  axis221      51      /* ROBOT#2 Joint-2 Vector to Link-1 Local Frame    */
#define  axis232      52      /* ROBOT#2 Joint-3 Vector to Link-2 Local Frame    */
#define  NOUT  12      /* Number of Output Variables to PRO/MECHANICA     */
#define  torq1   0     /* Follower Torque on ROBOT#1 Joint-1 Through vectj1*/
#define  torq2   1     /* Follower Torque on ROBOT#1 Joint-2 Through vectj2*/
#define  torq3   2     /* Follower Torque on ROBOT#1 Joint-3 Through vectj3*/
#define  torq4   3     /* Follower Torque on ROBOT#2 Joint-1 Through vectj4*/
#define  torq5   4     /* Follower Torque on ROBOT#2 Joint-2 Through vectj5*/
#define  torq6   5     /* Follower Torque on ROBOT#2 Joint-3 Through vectj6*/
#define  err1_der  6    /* ROBOT#1 Error-1 Derivative (d/dt of err1_int)    */
#define  err2_der  7    /* ROBOT#1 Error-2 Derivative (d/dt of err2_int)    */
#define  err3_der  8    /* ROBOT#1 Error-3 Derivative (d/dt of err3_int)    */
#define  err4_der  9    /* ROBOT#2 Error-1 Derivative (d/dt of err4_int)    */
#define  err5_der  10 /* ROBOT#2 Error-2 Derivative (d/dt of err5_int)    */
#define  err6_der  11 /* ROBOT#2 Error-3 Derivative (d/dt of err6_int)    */

/********************** Start of Main Program **********************/
void tworobot(flag,time,ints,reals,vecs,realout,vecout,ptlocs,err)
int   flag,ints[NIN][4],*err;
double time,reals[NIN][2],vecs[NIN][3][3];
double realout[NOUT],vecout[NOUT][3],ptlocs[NOUT][3];
{
.
.
.
x1 = (vecs[vecta1][0][0]);y1 = (vecs[vecta1][0][1]);z1 = (vecs[vecta1][0][2]);
x2 = (vecs[vecta2][0][0]);y2 = (vecs[vecta2][0][1]);z2 = (vecs[vecta2][0][2]);
x3 = (vecs[vecta3][0][0]);y3 = (vecs[vecta3][0][1]);z3 = (vecs[vecta3][0][2]);
x4 = (vecs[vecta4][0][0]);y4 = (vecs[vecta4][0][1]);z4 = (vecs[vecta4][0][2]);
x5 = (vecs[vecta5][0][0]);y5 = (vecs[vecta5][0][1]);z5 = (vecs[vecta5][0][2]);
x6 = (vecs[vecta6][0][0]);y6 = (vecs[vecta6][0][1]);z6 = (vecs[vecta6][0][2]);
x1t = (vecs[axis110][0][0]);y1t = (vecs[axis110][0][1]);z1t = (vecs[axis110][0][2]);
x2t = (vecs[axis121][0][0]);y2t = (vecs[axis121][0][1]);z2t = (vecs[axis121][0][2]);
x3t = (vecs[axis132][0][0]);y3t = (vecs[axis132][0][1]);z3t = (vecs[axis132][0][2]);
x4t = (vecs[axis210][0][0]);y4t = (vecs[axis210][0][1]);z4t = (vecs[axis210][0][2]);
x5t = (vecs[axis221][0][0]);y5t = (vecs[axis221][0][1]);z5t = (vecs[axis221][0][2]);
x6t = (vecs[axis232][0][0]);y6t = (vecs[axis232][0][1]);z6t = (vecs[axis232][0][2]);
px1 = (vecs[enefct1][0][0]);py1 = (vecs[enefct1][0][1]);pz1 = (vecs[enefct1][0][2]);
px2 = (vecs[enefct2][0][0]);py2 = (vecs[enefct2][0][1]);pz2 = (vecs[enefct2][0][2]);
realout[torq1] = tor1;realout[torq2] = tor2;realout[torq3] = tor3;
realout[torq4] = tor4;realout[torq5] = tor5;realout[torq6] = tor6;
//realout[err1_der] = error1;realout[err2_der] = error2;realout[err3_der] = error3;
//realout[err4_der] = error4;realout[err5_der] = error5;realout[err6_der] = error6;
}
```

APPENDIX C

Hardware Interface Specifications

C.1. Interface Card Specifications

For the purpose of interfacing the experimental set-up explained in chapter (6) to the host computer for the control algorithm testing purposes, an ADLINK® DAQ/PXI-2501 interface card with the following specifications has been used.

Analog Output (AO)

- Number of channels: 4-channels
- DA converter: AD7945
- Max update rate: 1MS/sec.
- Resolution: 12 bits
- FIFO buffer size: 8K byte
- Voltage reference: internal 10V or external up to ±10V
- Output range: Unipolar and Bipolar
- Settling time: 2μs.
- Offset error: ±2mv max

Analog Input (AI)

- Number of channels: 8-channels
- AD converter: LTC1416
- Max sampling rate: 400KS/sec.
- Resolution: 14 bits

- FIFO buffer size: 2K byte
- Input range: Bipolar ±10V or Unipolar up to +10V
- Settling time: 2μs.
- Offset error: ±1mv max

General Purpose Digital I/O (G.P. DIO)

- Number of channels: 24 programmable Input/Output
- Compatibility: TTL/CMOS
- Input voltage: Logic Low: 0.8V max, Logic High: 2.0V max
- Output voltage: Logic Low: 0.5V max, Logic High: 2.7V min

General Purpose Timer/Counter (G.P. TC)

- Number of channels: 2 UP/Down Timer/Counters
- Resolution: TTL/CMOS
- Resolution: 16 bits
- Clock source: Internal or external
- Max source frequency: 10MHz

C.2. Filters and Power Amplifiers Specifications

For each motor, a filter, an anti-aliasing filter, and a power amplifier is used to smoothen the input measurement, the continuous-time control, and to drive the motor. All the filters are chosen to be first-order RC filters and have the same cut-off frequency of 15 Hz. All anti-aliasing filters are chosen to be first-order RC filters as well with the same cut-off frequency of 33 Hz. All filters are cascaded with non-inverting mode operational amplifier with a gain of 2 for $R_i = R_f = 10K\Omega$. The

operational amplifiers are biased by ±15V DC external supply. Figure (C.1) shows the circuit diagram of one filter. The power amplifiers are LM12 linear series operational amplifiers. They are biased through ±13.8V, ±13.0A peak, ±10.0A continuous, regulated DC power supply. Figure (C.2) shows the circuit diagram for one power amplifier. Table (C.1) lists the design values for circuit's elements.

Filters		Anti-aliasing Filters	
R	C	R	C
10KΩ	1.0μf	10KΩ	47μf
Power Amplifiers			
R_1	C_1	R_2	C_2
1.0KΩ	220nf	4.0KΩ	220pf

Table (C.1). Design values for circuit's elements.

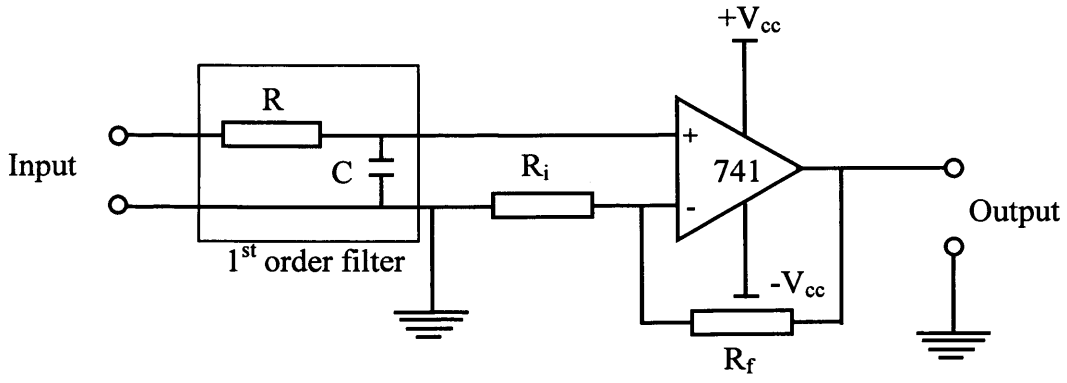


Figure (C.1). Circuit diagram for one motor filter/anti-aliasing filter.

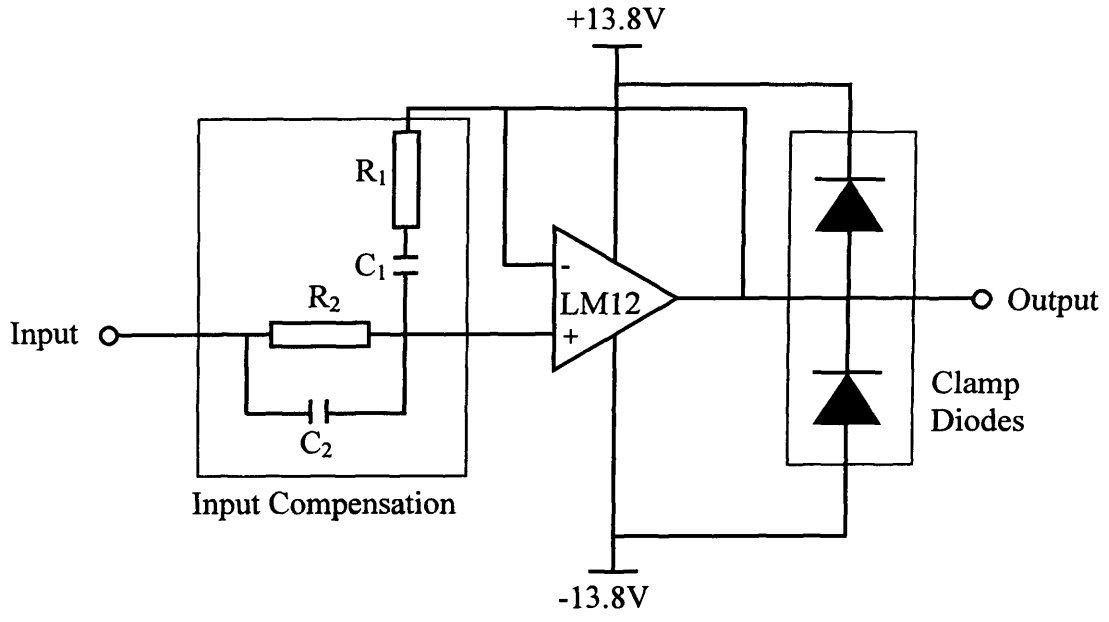


Figure (C.1). Circuit diagram for one motor power amplifier.

C.3. D.C. Motors Specifications

Each robot link is powered by permanent magnet DC motor equipped with position decoding potentiometers and suitable planetary gear head to increase the motor developed torque. Both of torque speed and torque current relationship for such type of motors are linear.

$$V_a - E_b = i_a R_a + L_a \frac{di_a}{dt} \tag{C.1}$$

$$E_b = k_\phi \omega \tag{C.2}$$

$$T = k_\phi i_a \tag{C.3}$$

$$T - T_L = \beta\omega + J\frac{d\omega}{dt} \tag{C.4}$$

where V_a is the input voltage of the DC motor, E_b is the motor armature back educed emf, i_a is the armature current, R_a is the armature resistance, L_a is the armature inductance, k_ϕ is the magnetisation constant, ω is the motor speed in rad/sec, T is the motor developed torque, T_L is the load torque, β is the equivalent friction constant at motor shaft, and J is the equivalent inertia constant at motor shaft. The motor equivalent circuit parameters are listed in table (C.2).

Rated Power	15 watt	Starting Current	4010 m.Amp.
Rated Voltage	12 volt	Armature Resistance	2.99 Ohm
No Load Speed	4590 rpm	Torque Constant	24.1 m.N.m./Amp.
Stall Torque	96.8 m.N.m.	Speed Constant	396 rpm/volt
ΔSpeed/ΔTorque	49.1 rpm/m.N.m.	Armature Inductance	0.21 m.H.
No Load Current	115 m.Amp.	Position Ratio	12 Degree/volt

Table (C.2). Motors equivalent circuit parameters.