Chapter 1
INTRODUCTION

1.1 General Background

With reference to the tasks assigned to an industrial robot, one important issue is to determine the motion of the joints and the end effector of the robot. Therefore, the purpose of the robot arm control, as Fu et al. wrote in one of the classical works on robotics, is to maintain the dynamic response of the manipulator in accordance with some prespecified performance criterion [19].

Among the early robots of the first generation, non-servo control techniques, such as bang-bang control and sequence control, were used. These robots move from one position to another under the control of limit switches, relays, or mechanical stops [19].

During the 1970s, a great deal of work was focused on including such internal state sensors as encoders, potentiometers, tachogenerators, etc., into the robot controller to facilitate manipulative operation [28, 64]. Since then, feedback control techniques have been applied for servoing robot manipulators.

Up till now, the majority of practical approaches to the industrial robot arm controller design, use traditional techniques, such as PD or PID controllers, by treating each joint of the manipulator as a simple linear servomechanism. In designing these kinds of controllers, the non-linear, coupled and time-varying dynamics of the mechanical part of the robot manipulator system are completely ignored, or are dealt with as disturbances. These methods generally give satisfactory performance when the
robot operates at a low speed. However, when the links are moving simultaneously and at a high speed, the non-linear coupling effects and the interaction forces between the manipulator links, may degrade the performance of the overall system and increase the tracking errors. The disturbances and uncertainties in a task cycle, may also reduce the tracking quality of robot manipulators. Thus, these methods are only suitable for relatively slow manipulator motion and for limited-precision tasks [30].

The Computed Torque Control (CTC) is commonly used in the research community. The CTC control law has the ability to make the error asymptotically stable if the dynamics of the robot are exactly known [43]. However, manipulators are subject to structured and/or unstructured uncertainty. Structured uncertainty is defined as the case of a correct dynamic model but with parameter uncertainty due to tolerance variances in the manipulator link properties, unknown loads, inaccuracies in the torque constants of the actuators, and others. Unstructured uncertainty describes the case of unmodeled dynamics which result from the presence of high-frequency modes in the manipulator, neglected time-delays and non-linear friction. It has been widely recognised that the tracking performance of the CTC method in high-speed operations is severely affected by the structured and unstructured uncertainties. To cope with the problem, some adaptive approaches have been proposed to maintain the tracking performance of the robotic manipulator in the presence of structured uncertainty [12]. Some other researchers have also tried to incorporate the neural network into the controller design and good results were reported [29,36].

In this thesis, another control strategy, namely, the Model Predictive Control (MPC), which had been used in industrial process control previously, is proposed for robot control.

It has been demonstrated that MPC can be applied to Multi-Input-Multi-Output (MIMO) [33] and non-linear systems [36]. MPC has been regarded as the only methodology that can handle process constraints in a systematic way [57]. By using predictions MPC can also deal with prescheduled reference trajectories during the design of the controller [57]. All above features naturally allow one to consider investigating the feasibility of MPC in robot control.
1.2 Model Predictive Control

The incipient interest in the applications of MPC can be dated back to the late 1970s. In 1978, Richalet et al.\cite{richalet1978}, presented the Model Predictive Heuristic Control (MPHC) method in which an impulse response model was used to predict the effect at the output of the future control actions. The control actions to be exerted on the system are determined by minimising the error between the predicted and the desired outputs of the system, subject to the operation restriction \cite{richalet1978}. In 1980, Culter and Ramaker proposed the Dynamic Matrix Control (DMC) method in which a step response model was used \cite{culter1980}. During 1980s, MPC quickly became popular particularly in chemical process industry due to the simplicity of the algorithm and to the use of the impulse or step response model, which is preferred, as being more intuitive and requiring no previous information for its identification \cite{marlin1997}.

The basic structure of the MPC is illustrated in Figure 1.1, in which a model is used to predict the future plant outputs, based on past and current values of plant status and on the proposed optimal future control actions. These control actions are calculated by the optimiser by taking into account the cost function (where the future tracking error is considered) as well as the constraints. The objective of the MPC is minimisation of the predicted output errors by adjusting control actions over a given horizon.

![Figure 1.1](image_url) Basic structure of MPC
As a consequence, the predictive model plays a decisive role in determining the optimal control actions. Therefore the chosen model must be capable of capturing the system's basic dynamic modes, so as to precisely predict the future outputs of the system. In addition, the model has to be simple enough to implement practically.

Most current MPCs are based on a linear mathematical description of the system, such as the above mentioned impulse and step response models and the transfer function model, because these models require no previous information for identification. Furthermore, the linear system theory is better developed.

However, a linear model, no matter how well structured and complicated it is, may only be acceptable in the case where the system is working around the operating point. If the system is highly non-linear, such as a robot manipulator, controls based on the prediction of a linear model, may result in unacceptable actuator response. In some cases, remarkable static errors exist, and in other cases, oscillation or even instability may occur [51]. Therefore, some kind of non-linear models should be used to describe the behaviour of a highly non-linear system.

1.3 Non-linear System Modelling and Prediction using Neural Networks

In recent years, neural networks (NNs) have become an attractive means of constructing the models of complex non-linear systems. This is because NNs have an inherent ability to approximate any arbitrary non-linear function. NNs allow many of the ideas of system identification and adaptive control originally applied to linear (or linearised) systems, to be generalised to cope with more severe non-linearity. Thus, NN provides a possible way to identify the model of a non-linear system effectively.
Based on their ability to learn sophisticated non-linear relationships, the effectiveness of NNs has been successfully demonstrated by previous studies when employed to model complex non-linear systems [6,7,42,58,60,63].

NNs generally consist of a number of interconnected processing elements called neurons. These processing elements are functions consisting of a summing junction and a non-linear operation (also known as activation function). How the neuron connections are arranged and the nature of the connections determine the structure of a network. How the strengths of the connections are adjusted or trained to achieve a desired overall behaviour of the network, is governed by its learning algorithm. Thus NNs can be classified according to their structures and learning algorithms.

In terms of their structures, NNs can be divided into two major categories: feedforward NNs (FNNs) and recurrent NNs (RNNs).

FNN is the structure in which signals flow from the input layer to the output layer via unidirectional forward connections (Figure 4.2). The neurons are connected from one layer to the next, but not within the same layer. The most commonly used FNN is the Multi-layer Perceptrons (MLP) [45]. Others are the Cerebellar Model Articulation Control (CMAC) network [23], the Learning Vector Quantisation (LVQ) network [34], the Group Method of Data Handling (GMDH) network [23], Radial Basis Function (RBF) network [22]. It has been proven [24] that a multilayer feedforward network can approximate any non-linear relationship between the inputs and the outputs to any predefined accuracy. FNN will be discussed in more details in Chapter 4.

RNN is another network structure in which the outputs of some neurons are fed back to the same neurons or to neurons in preceding layers. Signals can flow in both forward and backward directions (Figure 4.3). The most notable RNN is the Hopfield network [25]. Some others are the Jordan network [30], the Elman network [14], and the time-lag network [62]. RNN will be discussed in more details in Chapter 4.
Various NNs have been used to identify and predict non-linear system performance. Pham & Liu used the modified Elman network with dynamic backpropagation to identify a high-order non-linear system [44]. Chen et al. proposed an RBF network in which there are two-layer perceptrons for the identification of an MIMO non-linear system [6]. Morris and Manser used an ANN with an adaptive backpropagation algorithm and a lookup table to find the inverse kinematics of a robot manipulator moving along pre-defined trajectories [39]. Moreover, various results have also been reported using NNs for robot dynamics modelling. Eskandarian et al. used CMAC networks to model robot dynamics [15]. To construct such a CMAC model, five branch CMACs were used to represent five joint angle dependent terms, i.e., inertia matrix, Coriolis and centrifugal torques, gravity torques, external load torques, and friction, of the robot dynamic equations. By using these CMACs together with joint positions and velocities, and input torques, the joint accelerations can be calculated. Khemaissia & Morris estimated link parameters by using an FNN structure [32]. Kosmatopoulos et al. modeled robot dynamics by employing an FNN with dynamic elements [34]. Nebot and Fang tried to use an RNN to predict the behaviour of a robot for a certain number of time steps [41]. Fang and Dissanayake also proposed a method in which the dynamic behaviour of the robot is obtained by using the MLP to emulate the robot joint accelerations, based on the joint states (positions and velocities) and torques, together with a solver for Ordinary Differential Equations (ODEs) [18].

However, up to now there is still no pragmatic NN method to model the robot dynamics with an excellent predictive characteristic that is required by an MPC. Nebot and Fang's method [41] gives poor results for multiple step prediction. The method proposed by Fang and Dissanayake [18] is difficult to implement due to its use of accelerations, because the measurement of acceleration signals with high accuracy is difficult in practice. Furthermore, many other modelling methods in the literature either used simplified equations for robot dynamics, where important practical effects such as friction forces, joint position and velocity dependent torque limits were ignored [35], or impractically isolated each term in the dynamic equation, such as inertia matrix, Coriolis and centrifugal torques, gravity torques, external load torques and friction [15]. However these simplified models, can not precisely predict the future output of the robot because they differ greatly from the real robot dynamic equations,
and the isolation of the dynamic equation into above-mentioned separated terms is difficult to realise in practice.

1.4 Neuro-MPC Controllers for MIMO Non-linear Systems

To overcome the problems resulting from using linear models in the MPC, some researchers have tried to extend MPC to use non-linear models. The technique Joseph et al. used is to obtain a non-linear model through system analysis to help the control calculation arrive at an appropriate action [31]. The predictive methods using such non-linear models have also been made adaptive by estimating the parameters of the model that are most likely to change [31]. This requires the model to be of the correct structure, otherwise steady-state offsets from the setpoints may result despite parameter adaptation. Selecting such an accurate structure requires significant analysis and a prior knowledge of the system. However, due to the complexity of the practical systems, or lack of knowledge of critical parameters of the systems, in many cases it is impossible to obtain a suitable physically founded system model in an analytical way.

An alternative method to overcome the above problems is to use NNs as non-linear black-box models to predict the dynamic behaviour of the systems. There have also been many reports on using NNs for non-linear system control. Psaltis et al. proposed general learning architectures to learn the plant inverse dynamics for the FNN based feedforward controller [45]. Narendra and Parthasarathy developed generalised NN models for both identification and control of non-linear systems [40]. Hunt and Sbarbaro used NNs for non-linear internal control [26]. Fang and Dissanayake used an NN in time-optimal trajectory planning [17]. An extensive review on applying NNs to control has also been given by Hunt et al. [27].

For system identification, a feedforward MLP network could be used in two ways. One is that the network uses the past and current system inputs and system
outputs to predict the future system output. Such arrangement is referred to as a serial identification method, see Figure 1.2 (with 's' switched on). In contrast to the serial method, the network can also be arranged completely parallel to the system, which is referred to as a parallel identification method. Under this circumstance, the past system outputs are replaced by the outputs from the model itself. In the parallel arrangement as shown in Figure 1.2 (with the 'p' switched on), an NN model contains delayed recurrent connections from its output neurons back to its own input neurons. An NN with such recurrent connection is referred to as an external recurrent network [59].

![Diagram of serial and parallel identification](image)

- : Time-delay unit
- s: serial method
- p: parallel method

**Figure 1.2 Serial and parallel identification**

When incorporated with an MPC controller, NNs have also been applied to chemical process control. Draeger et al. [11] used the FNN as the non-linear prediction model in an extended DMC-algorithm to control the pH value in a laboratory-scale
neutralisation reactor. The FNN prediction model was arranged serially to the system. Current and the last four previous pH values and the corresponding five values of the impulse frequency (which determines the sodium hydroxide flow) are fed into the network to predict the pH value in the next time step. Experimental results show that an MPC controller based on this FNN model performs better than the conventional PI controller. In [59], Su and McAvoy also investigated the use of an NN model based on the parallel training approach for MPC control of a biological wastewater treatment system and a catalytic reforming system.

1.5 The Aim and Research Methods of the Thesis

The aim of this thesis is to investigate the feasibility of applying the MPC method to track predefined robot trajectories by using a neural network as the non-linear predictive model. The data used to train the NN model are obtained for the running of a robot model. In this thesis a computer simulation model is used as the robot model. This computer simulation model is built according to a modified PUMA robot. The simulation of the proposed neuro-MPC method will be carried out on this simulation model as well.

This thesis is organised into six chapters. Chapter 2 deals with the setting up of a robot simulation model. Here the PUMA industrial robot is chosen as the prototype of this simulation model, because it is the best known in the research community. To investigate the stability and robustness of the proposed control method on a robot using simulation, the robot should be simulated as accurately as possible. In Chapter 2, such factors as frictional forces, gear backlash, and other uncertainties which are often neglected by other researchers, will be incorporated into the simulation model to reflect the reality.

Chapter 3 investigates the feasibility of using model predictive control for robots. First the most representative MPC algorithm is reviewed. Then an MPC structure for robot trajectory control is proposed. The validity of the MPC method is
investigated by using a perfect model, and a model with both structured and unstructured uncertainties, in different predictive horizons. The effectiveness of the MPC when some constraints are added to control torques, will also be studied.

In Chapter 4 the method of constructing an NN predictor of a PUMA robot dynamics for the MPC control is discussed. One approach of using an FNN for modelling the robot forward dynamics is also provided. The FNN model is trained by using the data obtained from the robot’s dynamic responses to a set of random sinusoidal torque inputs within specific frequency ranges. To check the concordance of the responses of the robot dynamic system and the FNN model, a set of sinusoidal signals are used to excite the system and the FNN model. These sinusoidal signals are different from those used to train the FNN model, but are within the specified frequency range. Finally, the feasibility of a multi-step-ahead prediction using an FNN model is also investigated.

Chapter 5 integrates the methods developed in Chapters 3 and 4 together. In this chapter the position control of the robot is performed by using the MPC method discussed in Chapter 3, based on the neural network modelling method developed in Chapter 4. A series of simulations are carried out to perform the comparison between the model predictive control based on the neural network model and the model predictive control using a robot model described by the dynamic equations with parameter errors. Finally in Chapter 6 the conclusions of the thesis are given and some future research recommendations are offered.
Chapter 2

Simulation of a Robot System

2.1 Introduction

The aim of the robot simulation is to develop a complete mathematical representation of an open-loop robot system by incorporating actuator effects, gear backlash and dynamic equation with inertia, centrifugal and Coriolis, frictional and gravitational, and other uncertainties. After having been built up, this simulation model will be treated as the 'real robot'. All measurements and simulations through this thesis will be performed on this model.

A PUMA industrial robot is chosen as the prototype of the simulation model, because the dynamic characteristics of this kind of robots have been intensively researched. It is one of the best known robots to the research community.

Usually a PUMA robot has five or six degrees of freedom (DOF). For simplicity in this thesis it is assumed that the gripper and the wrist unit of the robot are fixed to the forearm forming a rigid body, that is, there are no relative movements among them. This assumption simplifies the PUMA to a three DOF robot. By doing so the simulation and computation time can be reduced greatly. In addition, due to the relatively small size and mass of the gripper and wrist, the effect of this simplification to the overall dynamics characteristics will not be very severe in comparison to the dynamics of PUMA's original arms.

It is also assumed that all the robot arms are rigid and there will be no elasticity modes existing in operation. Thus the dynamic equation of the robot can be derived based on the rigid body dynamics theory.
2.2 The Simulation Software

There are many software packages that could be used to perform dynamic simulation, such as Simulink, Adams, NIMBUS, etc. In this thesis Simulink is chosen to simulate the robot dynamic behaviour because it is easily accessible in this School. Simulink is a simulation environment attached to a software package named ‘Matlab’. Under the Simulink environment, mathematical representations are shown as a series of blocks interconnected by lines showing the flows of data. These blocks may contain a combination of standard library elements such as summers, gain blocks, function generators and many others. The users are also permitted to develop their own blocks with transfer functions of any type, such as the dynamic equations expressed in sections 2.4 and 2.7. Furthermore, the output from the Simulink allows for the production of plots and the export of data for analysis with other packages.

Simulink can simulate linear, non-linear, continuous-time, discrete-time, multivariable, and multirate systems [54]. It also allows the simulation of the performance of a dynamic system over time. The length of time to be simulated is limited only by the amount of memory the computer possesses and integration step size. Therefore, this software package is useful for emulating the non-linear characteristics of the robot dynamics.

2.3 PUMA Robot

The PUMA robot was initially built by the Unimation Inc. (now defunct), to specifications developed by General Motors. PUMA stands for Programmable Universal Machine for Assembly. This robot system was the first commercially available industry robot. It had all electric drives, and a reasonably sophisticated controller. Its controller could be disconnected and replaced by another custom-built controller. For these reasons,
PUMA became one of the most popular with robotic researchers around the world. Figure 2.1 shows an illustration of the PUMA robot. Table 2.1 lists the structural parameters and their values for a PUMA robot. These parameters were obtained from [18].

Figure 2.1 Illustration of a PUMA robot [19]
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Parameter values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$ (m)</td>
<td>0.0</td>
</tr>
<tr>
<td>$d_2$ (m)</td>
<td>0.1491</td>
</tr>
<tr>
<td>$a_1$ (m)</td>
<td>0.0</td>
</tr>
<tr>
<td>$a_2$ (m)</td>
<td>0.0</td>
</tr>
<tr>
<td>$a_3$ (m)</td>
<td>0.4318</td>
</tr>
<tr>
<td>$a_5$ (m)</td>
<td>-0.0203</td>
</tr>
<tr>
<td>Mass of the first link $M_1$ (kg)</td>
<td>12.96</td>
</tr>
<tr>
<td>Mass of the second link $M_2$ (kg)</td>
<td>22.34</td>
</tr>
<tr>
<td>Mass of the third link (including the 4th and 5th link) $M_3$ (kg)</td>
<td>6.07</td>
</tr>
<tr>
<td>Center of mass for the first link along x ($X_1$, m)</td>
<td>0.0</td>
</tr>
<tr>
<td>Center of mass for the first link along y ($Y_1$, m)</td>
<td>0.0388</td>
</tr>
<tr>
<td>Center of mass for the first link along z ($Z_1$, m)</td>
<td>0.3088</td>
</tr>
<tr>
<td>Center of mass for the 2nd link along x ($X_2$, m)</td>
<td>-0.3289</td>
</tr>
<tr>
<td>Center of mass for the 2nd link along y ($Y_2$, m)</td>
<td>0.005</td>
</tr>
<tr>
<td>Center of mass for the 2nd link along z ($Z_2$, m)</td>
<td>0.2038</td>
</tr>
<tr>
<td>Center of mass for the 3rd link along x ($X_3$, m)</td>
<td>0.0204</td>
</tr>
<tr>
<td>Center of mass for the 3rd link along y ($Y_3$, m)</td>
<td>0.0137</td>
</tr>
<tr>
<td>Center of mass for the 3rd link along z ($Z_3$, m)</td>
<td>0.1244</td>
</tr>
<tr>
<td>Moment of inertia for the first link along x ($I_{1x}$, $kg$-$m^2$)</td>
<td>2.82</td>
</tr>
<tr>
<td>Moment of inertia for the first link along y ($I_{1y}$, $kg$-$m^2$)</td>
<td>0.24</td>
</tr>
<tr>
<td>Moment of inertia for the first link along z ($I_{1z}$, $kg$-$m^2$)</td>
<td>2.82</td>
</tr>
<tr>
<td>Moment of inertia for the 2nd link along x ($I_{2x}$, $kg$-$m^2$)</td>
<td>1.596</td>
</tr>
<tr>
<td>Moment of inertia for the 2nd link along y ($I_{2y}$, $kg$-$m^2$)</td>
<td>3.636</td>
</tr>
<tr>
<td>Moment of inertia for the 2nd link along z ($I_{2z}$, $kg$-$m^2$)</td>
<td>4.056</td>
</tr>
<tr>
<td>Moment of inertia for the 3rd link along x ($I_{3x}$, $kg$-$m^2$)</td>
<td>0.3778</td>
</tr>
<tr>
<td>Moment of inertia for the 3rd link along y ($I_{3y}$, $kg$-$m^2$)</td>
<td>0.3754</td>
</tr>
<tr>
<td>Moment of inertia for the 3rd link along z ($I_{3z}$, $kg$-$m^2$)</td>
<td>0.012</td>
</tr>
<tr>
<td>Gear ratio of the first link drive chain ($r_1$)</td>
<td>62.61</td>
</tr>
<tr>
<td>Gear ratio of the second link drive chain ($r_2$)</td>
<td>107.82</td>
</tr>
<tr>
<td>Gear ratio of the third link drive chain ($r_3$)</td>
<td>53.71</td>
</tr>
<tr>
<td>Moment of inertia of the motor armature driving the first joint ($I_{m1}$)</td>
<td>0.0002</td>
</tr>
<tr>
<td>Moment of inertia of the motor armature driving the second joint ($I_{m2}$)</td>
<td>0.0002</td>
</tr>
<tr>
<td>Moment of inertia of the motor armature driving the third joint ($I_{m3}$)</td>
<td>0.0002</td>
</tr>
<tr>
<td>Viscous friction coefficient of the first joint ($v_1$, N-s)</td>
<td>0.01</td>
</tr>
<tr>
<td>Viscous friction coefficient of the second joint ($v_2$, N-s)</td>
<td>0.005</td>
</tr>
<tr>
<td>Viscous friction coefficient of the third joint ($v_3$, N-s)</td>
<td>0.003</td>
</tr>
<tr>
<td>Coulomb friction of the first joint ($c_1$, N-m)</td>
<td>0.6</td>
</tr>
<tr>
<td>Coulomb friction of the second joint ($c_2$, N-m)</td>
<td>0.5</td>
</tr>
<tr>
<td>Coulomb friction of the third joint ($c_3$, N-m)</td>
<td>0.9</td>
</tr>
<tr>
<td>Approximated Coulomb friction in low velocity of the first joint ($v_{11}$)</td>
<td>600</td>
</tr>
<tr>
<td>Approximated Coulomb friction in low velocity of the second joint ($v_{12}$)</td>
<td>500</td>
</tr>
<tr>
<td>Approximated Coulomb friction in low velocity of the third joint ($v_{13}$)</td>
<td>900</td>
</tr>
<tr>
<td>Rated voltage of the motor driving the first joint ($V_{m1}$ V)</td>
<td>40</td>
</tr>
<tr>
<td>Rated voltage of the motor driving the second joint ($V_{m2}$ V)</td>
<td>40</td>
</tr>
<tr>
<td>Rated voltage of the motor driving the third joint ($V_{m3}$ V)</td>
<td>40</td>
</tr>
</tbody>
</table>
2.4 PUMA Robot Arm Dynamics

The general form of the robot arm dynamic equations with \( n \) links can be obtained as following, based on the Lagrange-Euler formulation for rigid body dynamics [19]:

\[
\tau = D(\theta) \dot{\theta} + h(\theta, \dot{\theta}) + f(\dot{\theta}) + C(\theta) + T_d
\]  

(2-1)

where \( \tau \) is an \( n \times 1 \) vector of joint torques/forces supplied by the actuators; \( D(\theta) \) is the \( n \times n \) manipulator inertia matrix; \( h(\theta, \dot{\theta}) \) is an \( n \times 1 \) vector representing centrifugal and Coriolis effects; \( f(\dot{\theta}) \) is \( n \times 1 \) vector representing friction forces, both Coulomb and viscous; \( C(\theta) \) is \( n \times 1 \) vector representing gravity force; \( T_d \) is \( n \times 1 \) vector of unknown terms arising from unmodelled dynamics and external disturbances; and, \( \theta, \dot{\theta} \) and \( \ddot{\theta} \) are \( n \times 1 \) vectors of generalised joint positions, velocities and accelerations respectively.

In the case of a three DOF PUMA robot, variables in (2-1) are stated respectively as follow:

\[
\tau = \begin{bmatrix} \tau_{i1} & \tau_{i2} & \tau_{i3} \end{bmatrix}^T
\]  

(2-2)

\( \tau_{ii} (i=1,2,3) \): torque at joint \( i \); \( T^* \) stands for matrix transpose.

\[
\theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix}^T
\]  

(2-3A)

\[
\dot{\theta} = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_3 \end{bmatrix}^T
\]  

(2-3B)

\[
\ddot{\theta} = \begin{bmatrix} \ddot{\theta}_1 & \ddot{\theta}_2 & \ddot{\theta}_3 \end{bmatrix}^T
\]  

(2-3C)

\( \theta, \dot{\theta}, \ddot{\theta} \) (\( i = 1,2,3 \)): angular position, velocity and acceleration of joint \( i \) respectively.

\[
D(\theta) = \begin{bmatrix} d_{11}(\theta) & d_{12}(\theta) & d_{13}(\theta) \\ d_{12}(\theta) & d_{22}(\theta) & d_{23}(\theta) \\ d_{13}(\theta) & d_{23}(\theta) & d_{33}(\theta) \end{bmatrix} = \begin{bmatrix} \ddot{d}_{11}(\theta) & \ddot{d}_{12}(\theta) & \ddot{d}_{13}(\theta) \\ \ddot{d}_{12}(\theta) & \ddot{d}_{22}(\theta) & \ddot{d}_{23}(\theta) \\ \ddot{d}_{13}(\theta) & \ddot{d}_{23}(\theta) & \ddot{d}_{33}(\theta) \end{bmatrix} + \begin{bmatrix} \dddot{d}_{11}(\theta) & \dddot{d}_{12}(\theta) & \dddot{d}_{13}(\theta) \\ \dddot{d}_{12}(\theta) & \dddot{d}_{22}(\theta) & \dddot{d}_{23}(\theta) \\ \dddot{d}_{13}(\theta) & \dddot{d}_{23}(\theta) & \dddot{d}_{33}(\theta) \end{bmatrix}
\]

\[
= \ddot{D} + \dddot{D}(\theta)
\]  

(2-4)
(2-4) shows that the manipulator inertia matrix \( D(\theta) \) could be separated into a constant part \( \bar{D} \), and a variable \( \theta \)-related part \( \tilde{D}(\theta) \). The elements \( d_{ij} = \bar{d}_{ij} + \tilde{d}_{ij}(\theta) \) \((i,j = 1,2,3)\) in \( D(\theta) \) are scalar values. The calculation of \( d_{ij} \) can be found in Appendix A.2 \( (\bar{d}_{ij} \) is constant part, \( \tilde{d}_{ij}(\theta) \) is variable \( \theta \)-related part).

\[
\begin{align*}
    h(\theta, \dot{\theta}, \ddot{\theta}) &= \begin{bmatrix} h_1(\theta, \dot{\theta}) \\ h_2(\theta, \dot{\theta}) \\ h_3(\theta, \dot{\theta}) \end{bmatrix} = \begin{bmatrix} \dot{\theta}^T H_{1,\theta, \dot{\theta}} \\ \dot{\theta}^T H_{2,\theta, \dot{\theta}} \\ \dot{\theta}^T H_{3,\theta, \dot{\theta}} \end{bmatrix}  \quad (2-5A) \\
    H_{i,\theta} &= \begin{bmatrix} h_{i11} & h_{i12} & h_{i13} \\ h_{i21} & h_{i22} & h_{i23} \\ h_{i31} & h_{i32} & h_{i33} \end{bmatrix}  \quad (2-5B)
\end{align*}
\]

where \( h_{ik} \) \((i,j,k = 1,2,3)\) is scalar and the calculation of \( h_{ik} \) can be found in Appendix A.3.

\[
    \mathbf{f}(\dot{\theta}) = \begin{bmatrix} f_1(\dot{\theta}) \\ f_2(\dot{\theta}) \\ f_3(\dot{\theta}) \end{bmatrix}  \quad (2-6)
\]

The discussion about \( f_i \) \((i = 1,2,3)\) can be found in section 2.6.

\[
    \mathbf{c}(\theta) = \begin{bmatrix} c_1(\theta) \\ c_2(\theta) \\ c_3(\theta) \end{bmatrix}  \quad (2-7)
\]

The calculation of \( c_i \) \((i = 1,2,3)\) can be found in Appendix A.4.

\[
    T_d = \begin{bmatrix} T_{d1} & T_{d2} & T_{d3} \end{bmatrix}^T  \quad (2-8)
\]

\( T_d \) \((i = 1,2,3)\) denotes all unmodelled uncertainties and disturbances.

Therefore, the dynamic equation of a three d.o.f. PUMA robot is:

\[
\begin{bmatrix} r_n \\ r_{12} \\ r_{13} \end{bmatrix} = \begin{bmatrix} d_{11}(\theta) & d_{12}(\theta) & d_{13}(\theta) \\ d_{12}(\theta) & d_{22}(\theta) & d_{23}(\theta) \\ d_{13}(\theta) & d_{23}(\theta) & d_{33}(\theta) \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} h_1(\theta, \dot{\theta}) \\ h_2(\theta, \dot{\theta}) \\ h_3(\theta, \dot{\theta}) \end{bmatrix} + \begin{bmatrix} f_1(\dot{\theta}) \\ f_2(\dot{\theta}) \\ f_3(\dot{\theta}) \end{bmatrix} + \begin{bmatrix} c_1(\theta) \\ c_2(\theta) \\ c_3(\theta) \end{bmatrix} + \begin{bmatrix} T'_{d1} \\ T'_{d2} \\ T'_{d3} \end{bmatrix}  \quad (2-9)
\]

A routine named robdyn.m listed in Appendix A.5 is used to calculate the PUMA robot dynamics.
2.5 Actuator Model

This section describes the method that is used to build the actuator model in a single joint of a robot arm, assuming that the robot is electrically actuated. An analytical description of a DC actuator has been well established in the literature. An excellent treatment is given by Fu et al. [19] and this will be followed in the remainder of this section.

Before going into details, a few important variables are introduced in Table 2-2. They are armature voltage, armature inductance, back electromotive force, etc., which will be used to build up the actuator model.

- DC motor

Electrically driven manipulators are generally constructed with a DC permanent magnet torque motor for each joint. Basically the DC torque motor is a permanent magnet, armature excited, continuous rotation motor.

The torque produced by a DC motor is proportional to the armature current when the motor is operated in its linear range

\[ \tau_{mi} = k_{mi}i_i \quad (i = 1,2,3) \quad (2-11) \]

Where \( k_{mi} \) is known as the motor-torque proportional constant in N-m/A.

When the motor is rotating it acts like a generator and a voltage develops across the armature. This voltage is called back electromotive force (emf), which is proportional to a given armature angular velocity as:

\[ e_{bi}(t) = k_{bi}\dot{\theta}_{mi} \quad (i = 1,2,3) \quad (2-12) \]

\( k_{bi} \) is a proportionality constant in V/\text{S/ rad}. 
Table 2-2

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_a$</td>
<td>Armature voltage</td>
<td>volt</td>
</tr>
<tr>
<td>$L_a$</td>
<td>Armature inductance</td>
<td>Henry</td>
</tr>
<tr>
<td>$R_a$</td>
<td>Armature resistance</td>
<td>ohms</td>
</tr>
<tr>
<td>$i_a$</td>
<td>Armature current</td>
<td>amperes</td>
</tr>
<tr>
<td>$e_b$</td>
<td>Back electromotive force</td>
<td>volt</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Torque delivered by the motor</td>
<td>N-m</td>
</tr>
<tr>
<td>$\theta_m$</td>
<td>Angular displacement of the motor shaft</td>
<td>radians</td>
</tr>
<tr>
<td>$\theta_i$</td>
<td>Angular displacement of the load shaft</td>
<td>radians</td>
</tr>
<tr>
<td>$J_m$</td>
<td>Moment of inertia of the motor shaft</td>
<td>kg-m$^2$</td>
</tr>
<tr>
<td>$f_m$</td>
<td>Viscous-friction coefficient of the motor shaft</td>
<td>N-s</td>
</tr>
<tr>
<td>$J_L$</td>
<td>Moment of inertia of the load shaft</td>
<td>kg-m$^2$</td>
</tr>
<tr>
<td>$f_L$</td>
<td>Viscous-friction coefficient of the load shaft</td>
<td>N-s</td>
</tr>
</tbody>
</table>

- Equivalent Motor Circuit [19]

![Equivalent circuit of an armature - controlled DC motor](image)

**Figure 2.2** Equivalent circuit of an armature - controlled DC motor [19]

From Figure 2.2 an armature-controlled DC motor circuit can be described by a first-order differential equation given by:

\[
\begin{align*}
V_{ai}(t) - e_{ai}(t) &= L_{ai} \frac{di_{ai}(t)}{dt} + R_{ai}i_{ai}(t) \quad (2-13A) \\
V_{ai}(t) - k_{bi}\dot{\theta}_{ai}(t) &= L_{ai} \frac{di_{ai}(t)}{dt} + R_{ai}i_{ai}(t) \quad (2-13B)
\end{align*}
\]

\[i = 1,2,3.\]
• Connection of motor and load shaft

The following analysis is for one single joint. Without loss of generality, subscript $i (i = 1, 2, 3)$ is omitted for convenience.

![Diagram of gear train](image)

**Figure 2.3** Analysis of the gear train [19]

From Figure 2.3:

- On the load shaft: $\tau_l = Fr_l$  \hspace{1cm} (2-14)

- On the motor shaft: $\tau'_l = F'r_{in}$  \hspace{1cm} (2-15)

Where $F$, $F'$ represent the action force and reaction force between the motor gear and load gear. Obviously $F = F'$ in magnitude.
From (2-14) and (2-15):

\[
\frac{\tau_i}{\tau_i'} = \frac{r_i}{r_m} = n > 1
\]  

(2-16)

Applying Newton’s Second Law (without considering Coulumb friction) on motor shaft:

\[
\tau_m - \tau_i' - f_m \dot{\theta}_m = J_m \ddot{\theta}_m
\]  

(2-17)

From (2-16) and (2-17) we have:

\[
\tau_i = n[\tau_m - (J_m \ddot{\theta}_m + f_m \dot{\theta}_m)]
\]  

(2-18)

Without considering gear backlash, we have the relationship:

\[
\theta_m r_m = \theta_i r_i \quad (2-19A)
\]

that is:

\[
\frac{\theta_m}{\theta_i} = \frac{r_m}{r_i} \quad (2-19B)
\]

\[
\theta_m = n \theta_i \quad (2-19C)
\]

Thus from (2-18):

\[
\tau_i = n[\tau_m - (J_m n \ddot{\theta}_i + f_m n \dot{\theta}_i)]
\]  

(2-20)

- **Summary**

All above discussions can be summed up in Figure 2.4, where S denotes a differentiator.
2.6 Friction Model

Friction is present in power transmission elements such as gears and screws as well as in bearings, seals, hydraulic components and electric motors. For example, friction in bearings is a function of bearing size, type and design. Additional factors include speed, load type and magnitude as well as lubricant viscosity and flow [61]. While friction can be a function of many variables, it has been shown to be highly repeatable [13]. Thus it is possible for function modelling and parameter identification.

Friction models in individual components could be described by the friction-velocity curve. The characteristic friction-velocity curve for hard materials separated by liquid lubricants is referred to as the Strubeck curve shown as in Figure 2.5 [31].

![Friction-Velocity Curve](image)

Figure 2.5 Strubeck Friction-Velocity Curve [3]

Direct simulation of the behaviour shown in figure 2.5 is difficult. In this thesis the stick-slip effect in the friction-velocity model in figure 2.5 is ignored. The robot friction simulation is performed based on the modelling of both Coulomb and viscous friction described in the followings.
Independent of the area of contact, the Coulomb friction force always opposes relative motion and is proportional to the normal force of the contact \[ \text{[1]} \]. This force can be expressed as:

\[
F_{\text{Coulomb}} = \mu F_N \text{sgn}(\text{velocity})
\]  

(2-21)

where \( \mu \) is the coefficient of friction and \( F_N \) is the normal force. The sign operator is defined as:

\[
\text{sgn}(x) = \begin{cases} 
+1, & x > 0 \\
0, & x = 0 \\
-1, & x < 0 
\end{cases}
\]  

(2-22)

Due to its dependence on the sign of velocity, the Coulomb friction is discontinuous at zero velocity. In addition, the normal forces in the robot components vary with joint positions, velocities and accelerations. This indicates that the governing differential equations are discontinuous in the highest order derivatives terms.

To simplify the Coulomb friction model some researchers proposed to use a Coulomb friction coefficient \( c = \mu \bar{F}_N \) substituting the effect of \( \mu \) and \( F_N \) \[13\]. Where \( \bar{F}_N \) is the average of \( F_N \) within the whole motion range. So the friction force of a robot can be calculated as following when considering both Coulomb and viscous friction:

\[
f(\dot{\theta}) = c \text{sgn}(\dot{\theta}) + v \dot{\theta}
\]  

(2-23)

Where \( v \) denotes the viscous friction coefficient.

The simulation of equation (2-23) is time consuming. To simplify the simulation in this thesis, the discontinuity of the friction model is dispelled by using the following simplified model:

\[
f(\dot{\theta}) = \begin{cases} 
v_i \dot{\theta}, & |\dot{\theta}| \leq \dot{\theta}_c \\
c \text{sgn}(\dot{\theta}) + v \dot{\theta}, & |\dot{\theta}| > \dot{\theta}_c
\end{cases}
\]  

(2-24)

This method was suggested by Fang \[16\].
In (2-24), the Coulomb friction is approximated by viscous friction with large coefficient $v_1$, in the region where $\dot{\theta}$ is close to zero, as shown in Figure 2.6. The value of $\dot{\theta}$, at which the Coulomb friction switches between the approximation expression and the exact form, is called the critical velocity $\dot{\theta}_c$.

![Friction Model in a Robot Arm Joint](image)

**Figure 2.6** Friction Model in a Robot Arm Joint

In the simulation described in this thesis, the critical velocity, $\dot{\theta}_c$, is chosen to be 0.001 rad/sec for all robot joints. It was asserted by Fang that the behaviours of this simplified model and the actual model are very close by choosing this value for the critical velocity [16].
2.7 Gear Backlash

Backlash often occurs in the mechanical transmission systems consisting of mating gears. It is caused by the small gaps between a pair of mating gears, due to the unavoidable errors in manufacturing and assembly. Figure 2.7 illustrates a typical situation.

![Diagram of gear backlash](image)

**Figure 2.7** A Backlash Non-linearity

The hysteresis caused by gear backlash is a well-understood dynamic non-linearity. Discussions can be found in several books on non-linear control [51,55]. In this section, first the concept of gear backlash is introduced. Then an analysis on the effect of gear backlash in robot arm joint mechanism will be given.

- **Gear backlash phenomenon** [55]

As shown in Figure 2.7, when the driving gear rotates a smaller angle than the gap \( b \), the driven gear does not move at all, which corresponds to the dead-zone (OA segment in Figure 2.7(b)). After contact has been established between the two gears, the driven gear follows the rotation of the driving gear in a linear fashion (AB segment). When the driving gear rotates in the reverse direction by a distance of \( 2b \), the driven gear again does not move, corresponding to the BC gap.
segment in Figure 2.7(b). After contact between the two gears is re-established, the driven gear follows the rotation of the driving gear in the reverse direction (CD segment). Therefore, the gear backlash phenomenon is described clearly by the closed path EBCD when the driving gear is in periodic motion.

- **Gear backlash in the robot arm joint transmission mechanism**

The equations describing backlash hysteresis are given by \((2-25)\) \((22)\). The subscripts \(m\) and \(l\) denote the motor and load shaft quantities. The meanings of \(J_m, J_l, f_m, f_l, \tau_m, \tau_l, \theta_m, \theta_l\) and \(n\) are the same as those in section 2.5. \(b\) is the half of the angular dead-zone distance between gears on the motor shaft side.

\[
\frac{d\hat{\theta}_m}{dt} = J_m^{-1}[-f_m \hat{\theta}_m - \tau_l / n + \tau_m] \quad (1)
\]

\[
r = \begin{cases} 0 & -b < \theta_m - \theta_l \text{ and } i_n < b \\ \left(J_m + J_l / n\right)^{-1}(\hat{\theta}_m(t^-)/n - \hat{\theta}_l(t^-)) & \text{or:} \\ \text{max}(0, \tau_m) & \theta_m - \theta_l \text{ and } i_n = \text{band} \hat{\theta}_m > n \hat{\theta}_l \\ \text{min}(0, \tau_m) & \theta_m - \theta_l \text{ and } i_n = \text{band} \hat{\theta}_m < n \hat{\theta}_l \\ n[\tau_m - (J_m n \hat{\theta}_m + f_m n \hat{\theta}_l)] & \theta_m - \theta_l \text{ and } i_n = \text{band} \hat{\theta}_m = n \hat{\theta}_l = 0 \\ \end{cases} \quad (2-25)
\]

In \((2-25)\), \((1)\) comes from \((2-16)\) and \((2-17)\); \((2)\) indicates the gear torque is zero when the gear is in the dead-zone region; \((3)\) introduces torque impulses at the instant when the gears reach the boundaries of the dead-zone region with sufficient relative velocity to collide. The collision is inelastic. Thus the conservation of angular momentum during the collision \((2-26)\) are used to derive the shaft velocities after the collision in terms of the shaft velocities before the collision \((2-27)\). Taking the limit of \(J \Delta \omega / \Delta t\) as \(\Delta t \to 0\) yields the torque impulse in \((2-28)\).

\[
(J_m + J_l / n^2) \omega_l (t^+) = J_m \omega_m (t^-) / n + J_l \omega_l (t^-) / n^2 \quad (2-26)
\]

That is:

\[
\omega_l (t^+) = (J_m + J_l / n^2)^{-1}(J_m \omega_m (t^-) / n + J_l \omega_l (t^-) / n^2) \quad (2-27)
\]
So:

$$
\tau_i(t) = \lim_{\Delta t \to 0} J_i \frac{\omega_i(t^+) - \omega_i(t^-)}{\Delta t} = J_i J_m (J_m + J_i + J_l) \omega_m(t^+) - \omega_i(t^-) \delta(t)
$$

(2-28)

Equations (4) and (5) in (2-25) give the torque transmitted when the gears are engaged but not moving so that the static friction model is used. In this case, the gear reflects to the load shaft that portion of the motor torque — in the direction that the gear is engaged — not cancelled by motor shaft static friction; (6) gives the transmitted torque when the gears are engaged. Equations (2-14) to (2-18) are only applicable for this situation.

2.8 The Complete System

Figures 2-9 and 2-10 on the following two pages are the block diagram representation of the complete robot system. This includes actuator effects, gear backlash and robot arm dynamic equations with inertia, centrifugal, Coriolis, frictional and gravitational terms. By showing all these components on one block diagram, one can better understand the interaction between the subsystems and the role each term plays.

The simulation model is an open-loop system, corresponding to a robot system without feedback control components. Therefore this model can also be used to investigate the feasibility of certain control strategies by incorporating different controllers.

The current robot positions $\theta$ and velocities $\dot{\theta}$ are fed into the robot dynamic equations along with $\tau_i$ which is computed through a backlash model that takes into consideration the effects of motor states $\theta_m$, $\dot{\theta}_m$, and acceleration $\ddot{\theta}_m$, load shaft states $\theta_l$, $\dot{\theta}_l$, and acceleration $\ddot{\theta}_l$ and the gap between driving and driven gears.
Figure 2.8 is the general diagram of the open-loop robot system that shows the robot connections to the outside world. Every input voltage is connected to each corresponding joint motor, for example, input voltage 1 to joint motor 1, etc. The outputs are joint angular displacements and velocities respectively. Figure 2.9 is the detailed system diagram which shows the relationships and connections among all the subsystems and components within the robot. Figure 2.10 is the actuator’s Simulink model which comes from the analysis in section 2.5, in which we choose: $R_s = 5\Omega$, $L_s = 0.05H$, $K_m = 0.05 \text{ N-m/A}$, $K_b = 0.05 \text{ V-S/rad}$.
Figure 2.10  The Simulink model of the actuator
2.9 Simulated Open-loop Characteristics of the PUMA Robot

In this section the dynamic response characteristics of the simulated PUMA robot will be shown by inputting different signals. The inputs are the voltages to the joint motors whereas the outputs are the joint angular displacements and velocities respectively. To reflect the dynamic response characteristics more clearly special inputs are chosen as those in Table 2.2. Figures 2.11 to 2.14 are the robot response (angular displacement) curves to each of these input groups respectively. The initial positions for all joints in all cases are zero.

Table 2.2

<table>
<thead>
<tr>
<th>Case</th>
<th>Joint 1 Voltage (Volts)</th>
<th>Joint 2 Voltage (Volts)</th>
<th>Joint 3 Voltage (Volts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>step1*</td>
<td>step2**</td>
<td>step3***</td>
</tr>
<tr>
<td>2</td>
<td>20 sin2 ( \pi ) t</td>
<td>10 sin4 ( \pi ) t</td>
<td>5 sin6 ( \pi ) t</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>10 sin 4 ( \pi ) t</td>
<td>0</td>
</tr>
</tbody>
</table>

* Step1: Step signal. Initial value is 0, final value is 5. Step time is 0.5 second.
** Step2: Step signal. Initial value is 0, final value is 3. Step time is 0.5 second.
*** Step3: Step signal. Initial value is 0, final value is 1. Step time is 0.5 second.

Figure 2.11 depicts the system’s step response. The non-linearity of the system is shown in Figure 2.12, where the responses to sinusoidal inputs in all joints have been transformed completely. Figure 2.13 depicts the zero input response. Because the zero points are not the equilibrium positions of the robot, the links of the robot move under the action of gravitation force of each link, tending to some equilibrium points. The coupling effect of the system is shown in Figure 2.14, where a sinusoidal input to joint 2 affects the movements in both joint 1 and 3 as well.
Figure 2.11 (a) — joint 1

Figure 2.11 (b) — joint 2

Figure 2.11 (c) — joint 3

Figure 2.11 Case 1: The step response of the simulated robot system
Figure 2.12 (a) — joint 1

Figure 2.12 (b) — joint 2

Figure 2.12 (c) — joint 3

Figure 2.12 Case 2: The sinusoidal response of the simulated robot system
Figure 2.13 (a) — joint 1

Figure 2.13 (b) — joint 2

Figure 2.13 (c) — joint 3

Figure 2.13 Case3: The zero input response of the simulated robot system
Figure 2.14 The sinusoidal response of the simulated robot system
2.10 Summary

In this chapter a simulated robot system based on the PUMA robot dynamic equations has been built. This simulated robot system has incorporated the actuator effects, gear backlash and the robot dynamic equations with inertia, centrifugal and Coriolis, frictional and gravitational forces into the model. Upon set up, part or whole of this robot model will be treated as the ‘real’ robot, or the ‘real’ robot components. Research of the proposed modelling methods and control strategy will be carried out using this model.

From the equation (2-1) and above analysis it is found that this simulated robot system is one with strong non-linearity and coupling effects. According to the control theory, the conventional control strategies, such as independent joint control based on a PD or PID method, do not work very well when all robot arms are moving simultaneously and at high speeds. Thus new control strategies should be investigated to solve the problem. In the next chapter the Model Predictive Control strategy, which is mainly used in chemical engineering previously, is proposed for robot trajectory control. The validity of using this control strategy in robot control will be investigated.
Chapter 3

Model Predictive Control of Robots

3.1 Introduction

Since its introduction, MPC has become popular with chemical process plant operators and researchers due to its unique ability to systematically deal with constraints. All of MPC methods, such as DMC, MAC and GPC mentioned in Chapter 1, have certain common characteristics, most notably the on-line determination of a sequence of optimal control actions to minimise the deviation of the system outputs from its setpoints, subject to constraints. Since only the first of these computed control actions is implemented, the entire set is redetermined at each sample time, MPC functions over a “receding” or “moving” horizon.

As mentioned in Chapter 1, while MPC algorithms using linear predictive models have significant advantages in implementation as well as theoretical analysis, for a non-linear system, algorithms using non-linear predictive models are more advantageous, especially for an industrial robot. A number of MPC algorithms using non-linear models have been proposed [5,21,31]. Instead of developing a non-linear first principle model, in engineering, one seeks to establish a dynamic model directly from the input/output data sampled from the plant. This is because in many cases it is very difficult to obtain a suitable physically founded system model in an analytical way, due to the complexity of the practical systems, or lack of knowledge of critical parameters of the systems.

This chapter investigates the feasibility of the model predictive control for industrial robots based on a non-linear predictive model. First the most representative MPC algorithms will be reviewed. Then an MPC structure for robot position control is
proposed. The MPC controller is designed to control the PUMA simulation model discussed in chapter 2. In the last part of the chapter, the validity of the method will be investigated under different conditions.

3.2 The Concept of the MPC

The methodology of all the controllers belonging to the MPC family is characterised by the following strategies represented in Figure 3.1 [5].

![Figure 3.1](image)

**Figure 3.1** The MPC Strategy
Based on Figure 3.1 the basic procedure of the MPC can be described as follows:

1. The future outputs of a system for a certain horizon $N$, called the prediction horizon, are predicted at each instant $t$ using the system model. These predicted outputs $y(t+k | t)$ for $k=1, \ldots, N$ are calculated using the past inputs and outputs, and the future control signals $u(t+k | t)$, $k=0, \ldots, N-1$, which are to be determined.

2. The set of future control signals are calculated by optimising a predefined criterion in order to keep the system output as close as possible to the reference trajectory $w(t+k)$. This criterion usually takes the form of a quadratic function of the errors between the predicted output signal and the reference trajectory.

3. The control signal $u(t | t)$ is sent to the system while the next calculated control signals are rejected, because at the next sampling instant $y(t+1)$ is already known and step 1 is repeated with this new value and all the sequences are brought up to date. Thus the $u(t+1 | t+1)$ is calculated (which in principle will be different to the $u(t-1 | t)$, because of the new information available) using the receding horizon concept.

All the MPC algorithms using a linear model are very similar. Here we demonstrate only how DMC works.

The manipulated variables are selected to minimise a quadratic objective function:

$$
\min_{u(t+1 \ldots t+N)} \sum_{i=1}^{N} \| \hat{y}(k+l | k) - r(k+l) \|^2 + \| \Delta u(k+l-1) \|^2 \quad (3-1)
$$

$$
\hat{y}(k+l | k) = \sum_{i=1}^{l} H_i \Delta u(k+l-i) + \sum_{i=t+1}^{l} H_i \Delta u(k+i-l-i) + H_u u(k+l-n) + \hat{d}(k+l | k) \quad (3-2)
$$

$$
\hat{d}(k+l | k) = \hat{d}(k | k) = y_{\text{ref}}(k) - \sum_{i=t}^{l-1} H_{\text{int}} \Delta u(k-i) + H_{\text{int}} u(k+l-n) \quad (3-3)
$$
where:

\[ \hat{y}(k+1|k) = \text{predicted value of the system output at time } k+1 \text{ based on information available at time } k; \]

\[ r(k+1) = \text{desired system output at time } k+1; \]

\[ \hat{d}(k+1|k) = \text{predicted value of additive disturbances at process output at time } k+1 \text{ based on information available at time } k; \]

\[ u(k) = \text{the control signal at time } k; \]

\[ y_m(k) = \text{measurement of the system output at time } k; \]

\[ \Delta u(k+1) = u(k+1) - u(k+1-1); \]

\[ H_i (i=1, \ldots, n) = \text{model step response matrix coefficients}; \]

\[ n = \text{truncation order}; \]

\[ p = \text{horizon length}; \text{ and,} \]

\[ m = \text{number of manipulated variable moves in the future (} \Delta u(k+1) = 0, \forall i \geq m, m < p); \]

The prediction of output, (3-2), involves three terms on the right-hand side. The first term includes the present and all future moves of the manipulated variables which are to be determined so as to solve (3-1). The second term only includes past values of the manipulated variables and is completely known at time \( k \). The third term is the predicted disturbance. At time \( k \) it is estimated as the difference between the measured output \( y(k) \) and the output predicted from the model. Equations (3-1) — (3-3) define a non-linear optimisation program which is to be solved on-line at every time step.

3.3 A Model Predictive Controller for Robot Position Control

Based on the basic procedure of the MPC discussed in the previous section, we can have one model predictive control structure, as shown in Figure 3.2 for robot position control [21]. The implementation procedures are [18]:

[Note: The text continues with further details on the model predictive control for robot position and more implementation procedures.]

[End of Document]
1. Obtain a model of the robot to be controlled.
2. Use the model to predict the behaviour of the robot over a certain time horizon.
3. Determine the control action by optimising a performance index, which typically is the error between the output predicted from the model and the desired output, over the time horizon.
4. Apply the optimal control actions and measure the robot outputs over the time horizon. Use the last measurements of the robot states in the time horizon as the initial states of the model to perform the next iteration.
5. Repeat 2 to 4 until the end of the trajectory is reached.

![Block diagram of model predictive controller](image)

These steps can be expressed by the following equations:

\[
\min_{\tau(k)} \left( \sum_{l=1}^{N} [x_d(k+l) - x(k+l)]^2 \right) \tag{3-4}
\]

subject to

\[
\tau_{\text{min}} < \tau(k) < \tau_{\text{max}} \quad \forall k \tag{3-5}
\]
where \( k \) is the time step, \( \tau(k) \) is the control vector at time \( k \); \( x_d(k) \) and \( x(k) \) are the desired output and predicted output vectors. They may include positions and velocities of the robot to be controlled, at time \( k \) respectively, \( p \) is the prediction horizon. This control scheme clearly involves solving an optimisation problem. To perform optimisation and to determine the control actions, many optimisation methods can be used.

### 3.4 The Selection of the Optimisation Method

As mentioned above the model predictive control actions are determined by optimisation. The optimisation problem generally arise whenever there is a need to minimise (or maximise) an objective function that depends on a set of variables while satisfying some constraints. In terms of the properties of the objective functions and constraints, the optimisation problem can be divided into two main categories: linear and non-linear programming problems. In linear programming problems, the objective functions and all constraints are linear. Whereas in non-linear programming problems, the objective function and/or constraints are non-linear. The constraints may be equalities, inequalities or two sided form as \( l \leq c_i(x) \leq u_i \), where \( l \), and \( u \), are lower and upper bounds [8]. Obviously the MPC control problem (3-4) and (3-5) is a non-linearly bounded optimisation problem.

There are many ready-made software packages for non-linear constrained optimisation [9], among which NLPQL is chosen for the MPC control problem because the author is well acquainted with the related subroutine. However other packages are worth trying in the future to find out the most efficient optimisation method.

NLPQL is written in Fortran programming language and organised in the form of a subroutine. The internal algorithm is a sequential quadratic programming method.
Proceeding from a quadratic approximation of the Lagrangian function and a linearisation of the constraints, a quadratic subproblem is formulated and solved to get a search direction. Subsequently a line search is performed with respect to two alternative merit functions. The Hessian approximation is updated by the modified quasi-Newton formula. To use this subroutine, the user must provide the objective functions of the non-linear problem and analytical gradients in certain special subroutines \cite{98}. In Fortran PowerStation 4.0, the NLPQL algorithm is included in its internal subroutine library.

### 3.5 Validity of MPC Controller for Robot Control

This section investigates the validity of the MPC controllers described in the previous sections of this chapter when they are applied to the robot system. The investigation will be performed on the simulated system built in Chapter 2. Because the analyses of the convergence and divergence of the model predictive control method for non-linear systems in an analytical way, is very difficult and has not been solved \cite{51}, the experimental and simulation method will be used.

In this section only the dynamic equations of the simulated robot system (2-9) are considered. MPC control of the entire simulated system will be performed in Chapter 5. In the following parts of this section the dynamic equations of the robot are referred to as the "robot system".

All the desired trajectories used for simulations start from the origin point of the joint co-ordinate system. The range of the MPC controller is set to be ± 20 N-m, which is assumed to be the maximum torque the joint motor can supply. All the initial guesses to the control torque are 0 N-m. The control time span is one second. The control time step is 5 milliseconds.
• MPC controller based on a perfect prediction model

This part investigates the properties of the MPC controller when a perfect robot
dynamic prediction model exists. A perfect model has the same structure and parameters
as those in the simulated robot system, that means when the same torques are inputted,
the prediction model has the same outputs (displacements and velocities in all joints) as
those from the simulating system.

The desired trajectories for case 3.1 are shown in Figure 3.3. These trajectories
were obtained from exciting the robot dynamics model by the sinusoidal torques shown
in column 1 of Table 3.1. The tracking errors of the MPC control of the system are shown
in Figure 3.4. Through several rounds of calculation it is found that the different guesses
to the initial torques do not affect the results. It is also found that the effect of increasing
prediction steps when deciding control actions through optimisation is not obvious to
decrease the tracking errors in this circumstance. Because the desired trajectories are
obtained from exciting the robot using torque signals with amplitude well below their
limits, therefore, the small deviation of the tracking errors due to the prediction step
variation in each calculation can be easily accommodated by varying the torques at the
next time step.

The desired trajectories for case 3.2 are shown in Figure 3.5. These trajectories
were obtained from exciting the robot dynamics model by the sinusoidal torques shown
in column 2 of Table 3.1. The tracking errors of the MPC control of the system are shown
in Figure 3.6. Due to the same reasons stated previously the increase of the prediction
horizon does not have obvious effect on the control performance.
Table 3.1 Exciting torques at different joints to produce the desired trajectories for Cases 3.1, 3.2 and 3.3

<table>
<thead>
<tr>
<th></th>
<th>CASE 3.1</th>
<th>CASE 3.2</th>
<th>CASE 3.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint 1</td>
<td>sin(4πt)</td>
<td>9sin(10πt)</td>
<td>17sin(πt)</td>
</tr>
<tr>
<td>Joint 2</td>
<td>sin(5πt)</td>
<td>8sin(12πt)</td>
<td>20sin(1.5πt)</td>
</tr>
<tr>
<td>Joint 3</td>
<td>sin(6πt)</td>
<td>7sin(14πt)</td>
<td>7sin(2πt)</td>
</tr>
</tbody>
</table>

The desired trajectories for case 3.3 are shown in Figure 3.7. These trajectories were obtained from exciting the robot dynamics model by the sinusoidal torques shown in column 3 of Table 3.1. Case 3.3 is designed to investigate the MPC controller’s characteristics when tracking the trajectories resulted from the sinusoidal excitations with the amplitudes which are close to the limit of the MPC controller. The results of the MPC control of the system are shown in Figure 3.8. From this figure it could be found that, when one step ahead prediction is used, the errors between the actual trajectories and the desired trajectories are larger than those in both cases 3.1 and 3.2 in all three joints. This is because the control actions derived from the MPC controller may exceed the limit range for the robot system to best track the desired trajectories. When the control actions are coercively restricted within the limit range of the MPC controller, greater errors are caused. The tracking performance could be improved by increasing the prediction horizon. In this case, 5 and 10 step predictions are used. By extending the prediction horizon, more information is taken into consideration when performing the optimisation and thus the saturation of the control actions can be minimised. This fact is reflected in Figure 3.9, in which control torques, derived from one-step, five-step and ten-step MPC control separately, are compared.
Figure 3.3(a) — joint 1 position

Figure 3.3(b) — joint 2 position

Figure 3.3(c) — joint 3 position

Figure 3.3 Desired Trajectories — Model Predictive Control Based on a Perfect Model for Case 3.1
Figure 3.4(a)—joint 1

Figure 3.4(b)—joint 2

Figure 3.4(c)—joint 3

Figure 3.4 Tracking Errors  Model Predictive Control Based on a Perfect Model for Case 3.1
Figure 3.5(a)—joint 1

Figure 3.5(b)—joint 2

Figure 3.5(c)—joint 3

Figure 3.5  Desired Trajectories — Model Predictive Control Based on a Perfect Model for Case 3.2
Figure 3.6(a)—joint 1

Figure 3.6(b)—joint 2

Figure 3.6(c)—joint 3

Figure 3.6 Tracking Errors — Model Predictive Control Based on a Perfect Model for Case 3.2
Figure 3.7(a)—Joint 1 position

Figure 3.7(b)—Joint 2 position

Figure 3.7(c)—Joint 3 position

Figure 3.7 Desired Trajectories — Model Predictive Control Based on a Perfect Model for Case 3.3
Figure 3.8(a)—Joint 1

Figure 3.8(b)—Joint 2

Figure 3.8(c)—Joint 3

Figure 3.8  Tracking Errors — Model Predictive Control Based on a Perfect Model for Case 3.3
Figure 3.9 (a) — Joint 1

Figure 3.9 (b) — Joint 2

Figure 3.9 (c) — Joint 3

Figure 3.9 Control torques — Model Predictive Control Based on a Perfect Model for Case 3.3
• **MPC controller based on the measurements of the actual displacements and velocities with some errors**

In this part, the perfect prediction model is used, but some errors exist in the measurement of the actual displacements and velocities. This means that the prediction will be performed from a starting point that deviates from the actual trajectory point.

In the following cases, suppose there are some random errors in the measurement of the displacements and velocities. The maximum possible error values are ±2% of the displacements and ±5% of the velocities. The random numbers are produced by a Fortran 90 subroutine named RANDOM_NUMBER, which can produce uniformly distributed pseudo-random numbers in the range of (0,1) under the control of the system clock. As long as the system clock keeps ticking, different numbers can be produced at every time the program is run.

In case 3.4, the desired trajectories are the same as those in case 3.1 (Figure 3.3). The results of MPC control with measurement errors are shown in Figure 3.10. It is found from this figure that the tracking errors are greater than those for case 3.1 in all three joints, due to the effect of the measurement errors. It is also found that the effect of increasing prediction steps in MPC is not obvious to decrease the tracking errors due to the same reasons stated in case 3.1. However, because the prediction in an MPC controller is started from the actual points (displacements and velocities) in which there are some measurement errors, the tracking errors in this case are much greater than those in case 3.1.

In case 3.5, the desired trajectories come from the case 3.2 (Figure 3.5). The results of the MPC control under this circumstance are shown in Figure 3.11. Due to the same reasons stated for case 3.4 the effect of increasing prediction horizon is not obvious to decrease the tracking errors.
Figure 3.10 (a) — joint 1

Figure 3.10 (b) — joint 2

Figure 3.10 (c) — joint 3

Figure 3.10  Tracking errors
Model Predictive Control Based on a Perfect Model with Measurement Errors
Case 3.4
Figure 3.11 (a) — joint 1

Figure 3.11 (b) — joint 2

Figure 3.11 (c) — joint 3

Figure 3.11 Tracking errors
Model Predictive Control Based on a Perfect Model with Measurement Errors
Case 3.5
In case 3.6, the desired trajectories are the same as those in case 3.3 (Figure 3.7). The results of the MPC control are shown in Figure 3.12. When one step ahead prediction is used, the errors between the actual trajectories and the desired trajectories are larger than those in both cases 3.10 and 3.11 in all three joints. This is because under this circumstance, the tracking accuracy is determined not only by the magnitude of the measurement errors, but also by the saturation of the control torques. The tracking performance could be improved by increasing the prediction horizon. In this case, 5 and 10 step predictions are used. By extending the prediction horizon, more information is included in the optimisation process and thus the effect of the limit condition of the MPC controller to the control accuracy is reduced.

- An MPC controller based on a predictive model with structured and unstructured uncertainties

This part investigates the properties of the MPC controller for robot position control based on a predictive model with some structured and unstructured uncertainties. Structured uncertainties refer to the case when the proper model structure is determined, but there are uncertainties in the values of its parameters. However whenever part of the dynamic model is not present, it is usually referred to as unstructured uncertainties. In this part, suppose in the prediction model of the MPC controller there is a 10% of uncertainty in the parameters of the inertia term, and the friction terms are neglected.

In case 3.7, the desired trajectories are the same as those in case 3.1 (Figure 3.3). The results of the MPC control of the robot system based on a prediction model with uncertainties are shown in Figure 3.13. In this case the performance of the MPC controller could be improved by increasing the prediction horizon, even though the desired trajectories are derived from the sinusoidal exciting signals with the amplitudes which are much smaller than the limits of the MPC controller. This is because, by
Figure 3.12 (a) — joint 1

Figure 3.12(b) — joint 2

Figure 3.12(c) — joint 3

Figure 3.12 Tracking errors
Model Predictive Control Based on a Perfect Model with Measurement Errors
Case 3.6
extending the prediction horizon, more information is taken into consideration when performing the optimisation and thus the effect of the model uncertainties to the MPC controller is "smoothed" or "averaged" over the prediction horizon.

In case 3.8, the desired trajectories are the same as those in case 3.2 (Figure 3.5). The results of the MPC control of the robot system based on a prediction model with uncertainties are shown in Figure 3.14. Due to the same reason stated for case 3.7, in this case the performance of the MPC controller could also be improved by increasing the prediction horizon.

In case 3.9, the desired trajectories are the same as those in case 3.3 (Figure 3.7). When one step ahead prediction is used, the errors between the actual trajectories and the desired trajectories are larger than those in both cases 3.12 and 3.11 in all three joints. This is because under this circumstance, the tracking accuracy is determined by not only the model uncertainties, but also the limit conditions of the MPC controller. According to the previous discussions, the effect of the model uncertainties and the limit conditions of the MPC to the control accuracy could be reduced by extending the prediction horizon. In this case 5 and 10 step prediction are used. A comparison of the MPC control of the robot system based on a prediction model with uncertainties at different prediction steps is shown in Figure 3.15.

- **MPC controller when the frequency band of the controller is limited**

In the previous discussions there is no limitation to the frequency of the MPC controller. That is, the changing rate of the control torques could be unlimited to track the desired trajectories precisely. However, if the control frequency is too high the flexibility of the robot arm may be excited, and extensive vibration may then be caused. In this part some limitations will be put to the change rate of the control torque. Under this circumstance, the validity of the MPC controller for robot trajectory tracking will be investigated.
Figure 3.13(a) — joint 1

Figure 3.13(b) — joint 2

Figure 3.13(c) — joint 3

Figure 3.13 Tracking Errors —
Model Predictive Control Based on a Model with Uncertainties
Case 3.7
Figure 3.14(a) — joint 1

Figure 3.14(b) — joint 2

Figure 3.14(c) — joint 3

Figure 3.14 Tracking Errors — Model Predictive Control Based on a Model with uncertainties Case 3.8
Figure 3.15(a) — joint 1

Figure 3.15(b) — joint 2

Figure 3.15(c) — joint 3

Figure 3.15 Tracking Errors — Model Predictive Control Based on a Predict Model with uncertainties
Case 3.9
The limitations on the change rate of the control torques could be added to the optimisation objective function (3-4) as following:

$$\min \sum_{l=1}^{p} \left[ x_d(k+l) - x(k+l) \right]^2 + \sum_{l=1}^{p} [w \left[ \tau(k+l-1) - \tau(k+l) \right]^2]$$  \hspace{1cm} (3-6)$$

where the meanings of $k, \tau(k), x_d(k), x(k)$ and $p$ are all the same as those in (3-4); $w$ is the weighting factor of the limitation on the changing of the control torques to the objective function. The greater the $w$, the less variation the torque will have.

In case 3.10, the desired trajectories are the same as those in case 3.3 (Figure 3.7). Figure 3.16 depicts the control torques in three joints where there is no frequency limitation added to the MPC controller, or there are some limitations added to the controller with the weight mentioned in (3-6) equal to 0.001 and 0.01 respectively. Figure 3.17 shows the tracking errors in robot joints when weighting factor $w$ in equation (3-6) is 0.001 and 0.01 respectively. It is found that high order components of the control torques can be eliminated effectively by the introduction of the change rate limit condition of the control torques. The larger the weight, the smoother the control torques are. On the other hand, larger weight leads to larger tracking errors.

To investigate the ability of an MPC controller with control frequency limitations to restrain abundant higher order harmonic components in case 3.11 the desired robot trajectories are obtained using the following ramp functions (Figure 3.18):

In joint 1. \hspace{1cm} \tau_1(t) = \begin{cases} 40t & t < 0.5 \text{sec} \\ 20 & 0.5 \text{sec} \leq t < 1 \text{sec} \end{cases}

In joint 2. \hspace{1cm} \tau_2(t) = \begin{cases} 20t & t < 0.5 \text{sec} \\ 10 & 0.5 \text{sec} \leq t < 1 \text{sec} \end{cases}
Figure 3.16(a)—Joint 1

Figure 3.16(b)—Joint 2

Figure 3.16(c)—Joint 3

Figure 3.16  Control Torques—Case 3.10
Figure 3.17(a) — joint 1

Figure 3.17(b) — joint 2

Figure 3.17(c) — joint 3

Figure 3.17 Tracking errors—Case 3.10
In joint 3, \( r_3(t) = \begin{cases} 
10t & t < 0.5 \text{sec} \\
5 & 0.5 \text{sec} \leq t < 1 \text{sec} 
\end{cases} \)

When there is no torque limitation exerted to the MPC controller, the control torques in all three joints have high frequency components (Figure 3.19). All these fast changing components of the control torques could be reduced effectively by introducing the limiting condition to the control torques as that in case 3.10 (the weights for control torque limitations in case 3.11 are 0.001 and 0.01 respectively as well). Figure 3.20 are the comparison of the tracking errors to the control torques with the weighting factor in each instance being 0.001 and 0.01.

Figure 3.21 shows the robot dynamic system's response to a white noise excitation. The response is used as the desired trajectories in case 3.12. The white noise exciting signal comes from the SIMULINK signal-generating block. The powers of three noise excitations to three different joints are 10, 5 and 2.5 N-m respectively. Figure 3.22 depicts the control torques when the control frequency limitations with weights of 0.001 or 0.01 are exerted to the MPC controller. Figure 3.23 is the comparison of tracking errors when the controller has no frequency limitation, and has the limitation mentioned above.

### 3.6 Summary

In this chapter a method for robot position control, namely the model predictive control (MPC), is proposed. The feasibility of the proposed control method has also been investigated in this chapter. Through simulation it is found that the error sources of the MPC controller come mainly from the limit conditions of the MPC controller, measurement errors and model uncertainties. The accuracy of the MPC controller based on a perfect prediction model is of the order of \(10^{-3}\). The effect of the limit condition of the MPC controller and model uncertainties to the control accuracy could be improved by extending the prediction horizon. The high frequency components of the control torques
Figure 3.18(a) — joint 1

Figure 3.18(b) — joint 2

Figure 3.18(c) — joint 3

Figure 3.18 Desired Trajectories
Model Predictive Control for Case 3.11
Figure 3.19(a) — Joint 1

Figure 3.19(b) — Joint 2

Figure 3.19(c) — Joint 3

Figure 3.19  Control Torques—Case 3.11
Figure 3.20(a) — joint 1

Figure 3.20(b) — joint 2

Figure 3.20(c) — joint 3

Figure 3.20 Tracking Errors — Case 3.11
Figure 3.21 Desired Trajectories
Model Predictive Control for Case 3.12
Figure 3.22(a) — joint 1

Figure 3.22(b) — joint 2

Figure 3.22(c) — joint 3

Figure 3.22        Control Torque—Case 3.12
can be eliminated effectively by introducing the limiting condition to the control torques. However there is a trade-off between the frequency limiting weight of the control torque and tracking errors. That is, the larger the weight, the smoother the control torques are, but larger weight leads to larger tracking errors. Through the proper selection of the weight, higher accuracy of the trajectories can be obtained, whereas the smoothness of the control torque is maintained at a reasonable level.
Chapter 4
Neural Network Predictor for Robot

4.1 Introduction

The importance of the prediction model in an MPC controller has been demonstrated in Chapter 3. However in a real application accurate modelling using the exact analytical methods is impractical for robot trajectory tracking through MPC. Due to the complexity of most of systems to be modelled, it is difficult to establish a perfect prediction model using an exact mathematically analytical method as we did in the last chapter. When there are some parameter uncertainties in the prediction model, or the structure of the prediction model is different from that of the real system, greater control errors or even system instability may be caused.

Since the late 1980's, artificial neural networks (ANNs) have found wide applications in the engineering field, because of the development in ANN’s learning algorithms and computer technology. Most engineering researchers are interested in the following two properties of ANNs. The first is the ANNs’ learning and universal approximation ability [39]; that is, ANNs could be used to approximate any non-linear mapping relationship between the inputs and outputs. The second is ANN’s parallel processing abilities [49]. Based on above two properties engineering researchers have successfully applied ANNs to many engineering areas, such as pattern recognition [4], non-linear system identification [6,7], and control [14,60].

In this chapter a method of using ANNs to model the robot forward dynamics in a predictive way for the implementation of the MPC strategy is proposed. By applying the above mentioned ANN’s learning and approximation property one can use ANNs as the
non-linear black-box models of robot systems. The accuracy of the ANN model could be guaranteed through various structures and learning algorithms.

It is known that to model dynamic system processes for control purpose, two main formulations, *i.e.*, input/output model and state space model, can be used. In this thesis we choose input/output model for identification, because a discrete-time input/output model defines the outputs of the system only as a function of values of the outputs and inputs at previous and current time-steps. Obviously the input/output model takes a predictive form in straightforward way that is required by the MPC controller. The NN prediction model can then be trained by directly using the data collected from operating the real system to be controlled.

### 4.2 An Overview of Artificial Neural Networks

An artificial neural network is actually a network of interconnected elements. The models of these elements were inspired from the studies of the biological nervous systems so these elements are called neurons. Figure 4.1 depicts the schematic structure of a neuron. It is shown that a neuron receives *n* inputs *x*\(_i\), *i* = 1, 2, ..., *n*, which can be

![The structure of a neuron](image)
either from the external source or the other neurons. Each input is multiplied by a scalar value \( w_i \), known as the weight, and summed together with a bias \( x_0 \) to form the intermediate value \( y \). Finally, a non-linear transfer function \( f(.) \) is applied to calculate the output of the neuron. The equation can be expressed as:

\[
u = f\left( \sum w_i x_i + x_0 \right)
\]  

To form a network structure, the inputs and outputs of the neurons are connected in some way. According to the connection method, as mentioned in Chapter 1, neural networks could be divided into two major categories: feedforward neural networks (FNNs) and recurrent neural networks (RNNs).

- **Feedforward Neural networks**

A general fully connected feedforward multilayered neural network can be described in Figure 4.2, where the information propagates only in one direction (as indicated by the arrows). Each processing unit (denoted by a circle) is a neuron, and the interconnections between neurons are called synapses. The neuron first calculates the weighted sum of all synaptic signals from the previous layer plus a bias term, and then generates an output through its activation function. The input layer consists of only “fan-out” units. Each fan-out unit simply distributes an input to all neurons of the first hidden layer.

Mathematically, a feedforward multilayer neural network can be represented by the following equations:

\[
x^l = W^l u^{l-1} + \omega^l
\]  

\[
u^l = f^l(x^l)
\]  

(4-2)  

(4-3)
Where

$$W^l = \begin{bmatrix} w_{1,1}^{l} & w_{1,2}^{l} & \cdots & w_{1,N_l}^{l} \\ w_{2,1}^{l} & w_{2,2}^{l} & \cdots & w_{2,N_l}^{l} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N_l,1}^{l} & w_{N_l,2}^{l} & \cdots & w_{N_l,N_l}^{l} \end{bmatrix}$$

(4-4)

is the synaptic matrix in layer $l$: $w_{i,j}^{l}$ is the weight connecting the neurons $i$ in layer $l-1$, and the neuron $j$ in layer $l$.

$$F^l(x) = (f_1(x_1), f_2(x_2), \ldots, f_N(x_{N_l}))^T$$

(4-5)

is the activation vector in layer $l$: $f_i (i = 1, 2, \ldots, N)$ in $F^l$ denotes the activation function of the $i$th neuron; $u^l$ is the input to layer $l$; $\omega^l$ is the bias vector of the layer $l$. In practice, bias can be interpreted as the weight associated with a constant unity neuron at the previous layer (denoted by a small rectangle in Figure 4.2).

---

**Figure 4.2** Structure of a multilayer feedforward network
As shown in Figure 4.2, a multilayer feedforward neural network consists of several layers with different functions. The input layer is the one that receives inputs and typically performs no function other than buffering of the input signal. The outputs of the network are generated from the output layer. Any layer between the input and output layers is called a hidden layer because it is internal to the network and has no direct contact with the external environment. The neurons in a hidden layer and the output layer perform some activation functions. The most commonly-used activation functions can be found in Table 4.1. Such a feedforward neural network has been proven to have universal function approximation properties with only one hidden layer of sigmoid neurons [10].

Table 4.1 The most commonly-used activation functions

<table>
<thead>
<tr>
<th>Type of functions</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>$f(x) = x$</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>$f(x) = 1/(1 + \exp(-x))$</td>
</tr>
<tr>
<td>Hyperbolic tangent</td>
<td>$f(x) = (1 - \exp(-2x))/(1 + \exp(2x))$</td>
</tr>
</tbody>
</table>

- **Recurrent Neural Networks**

A recurrent network consists of neurons and synapses similar to those of a feedforward network, except that, as depicted in Figure 4-3, the information runs not only in a forward direction but also in a backward direction along the synapses. For example, in a fully connected recurrent neural network, every neuron is connected to every other one, and also to itself. Normally the output of the neuron is considered as having a one-step time delay. The discrete time equation of the output of each neuron is:

$$u_i(k + 1) = f\left(\sum_{j=1}^{N} w_{ij} u_j(k)\right)$$  \hspace{1cm} (4-6)

where $u_i(k)$ is expressed as:
\[
    u_i(k) = \begin{cases} 
        1 & i = 0 \\
        u_j(k) & i = 1, 2, \ldots, N \\
        v_{i,0}(k) & i = N + 1, N + 2, \ldots, N + M 
    \end{cases}
\] (4-7)

where \( u_i(k) \) is the state of the \( i \)th neuron at the time instant \( k \), \( w_j \) is the weight which connects neuron \( j \) to neuron \( i \), \( v_i \) is the external input to the \( i \)th neuron, and \( f(\cdot) \) is the activation function used to obtain the output. \( N \) and \( M \) are the number of neurons in the network and the number of external inputs to the network, respectively.

Figure 4-3 Structure of a recurrent neural network

4.3 Training of a Neural Network

It can be seen from section 4.2 that the output of the neuron, consequently the network, is determined by the values of the weights and biases of the network. To obtain the correct values of these parameters, a neural network has to go through a procedure, named learning (or training). There are three main categories of learning methods: supervised, reinforcement and unsupervised learning. In this thesis only the supervised learning method is used.

To perform the supervised learning, samples, also called teacher, i.e., the correct relationship of the inputs and the outputs, must be provided. During the network training
process, some learning algorithm should be used to adjust the weights and biases of the network to minimise the errors between the outputs in the samples and the overall network outputs. The basic training algorithm is called the back-propagation [22, 48]. In the back-propagation, the problem of the network training has been expressed as an optimisation problem so as to minimise the error $E$ defined as follows:

$$E = \frac{1}{2} \sum_{k=1}^{K_\gamma} [\hat{y}(k) - y(k)]' [\hat{y}(k) - y(k)]$$

(4-8)

where $y(k)$ is the desired output shown in the sample, $\hat{y}(k)$ is the overall network output, $K_\gamma$ is the total number of samples. The weight update is performed by taking the gradient of the total mean squared errors $E$ with respect to the weights and performing a gradient search in the weight space. This learning algorithm, normally called back-propagation algorithm or generalised delta rule, is applied to multilayer feedforward networks consisting of processing elements with continuous differentiable activation functions. When a sigmoid function is used as the activation function, the weight update equations can be expressed as follows:

$$w_{ij}(n+1) = \eta y_{jk}(n) + \alpha e_{kj} u_{ij} (1 - u_{ij}) u_{j(i+1)j}$$

(4-9)

where

$$e_{kj} = \sum_{a=1}^{N_j} u_{i(a+1)a} (1 - u_{i(a+1)a}) c_{i(a+1)a} w_{i(a+1)a}$$

(4-10)

$e_{kj}$ is the error term of the $k$th neuron in $i$th layer, $u_{kj}$ is the output of the $k$th neuron in the $i$th layer, $w_{kj}$ is the weight connecting the $j$th neuron in layer $(i-1)$ to the $k$th neuron in layer $i$. $\alpha$ is the training rate, $\eta$ is the momentum, and $N_j$ is the number of neurons in the $i$th layer.

Several learning algorithms for training RNNs have also been proposed in the literature [22, 23]. One of them is the back-propagation through time (BPTT) algorithm [24]. This algorithm converts the RNNs from feedback system into a purely feedforward system by unfolding the network over time. There are also some other algorithms for training RNNs, such as real-time recurrent learning [58], the modified BPTT [60].
4.4 Problem Formulation and Assumptions

This section discusses the viable ways for modelling robot dynamics using NNs in a predictive mode.

From the viewpoint of the system identification a parameterisation of the non-linear mapping \( f(.) \) by NN can be denoted as:

\[
y_k = f(x_{k-1}; \varphi) + e_k \tag{4-11}
\]

where

\[
x_{k-1} = [y_{k-1}, \ldots, y_{k-K}, u_{k-1}, \ldots, u_{k-K}]
\tag{4-12a}
\]

in the case of a Non-linear Autoregressive with eXogenous input (NARX) model and

\[
x_{k-1} = [y_{k-1}, \ldots, y_{k-K}, u_{k-1}, \ldots, u_{k-K}, e_{k-1}, \ldots, e_{k-K}]
\tag{4-12b}
\]

in the case of a Non-linear Autoregressive Moving Average with eXogenous input (NARMAX) model. The parameter vector \( \varphi \) in (4-11) contains the elements of the interconnection matrices. \( k_y, k_u \) and \( k_e \) are the number of the time delay of outputs, inputs and errors respectively.

If strong observability conditions are known (or assumed) to be satisfied in the system’s region of operation, two modes of identification and prediction using NN are possible:

- The serial mode

In this mode, the identification procedure using an FNN is straightforward. At each instant of time, the inputs consisting of the system’s past \( k_u \) input values and past \( k_y \) output values are fed into the NN. The network’s output \( \hat{y} \) is compared with the next observation of the system’s output to yield the error:

\[
e(k) = y(k) - \hat{y}(k) \tag{4-13}
\]

The weights of the network are then adjusted using static back-propagation (BP) to minimise the sum of the squared error.
Once its weights and biases are obtained, the NN model can be used to perform one-step-ahead prediction:

\[
\hat{y}(k) = f[y(k-1), \ldots, y(k-K_y), u(k-1), \ldots, u(k-K_u); \varphi]
\]

The prediction architecture is identical to the one used for identification (Figure 4.4). This method is referred to as the equation error method [58], or serial identification method [46].
- The parallel mode

The serial mode discussed above can only perform one-step-ahead prediction. If more than one-step-ahead prediction is required, the parallel mode must be used. There are also two possible operation modes under this circumstance.

1) Serial identification and parallel prediction

In this circumstance, identification architecture is identical to the one that is used in the serial mode. Once the weights and the biases are obtained, the NN model can be used to perform \(d\)-step-ahead \((d>1)\) prediction:

\[
\hat{y}'(k) = f_x[\hat{y}'(k-1),\ldots,\hat{y}'(k-K_y);u(k-1),\ldots,u(k-K_u);\varphi] \quad (k=1,\ldots,d) \quad (4-15)
\]

The prediction architecture can be shown in Figure 4.5. Hereinafter this structure is called parallel mode 1.

2) Parallel identification and parallel prediction

In this case, the inputs to the network consist of the system’s past \(K_u\) input values and NN’s past \(K_y\) output values. The network’s output \(\hat{y}'\) is compared with the next observation of the system’s output to yield the error:

\[
e(k+1) = y(k+1) - \hat{y}'(k+1) \quad (4-16)
\]

The weights of the network can then be adjusted to minimize the error using existing algorithms.

Once the weights and the biases of NN model have been obtained, the model can be used to carry out \(d\)-step-ahead \((d>1)\) prediction:

\[
\hat{y}'(k) = f_x[\hat{y}'(k-1),\ldots,\hat{y}'(k-K_y);u(k-1),\ldots,u(k-K_u);\varphi] \quad (k=1,\ldots,d) \quad (4-17)
\]

This identification and prediction architecture can be depicted by Figure 4.6.

Hereinafter this case is called parallel mode 2.
The last important question in system identification or modelling is that of system identifiability \( p_n \), i.e., given a particular model structure, can the system under study be adequately represented within that structure? In the absence of the concrete theoretical
results for NNs we proceed under the assumption that all systems we are studying belong to the class of systems that are identifiable by the chosen network.

\[ Y_k \]

\[ \hat{Y}_k \]

\[ \mathbf{h}_k \]

\[ \mathbf{k}_{u-1} \]

\[ \mathbf{k}_y \]

\[ \text{: Time-delay unit} \]

**Figure 4.6 Identification and Prediction architecture of parallel mode 2**
4.5 Methods of Modelling the Robot Dynamics

In order to model robot dynamics using neural network, the serial and parallel identification and prediction modes can be used. Various reports in the literature have shown applications of these methods to robot dynamics modelling.

In particular, Fang and Dissanayake [118] proposed a method by using an MLP as follows based upon the property of mechanical systems that instantaneous accelerations only depend on current positions, velocities and actuator forces.

The dynamic equation of a robot system (2-1) can be written as a set of ordinary differential equations as $D^{-1}$ always exists:

$$\ddot{q} = F(q(t), \dot{q}(t), \tau(t))$$

(4-18)

where

$$F(q(t), \dot{q}(t), \tau(t)) = D^{-1}(q(t))\{\tau - h(q(t), \dot{q}(t)) - C(q(t)) - f\}$$

Since MLPs can be used to approximate any non-linear function, it is possible to use such neural networks to emulate the function $F(.)$. The dynamic behaviour of the robot may then be obtained by solving the set of ordinary differential equations (4-18), using a numerical algorithm such as the Runge-Kutta method.

The MLP that emulates the function $F(.)$ can be trained using data obtained by exciting the robot with input signals and measuring the joint positions, velocities, accelerations and input torques. During the training process, joint accelerations in the resulting data sets are used as outputs whereas joint positions, velocities and torque signals are treated as inputs. Figure 4.7 shows the block diagram of this method.

However, in practice, the measurement of acceleration with high accuracy is difficult. The measurement errors may cause deterioration in the modelling results. Therefore, in this thesis a new method is proposed to overcome this problem. This method is to difference (4-18) by taking numerical differentiation and first proposed by the author of this thesis.
As we know for a one variable function \( q(t) \), one can expand it using Taylor formula:

\[
q(t + h) = q(t) + h q'(t) + \frac{h^2}{2!} q''(t) + \frac{h^3}{3!} q'''(t_1) + \cdots \quad t < t_1 < t + h \tag{4-19}
\]

\( \tau(t) \): input torque, \( a(t) \): robot acceleration, \( v(t) \): robot velocity, \( p(t) \): robot position

\[
\text{: time-delay unit}
\]

Figure 4.7 Method of modelling robot dynamics proposed by Fang [16]

or:

\[
q(t - h) = q(t) - h q'(t) + \frac{h^2}{2!} q''(t) - \frac{h^3}{3!} q'''(t_1) + \cdots \quad t - h < t_1 < t \tag{4-20}
\]

Through (4-19) - (4-20) we have:

\[
q'(t) = \frac{q(t + h) - q(t - h)}{2h} - \frac{h^2}{12} [q''(t_1) + q'''(t_3)]
\]

\[
= \frac{q(t + h) - q(t - h)}{2h} - \frac{h^2}{6} q'''(t_3) \quad t - h < t_1 < t + h \tag{4-21}
\]

(4-21) illustrates that there is an \( O(h^3) \) of error when using first order difference quotient as a substitute for first order derivative. Similarly, by applying Taylor formula:

\[
q(t + h) = q(t) + h q'(t) + \frac{h^2}{2!} q''(t) + \frac{h^3}{3!} q'''(t_1) + \frac{h^4}{4!} q^{(4)}(t_3) \quad (t > t_1 > t + h) \tag{4-22}
\]

\[
q(t - h) = q(t) - h q'(t) + \frac{h^2}{2!} q''(t) - \frac{h^3}{3!} q'''(t_1) + \frac{h^4}{4!} q^{(4)}(t_3) \quad (t > t_1 > t - h) \tag{4-23}
\]

From (4-22) + (4-23) we have:

\[
q'(t) = \frac{q(t + h) - 2q(t) + q(t - h)}{h^2} - \frac{h^2}{4!} [q^{(4)}(t_3) + q^{(4)}(t_3)]
\]
\[
\frac{q(t + h) - 2q(t) + q(t - h)}{h^2} - \frac{h^2}{12}q^{(4)}(t) \quad t - h > t_M > t + h
\]

(4-24)

(4-24) illustrates that there is an \( O(h^2) \) of error when using second order difference quotient as a substitute for second order derivative.

When \( h \) is small enough, one can use the first order difference quotient in (4-21) and second order difference quotient (4-24) to describe dynamic process (4-18). That is:

\[
\begin{align*}
q(t + h) &= G_1(q(t), q(t - h), \tau(t)) \\
\dot{q}(t + h) &= G_2(q(t), \dot{q}(t), q(t - h), \tau(t))
\end{align*}
\]

(4-25)

where \( G_1 \) and \( G_2 \) are certain non-linear functions. It means, when being inputted the robot's current and past states (displacements and velocities) and torque, the equation (4-25) can be used to calculate the future states of the robot. Thus if a neural network prediction model is trained by (4-25), the NN model can be used to model the robot dynamic behaviour without the measurement of the joint acceleration, for robot dynamic modelling.

4.6 Simulation Results

4.6.1 Neural network training

The simulation model described in chapter 2 is used in this section to demonstrate the proposed neural network for robot dynamics modelling methods. Training data for NN model are obtained by measuring the robot joint states and torques during the running of the robot simulation model. Here the SIMULINK model of the PUMA robot developed in Chapter 2 is used for this purpose. The parameters for this SIMULINK model are shown in column "Exact model" of Table 4.2.

To excite the robot system, sinusoidal torque signals \( \tau = \tau_{\text{rand}} \sin(2\pi f_{\text{rand}}t) \) are used with random magnitudes \( \tau_{\text{rand}} \) and random frequencies \( f_{\text{rand}} \). The maximum torque
(\(\tau_{\text{max}}\)) of each link is bounded by the capacity of its actuator as stated in Table 4.2. The frequency range of the sinusoidal signal was set to be lower than half of the link natural
frequencies, i.e., \(f_{\text{rand}} \leq 2.5\) Hz. Input torques \(\tau_1, \tau_2\) and \(\tau_3\), and corresponding time
responses at 5 msec interval are then collected. A total of 250 sets of torque signals are
used to excite the robot and 28,000 data sets are collected within the range of \(q \in [-\pi, \pi]\)
(rad), \(\dot{q} \in [-8, 8]\) (rad/sec).

To model the forward dynamics of a robot, a fully connected feedforward neural
network is used in this section. The feedforward neural network model has a structure of
15 inputs, which are three joint displacements \(q_1, q_2\) and \(q_3\), three joint velocities \(\dot{q}_1, \dot{q}_2\) and \(\dot{q}_3\) at the time of \(t\) and \(t-1\) separately, and three input torques \(\tau_1(t), \tau_2(t)\) and
\(\tau_3(t)\). Two hidden layers with 45 and 20 neurons respectively, and the output layer with
six neurons representing \(q_1(t+1), q_2(t+1), q_3(t+1), \dot{q}_1(t+1), \dot{q}_2(t+1)\) and \(\dot{q}_3(t+1)\).
The activation function used in each neuron in the hidden layers is the hyperbolic tangent
function expressed in (4-26). The activation function used in each neuron in the output
layer is the linear function as shown in (4-27). The algorithm used to train the neural
network model is the standard backpropagation. The momentum and training rate are set
to be 0.95 and 0.005 respectively.

\[
f(x) = \frac{(1 - \exp(-2x))}{(1 + \exp(2x))}
\]

\[
f(x) = x
\]

(4-26)

(4-27)

Then, the neural network model is trained with the data sets which are collected
from running the SIMULINK model that represents the real robot. The training is stopped
after 500,000 iterations, when the average errors are reduced to less than 5% with no
further significant reduction being observed.
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Nominal Model</th>
<th>Exact Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1 (m)$</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$d_2 (m)$</td>
<td>0.1491</td>
<td>0.1491</td>
</tr>
<tr>
<td>$d_3 (m)$</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$a_1 (m)$</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$a_2 (m)$</td>
<td>0.4318</td>
<td>0.4318</td>
</tr>
<tr>
<td>$a_3 (m)$</td>
<td>-0.0203</td>
<td>-0.0203</td>
</tr>
<tr>
<td>Mass of the first link $M_1 (kg)$</td>
<td>12.96</td>
<td>12.96</td>
</tr>
<tr>
<td>Mass of the second link $M_2 (kg)$</td>
<td>22.34</td>
<td>22.34</td>
</tr>
<tr>
<td>Mass of the third link (including the $4^{th}$ and $5^{th}$ link) $M_3 (kg)$</td>
<td>6.97</td>
<td>6.97</td>
</tr>
<tr>
<td>Center of mass for the first link along $x$ ($X_1, m$)</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Center of mass for the first link along $y$ ($Y_1, m$)</td>
<td>0.3088</td>
<td>0.3088</td>
</tr>
<tr>
<td>Center of mass for the first link along $z$ ($Z_1, m$)</td>
<td>0.0389</td>
<td>0.0389</td>
</tr>
<tr>
<td>Center of mass for the 2nd link along $x$ ($X_2, m$)</td>
<td>-0.3289</td>
<td>-0.3289</td>
</tr>
<tr>
<td>Center of mass for the 2nd link along $y$ ($Y_2, m$)</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>Center of mass for the 2nd link along $z$ ($Z_2, m$)</td>
<td>0.2038</td>
<td>0.2038</td>
</tr>
<tr>
<td>Center of mass for the 3rd link along $x$ ($X_3, m$)</td>
<td>0.0204</td>
<td>0.0204</td>
</tr>
<tr>
<td>Center of mass for the 3rd link along $y$ ($Y_3, m$)</td>
<td>0.0137</td>
<td>0.0137</td>
</tr>
<tr>
<td>Center of mass for the 3rd link along $z$ ($Z_3, m$)</td>
<td>0.1244</td>
<td>0.1244</td>
</tr>
<tr>
<td>Moment of inertia for the first link along $x$ ($I_{1xx} k g·m^2$)</td>
<td>2.35</td>
<td>2.82</td>
</tr>
<tr>
<td>Moment of inertia for the first link along $y$ ($I_{1yy} k g·m^2$)</td>
<td>0.2</td>
<td>0.24</td>
</tr>
<tr>
<td>Moment of inertia for the first link along $z$ ($I_{1zz} k g·m^2$)</td>
<td>2.35</td>
<td>2.82</td>
</tr>
<tr>
<td>Moment of inertia for the 2nd link along $x$ ($I_{2xx} k g·m^2$)</td>
<td>1.33</td>
<td>1.596</td>
</tr>
<tr>
<td>Moment of inertia for the 2nd link along $y$ ($I_{2yy} k g·m^2$)</td>
<td>3.03</td>
<td>3.636</td>
</tr>
<tr>
<td>Moment of inertia for the 2nd link along $z$ ($I_{2zz} k g·m^2$)</td>
<td>3.38</td>
<td>4.066</td>
</tr>
<tr>
<td>Moment of inertia for the 3rd link along $x$ ($I_{3xx} k g·m^2$)</td>
<td>0.3148</td>
<td>0.3778</td>
</tr>
<tr>
<td>Moment of inertia for the 3rd link along $y$ ($I_{3yy} k g·m^2$)</td>
<td>0.3128</td>
<td>0.3754</td>
</tr>
<tr>
<td>Moment of inertia for the 3rd link along $z$ ($I_{3zz} k g·m^2$)</td>
<td>0.01</td>
<td>0.012</td>
</tr>
<tr>
<td>Gear ratio of the first link drive chain ($r_1$)</td>
<td>0.0</td>
<td>62.61</td>
</tr>
<tr>
<td>Gear ratio of the second link drive chain ($r_2$)</td>
<td>0.0</td>
<td>107.82</td>
</tr>
<tr>
<td>Gear ratio of the third link drive chain ($r_3$)</td>
<td>0.0</td>
<td>53.71</td>
</tr>
<tr>
<td>Moment of inertia of the motor armature driving the first joint ($I_{m1}$)</td>
<td>0.0</td>
<td>0.0002</td>
</tr>
<tr>
<td>Moment of inertia of the motor armature driving the second joint ($I_{m2}$)</td>
<td>0.0</td>
<td>0.0002</td>
</tr>
<tr>
<td>Moment of inertia of the motor armature driving the third joint ($I_{m3}$)</td>
<td>0.0</td>
<td>0.0002</td>
</tr>
<tr>
<td>Viscous friction coefficient of the first joint ($v_{1, N-s}$)</td>
<td>4.0</td>
<td>0.01</td>
</tr>
<tr>
<td>Viscous friction coefficient of the second joint ($v_{2, N-s}$)</td>
<td>2.5</td>
<td>0.005</td>
</tr>
<tr>
<td>Viscous friction coefficient of the third joint ($v_{3, N-s}$)</td>
<td>2.5</td>
<td>0.003</td>
</tr>
<tr>
<td>Coulomb friction of the first joint ($c_{1, N-m}$)</td>
<td>0.0</td>
<td>0.6</td>
</tr>
<tr>
<td>Coulomb friction of the second joint ($c_{2, N-m}$)</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td>Coulomb friction of the third joint ($c_{3, N-m}$)</td>
<td>0.0</td>
<td>0.9</td>
</tr>
<tr>
<td>Approximated Coulomb friction in low velocity of the first joint ($v_{1l}$)</td>
<td>0.0</td>
<td>600</td>
</tr>
<tr>
<td>Approximated Coulomb friction in low velocity of the second joint ($v_{1l}$)</td>
<td>0.0</td>
<td>500</td>
</tr>
<tr>
<td>Approximated Coulomb friction in low velocity of the third joint ($v_{1l}$)</td>
<td>0.0</td>
<td>900</td>
</tr>
<tr>
<td>Rated voltage of the motor driving the first joint ($V_{m1}$)</td>
<td>0.0</td>
<td>40</td>
</tr>
<tr>
<td>Rated voltage of the motor driving the second joint ($V_{m2}$)</td>
<td>0.0</td>
<td>40</td>
</tr>
<tr>
<td>Rated voltage of the motor driving the third joint ($V_{m3}$)</td>
<td>0.0</td>
<td>40</td>
</tr>
</tbody>
</table>
4.6.2 Simulation Results

To examine the accuracy of this neural network model, a number of tests are performed. Results for two of these cases are presented. The driving torques for these two cases are given in Table 4.3. The neural network model is used to produce the approximated joint positions and angular velocities of the robot in response to the driving torques.

<table>
<thead>
<tr>
<th>Case</th>
<th>Joint 1 torque (N-m)</th>
<th>Joint 2 torque (N-m)</th>
<th>Joint 3 torque (N-m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 &amp; 4.3</td>
<td>15 sin(4πt)</td>
<td>8sin(3πt)</td>
<td>6sin(2πt)</td>
</tr>
<tr>
<td>4.2 &amp; 4.4</td>
<td>15sin(πt)</td>
<td>3sin(2πt)</td>
<td>1sin(3πt)</td>
</tr>
</tbody>
</table>

In cases 4.1 and 4.2, the neural network model performs a one-step-ahead prediction. That is, the neural network model is used in the form of the serial mode as mentioned in section 4.4. Figures 4.8 and 4.9 show the results obtained from the neural network model and the robot simulation system. As a comparison, the prediction errors of the neural network model and the model with the nominal parameters as shown in Table 4.2, are depicted in Figures 4.10 and 4.11. From these two cases it is found that the neural network model performs better than the nominal model. On average the prediction error using the neural network model is only 40% of that using the nominal model.

As discussed in Chapter 3, by extending the prediction horizon, the tracking errors of the MPC controller, caused by the model uncertainties and the limit conditions of the controller, could be reduced. This requires the prediction model to perform multi-step-ahead prediction. In cases 4.3 and 4.4, the feedforward neural network's ability to perform multi-step-ahead prediction is investigated. In these cases, the driving torques are given in Table 4.3. The neural network is used in the form of parallel mode 1 as mentioned in the section 4.4. That is, the outputs of the feedforward neural network
Figure 4.8 (a)—joint 1

Figure 4.8 (b)—joint 2

Figure 4.8 (c)—joint 3

Figure 4.8 The testing results obtained from MLP robot dynamic identifier — case 4.1
Figure 4.9 (a)—Joint 1

Figure 4.9 (b)—Joint 2

Figure 4.9 (c)—Joint 3

Figure 4.9: The testing results obtained from MLP robot dynamic identifier — case 4.2

+: NN
-: Robot
Figure 4.10 Prediction error comparison of the response of the NN identifier and the nominal model — case 4.1.
Figure 4.11 Prediction error comparison of the response of the NN identifier and the nominal model — case 4.2
Figure 4.12 (a)—joint 1

Figure 4.12 (b)—joint 2

Figure 4.12 (c)—joint 3

Figure 4.12 3 step ahead prediction — case 4.3
Figure 4.13 (a)—joint 1

Figure 4.13 (b)—joint 2

Figure 4.13 (c)—joint 3

Figure 4.13 5 step ahead prediction — case 4.3
Figure 4.14 (a)—joint 1

Figure 4.14 (b)—joint 2

Figure 4.14 (c)—joint 3

Figure 4.14 3 step ahead prediction — case 4.4
Figure 4.15 (a)—joint 1

Figure 4.15 (b)—joint 2

Figure 4.15 (c)—joint 3

Figure 4.15 5 step ahead prediction — case 4.4
model will be chained back to the model itself to perform multi-step-ahead prediction. Figures 4.12 — 4.15 are the results of 3-step-ahead and 5-step-ahead predictions for both cases 4.3 and 4.4. It is found that the fluctuation amplitudes of the outputs of the feedforward neural network model away from the responses of the real robot tend to become greater as the prediction steps increase.

From the results presented above, it is clear that the feedforward neural network, trained by the experimental data, can only be used to model the robot dynamics accurately in a predictive way within several predictive time steps. Because of the accumulated errors, the feedforward neural network model is not suitable to perform long-term prediction. Several experiments performed by the author showed that the errors increase drastically when the feedforward neural network is used to perform more than the 10-step-ahead prediction.

4.7 Summary

A new method of using neural networks to model the robot forward dynamics have been introduced in this chapter. This method predicts the robot dynamic behaviour without using the robot joint accelerations. As the NN model can be trained using data collected from operating the physical robot. Therefore, the NN model can potentially describe the robot dynamics more accurately than the mathematical model derived from theoretical analyses. It is shown that by using the back-propagation training method and experimental data, the feedforward neural network has been successfully applied to predict the robot dynamic behaviour for certain time steps. The limitation with this method is that it is not suitable to perform long term, that is, more than 10 time steps, prediction. However, if the model is only required to perform the prediction for certain time steps, which is the case in the MPC application, the proposed neural network model can be used satisfactorily.
In the next chapter, the validity of the MPC control strategy based on the FNN prediction model will be investigated. The robot to be controlled is the simulated robot system developed in Chapter 2.
Chapter 5
Neuro-MPC Control of a Robot Manipulator

In Chapter 3, it is shown that to improve the performance of the MPC controller, an accurate model is required. Furthermore, in Chapter 4, it is shown that a neural network can be used to model the robot dynamics accurately. Thus, by combining the MPC controller with the neural network model a more accurate control method may result. In this chapter the validity of the model predictive control method for robot position control proposed in Chapter 3, based on a feedforward neural network (FNN) prediction model, will be studied. The simulation will be performed on the complete simulated robot system developed in Chapter 2.

5.1 Fortran Model of the Simulated Robot System

Because the MATLAB optimisation toolbox can not be easily connected to a simulation model, and the MATLAB Model Predictive Control toolbox can not use a non-linear model, in this section, the Simulink model of the PUMA robot system as shown in Figure 2.9, will be rebuilt using the FORTRAN language. This is because in FORTRAN language there are many optimisation subroutines that can be used to realise the MPC. The study of the MPC control using an NN model of robot will be carried out using this FORTRAN robot system. The block diagram of the proposed FORTRAN model of the robot system is shown in Figure 5.1.

In Figure 5.1, the actuator subroutine is the FORTRAN realisation of the Figure 2.10. The gear backlash and dynamic equation subroutines are the FORTRAN realisation of the M-file subroutines backlash.m and robdyn.m which were separately described in
Figure 5.1 Block Diagram of FORTRAN Model of the Simulated Robot System
Chapter 2. Figures 5.2 and 5.3 are the responses of the Fortran model of the simulated robot system to the excitations shown as those in Table 5.1.

Table 5.1

<table>
<thead>
<tr>
<th>Case</th>
<th>Joint 1 Voltage (Volts)</th>
<th>Joint 2 Voltage (Volts)</th>
<th>Joint 3 Voltage (Volts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>step1 *</td>
<td>step2 **</td>
<td>step3 ***</td>
</tr>
<tr>
<td>5.2</td>
<td>20 sin2πt</td>
<td>10 sin3πt</td>
<td>5 sin4πt</td>
</tr>
</tbody>
</table>

* Step1: Step signal. Initial value is 0, final value is 1. Step time is 0.5 second.
** Step2: Step signal. Initial value is 0, final value is 3. Step time is 0.5 second.
*** Step3: Step signal. Initial value is 0, final value is 5. Step time is 0.5 second

Figures 5.4 and 5.5 are the errors between the running results of the Fortran model and SIMULINK model of the robot to those excitations as shown in Table 5.1. It can be seen that the differences are insignificant — about 1% of the actual outputs. It is believed that the errors mainly come from the different numerical methods for differentiation, integration and differential equation, and different rounding precisions during calculations. Therefore, this FORTEAN model will be used as the 'robot' to perform all the simulations in the following sections.

5.2 Training of the NN Prediction Model

This part discusses the training of the FNN prediction model for the MPC controller. Here the training data for the FNN prediction model are obtained by measuring the robot joint states and voltages during the running of the real robot. The Fortran model of the PUMA robot developed in section 5.1 is used for this purpose. The parameters for this model are shown in column "Exact model" of Table 4.2.
Figure 5.2(a) — joint 1 position

Figure 5.2(b) — joint 2 position

Figure 5.2(c) — joint 3 position

Figure 5.2 Case 5.1: The step response of the Fortran model of the simulated robot system
Figure 5.3(a) — Joint 1 position

Figure 5.3(b) — Joint 2 position

Figure 5.3(c) — Joint 3 position

Figure 5.3 Case 5.2: The sinusoidal response of the Fortran model of the simulated robot system
Figure 5.4(a) — joint 1

Figure 5.4(b) — joint 2

Figure 5.4(c) — joint 3

Figure 5.4 Errors between the Fortran model and SIMULINK model

Case 5.1
Figure 5.5(a) — Joint 1

Figure 5.5(b) — Joint 2

Figure 5.5(c) — Joint 3

Figure 5.5 Errors between the Fortran model and SIMULINK model

Case 5.2
To excite the robot system, sinusoidal voltage signals \( v = V_{\text{rand}} \sin(2\pi f_{\text{rand}} t) \) are used with random magnitudes \( V_{\text{rand}} \) and random frequencies \( f_{\text{rand}} \). The maximum voltage \( V_{\text{max}} \) of each link is bounded by the rated voltage of its actuator as stated in Table 4.2. The frequency range of the sinusoidal signal was set to be lower than half of the link natural frequencies, i.e., \( f_{\text{rand}} \leq 2.5 \) Hz. Input voltages \( v_1, v_2 \) and \( v_3 \), and corresponding time responses at 5 msec interval are then collected. A total of 250 sets of voltage signals are used to excite the robot and 35,000 data sets are collected within the range of \( q \in [-\pi, \pi] \) (rad), \( \dot{q} \in [-8, 8] \) (rad/sec).

To model the dynamic behaviour of the PUMA robot, a fully connected feedforward neural network is used in this section. The feedforward neural network model has a structure of 15 inputs, which are three joint displacements \( q_1, q_2 \) and \( q_3 \), three joint velocities \( \dot{q}_1, \dot{q}_2 \) and \( \dot{q}_3 \) at the time of \( t \) and \( t-1 \) separately, and three input voltages \( v_1(t), v_2(t) \) and \( v_3(t) \). Two hidden layers with 45 and 20 neurons respectively, and the output layer with six neurons representing \( q_1(t+1), q_2(t+1), q_3(t+1), \dot{q}_1(t+1), \dot{q}_2(t+1) \) and \( \dot{q}_3(t+1) \). The activation function used in the hidden layers for each neuron is the hyperbolic tangent function expressed in (4-26). The activation function used in the output layer for each neuron is the linear function as shown in (4-27). The algorithm used to train the neural network model is the standard backpropagation. The momentum and training rate are set to be 0.95 and 0.005 respectively.

Then, the feedforward neural network model is trained with the data sets which are collected from running the Fortran model that represents robot simulation model built up in Chapter 2. As stated above, the FNN prediction model has a structure of 15-45-20-6. The training is stopped after 2 million iterations, when the average errors are reduced to less than 5% with no further significant reduction being observed.
5.3 Simulation Results of the MPC Controller based on an FNN Prediction Model

Having been built up, the FNN prediction model could be used for the MPC control of the PUMA robot system.

To test the performance of the proposed model predictive control strategy, the desired trajectories for the robot system to follow are generated by inputting the system a group of sinusoidal excitations, in which the amplitudes and the frequencies are within the amplitude and frequency limits used in section 5.2.

In case 5.3, the desired trajectories, shown in Figure 5.3, are produced by inputting the system voltage signals as those of case 5.2 in Table 5.1. The errors for the PUMA robot system to track these trajectories, under the control of the MPC controller based on an FNN model, are shown in Figure 5.6. All the errors in this case are at $O(10^{-3})$.

In cases 5.4 and 5.5 the effect of the MPC control method using the neural network (annotated as NNMPC) will be compared with that of the MPC based on a nominal model with both structured and unstructured errors (annotated as NMMPC). The nominal model consists of the parameters which can be found in the column "nominal model" of Table 4.2. The desired trajectories for the robot to track are shown in Figure 5.7 and Figure 5.9. The tracking errors by applying NNMPC and NMMPC to track the above mentioned desired trajectories are shown in Figure 5.8 and Figure 5.10 separately. The results show that NNMPC provides more accurate tracking performance than the NMMPC, as predicted. It is clear, however, that the performance of the NMMPC will vary depend on the accuracy of the nominal model. For example, the tracking errors of the NMMPC can be reduced if the nominal model can be obtained more accurately.
Figure 5.6(a) — Joint 1 error

Figure 5.6(b) — Joint 2 error

Figure 5.6(c) — Joint 3 error

Figure 5.6 Tracking errors —

Model Predictive Control based on A Neural Network Model for case 5.3
Figure 5.7(a) — joint 1 position

Figure 5.7(b) — joint 2 position

Figure 5.7(c) — Joint 3 Position

Figure 5.7 Desired Trajectories — case 5.4
Figure 5.8(a) — joint 1 error

Figure 5.8(b) — joint 2 error

Figure 5.8(c) — joint 3 error

Figure 5.8 Tracking errors —
Model Predictive Control Based on a Neural Network Model for case 5.4
Figure 5.9(a) — Joint 1 position

Figure 5.9(b) — Joint 2 position

Figure 5.9(c) — Joint 3 Position

Figure 5.9 Desired Trajectories — case 5.5
Figure 5.10(a) — joint 1 error

Figure 5.10(b) — joint 2 error

Figure 5.10(c) — joint 3 error

Figure 5.10 Tracking Errors—
Model Predictive Control Based on a Neural Network Model for case 5.5
5.4 Summary

In this chapter a method of using MPC together with an NN model of the robot for robot trajectory control is proposed. As an accurate prediction model can be obtained by using the feedforward neural network that is trained using the actual measurement during operating the robot, therefore, better tracking performances by using MPC based on the NN model can be expected. It is shown using simulation that the proposed NNMPC method outperforms the NMMPC, when the nominal model has some uncertainties.

In next chapter, conclusions of this thesis will be given, and some recommendations will be discussed in regards to further this study.
Chapter 6
Conclusions and Recommendations

6.1 Summary and Conclusions

In this thesis a simulation model of a PUMA robot has been built. This model incorporates the actuator effects, gear backlash and robot dynamic equation with inertia, centrifugal and Coriolis, frictional and gravitational terms together. The model is based on a PUMA robot and is built in both Matlab/Simulink and FORTRAN programming languages. This simulation model was treated as the "real" robot after it had been built up. Researches of the proposed modelling methods and control strategy were carried out using this model.

The MPC control strategy has been proposed in this thesis for robot trajectory tracking. The feasibility of the MPC based on the robot forward dynamics model with different model uncertainties has also been investigated. Through simulation it has been found that the error sources of the MPC controller come mainly from the saturation of the control action, measurement errors and the model uncertainties. The effect of the saturation of the control action and the model uncertainties to the control accuracy can be improved by extending the prediction horizons. The high frequency components of the control actions can be eliminated effectively by introducing the limit condition on the control action. However smoother control actions lead to larger tracking errors. Through the proper selection of the limit condition higher accuracy of the trajectory tracking can be obtained whereas the smoothness of the control action can be maintained at a reasonable level.
A neural network method for robot modelling has also been developed in this thesis. An input-output prediction model, using feedforward neural network (multi-layer perception), has been established. The data to train this FNN model can come from the measurement of running a robot model. The simulation showed that the FNN model performs better than the nominal model with some uncertainties. The limitation with the FNN model is that it is not suitable to perform long term prediction. However, if the model is only required to perform the prediction for a certain time steps, which is the case in the MPC application, the proposed FNN model can be used satisfactorily.

By combining the MPC controller with the neural network model a more accurate control method resulted. The simulation showed that the proposed NNMPC method outperforms the NMMPC, in which the nominal model had some uncertainties.

6.2 Recommendations

From the results of this study and the knowledge obtained while working with the MPC control of the simulated robot system, the following recommendations are put forward.

The main problem with the MPC controller is its long computational time and thus it is difficult to implement in real time. Therefore the real time calculation of the MPC should be the future research topic. One of the possible ways to solve this problem is to use neural network’s parallel computational capability. Some researchers have suggested that neural networks have the potential to speed up the implementation of the MPC. Therefore the implementation of MPC computation using neural networks could be the field of future study.

It has been shown in previous chapters that the control accuracy of the MPC controller depends on the prediction steps. That is, the control accuracy of the MPC
controller could be improved by increasing the prediction steps used in MPC. Due to the one directional connection of the feedforward neural networks, the NN modelling method proposed in this thesis, can only perform precise prediction for limited time-steps. As it has been mentioned in Chapter 4, to carry out multi-step prediction, a parallel mode of the identification structure should be used. This implies that an RNN may be better than FNN to model the dynamic behaviour of a robot in multiple steps. Therefore, it is suggested that the feasibility of using RNN as the NNMPC prediction model be investigated in future research.
Appendices

Appendix A.1 Parameters for PUMA robot dynamic equation

Assume the values to the parameters in robot dynamic equation
a1 = 0;
a2 = 0.4318;
a3 = -0.02032;
d1 = 0;
d2 = 0.1491;
d3 = 0;
m1 = 12.96;
m2 = 22.34;
m3 = 6.97 + Mload;
x1 = 0;
x2 = -0.3289;
x3 = 0.0204 + (0.43307 + 0.05625 - 0.0204) * Mload / m3;
y1 = 0.3088;
y2 = 0.005;
y3 = 0.0137;
z1 = 0.0389;
z2 = 0.2038;
z3 = 0.1244;
i1xx1 = 2.82;
i1yy1 = 0.24;
i1zz1 = 2.82;
i2xx1 = 1.596;
i2yy1 = 3.636;
i2zz1 = 4.056;
i3xx1 = 0.3778;
i3yy1 = 0.3754 + Mload * (0.43307 + 0.05625) ^ 2;
i3zz1 = 0.012 + Mload * (0.43307 + 0.05625) ^ 2;
g = 9.81;
vrl1 = 600;
vrl2 = 500;
vrl3 = 900;
\[ f_{x1} = 0.6; \]
\[ f_{x2} = 0.5; \]
\[ f_{x3} = 0.9; \]
\[ v_{r1} = 0.01; \]
\[ v_{r2} = 0.005; \]
\[ v_{r3} = 0.003; \]
\[ n_{q1} = 62.61; \]
\[ n_{q2} = 107.82; \]
\[ n_{q3} = 53.71; \]
\[ j_{m1} = 0.0002; \]
\[ j_{m2} = 0.0002; \]
\[ j_{m3} = 0.0002; \]
\[ j_{gl} = 0; \]
\[ j_{g2} = 0; \]
\[ j_{g3} = 0; \]
\[ i_{1xx} = i_{1xx1}; \]
\[ i_{1yy} = i_{1yy1}; \]
\[ i_{1zz} = i_{1zz1}; \]
\[ i_{2xx} = i_{2xx1}; \]
\[ i_{2yy} = i_{2yy1}; \]
\[ i_{2zz} = i_{2zz1}; \]
\[ i_{3xx} = i_{3xx1}; \]
\[ i_{3yy} = i_{3yy1}; \]
\[ i_{3zz} = i_{3zz1}; \]
Appendix A.2 Calculation of the inertia term

% Calculation of inertia terms in the robot dynamic equation
% d11
% d111=a2*a2*cos(th2)*cos(th2)*m3;
d111s=d2*d2*m3+z2*a2*cos(th2)*cos(th2)*m2*x2;
d112=12xx+2*sin(th2+th3)*m3*x3*z3*cos(th2+th3)*iiyy;
d113=-cos(th2+th3)*cos(th2+th3)*i3xx+i3xx+2*d2*m3*y3;
d114=a2*a2*cos(th2)*cos(th2)*m2+d2*d2*m2;
d114s=-2*sin(th2)*m2*y2+a2*cos(th2);
d115=a3*a3*cos(th2+th3)*cos(th2+th3)*m3;
d115s=2*sin(th2+th3)*m3*z3*a3*cos(th2+th3);
d116=cos(th2+th3)*cos(th2+th3)*i3zz2*sin(th2)*m2*x2*y2*cos(th2);
d117=2*d2*m2*z2+2*sin(th2+th3)*m3*z3*a2*cos(th2);
d118=2*a2*cos(th2)*m3*x3*cos(th2+th3);
d118s=2*a2*cos(th2)*m3*a3*cos(th2+th3);
d119=2*a3*cos(th2+th3)*cos(th2+th3)*m3*x312xx*cos(th2)*cos(th2);
d1110=12yy*cos(th2)*cos(th2);
d110 = d111+d112+d113+d114+d115+d116+d117+d118+d119+d1110;
d111=d110+d111s+d114s+d115s+d118s+jm1*ngl*ng1+jgl;
%
% d12
% d121=m3*x3*y3*sin(th2+th3)*d2*m3*z3*cos(th2+th3)+d2*m3*a2*sin(th2);
d122=d2*m3*x3*sin(th2+th3)+m3*y3*a3*sin(th2+th3)+m3*y3*a2*sin(th2);
d123=d2*m2*z2*sin(th2)+d2*a2*m2*sin(th2)+m2*y2*z2*cos(th2);
d124=a2*m2*z2*sin(th2)+d2*m2*y2*cos(th2)*m3*y3*z3*cos(th3);
d125=m2*x2*z2*sin(th2)+d2*m3*a3*sin(th2+th3);
d12=d121+d122+d123+d124+d125;
%
% d13
% d131=m3*x3*y3*sin(th2+th3)-m3*y3*z3*cos(th2+th3);
d132=m3*y3*a3*sin(th2+th3)-d2*m3*z3*cos(th2+th3);
d133=d2*m3*a3*sin(th2+th3)+d2*m3*x3*sin(th2+th3);
d13=d131+d132+d133;
%
% d22
\[
\begin{align*}
d221 &= i_3yy + i_2zz + 2a_3m_3x_3 + 2a_2\sin(th2) \cdot m_3 \cdot a_3 \cdot \sin(th2+th3); \\
d222 &= -2a_2\sin(th2) \cdot m_3 \cdot z_3 \cdot \cos(th2+th3) + a_2a_2m_2 + a_3a_3m_3; \\
d223 &= 2a_2m_2x_2 + a_2a_2m_3 + 2\sin(th2+th3) \cdot m_3 \cdot z_3 \cdot a_2 \cdot \cos(th2); \\
d224 &= 2a_2\cos(th2) \cdot m_3 \cdot x_3 \cdot \cos(th2+th3); \\
d224s &= 2a_2\sin(th2) \cdot m_3 \cdot x_3 \cdot \sin(th2+th3); \\
d225 &= 2a_2\cos(th2) \cdot m_3 \cdot a_3 \cdot \cos(th2+th3); \\
d22 &= d221 + d222 + d223 + d224 + d225 + d224s + jm2*ng2*ng2+tg2; \\
\% d23 \\
d231 &= i_3yy + 2a_3m_3x_3 + a_2\sin(th2) \cdot m_3 \cdot a_3 \cdot \sin(th2+th3); \\
d232 &= -a_2\sin(th2) \cdot m_3 \cdot z_3 \cdot \cos(th2+th3) + a_3a_3m_3; \\
d232s &= \sin(th2+th3) \cdot m_3 \cdot z_3 \cdot a_2 \cdot \cos(th2); \\
d233 &= a_2\cos(th2) \cdot m_3 \cdot x_3 \cdot \cos(th2+th3) + a_2\sin(th2) \cdot m_3 \cdot x_3 \cdot \sin(th2+th3); \\
d234 &= a_2\cos(th2) \cdot m_3 \cdot a_3 \cdot \cos(th2+th3); \\
d23 &= d231 + d232 + d233 + d234 + d232s; \\
\% d33 \\
d33 &= i_3yy + 2a_3m_3x_3 + a_3a_3m_3 + jm3*ng3*ng3+jg3; 
\end{align*}
\]
Appendix A.3 Calculation of the centrifugal and Coriolis parameters

tCalculation of the centrifugal and Coriolis terms in robot dynamic equation

% h111
h111 = 0;
%
% h112
h1121 = cos(th2+th3)*sin(th2+th3)*i3zz2*a2*cos(th2)*cos(th2)*m2*y2;
% h1122 = 2*a2*cos(th2)*m2*x2*sin(th2)*a2*a2*cos(th2)*m2*sin(th2);
% h1123 = cos(th2)*sin(th2)*i2yy*cos(th2+th3)*sin(th2+th3)*i3xx;
% h1124 = m3*z3*a3-a3*a3*cos(th2+th3)*m3*sin(th2+th3);
% h1125 = a2*cos(th2)*m3*z3*cos(th2+th3);
% h1126 = 2*a3*cos(th2+th3)*cos(th2+th3)*m3*z3;
% h1126 = -a2*cos(th2)*m3*a3*sin(th2+th3);
% h1127 = cos(th2)*sin(th2)*i2xxa2*cos(th2)*m3*x3*sin(th2+th3);
% h1128 = a2*m2*y2+m2*x2*y2-a2*a2*cos(th2)*m3*sin(th2);
% h1129 = -2*a3*cos(th2+th3)*m3*x3*sin(th2+th3);
% h1129 = -a3*cos(th2+th3)*m3*a2*sin(th2);
% h11210 = m3*x3*z3*sin(th2+th3)*m3*z3*a2*sin(th2);
% h11210 = 2*cos(th2+th3)*cos(th2+th3)*m3*x3*z3;
% h11211 = cos(th2+th3)*m3*x3*a2*sin(th2)*2*cos(th2)*cos(th2)*m2*x2*y2;
% h1120 = h1121+h1122+h1123+h1124+h1125+h1126+h1127+h1128+h1129;
% h112 = h1120+h11210+h11211+h1126s+h1129s+h11210s;
%
% h113
h1131 = -cos(th2+th3)*sin(th2+th3)*i3zz;
% h113s = cos(th2+th3)*sin(th2+th3)*i3xx;
% h1132 = m3*z3*a3-a3*a3*cos(th2+th3)*m3*sin(th2+th3);
% h1133 = a2*cos(th2)*m3*z3*cos(th2+th3);
% h1134 = 2*a3*cos(th2+th3)*cos(th2+th3)*m3*z3;
% h1134 = -a2*cos(th2)*m3*a3*sin(th2+th3);
% h1135 = -2*a3*cos(th2+th3)*m3*x3*sin(th2+th3)-m3*x3*z3;
% h1136 = 2*cos(th2+th3)*cos(th2+th3)*m3*x3*z3;
% h113 = h1131+h1132+h1133+h1134+h1135+h1136+h1131s+h1133s+h1134s;
% h122
\[ h_{122} = d_2 + m_2 + x_2 \cos(th_2) + m_2 + x_2 \cos(th_2) + d_2 + a_2 + m_2 \cos(th_2); \]
\[ h_{122} = m_2 + y_2 + z_2 \sin(th_2) + d_2 + m_2 + y_2 + z_2 \sin(th_2) + a_2 + m_2 + z_2 \cos(th_2); \]
\[ h_{123} = d_2 + m_3 + x_3 \cos(th_2 + th_3) + d_2 + m_3 + x_3 \cos(th_2 + th_3) + a_2 + m_2 + x_2 \cos(th_2); \]
\[ h_{124} = d_2 + a_3 + x_3 \cos(th_2 + th_3) + d_2 + a_3 + x_3 \cos(th_2 + th_3) + m_3 + y_3 + a_2 + z_3 \cos(th_2); \]
\[ h_{125} = m_3 + y_3 + a_3 + z_3 \cos(th_2 + th_3) + m_3 + y_3 + z_3 \cos(th_2 + th_3) + a_2 + y_3 + m_3 \cos(th_2 + th_3); \]
\[ h_{125} = m_3 + x_3 + y_3 \cos(th_2 + th_3); \]
\[ h_{122} = h_{1221} + h_{1222} + h_{1223} + h_{1224} + h_{1225} + h_{1225} = 0; \]
\[
% h_{123}
\[ h_{1231} = m_3 + x_3 + y_3 \cos(th_2 + th_3) + m_3 + y_3 + z_3 \sin(th_2 + th_3); \]
\[ h_{1232} = m_3 + y_3 + a_3 \cos(th_2 + th_3) + d_2 + m_3 + a_3 \cos(th_2 + th_3); \]
\[ h_{1233} = d_2 + m_3 + x_3 \cos(th_2 + th_3) + d_2 + m_3 + x_3 \cos(th_2 + th_3); \]
\[ h_{123} = h_{1231} + h_{1232} + h_{1233} = 0; \]
\[
% h_{133}
\[ h = h_{133}; \]
\[
% h_1
\[ h_1 = h_{111} + h_{112} \cdot x_2 + h_{133} \cdot x_3 + h_{12} \cdot x_2 \cdot x_3 + h_{112} \cdot x_2 + h_{113} \cdot x_3 + h_{123} \cdot x_2 \cdot x_3); \]
\[
% h_{211}
\[ h_{211} = \cos(th_2 + th_3) \cdot \sin(th_2 + th_3) \cdot t_{123}; \]
\[ h_{2111} = 2 + a_2 + \cos(th_2 + th_3) \cdot \cos(th_2 + th_3) \cdot m_2 \cdot y_2; \]
\[ h_{2112} = 2 + a_2 + \cos(th_2 + th_3) \cdot \cos(th_2 + th_3) \cdot m_2 + \sin(th_2); \]
\[ h_{2113} = \cos(th_2 + th_3) \cdot \cos(th_2 + th_3) \cdot \sin(th_2 + th_3) \cdot t_{123}; \]
\[ h_{2114} = m_3 + x_3 + a_3 \cdot \cos(th_2 + th_3) \cdot m_3 + \sin(th_2 + th_3); \]
\[ h_{2115} = -a_2 + \cos(th_2 + th_3) \cdot m_3 \cdot z_3 \cdot \cos(th_2 + th_3); \]
\[ h_{2116} = -2 + a_2 + \cos(th_2 + th_3) \cdot \cos(th_2 + th_3) \cdot m_3 + z_3; \]
\[ h_{2117} = a_2 + \cos(th_2 + th_3) \cdot m_3 + a_3 \cdot \sin(th_2 + th_3) \cdot \cos(th_2 + th_3) \cdot \sin(th_2 + th_3) \cdot i_{23}; \]
\[ h_{2118} = a_2 + \cos(th_2 + th_3) \cdot m_3 + a_3 \cdot \sin(th_2 + th_3) \cdot \cos(th_2 + th_3) \cdot \sin(th_2 + th_3); \]
\[ h_{2119} = i_{23} + a_2 + \cos(th_2 + th_3) \cdot m_3 + \sin(th_2); \]
\[ h_{2119} = i_{23} + a_2 + \cos(th_2 + th_3) \cdot m_3 + \sin(th_2 + th_3); \]
\[ h_{21110} = a_2 + \cos(th_2 + th_3) \cdot m_3 + a_3 \cdot \sin(th_2 + th_3) \cdot m_3 + x_3 + z_3; \]
\[ h_{21111} = \sin(th_2 + th_3) \cdot m_3 + z_3 \cdot a_2 \cdot \sin(th_2); \]
\[ h_{21112} = -2 + \cos(th_2 + th_3) \cdot \cos(th_2 + th_3) \cdot m_3 + x_3 + z_3; \]
\[ h_{21113} = \cos(th_2 + th_3) \cdot m_3 + x_3 + a_2 \cdot \sin(th_2 + th_3) + 2 + \cos(th_2 + th_3) \cdot \cos(th_2 + th_3) \cdot m_2 + x_2 + y_2; \]
h2110 = h2111 + h2112 + h2113 + h2114 + h2115 + h2116 + h2117 + h2118 + h2119;
h211 = h2110 + h21110 + h21111 + h21112 + h21113 + h21114 + h21115 + h21116;

% h212
h212 = 0;
%
% h213
h213 = 0;
%
% h222
h222 = 0;
%
% h223
h2231 = a2 * cos(th2) * m3 * z3 * cos(th2 + th3) - a2 * cos(th2) * m3 * a3 * sin(th2 + th3);
h2232 = a2 * cos(th2) * m3 * x3 * sin(th2 + th3) + a3 * cos(th2 + th3) * m3 * a2 * sin(th2);
h2233 = sin(th2 + th3) * m3 * z3 + a2 * sin(th2) + cos(th2 + th3) * m3 * x3 + a2 * sin(th2);
h223 = h2231 + h2232 + h2233;
%
% h233
h233 = h223;
%
% h2
%
h2 = h211 * x1 + h222 * x2 + h233 * x3 + 2 * (h212 * x12 + h213 * x13 + h223 * x23);
%
% h311
h3111 = cos(th2 + th3) * sin(th2 + th3) * i3 * z3 * cos(th2 + th3) * sin(th2 + th3) * i3 * x3;
h3112 = m3 * z3 * a3 + a3 * a3 * cos(th2 + th3) * m3 * sin(th2 + th3);
h3113 = -a2 * cos(th2) * m3 * z3 * cos(th2 + th3);
h3113s = -2 * a3 * cos(th2 + th3) * cos(th2 + th3) * m3 * z3;
h3114 = a2 * cos(th2) * m3 * a3 * sin(th2 + th3) + a2 * cos(th2) * m3 * x3 * sin(th2 + th3);
h3115 = 2 * a3 * cos(th2 + th3) * m3 * x3 * sin(th2 + th3) + m3 * x3 * z3;
h3116 = 2 * cos(th2 + th3) * cos(th2 + th3) * m3 * x3 * z3;
h311 = h3111 + h3112 + h3113 + h3114 + h3115 + h3116 + h3113s;
%
% h312
h312 = 0;
%
\[ h_{313} \]
\[ h_{313} = 0; \]
\[
\%
\]
\[ h_{322} \]
\[ h_{3221} = -a_2 \cos(\theta_2) \cdot m_3 \cdot z_3 \cdot \cos(\theta_2 + \theta_3); \]
\[ h_{3221s} = a_2 \cos(\theta_2) \cdot m_3 \cdot a_3 \cdot \sin(\theta_2 + \theta_3); \]
\[ h_{3222} = a_2 \cos(\theta_2) \cdot m_3 \cdot x_3 \cdot \sin(\theta_2 + \theta_3); \]
\[ h_{3222s} = -a_3 \cos(\theta_2 + \theta_3) \cdot m_3 \cdot a_2 \cdot \sin(\theta_2); \]
\[ h_{3223} = -a_3 \cos(\theta_2 + \theta_3) \cdot m_3 \cdot z_3 \cdot a_2 \cdot \sin(\theta_2); \]
\[ h_{3223s} = -a_3 \cos(\theta_2 + \theta_3) \cdot m_3 \cdot x_3 \cdot a_2 \cdot \sin(\theta_2); \]
\[ h_{322} = h_{3221} + h_{3222} + h_{3223} + h_{3221s} + h_{3222s} + h_{3223s}; \]
\[
\%
\]
\[ h_{323} \]
\[ h_{323} = 0; \]
\[
\%
\]
\[ h_{333} \]
\[ h_{333} = 0; \]
\[
\%
\]
\[ h_{3} \]
\[ h_{3} = h_{311} \cdot x_{11} + h_{322} \cdot x_{22} + h_{333} \cdot x_{33} + 2 \cdot (h_{312} \cdot x_{12} + h_{313} \cdot x_{13} + h_{323} \cdot x_{23}); \]
\[
\%
Appendix A.4 Calculation of gravity term

Calculation of the gravitational forces in the robot dynamic equation

% c1
c1 = 0;
%
% c2
\[ c_{21} = -g*m_3*x_3*cos(th2+th3) - g*m_3*(a_2*cos(th2)+a_3*cos(th2+th3)); \]
\[ c_{22} = -g*a_2*m_2*cos(th2) - g*m_2*x_2*cos(th2) + g*m_2*y_2*sin(th2); \]
\[ c_{23} = -g*m_3*z_3*sin(th2+th3); \]
c2 = c21 + c22 + c23;
%
% c3
\[ c_{31} = -g*a_3*m_3*cos(th2+th3) - g*m_3*x_3*cos(th2+th3); \]
\[ c_{32} = -g*m_3*z_3*sin(th2+th3); \]
c3 = c31 + c32;
Appendix A.5 Calculation of the PUMA robot dynamics

%PUMA dynamic calculations
function pumaacc=robot(in)
%
global brake1 brake2 brake3 Mload ngl
brk1=brake1;
brk2=brake2;
brk3=brake3;
t1=in(1);
t2=in(2);
t3=in(3);
th1=in(4);
th2=in(5);
th3=in(6);
dth1=in(7);
dth2=in(8);
dth3=in(9);
para; %Parameters for the PUMA dynamic equations
x11 = dth1*dth1;
x12 = dth1*dth2;
x13 = dth1*dth3;
x22 = dth2*dth2;
x23 = dth2*dth3;
x33 = dth3*dth3;
d; %Inertia term of the PUMA dynamic equations
h; %Centrifugal and Coriolis forces
c; %Gravity force
f; %Friction force
v1 = h1+c1+f1;
v2 = h2+c2+f2;
v3 = h3+c3+f3;
%
% CHECK IF ANY BRAKE IS ON
%
if brk1 == 1
if brk2 == 1
    if brk3 == 1
        % All brakes are on
        pumaacc(1) = 0.0;
        pumaacc(2) = 0.0;
        pumaacc(3) = 0.0;
    else % Only brake 3 is off.
        Dm = [-1 0 d13;0 -1 d23;0 0 d33];
        pumaacc = inv(Dm)*[-v1;v2;v3];
        pumaacc(1) = 0.0;
        pumaacc(2) = 0.0;
    end
elseif brk3 == 1 % Only brake 2 is off.
    Dm = [-1 d12 0;0 d22 0;0 d23 -1];
    pumaacc = inv(Dm)*[-v1;t2-v2;v3];
    pumaacc(1) = 0.0;
    pumaacc(3) = 0.0;
else % Only brake 1 is on.
    Dm = [-1 d12 d13;0 d22 d23;0 d23 d33];
    pumaacc = inv(Dm)*[-v1;t2-v2;t3-v3];
    pumaacc(1) = 0.0;
end
elseif brk2 == 1
    if brk3 == 1 % Only brake 1 is off.
        Dm = [d11 0 0;d12 -1 0;d13 0 -1];
        pumaacc = inv(Dm)*[t1-v1;-v2;-v3];
        pumaacc(2) = 0;
        pumaacc(3) = 0;
    else % Only brake 2 is on.
        Dm = [d11 0 d13;d12 -1 d23;d13 0 d33];
        pumaacc = inv(Dm)*[t1-v1;-v2;t3-v3];
        pumaacc(2) = 0;
    end
else
    if brk3 == 1 % Only brake 3 is on.
        Dm = [d11 d12 0;d12 d22 0;d13 d23 -1];
        pumaacc = inv(Dm)*[t1-v1;t2-v2;-v3];
        pumaacc(3) = 0;
    else % All brakes are off.
        Dm = [d11 d12 d13;d12 d22 d23;d13 d23 d33];
        pumaacc = inv(Dm)*[t1-v1;t2-v2;t3-v3];
    end
end
Appendix B Calculation of the friction forces

% Frictional forces --- both viscous and Coulomb friction.
%
if (abs(dth1)) < 0.001
    f1 = vr11*dth1;
else
    f1 = fr1*sign(dth1)+vr1*dth1;
end
if (abs(dth2)) < 0.001
    f2 = vr21*dth2;
else
    f2 = fr2*sign(dth2)+vr2*dth2;
end
if (abs(dth3)) < 0.001
    f3 = vr31*dth3;
else
    f3 = fr3*sign(dth3)+vr3*dth3;
end
%
Appendix C Calculation of gear backlash

% Joint 1 backlash

function T11=Taoll(in)
global b1 deltaT1 SM1
ngl=62.61;
jml=0.002;
jll=4.2;
fml=0.05;
Thetall=in(1);
dThetall=in(2);
Taoll=in(3);
ddThetall=in(4);
dThetall=in(5);
Thetall=in(6);
if (abs(Thetall*ngl+Thetall)<b1)
    T11=0;
end
if (((Thetall*ngl+Thetall==b1)... & (dThetall>ngl*dThetall)||((Thetall*ngl+Thetall==b1)... & (dThetall<ngl*dThetall)))
    T11=(jml*jll1/((Thetall*ngl+Thetall)*deltaT1/(jml+jll1/((ngl*ngl)));
end
if ((Thetall+Thetall*ngl==b1) & (dThetall==0) & (ngl*dThetall==0))
    T11=max(0,Taoll);
else if ((Thetall*Thetall*ngl==b1) & (dThetall==0) & (ngl*dThetall==0))
    T11=min(0,Taoll);
else
    T11=ngl*(Taoll-(jml*ddThetall+fml*dThetall));
end
end
Appendix D.1 Neural predictor training program

% This program is used for NN training

load data.o  % this is the name of the data file.
load mlpwle; % this is the weights calculated in last iteration
dl=data;
mstp=size(dl,1);
j=1;
stp=1;
inp= [dl(1:stp:mstp,1),dl(1:stp:mstp,2),dl(1:stp:mstp,3),...
dl(1:stp:mstp,4),dl(1:stp:mstp,5),dl(1:stp:mstp,6),...
dl(1:stp:mstp,7),dl(1:stp:mstp,8),dl(1:stp:mstp,9),...
dl(1:stp:mstp,10),dl(1:stp:mstp,11),dl(1:stp:mstp,12),...
dl(1:stp:mstp,13)/65,dl(1:stp:mstp,14)/65,dl(1:stp:mstp,15)/65]';
outp=[dl(1:stp:mstp,16),dl(1:stp:mstp,17),...
dl(1:stp:mstp,18),dl(1:stp:mstp,19),...
dl(1:stp:mstp,20),dl(1:stp:mstp,21)]';
clear dl

% initialize the network. The NN structure is 15-45-20-6.
% [w1,b1,w2,b2,w3,b3]=initff(inp,45,'tansig',20,'tansig',6,'purelin');

while j<20001
  figure(1);
  tp=[10,1000,0.0001,0.00001,1.05,0.7,0.9,1.04];
  [w1,b1,w2,b2,w3,b3]=...
  trainbpx(w1,b1,'tansig',w2,b2,'tansig',w3,b3,'purelin',inp,outp,tp);

  % save the weights calculated.
  save mlpwle w1 b1 w2 b2 w3 b3

  j=j+1
end
Appendix D.2 Neural predictor testing program

% This program is used for NN testing
load data.o % This is the data file
load mlpwle % This is the weight file
step=1000;
indx=1:step;
d1=data(1:1:step,:);
inp= [d1(:,1),d1(:,2),d1(:,3),d1(:,4),d1(:,5),d1(:,6),d1(:,7),...
d1(:,8),d1(:,9),d1(:,10),d1(:,11),d1(:,12),...
d1(:,13)/65,d1(:,14)/65,d1(:,15)/65]';
otnn=simnff(inp,w1,b1,'tansig',w2,b2,'tansig',w3,b3,'purelin');
figure(1)
% plotting line 1 of the NN output matrix
plot(indx,otnn(1,1:1:step),'g')
figure(2)
% plotting line 3 of the NN output matrix
plot(indx,otnn(3,1:1:step),'g')
figure(3)
% plotting line 5 of the NN output matrix
plot(indx,otnn(5,1:1:step),'g')
References


16. Fang, G., "Neural Networks for Robot Trajectory Planning and Control", *Ph. D Thesis*, Department of Mechanical Engineering, University of Sydney, 1996


MODEL PREDICTIVE CONTROL
OF
A ROBOT
USING NEURAL NETWORKS

A THESIS SUBMITTED TO
THE SCHOOL OF MECHATRONIC, COMPUTER AND
ELECTRICAL ENGINEERING
THE UNIVERSITY OF WESTERN SYDNEY, NEPEAN
IN FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF ENGINEERING (Honours)

By
Zhouping Wei

June 1999
PLEASE NOTE

The greatest amount of care has been taken while scanning this thesis,

and the best possible result has been obtained.
Abstract

The aim of the thesis is to develop a model-based control strategy, namely, the Model Predictive Control (MPC) method, for robot position control using artificial neural networks. MPC is primarily developed for process control. Therefore its application in robot control has less been reported. In addition, conventional MPC uses linear model of the system for prediction which leads to inaccuracy for highly non-linear systems, such as robot.

In this thesis a simulation model of a modified PUMA robot is constructed. This model is built using both MATLAB/SIMULINK and FORTRAN languages. In this model, the full robot dynamics is used together with the realistic factors, such as the actuator effects and the gear backlash, to represent the real system accurately. All simulations throughout this thesis are carried out on this model.

A model predictive control strategy for robot trajectory tracking is also introduced in this thesis. The feasibility of the proposed MPC control method is studied based on a perfect prediction model, a model with uncertainties, and when the frequency band of the MPC controller is limited.

Furthermore, a new method of using neural networks for robot dynamics modelling is introduced. This method is developed on the basis of a numerical differential technique that eliminates the explicit requirement of robot joint accelerations. Therefore, this method can be easily implemented on physical systems. As the measurements of the robot joint positions, velocities, and torques collected from operating the robot can be used to train the neural network, a more accurate dynamic model can be obtained.

Finally the MPC control method and the neural network model are combined together to form a neural network based MPC controller. The validity of this method is verified by using simulation on the simulated robot system.
Statement of Originality

The work contained in this thesis is performed by myself. No part of this work has been submitted for the award of a higher degree or diploma unless otherwise indicated.

Zhouping Wei
Acknowledgments

I would like to express my sincere thanks to my supervisor Dr. Gu Fang for his patient supervision, concrete direction and instructive suggestion. His enthusiasm, understanding and cheerful manner have made this work possible.

Many thanks are also given to Professor Bryan Roberts and Dr. John Gal for their care, support and encouragement.

Last I am grateful to my family, friends and fellow postgraduate students in this school for their understanding, help and friendship throughout this study.
Contents

Abstract ......................................................................................................................... ii
Statement of Originality ................................................................................................. iii
Acknowledgements ........................................................................................................ iv
Contents .......................................................................................................................... v
Acronyms ....................................................................................................................... vii
List of Figures ................................................................................................................ viii

Chapter 1 Introduction ................................................................................................. 1
  1.1 General Background ............................................................................................... 1
  1.2 Model Predictive Control ....................................................................................... 3
  1.3 Non-linear System Modelling and prediction using Neural Networks .................... 4
  1.4 Neuro-MPC Controllers for MIMO Non-linear Systems ........................................ 7
  1.5 The Aim and Research Methods of the Thesis ...................................................... 9

Chapter 2 Simulation of a Robot System ................................................................. 11
  2.1 Introduction ........................................................................................................... 11
  2.2 The Simulation Software ..................................................................................... 12
  2.3 PUMA Robot ......................................................................................................... 12
  2.4 PUMA Robot Arm Dynamics ................................................................................ 15
  2.5 Actuator Model ..................................................................................................... 17
  2.6 Friction Model ....................................................................................................... 22
  2.7 Gear Backlash ........................................................................................................ 25
  2.8 The Complete System ........................................................................................... 27
  2.9 Simulated Open-loop Characteristics of the PUMA Robot ..................................... 31
  2.10 Summary ............................................................................................................. 36

Chapter 3 Model Predictive Control of Robots ..................................................... 37
  3.1 Introduction ........................................................................................................... 37
  3.2 The Concept of the MPC ...................................................................................... 38
  3.3 A Model Predictive Controller for Robot Position Control ..................................... 40
  3.4 The Selection of the Optimisation Method ............................................................ 42
  3.5 Validity of MPC Controller for Robot Control ..................................................... 43
  3.6 Summary .............................................................................................................. 65

Chapter 4 Neural Network Predictor for Robot ....................................................... 73
  4.1 Introduction ........................................................................................................... 73
  4.2 An overview of the Artificial Neural Networks .................................................... 74
  4.3 Training of a Neural Networks ............................................................................. 78
  4.4 Problem Formulation and Assumption .................................................................. 80
4.5 Methods of Modelling Robot dynamics .............................................. 85
4.6 Simulation Results ................................................................. 87
  4.6.1 Neural network training ....................................................... 87
  4.6.2 Simulation results ............................................................. 90
4.7 Summary ................................................................................. 99

Chapter 5 Neuro-MPC Control of a Robot Manipulator
.................................................................101

  5.1 Fortran Model of the Simulated Robot System ................................. 101
  5.2 Training of the NN Prediction Model .......................................... 103
  5.3 Simulation Results of the MPC Controller based on an FNN Prediction Model ......................................................... 109
  5.4 Summary ................................................................................. 115

Chapter 6 Conclusions and Recommendations ............ 116

  6.1 Summary and Conclusions ........................................................... 116
  6.2 Recommendations .................................................................... 117

References ................................................................. 119

Appendices

A.1 Parameters for PUMA robot dynamic equation ............................... 124
A.2 Calculation of the inertia term ...................................................... 126
A.3 Calculation of the centrifugal and coriolis parameters ....................... 128
A.4 Calculation of the gravity term ..................................................... 132
A.5 Calculation of the PUMA robot dynamics ..................................... 133

B Calculation of the friction forces ..................................................... 135

C Calculation of gear backlash .......................................................... 136

D.1 Neural predictor training program .................................................. 137
D.2 Neural predictor testing program .................................................. 138
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>BP</td>
<td>Back-Propagation</td>
</tr>
<tr>
<td>BPTT</td>
<td>Back Propagation Through Time</td>
</tr>
<tr>
<td>CMAC</td>
<td>Cerebellar Model Articulation Control</td>
</tr>
<tr>
<td>DMC</td>
<td>Dynamic Matrix Control</td>
</tr>
<tr>
<td>dof</td>
<td>degree of freedom</td>
</tr>
<tr>
<td>FNN</td>
<td>Feedforward Neural Network</td>
</tr>
<tr>
<td>GDR</td>
<td>Generalised Delta Rule</td>
</tr>
<tr>
<td>GMDH</td>
<td>Group Method of Data Handling</td>
</tr>
<tr>
<td>LVQ</td>
<td>Learning Vector Quantisation</td>
</tr>
<tr>
<td>MAC</td>
<td>Model Algorithmic Control</td>
</tr>
<tr>
<td>MBC</td>
<td>Model Based Control</td>
</tr>
<tr>
<td>MPC</td>
<td>Model Predictive Control</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-layer Perceptron</td>
</tr>
<tr>
<td>MIMO</td>
<td>Multi Input Multi Output</td>
</tr>
<tr>
<td>NARX</td>
<td>Non-linear AutoRegressive with eXogenous input</td>
</tr>
<tr>
<td>NARMAX</td>
<td>Non-linear AutoRegressive Moving Average with eXogenous input</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>NNMPC</td>
<td>Model Predictive Control based on a Neural Network Predictor</td>
</tr>
<tr>
<td>ODE</td>
<td>Ordinary Differential Equation</td>
</tr>
<tr>
<td>PRPE</td>
<td>Parallel Recursive Prediction Error</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>SISO</td>
<td>Single Input Single Output</td>
</tr>
</tbody>
</table>
# List of Figures

| Figure 1.1 | Basic structure of MPC | 3 |
| Figure 1.2 | Serial or parallel identification | 8 |
| Figure 2.1 | Illustration of PUMA robot | 13 |
| Figure 2.2 | Equivalent circuit of an armature-controlled DC motor | 18 |
| Figure 2.3 | Analysis of the gear train | 19 |
| Figure 2.4 | Block diagram of a robot arm actuator | 21 |
| Figure 2.5 | Striebeck Friction-Velocity Curve | 22 |
| Figure 2.6 | Friction model in a robot arm joint | 24 |
| Figure 2.7 | A backlash non-linearity | 25 |
| Figure 2.8 | PUMA simulation model — general diagram | 28 |
| Figure 2.9 | A Simulink model of the PUMA robot system | 29 |
| Figure 2.10 | The Simulink model of the actuator | 30 |
| Figure 2.11 | Case1: the step response of the simulated robot system | 32 |
| Figure 2.12 | Case2: the sinusoidal response of the simulated robot system | 33 |
| Figure 2.13 | Case3: the zero input response of the simulated robot system | 34 |
| Figure 2.14 | Case4: the sinusoidal response of the simulated robot system | 35 |
| Figure 3.1  | The MPC Strategy | 38 |
| Figure 3.2  | Block diagram of model predictive controller | 41 |
| Figure 3.3  | Desired trajectories — model predictive control based on a perfect model for case 3.1 | 46 |
| Figure 3.4  | Tracking errors — model predictive control based on a perfect model for case 3.1 | 47 |
| Figure 3.5  | Desired trajectories — model predictive control based on a perfect model for case 3.2 | 48 |
| Figure 3.6  | Tracking errors — model predictive control based on a perfect model for case 3.2 | 49 |
Figure 4.4  The serial identification structure 81
Figure 4.5  Prediction architecture of parallel mode 1 83
Figure 4.6  Identification and prediction architecture of parallel mode 2 84
Figure 4.7  Method of modelling robot dynamics proposed by Fang 86
Figure 4.8  The testing results obtained from MLP robot dynamic identifier 91
— case 4.1
Figure 4.9  The testing results obtained from MLP robot dynamic identifier 92
— case 4.2
Figure 4.10  Prediction error comparison of the response of the NN identifier and the nominal model — case 4.1 93
Figure 4.11  Prediction error comparison of the response of the NN identifier and the nominal model — case 4.2 94
Figure 4.12  3 step ahead prediction — case 4.3 95
Figure 4.13  5 step ahead prediction — case 4.3 96
Figure 4.14  3 step ahead prediction — case 4.4 97
Figure 4.14  5 step ahead prediction — case 4.4 98

Figure 5.1  Block diagram of Fortran model of the simulated robot system 102
Figure 5.2  Case5.1 The step response of the Fortran model of the simulated robot system 104
Figure 5.3  Case5.2 The sinusoidal response of the Fortran model of the simulated robot system 105
Figure 5.4  Errors between the Fortran model and SIMULINK model case 5.1 106
Figure 5.5  Errors between the Fortran model and SIMULINK model case 5.2 107
Figure 5.6  Tracking errors — model predictive control based on a neural network model for case 5.3 110
Figure 5.7  Desired trajectories — case 5.4 111
Figure 5.8  Tracking errors — model predictive control based on a neural network model for case 5.4 112
Figure 5.9  Desired trajectories — case 5.5  

Figure 5.10  Tracking errors — model predictive control based on a neural network model for case 5.5