MODELLING OF ROBOTIC MANIPULATORS

A DISSERTATION

SUBMITTED TO THE FACULTY OF ENGINEERING

OF GLASGOW UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Mohamed Abderrahim

December 1996

Dedication

This thesis is dedicated to my parents, Omelkheir and Ahmed Abderrahim, for their unfailing support throughout my life and encouragement to seek a higher education.

It is also dedicated to Silvia, my wife, for her encouragement and standing by me.

Abstract

The accuracy and performance of robotic manipulators are crucial to their commercial viability and widespread use in industry. Nowadays robotic manipulators are required to achieve more accurate positioning, high speeds and have the ability to interact with their environment. This increases the range of tasks for which they could be suitable.

The speed and accuracy of a manipulator are determined by the knowledge of its dynamic and kinematic characteristics and the capability of its control system. Improving the accuracy can be done through improving the controller by using accurate information about the dynamics and kinematics. Therefore, generating a model is the first step for the operation.

This thesis explores the different aspects of robotic manipulator modelling and covers both the dynamic and the kinematic issues for the purpose of improving the overall manipulator accuracy. It is shown that the modelling should not stop at producing the model, but rather the model should be validated. The thesis presents a description of the modelling process and examines the three most important formulations for dynamic modelling. A comparison of their performance and ease of use is made, both for manual and computer assisted implementation. Three commercial computer modelling packages are also described and compared with regard to their performance and ease of use for robotic manipulator modelling. It is shown that some software development is required to make the packages easy to use for manipulator specific modelling. As

part of this work, one such development was a programme written as a back end to AUTOLEV. This combination provides a powerful tool for dynamic modelling and simulation of manipulators. A more integrated computer aided engineering approach is also discussed through modelling a large industrial manipulator using a geometric modelling package along with another dynamic modelling and simulation program. This approach is very efficient in providing useful information which is difficult to otherwise obtain from direct measurements.

The thesis emphasises validation as part of the modelling process. A model does not have to be an exact mathematical description of the manipulator, inclusive of all characteristics, but rather a valid description for the intended use. It is shown that a manipulator model can be split into several joint models and validation performed on each using a parameter estimation technique. It is also shown that friction parameter tuning produces acceptable parameter values for a valid model of a Puma 560 manipulator.

As a result of this work it has been established that dynamic modelling and analysis do not solve all manipulator positioning deficiencies. It is necessary to perform kinematic modelling and kinematic model validation to ensure accurate positioning of the manipulator's end-effector. The thesis introduces a new methodology based on Stone's method to improve the kinematic model. The method is tested both experimentally and in simulation and yields good positioning improvement.

The work is extended to produce a specific dynamic model of a manipulator operating underwater. The hydrodynamic effects are evaluated through a series of simulations. The information gained provides a better understanding and may aid in designing a suitable controller for such manipulators.

Acknowledgements

I wish to express my thanks and gratitude to my supervisor, Dr. Arthur Whittaker for his support, skilful guidance and encouragement throughout the course of my Ph.D.

I would also like to thank the members of the robotics group for their constructive discussions and suggestions during our meetings. In particular, I thank Professor Peter Gawthrop and Dr. John Howell for their help and advise, Dr. David Roberts for the numerous discussions in the field robotics and Dr. Peter Larcomb for his advise. My thanks are also extended to my fellow members of the Control Group of Glasgow University for their friendship and help in numerous problems. I especially, thank Mrs Yasmine Mather and Dr. Donald Ballance for solving the numerous computing problems and Dr. Eduardo Liceaga-Castro for his help with the underwater manipulator model.

My thanks are extended to Professor G. Parker from Surrey University for allowing us use his laser tacking system.

Finally, I would like to thank the members of staff of Glasgow University Mechanical Engineering Department for their friendship and the Algerian Ministry of Higher Education for providing the financial support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The history of the human fascination with physically constructed life forms, which includes robots and automatic machines, is long. An historic overview of the fascination with regard to the robot evolution and its effects upon human life was presented by Mowforth [3]. He also discussed the growing expectations from the use of robots in contemporary industry.

The industrial robot was pioneered by George Duvall and Joe Englberger, who brought the first unimate to market in 1957 [3]. It was used to remove parts from die casting machines. In the late 1970's these devices became profitable in paint spraying and spot welding. Unfortunately, the new robot workforce did not live up to expectations and are still used throughout the industry in simple, repetitive tasks of the pick and place mode [4]. To attain commercial viability and more efficient industrial use, robotic manipulators must be developed to achieve more accurate positioning, high speeds and have the ability to interact with their environment.

To define a robotic manipulator a number of questions concerning the functional concepts, should be answered:

What is, and what is not, a robot ?

How is a robot constructed ?

How does it operate ?

Despite this, only a few manufacturers and associations would agree on one single definition [5]. Since there is no standard definition, it would be helpful to consider some of the attempts to provide one.

- The British Robot Association (BRA) emphasises the four degrees of freedom as one of the qualifications defining a robot as :

  A reprogrammable device with a minimum of four degrees of freedom designed to both manipulate and transport parts, tools or specialised manufacturing implements through variable programmed motions for the performance of the specific manufacturing task.

- The Robotics Institute of America (RIA) defines the robot as a reprogrammable multi-functional manipulator designed to move material, parts, tools or specialised devices through variable programmed motions for the performance of a variety of tasks.

  The RIA emphasises the programmable facilities, and its definition is widely accepted for an industrial robot.

- The Japan Industrial Robot Association (JIRA) and the Japanese Industrial Standards Committee define the robots at various levels as:

  manipulator : a machine which has functions similar to those of the human upper limbs, and moves the objects spatially, from one location to the other

  .. *playback robot*: a manipulator which is able to perform an operation by reading off the memorised information for an operating sequence, including positions and the like, which it learned by being taken manually through the routine beforehand ...

  and base higher levels definitions upon the first one.

Generally, a robotic manipulator is thought of as a programmable machine constructed by a chain of interconnected links by means of rotary or sliding joints.

where each joint can be actuated independently by its own actuator to allow the end effector to follow a defined trajectory in order to perform a defined task.

To attain the desired features a manipulator should be equipped with good sensors, a good control system with adequate computing power and light weight links. This is correct when developing new manipulators. However, there is still the need to improve the qualities of the existing ones. This can be done through implementing more sophisticated control algorithms or adjusting the existing ones.

Whether for developing new manipulators or improving existing ones, a good model of the system is required. An accurate mathematical model would be implemented in new control schemes such as computed torque control and model based control [4] [6]. A fully inclusive dynamic model could be complex and computationally expensive if implemented in real time applications. Generating such models for robotic manipulators is difficult and error prone, despite the existence of adequate formulations such as the Newton-Euler and Lagrange formulations. Computer automatic model generation using multibody dynamic systems modelling packages is obviously desirable. These do not support all the modelling activities required for robotic manipulators and may need further development, as shown in chapter 3. A good model is also required for computer simulation to predict the behaviour of a particular manipulator under particular conditions of actuation. The simulation exercise aids the analysis of the manipulator design and performance evaluation, as well as the evaluation of controller design.

Although the dynamic model is critical for the above activities several simplifying assumptions and approximations are considered during its development [7]. Therefore, the model does not have to be inclusive of all characteristics to be a valid description of the manipulator dynamics, but it is valid if it is eventually judged fit for the purpose for which it was intended [8] [9]. A validation process must therefore follow the model generation to establish if the model is

a valid representation of the real system. This issue is elaborated in chapter 5. where it is concluded that a good dynamic model and its analysis do not solve all manipulator end-effector positioning performance even when joint control is perfectly achieved. The end-effector positioning through off-line programming relies fundamentally on the manipulator internal functional relationship between the end-effector and the base. This relationship, *the kinematic model.* is unique for each manipulator and should be established accurately after manufacturing. as discussed by Roth *et al* [10] A chapter of the thesis is dedicated to improving manipulator end-effector positioning through improving its kinematic model. A new methodology based on Stone's method [11] is proposed, and its success is illustrated with both experimental and simulation results.

Manipulators operating underwater are not different kinematically from those operating in normal conditions, however their dynamics are severely affected by the hydrodynamic effects. A better understanding of their dynamics is required since there is an increased need for their use in underwater activities related to sea bed exploration, rescue and similar activities [12]. Although generic models of underwater manipulators have been proposed, no study of the particular hydrodynamic effects on a specific manipulator model has been reported. This issue is addressed in a separate chapter where hydrodynamic effects are explicitly calculated, extending previous works which were limited to generic models. The chapter provides greater understanding of the dynamics of underwater manipulators, which are part of the robotic manipulators family, and also aids designing suitable controllers according to the nature of their dynamics.

The aspects of robotic manipulators modelling studied in this work can be summarised as follows:

- Examine the aspects and requirements of dynamic modelling using the existing formulations, and comparing their efficiency.

- Study samples from the existing computer modelling packages with regard

to their use for robotic manipulator modelling. Establish a comparison of their ease of use and performance. Choose a computer package to perform modelling and simulation of a real industrial manipulator.

- Perform full dynamic modelling of an industrial manipulator and examine its validity considering that some approximations are included. Perform model validation based on measurements taken from the manipulator using a locally constructed measurement system. Raise the need for kinematic modelling and analysis.

- Perform kinematic modelling and validation to improve the positioning of the end-effector of the manipulator since it is part of the overall positioning accuracy and performance of the robot.

- Generate a specific model of a two links manipulator operating underwater and study the hydrodynamic effects through a series of simulations.

While each chapter deals with one major aspect of modelling, all aspects presented should be considered when designing and developing new robotic manipulators as well as when analysing existing ones for the purpose of improving their accuracy and performance.

1.1 Literature Survey

Dynamic models are useful for computer simulation of the robot arm motion, the design of suitable control and evaluation of the kinematic design, analysis of manipulator performance, evaluation of controller design, and form a major part of some controllers' algorithms. Kinematic manipulator models, on the other hand, constitute the essential part of the manipulator kinematic controllers that ensure the positioning of the end-effectors.

Formulation of the dynamic equations has been an active research topic concerning general mechanisms, especially robot manipulators. The two most commonly used formulations are Lagrange-Euler (L-E) and Newton-Euler (N-E) methods. Other methods such as Recursive Lagrangian and Generalised D'Alembert are cited in [6] as having been used.

Craig [4] presented a detailed analysis of the use of the N-E recursive formulation to generate rigid manipulators models, and also of the Lagrangian formulation of manipulator dynamics. Both methods were also described by Fu *et al* [13]. Kane *et al* [14] compared different existing methods to formulate the equations of motion for spacecraft, and presented a new, efficient method, known nowadays as *Kane's Formulation*, which has been used for manipulator dynamics. The Newton-Euler formulation has been regarded as useful in providing insight into the representation of manipulator dynamics, whilst the Lagrange formulation has been shown to be computationally more efficient [15].

Regardless of the method used, the symbolic expansion of the robot arm equations of motion by hand is a difficult and tedious task, which is both time consuming and error-prone. This created a great demand for the automatic generation of equations of motion which resulted in many commercial computer modelling packages several of which are listed in [16], which lists the modelling packages in separate chapters without consistently comparing them.

Dynamic model validation has long been recognised as an integral part of model development in textbooks such as [17] [18]. Although formal validation processes are emphasised in theory, Murray-Smith [19] argues that most application papers pass over questions of validation superficially. While validation appears to be of central importance in the past, mainly for a few safety-critical application, it is noted that it has extended in recent years to cover other applications such as robotics [20] [21].

The accuracy of the robotic manipulator kinematic model is crucial for

accurate end-effector positioning. There has been a considerable volume of work in the last decade on the subject of improving the positioning accuracy of manipulators, much of which was reviewed by Roth *et al* [10] mainly under titles with key words such as, *calibration* and *accuracy improvement*. The most commonly used kinematic model is the one using Denavit-Hartenberg representation [22]. The unique model for each manipulator should be established after it is manufactured. Despite the number of publications on the subject, real measurements were reported only in some works, especially the latest papers, such as [23] [24] [25] [26] [27] [28] and [29]. A variety of measurement techniques were used, including visual and automatic theodolites [23] [28] [29], acoustic sensors [24], laser tracking devices [25] and coordinate measuring machines [30]. Some other reported work was based on computer simulations. It is indicated that the method described by Driel *et al* [29] results in positioning accuracy of an order comparable to that of the repeatability of the given manipulator.

The use of underwater-robotic vehicles is nowadays common in maritime activities such as exploration, rescue and oil exploitation. The vehicles are usually equipped with manipulators [12], and a study of their dynamics should be done. Although modelling of vehicles themselves has been the topic of several papers such as [31] [32] [33], the dynamics of the manipulators were not considered specifically. Ioi *et al* [34] have proposed a generic model of an underwater manipulator and included added mass, drag and lift caused by the hydrodynamic effects. Nevertheless, no study of the hydrodynamic effects on a specific manipulator model seems to have been reported.

More references are cited in the appropriate places in the body of the thesis.

1.2 Thesis Outline

The thesis is organised as follows:

Chapter 2 outlines the basic steps and requirements for the modelling process of robotic manipulators. Several mathematical modelling formulations are presented and discussed with regard to their efficiency and ease of computer implementation for automatic model generation. The need for automatic computer modelling is raised.

In chapter 3 existing general multibody systems' dynamic modelling packages are highlighted and three of them examined closely. They are compared with regard to their efficiency and ease of use in robotic manipulator modelling. A back end computer program is written to complement one of the commercial packages where the combination is described as a powerful modelling tool and used throughout the work. A computer aided design and simulation approach is also presented in the form of an example using a commercial industrial manipulator, to show the benefits of the method in developing manipulators and providing essential information for dynamic modelling.

Chapter 4 describes the development and construction of an instrumentation system based on commercially available hardware and a personal computer. The information provided by the system is used in the following two chapters in the model validation process. Chapter 4 also describes other potential uses for the developed system.

Chapter 5 emphasises model validation as an integral part of the modelling process. The chapter shows how the dynamic model validation can be split into individual joint model validation, and how the relevant dynamic parameters are estimated using simple methods. Both experimental and simulation based results are presented. The chapter concludes that the generated dynamic model of a Puma 560 manipulator is valid for positioning purposes and the poor end-effector positioning is due to kinematic model deficiencies.

The sources of the kinematic model deficiencies are discussed in chapter 6, and a new methodology based on *Stone's Method* is introduced. The chapter

also shows the improvement through using experimental data as well as through simulations.

An explicit model of a two links underwater manipulator is developed in chapter 7. The chapter highlights the difficulties that arise when generating the dynamic model of such manipulators due to the complex terms caused by the hydrodynamic effects despite several simplifying assumptions. The chapter also states that the kinematic model is no different from that of manipulators operating in normal conditions.

Chapter 8 concludes the thesis and examines possible avenues for further research as a follow up to the thesis.

Chapter 2

Dynamic Modelling of Manipulators

2.1 Introduction

In practical engineering control problems, analysis starts with modelling of the robot arm or the physical system under study. The objective of the modelling is to establish the mathematical equations, *model*, as a set of analytical relations describing the dynamic behaviour of the robot arm. The modelling process depends on the characteristics of the arm to be studied and the physical details to be included. This is why dynamic modelling, according to Gawthrop [35] and Brussel *et al* [7], incorporates several stages which can be summarised in :

- physical modelling
- model simplification (schematic model)
- mathematical modelling
- mathematical model analysis
- model validation

In the above stages the robot arm. or generally a dynamic system. would undergo several transformation and simplifications, for instance in the first stage an imaginary model of the robot arm is built essentially like the real system or from the design requirements of a robot arm. At this stage many decisions are to be made concerning aspects such as friction, compliances of links and joints. linearity, noise, etc. A summery of the effects of some approximations on the mathematical model are shown in table 2.1 [7].

Table 2.1: Effects of approximations on the mathematical model

approximations	mathematical model simplification
1- Neglect small effects	Reduces number and complexity of differential equations.
2- Assume environment independent of system motions	Same as 1.
3- Replace distributed characteristics with appropriate lumped elements	Leads to ordinary, rather than partial, differential equations.
4- Assume linear relationships	Makes equations linear. allows superposition of solutions.
5- Assume constant parameters	Leads to constant coefficient in differential equations.

Model simplification consists of establishing links connectivity and the nature of their relative motion in a schematic form to give more insight to help generate the right equations of motion. In the mathematical modelling process, the first point to consider is to select the state variables, which describe essentially, the storage of energy and mass in the system. The state variables of a robot arm are the positions and velocities of its links when the rigid mechanical system only is considered. Next step is the application of balance equation for force, moment. mass, energy or writing system elements relations which describe relative motion of links. Mathematical model analysis (Simulation) is the next step. The obtained equations of motion are used to imitate the behaviour of the real system under a stimuli representing the action of a real control system or force/torque applied

to the system. The behaviour analysis at this stage is useful for the design of suitable control system and for the evaluation of the structure and parameters of the arm under consideration. Model validation is a necessary step at this stage. The obtained equations represent the dynamic model of the real system under study after several approximations. Therefore, it must be validated in order to obtain enough confidence that it adequately represents the arm dynamic behaviour under a set of conditions determined by the purpose of the modelling. This means that the validation process involves comparison of the mathematical model solutions(simulation) with the real arm behaviour subjected to the same stimuli. Usually, a model is not determined as absolutely valid, but rather, evaluation and model tuning are conducted until sufficient confidence is established within the context of intended uses of the robot arm.

A brief introduction of most important mathematical modelling formulations is given in the next section whereas section 2.3 contains a detailed description of each. In particular, the use of *Kane's formulation* is explained with the help of an example. Section 2.4 discusses the need for automatic mathematical modelling and section 2.5 concludes the chapter.

2.2 Dynamic Equations of Rigid Manipulators

The formulation of the equations of motion is always an important stage of multi-body mechanisms and robot manipulator design, and performance analysis. With regard to robotics, such equations are useful for computer simulation of manipulator arm motion, the design of a suitable control system, and the evaluation of the structure of the arm.

Formulation of the dynamic equations has been an active research topic concerning general mechanisms, especially robot manipulators and spacecrafts. Craig [4] presented a detailed analysis of the use of the *Newton-Euler* recursive

formulation to generate rigid manipulators models. and also the Lagrangian formulation of manipulator dynamics, also presented by Nicosia *et al* [36]. Kane & Levinson [14] compared different existing methods to formulate the equations of motion for spacecraft, and presented a new, and more efficient, method, known nowadays as *Kane's Formulation* which has been used for manipulator dynamics.

The Newton-Euler formulation has been regarded as useful in providing insight into the representation of manipulators dynamics, whilst the Lagrange formulation has been shown computationally more efficient. Kane's formulation is much more efficient than the previous two, and is also simpler. These –efficiency and simplicity– are the principal criteria by which a method would be assessed, and they become more important when dealing with more complex systems.

2.3 Dynamics

Dynamics of manipulators is a special case of dynamics of mechanisms, and is a field on which many books have been written. However, the work reported here is an attempt to analyse and use certain formulations of the dynamics problem which seem particularly well suited to application to manipulators. There are two major problems related to the dynamics of a manipulator that should be solved. In the first, a required trajectory is given in terms of Θ, $\dot{\Theta}$ and $\ddot{\Theta}$ and the vector of joint torques, τ, is to be found. This formulation is useful for the problem of controlling manipulators. The second problem is the opposite task to the first, which involves calculating how the mechanism will move under application of a set of joint torques. This is useful for manipulator simulation and some control schemes such as computed torque control.

2.3.1 Newton-Euler Recursive Equation

In this case the problem of computing the torques in terms of a given trajectory is considered, assuming that the position, the velocity and the acceleration of the joints $(\Theta, \dot{\Theta}, \ddot{\Theta})$, are known. With this knowledge, and the knowledge of the mass distribution of each link of the robot, the formulation of the equations of motion can be done in two main stages. In the first, we compute the rotational velocity, and linear and rotational acceleration of the centre of mass of each link of the manipulator, in order to compute the inertial forces and torques acting on the links at any given instant. This is done in an *outward* iterative manner, starting with link 1 and moving through to link n. In the second stage, an opposite procedure is applied. In an *inward* iterative way the joint forces and torques are calculated by writing force-balance and moment-balance equations based on a free body diagram for each link, starting at link n and moving, link by link, to link 1.

The outward iterations compute velocities and accelerations (kinematics elements).

If link i is rotational, then

$$\omega_i^i = R_{i-1}^i(\omega_{i-1}^{i-1} + Z_0\dot{q}_i) \tag{2.1}$$

$$\dot{\omega}_i^i = R_{i-1}^i[\dot{\omega}_{i-1}^{i-1} + Z_0\ddot{q} + \omega_{i-1}^{i-1} \times Z_0\dot{q}_i] \tag{2.2}$$

$$\dot{v}_i^i = \dot{\omega}_i^i \times p_i^i + \omega_i^i \times (\omega_i^i \times p_i^i) + R_{i-1}^i \dot{v}_{i-1}^{i-1} \tag{2.3}$$

$$a_i^i = \dot{\omega}_i^i \times s_i^i + \omega_i^i \times (\omega_i^i \times s_i^i) + \dot{v}_i^i. \tag{2.4}$$

If link i is translational, then

$$\omega_i^i = R_{i-1}^i \omega_{i-1}^{i-1} \tag{2.5}$$

$$\dot{\omega}_i^i = R_{i-1}^i \dot{\omega}_{i-1}^{i-1} \tag{2.6}$$

$$\dot{v}_i^i = R_{i-1}^i (Z_0 \ddot{q} + \dot{v}_{i-1}^{i-1}) + \dot{\omega}_i^i p_i^i + 2\omega_i^i \times (R_{i-1}^i Z_0 \dot{q}_i) + \omega_i^i \times (\omega_i^i \times p_i^i) \tag{2.7}$$

$$a_i^i = \dot{\omega}_i^i \times s_i^i + \omega_i^i \times (\omega_i^i \times s_i^i) + \dot{v}_i^i \tag{2.8}$$

where

n is the manipulator's number of degrees of freedom,

q is $n \times 1$ vector of joint variable positions,

$\dot{q}$ is $n \times 1$ vector of joint variable velocities,

$\ddot{q}$ is $n \times 1$ vector of joint variable accelerations,

R_j^i is 3×3 transformation matrix for jth link coordinates into ith link coordinate reference frame,

ω_i^i is 3×1 vector of the ith link coordinates angular velocity in the ith reference frame,

$\dot{\omega}_i^i$ is 3×1 vector of the ith link coordinates angular acceleration,

v_i^i is 3×1 vector of the ith link linear velocity,

$\dot{v}_i^i$ is 3×1 vector of the ith link coordinates linear acceleration,

a_i^i is 3×1 vector of the ith link mass centre linear acceleration,

s_i^i is the position vector of the ith link mass centre in terms of the reference frame (x_i, y_i, z_i),

$Z_0 = (0, 0, 1)^T$.

Backward equations $(i = n, n-1, ..., 1)$ compute the joint torques or forces corresponding to link motions (the dynamics elements).

$$F_i^i = m_i a_i^i \tag{2.9}$$

$$f_i^i = R_{i+1}^i f_{i+1}^{i+1} + F_i^i \tag{2.10}$$

$$n_i^i = R_{i+1}^i(n_{i+1}^{i+1} + p_i^{i+1^*} \times f_{i+1}^{i+1}) + (p_i^{i^*} + s_i^i) \times F_i^i + I_i^i\omega_i^i + \omega_i^i \times (I_i^i\omega_i^i) \quad (2.11)$$

If link i is rotational, then

$$\tau = (n_i^i)^T(R_{i-1}^i Z_0) + b_i\dot{q}_i \quad (2.12)$$

whilst if link i is translational,

$$\tau = (f_i^i)^T(R_{i-1}^i Z_0) + b_i\dot{q}_i \quad (2.13)$$

where:

f_i^i is 3×1 force vector exerted on link i by link $i-1$.

n_i^i is 3×1 torque vector exerted on link i by link $i-1$.

F_i^i is 3×1 vector of the total force exerted on link i.

N_i^i is 3×1 vector of the total torques exerted on link i.

m_i is the ith link total mass.

$p_i^{i^*}$ is the vector representing the origin of the ith coordinate system in terms of the coordinate system $i-1$.

I_i^i is the ith link inertia matrix about its mass centre.

Gravity effects have not been included in the formulation so far. This can be done simply by setting $\dot{v}_0^0 = G$, where G is the gravity vector. This is equivalent to saying that the base of the manipulator is accelerating upwards with an acceleration of $1g$. This '*fictious*' acceleration causes exactly the same effect on the links as real effect.

The robot arm set of the equations of motion are often represented in a single equation that shows some of the structure of the individual equations, but hides the details. After the equations of motion are evaluated symbolically for any rigid manipulator, they yield the following dynamic equation:

$$\tau = M(q)\ddot{q} + Q(q, \dot{q}) \quad (2.14)$$

where $M(q)$ is the $(n \times n)$ symmetric mass matrix of the manipulator and $Q(q, \dot{q})$ is an $(n \times 1)$ vector containing centrifugal, coriolis and gravity terms.

2.3.2 Lagrange-Euler Formulation

The Lagrangian formulation is an energy-based approach to dynamics. The equations of motion can be obtained by direct application of Lagrange-Euler formulation for non-conservative systems. In the case of manipulators we make use of the Denavit-Hartenberg [22] matrix representation to describe the displacement between the neighbouring link coordinate frames, to obtain the kinematics of each link, with the Lagrange dynamics technique used to derive the actual manipulator equations of motion. This method results in a convenient and compact algorithmic description of the equation of motion that facilitate both analysis and computer implementation.

The Lagrangian method provides a means of deriving the equations of motion from a scalar function called the **Lagrangian**, which is given by

$$\mathcal{L}(\Theta, \dot{\Theta}) = K(\Theta, \dot{\Theta}) - u(\Theta) \tag{2.15}$$

where $K(\Theta, \dot{\Theta})$ is the kinetic energy of the manipulator and $u(\Theta)$ is the potential energy. The vectors Θ and $\dot{\Theta}$ are position and velocity vectors.

The equations of motion for the manipulator are then given by

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\Theta}}\right) - \frac{\partial \mathcal{L}}{\partial \Theta} = \tau \tag{2.16}$$

Where τ is a vector of generalised forces or torques applied on the manipulator links.

One is required to choose the desired set of generalised coordinates to describe the system motion, (for example, relative or absolute angular displacements). They are used to describe the position and orientation of different manipulator links with respect to a reference coordinate frame, the so called *base* in this

case. The generalised coordinates for a manipulator with rotary joints can be chosen conveniently as the relative angles between links. because they are useful for the task of control. Fu *al* [13] give the total kinetic and potential energy of manipulator as

$$K = \sum_{i=1}^{n} \sum_{p=1}^{i} \sum_{r=1}^{i} Kc_{ipr}(q)\dot{q}_p\dot{q}_r \tag{2.17}$$

and

$$u = \sum_{i=1}^{n} -m_i g \overline{r}_i^0 \tag{2.18}$$

where $Kc_{ipr}(q_i)$ is a function of q_i, n is the number of links constituting the manipulator, m_i is the ith link mass and g is the gravity row vector in terms of the base reference frame. The vector $\overline{r}_i^0$ expresses the ith link mass centre from and in the base frame.

After using equation 2.15, 2.16, 2.17 and 2.18 the produced equations of motion of manipulator can be written in the following form:

$$D(q)\ddot{q} + H(q,\dot{q})\dot{q} + C(q,\dot{q}) = \tau \tag{2.19}$$

where $D(q)$ is an $n \times n$ inertial acceleration-related symmetric matrix whose elements are

$$d(i,j) = d(j,i) = \frac{\partial^2 K}{\partial \dot{q}_i \partial \dot{q}_j}. \tag{2.20}$$

$H(q,\dot{q})$ is an $n \times n$ nonlinear coriolis and centrifugal force-related matrix whose elements are

$$h(i,j) = \frac{\partial^2 K}{\partial \dot{q}_i \partial q_j} \neq \frac{\partial^2 K}{\partial q_i \partial \dot{q}_j} = h(j,i) \tag{2.21}$$

and $C(q,\dot{q})$ is an $n \times n$ gravity loading force vector whose elements are

$$c(i) = -\frac{\partial(K-P)}{\partial q_i}. \tag{2.22}$$

In particular the manipulator's total kinetic and potential energies are the sum of the individual links kinetic and potential energies, and are given by

$$K = \sum_{i=1}^{n} k_i; \qquad u = \sum_{i=1}^{n} u_i \tag{2.23}$$

where

$k_i = \frac{1}{2}\int_0^{L_i} \rho_i(\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2)dl$

$u_i = g\int_0^{L_i} \rho_i Z_i dl$

ρ_i = mass per unit length of the ith link

L_i = link i length.

Although this kind of algorithm is set to facilitate the formulation of the equations of motion, it is still difficult and time consuming to perform manually. especially for large systems. Li [15] has coded the above algorithm using the symbolic algebra language REDUCE to automatically generate the equations of motion for rotating manipulators.

2.3.3 Kane's Formulation

Kane's formulation is also known as Jourdain method. It applies to any material system which can be presented in a Newtonian reference frame in terms of generalised coordinates $q_1, ..., q_n$. This method involves two new quantities. namely *partial angular velocities* and *partial velocities.* These quantities are defined as follows:

If $u_1, ..., u_n$, called *generalised speeds*, are introduced as linear combinations of $\dot{q}_1, ..., \dot{q}_n$ by equations of the form

$$u_i = W_{ij}\dot{q}_j + X_i \qquad (i = 1, ..., n) \tag{2.24}$$

where W_{ij} and X_i are functions of $q_1, ..., q_n$ and the time t, and are chosen such that equation 2.24 can be solved uniquely for $\dot{q}_1, ..., \dot{q}_n$, then the angular velocity of any rigid body and the velocity of any point of the system can be expressed uniquely as a linear function of $u_1, ..., u_n$. In such a function. the vector which is the coefficient of u_i is the ith partial angular velocity of the rigid body or the ith partial velocity of the point. To make this task clearer. the example shown in Figure 2.1 is used as a sample of a system containing translational and angular

motion. Its equations of motion will be derived using Kane's formulation.

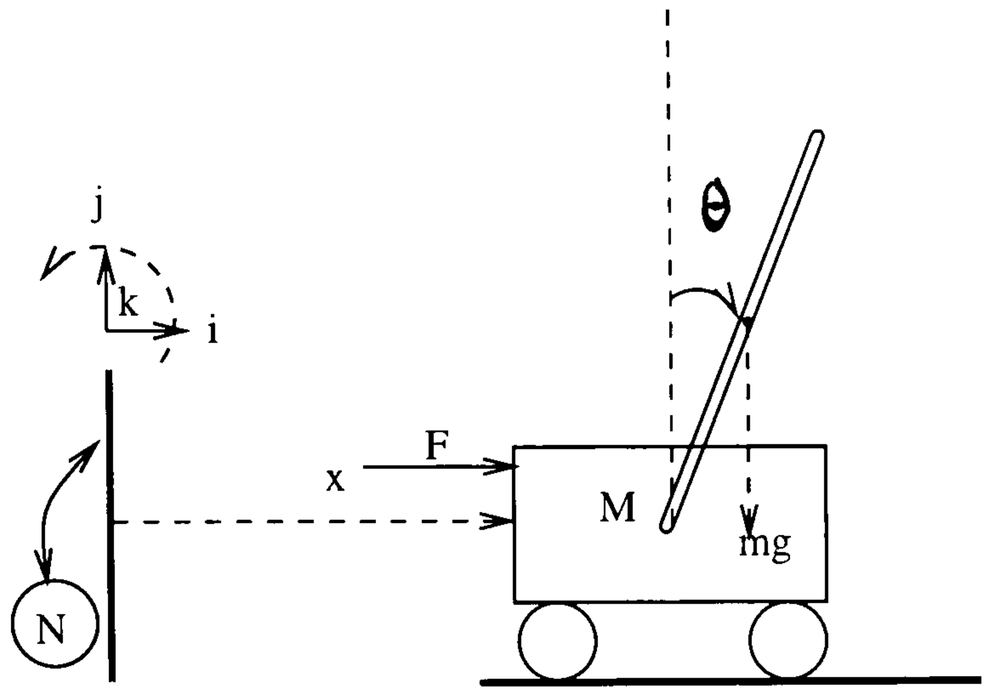


Figure 2.1: A trolley with inverted pendulum

The generalised coordinates in this case are x and θ, and the generalised speeds are u_1 and u_2. Indeed , these are the derivatives of x and θ, respectively.

The positions of the two bodies in the reference frame N are

$$r_1 = x\mathbf{i}, \tag{2.25}$$

$$r_2 = (x + \frac{l}{2}sin\theta)\mathbf{i} + \frac{l}{2}cos\theta\mathbf{j}. \tag{2.26}$$

The velocities in terms of N are

$$v_1 = \dot{r}_1 = \dot{x}\mathbf{i} \tag{2.27}$$

$$v_2 = \dot{r}_2 = (\dot{x} + \frac{l}{2}\dot{\theta}cos\theta)\mathbf{i} - (\frac{l}{2}\dot{\theta}sin\theta)\mathbf{j} \tag{2.28}$$

$$\omega_1 = 0 \tag{2.29}$$

$$\omega_2 = \dot{\theta}\mathbf{k} \tag{2.30}$$

From this set of equations and the definitions of partial velocities, the following are the partial angular velocities and partial velocities for the system of Figure 2.1:

$$\omega_1^1 = 0, \qquad \omega_2^1 = 0, \qquad \omega_1^2 = 0 \qquad \omega_2^2 = \mathbf{k} \tag{2.31}$$

$$v_1^1 = \mathbf{i}, \qquad v_2^1 = 0, \tag{2.32}$$

$$v_1^2 = \mathbf{i}, \qquad v_2^2 = \frac{l}{2}cos\theta\mathbf{i} - \frac{l}{2}sin\theta\mathbf{j} \tag{2.33}$$

To formulate the equations of motion using the partial angular velocities and the partial velocities two other forms of forces and/or torques are needed, namely, generalised active forces K_i and/or τ_i, and generalised inertia forces and/or torques K_i^* and/or τ_i^* $(i = 1, ..., n)$ such that

$$K_i = \sum_{y=1}^{\nu}(v_i^{p_y} \cdot F_y + \omega_i^y \cdot \tau_y) \qquad (i = 1, ..., n) \tag{2.34}$$

$$K_i^* = \sum_{y=1}^{\nu}(v_i^{p_y} \cdot F_y^* + \omega_i^y \cdot \tau_i^*) \qquad (i = 1, ..., n) \tag{2.35}$$

where ν is the number of particles in the system under consideration. The variable $v_i^{p_y}$ is the ith partial velocity of particle p_y. The force F_y is the resultant of all applied forces including gravitational forces acting on p_y whose mass is m_y. Similarly, F_y^* and τ_y^* are the resultants of the inertial forces and torques acting on particle p_y.

Dynamic equations of motion are formulated finally by equating to zero the sum of the generalised inertia active forces and inertia forces:

$$K_i + K_i^* = 0 \qquad (i = 1, ..., n) \tag{2.36}$$

The following is the application of the above on the system of Figure 2.1. The first quantities to be calculated are the active forces which act on the rigid bodies in this case.

$$F_1 = F\mathbf{i} - Mg\mathbf{j} \tag{2.37}$$

$$F_2 = -mg\mathbf{j} \tag{2.38}$$

The inertia forces and torques are given by

$$F_1^* = -M\ddot{x}\mathbf{i} \tag{2.39}$$

$$F_2^* = -m(\ddot{x} + \frac{l}{2}\ddot{\theta}cos\theta - \frac{l}{2}\dot{\theta}^2 sin\theta)\mathbf{i} + m(\frac{l}{2}\ddot{\theta}cos\theta + \frac{l}{2}\dot{\theta}^2 cos\theta)\mathbf{j} \quad (2.40)$$

$$\tau_2^* = -I_{pendulum}\ddot{\omega}_2 = \frac{ml^2}{12}\ddot{\theta}\mathbf{k} \quad (2.41)$$

and the remaining torques are all nil for this case, including the active ones as well.

Let us now calculate the generalised active and inertia forces:

$$K_1 = F \quad (2.42)$$

$$K_2 = \frac{ml}{2}gsin\theta \quad (2.43)$$

$$K_1^* = -(M+m)\ddot{x} - \frac{ml}{2}cos\theta\ddot{\theta} + \frac{ml}{2}sin\theta\dot{\theta}^2 \quad (2.44)$$

$$K_2^* = -\frac{ml}{2}cos\theta\ddot{x} - \frac{ml^2}{3}\ddot{\theta} \quad (2.45)$$

Finally, the two equations of motion of the system shown in Figure 2.1 can be formulated as follow:

$$F - (M+m)\ddot{x} - \frac{ml}{2}cos\theta\ddot{\theta} + \frac{ml}{2}sin\theta\dot{\theta}^2 = 0 \quad (2.46)$$

$$\frac{ml}{2}gsin\theta - \frac{ml}{2}cos\theta\ddot{x} - \frac{ml^2}{3}\ddot{\theta} = 0 \quad (2.47)$$

2.4 Automatic Mathematical modelling

A considerable effort has been spent in providing specific, formulations and methods for generation of equations of motion for dynamic systems, and specifically.

robot arm manipulators, see previous sections. This does not solve the problem of mathematical modelling completely, especially. when large systems and multilink manipulators are considered. Although, the use of the special formulation helps enormously, nevertheless, manual formulation of the model is tedious. time consuming, error-prone and extremely difficult to debug. It is also difficult to adapt to changing requirements and becomes too hard when the modelled system reaches certain complexity. This gives rise to the desire for automatic derivation of the equations using a computer, even for simple manipulator arms. This results in several important advantages. It reduces the time spent in deriving the equations of motion immensely and saves the good engineering talent to be invested in solving other issues related problems, such as control design.

Since good computer programs already exist, such as MACSYMA and REDUCE, to help mathematicians and engineers in performing complex mathematics, one can utilise the algorithms from previous sections in these programmes in order to generate the equations of motion for a given manipulator.

Li [15] has coded the Newton-Euler algorithm in REDUCE -Symbolic algebraic manipulation software- to generate automatically the dynamic equations for rotary manipulators. In the present work this code has been generalised to deal with manipulators with prismatic joints as well, see appendix A.1. It has been tested and generated successfully the correct dynamic equations for several example systems, including that of figure 2.1. The generated equations in symbolic form are easily edited and used in simulation to mimic the dynamics of the system under given conditions. The Lagrange-Euler formulation was also coded by Li [15] here at the Department of Mechanical Engineering using REDUCE to derive automatically the dynamic equations of manipulator in symbolic form.

When comparing the two codes, The Newton-Euler method[1] provides an

[1]It has been brought to attention during the *Viva*, that reference [101] is relevant to this chapter. Walker and Orin [101] discuss the efficiency of the Newton-Euler formulation and present four methods of calculating the forces acting on the manipulator links.

insight into the representation of system dynamics and a step by step check of the correctness of the generation of the equations. This method provides access to individual link forces and acceleration. The Lagrange-Euler algorithm is proved to be more efficient in calculating the equations of motion due to the fact that it need not to evaluate the acceleration and force for each link. Kane's Formulation, is claimed to be much more efficient especially when larger systems are under consideration. It is also simpler and more straightforward to program. It has been used in several powerful multibody systems modelling software such as SD/FAST and smaller ones such as AUTOLEV. Other possible formulations for multibody systems dynamics are order N, $O(N)$, and $O(N^3)$ formulations. Order N, being an option in SD/FAST is advised to be used for larger systems[2].

2.5 Conclusion

A general description of the modelling process of robot manipulator has been presented and led to the introduction of three methods of dynamic equations formulation. A comparison of the efficiency and ease of applicability is also presented and included comparison of programs for automatic derivation, in a symbolic form, of the dynamic equations. These programmes are written in REDUCE and based on the discussed algorithms. Several examples systems from [4] and [37] have been used to check the validity of these programs and produced the correct dynamic equations. Over the last five to ten years several software packages specialising in multi-body systems modelling and simulation have been developed. They are claimed to provide solutions to many engineering problems. Chapter 3 presents a comparison of some commercially available software packages with regard to their use and performance in the field of robotics.

---

[2]Featherstone [102] proves that $O(N)$ formulations are more efficient than $O(N^3)$ only for systems with more than 10 degrees of freedom.

Chapter 3

Existing Modelling Programs Evaluation

3.1 Introduction

From the previous chapter , it is apparent that several methods have been developed and are well established for the purpose of manipulator mathematical model formulation. These formalisms are valid for other multibody mechanisms and general multibody formalisms are applicable to robot arm modelling. The formulation of the mathematical model through the described methods is simple and straightforward, however it becomes complicated and cumbersome when the number of links exceeds two or three. In addition to the time it takes, it becomes error prone due to the large number of operations needed. To avoid all the problems associated with the hand formulation of the model a computer program shall be used to automatically generate the model and cope with the repetitive and tedious operation and hence save the engineering talent to be invested on other issues, such as the design of the suitable control.

Research and development in the field of multibody dynamics over the last two decades resulted in more than 20 computer programs and software packages, most

of which are commercially available [16]. Although all the programs generate the model of the mechanical system as a set of differential equations, only some are designed to provide these equations in a symbolic form. Most of the programmes described in [16] support robot systems, however, only one, *SYM*, is specifically designed to deal with serial-link robot manipulators modelling. The programmes described in [16] are presented in separate chapters and no consistent comparison is given, although the general advantages of each software are stated.

This chapter describes three commercially available modelling and simulation computer software packages and also present comparison of their ease and performance in view of their use for robotic manipulators in particular. The purpose of this experience based comparison is to provide guidelines to choosing the convenient modelling software from the numerous programs in the market. The studied programs were available in the Department and represent three different types. AUTOLEV is an example of a symbolic modelling software based on personal computers, DADS is a numeric modelling and simulation software package based on bigger computer platforms and SD/FAST is an example of symbolic modelling software available for bigger computer platforms. Some other non-commercial programs also will be considered during the comparison process.

Industrial robot manipulators are very complex mechanical systems, therefore a simple model has been chosen to undertake the modelling and simulation as a basis of the comparison. It consists of a double inverted pendulum fixed to a moving trolley. The lower end of the first rod is fixed to the trolley by means of rotational joint. For simplicity, motion is restricted to be two dimensional, as shown in figure 3.1. The ease of the modelling and the simulation time required by each software is used for the assessment.

An example of computer aided design and simulation is also presented in this chapter to show the success of this approach and the advantages provided by allowing an insight in the robot dynamics and the evaluation of the kinematic

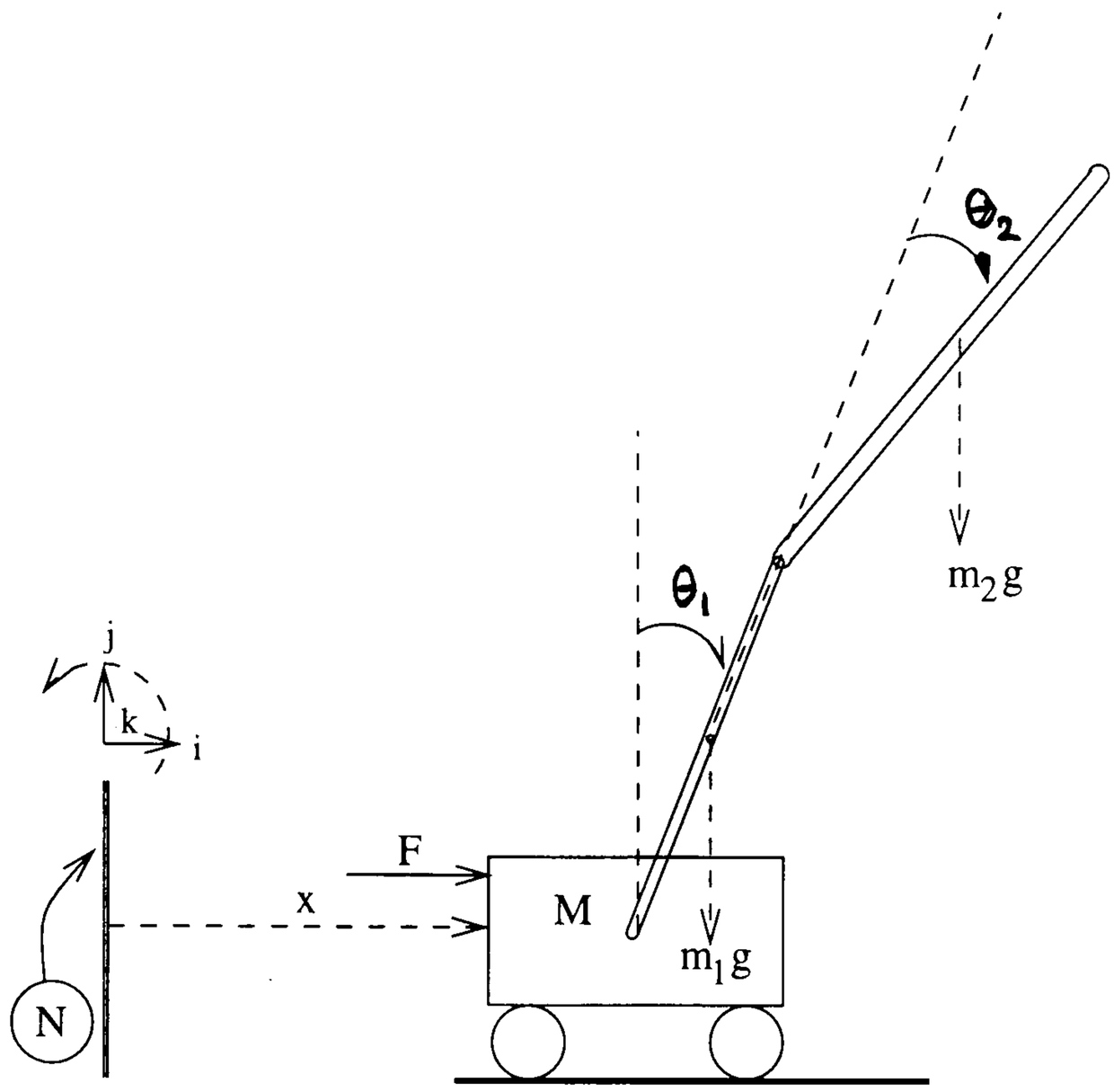


Figure 3.1: A trolley with double inverted pendulum

and dynamic design of the arm.

3.2 SD/FAST

The information in this section is mainly compiled through the use of SD/FAST acquired on a trial basis for a period of three months summarised in [38] and from [16]. SD/FAST is a multibody modelling software designed to ease producing multibody simulation with the best possible run-time performance. It is offered with choice between two formulations. The first, called *Kane's Formulation*, yields extremely good performance for smaller systems. The second is called *"Order (n) Formulation"*. The latter is preferred for larger systems, such as those which occur in spacecraft simulation.

One of the goals of SD/FAST is to produce a program that is considerably easier to learn and use than the existing numerical codes. This leads to a greater chance that the final simulation will actually be simulating the desired mechanical system.

SD/FAST provides the user with the following capabilities:

- It generates the model for free(e.g. spacecraft) or attached systems such as manipulators. This means, it is capable of modelling manipulators operating in space
- Rigid bodies and manipulators up to 50 DOF .
- 1,2, or 3-dimensional rotational joints and one-dimensional sliding joint.
- It supports serial-links manipulators as well as closed loop ones, and also non-holonomic constraints. Such constraint simulate, for instance, manipulator grasping an item on the ground or inside a spacecraft.

To accomplish the goals of SD/FAST, the task of generating the equations of motion is taken from simple, engineering-oriented system description. The system description is then provided to SD/FAST as an input file containing the appropriate information about the system geometry.

Appendix A.3 contains the complete input specification for the model shown in figure 3.1. This is presented as input file to SD/FAST, which can be edited using any available text editor and is completely free format. The words to the left of the equal sign have special meaning to SD/FAST while the others are just names and values provided by the user.

SD/FAST generates the equations of motion for the specified system and puts them in a subroutine which can be called from any available simulation medium such as ACSL, or a written FORTRAN program. SD/FAST also produces output in the ADSIM simulation language. At the time of making these comparisons it was planned that further releases would have additional languages, C and ADA. One has to mention at this stage that SD/FAST is not a simulation language, it only formulates the mathematical model and requires a simulation medium to accomplish the task.

3.3 AUTOLEV

AUTOLEV is an interactive symbolic manipulation program designed to assist the user in generating the equations of motion for multibody systems and in their FORTRAN coding to produce a simulation program.

Unlike SD/FAST and other multibody dynamics programs which can be run almost by anyone, AUTOLEV can be used effectively only by individuals with a good background in dynamics.

To accomplish the derivation of the equations of motion, the user must supply AUTOLEV with specific information concerning the particular dynamical system. Then the program performs the necessary operations and also the FORTRAN coding. AUTOLEV can be used in interactive mode or it can be supplied an input file containing the necessary information to model a mechanical system, and in both modes the intermediate calculations are displayed.

The information that must be supplied to AUTOLEV in order to generate the equations of motion of a multibody system are:

1. An expression for the angular velocity in inertial space of each rigid body in the system.

2. An expression for the velocity in inertial space for each rigid body mass centre and point at which a force contributing to generalised active force is applied in the system.

3. Expressions for force and/or torques

4. An expression for the angular acceleration in inertial space of each rigid body in the system, as well as, an expression for the acceleration in inertial space of every rigid body (link) mass-centre. These can be derived with AUTOLEV commands from the information already in the workspace.

Appendix A.4 shows AUTOLEV's input file containing the necessary specifications to generate the model for the trolley with double inverted pendulum mentioned above. The subsequent FORTRAN simulation program is presented in Appendix A.5. In the case of the interactive mode the lines constituting the input file are typed line by line.

AUTOLEV is designed for use on personal computers which are, by their nature, slower than mainframe computers and process less memory. Although this means that AUTOLEV cannot handle extremely large systems, it can do those of approximately 10 bodies efficiently [38] [39] .

The features of AUTOLEV could be summarised in the following;

- It generates the model for free(e.g. spacecraft) or attached systems such as manipulators. This means, it is capable of modelling manipulators operating in space
- Rigid bodies and manipulators up to 10 bodies.
- Produces complete, fully formatted, ready to compile and run FORTRAN simulation programs, where repeated strings have been replaced by new symbols.
- It supports serial-links manipulators and closed loops linkages and a variety of constraints
- Only available for desktop computers while the formatted simulation code can be run on any machine or platform that has a FORTRAN compiler.

The equations of motion for the system of figure 3.1 were generated, as well as the FORTRAN simulation program by AUTOLEV, using the input file of Appendix A.4. When the simulation program was run however, five seconds simulation took a considerably long time. The same program has been edited and put in ACSL form, only the DEQS subroutine was replaced by appropriate

commands to perform the integrations. Using the physical data given in [40], by Michel, and the controller to keep the two links in the upper vertical position, exactly the same response resulted in considerably improved run-time period.

3.4 DADS

DADS: the Dynamic Analysis and Design System software is a set of general purpose computer programs designed to model and simulate the behaviour of a variety of mechanical systems, including robot manipulators [16] [41].

DADS builds a mathematical model of the real system using a description of the system consisting of a set of data, which can be created interactively using the preprocessor program. The mathematical model calculates positions, velocities and accelerations of the bodies of the system.

DADS ,–as well as other programs–, provides the user with the possibility of simulating a wide range of alternate designs prior to building and testing prototypes, since it contains a large library of mechanical elements. The most important elements of DADS' library are: rigid and flexible bodies, joints and other constraints, force and torque elements and control and hydraulic elements. In addition DADS provides the possibility to create the model in two or three dimensions. The DADS related files representing the system of the trolley with double inverted pendulum are not included due to their considerable length. The joints and the bodies are separate built-in elements, the user must supply only the numerical data. Once the model has been created, the data are processed by the DADS analysis program which perform the mathematical assembly. Then the equations of motion are generated and solved numerically. The results wanted from this case are the position, the velocity and the acceleration of each body of the system since the analysis chosen is dynamic. DADS provides different types of analysis dynamic, kinematic, static and inverse dynamic as well as the choice

between global or local body-fixed reference frames.

DADS provides the user with the following capabilities;

- It simulates free and attached systems such as manipulators and therefore, it is capable of modelling manipulators operating in space
- It is capable of dealing with simple and complicated systems in two as well as in three dimensions.
- Its library contains several control elements.
- It supports serial-links manipulators as well as closed loop ones, and many types of constraints can be added to the model.
- DADS provides an animation program to permit a deep insight on the behaviour of a system.

DADS contains considerably big programs and can be supported more efficiently by big computing platforms, although a version for personal computers does exist. It is also quite slow when simulating three dimensional systems such as manipulators and does not produce the mathematical model in symbolic form in any case. The large number of the elements of the DADS library gives a big range of choice, however it makes the task of modelling more demanding

3.5 Comparison

Generally the modelling packages put an end to the tedious work of generating the equations of motion by hand and considerably reduce the required time to generate the equations of motion. However, the current packages, that are looked at so far, are good at producing models and simulation codes, but provide little support for other modelling activities such as control design and dynamic understanding.

SD/FAST is the easiest to use between the aforementioned modelling programs as the descriptive input file can be written by anyone with little knowledge of dynamics. Although DADS is conceptually straight forward to use, the fact that the information and data have to be entered in certain format and for each element, makes it hard for individuals with little knowledge of dynamics. AUTOLEV can be used properly *only* by individuals with good dynamics knowledge.

SD/FAST and AUTOLEV generate the equations of motion in a very similar format, in symbolic form, and also both use the same formalism. However AUTOLEV provides this equation inside a complete FORTRAN program ready for use and SD/FAST needs a simulation medium such as ACSL. DADS does not visualise any detail about the equations or their form and therefore they are not accessible.

Codes generated by AUTOLEV and SD/FAST are accessible, clearly commented and easily edited and the set of equations can readily be isolated, where the states are not confused. Within DADS the states are not always accessible for observation nor for control purposes, especially when they reflect angular rates, as is the case of robot manipulators. In the 3-dimensional DADS simulation. Euler parameters only are observable and can be used to feed back for control purposes. Often, they do not represent a physical quantity that is usable in a controller or to visualise physically how a given system (robot) behave. This represented a major handicap when DADS was used to model and simulate an industrial robot. Had the axes of rotation not been parallel the task would have been very difficult. The details of this simulation are given in section 3.8. The example of figure 3.1 was modelled with the three described programmes and simulation of the system behaviour under its own weight was executed. Matlab was used to call the equations generated by SD/FAST after they had been edited to a suitable format. The FORTRAN code generated by AUTOLEV was used without any changes. The simulation under the same condition was executed on the same machines and

resulted in distinctive execution times. These are presented in table 3.1 for the various programmes:

Table 3.1: Simulation execution time in the different software

	AUTOLEV	DADS3D	DADS2D	SD/FAST
CPU (sec)	84.9	319.26	65.28	24.83

In the light of the above comparison and taking into account the available computing resources choosing the suitable software for the task is made easier. One has also to consider the information given in the next section about updates and new versions, which have not been evaluated in current study. The ability to introduce extra features, such as joint compliance and link flexibility, have also to be considered for evaluating and for acquiring the suitable program and also the ability to link to other programs for control design and analysis. The last point, one should consider is the cost of the various programs, and it appears that AUTOLEV is the less expensive. It has to be noted that the Reduce code generated equation of motion could be used in simulation with Matlab/Simulink. The performance of the combination match that of many commercial packages.

3.6 Updates

According to [16] A new version of AUTOLEV in language C, was in progress and might be completed by now or in the near future. This version would make it possible to use AUTOLEV not only on personal computers, but also on most mainframe computers.

Very recently, the new feature added to DADS consist of "*DADS/Plant*", a new module to enable the user to perform time-domain and linear analysis of systems modelled in DADS to a controller modelled in SIMULINK. The latter is to apply forces/torques to the DADS system at run time using SIMULINK

S-Function block [42].

3.7 Interfacing AUTOLEV to MATLAB and SIMULINK

As mentioned in section 3.5, one has to look at the flexibility of the modelling software to link with some extra written codes and other existing useful computer programs, such as MATLAB and SIMULINK, for control design and analysis. AUTOLEV can generate the mathematical model and FORTRAN code, therefore it is conceivable to write a front-end to prepare an AUTOLEV input file format from an easy and more convenient format. This has been as a final year project and subsequently creating the model, for a serial link robot, using AUTOLEV is made effortless for specialists and easy for individuals with little knowledge of dynamics. The code to perform this function is listed in the report [43]. Although AUTOLEV produces complete simulation programs, these do not support any control design or linearisation of the model. SIMULINK S-function block allows the user to input his own equations. An interface was written to extract the equations from the AUTOLEV output code and generate the appropriate S-function for the manipulator under consideration. The S-function can be written in MATLAB code (m file) or compiled FORTRAN or C code (MEX files). The MEX files are compiled and therefore it is suggested that they are used for faster run time [44]. At the current stage numerical values for the manipulator should be entered to the AUTOLEV input file, or added in the appropriate place to the file *filename*.f mentioned later in this section. The code we propose is written for UNIX OS and will be even more convenient for the new version of AUTOLEV. The conversion program *a2m* given in appendix A.6 takes the output file *filename*.for and creates three more files: *filename*.f, *filename*g.f and *filename*s.m. The two .f

files are compiled and linked to create the *filename*.mex*** [1] and that is the only file needed for the simulation. It is called by SIMULINK S-function. *filenames*.m. every time step. This extra interface program makes AUTOLEV very attractive modelling software and gives it the added bonus to connect with other powerful control and analysis programs. To demonstrate the improvement added to the simulation, the **a2m** generated MEX file, model, of the system shown in figure 3.1 is run through SIMULINK under the conditions described in section 3.5. The execution time shown in table 3.1 is extremely improved and brought down to only 8.99 seconds which is the shortest simulation time when compared to all times seen so far. More details about this interface and others are compiled in the internal research report [45]. The added features to AUTOLEV can also be done to the written programs described in the previous chapter to make them more useful and comparable to commercially available modelling programs, provided that the necessary time is invested on them

3.8 Computer-Aided Modelling of an Industrial Robot

Following the discussion in the previous sections, modelling and simulation programs have to be chosen depending on the task. Although DADS sounds less favourable, it provides some features which are not supported by the other two. These features include, for instance, modelling manipulators with flexible links or joints as well as providing animation facilities. The latter provide a deep insight of the nature of the movement of the manipulator mechanisms. This section summarise the modelling and simulation of the AA300 robot situated at Lamberton Robotics Ltd., Coatbridge, Motherwell, Scotland, through use of computer-based geometric modelling package I-DEAS and DADS. The objective

[1] The *** extension is given to the filename to indicate the architecture of the machine

of this work is to show the suitability of the computer-aided approach for design, analysis and dynamic evaluation. The AA300 robot is a big sized robot designed to handle big loads, however it endured motor failure. It is thought, this was due to the required large torques to drive the big masses and inertias of the robot element plus the load. The solution lies in choosing an alternative motor which will sustain the load or altering the controller to limit the input currents to acceptable level. Both solutions can be performed using simulation to avoid trials on the real system and therefore a good model is required. I-DEAS was used for the geometric modelling of the AA300 to obtain the dynamic and inertial values of the different part of the manipulator. The final masses computed compared extremely well with those obtained form on site measurements. Moreover, I-DEAS gives the position of the centre of gravity and inertia of different links which are not as easy to measure in practice. The relevant dynamic and inertial values are shown in table 3.2 and the final assembled robot is shown in figure 3.2 and the detailed modelling is presented in [46] and [47] [48]. The I-DEAS built model composite system, *i.e.* assembled robot arm is shown in figure 3.2.

3.8.1 Simulation

DADS was chosen to perform the required numerical simulation using the data supplied by the geometric model. The initial DADS model included several simplifications which can be easily incorporated to the model at later stage. A torque/force is applied to each rotational/sliding joint directly. Therefore electrical properties are not included in the model. These can be added directly into a DADS simulation in a relatively straightforward manner. Although, one of the incentives for using DADS is to include link flexibility later, tests performed on the robot elements [49] suggest that robot elements can be considered structurally rigid.

After all the model specifications were entered to the DADS pre-processor,

Table 3.2: Dynamic data generated using I-DEAS for the robot AA300

Column			
mass(kg) 1123.099		*C.G.* (0.0059, 0.9383, 0.0638)	
/	$I_{xx} = 692.5912$	$I_{yy} = 85.0788$	$I_{zz} = 699.9323$
/	$I_{xy} = 7.2543$	$I_{xz} = -0.2016$	$I_{yz} = 22.1523$
Upper Arm			
mass(kg) 201.4534		*C.G.* (0.3267, 0.3938, 0.4836)	
/	$I_{xx} = 67.5724$	$I_{yy} = 49.6198$	$I_{zz} = 33.8151$
/	$I_{xy} = 0.0444$	$I_{xz} = -0.0430$	$I_{yz} = -17.8084$
Extension Tube			
mass(kg) 118.9522		*C.G.* (0.0012, 0.2900, −0.2111)	
/	$I_{xx} = 18.3198$	$I_{yy} = 2.7994$	$I_{zz} = 17.5240$
/	$I_{xy} = -0.0511$	$I_{xz} = 0.0292$	$I_{yz} = -1.4355$
Gripper			
mass(kg) 65.1572		*C.G.* (0.0773, 0.2194, −0.0953)	
/	$I_{xx} = 1.1798$	$I_{yy} = 0.6623$	$I_{zz} = 1.2533$
/	$I_{xy} = 0.0225$	$I_{xz} = -0.0140$	$I_{yz} = 0.0382$

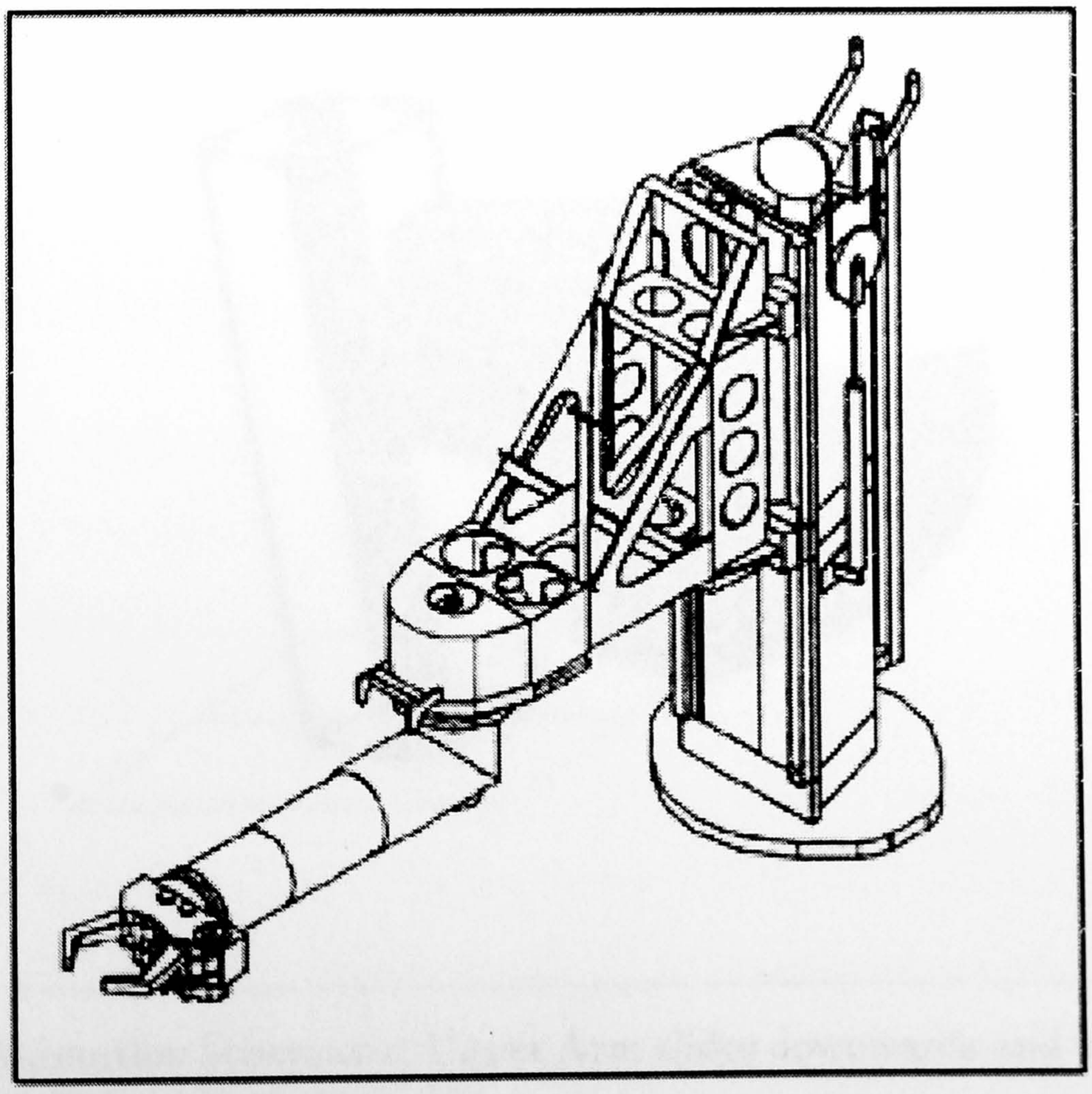


Figure 3.2: The AA300 robot manipulator assembled with I-DEAS

an inverse kinematic analysis is performed to obtain the set of joint parameters corresponding to any desired position of the end-effector. Second, a stand-alone FORTRAN program (which has been written externally) performs the path planning using the inverse kinematic analysis results to generate the curves to be followed by the joints of the robot. Finally, the actual simulation of the robot is accomplished under the action of forces and torques, the trajectories continuously monitored by a simple PD controller [38].

Several Simulation were run to mimic, typical manoeuvres required from the robot. The Upper Arm movement is unaffected by the other link whilst, the dynamics of the other two interacted, due to their parallel axes of rotation. Figure 3.3 shows a typical animation, produced by DADS, to illustrate the capability of showing the relative movement of all links of a robot, system, in one picture.

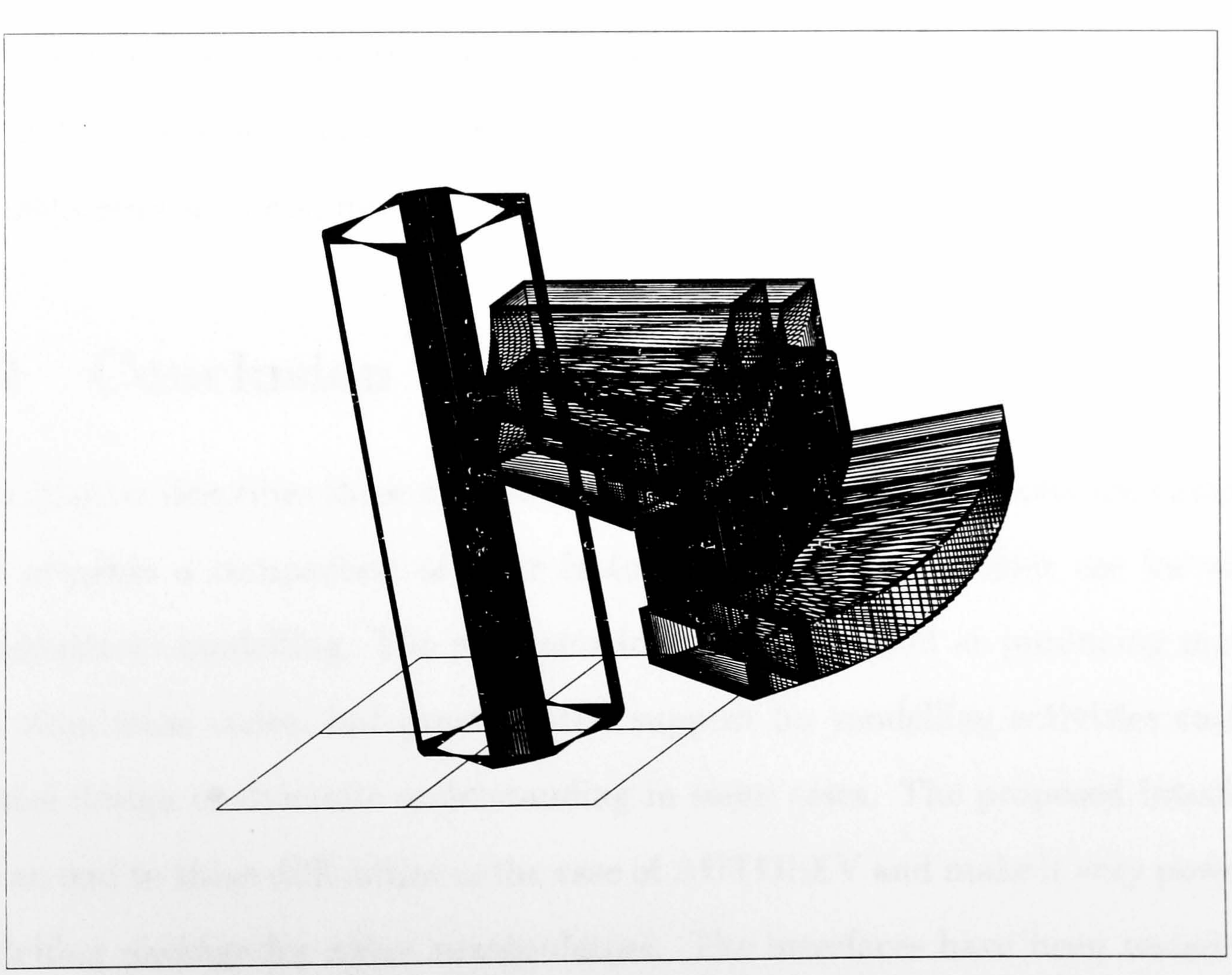


Figure 3.3: Animation Schematic: Upper Arm slides downwards and both Column and Extension Tube rotate clockwise

DADS provides several parameters for output and analysis. which include different type of energies, positions, Euler parameters. velocities. calculated applied forces and torques, and also the force *acting* at a point of particular element and the torque *acting* about a local axis. These forces and torques determine in a design process, what are the ranges expected from the actuators. In the light of this the motors are chosen, capable of delivering maximum torques not less than the maximum torques required for typical simulated manoeuvres of the robot arm. The Dynamic and electrical parameters of the motors are then added to the DADS model and final validation simulations tests are performed. In a recent work [50], DADS has also been used with other, existing finite element analysis, software in design optimisation of Flexible Mechanisms which are believed to be applicable to robot arm design. At the time of performing the aforementioned simulation, a limitation was noted, in that only the Euler parameters and angular velocities were provided for feedback; the angles are then obtained simply by integration. Obtaining the angles would prove to be a greater task for a robot with a more complex general configuration.

3.9 Conclusion

This chapter describes three modelling and simulation computer software packages and presents a comparison of their features with respect of their use for robot manipulators modelling. The programs looked at, are good at producing models and simulation codes, but provide little support for modelling activities such as control design or dynamic understanding in some cases. The proposed interfaces put an end to these difficulties in the case of AUTOLEV and make it very powerful modelling package for robot manipulators. The interfaces have been tested and results validated and produce the correct desired outcome. The problem of generating the equations of motion of manipulators is made an easy and effortless

task with computer programs. The talent and the time of engineers can be used in more effective and efficient way when using powerful modelling packages. and the comparison made in this chapter and the proposed interfaces provide guidelines to choosing the convenient program between the many offered in the market. The comparison is based on using and testing three available sample software programs representing symbolic and numeric modelling. Appendix B shows a list of the computer programs known to the author and it is thought there are still more in the market. Using numerical modelling software such as DADS to aid in the design process was discussed through an example and its use for design optimisation for flexible links is advised, based on published work [50]. While looking at the various modelling packages, it has been noticed that there was no support for manipulators operating in fluids like water. Underwater manipulator modelling is dealt with in a later chapter where the automatic model generation is also discussed.

Chapter 4

The PUMA 560 Instrumentation

4.1 Introduction

This chapter describes the development of a low cost instrumentation system based on commercially available low cost hardware and an IBM compatible personal computer. The main aim of the instrument is to measure the necessary data during a pre-determined robot arm move. The measured information is then used for the estimation of the dynamic parameters of each link of the robot arm. Although, the system was primarily designed to measure information relevant to dynamic parameter estimation, it proved to be easy to combine with other measurement systems by establishing the hand-shaking through producing or receiving a signal to trigger the sampling. This allowed the collection of more information that could be useful, for instance, for kinematic identification. The system was connected to a laser tracking system designed and built at Surrey University [51] and the combined instrument measured successfully the required information at various speeds and sampling rates. In addition to the description of the tests performed using the proposed measurement system, further uses and characteristics are also discussed in this chapter.

4.2 Background

Part of the *Metamodelling* project [52] was that the Lamberton AA300 robot arm would be modelled [48] and the model validated against the information provided by the industrial collaborator. When the data from the industrial collaborator *Lambertons* was not forthcoming, the PUMA 560 was chosen as an alternative robot to undergo the modelling and validation. The robot was modelled using different means i.e. DADS, AUTOLEV. The robot was also modelled using BONDGRAPHS as part of the Metamodelling project investigating the use of the technique in the field of robotics. The instrumentation of the robot was then deemed to be necessary for providing the measured information required for validating model. The minimum parameters needed to be measured are: the voltage/current and the angular positions of the first three motors. Ideally, however, the instrument should measure as many parameters as possible for all joints simultaneously (position of the end-effector, motor torques, current and voltage, joint angles, gearbox input and output, arm flexibility etc.), which proves to be a demanding task. An alternative could be achieved by designing the tests to cover only one joint at a time and measuring all the possible information related to the move involving the designated joint. In such a case, all relative movement of other joints must be prevented and joints locked. In fact, due to the size and the nature of our proposed system, the instrument is capable of collecting data from three links concurrently.

The instrument was developed here using commercially available plug-in cards on a 486 personal computer. An A/D card, *Lab-PC* [53], was used to capture signals from the Hall-effect devices, used to measure the motor currents and a counter/timer card was used to count the encoder pulses, used to measure the motor angular positions. The instrument system was tested and measured successfully the currents and the angular position of a PUMA560 robot arm. Extra encoders are available to be attached externally to the links to measure

their angular positions. The difference between the measurements of these and those attached to the motors reflect the gearbox and joint characteristics.

The measurement of the absolute end-effector position entailed the use of the *Optotrac* laser measurement system, described in section 4.4.1. This exercise demonstrate the flexibility of our instrument to combine with other existing measurement tools.

4.3 Description of The PUMA 560

A brief description of the main elements that constitute the PUMA 560 is given in this section to aid understanding the measurement process using this robot. The PUMA 560 is a six axis industrial manipulator manufactured by Unimation. Each motor of the PUMA 560 is instrumented with an incremental optical encoder. There are no tachometers in the PUMA 560; rather, joint positions are differenced on subsequent servo cycles to obtain an estimate of joint velocity.

The Unimation PUMA controller is a classical hierarchical controller as shown in figure 4.1 [54]. The host computer, in the factory configuration, an LSI-11/02 running the VAL robot language, communicates with the Arm Interface Board (AIB) over a bi-directional parallel bus. The LSI-11 computer carries all the high-level operations of the overall control system. It takes care of interpreting the VAL commands, performs any needed inverse kinematics, plans the desired trajectory via-points and communicates them every 28 milliseconds to the joint digital servo boards. The AIB in turn communicates with six digital servo [4] boards over a custom wired DEC double-height backplane. These boards execute commands (such as position setpoint setting, reading current encoder values. miscellaneous parameter setting) as well as implementing the position control loop. They connect through the backplane to the analog servo boards which implement nested velocity and current control loops. The older Mark I controller.

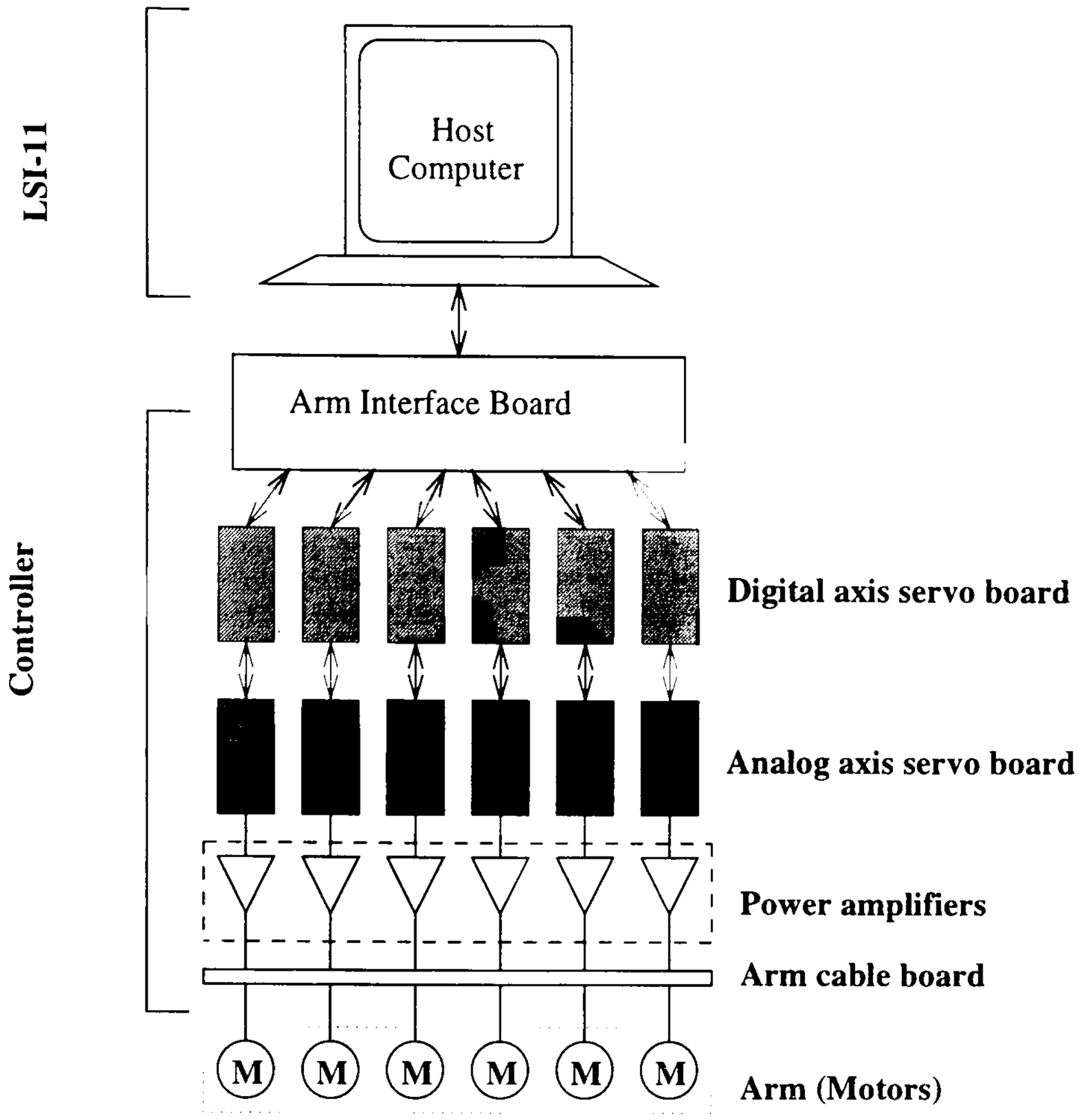


Figure 4.1: The PUMA 560 controller -Mark I

available for our tests, uses two boards per axis, one digital and one analog. In newer MARK II controllers, the digital and analog boards are combined into one single servo board per axis.

4.4 Measurement System Used

This section is divided into two main parts, the first details the characteristics of the hardware used and the second describe the software written to drive and control the measurement instrument.

4.4.1 Hardware

The main pieces of hardware used are available in the market, such as the counter card (MIT C12) and the multifunction analog and digital input/output board (Lab-PC). These and other devices used to build our measurement instrument

are described bellow.

The PC

The computer used to host the plug-in cards is a Viglen PC model Genie 4DX. It is based on, and is compatible with, the IBM personal computers with a 80486 processor and speed of $33MHz$. See [55] for more details about the computer.

The MIT C12 card

The MIT C12 is a universal counter card designed for connecting incremental shaft encoders to microcomputers. The card offers a powerful combination of incremental encoder interfacing and counter/timer functions [56].

There are three channels of incremental encoder interface with 24 bit resolution, a 32 bit incremental encoder interface and 3 channels of 16 bit programmable timer/counter.

The encoder-interface is designed to connect mechanical devices with quadrature signals to a PC. Each encoder can determine the direction and the displacement of the mechanical device. All the four encoder input channels may be independently programmed from software to operate on one of four different modes.

Lab-PC card

The Lab-PC is a low-cost multifunction analog, digital, and timing I/O board for the PC [53]. The Lab-PC contains 12 bit successive-approximation ADC with eight analog inputs, two 12 bit DACs with voltage outputs, 24 lines of TTL-compatible digital I/O, and six 16 bit counter/timer channels for timing I/O.

The multichannel analog input was used in this case for logging the output voltage of the hall-effect transducers to measure the current of the robot motors. These input channels could also be used in signal analysis. The two analog

output channels were made available to provide voltage which could be used to trigger the robot start the movement and/or generate a signal, to guarantee synchronisation with an external measurement device. In a similar way, two out of the 24 TTL-compatible lines were made available, as digital input channels. The first is for triggering the start of the data logging, if necessary, and the second is available for synchronisation. It waits for a signal from an external device to order a new reading, as required by the measurement procedure. The availability of two pairs of input and output channels dedicated to synchronisation with external devices bring about a great flexibility of measurement. For instance, during the measurement performed on the PUMA560, the *Optotrac*, described in the next section, failed to accept a synchronising signal from our instrument. Fortunately, the synchronisation was insured by a signal going in the opposite direction, generated by the Optotrac and read successfully by the Lab-PC card.

Only two counters, B1 and B2, out of the six are always available for counting/timing operations and the availability of B0 is subjected to whether it is used by the card for data acquisition/wavefrom generation. Counter B2 was then used for timing the data logging since it is not affected by any data acquisition and/or waveform generation. For more details about the Lab-PC see [53].

The Optotrack

The Optotrac, is an entirely independent measurement system , designed and built at Surrey University to measure the Cartesian position of a retroreflective target. The instrument consists of two tracking stations or sub-systems that each drive a laser beam towards a retroreflective *target* attached to the end effector of the robot by using two orthogonally mounted optical scanner units. Figure 4.2 shows the combined set-up with our measurement instrument.

During tracking, the reflected beam is laterally displaced by an amount proportional to the tracking error, which is detected within the sub-systems

and used by the controller to drive the scanners. Triangulations are then used to calculate the 3 dimensional position of the target. The instrument has a repeatability of approximately $\pm 0.1mm$ as stated in [57]. and has been used to track targets with velocities in excess of $5m/s$.

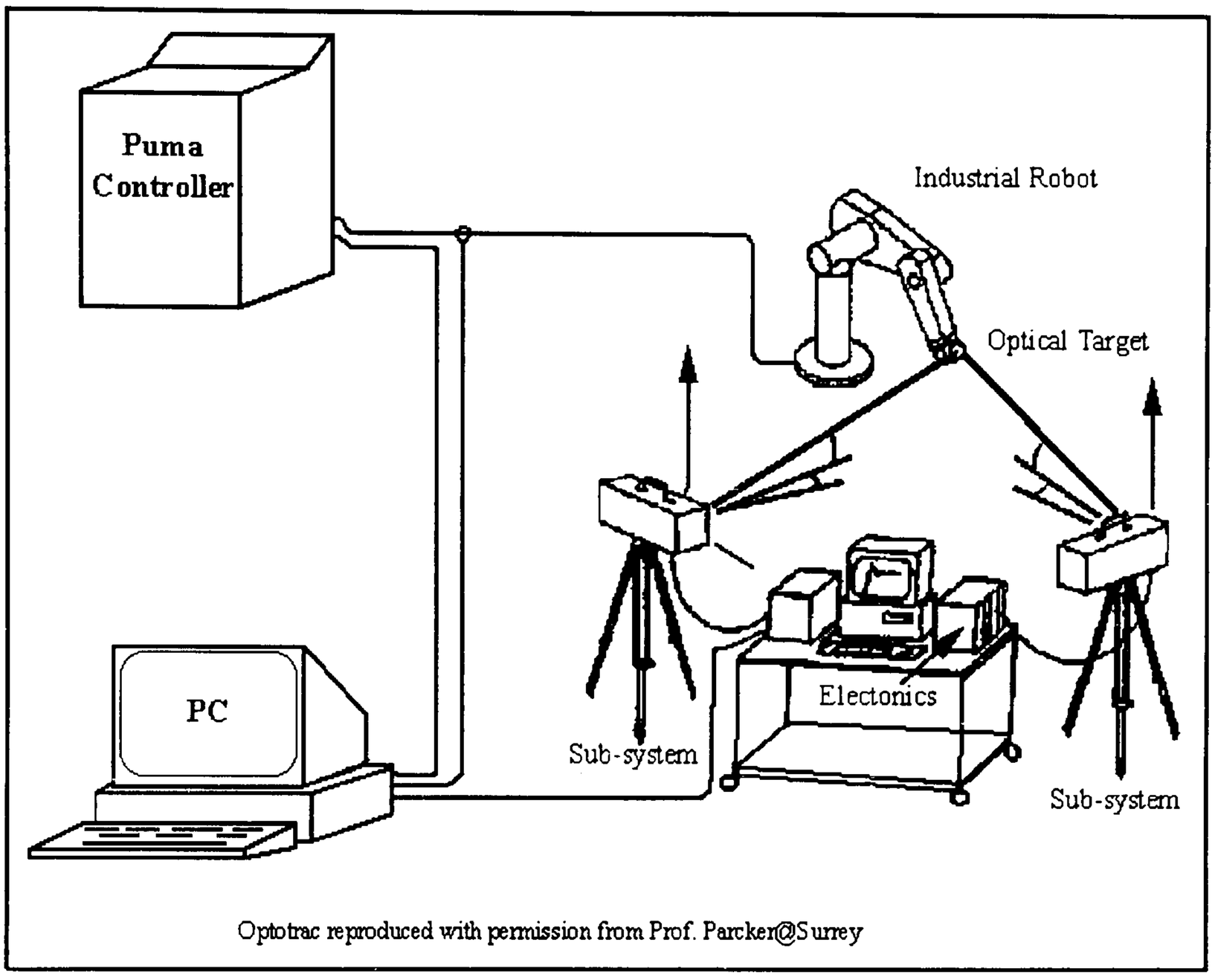


Figure 4.2: The whole data acquisition system

Circuits

Due to some difficulties in the direct measuring of some parameters, extra devices were used. To measure the torque delivered by the motors, the voltage or the current of the motors had to be measured, however the priority is for the measurement of the current since it is directly proportional to the torque:

$$T = k_m I_{ar},$$

where, T is the motor torque, k_m is the torque constant and I_{ar} is the motor armature current. Current transducers, based upon the Hall-effect. were then used

to generate a small voltage proportional to the measured current. The transducer [1] supplies an output that is linearly related to a current of 0 to 200 flowing in the centre core. Table 1 summarises the technical data of the current transducer used.

Input	Output	Supply voltage
$0 - 200Aa.c.$	$0 - \pm 10V$	$\pm 15V \pm 0.2V$

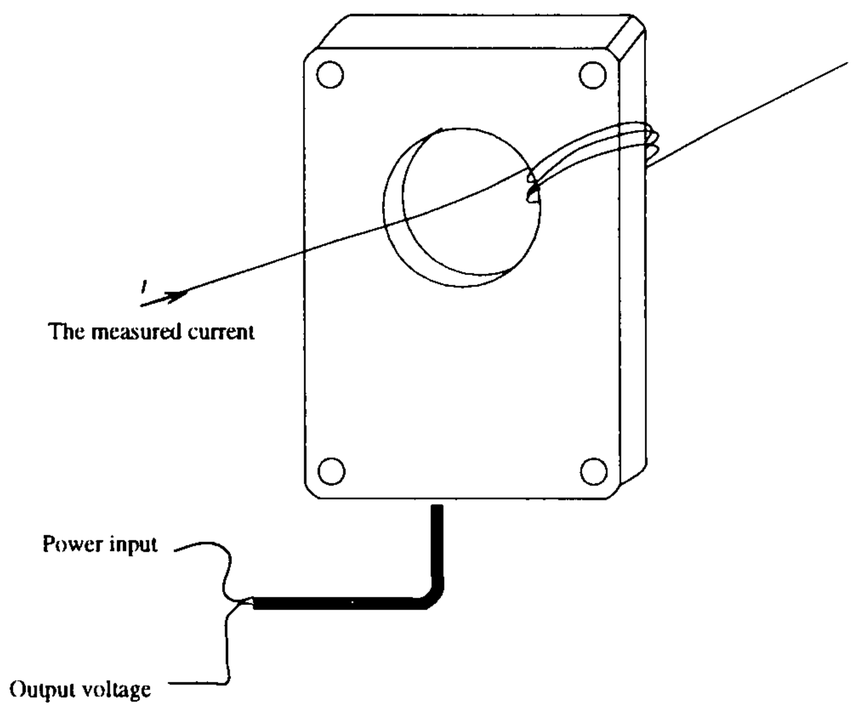


Figure 4.3: Current transducer connection

Figure 4.3 gives an idea about how the transducers are used to measure the current, which is calculated by the following formula:

$$I_w = \frac{20 \times V_t - C_{off}}{N},$$

where :

I_w is the current through the wire,

V_t is the measured voltage, N is the number of turns of the wire through the centre core of the transducer and

C_{off} is a calibration offset.

The signals generated by the PUMA 560 optical encoders are very poor and noisy such that they could not be read by the MIT-C12. An alternative solution would be tapping the encoder signals from the analog axis board and using the test pins. However the signals are sinusoidal signals and need some conditioning. The required electrical circiut for this task is given in figure 4.4.

[1] RS stock number for the current transducer used is RS 257-436

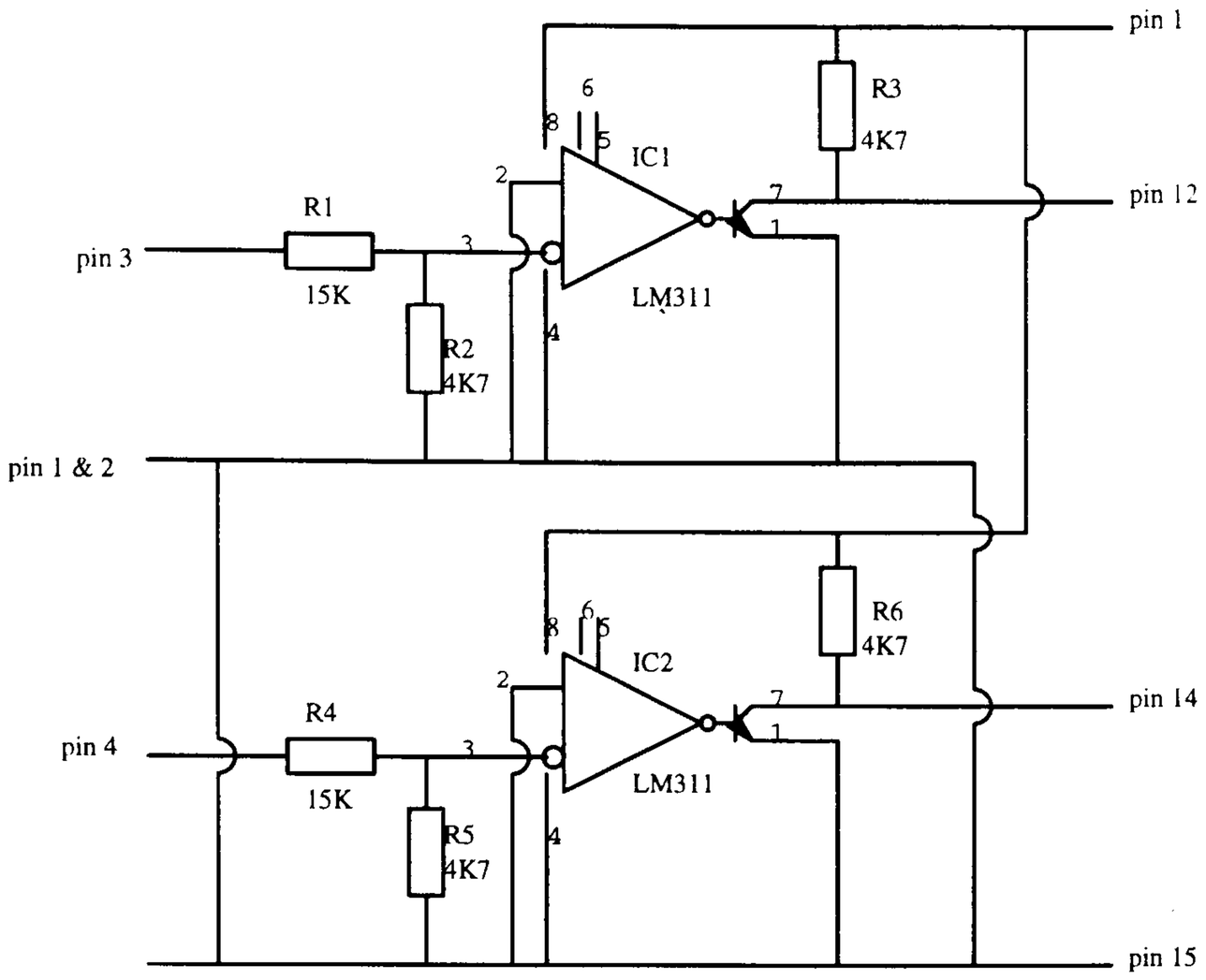


Figure 4.4: The used circuit to transform Sine signals to square signals

The conditioning circuits for the first three motor encoders signals of the PUMA 560 were fitted in one box, with inputs coming from the analog axis board and output going to MIT-C12 card in the PC.

4.4.2 Software

A program was written in C language to drive simultaneously both the MIT-C12 and the Lab-PC card plugged in the computer. All the tasks required from MIT-C12 are explicitly written within the program, including the channels needed to be read and the mode required for them. Although the MIT-C12 can handle four incremental encoders, only the three 24-bit channels were used, and programmed to operate with the same mode, to have the data with the same resolution. The mode options and other details about the MIT-C12 are given in [56]. The MIT-C12 encoder input channels start counting pulses as soon as they are configured, which is very useful for registering the initial positions of the motor shafts, and allows a consistent reading when focusing at an end of a particular move.

The functions required from the Lab-PC board are not explicitly written within the C program, rather *LabWindows* Data Acquisition Library functions

[58] were used which made the development of the program a simpler. and faster task. LabWindow data acquisition library provides high-level functions for controlling *National Instruments* data acquisition boards that can be used both in a LabWindows and standalone C programs. The user is required only to plug the right parameters in the function calls.

During each cycle of the data logging the readings include: the values of encoder pulses and the currents of the first three motors of the robot. The (X, Y, Z) of the target with reference to base frame are read by the laser system.

The hand-shaking (synchronisation) of the two systems, when using Optotrac, was insured by a square signal generated by the laser system controller that triggers the Lab-PC card. At every *low-to-high* change of the triggering signal a new set of readings are allowed to be taken by the Lab-PC and the MIT-C12. The triggering was done in this direction because the Optotrack has a maximum sampling rate of $1KHz$ and a maximum capacity of sampling 1000 samples, whereas the data acquisition system held by the PC could sample at a rate of $1.6KHz$, which makes it more convenient to receive the triggering signal. This also helps measuring the time lapse of the data logging.

4.5 Test DATA

Since the measurements were achieved by two independent systems, ours and the Optotrac, the data for each test are stored in two files. The first contains the time, the three motor angles and the three motor currents, and the second contains (x, y, z) of the end effector (target) of the robot with reference to the manipulator's base frame. The latter is stored directly by the Optotrac control system in a special format

4.5.1 The Arm Zero Position

The chosen zero position of the Arm is similar to the one given in [1], see Figure 4.5, and not the one that corresponds to the *Ready* position where the Arm is stretched up [59].

It was important to run the *diagnostic* program **POTCAL** to check the calibration parameters of the robot before any tests were performed. The new values of the calibration parameters were stored and used to replace the set contained in VAL by the VAL command *OVERLAY*. Then the calibration should be executed.

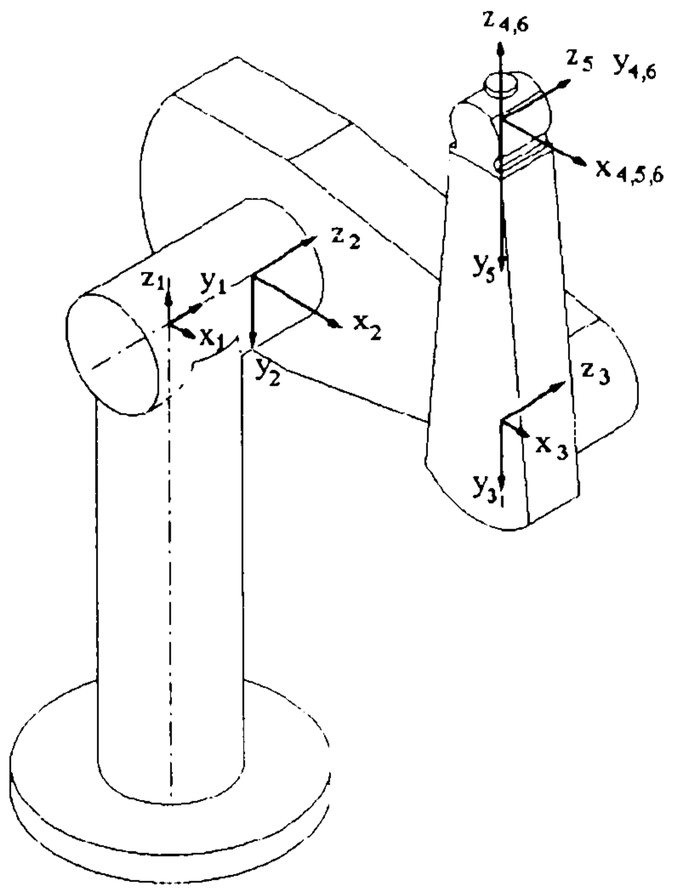


Figure 4.5: The PUMA 560 chosen Zero position; figure reproduced from [2] with permission

4.5.2 Tests

22 tests were performed, less than the number originally planed due to the time limits which include the long time of transferring the data from the Optotrac to the computer disks for each test. The data logging was performed with different

time steps for different sets of tests, depending on the time period of the move and the limit of 1000 samples of the Optotrac.

- The move for the tests concerning joint1 starts with (0,0,0) angles for joint1, 2 and 3 and 90° for joint5. This is the zero position given in [1] with joint5 rotated 90°.

- The move for the tests concerning joint2 starts from the position where the robot arm is stretched horizontally. Joint2 rotates 180° in each test.

- During joint3 tests the moves start from a configuration similar to joint2 start with joint3 rotated 90°.

4.5.3 Tests procedure

The Combined measurement instrument was configured to operate as follows: the PC begins by sending a *"start"* signal to both robot and Optotrac. From the start of the move, the Optotrac now ensures synchronisation of all measurements by sending a trigger signal back to the PC at the start of each sample cycle. Although the measurement covered the first three joints and the end-effector position during each test, only one joint is made to move at a time. Each of the joints was made to move 180 degrees at 50% and also at 100% speed. All tests were repeated with joints moving in opposite direction. More tests were performed involving the end-effector moving along the diagonal rectangle of a cubic shape according to the ISO standard test, described in the next section. Although, the measurements were collected from the three first joints, these tests involve motion of all joints of the manipulator. The Cartesian end-effector position is used in this case to evaluate the overshoot at the corners of the cube and also the offsets between the positions achieved by the manipulator and its model.

ISO Standard Test

According to [57], five test points were chosen for the overshoot measurement in accordance with the ISO test specification. The five points chosen lie on one of the four planes of the test cube which has the following properties:

1. The cube shall be located in the portion of the working space with the greatest anticipated use.

2. The cube shall have the maximum volume allowable with edges parallel to the base coordinate system.

The length chosen for the sides of the test cube is $500mm$, see Stanton [57].

4.6 Potential Uses for the System

In this section more characteristics and features of the measurement system are cited and in the light of that, more measurement procedures are proposed in order to quantify extra parameters of the robot manipulator. The available channels on the Lab-PC card can be used to measure the voltage supplied to the motor as well as the one generated by the tachometers and/or potentiometers when these are fitted to the robot motors. Voltage can also be generated to be used as a control signal. There are two optical encoders ready to be used in conjunction with the measurement system. An encoder can be fixed externally to a link and measure its exact position which would be compared to the reading of the corresponding motor position to check for gear backlash and flexibility. Generally, the use of the robot determines which details should be included in the robot model. A model used for force control purposes must include joint and link compliance's [60] which are included in position control only when significant. This is true also when the move involves contact with the end-effector and the manipulator's environment [4]. The use of the proposed measurement system is suggested as a tool in the

estimation of the link dynamic parameters. This is possible when the links are isolated and only one link is made to move at a time, during which all relative movement of other links are prevented and locked at known positions. Therefore, there are two main applications

- The determination of dynamic characteristics through measurement of angular position, velocity and motor current or voltage. This involves the use of a model of an electrical D.C. motor driving a load as detailed in [61] and a least squares identification method as detailed in [62]

- The determination of joint compliance through measurement of motor position and link position and motor current (torque) under special conditions such as constraining joint displacement. The example of such a setup is illustrated in figure 4.6 where the components of the PUMA manipulator are represented schematically. The original encoder is linked to the motor and records the rotation of the motor shaft only. The position of the link is then calculated using the gear ratio. The compliance present in the motor-link mechanism is not considered in the actual manipulator. By fixing the second encoder to the link, it would record the exact position of the link itself. This is then compared to the one calculated from the motor encoder to check for any backlash and compliance in the joint. The joint compliance is measured by moving the link against a fixed compliant work piece and recording the torque delivered by the motor, the motor position and the link position.

4.7 Examples of Measured Data

The following set of graphs is concerned with the test in which only joint 2 was made to move, where the details about the 3D coordinates of the end-effector (target) of the robot with reference to the robot base frame are shown. The angles

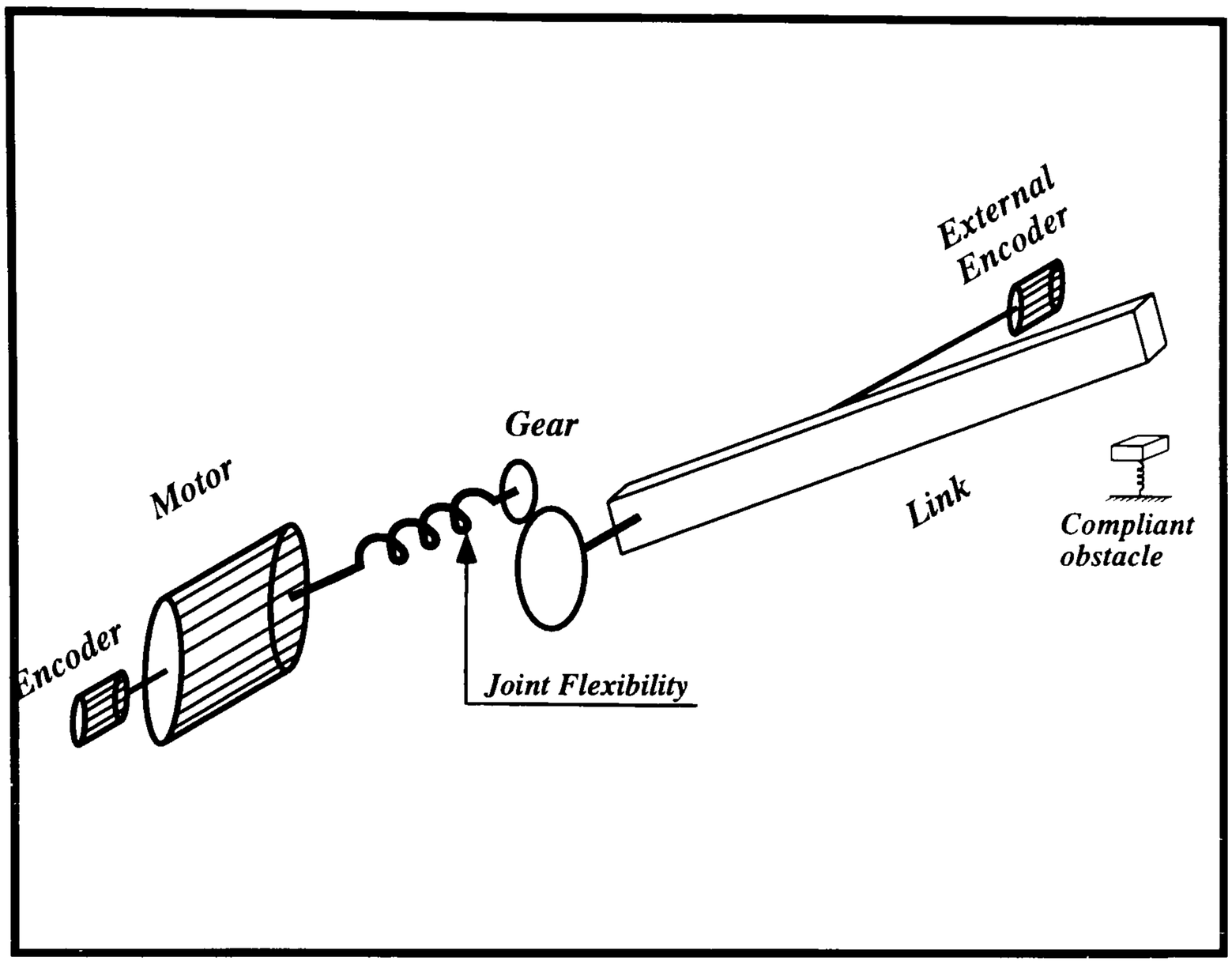


Figure 4.6: Schematic representation of a puma link with an external encoder

achieved by the three motors and their corresponding torques are also presented in figures 4.7 and 4.8.

Displacement on the x, y and z of the range of 0.1 mm and less, are observed in many of the measured data, which confirms the accuracy and the performance claimed for the Optotrac system.

4.8 Conclusion

In this Chapter the development of a low cost measurement system is described. The instrument proved to be flexible and easily configured to combine with an other instrument through receiving or generating the hand-shaking signal. It is also easily adaptable to suite a variety of measurements purposes and can be configured to be used as a control unit. The displacement measured during the tests are highly accurate and also the currents and the voltages. The Optotrac instrument provided Cartesian displacement of the order of $0.1mm$ and less which seem consistent with the nature of the moves. The obtained measurements also

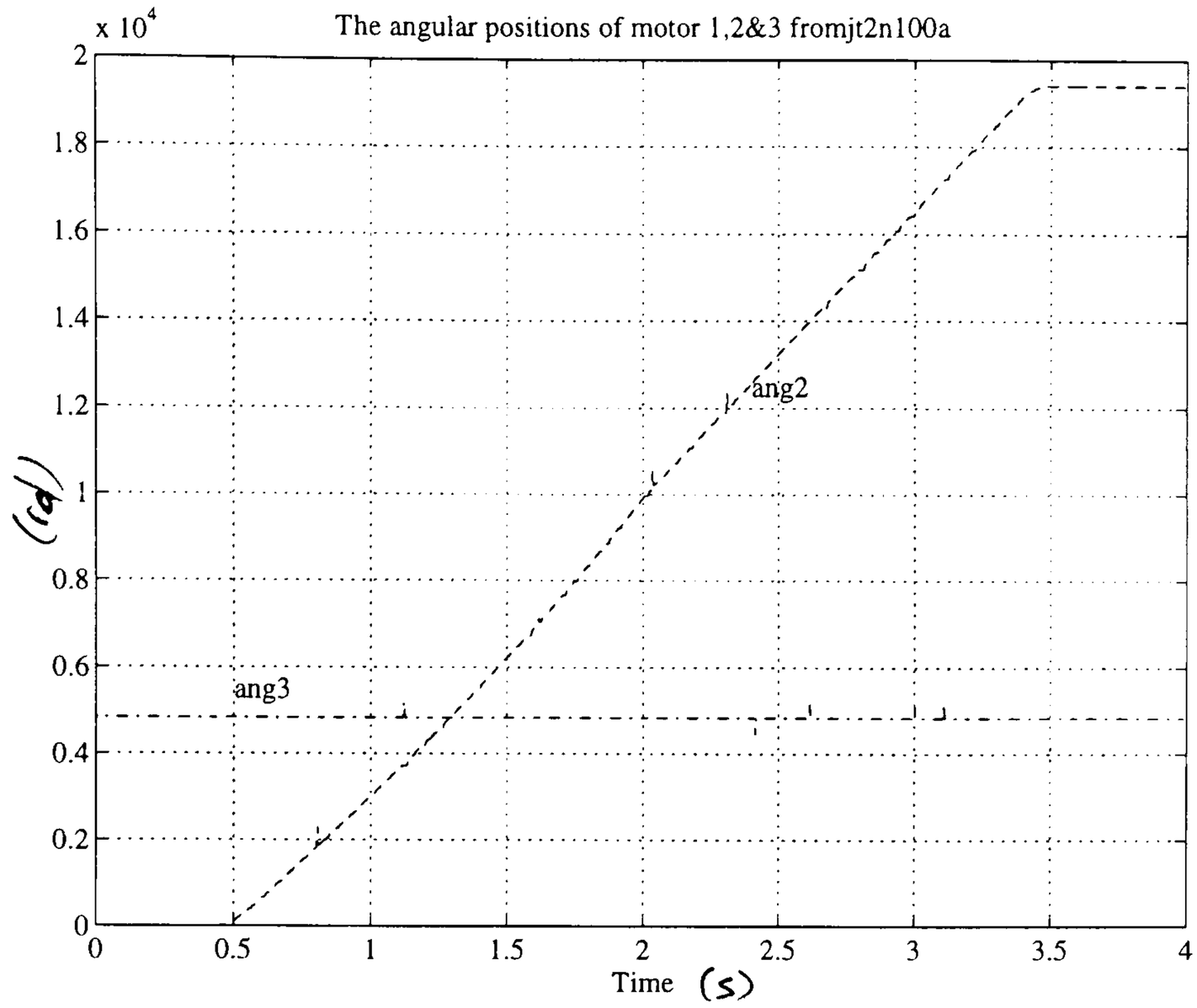


Figure 4.7: angles

showed the performance of the Puma 560 robot dynamic controller. The achieved positions by the motor compare with high accuracy to the demanded values but the positioning of the robot's end-effector is commonly known, and verified in this case, as of poor accuracy. This leaves the issue open as a kinematic problem and should be solved on kinematic level. The exact functional relationship between the end-effector position and the joints angle -*Kinematic Model* should be established and replace the existing one or used to compensate for the actual kinematic divergence

The Optotrac is a powerful and sophisticated instrument for measuring dynamically the end-point position of a manipulator moving at speeds up to 5 m/s, however it is not equipped with the necessary features to measure other related information such as joint position. It is therefore believed that the instrumentation system, developed as part of this work is the natural extension necessary to develop further the Optotrac.

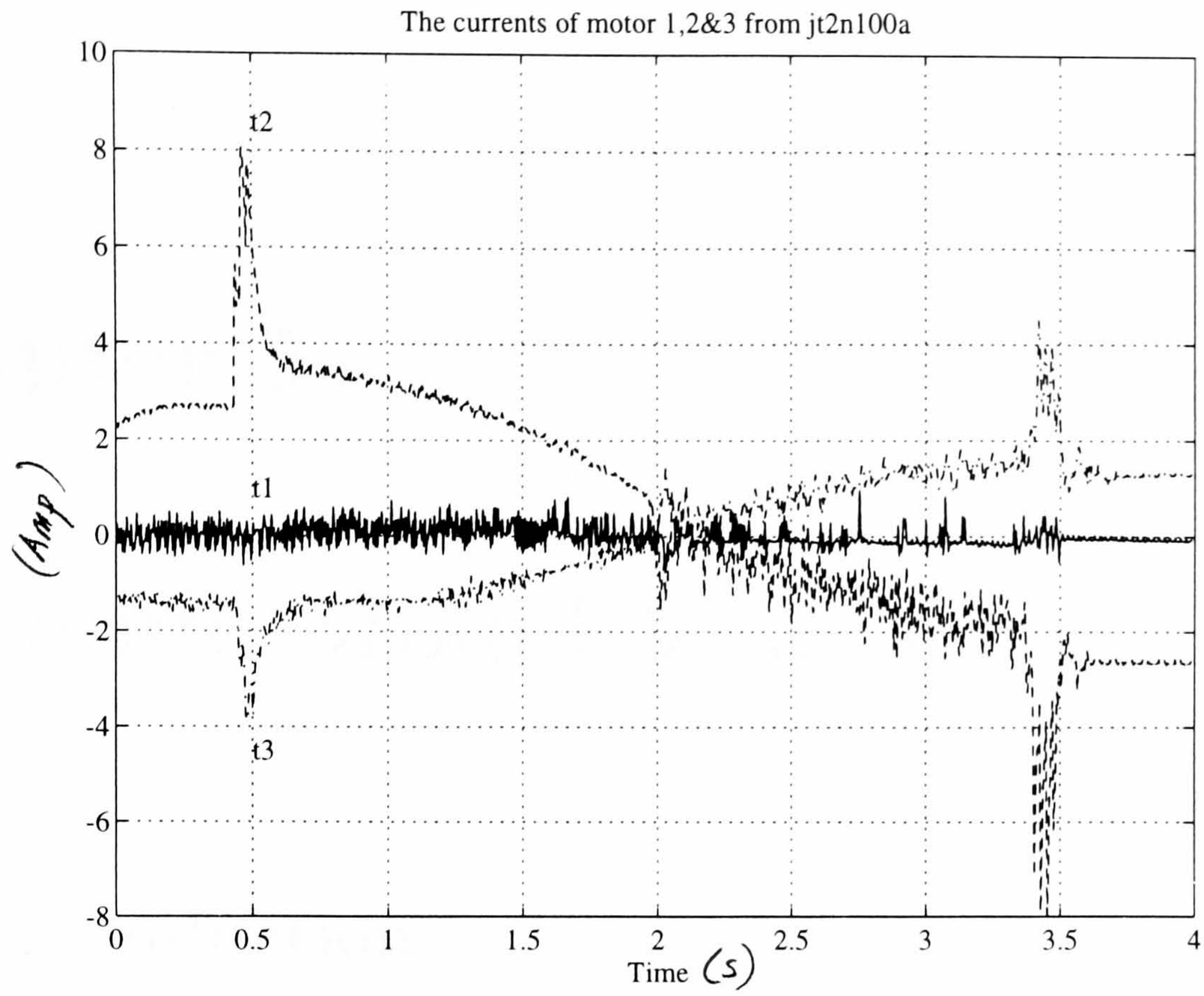


Figure 4.8: Currents

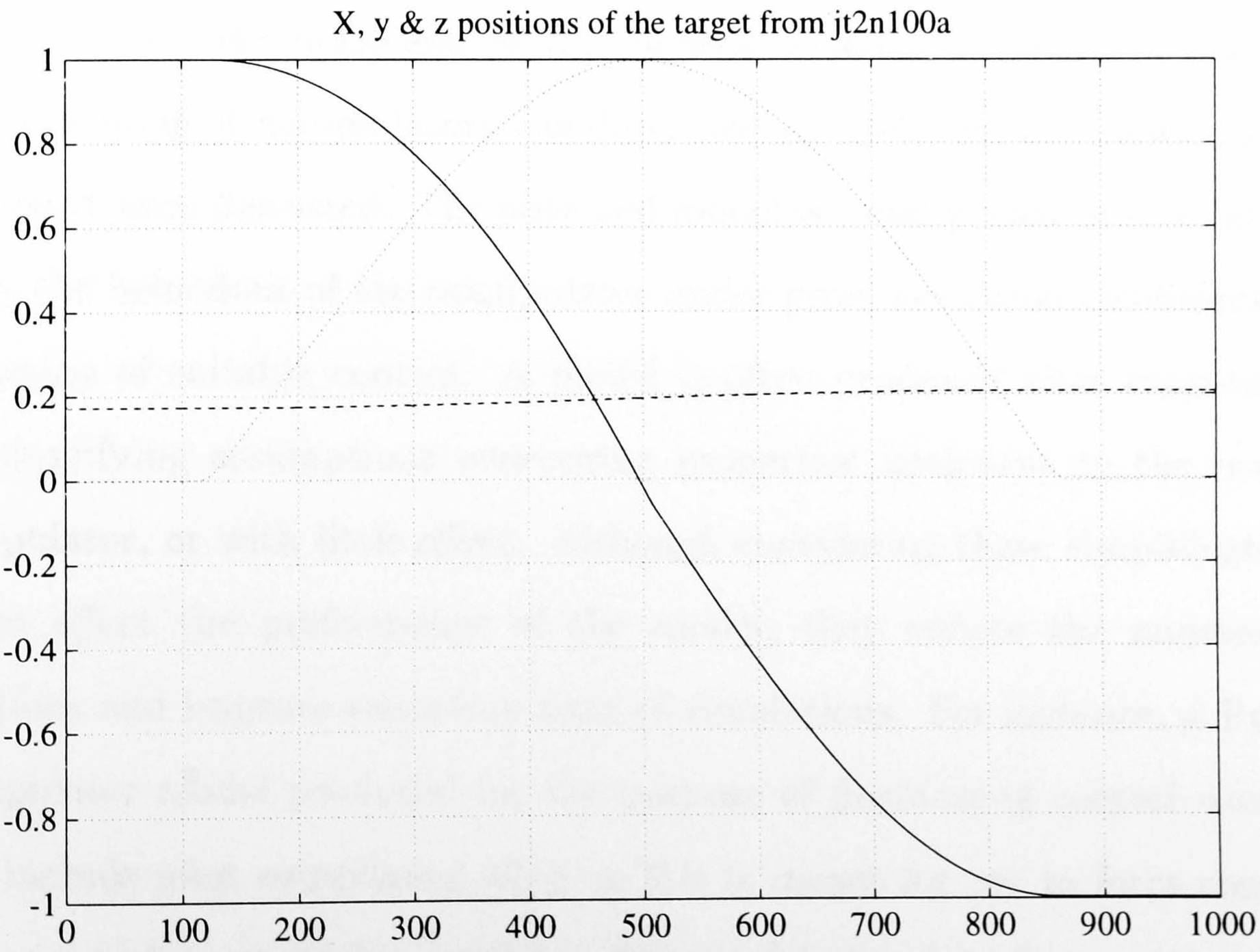


Figure 4.9: The (x, y, z) position of the target where only joint2 is moving

Chapter 5

Dynamic Model Validation

5.1 Introduction

A very important part of model development of robot manipulator or any dynamic system is ensuring that the underlying mathematical description (model) depicts to a high accuracy the behaviour of the physical system. In previous chapters dynamic modelling of manipulators was described and different automatic means of achieving it were discussed. The obtained model is usually used in simulations to predict the behaviour of the manipulator under given actuation conditions, or for the design of suitable control. A model is often produced after considering several simplifying assumptions concerning properties irrelevant to the use of the manipulator, or with little effect. Although considering these simplifications would not affect the performance of the model, they reduce the number of computations and improve execution time of simulations. For instance, a Puma 560 manipulator model produced for the purpose of positioning control usually does not include joint compliance whereas if it is meant for use in force control these characteristics would be essential. The model therefore does not have to be an exact mathematical description of the physical system (manipulator), but rather a **valid** description within the context of use of the system. Foss [8] agrees

with several authors [9] in the view that a model is validated if it is eventually judged as fit for the purpose for which it was intended. Validation has long been recognised as an integral part of model development in textbooks such as [17] [18]. Although formal validation processes are emphasised in theory, it is noted that most application papers pass over questions of validation in a superficial fashion [19]. While validation appears to have a central importance, in the past, mainly for a few safety-critical applications, the availability of low-cost high power computing facilities is extending the use of validation to other applications such as robotics [63] [64] [65] [20] [21]. Two Transactions of the Institute of Measurement and Control were dedicated to the subject of dynamic model validation [66] [9]. Dynamic model validation has close links with fault detection [20] and is based on system identification and parameter estimation techniques. Parameter sensitivity analysis is also of considerable value in establishing confidence in the validity of a given model.

Measurements performed on a PUMA 560 manipulator showed that the positions of the end-effector of the robot were not at the intended locations. At first instance, it was thought that the dynamics used for the design of the PID controller parameters were different from the actual manipulator parameters and therefore the model generated should be altered to describe this particular system (validated). A closer look at the obtained measurements revealed that the achieved joint angles of the manipulator are extremely close to the demanded values and hence the controller is highly reliable for its intended goal. It is concluded therefore that the discrepancies between the intended and the achieved positions of the end-effecor of the manipulator are due to deficiencies of the controller's internal kinematic model of the manipulator. The issue of kinematic positioning improvement is dealt with in chapter 6. The inaccurate kinematic parameters have little effect –in this case– on the dynamic performance of the manipulator although some of them, such as link lengths, appear in the dynamics.

The presence of gear-boxes increases the effective inertia of the motor rotor by the square of the gear-box ratio and hence the effects of link inertia are dominated by rotor inertia. Changes in link inertia caused by change of configuration or inaccurate knowledge of its value has little effect on the dynamic performance. Although manipulators are highly non-linear, multiple input, multiple output (MIMO) systems, most existing controllers, including the PUMA, treat them as a series of independent, linear systems; known as independent joint control [4] [6]. In this type of control, force and moment interaction between links and other non-linearities such as Coriolis and centrifugal forces and friction are ignored and considered as system disturbances. It is thought therefore that this technique could be used for validation by splitting the whole model into several individual joint models.

The dynamic model for positioning the first three links of the PUMA 560 manipulator was obtained using the software DADS and AUTOLEV described in chapter 3. The model is intended for purpose of positioning and therefore should be validated towards this end. The model is used in simulation to perform joint positioning similar to tasks programmed on a real manipulator. Measured input and output from the manipulator and its model are compared and the parameter values of the latter are tuned to produce similar responses to the real manipulator.

Butterfied [67] has pioneered a method called *model distortion method* which attempts to quantify the validity of the model by allowing change (distortion) of the model parameters as a function of time in order to obtain good data and model match in the frequency or time domains. An initial attempt to use this method implemented in MATLAB to validate one motor-link model failed to converge although it did produce a good match when only one parameter is unknown or uncertain. The method had been used successfully in the nuclear field, however it did not seem to have found widespread use in other application. Our initial result must not discourage further investigation for the applicability of the model

distortion method to robot manipulators and therefore it is included in the list of future work.

The rest of this chapter is organised in the following manner. The next section presents a motor-link pair modelling and validation and give a simulation example for the usefulness of the method. The validation of the Puma 560 manipulator is presented in section 5.3. A methodology of parameter estimation and model validation for SCARA type manipulator is proposed in section 5.4 and section 5.5 concludes the chapter.

5.2 Motor-Link Modelling and Validation

As mentioned in the previous section the independent joint control scheme employed in most of the geared robotic manipulators regard the joints as motors with dynamics combined with that of the link they actuate. For this reason, it necessary to develop a model of the motor-link pair and write it in the form best suited for system indentification, discussed further in this chapter. For the purpose of validation the model should be written in terms of the measurable parameters, and hence it is represented by these within a particular structure. While some parameters of the motor and link may be directly measurable, such as masses and motor torque constants, a manipulator needs to be disassembled to facilitate their measurement. These could also be obtained from the manufacturer. Joint and motor frictions on the other hand are extremely difficult to measure accurately. This isue is dealt with next.

5.2.1 Motor-Link Friction Model

The joint friction model is the combination of joint friction itself and the friction of the motor. In the presence of a gear box, the gear friction is also combined with the previous two. Although several friction models have been developed and

proved to be correct in each case [68] they all contain a linear viscous friction and a non-linear stiction and coulomb friction. A simple version of the friction model [15] [6] [68] is considered as shown in figure 5.1. By ignoring the stiction part as

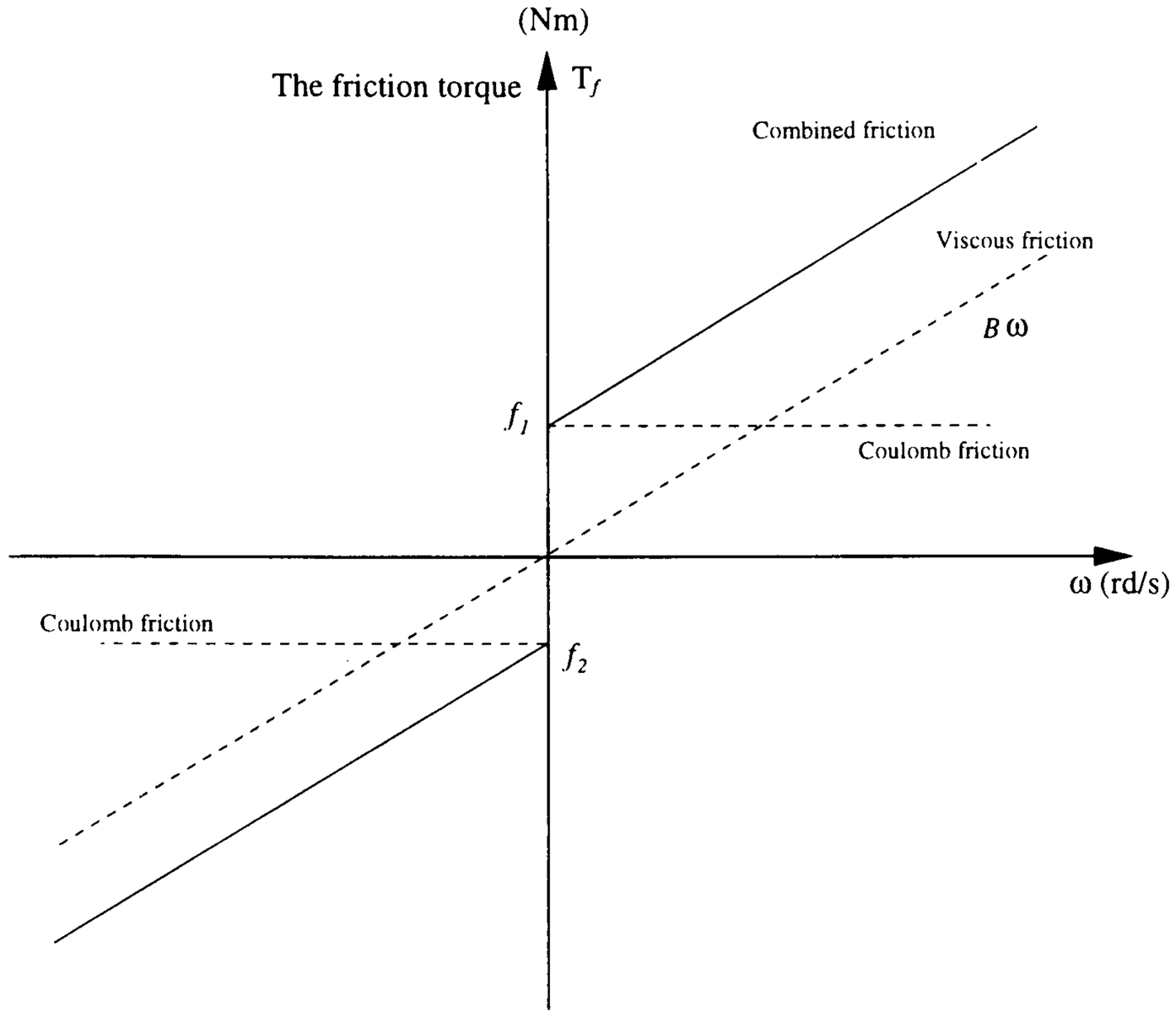


Figure 5.1: Friction Characteristics of Motor-Link

shown in the figure the friction resisting torque is written in the following form:

$$T_f(t) = -B\frac{d\theta}{dt} - f_1 sign(\omega(t) + |\omega(t)|) - f_2 sign(\omega(t) - |\omega(t)|) \tag{5.1}$$

where $\omega(t) = \frac{d\theta(t)}{dt}$.

This characteristic is included in the overall motor-link model next.

5.2.2 Motor-Link Model

Since the model is intended for a current driven motor as is the case of the PUMA manipulator it should relate the measurable parameters to the armature current input. The torque delivered by the motor is:

$$T(t) = ki_a(t) \tag{5.2}$$

where k and i_a are the motor torque constant and the armature current.

Vertical Axis of Rotation

Assuming the combined motor and link inertia seen by the motor is I then the torque is given by:

$$\begin{aligned} T(t) &= I\frac{d^2\theta(t)}{dt^2} + B\frac{d\theta(t)}{dt} + f_1 sign(\omega(t) + |\omega(t)|) \\ &\quad + f_2 sign(\omega(t) - |\omega(t)|) \end{aligned} \tag{5.3}$$

Horizontal Axis of Rotation

Although gravity loading has been included in the discussion parts of some publications such as [68] the author does not know of any that include it in the model for the purpose of system identification. In practice, many existing manipulators include links rotating around horizontal axes, therefore this case is included in this work.

When the axis of rotation is horizontal equation 5.3 becomes:

$$\begin{aligned} T(t) &= I\frac{d^2\theta(t)}{dt^2} + B\frac{d\theta(t)}{dt} + Mgl\sin(\theta(t)) + \\ &\quad f_1 sign(\omega(t) + |\omega(t)|) + f_2 sign(\omega(t) - |\omega(t)|) \end{aligned} \tag{5.4}$$

where M is the link mass and l is the distance between the axis of rotation and the link centre of gravity in the perpendicular plane to the axis of rotation. In the presence of a gear $\sin(\theta)$ becomes $\sin(\theta/n)$, with n the gear ratio.
The above models can therefore be used for system identification as shown in the next section.

5.2.3 System Identification of Motor-Link Model

Nonlinearities

A convenient way of model representation for parametric identification is the Laplace transform, however the models 5.3 and 5.4 include nonlinear terms. These are dealt with in the following manner;

Gawthrop [69] argues that the Laplace transform may exist for certain linear-in-parameters nonlinearities. If the differential equation is represented as:

$$y^{[n]}(t) + a_1 y^{[a-1]}(t) + \cdots + a_n y(t) + y_N(t) = 0 \tag{5.5}$$

where

$$y_N = \sum_{j=1}^{m} n_j N_j(y(t)) \tag{5.6}$$

and the nonlinear terms N_j are known functions of y but the scalar terms n_j are unknowns. It is assumed that each nonlinearity N_j, with the signal $y(t)$ as an input, defines a signal $N_j(y(t))$ at the nonlinearity output. This *measurable* output signal is assumed to be well behaved enough for its Laplace transform to exist and is represented by $\bar{N}_j(s)$ [69].

According to the above theory the friction torque 5.1 is rewritten as:

$$T_f(t) = -B\frac{d\theta}{dt} - f_1 f_{r1}(\theta(t)) - f_2 f_{r2}(\theta(t)) \tag{5.7}$$

where

$$f_{r1} = sign(\omega(t) + |\omega(t)|) \qquad \text{and} \tag{5.8}$$

$$f_{r2} = sign(\omega(t) - |\omega(t)|) \tag{5.9}$$

and its Laplace transform is therefore given by:

$$T_f(s) = -Bs\theta(s) - f_1 \bar{F}_{r1}(s) - f_2 \bar{F}_{r2}(s) \tag{5.10}$$

The identification method

The standard form for system identification is

$$y(s) = \frac{D(s)}{A(s)} u(s)$$

where $u(s)$ and $y(s)$ are respectively the system input and output.

Using 5.10 and the above form the rearranged Laplace transform of 5.3 results in:

$$\frac{s(s\theta(s))}{C(s)} = \frac{(k/I)I_a(s)}{C(s)} - \frac{(B/I)s\theta(s)}{C(s)} - \frac{(f_1/I)\bar{F}_{r1}(s)}{C(s)} - \frac{(f_2/I)\bar{F}_{r2}(s)}{C(s)} \quad (5.11)$$

The state variable filter, $C(s)$, used above is a second order polynomial introduced to avoid noise amplification effect of differentiation.

The above equation can be converted to its time domain form as:

$$\phi(t) = x^T(t)\,Q \quad (5.12)$$

where the filtered output $\Phi(s)$ is

$$\Phi(s) = \frac{s}{C(s)}(s\theta(s)) \quad (5.13)$$

and the filtered information vector $X(s)$ is

$$X(s) = \left[\frac{I_a(s)}{C(s)} \quad \frac{s\theta(s)}{C(s)} \quad \frac{\bar{F}_{r1}(s)}{C(s)} \quad \frac{\bar{F}_{r2}(s)}{C(s)}\right]^T \quad (5.14)$$

in its measurable signal form X is

$$X(s) = \left[\frac{I_a(s)}{C(s)}\ \frac{s\theta(s)}{C(s)}\ \frac{sign(\omega + |\omega|)}{C(s)}\ \frac{sign(\omega - |\omega|)}{C(s)}\right]^T \quad (5.15)$$

The parameter vector Q is

$$Q = \left[\frac{k}{I} \quad -\frac{B}{I} \quad -\frac{f_1}{I} \quad -\frac{f_2}{I}\right]^T \quad (5.16)$$

The parameter vector Q can be obtained from the standard least squares identification routine using, as input, the motor current, the angular velocity as well as the two *sign* signals constructed from the velocity. Since k is measurable or obtainable from the motor characteristics sheets the other parameters are easily calculated.

The above is not the only method for the estimation of the parameters, there are others such as the Singular Value Decomposition method which can be used effectively for the same purpose [6].

In the case of the model of equation 5.4, when the gravity loading is present. the nonlinearities are extended to include the $\sin(\theta(t))$ term. This is considered an accesible signal, and using the same theory inroduced by Gawthrop [69] discussed earlier, its Laplace transform is given the form shown in equation 5.17 bellow. Therefore, the model of the motor-link pair with a horizontal axis can be dealt with in its Laplace transform and is given by:

$$\begin{aligned}\frac{s(s\theta(s))}{C(s)} &= \frac{(k/I)I_a(s)}{C(s)} - \frac{(B/I)s\theta(s)}{C(s)} - \frac{(Mgl/I)\bar{G}_r(s)}{C(s)} \\ &\quad - \frac{(f_1/I)\bar{F}_{r1}(s)}{C(s)} - \frac{(f_2/I)\bar{F}_{r2}(s)}{C(s)}\end{aligned} \tag{5.17}$$

where $\bar{G}_r(s)$ is the Laplace transform corresponding to $G_r(\theta(t)) = \sin(theta(t))$. Hence the output and the information vectors are formed in the same manner as in equation 5.13 to 5.16 and the parameters are calculated using a least square routine.

An alternative way for dealing with estimation problem is using equations 5.2 and 5.4 and rearranging the model to the form:

$$b = A\,x$$

then the model is given by:

$$\begin{aligned}i_a(t) &= (I/k)\frac{d^2\theta(t)}{dt^2} + (B/k)\frac{d\theta(t)}{dt} + (Mgl/k)\sin(\theta(t)) + \\ &\quad (f_1/k)\,sign(\omega(t) + |\omega(t)|) + (f_2/k)\,sign(\omega(t) - |\omega(t)|)\end{aligned} \tag{5.18}$$

Therefore

$$b = i_a \tag{5.19}$$

$$A = \left[\frac{d^2\theta(t)}{dt^2}\ \frac{d\theta(t)}{dt}\ \sin(\theta(t))\ sign(\omega(t) + |\omega(t)|)\ sign(\omega(t) - |\omega(t)|)\right] \tag{5.20}$$

and the parameter vector x is

$$x = \left[\frac{I}{k}\quad \frac{B}{k}\quad \frac{Mgl}{k}\quad \frac{f_1}{k}\quad \frac{f_2}{k}\right]^T \tag{5.21}$$

In fact the above A and b are respectively an $(n \times m)$ matrix and a column vector of size m, where n is the number of parameters and m is the number of measured inputs.

The vector x is calculated with the Singular Value Decomposition routine as

$$x = A^{+} b \tag{5.22}$$

where A^{+} is the pseudo inverse of A formed by the singular value decomposition. It is obtained in MATLAB simply by the **pinv** function. The inputs forming A contain the angular acceleration which is not measurable. To avoid noise amplification through differentiation of the velocity the two sides of equation 5.18 can be integrated and solved for the parameters x as described above.

Test of the Gravity Loading Estimation

Since the system identification of motor-link with a vertical axis has been verified [6], in this section a simulation results of a motor-link is used to verify the second alternative, *i.e.* the case when the gravity loading parameter is present on the model. SIMULINK was used to simulate the movement of a motor-link with a horizontal axis under a PID[1] controller as illustrated in figure 5.2.

The model used for the simulation was chosen to have the following characteristics: link mass $M = 9.5kg$, motor torque constant $k = 0.2587$, combined inertia about centre of gravity $I = 2kgm^2$, $l = 0.1m$ viscous friction coefficient $B = 1Nm\,s/rad$ and Coulomb friction coefficients $f_1 = f_2 = 2Nm$. An idealised gearbox was also used with a ratio $n = 50$. The model is then made to follow a trajectory and the position, velocity and current shown in figure 5.3 to 5.5 were recorded for use in the identification routine. The recorded velocity includes some noise to simulate measurement noise, and is used with other recorded data to estimate the motor-link parameters according to the singular value decomposition

[1] the controller type is not relevant as far as the motor input current is measured

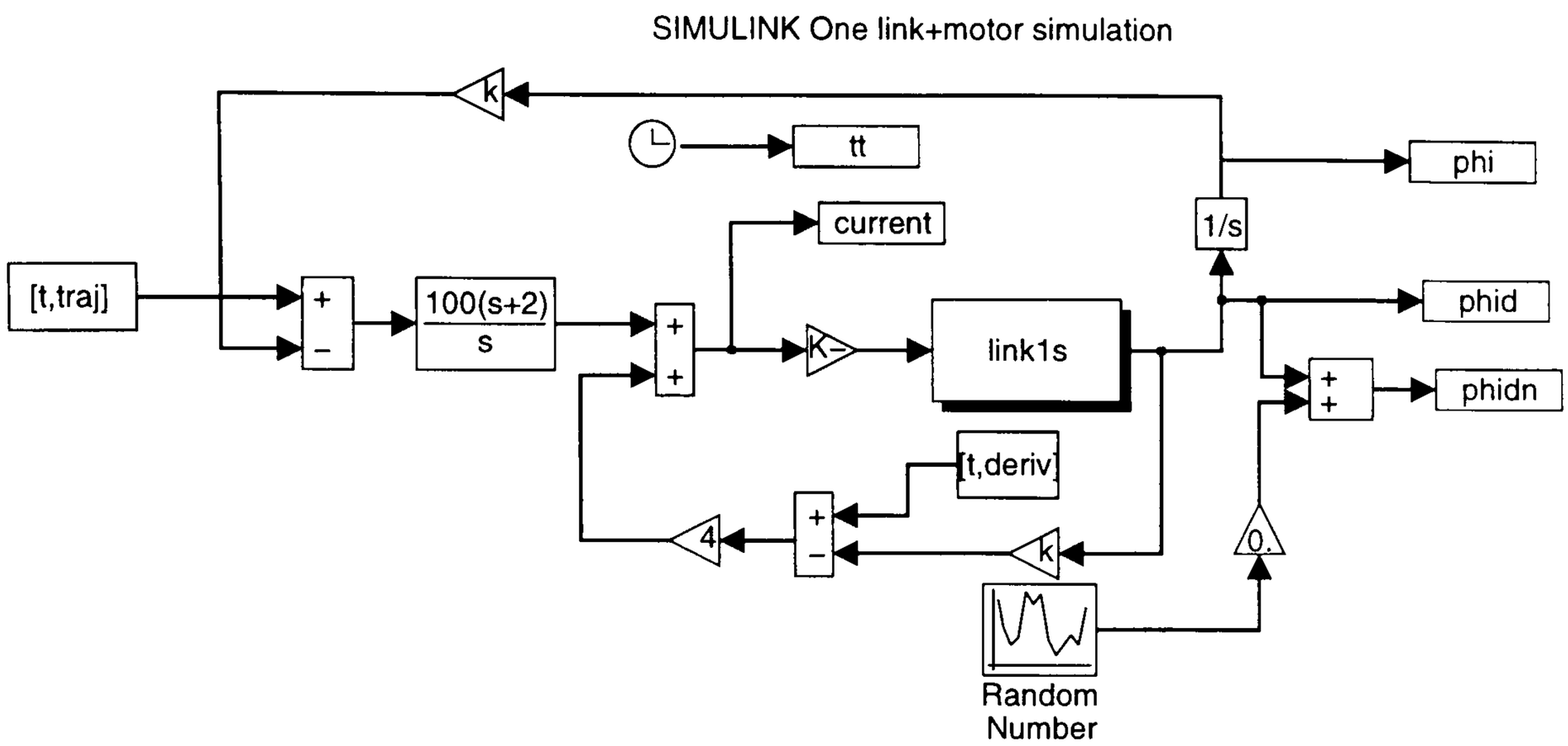


Figure 5.2: SIMULINK configuration for a motor-link simulation with gravity loading

described in previous section. The results are given in table 5.1 which demonstrate the success of the methodology. The shown values indicate that the gravity loading can be estimated with the other parameters in the case of vertically rotating links. The third column contains the parameters estimated from measurements without noise which give the exact parameter values where, for instance, the inertia is 0.0950 more than that of first column. This value is exactly the $M\,l^2$, because the estimated inertia is about the axis of rotation.

Figures 5.3, 5.4 and 5.5 also contain the position, velocity and current obtained from using the model with identified parameters from column 4 of table 5.1 in simulation. Although the values used are slightly different from the real parameter values, a good match is shown and therefore the identified model is valid for positioning purposes. The results of this section demonstrate that the method described is applicable to similar real cases.

Table 5.1: The estimated values of the motor-link from simulation result

Parameter	Given value	Estimated (no noise)	Estimated (with noise)
$I\,(kgm^2)$	2.00	2.0950	2.0743
$B\,(Nm\,s/rad)$	1.00	1.0000	0.9967
$Mgl\,(Nm)$	9.3163	9.3163	9.3288
$f_1\,(Nm)$	2	2.0000	2.0044
$f_2\,(Nm)$	2	2.0000	1.9953

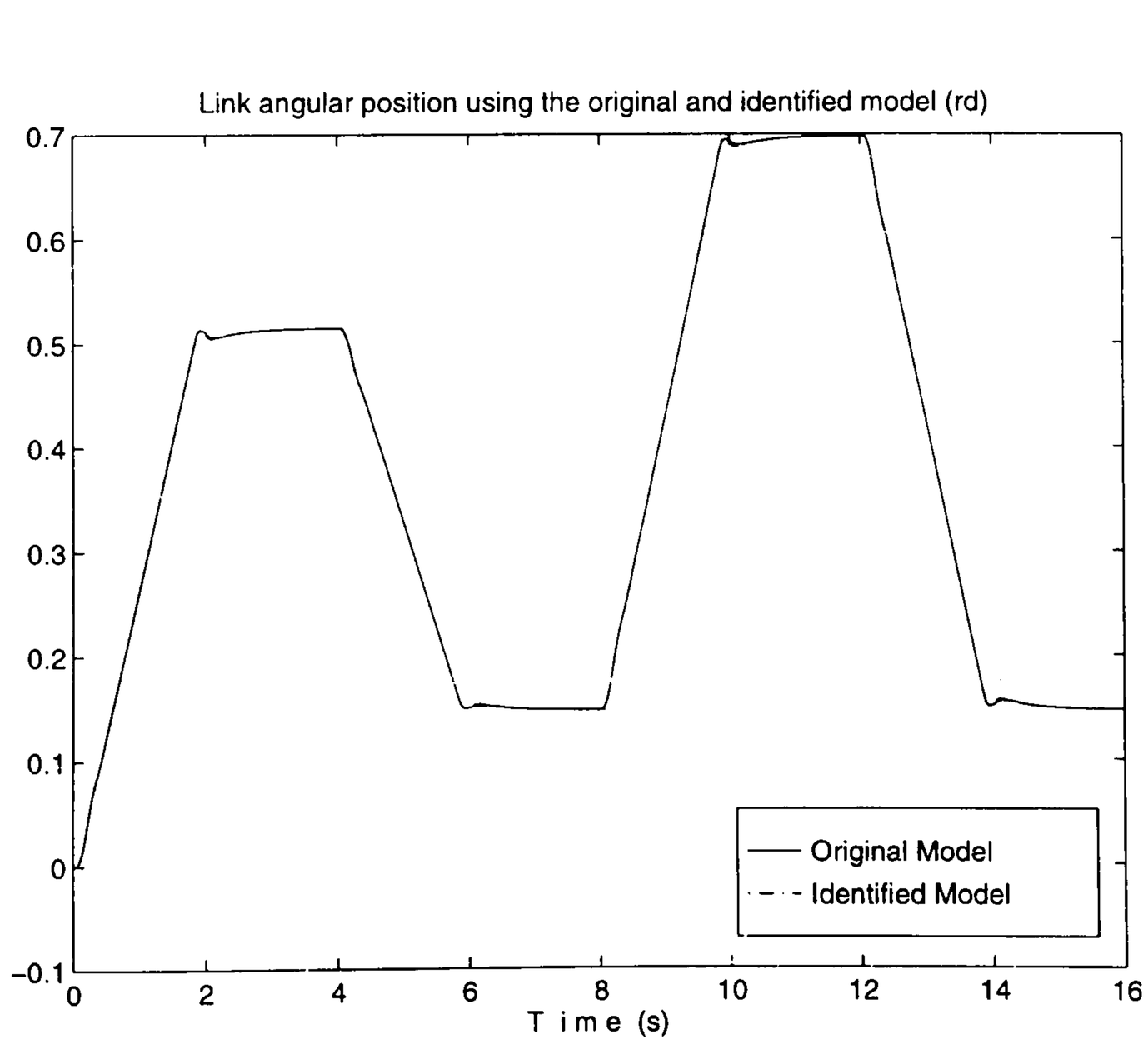


Figure 5.3: The output position of the link with gravity loading using the original and identified model show a good match(simulation)

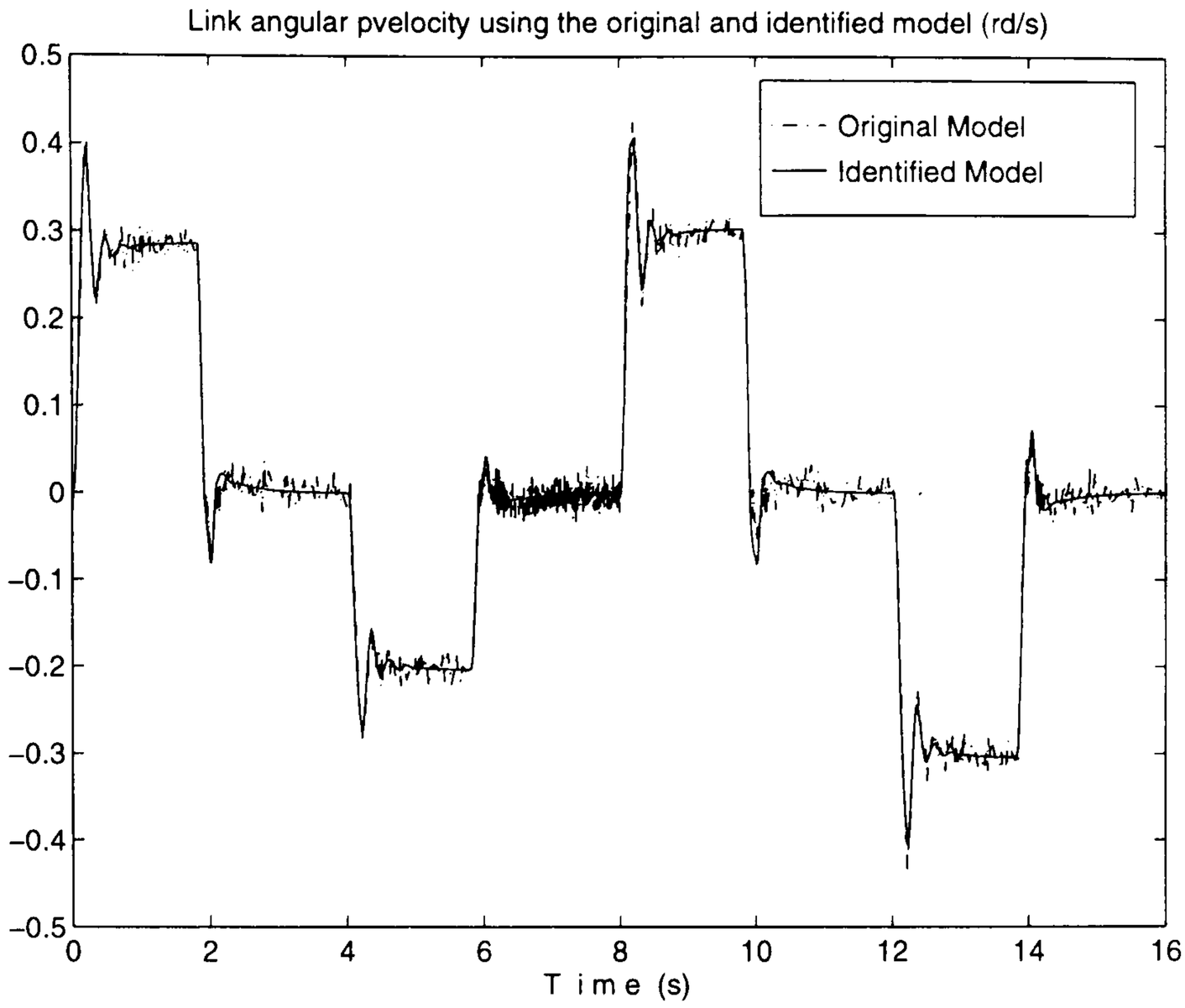


Figure 5.4: The output velocity of the link with gravity loading; the original contains white noise

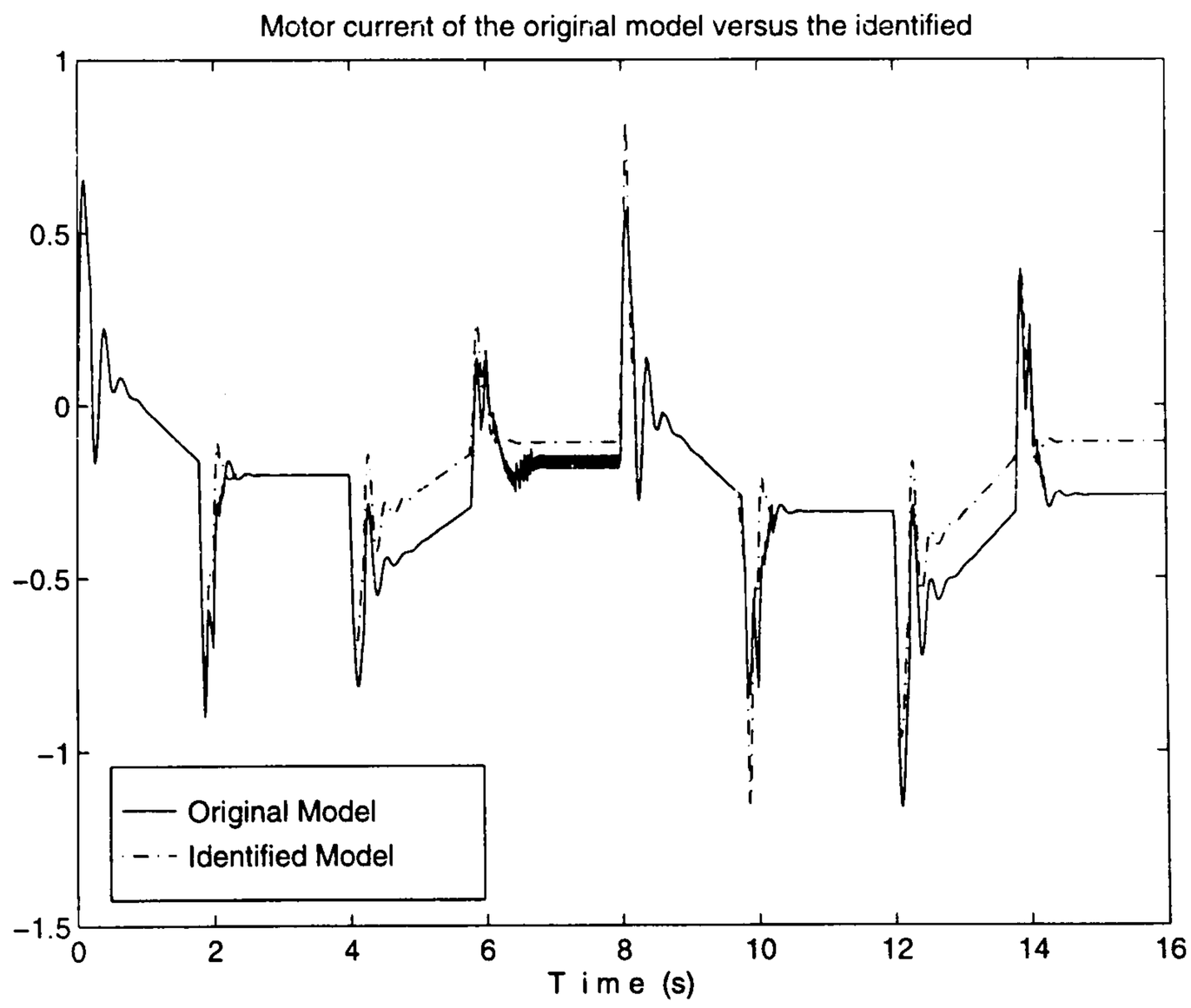


Figure 5.5: The current of the motor with gravity loading using the original and the identied model (simulation)

5.3 Puma560 Model Validation

The identification method described in the previous section is used in an attempt to identify some dynamic parameters of a Puma 560 manipulator. Measurements were collected from the first three links of the manipulator using the constructed instrument described in chapter 4. During each measurement, the current and the position of the first three motors are logged and stored while the manipulator is made to move. Although the data were collected from the three joints, only one link is made to move during the measurement. Since, the motors are controlled independently, the current and the position from each motor were used to identify the dynamics as if they were independent motor-link pairs. Link 1 used for identification the model with vertical axis whereas link 2 and 3 used the model with gravity loading. The identified parameters are tabulated in table 5.2. The motor torque constant used is $k = 0.2587$ obtained from direct measurement from joint 1 motor on test bench using a spring balance.

Table 5.2: The estimated values of the Puma first 3 links

Parameter	Link 1	Link 2	Link 3
$I\,(kgm^2)$	1.1089	0.0786	0.0123
$B\,(Nm\,s/rad)$	1.1565	0.1273	0.0015
$Mgl\,(Nm)$	-	74.9081	0.3371
$f_1\,(Nm)$	57.6922	4.5623	0.00
$f_2\,(Nm)$	85.9568	-12.5873	20.1496

By examining the estimated inertia values of table 5.2 and comparing them with the corresponding published values from table 5.3 they seem very far apart and therefore can not be relied on. The reasons for non convergence are thought to be several factors, summarised, by stating that the interactions between the links are too large to consider just like noise, especially during the estimation process. Secondly, when the measurements are taken, the non moving links should

be prevented from moving by blocking them rather than let the controller keep them at their constant position during the measurement. The author did not have access again to the manipulator to perform more measurements with links blocked to avoid interactions and identify the parameters. To verify that the stated reasons are enough to cause the estimation to produce incorrect values, more simulations similar to the one from the previous section were conducted. A two link manipulator model was created with the first link characteristic exactly similar to the model of the previous section and different parameters for the second link. In a simulated move, the first link was made to follow a given trajectory while the second was made to stay at its zero position. The current of the first motor, the position and velocity of the link were recorded and used in the estimation routine to estimate the first link parameters. Although, an ideal case was assumed with no noise, the routine has failed to estimate the correct values. The currents/torque produced to overcome the reactions to link 2 dynamics were confused with those caused by link 1 dynamics. The estimation routine considers the portion of the torque due interaction as if it was produced by the controller to overcome part of friction and actuate part of the inertia and hence their estimated parameters do not reflect the real system.

Even when the estimated inertias are acceptable a typical move of the Puma manipulator involves more inertia parameters which are not identifiable using the above method. Masses and inertias for manipulators with a complex shape such as that of the Puma are better obtained from measurements performed on the links. Measuring the inertias requires the use of indirect measuring techniques [1] and these can be extremely difficult, in which case using a geometric modelling programme is the ideal solution, as shown by the example of section 3.8. This method provides more information regarding the centre of gravity, inertias about different axes as well as masses.

The friction values of table 5.2 also have to be rejected and a different approach

has to be taken to determine their values.

5.3.1 Friction Coeffecient Determination Through Tuning

The inertial parameters of the Puma 560 have been published by many authors such Armstrong *et al* [1] and Corke *et al* [70]. The inertia parameters from [1] shown in table 5.3 were used in simulation by making the model follow the same moves the real Puma was performing during the measurement. The dynamic model used includes only the first three link of the manipulator and the three last links constituting the wrist were considered as a mass fixed to the end of the third link.

Table 5.3: The Puma parameters for the first 3 links from Armstrong [1]

Parameter	Link 1	Link 2	Link 3	Link 3 with wrist
$I_{xx}(kgm^2)$	-	0.130	0.066	0.192
I_{yy}	-	0.524	0.0125	0.0154
I_{zz}	0.35	0.539	0.089	0.212
I_{motor}	1.14	4.71	0.83	-
Centres of Gravity				
$r_x\,(m)$	-	0.068	0.0	0.0
r_y	-	0.006	-0.070	-0.143
r_z	-	-0.016	0.014	0.014

The links viscous and Coulomb friction were changed and their effect on the input and output (current and position) of the model were observed and compared to those measured from the manipulator. Since this exercise required the repeated change of the parameters and runung of simulation the more flexible software has to be used. DADS and AUTOLEV/SIMULINK produce identical simulation result as shown in figure 5.6 from the same example, however the latter's simulation run time is extremely faster.

SIMULINK was chosen for simulations, since the AUTOLEV generated

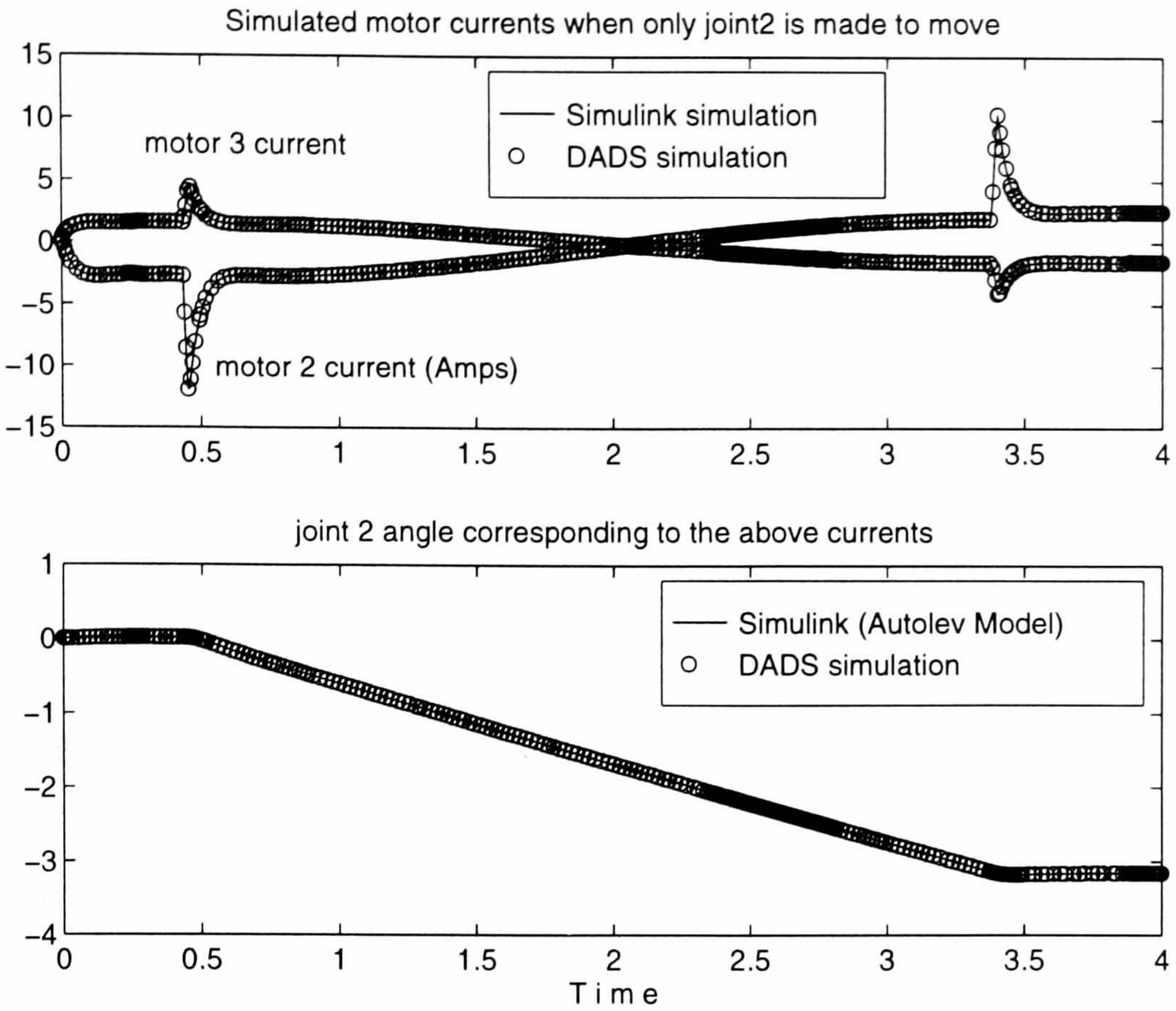


Figure 5.6: Identical simulation results from DADS and SIMULINK using the Autolev generated model: Puma 560 Manipulator

model is easily changable and the AUTOLEV to MATLAB interface shown in appendix A.6 produces and compiles the necessary files for the simulation quickly and fully automatically. The SIMULINK configuration for the simulation is shown in figure 5.7.

The Puma controller described in more details in section 4.3 regards the joint motors as independent and forces them to follow the desired trajectory considering all interactions as noise.

The parameter tuning was done link by link starting from joint 1 to joint 3. The parameters estimated in the previous section were used as a start and the Coulomb friction parameter f_1 and f_2 were tuned first. Each time the friction coefficient was changed the simulation was run and the currents and positions are compared with the measured ones and those from the previous simulation. The parameter is then increased or decreased in terms of the comparison until results are close to the measured values. The process is therefore repeated for the viscous friction coefficient and the whole exercise is repeated to cover joint 2 and then

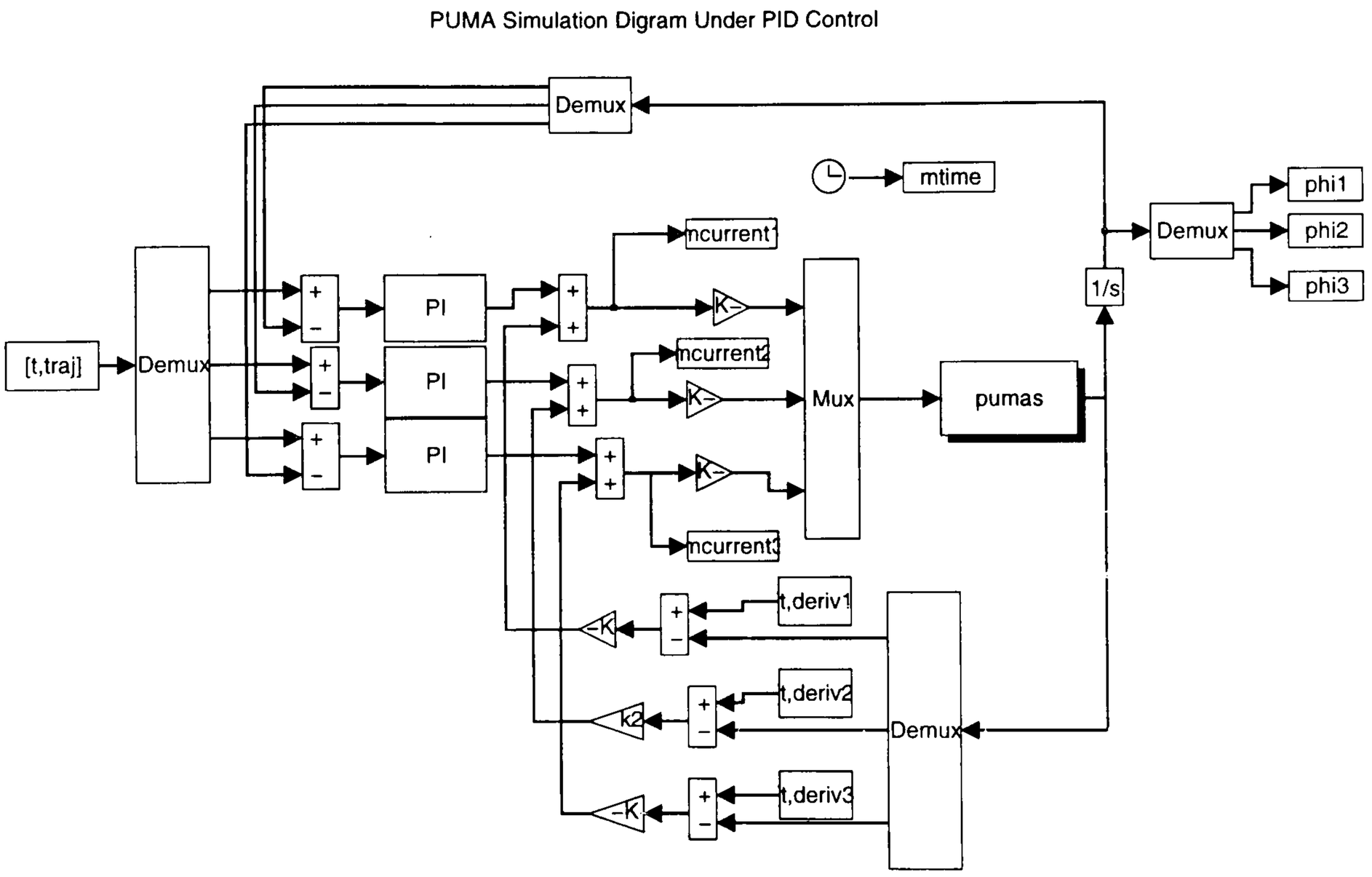


Figure 5.7: SIMULINK's block diagram representation of the simulations of the Puma 560.

joint 3. The final parameters are given in table 5.4 and performance of the whole model is compared with the real puma by plotting the currents and the positions related to each joint.

Table 5.4: The estimated values of the Puma first 3 links through tuning

Parameter	Link 1	Link 2	Link 3
$B\,(Nm\,s/rad)$	8.00	18.00	4.00
$f_1 = f_2\,(Nm)$	1.0	0.5	0.01

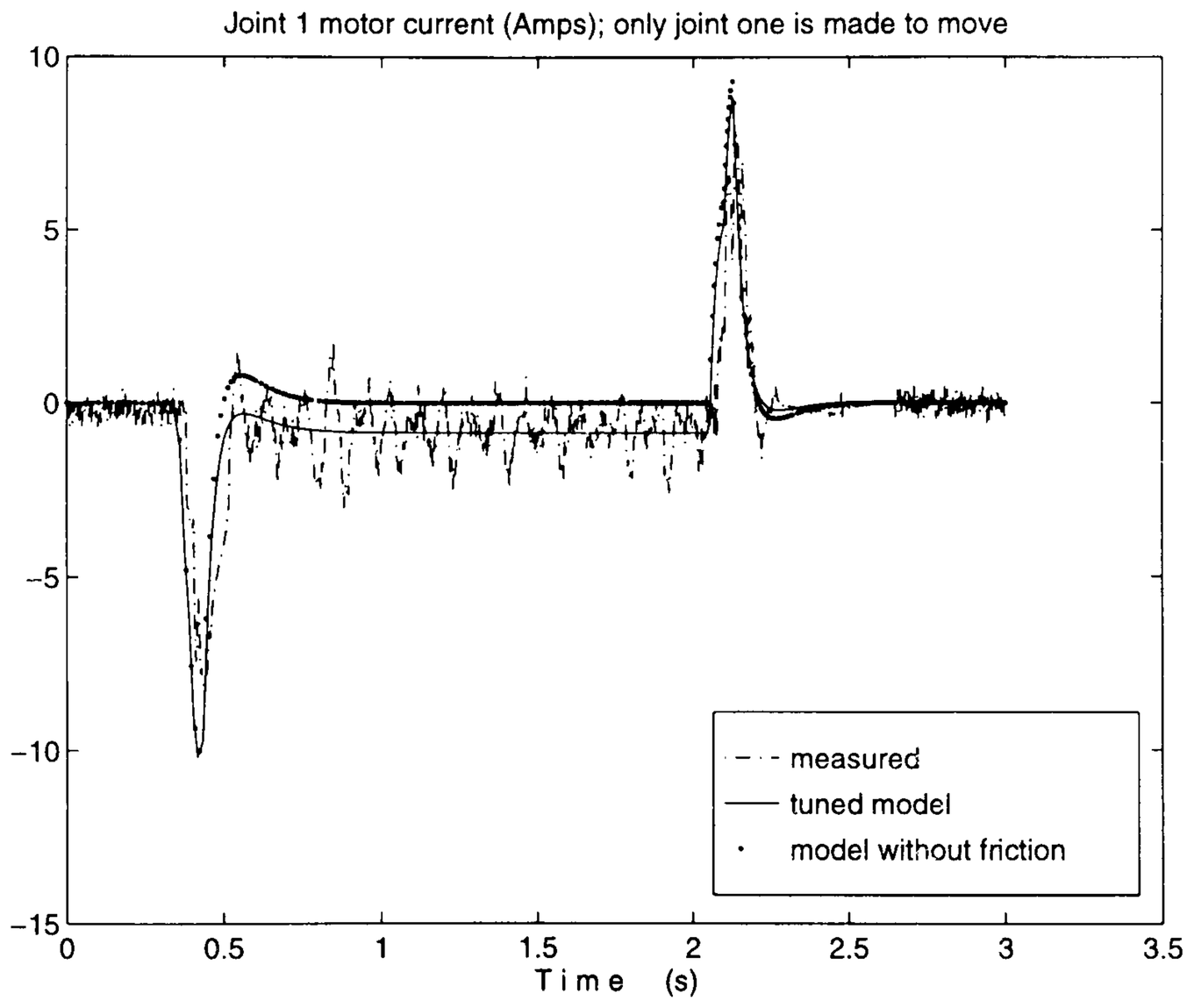


Figure 5.8: Motor 1 current from measured, tuned and without friction model

5.3.2 Discussion

By observing measurements and simulation results from the test when only link 1 was made to move, the current/time graphs of figure 5.8 show how the first motor current is affected by improving the friction parameters. For the same test, figure 5.10 and 5.11 show that changing joint 1 friction parameter values have a

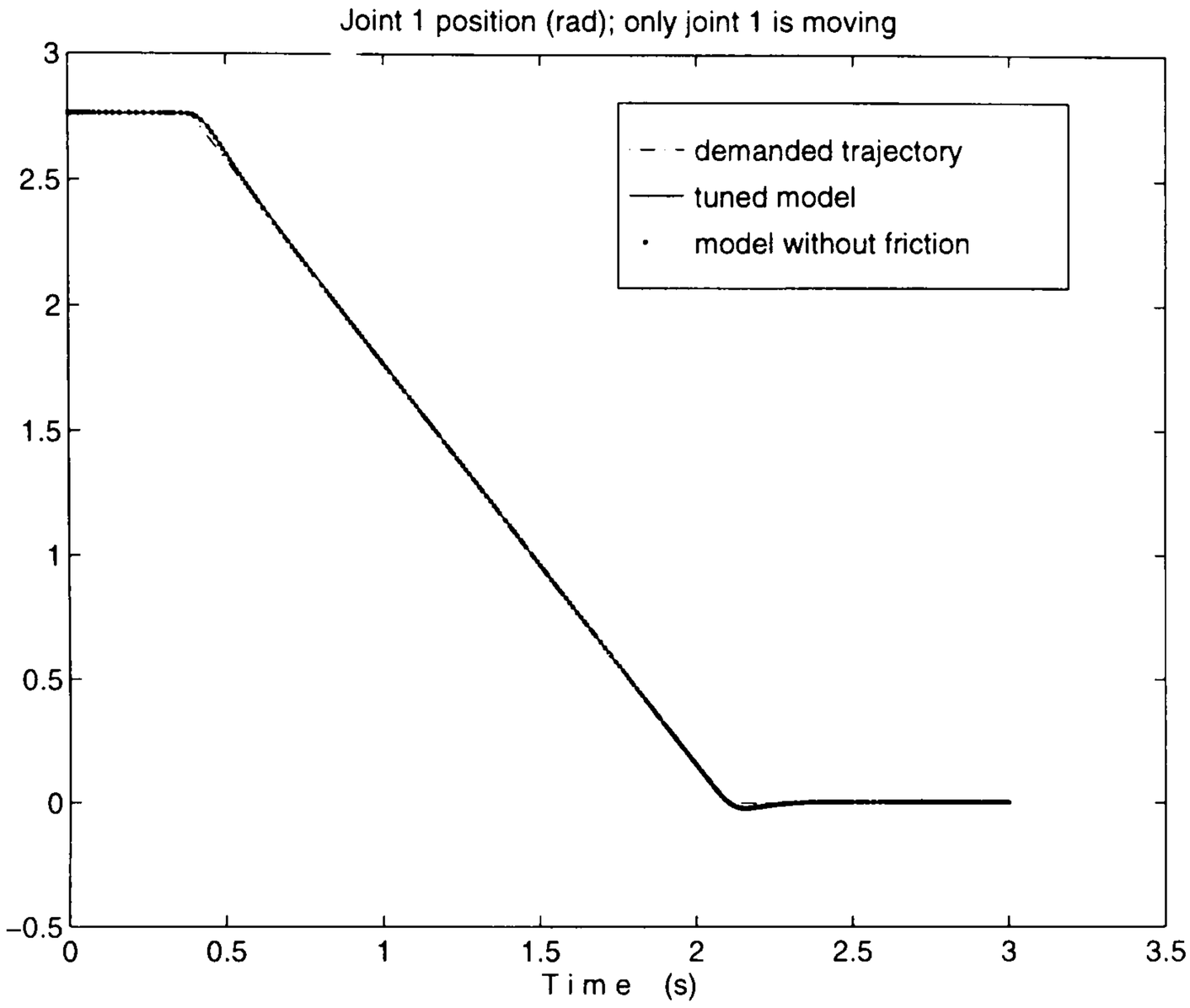


Figure 5.9: Joint 1 angular position demanded, tuned and without friction model

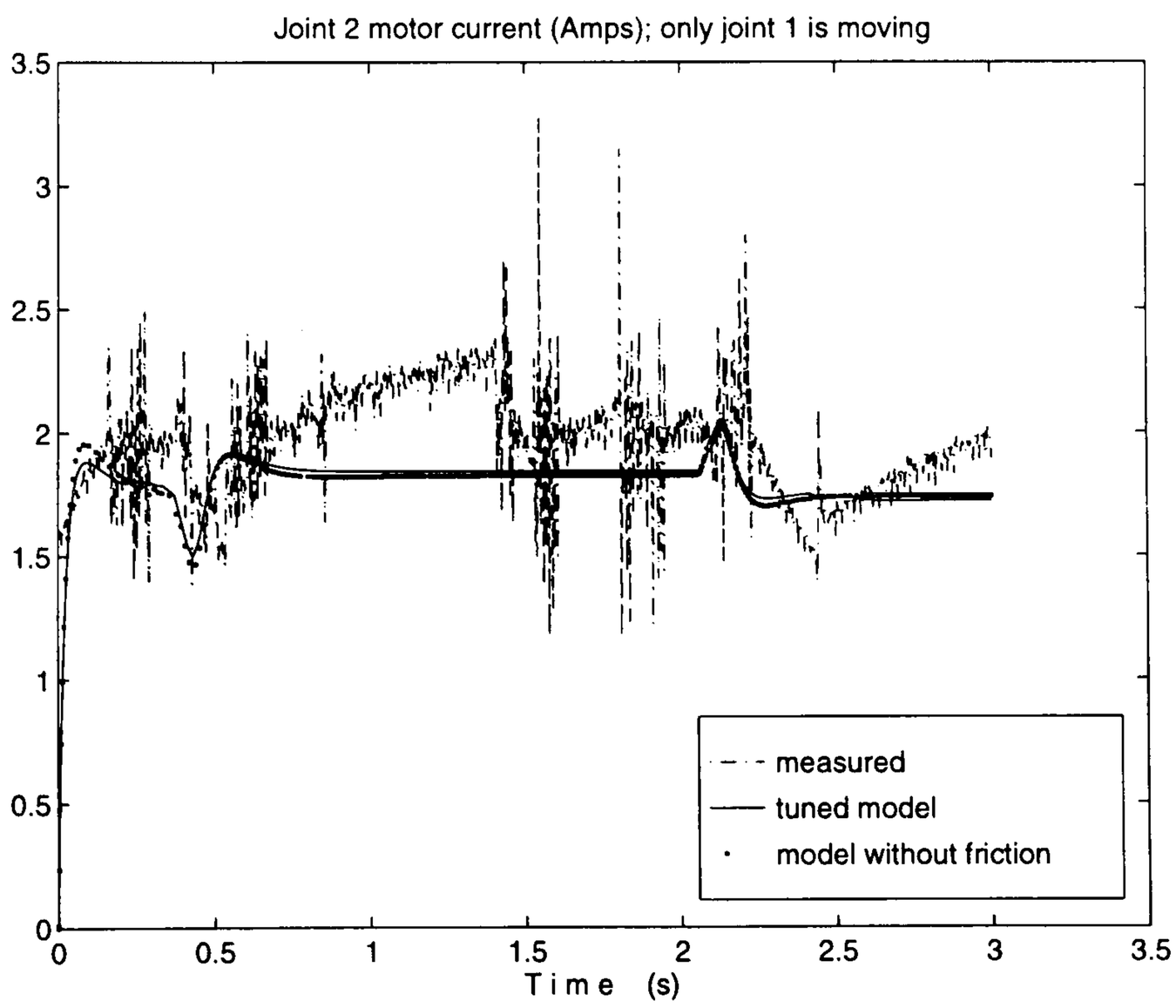


Figure 5.10: Motor 2 current measured and from tuned and without friction model

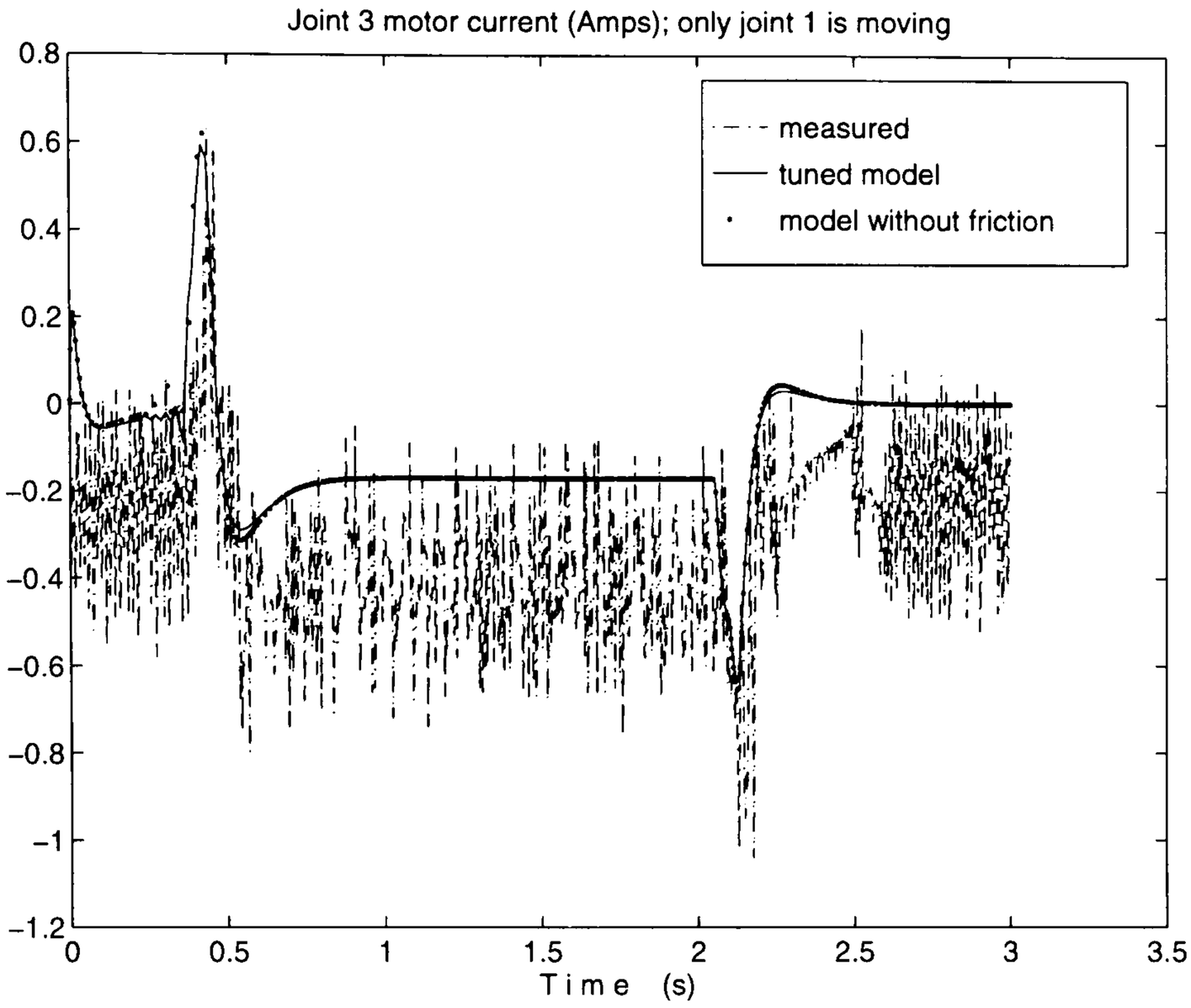


Figure 5.11: Motor 3 current measured and from tuned and without friction model

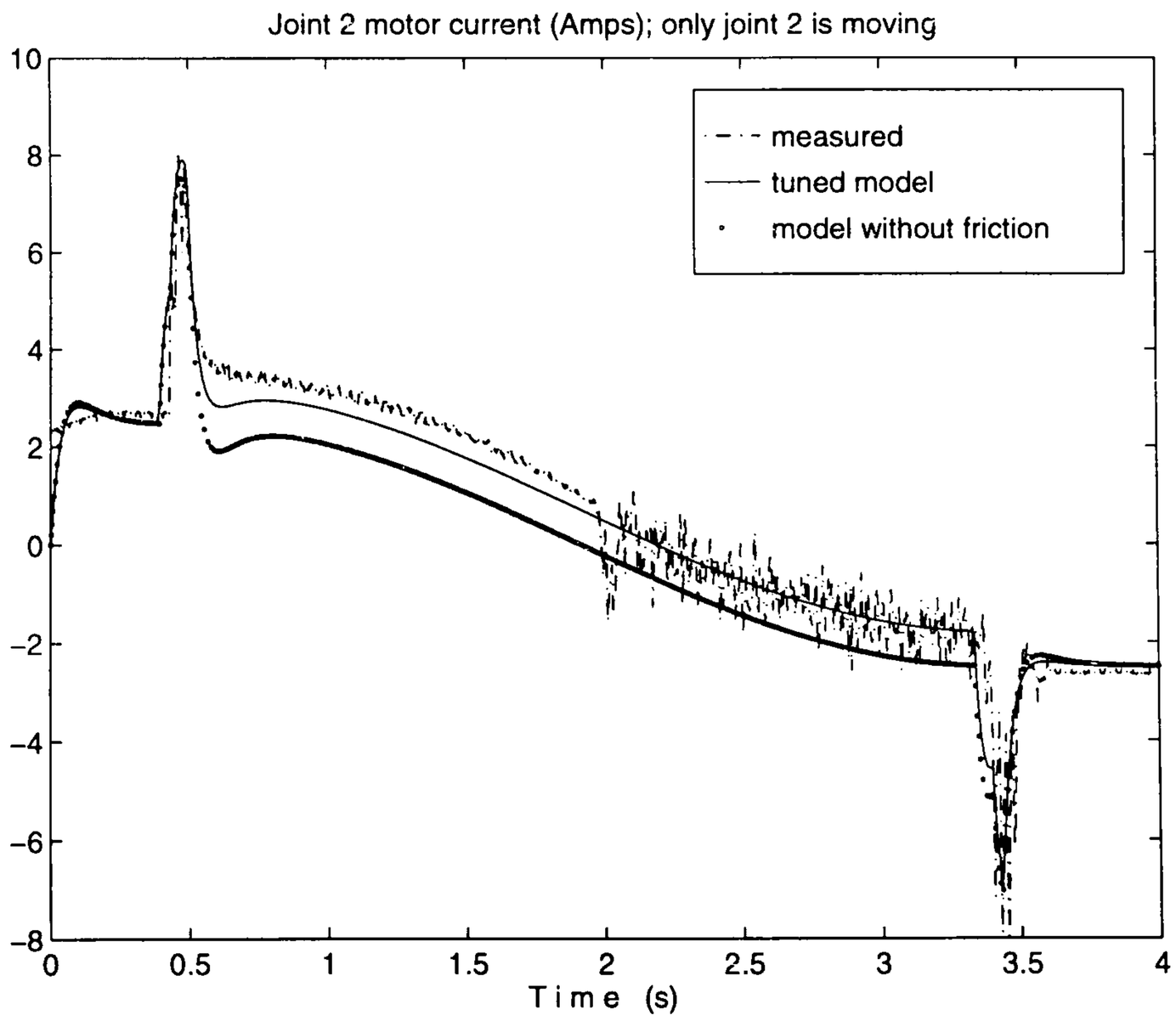


Figure 5.12: Motor 2 current measured and from tuned and without friction model when only joint 2 is made to move

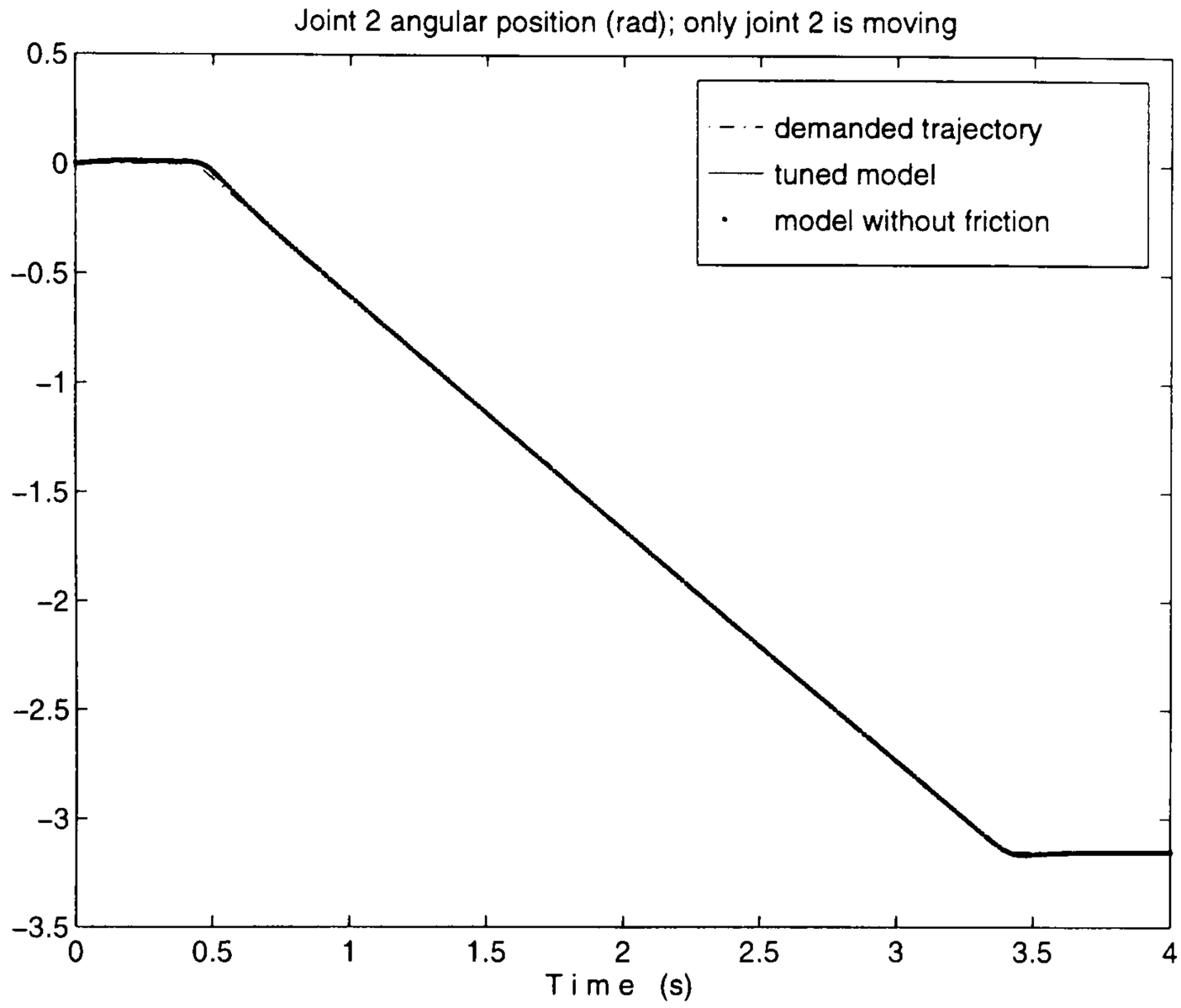


Figure 5.13: Joint 2 angular position demanded, tuned and without friction model

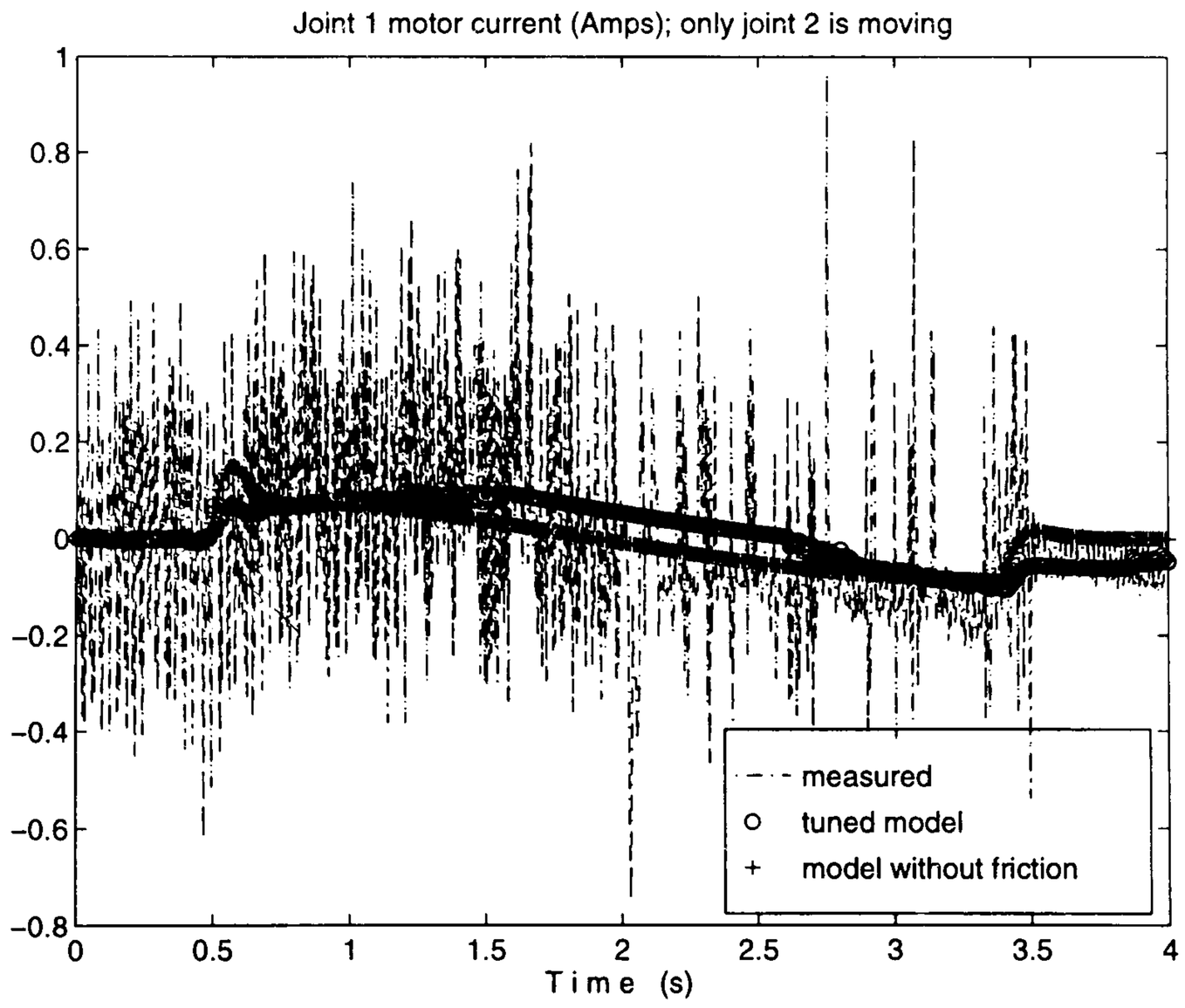


Figure 5.14: Motor 1 current measured and from tuned and without friction model when only joint 2 is made to move

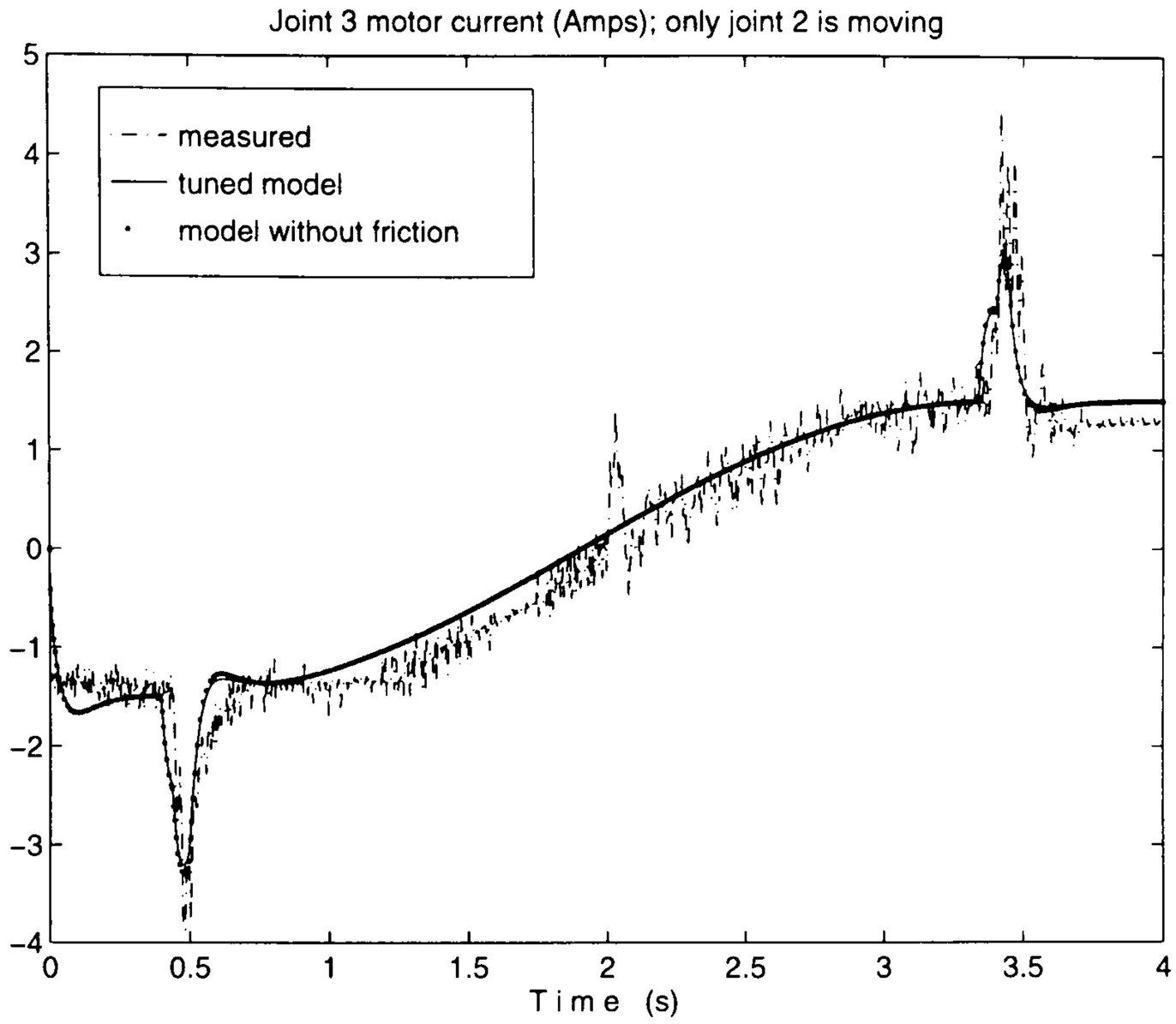


Figure 5.15: Motor 3 current measured and from tuned and without friction model when only joint 2 is made to move

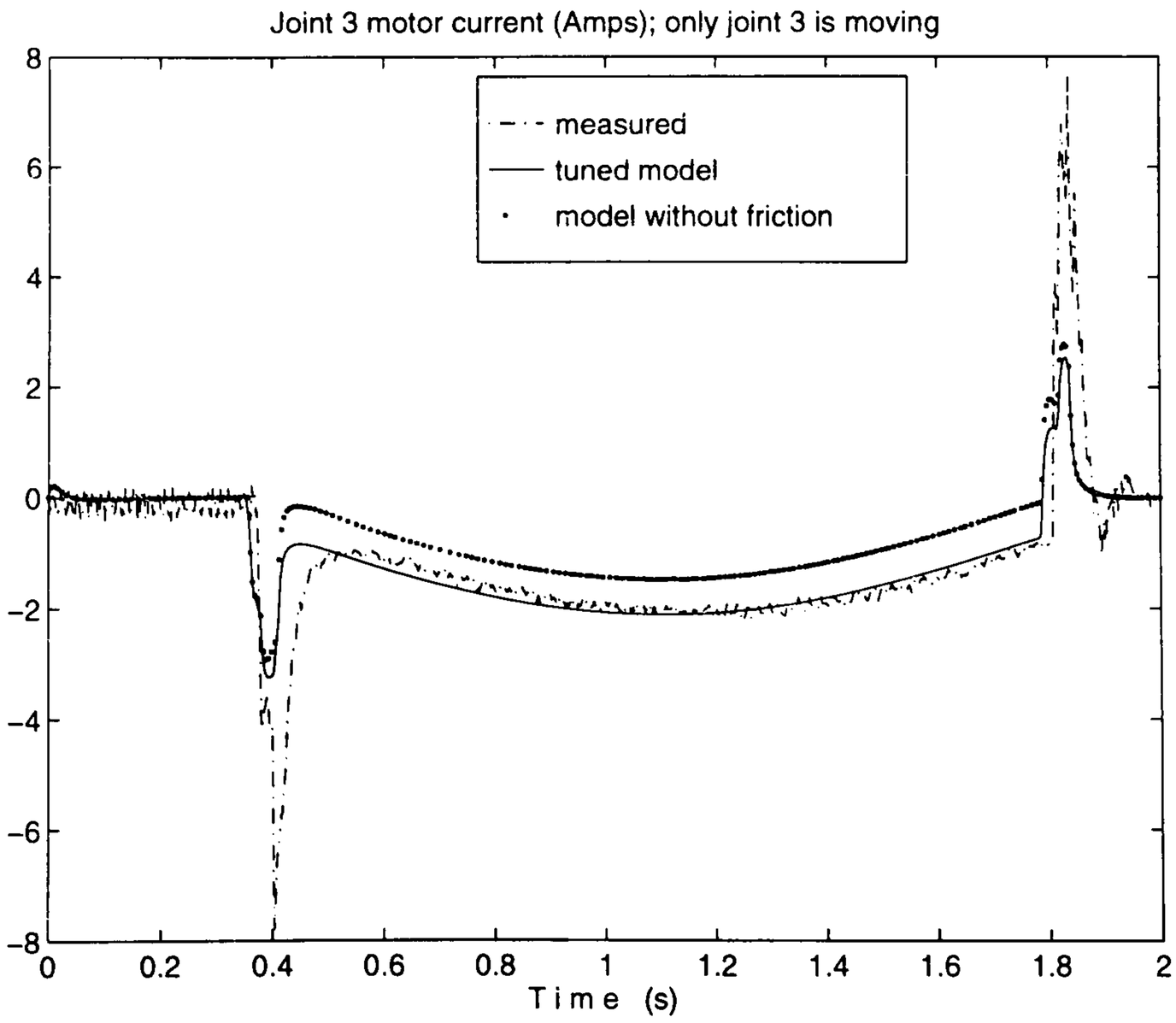


Figure 5.16: Motor 3 current measured and from tuned and without friction model when only joint 3 is made to move

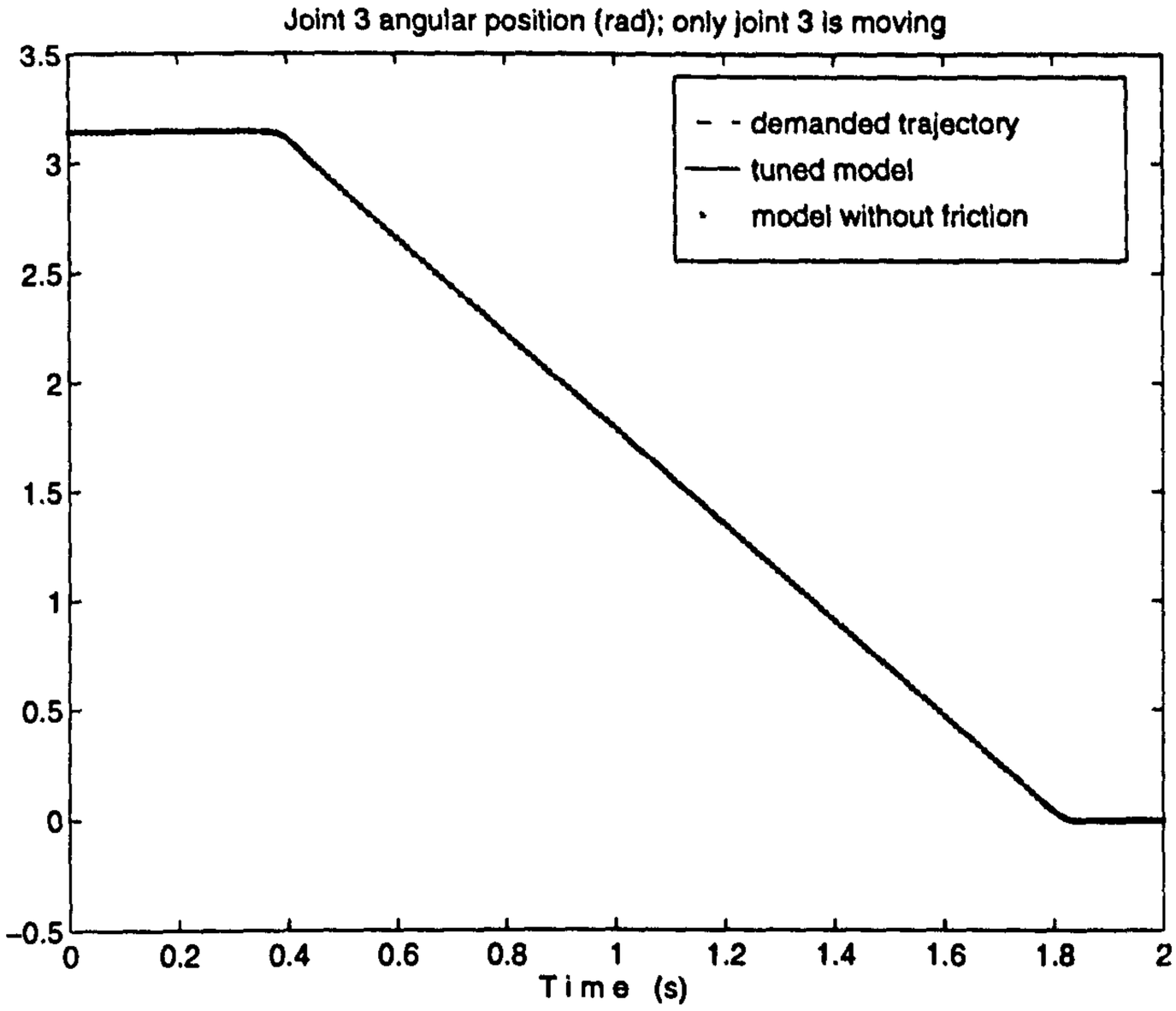


Figure 5.17: Joint 3 angular position demanded, tuned and without friction model; only joint 3 is made to move

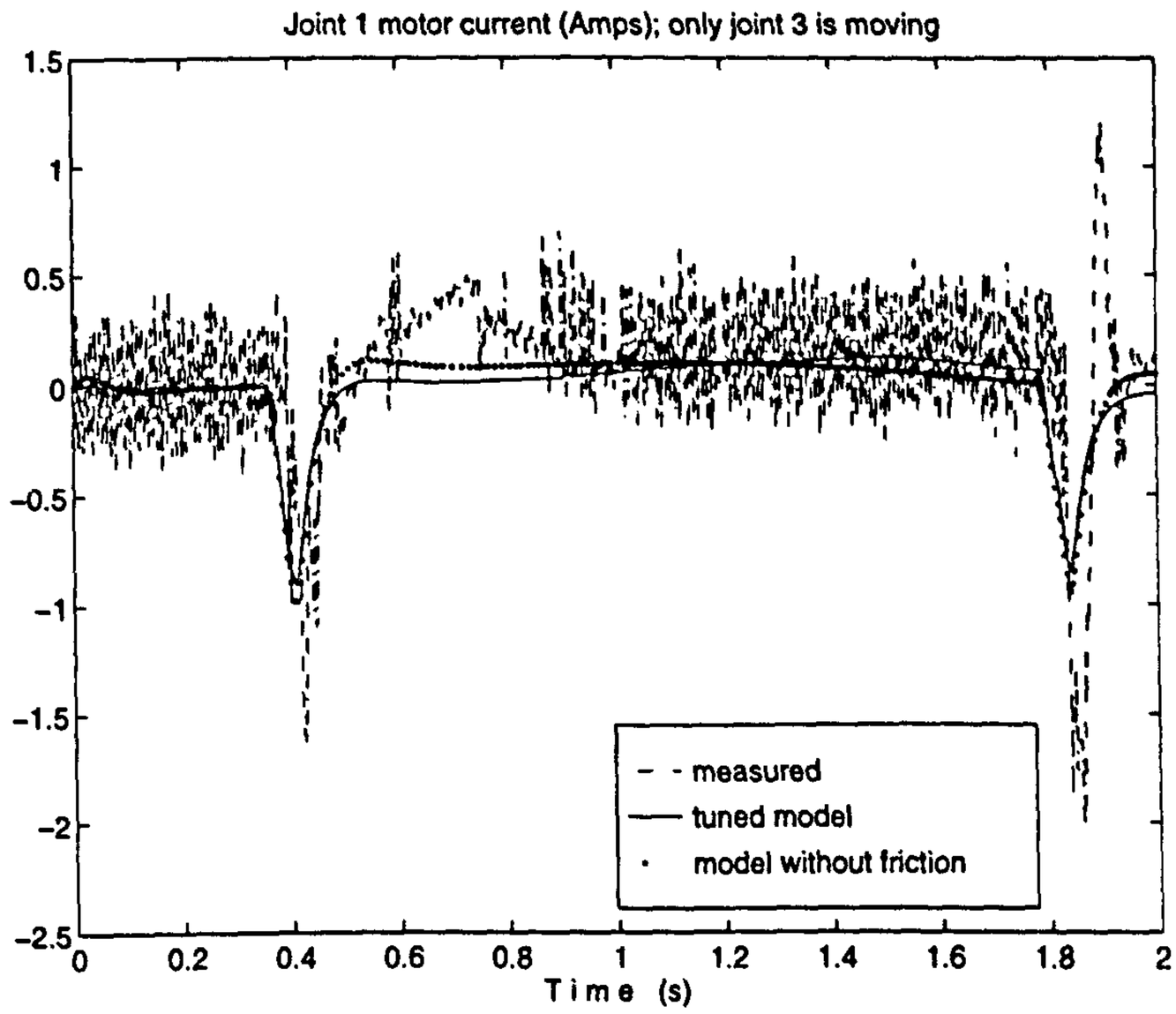


Figure 5.18: Motor 1 current measured and from tuned and without friction model when only joint 3 is made to move

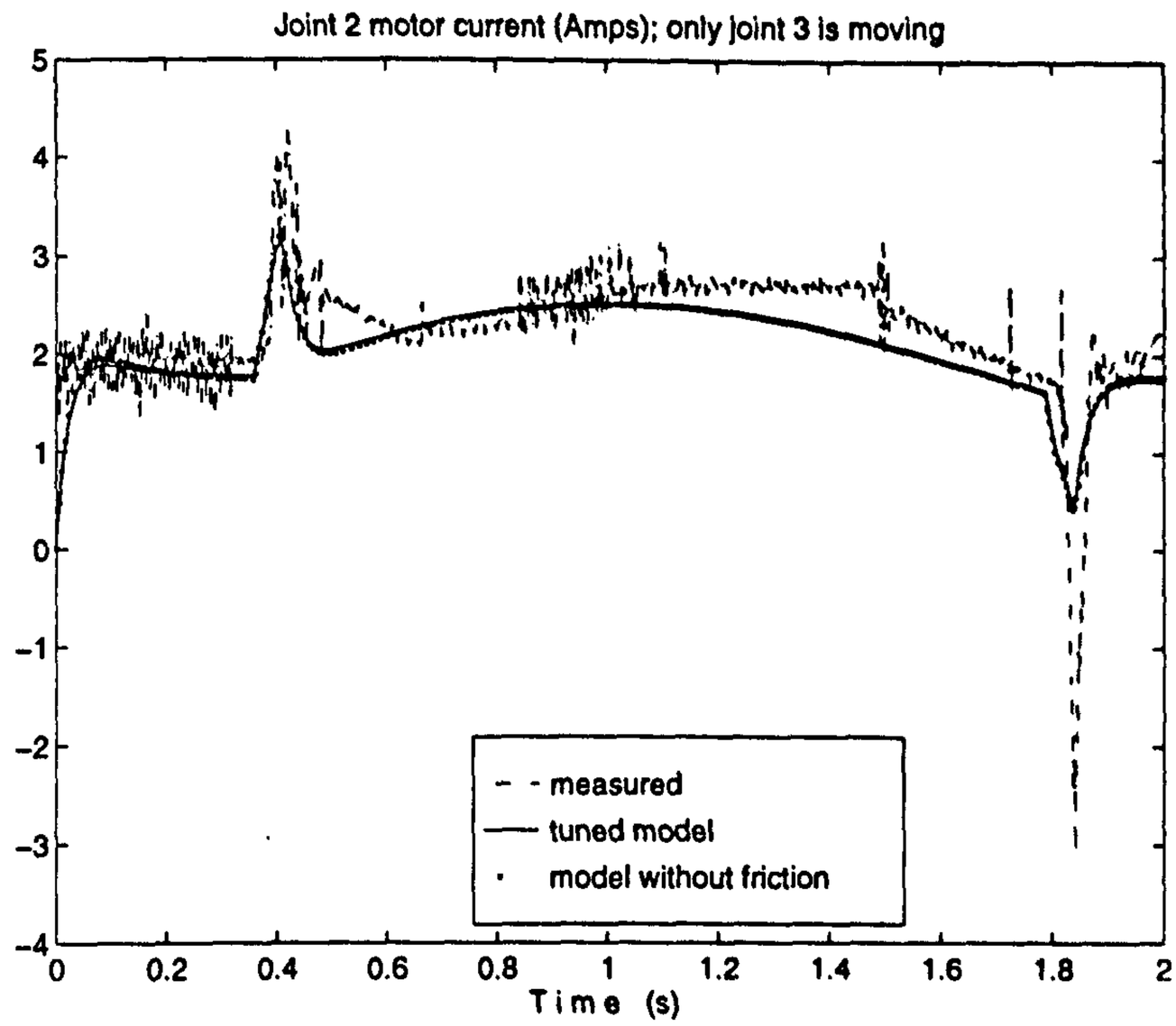


Figure 5.19: Motor 2 current measured and from tuned and without friction model when only joint 3 is made to move

smaller effect on joint 2 and 3 motor currents. Figure 5.9, on the other hand, shows that joint 1 desired angular position is achieved, with little effect from changing the friction parameter values. Similarly, figure 5.12 and 5.16 show that the friction parameter values have considerable effects on joint motor currents of link two and three only when these are moving. The motor currents of the two other stationary links during each simulation are affected very little by the tuning of the friction of the moving link, as shown in figures 5.14, 5.15, 5.18 and 5.19. The achieved position of each link also seems to be affected very little by tuning the friction parameters as shown in figure 5.13 and 5.17. where the desired positions are attained. This remark agrees with results in [6] in that although modelling joint friction slightly improves the accuracy, it is not critical for closed loop control as is the case of the Puma.

Figure 5.12 and 5.15 reveal the presence of an important gear backlash, in the measured current curves. A current jump in the middle appears when the second link passes by the vertical up position, the controller attempt to compensate

for the fast move in the gear from one limit to the other of the backlash. The interaction is very clear on joint 3 motor even when this one is not moving.

The backlash is not included in the model and the friction characteristics of the Puma manipulator are thought to be of a more complex form, however the simulations show that the model achieves the desired position similarly to the real manipulator. The model in its current form can therefore be used to predict the movement of the manipulator in a closed loop scheme controller and hence is valid for positioning purposes within the same context.

5.4 SCARA type Model Validation

In this section a methodology for the estimation of SCARA robot model parameters and model validation is proposed. The SCARA type manipulators have three links, the two first are rotational with vertical axes and the third moves along a prismatic joint up and down as illustrated in figure 5.20 . The method for system identification described above, unlike for the Puma, would be effective to estimate the relevant inertial and friction parameters. In SCARA type manipulator case, only the inertias about the vertical axis are relevant to the dynamic model and there is no interaction between the third and the first two links. A set of dynamic equation describing a SCARA manipulator are given in appendix C.2.

The identification would be performed from the outer link to the inner one in the folowing squence.

1. First, only the third link is made to move in up and down and measurements are collected during the move. A model for the third link is therefore used with the measured data to estimate its parameters; mass and friction parameters. The third link can be considered as point mass attached to the end of the second link.

2. The first link is blocked at a given position and the second link is made to

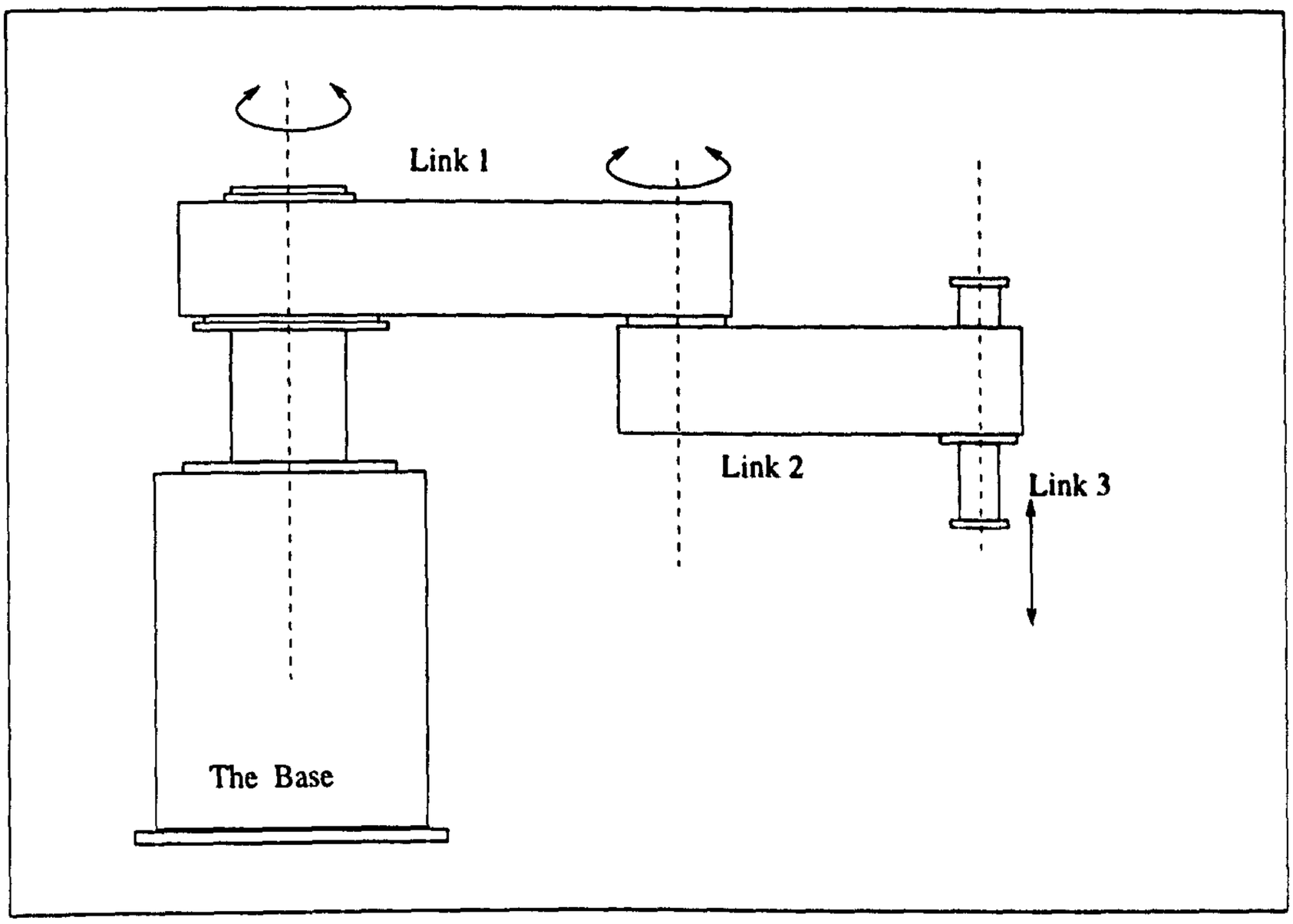


Figure 5.20: SCARA type manipulator schematic representation

move and measurments are collected during the move. The measurments are used then with a motor-link model to estimate the inertia and the friction parameters. The inertia obtained is about joint 2 axis and contains the third link contribution. Link 2 inertia about its axis of rotation can be isolated using link 3 mass and the link 2 length.

3. The second link is blocked stretched out, usualy is the zero position and prevented from any movement relative to link 1. Link 1 is then made to move and measurements are collected from it and similarly to the previous step the inertia and friction parameters are estimated. Link 1 inertia about its axis could be extracted from the estimated one using its length and link 2 inertia and mass.

4. the estimated values are then used in simulation to perform similar moves to the real manipulator and the achieved performance are compared. The model is therefore accepted or rejected in terms of the comparison with the real system.

5.5 Conclusion

In this chapter the model validation is emphasised to be a very important part of manipulator model development. The model has to be validated for the purpose it is intended for. The availability of low-cost high power computing facilities should make validation an integral part of manipulator model validation.

Simple one motor-link model can be used for parameter estimation and validation of more complex manipulator systems.

Gravity loading parameters can also be included in the motor-link model and estimated with the other parameters. This has been shown possible through a simulation example.

It has been shown that the model does not have to be an exact mathematical description of the physical system (manipulator), but rather a **valid** description within the context of use of the manipulator. Although the Puma manipulator is thought to have a complex friction model, a simplified one has been shown to be enough for positioning purposes. The final model can then predict the movement of the Puma 560 manipulator and could be used for instance for the design of suitable positioning controller. The presence of high gear ratios allowed the controller to overcome the interaction between the links.

The model parameters of SCARA type robots are thought to be less difficult to estimate and a methodology is proposed to facilitate this task.

The results show that, it is possible to tune the model parameters to produce an acceptable model. This is made possible because AUTOLEV produced a valid description of the Puma manipulator and the AUTOLEV to MATLAB interface allowed fast parameter tuning.

Parameter tuning can be done by automatic means, such as the model distortion method which has been used in the nuclear field [67]. The investigation of the use of this method in robotic manipulators model validation is included in the list of future work.

In conclusion, although it has been shown that the dynamic model of the Puma does not have to be the exact mathematical description and the one developed in this chapter is a valid description for the purpose of joint positioning, the end-effector position of the manipulator is still not at the right expected location. This demonstrates that developing a valid dynamic model and performing dynamic model analysis of a given manipulator does not necessarily solve all its positioning performance problems. The manipulator geometrical parameters errors which seem to have smaller effects on the dynamics of the manipulator could cause deficiencies that cannot be ignored on the kinematic front. The issue of kinematic model deficiencies and their improvement is treated in the next chapter.

Chapter 6

Kinematic Model Identification

6.1 Introduction

The relationship between the kinematics and the dynamics of robot manipulators is often overlooked when considering accuracy and performance. A good dynamic arm control ensures the achievement of the demanded position through a predetermined path for each link of the manipulator. However, if the robot controller internal kinematic model used for the calculation of the target position is inaccurate, the achieved robot manipulator position will then be incorrect. Therefore, both dynamic and kinematic aspects should be considered for the purpose of improving the performance and accuracy of a robot manipulator arm. When measured data are involved for the purpose of model validation, or to be fed back to the dynamic controller, it must be verified that they convey the exact values of the measured information. Consequently, accurate kinematic models are essential for the off-line kinematic programming. This chapter describes the development of a method for identifying the Denavit-Hartenberg (D-H) kinematic model parameters of serial link robot manipulators based on the application of the methodology described in the work of Stone *et al* [11] where joint *Features* were introduced and used for the identification of the *S-Model* parameters. Therefore,

the issues dealt with in this chapter are kinematic modelling and kinematic model identification.

The inaccuracies of the positioning of the end-effector of industrial robot manipulators are generally due to inaccurate knowledge of the kinematic functional relationship between the tool and the reference frame of the manipulator, known as *the kinematic model*, and to errors encountered when relating the base of the manipulator itself to the world reference.

At this point it is necessary to differentiate between two categories of inaccuracies. The first is observed when the taught positions of the arm are not achieved after one or more manoeuvres of the robot arm. These errors are related to the repeatability of the industrial robot and are usually very small. The repeatability is usually specified by the manufacturer as a quality of the manipulator, and has been addressed specifically in the work of Mooring et al [71]. The second category of positioning errors is encountered when the actual end-effector location differs from that programmed through off-line programming. Although repeatability is affected mainly by non-geometric sources such as encoder resolution, gear backlash and link and joint stiffness, these, together with other geometric errors discussed in [10] [72] [73] are also contributing factors to the accuracy. Improving the positioning accuracy of industrial manipulators is generally achieved through two methods. The first tackles the source of the problem itself, by imposing tighter tolerances during the manufacture of the robot arm components. This prevention is proven to be the expensive option and has to be decided on at the design stage. Consequently, it is not applicable to already manufactured manipulators. The more popular solution is to introduce changes to the robot positioning software by compensating for the errors after the identification of their values.

There has been a considerable volume of work in last decade on the subject of improving the positioning accuracy of industrial robot manipulators, much

of which was reviewed by Roth et al [10], mainly under titles with key words such as, calibration, accuracy improvement, parameter estimation or accuracy compensation. This work was the subject of many papers presented by Hayati [73] [74], Whitney et al [23], Chen et al [72], [75] Stone et al [11] [24], Stanton et al [25], Khalil et al [26], Driels et al [76] [77] [27] [78] [79] and Abderrahim et al [80]. Many of these authors considered only geometric errors as the main error source, with the exception of Chen et al [72] and Whitney et al [23], who included explicitly non-geometric errors as well. Generally the number of errors described is different depending upon the models employed by the different authors. Although some used particular models [23] the majority introduced models that are universally valid and widely used, such as the *Denavit-Hartenberg* or modified versions of it [26] [27] [81] and [82] [78]. Validation of the methods and real measurements were reported in some works, especially the latest papers which include [72] [23] [24] [25] [26] [27] [28] and [29]. A variety of measurement techniques were used which included visual and automatic theodolites [23] [28] [29], acoustic sensors [24], laser tracking devices [25] and coordinate measuring machines [30]. Some other reported work was based on computer simulations.

Some of the authors presented an estimation approach where the whole set of parameters are calculated simultaneously, while others split the estimation into sub tasks to cover individual links one by one [24], [25]. Stone [24] has developed a new model, *The S-Model*, and a general identification method to estimate the S-Model parameters from which the D-H parameters are extracted. The D-H Model was also described as not amenable to direct identification in [24]. However, with small modification, it was used successfully for calibration in several papers as stated above. The physical explicit significance of the D-H model parameters has led to its popularity and widespread use in robot control and justifies the importance of establishing the real D-H parameter values.

This work describes the identification of the D-H kinematic model parameters

by applying the methodology introduced by Stone without the intermediate step of identifying the S-Model parameters.

The position of the End-effector caused by the movement of **only** one joint axis is measured, and the process is repeated for each joint in an inward sequence, starting from joint n and ending at joint 1. In a similar sequence the measured Cartesian positions of the end-effector are then used for the estimation of each link (joint) *features*, introduced in [24]. These are in turn used for the estimation of link parameters. Despite the multiple methods that could be used to identify the features, the simplest and more intuitive method presented in [24] is used.

The next section reviews a number of kinematic models based on the assignment of a coordinate frame to each link of a manipulator , and section 6.3 describes the identification method. Section 6.4 presents an illustration of the identification method and section 6.6 concludes the chapter.

6.2 Coordinate Frame kinematic Models

Coordinate frame kinematic models are based upon the assignment of Cartesian coordinate frames fixed relative to each link of the robot arm. The position and orientation of the coordinate frames on the links vary from one type of model to another. The spatial transformation between two successive frames is a 4×4 *homogeneous* matrix, introduced first by Denavit and Hartenberg [22] and adopted by Paul [37] and others. The transformation matrix is a function of the type of the model, the type and position of the joint which connects the two links together. The D-H homogeneous transformation matrix has become the most common approach to describing robotic spatial transformation and it shares its

general form with other coordinate frame kinematic models and is given by:

$$\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.1}$$

which also could be written in terms of its vector components,

$$\begin{bmatrix} \vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}} & \vec{\mathbf{p}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

where

$$\vec{\mathbf{n}} = [n_x\ n_y\ n_z]^T, \tag{6.3}$$

$$\vec{\mathbf{o}} = [o_x\ o_y\ o_z]^T, \tag{6.4}$$

$$\vec{\mathbf{a}} = [a_x\ a_y\ a_z]^T, \text{and} \tag{6.5}$$

$$\vec{\mathbf{p}} = [p_x\ p_y\ p_z]^T. \tag{6.6}$$

The vectors $\vec{\mathbf{n}}$, $\vec{\mathbf{o}}$, and $\vec{\mathbf{a}}$ are the unit vectors of the second coordinate frame in terms of the first, whereas $\vec{\mathbf{p}}$ is the position of the origin of the second coordinate frame relative to the first one. Paul [37] has presented several mathematical properties of the above homogeneous transformation matrix which are used extensively in robotics.

6.2.1 Denavit-Hartenberg Model

The Denavit-Hartenberg is the most widely used model by the researchers in the field of robotics. It has been adopted due to its explicit physical interpretation of the mechanisms, and the relatively easy implementation in the programming of robot manipulator. The D-H model is presented as 4×4 matrix that results from the following product :

$$T = A_1 \cdot A_2 \cdot \ldots \cdot A_n \tag{6.7}$$

T defines the transformation of the link n coordinate frame into the base coordinate frame of the robot arm, which is chosen according to the D-H convention, coincident with the first link coordinate frame at its zero position. A_i designates the D-H transformation matrix relating frame i to the $i^{th}-1$ frame. The n^{th} link frame coincides with the end-effector's coordinate frame. Figure 6.1 illustrates the spatial relative position of two consecutive links and their associated coordinate frames, and the four parameters which define the spatial transformation between consecutive coordinate frames.

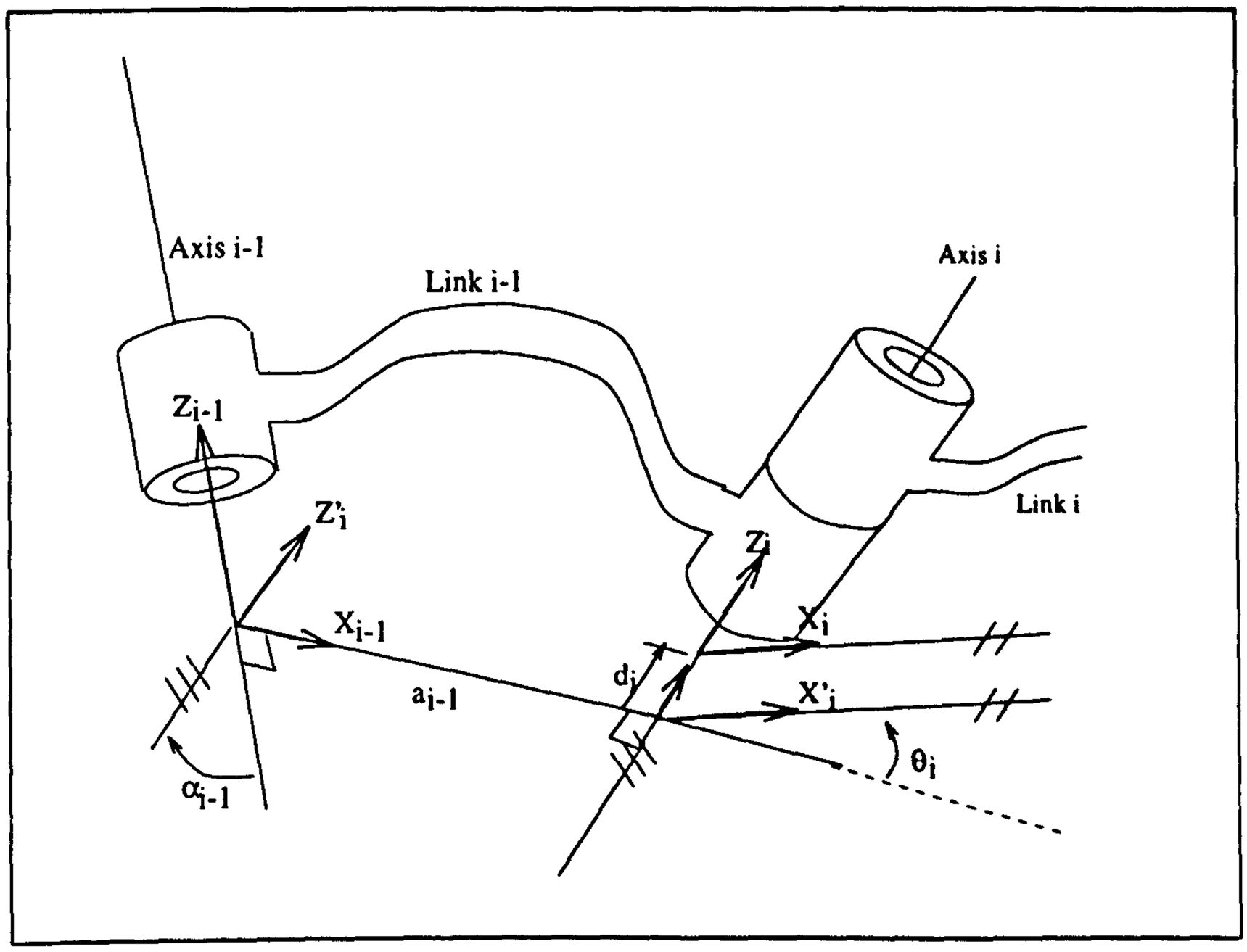


Figure 6.1: Denavit-Hartenberg parameters representation for a rotational joint

As mentioned previously, the spatial transformation between two consecutive links is a function of the joint type that connects them together. It is caused by a number of rotations and translations summarised by:

$$\begin{aligned} A_i = A_i(q_i) \equiv\ & Rot(z,\theta_i)Trans(0,0,\mathbf{d}_i)Trans(\mathbf{a}_{i-1},0,0) \\ & Rot(x,\alpha_{i-1}). \end{aligned} \tag{6.8}$$

$$Rot(x, \alpha_{i-1}). \tag{6.8}$$

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_{i-1} & \sin\theta_i \sin\alpha_{i-1} & \mathbf{a}_{i-1}\cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_{i-1} & -\cos\theta_i \sin\alpha_{i-1} & \mathbf{a}_{i-1}\sin\theta_i \\ 0 & \sin\alpha_{i-1} & \cos\alpha_{i-1} & \mathbf{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.9}$$

6.2.2 Whitney-Lozinski Model

The W-L model was developed by Whitney, Lozinski and Rourke [23] for formulating an approach to the kinematic identification problem. The formulated model is referred to in the literature as Whitney-Lozinski model (W-L) [24]. The complete W-L model includes geometric and non-geometric components. The geometric model is given by :

$$T_n = W_1 \cdot \ldots \cdot W_n, \tag{6.10}$$

where W_i is defined as :

$$\begin{aligned} W_i = W_i(q_i) \quad = \quad & Trans(0, y_i, z_i) Rot(z, \Phi_i) Rot(y, \Omega_i) Rot(x, \Psi_i) \\ & Rot(y, \theta_i), \end{aligned} \tag{6.11}$$

and depicts the transformation between two consecutive links. The complete geometric model includes an extra W' which describes the transformation between the link n coordinate frame and the tool coordinate frame. W_i yields,

$$W_i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & y_i \\ 0 & 0 & 1 & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\Phi_i & -\sin\Phi_i & 0 & 0 \\ \sin\Phi_i & \cos\Phi_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\Omega_i & 0 & \sin\Omega_i & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Omega_i & 0 & \cos\Omega_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} \cos\Psi_i & 0 & \sin\Psi_i & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\Psi_i & 0 & \cos\Psi_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_i & 0 & \sin\theta_i & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_i & 0 & \cos\theta_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.12}$$

The homogeneous transformation W_i is a function of six parameters y_i, z_i, Φ_i, Ω_i, Ψ_i and θ_i. Unlike the four parameters of the D-H model, the six parameters permit a greater degree of flexibility in choosing the locations of the link coordinate frames.

Amongst the six parameters, only one parameter per link is defined as the link generalised coordinate q_i. It is given as the joint angle θ_i for a revolute joint i or given as y_i for a prismatic joint i. The rest of the parameters are constant for either cases.

Although the W-L model parameters have no real physical significance, the model was designed to accommodate non geometric characteristics. These include joint compliance, gear transmission error and backlash, base motion and other effects.

In addition to the complexity of its transformation matrix W_i in comparison with the A_i matrix the W-L model lacks the apparent elegance of the D-H model. In the work presented by Whitney *et al* [23] on the calibration of a PUMA manipulator, the calibration did not cover the encoder reading error and the *zero* position error of the generalised coordinates of the robot manipulator.

6.2.3 The S-Model

The S-Model is a new parametric kinematic model developed by Stone *et al* [11] [24] for the robot arm signature[1] identification to improve the position accuracy. The S-Model is applicable to all rigid manipulators that allow Denavit-Hartenberg type modelling, which makes it a completely general method for describing rigid

[1] Arm signature is the manipulator real, ***unique***, kinematic model parameters.

manipulator kinematics. The S-Model is given by the matrix S_n where:

$$S_n = B_1 \cdot B_2 \cdot B_3 \ldots \cdot B_n, \tag{6.13}$$

which describes the position and orientation of link n coordinate frame relative to the base coordinate frame. The general transformation matrices between consecutive coordinate frames, B_i, are 4×4 homogeneous transformations. B_i is the transformation matrix between $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$ coordinate frames of the S-Model in a similar way to the matrix A_i in the D-H model.

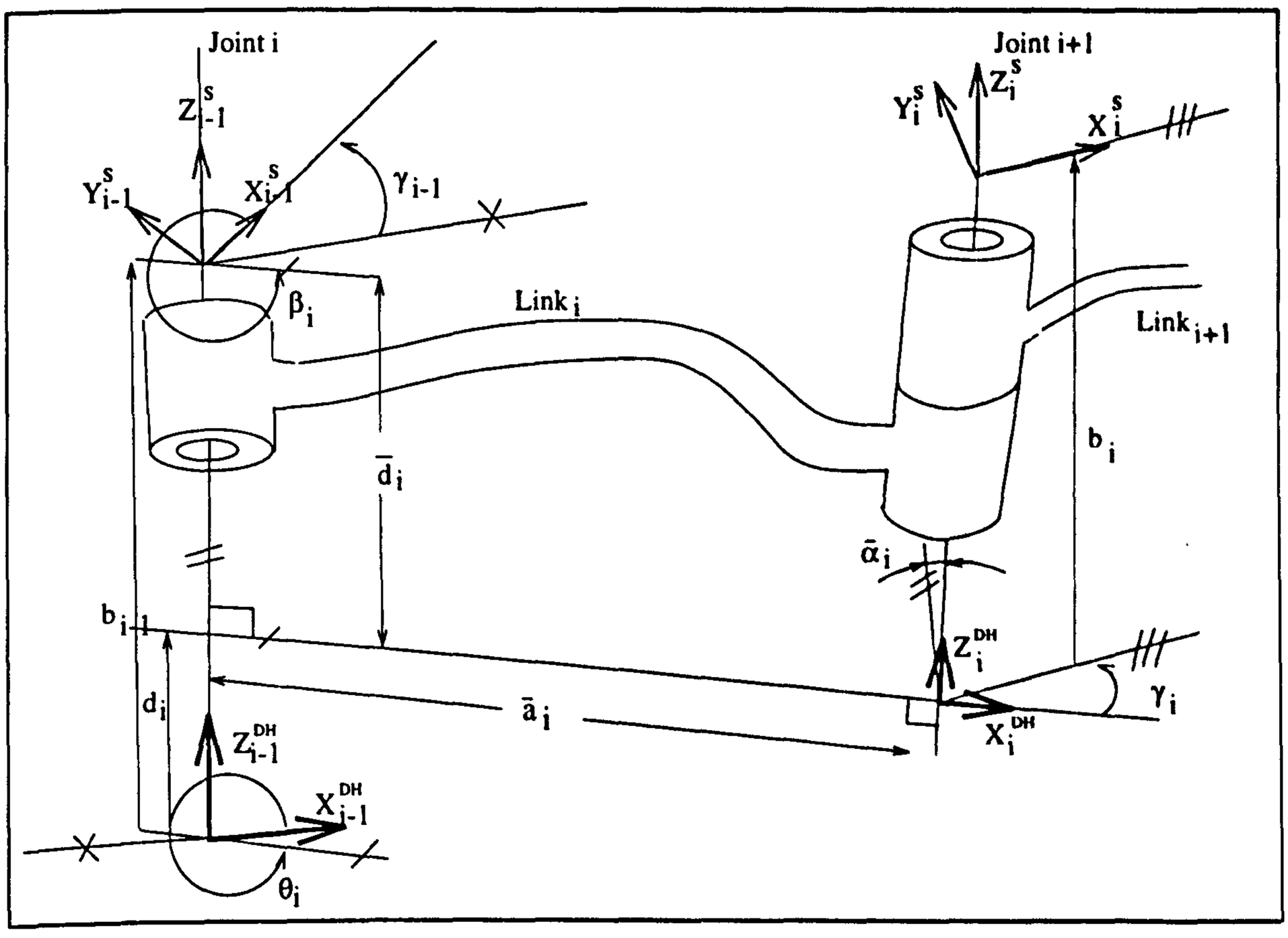


Figure 6.2: The S-Model parameters representation for a rotational joint .

Figure 6.2 illustrates the S-Model parameters for a revolute joint which depicts B_i as:

$$\begin{aligned} B_i \;=\; & Rot(z,\beta_i)Trans(0,0,\bar{d}_i)Trans(\bar{a}_i,0,0)Rot(x,\bar{\alpha}_i) \\ & Rot(z,\gamma_i)Trans(0,0,b_i) \end{aligned} \tag{6.14}$$

expanding the above B_i yields

$$S_i = \begin{bmatrix} o_x & n_x & \sin\beta_i \sin\alpha_i & p_x \\ o_y & n_y & -\cos\beta_i \sin\alpha_i & p_y \\ \sin\alpha_i \sin\gamma_i & \sin\alpha_i \sin\gamma_i & \cos\alpha_i & b_i \cos\alpha_i + \bar{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.15}$$

where

$$o_x = \cos\beta_i \cos\gamma_i - \sin\beta_i \cos\alpha_i \sin\gamma_i \tag{6.16}$$

$$o_y = \sin\beta_i \cos\gamma_i + \cos\beta_i \cos\alpha_i \sin\gamma_i \tag{6.17}$$

$$n_x = -\cos\beta_i \cos\gamma_i - \sin\beta_i \cos\alpha_i \cos\gamma_i \tag{6.18}$$

$$n_y = -\sin\beta_i \sin\gamma_i + \cos\beta_i \cos\alpha_i \cos\gamma_i \tag{6.19}$$

$$p_x = b_i \sin\beta_i \sin\alpha_i + \bar{a}_i \cos\beta_i \tag{6.20}$$

$$p_y = -b_i \cos\beta_i \sin\alpha_i + \bar{a}_i \sin\beta_i \tag{6.21}$$

The above expression is valid only if the S-Model convention is respected when assigning the various coordinate frames. The assignment of the coordinate frames should therefore satisfy the following points (see figure 6.2):

- The Z-axis of the frame $\mathcal{S}_{i-1}$ must be parallel to the joint i axis in the direction defined by the positive sense of rotation or translation.

- The origin of coordinate frame $\mathcal{S}_{i-1}$ must lie on the joint i axis.

- The Z-axis of the last coordinate frame $\mathcal{S}_n$ is parallel to the Z-axis of the next to last coordinate frame $\mathcal{S}_{n-1}$.

- The origin of $\mathcal{S}_n$ must lie on the joint n-1 axis.

From the above it is clear that these are a subset of the Denavit-Hartenberg convention, which make the S-Model less restrictive in locating the coordinate frame and more specific in choosing the origin of the frame and the X-axis

orientation, which is required to be only orthogonal to the Z-axis. Figure 6.2 also shows the nature of the relative link between the S-Model and the D-H coordinate frames. The actual transformation from $\mathcal{S}_{i-1}$ to $\mathcal{S}_i$ could easily be seen as a multiple transformation, involving intermediate frames. The sequence of the transformation is from $\mathcal{S}_{i-1}$ to $\mathcal{T}_{i-1}$ to $\mathcal{T}_i$ to $\mathcal{S}_i$ and is summarised by:

$$\begin{aligned} B_i &= Rot(z,\gamma_{i-1})Trans(0,0,-b_{i-1})\,[Rot(z,\theta_i)Trans(0,0,d_i)Trans(\bar{a}_i,0,0) \\ &\quad Rot(x,\bar{\alpha}_i)]\,Trans(0,0,b_i) \end{aligned} \tag{6.22}$$

From this self explanatory expression 6.22, the part embraced by the square brackets defines the Denavit-Hartenberg transformation A_i. The parameters α and a are taken as $\bar{\alpha}$ and $\bar{a}$ respectively, because they satisfy of the same condition. By combining terms according to the rules of homogenous transformations [37], B_i becomes,

$$\begin{aligned} B_i &= Rot(z,\theta_i-\gamma_{i-1})Trans(0,0,d_i-b_{i-1})Trans(a_i,0,0) \\ &\quad Rot(x,\alpha_i)Rot(z,-\gamma_i)Trans(0,0,b_i) \end{aligned} \tag{6.23}$$

By equating the right hand side of equations 6.22 and 6.23, we can extract the mapping expression between the S-Model and the Denavit-Hartenberg as follows:

$$\beta_i = \alpha_i - \gamma_{i-1} \tag{6.24}$$

$$\bar{d}_i = d_i - b_{i-1} \tag{6.25}$$

$$\bar{a}_i = a_i \tag{6.26}$$

$$\bar{\alpha}_i = \alpha_i \tag{6.27}$$

6.3 Kinematic Identification

The kinematic modelling is intended to identify the true(real) parameters of the kinematic model which describe the actual position and orientation of the end-effector relative to the base, for individual robot manipulators. The inaccuracy is

due to the unique kinematics and positioning performance of each manipulator, caused by manufacturing errors and the lack of sufficient knowledge of these parameters. Both the D-H and the S-Model provide an exact description of the actual rigid robot kinematics. The identification of these parameters, however, requires a detailed description of the model structure and an adequate procedure to measure robot configuration [24].

Kinematic parameter identification algorithms have been proposed by many authors: Stone *et al* [11], Hayati *et al* [73] [74], Whitney *at al* [23], Chen *et al* [75], Stanton *et al* [25] [83], Khalil *et al* [26] and Mooring *et al* [84]. The book edited by Bernhardt and Albright [85] also gathers together a number of works treating robot calibration techniques and modelling and parameter identification methods.

This chapter treats the application of the methodology proposed by Stone [24], for the direct identification of the D-H model parameters using the modified D-H A_i matrices.

While the D-H model is specified by a minimum number of parameters (four), Stone [24] claims that it is not amenable to direct identification. His new model, the S-Model, was developed based on the D-H model with two more parameters for each link as detailed in the previous section, although it was still described as incomplete in [25] and [86]. However, D-H model parameter and modified version of it were successfully estimated [74, 81] [26] and [27].

The added parameters in the S-Model are to allow free location of the link coordinate frame origin and free orientation of its X-axis. During the identification task, the six S-Model parameters for each link are determined through direct measurement of the joints' motions by attaching to the moving link a target whose position is used to estimate the joint features.

Stone allows a free positioning of the target on the moving link while Stanton specifies a target fixed to the end-effector of the arm, due to the nature of the measurement system. The measurement of the target Cartesian position

in our case uses the same system used by Stanton [25]. The joint features are identified first for each joint, and from these the D-H model parameters are then extracted. In most of the cases of the work we propose, the *plane-of-rotation* undergoes a translation along the axis of rotation to coincide with X-Y plane of the corresponding D-H coordinate frame.

6.3.1 Kinematic Features

While Stone has used a minimum number of features, $2n_r + n_p$, for the identification of the S-model parameters, where n_r is the number of the revolute joints and n_p is the number of the prismatic joints, we make use of a third feature associated to the revolute joint. Therefore, the three features of the revolute joint are *plane-of-rotation*, *centre-of-rotation* and *radius-of-rotation*, while the feature associated to a prismatic joint is called *line-of-translation*.

It is clear from the features' names that they are extracted from the nature of the motion of the joints. A point on a rotating link describes a circle situated on a plane called *plane-of-rotation*, the centre of the circle is a point of the plane and situated on the joint axis. It is called *centre-of-rotation* while the radius of the circle is called *radius-of-rotation*. For a prismatic joint, any point of the link situated on the joint axis describes a straight line parallel to the joint axis, which is called *line-of-translation*.

6.3.2 Features Identification

This section illustrates the different steps of the features identification process. During the development of the identification algorithm, the main similarities and differences with Stone's method are highlighted where appropriate. The next step involves the link coordinate frames specification, and is followed by the model parameters extraction. The prismatic joints have only one feature, which is the line-of-translation. Stone [24] gives a simple and easy approach for the estimation

of this feature. The work presented in this chapter concentrates upon revolute joints manipulators of which the identification method is described in the following

Plane-of-Rotation Estimation

As indicated in section 6.3 the measurement target is attached to the moving link during the measurement performed in [11], whereas the target in this case is attached to the end-effector. Therefore, the conditions that make the target describe a circle or a line in space are that the joints between the moving and the last joint(end-effector) and the joints between the manipulator base and the one concerned with the measurement should remain in the same configuration during the measurement. The combination of this, with the definition of the plane-of-rotation means that the plane-of-rotation is shifted along the Z-axis of the moving link coordinate frame. The distance between the aforementioned plane-of-rotation and the D-H x-y plane is determined by the configuration (joint angles) and also the identified radius of rotation from previous measurements. The identification process starts from joint n and works joint by joint backwards to joint 1.

Identifying the plane-of-rotation is a straightforward task. When joint i is made to rotate and the rest of the joints (1 to $i-1$ and $i+1$ to n) are locked, the target fixed relative to the end-effector traces a circle in space. The coefficients of the plane-of-rotation can be estimated from a fitting of the m measured Cartesian positions of the target corresponding to m different positions of joint i to a plane. Many methods have been suggested to solve the above problem, some of which are exact and make use of an eigenvalues solution , and others are based on minimisation techniques such as linear least squares [24], or a non linear optimisation technique suggested by Stanton [83]. For reasons of simplicity, Stone's linear least square technique is used in this work. More details of the other methods can be found in the above references.

This work presents a method of fitting to a plane the measured target Cartesian

positions caused by the movement of a revolute joint. The method attempts to minimise the sum of the perpendicular distances between the measured points and the estimated plane-of-rotation. To complete this task, an approximation of the minimisation is achieved by the repeated application of a linear least squares regression. For the purpose of making the measured target positions simple to picture the following suggestion proves to be advantageous. While measuring the target positions corresponding to the motion of the i^{th} joint (called the i^{th} target), joint 1 through to joint $i-1$ are required to be in their corresponding zero positions. It is also preferable that joints $i+1$ through to n, are at their zero positions, though they could be locked at any arbitrary configuration during the measurement. To make the suggested method most efficient and reliable a set of conditions and guidelines introduced by Stone [24] should be observed.

- the measured target positions should be uniformly distributed between the upper and the lower limits of the joint's travel.
- The manipulators revolute joints have a maximum travel rotation (180 degrees or more).
- The standard step in the measurements of the target's Cartesian position should be several orders of magnitude less than the nominal distance between the target and the axis of rotation.

Independently, each joint of the manipulator is made to move through the required m positions, maintaining a correct sense of rotation. It is assumed that the joint angles $q_{i,j}$ are ordered such that $q_{i,j} < q_{i,j+1}$ where i is the joint number and j is the order of the corresponding measured end-effector position $\vec{p}_j = [x_j\ y_j\ z_j]^T$.

The general equation of a plane is given by:

$$Ax + By + Cz + D = 0 \tag{6.28}$$

where x, y and z are the coordinate of the points of the plane and the coefficients A, B, C and D are to be identified.

The general equation of a plane can be rewritten as:

$$Z = Ex + Fy + G = \phi^T \Theta \tag{6.29}$$

where

$$\phi = [x \ y \ 1]^T \tag{6.30}$$

$$\Theta = [E \ F \ G]^T, \tag{6.31}$$

which are defined to be the information vector and the parameter vector. The Z coordinate is defined to be the output of the equation 6.29. A simple regression of Z on x, y and 1 corresponds to the minimisation of the sum of the squared errors in the Z coordinate.

$$\Xi \equiv \sum_{j=1}^{m} (Z - Z_j)^2 = \sum_{j=1}^{m} (\phi^T \Theta - Z_j)^2. \tag{6.32}$$

The minimisation of the above by equating it to zero yields the linear least squares solution [87]:

$$\Theta = \left[\Phi^T \Phi\right]^{-1} \Phi Z \tag{6.33}$$

where

$$\Phi = [\phi_1 \ \phi_2 \ \phi_3, \ldots, \phi_m]^T \tag{6.34}$$

$$Z = [Z_1 \ Z_2 \ Z_3, \ldots, Z_m] \tag{6.35}$$

The $\Phi^T \Phi$ is the correlation matrix which is composed of the sums of products of x, y and 1. The application of the L-S assumes that the coordinates of the information vector are independent variables and measured without error. Consequently the closer the plane-of-rotation is to being parallel with the Z-axis the farther the solution 6.33 is from the true plane-of-rotation [24]. To avoid this

problem the measured data are required to be transferred to a new coordinate frame in which the measured target positions lie as close as possible to its X-Y plane. Since the first assumption ensures that the measured data fits as closely as possible a plane, the application of solution 6.33 means the minimisation of the sum of the perpendicular errors between the plane-of-rotation and the measured z coordinates.

Three points of the measured target positions are then used for the formation of the new coordinate frame. These 3 points uniquely form an initial approximation of X-Y plane, hence the plane-of-rotation. The points are chosen to be mutually most distant from one another and are denoted by $\vec{p}_k, \vec{p}_l$ and $\vec{p}_m$. Again, $k < l < m$ is required which means that $q_{i,k} < q_{i,l} < q_{i,m}$ in order to preserve the sense of rotation .

Figure 6.3 assists in the understanding of the formation of the new coordinate frame where the plane of rotation is estimated.

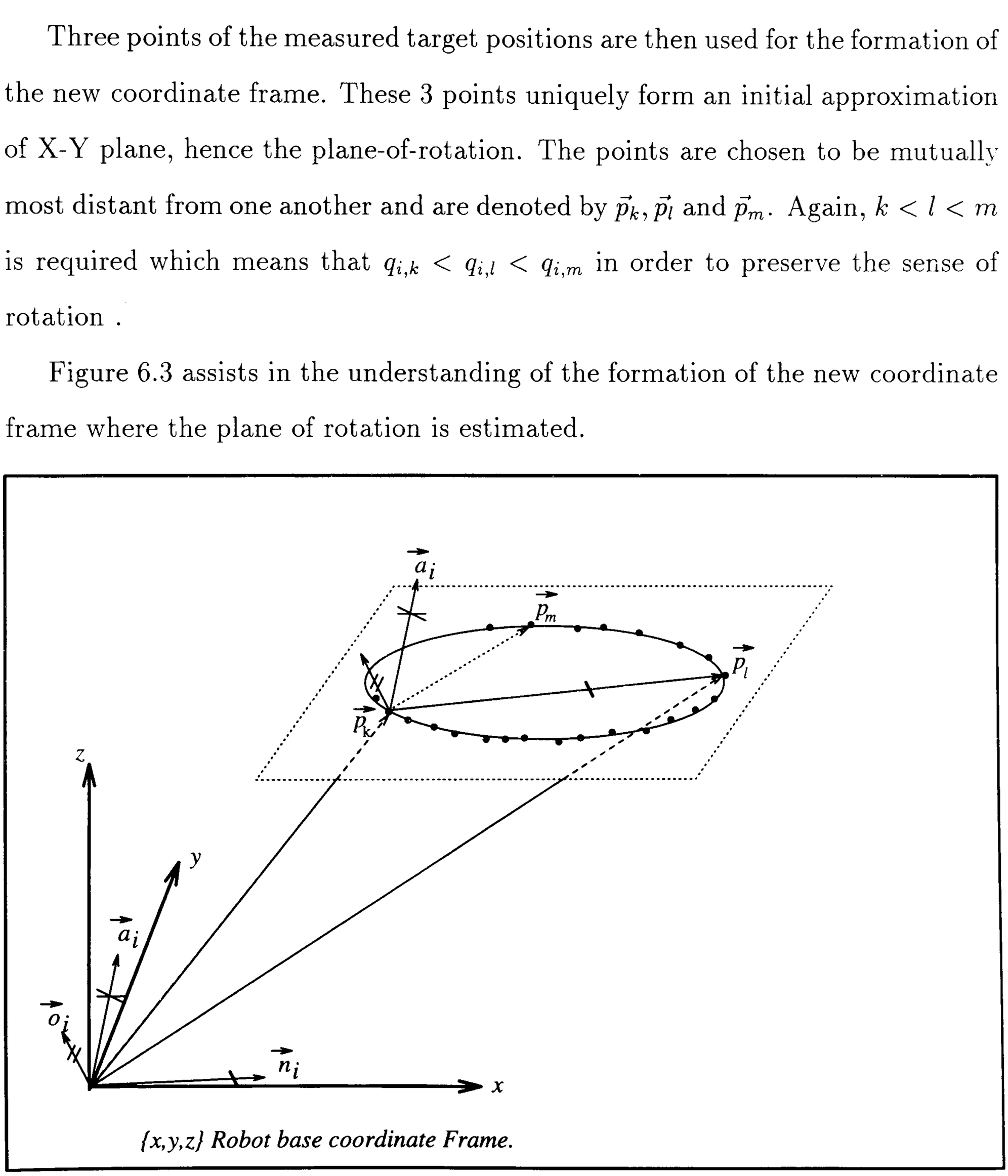


Figure 6.3: The formation of a new coordinate frame

The X-axis is chosen to be parallel to the line joining $\vec{p}_k$ to $\vec{p}_l$; The Z-axis is perpendicular to this axis and to the line joining $\vec{p}_k$ to $\vec{p}_m$. The Y-axis completes the orthogonal system. The origin of this initial coordinate frame is coincident with the origin of the frame in which the data are measured (presented). which in our case is the robot arm base coordinate frame. The unit vectors of the newly formed coordinate frame are calculated as follows:

The X-axis unit vector is

$$\vec{\mathbf{n}} = \frac{\vec{p}_l - \vec{p}_k}{\|\vec{p}_l - \vec{p}_k\|} \tag{6.36}$$

The Z-axis unit vector is perpendicular to the plane and is therefore given by:

$$\vec{\mathbf{a}} = \frac{(\vec{p}_l - \vec{p}_k) \times (\vec{p}_m - \vec{p}_k)}{\|(\vec{p}_l - \vec{p}_k) \times (\vec{p}_m - \vec{p}_k)\|}, \tag{6.37}$$

and finally the Y-axis unit vector completes the orthogonal system by:

$$\vec{\mathbf{o}} = \vec{\mathbf{a}} \times \vec{\mathbf{n}} \tag{6.38}$$

Therefore, the homogeneous transformation matrix between the measurement frame and the new frame $\mathcal{C}_0$ is given by:

$$R_i = \begin{bmatrix} \vec{\mathbf{n}} & \vec{\mathbf{o}} & \vec{\mathbf{a}} & \vec{0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.39}$$

which describes a pure rotation. After the data are transferred to the new coordinate frame $\mathcal{C}_0$, the estimation is then executed using the L-S method as described above and the plane-of-rotation coefficients are transferred back to the base coordinate frame via:

$$[A\ B\ C\ D]^T = R\left[A^0 B^0 C^0 D^0\right]^T \tag{6.40}$$

To achieve increased accuracy of solution, more iteration may be needed. At this stage, the estimated plane is used as the X-Y plane of a new coordinate frame, from which a more accurate plane-of-rotation is calculated. The unit normal

vector which will be used in the next step is calculated using the coefficients of the estimated plane-of-rotation, according to [88] by:

$$\vec{\mathbf{a}} = [Aw \quad Bw \quad Cw] \tag{6.41}$$

where

$$w = \frac{1}{\sqrt{A^2 + B^2 + C^2}} \tag{6.42}$$

The above process is repeated until the desired minimisation is adequately solved and the X-Y plane of $\mathcal{C}_0$ is as close as possible to being parallel to the plane-of-rotation .

At the i^{th} iteration, the transformation matrix $R = R_i$ is computed using the normal vector to the plane-of-rotation from the previous iteration calculated by equation 6.41. The new coordinate frame is completed by choosing the X and Y axes arbitrarily. This is done by choosing the vector $\vec{\mathbf{n}}$ and completing the orthogonal system using 6.38. The new transformation matrix is formed as in the previous step, using 6.39. The linear least squares solution is then applied again and the whole process is repeated until terminated when the difference between the two consecutive estimates of the plane-of-rotation satisfies a pre-set condition.

The plane-of-rotation should be translated for most of the links to coincide with the D-H X-Y plane. This is first done by using the offsets between the n^{th} axis and the target attached to the n^{th} link. Consequently, the accurate knowledge of the target location with reference to the last link is essential. In the case of other links, the parameters previously estimated could be used unless this information is not available in which case nominal lengths are used. The translation would be along the Z-axis of the $\mathcal{C}_i$ for each link. The value and direction of the translation depend on the configuration of the links $i + 1$ to n as well as the coordinates of the target point relative to the n^{th} coordinate frame.

Centre-of-Rotation Estimation

The Centre-of-rotation feature is a point of the link that lies on the axis of rotation. This section presents a methodology for the estimation of the Cartesian coordinates of this point relative to the robot base frame.

Ideally, as mentioned in the previous section, the measured data points $\vec{P}_{i,j}$ in addition to their lying on a plane on space, lie on a circle whose centre is the centre-of-rotation. The estimation of the Cartesian coordinates of the centre-of-rotation results from fitting the measured points of the target to the circle equation. The objective in this case would be the minimisation of the sum of the perpendicular distances between the estimated and measured circles. Again in this case the repeated application of the linear least squares regression is chosen.

The general equation of a circle is given by:

$$(x-g)^2 + (y-h)^2 = r^2 \tag{6.43}$$

where g and h are the Cartesian coordinates of the centre of the circle which are to be estimated in this exercise, and r is the radius of the circle, hence the radius-of-rotation. While a circle can lie in a three dimensional space which is the case of the one described by the target, equation 6.43 describes a circle that lies only on the X-Y plane. That is revealed by the absence of the z parameter from the equation. This motivates the transformation of the data to a coordinate frame where all points lie on the X-Y plane, or at least on a plane parallel to the X-Y. The solution of the minimisation problem is therefore evident. The plane-of-rotation is the obvious candidate for this transformation and the new measured data points are therefore given by:

$$\vec{p_j^o} = R^{-1}\vec{p_j} \tag{6.44}$$

for $j = 1, ..., m$. If the estimate plane-of-rotation is only parallel to the X-Y plane of $\mathcal{C}$, the projection of x and y of $\vec{p_j^o}$ constitute the projection of the target positions. The equation 6.43 becomes:

$$x^2 + y^2 = 2gx + 2hy + r^2 - g^2 - h^2 \tag{6.45}$$

By taking $w = x^2 + y^2$ and combining the coefficients on the right hand side, equation 6.45 is rewritten as:

$$w = Hx + Jy + K = \phi^T \Theta \tag{6.46}$$

where $\phi = [x\ y\ 1]^T$

$\Theta = [H\ J\ K]^T$

The output of 6.46, w, is the squared distance between a point on the circle and origin. A simple regression of w on x^o, y^o and 1 corresponds to the minimisation:

$$\Xi \equiv \sum_{j=1}^{m} \xi_j = \sum_{j=1}^{m} (w - w_j)^2 = \sum_{j=1}^{m} (\phi^T \Theta - w_j)^2 \tag{6.47}$$

where $w_j = x_j^{o2} + y_j^{o2}$

Minimising 6.47 does not correspond to minimising the sum of the squared distances between the measured and the estimated circles, unless the origins are coincident with the origin of $\mathcal{C}$. We therefore opt for the repeated application of the linear least squares solution and a coordinates transformation. At the i^{th} iteration we translate the x and y components of the originally projected measured points $\vec{p_j^o}$ by:

$$x_j^{oi} = x_j^o - g_{i-1} \tag{6.48}$$

$$y_j^{oi} = y_j^o - h_{i-1} \tag{6.49}$$

where g_{i-1} and h_{i-1} are the estimated coordinates of the centre-of-rotation from previous iteration, $(i-1)^{th}$.

By repeated translation of the original data during each iteration, the centre of the circle approaches zero which means it approaches the origin of the frame. The L-S solution using x_j^{oi} and y_j^{oi} therefore approaches the solution of the desired minimisation problem 6.47.

The coordinates of the centre-of-rotation is then given by:

$$\vec{p_i^c} = R^T[g\ h\ z_c] \tag{6.50}$$

where z_c could be computed by different means, for example using g and h as x and y coordinates in the plane-of-rotation equation or as given by Stone [24]:

$$z_c^o = \frac{D}{\sqrt{A^2 + B^2 + C^2}} \tag{6.51}$$

The set of centre-of-rotation vectors and the normal vectors to the plane-of-rotations are used in the next section to construct a kinematic model of the manipulator under consideration.

6.3.3 Link Coordinate Frame Construction

This part makes use of the set of the estimated n normal vector to the plane-of-rotations $\vec{\mathbf{a}}_i$ and n centre-of-rotations, from the previous sections, to specify the locations and orientations of the D-H model link coordinate frames.

First of all, partial D-H models are specified, which define the location and orientation of the individual D-H coordinate frames in terms of the base frame of the robot arm which are given by:

$$T_i = A_1 \cdot A_2 \cdot ..... \cdot A_i \tag{6.52}$$

where A_i are the D-H homogeneous transformation matrices between consecutive coordinate frames. We start at this stage by defining a set of constant T_i matrices which describe the kinematics of the manipulator in the zero configuration only. From the set of T_i matrices, the D-H parameters are calculated using the general structure of the homogeneous transformation matrix A_i.

Figure 6.4 illustrates the construction of the coordinate frame $\mathcal{A}_i$ when joint $i = 1$ is revolute.

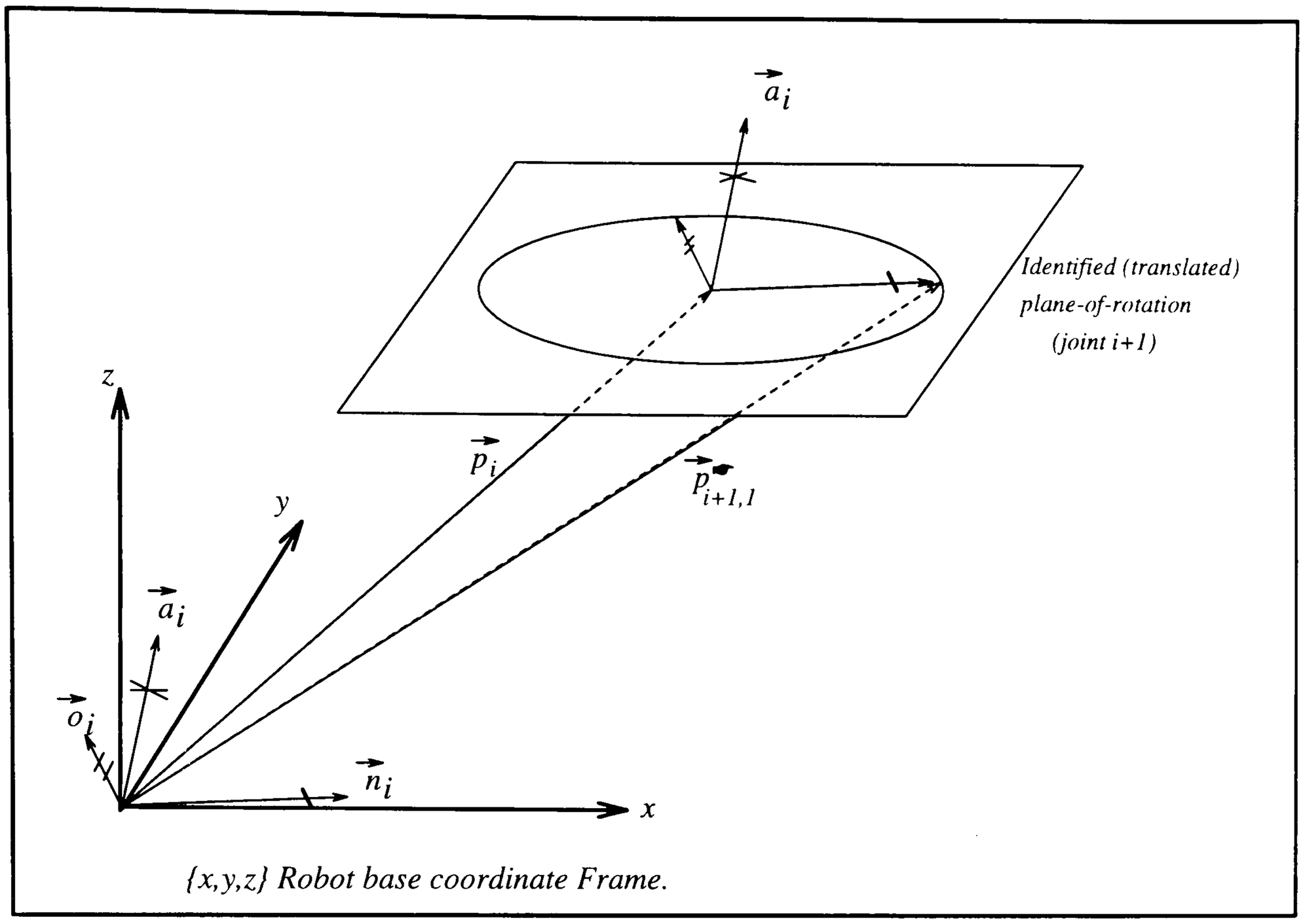


Figure 6.4: The $\mathcal{A}_i$ coordinate frame construction

The construction of the $\mathcal{T}_i$ coordinate frames –denoted above by $\mathcal{A}_i$– is specified by the identified joint $i+1$ features, and T_i is given by :

$$T_i = \begin{bmatrix} \vec{\mathbf{n}}_i & \vec{\mathbf{o}}_i & \vec{\mathbf{a}}_i & \vec{p}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.53}$$

where $\vec{\mathbf{a}}_i$ is the Z-axis unit vector, of the $\mathcal{T}_i$, estimated in previous sections. The direction of the X and Y axes, defined by $\vec{\mathbf{n}}_i$ and $\vec{\mathbf{o}}_i$, are not arbitrary in this case since $\mathcal{T}_i$ should satisfy the D-H convention. $\vec{\mathbf{n}}_i$ is chosen to be along the length of the link, whereas $\vec{\mathbf{o}}_i$ completes the orthogonal system.

The X unit vector is chosen as:

$$\vec{\mathbf{n}}_i = \frac{\vec{p^*_{i+1,1}} - \vec{p^*_{i+1,c}}}{\|\vec{p^*_{i+1,1}} - \vec{p^*_{i+1,c}}\|}, \tag{6.54}$$

where $\vec{p^*_{i+1,1}}$ is the location of the first target position and corresponds to joint angle $\theta_{i+1,1}$ at its zero position. The first target position is used for convenience. The unit vector $\vec{\mathbf{n}}_i$ can also be chosen, in some cases, along the line joining the two

consecutive centres of rotation. This has been done successfully between joint 2 and 3 for the PUMA 560 presented in section 6.4. The origin of the i^{th} coordinate frame $\vec{p_i}$ is the centre-of-rotation after it undergoes a translation to approach the D-H frame origin, $\vec{p^*_{i+1,c}}$.

The location of the first target position should be adjusted to lie on the link length, using the previously estimated parameters from outer links. This could also be achieved by choosing a proper links alignment during the measurement stage.

Stone [24] and Stanton [25] both assume no degree of freedom between the last link and the tool attached to it, and define the same tool attachment point (TAP), for the case of the PUMA560. However, Stanton designates the S-model as incomplete, indicating that a relationship between the base of the manipulator and a coordinate frame attached to the TAP of the end-effector cannot be defined. While Stone [11] [24] associates no features with the last coordinate frame, this will try to make use of these estimated feature to help defining more accurately the position of the target relative to $\mathcal{T}_n$. The transformation between the last link and the Tool coordinate frames is a constant, and the position of the target could be measured using the last one or two joints radius of rotation features.

Assuming that the tool coordinate frame has the same orientation as the n^{th} link coordinate frame, T_{tool} is therefore:

$$T_{Tool} = \begin{bmatrix} \vec{\mathbf{n}}_n & \vec{\mathbf{o}}_n & \vec{\mathbf{a}}_n & \vec{p_n} + \vec{p_T} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.55}$$

where $\vec{p}_T$ is the vector position of the target relative to the n^{th} coordinate frame.

The construction of the T_i matrices leads to the computation of the D-H model parameters presented in the next section.

6.3.4 Denavit-Hartenberg

At this stage, the transformation matrices A_i are computed from the T_i matrices. Equation 6.52 can be rewritten as

$$T_i = T_{i-1} \cdot A_i \tag{6.56}$$

Therefore,

$$A_i = T_{i-1}^{-1} \cdot T_i \quad for \;\; i = 1, ..., n \tag{6.57}$$

The computed A_i are written as:

$$A_i = \begin{bmatrix} n_{i,x} & o_{i,x} & a_{i,x} & p_x \\ n_{i,y} & o_{i,y} & a_{i,y} & p_y \\ n_{i,z} & o_{i,z} & a_{i,z} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.58}$$

where the individual elements are known.

Paul's backwards multiplication technique [37] is used to compute the constant transformation parameters, θ, α, a and d, at the given signature configuration.

The joint variable for a revolute joint is θ. This may deviate from its true value due to some mechanical consideration. Consequently, the measured joint variable contains a number of offsets which include, an encoder error, $\theta_i^{encoder}$, a machining error offset θ_i^{off} and a mounting zero error. Therefore, during the manipulator normal operation, the programmed controller joint angle differs from the real joint position by the sum of the aforementioned errors in addition to backlash offset when a gearbox is present in the joint (see figure 6.5).

$$\theta_i^* = \theta_i + (\theta_i^{encoder} + \theta_i^{off} + \theta_i^{zero}) \tag{6.59}$$

where θ_i^* is the real value of the joint angle. Since, the identification of the individual errors is not possible only one combined joint angle error is considered.

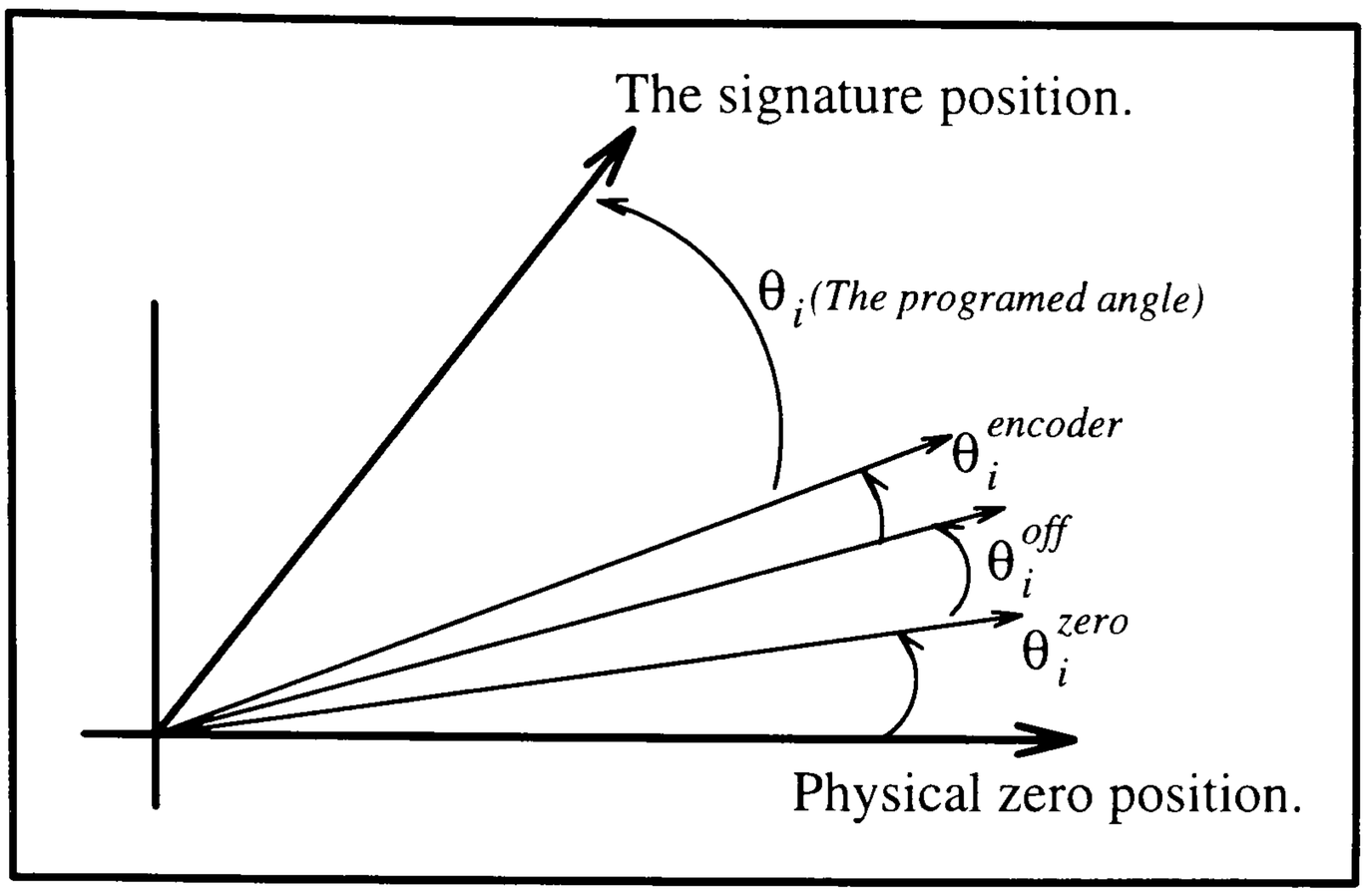


Figure 6.5: A joint signature errors

θ_i^{error}. Equation 6.59 then becomes;

$$\theta_i^* = \theta_i + \theta_i^{error} \tag{6.60}$$

By equating the two expressions of A_i 6.9 and 6.58 and replacing θ_i by θ_i^*

$$\theta_i^* = \begin{cases} 0, & \text{if} \quad a_{i,x} = a_{i,y} = 0; \\ \text{atan}(-a_{i,x}/a_{i,y}), & \text{otherwise.} \end{cases} \tag{6.61}$$

From the above and the measured θ_i, the error could be calculated according to 6.60. The corresponding twist angle appears in many elements of the matrix and could be calculated using:

$$\alpha_{i-1} = \text{atan}\frac{-a_{i,y}}{o_{i,y}} \tag{6.62}$$

and the parameter d is given by:

$$d_i = p_z \tag{6.63}$$

Finally, the D-H fourth parameter a is calculated using:

$$a_{i-1} = \sqrt{p_{i,x}^2 + p_{i,y}^2} \tag{6.64}$$

The parameter a_{i-1} could be estimated, in some cases, directly using the radius of rotation.

When two consecutive joint axes are parallel the D-H model suffers disproportion and the above calculation would not reflect the link parameters. The modifications proposed by Hayati [81] and Mooring [82] are then used to account for disproportional models [86]. The transformation 6.8 takes a new form given by:

$$A_i = A_i(q_i) \equiv Rot(z,\theta_i)Trans(0,0,\mathbf{d}_i)Trans(\mathbf{a}_{i-1},0,0) \\ Rot(x,\alpha_{i-1})Rot(y,\beta_i) \tag{6.65}$$

and the matrix A_i is then expressed as

$$A_i = \begin{bmatrix} C\theta_i C\beta_i - S\theta_i S\alpha_{i-1} S\beta_i & -S\theta_i C\alpha_{i-1} & C\theta_i S\beta_i + S\theta_i S\alpha_{i-1} C\beta_i & \mathbf{a}_{i-1} C\theta_i \\ S\theta_i C\beta_i + C\theta_i S\alpha_{i-1} S\beta_i & C\theta_i C\alpha_{i-1} & S\theta_i S\beta_i - C\theta_i S\alpha_{i-1} C\beta_i & \mathbf{a}_{i-1} S\theta_i \\ -C\alpha_{i-1} S\beta_i & S\alpha_{i-1} & C\alpha_{i-1} C\beta_i & \mathbf{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.66}$$

where $C\theta = \cos\theta$ and $S\theta = \sin\theta$ and and the same for α and β.

The values of the 5 kinematic parameters are extracted in a similar way to above, by equating the new expression of A_i 6.66 and 6.58.

The estimation process would be applied according to one of the two forms described above over the whole joint space of the robot arm, in an inward iterative manner, starting with the outer link and going link by link through to the first one. A proper assessment of the errors and the effect of various errors on the overall error of the end-effector would be then established. The estimation technique described is applied to estimate the kinematic parameters of a PUMA560 robot arm in the next section.

6.4 PUMA560 Kinematic Model Identification

The original measurements performed on the robot were destined for a dynamic model validation, and were restricted only to the first three links of the robot,

due to hardware limitations. The Cartesian position of a target fixed to the end-effector was measured relative to the robot base coordinate frame. as well as the first three links angular positions. A set of three measurement was performed such that only one joint was made to move during each measurement. The whole set was repeated for a different speed to later compare its effects on the accuracy of the robot. The set with the lower speed was used for the estimation of the kinematic model parameters using the technique described in the previous sections.

During the measurement of joint one tests all other joints were locked at their zero position and this was true for joint 3 test as well. Joint 3 was locked at 90° during measurements involving joint 2 movement, making the arm stretched horizontally before the start of the move. The description of the instrumentation is presented in chapter 4, though a brief description of the Cartesian position of the end-effector measurement instrument is given for completeness. The identified parameters and the change of the kinematic model are also given in this section.

To test the proposed method further the estimation of the complete model of a Puma 560 based on simulation is also presented and the improvement is assessed using a five points ISO test [25]. During the simulations operation it was indicated that the offsets of the target from the last link of the manipulator have an important effect on the accuracy of the identified parameters. Although these offsets are needed for establishing the last coordinate frames (wrist) they need to be minimal when measurements involve the first three links and therefore minimal plane-of-rotation is needed. On the other hand, when assessing the improvement of a calibrated manipulator end-effector offsets are necessary to establish the effects of wrist errors (angles) effects on accuracy.

6.4.1 The Measurement System

The instrument used to measure the Cartesian end-effector position of the Puma manipulator is a laser tracking system designed and built at Surrey University.

England [51]. The instrument consists of two tracking sub-systems that each drive a laser beam towards retroreflective target, attached to end effector of the robot arm, by using two orthogonally mounted optical scanner units. see figure 4.2. The tracking errors resultant from driving the scanners are used to calculate the 3-dimensional position of the target. The instrument is believed to have a repeatability of $\pm 0.1mm$ [25].

6.4.2 The PUMA560 D-H Model Parameters

During the estimation process the fourth to the sixth links of the manipulator were considered a fixed extension of the third link. This was done to minimise the effect of their errors on the estimated parameters of the first three links. Table 6.1 summarises the D-H model, parameters of a Puma 560, nominal and estimated using the measurement obtained from the Optotrac. Results and assessment of the improvement of the kinematic model are discussed in the next section.

Table 6.1: The D-H parameters: nominal and the estimated(*), for the first 3 links.

Link	θ_i (deg)	α_i (deg)	a_i (cm)	d_i (cm)	β_i (deg)
1	0.0	-90.0	0.0	0.0	0.0
2	0.0	0.0	43.18	14.909	0.0
3	0.0	90.0	-2.032	0.0	0.0
4	0.0	-90.0	0.0	43.307	0.0
5	0.0	90.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0
1*	-1.3037	-90.0013	0.0	0.0	0.0
2*	0.1481	-0.1343	43.210	14.906	0.0
3*	2.751	89.998	-2.0	0	-0.2942
4*	...	...	...	43.322	0.0

A complete D-H model was estimated using simulation results from a Puma 560 model after it was subjected to several alterations to imitate the manufacturing errors . The end effector Cartesian positions were stored and used as the measured

data in the estimation process [80]. The resultant estimated parameters are given in table 6.2

Table 6.2: Puma 560 D-H parameters estimated, from a modified model.

Link	θ_i (deg)	α_i (deg)	a_i (mm)	d_i (mm)	β_i (deg)
1	-1.306	-90.00	0.0	0.0	0.
2	0.0011	0.00	432.3001	148.9911	0.
3	1.9855	90.0545	-19.9898	0.0	0.0027
4	0.002	-89.94	0.0	434.0384	0.
5	$-2.5e^{-5}$	90.00	0.0	0.0	0.
6	$8.3e^{-7}$	0.0	0.0	0.0	0.

6.5 Results

The results are divided into two parts, the first of which contains the results of the estimation based on the measured values and are given in forms of graphs. The second part is based on the simulation values where the improvement is assessed according to an ISO test. Figure 6.6 and 6.7 illustrate the improvement introduced to target position after using the estimated parameters for joint 2 and 3 movement respectively. In this case a simple approach is used to assess the improvement. The spatial 3-D distance between the measured target and the one according to the kinematic model are calculated for all the measured targets. This distance is the value of the positioning error. These are then compared for the two cases; when using the nominal model, and when using the estimated parameters. A positioning error of the order of 30 mm in figure 6.6 is not exaggerated when taking in account 1.3° error of θ_1 and the arm fully stretched. The average positioning error reduces from 29.62 mm to 5.05 mm after using the estimated values, showing an improvement of about 82.92%. According to figure 6.7 the average error reduces from 25.67 mm to 6.33 mm, indicating an improvement of 75.45%. The average improvement concerning the positioning caused joint 2 and 3 is 79.19%, which is

considerable . This agrees with results presented by Robinson et al [89], where an average improvement of about 80.22% was achieved through improving the kinematic model of the first three links of the Puma 560.

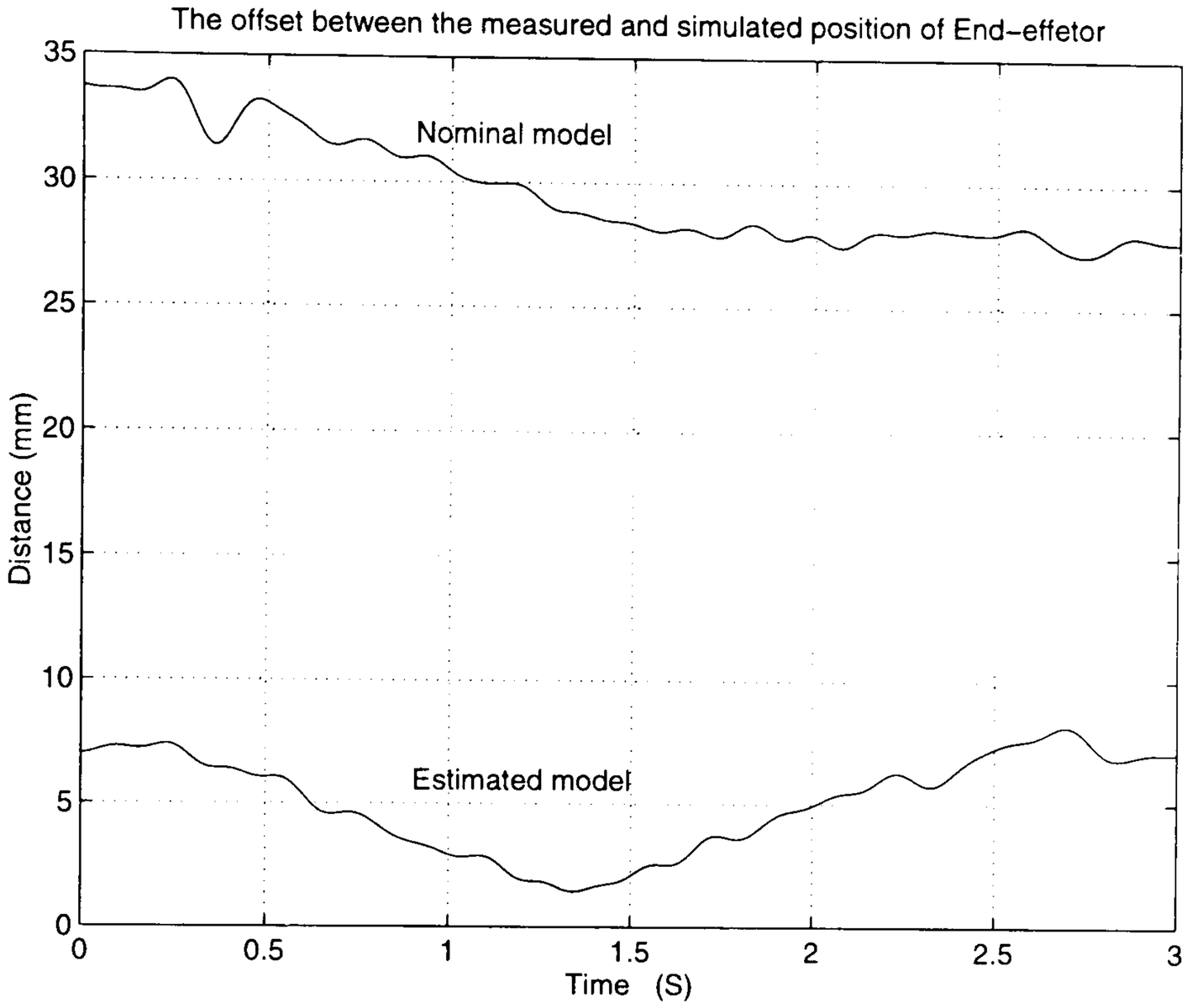


Figure 6.6: The distance between the measured target and the model target using standard and estimated model when joint 2 only is moving.

The second part of the results concerns the estimated Puma 560 model shown in table 6.2. Simulation of the manipulator positioning was executed using the nominal and estimated model to assess the improvement. Again the error is considered to be the distance between the end-effector measured position and the one using the model, nominal or estimated. In this case the measured position is the one obtained from the modified Puma model. The evaluation of the improvement of the accuracy was performed at 5 points of the ISO test cube defined in [25] and results are shown in table 6.3. The points P2 to P5 are the diagonal corners of cube of 500 mm side situated in the area with the most anticipated use of the manipulator. The average accuracy improvement is estimated to exceed 82.70%.

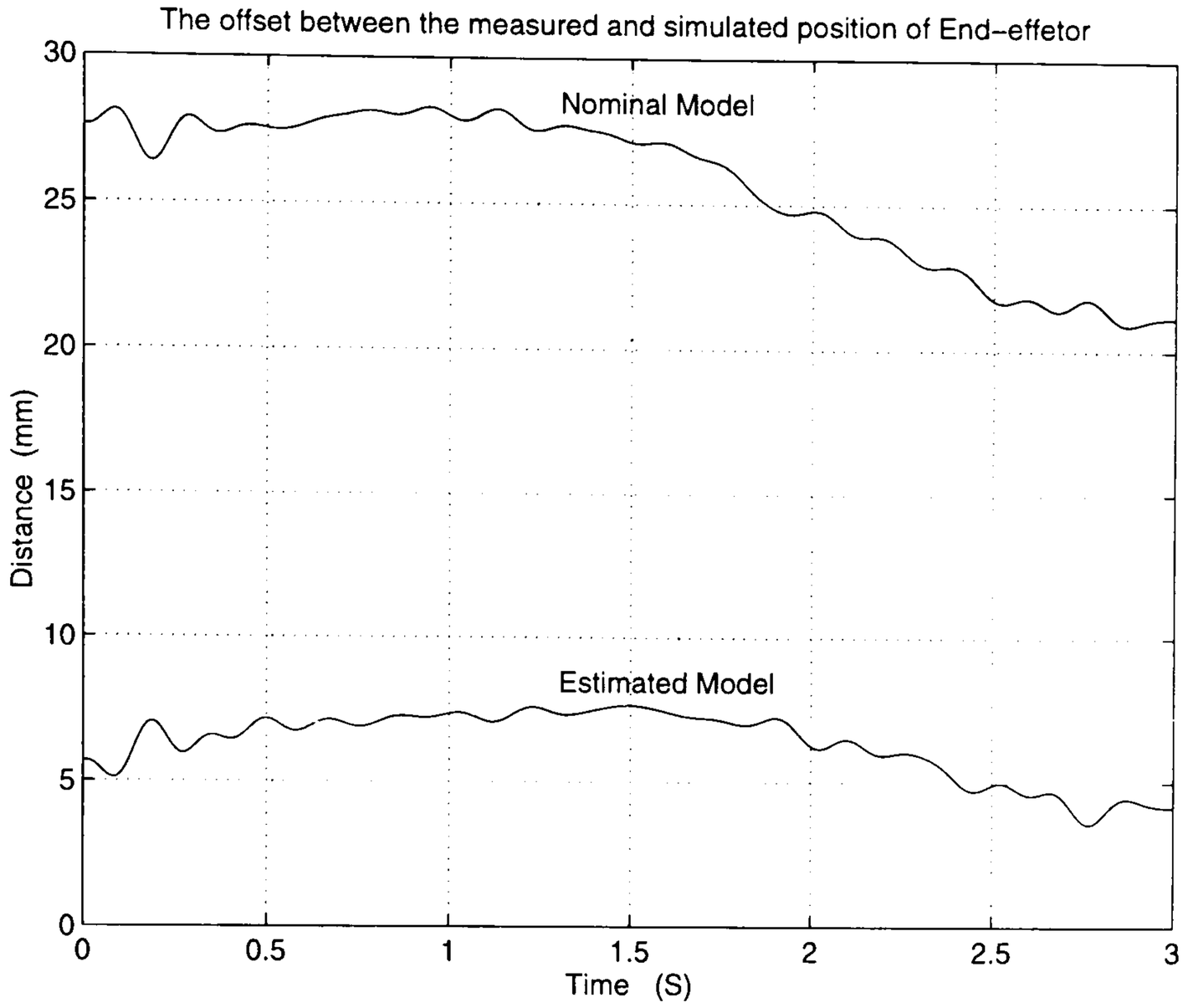


Figure 6.7: The distance between the measured target and the model target using standard and estimated model when joint 3 only is moving.

Table 6.3: The ISO test points error improvement using simulation.

	P1	P2	P3	P4	P5
Nominal Accuracy (mm)	17.6922	16.0544	16.4298	23.0123	18.3326
Improved Accuracy (mm)	3.0494	2.9483	3.0159	3.5302	3.1533
Improvement (%)	82.765	81.636	81.644	84.660	82.800
Average Improvement (%)	82.7074				

6.6 Conclusion

Although the S-Model is an exact kinematic description, it does not give an insight of the physical parameters of the robot arm. This often leads to establishing different parameters for the same model, due to the wide choice of possible locations of the links coordinate frames. This arbitrary choice of locations is avoided when using the D-H model. Instead of having several possible values in the S-Model, the identified parameters are constant for the D-H model. This facilitates the task of improving the existing kinematic control, rather than creating new algorithms based on the implementation of the S-Model. The method presented in this work concentrates on the direct identification of the D-H model parameters, or the modified version of it that include the new parameter β to account for disproportional models when two consecutive joint axes are nominally parallel. The S-Model, on the other hand, suffers the same discontinuity as the D-H model when two consecutive joint axes are parallel.

Moreover, the D-H model identification algorithm possesses most of the features as the S-Model for accurate kinematic parameter estimation. The sensor system for measuring the position of the target point is independent of the manipulator, and does not require an accurate positioning relative to the robot arm. Simple linear least-squares algorithms are applied to estimate the link features and non-linear minimisation problems are thus avoided. The effects of measurement noise on the estimated parameters can be reduced substantially by increasing the number of target locations measured per circle.

The success of the proposed method is illustrated by the more than 80% positioning accuracy improvement shown in the previous section. Despite the increasing number of methods of improving manipulator positioning in the last few years, the proposed method shows one way of achieving kinematic model improvement and therefore improving manipulator end-effector positioning accuracy. This will respond to the increase in demand for off-line programming

and restore the programmability for which manipulators are designed in the first place. This also fits in with the goals of the thesis, by exploring the kinematic modelling aspect and showing the physical problems encountered in this side of manipulator modelling, by showing their sources and suggesting a possible remedy. Like most of the methods in the literature this method suffers from a few drawbacks: a large number of measured points is needed to achieve a good estimate of the parameters, and measuring the end-effector Cartesian position can only be accomplished by a sophisticated system such as the Optotrack.

In the course of completing this work, it has been noticed that there are indications that a higher accuracy may be achieved by combining methods and measurement systems from different research groups active in the subject of manipulator calibration. For instance the use of the Optotrack developed in Surrey University, England by Professor Parker's research group [51] to provide measurements in combination with the methodology described by Driel in [27] should be investigated. The measurement can be obtained from a manipulator without removing it from its working environment and also without the constraints and conditions described in [27]. The estimated model parameters, on the other hand, seem to improve the accuracy to the same order of magnitude as the repeatability [27].

Chapter 7

Underwater Robot Modelling

7.1 Introduction

In previous chapters the modelling issue dealt with robot arm operating in normal conditions, whether that is in factory floors or in research labs, the only external force considered in this case is the gravity attraction. When this force is cancelled the model therefore depicts the behaviour of manipulator operating in space. It has also been established that obtaining such models automatically using existing modelling software is easily achieved by equalising the gravity force to zero. On the other hand, modelling robot arms operating underwater is not so easy and, still, not supported by commercially existing computer modelling software packages. In addition to changes in the gravity effects, the hydrodynamic effects are complex due to their dependence on the absolute value of the linear velocity of each link at various points and the shape of the links themselves.

The aim of this work is to investigate the dynamical changes a typical robot arm would endure when it is operated underwater and to include such changes in its model.

The use of underwater-robotic vehicles is nowadays common in maritime activities such as sea bed exploration, rescue and oil exploitation. Several

examples of such equipment are described in [90] and [91]

The importance of sea exploitation and the equipment involved in such activities has been addressed in a special issue of the I. E. E. E. Journal of Oceanic Engineering [12]. A key issue involved in underwater activities is that, due to the hostility of the under sea environment, the use of remotely or autonomous robotic vehicles is necessary and therefore is expanding rapidly

There is a considerable amount of reports on the modelling and control of underwater remotely operated vehicles (ROV's), for instance [31, 32, 33].

It is also common to provide ROV's with one or two robot arms in order to perform specific or generic tasks. In general the vehicle and its manipulators are operated remotely through direct human intervention [90, 91]. Many of the operations would be facilitated if the equipment could perform some of the tasks autonomously. In the case of manipulators, such a step requires a better understanding of their dynamics and also the interaction between the manipulator dynamics and that of the supporting platforms (*e.g.*. ROV).

Usual models of 'dry' arms require further considerations due to hydrodynamic effects: added mass, added inertia, drag, lift and buoyancy. In [34] a generic model for underwater arms has been proposed. This model includes added mass, drag and lift. Meanwhile in [92] the design of adaptive control schemes of underwater robot arms, modelled as suggested in [34], has been addressed. Nevertheless, a study of the particular hydrodynamic effects on a specific 'dry' model has not been reported.

In this chapter the hydrodynamic effects on a typical planar two degrees of freedom robot arm have been calculated. Added mass and drag torques and forces of the robot arm have been derived in an explicit form. Their effects were evaluated through a series of simulations, individually and in their combined complex form.

A controller was designed and applied to the two link manipulator based on disturbance attenuation. It controlled the arm movement successfully, especially

with the aid of the high damping caused by the drag.

The chapter is organised as follows. A review of a dry model is included in Section 7.2. The hydrodynamic effects and simplifying assumptions considered are presented in Section 7.3. The calculations of the added mass and drag effects are included in Subsections 7.3.1 and 7.3.2 respectively. Subsection 7.3.3 summarises the underwater robot arm model.

The results of the evaluation of the hydrodynamic effects through simulations are presented in Section 7.4. In Section 7.5 a control system for the underwater arm is proposed and its movement under active control is evaluated through simulation in section 7.6. Finally some conclusions and remarks end the chapter.

7.2 A typical robot arm model

The hydrodynamic forces acting on a body are described by functions which depend on the geometry of the body, the physical characteristics of the fluid and the relative velocity between the fluid and the body.

It is clear that hydrodynamic effects also affect robot arms operating under water. In [34] a generic model which incorporates these additional loads has been introduced. Nevertheless, it is desirable from the point of view of control design to have a more specific description of the dynamical changes endured by the mechanism. Such a description can only be obtained — as pointed out above, if the architecture of the arm and the geometry of its links are known.

In order to have a more descriptive representation of the hydrodynamic effects, a planar two degrees of freedom arm is considered as an example. The links of the arm are assumed to be circular cylinders.

The 'dry' model of such mechanism is well known [93]. This is presented next for completeness:

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}}(t) = \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{B}(\dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) + \boldsymbol{\tau} \tag{7.1}$$

Where:

$\boldsymbol{\theta} = (\theta_1, \theta_2)^T$ represents the joint angles,

$\boldsymbol{\tau} = [\tau_1, \tau_2]^T$ represents the drive torques applied at the joints.

$\mathbf{M}(\boldsymbol{\theta})$ represents the mass matrix.

$\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ represents the Coriolis and centripetal effects.

$\mathbf{B}(\dot{\boldsymbol{\theta}})$ is the term related to the viscous friction.

$\mathbf{G}(\boldsymbol{\theta})$ represents the gravity effects.

For the arm considered above and shown in figure 7.1.

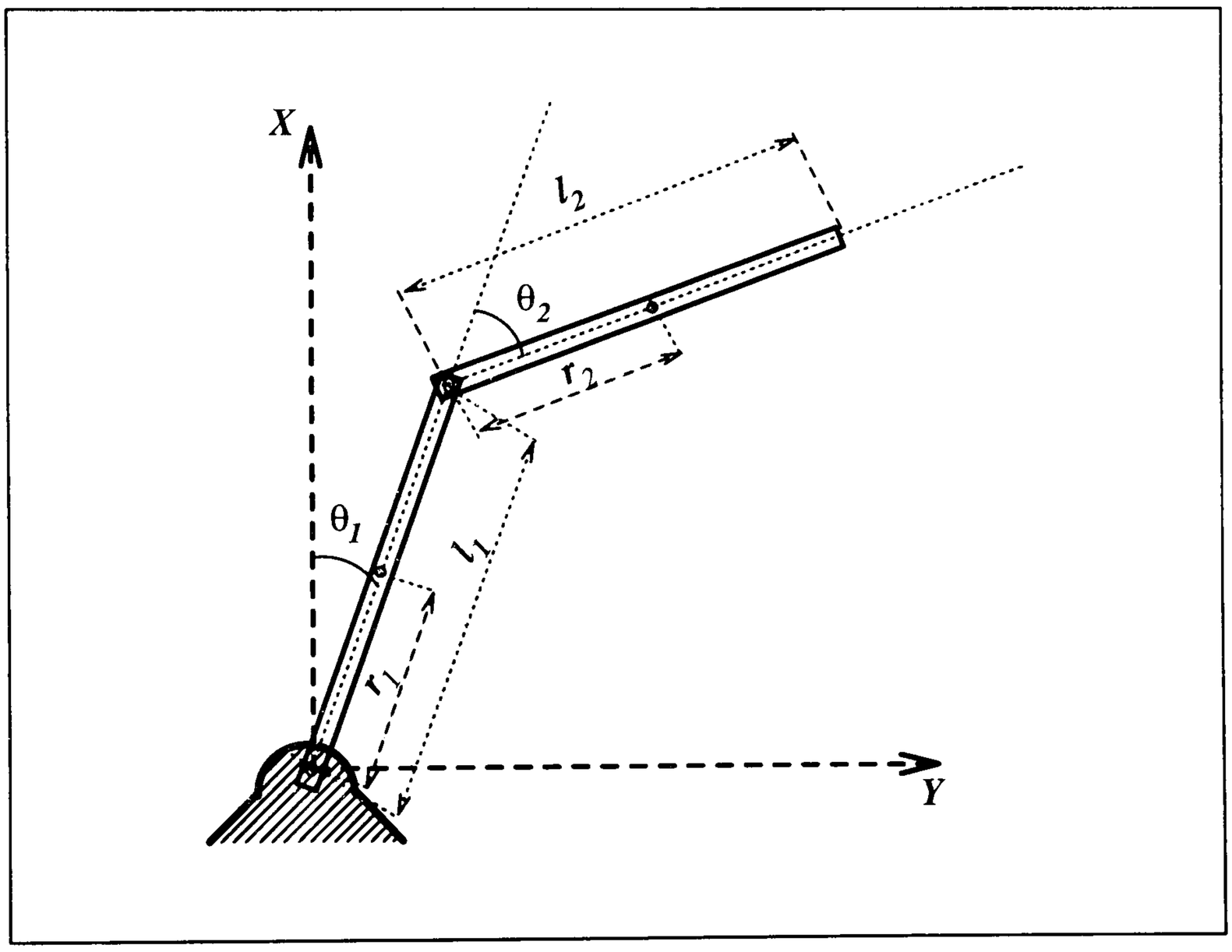


Figure 7.1: Two links planar manipulator

$$M(\boldsymbol{\theta}) = \begin{bmatrix} m_{11}(\boldsymbol{\theta}) & m_{12}(\boldsymbol{\theta}) \\ m_{21}(\boldsymbol{\theta}) & m_{22}(\boldsymbol{\theta}) \end{bmatrix} \tag{7.2}$$

$$m_{11}(\boldsymbol{\theta}) = I_{I0} + m_1 r_1^2 l_1^2 + m_2 r_2 l_2 C_2 + m_2 l_1^2 + m_{12}(\boldsymbol{\theta})$$

$$m_{12}(\boldsymbol{\theta}) = I_{02} + m_2 r_2 l_1 l_2 C_2 + m_2 r_2^2 l_2^2$$

$$m_{21}(\boldsymbol{\theta}) = m_{12}(\boldsymbol{\theta});$$

$$m_{22}(\boldsymbol{\theta}) = I_{02} + m_2 r_2^2 l_2^2$$

Where I_{0i} is the moments of inertia of link i; m_i is the mass of link i: r_i is the position of the centre of mass of link i; l_i is the length of link i and C_2 indicates $\cos\theta_2$, for $i = 1, 2$.

The vector function $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is:

$$\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = m_2 r_2 l_1 l_2 S_2 \begin{bmatrix} -1 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1^2 \\ (\dot{\theta}_1 + \dot{\theta}_2)^2 \end{bmatrix}$$

Where S_2 indicates $\sin\theta_2$

The friction term $\mathbf{B}(\dot{\boldsymbol{\theta}})$ can be represented as:

$$\mathbf{B}(\dot{\boldsymbol{\theta}}) = \begin{bmatrix} B_1\dot{\theta} \\ B_2\dot{\theta} \end{bmatrix}$$

In order not to complicate the model more than necessary, it will be assumed that the arm operates on a horizontal plane, thus the term due to gravity and buoyancy will not be included.

7.3 Hydrodynamic Effects

The relative motion between a cylinder immersed in a fluid and the fluid (water) gives rise to an interacting force opposing the motion. Such force can be expressed as the addition of two effects, namely:

$$\text{Hydraulic force} = \text{drag force} + \text{inertia force}$$

The drag term is associated to the change of pressure the fluid experiences when is perturbed by the cylinder. If it is assumed that the fluid is at rest and the cylinder moves, the inertial term accounts for the amount of fluid transported together with the cylinder.

The hydraulic force can be calculated according to the so called Morison equation [94]:

$$F_h = \int_0^l RC_d\rho|v(x)|v(x)dx + \int_0^l C_a\rho\pi R^2\dot{v}(x)dx \tag{7.3}$$

Where:

C_d is the drag coefficient,

ρ is the density of the water,

R is the radius of the cylinder,

l is the length of the cylinder,

$v(x)$ is the perpendicular component of the fluid velocity with respect to the cylinder

$\dot{v}(x)$ is the time derivative of $v(x)$.

The Morison equation (7.3) is fundamental for the calculation of the hydrodynamics of the arm. Its application depends on the relative velocity $v(x)$ associated with each link. It is also clear that added mass and drag effects can be calculated separately. These calculations are presented and discussed in the next few sections.

7.3.1 Added mass

The principle involved in the concept of this additional inertia effect, is that a water particle moving in a flow carries a momentum with it. As water particles pass around a circular cylinder they accelerate and then decelerate. Therefore, work has to be done through the application of a force on the cylinder to increase this momentum. The same principle applies if the water is at rest and the cylinder moves.

The calculation of the added mass effect on the robot arm described can be carried out according to the second term of the Morison equation (7.3):

$$F_{a\,i}(\dot{v}) = \int_0^l C_a \rho \pi R^2 \dot{v}_i(x) dx \tag{7.4}$$

By examining the second part of the above equation before integration, $C_a \rho \pi R^2$, it represents the added *apparent mass* per unit length. This results from some particles of the water being permanently displaced by the moving cylindrical link. Before applying (7.4) the following considerations are introduced:

C.1 The fluid is considered to be at rest while the arm moves,

C.2 the term $v_i(x)$ is the absolute linear velocity of a point on link-i situated at a distance x from the joint,

C.3 the added mass coefficients are considered constant, although the added mass coefficients are functions of the posture of the arm and adjacent bodies.

C.4 the added mass due to the flow parallel to the second link is negligible

In order to apply (7.4) the velocity components of the fluid perpendicular to the links are first determined.

The linear velocity of a point on link-1 at a distance x from the axis of rotation is:

$$v_1(x) = \dot{\theta}_1 x, \tag{7.5}$$

Thus:

$$\frac{d}{dt} v_1(x) = \ddot{\theta}_1 x. \tag{7.6}$$

The linear velocity of a point of link-2 at a distance x from the joint axis has two components, one perpendicular and the other parallel to the link. The perpendicular component is:

$$v_{2n}(x) = \dot{\theta}_1 l_1 C_2 + (\dot{\theta}_1 + \dot{\theta}_2) x, \tag{7.7}$$

whereas the parallel is:

$$v_{2l}(x) = \dot{\theta}_1 l_1 S_2, \tag{7.8}$$

By considering C.4 the added mass due to $dv_{2l}(x)/dt$ is neglected. This simplification is justified by the fact that the area of the cross section of link-2 is negligible compared with the area exposed to the velocity component v_{2n}.

Then

$$\frac{d}{dt} v_{2n}(x) = (l_1 C_2 + x)\ddot{\theta}_1 + x\ddot{\theta}_2 - (l_1 S_2)\dot{\theta}_1\dot{\theta}_2 \tag{7.9}$$

By replacing the expression for v_1 in (7.4) the force caused by the added mass of link-1 is obtained:

$$f_1 = C_a \rho \pi R_1^2 \int_0^{l_1} \ddot{\theta}_1 \, x dx \tag{7.10}$$

Meanwhile the added mass (force) of link-2 is given by:

$$f_2 = C_a \rho \pi R_2^2 \int_0^{l_2} \left[l_1 C_2 \ddot{\theta}_1 + x(\ddot{\theta}_1 + \ddot{\theta}_2) - (l_1 S_2)\dot{\theta}_1\dot{\theta}_2 \right] dx \tag{7.11}$$

Then it is clear that the torque due to the added mass (7.10) on link-1 is:

$$\tau_1 = \frac{1}{3} K_{a1} l_1^3 \, \ddot{\theta}_1 \tag{7.12}$$

and the torque due to the added mass (7.11) on link-2 is:

$$\tau_2 = K_{a2} \left(\frac{1}{3} l_2^3 + \frac{1}{2} l_1 l_2^2 C_2 \right) \ddot{\theta}_1 + K_{a2} \frac{1}{3} l_2^3 \, \ddot{\theta}_2 - K_{a2} \frac{1}{2} l_1 l_2^2 S_2 \, \dot{\theta}_1 \dot{\theta}_2, \tag{7.13}$$

where the constants K_{ai} are defined as follows [94]:

$$K_{a1} = C_a \rho \pi R_1^2 \tag{7.14}$$

$$K_{a2} = C_a \rho \pi R_2^2 \tag{7.15}$$

If the added mass effect is considered as an external load then it can be expressed as:

$$T_{added} = \begin{bmatrix} \tau_{a1} + \tau_{a2} \\ \\ \tau_{a2} \end{bmatrix} \tag{7.16}$$

It is evident therefore that this external load has an added mass/inertia effect as well as a Coriolis effects caused by the appearance of the term related to $\dot{\theta}_1\dot{\theta}_2$ and hence (7.16) can be rewritten as:

$$T_{added} = M_a(\theta_1, \theta_2) \begin{bmatrix} \ddot{\theta}_1 \\ \\ \ddot{\theta}_2 \end{bmatrix} + C_a\, \dot{\theta}_1\dot{\theta}_2. \tag{7.17}$$

Where,

$$M_a(\theta_1, \theta_2) = \begin{bmatrix} \frac{1}{3}K_{a1}l_1^3 + K_{a2}l_1^2S_2^2 + K_{a2}\left(\frac{1}{3}l_2^3 + \frac{1}{2}l_1l_2^2C_2\right) & K_{a2}\frac{1}{3}l_2^3 \\ K_{a2}\left(\frac{1}{3}l_2^3 + \frac{1}{2}l_1l_2^2C_2\right) & K_{a2}\frac{1}{3}l_2^3 \end{bmatrix} \tag{7.18}$$

and

$$C_a(\theta_1, \theta_2) = \begin{bmatrix} K_{a2}l_1^2S_2C_2 - K_{a2}\frac{1}{2}l_1l_2^2S_2 \\ \\ -K_{a2}\frac{1}{2}l_1l_2^2S_2 \end{bmatrix} \tag{7.19}$$

It is easy to show that matrix (7.18) is in effect positive definite with constant determinant

$$\frac{1}{9}K_1K_2l_1^3l_2^3$$

Thus the addition of (7.18) to (7.2) can be considered a virtual mass matrix.

7.3.2 Drag

The drag effects can be determined by considering the first term of the Morison equation (7.3), that is:

$$d\,f_D = -RC_d\rho\,|v(x)|\,v(x) \tag{7.20}$$

This is defined as the force per unit length necessary to hold a cylinder at rest, to a constant free stream velocity v [94]. In the case where the cylinder is moving in a constant fluid, it is the force per unit length resisting the motion.

The drag torque endured by link-1 of the manipulator is calculated as follow:

$$T_{11} = -R_1^2 C_d \rho \int_0^{l_1} |\dot{\theta}_1 x| \, \dot{\theta}_1 x \, x dx \tag{7.21}$$

The evaluation of the above integral (7.21) results in:

$$T_{11} = \frac{1}{4} K_{d1} \, \text{sign}(\dot{\theta}_1) \, \dot{\theta}_1{}^2 \, l_1{}^4 \tag{7.22}$$

where

$$K_{d1} = -R_1{}^2 \, C_d \, \rho. \tag{7.23}$$

The drag effect of link-2 is determined by the relative velocity between the link and the fluid, that is equation (7.7). Thus the torque due to drag on link-2 about its axis is represented by:

$$T_{22} = K_{d2} \int_0^{l_2} \left| (\dot{\theta}_1 + \dot{\theta}_2)x + \dot{\theta}_1 l_1 C_2 \right| \left[(\dot{\theta}_1 + \dot{\theta}_2)x + \dot{\theta}_1 l_1 C_2 \right] \, x dx \tag{7.24}$$

where

$$K_{d2} = -R_2{}^2 \, C_d \, \rho. \tag{7.25}$$

The velocity term is not simple and the evaluation of its absolute value yields several possibilities. In general the linear normal velocity of link-2 could change its sign along the length depending on the velocity values at the extremities of the link. This evaluation is closely related to the angular velocities of the two links and is the cause of the presence of the reverse flow. The reverse flow occurs when the normal velocity changes sign along the length of link-2.

In order to evaluate the drag torque term (7.24), let equation (7.7) be evaluated at the joint of link-2, that is $x = 0$:

$$V1 = \dot{\theta}_1 \, l_1 \, C_2, \tag{7.26}$$

and at the end of link-2, that is $x = l_2$:

$$V2 = (\dot{\theta}_1 + \dot{\theta}_2)\, l_2 + \dot{\theta}_1\, l_1\, C_2, \tag{7.27}$$

Let l_0 be a function which reflects the position where the direction of the velocity changes along link-2 as shown in figure 7.4:

$$l_0 = -\frac{l_2\, V1}{(V2 - V1)} = -\frac{\dot{\theta}_1\, l_1\, C_2}{(\,\dot{\theta}_1 + \dot{\theta}_2\,)}. \tag{7.28}$$

The value of l_0 determines the presence of reverse flow on link-2 and therefore assist the evaluation of equation (7.24). As shown in figure 7.4, both possibilities of reverse flow are characterised by:

$$l_0 \in \ ]\, 0\ ,\ l_2\, [\,. \tag{7.29}$$

Considering the signs of the linear velocity at the extremities of link-2 we have four possibilities for the evaluation of equation (7.24). These are summarised in the following four cases.

Case 1: $V1 \geq 0$ and $V2 \geq 0$

In this case there is no presence of reverse flow acting on link-2 and its normal velocity is therefore positive along the its length except when $V1 = V2 = 0$ where, $v_{2n}(x)$ is nil for any distance x between 0 and l_2. The distribution of the normal velocity for this case is illustrated in figures 7.2 a and 7.3 a.

Equation (7.24) is written therefore in the following form:

$$T_{22} = K_{d2} \int_0^{l_2} \left((\dot{\theta}_1 + \dot{\theta}_2)\, x + \dot{\theta}_1 l_1 C_2\right)^2 x dx. \tag{7.30}$$

The evaluation of the above integral as a function of x is:

$$T_3(x) = K_{d2} \left[\frac{1}{3}\,(\dot{\theta}_1 + \dot{\theta}_2)^2\, x^3 + \frac{1}{2}\,(\dot{\theta}_1 l_1 C_2)^2\, x^2 + \frac{2}{3}\,\dot{\theta}_1 l_1 C_2\,(\dot{\theta}_1 + \dot{\theta}_2)\, x^3\right]. \tag{7.31}$$

Considering the limits of (7.30) in (7.31) the value of the torque (7.24) is calculated by:

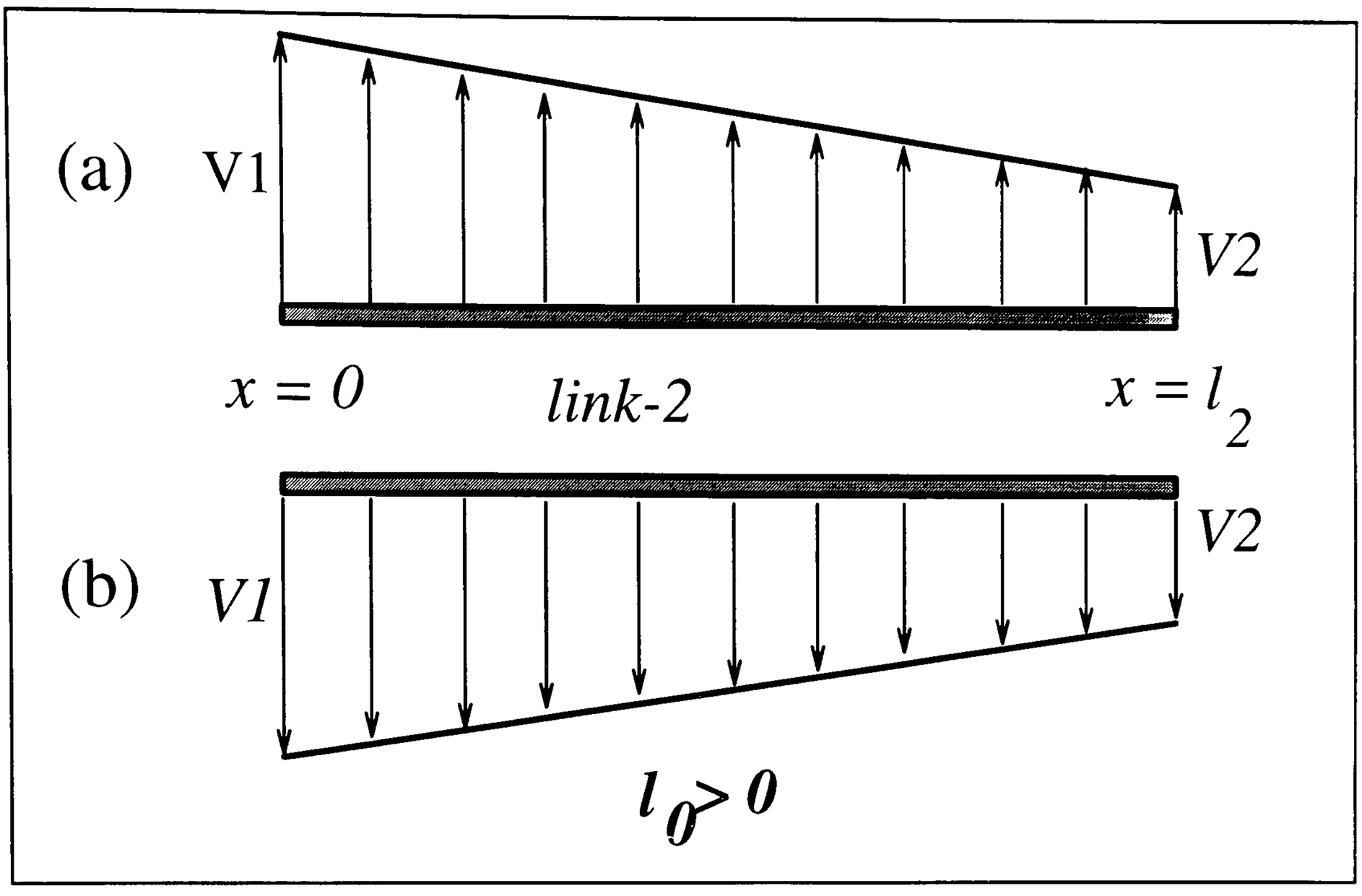


Figure 7.2: Velocity distribution of link-2 with $l_0 \geq l_2$; no reverse flow.

$$T_{22} = T_3(l_2). \tag{7.32}$$

Case 2: $V1 \geq 0$ and $V2 < 0$

This case is characterised by the presence of reverse flow which is caused by the change of direction of the normal velocity $v_{2n}(x)$, at a point l_0, somewhere along link-2. That is $l_0 \in]0,\ l_2[$ see figure 7.4 a. The integral of (7.24) is therefore split, according to the sign of the velocity, into two integrals of the following form :

$$T_{22} = K_{d2} \left[\int_0^{l_0} \left((\dot{\theta}_1 + \dot{\theta}_2)\, x + \dot{\theta}_1 l_1 C_2 \right)^2 - \int_{l_0}^{l_2} \left((\dot{\theta}_1 + \dot{\theta}_2) x + \dot{\theta}_1 l_1 C_2 \right)^2 \right] x dx \tag{7.33}$$

The above expression is solved as a sum of two integrals which differ only in sign and limits. Each term of (7.33) is of the form of equation (7.31), and therefore the sum of the above integrals is given by:

$$T_{22} = [\,(T_3(l_0) - T_3(0)\,) - (\,T_3(l_2) - T_3(l_0)\,)] \tag{7.34}$$

and finally it becomes:

$$T_{22} = 2\,T_3(l_0) - T_3(l_2) \tag{7.35}$$

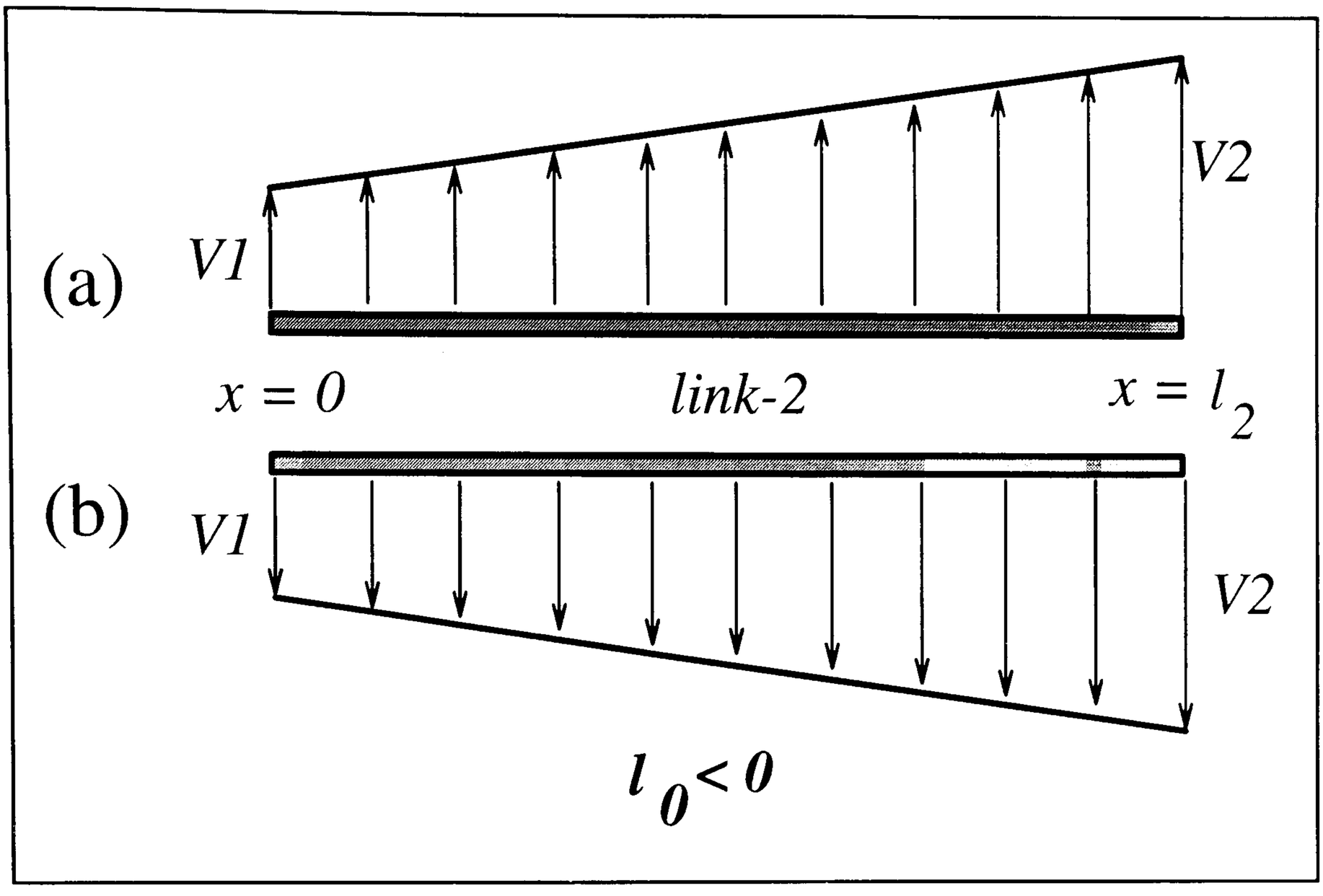


Figure 7.3: Velocity distribution of link-2 with $l_0 \leq 0$; no reverse flow.

Case 3: $V1 < 0$ and $V2 > 0$

This case is symmetric to the previous as illustrated in figure 7.4 b and is also characterised by the presence of reverse flow. Therefore the integral expression of (7.24) is given by:

$$T_{22} = K_{d2}\left[-\int_0^{l_0}\left((\dot{\theta}_1+\dot{\theta}_2)\,x+\dot{\theta}_1 l_1 C_2\right)^2+\int_{l_0}^{l_2}\left((\dot{\theta}_1+\dot{\theta}_2)x+\dot{\theta}_1 l_1 C_2\right)^2\right]\,x dx \tag{7.36}$$

The above expression differs only in sign from equation (7.33) and all other condition are similar. Thus the drag expression is:

$$T_{22} = -T_3(l_2) - 2\,T_3(l_0), \tag{7.37}$$

which is (7.35) multiplied by minus one.

Case 4: $V1 < 0$ and $V2 \leq 0$

In this case no change of velocity sign takes place along the length of link-2. see figure 7.2 b and 7.3 b. The value of the velocity $v_{2n}(x)$ is negative for all $x \in [0,\, l_2]$ except when $V2 = 0$. In this particular case $l_0 = l_2$ which still does not

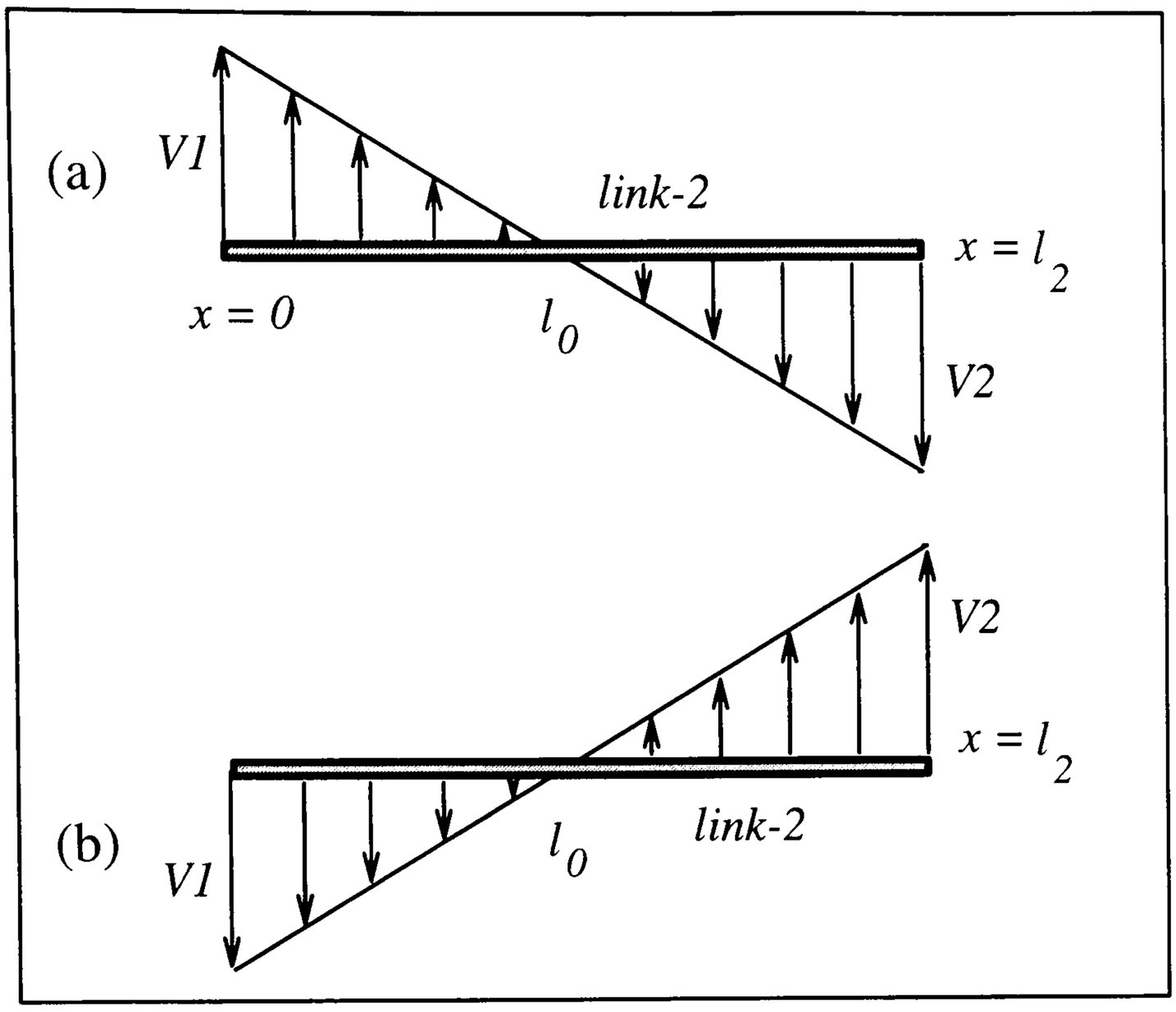


Figure 7.4: The two possibilities of Velocity distribution of link-2 causing reverse flow.

cause reverse flow. The drag torque expression (7.24) is rewritten therefore in the following form:

$$T_{22} = K_{d2} \int_0^{l_2} -\left((\dot{\theta}_1 + \dot{\theta}_2)\, x + \dot{\theta}_1 l_1 C_2\right)^2 x dx, \tag{7.38}$$

which is exactly the negative form of (7.30) and hence the solution yields the negative form of (7.32) and the drag torque is then given by:

$$T_{22} = -T_3(l_2). \tag{7.39}$$

The total drag effect of the two link cylindrical manipulator can be written in a vector form as follows:

$$\boldsymbol{\tau}_{drag} = \begin{bmatrix} \tau_{drag1} + \tau_{drag2} \\ \\ \tau_{drag2} \end{bmatrix}, \tag{7.40}$$

where:

$\tau_{drag1} = T_{11}$ according to equation (7.22) and $\tau_{drag2} = T_{22}$. With the function T_{22}

being determined by the angular velocities of the links as indicated in the above cases.

7.3.3 The Underwater Robot Model

After producing the model of the dry robot and also calculating the hydrodynamic effects it is possible to superimpose these effects upon the dry robot model in order to formulate the model of an underwater robot. The combined model therefore depicts the behaviour of a two link manipulator with cylindrical links operating underwater. The model is then presented in the general form:

$$\begin{aligned}[\mathbf{M}(\boldsymbol{\theta}) + \mathbf{M}_a(\boldsymbol{\theta})]\,\ddot{\boldsymbol{\theta}}(t) \;&=\; \mathbf{C}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \mathbf{C}_a(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \mathbf{B}(\dot{\boldsymbol{\theta}}) \\ &\quad + \boldsymbol{\tau}_{drag}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) + \boldsymbol{\tau},\end{aligned} \tag{7.41}$$

A MATLAB file containing the above underwater two link manipulator model is presented appendix D.

7.4 Evaluation of the hydrodynamic effects

In addition to being cylindrical, let also the links be of homogeneous material. Then the values of r_1 and r_2 of equation (7.2) are equal to $\frac{1}{2}$, which means that the centre of gravity of each link is half way along its length.

In order to evaluate the hydrodynamic effects, these are included in the dry robot model separately. The dynamical changes under the hydrodynamic effects are evaluated through simulations, which consider a combination of these effects.

For all simulations the driving torques applied on link-1 and link-2 are:

$$\tau_1 = \begin{cases} 1 \text{ Nm} & \text{if } 0 \le t < 1\,sec \\ 0 & \text{otherwise} \end{cases}$$

and

$$\tau_2 = \begin{cases} 1 \text{ Nm} & \text{if } 5 \le t < 6\,sec \\ 0 & \text{otherwise.} \end{cases}$$

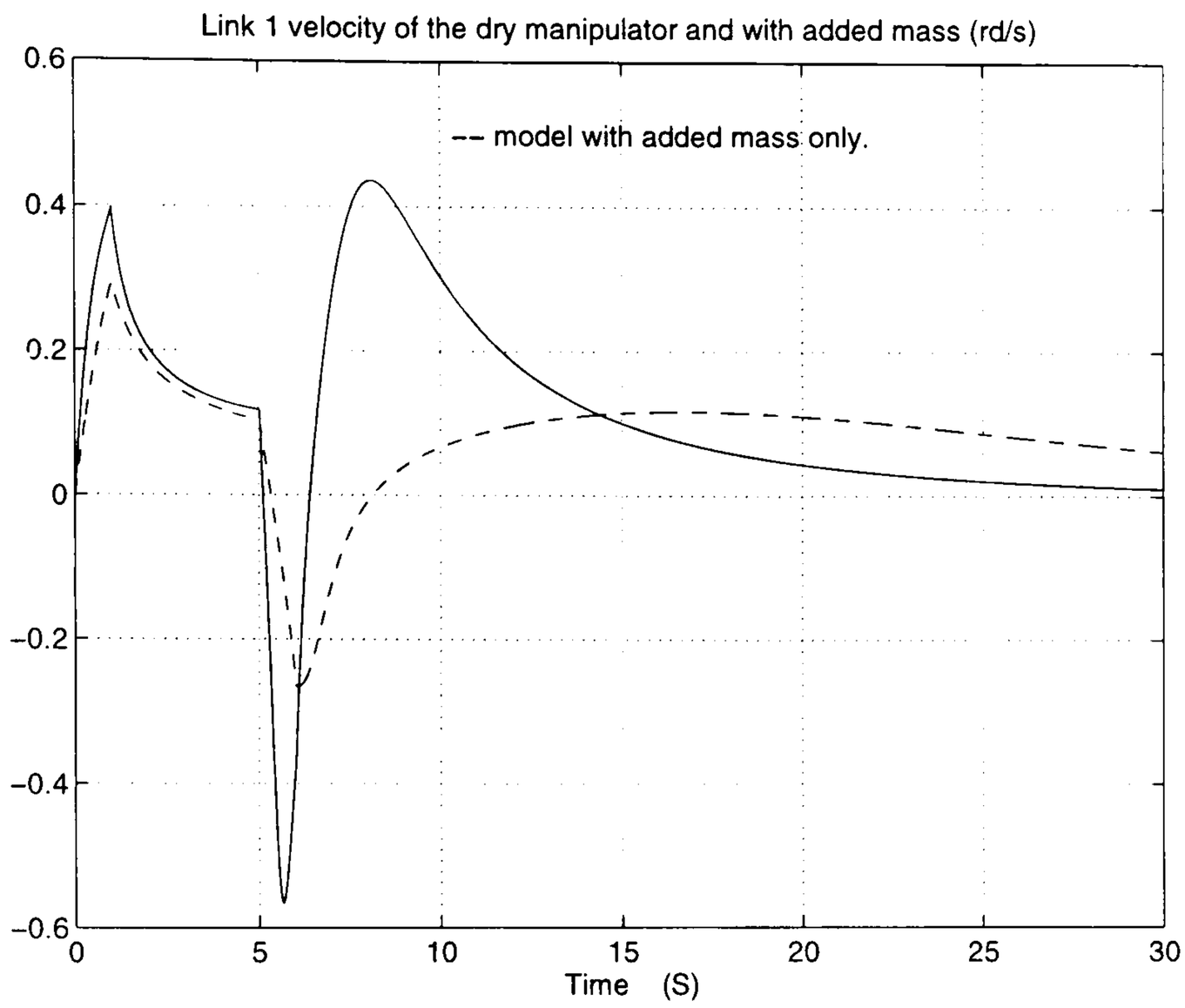
Other parameters needed for the simulations are the drag coefficient. added mass coefficient and the water density which are set to $C_d = 1$. $C_a = 2$ according to the shape and dimensions of the links [94] and $\rho = 1025kg/m^3$. The masses of the links are set to $2kg$ each and their lengths both to $1m$.

The added mass effect is equivalent to increasing the mass of the two links. It was also noted that the added Coriolis part (7.18) caused by the added mass has smaller effects when compared with the inertia part. The change of behaviour of the dry arm with respect to the dry arm plus the added mass terms is illustrated in figure 7.5 and 7.6.

Figure 7.7 and 7.8 illustrate the case when only the drag effects are included in the model. The links of the manipulator eventually stop moving under the drag force resisting the movement. The difference between the positions of the dry robot model and that with drag effects appear just after the start of the move and are considerable. Figure 7.9 and 7.10, however depict the comparison between a manipulator with drag effects only and one with complete hydrodynamic effects (*i.e.* underwater manipulator). It is evident from the response of link-1 that the added mass has a lesser effect in comparison with the drag effects. However, both effects have important values and neither of them can be neglected.

7.5 Control design

The control design proposed next is based on the design of two scalar controllers for each link. This approach results in a multivariable diagonal controller. The interaction of the links, which is very strong, has been considered as disturbances. The aim of the design is to achieve trajectory tracking with disturbance rejection. The original scalar design was first proposed in [95] and is summarised as follows.



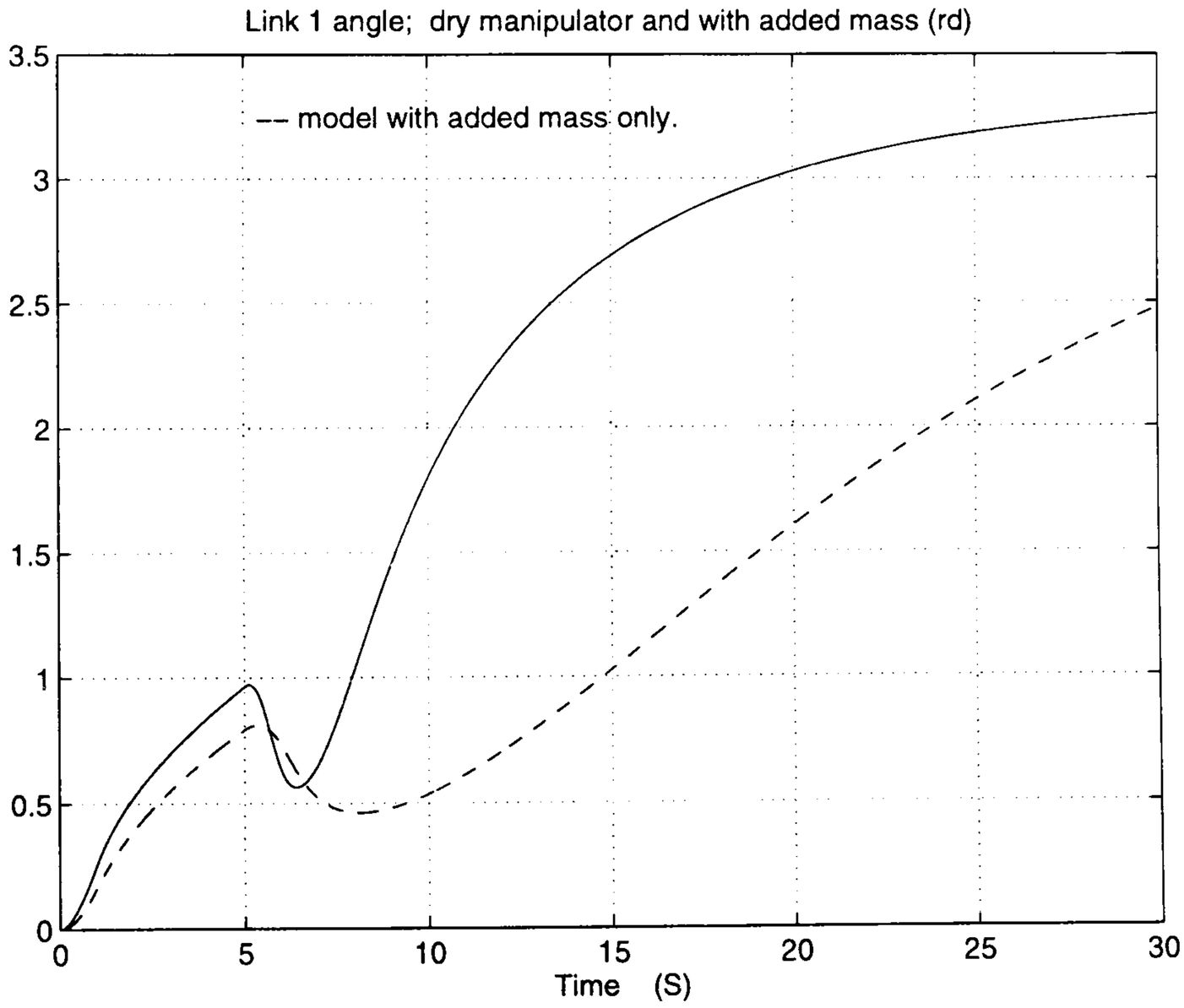


Figure 7.5: Comparison of dry manipulator and the one with added mass effects only: link 1 velocity and angular position.

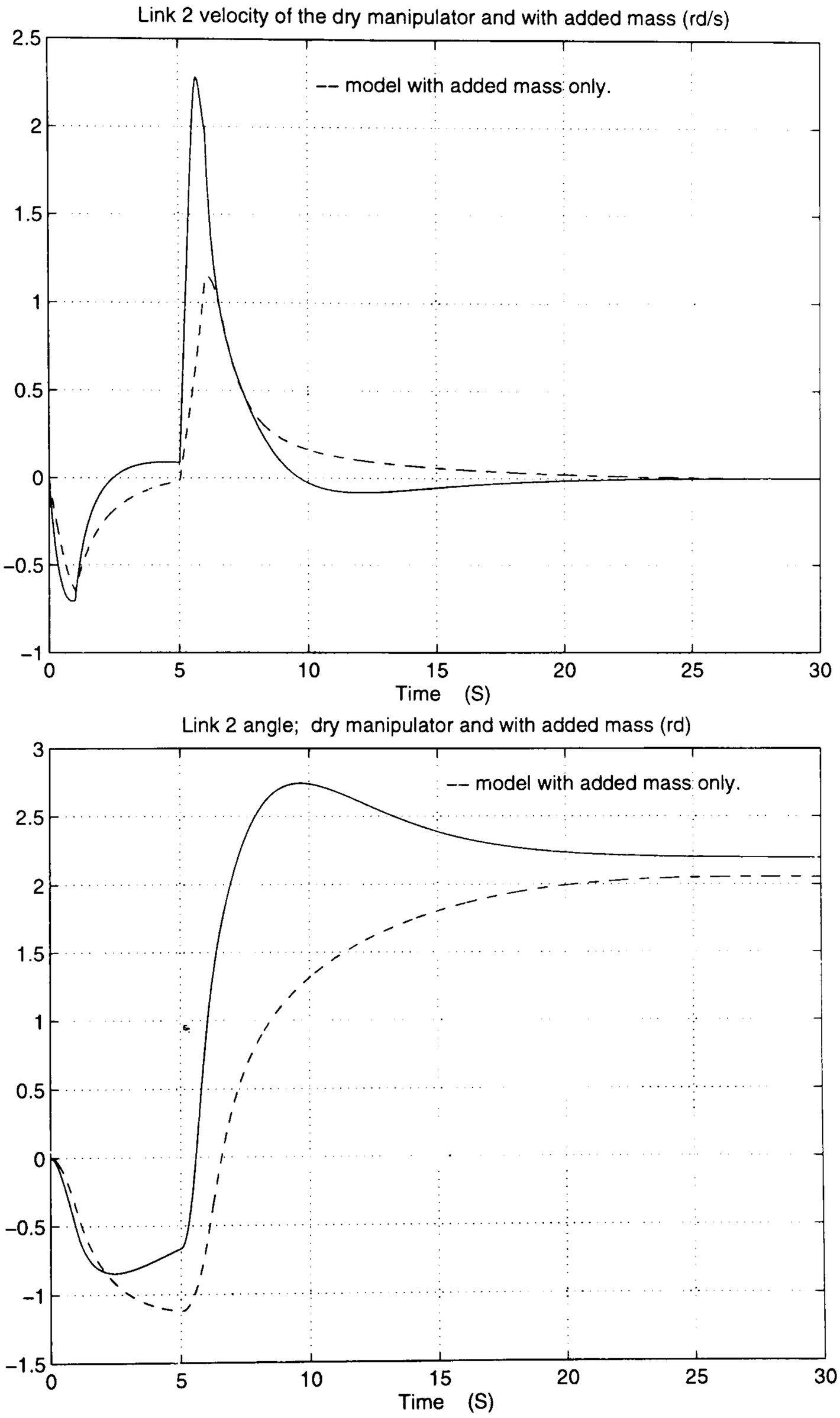
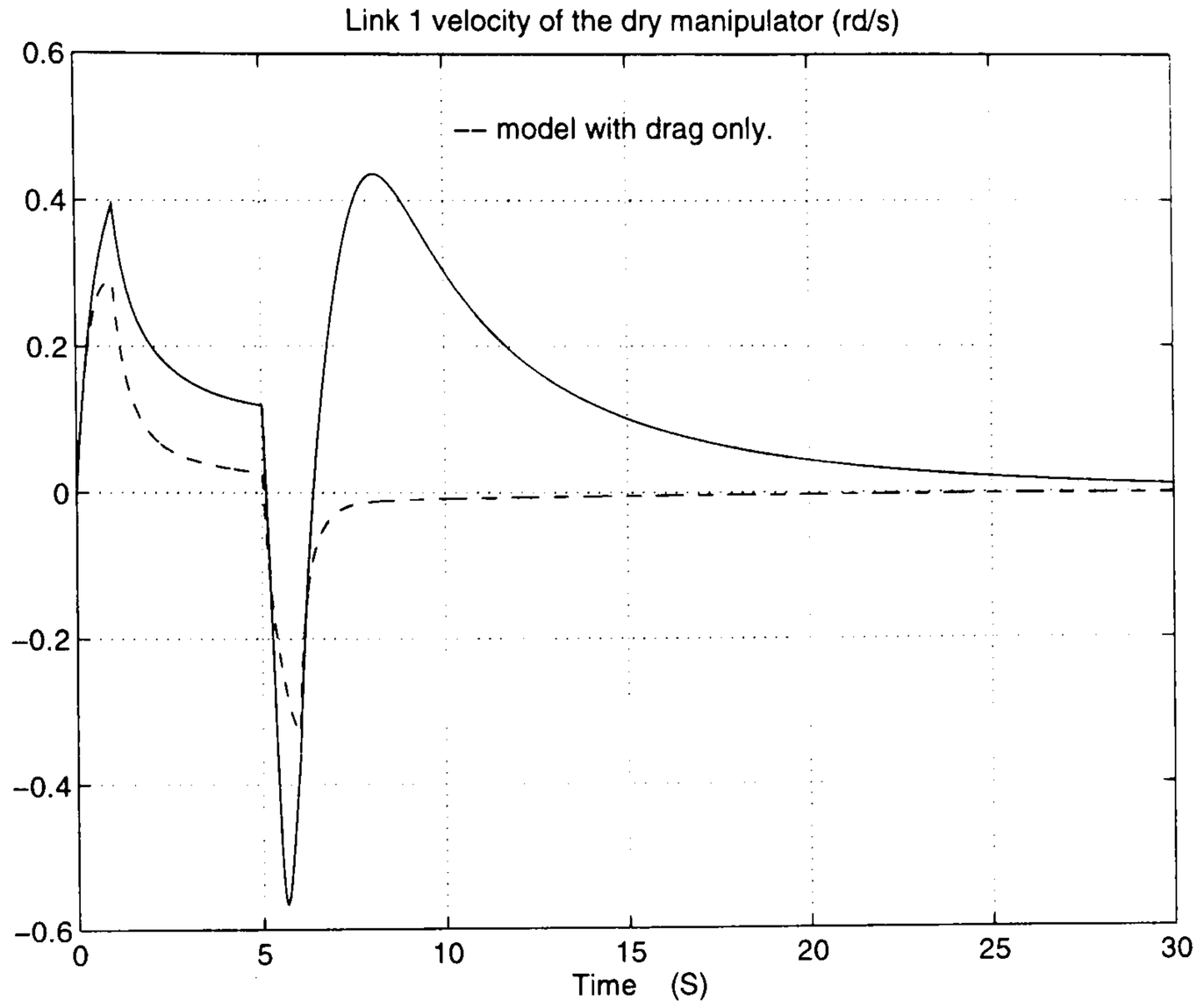


Figure 7.6: Comparison of dry manipulator and the one with added mass effects only; link 2 velocity and angular position.



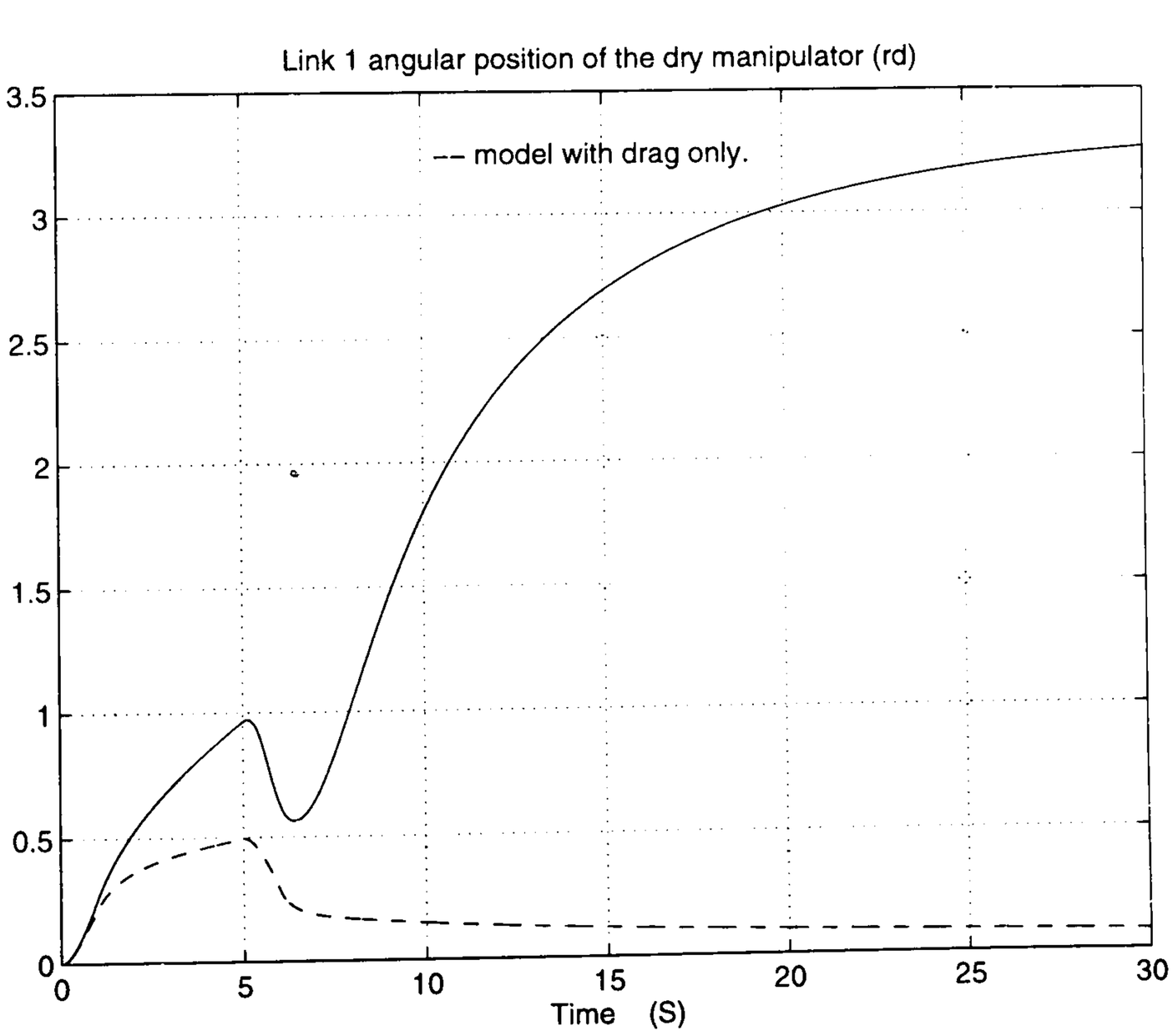


Figure 7.7: Comparison of dry manipulator and the one with drag effects only: link 1 velocity and angular position.

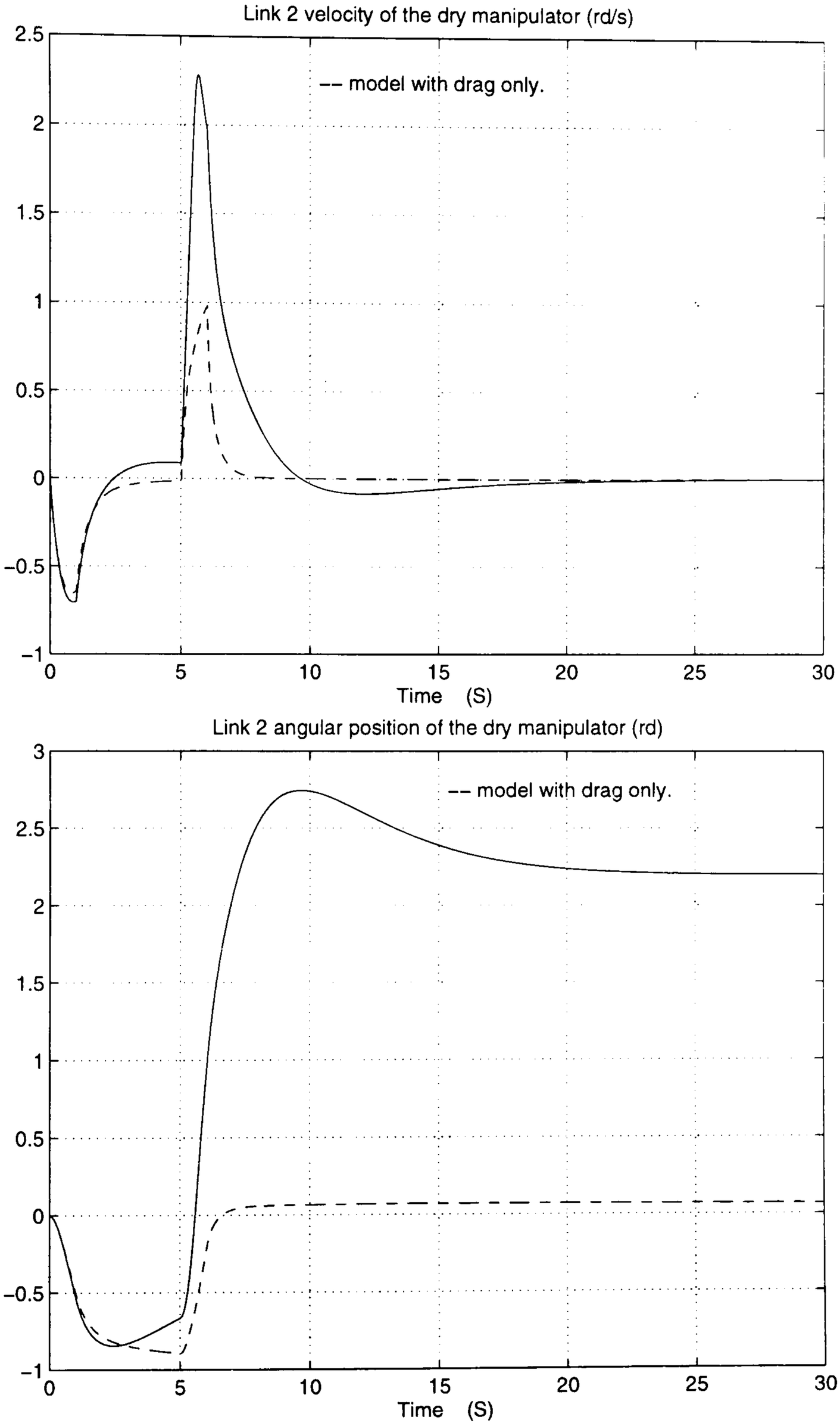


Figure 7.8: Comparison of dry manipulator and the one with drag effects only: link 2 velocity and angular position.

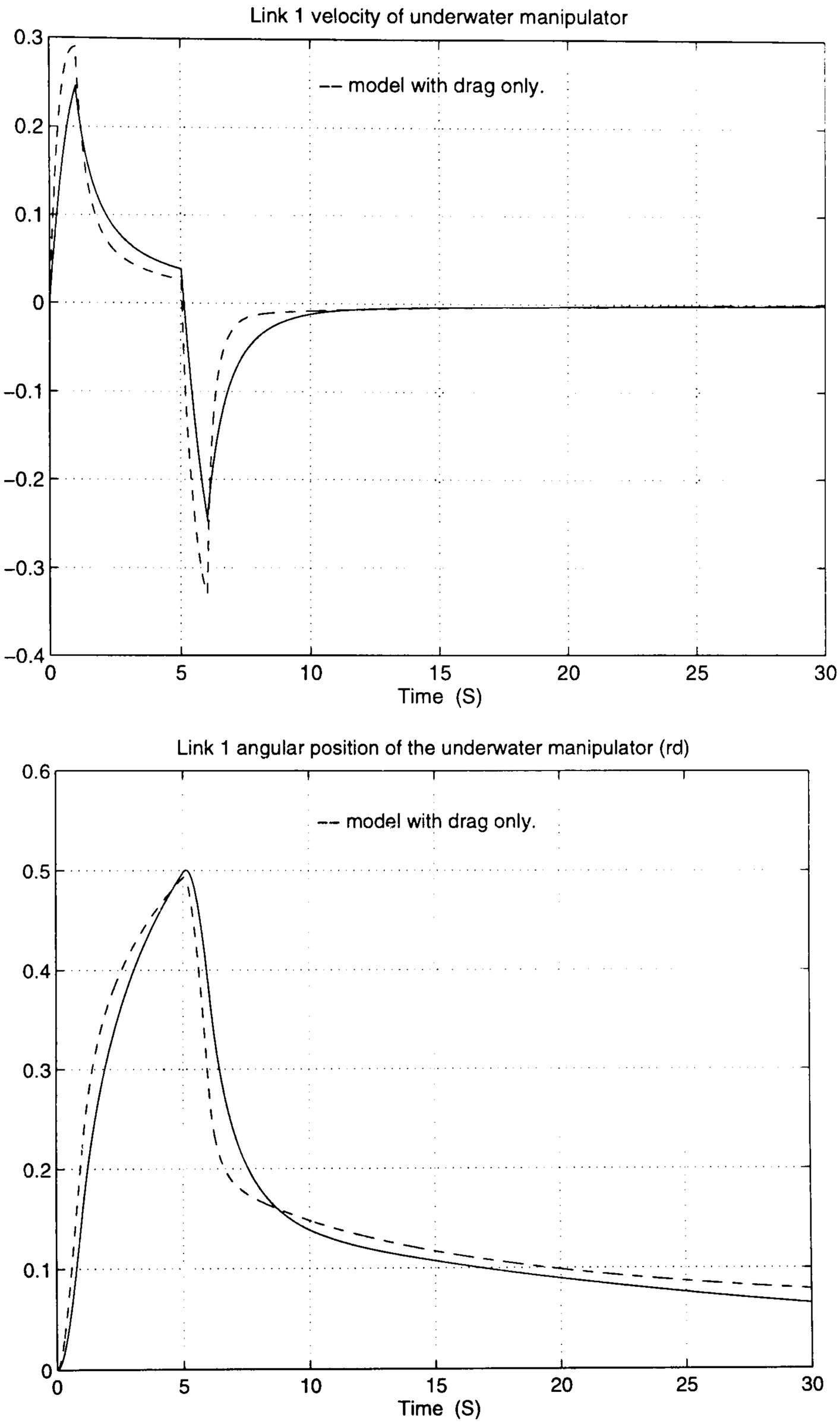


Figure 7.9: Comparison of underwater manipulator and one with drag effects only; link 1 velocity and angular position.

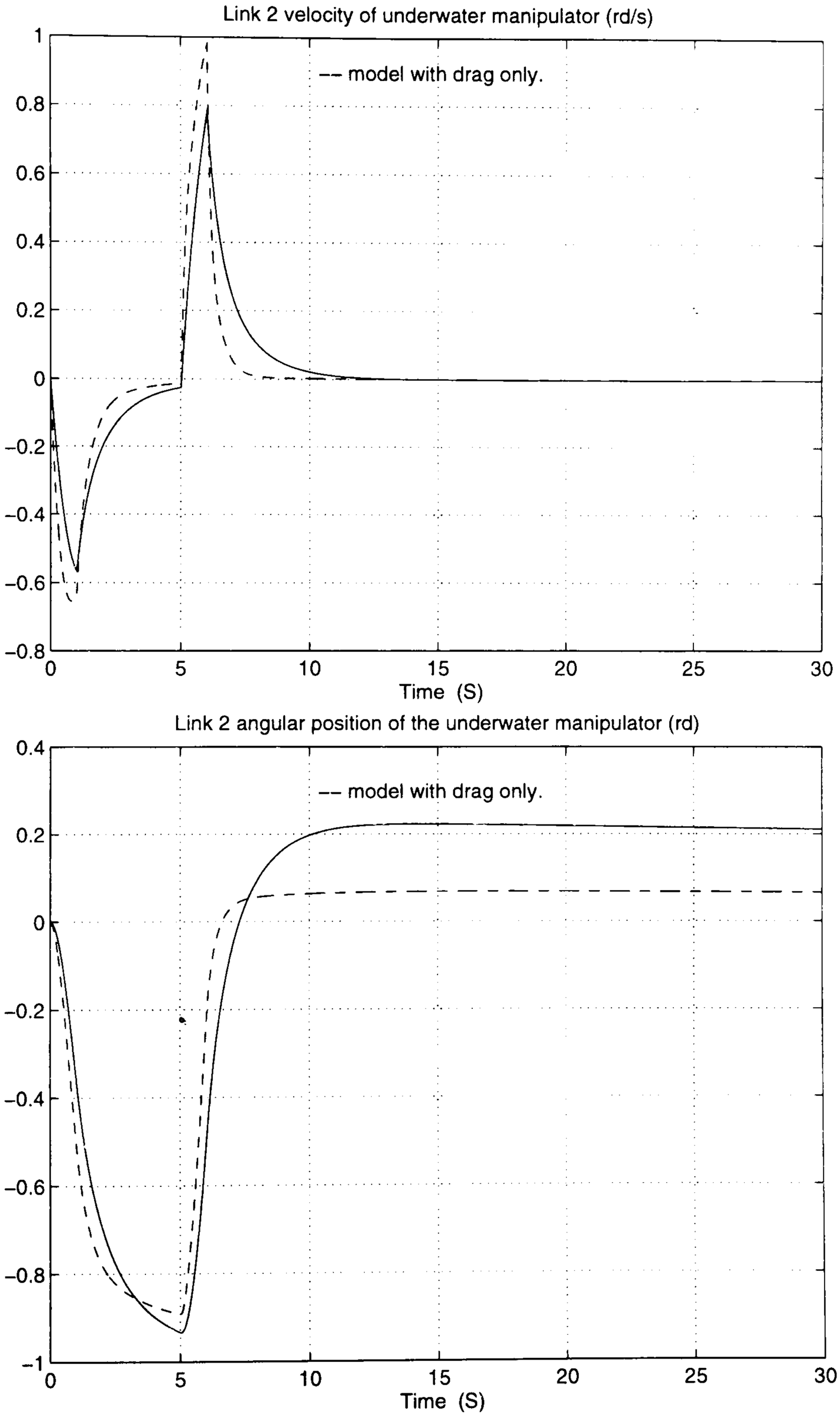


Figure 7.10: Comparison of underwater manipulator and one with drag effects only; link 2 velocity and angular position.

7.5 Control design

The control design proposed next is based on the design of two scalar controllers for each link. This approach results in a multivariable diagonal controller. The interaction of the links, which is very strong, has been considered as disturbances. The aim of the design is to achieve trajectory tracking with disturbance rejection. The original scalar design was first proposed in [95] and is summarised as follows.

In order to describe the control design let the underwater arm be described by

$$\Sigma : \begin{cases} \dot{x} = f(x) + g_1(x)u_1 + g_2(x)u_2, \\ y = [h_1(x) \quad h_2(x)]^T \end{cases} \tag{7.42}$$

The control system is composed of individual controllers formed by the following subsystems:

M.1 A non-linear compensator described by differential equations of the form:

$$\Sigma_m : \begin{cases} \dot{x}_m = f_m(x_m) + g_m(x_m)u, \\ y_m = h_m(x_m) \end{cases} \tag{7.43}$$

where the state $x_m \in I\!R^n$, $u \in I\!R$ is the same control input as in (7.42), and $y_m \in I\!R$ is the model output. f_m and g_m are smooth vector fields and h_m is a smooth function. Moreover, it is assumed that the point x_{m_0} is an equilibrium point for the unforced dynamics $\dot{x}_m = f_m(x_m)$, i.e., $f_m(x_{m_0}) = 0$, and $h_m(x_{m_0}) = 0$.

M.2 A *tracking model* represented by

$$\Sigma_t : \begin{cases} \dot{x}_t = A_t x_t + b_t z, \\ y_t = C_t x_t \end{cases} \tag{7.44}$$

where the state $x_t \in I\!R^{n_t}$, $z \in I\!R$ is the input to the tracking model and $y_t \in I\!R$ is the corresponding output. $A_t \in I\!R^{n_t \times n_t}$, $b_t \in I\!R^{n_t \times 1}$, and

where the state $x_r \in \mathbb{R}^{n_r}$, while $\varepsilon \in \mathbb{R}$ and $y_r \in \mathbb{R}$ are the regulation model input and output, respectively. $A_r \in \mathbb{R}^{n_r \times n_r}$, $b_r \in \mathbb{R}^{n_r \times 1}$, and $C_r \in \mathbb{R}^{1 \times n_r}$, are real matrices. As in the tracking model, the matrix A_r has eigenvalues with negative real part.

The input z, of the tracking model Σ_t, represents the desired output of the real process, thus a tracking error can be defined, for instance:

$$e_t = y_t - y \tag{7.46}$$

Meanwhile, the input ε, of the regulation model Σ_r, is defined as:

$$\varepsilon = y - y_m \tag{7.47}$$

In addition, we assume that Σ_m, Σ_t and Σ_r have relative degrees d_m, d_t and d_r, respectively.

An associated extended system is formed as:

$$\Sigma_E : \begin{cases} \dot{x}_E = f_E(x_E) + g_E(x_E)u + p_{E_1}(x_E)z + p_{E_2}(x_E)y, \\ y_E = h_E(x_E) \end{cases} \tag{7.48}$$

with state $x_E = \mathrm{col}(x_m, x_t, x_r)$, inputs u, z and y, with

$$f_E(x_E) = \begin{bmatrix} f_m(x_m) \\ A_t x_t \\ A_r x_r - b_r h_m(x_m) \end{bmatrix}, \qquad g_E(x_E) = \begin{bmatrix} g_m(x_m) \\ 0 \\ 0 \end{bmatrix},$$

$$p_{E_1}(x_E) = \begin{bmatrix} 0 \\ b_t \\ 0 \end{bmatrix}, \qquad p_{E_2}(x_E) = \begin{bmatrix} 0 \\ 0 \\ b_r \end{bmatrix},$$

$$h_E(x_E) = -h_m(x_m) + C_t x_t - C_r x_r,$$

As the extended system has been constructed by linking Σ_m, Σ_t and Σ_r it is possible to define d_{E_u}, d_{E_z} and d_{E_y} as the relative degree of the extended system

Σ_E with respect to u, z and y, respectively. Then

$$d_{E_u} = d_m,\ d_{E_z} = d_t,\ \text{and}\ d_{E_y} = d_r.$$

The solution of the disturbance decoupling problem with measurements (DDPM) associated to Σ_E (e.g. decoupling the output $y_E = h_E(x_E)$ from the signals z and y) can be expressed in terms of the relative degrees of Σ_m, Σ_t and Σ_r. As a matter of fact its solution exists if and only if [96]

$$d_r \geq d_m \text{ and } d_t \geq d_m,$$

and its stability can be guaranteed if the following assumptions are satisfied [95]:

(A1) The input to the tracking model $z(t)$ and the output modelling error $\varepsilon(t)$ satisfy the inequalities

$$\begin{aligned} &| z(t) | \leq K_1, \text{ for all } t \geq 0, \\ &| \varepsilon(t) | \leq K_2, \text{ for all } t \geq 0, \end{aligned} \tag{7.49}$$

where K_1 and K_2 are positive constants.

(A2) Σ_m has a global inverse and is hyperbolically minimum phase.

In [97] it is also shown that the control law that solves the DDPM problem of (7.48), can be written as:

$$u = \alpha_E(x_E) + \beta_E(x_E)V_E + \gamma_{E_1}(x_E)z + \gamma_{E_2}(x_E)y \tag{7.50}$$

where

$$\alpha_E(x_E) = (-L_{g_m} L_{f_m}^{d_m-1} h_m(x_m))^{-1} \{ L_{f_m}^{d_m} h_m(x_m) - C_t A_t^{d_m} x_t$$

$$+ C_r A_r^{d_m} x_r - C_r A_r^{d_m-1} b_r h_m(x_m) \},$$

$$\beta_E(x_E) = (-L_{g_m} L_{f_m}^{d_m-1} h_m(x_m))^{-1}, \tag{7.51}$$

$$\gamma_{E_1}(x_E) = (-L_{g_m} L_{f_m}^{d_m-1} h_m(x_m))^{-1} (-C_t A_t^{d_m-1} b_t)$$

$$\gamma_{E_2}(x_E) = (-L_{g_m} L_{f_m}^{d_m-1} h_m(x_m))^{-1} (C_r A_r^{d_m-1} b_r).$$

The Lie derivative is defined in [97], where V_E is also given as

$$V_E(x_E) = \sum_{i=0}^{d_m-1} a_i (-L_{f_m}^i h_m(x_m) + C_t A_t^i x_t - C_r A_r^i x_r) \tag{7.52}$$

and the coefficients $a_0, \ldots, a_{d_m-1}$ form the Hurwitz polynomial

$$p(s) = s^{d_m} + a_{d_m-1} s^{d_m-1} + \cdots + a_1 s + a_0 \tag{7.53}$$

It is not difficult to show that, if the relative degrees of (Σ_m) and the plant (Σ) coincide, then the tracking dynamics are described by [97]:

$$\sum_{i=0}^{d_m} a_i e_t^{(i)} = \sum_{i=0}^{d_m} a_i (\varepsilon^{(i)} - y_r^{(i)}) \tag{7.54}$$

Which indicates that the tracking error asymptotically approaches the origin if the right hand side of (7.54) is zero, e. g.:

$$p(s) e_t = 0 \tag{7.55}$$

which corresponds to the case when the Σ_m and process are identical and with the same initial conditions.

In situations in which the right hand side of (7.54) does not vanish, it may still be possible to reduce the tracking error by selecting an appropriate regulation

filter. Namely, if Σ_r has unitary gain and a bandwidth higher than that of the tracking mismatch dynamics. If this is the case

$$y^{(i)} \longrightarrow (\varepsilon^{(i)} - y_r^{(i)}) + y_t^{(i)} \quad \text{as} \quad t \longrightarrow \infty \tag{7.56}$$

which is a solution of (7.54) when $t \longrightarrow \infty$, indicates that ϵ_t would converge to zero with a rate determined by the regulation filter dynamics.

In the case of the arm the terms ε^i are originated by the tracking error and the interaction between the links.

Note.

In the original design of the above control structure Σ_m was considered a plant model, nevertheless such approach is limited by assumption **A2**. An alternative option is to consider Σ_m a non-linear compensator. This aspect is currently under research [97].

Two individual controllers were designed and then applied in simulation to the underwater model arm presented above. The controller was defined by the following subsystems.

For Σ_{ti}:

$$\frac{x_{ti}(s)}{z_i(s)} = g_{ti}(s) \tag{7.57}$$

with

$$g_{ti}(s) = \frac{a_{t0i}}{s^2 + a_{t1i}s + a_{t0i}} \ ,$$

where $i = 1, 2$ corresponds to the link-i. The values of the parameter were selected as: $a_{t01} = 0.25$, $a_{t11} = 1$, $a_{t02} = 1$ and $a_{t12} = 2$. z_i represents the output reference.

For Σ_r

$$\frac{x_{ri}(s)}{\varepsilon_i(s)} = g_{ri}(s) \tag{7.58}$$

with

$$g_{ri}(s) = \frac{a_{r0i}}{s^2 + a_{r1i}s + a_{r0i}} \ ,$$

where $i = 1, 2$ corresponds to the link-i. The values of the parameter were selected as: $a_{r01} = 25$, $a_{r11} = 10$, $a_{r02} = 36$ and $a_{r12} = 12$. $\varepsilon_i = \theta_i - x_{mi}$ where x_{mi} is defined next.

For Σ_{mi}:

$$\ddot{x}_{m1} = -4\,\dot{x}_{m1} - \dot{x}_{m1}^3 + 0.02\,u_1 \tag{7.59}$$

and

$$\ddot{x}_{m2} = -\dot{x}_{m2} - \dot{x}_{m2}^3 + 0.02\,u_2 \tag{7.60}$$

A block diagram describing the structure of the individual controllers applied to each link is shown in figure 7.11. In the ideal representation of the manipulator the model is free from noise although it is possible to include it in the simulation. However, ε in this case includes implicitly the undesired interaction between the manipulator links. The interaction is therefore treated as disturbance and dealt with by each link controller through driving the extended system error to zero. This illustrates an efficient implementation of a controller, designed primarily for a SISO system [98, 99], on MIMO system (manipulator) as shown in next the section.

7.6 Simulation Results

The simulation program was written in MATLAB. Appendix D lists the file containing the underwater two link manipulator with the above described controller. For the purpose of simulation the dynamic parameters of the manipulator as well as the hydrodynamic conditions are kept the same as in section 7.4. Considering the driving force/torque limitation that possibly encounter real systems, a limiter was imposed on driving torque generated by the controllers.

During the simulation the maximum absolute value of the control input was bounded for each link as follows: $max(|u_1|) = 10$ Nm and $max(|u_2|) = 20$ Nm. The effect of these two limiters on the performance of the controller is also assessed

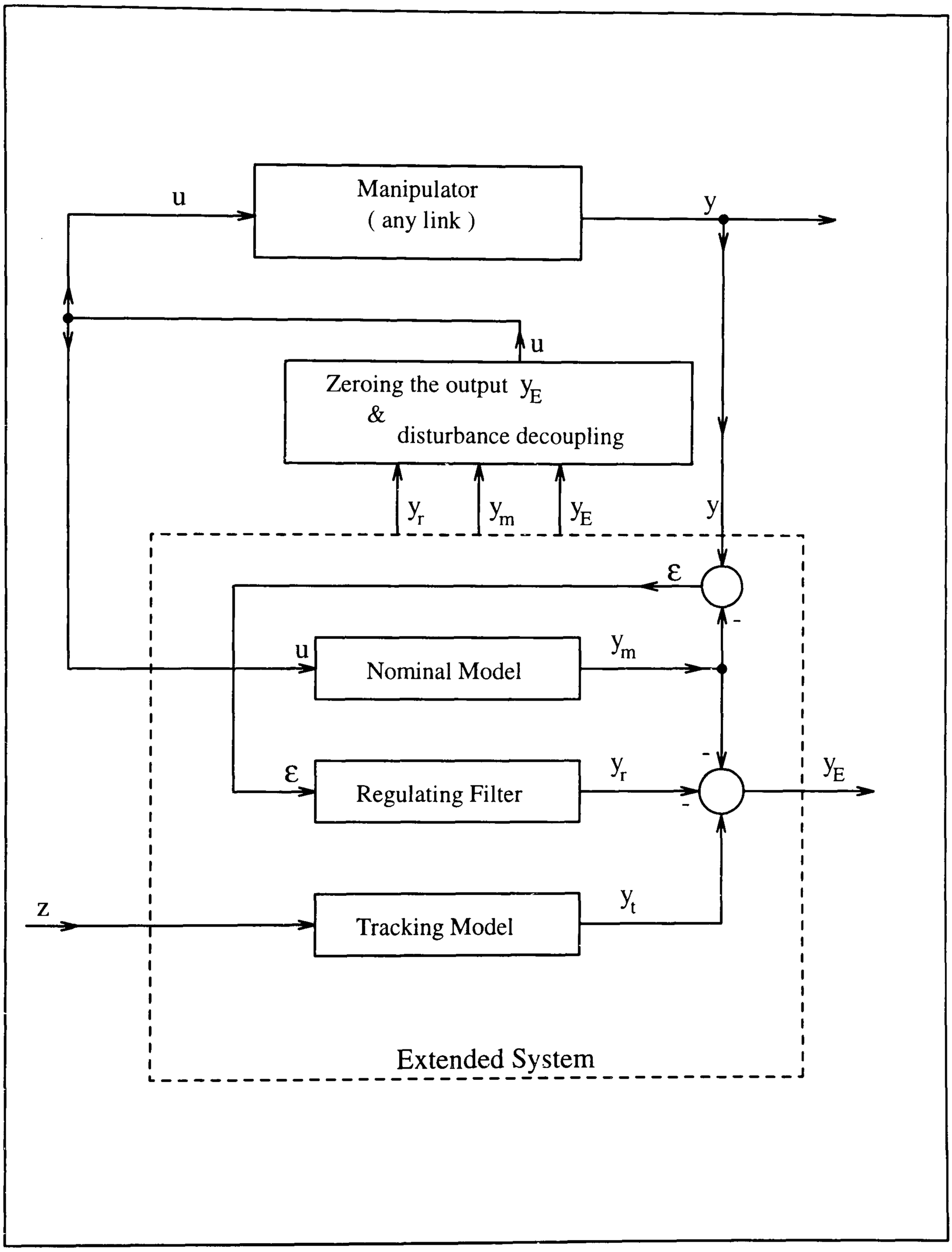


Figure 7.11: The structure of the control system applied to each link of the underwater manipulator.

through simulations. This is done by comparing the achieved manipulator joint angles under active control with and without limiter.

The simulation results presented next, show that the underwater arm can be controlled with relatively simple designs. Note that Σ_{mi} are far more simple than the actual model of the underwater robot arm. Figure 7.12 shows simultaneous link displacement movement, achieved by the controller in smooth trajectory. The interactions are kept very small and not visible on the graphs. The same example of the time response of the closed-loop system results in the tracking errors shown in figure 7.13. This demonstrates the possibility of achieving manipulator positions with negligible tracking errors and reduced link interactions.

Figures 7.14 to 7.17, on the other hand, illustrate the effect of the torque limiter, used here to mimic the limitation encountered in real systems where maximum torques cannot be exceeded even when higher values are required. Although the manoeuvres are the same as in the previous figures, the torques generated by the controllers pass through more rigorous limiters with ± 4Nm and ± 10Nm maximum for link 1 and 2 consecutively. The simulation of the above mentioned manoeuvre was executed with the limiters and a second time without the limiters at all, to allow results comparison. Figures 7.14 and 7.15 show the controllers generated torques for both cases. They focus only on the relevant part of the move, where the input torques to the links are different and they converge to the same values during the rest of the move.

The achieved link positions in both cases are very close and the difference between the angular responses cannot be noticed on the same graph. Figures 7.16 and 7.17 therefore show the difference between the positions achieved by the links. and again they focus on the first part where the difference is highest. During the rest of the move these differences converge to zero.

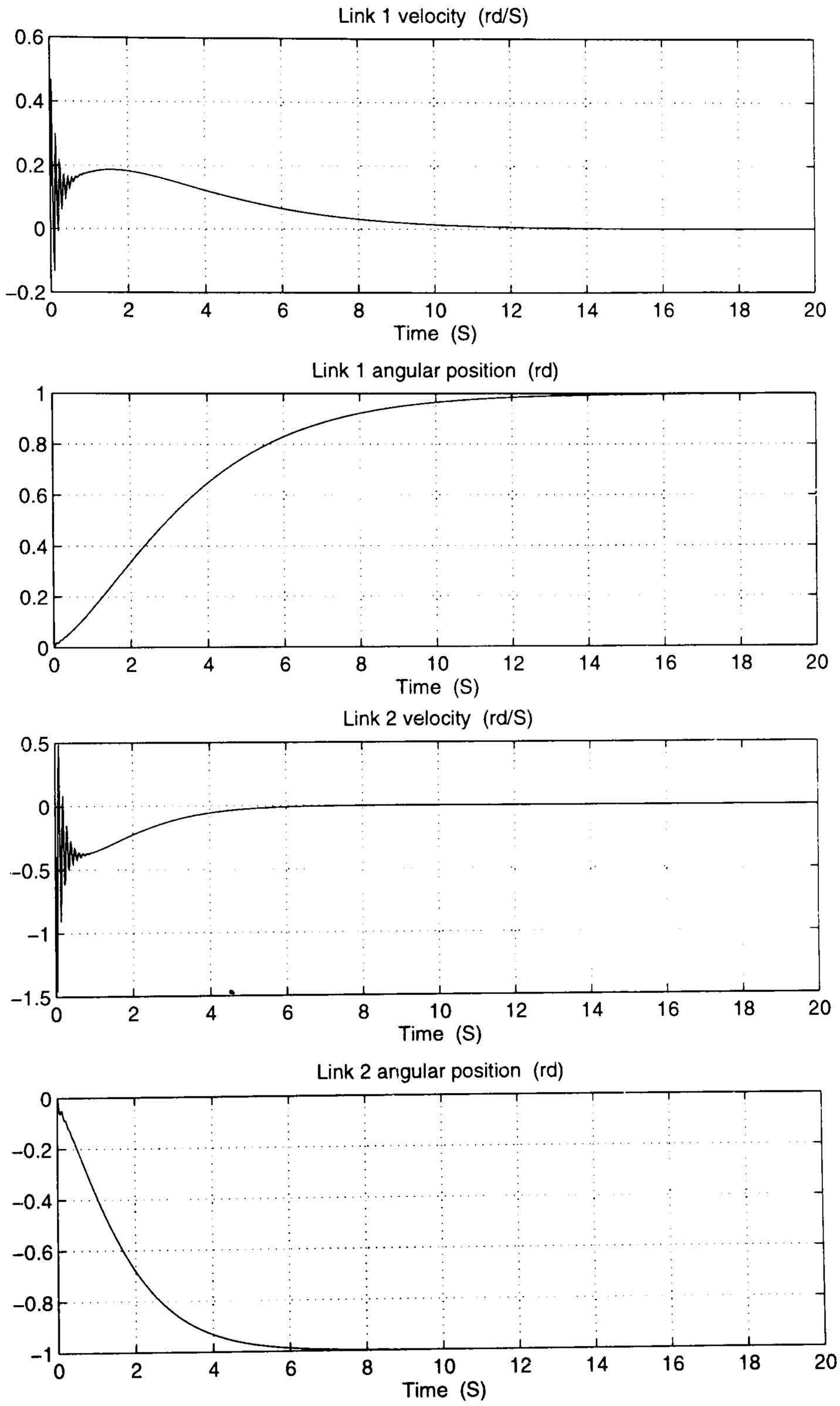


Figure 7.12: Response of the underwater arm under active control

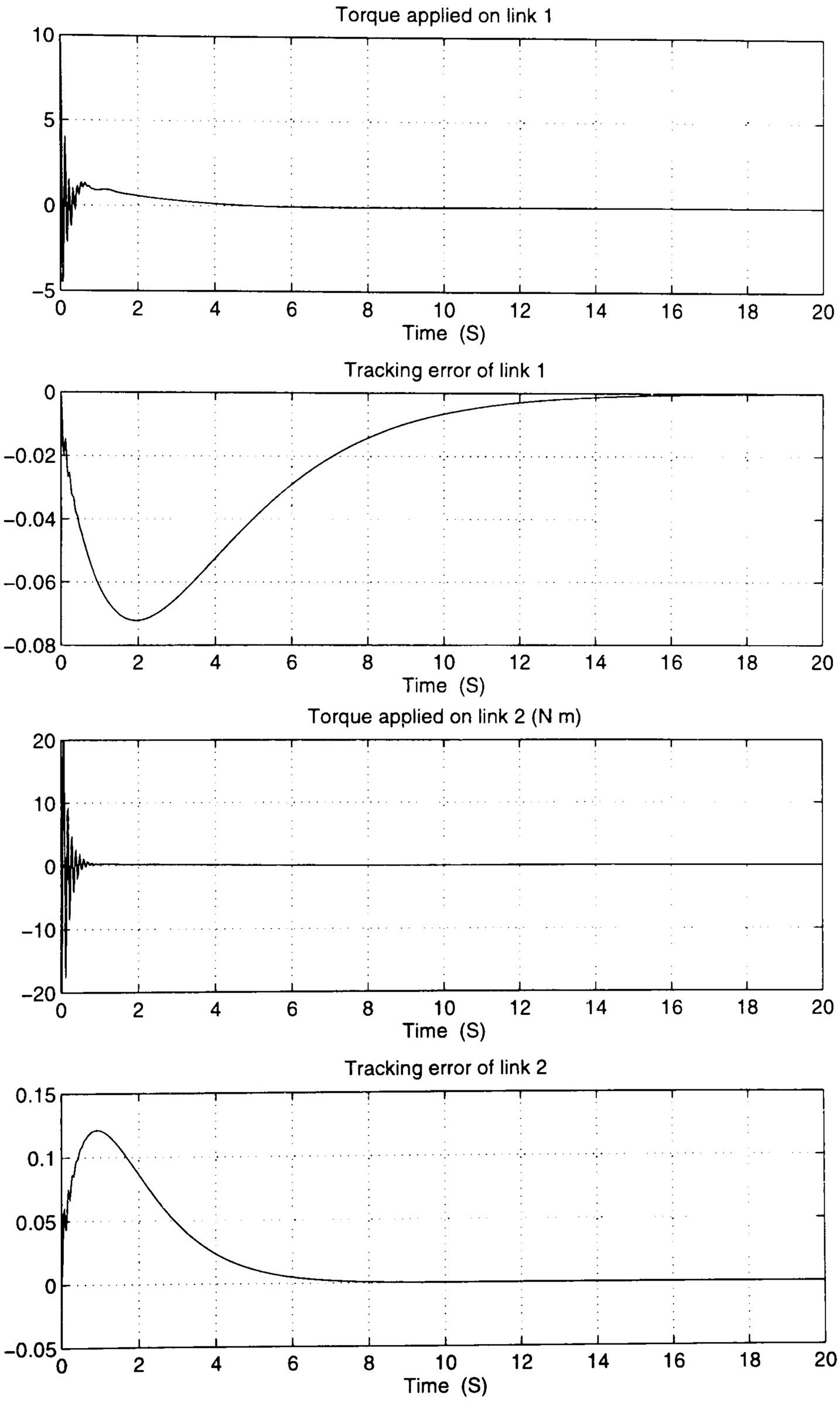


Figure 7.13: Torques and tracking errors.

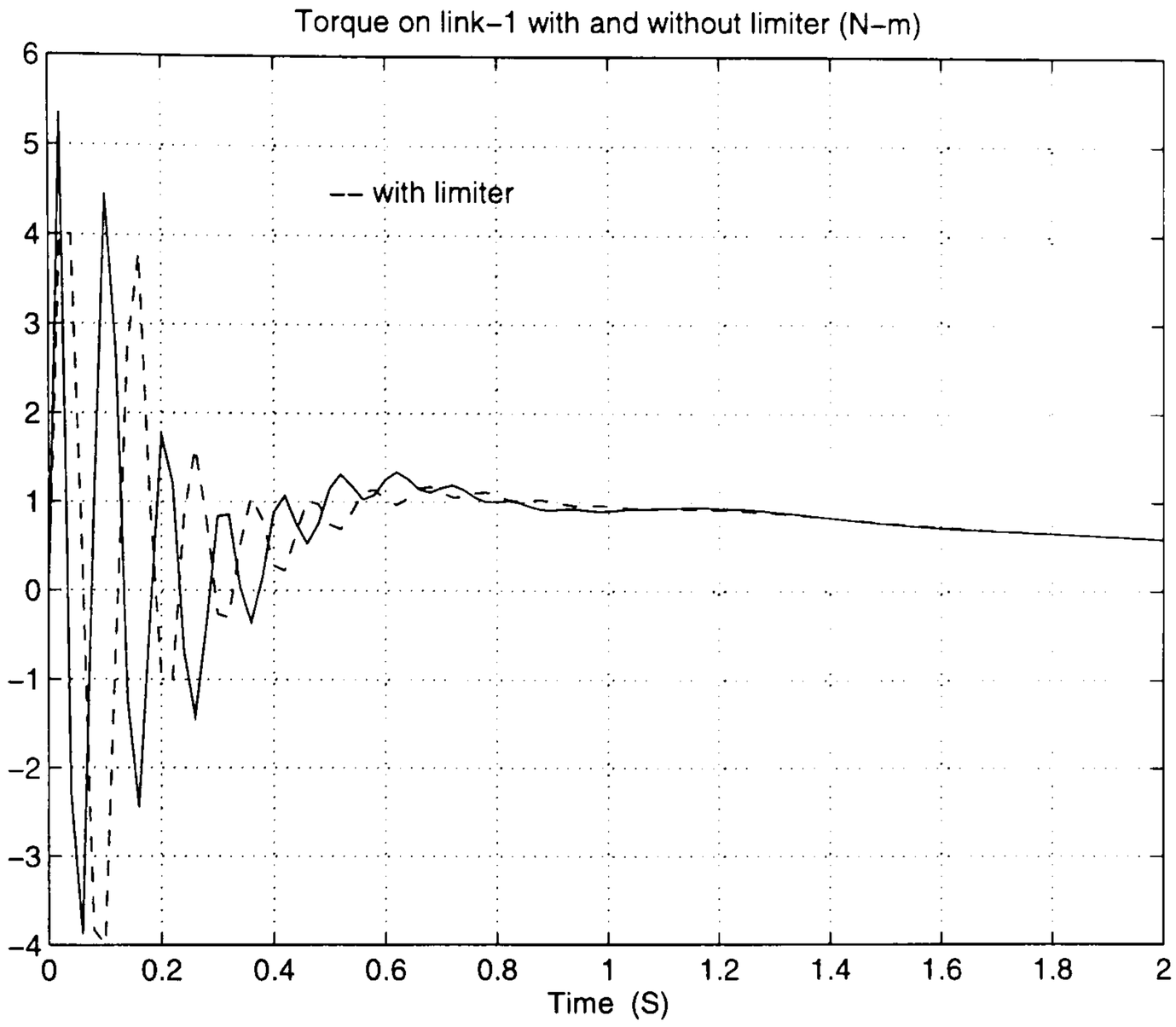


Figure 7.14: The torque applied on link 1, with and without limiter ($\pm$ $4Nm$)

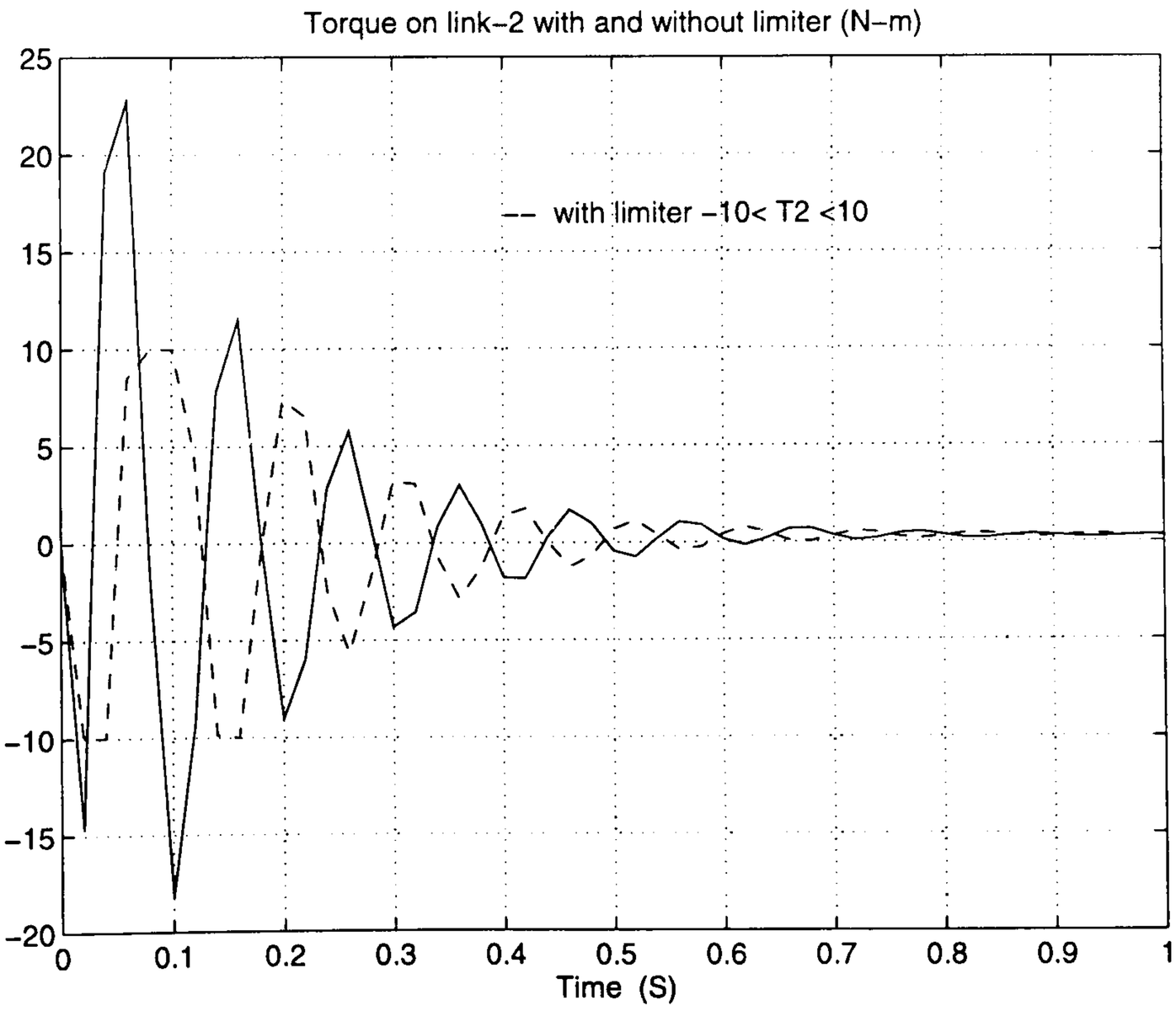


Figure 7.15: The torque applied on link 2, with and without limiter ($\pm$ $10Nm$)

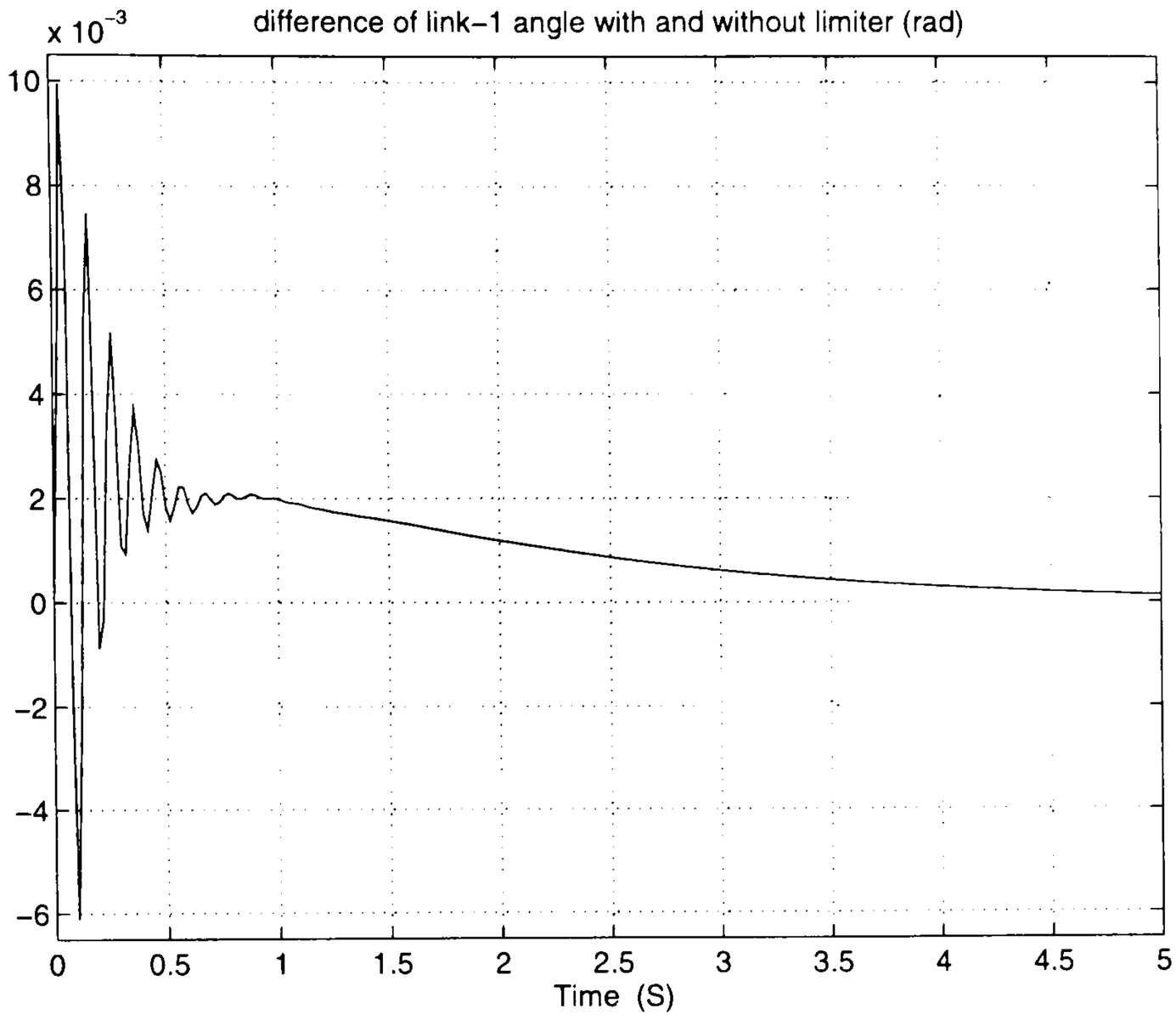


Figure 7.16: The difference of link 1 angular position with and without limiter.

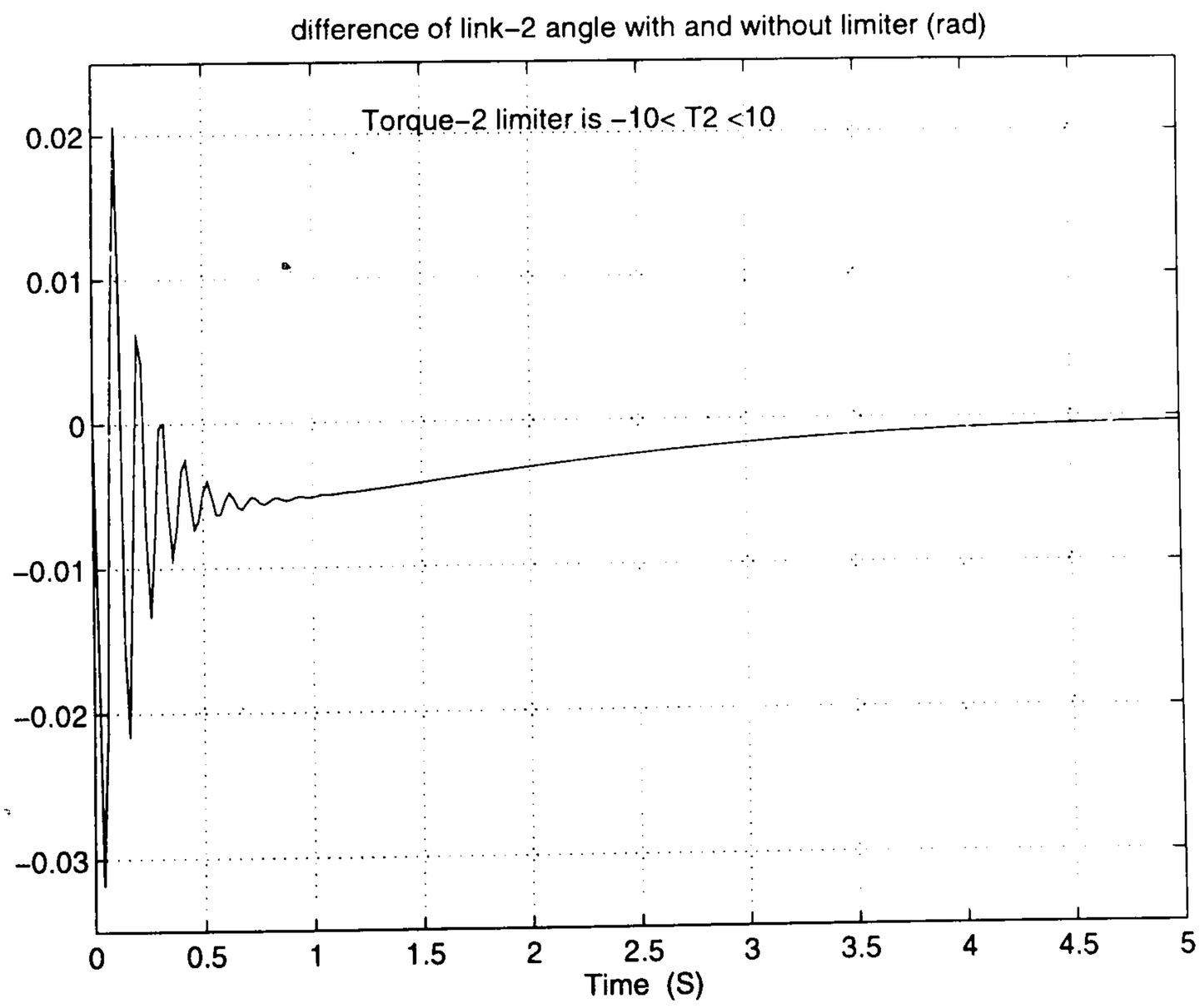


Figure 7.17: The difference of link 2 angular position with and without limiter.

7.7 Remarks and Conclusions

A model for a particular underwater robot arm was developed. The model was obtained by including the hydrodynamic effects due to drag and added mass into the original dry arm model. These effects were calculated explicitly extending previous works which were limited to implicit generic models.

The model derived here is rather complex and cannot in general be written in closed form. This is due to the fact that in order to calculate the drag effect. an integral, with several possible solutions depending on the state of the arm, is involved.

It is also remarked that in spite of several simplifying assumptions (constant added mass and drag coefficients) the model is still not simple. Moreover. some effects were not considered, such as lift, buoyancy and the added mass related to the velocity parallel to the second link.

This work shows that implementing an automatic model generation of underwater manipulators may be of high complexity, in comparison to general multi-body dynamics.

Considering the mathematical complexity of hydrodynamic effects it is thought that the interaction of the underwater manipulator and the supporting platform such as ROV is more complicated than suggested in [100]. A study of this aspect is planned on the basis of the derived model.

For arms operating in a three dimensional space it is necessary to consider the Karman vortex effects [94].

In the example treated here, in spite of the complexity of the arm dynamics. the control problem is not difficult to solve. The extra damping caused by the drag allows effective implementation of the proposed simple control laws.

A future extension of this work is to investigate the feasibility of adding a module to DADS and AUTOLEV to support such a manipulator. Although preliminary results suggest that a fully automatic model generation may not be

achievable.

Chapter 8

Conclusion

It has been shown during this research that mathematical modelling is essential for the design development and analysis of robotic manipulator systems. For manipulators, modelling is not limited to the dynamics, but should be extended to the kinematics as well, to cover the overall positioning performance. In both aspects, the modelling does not stop at producing the model, but rather a validation of the model should be considered an integral part of the modelling process. It has been shown on an existing manipulator that the controller achieved the required joint rates for a move. The developed dynamic model is valid for positioning the joints, but the achieved end-effector position of the manipulator differs from the one predicted by the kinematic model. A more accurate kinematic model has been established to replace the one contained in the manipulator controller. This has shown that, while many simplifications are permissible in a dynamic model without affecting its validity for its intended use. accuracy of kinematic parameters is crucial to the end-effector positioning. With a poor end-effector positioning, a manipulator still can be used effectively in a position teaching mode, however the off-line programming facility available on the manipulator cannot be used.

A general description of the modelling process has been presented, and

this has led to the introduction of the three most important formulations for producing manipulator dynamic equations. A comparison of their performance has been discussed. It has been shown they could be easily coded in a symbolic manipulation program, such as REDUCE, to produce the correct dynamic models for rigid manipulators. With the advances in computing, the growing need for automatic modelling has resulted in many modelling computer packages in the market. Three samples of these were described and compared with regard to their use for modelling manipulators. It has been concluded that they are good at producing models and simulation codes, but provide little support for other activities such as control design and dynamic understanding in some cases. The proposed interfaces put an end to these difficulties for the software AUTOLEV and made it a very powerful modelling tool for manipulators. The final combined package was used in producing efficient simulations throughout the rest of this work. With such powerful modelling programs the talent of the engineer can be focused on, and invested in other activities, such as model validation and design of a suitable controller for the manipulator. The use of more computing resources for the design, modelling and analysis in robotic manipulator is illustrated through an example, where the inertial parameters of the Lambertons' AA300 manipulator were produced by the geometric modelling package I-DEAS and fed to DADS to perform the simulations. This methodology proves to be highly effective in producing accurate information which is difficult to obtain through direct measurements.

A low cost measurement system was developed to measure the currents and positions from the three first link motors of a Puma 560 manipulator simultaneously. The system successfully measured the required information and proved to be flexible enough to connect with other instruments, such as the Optotrac, to measure in combination more information. The developed system can be used with little modification for measuring similar information from other

manipulators.

It has been emphasised that a validation should be considered an integral part of model development. It has been shown that although a dynamic model of a manipulator could contain several simplifying assumptions and slightly inaccurate parameter values, it is still considered a **valid** description for a given intended use of the model. The validation process is done through model parameter estimation. A manipulator model can be split into several joint models and validation performed on each. The joint model parameter estimation has been proven to provide a good estimate of model parameters. A simple least squares technique or a singular value decomposition method is used for the estimation process where gravity loading is also shown to be estimated in the case of joints rotating about horizontal axes. This method is predicted to produce more satisfactory results for SCARA and Cartesian type manipulators, according to the proposed methodology in the thesis. The method requires simple hardware to provide the measurement for joint model estimation and then the information is used to build the valid overall model. Friction parameter tuning has been shown to produce acceptable parameter values for a valid model of the Puma 560, compared with measurement taken from a real manipulator.

Working on the dynamic model has revealed the necessity for kinematic model improvement, as it is responsible for the poor manipulator end-effector positioning. A new methodology based on Stone's method was developed and yields satisfactory improvement. It has been tested on the Puma 560 model, both experimentally and through simulation, and produced more than 80% positioning improvement. Establishing a more accurate kinematic model improves manipulator end-effector positioning accuracy and allows effective use of the off-line programming facility of the manipulator.

It has been noticed that none of the computer modelling packages is designed to support manipulators operating under water. A model was developed by

including in the dry arm model the hydrodynamic effects due to drag and added mass. Although some simplifying assumptions were imposed, the obtained model is rather complicated and therefore computer automatic modelling is thought to be a difficult task. The hydrodynamic effects were calculated explicitly extending previous works which were limited to implicit generic models. A control scheme based on noise rejection is proposed as a solution to controlling the underwater arm and produced acceptable positioning performance. The model is therefore used to predict the dynamic behaviour of underwater arms whereas the kinematics are no different from those of manipulators operating in normal conditions.

Although this thesis has covered several aspects of manipulator modelling, the work focused on rigid manipulators. The need for developing lighter, faster robotic manipulators with high accuracy is on the increase. However, these would possess considerable joint and link flexibility. The control of such manipulators is complex due the higher number of degrees of freedom caused by the structural flexibility. Despite the numerous publications on the modelling and control of flexible manipulators, these are still of simple shapes and limited number of links. In addition, they are still confined to research laboratories. The answer to the demand of such manipulators is thought to lie in developing new materials which are low in weight but stiff enough to make links that behave as rigid manipulator links. Joint flexibility, on the other hand, is relatively easy to model and can be exploited for use in force control when it is accurately known.

8.1 Future Work

The major avenues in which the author wishes to carry out further research as a follow up to this thesis can be summarised in the following four points:

- Joint stiffness modelling has already been considered in a DADS generated model, which is relatively straightforward. It can be added to the manipulator model through introducing *spring-damper* elements to the joints. Including the stiffness in a symbolic model should not be difficult. One degree of freedom is added to the manipulator at each flexible joint. The position of the actuator and the position of the link are connected through the flexible joint. The author intends to explore the use of joint flexibility for manipulator force control.

- The applicability of the model distortion method introduced by Butterfield [67] to manipulator model validation should be explored. It provides parameter estimation through parameter tuning which is done as a function of time. It has been indicated that the method has been used successfully in the nuclear field and could be applied to manipulator models.

- The measured data from the Puma 560 using the developed measurement system combined with the Optotrac are to be used within the methodology described by Driel in [27] to explore the level of kinematic model improvement and thus the end-effector positioning improvement. The combination is predicted to produce accuracy of the order of the repeatability [27].

- Further investigation is needed in the line of underwater manipulators, to verify that the model developed in this thesis does reflect the dynamics of a manipulator operating underwater, using an experimental test rig. The pursuit of this line of work depends, however, on the availability of resources. The feasibility of automatic underwater manipulator modelling also needs to be explored. The dynamic interaction between an underwater manipulator and the supporting platform such as ROV should also be studied.

Appendix A

Commercial Modelling Programs Related Files

A.1 The Reduce Code for generating manipulator's equations

This file requires the number of links to be entered as well as a file containing the names or the values of some parameters such as the force acting at the end of last link and some symbolic triangular relationships for the Reduce program. An example input file is given in the next section.

```
% Enter N := the number of links of the manipulator

pause;

MATRIX MA(N,1),IL(3*N,3),THE(N,1),D1(N,1),D2(N,1),
       ALPHA(N,1),DX(N,1),DZ(N,1),POS(N,3),ENDN(3,1),
       ENDF(3,1),W0(3,1),E0(3,1),A0(3,1)$
%

PAUSE;
      MATRIX E1(3,3),E2(3,3),E3(3,3),H1(1,3),H2(1,3),H3(1,3),
           U1(1,1),U2(1,1),U3(1,1),Z(3,1),M(N,N),Q(N,1),Z0(1,3),
           RA(3,3),RRA(3,3),WA(3,1),WWA(3,3),EA(3,1),EEA(3,3),
          AA(3,1),ACA(3,1),FA(3,1),RPA(3,3),RP1A(3,3),VA(3,1),
           TTA(1,1),IA(3,3),P1A(3,1),PA(3,1),DA(3,1),R1A(3,3),
           C(N,N),GA(N,N),QA(N,1);
ARRAY R(N+1),RR(N+1),W(N),WW(N),E(N),EE(N),A(N),
      AC(N),F(N+1),RP(N+1),RP1(N+1),V(N+1),TT(N),I(N),
      P1(N),P(N),D(N),R1(N+1,N)$
W(0):=W0$
E(0):=E0$
```

```
A(0):=A0$
E1:=MAT((0,0,0),(0,0,-1),(0,1,0))$
E2:=MAT((0,0,1),(0,0,0),(-1,0,0))$
E3:=MAT((0,-1,0),(1,0,0),(0,0,0))$
H1:=MAT((1,0,0))$H2:=MAT((0,1,0))$H3:=MAT((0,0,1))$
Z:=MAT((0),(0),(1))$Z0:=MAT((0,0,1))$
F(N+1):=ENDF$V(N+1):=ENDN$
WW(0):=MAT((0,0,0),(0,0,0),(0,0,0))$
      RR(N+1):=MAT((1,0,0),(0,1,0),(0,0,1))$
      R(N+1):=RR(N+1)$
% calculate the Transformation Matrices,

FOR J:=1 STEP 1 UNTIL N DO
      <<RR(J):=MAT((COS(THE(J,1)),-SIN(THE(J,1))*COS(ALPHA(J,1)),
          SIN(THE(J,1))*SIN(ALPHA(J,1))),(SIN(THE(J,1)),
          COS(THE(J,1))*COS(ALPHA(J,1)),-COS(THE(J,1))
          *SIN(ALPHA(J,1))),(0,SIN(ALPHA(J,1)),COS(ALPHA(J,1))))$
        R(J):=MAT((COS(THE(J,1)),SIN(THE(J,1)),0),(-SIN(THE(J,1))*
         COS(ALPHA(J,1)),COS(THE(J,1))*COS(ALPHA(J,1)),SIN(ALPHA(J,1))),
         (SIN(THE(J,1))*SIN(ALPHA(J,1)),-COS(THE(J,1))*SIN(ALPHA(J,1)),
         COS(ALPHA(J,1))))>>$
%

    FOR J:=1 STEP 1 UNTIL N DO
     <<WA:=W(J-1);RA:=R(J);WA:=RA*(WA+Z*D1(J,1));W(J):=WA;
       U1:=H1*WA$O1:=U1(1,1)$
       U2:=H2*WA$O2:=U2(1,1)$
       U3:=H3*WA$O3:=U3(1,1)$
       WWA:=O1*E1+O2*E2+O3*E3;WW(J):=WWA>>;PAUSE;
      FOR J:=1 STEP 1 UNTIL N DO
       <<EA:=E(J-1);RA:=R(J);WWA:=WW(J-1);
       EA:=RA*(EA+Z*D2(J,1)+WWA*(Z*D1(J,1)));E(J):=EA;
       U1:=H1*EA$O1:=U1(1,1)$
       U2:=H2*EA$O2:=U2(1,1)$
      U3:=H3*EA$O3:=U3(1,1)$
       EEA:=O1*E1+O2*E2+O3*E3;EE(J):=EEA>>;PAUSE;
    FOR J:=1 STEP 1 UNTIL N DO
       <<RA:=R(J);
       PA:=RA*MAT((COS(THE(J,1))*DX(J,1)),
              (SIN(THE(J,1))*DX(J,1)),(DZ(J,1)));P(J):=PA;
       D(J):=MAT((POS(J,1)),(POS(J,2)),(POS(J,3))),
       EEA:=EE(J);DA:=D(J);PA:=P(J);WWA:=WW(J);RA:=R(J);AA:=A(J-1);
       AA:=EEA*PA+WWA*WWA*PA+RA*AA;A(J):=AA;
       ACA:=EEA*DA+WWA*WWA*DA+AA;AC(J):=ACA>>;
     FOR J:=0 STEP 1 UNTIL N DO
```

```
      <<R1(N+1,J):=R1(N,J)>>$
PROCEDURE FFF(N1,N2)$
     FOR J:=N1 STEP -1 UNTIL N2 DO
      <<PA:=P(J);DA:=D(J);RA:=R(J+1);FA:=F(J+1);ACA:=AC(J);
      RRA:=RR(J+1);FA:=RRA*FA+MA(J,1)*ACA;F(J):=FA;
      U1:=H1*(PA+DA)$O1:=U1(1,1)$
      U2:=H2*(PA+DA)$O2:=U2(1,1)$
      U3:=H3*(PA+DA)$O3:=U3(1,1)$
      RPA:=O1*E1+O2*E2+O3*E3;
      U1:=H1*(RA*PA)$O1:=U1(1,1)$
      U2:=H2*(RA*PA)$O2:=U2(1,1)$
      U3:=H3*(RA*PA)$O3:=U3(1,1)$
      RP1A:=O1*E1+O2*E2+O3*E3$
      IA:=MAT((IL(J*3-2,1),IL(J*3-2,2),IL(J*3-2,3)),
              (IL(J*3-1,1),IL(J*3-1,2),IL(J*3-1,3)),
              (IL(J*3,1),IL(J*3,2),IL(J*3,3))),
      RRA:=RR(J+1);EA:=E(J);WWA:=WW(J);WA:=W(J);ACA:=AC(J);
      VA:=V(J+1);FA:=F(J+1);
      VA:=RRA*(VA+RP1A*FA)+RPA*MA(J,1)*ACA+IA*EA+WWA*IA*WA;V(J):=VA;
      RA:=R(J);TTA:=TP(VA)*(RA*Z);
      O4:=TTA(1,1);
      FOR K:=1 STEP 1 UNTIL N DO
        <<M(J,K):=DF(O4,D2(K,1))>>$
      Q(J,1):=O4-FOR K:=1 STEP 1 UNTIL N SUM M(J,K)*D2(K,1)$>>$
 ;END;
```

A.2 An example of input file

```
MA:=MAT((M1),(M2),(m3));
ENDF:=MAT((0),(0),(0));
ENDN:=MAT((0),(0),(0));
AO:=MAT((0),(g),(0));
WO:=MAT((0),(0),(0));
EO:=MAT((0),(0),(0));
TH:=MAT((TH1),(TH2),(th3));
D1:=MAT((TH1D),(TH2D),(th3d));
D2:=MAT((TH1DD),(TH2DD),(th3dd));
DX:=MAT((0),(L1),(L2));
DZ:=MAT((0),(0),(0));
POS:=MAT((L1,0,0),(l2,0,0),(l3,0,0));
for all x,y let cos(x)*cos(Y)=(cos(x+y)+cos(x-y))/2,
```

```
 cos(x)*sin(y)=(sin(x+y)-sin(x-y))/2,
 sin(x)*sin(y)=(cos(x-y)-cos(x+y))/2,
 sin(x)**2+cos(x)**2=1;

;end;
```

A.3 SD/FAST Cart with Double Pendulum Input File

```
#This is an SD/FAST input file
# for the 2 rods problem
grounded
gravity = 0.0   9.810   0.0

 body=cart        joint =slider
        mass= 1
        inertia = 0.0      0.0       0.0
        bodytojoint =0.0       0.0       0.0
        pin=1   0    0

body =rod1      inb =cart        joint =pin
        mass = 0.5
        inertia =0.0       0.0       0.0026
        bodytojoint= 0.0       -0.125       0.0
        inbtojoint = 0.0        0.0         0.0
        pin = 0       0       -1

body= rod2      inb =rod1      joint =pin
        mass =0.5
        inertia =0.0       0.0       0.0026
        bodytojoint =0.0      -0.125       0.0
        inbtojoint = 0.0       0.125       0.0
        pin = 0       0      -1
```

A.4 AUTOLEV Input File

```
DOF(3)
AUTOZ(OFF)
FRAMES(A,B,C)
INERTIA(A,0,0,0,0,0,0)
INERTIA(B,IB,IB,IB,0,0,0)
```

```
INERTIA(C,IC,IC,IC,0,0,0)
CONST(L1,L2,G,DBUG)
VAR(X,TH1,TH2)
DIRCOS(N,A,1,0,0,0,1,0,0,0,1)
POINTS(O,P)
MASSLESS(O,P)
SIMPROT(A,B,3,TH1)
SIMPROT(B,C,3,TH2)
DIRCOS(N,B)
DIRCOS(N,C)
DIRCOS(A,C)
X'=U1
TH1'=U2
TH2'=U3
WAN=0
WBN=U2*B3
WCN=(U2+U3)*C3
ALFAN=DERIV(WAN,T,N)
ALFBN=DERIV(WBN,T,N)
ALFCN=DERIV(WCN,T,N)
VASTARN=U1*A1
PASTARBSTAR=(L1/2)*B1
V2PTS(N,B,ASTAR,BSTAR)
PBSTARP=(L1/2)*B1
V2PTS(N,B,BSTAR,P)
PPCSTAR=(L2/2)*C1
V2PTS(N,C,P,CSTAR)
AASTARN=DERIV(VASTARN,T,N)
ABSTARN=DERIV(VBSTARN,T,N)
ACSTARN=DERIV(VCSTARN,T,N)
FRSTAR
FORCE(ASTAR)=-MASSA*G*N2+FA1*N1
FORCE(BSTAR)=-MASSB*G*N2
FORCE(CSTAR)=-MASSC*G*N2
TORQUE(B)=TB1*B1+TB2*B2+TB3*B3
TORQUE(C)=TC1*C1+TC2*C2+TC3*C3
CONTROLS(FA1,TB1,TB2,TB3,TC1,TC2,TC3)
FA1=0.0
FR
KANE
```

A.5 Autolev Generated Code

```
C       THE NAME OF THIS PROGRAM IS TROLL2.FOR
```

```
C      CREATED BY AUTOLEV ON 01-01-1980 AT 00:16:36
C
       IMPLICIT DOUBLE PRECISION (A-Z)
       INTEGER JLOOP,NSTEPS,NCUTS,NEQS,ILOOP
       LOGICAL STPSZ
       EXTERNAL EQNS
       CHARACTER MSG(75)
       DIMENSION U(6)
       COMMON/CPAR/IB,IC,MASSA,MASSB,MASSC,PI,L1,L2,G,DBUG
       COMMON/CONT/FA1,TB1,TB2,TB3,TC1,TC2,TC3
       COMMON/DFQLST/T,STEP,RELERR,ABSERR,NCUTS,NEQS,STPSZ
C
       OPEN(UNIT=11,FILE='TROLL2.IN ',STATUS='old')
       OPEN(UNIT=12,FILE='TROLL2.OU1',STATUS='new')
       OPEN(UNIT=13,FILE='TROLL2.OU2',STATUS='new')
       PI = 4.0*DATAN(1.0D0)
       WRITE(*,601)
C
*****************************************************************************
*                                                                           *
*                                  NOTE                                     *
*                                                                           *
*  The user must supply an input data file to this program. The             *
*  file must be named FILENAME.IN , where FILENAME is obtained from         *
*  the first line of this program. The data must be arranged in an          *
*  order consistent with the READ statements that follow this NOTE.        *
*                                                                           *
*  The output from the program is sent to FILENAME.OUi (i = 1, 2,           *
*  ...). The first column in each such file is the time T in                *
*  seconds, running from zero to TMAX in increments of STEP; the            *
*  values of TMAX and STEP are input by the user from the terminal at       *
*  run time. There are at most five more columns in each output             *
*  file. These contain the time histories of all of the generalized         *
*  speeds and all kinematical quantities, except Euler parameters,          *
*  that result from solution of kinematical differential equations.         *
*  The column headings all contain the word UNITS. The user can             *
*  modify the associated FORMAT statements in the FORTRAN program to        *
*  reflect the units used in the particular problem being solved. If        *
*  the AUTOLEV ANGMOM and/or ENERGY options have been used, time            *
*  histories of these quantities are on files FILENAME.H and/or             *
*  FILENAME.NRG.                                                            *
*                                                                           *
*****************************************************************************
C
      READ(11,*) L1,L2,G,DBUG
```

```
      READ(11,*) MASSA,MASSB,MASSC
      READ(11,*) IB
      READ(11,*) IC
      READ(11,*) (U(ILOOP),ILOOP = 1,3)
      READ(11,*) X,TH1,TH2
C
      WRITE(*,602)
      READ(*,*) TMAX,STEP
      WRITE(*,603)
      READ(*,604) (MSG(ILOOP),ILOOP = 1,75)
C
      WRITE(* ,605)
      WRITE(* ,606) (MSG(ILOOP),ILOOP = 1,75)
      WRITE(12,605)
      WRITE(12,606) (MSG(ILOOP),ILOOP = 1,75)
      WRITE(13,605)
      WRITE(13,606) (MSG(ILOOP),ILOOP = 1,75)
      WRITE(* ,607)
      WRITE(12,607)
C
      WRITE(* ,6070) L1,L2,G,DBUG
      WRITE(12,6070) L1,L2,G,DBUG
      WRITE(* ,6082) IB
      WRITE(12,6082) IB
      WRITE(* ,6083) IC
      WRITE(12,6083) IC
      WRITE(* ,610) MASSA,MASSB,MASSC
      WRITE(12,610) MASSA,MASSB,MASSC
      WRITE(* ,611)  (U(ILOOP),ILOOP = 1,3)
      WRITE(12,611)  (U(ILOOP),ILOOP = 1,3)
      WRITE(* ,6111) X,TH1,TH2
      WRITE(12,6111) X,TH1,TH2
      WRITE(* ,612) TMAX,STEP
      WRITE(12,612) TMAX,STEP
      U(4) = X
      U(5) = TH1
      U(6) = TH2
C
      NEQS = 6
      WRITE(* ,6171)
      WRITE(12,6171)
      WRITE(13,6172)
C
      NCUTS  = 20
      T      = 0.0
```

```
      RELERR = 1.0D-8
      ABSERR = 1.0D-8
      STPSZ  = .FALSE.
      NSTEPS = IDINT(TMAX/STEP+0.1)+1
C
      DO 1000 JLOOP = 1 , NSTEPS
      WRITE(12,6181)  T,(U(ILOOP),ILOOP = 1,5)
      WRITE(* ,6181)  T,(U(ILOOP),ILOOP = 1,5)
      WRITE(13,6181)  T,(U(ILOOP),ILOOP = 6,6)
      IF (JLOOP.EQ.NSTEPS) GO TO 1000
      CALL DEQS(EQNS,U,*99)
 1000 CONTINUE
C
      WRITE(*,620)
C
      STOP
   99 WRITE(*,616)
C
  601 FORMAT(1X,'SYSTEM PARAMETERS AND INITIAL CONDITIONS'/
     &       2X,'ARE NOW BEING READ FROM THE INPUT FILE'/)
  602 FORMAT(1X,'INPUT TMAX,STEP (S)  ')
  603 FORMAT(1X,'INPUT A DESCRIPTION OF THIS RUN'/)
  604 FORMAT(75A1)
  606 FORMAT(1X,'*** ',75A1)
  607 FORMAT(///1X,'SYSTEM PARAMETERS'//)
  612 FORMAT(//11X,'TMAX = ',1PD12.5,' S'/11X,'STEP = ',1PD12.5,' S',
     &       /'1')
  613 FORMAT(//1X,'SIMULATION RESULTS'//
     &        7X,'T',8X,'HN',11X,'HN1',10X,'HN2',10X,'HN3'
     &       /6X,'(S)',5X,'(N M S)',3(6X,'(N M S)')/)
 6151 FORMAT(//1X,'SIMULATION RESULTS'//
     &        7X,'T',6X,'ENERGY'/6X,'(S)',6X,'(N M)'/)
  616 FORMAT(1X,'STEPSIZE HALVED TOO MANY TIMES'/)
 6181 FORMAT(1X,F9.3,5(1X,1PD12.5))
  605 FORMAT(1X,'OUTPUT FROM PROGRAM TROLL2.FOR'//)
 6070 FORMAT(/13X,'L1 = ',1PD12.5,' UNITS'/13X,'L2 = ',1PD12.5,' UNITS'/
     &14X,'G = ',1PD12.5,' UNITS'/11X,'DBUG = ',1PD12.5,' UNITS'//)
 6082 FORMAT(13X,'IB = ',1PD12.5,' UNITS'/)
 6083 FORMAT(13X,'IC = ',1PD12.5,' UNITS'/)
  610 FORMAT(/10X,'MASSA = ',1PD12.5,' UNITS'/10X,'MASSB = ',1PD12.5,' U
     &NITS'/10X,'MASSC = ',1PD12.5,' UNITS'//)
  611 FORMAT(//1X,'INITIAL CONDITIONS'//10X,'U1(0) = ',1PD12.5,' UNITS'/
     &10X,'U2(0) = ',1PD12.5,' UNITS'/10X,'U3(0) = ',1PD12.5,' UNITS'//)
 6111 FORMAT(/11X,'X(0) = ',1PD12.5,' UNITS'/9X,'TH1(0) = ',1PD12.5,' UN
     &ITS'/9X,'TH2(0) = ',1PD12.5,' UNITS'//)
```

```
 6171 FORMAT(//1X,'SIMULATION RESULTS'//7X,'T',8X,'U1',11X,'U2',11X,'U3'
     &,12X,'X',11X,'TH1'/6X,'(S)',5X,'(UNITS)',6X,'(UNITS)',6X,'(UNITS)'
     &,6X,'(UNITS)',6X,'(UNITS)',6X/)
 6172 FORMAT(//1X,'SIMULATION RESULTS'//7X,'T',8X,'TH2'/6X,'(S)',5X,'(UN
     &ITS)',6X/)
  620 FORMAT(//1X,'OUTPUT IS ON FILES:  ','TROLL2.OU1'/22X,'TROLL2.OU2'/
     &22X,/)
      END
C
C
      SUBROUTINE EQNS(T,U,UDOT)
      IMPLICIT DOUBLE PRECISION (A-Z)
      INTEGER IPS(3)
      DIMENSION U(6),UDOT(6),COEF(3,3),RHS(3)
      COMMON/CPAR/IB,IC,MASSA,MASSB,MASSC,PI,L1,L2,G,DBUG
      COMMON/CONT/FA1,TB1,TB2,TB3,TC1,TC2,TC3
C
C
C
      X = U(4)
      TH1 = U(5)
      TH2 = U(6)
C
       S1 = DSIN(TH1)
       C1 = DCOS(TH1)
       S2 = DSIN(TH2)
       C2 = DCOS(TH2)
C
       COEF(1,1) = -MASSA-MASSB-MASSC
       COEF(1,2) = (-.5*(-C1*S2-S1*C2)*L2+L1*S1)*MASSC+.5*L1*MASSB*S1
       COEF(1,3) = -.5*(-C1*S2-S1*C2)*L2*MASSC
       COEF(2,1) = COEF(1,2)
       COEF(2,2) = (-C2*L1*L2-L1*L1-.25*L2*L2)*MASSC-IB-IC-.25*L1*L1*MASS
     &  B
       COEF(2,3) = (-.5*C2*L1*L2-.25*L2*L2)*MASSC-IC
       COEF(3,1) = COEF(1,3)
       COEF(3,2) = COEF(2,3)
       COEF(3,3) = -IC-.25*L2*L2*MASSC
C
       CALL CNTRL(T,U)
C
       RHS(1) = (-.5*(C1*C2-S1*S2)*(U(2)+U(3))*(U(2)+U(3))*L2-C1*L1*U(2)*
     &  U(2))*MASSC-.5*C1*L1*MASSB*U(2)*U(2)-FA1
       RHS(2) = (-.5*(U(2)+U(3))*(U(2)+U(3))*L1*L2*S2+.5*L1*L2*S2*U(2)*U(
     &  2))*MASSC+.5*(-S1*S2+C1*C2)*G*L2*MASSC+.5*C1*G*L1*MASSB+C1*G*L1*
```

```
     &  MASSC-TB3-TC3
      RHS(3) = .5*L1*L2*MASSC*S2*U(2)*U(2)+.5*(-S1*S2+C1*C2)*G*L2*MASSC-
     &  TC3
C
      CALL DECMP2(3,COEF,3,COEF,IPS,*9996,*9997)
      CALL SOLVE2(3,COEF,3,RHS,UDOT,IPS)
C
C
C     U4 IS DEFINED TO BE X
      UDOT(4) = U(1)
C
C     U5 IS DEFINED TO BE TH1
      UDOT(5) = U(2)
C
C     U6 IS DEFINED TO BE TH2
      UDOT(6) = U(3)
C
C
      RETURN
 9996 WRITE(*,9998)
      STOP
 9997 WRITE(*,9999)
      STOP
 9998 FORMAT(/1X,'ALL ELEMENTS IN A ROW OF COEF ARE ZEROS'/)
 9999 FORMAT(/1X,'A PIVOT ELEMENT ENCOUNTERED IN THE DECOMPOSITION',
     C        ' OF COEF IS ZERO')
      END
C
C
      SUBROUTINE CNTRL(T,U)
      IMPLICIT DOUBLE PRECISION (A-Z)
      DIMENSION U(6)
      COMMON/CPAR/IB,IC,MASSA,MASSB,MASSC,PI,L1,L2,G,DBUG
      COMMON/CONT/FA1,TB1,TB2,TB3,TC1,TC2,TC3
C
      X = U(4)
      TH1 = U(5)
      TH2 = U(6)
      FA1 = 0.0
      TB1 = 0.0
      TB2 = 0.0
      TB3 = 0.0
      TC1 = 0.0
      TC2 = 0.0
      TC3 = 0.0
```

```
C
      RETURN
      END
C
C
C
C
      SUBROUTINE DECMP2(N,A,IDIM,LU,IPS,*,*)
      IMPLICIT DOUBLE PRECISION (A-Z)
      INTEGER N,IDIM,IPS(N),I,J,K,IP,KP,KP1,NM1,IDXPIV
      DIMENSION A(IDIM,N),LU(IDIM,N),SCALES(100)
      ZERO=0.0D0
      DO 5 I=1,N
      IPS(I)=I
      ROWNRM=0.0D0
      DO 2 J=1,N
      LU(I,J)=A(I,J)
      ROWNRM=DMAX1(ROWNRM,DABS(LU(I,J)))
2     CONTINUE
      IF(ROWNRM.EQ.ZERO) RETURN 1
      SCALES(I)=1.0/ROWNRM
5     CONTINUE
      NM1=N-1
      DO 17 K=1,NM1
      BIG=0.0D0
      DO 11 I=K,N
      IP=IPS(I)
      SIZE=DABS(LU(IP,K))*SCALES(IP)
      IF(SIZE.LE.BIG) GO TO 11
      BIG=SIZE
      IDXPIV=I
11    CONTINUE
      IF(BIG.EQ.ZERO) RETURN 2
      IF(IDXPIV.EQ.K) GO TO 15
      J=IPS(K)
      IPS(K)=IPS(IDXPIV)
      IPS(IDXPIV)=J
15    KP=IPS(K)
      PIVOT=LU(KP,K)
      KP1=K+1
      DO 16 I=KP1,N
      IP=IPS(I)
      EM=LU(IP,K)/PIVOT
      LU(IP,K)=EM
      DO 16 J=KP1,N
```

```
            LU(IP,J)=LU(IP,J)-EM*LU(KP,J)
   16       CONTINUE
   17       CONTINUE
            IF(LU(IPS(N),N).EQ.ZERO) RETURN 2
            RETURN
            END
C
C
C
            SUBROUTINE SOLVE2(N,LU,IDIM,B,X,IPS)
            IMPLICIT DOUBLE PRECISION (A-Z)
            INTEGER I,J,IP,IP1,IM1,NP1,IBACK,N,IDIM,IPS(N)
            DIMENSION LU(IDIM,N),B(N),X(N)
            NP1=N+1
            X(1)=B(IPS(1))
            DO 2 I=2,N
            IP=IPS(I)
            IM1=I-1
            SUM=0.0D0
            DO 1 J=1,IM1
            SUM=SUM+LU(IP,J)*X(J)
   1        CONTINUE
            X(I)=B(IP)-SUM
   2        CONTINUE
            X(N)=X(N)/LU(IPS(N),N)
            DO 4 IBACK=2,N
            I=NP1-IBACK
            IP=IPS(I)
            IP1=I+1
            SUM=0.0D0
            DO 3 J=IP1,N
            SUM=SUM+LU(IP,J)*X(J)
   3        CONTINUE
   4        X(I)=(X(I)-SUM)/LU(IP,I)
            RETURN
              END
C
C
            SUBROUTINE DEQS(F,Y,*)
            IMPLICIT DOUBLE PRECISION (A-Z)
            INTEGER I,NCUTS,NEQ
            LOGICAL DBL,STPSZ
            EXTERNAL F
            COMMON/DFQLST/T,STEP,REL,ABS,NCUTS,NEQ,STPSZ
            DIMENSION F0(200),F1(200),F2(200),Y1(200),Y2(200),Y(NEQ)
```

```
      DATA HC/0.0D0/
C *** CHECK FOR INITIAL ENTRY AND ADJUST HC, IF NECESSARY.
      IF(NEQ.NE.0) GO TO 10
      HC=STEP
      RETURN
   10   IF(STEP.EQ.0.0D0) RETURN 1
C *** CHANGE DIRECTION, IF REQUIRED.
      IF(HC*STEP) 20,30,40
   20      HC=-HC
      GO TO 40
   30   HC=STEP
C *** SET LOCAL VARIABLES
   40   EPSL=REL
      FINAL=T+STEP
      H=HC
      TT=T+H
      T=FINAL
      H2=H/2.0D0
      H3=H/3.0D0
      H6=H/6.0D0
      H8=H/8.0D0
C *** MAIN KUTTA-MERSON STEP
50    IF((H.GT.0.0D0.AND.TT.GT.FINAL).OR.
  C      (H.LT.0.0D0.AND.TT.LT.FINAL)) GO TO 190
60    CALL F(TT-H,Y,F0)
      DO 70 I=1,NEQ
70    Y1(I)=F0(I)*H3+Y(I)
      CALL F(TT-2.0*H3,Y1,F1)
      DO 80 I=1,NEQ
80    Y1(I)=(F0(I)+F1(I))*H6+Y(I)
      CALL F(TT-2.0*H3,Y1,F1)
      DO 90 I=1,NEQ
90    Y1(I)=(F1(I)*3.0+F0(I))*H8+Y(I)
      CALL F(TT-H2,Y1,F2)
      DO 100 I=1,NEQ
100   Y1(I)=(F2(I)*4.0-F1(I)*3.0+F0(I))*H2+Y(I)
      CALL F(TT,Y1,F1)
      DO 110 I=1,NEQ
110   Y2(I)=(F2(I)*4.0+F1(I)+F0(I))*H6+Y(I)
C *** DOES THE STEPSIZE H NEED TO BE CHANGED?
      IF(EPSL.LE.0.0D0) GO TO 170
      DBL=.TRUE.
      DO 160 I=1,NEQ
      ERR=DABS(Y1(I)-Y2(I))*0.2
      TEST=DABS(Y1(I))*EPSL
```

```
      IF(ERR.LT.TEST.OR.ERR.LT.ABS) GO TO 150
C *** HALVE THE STEPSIZE
      H=H2
      TT=TT-H2
      IF(.NOT.STPSZ) GO TO 120
      TEMP=TT-H2
      WRITE(*,200) H,TEMP
C *** HAS THE STEPSIZE BEEN HALVED TOO MANY TIMES?
  120   NCUTS=NCUTS-1
      IF(NCUTS.GE.0) GO TO 130
      T=TT-H2
      WRITE(*,210) T
      RETURN 1
C *** IF STEPSIZE IS TOO SMALL RELATIVE TO TT TAKE RETURN 1
  130   IF(TT+H.NE.TT) GO TO 140
      T=TT
      RETURN 1
  140   H2=H/2.0D0
      H3=H/3.0D0
      H6=H/6.0D0
      H8=H/8.0D0
      GO TO 60
  150   IF(DBL.AND.64.0D0*ERR.GT.TEST
     C         .AND.64.0D0*ERR.GT.ABS) DBL=.FALSE.
  160   CONTINUE
C *** DOUBLE THE STEPSIZE, MAYBE.
      IF(.NOT.DBL.OR.DABS(2.0D0*H).GT.DABS(STEP).OR.
     C     DABS(TT+2.0D0*H).GT.DABS(FINAL).AND.
     C     DABS(TT-FINAL).GT.DABS(FINAL)*1.0D-7) GO TO 170
      H2=H
      H=H+H
      IF(STPSZ) WRITE(*,200) H,TT
      H3=H/3.0D0
      H6=H/6.0D0
      H8=H/8.0D0
      NCUTS=NCUTS+1
  170   DO 180 I=1,NEQ
  180   Y(I)=Y2(I)
      TT=TT+H
      GO TO 50
  190   IF(EPSL.LT.0.0D0) RETURN
C *** NOW BE SURE TO HAVE T=FINAL.
      HC=H
      H=FINAL-(TT-H)
      IF(DABS(H).LE.DABS(FINAL)*1.0D-7) RETURN
```

```
      TT=FINAL
      EPSL=-1.0D0
      H2=H/2.0D0
      H3=H/3.0D0
      H6=H/6.0D0
      H8=H/8.0D0
      GO TO 60
  200 FORMAT(1X,'THE STEPSIZE IS NOW ',1PD12.4,' AT T = ',1PD12.4)
  210 FORMAT(1X,'THE STEPSIZE HAS BEEN HALVED TOO MANY TIMES; ',
     C       'T = ',1PD12.4)
      END
```

A.6 Autolev to SIMULINK Interface Code

```
#!/bin/sh
#
FILENAME=${1}
AUTOLEV_OUTPUT=${FILENAME}.for   # 1st Input Varilable filename.for
MATLAB_GATEWAY=${FILENAME}g.f  # this should be filenameg.f
FORTRAN_FILE=${FILENAME}.f     # this should be filename.f
MATLAB_FILE=${FILENAME}s.m      # this should be filename.m
#(an s function for simulab)
TEMP1=/var/tmp/temp1.$$
TEMP2=/var/tmp/temp2.$$
TEMP3=/var/tmp/temp3.$$

# Find the dimension of the state vector
DIME1=`grep "DIMENSION U(.),UDOT(.),COEF" ${AUTOLEV_OUTPUT} | \
sed -e "s/      DIMENSION U(//" | awk -F\) '{print $1}'`
# Find the number of inputs (forces or torques)
DIME2=`grep "CALL UNCUPL(" ${AUTOLEV_OUTPUT} | sed -e "s/   \
CALL UNCUPL(//" | awk -F\, '{print $1}'`
# Find the line on which the useful part of the autolev output starts
LINE1=`grep -hn "SUBROUTINE EQNS" ${AUTOLEV_OUTPUT} | awk -F: '{print $1}'`
#
# Put useful code into TEMP1
#
tail +${LINE1} ${AUTOLEV_OUTPUT} > ${TEMP1}
#
#Find the line on which the useful part of the autolev output finishes
#
LINE2=`grep -hn "SUBROUTINE DEQS" ${TEMP1} | awk -F: '{print $1}'`
#
# Create 2 lines to go in front of autolev output to allow force to be
```

```
# passed in argument list.
#
cat > ${TEMP2} <<EOF
      SUBROUTINE EQNS(T,U,UDOT,F)
      DIMENSION F(${DIME2})
EOF
cat ${TEMP2} ${TEMP1} > ${TEMP3}
rm -f ${TEMP1}
rm -f ${TEMP2}
#
# Put editor instructions in TEMP1 to delete original line 1
#
cat > ${TEMP1} <<EOF
3
d
w
q
EOF
ed -s  ${TEMP3} < ${TEMP1}
rm -f ${TEMP1}
#
# Remove redundant end part of autolev output file
#
head -${LINE2} ${TEMP3} > ${FORTRAN_FILE}
rm -f ${TEMP3}
#
cat > ${MATLAB_GATEWAY} <<EOF
C ${MATLAB_GATEWAY} - Gateway function for ${FORTRAN_FILE}
C
C This is an example of the FORTRAN code required for interfacing
C a .MEX file to MATLAB.
C
C This subroutine is the main gateway to MATLAB.  When a MEX function
C  is executed MATLAB calls the USRFCN subroutine in the corresponding
C  MEX file.
C
      SUBROUTINE MEXFUNCTION(NLHS, PLHS, NRHS, PRHS)
      INTEGER*4 PLHS(*), PRHS(*)
      INTEGER   NLHS, NRHS
C
      INTEGER*4 MXCREATEFULL, MXGETPR
      INTEGER   MXGETM, MXGETN
C
C KEEP THE ABOVE SUBROUTINE, ARGUMENT, AND FUNCTION DECLARATIONS
C FOR USE  IN ALL YOUR FORTRAN MEX FILES.
```

```
C-----------------------------------------------------------------------
C
      INTEGER*4 YPP, TP, YP,FR
      INTEGER   M, N, MM, NN
      REAL*8 RYPP(${DIME1}), RTP, RYP(${DIME1}), RFR(${DIME2})
C RYPP Here is equivalent to UDOT in AUTOLEV
C RTP Here is equivalent to T in AUTOLEV
C RYP Here is equivalent to U in AUTOLEV
C FR IS AN INPUT FORCE OR TORQUE
C
C CHECK FOR PROPER NUMBER OF ARGUMENTS
C
      IF (NRHS .NE. 3) THEN
        CALL MEXERRMSGTXT('${FILENAME} requires three input arguments')
      ELSEIF (NLHS .NE. 1) THEN
        CALL MEXERRMSGTXT('${FILENAME} requires one output argument')
      ENDIF
C
C CHECK THE DIMENSIONS OF Y.  IT CAN BE ${DIME1} X 1 OR 1 X ${DIME1}.
C
      M = MXGETM(PRHS(2))
      N = MXGETN(PRHS(2))
C
      IF ((MAX(M,N) .NE. ${DIME1}) .OR. (MIN(M,N) .NE. 1)) THEN
        CALL MEXERRMSGTXT('${FILENAME} requires that Y be
& a ${DIME1} x 1 vector')
      ENDIF
C
C CHECK THE DIMENSIONS OF FR.  IT CAN BE ${DIME2} X 1 OR 1 X ${DIME2}.
C
      MM = MXGETM(PRHS(3))
      NN = MXGETN(PRHS(3))
C
      IF ((MAX(MM,NN) .NE. ${DIME2}) .OR. (MIN(MM,NN) .NE. 1)) THEN
        CALL MEXERRMSGTXT('${FILENAME} requires that FR be
&   a ${DIME2} x 1 vector')
      ENDIF

C
C CREATE A MATRIX FOR RETURN ARGUMENT
C
      PLHS(1) = MXCREATEFULL(M,N,0)
C
C ASSIGN POINTERS TO THE VARIOUS PARAMETERS
C
```

```
      YPP = MXGETPR(PLHS(1))
C
      TP = MXGETPR(PRHS(1))
      YP = MXGETPR(PRHS(2))
      FR = MXGETPR(PRHS(3))
C
C COPY RIGHT HAND ARGUMENTS TO LOCAL ARRAYS OR VARIABLES
      CALL MXCOPYPTRTOREAL8(TP, RTP, 1)
      CALL MXCOPYPTRTOREAL8(YP, RYP, ${DIME1})
      CALL MXCOPYPTRTOREAL8(FR, RFR, ${DIME2})
C
C DO THE ACTUAL COMPUTATIONS IN A SUBROUTINE
C       CREATED ARRAYS.
C
      CALL EQNS(RTP,RYP,RYPP,RFR)
C
C COPY OUTPUT WHICH IS STORED IN LOCAL ARRAY TO MATRIX OUTPUT
      CALL MXCOPYREAL8TOPTR(RYPP, YPP, ${DIME1})
C
      RETURN
      END
EOF
#
#  Create s function for use with simulab
#
cat > ${MATLAB_FILE} << EOF
function [sys, x0] = ${FILENAME}(t,x,u,flag,x0,out)
%
% t = time
% x = state vector (all velocities first, then all displacements )
% u = input vector (torques or forces)
% flag = determines what is returned in sys
% x0 = initial conditions set by user in simulab calling block
% out = vector of 1s and 0s set by user in simulab calling
% block to choose
% outputs.  A 1 means corresponding element in x is to be for output.
%
if abs (flag) == 1,
% return state derivatives
        sys  = ${FILENAME}(t,x,u);
elseif flag == 0,
% return size of parameters
%
% These may have to be changed manually for other than
% indicated default values.
```

```
%
size1 = ${DIME1}; % number of continuous states
size2 = 0;        % number of discrete states
size3 = sum(out); % number of outputs
size4 = ${DIME2}; % number of inputs  (Default:
  % half number of states)
size5 = 0;  % number of discontinuous roots )
size6 = 0;  % flag for direct feedthrough
        sys = [size1;size2;size3;size4;size5;size6];
%       x0 = [0;0;0;0;0;0]; % remove 1st % if you want to specify
% initial conditions here
%                                 otherwise, passed as parameter.
elseif flag == 3,
% return output
% out vector is a series of 1s and 0s to define whch
% elements of state vector are to be passed back as outputs.
%
sys = x(out,:);

else
        sys = [];
end
EOF
#
echo  " ____________________________________________________"
echo  "* Create mex function requires matlab, make sure you typed  *"
echo  "* 'use matlab' before runing this program.                   *"
echo  "*____________________________________________________*"
#
#
fmex ${FORTRAN_FILE} ${MATLAB_GATEWAY}

echo "*************************************************************"
echo " The path to the shared object libraries must be set before  "
echo " runing matlab, when using MEX files. Type 'use lang' on the "
echo " faculty computers before runing matlab.
echo " To change or input the model parameters edit the file       "
echo " ${FORTRAN_FILE} and recreat the MEX file by typing:         "
echo " 'fmex ${FORTRAN_FILE} ${MATLAB_FILE}'                       "
echo "*************************************************************"
```

Appendix B

Some Existing Modelling Software

B.1 Some Existing Modelling Software Packages

This section contains a list of some commercially available modelling programs and also a few non-commercial programs. Although this list does not contain all existing programs, its purpose is to give an idea on the multitude of modelling programs.

- **AUTOLEV**
  An interactive symbolic dynamics program based on Kane's formulation.
  From:
  Online Dynamics, Inc., Sunnyvale, CA, USA.

- **DADS**
  Dynamic Analysis and Design System.
  From:
  Computer Aided Design, Inc., Oakdale, Iowa, USA.

- **ADAMS**
  Multibody System Analysis.
  From:
  Mechanical Dynamics, Inc., Ann Arbor, Michigan, USA.

- **AUTOSIM**
  A software package for automatic modelling of multibody systems.
  From:
  MGA Software,
  200 Baker Avenue, Concord, MA 01742, USA.

- **MTT**
  Model Transformation Toolbox (Bound Graph Modelling).

Contact:
Prof. P. Gawthrop,
Mechanical Engineering Department. University of Glasgow. UK.

- **SD/FAST**
  Symbolic dynamic modelling software.
  From:
  Symbolic Dynamics, Inc., in the USA.
  In Europe Contact:
  Rapid Data LTD.
  Crescent House, Crescent Road, Worthing, BN11 5RW. UK.

- **SIMPACK**
  A Computer Program for Simulation of Large-motion Multibody Systems.
  From:
  MAN Technology AG, 8000 München, Germany.

- **AUTODYN & ROBOTRAN**
  Multibody dynamics programmes.
  Contact:
  Prof. P. Willems, Universite Catholique de Louvain, Belgium.

- **MEDYNA**
  An Interactive Analysis and Design Program for Geometrically Linear and Flexible Multibody Systems.
  From:
  T-Programm Gmbh, Technische Software and Engineering, Reutllingen, Germany.

- **NUBEMM**
  Dynamic Simulation Program (designed primarily for studying vehicle, handeling).
  Contact:
  Dr. E. Pankiewicz, BMW AG., 8000 München, Germany.

- **SYM**
  Program Package for Computer-aided Generation of Optimal Symbolic Models of Robot Mamipulators.
  Contact:
  Prof. M. Vukobratovic, Institut Mihailo Pupin, Beograd 11000. Yugoslavia.

- **CAMS**
  A Graphical Interactive System for Computer Simulation and Design of Multibody Systems.
  from:
  MECHATRONINA, 1126 Sofia, Bulgaria.

- **UCIN-DYNOCOMBS**
  Software for Dynamic Analysis of Constrained Multibody Systems.
  Contact:
  Prof. R. Huston. Department of Mech. and Ind. Engineering. University of Cincinnati, USA.

- **SPACAR**
  Computer Program for Dynamic Analysis of Flexible Mechanisms and Manipulators.
  From:
  Laboratory of Engineering Mechanics, Delft University of Technology. The Netherlands.

- **DISCOS & NBOD**
  Dynamic Interaction Simulation of Controls and Structure.
  From:
  COSMIC Program No. GSG-12810, GSG-12846
  NASA's Computer Software Management and Information Center.
  The University of Georgia, Athens, USA.

- **NEWEUL**
  Software for Generation of Symbolical Equations of Motion.
  from:
  Institue B of Mechanics, University of stuttgart, Germany.

- **COMPAMM**
  Computer Analysis of Machines and Mechanisms.
  A Simple and Efficient Code for Kinematic and Dynamic Numerical Simulation of 3-D Multibody Systems with Realistic Graphics.
  From:
  CORITEL S. A.
  Pl. Carlos Trias Bertran, 28020 Madrid, Spain.

- **DYMAC & DYSPAM**
  Programs for the Dynamic Analysis and Simulation of Planar Mechanisms and Multibody Systems.
  From:
  B. Paul Associates,
  204 Dodds Lane, Princeton, NJ 08540, USA.

- **MESA VERDE**
  A General-purpose Program Package for Symbolical Dynamics Simulations of Multibody Systems.
  From:
  Ingenieurgemeinschaft,
  Prof. R. Gnadler, Kaiserallee 111, 750 Karlsruhe 21, Germany.

- **PLEXUS**
  Software for the Numerical Analysis of the Dynamical Behavior of Rigid and Flexible Mechanisms.
  Contact:
  Prof. A. Barraco,
  E.N.S.A.M., 151 Boulevard de l'Hopital. Paris Cedex 13. France.

- **DYMOLA**
  Dynamic Modelling Laboratory.
  From:
  Dynasim AB,
  Research Park Ideon, S-223 70 Lund. Sweden.

Appendix C

Model Files

C.1 The Puma 560 three links model code

The following are relevant subroutines from the AUTOLEV generated FORTRAN code of the Puma 560 model. The inertial parameters used are those extracted from Armstrong's paper and friction parameters are included in the `CNTRL` subroutine. These are the values obtained after tuning. This file was used in simulation with SIMULINK.

```
C
C
C
      SUBROUTINE EQNS(T,U,UDOT)
      IMPLICIT DOUBLE PRECISION (A-Z)
      DIMENSION U(6),UDOT(6),COEF(3,3),RHS(3)
      COMMON/CPAR/PI,DEGTORAD,RADTODEG
      COMMON/CONT/T1,T2,T3
C
C
C
      TH1 = U(4)
      TH2 = U(5)
      TH3 = U(6)
C
C
      CALL CNTRL(T,U)
C
C
      S1 = DSIN(TH2)
      C1 = DCOS(TH2)
      S2 = DSIN(TH2+TH3)
      C2 = DCOS(TH2+TH3)
C
```

```
      COEF(1,1) = -3.197+8.688000000000001*(-2.706025E-002-(0.+0.*C2+0.*
     &  S1+.4318*C1+.244*S2)*(0.+0.*C2+0.*S1+.4318*C1+.244*S2))+17.4*(-5
     &  .175625E-002-((-6E-003)*S1+(6.8E-002)*C1+0.)*((-6E-003)*S1+(6.8E
     &  -002)*C1+0.))+.394*S1*S1+.85000000000000001*S2*S2
      COEF(1,2) = 3.9585*((-6.8E-002)*S1+(-6E-003)*C1)+1.429176*(0.*C1-0
     &  .*S2+.244*C2-.4318*S1)
      COEF(1,3) = 1.429176*(-0.*S2+.244*C2)
      COEF(2,1) = COEF(1,2)
      COEF(2,2) = -5.461+17.4*(-((-6.8E-002)*S1+(-6E-003)*C1)*((-6.8E-00
     &  2)*S1+(-6E-003)*C1)-(-(-6E-003)*S1-(6.8E-002)*C1)*(-(-6E-003)*S1
     &  -(6.8E-002)*C1))+8.688000000000001*(-(-0.*C2-0.*S1-.4318*C1-.244
     &  *S2)*(-0.*C2-0.*S1-.4318*C1-.244*S2)-(0.*C1-0.*S2+.244*C2-.4318*
     &  S1)*(0.*C1-0.*S2+.244*C2-.4318*S1))
      COEF(2,3) = -.212+8.688000000000001*(-(-0.*C2-.244*S2)*(-0.*C2-0.*
     &  S1-.4318*C1-.244*S2)-(-0.*S2+.244*C2)*(0.*C1-0.*S2+.244*C2-.4318
     &  *S1))
      COEF(3,1) = COEF(1,3)
      COEF(3,2) = COEF(2,3)
      COEF(3,3) = -.212+8.688000000000001*(-(-0.*C2-.244*S2)*(-0.*C2-.24
     &  4*S2)-(-0.*S2+.244*C2)*(-0.*S2+.244*C2))
C
      RHS(1) = 8.688000000000001*(-.1645*((-0.*C2*U(2)-0.*S1*U(2)-.4318*
     &  C1*U(2)-.244*S2*U(2)-.488*S2*U(3))*U(2)+(-0.*C2*U(3)-.244*S2*U(3
     &  ))*U(3)-(0.+0.*C2+0.*S1+.4318*C1+.244*S2)*U(1)*U(1))+((-0.*S2+.2
     &  44*C2)*U(3)+(0.*C1-0.*S2+.244*C2-.4318*S1)*U(2)-(U(2)+U(3))*0.*S
     &  2+.244*(U(2)+U(3))*C2+0.*C1*U(2)-.4318*S1*U(2)-.1645*U(1))*(0.+0
     &  .*C2+0.*S1+.4318*C1+.244*S2)*U(1))+17.4*(-.2275*(((-6.8E-002)*C1
     &  *U(2)-(-6E-003)*S1*U(2))*U(2)-((-6E-003)*S1+(6.8E-002)*C1+0.)*U(
     &  1)*U(1))+(((-6.8E-002)*S1+(-6E-003)*C1)*U(2)+(-6E-003)*C1*U(2)-(
     &  6.8E-002)*S1*U(2)-.2275*U(1))*((-6E-003)*S1+(6.8E-002)*C1+0.)*U(
     &  1))+(-.333*(-U(2)-U(3))*C2-.971*(U(2)+U(3))*C2)*S2*U(1)-(-1.183*
     &  (-U(2)-U(3))*S2-.121*(U(2)+U(3))*S2)*C2*U(1)-.787999999999998*C
     &  1*S1*U(1)*U(2)-F(1)-T1
      RHS(2) = 17.4*((((-6.8E-002)*C1*U(2)-(-6E-003)*S1*U(2))*U(2)-((-6E
     &  -003)*S1+(6.8E-002)*C1+0.)*U(1)*U(1))*((-6.8E-002)*S1+(-6E-003)*
     &  C1)+(-(-6E-003)*C1*U(2)+(6.8E-002)*S1*U(2))*(-(-6E-003)*S1-(6.8E
     &  -002)*C1)*U(2))+8.688000000000001*(((-0.*C1*U(2)+0.*S2*U(2)-.244
     &  *C2*U(2)-.488*C2*U(3)+.4318*S1*U(2))*U(2)+(0.*S2*U(3)-.244*C2*U(
     &  3))*U(3))*(-0.*C2-0.*S1-.4318*C1-.244*S2)+((-0.*C2*U(2)-0.*S1*U(
     &  2)-.4318*C1*U(2)-.244*S2*U(2)-.488*S2*U(3))*U(2)+(-0.*C2*U(3)-.2
     &  44*S2*U(3))*U(3)-(0.+0.*C2+0.*S1+.4318*C1+.244*S2)*U(1)*U(1))*(0
     &  .*C1-0.*S2+.244*C2-.4318*S1))+170.63571*(-(-6E-003)*S1-(6.8E-002
     &  )*C1)+85.2001752*(-0.*C2-0.*S1-.4318*C1-.244*S2)+.394*C1*S1*U(1)
     &  *U(1)+.85000000000000001*C2*S2*U(1)*U(1)-F(2)-T2
      RHS(3) = 8.688000000000001*(((-0.*C1*U(2)+0.*S2*U(2)-.244*C2*U(2)-
```

```
     &   .488*C2*U(3)+.4318*S1*U(2))*U(2)+(0.*S2*U(3)-.244*C2*U(3))*U(3))
     &   *(-0.*C2-.244*S2)+((-0.*C2*U(2)-0.*S1*U(2)-.4318*C1*U(2)-.244*S2
     &   *U(2)-.488*S2*U(3))*U(2)+(-0.*C2*U(3)-.244*S2*U(3))*U(3)-(0.+0.*
     &   C2+0.*S1+.4318*C1+.244*S2)*U(1)*U(1))*(-0.*S2+.244*C2))+85.2001
     &   52*(-0.*C2-.244*S2)+.85000000000001*C2*S2*U(1)*U(1)-F(3)-T3
C
      CALL UNCUPL(3,COEF,RHS,UDOT)
C
C
C     U4 IS DEFINED TO BE TH1
      UDOT(4) = U(1)
C
C     U5 IS DEFINED TO BE TH2
      UDOT(5) = U(2)
C
C     U6 IS DEFINED TO BE TH3
      UDOT(6) = U(3)
C
C
      RETURN
      END
C
C
C
C
C
      SUBROUTINE CNTRL(T,U)
      IMPLICIT DOUBLE PRECISION (A-Z)
      DIMENSION U(6)
      COMMON/CPAR/PI,DEGTORAD,RADTODEG
      COMMON/CONT/T1,T2,T3
C
      TH1 = U(4)
      TH2 = U(5)
      TH3 = U(6)
C
      if (U(1).EQ.0.) then
      FRIC1 = 0.0
      else
      FRIC1 = DSIGN(1,U(1))
      endif
      if (U(2).EQ.0.) then
      FRIC2 = 0.0
      else
      FRIC2 = DSIGN(1,U(2))
```

```
      endif
      if (U(3).EQ.0.) then
      FRIC3 = 0.0
      else
      FRIC3 = DSIGN(1,U(3))
      endif

      T1 = -8*U(1)-1*FRIC1
      T2 = -18*U(2)-0.5*FRIC2
      T3 = -4*U(3)-0.01*FRIC3
C

      RETURN
      END
C
C
```

C.2 SCARA manipulator model

This is the relevant FORTRAN portion of an AUTOLEV generated dynamic model of a SCARA type manipulator

```
C
C
      SUBROUTINE EQNS(T,U,UDOT)
      IMPLICIT DOUBLE PRECISION (A-Z)
      DIMENSION U(6),UDOT(6),COEF(3,3),RHS(3)
      COMMON/CPAR/IAXX,IAYY,IAZZ,IBXX,IBYY,IBZZ,ICXX,ICYY,ICZZ,MA,MB,MC,
     &  PI,DEGTORAD,RADTODEG,G,L1,L2,CX1,CX2,CZ3,CO1,CO2,CO3
      COMMON/CONT/T1,T2,T3,FOR1,FOR2,FOR3
C
C
C
      TH1 = U(4)
      TH2 = U(5)
      Z3 = U(6)
C
C
C
      CALL CNTRL(T,U)
C
C
      S1 = DSIN(TH2)
```

```
      C1 = DCOS(TH2)
C
      COEF(1,1) = (-2*C1*CX2*L1-CX2*CX2-L1*L1)*MB+(-2*C1*L1*L2-L1*L1-L2*
     &  L2)*MC-CX1*CX1*MA-IAZZ-IBZZ-ICZZ
      COEF(1,2) = (-C1*CX2*L1-CX2*CX2)*MB+(-C1*L1*L2-L2*L2)*MC-IBZZ-ICZZ
      COEF(1,3) = 0.0
      COEF(2,1) = COEF(1,2)
      COEF(2,2) = -CX2*CX2*MB-IBZZ-ICZZ-L2*L2*MC
      COEF(2,3) = 0.0
      COEF(3,1) = 0.0
      COEF(3,2) = 0.0
      COEF(3,3) = -MC
C
      RHS(1) = (((-(U(1)+U(2))*CX2-CX2*U(2))*U(1)-CX2*U(2)*U(2))*L1*S1+C
     &  X2*L1*S1*U(1)*U(1))*MB+(((-(U(1)+U(2))*L2-L2*U(2))*U(1)-L2*U(2)*
     &  U(2))*L1*S1+L1*L2*S1*U(1)*U(1))*MC-FOR1-TF1
      RHS(2) = CX2*L1*MB*S1*U(1)*U(1)-FOR2+L1*L2*MC*S1*U(1)*U(1)-TF2
      RHS(3) = -FF3-FOR3+G*MC
C
      CALL UNCUPL(3,COEF,RHS,UDOT)
C
C
C     U4 IS DEFINED TO BE TH1
      UDOT(4) = U(1)
C
C     U5 IS DEFINED TO BE TH2
      UDOT(5) = U(2)
C
C     U6 IS DEFINED TO BE Z3
      UDOT(6) = U(3)
C
C
      RETURN
      END
C
```

Appendix D

Underwater Manipulator Model File

the following is the matlab file containg the two link underwater manipulator and its controller.

twolmar.m

```
function [u1,funeval1,detmass] = mar1(xi,h,t,p);
%
%
U(1) = xi(1) ;
U(2) = xi(2) ;
TH1 = xi(3) ;
TH2 = xi(4) ;
%
  S1 = sin(TH2);
  C1 = cos(TH2);
%
 C2=C1; S2=S1;
%
%%%%%%%%%% The Constant Values related to mass, inertia ...etc.
CX1 = 0.5 ; CX2 = 0.5 ;
L1 = 1.0 ; L2 = 1.0 ; G = 10. ;
MA = 2.0 ; MB = 2.0 ; Fric1 = 0.3 ; Fric2 = 0.3;
IAZZ = 0.167; IBZZ = 0.167;
Ca=2; Cd=1; Ro= 1025; rayon1=0.025; rayon2=0.025;

%%%%%% The Added mass effect parameters %%%%%%%%
%Ca=0; % this cancels the added mass effects when uncommented
K1= -Ca*Ro*pi*(rayon1*rayon1) ;
K3= -Ca*Ro*pi*(rayon2*rayon2) ;
```

```
K2= K3*rayon2 ;
% added torques are split to added mass
% component and added coriolis component
% for each added torque.
%
% Torq1=(1/3)K1*L1^3(Udot(1)) then admas1
admas11=(1/3)*K1*(L1^3)    ;
%
% Torq2=k2(L1^2)(S2^2)Udot(1)+k2(L1^2)S2C2(U(1)U(2)) Then
%admas21=K2*(L1^2)*(S2^2) ;
%adtorq21=K2*(L1^2)*S2*C2*(U(1)*U(2)) ;
admas21 =0.; adtorq21=0.0; % comment this line to set admas21 &
                           % adtorq21 to their calculated values
%
% Torq3=k3*((1/3)(L2^3)+(1/2)L1(L2^2)C2)Udot(1)+K3(1/3)
% (L2^3)Udot(2) - k3(1/2)L1(L2^2)S2(U(1)U(2))   Then....
admas22(1)=K3*((1/3)*(L2^3)+(1/2)*L1*(L2^2)*C2)  ;
admas22(2)=K3*(1/3)*(L2^3) ;
adtorq22= - K3*(1/2)*L1*(L2^2)*S2*(U(1)*U(2))  ;
%
%%%%%%%%%%%%%% Added mass %%%%%%%%%%%%%%%%%%
mass_ad(1,1)=admas11+admas21+admas22(1) ;
mass_ad(1,2)=admas22(2) ;
mass_ad(2,1)=admas22(1) ;
mass_ad(2,2)=admas22(2) ;

% Added Torque;

Torq_ad(1)= adtorq21 +adtorq22;
Torq_ad(2)= adtorq22 ;
%%%%%%%%%%%%%% The Drag FORCE/TORQUE Parameters %%%%%%%%
%Cd =0; % this cancel the drag effect when uncommented.
V1= U(1)*L1*C2 ;
V2= (U(1)+U(2))*L2+V1 ;
KD1= -(Ro)*Cd*rayon1;
KD2= -(Ro)*Cd*rayon2;

%The torq applied on link1 by link1 drag is :
T11 = (0.25)*KD1*sign(U(1))*U(1)*U(1)*(L1)^4 ;
%
% The torq applied on link2 caused by link2 drag
% is complicated and depend on many factors:

if ( U(1) == -U(2) ),  % start of if loop (0)
   T22= 0.5*KD2*sign(V1)*((U(1)*L1*C2)^2)*L2^2;
```

```
else,     % if loop (0)
 L0= -(V1)/(U(1)+U(2));
Td_L0=(1/3)*((U(1)+U(2))^2)*(L0^3)+(0.5)*((U(1)*L1*C2)^2)* ...
(L0^2)+(2/3)*U(1)*L1*C2*(U(1)+U(2))*(L0^3);
Td_L2=(1/3)*((U(1)+U(2))^2)*(L2^3)+(0.5)*((U(1)*L1*C2)^2)* ...
(L2^2)+(2/3)*U(1)*L1*C2*(U(1)+U(2))*(L2^3) ;

 if ( V1 >= 0),    % start of big (drag) if loop         (1)

    if ( V2 >= 0 ),  % second if loop(2) No reverse flow;
      Tdrag2 = KD2*Td_L2;
      T22 = Tdrag2;
    else  % (2)     There is reverse flow and L0=[0,..L2]
      Tdrag2 = KD2*(2*Td_L0 - Td_L2) ;
      T22 = Tdrag2 ;
    end  %  (2)

 else,                        % V1 is negative               (1)

    if ( V2 > 0.0),     % Third if loop (3) there revese flow
      Tdrag2 = KD2*(2*Td_L0 - Td_L2);
      T22 = -Tdrag2;

    else,  %  (3) V2 is negative here or nill. No Reverse flow
      Tdrag2 = KD2*Td_L2;
      T22 = -Tdrag2;
    end  % end of third if loop (3)

 end           % End of big (drag) if loop (1)
end    % End of if loop (0)
Tdr2 = T22;
Tdrag_ad(1) =T11+T22;
Tdrag_ad(2) =  T22;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     Control Design
%
z1=1;
a11=1; a01=0.25; af11=10; af10=25;

xt1= xi(9);
dxt1= xi(10);
ddxt1= -a01*xi(9)-a11*xi(10)+a01*z1;
```

```
xf1=xi(13);
dxf1= xi(14);
ddxf1= -af10*xi(13)-af11*xi(14)+af10*(xi(3)-xi(5));
xm1=xi(5);
dxm1= xi(6);
L2fh1= -0.*xi(5)-0.*xi(5)^3 -4*xi(6)-1*xi(6)^3; LgLfh1=0.02;
Tau1= (LgLfh1)^(-1)*(-L2fh1+(ddxt1-ddxf1)+2*(dxt1-dxf1-dxm1)+ ...
1*(xt1-xf1-xm1));

if ( Tau1 > 10),
Tau1= 10;
elseif( Tau1 < -10),
Tau1= -10;
end

ddxm1= L2fh1+LgLfh1*Tau1;


z2=-1;
a12=2; a02=1; af21=12; af20=36;

xt2= xi(11);
dxt2=xi(12);
ddxt2= -a02*xi(11)-a12*xi(12)+a02*z2;
xf2= xi(15);
dxf2= xi(16);
ddxf2= -af20*xi(15)-af21*xi(16)+af20*(xi(4)-xi(7));
xm2= xi(7);
dxm2= xi(8);
L2fh2= -0.0*xi(7) - 1*xi(8) - 1*xi(8)^3; LgLfh2=0.02;
Tau2= (LgLfh2)^(-1)*(-L2fh2+(ddxt2-ddxf2)+2*(dxt2-dxf2-dxm2)+ ...
1*(xt2-xf2-xm2));

if ( Tau2 > 20),
Tau2= 20;
elseif( Tau2 < -20),
Tau2= -20;
end

ddxm2= L2fh2+LgLfh2*Tau2;


actorq = [ Tau1 Tau2]';
```

```
%
% Equations of the dry manipulator
%
      COEF(1,1) = (-(C1*CX2+L1)*(C1*CX2+L1)-CX2*CX2*S1*S1)*MB ...
      -CX1*CX1*MA -IAZZ-IBZZ;
      COEF(1,2) = (-(C1*CX2+L1)*C1*CX2-CX2*CX2*S1*S1)*MB-IBZZ;
      COEF(2,1) = (-(C1*CX2+L1)*C1*CX2-CX2*CX2*S1*S1)*MB-IBZZ;
      COEF(2,2) = -CX2*CX2*MB-IBZZ;
%
      RHS(1) = (-((-(C1*CX2+L1)*U(1)-2*C1*CX2*U(2))*U(1) ...
      -C1*CX2*U(2)*U(2))*CX2*S1+((-CX2*S1*U(1)-2*CX2*S1*U(2))* ...
      U(1)-CX2*S1*U(2)*U(2))*(C1*CX2+L1))*MB +Fric1*U(1);
      RHS(2) = (-((-(C1*CX2+L1)*U(1)-2*C1*CX2*U(2))*U(1) ...
      -C1*CX2*U(2)* U(2))*CX2*S1+((-CX2*S1*U(1)-2*CX2*S1* ...
      U(2))*U(1)  -CX2*S1*U(2)*U(2))*C1*CX2)*MB +Fric2*U(2);
%
%
%%%%%%%%%%%%%%%%%% Calculate angular dotdot  %%%%%%
%% The form of equation initially is:
%% [COEF]*[UDOT]=RHS
%% New effects are added:
%% mass = [COEF]+[mass_ad]
%% RHS = RHS+[Torq_ad]+[Tdrag_ad]

mass_new = COEF+mass_ad;

detmass=[det(COEF)  det(mass_new)];

RHS_new = RHS'-(Torq_ad)'-(Tdrag_ad)';
inv_mass = inv(mass_new);

 Udot = inv_mass*RHS_new -inv_mass*actorq;

funeval1(1) = Udot(1) ;
funeval1(2) = Udot(2) ;
funeval1(3) = U(1) ;
funeval1(4) = U(2) ;
funeval1(5)= dxm1;
funeval1(6)= ddxm1;
funeval1(7)=dxm2;
funeval1(8)=ddxm2;
funeval1(9)= dxt1;
funeval1(10)= ddxt1;
funeval1(11)=dxt2;
funeval1(12)=ddxt2;
```

Appendix D.

```
funeval1(13)= dxf1;
funeval1(14)= ddxf1;
funeval1(15)=dxf2;
funeval1(16)=ddxf2;
u1(1)=Tau1 ;
u1(2)=Tau2 ;
funeval1= funeval1';
```

Bibliography

[1] B. Armstrong, O. Khatib, and J. Burdik. "The explicit dynamic model and inertial parameters of the puma 560 arm." in *The IEEE International Conference on Robotics and Automation, San Francisco*, pp. 510–518, 1986.

[2] P. I. Corke, *Visual Control of Robots: High-Performance Visual Servoing.* John Wiley, 1996.

[3] P. Mowforth, "Intelligent robot: the state of play." *A seminar presented at the International Robot Olympics, Strathclyde University, Glasgow, Scotland*, Sept. 1990.

[4] J. J. Craig, *Introduction to Robotics: Mechanics and Control(2nd ed.).* Addison-Wesley, 1989.

[5] F. N-Nagy and A. Siegler, *Engineering Foundations of Robotics.* Prentice-Hall International (UK) Ltd, 1987.

[6] D. W. Roberts, *Bond Graph Model Based Control of Robotic Manipulators.* Ph.D. Thesis, Glasgow University, Dept. of Mechanical Engineering, 1993.

[7] H. V. Brussel and K. Jankowski, "Dynamic modelling of multibody systems," in *Lecture Notes of the 1990 Integrated European Course in Mechatronics, Computer Controlled Motion and Robotics* (J. D. Schutter and H. V. Brussel, eds.), Haverlee, Belgium, June 1990.

[8] A. M. Foss, "Towards a general procedure for dynamic model validation." *Transaction of the Institute of Measurement and Control, Special Issue on Model Validation*, vol. 12, no. 4, pp. 174–177, 1990.

[9] "Special issue on model validation," *Transactions of the Institute of Measurement and Control*, vol. 12, no. 4, pp. 166–207, 1990.

[10] Z. S. Roth, B. W. Mooring, and B. Ravani, "An overview of robot calibration," *The IEEE Journal of Robotics and Automation*, vol. RA-3, pp. 377–385, Oct. 1987.

[11] H. W. Stone, A. C. Sanderson, and C. P. Neuman, "Arm signature identification," in *The IEEE International Conference on Robotics and Automation, San Francisco, CA*, pp. 41–48, April 1986.

[12] *The IEEE Journal of Oceanic Engineering*, vol. OE-15, July 1990.

[13] K. S. Fu, R. C. Gonzalez, and C. S. Lee. *Robotics: Control, Sensing, Vision and Intelligence*. McGraw-Hill, Inc., 1987.

[14] T. R. Kane and D. A. Levinson, *Dynamics: Theory and Applications*. McGraw-Hill Book Company, 1985.

[15] X. Li, "Modelling and validation of robot manipulators," M.Sc. Thesis, Glasgow University, Dept. of Mechanical Engineering, 1989.

[16] W. Schiehlen, ed., *Multibody Systems Handbook*. Springer-Verlag, 1990.

[17] R. E. Shannon, *System Simulation: The Art and the Science*. Prentice-Hall, Englewood Cliffs, USA, 1975.

[18] J. A. Spriet and G. C. Vansteenkiste, *Computer-Aided Modelling and Simulation*. Academic Press, London, 1982.

[19] D. J. Murray-Smith, "Problems and prospects in the validation of dynamic models," in *Proc. of the 4th Int. Symp. on Systems Analysis and Simulation: Computational Systems Analysis, Berlin*, pp. 21–27, August 1992.

[20] P. J. Gawthrop, H. Mirab, and X. Li, "Robot model validation," *Transaction of the Institute of Measurement and Control, Special Issue on Model Validation*, vol. 12, no. 4, pp. 197–207, 1990.

[21] B. Raucent and J. Samin, "Modelling and identifying the dynamic parameters of robot manipulators," in *Robot Calibration* (R. Bernhardt and S. L. Albright, eds.), 1993.

[22] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 2, pp. 215–221, June 1955.

[23] D. E. Whitney, C. A. Lozinski, and J. M. Rourke, "Industrial robot forward calibration method and results," *Trans. of ASME: Journal of Dynamic Systems, Measurement, and Control*, vol. 108, pp. 1–8, Mar. 1986.

[24] H. W. Stone, *Kinematic Modeling, Identification, and Control of Robotic Manipulators*. Kluwer Academic Publishers, 1987.

[25] D. Stanton and G. A. Parker, "Experimental kinematic calibration using a modified s-model," in *ICARCV'92: The Second Int. Conf. on Automation, Robotics and Computer Vision*, vol. 3, Sept. 1992.

[26] W. Khalil, M. Gautier, and C. Enguehard, "Identifiable parameters and optimum configurations for robots calibration," *Robotica*, vol. 9, pp. 63–70, 1991.

[27] M. R. Driels, "Using passive end_point motion constraints to calibrate robot manipulators," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 115, no. 3, pp. 560–566, 1993.

[28] M. R. Driels and U. S. Pathre, "Vision_based automatic theodolite for robot calibration," *IEEE Transaction on Robotics and Automation*, vol. 7, no. 3, pp. 351–360, 1991.

[29] M. R. Driels and U. S. Pathre, "Robot calibration using an automated theodolite," *International Journal of Advanced Manufacturing Technology*, vol. 9, no. 2, pp. 114–125, 1994.

[30] B. W. Mooring and S. S. Padavala, "The effect of kinematic model complexity on manipulator accuracy," in *The IEEE International Conference on Robotics and Automation, Boston Spa, USA*, pp. 593–598, 1989.

[31] D. Yoerger and J.-J. E. Slotine, "Robust trajectory control of underwater vehicles," *The IEEE Journal of Oceanic Engineering*, vol. OE-15, July 1990.

[32] T. I. Fossen and S. I. Sagatun, "Adaptive control of nonlinear underwater robotic system," *Modeling, Identification and Control*, vol. 12, no. 2, 1991.

[33] B. Jalvin and N. Storkersen, "The control of an autonomous underwater vehicle," in *Proceeding of the IEEE Conference on Control Applications, Glasgow*, 1994.

[34] K. Ioi and K. Itoh, "Modelling and simulation of an underwater manipulator," *Advanced Robotics*, vol. 4, no. 4, pp. 303-317, 1990.

[35] P. J. Gawthrop, "Automated system modelling," in *IEE Colloquium on Bond Graphs in Control*, 1990.

[36] S. Nicosia, P. Tornei, and A. Tornambe, "Dynamic modelling of flexible robot manipulators," *Rep. CH2282-2/86/0000/0365*, 1986.

[37] R. P. Paul, *Robot Manipulators: Mathematics, Programming and Control*. The MIT Press Series in Artificial Intelligence, 1981.

[38] M. Abderrahim, "Robot modelling: Evaluation of computer packages." Control Group Research Report R-91/2. Glasgow University, Dept. of Mechanical Engineering, 1991.

[39] D. B. Scaechter and D. A. Levinson, *AUTOLEV User's Manual*. Online Dynamics Inc., 1988.

[40] *SD/FAST Application Note: Balancing Sticks Using SD/FAST with ACSL and PRO-MATLAB*. Symbolic Dynamics Inc., September, 1988.

[41] *DADS User's Manual*. Rev. 6.0, Computer Aided Design Software Inc., 1989.

[42] *MATLAB News & Notes: The Newsletter for MATLAB, SIMULINK and Toolbox Users.* The MathWorks, Inc., Summer 1995.

[43] T. H. Ang, "Front end to autolev," Final Year Project Report No. 910883, Glasgow University, Dept. of Mechanical Engineering, 1993.

[44] C. Moler, J. Little, and S. Bangert, *MATLAB user's Guide.* Mathworks Inc., 1992.

[45] A. Whittaker, "Interfacing autosim and autolev to matlab and simulab with validation using puma models," Control Group Research Report R-92/13, Glasgow University, Dept. of Mechanical Engineering, 1992.

[46] D. Kerr and P. J. Larcombe, "The AA 300 robot, lamberton robotics ltd: Geometric model and dynamic data," Control Group Research Report R-90/18, Glasgow University, Dept. of Mechanical Engineering, 1990.

[47] M. Abderrahim, P. J. Larcombe, and A. R. Whittaker, "The three dimensional computer-aided model and simulation of an industrial robot: The aa 300 robot, lamberton robotics, ltd.," Control Group Research Report R-91/16, Glasgow University, Dept. of Mechanical Engineering, 1991.

[48] M. Abderrahim, P. J. Larcombe, and A. R. Whittaker, "A computer-aided model and simulation of an industrial robot," in *Eighth International Conference on Computer-Aided Production Engineering, Edinburgh*, pp. 201–209, Aug. 1992.

[49] A. R. Whittaker, M. Abderrahim, and D. W. Roberts, "Vibration tests on components of the lamberton aa 300 robot," Control Group Research Report R-91/3, Glasgow University, Dept. of Mechanical Engineering, 1991.

[50] T. E. Bruns and D. A. Tortorelli, "Computer-aided optimal design of flexible mechanisms," in *The Twelfth Conference of the IMC, Competitive Manufacturing, Cork, Ireland*, pp. 29–36, Sept. 1995.

[51] R. Mayer and G. A. Parker, "Calibration and assessment of a laser based instrument for robot dynamic measurement," in *IMEKO XI, 11th Triennial World Congress of the International Measurement Confederation, Houston, Texas*, p. Paper No 172, Oct. 1988.

[52] P. J. Gawthrop, A. R. Whittaker, J. Howell, P. J. Larcombe, D. W. Roberts, and M. Abderrahim, "Metamodelling of robots, 1st progress review report: 10/89–5/90, GR/F 57779: October 1989–September 1992," Control Group Research Report R-90/10, Glasgow University, Dept. of Mechanical Engineering, 1990.

[53] *Lab-PC User's Manual.* National Instrument Corporation, Part Number 320205-01, September, 1991.

[54] P. I. Corke, "Operational details of the unimation puma servo system," *CSIRO Division of Manufacturing Technology, Preston, Australia*, Sept. 1991.

[55] *Viglen Genie 4 User's Guide.* Viglen Limited. Document MAG040-00-01, 1990.

[56] *Instruction Manual for the MITC-12.* 3D Digital Design and Development Ltd., London, 1991.

[57] D. Stanton and G. A. Parker, "The condition vector as a measure of robot kinematic performance for off-line programming, theory and experimental validation," in *The ASME WAM, San Francisco, CA* (Y. Youssef-Toumi and H. Kazerooni, eds.), vol. DSC-14, pp. 69–76, Dec. 10-15th 1989.

[58] *LabWindows User's Manual.* National Instrument Corporation, 1989.

[59] *PUMA 500/600 Robot Technical Manual.* Unimation (Europe) Ltd Telford, Shropshire, England, Nov. 1982.

[60] P. Elosegui, "Development and implementation of an on-line multivariable control algorithm for the force control of an industrial robot," Tech. Rep. No. OUEL 1748/88, University of Oxford, Department of Engineering Science, 1990.

[61] D. W. Roberts, "Design and construction of a two-link manipulator," Control Group Research Report R-90/11, Glasgow University, Dept. of Mechanical Engineering, 1990.

[62] P. J. Gawthrop, "Automatic tuning of a motion controller," Control Engineering Report 88, Glasgow University, Dept. of Mechanical Engineering, 5, August 1988.

[63] C. P. Neuman and P. K. Khosla, "Parameter identification for robot control," in *Winter Meeting of ASME; Dynamic Systems Modelling and Control*, pp. 213–235, Nov 1985.

[64] C. Atkeson, C. An, and J. Hollerbach, "Estimation of inertial parameters of manipulator load and links," *Int. Journal of Robotics Research*, vol. 5, no. 3, pp. 101–119, 1986.

[65] M. Gautier and W. Khalil, "A direct determination of minimum inertial parameters of robots," in *Proc. of the IEEE Conference on Robotics and Control*, pp. 1682–1687, 1988.

[66] "Special issue on model validation," *Transactions of the Institute of Measurement and Control*, vol. 8, pp. 170–232, 1986.

[67] M. H. Butterfield, "A method of quantitative validation based on model distortion," *Transaction of the Institute of Measurement and Control. Special Issue on Model Validation*. vol. 12. no. 4. pp. 167–173. 1990.

[68] C. Canudas de Witt, *Adaptive Control for Partially Known Systems*. Amsterdam: Elsevier, 1988.

[69] P. J. Gawthrop, "Parametric identification of transient signals." *IMA Journal of Mathematical Control and Information*. vol. 1. pp. 117–128. 1984.

[70] P. I. Corke and B. Armstrong, "Puma 560 dynamics: A numeric and symbolic model," *A Draft paper*, Sept. 1992.

[71] B. M. Mooring and T. J. Pack, "Aspects of robot repeatability." *Robotica*. vol. 5, pp. 223–230, 1987.

[72] J. Chen and L.-M. Chao, "Positioning error analysis for robot manipulators with all rotary joints," *The IEEE Journal of Robotics and Automation*. vol. RA-3, pp. 539–545, Dec. 1987.

[73] S. Hayati, K. Tso, and G. Roston, "Robot geometry calibration." in *Proceedings of The IEEE Decision and Control Conference*. pp. 798–80. 1988.

[74] S. Hayati and M. Mirmirani, "Puma 600 robot arm geometric calibration." in *The First IEEE International Conference on Robotics. Atlanta*. March. 13 1984.

[75] J. Chen, C. B. Wang, and C. S. Yang, "Robot positioning accuracy improvement through kinematic parameter identification," in *The Proceedings of the Third Canadian CAD/CAM in Robotics Conference, Toronto*. pp. 4.7–4.12. June 1984.

[76] M. R. Driels and U. S. Pathre, "Generalized joint model for robot manipulator kinematic calibration and compensation," *Journal of Robotic Systems*. vol. 4, no. 1, pp. 77–114, 1986.

[77] U. S. Pathre and M. R. Driels, "Simulation experiments in parameter identification for robot calibration," *International Journal of Advanced Manufacturing Technology*, vol. 5, pp. 13–33, 1990.

[78] S. M. Harb and M. Burdekin, "A systematic approach to identify the error motion of an n-degree of freedom manipulator," *International Journal of Advanced Manufacturing Technology*, vol. 9, no. 2, pp. 126–133. 1994.

[79] M. R. Driels and W. E. Swaze, "Automated partial pose measurement system for manipulator calibration experiments," *IEEE Transaction on Robotics and Automation*, vol. 10, no. 4, pp. 430–440, 1994.

[80] M. Abderrahim and A. R. Whittaker, "Positioning improvement of industrial manipulators," in *The Twelfth Conference of the IMC, Competitive Manufacturing, Cork, Ireland*, pp. 37–44. Sept. 1995.

[81] S. Hayati, "Robot arm geometric link parameter estimation," in *The 22nd IEEE Conference on Decision and Control, San Antonio, Texas*, pp. 1477–1483, Dec 1983.

[82] B. W. Mooring, "The effect of joint axis misalignment on robot positioning accuracy," in *Proc. ASME 1983 International Computer in Engineering Conference, Chicago, Illinois*, vol. 2, pp. 151–155. 1983.

[83] D. Stanton, *Enhancement to Off-Line Programing Through Improvements to Robot Kinematic Performance*. D.Phil. thesis, Surrey University, 1992.

[84] B. W. Mooring and G. R. Tang, "An improved method for identifying the kinematic parameters in a six axis robot," in *The Proceeding of the International Computers in Engineering Conference*, pp. 79–84. ASME, New York, August 1984.

[85] R. Bernhardt and S. L. Albright, eds., *Robot Calibration*. Chapman & Hall, 1993.

[86] L. J. Everett and T. W. Hsu, "The theory of kinematic parameter identification for industrial robots," *Trans. ASME. Journal of Dynamic Systems, Measurement and Control*, vol. 110, pp. 96–100, March 1988.

[87] T. C. Hsia, *System Identification: Least Squares method*. Lexington Books, 1977.

[88] D. James, D. Burley, D. Clements, P. Dyke, J. Searl, and J. Wright, *Modern Engineering Mathematics*. Addison-Wesley, 1992.

[89] P. Robinson and P. Orzechowski, "Experiences of robot simulation & off-line programming using workspace," in *The Twelfth Conference of the IMC, Competitive Manufacturing, Cork, Ireland*, pp. 141–148, Sept. 1995.

[90] D. Liddle, "Trojan: Remotely operated vehicle," *The IEEE Journal of Oceanic Engineering*, vol. OE-11, July 1986.

[91] M. Nomoto and M. Hattori, "A deep ROV 'dolphin 3k': Design and performance analysis," *The IEEE Journal of Oceanic Engineering*, vol. OE-11, July 1986.

[92] A. K. Ramadorai and T. J. Tarn, "On modeling and adaptive control of underwater robots," in *Proc. of a U.S.-Italy Workshop in honour of Prof. Antonio Ruberti* (A. Isidori and T. J. Tarn, eds.), Capri, 15-17 June 1992.

[93] A. Koivo, *Fundamentals for Control of Robotic Manipulators*. J. Wiley and Sons, 1989.