



Pilkington Library

Author/Filing Title HAY

Vol. No. Class Mark T

**Please note that fines are charged on ALL
overdue items.**

LOAN COPY

0402152603



BADMINTON PRESS
UNIT 1 BROOK ST
SYSTON
LEICESTER, LE7 1GD
ENGLAND
TEL : 0116 260 2917
FAX : 0116 269 6639



**IDENTIFICATION OF ROBOTIC MANIPULATORS'
INVERSE DYNAMICS COEFFICIENTS
VIA MODEL-BASED ADAPTIVE NETWORKS**

by


Robert J. Hay

A Doctoral Thesis

Submitted in partial fulfilment of the requirements
for the award of
Doctor of Philosophy of Loughborough University

September 1998

© by R.J. Hay 1998

 Leeds University Pitt Rivers Library
Date Dec 99
Class
Acc No. 040215 260

M0000928LB

ACKNOWLEDGMENTS

Dr. I.P.W. Sillitoe (Assistant Professor and joint Supervisor, Borås University)
for expert and friendly advice on all aspects of this research.

Dr. R.M. Voyles (Assistant Professor, University of Minnesota)
for helping a complete stranger and generating the case study data.

Dr. D.J. Mulvaney (Lecturer and joint Supervisor, Loughborough University)
for additional expert advice and support.

CONTENTS

ABSTRACT	1
1. INTRODUCTION	2
1.1 Statement of the Problem	2
1.2 Method of Solution	2
1.3 Related Works	3
1.4 Layout of this Thesis	4
1.5 Declaration of Originality	5
1.6 Conventions and Nomenclature	6
References	7
2. THE MOTION OF ROBOTIC MANIPULATORS	9
2.1 The Structure of a Manipulator	9
2.2 Manipulator Dynamics	10
2.3 Lagrange's Equation of Motion	11
2.4 The Euler-Lagrange Equation	17
2.5 Non-Lagrangian Dynamics	18
2.6 The Inverse Dynamics of the PUMA 560	24
2.7 Model-Based Control Algorithms	31
2.8 Summary	33
References and Further Reading	34
3. ADAPTIVE NETWORKS	38
3.1 General Adaptive Network Definitions	38
3.2 Properties of Adaptive Networks	40
3.3 Network Architectures	41
3.4 Learning Algorithms	47
3.5 Adaptive Networks and Manipulator Dynamics	54
3.6 Overtraining and Generalisation	56
3.7 Summary	58
References and Further Reading	59
4. ANALYSIS OF TRAINING IN ANALOGUE OUTPUT NETWORKS ..	66
4.1 Why Curve-Fitting is Analogous to Network Training	66
4.2 Network Order	68
4.3 Network Flexibility	70
4.4 Summary	73
References and Further Reading	73

5. THE CSLC NETWORK FOR INVERSE DYNAMICS	74
5.1 Decomposition of the Inverse Dynamics of a PUMA 560	74
5.2 Properties of the CSLC Network	83
5.3 Preliminary Parameter Identification Simulation	86
5.4 Summary	87
References and Further Reading	88
6. PROPORTIONAL ERROR ALLOCATION	90
6.1 Theoretical Basis	90
6.2 Practical Application	93
6.3 Empirical Results	95
6.4 Summary	101
References and Further Reading	102
7. COEFFICIENT IDENTIFICATION FOR A PUMA 560	103
7.1 Equipment	103
7.2 Source of Noise	104
7.3 Spline Smoothing of Measured Positions	110
7.4 Data Gathering Trajectories	114
7.5 Network Training	118
7.6 Results	119
7.7 Summary	127
References and Further Reading	129
8. CONCLUSION	130
8.1 Achievements	130
8.2 Further Work	132
References	135
APPENDIX A - Table Of Unique Terms	136
APPENDIX B - Terms and Values for the PUMA 560 CSLC Network ..	138
B.1 Mass Term Subnetwork	138
B.2 Velocity Term Subnetwork	145
B.3 Gravity Term Subnetwork	150
B.4 Viscous Friction Subnetwork	153
B.5 Dry Friction Subnetwork	154
B.6 Static Friction Velocity Thresholds	155
APPENDIX C - Technical Summary	156
C.1 Manipulator Inverse Dynamics	156
C.2 Other Systems	159

ABSTRACT

The values of a given manipulator's dynamics coefficients need to be accurately identified in order to employ model-based algorithms in the control of its motion. This thesis details the development of a novel form of adaptive network which is capable of accurately learning the coefficients of systems, such as manipulator inverse dynamics, where the algebraic form is known but the coefficients' values are not. Empirical motion data from a pair of PUMA 560's has been processed by the Context-Sensitive Linear Combiner (CSLC) network developed, and the coefficients of their inverse dynamics identified. The resultant precision of control is shown to be superior to that achieved from employing dynamics coefficients derived from direct measurement.

As part of the development of the CSLC network, the process of network learning is examined. This analysis reveals that current network architectures for processing analogue output systems with high input order are highly unlikely to produce solutions that are good estimates throughout the entire problem space. In contrast, the CSLC network is shown to generalise intrinsically as a result of its structure, whilst its training is greatly simplified by the presence of only one minima in the network's error hypersurface. Furthermore, a fine-tuning algorithm for network training is presented which takes advantage of the CSLC network's single adaptive layer structure and does not rely upon gradient descent of the network error hypersurface, which commonly slows the later stages of network training.

1. INTRODUCTION

1.1 STATEMENT OF THE PROBLEM

Accurate dynamic control of robotic manipulators' motion is clearly desirable for many applications, however the design of such systems is hampered by the complex manner in which forces and torques applied to a manipulator interact to influence its motion. The acceleration response of a manipulator is clearly a function of the forces and/or torques applied to it, but in general this function is nonlinear and *configuration dependent*, that is, varies with the instantaneous position and velocity of each section of the manipulator. Continuously precise control of a manipulator's motion cannot therefore be achieved by purely error driven control algorithms (such as PID), as their performance will vary greatly over the manipulator's work space and any specified steady state positions will not be achieved by sections subject to external forces such as gravity.

Model-based control algorithms allow some or all of a system's nonlinearity and configuration dependency to be compensated for, by incorporating models of these characteristics into the control algorithm. A fully compensated model-based controller can therefore provide ideal linear control of a system. Such an approach has often been proposed for controlling robotic manipulators as the form of their dynamics is relatively well understood. However, the performance of such model-based controllers is greatly affected by the degree to which the values of the coefficients within the manipulator's dynamics can be identified. These coefficients consist of grouped physical parameters such as inertias and positions of centroids, which are difficult to measure accurately and usually require the disassembly of the manipulator. Furthermore, grouping these parameters together compounds the effects of the inherent measurement errors, such that the resultant control performance is often unacceptable.

1.2 METHOD OF SOLUTION

A class of adaptive algorithms known as *adaptive networks* are employed. By using information on a given manipulator's motion to adapt a network it is possible to obtain a model of the manipulator's inverse dynamics. A novel form of network is devised, to be known as the Context Sensitive Linear Combiner (CSLC) network, where the

network's structure exactly matches the algebraic form of the system to be modelled, in this case the manipulator's inverse dynamics. This means that the parameter values within the adapted network are direct estimates of the manipulator dynamics coefficients, and that these values can be used independently of the network, in particular, within model-based control algorithms. Additionally, it will be shown that this type of network overcomes many of the traditional problems associated with adaptive networks, such as failure to obtain the optimal system model, failure of the network solution to generalise and non-transparency of operation.

The accuracy to which the manipulator dynamics coefficients will be identified by the CSLC network is expected to be largely dependent upon the accuracy of the motion data used to adapt the network. In general, motion data measurement error is much smaller than that inherent in measuring a manipulator's physical parameters directly.

1.3 RELATED WORKS

The majority of technical references are made in the main body of the thesis, after appropriate terminology and concepts have been introduced. However, in order to provide a context for the work, it is useful to review here a number of studies made by other researchers.

A review of manipulator control laws is presented by [Leahy & Saridis 1989] which examines a variety of both simple error driven algorithms and model-based controllers. Such model-based controllers, when applied to manipulator dynamics, are known as *computed torque* algorithms. Other studies into computed torque techniques include [Paul 1972, Fu et al. 1987, Luh et al. 1980, Lee et al. 1982, Neuman & Tourassis 1987]. Such algorithms are shown to be capable of providing excellent control characteristics, as shown by [An et al. 1987, Khosla & Kanada 1988, Leahy & Saridis 1987], but, in practice, have been hampered by an inability to provide them with accurate values for the dynamics coefficients. Crucially, [Whitcomb et al. 1993] demonstrates how computed torque controllers can perform less well than simple error driven algorithms when the control model is inaccurate, stating that "the degree of performance improvement afforded by all model-based algorithms is strictly limited by the accuracy of the plant model employed."

The identification of manipulator dynamics coefficients has, up until now, been attempted through direct measurement of the many physical parameters that comprise them. An appreciation of the practical difficulties, and inherent inaccuracies, involved in this task can be gained by consideration of [Corke & Armstrong-Hélouvy 1994]. This review of reported dynamics coefficients for the most well documented manipulator found in the

research community is revealingly entitled "A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot." It demonstrates how the values given by 11 different sources for the manipulator's physical parameters vary enormously: in particular, the stated inertia values for differing components vary by between 200% and 450%. Of all the identification experiments examined, perhaps the most thorough is that of [Armstrong et al. 1986]. Not only does this study provide estimates of the dynamics coefficients, but also reports expected error margins. These margins have magnitudes of up to 50% of the stated values for the more significant coefficients, and up to 400% for those of less significance.

Adaptive techniques of many types have been employed in efforts to identify various properties of manipulators' dynamics; common amongst these techniques are adaptive networks (often called *neural networks* when representing nonlinear systems). Broadly speaking, the adaptive networks applied to solving manipulator dynamics can be split into two categories; those, such as employed by [Bassi & Bekey 1989], that attempt to learn the values that are constant across the whole of the problem space, and those such as [Miller et al. 1990], that learn values of localised variables. The advantage of attempting to find good estimates to a localised model is the reduced complexity of such problems, however, such adaption must be performed on line, which often requires an abbreviation of the dynamics model due to the requirement to limit computation time. The estimation of problem-space-wide parameters, on the other hand, may be performed either on or off line, allowing it to be considered separately from the control scheme.

1.4 LAYOUT OF THIS THESIS

The motion of robotic manipulators is examined in Chapter 2. In particular, the inverse dynamics equations for a manipulator are derived, first in general, and then in respect to the chosen case study manipulator, the PUMA 560. The need for a complete model of the inverse dynamics, with accurately identified coefficients, is shown in relation to the ideally linearised computed torque control algorithm.

An introduction to adaptive networks is given in Chapter 3. The major network types and methods of adaption are discussed. The use of adaptive networks in the modelling of manipulator dynamics is also reviewed, the limited practical success obtained noted and current areas of difficulty identified.

The mechanism of adaption in networks with analogue valued outputs is analysed in Chapter 4. This often hard to visualise process is both clarified and better defined by considering the analogous operation of fitting a polynomial curve to a given set of points. The analogy gives rise to the definition of a network's order, and helps demonstrate why

conventional adaptive networks cannot be trained to model high-order systems with analogue outputs across the whole of the system's problem space.

Chapter 5 presents a novel adaptive network structure, to be known as the Context Sensitive Linear Combiner network. This type of network is shown to be capable of accurately modelling a high-order analogue system (with known form, but initially unknown coefficients) across all of its problem space. A number of other advantages over conventional network types are also discussed, and the structure a CSLC network for identifying a PUMA 560's dynamics coefficients illustrated.

In order to hasten the often laborious process of network adaption, a novel network fine-tuning algorithm is presented in Chapter 6. Based on the concept of assigning adjustments to each adaptive network component relative to its significance, the algorithm is to be known as Proportional Error Allocation (PERAL). Empirical comparisons are made between PERAL and the ubiquitous Backpropagation algorithm in the modelling of two different classes of problem.

The implementation of a CSLC network to identify the dynamics coefficients of a given manipulator is discussed in Chapter 7. The results of processing empirical motion data from a pair of PUMA 560s are presented, and the coefficients identified are shown to provide a significantly improved model of the manipulators' motion than those identified from direct measurement.

The conclusions that can be drawn from this thesis are detailed in Chapter 8. Also discussed are the avenues for further investigation that follow from this work.

For those who may wish to perform similar studies, a technical summary of the identification methodology developed in the main body of the thesis is additionally presented in Appendix C.

1.5 DECLARATION OF ORIGINALITY

The information and theory presented in chapters four to eight are the result of original research, and, to the best of the author's knowledge, does not yet appear in any other publication. In particular, the major contributions made by this thesis are as follows.

- The dissection of the network adaption process, and identification of the primary reasons why conventional network types fail to produce problem-space-wide models of high-order analogue systems.

- The derivation of a relative measure of the amount of data required to cause a network to adapt successfully.
- The definition of a novel type of adaptive network which is capable of modelling high-order analogue systems in general, and the identification of a manipulator's dynamics coefficients in particular.
- The derivation of a novel algorithm for the fine-tuning of adaptive networks.
- The identification of dynamics coefficients for the PUMA 560 manipulator, from motion data alone, which afford a superior dynamics model than that provided by coefficients obtained from direct measurement.

1.6 CONVENTIONS AND NOMENCLATURE

Throughout this work, the first use of a significant technical term or parameter within the text is denoted by the expression appearing italicised. In all such cases it is accompanied by a definition, and in the case of parameters, its algebraic representation is also noted if it is to be used in any equations.

So as to make algebraic terms clearly distinguishable, Roman alphabetical characters within a parameter name always appear italicised, however, Greek characters and characters representing algebraic terms that are not parameters (such as numerals or operators) do not. The commonly accepted algebraic notation for each parameter has been adopted wherever possible. Since this work encompasses both the fields of manipulator dynamics and adaptive networks, this has led to the occasional repetition of notation between differing parameters, however, this is restricted to parameters with clearly different usages. Although each parameter's notation is defined at the appropriate points in the main body of this work, a list of all unique algebraic parameter labels, complete with shortened definitions, is supplied in Appendix A for ease of reference. Without exception, scalar parameters are denoted by lower-case characters and matrices by uppercase characters, whilst vector parameters may be denoted by either. Note that the sometimes used convention of lowering a parameter label's case, whilst referring to a component element of that parameter, is not used. That is, the top left element of the matrix A is referred to here as A_{11} , and not as a_{11} .

Chapters, sections and subsections are labelled in numerical order and delimited by periods, such that the fifth subsection of the fourth section of the third chapter is labelled 3.4.5. Both equations and figures are independently labelled with respect to the section in which they appear, in order of appearance. Thus the fourth equation appearing in

section three of the second chapter is labelled 2.3-4 (regardless of which sub-section, if any, it appears in). Note that the term "figure" is used to refer to all plots, images, diagrams and tables.

Due to the disparate topics covered in this thesis, and for ease of reading, the lists of references and suggested further reading material are presented at the end of each chapter.

REFERENCES

- An, C.H., Atkeson, C.G., Griffiths, J.D., and Hollerbach, J.M., "Experimental evaluation of feedforward and computed torque control," *Proc. IEEE Conf. on Robotics and Automation* (Raleigh, NC, March 1987), pp. 165-168.
- Armstrong, B., Khatib, O., and Burdick, K., "The explicit dynamic model and inertial parameters of the PUMA 560 arm," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 510-518, 1986.
- Bassi, D.F., and Bekey, G.A., "Decomposition of neural network models of robot dynamics: a feasibility study," *Simulation and AI, Conf. Simulation Series*, vol. 20, no. 3, pp. 8-13, 1989.
- Corke, P.I., and Armstrong-Hélouvry, B., "A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot," *IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1608-1613, 1994
- Fu, K.S., Gonzales, R.C., and Lee, C.S.G., *Robotics: Control, Sensing, Vision and Intelligence*, Int. Ed. New York: McGraw-Hill, 1987.
- Khosla, P.K., and Kanada, T., "Experimental evaluation of nonlinear feedback and feedforward control schemes for manipulators," *Int. J. Robotics Res.*, vol. 7, no. 1, pp. 18-28, 1988.
- Leahy, M.B., Jr., and Saridis, G.N., "Compensation of industrial manipulator dynamics," RAL Technical Report no. 101. Troy, N.Y.: Rensselaer Polytechnic Institute, Dept. of Electrical, Computer, and Systems Engineering, 1987.
- Leahy, M.B., Jr., and Saridis, G.N., "Compensation of industrial manipulator dynamics," *Int. J. Robotics Res.*, vol.8, no.4, pp. 73-84, 1989.
- Lee, C.S.G., "Robot arm kinematics, dynamics and control," *IEEE Computer*, vol. 15, pp. 62-80, Dec. 1982.
- Lee, C.G.S., Chung, M.J., Mudge, T.N., and Turney, J.L., "On the control of mechanical manipulators," *Proc. 6th IFAC Symp. on Identification and System Parameter Estimation* (Washington, DC, June, 1982), pp. 1454-1459.
- Liu, M., "Puma 560 robot arm analogue servo system parameter identification," Tech. Rep. ASR-91-1, University of Melbourne, Dept. Mech and Manu. Eng., Feb. 1991.
- Lloyd, J., "Implementation of a robot control development environment," Master's thesis, Mc Gill University, Dec. 1985.
- Luh, J.Y.S., Walker, M.W., and Paul, R.P., "Resolved-acceleration control of mechanical Manipulators," *IEEE Trans. Automat. Control*, vol. AC-25, pp. 468-474, 1980.

- Miller, W.T., Hewes, R.P., Glanz, F.H., and Kraft, L.G., "Real-time dynamic control of an industrial manipulator using a neural-network based learning controller," *IEEE Trans. Robotics and Automation*, vol. 6, no. 1, pp. 1-9, 1990.
- Neuman, C.P., and Tourassis, V.D., "Robust discrete nonlinear feedback control for robotic manipulators," *J. Robotic Syst.*, vol. 4, pp. 115-143, Feb. 1987.
- Paden, B., and Panja, R., "Globally asymptotically stable PD+ controller for robot manipulators," *Int. J. Control*, vol. 47, no. 6, pp. 1696-1712, 1988.
- Paden, B., and Riedle, R., "A positive-real modification of a class of nonlinear controllers for robot manipulators," *Proc. Amer. Control Conf.*, Atlanta, GA, pp. 1782-1785, 1988.
- Paul, R.P., "Modelling, trajectory calculation and servoing of a computer controlled arm," Stanford Artificial Intelligence Lab. Memo AM-177, Nov. 1972.
- Paul, R., Rong, M., and Zhang, H., "Dynamics of puma manipulator," *American Control Conference*, 1983.
- Paul, R.P., and Zhang, H., "Computationally efficient kinematics for manipulators with spherical wrists," *Int. Journal of Robotics Research*, vol. 5, no. 2, 1986.
- Sadegh, N., and Horowitz, R., "Stability and robustness analysis of a class of adaptive controllers for robotic manipulators," *Int. J. Robotics Res.*, vol. 9, no. 3, pp. 74-92, 1990.
- Tarn, T.J., Bejczy, A.K., Han, S., and Yun, X., "Inertia parameters of puma 560 robot arm," Tech. Rep. SSM-RL-85-01, Washington Uni., St. Louis, MO., Sept. 1985.
- Wen, J.T., and Bayard, D.S., "New class of control laws for robotic manipulators," *Int. J. Control*, vol. 47, no. 5, pp. 1361-1406, 1988.
- Whitcomb, L.L., Rizzi, A.A., and Kodischek, D.E., "Comparative experiments with a new adaptive controller for robot arms," *IEEE Trans. Robotics and Automation*, vol. 9, no. 1, pp. 59-69, 1993.

2. THE MOTION OF ROBOTIC MANIPULATORS

Given the desire for model-based control of manipulators, a representation of manipulator motion is required. Several aspects of manipulator motion are discussed in this chapter, most especially, the derivation of the equations of motion for the general rigid bodied case. Phenomena that are not modelled in the ideal case, such as friction and mechanical imperfections, are also considered, and compensation for their effects incorporated into the general model of a manipulator's inverse dynamics. Examination of this generalised inverse dynamics model provides valuable insights into the problem of identifying a given manipulator's dynamics coefficients.

Section 2.6 covers the derivation of dynamics components specific to a particular manipulator, using the PUMA 560 as its example. In particular, it is shown how the definition of a suitable algebraic representation of the PUMA 560's geometry allows the manipulator's inverse dynamics coefficients to be expressed solely in terms of physical parameters such as lengths and masses. Section 2.7 provides a context for the coefficient identification problem by describing the fully compensated computed torque control law, and discusses the performance of such algorithms compared to a non model-based controllers.

2.1 THE STRUCTURE OF A MANIPULATOR

For the purposes of analysis, a manipulator can be considered to be a set of *links* (or limbs), interconnected by *joints*. Although there are many possible designs of joint, they can all be thought of as being made up of just two fundamental types, namely *revolute*, with a single rotational degree of freedom, and *prismatic*, with a single translational degree of freedom. Likewise, the links ascribed to a given manipulator need not resemble precisely its physical components, as long as one link and one joint are associated with each degree of freedom of the manipulator, and the system so defined is capable of all configurations possible by the physical manipulator. For example, a ball-and-socket joint can be considered to be two revolute joints, with different axes of rotation, interconnected by a link of zero length.

2.2 MANIPULATOR DYNAMICS

Robotic manipulators are, in general, constructed to operate as rigid bodied machines to within engineering accuracy, that is, the shape of each manipulator link does not vary significantly. The rare exceptions to this include the NASA Space Shuttle manipulator arm, where minimising mass was considered more desirable than simplifying the control, such that the manipulator links are prone to significant flexing [Lowe et al. 1995]. However, the over-whelming majority of manipulators can be modelled satisfactorily using rigid body mechanics, which is the case considered here.

There exist two principal methods for analysing the dynamics of rigid bodied robotic manipulators, known as the *Lagrangian* and the *Newton-Euler* formulations. These formulations result in generalised expressions for the equations of motion that are quite different in form, but, when evaluated for a given manipulator, produce exactly the same algebraic solutions. In the Lagrangian formulation, the manipulator is treated as a whole and the analysis performed using a Lagrangian function (the difference between the kinetic energy and the potential energy). In contrast, in the Newton-Euler formulation, the dynamics of each link of the robot is described in turn, and a so-called forward-backward recursion is then performed which combines the dynamics of all the links, and leads to a description of the manipulator as a whole.

Both formulations are regarded as having their own distinct merits, additionally, a variety of expressions for the equations of motion, each with different properties, can be obtained from each formulation. The Newton-Euler formulation results in forms of the generalised inverse dynamics that require significantly fewer numerical operations to evaluate than those obtainable from the Lagrangian formulation. This can be an important consideration for the real-time control of a manipulator, however, in practise the greatest computational efficiency is usually obtained by expressing the inverse dynamics explicitly in manipulator specific terms and combining constant terms that factor common variables [Armstrong et al. 1986]. The Lagrangian formulation is commonly used to produce a closed form differential equation representation of the generalised inverse dynamics, where the contributions to the generalised force from the separate mechanical effects acting on a manipulator are clearly identifiable. This explicitly provides a state equation for a manipulator's dynamics and enhances ease of comprehension and analysis compared with the generalised inverse dynamics obtained from the Newton-Euler formulation, which consist of a set of recursive equations with less distinct structures and containing cross-product operators.

Although the consideration of both formulations may provide additional insight into manipulator dynamics, such detail is beyond the scope of this work. Due to its results being easier to understand, analyse and convert into control algorithms, the Lagrangian

formulation will be the method explained within this thesis. For those wishing to study the equations of motion obtainable from the Newton-Euler formulation, some of the first work in this area was done by [Armstrong 1979, Orin et al. 1979], whilst an even more computation efficient variant was produced by [Luh et al. 1980] by stating quantities with respect to their local links. Variants obtainable from the Lagrange formulation, but which are not discussed here include the generalised d'Alembert equations of motion [Lee et al. 1983], and the recursive Lagrange equations derived by [Hollerbach 1980]. For discussions of the equivalence of the two formulations see [Turney 1980, Silver 1982].

2.3 LAGRANGE'S EQUATION OF MOTION

The governing dynamic equations of motion for a manipulator comprising of rigid links will now be defined, in the general case, starting with the *Lagrange equations of motion of the second kind* for a conservative system [Lagrange 1965, Marion 1965, Whittaker 1944]. These are a set of differential equations that can describe the time evolution of any system which is subject to holonomic constraints that satisfy the principle of virtual work. A system of material points is said to be holonomic if the position of the points are either not constrained, or constrained by a function of the system coordinates and time [Rutherford 1957]. The kinetic and potential energies of a manipulator will then be derived and substituted back into the Lagrange equations to form what is known as the *Euler-Lagrange* equations [Uicker 1965, Bejczy 1974, Lewis 1974].

If there exists a vector of generalised coordinates, q , and the corresponding vector of generalised forces/torques is defined as τ , then the Lagrange equations can be written in the form

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau \quad (2.3-1)$$

where L is called the *Lagrangian*, and is equal to $K - P$, where K is the kinetic energy, and P is the potential energy, within the system. In a manipulator's case, q will be the variable joint position vector, consisting of angles in the case of revolute joints, and offsets in the case of prismatic joints. Its first derivative, \dot{q} , is termed the generalised joint velocity vector, and the second derivative, \ddot{q} , is the generalised joint acceleration vector. Thus τ will be a vector containing torque elements corresponding to the revolute joints, and force elements corresponding to the prismatic joints.

2.3.1 Reference Coordinate Frames

Before developing the terms of the Lagrange equation, it is necessary to first define the coordinate system that will be employed. For any point expressed in the coordinate frame i as

$${}^i r = [x \ y \ z \ 1]^T \quad (2.3-2)$$

its position expressed in a different frame j can be found from

$${}^j r = {}^j T_i {}^i r \quad (2.3-3)$$

where ${}^j T_i$ is a 4x4 matrix known as a *homogeneous transform* [Denavit & Hartenberg 1955], of the form

$${}^j T_i = \begin{bmatrix} {}^j R_i & {}^j p_i \\ 0 & 1 \end{bmatrix} \quad (2.3-4)$$

where ${}^j R_i$ is a 3x3 *rotation matrix*, ${}^j p_i$ is a *translation vector* and the 1 represents a *scaling factor*. A homogeneous transform can be thought of as both a transformation that takes the representation of a vector in frame i to its representation in frame j , and as a description of frame i in terms of frame j . ${}^j R_i$ describes the orientation of the axes of frame i in terms of frame j , and ${}^j p_i$ describes the origin of frame i in terms of the coordinates of frame j .

One important property of homogeneous transforms, that will be used later, is

$${}^k T_i = {}^k T_j {}^j T_i \quad (2.3-5)$$

By convention, the *base* or *world* coordinate frame is denoted as frame 0; thus a homogeneous transform to the base frame is written with either a leading 0 superscript or, as is adopted here, no leading superscript at all. For a more detailed discussion of spatial descriptions and transformations see [Craig 1986, Fu et al. 1987].

2.3.2 Manipulator Kinetic Energy

Given a point on link i with coordinates of ${}^i r$ (with respect to a frame i), then r , the coordinates of that point with respect to the base frame, can be found from

$$r = T_i {}^i r \quad (2.3-6)$$

Note that T_i is a function of q_1, q_2, \dots, q_i . Consequently, the velocity v of the point expressed in base coordinates is

$$v = \frac{dr}{dt} = \sum_{j=1}^i \left(\frac{\partial T_i}{\partial q_j} \dot{q}_j \right) {}^i r \quad (2.3-7)$$

Also, as

$$\frac{\partial T_i}{\partial q_j} = 0 \quad \text{for } j > i$$

the upper summation limit in Eqn. 2.3-7 can be replaced by n , the number of links.

Now, the kinetic energy in terms of a stationary base frame of an infinitesimal mass, dm , moving with velocity \mathbf{v} , can be expressed as

$$dK_i = \frac{1}{2} \mathbf{v}^T \mathbf{v} \, dm = \frac{1}{2} \text{Tr}(\mathbf{v} \mathbf{v}^T) \, dm \quad (2.3-8)$$

where $\text{Tr}(\cdot)$ denotes the *trace* of a matrix (the sum of the leading diagonal). Substituting in Eqn. 2.3-7, and taking advantage of the matrix identity $A^T B^T \equiv (BA)^T$, the following is obtained

$$dK_i = \frac{1}{2} \text{Tr} \left(\sum_{j=1}^n \sum_{k=1}^n \frac{\partial T_i}{\partial q_j} ({}^i r \, {}^i r^T \, dm) \frac{\partial T_i^T}{\partial q_k} \dot{q}_j \dot{q}_k \right) \quad (2.3-9)$$

Clearly, the total kinetic energy for link i is given by

$$K_i = \int_{\text{link } i} dK_i \quad (2.3-10)$$

Substituting Eqn. 2.3-9 into the above, the integral may be taken inside the summations. Thus, defining the 4×4 *pseudo-inertia matrix* for link i as

$$I_i \equiv \int_{\text{link } i} {}^i r \, {}^i r^T \, dm \quad (2.3-11)$$

the kinetic energy equation for link i can be expressed as

$$K_i = \frac{1}{2} \text{Tr} \left(\sum_{j=1}^n \sum_{k=1}^n \frac{\partial T_i}{\partial q_j} I_i \frac{\partial T_i^T}{\partial q_k} \dot{q}_j \dot{q}_k \right) \quad (2.3-12)$$

Before proceeding with the definition of the total manipulator kinetic energy, it is useful to examine the pseudo-inertia matrix. Expanding Eqn. 2.3-11 yields

$$I_i = \begin{pmatrix} \int x^2 \, dm & \int yx \, dm & \int zx \, dm & \int x \, dm \\ \int xy \, dm & \int y^2 \, dm & \int zy \, dm & \int y \, dm \\ \int xz \, dm & \int yz \, dm & \int z^2 \, dm & \int z \, dm \\ \int x \, dm & \int y \, dm & \int z \, dm & \int dm \end{pmatrix} \Big|_{\text{link } i} \quad (2.3-13)$$

Where x , y , z are the Cartesian coordinates of the infinitesimal mass dm , with respect to frame i , and the integrals are taken over the volume of link i . If frame i is attached to link i , and link i is rigid, then I_i is a constant matrix dependent only on the geometry and mass distribution of the link. In other words, it is dependent upon the *mass moments of*

inertia

$$\begin{aligned} {}^i I_{xx} &= \int (y^2 + z^2) dm \\ {}^i I_{yy} &= \int (x^2 + z^2) dm \\ {}^i I_{zz} &= \int (x^2 + y^2) dm \end{aligned} \quad (2.3-14)$$

the *mass cross-products of inertia*

$$\begin{aligned} {}^i I_{xy} &= \int xy dm \\ {}^i I_{xz} &= \int xz dm \\ {}^i I_{yz} &= \int yz dm \end{aligned} \quad (2.3-15)$$

and the *first moments*

$$\begin{aligned} m_i {}^i r_x &= \int x dm \\ m_i {}^i r_y &= \int y dm \\ m_i {}^i r_z &= \int z dm \end{aligned} \quad (2.3-16)$$

where m_i is the total mass of link i and $({}^i r_x \ {}^i r_y \ {}^i r_z \ 1)^T$ are the coordinates, in frame i , of its centre of gravity.

Thus, rewriting I_i in terms of these quantities produces

$$I_i = \begin{pmatrix} \frac{-{}^i I_{xx} + {}^i I_{yy} + {}^i I_{zz}}{2} & {}^i I_{xy} & {}^i I_{xz} & m_i {}^i r_x \\ {}^i I_{xy} & \frac{{}^i I_{xx} - {}^i I_{yy} + {}^i I_{zz}}{2} & {}^i I_{yz} & m_i {}^i r_y \\ {}^i I_{xz} & {}^i I_{yz} & \frac{{}^i I_{xx} + {}^i I_{yy} - {}^i I_{zz}}{2} & m_i {}^i r_z \\ m_i {}^i r_x & m_i {}^i r_y & m_i {}^i r_z & m_i \end{pmatrix} \quad (2.3-17)$$

In general, given a reference frame whose origin is fixed with respect to a body, the axes of the frame can be orientated so that all the body's mass products of inertia are zero [Shames 1997]. In this case, the axes of the frame are known as the body's *principal axes*, and the mass moments of inertia the *principal moments*, for that reference frame origin.

Now consider the parallel axis theorem [Shames 1997], which states that for any given body of mass m

$${}^a I_{zz} = {}^c I_{zz} + m({}^a r_x^2 + {}^a r_y^2) \quad (2.3-18)$$

$${}^a I_{xy} = {}^c I_{xy} + m {}^a r_x {}^a r_y \quad (2.3-19)$$

where the frame c is located at the centre of gravity, and a denotes an arbitrarily translated frame. (Note: Eqn. 2.3-19 is incorrectly stated in [Craig 1986], though correct

in the earlier edition of that book).

Making use of both the principal axes and parallel axis theorems, I_i can be restated, for any coordinate frame whose axes are parallel to the principal axes which lie at the centre of gravity of link i , as

$$I_i = \begin{pmatrix} \frac{-{}^pI_{xx} + {}^pI_{yy} + {}^pI_{zz}}{2} + m_i {}^i r_x^2 & m_i {}^i r_x {}^i r_y & m_i {}^i r_x {}^i r_z & m_i {}^i r_x \\ m_i {}^i r_x {}^i r_y & \frac{{}^pI_{xx} - {}^pI_{yy} + {}^pI_{zz}}{2} + m_i {}^i r_y^2 & m_i {}^i r_y {}^i r_z & m_i {}^i r_y \\ m_i {}^i r_x {}^i r_z & m_i {}^i r_y {}^i r_z & \frac{{}^pI_{xx} + {}^pI_{yy} - {}^pI_{zz}}{2} + m_i {}^i r_z^2 & m_i {}^i r_z \\ m_i {}^i r_x & m_i {}^i r_y & m_i {}^i r_z & m_i \end{pmatrix} \quad (2.3-20)$$

where

${}^pI_{xx}, {}^pI_{yy}, {}^pI_{zz}$ = the principal moments of inertia about link i 's centre of gravity

Although inertia terms cannot be measured directly, the mass moments (but not products) of inertia around a link's centre of gravity can be crudely estimated from measurements of the detached limb, using a device known as a inertia pendulum [Armstrong et al. 1986].

Returning to the development of the Euler-Lagrange equation, the total manipulator kinetic energy may now be written as

$$K = \sum_{i=1}^n K_i = \frac{1}{2} \sum_{i=1}^n \text{Tr} \left(\sum_{j=1}^n \sum_{k=1}^n \frac{\partial T_i}{\partial q_j} I_i \frac{\partial T_i^T}{\partial q_k} \dot{q}_j \dot{q}_k \right) \quad (2.3-21)$$

Since the trace of a sum of matrices is equal to the sum of their individual traces, the summations may be interchanged with the trace operator to obtain

$$K = \frac{1}{2} \sum_{j=1}^n \sum_{k=1}^n M_{jk} \dot{q}_j \dot{q}_k \quad (2.3-22)$$

or, in matrix terminology

$$K = \frac{1}{2} \dot{q}^T M(q) \dot{q}$$

where the $n \times n$ mass matrix, $M(q)$, (also known as the inertia matrix, dynamic inertia matrix, or kinetic energy matrix) is defined as

$$M_{jk} \equiv \sum_{i=1}^n \text{Tr} \left(\frac{\partial T_i}{\partial q_j} I_i \frac{\partial T_i^T}{\partial q_k} \right) \quad (2.3-23)$$

Since

$$\frac{\partial T_i}{\partial q_j} = 0 \quad \text{for } j > i$$

the mass matrix may be defined more efficiently as

$$M_{jk} \equiv \sum_{i=\max(j,k)}^n \text{Tr} \left(\frac{\partial T_i}{\partial q_j} I_i \frac{\partial T_i^T}{\partial q_k} \right) \quad (2.3-24)$$

Furthermore, since performing the product of three $n \times n$ matrices and taking the trace of the result is computationally very inefficient, one can instead take advantage of the identity

$$\text{Tr}(ABC) \equiv \sum_{h=1}^n \sum_{g=1}^n \sum_{f=1}^n A_{hf} B_{fg} C_{gh}$$

As $M_{jk} = M_{kj}$, the mass matrix is *symmetric*. Since kinetic energy is positive, vanishing only when the generalised velocity \dot{q} is equal to zero, the mass matrix is also *positive definite*.

We now have a convenient expression for the manipulator kinetic energy in terms of physical parameters and the joint variables q and \dot{q} .

2.3.3 Manipulator Potential Energy

If link i has a mass m_i and a centre of gravity at ${}^i r_c$, then the potential energy of the link is given by

$$P_i = -m_i g_{vec}^T T_i {}^i r_c \quad (2.3-25)$$

where g_{vec} is the *gravity vector* (in base coordinates) defined as

$$g_{vec}^T = (g_x \ g_y \ g_z \ 0) \quad (2.3-26)$$

If, as is normally the case, the manipulator's base coordinate frame's z-axis is aligned vertically upward, then

$$g_{vec}^T = (0 \ 0 \ -g \ 0)$$

where g is the scalar value of the local acceleration due to gravity. For example, if the manipulator is situated at sea level then $g = 9.8062 \text{ m/s}^2$.

Therefore, the total manipulator potential energy term is

$$P = -\sum_{i=1}^n m_i g_{vec}^T T_i {}^i r_c \quad (2.3-27)$$

Note that P is dependent on the joint positions, q , only. Noting also that $m_i {}^i r_c$ is the last

column of the pseudo-inertia matrix I_i , it is possible to rewrite the above equation as

$$P = -\sum_{i=1}^n g_{vec}^T T_i(q) I_i l_4 \quad (2.3-28)$$

where l_4 is the last column of the 4×4 identity matrix, that is $l_4 = (0 \ 0 \ 0 \ 1)^T$.

2.4 THE EULER-LAGRANGE EQUATION

The manipulator Lagrangian can be expressed as

$$L(q, \dot{q}) = K(q, \dot{q}) - P(q) = \frac{1}{2} \dot{q}^T M(q) \dot{q} - P(q) \quad (2.4-1)$$

Note that the kinetic energy is a quadratic function of the joint velocity vector and the potential energy is independent of \dot{q} .

The terms within Lagrange's equation (Eqn. 2.3-1) are as follows.

$$\begin{aligned} \frac{\partial L}{\partial \dot{q}} &= \frac{\partial K}{\partial \dot{q}} = M(q) \dot{q} \\ \frac{d}{dt} \frac{\partial L}{\partial \dot{q}} &= \frac{\partial K}{\partial \dot{q}} = M(q) \ddot{q} + \dot{M}(q) \dot{q} \\ \frac{\partial L}{\partial q} &= \frac{1}{2} \frac{\partial}{\partial q} (\dot{q}^T M(q) \dot{q}) - \frac{\partial P(q)}{\partial q} \end{aligned}$$

Therefore the Euler-Lagrange equation for a manipulator's dynamics is

$$M(q) \ddot{q} + \dot{M}(q) \dot{q} - \frac{1}{2} \frac{\partial}{\partial q} (\dot{q}^T M(q) \dot{q}) + \frac{\partial P(q)}{\partial q} = \tau \quad (2.4-2)$$

This form of the equation presents a difficulty: the differentiation of a matrix, namely $M(q)$, by a vector, q . Such derivatives are not matrices, but *tensors* of order three—that is, they have three indices, not two. This presents no problem to those familiar with tensor mathematics, however those in the field of robotics have tended to adopt matrix devices such as *Kronecker Product Analysis* [Brewer 1978] instead. It is possible to avoid all such conceptual difficulties by breaking down the matrix operations involved into scalar mathematics. Thus, combining the second and third terms of the above equation and defining them as V , the *velocity term*, they can be expressed in scalar form, for link i , as

$$V_i = \sum_k^n \sum_j^n \left(\frac{\partial M_{ik}}{\partial q_j} - \frac{1}{2} \frac{\partial M_{jk}}{\partial q_i} \right) \dot{q}_j \dot{q}_k \quad (2.4-3)$$

Now, taking advantage of the mass matrix's symmetry this can be rewritten as

$$V_i = \sum_k^n \sum_j^n \frac{1}{2} \left(\frac{\partial M_{ik}}{\partial q_j} + \frac{\partial M_{ij}}{\partial q_k} - \frac{\partial M_{jk}}{\partial q_i} \right) \dot{q}_j \dot{q}_k \quad (2.4-4)$$

Thus the *Christoffel symbols (of the first kind)* [Corben & Stehle 1950, Borisenko & Tarapov 1968] can be defined as

$$c_{ijk} \equiv \frac{1}{2} \left(\frac{\partial M_{ik}}{\partial q_j} + \frac{\partial M_{ij}}{\partial q_k} - \frac{\partial M_{jk}}{\partial q_i} \right) \quad (2.4-5)$$

Given that the mass matrix is symmetrical, it is clear that $c_{ijk} = c_{ikj}$. Further computational efficiency may be gained by noting that

$$\frac{\partial M_{ij}}{\partial q_k} = 0 \quad \text{for} \quad i \geq k \leq j$$

which in turn implies that for any i, j & k , one term in Eqn. 2.4-5 is equal to zero, and that $c_{ijk} = 0$ for $i = k \geq j$.

The velocity term of the Euler-Lagrange equation can now be expressed as

$$V_i(q, \dot{q}) = \sum_k^n \sum_j^n c_{ijk} \dot{q}_j \dot{q}_k \quad (2.4-6)$$

The terms in the above equation can be classified into two types. Those involving a product of the type \dot{q}_i^2 , which represent *centrifugal* effects, and those involving a product of the type $\dot{q}_i \dot{q}_j$ where $i \neq j$, which represent the *Coriolis* effects [Koditschek 1984, Gu & Loh 1988].

Finally, if the *gravity effect vector* (also known as simply the *gravity term*), G , is defined as

$$G(q) = \frac{\partial P(q)}{\partial q} \quad (2.4-7)$$

then the Euler-Lagrange equation may be rewritten thus

$$M(q)\ddot{q} + V(q, \dot{q}) + G(q) = \tau \quad (2.4-8)$$

2.5 NON-LAGRANGIAN DYNAMICS

So far the Euler-Lagrange equation derived includes terms for inertia, velocity and gravity effects. However, the dynamics of a real manipulator are also significantly affected by several further effects:

2.5.1 Friction Effects

All practical mechanisms are, of course, affected by frictional forces and torques. Although friction is relatively complex to model, it is at least decoupled between joints, that is the frictional effects acting upon a joint are dependent only upon the properties of

that joint. From empirical observation it is apparent that the friction experienced by a manipulator's joint is almost entirely dependent on the joint's velocity [Leahy 1985]. Friction can also be affected by joint position due to, for instance, gears which are not perfectly circular. However, for most manipulators, including the PUMA 560 [Leahy & Saridis 1989], the frictional effects can be acceptably modelled for engineering purposes as a function of the joint velocity alone.

The majority of a friction profile can usually be modelled by *viscous friction*, in which the frictional effect is proportional to the joint velocity. Thus if the *friction effect vector* is denoted as F , then the friction model can be defined as

$$F_i = k_i^v \dot{q}_i \quad (2.5-1)$$

where k^v is the viscous friction coefficient vector, and the Euler-Lagrange equation is extended to include friction effects, such that

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) + F(\dot{q}) = \tau \quad (2.5-2)$$

A refinement to the friction model is the inclusion of *dry friction* (sometimes called *Coulomb friction* or *dynamic friction*)[Smith & Smith 1935], which allows for offsets in the value of F_i , with respect to \dot{q}_i . Dry friction's magnitude is constant, and opposes the motion of the joint. Hence the friction model becomes

$$F_i = k_i^v \dot{q}_i + k_i^d \text{sgn}(\dot{q}_i) \quad (2.5-3)$$

where k^d is the dry friction coefficient vector, and $\text{sgn}(\cdot)$ denotes the signum or sign function.

The above combination models the friction profile of most manipulators adequately for engineering purposes, except when \dot{q}_i is close to zero, where the resistance to movement can be much higher than predicted. To model this region, a third component known as *static friction* is defined [Schilling 1990], so that the total frictional effect is given by

$$F_i = k_i^v \dot{q}_i + \text{sgn}(\dot{q}_i) \left[k_i^d + (k_i^s - k_i^d) \exp\left(\frac{-|\dot{q}_i|}{\gamma_i}\right) \right] \quad (2.5-4)$$

where k^s is the static friction magnitude coefficient vector, and γ is the static friction slope coefficient vector.

Thus the three part friction model described by Eqn. 2.5-4 results in the profile shown in Fig. 2.5-1.

Commonly, friction in manipulators is modelled by either the viscous component alone, or by the combination of viscous and dry friction terms. Although the significance of static friction effects varies depending on the type of joint in question, one of the main

reasons that a static friction term is commonly not included in the friction model is it that many manipulators do not have velocity sensors. This requires \dot{q} to be derived by numerical differentiation of position measurements, which is inherently inaccurate. Thus when \dot{q}_i is close to zero (the only time when the added static friction effect is not vanishingly small), the sign of \dot{q}_i cannot be known with confidence.

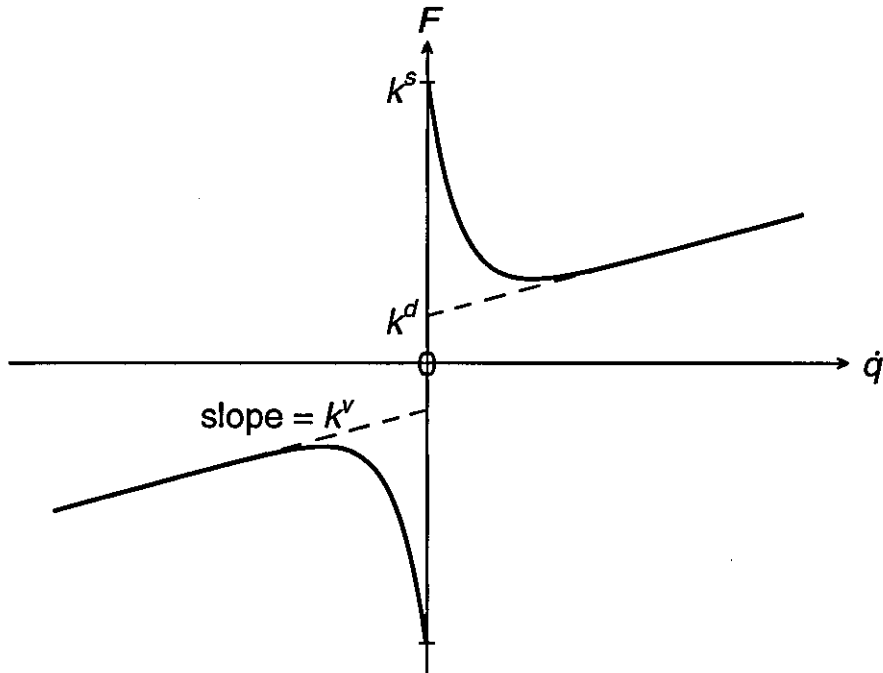


Figure 2.5-1: The friction profile produced by the three part model in Eqn. 2.5-4
(Subscripts omitted: all terms refer to a single given joint)

Note that the use of dry and/or static friction components in the friction model introduces discontinuities into the otherwise continuous Euler-Lagrange equation.

2.5.2 Actuator Dynamics

The derivation of the inverse dynamics performed so far has assumed that a manipulator is constructed solely from a set of solid links. However, manipulators need actuators to move them; these are generally either electrically or hydraulically driven. In either case, it can be shown [de Silva 1989] that the actuator effects are decoupled. Therefore, the friction acting upon the actuators is indistinguishable from the friction acting upon the joints, and the friction coefficient vectors can be chosen to encompass both sources. Furthermore, the motion of the actuators does not affect the gravity term, consequently the gravity model is unchanged as long as the values for link masses and centres of mass take into account the masses of the actuators.

In the case of electric motors, when a given link moves, its actuator's axis of rotation is

usually in motion relative to the coordinate frame attached to that link. This motion gives rise to a gyroscopic couple. However, it can be shown [Murphy & Wen 1993] that such gyroscopic forces have negligible effect upon the manipulator's links, although they can have significant effects on a manipulator's platform if it is free to move. Therefore, gyroscopic effects can be safely neglected except in special cases such as when the manipulator in question is mounted on a space vehicle.

However, the torque or force acting upon a link must overcome the resistance to motion caused by the mass of the moving parts of the attached actuator. In the case of a motor, this resistance is closely related to its mass moment of inertia measured around the axis of rotation. For a piston, it is closely related to the mass. The effect of these quantities at each joint is dependent on the transmission between the joints and the actuators, however these effective inertias/masses are constant for all common transmission devices. For example, taking the most common arrangement of a motor, embedded in the preceding link and driving a revolute joint via a simple gear train; if the gear ratio is denoted r (such that the joint speed is r times the motor speed), then

$${}^mI = \frac{I_{kk}}{r^2} \quad (2.5-5)$$

where

mI = the effective inertia of the motor acting at the joint

I_{kk} = the mass moment of inertia of the motor about its axis

As the effect of actuators on the mass term does not vary with q , then by the definition of the velocity term (Eqn. 2.4-3), there is no contribution to the velocity term due to the presence of actuators.

Consequently, the Euler-Lagrange equation can be employed as stated in Eqn. 2.5-2, but with the mass matrix redefined as

$$M_{ij} = \begin{cases} M'_{ij} + {}^mI_i & \text{for } i=j \\ M'_{ij} & \text{otherwise} \end{cases} \quad (2.5-6)$$

where M' is the mass matrix excluding actuator effects (Eqn. 2.3-24), and mI is the constant and generalised *effective motor inertia vector*.

2.5.3 Compliance and Backlash

Power transmission devices include gears, power screws, pulley systems, chain drives and harmonic drives [Groover et al. 1986, Stadler 1995]. All of these are prone to two unwanted effects, namely *compliance* and *backlash*.

Compliance is defined as the deflection under load of the power transmission device, essentially the degree of spring in the transmission. For example, in a gear train this might be due to the bending of the individual teeth, or due to deflections of the bearing supporting the gears. The compliance of a manipulator as a whole is mainly the result of transmission compliance. Compliance is not commonly incorporated into models of manipulators' dynamics, as its effects are usually small in comparison to those of other sources of system disturbance. Furthermore, compensation for compliance would hugely increase the complexity of the inverse dynamics model as it varies with time and so cannot be calculated for a given instant without knowledge of its prior value [Shames 1997].

Backlash represents the free movement within the transmission. In gears, for example, this is can be due to spaces between the gear teeth mesh, as shown below in Fig. 2.5-2.

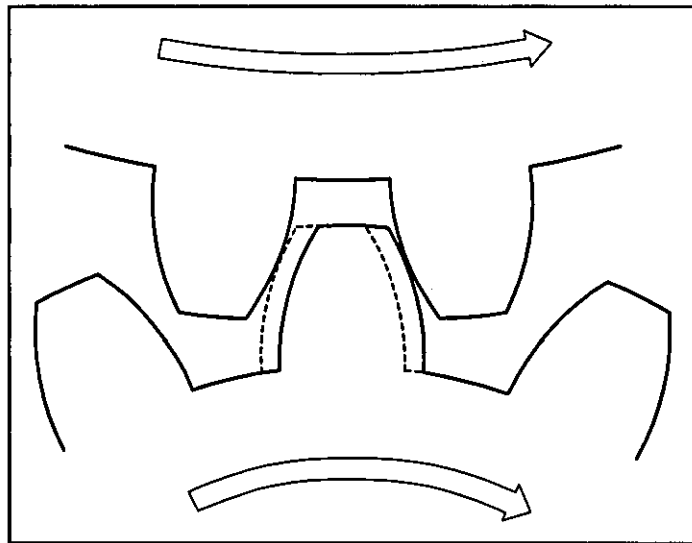
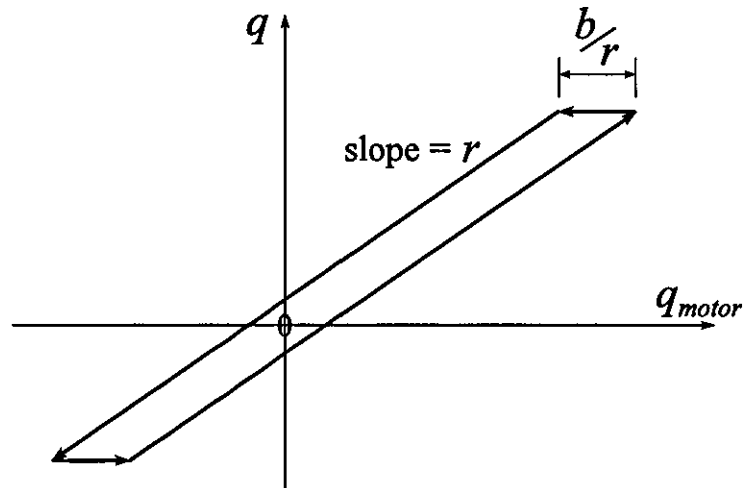


Figure 2.5-2: Backlash shown at a gear interface
(the lower gear is driving the upper gear)

Here the lower gear is driving the upper gear in the direction shown by the arrows, if the motion of the lower gear were to change direction, the tooth currently driving the upper gear would move from its present position to the position shown by the dotted lines, whilst the upper gear free-wheels. The *degree of backlash* for joint i , denoted b_i , is the maximum angle or distance through which joint i can be moved freely, whilst its actuator remains stationary.

All transmission devices display backlash effects to some degree, giving rise to hysteresis in the drive system. In hydraulic systems, for instance, backlash relates to compression of the transmission fluid. An absquare hysteresis profile is shown in Fig. 2.5-3 which describes a transmission with idealised backlash effects and zero compliance. The acute corners of the hysteresis loop represent occasions when the actuator velocity has changed

direction. Whilst the transmission system is described by the horizontal sections of the loop, the joint is said to be experiencing backlash. In reality, the hysteresis loop of a manipulator's transmission system is rarely as neat as the trapezium in Fig. 2.5-3 due to such occurrences as gears rebounding from each other at the end of a backlash period and mechanical compliance. However, consideration of Figs. 2.5-2 and 2.5-3 do allow one to appreciate the effects backlash will have on manipulator dynamics.



*Figure 2.5-3: An absquare hysteresis profile caused by backlash
(Subscripts omitted: all terms refer to a single joint,
arrows show direction of path around hysteresis loop)*

If the value of q_i is derived from measurement of the actuator position, as is often the case, then it is clear that q_i suffers a disturbance that can be of either sign and of up to b_i in magnitude depending upon the conditions prevailing when the joint position was aligned with the actuator position. Although this disturbance will be either of two values (whose magnitudes total b_i) whilst the transmission can be said to be “tight”, it will vary smoothly from one to the other whilst the joint is experiencing backlash. Furthermore, if the joint velocity and/or acceleration is computed from measurements of the actuator position, then these values will also contain disturbances whilst the joint is experiencing backlash. However, more serious is the fact that during a backlash period the actuator is detached from the link, thus the torque applied to a link by its actuator will be zero (in the absence of compliance), the geartrain contribution to the mass matrix will decrease (possibly to zero, depending on which gear interface in the train is experiencing backlash) and the friction coefficients will also lessen. In effect, to comprehensively model a manipulator's motion one requires two sets of values for many of the dynamics' coefficients, as well as an accurate method for detecting when a joint is experiencing backlash.

To date, to the best of the author's knowledge, no published manipulator control

algorithm has attempted to compensate for backlash through modelling the resultant dynamics. There are two main reasons for this; Firstly, it adds a large degree of complexity to the controller to compensate for a relatively short lived effect. Secondly, there exist techniques and devices, such as harmonic drives and transmissions with high (unity) gear ratios, for minimising the effects of backlash [Dagalakis & Myers 1985].

However, even if backlash effects are not modelled it is still extremely important to identify if joints are experiencing backlash when employing an adaptive coefficient identification system, so that the backlash periods can be neglected for the purposes of adaption such that they do not affect the resultant estimates. To this end, if one considers the transmission for joint i as being a single interface, it is possible to describe three conditions when backlash will not occur;

- 1) $\text{sgn}(\tau_i) \neq \text{sgn}(\dot{q}_i)$
- 2) $\text{sgn}(\tau_i) \neq \text{sgn}(\ddot{q}_i)$
- 3) $\text{sgn}(\tau_i + {}^mF_i) = \text{sgn}(V_i + G_i + {}^lF_i)$

where

- mF_i = Frictional effects of the motor or other actuator
- lF_i = Frictional effects of the link's joint

Condition 3 is difficult to verify in practice, so a convenient logical statement that includes at least all the occurrences of backlash at joint i is;

$$\text{sgn}(\tau_i) = \text{sgn}(\dot{q}_i) \text{ AND } \text{sgn}(\tau_i) = \text{sgn}(\ddot{q}_i) \quad (2.5-7)$$

2.6 THE INVERSE DYNAMICS OF THE PUMA 560

In Sections 2.3 through to 2.5, the equations derived for manipulators' inverse dynamics are in terms of the general case. At various points within this work it will be desirable to examine processes associated with manipulator dynamics with respect to the detailed Euler-Lagrange equations for a specific manipulator. In all such cases the manipulator chosen is the PUMA 560. This section describes the PUMA 560 and the theory required to derive its specific algebraic expressions for the components of the full Euler-Lagrange equation (Eqn. 2.5-2).

The PUMA[®] (Programmable Universal Machine for Assembly) series of industrial manipulators was designed by Vic Schienman and manufactured for many years by Unimation Ltd, before that company was bought Westinghouse, and then by Stäubli International AG, whereupon they were marketed under the composite name [Stäubli-Unimation]. Although some spares and servicing are still available, the PUMA 560 is no

longer produced, however, it is still the most common manipulator to be found in robotics research laboratories, due, in part, to its relative flexibility of operation compared to most other industrial manipulators. Because of its ubiquity, there exists probably more published attempts at identifying the values of its dynamics coefficients than for any other manipulator (for example [Paul et al. 1983, Tarn et al. 1985, Armstrong et al. 1986, Leahy & Saridis 1989, Liu 1991, Kozlowski 1992]). Fig. 2.6-1 contains a picture of a PUMA 560 C, the most modern of the three PUMA 560 variants. The differences between the 560 Mk. 1, 560 Mk. 2 and the 560 C are largely concerned with the control systems and do not affect the form of the inverse dynamics.

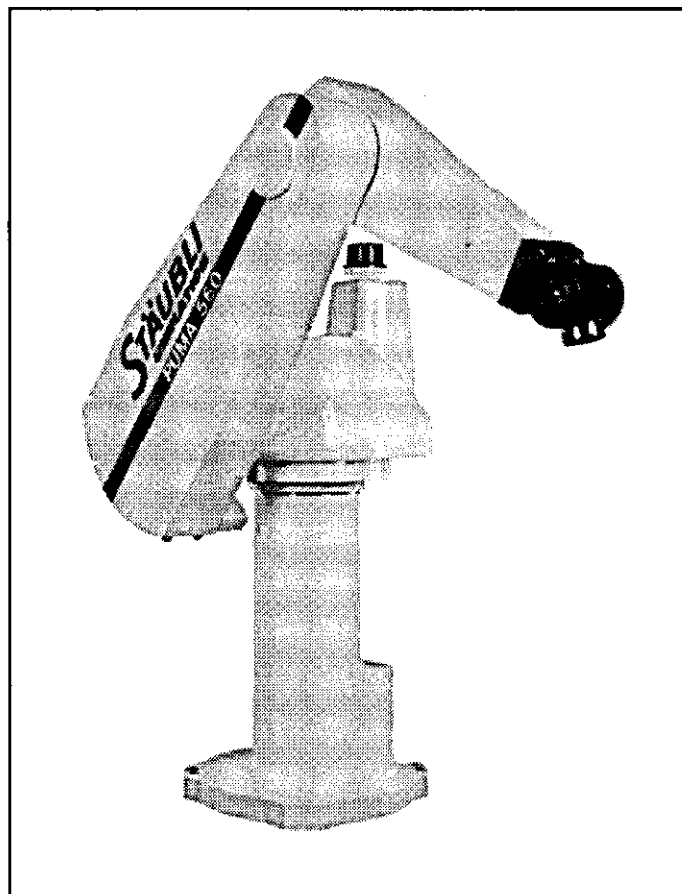
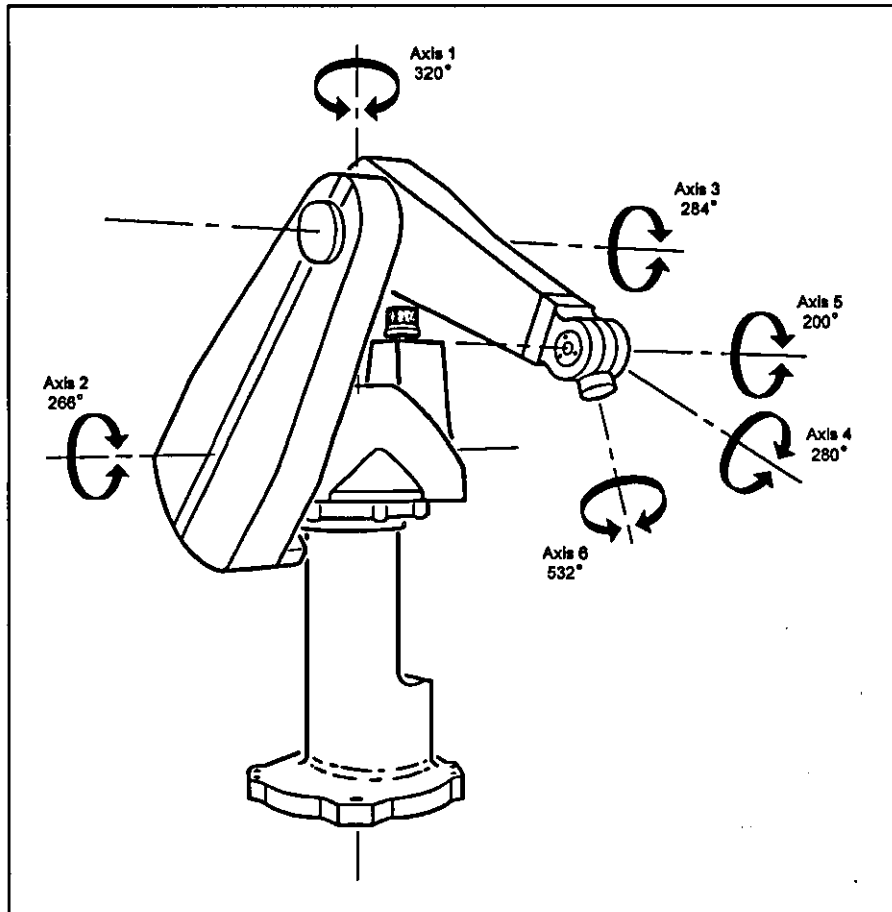


Figure 2.6-1: A PUMA 560 C

The PUMA 560 possesses six degrees of freedom, with all of its joints being revolute. As can be seen from Fig. 2.6-2, this manipulator arm is of anthropomorphic design, that is, its configuration resembles that of a human arm. The first joint can be thought of as the manipulator's waist, the second as a shoulder, the third an elbow, and the fourth and fifth as a wrist. The rolling motion of the wrist is performed by the fourth joint and the flexion by the fifth. The motion of the sixth joint however has no direct anthropomorphic parallel, being a further rolling rotation of the tool mount at the tip of the arm, known as the *end-effector*.



*Figure 2.6-2: The six axes of rotation possessed by a PUMA 560
(Stated values denote range of movement for each axis)*

2.6.1 Frame Assignment and Notation

As can be seen from Eqns. 2.3-24, 2.3-28, 2.4-4, 2.4-7, 2.5-2 & 2.5-4, the precise algebraic form of the inverse dynamics for a specific manipulator is dependent only upon the base homogeneous transforms which describe the configuration of the manipulator's joints.

A manipulator's homogeneous transforms are in turn dependent on the orientation of the coordinate frames attached to each link. There are several different conventions and notations for the task of assigning frames to manipulators' links, including the original Denavit-Hartenberg method [Denavit-Hartenberg 1955], the modified Denavit-Hartenberg method [Craig 1985] and ISO standard number 9787 (1990), which is reproduced in the European standard EN 29787. Of these, the modified D-H method is the simplest to perform and results in the most efficient algebraic representations of the homogeneous transforms. Furthermore, the modified D-H method was used by [Armstrong et al. 1986] to produce the inverse dynamics for the PUMA 560, whose coefficients were then estimated from empirical measurements of the dismantled manipulator. To date, this work

represents the most thorough investigation of its kind, therefore the adoption of the same frame assignment method will allow for direct comparison of coefficient values.

Within the modified D-H method, the assignment convention rigidly attaches frame i to link i , such that the z -axis of the frame coincides with the axis of joint i , and the x -axis of the frame is perpendicular to the axis of joint $i+1$. For simplicity, the base or reference frame (frame 0) is positioned such that its origin and z -axis coincide with those of frame 1. Note that this convention does not necessarily produce a unique set of possible frame orientations, thus, for any given manipulator there is often an element of choice. Here, the same frame assignments as specified by Armstrong et al. are made (see Fig. 2.6-3).

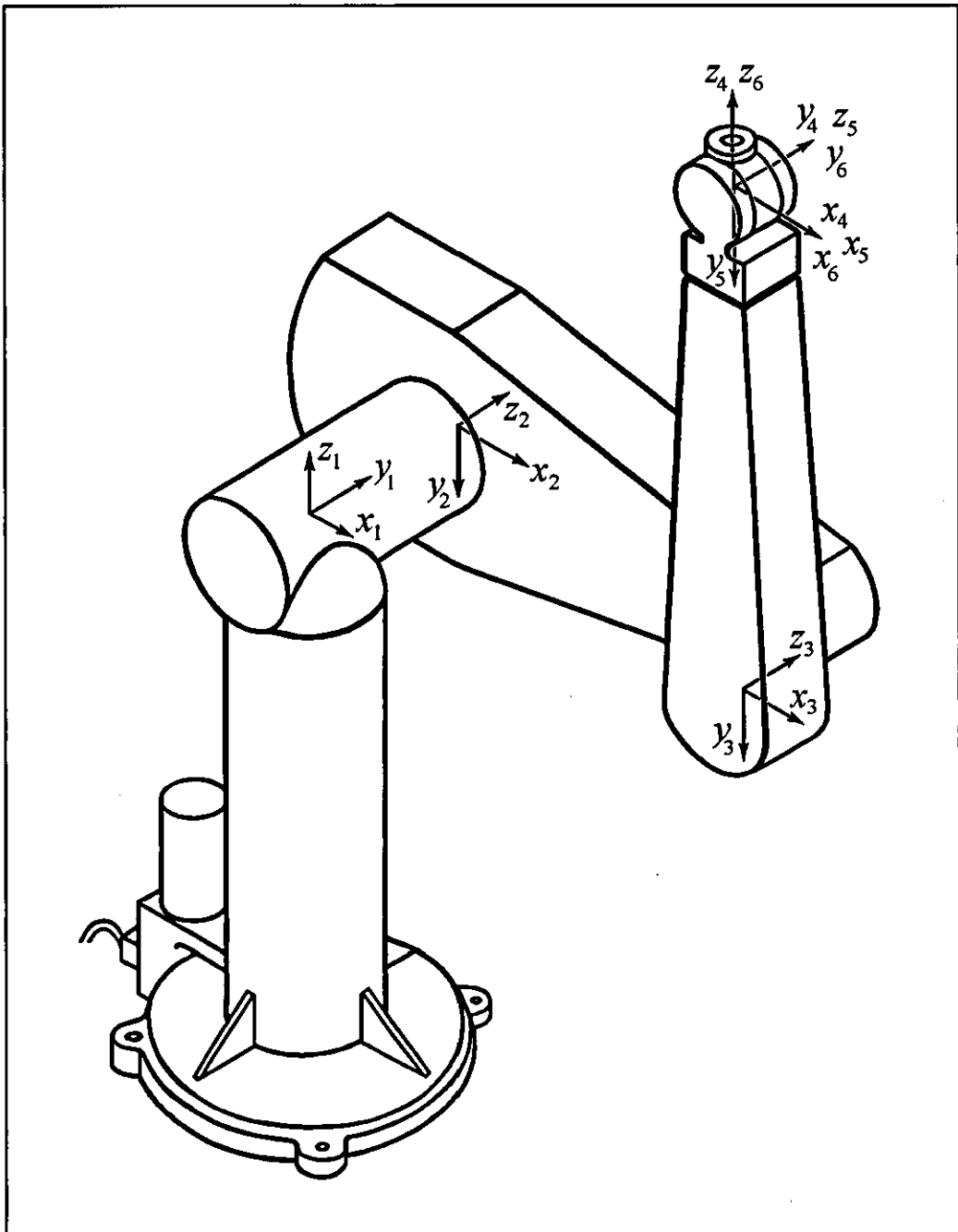


Figure 2.6-3: The frame assignments for a PUMA 560 shown at its zero position

In general, the position and orientation of any given set of axes i , with respect to another set $i-1$, can be expressed by just four quantities known as Denavit-Hartenberg parameters. In terms of a manipulator's frames, these parameters are; the twist and length of link $i-1$ (α_{i-1} & a_{i-1}), the offset of link i (d_i) and the angle of joint i (θ_i). The first two of these parameters between them define the fixed relationship between the two joint axes, whilst the second pair describe the nature of the connection between neighbouring links. They are defined as follows,

α_{i-1} = the angle between the axes of joints $i-1$ and i when projected onto a plane whose normal is orthogonal to both axes.

a_{i-1} = the distance between the axes of joints $i-1$ and i measured along a line orthogonal to both axes.

d_i = the distance between the mutual orthogonal of axes $i-1$ and i , and the mutual orthogonal of axes i and $i+1$, measured along the axis of joint i .

θ_i = the angle between the mutual orthogonals of axes $i-1$ & i , and axes i & $i+1$, when projected onto a plane orthogonal to the axis of joint i .

Note that for any one joint, three of the Denavit-Hartenberg parameters will be constant, and one will be variable. In the revolute case, the joint variable is θ_i , for a prismatic joint, d_i varies with movement. When the modified D-H convention for frame assignment has been employed, the definitions of the D-H parameters become greatly simplified, such that

α_{i-1} = the angle from z_{i-1} to z_i , measured about x_{i-1} .

a_{i-1} = the distance from z_{i-1} to z_i , measured along x_{i-1} .

d_i = the distance from x_{i-1} to x_i , measured along z_i .

θ_i = the angle from x_{i-1} to x_i , measured about z_i .

To finalise the assignment and notation of a manipulator's coordinate frames, one must define a datum position where each of the joint variables is zero. There are two popular choices for a PUMA 560's *zero position*, the first has the arm pointing directly upwards, the second is the crooked arm configuration shown in Fig. 2.6-3. Again, the crooked zero position was selected in order to be able to compare results to those of Armstrong et al.

Thus, from inspection only of the assembled and static PUMA 560, its links' relative orientations and positions, known as the manipulator's *kinematics*, can be described by the 24 modified D-H parameters which appear in Fig. 2.6-4. Note that from thoughtful placement of the coordinate frames, seven of the twelve distance terms have been defined as being zero, leaving just five unknown D-H parameter constants to appear in the manipulator's inverse dynamics.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	q_1
2	-90°	0	d_2	q_2
3	0	a_2	d_3	q_3
4	90°	a_3	d_4	q_4
5	-90°	0	0	q_5
6	90°	0	0	q_6

Figure 2.6-4: The modified Denavit-Hartenberg parameters for a PUMA 560, as identifiable without measurement

2.6.2 Formulation of Homogeneous Transforms

The relative homogeneous transforms for the PUMA 560 can be easily found from the values in Fig. 2.6-4, and from these, the base homogeneous transforms can be determined (Eqn. 2.3-5). In the table of modified D-H parameters, each row contains the information to move to the correspondingly numbered frame from its predecessor, such that in general, for a given frame i , the relative homogeneous transform ${}^{i-1}T_i$ is given by

$${}^{i-1}T_i = \text{Rot}(x_{i-1}, \alpha_{i-1}) \text{Trans}(x_{i-1}, a_{i-1}) \text{Trans}(z_i, d_i) \text{Rot}(z_i, \theta_i) \quad (2.6-1)$$

where

$\text{Rot}(u, \phi)$ = the 4x4 transform describing a rotation of ω about the u -axis

$\text{Trans}(u, l)$ = the 4x4 transform describing a translation of l parallel to the u -axis

Specifically,

$$\text{Rot}(\hat{x}, \omega) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\omega) & -\sin(\omega) & 0 \\ 0 & \sin(\omega) & \cos(\omega) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Trans}(\hat{x}, l) = \begin{bmatrix} 1 & 0 & 0 & l \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}(\hat{z}, l) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Rot}(\hat{z}, \omega) = \begin{bmatrix} \cos(\omega) & -\sin(\omega) & 0 & 0 \\ \sin(\omega) & \cos(\omega) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To verify Eqn. 2.6-1, consider Fig. 2.6-3 and select any of the first five frames, imagine another set of axes coincident with the selected frame, then perform each of the four transformations described in Eqn. 2.6-1 (with values taken from the table in Fig. 2.6-4) to that set of axes in sequence (left to right). The resulting position and orientation of the imagined set of axes will coincide with the coordinate frame which the originally selected frame precedes.

Now that the PUMA 560's relative homogeneous transforms have been defined, its base homogeneous transforms can be calculated from an extension of Eqn 2.3-5, thus:

$$T_i = {}^0T_1 {}^1T_2 \dots {}^{i-1}T_i \quad (2.6-2)$$

Note that for any manipulator with parallel consecutive revolute joint axes, such as the PUMA 560's second and third joints, premultiplying the corresponding relative homogenous transforms within Eqn. 2.6-2, in this case 1T_2 and 2T_3 , enables one to take advantage of the sum of angles identities and thus simplify the resultant expression.

2.6.3 Non-Lagrangian Effects

The first three links of the PUMA 560 are neither small nor light, therefore it is not surprising that frictional effects have been reported as being significant to the manipulator's motion [Leahy & Saridis 1989]. Although often not modelled, and occurring only at low joint speeds, even static friction can have detectable effects.

All the driving motors within a PUMA 560 have geartrains with ratios (r) much less than unity [Armstrong et al. 1986]. From Eqn. 2.5-5, it can be seen that the inertias of these motors are greatly magnified in terms of their effect on the manipulator's motion, and thus must be incorporated in the inverse dynamics model.

2.6.4 Detailed Inverse Dynamics

The equations for the fully detailed inverse dynamics of the PUMA 560 can be found in Appendix B, these are exactly equivalent to the equations reported by [Armstrong et al. 1986], with the addition of a friction model. The equations have been parameterised and presented in a style suitable for use with an adaptive network (see Chapter 5). Note that the parameterisation performed by Armstrong et al. (for computational efficiency) produces a different set of coefficients than those found in the appendix. Although under less stringent restrictions as to how to group physical parameters within the inverse dynamics into constant coefficients, Armstrong et al. chose to treat the effective motor inertias separately, hence producing a longer list of mass and velocity term coefficients

than achieved here (29 instead of 26).

The generalised Euler-Lagrange equation for manipulators inverse dynamics (Eqn. 2.5-2) hides considerable complexity. It can be seen from Eqn. 2.6-2 that the base homogeneous transform for a given frame grows in complexity the greater the number of intermediate frames between it and the base frame. Now consider Eqn. 2.3-24, the definition of the mass matrix (without actuator effects), from this one can appreciate how large expressions within the inverse dynamics can become. In the case of the PUMA 560, with frame assignments as described in this section, the top left most element of the mass matrix alone has 224 separate terms when fully expanded, similarly, several of the 105 unique Christoffel symbols also each contain over two hundred separate terms.

2.7 MODEL-BASED CONTROL ALGORITHMS

Although the method for identifying manipulator dynamics coefficients presented in this work is independent of which algorithm is employed to control the manipulator's motion, a brief description of a model-based algorithm is given here to aid the reader's understanding of the purpose of this work. By incorporating a model of the manipulator's inverse dynamics into a control algorithm, often in conjunction with one or more error based terms, the nonlinear and configuration dependent characteristics of the manipulator's dynamics can be compensated for. Known as computed torque control [Paul 1972, Fu et al. 1987], various degrees of compensation for the manipulator's dynamics are possible. However, it has been shown [Leahy & Saridis 1989] that the more complete the model of the inverse dynamics which is incorporated into the controller, the better the trajectory-following performance. Another study that demonstrates the superiority of model-based control over conventional PD control [Whitcomb et al. 1993], also points out the importance of having accurately identified system parameters, describing how a model-based controller with only approximately correct parameters may perform less well than a simple PD controller. Hence, accurately identified values of a manipulator's dynamics coefficients are vital for the advantages of model-based control are to be realised in practice.

2.7.1 Fully Compensated Computed Torque

Recalling that the complete inverse dynamics for a rigid bodied manipulator (Eqn. 2.5-2) can be described as

$$\tau = M\ddot{q} + V + G + F$$

and given that a comprehensive controller should adjust the applied torques/forces to correct for deviations from a desired trajectory, whilst compensating for the orientation

dependent dynamics, then it is clear to see how *fully compensated computed torque* can be defined as

$$\tau = M(\ddot{q}_d + K_v(\dot{q}_d - \dot{q}) + K_p(q_d - q)) + V + G + F \quad (2.7-1)$$

where, for a chosen trajectory,

q_d = the desired general position vector for the given instant

\dot{q}_d = the desired general velocity vector for the given instant

\ddot{q}_d = the desired general acceleration vector for the given instant

and

K_v = the (diagonal) velocity error gain matrix

K_p = the (diagonal) positional error gain matrix

It can be seen that the control law represented by Eqn. 2.7-1 contains information on the desired movement at each given instant, in the form of the desired joint accelerations, two error driven terms based on deviations in expected joint positions and velocities, and compensation for mass, velocity, gravity and frictional effects. The operation of the fully compensated computed torque control scheme is shown graphically in Fig 2.7-1. Note the inclusion of a variable disturbance term, D , which occurs due to noise and imperfectly modelled inverse dynamics.

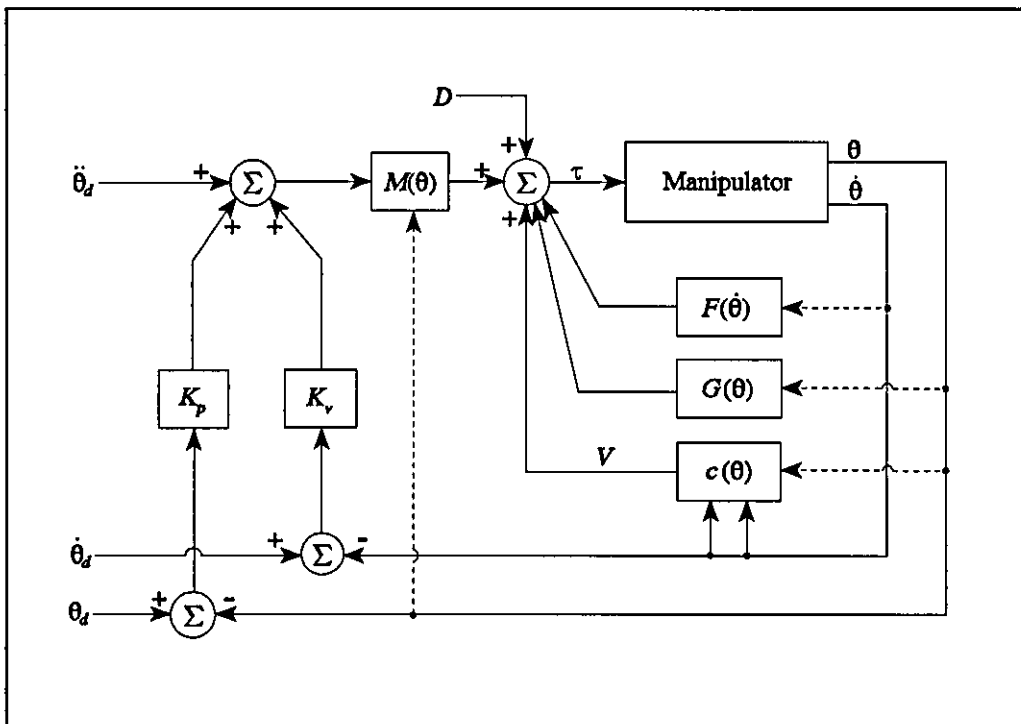


Figure 2.7-1: Fully compensated computed torque control
(Dashed lines show nonlinear dependencies)

For the ideal case, where the model of the manipulator's inverse dynamics is precise and the measurement of positions and velocities are exact, the disturbance term D is zero. In these circumstances, if the gain matrices (K_p & K_v) are diagonal, Eqn 2.7-1 represents a perfectly linearised controller. That is, the acceleration induced in a given joint is a linear function of the desired acceleration, position error and velocity error of solely that joint. Furthermore, it can be shown [Leahy & Saridis 1989] that for such a perfectly linearised controller

$$K_{v_{ii}} = 2 \zeta_i \omega_{n_i} \quad \text{and} \quad K_{p_{ii}} = \omega_{n_i}^2 \quad (2.6-2)$$

where

$$\begin{aligned} \zeta &= \text{the error response damping ratios vector} \\ \omega_n &= \text{the error response natural frequencies vector} \end{aligned}$$

2.8 SUMMARY

Within this chapter, the Euler-Lagrange equation for a rigid bodied manipulator's inverse dynamics has been derived from first principles and in the general case. Compensation has been included in the inverse dynamics model for the effects of both friction and actuators, and the consequences of compliance and backlash within the geartrain have been considered. The importance of discerning when a joint is experiencing backlash was also noted for systems where measurements of the manipulator's motion are used for identifying the inverse dynamics.

The PUMA 560 manipulator has been described and its configuration defined using the modified Denavit-Hartenberg technique. The spatial relationship parameters obtained were used to form the fully detailed algebraic representation of the PUMA 560's inverse dynamics. This includes expressions for the inverse dynamics coefficients that entirely consist of physical parameters, such as lengths and masses. The considerable size and complexity of the inverse dynamics model was also noted. Furthermore, all of the dynamical components for which terms have been derived in this chapter have been reported by other researchers to have significant effects on the motion of the PUMA 560 manipulator. Thus the suitably comprehensive Euler-Lagrange model of its inverse dynamics is highly non-linear, discontinuous, configuration dependent and contains a huge number of terms.

For most manipulators it is not possible to measure the physical parameters that appear in their inverse dynamics, in particular the frictional and inertial parameters, to an acceptable degree of accuracy. Therefore, the relationship between a manipulator's instantaneous generalised positions, velocities and accelerations, and the forces and/or

torques applied to it must be identified indirectly. As the majority manipulators are constructed to be feedback controlled devices, information on a manipulator's motion is commonly available in the form of instantaneous joint position measurements, which can be compared to the applied forces and/or torques. Some manipulators also possess encoders that directly measure instantaneous velocities, whilst a few additionally have encoders capable of measuring the instantaneous accelerations. For the majority of manipulators, such as the unmodified PUMA 560, that do not have velocity or acceleration sensors, the instantaneous values of \dot{q} and \ddot{q} can be estimated from numerical differentiation.

In the next chapter, adaptive networks are introduced for the purpose of modelling manipulator dynamics. By iteratively processing data on a given manipulator's motion, a network can adapt to form a model of that specific manipulator's inverse dynamics, thus providing information on the values of the dynamics coefficients.

REFERENCES AND FURTHER READING

- Arimoto, S., Miyazaki, F., "Stability and robustness of PID feedback control for robot manipulators of sensory capability," *Proc. First Int. Symp.*, pp. 783-799, MIT, Cambridge, MA, 1984.
- Armstrong, B., Khatib, O., and Burdick, K., "The explicit dynamic model and inertial parameters of the PUMA 560 arm," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 510-518, 1986.
- Armstrong, W.W., "Recursive solution to the equations of motion of an n -link manipulator," *Proc. Fifth World Congress on Theory of Machines and Mechanisms*, Montreal, 1979.
- Bejczy, A.K. "Robot Arm Dynamics and Control," Technical Memo 33-699, Jet Propulsion Laboratory, Pasadena, California, 1974.
- Borisenko, A.I., and Tarapov, I.E., *Vector and Tensor Analysis with Applications* Englewood Cliffs, NJ: Prentice Hall, 1968.
- Brewer, J.W., "Kronecker products and matrix calculus in system theory," *IEEE Trans. Circuits Syst.*, vol. CAS-25, no. 9, pp. 772-781, Sept. 1978.
- Corben, H.C., and Stehle, P., *Classical Mechanics*. New York; Wiley, 1950.
- Craig, J.J, *Introduction to Robotics: Mechanics & Control*, 2nd Ed. Reading, MA: Addison-Wesley, 1986.
- Dagalakis, N.G., and Myers, D.R., "Adjustment of robot gear backlash," *Int. J. Robotics Res.*, vol. 4, no. 2, pp. 65-79, 1985.
- Denavit, J., and Hartenberg, R.S., "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics.*, vol. 77, pp. 215-221, 1955.
- Fu, K.S., Gonzalez, R.C., and Lee, C.S.G., *Robotics: Control, Sensing, Vision and Intelligence*, Int. Ed. New York: McGraw-Hill, 1987.

- Goldstein, H., *Classical Mechanics*. Reading, MA: Addison-Wesley, 1981.
- Groover, M.P., Weiss, M., Nagel, R.N., and Odrey, N.G., *Industrial Robotics: Technology, Programming, and Applications*, Int. Ed. New York: McGraw-Hill, 1986.
- Gu, Y.-L., and Loh, N.K., "Dynamic model for industrial robots based on a compact Lagrangian formulation," *Proc. IEEE Conf. Decision Control*, pp. 1497-1501, 1985.
- Gu, Y.-L., and Loh, N.K., "Dynamic modeling and control by utilizing an imaginary robot model," *IEEE J. Robot. Autom.*, vol. 4, no. 5, pp. 532-534, Oct. 1988.
- Hemami, H., Jaswa, V.C., and McGhee, R.B., "Some alternative formulations of manipulator dynamics for computer simulation studies," *Proc. Allerton Conf. on Communication, Control and Computing*, Monticello, IL, Oct., 1975.
- Hollerbach, J.M., "A recursive formulation of Lagrangian manipulator dynamics," *IEEE Trans. Sys., Man, and Cyber.*, vol. 10, no. 11, 1980.
- Koditschek, D., "Natural motion for robotic arms" *Proc. IEEE Conf. Decision Control*, pp. 733-735, Dec. 1984.
- Kozłowski, K., "Identification of model parameters of robot manipulators," *Systems Science (Poland)*, vol. 18, no. 1-2, pp. 165-187, 1992.
- Kubo, T., Anwar, G., and Tomizuka, M., "Application of nonlinear friction compensation to robot arm control," *Proc. IEEE Intl. Conf. Robotics and Autom.*, pp. 722-727, April 1986.
- Lagrange, J.L., *Mécanique Analytique*, parts 1-2, A. Blanchard, reprint, 1965
- Leahy, M.B., Jr., "Performance characterization of a PUMA-600 robot," RAL Technical Report No. 56., Troy, N.Y.: Rensselaer Polytechnic Institute, Dept. of Electrical, Computer, and Systems Engineering, 1985.
- Leahy, M.B., Jr., and Saridis, G.N., "Compensation of industrial manipulator dynamics," *Int. J. Robotics Res.*, vol. 8, no. 4, pp. 73-84, 1989.
- Lee, C.S.G., "Robot arm kinematics, dynamics, and control," *Computer*, vol. 15, no. 12, pp. 62-80, 1982.
- Lee, C.S.G., Lee, B.H., and Nigam, R., "Development of the Generalized d'Alembert Equations of Motion for Mechanical Manipulators," *Proc. 22nd Conf. Decision and Control*, San Antonio, Tex., pp. 1205-1210, 1983.
- Lewis, R.A., "Autonomous Manipulation on a Robot: Summary of Manipulator Software Functions," Technical Memo 33-679, Jet Propulsion Lab., Pasadena, Calif., 1974.
- Liu, M., "Puma 560 robot arm analogue servo system parameter identification," Tech. Rep. ASR-91-1, University of Melbourne, Dept. Mech. and Man. Eng., Feb. 1991.
- Lowe, J.D., Lin, Y., and Nguyen, L., "Development, Validation, and Use of a Flexible Multi-Body Dynamic Model of the Space Shuttle Remote Manipulator System", *The Fourth Int. Symposium on Measurement and Control in Robotics*, Smolenice Castle, Slovakia, June 1995.
- Luh, J.Y.S., Walker, M.W., and Paul, R.P., "On-Line Computational Scheme for Mechanical Manipulators," *Trans. ASME, J. Dynamic Systems, Measurements and Control*, vol. 120, pp. 69-76, 1980.
- Marilier, T., and Richard, J.A., "Non-linear mechanic and electric behavior of a robot axis with a harmonic drive gear," *Robot. Computer-Integrated Manufact.*, vol. 5, nos. 2-3, pp. 129-136, 1989.

- Marion, J.B., *Classical Dynamics*, New York: Academic Press, 1965.
- Murphy, S.H., and Wen, J.T., "Analysis of Active Manipulator Elements in Space Manipulation," *IEEE Trans. Robotics and Autom.*, vol.9, no.5, pp.544-552 1993.
- Nagy, P.V., "The PUMA 560 industrial robot: Inside-out," *Robots 12*, pp. 4-67, June 1988.
- Orin, D.E, McGhee, R.B., Vukobratovic, M. and Hartoch, G., "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods," *Math. Biosci.*, vol. 43, pp. 107-130, 1979.
- Paul, R.P.C., "Modelling, trajectory calculation, and servoing of a computer controlled arm," AI Laboratory Memo Am-177. Palo Alto, Cal: Stanford University, 1972
- Paul, R., Rong, M., and Zhang, H., "Dynamics of puma manipulator," *American Control Conf.*, 1983.
- Rutherford, D.E., *Classical Mechanics*, Oliver & Boyd, 1957.
- Schilling, R.J., *Fundamentals of Robotics: Analysis and Control* Englewood Cliffs, N.J.: Prentice Hall, 1990.
- Shames, I.H., *Engineering Mechanics: Statics and Dynamics*, 4th Ed., Upper Saddle River, N.J.: Prentice Hall, 1997.
- Silver, D.B., "On the equivalence of Lagrangian and Newton-Euler dynamics for manipulators," *Int. J. Robotics Res.*, vol. 1, no. 2, 1982.
- Smith, P., and Smith, R.C., *Mechanics*, 2nd Ed., Chichester, West Sussex: Wiley, 1935.
- Spong, M.W., and Vidyasagar, M., *Robot Dynamics and Control*. New York: Wiley, 1989.
- Stadler, W., *Analytical Robotics and Mechatronics*, Int Ed. New York: McGraw-Hill, 1995.
- Stäubli-Unimation Inc. USA
Pittsburgh, PA 15143-2305. Tel.: (412) 741 1740, Fax: (412) 741 1789
- Stäubli-Unimation Ltd. GB
Telford, Shropshire TF3 3BN, Tel.: (1952) 290931, Fax: (1952) 290057
- Tarn, T.J., Bejczy, A.K., Han, S., and Yun, X., "Inertia parameters of puma 560 robot arm," Tech. Rep. SSM-RL-85-01, Washington University, St. Louis, MO., Sept. 1985.
- Tarn, T., Bejczy, A.K., Yun, X., and Li, Z., "Effect of motor dynamics on nonlinear feedback robot arm control," *IEEE Trans. Robotics and Automation*, vol. 7, pp. 114-122, Feb. 1991.
- Turney, J.L., Mudge, T.N., and Lee, C.S.G. "Equivalence of Two Formulations for Robot Arm Dynamics," SEL Report 142, ECE Department, University of Michigan, Ann Arbor, Mich., 1980.
- Tuttle, T.D., and Seering, W.P., "A nonlinear model of a harmonic drive gear transmission," *IEEE Trans. Robot. and Autom.*, vol. 12, no. 3, 1996.
- Uicker, J.J., "On the Dynamic Analysis of Spatial Linkages using 4x4 Matrices," Ph.D dissertation, Northwestern University, Evanston, Ill., 1965.
- Uicker, J.J., Jr., Denavit, J, and Hartenberg, R.S., "An Iterative Method for the Displacement Analysis of Spatial Mechanisms," *Trans. ASME, J. Appl. Mech.*, vol. 31, Series E, pp. 309-314, 1964.

Whitcomb, L.L., Rizzi, A.A., and Koditschek, D.E., "Comparative experiments with a new adaptive controller for robot arms," *IEEE Trans. Robotics and Automation*, vol. 9, no. 1, pp. 59-70, Feb. 1993.

Whittaker, E.T., *Analytical Dynamics of Particles and Rigid Bodies*, Dover, reprint, 1944.

3. ADAPTIVE NETWORKS

Adaptive networks are connectionist models of systems with initially unknown properties. Automatic adaption techniques can be employed on these networks to improve their processing performance, in the absence of system knowledge, and using only potentially imperfect sample data from the modelled system. Originally developed as classifying and decision making tools for discrete systems, adaptive networks are now used in a vast assortment of applications, including the modelling of systems with analogue inputs and/or outputs. As such, it is intended to use an adaptive network to identify the inverse dynamics coefficients of a given manipulator from measurements of its motion.

A general overview of adaptive networks, including an introduction to the accompanying terminology, is given in Section 3.1. The advantageous properties common to all adaptive networks are discussed in Section 3.2, illustrating the primary reasons for their use in the coefficient identification problem. Various aspects of network design are examined in Section 3.3, whilst automatic adaption methods are investigated in Section 3.4. The analysis of both is general, but particular emphasis is made of their use in modelling analogue output systems such as the chosen application. The specifics of applying adaptive networks to problems involving manipulator dynamics are examined in Section 3.5, including a review of other researchers' work in this area. The major obstacles currently preventing acceptable coefficient identification via adaptive networks are investigated, leading to a discussion of the currently postulated theories for poor network generalisation in Section 3.6.

3.1 GENERAL ADAPTIVE NETWORK DEFINITIONS

Due to their original source of inspiration, and the properties they share with biological neural systems, adaptive networks (especially those containing nonlinear functions) are often referred to as *artificial neural networks*.

All adaptive networks are based on the same structure; data, in the form of numeric values, is passed through a network of *nodes* (sometimes called *cells*, *processing elements* or *artificial neurons*), each of which performs some function on its inputs to produce a single output. The connections between nodes are called *links*, and each one has a *weight*

associated with it which factors values passing along that link. Nodes are usually organised into distinct *layers*, where the outputs of the nodes in a given layer do not feed into other nodes within the same layer. The example network in Fig. 3.1-1 has two layers, one of whose outputs are also the network outputs, and is therefore known as the output layer. Layers which are not output layers are known as hidden layers, a network may have any number of hidden layers including zero.

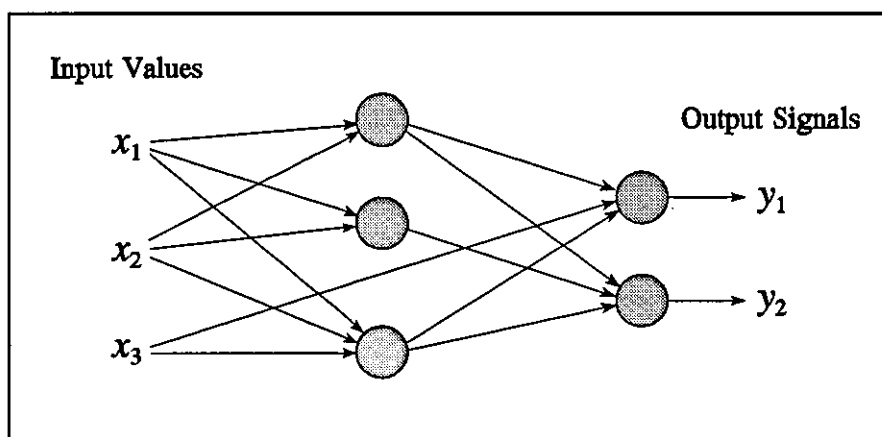


Figure 3.1-1: An example network (nodes shown shaded, weights not shown)

The adjustment of a network's weight values, often performed iteratively, is known as *training*. The aim of training is to increase some measure of the network's performance in processing supplied data. The weight adjustment values are calculated by a chosen *learning algorithm*, of which there are three main categories, those that perform *supervised learning*, those that perform *reinforcement learning* and those that perform *unsupervised learning*.

In the case of supervised learning, the output signal from the network is compared to a desired target set of values to create an error signal. This error is then operated upon by a learning algorithm with the aim of producing new weight values that will cause the network to yield smaller errors when processing similar inputs. To provide a measure of a network's processing performance, networks usually have associated with them a cost function, this acts upon the network errors to produce an *error cost*. A network is said to be *fully trained* when it produces the minimum possible error cost. A single vector of input values, plus any associated vector of target outputs, is called a *training pattern*, and the complete set of patterns used in a training process is called the *training set*.

Reinforcement learning is similar to supervised learning except that the learning algorithm is supplied with only an evaluation score or grade, instead of a target value for each output. These grades need not be supplied at every training iteration, and obviate the need for a precise target value for each training pattern to be known. Due to this, reinforcement learning is particularly suited to working in a dynamic environment, such

as exists in the local optimisation of robot dynamics [Zomaya & Nabhan 1993]. However, the identification of a manipulator's global inverse dynamics parameters is not a dynamic problem, and in such cases reinforcement learning usually requires more iterations than supervised learning to fully train the network.

Unsupervised learning algorithms (which are employed on *self-organising* networks) receive no external measure of performance but instead use in-built rules to modify the network weights. The aim of such algorithms is usually to produce consistent outputs, that is, similar input patterns should produce the same pattern of outputs. The training process, therefore, extracts the statistical properties of the set of inputs used and groups them into classes. It is clear that such qualitative analysis is not readily suited to the task of parameter identification.

Note that in Fig. 3.1-1 all the data within the network flows one way, that is, there are no feedback loops. Such networks are called *feedforward* or *nonrecurrent* networks and are the most common choice of configuration for system modelling.

3.2 PROPERTIES OF ADAPTIVE NETWORKS

Several characteristics of adaptive networks set them apart from conventional computing and artificial intelligence approaches. This section outlines three important and advantageous properties that all adaptive networks share, namely, a mathematical basis, an ability to learn and an ability to generalise.

3.2.1 Mathematical Basis

Adaptive networks are one of the few AI-related technologies that have a rigorous mathematical foundation. This produces a comparative ease of analysis, as well as allowing the incorporation of other, of already well understood, mathematical tools and techniques.

3.2.2 Learning

Adaptive networks can modify their behaviour in response to their environment. This factor, more than any other, is responsible for the interest they have received. Given a set of inputs (perhaps with desired outputs), they can automatically self-adjust to produce consistent responses without first possessing any knowledge of the system which produced the training set. Many dozens of learning algorithms have been developed to

perform this task, some specific to certain network types, others general, and all with their own strengths and weaknesses [Rumelhart & McClelland 1986, Anderson & Rosenfeld 1988, Hassoun 1995].

3.2.3 Generalisation

Once trained, a network can, to a degree, correctly respond to inputs that it has not processed before. Generalisation takes the ability to learn and self-adjust a step further: the network is able to hypothesise a response based on previous experiences. Note that adaptive networks generalise automatically as a result of their structure, not by using human intelligence embedded in the form of ad hoc computer programs.

Although the property of generalisation is often cited, in practice many networks' ability to accurately generalise extends only to interpolation between supplied training data, as opposed to the desired extrapolation to the whole problem space [Caelli et al. 1993, Bishop 1995]. The analysis of network training presented in Chapter 4 will show that the majority of high input order networks with analogue valued outputs can never be made to generalise to the whole problem space as a result of training.

3.3 NETWORK ARCHITECTURES

The quality which differentiates networks is their *architecture*, that is, their number of nodes, the connectivity of the nodes and the functions present within the nodes. Some architectures require the use of specialist learning algorithms, but many are capable of being trained using one of a selection of learning algorithms. The different types of node are often named after the class of function they contain.

This section introduces the three major types of node, namely the hard-limiters, linear combiners and locally sensitive nodes. The use of network input preprocessing and of context sensitive weights are also discussed.

3.3.1 The Hard-Limiter

Historically the first type of trainable adaptive network node developed [Rosenblatt 1958], a *hard-limiter* (also known as a *perceptron*) is a two-class classifier, such as the step function shown in Fig 3.3-1; If the sum of its weighted inputs is greater than or equal to zero, its output is one, otherwise its output is zero.

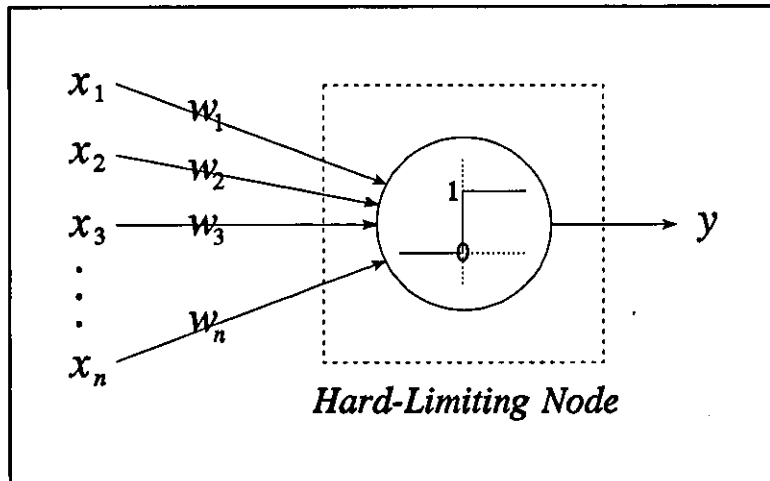


Figure 3.3-1: The functionality of a Hard-Limiting Node

To express the operation performed by a hard-limiter algebraically, the node's operation is broken down into two parts. The first is the summation of the weighted inputs, which is general to almost all types of network node and produces the scalar value s , such that

$$s = \sum_{i=1}^n x_i w_i \quad (3.3-1)$$

The second is particular to the type of node, in the case of a zero/one classifying hard-limiter this function is the step function, expressed as

$$y = \begin{cases} 1 & \text{if } s \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.3-2)$$

The hard-limiting node was intended to be a simplified model of a biological neuron within the brain, it having been noted that neurons either fire an impulse of a fixed magnitude, or do not, based on the sum charge applied to them. Because of this, the function performed by a node on the summed input value s is often called the *activation function*. Note that in order to provide a non-zero threshold for the activation function it is common to fix one of the network's inputs to a constant value of one, the weight associated with the link from this input to the node in question is then known as the node's *bias*. Biasing is common in many different types of network.

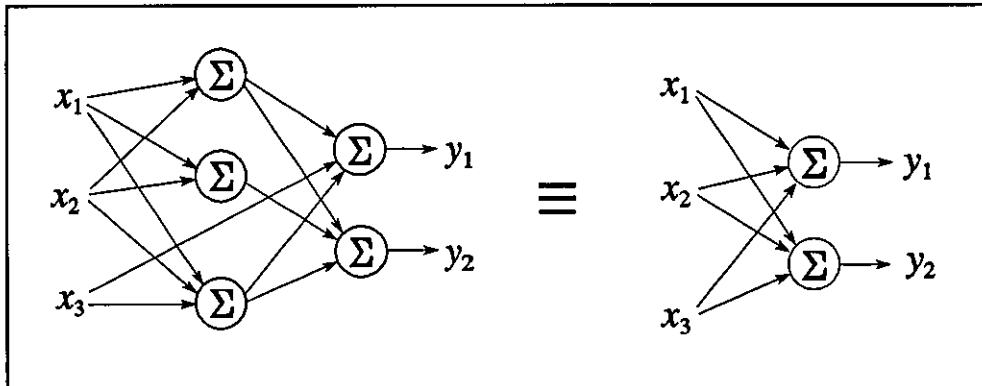
Note that although the hard-limiter was originally known as the perceptron, this name has been corrupted to refer to almost any node containing a step-like, or even just locally sensitive, activation function. Hence, use of the name perceptron is avoided in this report.

The discrete nature of a hard-limiter's output is clearly better suited to problems with discrete answers than to those with analogue ones. Hard-limiter networks are therefore principally used for problems pertaining to classifying data.

3.3.2 The Linear Combiner

Developed by Widrow and Hoff, the adaptive linear combiner element (ADALINE) [Widrow 1962] possesses no activation function as such, instead the output is simply the sum of the weighted inputs, that is $y = s$. When such networks have an error cost function associated with them that is a polynomial function of the output error moduli (such as the mean squared error or mean weighted error moduli), they possess a very important property. If one plots the error cost of a network containing n weights, for all possible weight values, in an $n+1$ dimensional space, then the hypersurface which describes the error cost is continuous, smooth and contains only one minima. That minima represents the optimal state of the network weights for the training set employed. Furthermore, any learning algorithm which correctly follows the gradient of the error cost hypersurface downwards will ultimately train the network to this state.

Despite the desirable properties stated above, standard feedforward networks (such as in Fig. 3.1-1) which are entirely made up from linear combiner nodes are rarely used, this is because they can only ever model linear systems. It can be shown [Wasserman 1989], that regardless of the number of nodes or how they are interconnected, a standard feedforward linear combiner network is always equivalent to a linear combiner network with only a single layer (where all the nodes' outputs are network outputs and do not feed into other nodes). An example of this is shown in Fig 3.3-2;



*Figure 3.3-2: Linear Combiner network equivalence
(The two layer network on the left is equivalent in operation to
the single layer network on the right)*

Clearly, single layer linear combiner networks can be represented by a single matrix of weight values acting on the input vector. For example, the right-hand representation of the network in Fig. 3.3-2 could be described by

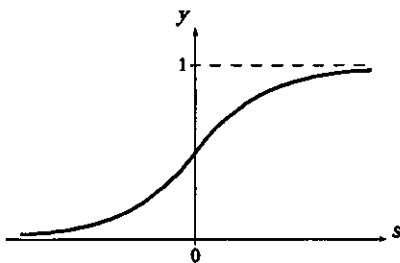
$$xW = y$$

where x and y are the input and output row vectors, and W denotes a 3×2 weights matrix. Therefore, any system adequately modelled by a conventional linear combiner network

can, for many purposes, be better analysed by linear algebra techniques [Strang 1988, Faddeev & Faddeeva 1963, Kolman 1988].

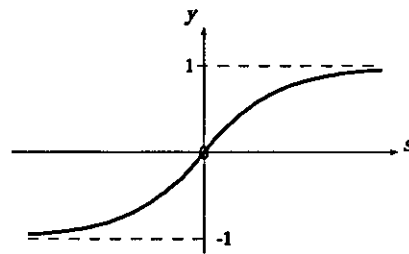
3.3.3 Locally Sensitive Nodes

There exists a type of node whose activation function is continuous but has near zero gradient for all but one localised area of the node's input space, such nodes are referred to as being *locally sensitive*. In particular, a subcategory of these are designed to combine the decision making property of hard-limiters with the continuous quality of linear combiners and are collectively called *smoothed step* or *sigmoidal* nodes [Nelson & Illingworth 1991]. These contain activation functions which approximate a step, so producing nonlinearities, but are differentiable, ensuring a continuous error cost hypersurface. Two examples of sigmoidal functions are shown below;



The Logistic Function

$$y = \frac{1}{1 + e^{-s}}$$



The Hyperbolic Tangent Function

$$y = \tanh(s)$$

Another important subcategory of locally sensitive activation functions are *Radial Basis Functions* (RBFs) [Powell 1987]. Although theoretically not restricted to being locally sensitive, almost all RBFs used in adaptive networks are chosen to be so, the most common amongst these being the Gaussian (bell-shaped) function. RBF nodes have a more complex form than the simpler smoothed step nodes, such that their area of local sensitivity is defined in the multi-dimensional node input space, rather than the one dimensional summed node input space. This in part gives rise to a number of properties which are useful for system approximation and adaption (regularisation; [Poggio and Girosi 1990], best approximation; [Girosi and Poggio 1990], kernel regression [Scott 1992], noisy interpolation; Webb [1994]), however, RBFs require specialised learning algorithms and place restrictions on the topology of the network.

Analogue output adaptive networks commonly contain locally sensitive nodes. One important reason for this is that many locally sensitive continuous functions are capable

of universal approximation, that is, a linear superposition of enough such functions can approximate any continuous real-valued system function [Hornik et al. 1989, Baldi 1990, Light 1992, Park and Sandberg 1993].

3.3.4 Network Preprocessing

Preprocessing of network inputs can have significant benefits, particularly where it is possible to reduce the order of the system that the network models [Specht 1966, Barron 1984]. A good example of this is the logical exclusive or (XOR) problem; Marvin Minsky infamously co-wrote a book [Minsky & Papert 1969] with the intention of demonstrating the inadequacies of adaptive networks. In this he “proved” that a single-layer hard-limiter based network can never model an XOR operator, regardless of weight values or topology (multilayer network training algorithms did not exist at that time, see Section 3.4). This is due to the linear operation of such networks and the nonlinear nature of the XOR operator. However, it can be seen from Figs. 3.3-3 & 3.3-4 that with relatively simple preprocessing of the network inputs, and the addition of a bias, the XOR problem can in fact be modelled by a network consisting of just a single hard-limiting node.

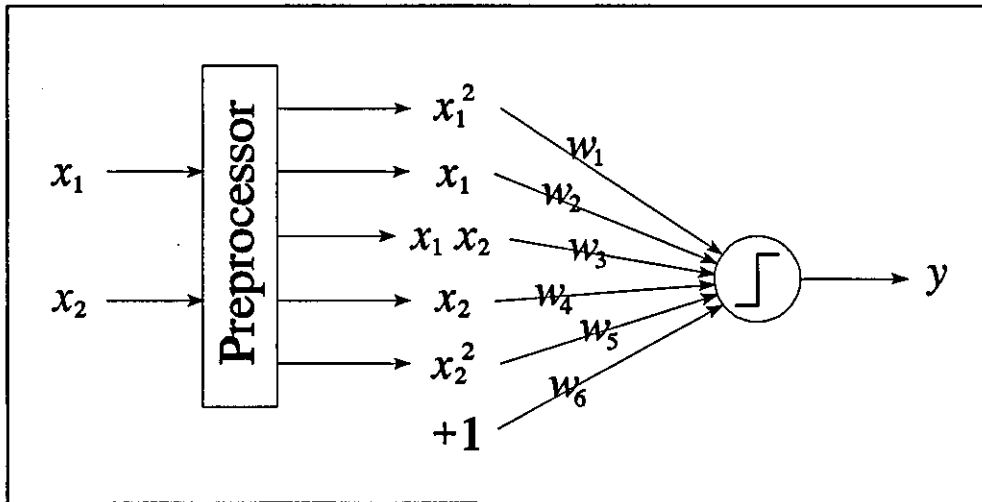
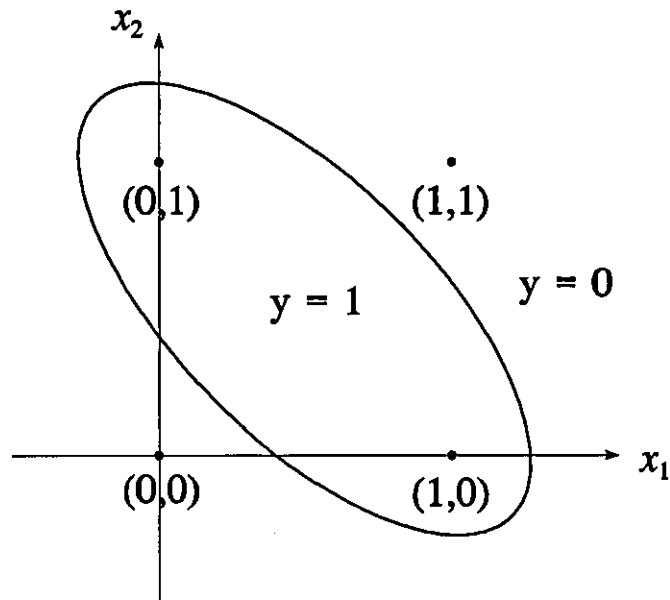


Figure 3.3-3: XOR modelling hard-limiter network employing input preprocessing

The preprocessor in the XOR modelling network performs nonlinear combinations of the two inputs (x_1, x_2) to produce $x_1^2, x_1 x_2$ and x_2^2 . These three quantities, along with the original two inputs and a constant bias are then used as input data for the single hard-limiting node. Since there is no requirement for network inputs to be independent of each other, the preprocessing performed in the XOR problem has been able to take a system that is not linearly separable in (x_1, x_2) space (see Fig. 3.3-4) and made it instead linearly separable in a five dimensional space which has a direct mapping from (x_1, x_2) .



*Figure 3.3-4: Graphical representation of the trained XOR network's output
(ellipse shows where the sum of the node's inputs equals zero)*

The importance and usefulness of preprocessing network inputs has been increasingly accepted during recent years. However it is evident that one requires some knowledge of the form of the modelled system in order to sensibly choose which preprocessing to employ. In the absence of such a priori knowledge, statistical methods can be used to find correlations between the system output and arbitrarily processed inputs which are stronger than the correlations between the system output and the unprocessed inputs. However, as there are an infinite number of possible preprocessing functions, such statistical methods are not reliable substitutes for genuine understanding of the form of the modelled system.

Preprocessing therefore provides a good opportunity to incorporate a priori system knowledge into a network's design, to the benefit of both the network's simplicity and its accuracy once trained. For an example of its use in a practical application see [Specht 1967].

3.3.5 Context Sensitive Networks

First proposed by [Yeung 1989, Yeung & Gekey 1989], context sensitive networks differ from other networks by having weights that take different values for each pattern processed, often these context sensitive weight values are the outputs of another subnetwork. Such networks are fundamentally more sophisticated than the conventional networks so far discussed, in which data is weighted, summed and then acted upon by a single-input-single-output activation function whose output may then be weighted,

summed with some more data, then acted upon, and so on until it is outputted. In a context sensitive link, data is actually multiplied by other data, producing nonlinear double-input-single-output combinations. This expansion in the kind of information processing possible within a network allows context sensitive networks to potentially model many more types of system than is possible with conventional network architectures.

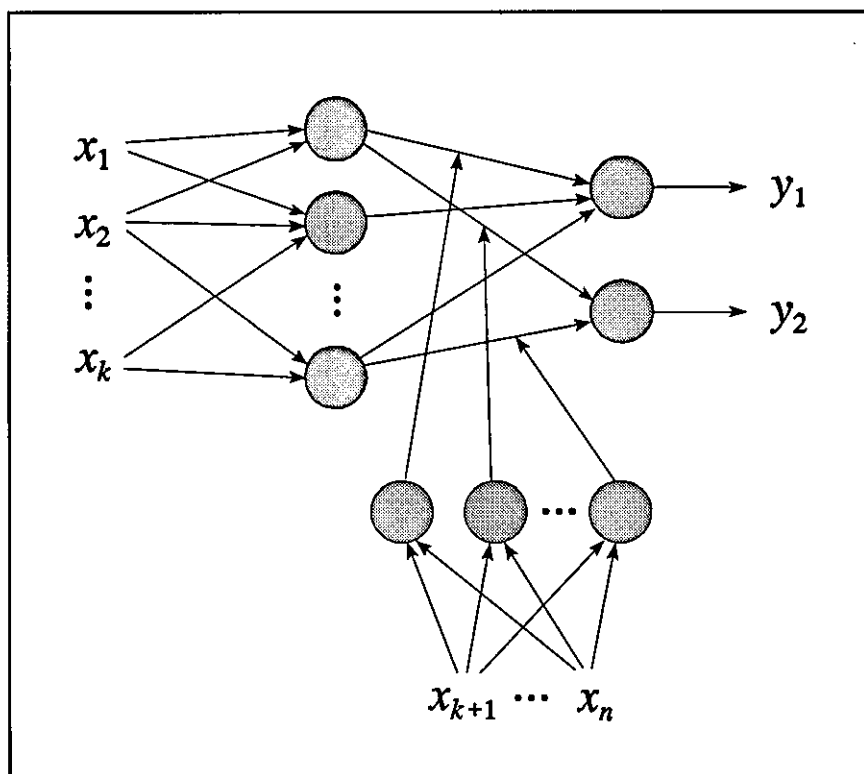


Figure 3.3-5: Example of a context sensitive network

A diagram of an example context sensitive network is shown in Fig. 3.3-5. The weight values for the links connecting to the output nodes are calculated by the subnetwork in the lower part of the figure. These weights are context sensitive, as their values vary with each pattern processed. Whereas the remaining links in the network possess conventional weights, whose values are constant throughout a training set but are subject to training. Clearly, context sensitive weights are themselves not adjusted by training, however, it has been shown [Yeung 1990] that the training of context sensitive networks can be accomplished by standard learning algorithms (which adjust only the values of the conventional weights).

3.4 LEARNING ALGORITHMS

To date, many dozens of learning algorithms have been formulated (for descriptions and reviews of relative merits see [Rumelhart & McClelland 1986, Anderson & Rosenfeld

1988, Nelson & Illingworth 1991, Hassoun 1995]). This section will detail the two most well-known algorithms. Both are supervised learning techniques and as such aim to minimise the error cost associated with the training patterns. The first algorithm examined, the Perceptron Rule, operates exclusively on discrete output single-layer networks, it is historically important but very limited in its application. The second, Backpropagation, overcomes many of the limitations experienced by the Perceptron Rule (and others), operates exclusively on networks with continuous activation functions, and is an example of a *gradient descent* method (see Subsection 3.4.2 for definition). These two algorithms are discussed here because of the Perceptron Rule's simplicity and Backpropagation's general applicability and gradient descent behaviour, furthermore, between them they demonstrate many of the important concepts in network training.

A third learning algorithm developed as part of the current work is termed Proportional Error Allocation (PERAL), and is presented in detail in Chapter 6.

Regardless of which learning algorithm is employed, it can be applied to a network in one of two ways. In *sequential training*, the network processes a single training pattern, then its weights are adjusted by the learning algorithm before the next pattern is processed. In *batch training*, the entire training set is processed before the network's weights are adjusted, using values that are averages determined by the learning algorithm from across the whole set. In either case, the processing of all the data in the training set and the associated training, is collectively known as an *epoch*.

In a sequential training, the order in which the training patterns are processed can affect the network's learning, thus the order in which the training patterns are considered is usually changed between epochs to minimise such ordering effects. Also, when training is terminated, the network obtained will clearly be disproportionately dependent on the last few patterns processed than on those earlier within the training set.

The aim of training is usually to minimise the error cost of each training pattern averaged throughout the entire set, thus batch training is the more intuitive, and mathematically rigorous, choice. However, the random disturbances to the learning caused by sequential training can increase the effectiveness of certain learning algorithms, allowing them to advance the state of a network past non-optimal weight values at which the individual errors in the set cancel each other out or otherwise fail to provide a learning impulse [Wasserman 1989]. Though sequential training can be faster, due to many weight changes per epoch rather than just one, the equations that govern both network processing and learning can usually be expanded into matrix form to provide rapid parallel calculation methods. Furthermore, batch training is more resilient to random (zero mean) noise in the training data due to the averaging out of its effects.

For ease of comprehension, the equations that follow refer to the learning algorithms' application in sequential mode. To obtain the equivalent batch equations, one needs only to expand the appropriate vectors into matrices with one row per training pattern, and divide the resultant weight adjustments by the number of patterns in the training set.

3.4.1 The Perceptron Rule

The Perceptron Rule [Rosenblatt 1958] is specific to single layer adaptive networks employing hard-limiting nodes, and, although rarely used today, is significant as being the earliest learning algorithm actually used with a real information processing network. Its limitations also allow one to better appreciate the more sophisticated, though usually more complex, learning algorithms.

Recalling Subsection 3.3.1, the operation of a single layer hard-limiter based network can be expressed as

$$y = \Gamma(xW)$$

where

y = network output vector, of size $1 \times o$

x = network input vector, of size $1 \times o$

W = weights matrix, of size $i \times o$

Γ = elementwise step function

and

i = number of input nodes

o = number of output nodes

The weights matrix may possess arbitrary values prior to training. For each epoch of training, the Perceptron Rule states that the adjustments to the weights matrix is given by

$$\Delta W = \eta x^T (t - y) \quad (3.4.1)$$

where

ΔW = the increases in the values of the weights matrix

η = learning rate (usually a constant scalar with a value less than one)

t = desired target output vector

The difference between the desired target values and the network output values clearly provide an error signal, with possible values of 1, 0 or -1. Thus it can be seen that each individual weight will have its value increased by the product of its corresponding input and error (averaged, in the case of batch training, over the set of patterns processed).

The reason why the Perceptron Rule is limited to single layer networks is clear since the

algorithm is driven by a measure of each node's error. This is simple to calculate for an output layer node, as the value it outputs can be compared to the corresponding target value, but there is no such measure for nodes in preceding layers.

Restricting a network to a single layer means that each network output is the result of just one single node. Therefore, for a network trained by the Perceptron Rule to be potentially capable of correctly modelling a system, it must fulfil the condition of *linear separability*. This states that for each output, it must be possible to draw a hyperplane through the i -dimensional space that contains the network's input patterns, separating the data points into the desired two classes.

Furthermore, it can be shown that, given linear separability, the Perceptron Rule will develop a weights matrix that correctly separates the classes in a *finite number* of training epochs, regardless of the initial values of the weights [Rosenblatt 1958].

3.4.2 Backpropagation

The invention of the Backpropagation algorithm played a large part in the resurgence of interest in adaptive networks that occurred in the 1980s. Backpropagation was the first systematic method for training multilayered networks, thus vastly increasing the complexity and therefore usefulness of trainable adaptive networks. The basic algorithm was first described by [Werbos 1974], was made popular by [Rumelhart et al. 1986], and draws upon the theory of error gradient descent, which was developed independently by [Amari 1967 & 1968, Bryson & Ho 1969, Werbos 1974, Parker 1985]. The premise of gradient descent learning is to formulate an error function for the network and then to minimise it locally over the space of possible weight settings. In particular, if the error function used is differentiable with respect to the network weights, (requiring the use of differentiable activation functions within the network nodes) gradient descent will naturally lead to a learning rule.

Almost universally in Backpropagation training, the error function minimised is the instantaneous (half) sum of squared error [Widrow & Hoff 1960, Hassoun 1995], given by

$$\varepsilon = \frac{1}{2} \sum_{i=1}^o (t_i - y_i)^2 \quad (3.4-2)$$

Consider the network in Fig. 3.4-1. The weights have been grouped into two matrices such that W_{jk}^A is the weight associated with the link from input j to node k in the hidden layer. Similarly, W_{kl}^B is the weight associated with the link from node k in the hidden layer to node l in the output layer. In the following formulation the two processing layers are also denoted as A , the hidden layer, and B , the output layer. As before, i is the

number of inputs, o the number of output nodes and h is defined as the number of hidden nodes.

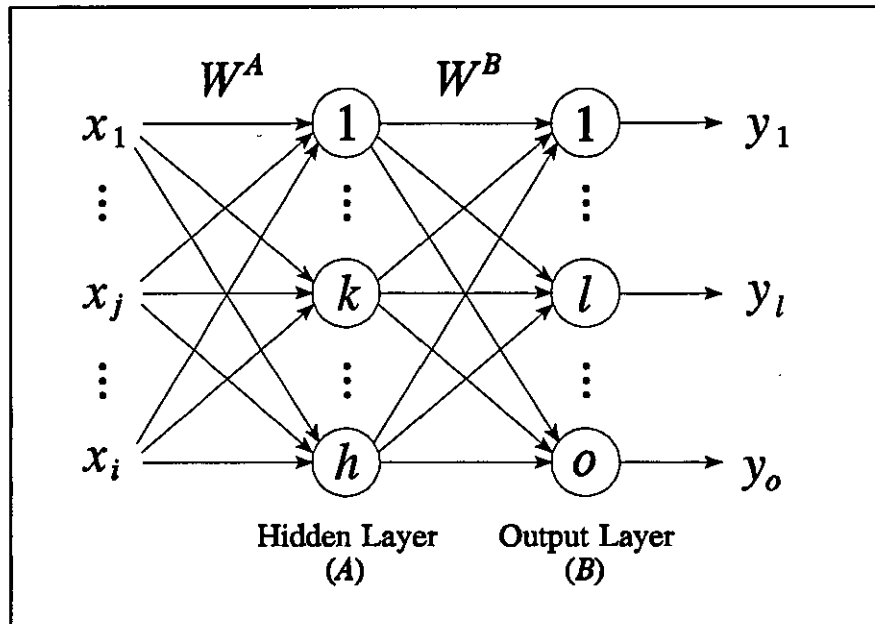


Figure 3.4-1: A generalised two layered network

Since the targets for the output layer nodes are explicitly stated, the output layer is the simplest for which to formulate the learning algorithm. In terms of the weight increment matrix, the error gradient descent method produces

$$\Delta W_{kl}^B = -\eta \frac{\partial \epsilon}{\partial W_{kl}^B} = \eta (t_l - y_l) f_B'(S_l^B) f_A'(S_k^A) \quad (3.4-3)$$

where

S^A = vector of summed weighted inputs for layer A

f_A = activation function for layer A

Note that the last function in Eqn. 3.4-3 is simply the output of the preceding layer. Also, although η is commonly defined to be constant throughout a network, it has been shown that better training performance is likely when the learning rate is successively lower for each layer after the first (such that, in this case, the learning rate for ΔW^A should be less than the learning rate for ΔW^B) [Rigler et al. 1991].

The provision of a learning algorithm for the hidden layer's weights (W^A) is not as straightforward as for the output layer, as there is no available set of target values for the hidden nodes. However, one may derive an algorithm by attempting to minimise the output layer's error. This amounts to propagating the output errors ($t - y$) back through the output layer toward the hidden layer, in an attempt to estimate targets for these nodes.

Therefore gradient descent is performed on the error function (Eqn. 3.4-2), but this time the gradient is calculated with respect to the hidden weights:

$$\Delta W_{jk}^A = -\eta \frac{\partial \varepsilon}{\partial W_{jk}^A} \quad (3.4-4)$$

Using the chain rule for partial derivatives, one may express the partial differentiation in the above as

$$\frac{\partial \varepsilon}{\partial W_{jk}^A} = \frac{\partial \varepsilon}{\partial f_A(S_k^A)} \frac{\partial f_A(S_k^A)}{\partial S_k^A} \frac{\partial S_k^A}{\partial W_{jk}^A} \quad (3.4-5)$$

where

$$\frac{\partial S_k^A}{\partial W_{jk}^A} = x_j \quad (3.4-6)$$

and

$$\begin{aligned} \frac{\partial \varepsilon}{\partial f_A(S_k^A)} &= \frac{\partial}{\partial f_A(S_k^A)} \left[\sum_{i=1}^o (t_i - f_B(S_i^B))^2 \right] \\ &= - \sum_{i=1}^o (t_i - f_B(S_i^B)) \frac{\partial f_B(S_i^B)}{\partial S_k^A} \\ &= - \sum_{i=1}^o (t_i - y_i) f_B'(S_i^B) W_{ki}^B \end{aligned} \quad (3.4-7)$$

Substituting Eqn.s 3.4-6 and 3.4-7 back into Eqn. 3.4-5, and using Eqn. 3.4-4, the desired learning algorithm is obtained:

$$\Delta W_{jk}^A = \eta \left[\sum_{i=1}^o (t_i - y_i) f_B'(S_i^B) W_{ki}^B \right] f_A'(S_k^A) x_j \quad (3.4-8)$$

By comparing the above with Eqn. 3.4-3, it is clear that the summed terms within the square bracket represent an estimate of the difference between the desired and actual output of the hidden layer. The Backpropagation equations can therefore be generalised for similar networks of any number of layers. To achieve this, A and B are generalised to denote any two consecutive layers, followed, in the case of B not being the output layer, by a further layer C . Then, taking advantage of matrix algebra, the Backpropagation algorithm for the weights matrix preceding layer B can be expressed as

$$\Delta W^B = \eta f_A'(S^A)^T \delta^B \quad (3.4-9)$$

where

$$\delta^B = \begin{cases} f_B'(S^B) * (t - y) & \text{if } B \text{ is the output layer} \\ f_B'(S^B) * (W^C \delta^C)^T & \text{otherwise} \end{cases}$$

and $*$ denotes the element by element multiplication of two vectors/matrices.

Clearly, when B is the first processing layer of a network, the output of the previous layer, $f_A(S^A)$, is simply the network inputs, x .

Due to the need for differentiable activation functions, discontinuous functions, such as hard-limiters, cannot be used. Sigmoidal functions, however, are commonly used and usually have derivatives which can be expressed in terms of the original function. For example the logistic function

$$f(S_i) = \frac{1}{1 + e^{-S_i}} \quad , \quad f'(S_i) = f(S_i)(1 - f(S_i))$$

and the hyperbolic tangent function

$$f(S_i) = \tanh(S_i) \quad , \quad f'(S_i) = 1 - f^2(S_i)$$

Simpler still is the case of linear combiner functions;

$$f(S_i) = S_i \quad , \quad f'(S_i) = 1$$

As has already been stated, the universal use of linear combiner nodes within a network guarantees that the error hypersurface (which is a function of the network weights) will be smooth and possess a single minima which represents the network's optimal state. Thus, an error gradient descent method such as Backpropagation, employing an infinite number of infinitely small iterations, will always tend towards this minima, regardless of the initial values of the network weights. For practical application, a suitable finite iteration size (related to the learning coefficient) can be found from trial and error or from automatic algorithms such as proposed by [Jacobs 1988, Chen & Mars 1990, Le Cun et al. 1993]. In contrast, due to the localised effect of sigmoidal functions, a network employing such activation functions will have a "lumpy" and irregular error hypersurface. The most significant aspect of this is that Backpropagation may cause the network weights to tend toward one of the local (non-optimal) minima instead of the global (optimal) one. Therefore, the state into which the network settles will, in this case, depend on the initial values of the network's weights [Kolen & Pollack 1990]. Local minima are a particular example of the more general problem of *flat-spots*, areas of the error hypersurface with near zero gradient. As both the size and direction of the Backpropagation learning increment are related to this gradient, flat-spots of all types cause standard Backpropagation learning to become unacceptably slow [Fahlman 1988]. Clearly, the more nonlinear activation functions present in a network, the more convoluted the error hypersurface with a greater number of flat-spots within it. Studies of error hypersurface shapes are presented in [Sontag & Sussmann 1989, Chen et al. 1993, Liang et al. 1995].

Numerous variants of the standard Backpropagation algorithm have been proposed

[Stornetta & Huberman 1987, Fahlman 1988, Hagiwara 1990, Li 1990], many of which are intended to learn faster and/or allow training to evolve a network out of a non-optimal state and across flat-spots. However, most of these variations' enhancements are gained at the expense of the standard Backpropagation's mathematical pedigree and gradient descent behaviour [Bishop 1995].

3.5 ADAPTIVE NETWORKS AND MANIPULATOR DYNAMICS

Adaptive networks have been applied to almost all areas of robotics, including image processing, mobile robot navigation, kinematics, and of course, dynamics. One of the early uses of an adaptive network was the motion control of a manipulator using the *Cerebellar Model Articulation Controller* (CMAC) [Albus 1975], which incorporated look-up tables in the determination of suitable control torques. Although adaptive networks have been popular tools for the enhancement of manipulator control for some time, resulting in many differing schemes being proposed, the work done to date can be split into two fundamentally different approaches:

Some researchers have attempted to train a network to directly model a manipulator's problem-space-wide dynamics [Kawato et al. 1987, Psaltis et al. 1987, Miyamoto et al. 1988, Bassi & Bekey 1989, Kuschewski et al. 1993, Sillitoe et al. 1994]. As can be seen from Chapter 2, the complete inverse dynamics of multi-jointed manipulators can be extremely complicated, requiring a vast amount of training data to representatively span the problem space. Attempts at learning the inverse dynamics of manipulators with more than three degrees of freedom, from empirical motion data, have yet to produce acceptable results throughout the whole problem space. However, this approach is still the subject of much work as the successful approximation of a manipulator's inverse dynamics obviates the need for on-line training and allows the development of near ideal controllers.

Other researchers have employed adaptive networks as localised inverse dynamics models or as adjustment terms to a standard controller (either non-model-based or using estimates of the dynamics coefficients derived by other means) [Seraji 1989, Ishiguro et al. 1992, Jacobs & Jordan 1993, Zomaya & Nabhan 1993, Whitcomb et al. 1993, Yamada & Yabuta 1993, Sanger 1994], such that training is either performed continuously as the manipulator moves or else performed on data from a specific trajectory that is localised within the problem space. This approach deliberately tackles a very much simpler problem than that of system-wide inverse dynamics identification, as it only needs to model localised dynamics. There are, however, several disadvantages with these techniques. Firstly, if on-line adaption is used to adjust a control algorithm, then the resultant controller is liable to be non-ideal, that is, the relationship between the desired

and actual movement will be nonlinear and the controllers performance will vary over the space. Furthermore, it may not be possible to train the network quickly enough to cope with rapid movement across the problem space. Optimising control for a specific trajectory (to the detriment of others) is clearly suitable for repetitive motion tasks such as may be required by a production manipulator, but is of limited use elsewhere.

Several ideas expressed in the literature are of significant note, and, in the author's opinion, are of great potential in the modelling of manipulator inverse dynamics. To tackle the problem of system complexity, [Bassi & Bekey 1989] suggested the decomposition of the network representation of the inverse dynamics, such that individual subnetworks model separate terms within the Euler-Lagrange equation. A related proposal was made for localised network controllers by [Jacobs & Jordan 1993], suggesting the training of several networks, each for a different trajectory/payload, and switching between them. To increase network training rate and accuracy, [Armstrong 1989] developed a technique which transforms a given training data trajectory to one locally optimised to cause the minimum error in the values of identified parameters due to experimental noise. Importantly, [Craig 1988] noted that inverse dynamics coefficients are linearly related (by the generalised position, velocity and acceleration vectors) to the torque/force vector, and that this could be exploited in adaptive coefficient identification. Lastly, several researchers have noted the applicability of context sensitive networks (see Subsection 3.3.5) to modelling manipulator inverse dynamics, given that they are known to be configuration dependent.

The majority of networks currently proposed for modelling problem-space-wide inverse dynamics consist of a large numbers of nodes with continuous and locally sensitive activation functions, commonly either radial basis or sigmoidal functions. However, the vast size of such networks is often due to reliance upon the property of universal approximation (where a given network is capable of approximating any continuous system if it contains a large enough number of suitable nodes) and belies the lack of analysis performed on the inverse dynamics. For example, as has been stated previously in Subsection 2.5.1, manipulator dynamics are known to be discontinuous due to the effect of dry and static friction, and that this discontinuity is often significant. Furthermore, in many networks, the values of revolute joint positions are directly fed into an adaptive network as inputs (without preprocessing) despite this clearly making it improbable that the trained network will provide similar outputs for pairs of position values such as zero and 2π radians.

The major challenge, at present, in the area of adaptive network modelling of inverse dynamics, and indeed of many other complex systems, is that of obtaining an accurate representation of the entire system by training a network on a subset of it. In other words, the problem is one of generalisation, or rather the lack of it routinely displayed by

networks trained on empirical data. The next section discusses generalisation and two techniques currently used to enhance it in networks subject to supervised learning.

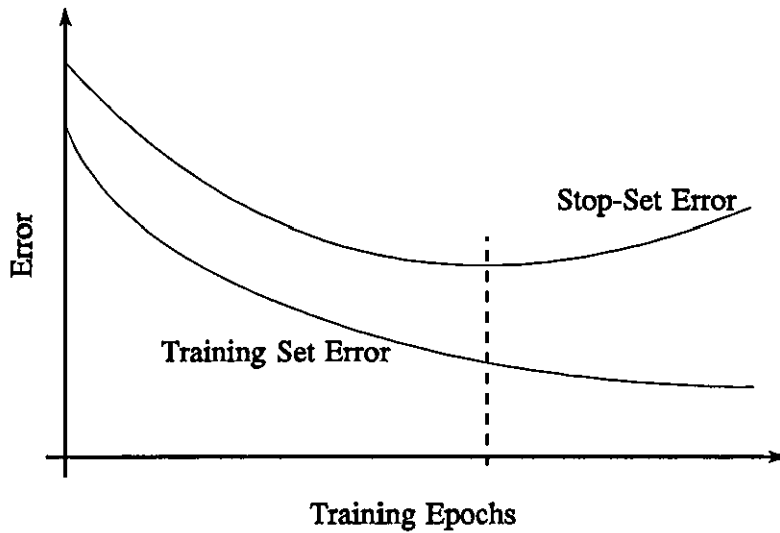
3.6 OVERTRAINING AND GENERALISATION

When an adaptive network is trained using a supervised learning technique, the weights within the network are adjusted in order to decrease the error produced by the training patterns. The aim of such training, however, is for the network to become a better model of the generalised system of which the training set is a sample. In particular, a training set can never represent all of the points within a problem space with analogue inputs or outputs and will commonly contain some degree of spurious information due to sampling, such as measurement errors. Assuming that the network has at least as many nodes as would be required to be capable of accurately approximating the system, then at some point during training, often in the later stages once the more obvious features of the system have been learnt, the network will start to reduce the training set error by learning these idiosyncrasies of the training data that do not exist in the system in general. This phenomenon is termed *overtraining* or *overfitting*, an example and discussion of which is presented in [Chauvin 1990]. The consequences of overtraining, and a means of largely avoiding it, are fully discussed as part of the concept of network flexibility in Chapter 4, however, at present, overtraining is mitigated by the use of either *stop sets* or *network pruning*.

3.6.1 Stop Sets

This method involves using two sets of data, a training set and a validation set (or stop set) [Hecht-Nielsen 1990]. The network is trained as normal using the training set, but the network error produced by the stop set is noted periodically. The stop set error is assumed to be a valid indicator of the network's performance at modelling the problem system in general, and so training is terminated when the stop set error ceases to decrease between training epochs. A representation of overtraining is shown in Fig. 3.6-1, complete with a comparative stop set error curve.

The ability of the stop set to accurately represent the system in general is clearly a major factor in the effectiveness of the stop set technique. If the stop set represents only a subsection of the whole system, then it can only provide information on the network's processing performance for that subsection of the problem space. This may cause the termination of learning, as indicated by the increase in stop set error cost, to be either earlier or later than optimal.



*Figure 3.6-1: Overtraining to the detriment of generalised performance
(The optimal point to cease training, as indicated by the
stop set, is marked by the vertical dashed line)*

3.6.2 Network Pruning

An alternative method of limiting overtraining is to restrict the ability of the network to learn the spurious correlations in the training data. This can be done by finding the least complex network that will adequately fit the training data, assuming that the idiosyncrasies of the training set used are less significant than the system response characteristics. Training many different network configurations would clearly be very time consuming, so a less arduous method is to train a large network (with many nodes and links) and to then apply some form of pruning technique.

The simplest kind of pruning technique tests the sensitivity of the training set error to the removal of each network link. If the removal of a link produces a marked increase in the error it is restored, otherwise it is permanently removed. If a node has all the links leading from it removed, then it is itself removed, along with all of the links leading to it. Once all of the insensitive links have been removed it is assumed that what is left is the minimum complexity network required to model the generalised system.

A survey of different pruning techniques is presented in [Reed 1993].

The basics of adaptive network theory have been introduced in this chapter, with particular reference to the modelling of analogue output systems, specifically inverse dynamics. The three major node types, hard-limiters, linear combiners and locally sensitive nodes, have been examined and the properties with which they endow a network reviewed. In particular, the effect of different activation functions on the form of network output error hypersurface, a function of the network weights, was discussed. It was noted that continuous activation functions produce continuous error hypersurfaces, but that with the significant exception of linear combiners, these surfaces are often complicated, hard to visualise or analysis and contain multiple local minima. The benefits of input preprocessing and context sensitivity in networks were shown, both of which extend the types of problems that adaptive networks can be applied to.

The process of network training was illustrated using the Perceptron Rule and Backpropagation learning algorithms. In particular, Backpropagation was noted as an example of an error gradient descent method, applicable to a large variety of network types, but prone to slow or unsuccessful learning due to flat-spots and local minima when applied to networks with nonlinear activation functions.

The networks employed by other researchers in the area of manipulator motion control were reviewed, in general they can be separated into those that attempt to model the entire problem-space-wide inverse dynamics, and those that learn localised relationships. The relative benefits of both were discussed, with the former being presented as the ideal case if successful training can be achieved, however, to date, this approach has failed to provide a generalised model for all but the simplest of manipulators. Important observations and ideas from the current literature and relevant to the modelling of inverse dynamics with adaptive networks were also reported. These included the functional decomposition of networks, selecting training data with high robustness to noise, the linear relationship between inverse dynamics coefficients and the generalised force/torque, and the suitability of context sensitivity to modelling inverse dynamics.

The often minimal degree of analysis performed on the form of manipulator inverse dynamics prior to the attempted modelling of such a system was commented upon, with reference to some of the deficiencies commonly found in such networks.

The reported poor performance of most analogue output networks' solution to accurately generalise between or beyond the data with which the networks were trained was noted. In particular, the problems associated with overfitting the network model were discussed and the use of stop sets and network pruning as a means of reducing overfitting were explained.

It is clear that the behaviour of network models undergoing supervised learning is not fully understood at present, this is not surprising given the high dimensionality and nonlinearity of most networks. However, it is apparent that this process must be more closely examined if the problem of poor generalisation is to be properly addressed. The analysis performed in the following chapter sheds new light on the general problem of training analogue output networks and the reasons for their degree of generalisation.

REFERENCES AND FURTHER READING

- Albus, J.S., "A new approach to manipulator control: the cerebellar model articulation controller (CMAC)" & "Data storage in the cerebellar model articulation controller (CMAC)," *J. Dyn. Sys., Meas., Contr.*, vol. 97, pp. 220-227 & 228-233, 1975.
- Amari, S.-I., "Theory of adaptive pattern classifiers," *IEEE Trans on Electronic Computers*, vol. EC-16, pp. 299-307, 1967.
- Amari, S.-I., "A universal theorem on learning curves," *Neural Networks*, vol. 6, no. 2, pp. 161-166, 1993.
- Anderson, J.A., and Rosenfeld, E, Eds., *Neurocomputing: Foundations of Research*. Cambridge, MA: M.I.T. Press 1988.
- Armstrong, B., "On Finding Exciting Trajectories for Identification Experiments Involving Systems with Nonlinear Dynamics," *Int. Journal of Robotics Research*, vol. 8, no. 6, pp. 28-48, Dec. 1989.
- Baldi, P., "Computing with arrays of bell-shaped and sigmoid functions," *Proc. of Neural Information Processing Systems 3* (Denver). San Mateo, Calif.: Morgan Kaufmann, pp. 735-742, 1990.
- Barron, A.R., "Predicted squared error: A criterion for automatic model selection," in *Self-Organizing Methods in Modelling*, S.J. Farlow, Ed., New York: Marcel Dekker Inc., pp. 87-103, 1984.
- Barto, A.G., Sutton, R.S., and Brouwer, P.S., "Associative search network: A reinforcement learning associative memory," *Biological Cybernetics*, vol. 40, pp. 201-211, 1981.
- Barto, A.G., Sutton, R.S., and Anderson, C.W., "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-13, pp. 834-846, 1983.
- Bassi, D.F., and Bekey, G.A., "Decomposition of neural network models of robot dynamics: a feasibility study," *Soc. for Computer Sim. Int. Conf. on Simulation and AI*, vol. 20, no. 3, pp. 8-13, 1989.
- Bishop, C.M., *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.
- Broomhead, D.S., and Lowe, D., "Multivariate functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321-355, 1988.
- Bryson, A.E., and Ho, Y.-C., *Applied Optimal Control*. New York: Blaisdell, 1969.
- Caelli, T.M., Squire, D.McG., and Wild, P.J., "Model-Based Neural Networks", *Neural Networks*, vol. 6, pp. 613-625, 1993.

- Carpenter, G.A., and Grossberg, S., "A massively parallel architecture for a self-organising neural pattern recognition machine," *Computer Vision, Graphics and Image Pro.*, vol. 37, pp. 54-115, 1983.
- Carpenter, G.A., and Grossberg, S., "Art 2: Self-organization of stable category recognition codes for analog output patterns," *Applied Optics*, vol. 26, pp. 4919-4930, Dec. 1987.
- Carpenter, G.A., and Grossberg, S., "Art 3 hierarchical search: Chemical transmitters in self-organizing pattern recognition architectures," *Proc. Int. Joint Conf. on Neural Networks*, Washington DC, vol. 2, pp. 30-33, Jan. 1990.
- Chen, J.R., and Mars, P., "Stepsize variation for accelerating the back propagation algorithm," *Proc. Int. Joint Conf. on Neural Networks*, Washington DC, vol. I, pp. 601-604, 1990.
- Chen, A.M., Lu, H. and Hecht-Nielsen, "On the geometry of feedforward neural network error surfaces," *Neural Computation*, vol. 5, pp. 910-927, 1993.
- Craig, J.J., *Adaptive Control of Mechanical Manipulators*. Reading, MA: Addison-Wesley, 1988.
- Cun, Y. le, "A theoretical framework for back-propagation," *Proc. 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton and T. Sejnowski, Eds., pp. 21-28. San Mateo, CA: Morgan Kaufmann, 1988.
- DeMers, D., and Cottrell, G., "Non-linear dimensionality reduction," *Proc. of Advances in Neural Info. Processing Systems 5*, (Denver). Hanson, Cowan & Giles, Eds., San Mateo, CA: Morgan Kaufmann, pp. 550-587, 1992.
- Faddeev, D.K., and Faddeeva, V.N., *Computational Methods of Linear Algebra*. San Francisco: Freeman and Co., 1963.
- Fahlman, S.E., "Faster-learning variations on back-propagation: an empirical study," Touretzky, Hinton and Sejnowski (Eds.), *Proc. Connectionist Models Summer School*, San Mateo, CA: Morgan Kaufmann, pp. 38-51, 1988.
- Fukushima, K., "Cognitron: A self-organizing multilayered neural network," *Biolog. Cybernetics*, vol. 20, pp. 121-136, 1975.
- Fukushima, K., "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biolog. Cybernetics*, vol. 36, pp. 193-202, 1980.
- Girosi, F., and Poggio, T., "Representation properties of networks: Kolmogorov's theorem is irrelevant," *Neural Computation*, vol. 1, no. 4, pp. 465-469.
- Grossberg, S., "Adaptive pattern classification and universal recoding," *Biolog. Cybernetics*, vol. 20, pp. 121-202, 1976.
- Guez, A., and Ahmad, Z., "Solution to the inverse problem in Robotics by Neural Networks," *Proc. IEEE Int. Conf. on Neural Networks*, San Diego CA, vol. II, pp. 625-631, July 1988.
- Gullapalli, V., "A stochastic reinforcement learning algorithm for learning real-valued functions," *Neural Networks*, vol. 3, pp. 671-692, 1990.
- Hassoun, M.H., *Fundamentals of artificial neural networks*. Cambridge, Mass.: MIT Press, 1995
- Harp, S.A., Samad, T., and Guha, A., "Designing application-specific neural networks using the genetic algorithms," *Proc. Advances in Neural Info. Pro. Sys. 2*, (Denver)

- Touretzky, Ed., San Mateo, Calif.: Morgan Kaufmann, pp. 447-454, 1989.
- Hartman, E.J., Keeler, J.D., and Kowalski, J.M., "Layered neural networks with Gaussian hidden units as universal approximators," *Neural Computation*, vol. 2, pp. 210-215, 1990.
- Hebb, D.O., *The Organization of Behavior*. New York: Wiley, 1949.
- Hecht-Neilsen, R., *Neurocomputing*. Reading, MA: Addison-Wesley, 1990.
- Hagiwara, M., "Novel back-propagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection," *Int. Joint Conf. on Neural Networks*, vol. I, pp. 625-630, 1990.
- Hinton, G.E., Sejnowski, T.J., and Ackley, "Boltzmann machines: Constraint satisfaction networks that learn," Tech. Rep. CMU-CS-84-119, Carnegie-Mellon University, Dept. of Computer Science, 1984.
- Hinton, G.E., and Sejnowski, T.J., "Learning and relearning in Boltzmann machines," *Parallel Distrib. Processing*, vol. 1, ch. 7, D.E. Rumelhart and J.L. McClelland, Eds. Cambridge, MA: M.I.T Press, 1986.
- Hopfield, J.J., "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci.*, vol. 79, pp. 2554-2558, Apr. 1982.
- Hopfield, J.J., "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci.*, vol. 81, pp. 3088-3092, May 1984.
- Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989.
- Hornik, K., Stinchcombe, M., and White, H., "Universal approximations of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Networks*, vol. 3, no. 5, pp. 551-560, 1990.
- Hornik, K., "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 6 no. 8, pp. 1069-1072, 1991.
- Hu, M.J.C., *Application of the Adaline System to Weather Forecasting*. Thesis, Tech. Rep. 6775-1, Stanford Electron. Labs., Stanford, CA, June 1964.
- Hush, D.R., Salas, J.M., and Horne, B., "Error surfaces for multi-layer perceptrons," *Proc. Int. Joint Conf. on Neural Networks*, (Seattle). New York: IEEE, vol. I, pp. 759-764, 1991.
- Ishiguro, A., Furuhashi, T., Okuma, S., and Uchikawa, Y., "A Neural Network Compensator for Uncertainties of Robotics Manipulators," *IEEE Trans. on Industrial Electronics*, vol. 39, no. 6, pp. 565-570, 1992.
- Jacobs, R.A., "Increased rates of convergence through learning rate adaption," *Neural Networks*, vol. 1, pp. 295-308, 1988.
- Jacobs, R.A., and Jordan, M.I., "Learning Piecewise Control Strategies in a Modular Neural Network Architecture," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, no. 2, pp. 337-345, 1993.
- James, W., *Psychology (Briefer Course)*, New York: Holt, 1890.
- Karhunen, J., "Optimization criteria and nonlinear PCA neural networks," *Proc. IEEE Int. Conf. on Neural Networks*, (Orlando). New York: IEEE, vol. II, pp. 1241-1246, 1994.
- Kawato, M., Uno, Y., Isobe, M., and Suzuki, R., "A Hierarchical model for voluntary

- movement and its application to robotics," *Proc. IEEE First Int. Conf. on Neural Networks*, San Diego CA, vol. IV, pp. 573-582, June 1987.
- Kohonen, T., "Self-organized formation of topologically correct feature maps," *Biolog. Cybernetics*, vol. 43, pp. 59-69, 1982.
- Kohonen, T., *Self-Organization and Associative Memory*. New York: Springer-Verlag, 2nd Ed., 1988.
- Kolen, J.F., and Pollack, J.B., "Back propagation is sensitive to initial conditions," *Proc. Advances in Neural Info. Pro. Systems 3*, (Denver). Lippmann, Moody & Touretzky, Eds., San Mateo, Calif.: Morgan Kaufmann, pp. 860-867, 1990.
- Kolman, B., *Introductory Linear Algebra with Applications*. New York: Macmillan, 4th Ed., 1988.
- Kosko, B., "Adaptive bidirectional associative memories," *Appl. Optics*, vol. 26, pp. 4947-4960, Dec. 1987.
- Kuschewski, J.G., Hui, S., and Stanislaw, H.Z., "Application of Feedforward Neural Networks to Dynamical System Identification and Control," *IEEE Trans. on Control Systems Tech.*, vol. 1, no. 1, pp. 37-49, 1993.
- Le Cun, Y., Simard, P.Y., and Pearlmutter, B., "Automatic learning rate maximization by on-line estimation of the Hessian's eigenvectors," Hanson, Cowan and Giles (Eds.), *Advances in Neural Info. Processing Systems*, San Mateo, CA: Morgan Kaufmann, vol. 5, pp. 156-163, 1993.
- Lee, S., and Kil, R., "Multilayer feedforward potential function networks," *Proc. of the 2nd IEEE Int. Conf. on Neural Networks*, vol. 4, pp. 588-599, 1988.
- Li, S., "An optimized backpropagation with minimum norm weights," *Proc. Int. Joint Conf. on Neural Networks*, San Diego, vol. I, pp. 697-702, 1990.
- Liang, X., Wang, X., and Bode, J., "Research on Error Hypersurfaces in Multi-Output Multilayer Perceptrons," *Proc. Neural, Parallel and Scientific Computations*, vol. 1, pp. 273-276, 1995.
- Light, W.A., "Ridge functions, sigmoidal functions and neural networks," *Proc. of Approximation Theory VII*, E.W. Cheeny, C.K. Chui and L.L. Schumaker, Eds., New York: Academic Press, pp. 163-206, 1992
- Lippman, R.P., "An Introduction to computing with neural nets," *IEEE ASSP Mag.*, April 1987.
- Lucky, R.W., "Automatic equalization for digital communication," *Bell Syst. Tech. J.*, vol. 44, pp. 547-588, Apr. 1965.
- Lucky, R.W., et al., *Principles of Data Communication*. New York: McGraw-Hill, 1968.
- Malsburg, C. von der, "Self-organizing of orientation sensitive cells in the striate cortex," *Kybernetik*, vol. 14, pp. 85-100, 1973.
- McCulloch, W.S., and Pitts, W., "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5 pp. 115-133, 1943.
- Miller, W.T. III, "Sensor-based control of robotic manipulators using a general learning algorithm." *IEEE Robotics and Automat.*, vol. RA-3, pp. 157-165, Apr. 1987.
- Minsky, M., "Neural nets and the brain - Model problem," Doctoral Dissertation, Princeton University, Princeton NJ, 1954.
- Miyamoto, H., Kawato, M., Setoyama, T., and Suzuki, R., "Feedback Error Learning Neural Networks for Trajectory Control of a Robotic Manipulator," *Neural*

- Networks*, vol. 1, pp. 251-265, 1988.
- Moody, J., and Darken, C., "Learning with localized receptive fields," *Proc. of the Connectionist Models Summer School*. San Mateo, Calif.: Morgan Kaufmann, pp. 174-185, 1988.
- Moody, J., and Darken, C., "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp 281-294, 1989.
- Moore, B., "Art 1 and pattern clustering," *Proc. 1988 Connectionist Models Summer School*, D. Touretzky, G. Hinton and T. Sejnowski, Eds., pp. 174-185, San Mateo, CA: Morgan Kaufmann, June 1988.
- Nelson M.M., and Illingworth, W.T., *A Practical Guide to Neural Nets*. Reading, Mass: Addison-Wesley, 1991.
- Neumann, J. Von, "The general and logical theory of automata," in L.A. Jeffress, Ed., *Cerebral Mechanisms in Behavior*, New York: Wiley, pp. 1-41, 1951.
- Neumann, J. Von, *The Computer and the Brain*, Yale University Press, 1958.
- Nilsson, N., *Learning Machines*. New York: McGraw-Hill, 1965.
- Niranjan, N., and Fallside, F., "Neural Networks and Radial Basis Functions in Classifying Static Speech Patterns," Tech. Report CUEDIF-INFENG17R22, Engineering Dept., Cambridge University, 1988.
- Park, J., and Sandberg, I.W., "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246-257, 1991.
- Park, J., and Sandberg, I.W., "Approximation and radial-basis-function networks," *Neural Computation*, vol. 5, no. 2, pp. 305-316, 1993.
- Parker, D., "Learning-logic," Invention Rep. S81-64, File 1, Office of Tech. Licensing, Stanford University, Stanford, CA, Oct. 1982.
- Parker, D., "Learning-logic," Tech. Rep. TR-47, Center for Computational Research in Economics and Management Science, M.I.T, April 1985.
- Poggio, T., and Girosi, F., "A Theory of Networks for Approximation and learning," A.I. Memo 1140, MIT, Cambridge, Mass., 1989.
- Poggio, T., and Girosi, F., "Regularization algorithms for learning that are equivalent to multilayer networks," *Science*, vol. 247, pp. 978-982, 1990.
- Powell, M.J.D., "Radial basis functions for multivariable interpolation: a review," Mason and Cox (Eds.), *Algorithms for Approximation*. Oxford: Clarendon Press, pp. 143-167, 1987.
- Psaltis, D., Sideris, A., and Yamamura, A., "Neural Controllers," *Proc. IEEE Int. Conf. on Neural Networks*, pp. 551-558, 1987.
- Reed, R., "Pruning Algorithms — A Survey," *IEEE Trans. on Neural Networks*, vol. 4 no. 5, pp. 740-747, 1993.
- Rigler, A.K., Irvine, J.M., and Vogl, T.P., "Rescaling of variables in backpropagation learning," *Neural Networks*, vol. 4, pp. 225-229, 1991.
- Rosenblatt, F., "The Perceptron: A probabilistic model for information storage and organisation in the brain," *Psychol. Rev.*, vol. 65, pp. 386-408, 1958.
- Rosenblatt, F., *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, DC: Spartan Books, 1962.
- Rumelhart, D.E., Hinton, G.E., and Williams, R.J., " Learning internal representations

- by error propagation," ICS Report 8506, Institute for Cognitive Science, University of California at San Diego, La Jolla, CA, Sept. 1985.
- Rumelhart, D.E., Hinton, G.E., and Williams, R.J., " Learning internal representations by error propagation," *Parallel Distributed Processing*, vol. 1, ch. 8, D. Rumelhart and J. McClelland, Eds., Cambridge, MA: M.I.T. Press, 1986.
- Rumelhart, D.E., and McClelland, J.L., *Parallel Distributed Processing*. Cambridge, MA: M.I.T. Press, 1986.
- Sanger, T.D., "Neural Network Learning Control of Robot Manipulators Using Gradually Increasing Task Difficulty," *IEEE Trans. on Robotics and Automation*, vol. 10, no. 3, pp. 323-333, 1994.
- Scott, D.W., *Multivariate Density Estimation: Theory, Practice, and Visualization*. New York: John Wiley, 1992.
- Seraji, H., "Decentralized Adaptive Control of Manipulators: Theory, Simulation, and Experimentation," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 2, pp. 183-201, 1989.
- Sillitoe, I.P.W., Hay, R.J., and Mulvaney, D.J., "A Comparison of Neural Network Structures for use in the Learning of Robotic Manipulator Dynamics," *Proc. Int. Conf. on Systems Engineering*, 1994.
- Sira-Ramírez, H.J., and Stanislaw, H.Z., "The Adaption of Perceptrons with Applications to Inverse Dynamics Identification of Unknown Dynamic Systems," *IEEE Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 634-643, 1991.
- Sondhi, M.M., "An adaptive echo canceller," *Bell Syst. Tech. J.*, vol. 46, pp. 497-511, March 1967.
- Sontag, E.D., and Sussmann, H.J., "Backpropagation can give rise to spurious local minima even for networks without hidden layers," *Complex Systems*, vol. 3, pp. 91-106, 1989.
- Specht, D.F., *Generation of Polynomial Discriminant Functions for Pattern Recognition*. PhD thesis, Tech. Rep. 6764-5, Stanford Elec. Labs., Stanford, CA, May 1966.
- Specht, D.F., "Vectorcardiographic diagnosis using the polynomial discriminant method of pattern recognition," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 308-319, June 1967.
- Stark, L., Okajima, M., and Whipple, G.H., "Computer pattern recognition techniques: Electrocardiographic diagnosis," *Commun. Ass. Comput. Mach.*, vol. 5, pp. 527-532, Oct. 1962.
- Steinbuch, K., and Piske, V.A.W., "Learning matrices and their applications," *IEEE Trans. Electron. Comput.*, vol. EC-12, pp. 840-862, Dec. 1963.
- Stornetta, W.S., and Huberman, B.A., "An Improved Three-Layer Back-Propagation Algorithm", *Proc. IEEE Int. Conf. on Neural Networks*, San Diego, pp. 637-644, 1987.
- Strang, G., *Linear Algebra and its Applications*. Fort Worth: Harcourt Brace Jovanovich, 3rd Ed., 1988.
- Talbert, L.R., et al., "A real-time adaptive speech-recognition system," Tech. Rep., Stanford University, 1963.
- Tsetlin, M.L., *Automaton theory and modeling of biological systems*. New York: Academic Press, 1973.
- Villiers, J., and Barnard, E., "Backpropagation neural nets with one or two hidden layers,"

- IEEE Trans. on Neural Networks*, vol. 4, no. 1, pp. 136-141., 1993.
- Wasserman, P.D., *Neural Computing: theory and practice*, New York: Van Nostrand Reinhold, 1989.
- Webb, A.R., "Functional approximation by feed-forward networks: a least-squares approach to generalisation," *IEEE Trans. on Neural Networks*, vol. 5, no. 3, pp. 363-371, 1994.
- Werbos, P., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, Aug. 1974.
- Whitcomb, L.L., Rizzi, A.A., and Koditschek, D.E., "Comparative Experiments with a New Adaptive Controller for Robot Arms," *IEEE Trans. on Robotics and Automation*, vol. 9, no. 1, pp. 59-70, 1993.
- White, H., "Learning in artificial neural networks: A statistical perspective," *Neural Networks*, vol. 1, pp. 425-464, 1989.
- Widrow, B., and Hoff, M.E., "Adaptive switching circuits," Tech. Rep. 1553-1, Stanford Electronics Labs., Stanford, CA June 30, 1960.
- Widrow, B., "Generalization and information storage in networks of adaline neurons," *Self-Organizing Systems 1962*, M. Yovitz, G. Jacobi and G. Goldstein, Eds. Washington, DC: Spartan Books, pp. 435-461, 1962.
- Widrow, B., "Bootstrap learning in threshold logic systems," Presented at the American Automatic Control Council (Theory Committee), IFAC Meeting, London, England, June 1966.
- Widrow, B., Mantey, P., Griffiths, L., and Goode, B., "Adaptive antenna systems," *Proc. IEEE*, vol. 55, pp. 2143-2159, Dec. 1967.
- Widrow, B., Gupta, N.K., and Maitra, S., "Punish/reward: Learning with a critic in adaptive threshold systems," *IEEE Trans. Syst., Man, Cybernetics*, vol. SMC-3, pp. 455-465, Sept. 1973.
- Widrow, B., "The original adaptive neural net broom-balancer," *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 351-357, Phila., PA, May 1987.
- Widrow, B., Winter, R.G., and Baxter, B., "Learning phenomena in layered neural networks," *Proc. 1st IEEE Intl. Conf. on Neural Networks*, vol. 2, pp. 411-429, San Diego, CA June 1987.
- Yamada, T., and Yabuta, T., "Dynamic System Identification Using Neural Networks," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, no. 1, pp. 204-211, 1993.
- Yeung, D.Y., and Gekey, G.A., "Using a Context-Sensitive Learning Network for Robot Arm Control," *Proc. 6th IEEE Conf. on Robotics and Automation*, vol. 3, pp. 1441-1447, 1989.
- Yeung, D.Y., *Modularity and Transfer in Connectionist Learning*. PhD thesis, Computer Science Dept., University of Southern California, 1990.
- Zomaya, A.Y., and Nabhan, T.M., "Centralized and Decentralized Neuro-Adaptive Robot Controllers," *Neural Networks*, vol 6, pp. 223-244, 1993.

4. ANALYSIS OF TRAINING IN ANALOGUE OUTPUT NETWORKS

The behaviour of adaptive networks and their learning algorithms is difficult to visualise, partly due to the high dimensionality of most networks, and partly due to the “black-box” nature in which the process of training is often treated. In particular, the processes involved in training networks that model systems with analogue outputs, and the reasons why these networks usually fail to generalise outside of the training data, are especially poorly understood. When dealing with complex or poorly understood processes the ability to make an analogy between it and a more readily comprehensible mechanism can be enormously useful. Whilst one must, of course, make sure that any corollaries drawn are valid, many valuable insights into difficult problems can often be gained by considering simpler, but analogous, ones.

In this chapter an analogy is drawn between the training of analogue output adaptive networks via supervised learning algorithms and the relatively well-understood problem of finding the best fit for a polynomial curve to a given set of points. This has previously been touched upon by [Bishop 1995], however, the analogy is examined here in much greater detail. The insights gained are used to show why most networks employed for modelling analogue output systems, including those currently used for the estimation of manipulator dynamics, are inappropriate. This analysis also provides the inspiration for a valid network architecture for system-space-wide modelling of manipulator inverse dynamics and other systems where the algebraic form is known but the coefficients are not.

4.1 WHY CURVE-FITTING IS ANALOGOUS TO NETWORK TRAINING

In the curve-fitting problem, the objective is to find a polynomial curve which will pass through, or close to, every point in a set of data such that the errors arising from the curve not passing directly through each point are minimised. Conceptually this is a very similar operation to network training. The set of data points are equivalent to the training set, the curve is the network and the errors in both systems are the differences between the data points’ and the model’s outputs evaluated at the data points’ input values.

Indeed, the only major differences between polynomial curve-fitting and training analogue output networks are that networks may have responses other than polynomials, and the input and output spaces for the curve-fitting problem are limited to one dimension. This apparent obstacle to the analogy caused by networks having higher dimensionality is easily surmounted, one needs only to imagine two mappings that take the i -dimensional network input space, and the o -dimensional output space, and transform them into the one dimensional spaces represented by the x -axis and y -axis values (respectively) in which the curve exists.

The curve-fitting problem is not just similar to network training, it is exactly the same process performed on systems of a more restricted class, and as such can provide practical mathematical information. To demonstrate this, consider the problem of training a linear combiner network with p training patterns, and compare it to the problem of fitting an n^{th} order polynomial (possessing only non-zero powers of x) to p data points. The solution method to the curve-fitting problem is well known. Let A be a matrix such that each row refers to a different data point's x -value, and each column is a different power of x from 1 to n . Furthermore let c denote the column vector whose elements are the desired polynomial coefficients, in the order corresponding to the columns in A (that is, 1 to n), and lastly, let B denote the column vector whose elements are the y -values of the data points, in the same order as the rows in A (that is, 1 to p), such that

$$A = \begin{bmatrix} {}^1x & {}^1x^2 & \dots & {}^1x^n \\ {}^2x & {}^2x^2 & \dots & {}^2x^n \\ \vdots & \vdots & & \vdots \\ {}^px & {}^px^2 & \dots & {}^px^n \end{bmatrix} \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \quad B = \begin{bmatrix} {}^1y \\ {}^2y \\ \vdots \\ {}^py \end{bmatrix} \quad (4.1-1)$$

where the leading superscripts denote individual data points. The coefficients of the best-fit curve (in a least squared error sense) can then be found from

$$c = A \setminus B \quad (4.1-2)$$

where the backslash character denotes the matrix division of A into B . If A is an $n \times n$ matrix and B is a column vector with n components, or a matrix with several such columns, then $c = A \setminus B$ is the exact solution to the equation $Ac = B$ via Gaussian elimination (with pivoting). If A is an $p \times n$ matrix (with p not equal to n) and B is a column vector with p components, or a matrix with several such columns, then $c = A \setminus B$ is the solution in the least squares sense to the under- or over-determined system of equations $Ac = B$.

Clearly, the solution method employed on the curve-fitting problem is directly applicable to the (single-layer) linear combiner network. If X is a $p \times n$ matrix, where each row is the network input vector (x) for a particular pattern, w is the weights vector and T is the

vector of target output values for all patterns, such that

$$X = \begin{bmatrix} {}^1x_1 & {}^1x_2 & \dots & {}^1x_n \\ {}^2x_1 & {}^2x_2 & \dots & {}^2x_n \\ \vdots & \vdots & & \vdots \\ {}^px_1 & {}^px_2 & \dots & {}^px_n \end{bmatrix} \quad w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \quad T = \begin{bmatrix} {}^1t \\ {}^2t \\ \vdots \\ {}^pt \end{bmatrix} \quad (4.1-3)$$

where the leading superscripts denote individual training patterns. Then the optimal weights vector (in a least squared error sense) for the patterns processed, denoted \check{w} , can be found from

$$\check{w} = X \backslash T \quad (4.1-4)$$

Furthermore, if the network concerned has more than one output, such that W denotes the weights matrix (defined in the usual way) and T is instead a $p \times o$ matrix of target values, where o is the number of network outputs, such that

$$T = \begin{bmatrix} {}^1t_1 & {}^1t_2 & \dots & {}^1t_o \\ {}^2t_1 & {}^2t_2 & \dots & {}^2t_o \\ \vdots & \vdots & & \vdots \\ {}^pt_1 & {}^pt_2 & \dots & {}^pt_o \end{bmatrix} \quad (4.1-5)$$

then, once again, the optimal least square error solution for the weights matrix, denoted \check{W} , can be found from

$$\check{W} = X \backslash T \quad (4.1-6)$$

4.2 NETWORK ORDER

A significant concept that can be taken from curve-fitting is the idea of network order. It is well known that to determine the coefficients of the polynomial curve uniquely, for the case so far presented where the curve possesses no constant term, one requires at least as many independent data points as the order of the polynomial. The same is apparent from Eqn. 4.1-2, where, if the rank of A is less than the number of coefficients to be determined, no unique solution to the equation is obtainable. This rule will also apply to training networks with learning algorithms.

In general, for a given network with analogue outputs, there will exist a single optimal state for approximating the whole of the modelled system. If, however, there is no unique solution for a given training set, then that set must be inadequately representative of the whole of the system being modelled, and the trained network state obtained from using

that set will almost certainly not be a close approximation of the system-wide optimal state. Thus it is vitally important to ensure that any training set used is likely to provide a unique solution for the network being trained.

For a fully connected feed-forward one-layer network, such as those discussed in the previous section, where each link has a unique weight associated with it, the order of the network is clearly equal to the number of input nodes. This is true regardless of the activation functions chosen for the output nodes, as long as they are uniquely invertible across the function's output range (that is, strictly increasing or decreasing). The number of output nodes also has no effect on the network order, as each output can be regarded as part of a separate system.

For the purpose of determining the orders of multi layered networks, and/or those that are less than fully connected, consider the network in Fig. 4.2-1. As previously stated, the two outputs can each be thought of as representing separate hypersurfaces which can be determined independently from each other with respect to the output layer weights. Thus it can be seen from the number of links leading to the output nodes that y_1 has an order of two, in terms of the values produced by the hidden layer, and y_2 an order of three. If one now ignores the output layer and instead regards the hidden layer as representing hypersurfaces that must be uniquely defined, then the four hidden nodes can be seen to have orders of 2, 3, 3, & 2, viewed top to bottom respectively, in terms of the network inputs.

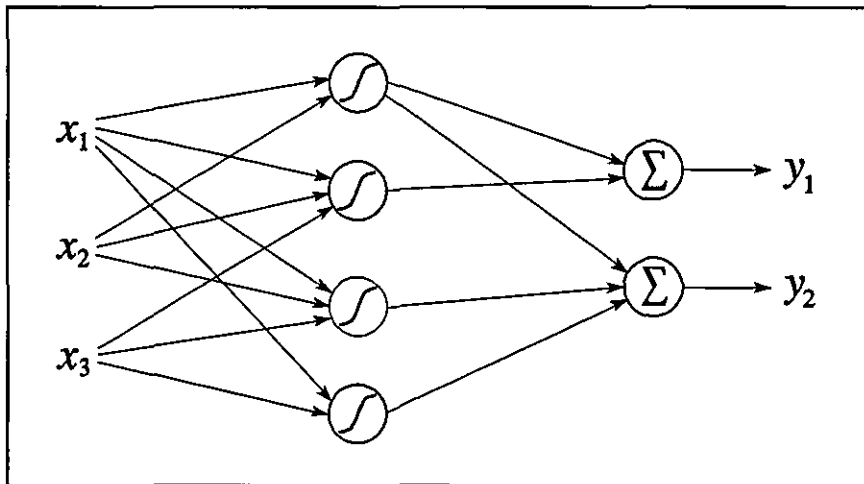


Figure 4.2-1: A less than fully connected two-layer adaptive network

The order of an output node in terms of the inputs, here denoted as $\sigma(y_i)$, is simply the highest valued product of their order with respect to the hidden nodes, and the associated hidden nodes' order with respect to the inputs. Thus,

$$\sigma(y_1) = 2 \times \max(2,3) = 6$$

$$\sigma(y_2) = 3 \times \max(2, 3, 2) = 9$$

Therefore, the order of the network as a whole is simply the higher of the two outputs' orders, namely nine.

This method allows one to calculate the order of any given feedforward network (with strictly increasing or decreasing activation functions) which is in its most compact form, that is, not equivalent to a network with fewer nodes or links (see Subsection 3.3.2). If every node within a given network is assigned a value equal to the number of independently weighted links that pass signals to that node, then the network's order is equal to the highest product of these values from any one route connecting the input layer to the output layer.

In practical applications, the data used to form a network's training patterns is almost always taken from empirical samples of the system to be modelled. Prior to training, there is often no simple way of determining whether the training patterns formed from this data are independent of one another, or to what degree measurement noise will detrimentally affect learning. Due to this, it is clear that the order of a network provides only a lower bound for the required number of training patterns, not an exact value. However, it is still useful to know the order of a network in order to estimate the number of patterns required in a given training set. For instance, if a network's training set contained 100 patterns, and the network was found to have an order of 85, then there would be a significant probability that the trained state of the network was not a unique solution to the training set, and therefore very unlikely to be an approximation of the system-wide solution.

4.3 NETWORK FLEXIBILITY

In the previous section, the order of the network was compared with the number of patterns in the training set. Here, the importance of a network's order is examined in relation to the order of the system being modelled. For the generalised problem of modelling a continuous analogue output system, one can visualise the system output(s) as an unbroken but "lumpy" hypersurface when plotted against the system inputs. The output(s) of the network chosen to model the system will form a similar hypersurface, existing in the same space, but initially of a different shape. If, for instance, all of the network's weights are initialised to zero, before training, then the network output in this state would form a level hyperplane. The process of training the network, with example patterns taken from the system space, is intended to alter the shape of the network hypersurface and mould it into a close approximation of the system hypersurface. With the curve-fitting analogy in mind, it is clear that the network must possess at least as high

an order as the modelled system for its hypersurface to be sufficiently flexible to be moulded into the shape of the system hypersurface. Note that a network's flexibility is closely related to the network's order, but is also dependent on architectural characteristics, such as the type of activation functions used. However, when the network's order exceeds that of the system being modelled then the problem of potentially localised learning emerges. To demonstrate this, consider again the curve-fitting problem, as shown in Fig. 4.3-1, where the order of the polynomial being fitted is higher than the order of the system that generated the data points.

In Fig. 4.3-1, the data points were generated from a cubic system equation, with a small disturbance term added to represent measurement noise. The best-fit curves are both quintics and were fitted to six data points each. Clearly both curves provide good estimates of the system response within the region spanned by the data points, however, outside this region these estimates vary wildly and produce very poor approximations to the system equation.

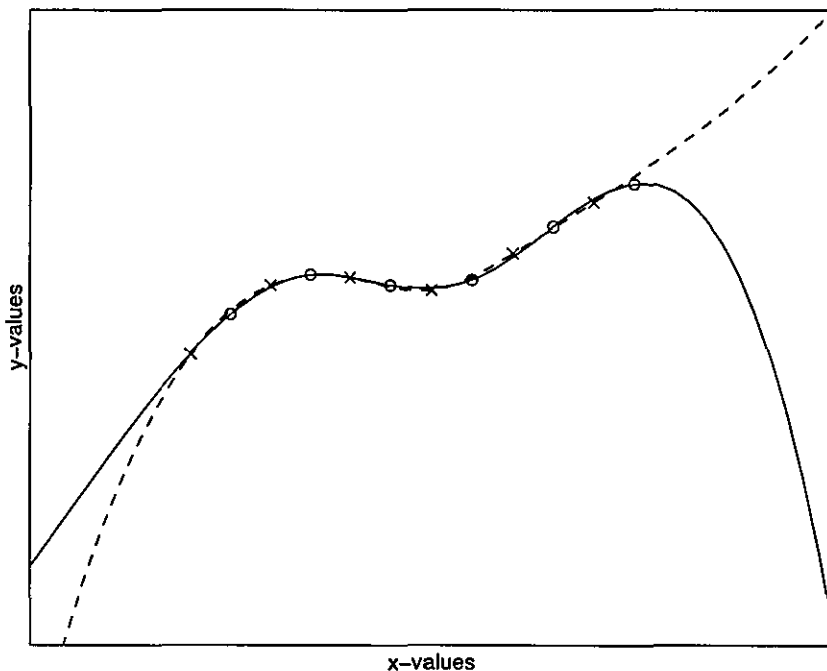


Figure 4.3-1: Two best-fit curves showing localised modelling of underlying system (solid line fitted to circles, dotted line to crosses)

This problem of over flexibility and localised learning also applies to training networks. When the network hypersurface has degrees of flexibility not possessed by the modelled system then training will cause these extra terms to model the noise in the training set, thus learning characteristics that do not exist in the underlying system. The effect of these extra terms may be small within the region spanned by the training patterns, but are likely to swamp the true response elsewhere. In an attempt to address this problem, literature

on adaptive network applications often glibly states that "a large training set was used, with patterns from across the problem space, to ensure the system is well represented." Similarly, when stop-sets are employed in attempts to avoid overtraining (see Subsection 3.6.1), a symptom of network over-flexibility, they themselves must be representative of the whole system. This naturally leads to the question: just how big a data set does one need for it to be representative? As a pertinent example of a type of complex system, consider the inverse dynamics of a six degrees of freedom manipulator (such as the PUMA 560). This system has, in general, 18 inputs (six positions, six velocities and six accelerations) and 6 outputs (the applied forces and/or torques), furthermore it is known that the system response is highly nonlinear and liable to vary considerably over the input space. If one was optimistic and believed that each input dimension could be representatively spanned by just 10 measurements, taken at intervals from across the whole of that input's possible working range, then the network's training set would require 10^{18} training patterns from interspaced points within the problem space. Even this conservative value clearly far exceeds the number of measurements that could be gathered in a human lifetime.

This then presents a serious dilemma for the modelling of an analogue output system. If the chosen network's hypersurface is less flexible than that of the system it is modelling, it cannot form an accurate approximation. Conversely, if the network is more flexible than the system, and the system has many inputs, then there is no practical way of gathering enough training data to make the network approximate the whole system. A solution to this predicament is to build a network whose response has exactly the same form as that of the system's response, thus, as the network weights are adjusted by training, there are no extraneous terms that can learn from input independent noise, and the network's hypersurface can precisely fit the system's hypersurface. Although this is similar in principle to using network pruning (see Subsection 3.6.2), there is a crucial difference: pruning reduces network flexibility by decreasing the number of nodes and links within a network, however, it does not guarantee a close match between the forms of the network and system responses due to the fact that a large degree of the network response's form is dictated by features such as the activation functions. For example, consider the problems inherent in using a network with sigmoidal activation functions, to model a system with a cyclic response (such as a trigonometric function). Furthermore, pruning algorithms intrinsically require data sets which are representative of the whole system in order to test the significance of network components. Therefore, to achieve an exact match in response forms for the majority of complex systems where the form, but not the coefficients, of the system equation are known, requires the development of the Context Sensitive Linear Combiner network (see Chapter 5).

4.4 SUMMARY

The process of training an analogue output adaptive network has been analysed by making use of the much more readily understood analogy of curve-fitting. From this, the concept of network order has been obtained and a method for determining the order of a given network (which exclusively contains nodes with strictly increasing or decreasing activation functions) was derived and demonstrated. This involved determining the number of unique weights contributing to each node, known as the node order, and then finding the highest possible product of these node orders from any one signal path connecting the inputs to the output layer.

By examining the consequences of a network's order with respect to its training set and to the dimensionality of the system being modelled, it has been shown that for systems with high input order and analogue outputs (which comprise the majority of engineering problems), networks which do not directly replicate the form of the modelled system are unlikely to accurately generalise as a result of supervised learning, when trained with real (partial and imperfect) data.

Given that the networks currently used for modelling manipulator inverse dynamics do not meet this requirement, a method for constructing analogue output networks which are capable of producing generally accurate representations of the modelled system, from partial and imperfect data, has been outlined and will be developed in the following chapter.

REFERENCES AND FURTHER READING

- Bishop, C.M., *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.
- Faddeev, D.K., and Faddeeva, V.N., *Computational Methods of Linear Algebra*. San Francisco: Freeman and Co., 1963.
- Kolman, B., *Introductory Linear Algebra with Applications*. New York: Macmillan, 4th Ed., 1988.
- Sillitoe, I.P.W., Hay, R.J., and Mulvaney, D.J., "A Comparison of Neural Network Structures for use in the Learning of Robotic Manipulator Dynamics," *Proc. Int. Conf. on Systems Engineering*, 1994.
- Strang, G., *Linear Algebra and its Applications*. Fort Worth: Harcourt Brace Jovanovich, 3rd Ed., 1988.

5. THE CONTEXT SENSITIVE LINEAR COMBINER NETWORK FOR INVERSE DYNAMICS

In this chapter, a novel Context Sensitive Linear Combiner (CSLC) network is developed for the modelling of a manipulator's inverse dynamics. Although expressed in terms of the chosen application, CSLC networks are applicable, indeed often necessary (see Section 4.3), for a large number of complex systems whose algebraic forms are known, but whose coefficients are not. The form of the CSLC network is developed by examining the detailed algebraic representation of a PUMA 560's inverse dynamics. Its properties are discussed, and its performance in modelling simulated inverse dynamics investigated.

The basis of the CSLC network is that it should possess exactly the same algebraic form as the system being modelled (in this case the inverse dynamics), thus avoiding the problems of over or under flexibility. This allows the network to be trained on a reasonable number of training patterns and be able to accurately approximate the inverse dynamics across the whole of the problem space. Furthermore, as with any engineering model, it is always desirable to use as much of the available a priori knowledge about the system as possible in an adaptive network's design. Therefore, the starting point for the CSLC network's development is a thorough analysis of the chosen system, that is, the inverse dynamics of a PUMA 560.

5.1 DECOMPOSITION OF THE INVERSE DYNAMICS OF A PUMA 560

The general Euler-Lagrange equation for a manipulator's inverse dynamics was developed in Sections 2.4 & 2.5, and was stated in Eqn. 2.5-2 as being

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) + F(\dot{q}) = \tau$$

As has been noted previously, the detailed expressions for the elements of the above equation are wholly dependent on the manipulator's configuration (as described by its homogeneous transforms, see Section 2.6) and may contain many hundreds of terms. However, it has been shown that significant computational efficiency gains can be obtained from combining constants that multiply common variable parameters [Murray & Neuman 1984, Armstrong et al. 1986]. Furthermore, in the case of the PUMA 560, it

has been shown that the majority of terms that appear within the detailed expressions of elements of the mass matrix and the Christoffel symbols are insignificant, specifically, their values are less than 1% of the largest term in the same element, or less than 0.1% of the largest constant term relevant to the same joint [Armstrong et al. 1986]. By disregarding these insignificant terms a less complex, more computationally efficient, network could be constructed. However, judgements upon the comparative magnitudes of terms within a manipulator's inverse dynamics can only be made after the constant coefficients (which are combinations of the manipulator's physical parameters) have been accurately identified. Therefore, in order to demonstrate that the design method of the CSLC network is generally applicable to any manipulator, it will be assumed that no estimates of the manipulator's physical parameter values are available. The network designed must therefore model the manipulator's inverse dynamics in their entirety.

Given that the terms in Eqn. 2.5-2 are functions of different combinations of q , \dot{q} and \ddot{q} , it is proposed to construct the CSLC network from several distinct subnetworks. This partitioned network approach was suggested by both [Bassi & Bekey 1989] and [Zomaya & Nabhan 1993] specifically for the modelling of manipulator dynamics. The subnetworks are considered individually in the following subsections.

5.1.1 The Mass Term

Consider the problem of representing the mass matrix as part of an adaptive network. For any manipulator of n degrees of freedom, M is known to have $\frac{1}{2}(n^2 + n)$ unique elements, and for the PUMA 560 in particular, it is known that each of these elements contain between zero and 113 separate terms when fully expanded. Despite the apparent complexity of some of these expressions however, they share a common format. To demonstrate this, consider the following partial expression of an element of the PUMA 560's mass matrix;

$$M_{23} = {}^pI_{zz} + a_3^2 m_4 - {}^pI_{yy} c_5^2 + a_2 {}^6r_z m_6 c_2 s_{23} c_5 + a_2 {}^6r_z m_6 s_2 c_{23} c_5 + \dots \quad (5.1-1)$$

where

$$c_i = \cos(q_i) , \quad s_i = \sin(q_i) , \quad c_{ij} = \cos(q_i + q_j) , \quad s_{ij} = \sin(q_i + q_j)$$

The first two terms in Eqn. 5.1-1 are constant, the others vary with trigonometric functions of q . These trigonometric functions are a consequence of the PUMA 560's revolute joints. In the case of prismatic joints, the corresponding q_i would appear without being acted upon by such functions. The c_{23} and s_{23} functions are the result of simplifications made possible by the z -axes of frames two and three being parallel. There are of course no \dot{q} or \ddot{q} terms within M , although the general acceleration vector is multiplied by the mass matrix in order to form the mass term, which it is now useful to

denote as τ^M .

Thus, for any given manipulator, it is possible to factorise the elements of the mass matrix for each unique product of the position dependent quantities they contain. This forms each element of M into a summation of terms, where each term consists of a unique product of position dependent quantities, or unity, factored by a group of constant parameters. Stating this explicitly for the example expression given in Eqn. 5.1-1, the following is obtained

$$M_{23} = \left(p^6 I_{zz} + a_3^2 m_4 + \dots \right) + \left(-p^5 I_{yy} + \dots \right) c_5^2 \\ + \left(a_2^6 r_z m_6 + \dots \right) c_2 s_{23} c_5 + \left(a_2^6 r_z m_6 + \dots \right) s_2 c_{23} c_5 + \dots$$

Performing the summation of the individual terms in the above expression requires only a linear combiner layer within the network, where each bracketed group of constant parameters would be modelled by a weight value, and the weight matrix would possess only those links needed to perform the required factoring. However, the nonlinear problem remains of how to produce the various groups of position dependent quantities from the link position values. The solution is remarkably simple: as there is no requirement for a network's inputs to be independent of each other, the joint positions can simply be preprocessed to form all of the different products of position dependent quantities that appear with the algebraic representation of the inverse dynamics (see Fig. 5.1-1).

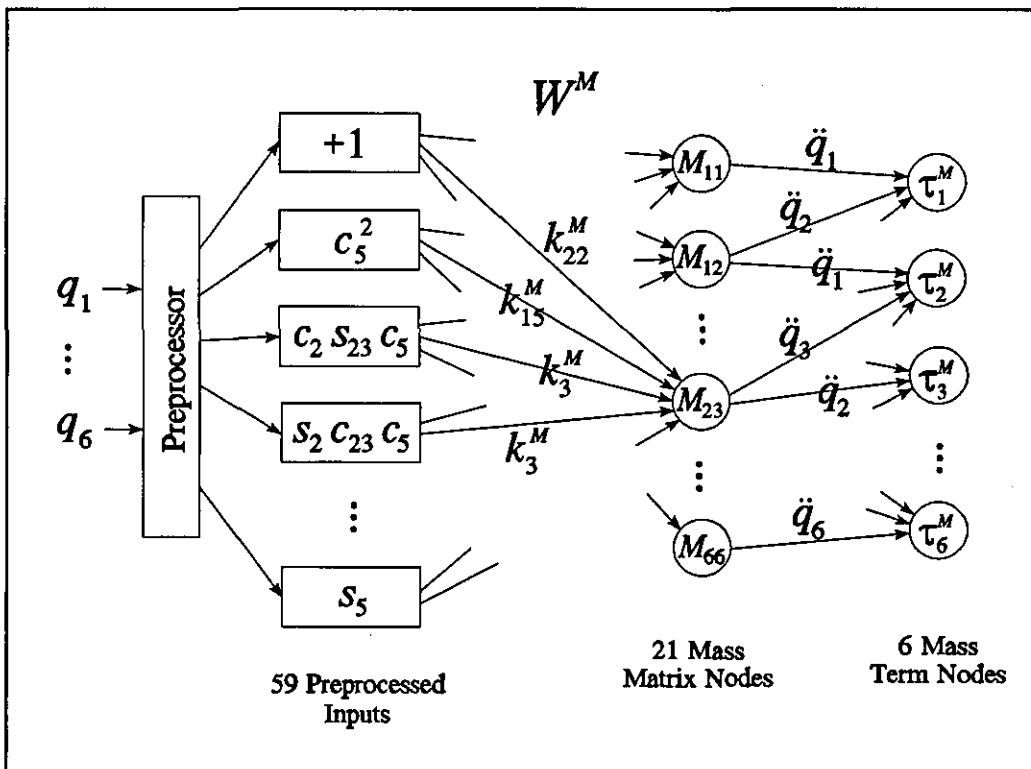


Figure 5.1-1: The mass term section of the PUMA 560 CSLC network (Only those links of W^M relevant to the partial Eqn. 5.1-1 are shown in full)

Now that a partial network can be formed to produce the elements of the mass matrix, it needs to be extended to incorporate the matrix multiplication of the link acceleration vector. To perform multiplications within an adaptive network, between sets of values which both vary with input patterns, the principle of context sensitivity must be used. To perform the matrix multiplication, each of the output values of the nodes calculating the unique mass matrix values needs to be directly multiplied by either one, or in the case of off-diagonal elements, two, of the acceleration values: the appropriate summations can then be made to obtain the six elements of the mass term. This can be achieved by the addition of another linear combiner layer to the partial network so far described, where the weight values are the link accelerations (thus using system inputs directly as context-sensitive weights), and the weight matrix has only those links required to perform the matrix multiplication (see Fig. 5.1-1).

Within the mass term subnetwork shown in Fig. 5.1-1, W^M denotes the weights matrix linking the preprocessed input layer and the first linear combiner layer (whose outputs are the elements of the mass matrix). Note that these layers are not fully connected, therefore many of the elements of W^M are null, that is, do not exist. Those links that do exist have associated weights equal to elements of k^M , the vector of mass coefficients whose values it is the intention to identify, where each coefficient is a collection of constant physical parameters. Comparing the mass term subnetwork (Fig. 5.1-1) to Eqn. 5.1-1, it can be seen that

$$\begin{aligned}
 k_{22}^M &= {}^p I_{zz} + a_3^2 m_4 + (\text{all other constant terms within } M_{23}) \\
 k_{15}^M &= -{}^p I_{yy} + (\text{all other constant terms which factor } c_5^2 \text{ within } M_{23}) \\
 k_3^M &= a_2 {}^6 r_z m_6 + (\text{all other constant terms which factor } c_2 s_{23} c_5 \text{ within } M_{23})
 \end{aligned}
 \tag{5.1-2}$$

Note that not all of the elements of W^M are unique, one of several alternative definitions of k_3^M is given by

$$k_3^M = a_2 {}^6 r_z m_6 + (\text{all other constant parameters factored by } s_2 c_{23} c_5 \text{ within } M_{23})$$

In total, there are 114 links within W^M , sharing integer multiples of just 26 coefficients.

Whereas the inputs to the first linear combiner layer are highly dependent on the kinematics of the manipulator in question, the inputs to the second layer are fixed. For any manipulator of n degrees of freedom there will be $\frac{1}{2}(n^2 + n)$ mass matrix nodes and, of course, n mass term nodes. Each mass term node will have n links to it from mass matrix nodes which share one of the same indices, that is, a mass matrix node denoted

M_{ii} is linked only to τ_i^M and weighted by \ddot{q}_i , a node denoted M_{ij} is linked to both τ_i^M and τ_j^M , and weighted by \ddot{q}_j and \ddot{q}_i respectively.

5.1.2 The Velocity Term

It is clear from Eqn.s 3.4-5 & 3.4-6 that the form of the velocity term is closely related to that of the mass term. However, instead of a matrix being multiplied by the acceleration vector, the tensor of Christoffel symbols is multiplied twice by the velocity vector. This implies that the velocity section of the inverse dynamics CSLC network will require two context sensitive layers rather than just the one that appears in the mass section. As for calculating the Christoffel symbols themselves, it can be shown that they have the same general algebraic form as the mass matrix elements, and therefore, can be computed in a similar fashion by a single layer of linear combiner nodes.

Eqn. 2.4-5 defines the Christoffel symbols as follows;

$$c_{ijk} \equiv \frac{1}{2} \left(\frac{\partial M_{ik}}{\partial q_j} + \frac{\partial M_{ij}}{\partial q_k} - \frac{\partial M_{jk}}{\partial q_i} \right)$$

Therefore, although the particular products of position dependent quantities will differ from those appearing in the mass term (due to differentiation), the constant coefficients that occur in each of the Christoffel symbols' algebraic representations will consist of one or more of the mass matrix's coefficients. Therefore, instead of requiring another vector of coefficients for the velocity section of the CSLC network, the weight values of the links connecting the preprocessed inputs and the Christoffel symbol nodes can be expressed in terms of the mass coefficients. In particular, due to one of the differentiated terms in Eqn. 2.4-5 always being zero, the links connecting to a node denoted c_{ijk} have weights equal to plus or minus half the value of either one or a pair of the weights associated with links connecting to M_{ij} , M_{jk} and M_{ik} .

To illustrate the functionality of the velocity term subnetwork, consider the following two representative Christoffel symbols

$$c_{345} = \left(\frac{1}{2}k_{15}^M - \frac{1}{2}k_9^M \right) s_4 - k_{15}^M s_4 c_5^2 + k_1^M s_4 c_5 \quad (5.1-3)$$

$$c_{356} = \frac{1}{2}k_4^M s_4 c_5 \quad (5.1-4)$$

and the diagram of the subnetwork presented in Fig. 5.1-2.

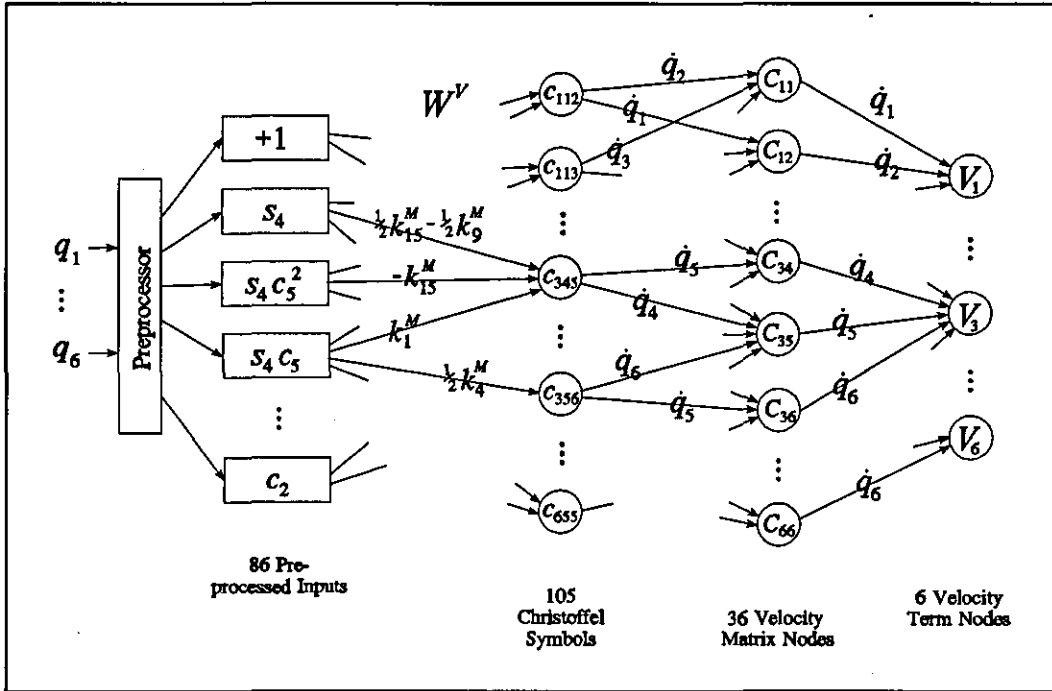


Figure 5.1-2: The velocity term section of the PUMA 560 CSLC network (Only those links of W^V relevant to Eqn.s 5.1-3 & 5.1-4 are shown in full)

The weights matrix W^V is defined in a similar fashion to W^M , that is, not all of its elements exist or have unique weights, and the element W_{ij}^V is the weight associated with the link from the i^{th} preprocessed input to the j^{th} Christoffel symbol node. The outputs of the second hidden layer are denoted as elements of the velocity matrix, C , which is defined as

$$C_{ij} = \sum_{k=1}^n c_{ijk} \dot{q}_k \quad (5.1-5)$$

such that

$$V_i = \sum_{j=1}^n C_{ij} \dot{q}_j \quad (5.1-6)$$

Thus the combination of Eqn.s 5.1-5 & 5.1-6 (and the identity $c_{ijk} = c_{ikj}$) produces the same definition of the velocity term as Eqn. 3.4-6. Note, however, that the described velocity subnetwork contains only those nodes pertaining to Christoffel symbols which are non-zero and unique in the general case. Hence, for a manipulator of n degrees of freedom, there will be $\frac{1}{2}(n^3 - n)$ Christoffel symbol nodes, where each node denoted c_{ijj} will have just the one link leading from it to the velocity matrix layer, whilst those denoted c_{ijk} , where $j \neq k$, will have two. As there is no inherent redundancy within the velocity matrix, the second hidden layer contains all n^2 velocity matrix nodes, each with a single link leading from it to the corresponding velocity term node. As an aid to comprehension, it is useful to remember that the first index of c , C and V always refers

to the particular output/joint to which the term in question contributes.

As in the mass section of the CSLC network, the important and unconventional aspects of the velocity term subnetwork are the preprocessing of the inputs by functions determined from factoring the algebraic representation of the inverse dynamics, and the use of input values, in this case \dot{q} , as (non-trainable) weights.

5.1.3 The Gravity Term

In contrast to the previous two terms, it can be seen from Eqn.s 2.4-7 & 2.3-28 that there is no matrix multiplication of input vectors required in the calculation of G , the gravity term. Although G is a nonlinear function of q , examination shows that, in general, there exists linear relationships between the elements of the gravity term and a set of functions of q , where these functions can be determined in the same manner as those of the mass matrix and the Christoffel symbols.

In the case of the PUMA 560, the elements of the gravity term can be factored such that

$$\begin{aligned}
 G_1 &= 0 \\
 G_2 &= k_4^G c_2 + k_1^G s_2 + k_5^G s_{23} + k_2^G s_{23} c_5 + k_3^G c_{23} + k_2^G c_{23} c_4 s_5 \\
 G_3 &= k_5^G s_{23} + k_2^G s_{23} c_5 + k_3^G c_{23} + k_2^G c_{23} c_4 s_5 \\
 G_4 &= -k_2^G s_{23} s_4 s_5 \\
 G_5 &= k_2^G s_{23} c_4 c_5 + k_2^G c_{23} s_5 \\
 G_6 &= 0
 \end{aligned}$$

where k^G is the vector of gravity coefficients, defined in this case as

$$\begin{aligned}
 k_1^G &= {}^2r_y m_2 g \\
 k_2^G &= -{}^6r_z m_6 g \\
 k_3^G &= -a_3 m_4 g - a_3 m_5 g - a_3 m_6 g \\
 k_4^G &= -a_2 m_3 g - a_2 m_4 g - a_2 m_5 g - a_2 m_6 g - {}^2r_x m_2 g \\
 k_5^G &= {}^3r_y m_3 g - {}^4r_z m_4 g - d_4 m_4 g - d_4 m_5 g - d_4 m_6 g
 \end{aligned}$$

Thus the gravity term of the PUMA 560 can be represented by the subnetwork shown in Fig. 5.1-3.

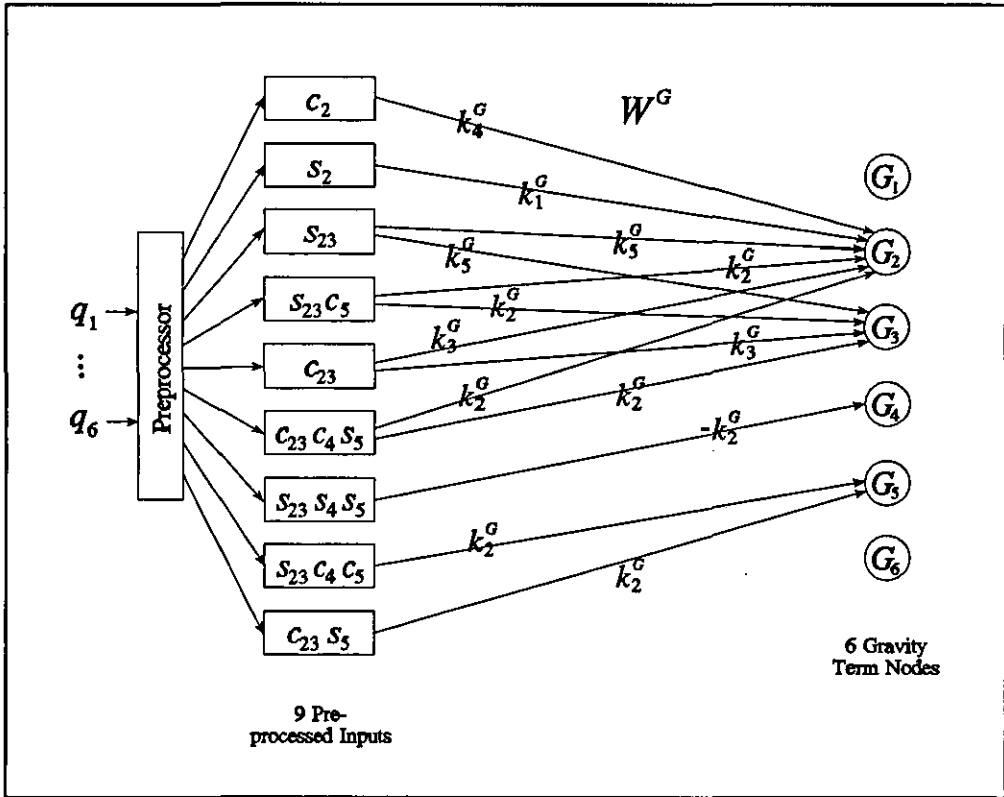


Figure 5.1-3: The gravity term section of the PUMA 560 CSLC network

Through the use of the preprocessing layer, and analysis of the algebraic representation of the gravity term, it has been possible to produce a subnetwork which is entirely linear with respect to the processed inputs and contains only conventional (context insensitive) trainable weights.

5.1.4 The Friction Term

From Subsection 2.5.1, it can be seen that the effects of friction are decoupled, that is, they are independent of joints other than the one upon which they act. Furthermore, the stated friction term models are all functions of the generalised velocity vector only. Eqn. 2.5-3 shows how the friction in manipulators can be modelled by a combination of a viscous friction term and a dry friction term. As has been previously stated, the effects of static friction upon a PUMA 560 are not insignificant, but the PUMA 560 is not, as standard, equipped with velocity transducers. This makes modelling the static friction of doubtful value given that the estimation of joint velocities from joint position measurements is inherently noisy, leading to uncertainty in the sign of the static friction term when its magnitude is at its greatest (when the joint velocity is near zero).

The viscous friction term, denoted τ^v , can be expressed as a very simple linear combiner

subnetwork without any preprocessing of terms, as can be seen in Fig. 5.1-4.

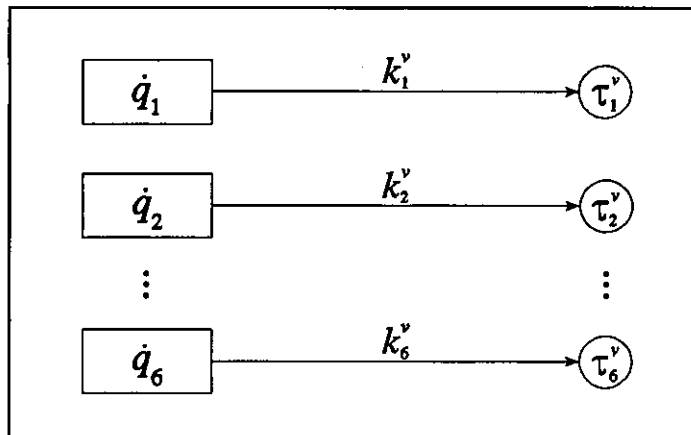


Figure 5.1-4: The viscous friction section of the PUMA 560 CSLC network

The elements of the dry friction term, denoted τ^d , are proportional to the signum of the joint velocities. Therefore, to express their calculation as a subnetwork requires only a linear combiner layer with a preprocessor to find the signs of \dot{q} . The resultant subnetwork is shown in Fig. 5.1-5.

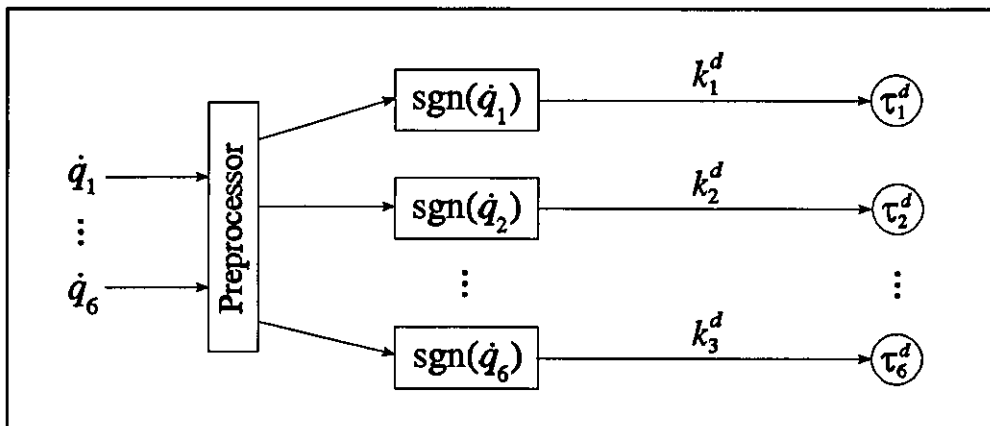


Figure 5.1-5: The dry friction section of the PUMA 560 CSLC network

5.1.5 The Complete CSLC Network

Now that subnetworks have been formulated for each term within the Euler-Lagrange equation, the full network can be described by simply joining the five subnetworks (Figs. 5.1-1 to 5.1-5), superimposing the output nodes of each subnetwork to form the six nodes that calculate the elements of τ . An overview of the resultant CSLC network's functionality is presented in Fig. 5.1-6. Details of the composition of the processed input

vectors, the weights matrices (W^M , W^V & W^G) and the coefficient vectors k^M and k^G , for a PUMA 560, can all be found in Appendix B.

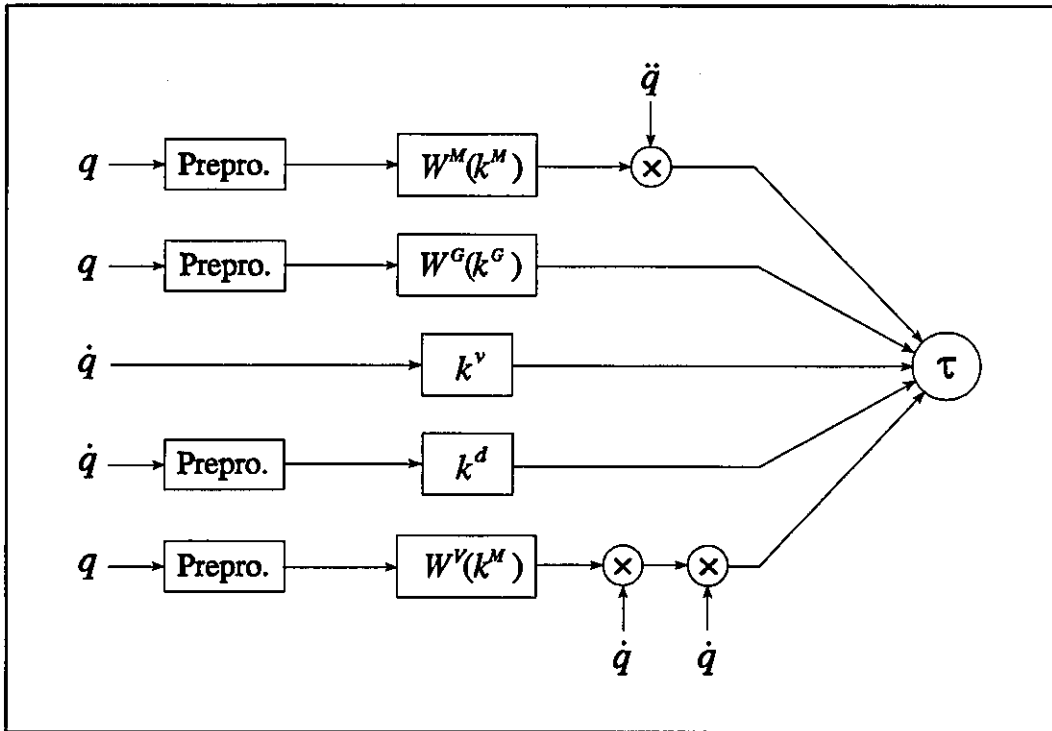


Figure 5.1-6: Functionality of the CSLC network for manipulator inverse dynamics (depicting, from top to bottom, the mass, gravity, viscous friction, dry friction and velocity subnetworks)

5.2 PROPERTIES OF THE CSLC NETWORK

The advantageous properties of the CSLC network, compared with conventional networks, are listed below:

- Good modelling capability
- Good generalisation capability
- Single minima in network's error hypersurface
- Transparency of operation
- Network modularity
- Low network order
- Computational efficiency
- Unrestricted choice of learning algorithm(s)

The principal limitation of CSLC networks is that they require the algebraic form of the

system being modelled to be known. (Note that this cannot be viewed as a disadvantage when compared to conventional networks, which, as discussed in Chapter 4, are incapable of accurately learning problem-space-wide models of high input order systems such as the PUMA 560 from empirical data).

These qualities are general to any CSLC network applied to any suitable application. Each of the listed properties is discussed in detail below with respect to learning the PUMA 560's inverse dynamics and thus identifying its dynamics coefficients.

5.2.1 Good Modelling and Generalisation Capabilities

The function performed by the network is exactly that of the most complete set of inverse dynamics equations derived to date, namely the Euler-Lagrange equations with compensation for friction and actuator effects. The minor differences that will inevitably exist between a real manipulator's motion, and that predicted for it by the Euler-Lagrange equation, are expected to be a lesser source of noise than disturbances such as quantisation error (see Section 7.2 for a full discussion of possible sources of noise). Due to this match between the algebraic forms of the network and of the manipulator's inverse dynamics, the network is capable of accurately modelling the PUMA 560's motion. For the same reason, the solution obtained as a result of fully training the network is expected to be general to the whole of the problem space.

5.2.2 Sole Error Hypersurface Minima

Although the outputs of the CSLC network are nonlinear functions of the input values, and discontinuous in terms of the non-preprocessed inputs, they are linearly related to the trainable weight values. This means that the network's $n+1$ dimensional error hypersurface (where n is the number of unique weights, 43 in the case of the PUMA 560) is continuous and possesses only one minima. This greatly simplifies learning compared to networks which employ non-linear activation functions: in the absence of noise, gradient descent learning algorithms, such as Backpropagation, are guaranteed to eventually identify the optimal solution, irrespective of initial weight values.

5.2.3 Transparent Operation

In marked contrast to most adaptive networks, all the values computed in the operation of the CSLC network represent real quantities, each being some part of the well understood inverse dynamics equations. This compares favourably with the obscure

“black-box” type of operation typical of most conventional networks, replacing it with the ability to analyse both weights and node outputs from a position of understanding as to what they each represent. For instance, in the case of the PUMA 560, several of the mass and velocity subnetworks’ weight coefficients are the sums of various masses, inertias and/or squared terms, this means that these coefficients’ true values must be positive. This information can be used to sensibly restrict the initial values of the corresponding weights to the correct signs, and as an aid to checking the state of the network during training.

5.2.4 Network Modularity

Each separate term within the Euler-Lagrange equation is represented by a distinct subnetwork, this allows each to be analysed, trained or calculated independently. For instance, if a manipulator which has had its physical parameters identified by a CSLC network has its bearings cleaned, it would be desirable to relearn the friction coefficients, but unnecessary to expend computational effort in relearning the gravity or mass/velocity coefficients. This can be achieved by allocating separate learning rates to each subnetwork, and setting all but the dry and viscous friction subnetworks’ to zero.

The ability to set different learning rates for different network sections is of general usefulness when employing learning algorithms, such as Backpropagation, that can overemphasise the training of significant terms. As can be seen from Eqn. 3.4-9, the effect of Backpropagation is to concentrate learning on those weights which cause the greatest contributions to the network outputs. Although this property can initially hasten the decline in network error, if not countered it can make the identification of less significant weights more difficult, ultimately causing training to take longer to reach a given error tolerance. Therefore, it is useful to define separate values for Backpropagation’s learning rate, η , for each subnetwork, dependent upon their significance.

5.2.5 Low Network Order

Despite containing a relatively large number of nodes, the CSLC network is sparsely connected and contains only 43 independent weights (26 mass/velocity coefficients, 5 gravity, 6 viscous friction and 6 dry friction). This accounts for the CSLC network’s low order of just 28, which is due to the most dependent output, τ_2 , being a function of 28 independent weights (21 mass/velocity coefficients, 5 gravity, 1 viscous friction and 1 dry friction). This means that, in the absence of noise, the CSLC network will reliably learn the global solution to the PUMA 560’s inverse dynamics from as few as 28 training patterns whose input values are linearly independent.

5.2.6 Computational Efficiency

Due to the sparsity of network links, the sharing of common weight values and the independence of the subnetworks, the number of mathematical operations required to process and train the CSLC network is relatively small compared to that required by other networks proposed for modelling manipulator dynamics (for example [Bassi & Bekey 1989, Miller et al. 1990]). Also, during training, the preprocessing of the input values needs only be performed once, rather than for every epoch.

Furthermore, as the weight values associated with links in the velocity term subnetwork are a subset of those associated with links in the mass term subnetwork, it is not strictly necessary to apply a learning algorithm to the velocity term section of the network. This would mean that the adjustment to a common weight value would be equal to the average of the relevant elements of ΔW^M , rather than elements of both ΔW^V and ΔW^M . Disregarding the effects of noise, the final estimate of common weight values will be identical to those obtained from applying learning to both subnetworks. Given that the velocity term subnetwork is by far the largest component of the total network, disregarding it for the purposes of calculating weight adjustments greatly reduces the computational effort required for each training epoch.

5.2.7 Unrestricted Choice of Learning Algorithm

Unlike some network architectures, the CSLC network does not require a specialised learning algorithm. Neither the input preprocessing nor the context-sensitivity place any restrictions on the kinds of learning algorithm that can be employed. Therefore, any of the large number of learning algorithms that are appropriate for training linear combiner nodes (such as Backpropagation) can be used in the training of the CSLC network.

5.3 PRELIMINARY PARAMETER IDENTIFICATION SIMULATION

The training of the PUMA 560 CSLC network on real empirical data is presented in detail in Chapter 7, however, it is useful at this point to briefly discuss the training of the CSLC network on simulation data. The analyse of learning is made much simpler when dealing with data produced by simulation, due to the ideal network state being known.

As the CSLC network's error hypersurface has only one minima, gradient descent learning algorithms would appear well suited. However, whilst using Backpropagation to train the PUMA 560 network, the rate of learning progressively slowed as training advanced. This is to be expected given that the error hypersurface gradient that

Backpropagation relies upon to provide a learning impulse tends towards zero as the network state approaches the optimal solution. This is because the optimal solution exists at the error hypersurface's minima, where the error gradient is zero in all dimensions, that is, for all weights. Thus the region surrounding the minima comprises a flat-spot (a region of the error hypersurface with near zero gradient). Furthermore, the spatial interval between re-evaluating the instantaneous error gradient must be kept small as the network approaches its optimal state, in order for Backpropagation to reliably produce weight adjustments that cause the network's error cost to decrease between epochs. This translates to ever smaller values of η being required to ensure positive learning as training progresses.

As has already been mentioned in Subsection 3.4.2, there have been several variants of Backpropagation proposed specifically for overcoming the difficulties caused by flat-spots and local minima [Stornetta & Huberman 1987, Fahlman 1988, Hagiwara 1990, Li 1990], however, these algorithms often have little mathematical foundation [Bishop 1995], and by definition, are not gradient descent techniques. The preferable solution is to employ a learning algorithm which does not rely upon determining the error surface gradient and which is capable of rapidly training a CSLC network, most particularly when the network weights are close to their optimal values.

5.4 SUMMARY

The formulation of a Context Sensitive Linear Combiner network has been demonstrated in this chapter, using the inverse dynamics of a PUMA 560 as the example application. Forming an exactly matching network representation of the system required the decomposition of the Euler-Lagrange equation into several subnetworks, the application of input preprocessing and the use of some system inputs (\dot{q} & \ddot{q}) as context sensitive weights. The detailed form of the Euler-Lagrange equation specific to the PUMA 560 was examined. It was shown how the elements of the mass matrix, the gravity effect vector and the Christoffel symbols could all be expressed as individual summations of terms, where each term consists of a group of manipulator physical parameters, defined to be an inverse dynamics coefficient, which, for nonconstant terms, factors a trigonometric function, or group of functions, that vary with q . This representation of the dynamics' elements leads naturally to the formation of the corresponding subnetworks, where the trigonometric functions form the (preprocessed) inputs, and the coefficients are the weights of the network links leading from them. With the position dependent quantities represented in a network form, context sensitive links are then required to perform the vector products which occur in the mass and velocity terms. This capability to combine information from two pattern dependent sources through multiplication is unique to context sensitive networks. It was also noted that the velocity subnetwork weights are

linear functions of the mass subnetwork weights.

The properties of the CSLC network were discussed, noting that its properties are general to this type of network and not restricted to the inverse dynamics application. These include good modelling capability and generalisation of the obtained model to the whole problem space due to the match between the algebraic form of the network and that of the system, the existence of only one minima in the network's error hypersurface, the transparency of operation due to the weights and node outputs within the network being recognisable quantities that appear in the system equation, network modularity due to the dependencies of each term within the system equation being known, the lack of restrictions on the learning algorithm that can be used for training, the low network order due to the sparsity of node links and the sharing of weights between several links, and the low computational effort required to process the network. For a manipulator inverse dynamics CSLC network in particular, it was explained how the velocity subnetwork (which is by far the largest subnetwork in terms of nodes, links and computational load) can be omitted from the learning segment of the training process (but not the calculation of the network output) in order to decrease the time taken to process each epoch. It is important to note that many of the CSLC network's properties, significantly those of generalisation and single minima, are not possessed by conventional networks when used to model nonlinear systems (such as manipulator inverse dynamics).

Finally, a preliminary examination of Backpropagation training has shown that, although the optimal solution was reliably identified by gradient descent of the network's error hypersurface, training rates progressively decreased, making training tedious. To address this problem, a novel learning algorithm, which does not rely upon gradient descent, will be introduced in the next chapter.

REFERENCES AND FURTHER READING

- Armstrong, B., Khatib, O., and Burdick, K., "The explicit dynamic model and inertial parameters of the PUMA 560 arm," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 510-518, 1986.
- Bassi, D.F., and Bekey, G.A., "Decomposition of neural network models of robot dynamics: a feasibility study," *Simulation and AI; Conf. Simulation Series*, vol. 20, no. 3, pp. 8-13, 1989.
- Bishop, C.M., *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.
- Fahlman, S.E., "Faster-learning variations on back-propagation: an empirical study," Touretzky, Hinton and Sejnowski (Eds.), *Proc. Connectionist Models Summer School*, San Mateo, CA: Morgan Kaufmann, pp. 38-51, 1988.

- Hagiwara, M., "Novel back-propagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection," *Int. Joint Conf. on Neural Networks*, vol. I, pp. 625-630, 1990.
- Li, S., "An optimized backpropagation with minimum norm weights," *Proc. Int. Joint Conf. on Neural Networks*, San Diego, vol. I, pp. 697-702, 1990.
- Miller, W.T., Hewes, R.P., Glanz, F.H., Kraft, L.G., "Real-Time Dynamics Control of an Industrial Manipulator Using a Neural-Network-Based Learning Controller," *IEEE Trans. on Robotics and Automation*, vol. 6, no. 1, Feb. 1990.
- Murray, J.J., and Neuman, C.P., "ARM: an algebraic robot dynamic modeling program," *Proc. 1984 Int. Conf. on Robotics*, Atlanta, Georgia, pp. 103-114, March 1984.
- Stornetta, W.S., and Huberman, B.A., "An Improved Three-Layer Back-Propagation Algorithm", *Proc. IEEE Int. Conf. on Neural Networks*, San Diego, pp. 637-644, 1987.
- Tabary, G., and Salaün, I., "Control of a redundant articulated system by neural networks," *Neural Networks*, vol. 5, pp. 305-311, 1992.
- Zomaya, A.Y., and Nabhan, T.M., "Centralized and Decentralized Neuro-Adaptive Robot Controllers," *Neural Networks*, vol 6, pp. 223-244, 1993.

6. PROPORTIONAL ERROR ALLOCATION

In order to hasten network training, and in particular to prevent the maximum attainable rate of error cost reduction tending to zero as a given network approaches its optimal state, a novel learning algorithm has been developed which does not rely upon gradient descent, but which is capable of high error cost reduction rates, especially when the network weights are near-optimal. The algorithm is termed Proportional Error Allocation (PERAL), and its derivation is described in this chapter. The theoretical basis for PERAL is first described and analysed. This is followed by an explanation of its practical implementation and limitations. Empirical results from training two different classes of networks are also presented, complete with comparisons of the PERAL learning algorithm's performance to that achieved with Backpropagation.

Although PERAL will be shown to be a useful tool in the training of the adaptive networks considered, a full study of its applicability has not yet been completed (see Section 8.2). Neither has there yet been a comparison of its training performance with a broader range of published learning algorithms, in particular those designed to perform rapid training, such as [Battiti 1992, Riedmiller & Bravin 1993, Moller 1993]. However, PERAL is presented here not only because of its potential capability to reduce training times, but also because it employs a novel approach to minimising a given network's output error, which is not dependent upon the error cost hypersurface.

6.1 THEORETICAL BASIS

The aim of the PERAL algorithm is to adjust the value passed through each link in a given network by a fraction of the network output error, where that fraction is proportional to the link's significance. The significance of a link is regarded as the relative contribution it makes to a node, compared to the contributions of all the links that connect to the same node.

To examine this principle consider the single summation node network in Fig. 6.1-1, which can be thought of as a network in its own right or as a subsection of a more complex network.

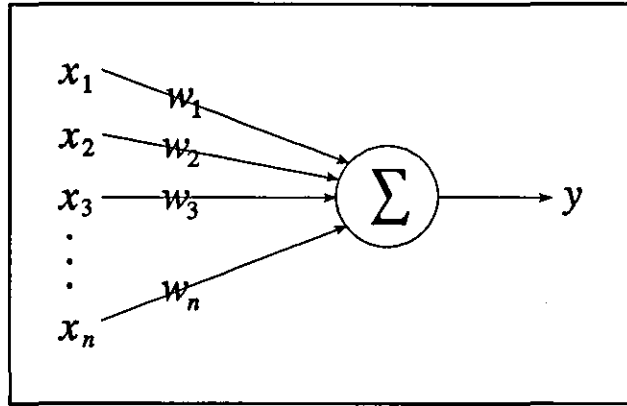


Figure 6.1-1: A single summation node network

Let \check{w} denote the vector of optimal weight values, of which w is the current estimate. Associated with the output, for each training pattern, will be a desired target value t . The simple difference output error for a given pattern is thus defined as

$$e = t - y \quad (6.1-1)$$

Therefore, for the network in Fig. 6.1-1, it is desirable to increase the weighted values in the n links by a sum total of e . If the adjustments made are proportional to each link's true significance, in other words their significance when the network is in its optimal state, then, for a given link j ,

$$\Delta(x_j, w_j) = \frac{|x_j \check{w}_j| e}{\sum_{m=1}^n |x_m \check{w}_m|} \quad (6.1-2)$$

As the input vector is constant, the following learning algorithm can be obtained:

$$\Delta w_j = \frac{\text{sgn}(w_j) |\check{w}_j| e}{\sum_{m=1}^n |x_m \check{w}_m|} \quad (6.1-3)$$

Clearly this equation in its present form cannot be used on practical problems, since it requires the optimal weight values to be known. However Eqn. 6.1-3 is worth examining for the desirable properties it displays. In the trivial case of n being equal to one, and in the absence of noise, all training patterns will produce the same Δw , which adjusts the weights vector to its optimal value. In the more general case of $n > 1$, Eqn. 6.1-3 describes an iterative training process, such that for any given iteration and any given training pattern, the algorithm produces an adjustment to w , which, in the absence of noise, will in turn cause the network to produce a zero error in response to that pattern (and, in general, that pattern only). Thus, the value of Δw obtained will differ for each pattern. However, if a batch learning approach is used, that is, if the mean Δw from across all the patterns is employed at each iteration, then the adjustments to the weights will always take the network closer to its optimum state. This ensures that, for any initial

weight values (except $\prod w_j = 0$), the weights' optimal values will be identified, to within any chosen accuracy, in a finite number of iterations.

There is an important exception to this process. The above does not hold if there are fewer independent signum of the training patterns than there are weights. That is, if one forms a matrix where each row is the signum function of x for a specific pattern, then the mean of the weight adjustments specified by Eqn. 6.1-3 is only guaranteed to progress the network towards its optimal state if the matrix has full rank. The reason for this is straightforward: to identify uniquely the optimal weights for the training set one requires at least as many patterns as weights (see Section 4.2). However, if the training set contains two or more patterns whose vectors of input value signum are not linearly independent, then training will behave exactly as if those patterns had been replaced by a single pattern equal to their sum. Therefore the effective training set can always be reduced to contain only independent signum, but to be capable of inducing successful learning must still have at least as many patterns as weights.

Clearly this idealised version of PERAL learning is potentially very useful when processing data produced by a linear combination and devoid of noise, however, a practical learning algorithm should also be robust to non-ideal disturbances within its training set. If one regards all the sources of noise within a given training pattern in terms of the network output, such that

$$x \check{w} = y - k \quad (6.1-4)$$

where

$$k = \text{disturbance term due to noise}$$

then, from Eqns. 6.1-1 and 6.1-4, it can be seen that the training described by Eqn. 6.1-3 will be adversely affected unless the noise is uncorrelated, and

$$\sum_{\text{all patterns}} x_j k = 0 \quad \text{for all } j \quad (6.1-5)$$

The simplest way to approximate the above condition is to employ a large training set. Thus, if the disturbances due to noise are uncorrelated and have zero mean, Eqn. 6.1-5 is likely to be approximately true and training will not be significantly affected.

The choice of initial weights can begin to affect the result of training when the training set contains noise, but evidence so far collected suggests that the tolerances within which the network's optimal weight values can be identified are largely related to

$$\frac{1}{p} \sum_{\text{all patterns}} x k \quad (6.1-6)$$

(where p , as before, denotes the number of patterns in the training set).

6.2 PRACTICAL APPLICATION

Within this section the PERAL learning algorithm developed so far is generalised and modified for practical application. This requires the approximation of \tilde{w} by w , which restricts the use of PERAL to fine-tuning, that is, w must be partly trained or some other rough estimate of the optimal weights obtained prior to employing PERAL.

As it is not clear how best to produce an estimate of output error for nodes which precede further adaptive (context insensitive) layers, the description of PERAL's practical application presented here is restricted to networks which possess just one adaptive layer. The same generalised two-layer adaptive network (and accompanying notation) as was used in the development of Backpropagation (Subsection 3.5.2), is used here (reproduced as Fig. 6.2-1), but with the weights matrix W^B defined as being context sensitive and employing network inputs directly as weight values. The CSLC mass subnetwork is therefore a specialisation of the network shown below. Furthermore, as with networks trained using Backpropagation, the activation functions performed by the network nodes are assumed to be continuous, which implies that the network contains only linear combiner and/or locally sensitive nodes.

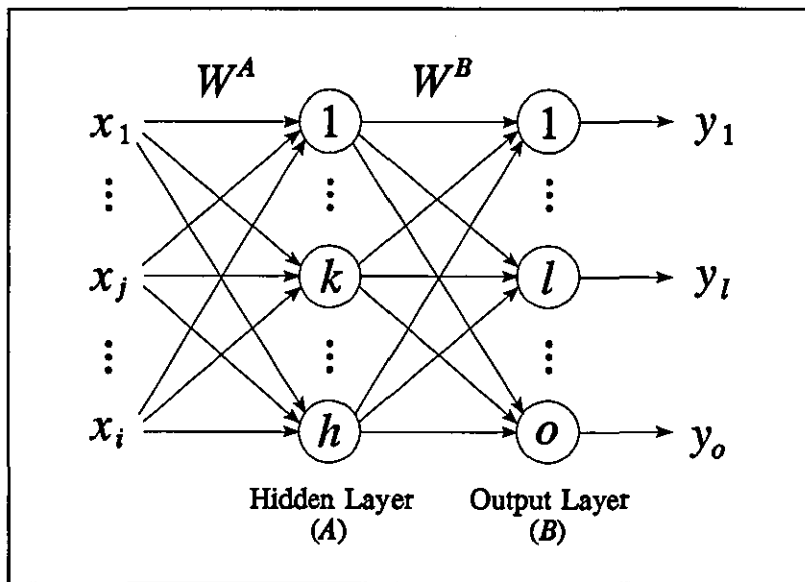


Figure 6.2-1: A generalised two-layer network

Consider the arbitrary output node l , which performs the activation function f_B . If it produces an output y_l which is an estimate of a target value of t_l , and f_B is locally sensitive, then one may well presume that it is desirable to adjust the summed input to node l by $f_B^{-1}(t_l) - f_B^{-1}(y_l)$. However, to do this would mean that one is effectively training a summation layer with target values of $f_B^{-1}(t_l)$ and ignoring the purpose of the activation function, which is to produce areas of relative sensitivity to errors. That is,

because locally sensitive activation functions bias their associated output errors within particular regions of interest, training should also be likewise biased to enhance learning within these regions. Thus if, for example, f_B is a hyperbolic tangent function, learning should be concentrated at the region where S_l^B (the summed input to node l in layer B) is close to zero. Therefore, it is desirable to make adjustments so that

$$\Delta S_l^B = \begin{cases} t_l - y_l & \text{if } f_B \text{ is strictly increasing} \\ y_l - t_l & \text{if } f_B \text{ is strictly decreasing} \end{cases} \quad (6.2-1)$$

Note that linear combiner nodes can be thought of having an identity activation function, which is strictly increasing.

Now consider a link which connects node k of the hidden layer with node l of the output layer: the weighted value it passes to the output layer is $f_A(S_k^A) W_{kl}^B$. Thus, the desired adjustment to this value, as an approximation of Eqn 6.1-2, is

$$\Delta (f_A(S_k^A) W_{kl}^B) = \frac{|f_A(S_k^A) W_{kl}^B| \Delta S_l^B}{\sum_{m=1}^h |f_A(S_m^A) W_{ml}^B|} \quad (6.2-2)$$

Therefore, the adjustment to the output of node k in the hidden layer, averaged across the o nodes to which it connects, is

$$\begin{aligned} \Delta (f_A(S_k^A)) &= \frac{1}{o} \sum_{l=1}^o \left(\frac{|f_A(S_k^A) W_{kl}^B| \Delta S_l^B}{W_{kl}^B \sum_{m=1}^h |f_A(S_m^A) W_{ml}^B|} \right) \\ &= \frac{1}{o} \sum_{l=1}^o \left(\frac{|f_A(S_k^A)| \operatorname{sgn}(W_{kl}^B) \Delta S_l^B}{\sum_{m=1}^h |f_A(S_m^A) W_{ml}^B|} \right) \end{aligned} \quad (6.2-3)$$

This equation can be generalised for the whole layer such that

$$\Delta (f_A(S^A)) = \frac{1}{o} \left[\left(\frac{\Delta S^B}{|f_A(S^A)| |W^B|} \right) \operatorname{sgn}(W^B)^T \right] * |f_A(S^A)| \quad (6.2-4)$$

where both the division inside the brackets, and the multiplication denoted by $*$, are performed element by element.

Finally, the adjustments to the weights matrix connecting the inputs to the hidden layer (W^A) are determined. Again, an appropriately biased value for the desired adjustments to the summed inputs of the hidden layer can be obtained by specifying

$$\Delta S^A = \begin{cases} \Delta(f_A(S^A)) & \text{if } f_A \text{ is strictly increasing} \\ -\Delta(f_A(S^A)) & \text{if } f_A \text{ is strictly decreasing} \end{cases} \quad (6.2-5)$$

Thus the adjustment to the weight associated with the link from input j to the hidden layer node k is obtained from

$$\Delta W_{jk}^A = \frac{\eta \operatorname{sgn}(x_j) |W_{jk}^A| \Delta S_k^A}{\sum_{m=1}^I |x_j W_{mk}^A|} \quad (6.2-6)$$

The inclusion here of a learning coefficient (η) of less than unitary value, increases the probability of the training process being stable (successively reducing the network error cost) and thus ultimately producing the optimal network state.

The above equation can be generalised for the whole weights matrix, such that

$$\Delta W^A = \eta \left[\operatorname{sgn}(x^T) \left(\frac{\Delta S^A}{|x| |W^A|} \right) \right] * |W^A| \quad (6.2-7)$$

As with the equations which describe the learning algorithms in Section 3.4, Eqns. 6.2-4 and 6.2-7 can both be generalised for batch training, where the whole training set is processed in parallel, by replacing the row vectors with corresponding matrices (that contain one row for each pattern) and dividing by p , the number of patterns.

6.3 EMPIRICAL RESULTS

This section details the training of two very different types of network in order to compare the batch forms of PERAL and Backpropagation. The first system examined, that of object categorisation, is a classification problem with ideally discrete outputs. In contrast, the second system, the PUMA 560 simulated inverse dynamics, is a continuous output modelling problem. Consequently, together these provide examples from both sides of the most significant distinction between network types. Historically, the theory of adaptive networks was developed mainly for use on classification problems, but more recently, has been increasingly employed on systems with analogue outputs.

6.3.1 Ultrasound Object Identification

This problem is concerned with the identification of three dimensional objects encountered by a mobile robot [Sillitoe & Elomaa 1994]. An ultrasound emitter mounted

on the robot periodically sends out pulses into the area in front of the vehicle, two receivers mounted equidistantly on either side of the emitter then pick up the signals reflected back towards the robot by any objects within this detection field (see Fig. 6.3-1). The aim is to train an adaptive network to recognise objects and surfaces, by correctly classifying them as one of ten broadly defined categories, from data of the reflected signals picked up by the receivers.

To create the data set the robot was kept stationary and its emitter fired just once per pattern. For each pattern, a single object or structure which clearly fell into one of the ten chosen object categories was placed within the detection field. To create the network input values, the two received signals were processed in order to provide the angle to which the object lies off the robot's centre line, the distance from the robot to the object and a measure of the total power of the signals received. In addition to these, the two signal profiles were broken down into 11 coefficients each, to provide data on the shape (over time) of the received signals. Thus each training pattern consists of 25 input values and 10 target outputs (one for each object category), where only one of the targets will have a value of 1, denoting the correct category, whilst the other nine will have values of -1. In total, 5000 patterns were recorded, of which 3150 were used as the training set (315 of each object type) and the rest formed the comparative set.

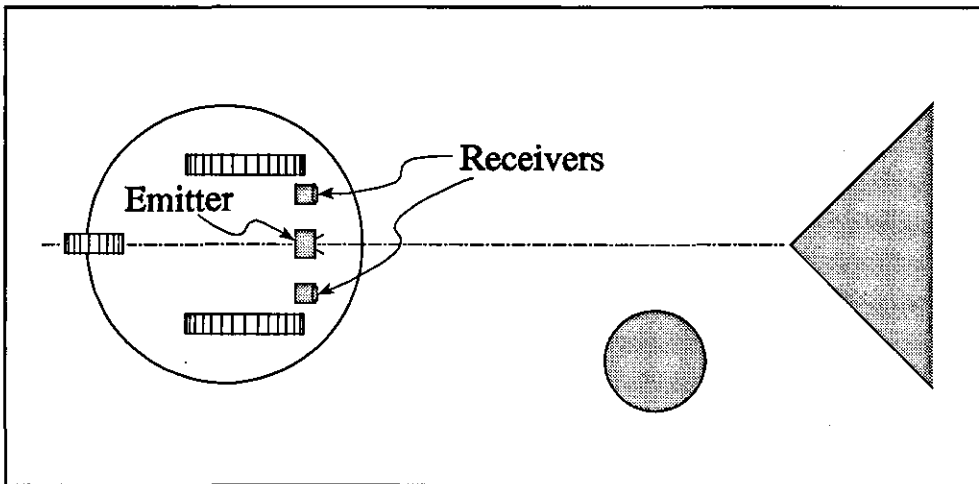


Figure 6.3-1: Plan view of mobile robot with a 90° convex edge directly in front of it and a pole slightly to its right

Several different network architectures were tested, with varying degrees of success. For ease of analysis, the simplest of these architectures is discussed here. The network consists of just one fully connected weights matrix connecting the 25 inputs and one bias term to 10 output nodes with hyperbolic tangent activation functions, as shown in Fig. 6.3-2. The activation functions are incorporated so as to restrict the network output values to a finite range, as well as to polarise the outputs between the “true” and “false” values.

For the task of gauging whether the network has correctly identified a given input pattern, a simple winner-takes-all system is used, such that the category associated with the highest valued output is taken to be the one proposed by the network. However, the ten outputs also provide quantitative measures for the degree of certainty associated with the network's selection. Output values of near 1 or -1 show that the network has decisively selected or ruled out a category, whilst near zero values demonstrate uncertainty. Both Backpropagation and PERAL require this quantitative measure of error in order to induce learning.

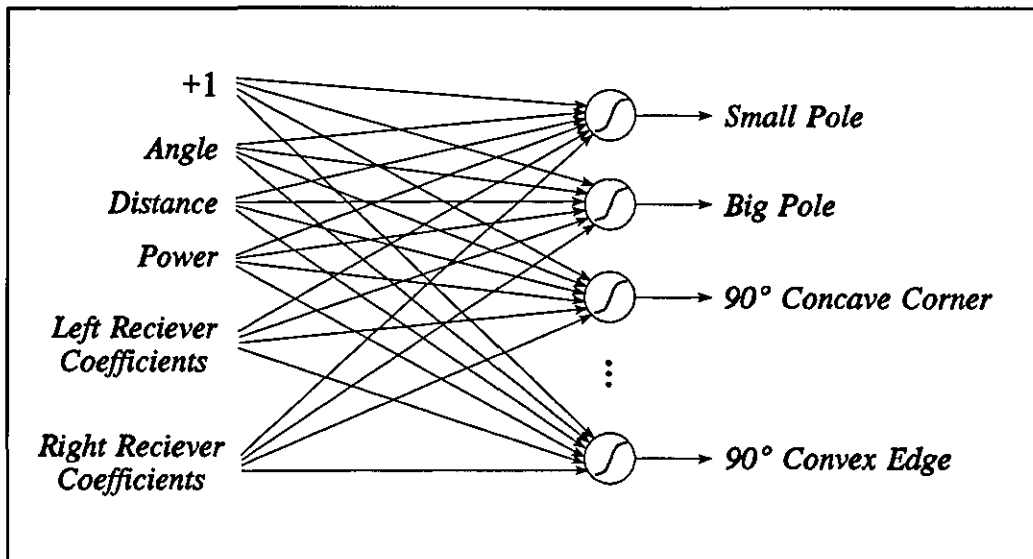


Figure 6.3-2: Ultrasound object identification network

Several training sessions were performed, using random initial weight values. To compare the fine-tuning performance of PERAL, the weight values learnt from incomplete Backpropagation training were recorded at the point that the learning network first achieved a mean correct classification rate of 30% or more, and then used as the initial weights for PERAL training. To ensure that both learning algorithms made weight adjustments at their optimal (highest whilst remaining stable) rate, the learning parameter (η) was modified before each epoch such that if a 10% increase in η produced a further 8% or greater reduction in the mean error modulus then the learning rate was increased by 10%, or if a 10% decrease in η caused any further reduction in the mean error modulus then the learning rate was reduced by 10%. This approach increased the actual time spent calculating each epoch, but maximised the error reduction per epoch caused by each learning algorithm.

As could be expected from such a simple network architecture, the final network solutions obtained are not completely successful in categorising all of the training objects. (Certain, more complex, architectures have achieved 100% correct categorisation

[Mulvaney & Sillitoe 1995]). However, a high error optimal solution is ideal for testing the ability of PERAL to learn from training data with significant disturbance terms.

A plot of the correct classification percentage for a typical comparison experiment is shown in Fig 6.3-3. This shows how Backpropagation was initially used to train the network for 41 epochs, at which point the network had a correct classification rate of 30%. Both Backpropagation and PERAL were then used to train from this point, with PERAL showing a markedly faster progression towards the optimal solution region. If the end of training is defined as the point when the classification rate of the training set reaches 50%, then, for this particular comparison, the Backpropagation training took a total of 382 epochs to complete, meaning that the fine-tuning section of the learning was approximately five times faster (measured in epochs) under PERAL.

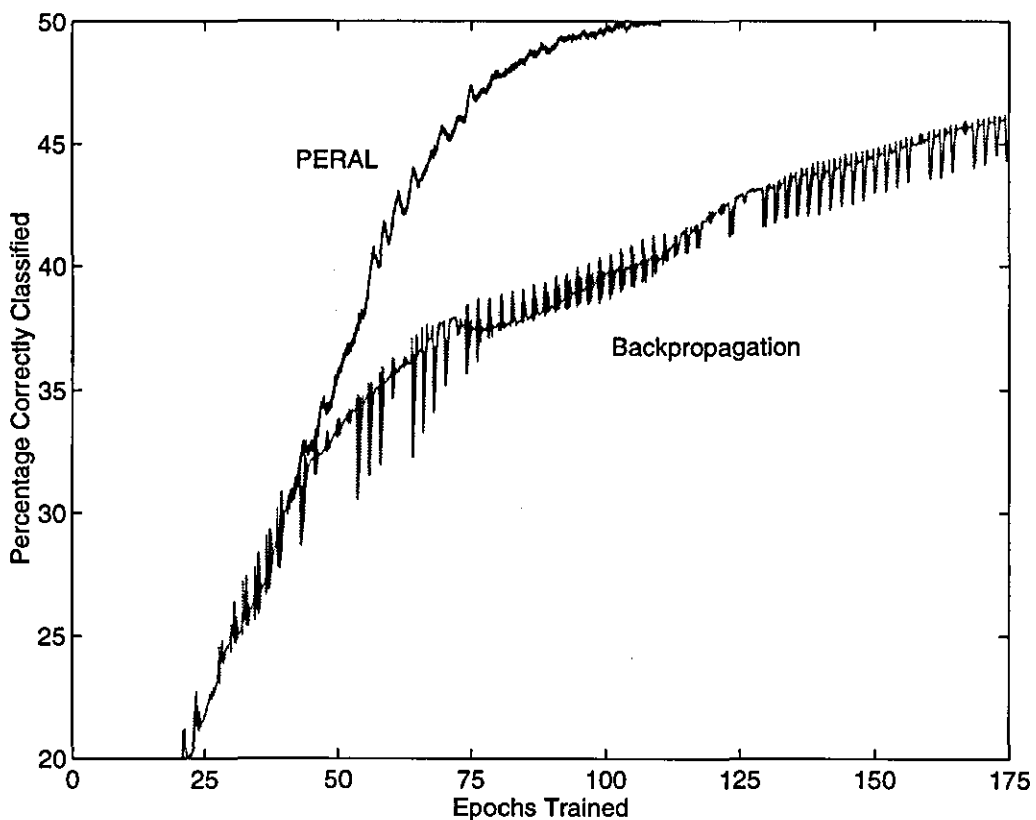


Figure 6.3-3: Comparative plot of the error curves produced by PERAL and Backpropagation when applied to the ultrasound object identification problem

6.3.2 Simulated Manipulator Inverse Dynamics

This subsection compares the training under PERAL and Backpropagation of the CSLC network described in Chapter 5, using data obtained from a simulation of the PUMA 560's inverse dynamics (basic model taken from [Armstrong et al. 1986] with friction

components from [Leahy & Saridis 1989]). A full discussion of the issues involved in training the CSLC network are presented in Chapter 7, where the added complexities of empirical data gathering can also be addressed. For now though, it is important only to note that the PUMA 560 simulation contains a static friction term, whereas the CSLC network does not, thus providing a source of correlated noise.

Multiple training sessions were performed, with different levels of added noise and different initial weight values. The same method as described in subsection 6.3.1 was employed to ensure optimal rates of learning for both learning algorithms, though the CSLC network employed five separate learning rates, one for each subnetwork. Fig. 6.3-4 shows a typical comparative error plot of a training attempt, where both the (unprocessed) input data and target torques data contain large Gaussian noise terms.

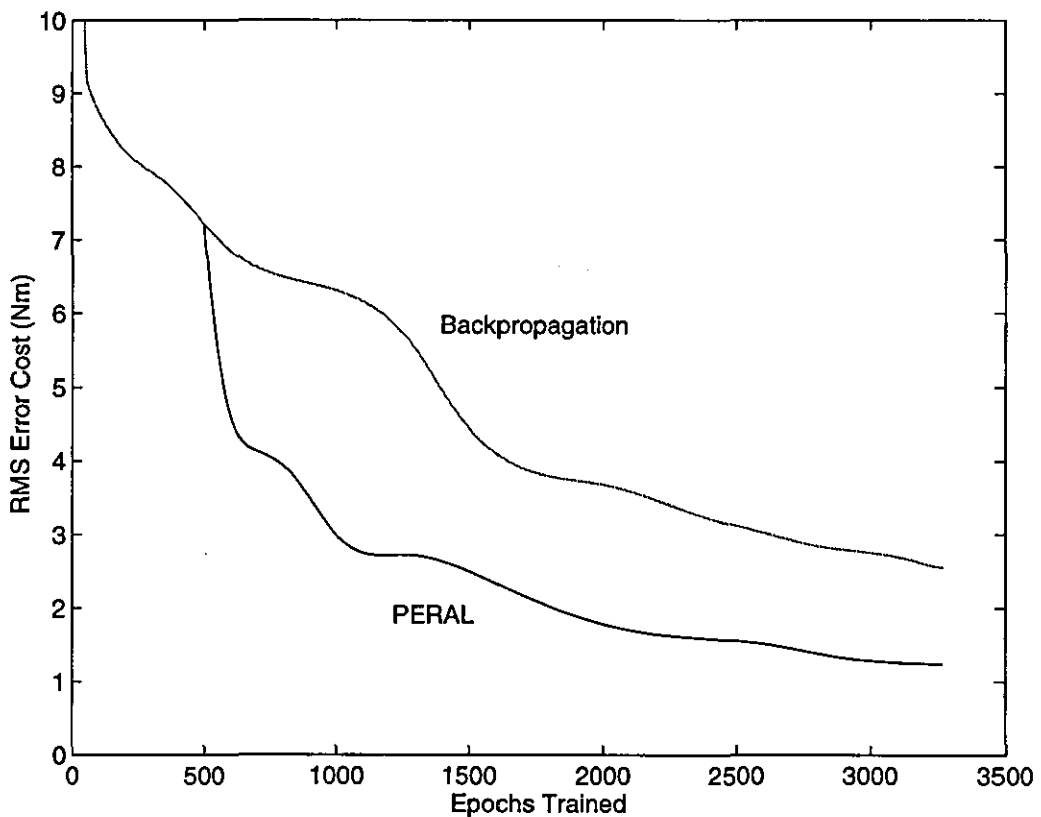


Figure 6.3-4: Comparative plot of the error curves produced by PERAL and Backpropagation when applied to the simulated inverse dynamics problem

For the training comparison shown in Fig. 6.3-4, the PERAL algorithm was applied after 500 epochs of training by Backpropagation. As with all instances where PERAL was introduced at this point or later, it immediately made a significant and sustained reduction in the mean network error cost.

6.3.3

Discussion Of Results

The Proportional Error Allocation algorithm clearly has potential for shortening the network training process, in terms of the number of epochs of learning required to reach a suitably low average error. The actual time taken to perform each epoch will depend on a number of factors, such as the speed of the computer used, but any difference in learning algorithms' computation time will be largely due to the number of mathematical operations each requires. Operations such as floating point multiplications or additions take significantly more time than operations such as signum or modulus functions, therefore, for comparing the relative time taken to perform either Backpropagation or PERAL, only the floating point operations are considered. Similarly, the operations required to determine terms whose values do not vary between epochs are also not counted, as it is assumed such values are only calculated once in the entire training process. Furthermore, values derived in the calculation of the network's outputs, during what is often called the *feedforward phase* of training, are assumed to be retained if used by the learning algorithm during the weight adjustment (*feedback*) phase. For the general two-layer network, each of the nodes is assumed to contain a hyperbolic tangent, or other similar activation function, whose derivative is easily expressible in terms of the original function and therefore requires only two floating point operations per pattern to calculate (if this is not the case then the performance of the Backpropagation algorithm may require considerably more floating point operations). Note that the values in the table below (Fig. 6.3-5) include the number of floating point operations performed in the feedforward phase of learning.

Network Type	Backpropagation	PERAL
General Two-Layer (Fig. 6.2-1)	$4ih + 7h + 2ho + 9o$	$7ih + 4h + 6ho + 4o$
Object Identification (Fig. 6.3-2)	1080	1850
PUMA 560 CSLC (Figs. 5.1-1 to 5.1-6)	2904	4875

Figure 6.3-5: Total no. of floating point operations required to perform one iteration of the indicated learning algorithm upon one training pattern (i = no. of inputs, h = no. of hidden nodes, o = no. of outputs)

It can be seen from the table that, for most network architectures, PERAL requires more floating point operations to perform than Backpropagation. The exact comparative ratio

depends on the particular network considered. However, in the two test cases, PERAL's superior training rate per epoch more than made up for its lower rate of epochs per second.

An important property of the PERAL algorithm is that it does not rely upon the error hypersurface gradient to provide information on weight adjustments, and is therefore not affected by flat-spots in the error hypersurface. Furthermore, as the network approaches its optimal solution state, the estimate-based PERAL algorithm (Eqn. 6.2-2) more closely approximates the ideal algorithm (Eqn. 6.1-2). Thus, PERAL's learning rate parameter can be increased as training advances, whilst maintaining stable successive error cost reduction. This is in marked contrast to Backpropagation, where the learning rate must necessarily be reduced as training advances, in order to maintain the same stable learning.

6.4 SUMMARY

A novel learning algorithm has been introduced, based upon adjusting a network's weights in accordance with the relative magnitude of the corresponding links contribution to the network outputs. The beneficial consequences of employing such Proportional Error Allocation learning were examined under the theoretical case of the optimal weight values being known, and the algorithms robustness to noise evaluated.

A practical implementation of the PERAL principle was developed in Section 6.2 by approximating the optimal weight values with the current estimates. This approximation limits the operational use of the algorithm to the fine-tuning of a given network subsequent to some other means having been used to identify an approximate solution. However, by considering the results shown in the two case studies, it can be seen that this region of adequate solution approximation can be substantial, and that the use of PERAL can greatly hasten training compared to the Backpropagation algorithm.

The benefits of the PERAL algorithm not being based upon gradient descent of the network error hypersurface were discussed. It was also noted that, in contrast to gradient descent techniques, the constructive weight adjustments performed by PERAL do not become vanishingly small as the network approaches its optimal state.

Up until now, the theory surrounding the identification of a manipulator's dynamics has been almost entirely developed in either a theoretical or simulated context. Given the information and knowledge that has been acquired, it will be possible in the next chapter to progress the examination of the identification problem into the context of empirical experimentation and practical results.

REFERENCES AND FURTHER READING

- Armstrong, B., Khatib, O., and Burdick, K., "The explicit dynamic model and inertial parameters of the PUMA 560 arm," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 510-518, 1986.
- Battiti, F., "First and second order methods for learning: between steepest descent and Newton's method," *Neural Computation*, vol. 4, pp. 141-166, 1992.
- Fahlman, S.E., "An empirical study of learning speed in back-propagation networks", Tech. Rep. CMU-CS-88-162, Carnegie-Mellon University, Dept. of Computer Science, Sept. 1988.
- Leahy, M.B., Jr., and Saridis, G.N., "Compensation of industrial manipulator dynamics," *Int. J. Robotics Res.*, vol.8, no.4, pp. 73-84, 1989.
- Moller, M.F., "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525-533, 1993.
- Mulvaney, D.J., and Sillitoe, I.P.W., "A comparison of neural network and pattern matching approaches to the classification of ultrasonic signals," *Signal & Image Processing*, Las Vegas, Nov. 1995.
- Riedmiller, H., and Bravin, H., "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," Ruspini Ed., *Proc. IEEE Int. Conf. on Neural Networks*, pp. 536-591, San Francisco, 1993.
- Sillitoe, I., and Elomaa, T., "Learning Decision Trees for Mapping the Local Environment in Mobile Robot Navigation," *Proc. Machine Learning - Colt Workshop on Robot Learning*, pp. 119-125, July 1994.
- Vogl, T.P., Mangis, J.K., Rigler, A.K., Zink, W.T., and Alkon D.L., "Accelerating the convergence of the backpropagation method," *Biological Cybernetics*, vol. 59, pp. 257-263.

7. COEFFICIENT IDENTIFICATION FOR A PUMA 560

This chapter details and discusses a coefficient identification experiment, performed on two real PUMA 560s, using the CSLC network developed in Chapter 6. The hardware involved in the experiment is first described, then the potential sources of disturbances to the Euler-Lagrange representation of the inverse dynamics are reviewed. Of particular concern is the need to derive the manipulators' joint velocities and accelerations from the sampled joint positions, as this is a notorious source of input noise. In order to address this, a novel application of spline fitting is developed, which is capable of smoothing the quantised joint position profiles and can be algebraically differentiated.

The requirements of a good data set, and the measures taken in order to achieve them, are discussed in Section 7.4. The details of how training was applied are presented in Section 7.5, and the results of the identification experiment are discussed in Section 7.5, where comparisons are made with coefficients identified from direct measurement and a combination of direct measurement and regression.

7.1 EQUIPMENT

The two PUMA 560 Mk. 2 manipulators under investigation reside at the [Robotics Institute] of Carnegie Mellon University. Data on the manipulators' motion was kindly collected by Dr. Richard Voyles, a then Ph.D student at the institute. Coefficient identification was performed on two identical manipulators so that it would be possible to verify the results obtained through comparison.

Both manipulators are controlled by TRC004 PUMA interface boards, these allow low-level control of the manipulators' motion [TRC004 User's Manual]. Combined with a real-time computer, the TRC004 replaces the archaic LSI/11 controller that PUMA 560 Mk. 2s were originally supplied with. The TRC004 interface board is manufactured by [Trident Robotics and Research Inc.], a spinoff of Carnegie Mellon and Stanford Universities that produces various low-cost products aimed at the robotics research community. The TRC004 consists of an I/O board with A/D converters, D/A converters,

encoder counters and digital I/O lines, capable of connecting to several bus architectures including VMEbus, IBM-PC bus, Multibus and IndustryPack bus. It allows direct control of the PUMA 560's six motors, and measurement of the joint positions.

Position encoders are attached to all of the manipulators' motors. Under ideal transmission conditions, that is, in the absence of backlash or mechanical compliance in the transmission components, the encoders have joint position measurement precisions of [0.10035 0.073156 0.11700 0.082663 0.087376 0.081885] milliradians respectively (from joint one to six). Note that these values are encoder precisions, expressed in terms of the joint positions, that is, multiplied by the gear ratio r .

7.2 SOURCES OF NOISE

There are several known sources of disturbance to the Euler-Lagrange equation (when applied to a real manipulator) which have the potential to cause noise in the data collected. These include the principal inertia substitution within the pseudo-inertia matrices, the lack of a static friction term, the motor torque generation linearity limits, backlash, compliance, quantisation and estimation of the joint velocities and accelerations. Each of these is discussed below.

7.2.1 The Principal Inertia Substitution

In the development of the kinetic energy term for a general manipulator (Subsection 2.3.2), it was useful to define the pseudo-inertia matrix for each link in terms of their mass moments of inertia, products of inertia, and first moments (Eqn. 2.3-17). It was then shown how the expression for any given pseudo-inertia matrix could be simplified, by use of the parallel axis theorem (Eqns. 2.3-18 & 2.3-19), so as to contain only the principal mass moments of inertia at the body's centre of gravity and the first moments (Eqn. 2.3-20). As stated in Section 2.3, this simpler representation is only correct when the axes of the coordinate frame attached to the body are parallel to the corresponding principal axes at the body's centre of gravity. However, in the case of manipulator links, these principal axes are usually impossible to determine exactly. In practice, it is commonly assumed that the principal axes at the link's centre of gravity are closely approximated by a set of axes, two pairs of which each describe a plane of symmetry in the link's external dimensions. This estimated set of axes and the true principal axes only ever coincide exactly when the given link is mass symmetric in the same planes as its exterior dimensions are volumetrically symmetric. As this is extremely unlikely in a real manipulator, this simplification introduces errors into the Euler-Lagrange equations, caused by the existence of neglected products of inertia in the manipulator's true inverse

dynamics for the frames chosen.

This simplification has in the past been universally used during the derivation of specific manipulators inverse dynamics, the reason for this is the difficulty in finding a link's products of inertia. The inertia pendulum method [Armstrong et al. 1986] is the most practical way of determining the inertia characteristics of a link, once it has been detached from the rest of the manipulator, but this only provides information on the mass moment of inertia about a chosen axis. Therefore, when using the inertia pendulum, it is assumed that the chosen axis is a principal axis of the link, because if it is not, there is no practical way to measure the arising non-zero products of inertia.

Given that the method of coefficient identification used here is not dependent upon direct measurement of physical parameters, it would be a simple matter to develop the Euler-Lagrange equation without the approximation presented by the principal axes theorem. This would result in a slightly more complex representation of the manipulator's inverse dynamics, for which a CSLC network could be just as readily formulated as for the simpler version so far discussed. However, in order for the parameter values obtained from the identification method to be directly comparable to those obtained from direct measurement, the less accurate model used by other researchers (specifically [Armstrong et al. 1986]) is adopted. The errors caused by using the simplified inverse dynamics will act as noise in the coefficient identification process. However, the cross products of inertia at the links' centroids (the neglected quantities) will always be small compared to terms such as the moments of inertia (see Eqns. 2.3-14 & 2.3-15), regardless of the axes orientation chosen. Thus the effect of the principal inertia substitution is expected to be correspondingly small.

7.2.2 Static Friction Term

The other known deficiency of the CSLC network's representation of the PUMA 560's inverse dynamics is the lack of a static friction term. For most of the problem space this term is negligible (see Subsection 2.5.1), but at low joint velocities, it can begin to have significant effects. As discussed in Subsection 2.5.1, the estimation of link velocities from the corresponding link positions, as is required in this case study, causes the sign of a given joint's static friction term to be in doubt at the only time its magnitude is not vanishingly small. It has therefore been decided not to include a model of the static friction in the CSLC network, as it would be unlikely to reduce the overall modelling error.

The disturbance within the training data caused by the omission of a static friction model is not expected to significantly affect the training of the mass, gravity of velocity sections

of the CSLC network. This is because the mass and gravity terms are uncorrelated with joint velocities, thus, if the training set contains a roughly equal number of low velocities in the positive direction as for the negative direction, for all joints, then the averaged error will be approximately zero. The velocity term is effectively uncorrelated as it approaches zero for low joint velocities. However, the viscous and dry friction terms are clearly correlated, so to avoid affecting their training a threshold velocity magnitude is defined for each joint, below which static friction is significant. During training, all patterns possessing velocity magnitudes below this threshold are neglected for the purposes of training the viscous and dry friction subnetworks only. As the frictional effects are decoupled, different patterns can be neglected for different joints. By consideration of Fig. 2.5-1 it can be seen that, without the use of suitable thresholds, the corresponding subnetworks are liable to learn erroneously low values for the viscous friction coefficients and erroneously high values for the dry friction, in order to minimise the training set's error cost.

7.2.3 Motor Linearity Limits

The electric motors within a PUMA 560 are supplied by the manufacturer with accurate ratings for their torque generation per unit of power. However, this linear relationship between torque and power is only true for a fraction of the motor's range. As can be seen from Fig. 7.2-1, the torque-power relationship for a given motor is linear at low demands, but starts to produce progressively less torque per unit power at higher demands. The point at which the relationship stops being linear is known as that motor's limit of linearity.

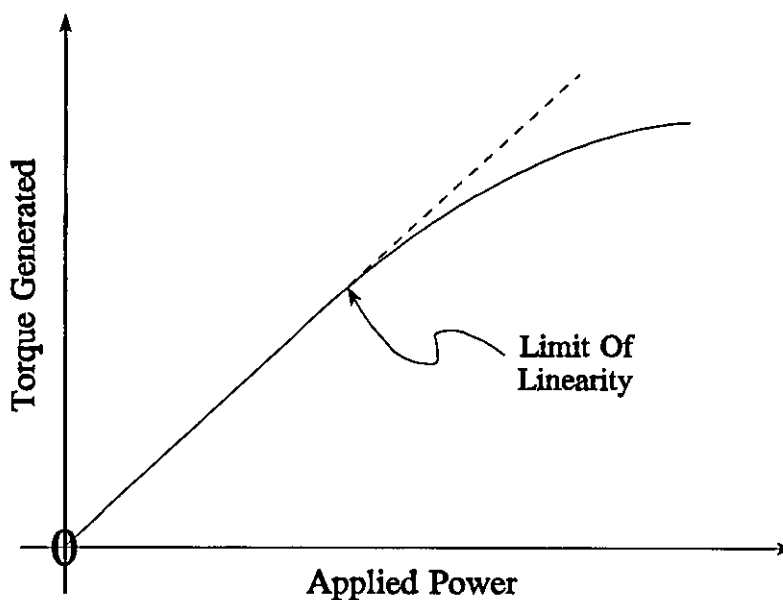
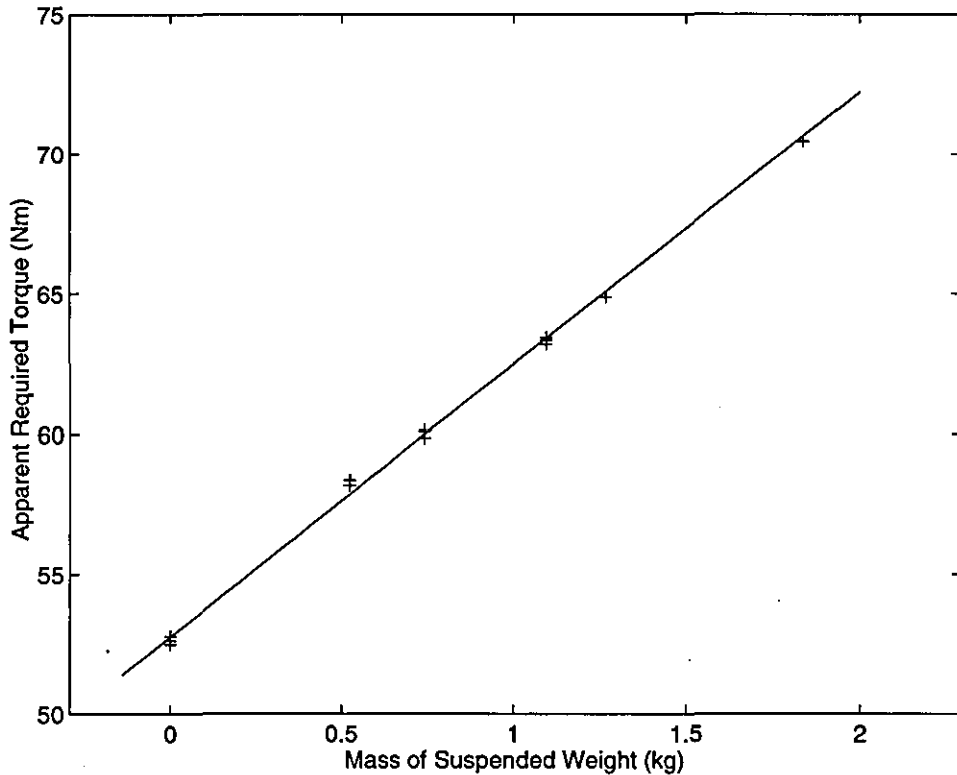


Figure 7.2-1: Motor response curve with limit of linearity

Therefore, in order to not introduce errors between demanded and generated torque, the torque-power relationship of the PUMA 560s motors must be checked, to ascertain their limits of linearity. The PUMA 560s' motors are all of the same design, but vary in rated maximum output, it is therefore reasonable to assume that they share approximately the same torque-power profiles, when scaled down by their maximum output, and that their limits of linearity are similarly related. Thus an experiment was devised to check the motor response of joint 2, which would act as the datum for approximating the limits for the rest of the motors.

The experiment consists of locking all of the joints except joint 2 with the arm pointing directly outwards, suspending a range of weights from the flange of the manipulator's end effector, then using a standard PD controller with gravity compensation (the gravity section of the abbreviated model with relevant parameter values from [Armstrong et al. 1986]) to oscillate link 2 around the horizontal position. The average applied torque required to do this is then plotted against the mass of the suspended weight. The arm is oscillated, rather than simply commanded to be stationary in the horizontal position, so as to average out the effects of friction. If the arm was stationary, one would not know if the frictional effects were acting in the same direction between measurements. It is not necessary to know the distance from the axis of joint 2 to the point through which the suspended weights act, only that it is constant. Clearly, whilst the motor is acting within its range of linearity, the plotted results should lie on a straight line, however, the measured "torques" are in fact measurements of the power required by the motor, expressed in terms of torques (applied at the joint). Thus if a large weight was used that caused the motor to exceed its limit of linearity, it would appear to require more torque per unit of added mass than for lesser weights.

The results of this experiment are shown in Fig. 7.2-2, in which a total of 12 measurements are plotted, including three where no weight was used. The best-fit line (in a least squares sense) of the measurements pertaining to the 11 lightest weights is also shown. If the motor's limit of linearity was less than the torque required by the heaviest weight, then the point on the graph corresponding to that weight would be expected to appear above the best-fit line. As this is not the case, it is reasonable to assume that the limit of linearity lies beyond the torque range of this experiment. Furthermore, as the required torques for the data gathering trajectories (see Section 7.4) do not exceed this range, or its equivalent for joints other than number two, it is considered unnecessary to investigate the limits of linearity further.



*Figure 7.2-2: Motor response for joint of a PUMA 560
(best-fit line found for all points except the furthest right)*

As a double-check, the same experiment was performed on joint 3, with similar results.

7.2.4 Backlash and Compliance

The joint angular position encoders on the studied PUMA 560s are attached directly to the respective motor shafts, thus creating the potential for positional errors whilst joints are experiencing backlash. To reduce the effects of backlash, forces are applied to the cogs in each of the geartrains, pushing them against each other at the cogteeth interfaces. This technique is known as *overmeshing*: it can greatly reduce the degree of backlash in a manipulator's joints, but it also increases the effective friction, and, at high levels, can increase the degree of mechanical compliance. A moderate level of overmeshing is employed on the PUMA 560s, such that any offset between measurements of a given joint's position taken before and after a backlash period (b_i) are assumed to be small.

However, as discussed in Subsection 2.5.3, it is important to identify when a joint is experiencing backlash so that the characteristics of non-driven motion are not learnt by the CSLC network. As joints are likely to experience backlash at different times from one another, disregarding a whole training pattern because of backlash in one or more of the

joints is likely to greatly increase the amount of raw data required. However, the disturbance to the inverse dynamics caused by backlash is decoupled, that is, if a given joint is experiencing backlash, then there will be a reduction in the effective values of the friction coefficients and the motor inertia for that joint only. Therefore, it is possible to use all the available training patterns by ignoring the network output errors (for a given pattern) of only those links identified by Eqn. 2.5-7 as being likely to be experiencing backlash.

Disturbances due to mechanical compliance can be largely characterised by two types of phenomenon, those of sag and elasticity. Sag represents the time independent deformation of the manipulator due to its configuration, such as bowing of a link (or more plausibly, its actuator) due to the weight acting upon it. Given the PUMA 560's metallic construction, the degree sag is expected to be small. Elasticity within a manipulator's components cause its links to oscillate about their steady-state positions, as such, its effects are clearly time dependent. However, as they are not significantly correlated with any of the CSLC network's inputs, the effects of elasticity are expected to cancel out when considered over the whole of a training set.

7.2.5 Quantisation And Input Estimation

The link positions are measured in terms of movement counts of the corresponding encoders, thus the values for joint positions are integer multiples of the measurement precisions stated in Section 7.1. This causes what is known as quantisation, where the measured values for a given joint's position are constrained to be members of a set of discrete values, in this case, with a constant interval equal to the encoder precision. Thus, the measured motor positions (evaluated in terms of the joint positions) may deviated from the true positions by up to the value of the corresponding encoder tolerance. Even given that the true measurement tolerances of joint positions will be somewhat higher than the value derived from the encoders' precisions, due to mechanical compliance and backlash in the transmission, the measured values of q , and therefore the preprocessed CSLC network inputs, are acceptably accurate estimations of the true values.

However, the problems caused by the quantisation of q become much more significant when attempting to derive values for \dot{q} and \ddot{q} . Thus, the method for deriving the joint velocities and accelerations from position measurements needs to be carefully considered. For the purposes of demonstrating the potential problems connected with quantisation, consider the estimation of a joint's velocity and acceleration from simple numerical differentiation of neighbouring patterns. If the estimate of a given joint's positions, denoted \tilde{q} , is set equal to the corresponding measured values, numerical differentiation

of the consecutive patterns $n-2$ to $n+2$ can then be expressed as

$$\dot{\tilde{q}}_n \approx \frac{\tilde{q}_{n+1} - \tilde{q}_{n-1}}{2\lambda} \quad (7.2-1)$$

$$\ddot{\tilde{q}}_n \approx \frac{\dot{\tilde{q}}_{n+1} - \dot{\tilde{q}}_{n-1}}{2\lambda} = \frac{\tilde{q}_{n+2} - 2\tilde{q}_n + \tilde{q}_{n-2}}{4\lambda^2} \quad (7.2-2)$$

where

\tilde{q}_n = estimated value of the joint position for pattern n

$\dot{\tilde{q}}_n$ = estimated value of the joint velocity for pattern n

$\ddot{\tilde{q}}_n$ = estimated value of the joint acceleration for pattern n

λ = time interval between measurements

As with any numerical differentiation technique, the estimation accuracy is clearly related to the length of the measurement interval, such that a smaller λ yields greater accuracy. However, it can be seen from Eqn 7.2-1 that the quantisation intervals of \dot{q} will be equal to the quantisation intervals of q divided by 2λ , and in turn, from Eqn. 7.2-2, the quantisation intervals of \ddot{q} will be equal to the quantisation intervals of q divided by $4\lambda^2$. Therefore, if measurements were taken every thousandth of a second, the quantisation intervals for the joint accelerations will be 250 thousand times greater than the those of the positions, and clearly a major cause of noise.

In the following section a technique to be known as spline smoothing is described, which, under certain predictable conditions, eliminates quantisation for joint positions and thus allows accurate estimation of the joint velocities and accelerations.

7.3 SPLINE SMOOTHING OF MEASURED POSITIONS

In Subsection 7.2.5 the need for eliminating quantisation effects from the measured positions was identified. Within this section, a novel application of splines, for the smoothing of quantised data, is discussed. Splines take their name from the thin strips of wood that ship designers would bend between pegs to provide natural looking curves between specific points. In particular, cubic splines provide the most aesthetically pleasing smooth curves between several given points in two dimensions. A cubic spline is made up from a series of cubic curves, each one having different coefficients and smoothly joining to the next in the series at one of the given points. The conventional use of splines, however, is to provide interpolation between known values. In the case of position measurements though, the true values are not known exactly (due to quantisation, backlash and compliance) and the only data available to aid in correcting a given measurement is from other measurements taken around the same time.

The technique devised involves creating cubic splines from different subsets of the measured values, then averaging these splines to create the desired smooth curve. For a given link and smooth trajectory, if one takes every j^{th} point from the set of joint position measurements, then one can create a cubic spline which will exactly pass through the chosen points, whilst closely and smoothly approximating those in between. There are a total of j possible splines thus obtainable, with different choices of the starting point for the spline (the first, second, ... or j^{th} member of the data set). Thus, the average of these cubic splines provides a smooth curve which closely approximates all the measurements, without necessarily passing through any of them.

To demonstrate the usefulness of this technique, consider Figs. 7.3-1 & 7.3-2. The former shows a selection of points whose y-values are the quantised output of a quadratic equation, three cubic splines are also shown, each one fitted to every third point. The range of the data extends either side of the graph, but attention is focused on the saddle point as it is the most difficult part of a curve to approximate from quantised data. Note in particular how quantisation has apparently flattened the original equation's representation at the bottom of the curve.

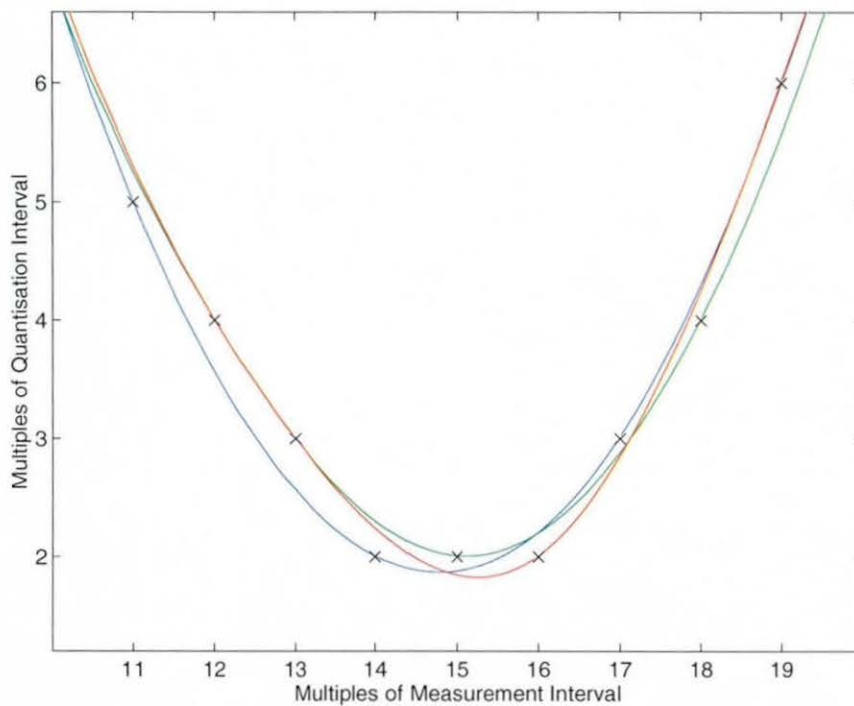


Figure 7.3-1: Three cubic splines passing through three subsets of quantised data

From consideration of Fig. 7.3-2 it can be seen that the average of the three fitted spline curves shown in Fig. 7.3-1 provides a smooth, close approximation to the original (nonquantised) data. In particular, note how the spline smoothing has reversed the flattening due to quantisation of the quadratic equation at the bottom of the saddle point.

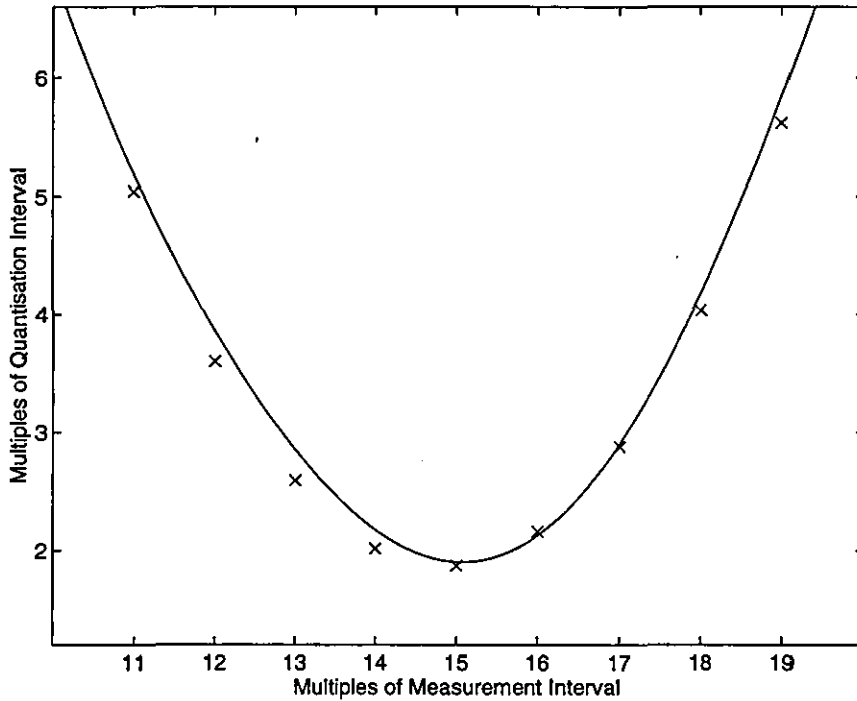


Figure 7.3-2: Comparison of averaged cubic curve and nonquantised data

Now that a suitably accurate and nonquantised approximation of the true joint positions is possible, within certain limitations which are discussed below, it can be used to provide nonquantised estimates of the joint velocities and accelerations. The estimate of a given joint's position profile (\bar{q}) is a cubic expression, with differing coefficients in each measurement interval, such that, for a given pattern n , it can be expressed as

$$\bar{q}_n = \kappa_0 + \kappa_1 n + \kappa_2 n^2 + \kappa_3 n^3 \quad (7.3-1)$$

where

$$\kappa_i = \text{coefficient of the } i^{\text{th}} \text{ order element, for the current measurement interval}$$

Note that as measurement intervals join at measurement points, there will, in general, be two different sets of values that can be employed as coefficients in Eqn 7.3-1.

Estimates of a given joint's velocity and acceleration can be found from the first and second instantaneous differentials of Eqn. 7.3-1. Thus, for the n^{th} estimated position, the following are obtained

$$\dot{\bar{q}}_n = \frac{\kappa_1}{\lambda} + \frac{2\kappa_2}{\lambda}n + \frac{3\kappa_3}{\lambda}n^2 \quad (7.3-2)$$

$$\ddot{\bar{q}}_n = \frac{2\kappa_2}{\lambda^2} + \frac{6\kappa_3}{\lambda^2}n \quad (7.3-3)$$

Clearly j , the number of splines used, has an effect on the approximations obtained. The larger the value of j , the less sensitive the resultant approximation will be to changes in the data that occur over short periods. This insensitivity will make the approximations robust to noise, but if j is set too high, the approximations may fail to capture legitimate features of the underlying data, for example, approximating acute peaks by more shallow ones. Thus the value of j should be adjusted to best suit the data being examined.

The primary limitation of the spline smoothing technique is that to smooth a given pattern requires information from patterns measured both before and after it. Therefore, this technique can not be used in real-time, that is, smoothing can only be performed sometime after the measurements are taken. However, this does not present a problem for the currently considered application of smoothing joint position measurements in order to create the input data for the CSLC network.

A second limitation is that the technique requires the underlying curve that it approximates to be continuous and smooth. Fortunately, in the case of joint positions and velocities, their true values are guaranteed to be continuous. Any lack of smoothness is related to discontinuities (within the measurement timescale) in the acceleration profile. When controlling a manipulator, the signal that regulates how much torque the motors apply to each joint is updated at discrete intervals, causing the motor torque profile (against time) to be a series of steps. This will theoretically cause the joint accelerations to be step-like too. However, compliance within the transmission is liable to cause a degree of smoothing in the torque profile applied to the manipulator joints. What is more, if the torque update interval is set equal to the measurement interval, then the acceleration profile will appear continuous. Any errors arising from the treatment of a series of steps as a smooth curve are expected to be small if the measurement interval is small, as well as averaging to zero if the mean of the acceleration profile's derivative is zero (see Subsection 7.4.1).

Another, potentially more serious, source of discontinuity in the acceleration profiles is the effects of friction. If one considers Eqn. 2.5-4, it is clear that both the dry and static friction components are discontinuous, taking the same signs as the corresponding link velocities. In particular, static friction is expected to be the greater contributor to the discontinuity that occurs in a joint's acceleration as that joint's velocity traverses zero, due to being closely modelled by an exponential function of \dot{q} . So as not to remove information on such events from the estimate of \ddot{q} , the angular position profiles for each joint are subdivided into sections with either monotonically increasing or monotonically decreasing changes in value between sample points. These are then smoothed separately, such that the resultant smoothed velocities for each section all share the same sign, and discontinuities can exist in the acceleration profiles at the points that the corresponding velocity profiles transverse zero.

7.4 DATA GATHERING TRAJECTORIES

The trajectories performed by the manipulators whilst gathering data are discussed in this section. The distinction is made between the trajectory chosen for the process of gathering training data, and the trajectory used for testing the model represented by the trained CSLC network. The former is designed to minimise any disturbance in the training set obtained, whilst the latter is representative of the most common class of trajectory demanded of this type of manipulator.

In both cases the encoder values were sampled every two thousandths of a second, at (effectively) the same instants as the motor demands were updated. This means that the applied torques are theoretically discontinuous at the sampling points. However, in practice, the change in the torque applied by each of the manipulators' motors at each update point is not instantaneous, primarily due to mechanical compliance in the transmission. An (exaggerated) example of an applied torque profile is shown in Fig. 7.4-1. Assuming that there is no error between the applied torques and those demanded by the controller, within a period equal to the sampling/update interval, it can be seen that the applied torque that generates the manipulator configuration sampled at n , is equal to the torque demanded at $n-1$. Thus each pattern processed by the CSLC network consists of input values derived from a given sampling/update point, and the demanded torque values for the previous point.

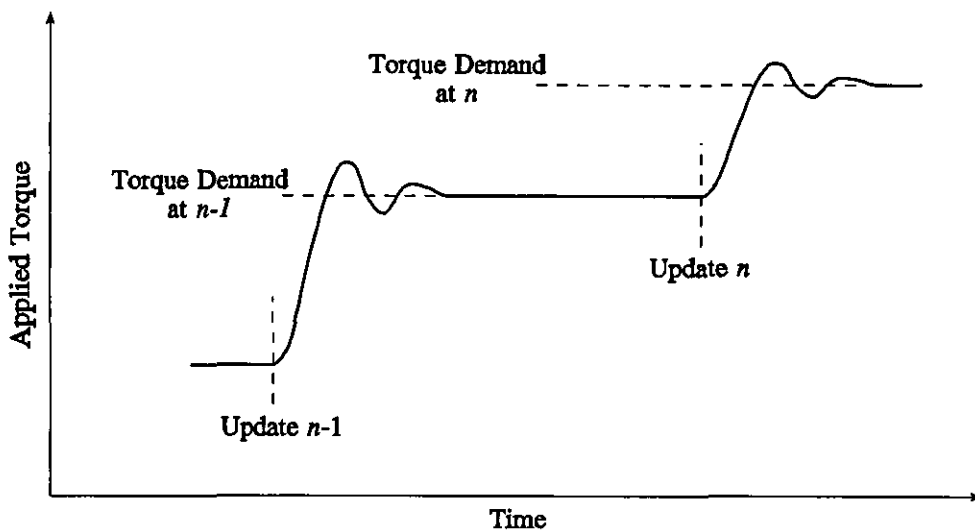


Figure 7.4-1: Comparison between demanded and applied torques

7.4.1 Training Trajectory

As discussed in Sections 6.1 & 7.2, the degradation of the CSLC network's learning caused by many of the disturbances to the manipulator model can be greatly reduced or eliminated if the training data contains disturbance terms of zero mean. The errors caused by the principal inertia substitution (Subsection 7.2.1), the lack of a static friction term (Subsection 7.2.2) and backlash offsets (Subsection 7.2.4) are all configuration dependent, that is they vary with the manipulator joints' positions, velocities and/or accelerations. Thus, a simple method for minimising the effects of disturbance terms is to design a trajectory where, for each joint, the sections of the velocity and acceleration profiles in the positive direction are matched as much as possible by counterpart sections in the negative direction. Practical considerations require that the trajectory's initial velocities are zero, which implies that the position profiles are matched about their mean values in a similar manner to the velocities and accelerations. Furthermore, as discussed in Section 7.3, it is also desirable for the acceleration profiles to be smooth. Therefore, the trajectory chosen for accumulating training data consists of a sinusoidal path for each link, such as that shown in Fig. 7.4-2.

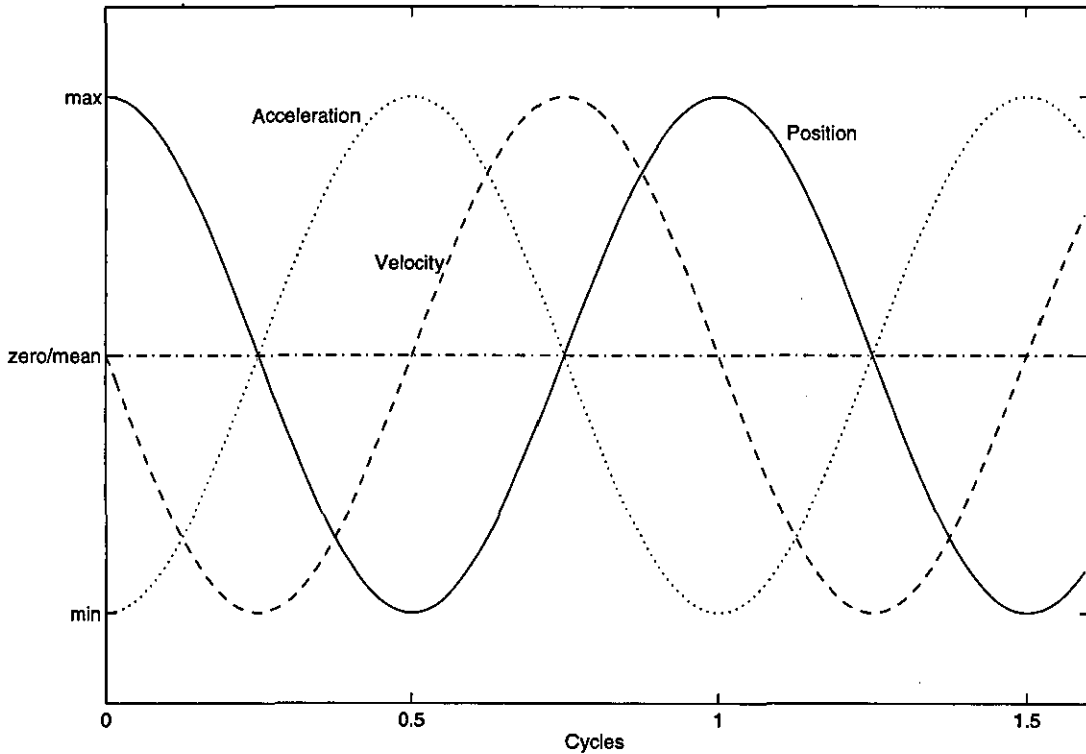


Figure 7.4-2: Desired sinusoidal position, velocity and acceleration profiles for a given joint (centre-line denotes zero velocity & acceleration, and mean position)

To calculate an applied torque profile that would perform such a trajectory precisely would require an accurate model of the manipulator's dynamics, which of course is not available prior to the accurate identification of its dynamics coefficients. However, it is known that the highly significant elements of the mass matrix lie on its leading diagonal (as these represent the effective link inertias, as opposed to the off-diagonal coupling inertias). Thus, the motion of the PUMA 560's joints can often be thought of as being largely decoupled. Furthermore, if a sinusoidal trajectory was approximated, it is clear that the frictional effects on each joint would average out as zero over time. Therefore, approximately sinusoidal acceleration profiles can be obtained by employing applied torque profiles which consist of sinusoidal terms plus some form of gravity compensation.

A further important consideration for a training trajectory is its degree of *excitation* [Craig 1986, Sastry 1984], which can be thought of as a measure of disparateness in the patterns obtained, and is related to the training efficiency and immunity to noise of the training set [Armstrong 1989]. It is therefore desirable to maximise the degree of uniqueness in the patterns obtained from the sinusoidal acceleration based training trajectory by ensuring that no configurations are repeated, thereby increasing the training set's degree of excitation. For the chosen training trajectory, the position, velocity and acceleration of each joint will be approximately cyclic, thus different cycle periods for each joint are employed to enlarge the unique configurations span within the trajectory. The use of prime number factors of a base time unit for the period lengths ensures the absence of configuration repetition for the maximum span, the length of which is the product of all the periods, measured in terms of the base time unit. To further increase the disparateness of the information on the dynamics' configuration dependency incorporated within the training data, the torque profile amplitudes are adjusted between successive trials, so that ultimately, each joint sweeps the majority of its positional range during the trajectory.

Therefore, the training trajectory employed with the PUMA 560s consists of sinusoidal applied torques with amplitudes of [25 13 14 8 10 60] Nm, initial phases of $\frac{3\pi}{2}$ rads, and periods of [4.1 2.7 1.7 1.1 7 5] s, with the addition of the abbreviated gravity compensation model proposed by [Armstrong et al. 1986]. The initial joint positions are [0.4 -1.2 0.3 0 0 0] rads. Due to the coupled dynamics of the manipulator, and the inaccuracies of the gravity compensation, the motion of the manipulator's joints will become non-sinusoidal with time. Therefore, as a safety measure, a further term is added to each joint's applied torque which is vanishingly small throughout the majority of a given joint's workspace, but increases sharply to repulse a joint as it approaches either a limit of its positional range or a position which could potentially cause the end-effector to strike either the ground or the manipulator itself. The theoretical span of unique configurations from this trajectory is over eight days in length, in practice, the number of measurements that can be taken is constrained by the rapidity

with which the sinusoidal motion breaks down and the size of the recording buffer's memory. In this case, a total of 3300 patterns were taken, all from the approximately sinusoidal section of the trajectory. Recalling that the lower bound for the number of patterns required to achieve good network training is equal to the network order (Section 4.2), it can be seen that this training set has 117 times the theoretical minimum number of patterns. This high degree of redundancy is desirable, as it reduces the likelihood that uncorrelated noise and linearly related patterns will adversely affect the network training process.

7.4.2 Comparison Trajectory

To test the accuracy with which the CSLC network has learnt the PUMA 560s' coefficients of motion in a conscientious manner, one needs to employ a comparison trajectory which is quite different in nature to that used to train the network, so as to be able to gauge whether the trained network has achieved a problem-space-wide solution to the inverse dynamics. For this task it would seem appropriate to use a the most common type of movement requested of a manipulator, the simple go to/from trajectory, where the manipulator starts and finishes the trajectory at rest, and a control algorithm causes the links to smoothly move the end-effector from a starting position to the final position. However, due to the large degree of redundancy in a PUMA 560's workspace, in terms of the potential configurations capable of obtaining a given end-effector position, such trajectory's often cause only minimal movement in some joints. So instead, a similar trajectory is used, where all of the joints are simultaneously moved through a full 180° under the control of a PID algorithm with gravity compensation.

In detail, the comparison trajectory employed for both PUMA 560s consists of moving the joints from $[0 \quad -\frac{\pi}{2} \quad \frac{\pi}{2} \quad 0 \quad 0 \quad 0]$ to $[-\frac{\pi}{2} \quad 0 \quad 0 \quad \frac{\pi}{2} \quad \frac{\pi}{2} \quad \frac{\pi}{2}]$ rads under the influence of a PID controller with positional gains of $[4000 \quad 10000 \quad 2000 \quad 500 \quad 310 \quad 300]$, integral gains of $[5 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5]$ and velocity gains of $[80 \quad 114 \quad 25 \quad 25 \quad 12 \quad 17]$, and the abbreviated model gravity compensation from [Armstrong et al. 1986]. This trajectory takes approximately two and half seconds to complete, and calls for applied torques of up to $[13.6 \quad 42.8 \quad 8.59 \quad 1.44 \quad 1.77 \quad 1.76]$ Nm in magnitude. The control algorithm and its gains were not selected specially, they were simply the default setup for these manipulators at the time the tests were performed.

By disregarding data from the very beginning and end of the trajectory, any contributions from static friction effects can be removed. Furthermore, the manipulators do not experience backlash during the comparison trajectory. Therefore, all of the comparison data represent examples of the manipulators' dynamics that theoretically can be modelled by the CSLC network, thus providing a fair measure of the degree identification achieved.

7.5 NETWORK TRAINING

The details of the training process performed on the PUMA 560 CSLC network are discussed in this section. Fundamental to the principle of supervised learning is the definition of a network error cost for each pattern processed. This is the quantity that the application of training should reduce. The error cost must be derived from the network output errors (the differences between the desired target values and the network's output), which are applied torques values (measured in Newton metres). Thus, the error cost can not be expressed in terms of some combination of the induced accelerations, as the calculation of these values from the output errors would require the manipulator dynamics to be known. Using the current network estimate of the manipulator dynamics during training to approximate the acceleration error would introduce an undesirable nonlinear relationship into the identification of the optimal network solution. Therefore, as a quantitative measure of error is desired, which takes all six joints into account, the logical choice of error cost is some combination of the six output error values. Arguably, the output errors for the lower numbered joints should be given proportionally larger weightings, as positional errors in these joints have a disproportionately large effect upon the position of the end-effector (relative to the base frame). Conversely, it could be argued that the higher numbered joints should receive the larger weightings, as an output (torque) error of a given size will have a greater effect upon the motion of the lighter links within the manipulator. It is clear that a chosen application for the manipulator could have considerable bearing upon the most suitable method of weighting the output errors. However, in the absence of a chosen application, and for the sake of simplicity, the network error cost in this case is chosen to be half the root mean square of the output errors, denoted ϵ (as described in Eqn 3.4-2). Thus the forms of Backpropagation and PERAL learning described in Subsection 3.4.2 and Section 6.2 are appropriate without modification.

Before training can begin, an initial set of network weight values must be specified. Commonly, in other types of network, these values are generated randomly. This is because the optimal network solution may not be obtainable through learning from all possible initial network states, and multiple training attempts often necessary. This is not the case for the CSLC network. As has been discussed previously, training could potentially be abbreviated by using initial weight values derived from other studies of the PUMA 560, however, to demonstrate the general applicability of the CSLC network approach to coefficients identification this is not done. Furthermore, even in the absence of a priori knowledge, it is still possible to calculate the sign of many of the dynamics coefficients due to many of them representing combinations of masses and inertias, and/or containing squared terms. Although this feature is used to assess the results of training, the initial weight values for the CSLC network are chosen to be set to zero. In combination with a small learning rate, such that the high errors at the start of the

training process do not cause large changes in the weight values in any single epoch of learning, this is expected to cause only the significant network weights (in terms of reducing the network error cost) to evolve away from zero.

Two separate training sessions were performed, one for each manipulator, which it is now useful to differentiate between by denoting them as PUMA A and PUMA B. In both cases the Backpropagation learning algorithm was initially applied. Once the error cost had fallen below that obtained from processing the training trajectories with the coefficients identified by [Armstrong et al. 1986] and [Leahy & Saridis 1989], (2.63 Nm for PUMA A, 2.44 Nm for PUMA B), training was instead performed with a combination of PERAL and Backpropagation. This was done because it was observed that, unlike the examples in Section 6.3, acceptable error cost reduction rates were best maintained by performing bouts of training with alternating learning algorithms. Training was terminated when the maximum possible reduction between epochs of the network's error cost (by either method) fell below 0.001 Nm.

Patterns which may contain information on the motion of joints that were experiencing backlash were identified by Eqn. 2.5-7. These particular pattern and joint combinations are ignored for the purposes of training, so as not to cause the network to incorporate aspects of backlash behaviour in its identification of the (nonbacklash) inverse dynamics coefficients (see Subsections 2.5.3 and 7.2.4). The velocity thresholds (used to prevent the velocity subnetworks from being corrupted by the effects of static friction, see Subsection 7.2-2) were initially arbitrarily set to one hundredth of the maximum magnitude of each joint's training trajectory velocity. Once the network had learnt enough for the regions of significant static friction to be identifiable (see Fig. 7.6-4) the thresholds were updated to values obtained from observation of the velocities at the brink of these regions. These values are similar both manipulators, and are listed in the appendices in Section B.4.

The results of the described training are illustrated and discussed in the following section.

7.6 RESULTS

To examine the results of the CSLC network's coefficients identification, and in particular to check the problem-space-wide generalisation of the network solution, the comparison trajectory discussed in Subsection 7.4.2 is employed. Since the purpose of coefficient identification is to enable the accurate modelling of a manipulator's dynamics, the accuracy of a given identification technique can be assessed by comparing the predicted torque that the identified coefficients give rise to, in response to a set of real input data, against the corresponding real measured torque.

To make a comparison with identification through direct measurement, the model produced by utilising the mass/velocity and gravity coefficients reported by [Armstrong et al. 1986] and the friction coefficients reported by [Leahy & Saridis 1989] is also assessed. This set of coefficients will, from now on, be simply referred to by the designator “Ar+LS”. Although the PUMA 560s that these researchers performed their identification experiments upon can be assumed to have similar mass/velocity and gravity coefficients to those at the Robotics Institute, the friction coefficients could easily be very different, especially as a large proportion of the effective frictional forces are due to the drive mechanisms and are affected by factors such as overmeshing. Demonstrating the superiority of the CSLC network identified coefficients over the Ar+LS model would therefore not provide a convincing argument for the superiority of the CSLC identification technique. Instead, the decoupled nature of the CSLC friction model is exploited to produce a set of friction coefficients for each of the manipulators, which is optimal in terms of error cost, given the adoption of the Armstrong et al. coefficients. This is achieved by calculating the network output error for the two training sets when using the Armstrong et al. coefficient values and zero friction coefficient values, and then regressing this against the friction subnetwork inputs for each joint. That is, if the column vector of errors (target torques minus predicted torques) for the whole of a training set and a given joint i , is denoted e_i , then the optimal friction coefficients for that particular training set and the chosen mass/velocity and gravity coefficients is given by

$$[k_i^v \ k_i^d] = [\dot{q}_i \ \text{sgn}(\dot{q}_i)] \backslash e_i \quad (7.6-1)$$

where \dot{q}_i is the column vector of joint velocities over the whole training set, and the backslash denotes matrix division (as described for Eqn. 4.1-2). In addition, patterns likely to be experiencing backlash or with near zero velocities for joint i are disregarded in the same fashion as described for training the CSLC network. Note that this direct (non-iterative) solution for the friction coefficients could be incorporated into the CSLC network. This was not done in this study, so as not to obscure discussion and analysis of the CSLC network’s functionality. The dynamics model identified by the combination of the Armstrong et al. coefficients and regression shall be designated by “Ar+Rg”. Note that the friction coefficients obtained from Eqn. 7.6.1 differ between PUMA A and B.

The six graphs in Fig. 7.6-1 show the torques predicted by the inverse dynamics model when employing the coefficients produced by each of the three identification techniques, as well as the actual experimental profile, for the comparison trajectory of PUMA A. Fig. 7.6-2 similarly compares the identification techniques performance for PUMA B. Note that the comparison trajectory data has not been used in the adaption of either the CSLC network or the friction regression equation (Eqn. 7.6-1). Thus the results obtained from this comparison of the modelled and actual torques are assumed to be indicative of the identified coefficients’ ability to model the manipulator dynamics throughout the whole of the problem space.

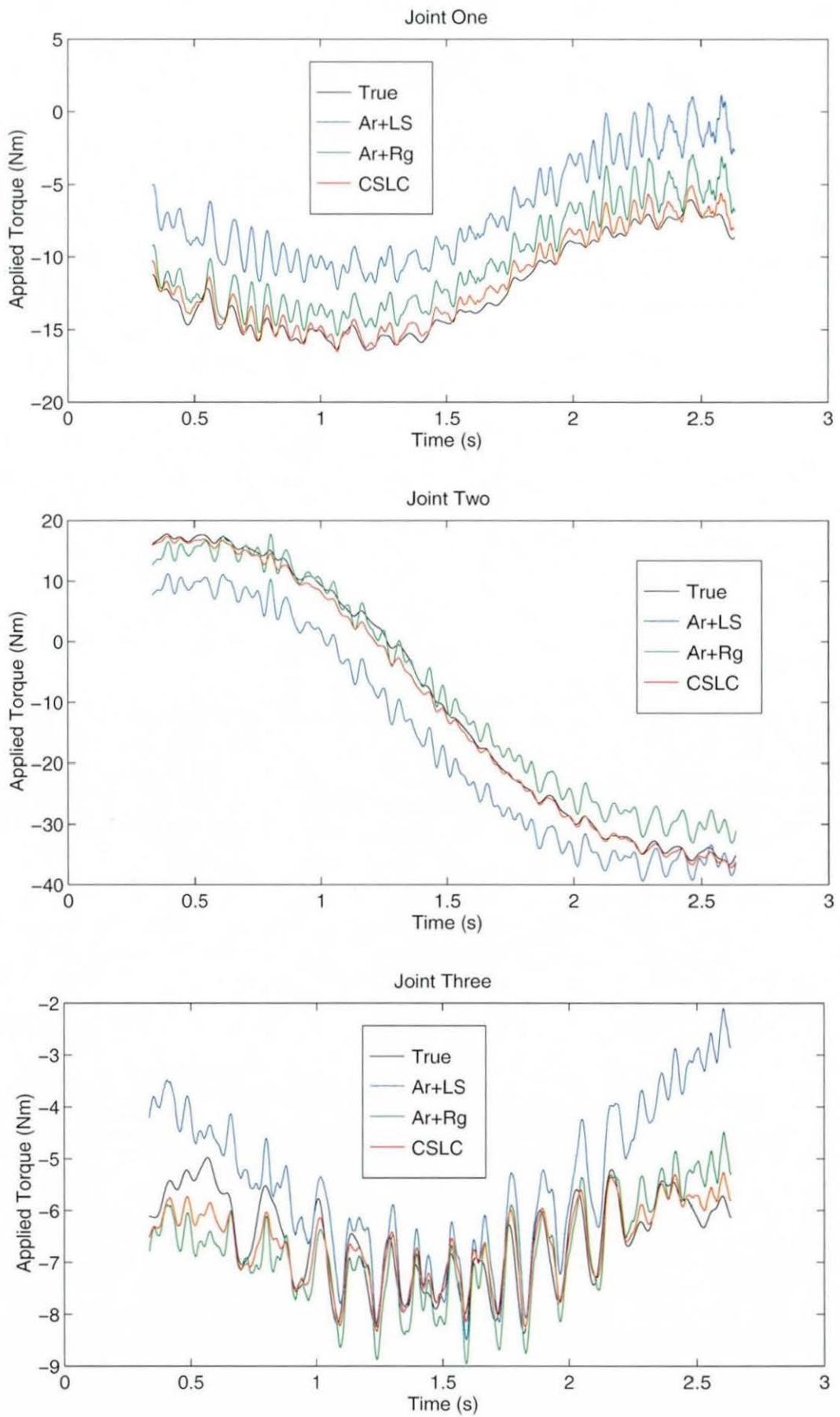


Figure 7.6-1a: Comparison of the predicted torques for joints 1 to 3 of PUMA A.

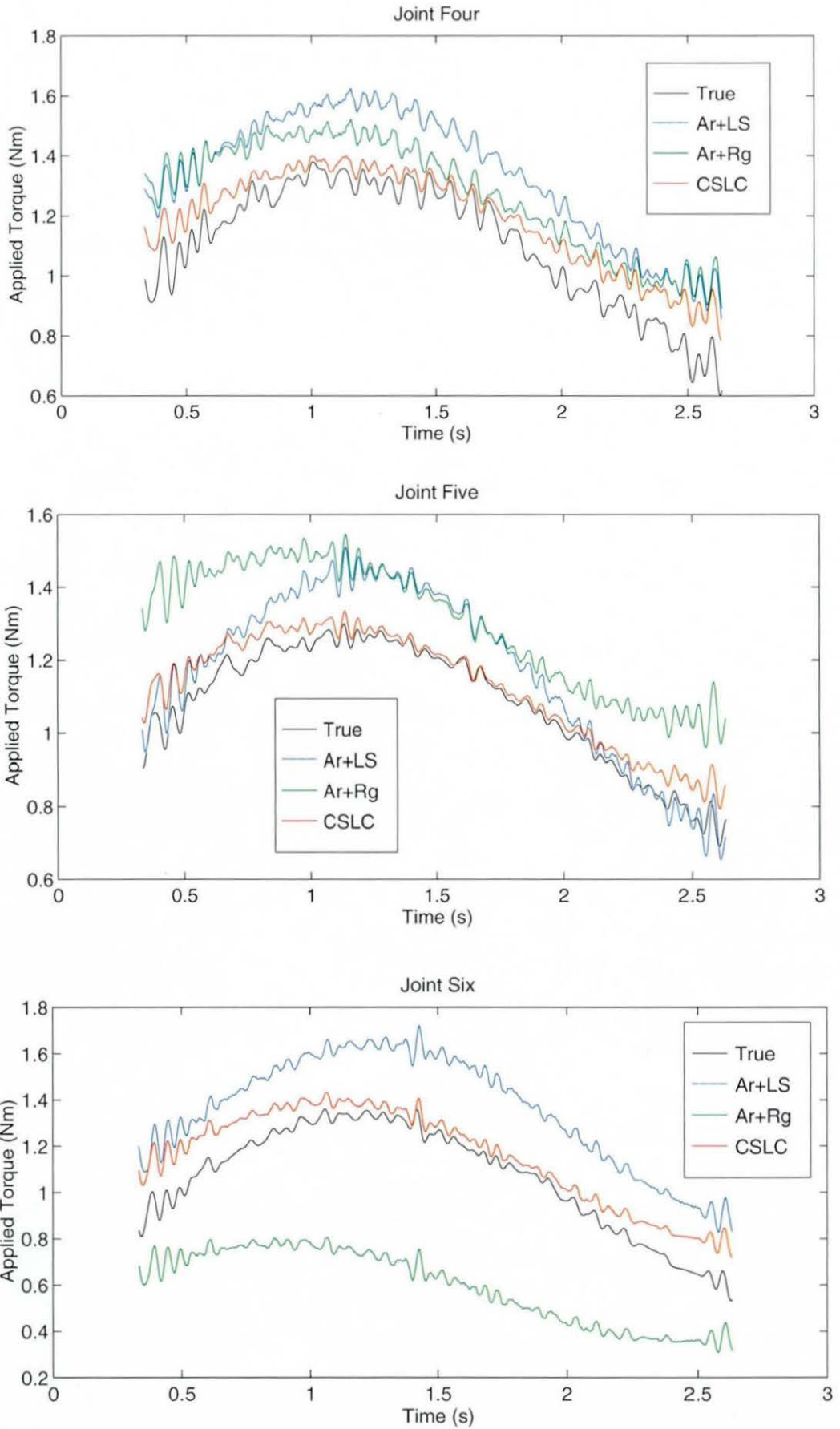


Figure 7.6-1b: Comparison of the predicted torques for joints 4 to 6 for PUMA A.

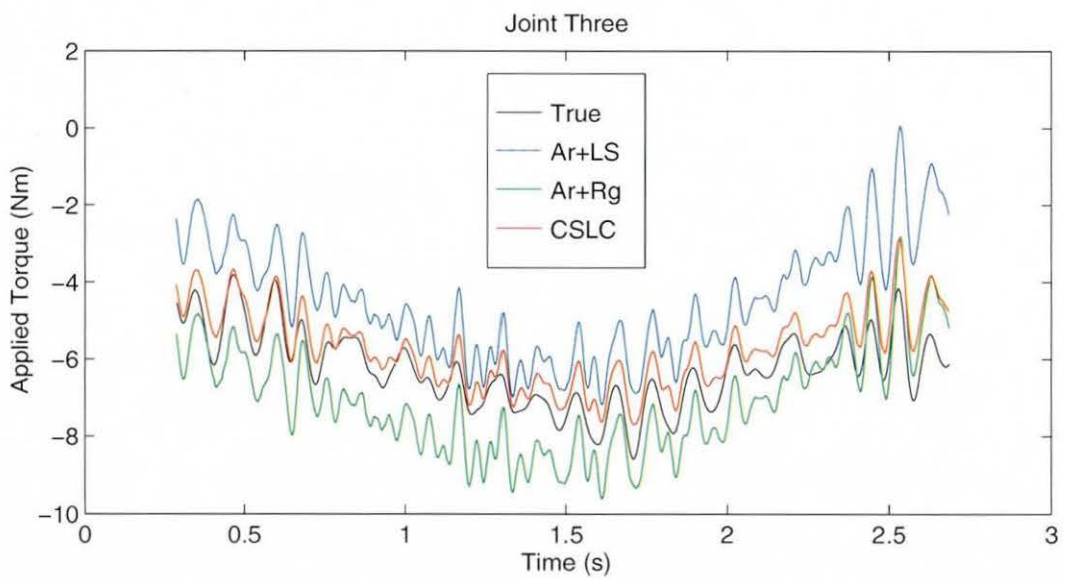
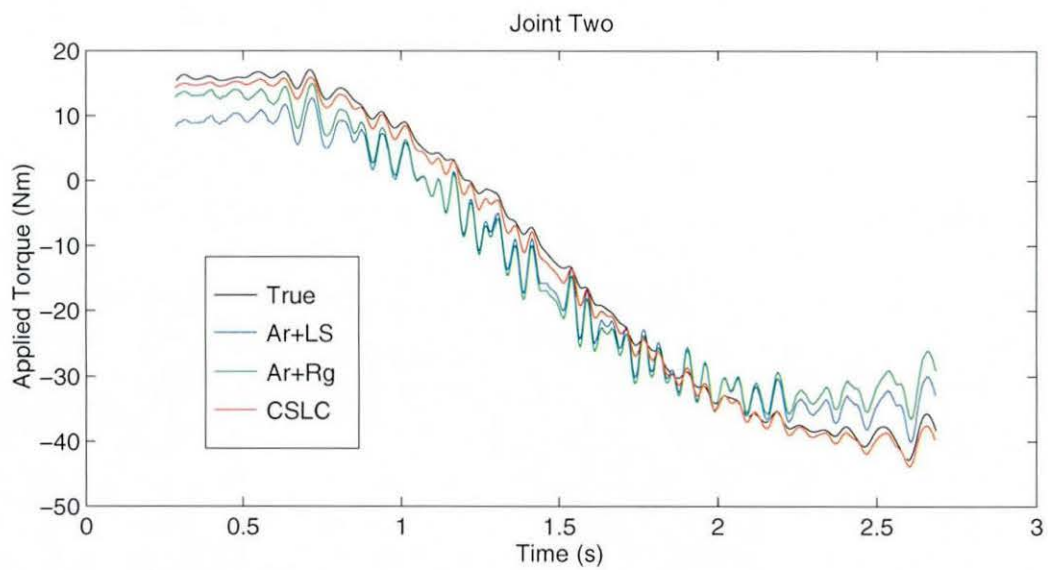
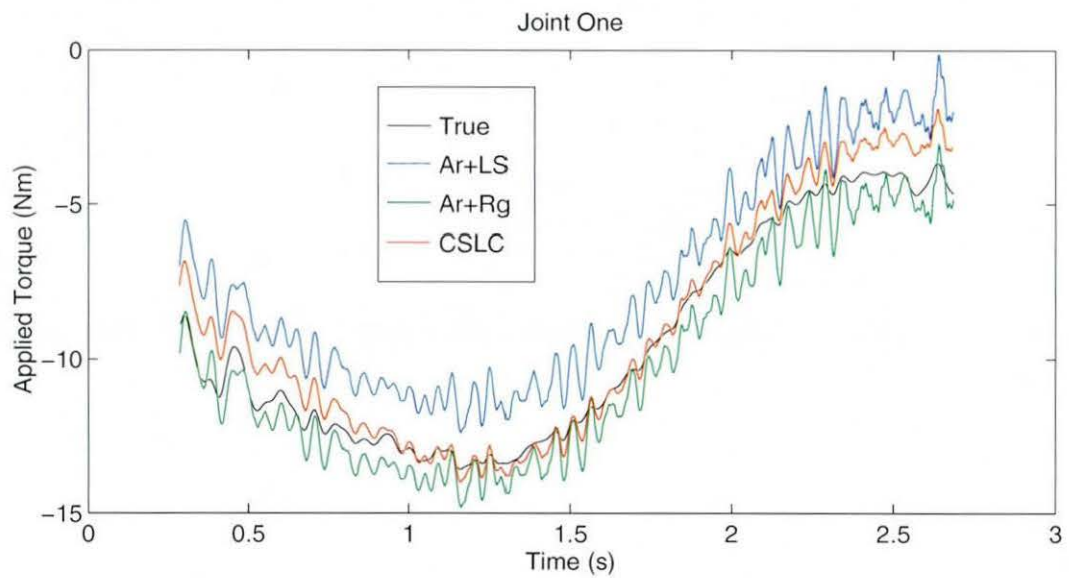


Figure 7.6-2a: Comparison of the predicted torques for joints 1 to 3 for PUMA B.

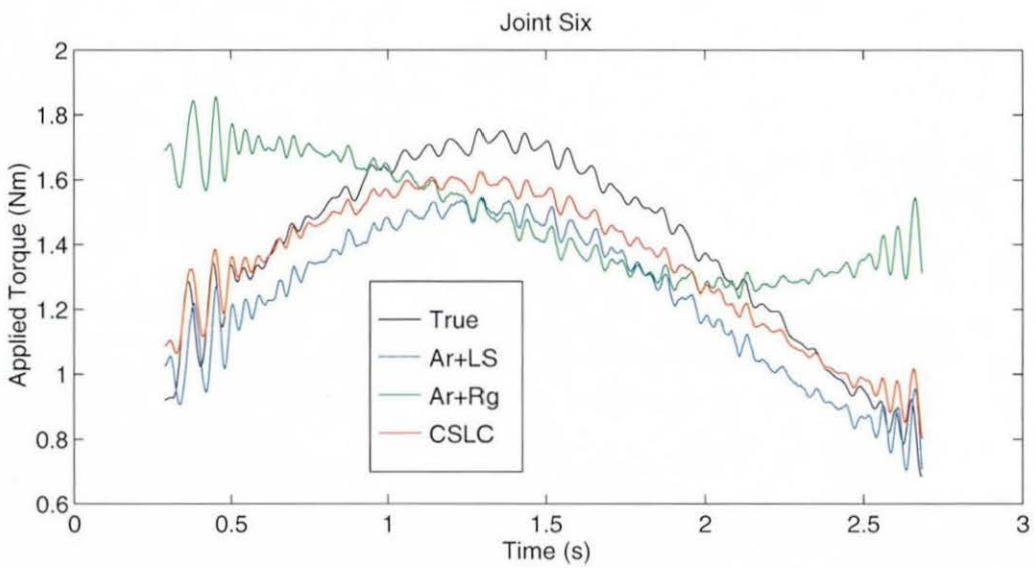
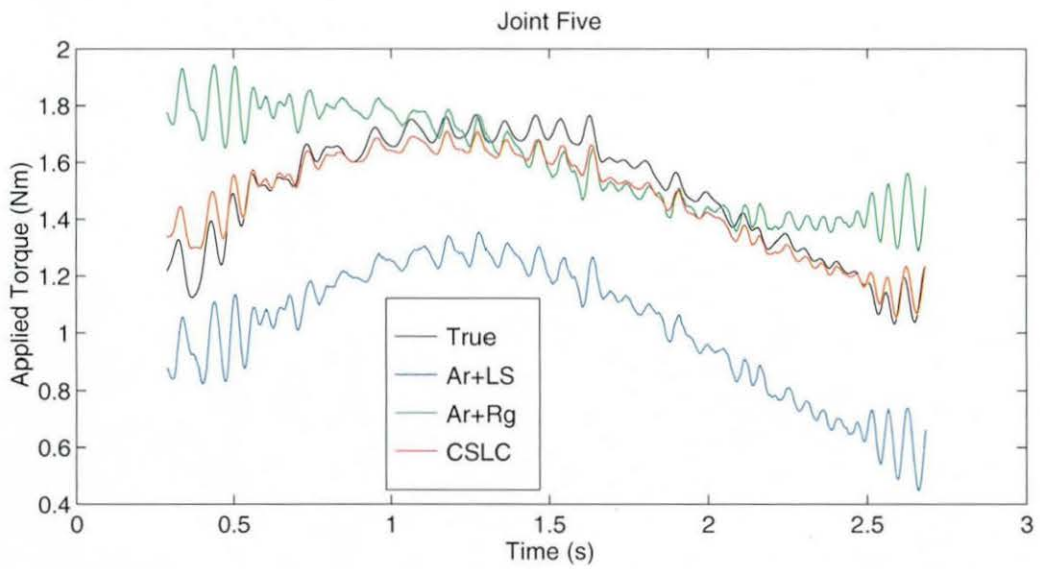
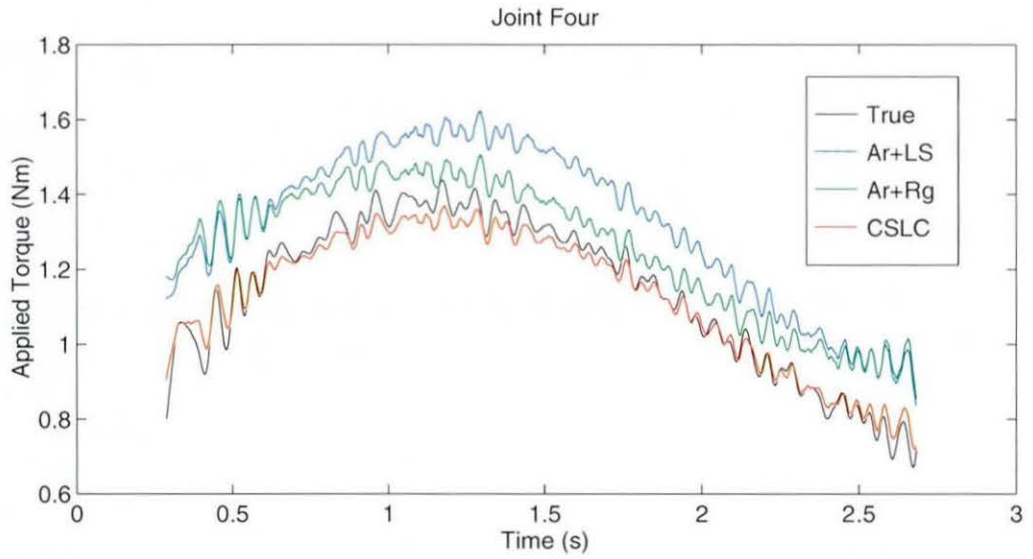


Figure 7.6-2b: Comparison of the predicted torques for joints 4 to 6 for PUMA B.

From consideration of Figs. 7.6-1 & 7.6-2 it can be seen that the overall modelling performance of the dynamics coefficients identified by the CSLC network is significantly superior to those identified by either of the direct measurement based techniques, for both PUMA 560s. These coefficient values are listed in Appendix B. The accuracy achieved by the different identification techniques in providing a model of the PUMA 560s' inverse dynamics is summarised in Figs. 7.6-3 & 7.6-4.

	Ar+LS	Ar+Rg	CSLC
mean($ e_1 $)	5.345	1.891	0.5408
mean($ e_2 $)	6.523	2.435	0.8877
mean($ e_3 $)	1.115	0.5177	0.2774
mean($ e_4 $)	0.2314	0.1665	0.08062
mean($ e_5 $)	0.09853	0.2154	0.04401
mean($ e_6 $)	0.2921	0.4733	0.09072
mean(ϵ)	1.8155	0.698	0.2498

Figure 7.6-3: Comparison trajectory mean error moduli and error cost for PUMA A. (all values given in Newton metres)

	Ar+LS	Ar+Rg	CSLC
mean($ e_1 $)	1.905	0.7166	0.6032
mean($ e_2 $)	4.458	4.476	1.093
mean($ e_3 $)	1.781	1.079	0.5394
mean($ e_4 $)	0.1831	0.1169	0.02830
mean($ e_5 $)	0.4743	0.1614	0.05301
mean($ e_6 $)	0.1629	0.2494	0.08132
mean(ϵ)	1.105	0.979	0.3085

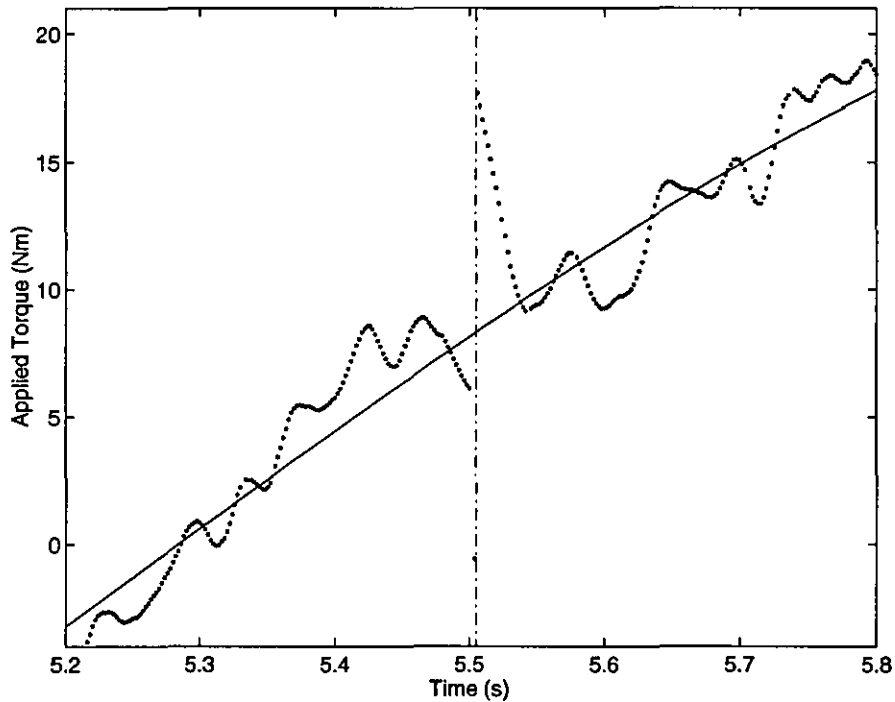
Figure 7.6-4: Comparison trajectory mean error moduli and error cost for PUMA B. (all values given in Newton metres)

As previously stated, there is little reason to suspect that mass/velocity or gravity coefficients would vary greatly between different manipulators of the same design. From consideration of the values presented in Figs. B.1-1, B.1-2, B.3-1 & B.3-2 it can be seen that the CSLC network identified broadly similar values for both PUMA A and B, for each mass/velocity and gravity coefficient. The fact that approximately the same solution was identified from two different sets of data implies that the training of the networks was not significantly affected by noise in the training data, and that the network solutions were not dominated by training set specific features that do not appear in the problem-space-wide dynamics. However, it is important to determine just how significant the differences in identified coefficient values are; to do this the sensitivity of the calculated torque to the value of each coefficient is determined. As the torque is linearly related to the dynamics coefficients, these sensitivities are constant for a given data set. Thus, the torque sensitivity for a given coefficient can be defined as the difference in the CSLC's output caused by a unit change in the coefficient's value, averaged across all joints and all patterns in the training set. This can be simply determined by setting the value of the particular coefficient in question equal to unity, all other coefficients to zero, and calculating the predicted torque's mean magnitude. These sensitivities are also listed in Figs. B.1-1, B.1-2, B.3-1 & B.3-2. By examining these values it can be seen that there does appear to be a correlation between the variation in identified coefficient values for the two manipulators and the coefficients' sensitivities, such that the largest differences occur for coefficients with some of the lowest sensitivities. It can be shown that all products of coefficient variations and sensitivities yield values of less than 0.001 Nm for the mass/velocity coefficients, and less than 0.3 Nm for the gravity coefficients.

As discussed in Subsection 5.2.3, a further check upon the validity of the coefficient values identified can be made by comparison with their algebraic representation. In the cases of k_4^M , k_6^M , k_9^M and k_{11}^M , their true values must all be positive, due to the nature of the physical parameters of which they are comprised. By consideration of Figs. B.1-1 and B.1-2, it can be seen that the CSLC network identified values of the correct sign for all four of these coefficients, for both PUMA 560s.

Also worthy of analysis are those periods when a given joint's velocity traverses zero. Fig. 7.6-4 shows a typical such region (part of the training trajectory for joint one of PUMA A), where error is the difference between the smooth target (demand) torque and the predicted torque. The steep ramps in the level of error are believed to be due to unmodelled static friction, as they occur either side of the line denoting when the joint's velocity passed through zero, according to the smoothed spline estimate given by Eqn. 7.6-4. As it seems reasonable to assume that these error ramps are indeed due to static friction, they can be used to indicate where the true joint velocity passed through zero. In the case shown in Fig. 7.6-4, and in fact all such cases identified (for both of the manipulators and all joints), the velocity zero-point estimated from the spline smoothing

of the recorded motion data lies within exactly the same measurement interval as that indicated by the characteristic static friction error ramps.



*Figure 7.6-4: Expanded view of joint one's torque profiles
from the training trajectory of PUMA A.*

*(Continuous line shows actual demand torque, dots are predicted torques,
the vertical line denotes the zero-point of the estimated joint velocity)*

7.7 SUMMARY

This chapter detailed the practical implementation of coefficients identification with a CSLC network, for two PUMA 560 manipulators. Having described the equipment involved, a review of the possible sources of system disturbance was conducted. These included the errors introduced into the algebraic representation of the manipulators' inverse dynamics by making the principal inertia substitution, the lack of a static friction model, operating above the drive motors' limits of linearity, backlash, compliance, quantisation and estimation of system inputs such as the joint velocities and accelerations, for which there are no direct measurements.

The related problems of quantisation and input estimation were tackled by developing a novel use of spline fitting, such that the quantised joint positions could be approximated by a smooth cubic spline curve which did not necessarily pass through any of the data

points. As the spline derived for any position profile is algebraically differentiable, good estimates of the joint velocities and accelerations are also achieved through the spline smoothing process. The potential limitations of spline smoothing were also discussed, in particular, the effect of the discontinuities within the manipulators' inverse dynamics were examined, and a means of dealing with them explained.

The issues involved in gathering data of the PUMA 560s' motion were then discussed, and a clear distinction made between the requirements of a network training set and a comparison set. A scheme for obtaining a training trajectory with a high degree of excitation was developed, where the data collected is as disparate as possible and no configuration of the manipulator's links is repeated. A comparison trajectory was designed which would represent the common start-here/go-there class of motion, as well as producing a data set unaffected by static friction or backlash.

The process of training the CSLC network on the gathered data was then detailed. An initially low training rate was used, in an attempt not to cause large changes in the weights matrices that might excite insignificant coefficient. This caused the training process to take several thousand epochs, but appeared to achieve its goal. The dynamics coefficients identified by training the CSLC network were then evaluated by comparing their ability to model the comparison trajectory to that of the coefficients identified through direct measurement. In order to allow for differences in the values of the friction coefficients that can easily occur between different manipulators, advantage was taken of the decoupled nature of the friction model, and a regression method for deriving optimal friction coefficients explained. This is done by finding the difference between the target torque and that predicted by a given set of mass/velocity and gravity coefficients, for multiple patterns, and then regressing this error matrix against the joint velocities, which produces a set of friction coefficients that cause the minimum network error for the data set used.

Illustrating the modelling performance of the variously identified coefficients both graphically and in tabular form, the overall superiority of those identified by the CSLC network was demonstrated. The successful modelling of a comparison trajectory of entirely different design to that of the training trajectory, and the agreement in the mass/velocity and gravity coefficient values identified for the two manipulators, imply that the inverse dynamics model obtained from these coefficients is generally accurate for the whole of the problem space. These results, therefore, affirm the concept that greater degrees of accuracy can be achieved through the processing of motion data by an adaptive network than through direct measurement of a manipulator's physical components.

REFERENCES AND FURTHER READING

- Armstrong, B., Khatib, O., and Burdick, K., "The explicit dynamic model and inertial parameters of the PUMA 560 arm," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 510-518, 1986.
- Armstrong, B., "On Finding Exciting Trajectories for Identification Experiments Involving Systems with Nonlinear Dynamics," *Int. Journal of Robotics Research*, vol. 8, no. 6, pp. 28-48, Dec. 1989.
- Craig, J.J., "Adaptive control of mechanical manipulators," Ph.D Thesis, Elec. Eng. Dept., Stanford University, 1986.
- Leahy, M.B., Jr., and Saridis, G.N., "Compensation of industrial manipulator dynamics," *Int. J. Robotics Res.*, vol.8, no.4, pp. 73-84, 1989.
- Robotics Institute
Smith Hall 113, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3890, U.S.A. Tel.: (412) 268 3818, Fax: (412) 268 5571
- Sastry, S.S., "Model-reference adaptive control — stability, parameter convergence and robustness," *IMA Journal of Math. Control Information*, no. 1, pp. 27-66, 1984.
- TRC004 User's Manual, "Richard Voyles PUMA Page," Computer Sciences, Carnegie Mellon University. <http://www.cs.cmu.edu/~deadslug/puma.html>
- Trident Robotics and Research, Inc.
2516 Matterhorn Drive, Wexford, PA 15090-7962, U.S.A. Tel.: (412) 934 8348

8. CONCLUSION

The contributions made by this thesis to the fields of robotics and artificial intelligence are discussed in this chapter. The novel aspects of the research carried out in order to provide a method of accurate dynamics coefficients identification are listed, and their significance evaluated. The promising avenues of further investigation are also discussed.

8.1 ACHIEVEMENTS

The advancements in the state of robotics and artificial intelligence theory contributed by this work can be summarised as follows

- A technique for accurately identifying the dynamics coefficients of any given robotic manipulator from motion data alone.
- The identification of reasons why conventional adaptive networks do not achieve problem-space-wide solutions to high order analogue systems.
- The development of the novel context sensitive linear combiner network architecture.
- The development of the novel proportional error allocation network learning algorithm
- The derivation of network order as a means for comparing network complexity and obtaining relative measures of the size of training sets required.

Each of these items is discussed in detail below.

8.1.1 Dynamics Coefficients Identification

The dynamics of the case study PUMA 560 manipulator were identified from motion data alone, without the need for direct measurement of the manipulator's physical parameters. The results of Chapter 7 demonstrated the superior dynamics modelling performance

obtained from these coefficients compared to that obtained from coefficients derived from direct measurement. The technique, as described in this thesis, is generally applicable to all robotic manipulators whose motion can be described by the rigid bodied inverse dynamics relationship known as the Euler-Lagrange equation (Eqn. 2.5-2). A technical summary of the identification methodology is provided in Appendix C for those who may wish to attempt similar coefficient identification studies.

8.1.2 Unsuitability of Conventional Network Architectures

The analysis of the learning process in networks which model analogue systems, via the use of the curve-fitting analogy, led to the identification of the reasons for conventional network architectures yielding either poor modelling performance or non-problem-space-wide solutions. The former occurs in networks with less flexibility in their algebraic forms than that possessed by the system they are attempting to model, whilst the latter occurs in networks with more flexibility than the modelled system when trained with a data set that is not representative of the whole problem space. Given the impossibility, for many high order systems, of collecting enough disparate training patterns to achieve such problem-space-wide representation, the unsuitability of network architectures that do not exactly match the algebraic form of the modelled system is made apparent.

The inverse dynamics of many robotic manipulators constitute such high order systems, given that the number of input variables for a n degree of freedom manipulator is, in general $3n$. This explains why studies on coefficient identification have previously only reported success for manipulators of 2 or 3 degrees of freedom.

8.1.3 The Context Sensitive Linear Combiner Network

The CSLC network was developed in order to provide a generally applicable method for representing the algebraic form of systems such as the Euler-Lagrange equation as an adaptive network exactly. This exact match in algebraic forms enables the training process to achieve an accurate problem-space-wide solution from a non-problem-space-wide representative set of training data. Furthermore, the output of a CSLC network is a linear function of its adaptive weights, therefore ensuring that there is just one minima in the network's error hypersurface, which can be reliably identified by error gradient descent learning algorithms.

8.1.4 Proportional Error Allocation

In order to avoid tediously lengthy training sessions, as can be caused by the inevitable

decline in learning rate achieved by purely gradient descent learning algorithms as they approach the network optimal solution, the PERAL fine-tuning algorithm was developed. This algorithm does not rely upon the error gradient, and has been shown to be capable of significantly reducing training times, once an approximate network solution has been identified.

8.1.5 Network Order

The definition of network order has provided a lower bound on the number of patterns required to prevent a given network from being underdetermined by its training set. Thus, networks trained with fewer patterns than the value of their order, are, in the presence of noise, guaranteed to produce network solutions local to that training set.

Given the effects of noise and the difficulty in determining the independence of training patterns, the ratio of the number of training patterns to the network order can be used as a measure of the likelihood that the optimal solution obtained from the training set is unique, and thus potentially equal to the desired problem-space-wide solution.

8.2 FURTHER WORK

As with any productive study, this work points to many more potential avenues of research. The most promising and significant of these are discussed below.

8.2.1 Improved CSLC Network Modelling Accuracy

The accuracy of a dynamics model/controller which employs the dynamics coefficients identified by the CSLC network could be improved by improving the network's representation of the inverse dynamics. As stated in Subsection 7.2.1, the algebraic representation of a given manipulator's dynamics would be more accurate if it did not employ the principal inertia substitution (Eqn. 2.3-20). Also, minor simplifications could be made to the lists of dynamics coefficients by employing numeric values for link lengths and offsets (a & d). These are the easiest physical parameters to measure directly, and the measurement error margins can be expected to be low. Some coefficients would then not need to be identified separately, for instance, for the PUMA 560, the unknown mass coefficients k_1^M , k_2^M and k_3^M could be replaced by three known multiples of a single unknown coefficient.

Another known inaccuracy of the CSLC representation described in Chapter 5 is the lack

of a static friction term. As stated in Subsection 7.2.2, a model of the static friction was not incorporated into the CSLC network due to the expected difficulty in correctly identifying the sign of \dot{q} at low joint speeds. If one considers Fig. 7.6-4, it can be seen that the use of spline smoothing in the calculation of the estimated joint velocities has in fact provided an excellent identification of the direction of joint motion at low speeds. Thus, a static friction subnetwork could be incorporated into the CSLC network, such as that shown in Fig. 8.2-1 for a single joint, which would calculate the static friction term, denoted τ_i^s , as described in Eqn. 2.5-4.

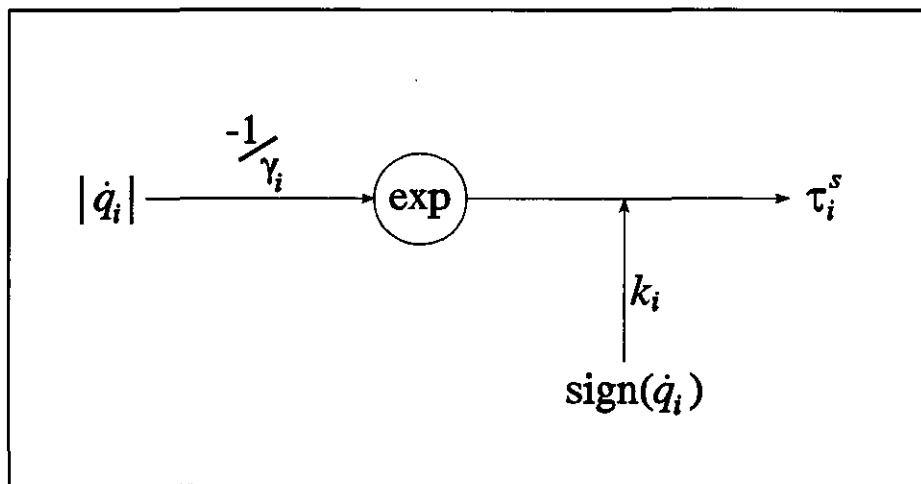


Figure 8.2-1: A static friction CSLC subnetwork for a given single joint

The static friction model shown in Fig. 8.2-1 uses the joint velocity magnitude and sign as two separate inputs, and makes use of context sensitivity to combine these two sources of information. However, it also requires a nonlinear activation function, in particular, the exponential function. The inclusion of a nonlinear function would mean forgoing the CSLC network's attribute of having its outputs strictly linearly related to its adaptive coefficients, along with the advantageous properties this bestows. In order to keep this linearity during training, the static friction slope coefficients (γ) could be kept constant, and only the static friction magnitude coefficients learnt. If desired, both sections of the static friction subnetwork could then be trained in isolation after all the other subnetworks have been fully trained.

Many friction models exist in the literature, of which the model developed in Subsection 2.5.1 is one of the most common, however, there is evidence to suggest that the friction forces acting upon the joints of a PUMA 560 have differing magnitudes depending upon the direction of motion [Corke & Armstrong-Hélouvy 1994]. This implies that the CSLC friction subnetworks could achieve greater accuracy by possessing two sets of inputs for each joint, where each input is set to zero for motion in a particular direction. For instance, the viscous friction model for a given joint could be as shown in Fig. 8.2-2.

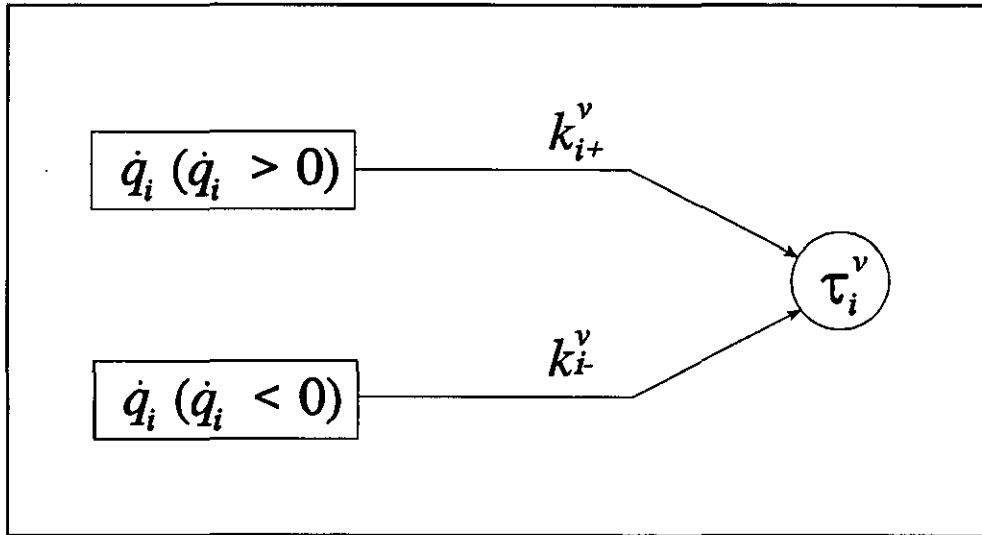


Figure 8.2-2: Asymmetric viscous friction subnetwork for a given joint

8.2.2 Proportional Error Allocation

The PERAL learning algorithm has been demonstrated in Chapter 6 to be of potential benefit in the hastening of network training. However, further work is required to identify the classes of network to which it can be successfully applied, and comparisons made with a larger number of alternative learning algorithms. It may also be possible to derive some general indicator of how soon in a network's training the switch to PERAL learning would be of benefit.

8.2.3 Error Hypersurfaces and Analytical Solutions

It is clear from the review of adaptive network theory carried out in Chapter 3 that a better understanding of the manner in which a network's error hypersurface affects training would help advance the state of the field. Furthermore, as discussed in Section 4.1 and Subsection 3.3.2, greater use of linear algebra techniques is expected to yield significant advances in the speed, accuracy, and understanding of the training process. In particular, given the linear relationship of a CSLC network's output to its coefficients, there may well exist a method for obtaining a direct (non-iterative) optimal solution for such networks, however, the derivation of such a method is not trivial due to the nonlinear coupling between pattern dependent inputs. The technique presently considered to be the most promising in this respect involves solving the error hypersurface equation for the point where its derivative is equal to zero, which for a CSLC network would uniquely identify the optimal network state for a given training set.

Though clearly not always a simple matter, in the opinion of the author, many aspects of adaptive network theory would benefit from a more analytical based understanding being developed, to replace the present, often empirical based, perceptions.

8.2.4 Payload Identification

Given the success in identifying the dynamics coefficients for a whole manipulator, it is envisaged that a logical extension of the identification technique could allow for the dynamics effects of a payload of initially unknown characteristics to be compensated for during its transport. The payload could be modelled as an additional link of the manipulator, and its characteristics learnt on-line as the manipulator picks up and moves the payload. Assuming the unladen manipulator's dynamics had been accurately identified previously, only the extra dynamics components due to the payload need to be learnt on-line. Thus, assuming rapid learning, the payload could be placed with accuracy at the end of its transportation.

8.2.5 The CSLC Network

Although the CSLC network was developed in this thesis in order to solve the problem of manipulator dynamics coefficients identification, it is suitable for many different applications. Given the major benefits a CSLC network provides over conventional network architectures, as discussed in Section 5.2 but in particular those of generalisation and sole error hypersurface minima, there are expected to be many systems and problems to which the application of a CSLC network would provide a significant positive contribution. An overview of the identification methodology for non-manipulator systems is given in Appendix C.

There are expected to be still further problems for which a strictly linear combiner based network may not be appropriate, due to the modelled system not being linearisable in its coefficients, but for which the concept of designing a network with exactly matching algebraic form will be of significant benefit.

REFERENCES

Corke, P.I., and Armstrong-Hélouvy, B., "A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot," *IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1608-1613, 1994.

APPENDIX A

TABLE OF UNIQUE TERMS

The algebraic expressions which have a unique meaning within this thesis are listed below. A concise definition of each term is provide, as well as a reference to the section or subsection in which that term was first introduced and the full definition stated.

<u>Expression</u>	<u>Definition</u>	<u>Stated</u>
α	Link twist vector	2.6.1
ϵ	Error cost: half root mean square of output errors	3.4-2
η	Learning rate	3.4.1
θ	Joint angle vector	2.6.1
σ	Network order	5.2
τ	Generalised applied force/torque vector	2.3
a	Link length vector	2.6.1
b	Degree of backlash vector	2.5.3
C	Velocity Matrix	6.1.2
c_{ijk}	Christoffel symbol (of the first kind)	2.4
d	Link offset vector	2.6.1
e_i	Output error vector for joint i	7.6-1
F	Friction term vector	2.5.1
$f_A(\cdot)$	Activation function for nodes in layer A	3.4.2
G	Gravity term vector	2.4
g	Local downward acceleration due to gravity	2.3.3
g_{vec}	Acceleration due to gravity vector (with respect to base frame)	2.3.3
I_i	Pseudo-inertia matrix for link i	2.3.2
${}^m I$	Effective motor inertia vector (acting about joint axis)	2.5.2
${}^i I_{xx}$	Mass moment of inertia of link i about the x-axis of frame i	2.3.2
${}^p I_{xx}$	Principal mass moment of inertia of link i at its c.o.g.	2.3.2
${}^i I_{xy}$	Mass cross-product of inertia of link i between axes of frame i	2.3.2

K	Kinetic energy vector	2.3
k^d	Dry friction coefficients vector	2.5.1
k^G	Gravity term coefficients vector	6.1.3
k^M	Mass term coefficients vector	6.1.1
k^s	Static friction magnitude coefficient vector	2.5.1
k^v	Viscous friction coefficients vector	2.5.1
L	Lagrangian vector ($K - P$)	2.3
M	Mass matrix of entire manipulator	2.3.2
m_i	Mass of link i	2.3.3
o	Number of output nodes	3.4.1
P	Potential energy vector	2.3
p	Number of patterns in training set	4.1
q	Generalised joint position vector	2.3.2
\dot{q}	Generalised joint velocity vector	2.3
\ddot{q}	Generalised joint acceleration vector	2.3
${}^i r$	A given point expressed in coordinate frame i	2.3.1
${}^i r_c$	Coordinate vector in frame i of the centre of gravity of link i	2.3.3
${}^i r_x$	X-coordinate in frame i of the centre of gravity of link i	2.3.2
s	Sum of weighted inputs to the node in question	3.3.1
S^A	Summed weighted inputs vector for layer A	3.4.2
t	Network target vector	3.4.1
T_i	Homogenous transform of frame i relative to the base frame	2.3.1
${}^j T_i$	Homogenous transform of frame i relative to frame j	2.3.1
V	Velocity term vector	2.4
W	Weights matrix	3.3.2
W^G	Gravity term weights matrix for CSLC network	6.1.3
W^M	Mass term trainable weights matrix for CSLC network	6.1.1
W^V	Velocity term trainable weights matrix for CSLC network	6.1.2

APPENDIX B

TERMS AND VALUES FOR THE PUMA 560 CSLC NETWORK

This appendix details all of the components of the PUMA 560 CSLC network, including the identified values for the dynamics coefficients. Each of the subnetworks is described in turn. Note that, in every case, multiplying a subnetwork input vector by the appropriate weights matrix produces exactly the same algebraic expressions for the mass matrix, Christoffel symbols and gravity effect vector as described in [Armstrong et al. 1986]. Also, the algebraic form used for describing the inputs to each subnetwork holds in both the single pattern (algebraic or sequential processing) case and the multi-pattern (parallel computation) case. For the single input pattern case, a subnetwork's input is a row vector, whilst for the parallel computation case the algebraic input vector is evaluated for each pattern, producing an input matrix with one row per pattern.

In tables throughout this appendix, "Armstrong et al." refers to [Armstrong et al. 1986], and "Leahy & Saridis" refers to [Leahy & Saridis 1989].

B.1 MASS TERM SUBNETWORK

The mass subnetwork possesses 59 preprocessed inputs, which are described in Subsection B.1.1. The weights matrix connecting the inputs to the 21 nodes representing the unique mass matrix elements possesses 114 finite elements and 1125 null elements (these symbolise non-existent links, for the purposes of matrix algebra however, they can be represented by zero valued weights which are not trained).

B.1.1 Mass Subnetwork Inputs

Denoting the row vector of preprocessed mass subnetwork inputs as X^M , its elements are defined as follows.

$$\begin{array}{lll}
 X_1^M = 1 & X_2^M = c_5 & X_3^M = c_{23} s_{23} c_4 c_5 s_5 \\
 X_4^M = c_{23} s_{23} c_5 & X_5^M = c_{23}^2 c_5 & X_6^M = c_2 s_{23} c_5 \\
 X_7^M = c_4^2 c_5^2 & X_8^M = c_{23}^2 c_4^2 c_5^2 & X_9^M = c_{23}^2 c_5^2 \\
 X_{10}^M = c_{23} s_{23} & X_{11}^M = c_{23} s_{23} c_4 s_5 & X_{12}^M = c_2 c_{23} c_4 s_5 \\
 X_{13}^M = c_2 c_{23} & X_{14}^M = c_{23}^2 & X_{15}^M = c_{23}^2 c_4^2 \\
 X_{16}^M = c_{23}^2 c_4 s_5 & X_{17}^M = c_4^2 & X_{18}^M = c_2 s_{23} \\
 X_{19}^M = s_4 s_5 & X_{20}^M = c_2^2 & X_{21}^M = c_2 s_2 \\
 X_{22}^M = c_{23} s_4 c_5 s_5 & X_{23}^M = c_{23} s_4 s_5 & X_{24}^M = s_2 s_4 s_5 \\
 X_{25}^M = s_{23} s_4 s_5 & X_{26}^M = s_{23} c_4 s_5 & X_{27}^M = s_{23} c_4 s_4 \\
 X_{28}^M = s_{23} c_4 s_4 c_5^2 & X_{29}^M = c_{23} & X_{30}^M = c_{23} c_5 \\
 X_{31}^M = s_2 & X_{32}^M = s_{23} & X_{33}^M = c_2 \\
 X_{34}^M = s_{23} c_4 c_5 s_5 & X_{35}^M = c_2 c_4 s_5 & X_{36}^M = c_{23} c_4 s_5 \\
 X_{37}^M = c_{23} c_5^2 & X_{38}^M = c_2 s_4 c_5 & X_{39}^M = s_{23} s_4 c_5 \\
 X_{40}^M = c_{23} s_4 c_5 & X_{41}^M = c_{23} c_4 c_5 & X_{42}^M = s_{23} s_4 \\
 X_{43}^M = s_{23} s_5 & X_{44}^M = c_5^2 & X_{45}^M = s_2 c_{23} c_5 \\
 X_{46}^M = s_2 s_{23} & X_{47}^M = s_2 s_{23} c_4 s_5 & X_{48}^M = s_2 c_{23} \\
 X_{49}^M = c_4 s_5 & X_{50}^M = c_2 s_{23} s_4 s_5 & X_{51}^M = s_4 c_5 s_5 \\
 X_{52}^M = s_2 c_{23} s_4 s_5 & X_{53}^M = c_4 & X_{54}^M = c_4 c_5 \\
 X_{55}^M = s_2 c_{23} c_4 c_5 & X_{56}^M = c_2 s_{23} c_4 c_5 & X_{57}^M = s_2 s_{23} s_5 \\
 X_{58}^M = c_2 c_{23} s_5 & X_{59}^M = s_5 &
 \end{array}$$

where

$$\begin{array}{l}
 c_i = \cos(q_i) \\
 s_i = \sin(q_i) \\
 c_{ij} = \cos(q_i + q_j) \\
 s_{ij} = \sin(q_i + q_j)
 \end{array}$$

B.1.2 Mass Subnetwork Weights Matrix

The 21 unique elements of the mass matrix can be found from $X^M W^M$, where the weights matrix elements are simply integer multiples of one of the 26 mass/velocity coefficients, as described below.

$$\begin{array}{llll}
 W_{1,1}^M = k_{26}^M & W_{1,7}^M = k_{25}^M & W_{1,8}^M = k_{22}^M & W_{1,12}^M = k_{23}^M \\
 W_{1,16}^M = k_{16}^M & W_{1,19}^M = k_{11}^M & W_{1,21}^M = k_6^M & W_{2,1}^M = -2k_1^M \\
 W_{2,7}^M = -2k_1^M & W_{2,8}^M = -2k_1^M & W_{2,12}^M = -2k_1^M & W_{2,18}^M = k_4^M \\
 W_{3,1}^M = 2k_{15}^M & W_{4,1}^M = 2k_2^M & W_{5,1}^M = 2k_1^M & W_{6,1}^M = 2k_3^M \\
 W_{6,7}^M = 2k_3^M & W_{6,8}^M = k_3^M & W_{7,1}^M = k_{15}^M & W_{7,7}^M = -k_{15}^M \\
 W_{7,8}^M = -k_{15}^M & W_{7,12}^M = -k_{15}^M & W_{8,1}^M = -k_{15}^M & W_{9,1}^M = -k_{15}^M \\
 W_{10,1}^M = k_{12}^M & W_{11,1}^M = -2k_1^M & W_{12,1}^M = 2k_3^M & W_{12,7}^M = 2k_3^M \\
 W_{12,8}^M = k_3^M & W_{13,1}^M = 2k_{10}^M & W_{13,7}^M = 2k_{10}^M & W_{13,8}^M = k_{10}^M \\
 W_{14,1}^M = k_{24}^M & W_{15,1}^M = k_{18}^M & W_{16,1}^M = 2k_2^M & W_{17,1}^M = -k_{18}^M \\
 W_{17,7}^M = k_{18}^M & W_{17,8}^M = k_{18}^M & W_{17,12}^M = k_{18}^M & W_{18,1}^M = -2k_{14}^M \\
 W_{18,7}^M = -2k_{14}^M & W_{18,8}^M = -k_{14}^M & W_{19,1}^M = 2k_7^M & W_{19,9}^M = k_1^M \\
 W_{19,11}^M = k_4^M & W_{19,13}^M = k_1^M & W_{19,15}^M = k_4^M & W_{20,1}^M = k_{19}^M \\
 W_{21,1}^M = k_5^M & W_{22,2}^M = -k_{15}^M & W_{22,3}^M = -k_{15}^M & W_{23,2}^M = k_1^M \\
 W_{23,3}^M = k_1^M & W_{23,4}^M = k_7^M & W_{24,2}^M = k_3^M & W_{25,2}^M = k_2^M \\
 W_{25,3}^M = k_2^M & W_{26,2}^M = k_7^M & W_{26,3}^M = k_7^M & W_{26,4}^M = -k_1^M \\
 W_{26,6}^M = -k_4^M & W_{27,2}^M = k_{18}^M & W_{27,3}^M = k_{18}^M & W_{28,2}^M = -k_{15}^M \\
 W_{28,3}^M = -k_{15}^M & W_{29,2}^M = k_{20}^M & W_{29,3}^M = k_{20}^M & W_{29,4}^M = k_{13}^M \\
 W_{30,2}^M = -k_7^M & W_{30,3}^M = -k_7^M & W_{30,6}^M = k_4^M & W_{31,2}^M = k_{21}^M \\
 W_{32,2}^M = k_{17}^M & W_{32,3}^M = k_{17}^M & W_{33,2}^M = k_8^M & W_{34,4}^M = k_{15}^M \\
 W_{35,4}^M = k_3^M & W_{36,4}^M = k_2^M & W_{37,4}^M = -k_{15}^M & W_{38,5}^M = k_3^M \\
 W_{39,5}^M = -k_1^M & W_{40,5}^M = k_2^M & W_{41,5}^M = -k_7^M & W_{42,5}^M = k_9^M \\
 W_{43,5}^M = k_7^M & W_{44,7}^M = k_{15}^M & W_{44,8}^M = k_{15}^M & W_{44,12}^M = k_{15}^M \\
 W_{44,16}^M = -k_{15}^M & W_{45,7}^M = -2k_3^M & W_{45,8}^M = -k_3^M & W_{46,7}^M = 2k_{10}^M \\
 W_{46,8}^M = k_{10}^M & W_{47,7}^M = 2k_3^M & W_{47,8}^M = k_3^M & W_{48,7}^M = 2k_{14}^M
 \end{array}$$

$$\begin{array}{llll}
W_{48,8}^M = k_{14}^M & W_{49,7}^M = 2k_2^M & W_{49,8}^M = 2k_2^M & W_{49,12}^M = 2k_2^M \\
W_{50,9}^M = -k_3^M & W_{51,9}^M = -k_{15}^M & W_{51,13}^M = -k_{15}^M & W_{52,9}^M = k_3^M \\
W_{53,10}^M = k_9^M & W_{53,14}^M = k_9^M & W_{54,10}^M = -k_1^M & W_{54,14}^M = -k_1^M \\
W_{55,10}^M = -k_3^M & W_{56,10}^M = k_3^M & W_{57,10}^M = k_3^M & W_{58,10}^M = k_3^M \\
W_{59,10}^M = k_2^M & W_{59,14}^M = k_2^M & &
\end{array}$$

(all other elements are zero/null)

B.1.3 Mass/Velocity Coefficients

The vector of mass/velocity coefficients is composed so as produce a list of expressions with a monotonically increasing number of terms. Its elements are defined as follows.

$$k_1^M = -d_4 {}^6r_z m_6$$

$$k_2^M = a_3 {}^6r_z m_6$$

$$k_3^M = a_2 {}^6r_z m_6$$

$$k_4^M = {}^p I_{zz}$$

$$k_5^M = -2 {}^2r_x {}^2r_y m_2$$

$$k_6^M = m I_6 + {}^p I_{zz}$$

$$k_7^M = d_2 {}^6r_z m_6 + d_3 {}^6r_z m_6$$

$$k_8^M = d_2 {}^2r_y m_2 + {}^2r_y {}^2r_z m_2$$

$$k_9^M = {}^6r_z^2 m_6 + {}^p I_{zz} + {}^p I_{xx}$$

$$k_{10}^M = a_2 a_3 m_4 + a_2 a_3 m_5 + a_2 a_3 m_6$$

$$k_{11}^M = m I_5 + {}^6r_z^2 m_6 + {}^p I_{zz} + {}^p I_{xx}$$

$$k_{12}^M = 2 a_3 d_4 m_4 + 2 a_3 d_4 m_5 + 2 a_3 d_4 m_6 + 2 a_3 {}^4r_z m_4$$

$$k_{13}^M = {}^p I_{zz} + {}^p I_{xx} + {}^6r_z^2 m_6 + {}^p I_{xx}$$

$$k_{14}^M = -a_2 d_4 m_4 - a_2 d_4 m_5 - a_2 d_4 m_6 + a_2 {}^3r_y m_3 - a_2 {}^4r_z m_4$$

$$k_{15}^M = -{}^p I_{yy} - {}^p I_{zz} + {}^p I_{xx} + {}^6r_z^2 m_6 + {}^p I_{xx}$$

$$\begin{aligned}
k_{16}^M &= {}^m I_4 + {}^{p^4} I_{zz} + {}^{p^5} I_{xx} + {}^{6r_z^2} m_6 + {}^{p^6} I_{xx} \\
k_{17}^M &= a_3 d_2 m_4 + a_3 d_2 m_5 + a_3 d_2 m_6 + a_3 d_3 m_4 + a_3 d_3 m_5 + a_3 d_3 m_6 \\
k_{18}^M &= -{}^{p^4} I_{xx} + {}^{p^4} I_{yy} + {}^{6r_z^2} m_6 + {}^{p^5} I_{zz} + {}^{p^6} I_{xx} - {}^{p^5} I_{yy} - {}^{p^6} I_{zz} \\
k_{19}^M &= a_2^2 m_3 + a_2^2 m_4 + a_2^2 m_5 + a_2^2 m_6 + {}^2 r_x^2 m_2 - {}^2 r_y^2 m_2 - {}^{p^2} I_{xx} + {}^{p^2} I_{yy} \\
k_{20}^M &= -d_2 d_4 m_4 - d_2 d_4 m_5 - d_2 d_4 m_6 + d_2^3 r_y m_3 - d_2^4 r_z m_4 - d_3 d_4 m_4 - d_3 d_4 m_5 \\
&\quad - d_3 d_4 m_6 + d_3^3 r_y m_3 - d_3^4 r_z m_4 + {}^3 r_y^3 r_z m_3 \\
k_{21}^M &= a_2 d_2 m_3 + a_2 d_2 m_4 + a_2 d_2 m_5 + a_2 d_2 m_6 + a_2 d_3 m_3 + a_2 d_3 m_4 + a_2 d_3 m_5 \\
&\quad + a_2 d_3 m_6 + a_2^3 r_z m_3 + d_2^2 r_x m_2 + {}^2 r_x^2 r_z m_2 \\
k_{22}^M &= a_3^2 m_4 + a_3^2 m_5 + a_3^2 m_6 + d_4^2 m_4 + d_4^2 m_5 + d_4^2 m_6 + 2 d_4^4 r_z m_4 + {}^3 r_y^2 m_3 \\
&\quad + {}^4 r_z^2 m_4 + {}^{p^3} I_{zz} + {}^{p^4} I_{xx} + {}^{p^5} I_{yy} + {}^{p^6} I_{zz} \\
k_{23}^M &= {}^m I_3 + a_3^2 m_4 + a_3^2 m_5 + a_3^2 m_6 + d_4^2 m_4 + d_4^2 m_5 + d_4^2 m_6 + 2 d_4^4 r_z m_4 + {}^3 r_y^2 m_3 \\
&\quad + {}^4 r_z^2 m_4 + {}^{p^3} I_{zz} + {}^{p^4} I_{xx} + {}^{p^5} I_{yy} + {}^{p^6} I_{zz} \\
k_{24}^M &= a_3^2 m_4 + a_3^2 m_5 - d_4^2 m_4 + a_3^2 m_6 - d_4^2 m_5 - d_4^2 m_6 - {}^3 r_y^2 m_3 - {}^4 r_z^2 m_4 - {}^{p^5} I_{zz} \\
&\quad + {}^{p^5} I_{xx} + {}^{p^4} I_{zz} - {}^{p^4} I_{yy} + {}^{p^3} I_{yy} - 2 d_4^4 r_z m_4 - {}^{p^3} I_{xx} \\
k_{25}^M &= a_2^2 m_3 + a_2^2 m_4 + a_2^2 m_6 + a_2^2 m_5 + {}^2 r_x^2 m_2 + {}^2 r_y^2 m_2 + a_3^2 m_4 + a_3^2 m_5 + d_4^2 m_4 \\
&\quad + a_3^2 m_6 + d_4^2 m_5 + d_4^2 m_6 + {}^3 r_y^2 m_3 + {}^4 r_z^2 m_4 + {}^{p^6} I_{zz} + {}^m I_2 + {}^{p^5} I_{yy} + {}^{p^4} I_{xx} \\
&\quad + {}^{p^2} I_{zz} + 2 d_4^4 r_z m_4 + {}^{p^3} I_{zz} \\
k_{26}^M &= {}^{6r_z^2} m_6 + {}^2 r_y^2 m_2 + d_4^2 m_4 + d_4^2 m_5 + d_4^2 m_6 + {}^3 r_y^2 m_3 + {}^4 r_z^2 m_4 + d_2^2 m_2 + d_2^2 m_3 \\
&\quad + d_2^2 m_5 + d_2^2 m_4 + d_2^2 m_6 + d_3^2 m_3 + d_3^2 m_4 + d_3^2 m_5 + d_3^2 m_6 + {}^1 r_y^2 m_1 + {}^2 r_z^2 m_2 \\
&\quad + {}^3 r_z^2 m_3 + {}^{p^5} I_{zz} + {}^{p^6} I_{xx} + {}^m I_1 + {}^{p^4} I_{yy} + {}^{p^2} I_{xx} + 2 d_4^4 r_z m_4 + {}^{p^1} I_{zz} + {}^{p^3} I_{xx} + 2 d_2 d_3 m_3 \\
&\quad + 2 d_2 d_3 m_4 + 2 d_2 d_3 m_5 + 2 d_2 d_3 m_6 + 2 d_2^2 r_z m_2 + 2 d_2^3 r_z m_3 + 2 d_3^3 r_z m_3
\end{aligned}$$

B.1.4 Mass/Velocity Coefficient Values

The identified values and sensitivities of the mass/velocity coefficients are given below.

Coeff. Name	Value Identified by CSLC Network	Value Reported by Armstrong et al.	Output Sensitivity (Training Data)
k_1^M	-0.002847	-0.001247	13.18
k_2^M	0.0003905	-0.00005846	12.76
k_3^M	0.005619	0.001244	7.201
k_4^M	0.00003692	0.00004000	20.34
k_5^M	-0.4562	-0.01420	0.2150
k_6^M	0.2042	0.1930	6.622
k_7^M	0.007266	0.0004323	5.330
k_8^M	-0.2149	0.02375	0.201
k_9^M	0.0006139	0.0006422	11.12
k_{10}^M	0.1387	-0.01096	1.042
k_{11}^M	0.1616	0.1796	4.903
k_{12}^M	-0.2533	-0.02135	0.4104
k_{13}^M	0.03398	0.001842	3.376
k_{14}^M	-0.1557	-0.3721	1.194
k_{15}^M	0.001942	0.0002022	9.115
k_{16}^M	0.2002	0.2018	2.858
k_{17}^M	-0.07031	-0.003809	0.8882
k_{18}^M	0.01923	0.0003022	2.334
k_{19}^M	0.9278	1.602	0.2055
k_{20}^M	-0.1710	-0.1341	1.379
k_{21}^M	0.3631	0.6903	0.3905
k_{22}^M	0.2676	0.3335	0.8969
k_{23}^M	1.036	1.163	0.8186
k_{24}^M	0.4285	-0.2984	0.5072
k_{25}^M	3.757	6.792	0.07830
k_{26}^M	2.230	2.992	0.3628

Figure B.1-1: Identified values and sensitivities for the mass/velocity coefficients of PUMA A.

Coeff. Name	Value Identified by CSLC Network	Value Reported by Armstrong et al.	Sensitivity using Training Data
k_1^M	-0.001617	-0.001247	11.65
k_2^M	0.006519	-0.00005846	10.82
k_3^M	0.009809	0.001244	7.386
k_4^M	0.0001137	0.00004000	18.94
k_5^M	-0.3535	-0.01420	0.2102
k_6^M	0.1921	0.1930	6.145
k_7^M	-0.007015	0.0004323	4.887
k_8^M	-0.2205	0.02375	0.2855
k_9^M	0.0004397	0.0006422	9.830
k_{10}^M	0.1211	-0.01096	1.075
k_{11}^M	0.1613	0.1796	4.335
k_{12}^M	-0.2431	-0.02135	0.4555
k_{13}^M	0.02788	0.001842	3.535
k_{14}^M	-0.2372	-0.3721	1.344
k_{15}^M	0.006201	0.0002022	9.008
k_{16}^M	0.2020	0.2018	2.892
k_{17}^M	-0.2670	-0.003809	0.6968
k_{18}^M	0.04889	0.0003022	2.044
k_{19}^M	1.275	1.602	0.3235
k_{20}^M	-0.1673	-0.1341	1.393
k_{21}^M	0.4887	0.6903	0.3494
k_{22}^M	0.2673	0.3335	0.9260
k_{23}^M	1.026	1.163	0.8128
k_{24}^M	0.2860	-0.2984	0.5643
k_{25}^M	4.485	6.792	0.1132
k_{26}^M	2.466	2.992	0.3521

Figure B.1-2: Identified values and sensitivities for the mass/velocity coefficients of PUMA B.

By consideration of their algebraic representations (Subsection B.1.3), it can be seen that the true values of k_4^M , k_6^M , k_9^M and k_{11}^M must all be positive.

B.2 VELOCITY TERM SUBNETWORK

The velocity subnetwork has 86 preprocessed inputs, which are described in Subsection B.2.1. The adaptive weights matrix connecting these inputs to the first hidden layer has 408 finite elements, sharing simple multiples of 17 of the 26 mass/velocity coefficients (see Subsection B.1.3).

B.2.1 Velocity Subnetwork Inputs

Denoting the row vector of preprocessed velocity subnetwork inputs as X^V , its elements can be described as follows.

$$\begin{array}{llll}
 X_1^V = 1 & X_2^V = c_4 s_5 & X_3^V = c_{23}^2 c_4 s_5 & X_4^V = c_{23}^2 c_4 c_5 s_5 \\
 X_5^V = c_{23} s_{23} c_4 s_5 & X_6^V = s_2 c_{23} c_4 s_5 & X_7^V = c_4 c_5 s_5 & X_8^V = c_2 s_{23} c_4 s_5 \\
 X_9^V = c_{23} s_{23} c_4^2 c_5^2 & X_{10}^V = c_{23} s_{23} c_4^2 & X_{11}^V = c_{23} s_{23} c_5^2 & X_{12}^V = c_5 \\
 X_{13}^V = s_2 s_{23} c_5 & X_{14}^V = c_{23} s_{23} c_5 & X_{15}^V = c_2 c_{23} c_5 & X_{16}^V = c_{23}^2 c_5 \\
 X_{17}^V = c_{23}^2 & X_{18}^V = c_{23} s_{23} & X_{19}^V = s_2 c_{23} & X_{20}^V = c_2 c_{23} \\
 X_{21}^V = s_2 s_{23} & X_{22}^V = c_2 s_{23} & X_{23}^V = c_2 s_2 & X_{24}^V = c_2^2 \\
 X_{25}^V = c_{23}^2 c_4 s_4 c_5^2 & X_{26}^V = c_{23}^2 c_4 s_4 & X_{27}^V = c_4 s_4 & X_{28}^V = c_4 s_4 c_5^2 \\
 X_{29}^V = c_{23} s_{23} s_4 c_5 s_5 & X_{30}^V = c_{23} s_{23} s_4 s_5 & X_{31}^V = c_2 c_{23} s_4 s_5 & X_{32}^V = c_{23}^2 s_4 s_5 \\
 X_{33}^V = c_{23} s_{23} c_4 c_5^2 & X_{34}^V = c_{23}^2 c_4^2 c_5 s_5 & X_{35}^V = c_{23}^2 c_4 c_5 & X_{36}^V = c_{23}^2 c_5 s_5 \\
 X_{37}^V = c_2 c_{23} c_4 c_5 & X_{38}^V = c_{23} s_{23} c_4 c_5 & X_{39}^V = c_4^2 c_5 s_5 & X_{40}^V = s_4 c_5 \\
 X_{41}^V = c_{23} s_{23} c_4 & X_{42}^V = c_{23}^2 s_5 & X_{43}^V = c_{23} s_{23} s_5 & X_{44}^V = s_5 \\
 X_{45}^V = c_2 s_{23} s_5 & X_{46}^V = s_{23} s_4 s_5 & X_{47}^V = s_{23} s_4 c_5 s_5 & X_{48}^V = c_{23} s_4 s_5 \\
 X_{49}^V = c_2 s_4 s_5 & X_{50}^V = c_{23} c_4 s_5 & X_{51}^V = c_{23} c_4 s_4 & X_{52}^V = c_{23} c_4 s_4 c_5^2 \\
 X_{53}^V = s_{23} & X_{54}^V = s_{23} c_5 & X_{55}^V = c_{23} & X_{56}^V = c_2 \\
 X_{57}^V = s_2 & X_{58}^V = s_{23} c_4^2 & X_{59}^V = s_{23} c_4^2 c_5^2 & X_{60}^V = s_{23} c_5^2 \\
 X_{61}^V = c_{23} s_4 & X_{62}^V = c_{23} s_4 c_5^2 & X_{63}^V = s_{23} c_4 s_4 c_5 s_5 & X_{64}^V = c_{23} s_5 \\
 X_{65}^V = s_{23} c_4 c_5 & X_{66}^V = s_{23} c_4 & X_{67}^V = s_{23} c_4 c_5^2 & X_{68}^V = c_2 c_4 c_5 \\
 X_{69}^V = c_{23} c_4 c_5 & X_{70}^V = c_{23} s_4 c_5 & X_{71}^V = c_{23} c_5 s_5 & X_{72}^V = c_{23} c_4 c_5 s_5 \\
 X_{73}^V = s_2 c_4 s_5 & X_{74}^V = s_{23} c_4 s_5 & X_{75}^V = s_{23} s_4 c_5 & X_{76}^V = s_2 s_4 c_5
 \end{array}$$

$$\begin{array}{llll}
X_{77}^V = s_4 s_5 & X_{78}^V = s_2 s_{23} s_4 s_5 & X_{79}^V = s_2 s_{23} c_4 c_5 & X_{80}^V = c_4 c_5 \\
X_{81}^V = c_3 s_5 & X_{82}^V = s_2 c_{23} s_5 & X_{83}^V = c_2 s_{23} s_4 c_5 & X_{84}^V = s_4 \\
X_{85}^V = s_4 c_5^2 & X_{86}^V = s_2 c_{23} s_4 c_5 & &
\end{array}$$

B.2.2 Velocity Subnetwork Weights Matrix

The 105 unique Christoffel Symbols can be determined from $X^V W^V$, where the 408 finite elements of the weights matrix are simple multiples of 17 of the 26 mass/velocity coefficients, as described below.

$$\begin{array}{llll}
W_{1,1}^V = -\frac{1}{2}k_5^M - \frac{1}{2}k_{12}^M & W_{1,2}^V = -\frac{1}{2}k_{12}^M & W_{1,21}^V = \frac{1}{2}k_5^M + \frac{1}{2}k_{12}^M & W_{1,40}^V = \frac{1}{2}k_{12}^M \\
W_{2,1}^V = k_1^M & W_{2,2}^V = k_1^M & W_{2,3}^V = k_7^M & W_{2,21}^V = -k_1^M \\
W_{2,34}^V = k_1^M & W_{2,36}^V = \frac{1}{2}k_4^M & W_{2,37}^V = k_1^M & W_{2,40}^V = -k_1^M \\
W_{2,52}^V = k_1^M & W_{2,54}^V = \frac{1}{2}k_4^M & W_{2,55}^V = k_1^M & W_{2,58}^V = -k_7^M \\
W_{2,66}^V = -\frac{1}{2}k_4^M & W_{2,69}^V = -\frac{1}{2}k_4^M & W_{2,98}^V = \frac{1}{2}k_4^M & W_{2,101}^V = \frac{1}{2}k_4^M \\
W_{3,1}^V = -2k_1^M & W_{3,2}^V = -2k_1^M & W_{3,21}^V = 2k_1^M & W_{3,40}^V = 2k_1^M \\
W_{4,1}^V = 2k_{15}^M & W_{4,2}^V = 2k_{15}^M & W_{4,21}^V = -2k_{15}^M & W_{4,40}^V = -2k_{15}^M \\
W_{5,1}^V = -2k_2^M & W_{5,2}^V = -2k_2^M & W_{5,21}^V = 2k_2^M & W_{5,40}^V = 2k_2^M \\
W_{6,1}^V = -k_3^M & W_{6,21}^V = k_3^M & W_{6,26}^V = k_3^M & W_{6,30}^V = k_3^M \\
W_{6,34}^V = k_3^M & W_{6,37}^V = k_3^M & W_{6,45}^V = -k_3^M & W_{7,1}^V = -k_{15}^M \\
W_{7,2}^V = -k_{15}^M & W_{7,21}^V = k_{15}^M & W_{7,34}^V = -k_{15}^M & W_{7,40}^V = k_{15}^M \\
W_{7,52}^V = -k_{15}^M & W_{8,1}^V = -k_3^M & W_{8,2}^V = -k_3^M & W_{8,21}^V = k_3^M \\
W_{8,26}^V = -k_3^M & W_{8,30}^V = -k_3^M & W_{8,34}^V = -k_3^M & W_{8,37}^V = -k_3^M \\
W_{8,40}^V = k_3^M & W_{8,45}^V = k_3^M & W_{9,1}^V = k_{15}^M & W_{9,2}^V = k_{15}^M \\
W_{9,21}^V = -k_{15}^M & W_{9,40}^V = -k_{15}^M & W_{10,1}^V = -k_{18}^M & W_{10,2}^V = -k_{18}^M \\
W_{10,21}^V = k_{18}^M & W_{10,40}^V = k_{18}^M & W_{11,1}^V = k_{15}^M & W_{11,2}^V = k_{15}^M \\
W_{11,21}^V = -k_{15}^M & W_{11,40}^V = -k_{15}^M & W_{12,1}^V = -k_2^M & W_{12,2}^V = -k_2^M \\
W_{12,21}^V = k_2^M & W_{12,37}^V = k_2^M & W_{12,40}^V = k_2^M & W_{12,55}^V = k_2^M \\
W_{13,1}^V = -k_3^M & W_{13,21}^V = k_3^M & W_{13,26}^V = k_3^M & W_{13,30}^V = k_3^M
\end{array}$$

$W_{13,37}^V = k_3^M$	$W_{13,45}^V = -k_3^M$	$W_{14,1}^V = -2k_1^M$	$W_{14,2}^V = -2k_1^M$
$W_{14,21}^V = 2k_1^M$	$W_{14,40}^V = 2k_1^M$	$W_{15,1}^V = k_3^M$	$W_{15,2}^V = k_3^M$
$W_{15,21}^V = -k_3^M$	$W_{15,26}^V = k_3^M$	$W_{15,30}^V = k_3^M$	$W_{15,37}^V = k_3^M$
$W_{15,40}^V = -k_3^M$	$W_{15,45}^V = -k_3^M$	$W_{16,1}^V = 2k_2^M$	$W_{16,2}^V = 2k_2^M$
$W_{16,21}^V = -2k_2^M$	$W_{16,40}^V = -2k_2^M$	$W_{17,1}^V = k_{12}^M$	$W_{17,2}^V = k_{12}^M$
$W_{17,21}^V = -k_{12}^M$	$W_{17,40}^V = -k_{12}^M$	$W_{18,1}^V = -k_{24}^M$	$W_{18,2}^V = -k_{24}^M$
$W_{18,21}^V = k_{24}^M$	$W_{18,40}^V = k_{24}^M$	$W_{19,1}^V = -k_{10}^M$	$W_{19,21}^V = k_{10}^M$
$W_{19,26}^V = k_{10}^M$	$W_{19,30}^V = k_{10}^M$	$W_{19,45}^V = -k_{10}^M$	$W_{20,1}^V = -k_{14}^M$
$W_{20,2}^V = -k_{14}^M$	$W_{20,21}^V = k_{14}^M$	$W_{20,26}^V = -k_{14}^M$	$W_{20,30}^V = -k_{14}^M$
$W_{20,40}^V = k_{14}^M$	$W_{20,45}^V = k_{14}^M$	$W_{21,1}^V = k_{14}^M$	$W_{21,21}^V = -k_{14}^M$
$W_{21,26}^V = -k_{14}^M$	$W_{21,30}^V = -k_{14}^M$	$W_{21,45}^V = k_{14}^M$	$W_{22,1}^V = -k_{10}^M$
$W_{22,2}^V = -k_{10}^M$	$W_{22,21}^V = k_{10}^M$	$W_{22,26}^V = -k_{10}^M$	$W_{22,30}^V = -k_{10}^M$
$W_{22,40}^V = k_{10}^M$	$W_{22,45}^V = k_{10}^M$	$W_{23,1}^V = -k_{19}^M$	$W_{23,21}^V = k_{19}^M$
$W_{24,1}^V = k_5^M$	$W_{24,21}^V = -k_5^M$	$W_{25,3}^V = k_{15}^M$	$W_{25,58}^V = -k_{15}^M$
$W_{26,3}^V = -k_{18}^M$	$W_{26,58}^V = k_{18}^M$	$W_{27,3}^V = k_{18}^M$	$W_{27,27}^V = -k_{18}^M$
$W_{27,31}^V = -k_{18}^M$	$W_{27,46}^V = -k_{18}^M$	$W_{27,49}^V = -k_{18}^M$	$W_{27,58}^V = -k_{18}^M$
$W_{27,63}^V = k_{18}^M$	$W_{27,64}^V = k_{18}^M$	$W_{27,67}^V = k_{18}^M$	$W_{28,3}^V = -k_{15}^M$
$W_{28,27}^V = k_{15}^M$	$W_{28,31}^V = k_{15}^M$	$W_{28,46}^V = k_{15}^M$	$W_{28,49}^V = k_{15}^M$
$W_{28,58}^V = k_{15}^M$	$W_{28,63}^V = -k_{15}^M$	$W_{28,64}^V = -k_{15}^M$	$W_{28,67}^V = -k_{15}^M$
$W_{29,3}^V = -k_{15}^M$	$W_{29,58}^V = k_{15}^M$	$W_{30,3}^V = k_1^M$	$W_{30,58}^V = -k_1^M$
$W_{31,3}^V = -k_3^M$	$W_{31,27}^V = -k_3^M$	$W_{31,31}^V = -k_3^M$	$W_{31,58}^V = k_3^M$
$W_{31,63}^V = k_3^M$	$W_{32,3}^V = -k_2^M$	$W_{32,58}^V = k_2^M$	$W_{33,4}^V = 2k_{15}^M$
$W_{33,75}^V = -2k_{15}^M$	$W_{34,4}^V = k_{15}^M$	$W_{34,75}^V = -k_{15}^M$	$W_{35,4}^V = k_2^M$
$W_{35,75}^V = -k_2^M$	$W_{36,4}^V = k_{15}^M$	$W_{36,75}^V = -k_{15}^M$	$W_{37,4}^V = k_3^M$
$W_{37,28}^V = k_3^M$	$W_{37,32}^V = k_3^M$	$W_{37,75}^V = -k_3^M$	$W_{37,80}^V = -k_3^M$
$W_{38,4}^V = -k_1^M$	$W_{38,75}^V = k_1^M$	$W_{39,4}^V = -k_{15}^M$	$W_{39,28}^V = k_{15}^M$
$W_{39,32}^V = k_{15}^M$	$W_{39,47}^V = k_{15}^M$	$W_{39,50}^V = k_{15}^M$	$W_{39,75}^V = k_{15}^M$
$W_{39,80}^V = -k_{15}^M$	$W_{39,81}^V = -k_{15}^M$	$W_{39,84}^V = -k_{15}^M$	$W_{40,4}^V = k_7^M$

$$\begin{array}{llll}
W_{40,35}^V = k_1^M & W_{40,38}^V = \frac{1}{2}k_4^M & W_{40,53}^V = k_1^M & W_{40,56}^V = \frac{1}{2}k_4^M \\
W_{40,75}^V = -k_7^M & W_{40,83}^V = -\frac{1}{2}k_4^M & W_{40,86}^V = -\frac{1}{2}k_4^M & W_{40,99}^V = \frac{1}{2}k_4^M \\
W_{40,102}^V = \frac{1}{2}k_4^M & W_{41,4}^V = -k_{15}^M & W_{41,75}^V = k_{15}^M & W_{42,4}^V = -k_1^M \\
W_{42,75}^V = k_1^M & W_{43,4}^V = -k_2^M & W_{43,75}^V = k_2^M & W_{44,4}^V = k_1^M \\
W_{44,28}^V = k_1^M & W_{44,32}^V = k_1^M & W_{44,47}^V = k_1^M & W_{44,50}^V = k_1^M \\
W_{44,73}^V = -\frac{1}{2}k_4^M & W_{44,75}^V = -k_1^M & W_{44,80}^V = -k_1^M & W_{44,81}^V = -k_1^M \\
W_{44,84}^V = -k_1^M & W_{44,88}^V = \frac{1}{2}k_4^M & W_{44,104}^V = -\frac{1}{2}k_4^M & W_{45,4}^V = -k_3^M \\
W_{45,28}^V = -k_3^M & W_{45,32}^V = -k_3^M & W_{45,75}^V = k_3^M & W_{45,80}^V = k_3^M \\
W_{46,6}^V = -k_1^M & W_{46,7}^V = -k_1^M & W_{46,8}^V = -k_7^M & W_{46,11}^V = -k_1^M \\
W_{46,12}^V = -k_7^M & W_{46,15}^V = k_1^M & W_{46,17}^V = \frac{1}{2}k_4^M & W_{46,18}^V = k_1^M \\
W_{46,62}^V = -\frac{1}{2}k_4^M & W_{46,94}^V = \frac{1}{2}k_4^M & W_{47,6}^V = k_{15}^M & W_{47,7}^V = k_{15}^M \\
W_{47,11}^V = k_{15}^M & W_{47,15}^V = -k_{15}^M & W_{48,6}^V = k_2^M & W_{48,7}^V = k_2^M \\
W_{48,11}^V = k_2^M & W_{48,15}^V = -k_2^M & W_{48,18}^V = -k_2^M & W_{49,6}^V = k_3^M \\
W_{49,15}^V = -k_3^M & W_{49,18}^V = -k_3^M & W_{50,6}^V = k_7^M & W_{50,7}^V = k_7^M \\
W_{50,10}^V = -\frac{1}{2}k_4^M & W_{50,11}^V = k_7^M & W_{50,14}^V = -\frac{1}{2}k_4^M & W_{50,15}^V = k_7^M \\
W_{50,18}^V = k_7^M & W_{50,23}^V = k_1^M & W_{50,25}^V = \frac{1}{2}k_4^M & W_{50,42}^V = k_1^M \\
W_{50,44}^V = \frac{1}{2}k_4^M & W_{50,59}^V = -k_1^M & W_{50,60}^V = -k_1^M & W_{50,92}^V = -\frac{1}{2}k_4^M \\
W_{50,93}^V = -\frac{1}{2}k_4^M & W_{51,6}^V = k_{18}^M & W_{51,7}^V = k_{18}^M & W_{51,11}^V = k_{18}^M \\
W_{52,6}^V = -k_{15}^M & W_{52,7}^V = -k_{15}^M & W_{52,11}^V = -k_{15}^M & W_{53,6}^V = -k_{20}^M \\
W_{53,7}^V = -k_{20}^M & W_{53,8}^V = -\frac{1}{2}k_{13}^M - \frac{1}{2}k_{18}^M & W_{53,11}^V = -k_{20}^M & W_{53,12}^V = -\frac{1}{2}k_{13}^M - \frac{1}{2}k_{18}^M \\
W_{53,23}^V = \frac{1}{2}k_{13}^M - \frac{1}{2}k_{18}^M & W_{53,42}^V = \frac{1}{2}k_{13}^M - \frac{1}{2}k_{18}^M & & W_{53,59}^V = -\frac{1}{2}k_{13}^M + \frac{1}{2}k_{18}^M \\
W_{53,60}^V = -\frac{1}{2}k_{13}^M + \frac{1}{2}k_{18}^M & W_{54,6}^V = k_7^M & W_{54,7}^V = k_7^M & W_{54,10}^V = -\frac{1}{2}k_4^M \\
W_{54,11}^V = k_7^M & W_{54,14}^V = -\frac{1}{2}k_4^M & W_{54,18}^V = k_7^M & W_{54,25}^V = \frac{1}{2}k_4^M \\
W_{54,44}^V = \frac{1}{2}k_4^M & W_{54,92}^V = -\frac{1}{2}k_4^M & W_{54,93}^V = -\frac{1}{2}k_4^M & W_{55,6}^V = k_{17}^M \\
W_{55,7}^V = k_{17}^M & W_{55,11}^V = k_{17}^M & W_{56,6}^V = k_{21}^M & W_{57,6}^V = -k_8^M \\
W_{58,8}^V = k_{18}^M & W_{58,12}^V = k_{18}^M & W_{58,23}^V = k_{18}^M & W_{58,42}^V = k_{18}^M
\end{array}$$

$$\begin{array}{llll}
W_{58,59}^V = -k_{18}^M & W_{58,60}^V = -k_{18}^M & W_{59,8}^V = -k_{15}^M & W_{59,12}^V = -k_{15}^M \\
W_{59,23}^V = -k_{15}^M & W_{59,42}^V = -k_{15}^M & W_{59,59}^V = k_{15}^M & W_{59,60}^V = k_{15}^M \\
W_{60,8}^V = k_{15}^M & W_{60,12}^V = k_{15}^M & W_{61,9}^V = \frac{1}{2}k_9^M + \frac{1}{2}k_{15}^M & W_{61,13}^V = \frac{1}{2}k_9^M + \frac{1}{2}k_{15}^M \\
W_{61,24}^V = -\frac{1}{2}k_9^M + \frac{1}{2}k_{15}^M & W_{61,43}^V = -\frac{1}{2}k_9^M + \frac{1}{2}k_{15}^M & & W_{61,76}^V = \frac{1}{2}k_9^M - \frac{1}{2}k_{15}^M \\
W_{61,77}^V = \frac{1}{2}k_9^M - \frac{1}{2}k_{15}^M & W_{62,9}^V = -k_{15}^M & W_{62,13}^V = -k_{15}^M & W_{62,24}^V = -k_{15}^M \\
W_{62,43}^V = -k_{15}^M & W_{62,76}^V = k_{15}^M & W_{62,77}^V = k_{15}^M & W_{63,9}^V = k_{15}^M \\
W_{63,13}^V = k_{15}^M & W_{63,24}^V = k_{15}^M & W_{63,43}^V = k_{15}^M & W_{63,76}^V = -k_{15}^M \\
W_{63,77}^V = -k_{15}^M & W_{64,9}^V = k_7^M & W_{64,13}^V = k_7^M & W_{64,19}^V = -\frac{1}{2}k_4^M \\
W_{64,79}^V = \frac{1}{2}k_4^M & W_{64,95}^V = -\frac{1}{2}k_4^M & W_{65,9}^V = k_7^M & W_{65,13}^V = k_7^M \\
W_{65,16}^V = -k_1^M & W_{65,19}^V = -\frac{1}{2}k_4^M & W_{65,79}^V = \frac{1}{2}k_4^M & W_{65,95}^V = -\frac{1}{2}k_4^M \\
W_{66,16}^V = \frac{1}{2}k_9^M - \frac{1}{2}k_{15}^M & W_{66,61}^V = -\frac{1}{2}k_9^M - \frac{1}{2}k_{15}^M & & W_{66,78}^V = \frac{1}{2}k_9^M + \frac{1}{2}k_{15}^M \\
W_{67,16}^V = k_{15}^M & W_{67,61}^V = k_{15}^M & W_{67,78}^V = -k_{15}^M & W_{68,16}^V = k_3^M \\
W_{69,16}^V = k_2^M & W_{70,16}^V = k_7^M & W_{70,24}^V = k_1^M & W_{70,43}^V = k_1^M \\
W_{70,76}^V = -k_1^M & W_{70,77}^V = -k_1^M & W_{71,16}^V = k_{15}^M & W_{71,61}^V = k_{15}^M \\
W_{71,78}^V = -k_{15}^M & W_{72,23}^V = -k_{15}^M & W_{72,42}^V = -k_{15}^M & W_{72,59}^V = k_{15}^M \\
W_{72,60}^V = k_{15}^M & W_{73,23}^V = k_3^M & W_{73,59}^V = -k_3^M & W_{74,23}^V = k_2^M \\
W_{74,42}^V = k_2^M & W_{74,59}^V = -k_2^M & W_{74,60}^V = -k_2^M & W_{75,24}^V = k_2^M \\
W_{75,43}^V = k_2^M & W_{75,76}^V = -k_2^M & W_{75,77}^V = -k_2^M & W_{76,24}^V = k_3^M \\
W_{76,76}^V = -k_3^M & W_{77,27}^V = -k_2^M & W_{77,31}^V = -k_2^M & W_{77,46}^V = -k_2^M \\
W_{77,49}^V = -k_2^M & W_{77,63}^V = k_2^M & W_{77,64}^V = k_2^M & W_{77,67}^V = k_2^M \\
W_{78,27}^V = -k_3^M & W_{78,31}^V = -k_3^M & W_{78,63}^V = k_3^M & W_{79,28}^V = k_3^M \\
W_{79,32}^V = k_3^M & W_{79,80}^V = -k_3^M & W_{80,28}^V = k_2^M & W_{80,32}^V = k_2^M \\
W_{80,47}^V = k_2^M & W_{80,50}^V = k_2^M & W_{80,80}^V = -k_2^M & W_{80,81}^V = -k_2^M \\
W_{80,84}^V = -k_2^M & W_{81,28}^V = -k_{15}^M & W_{81,32}^V = -k_{15}^M & W_{81,47}^V = -k_{15}^M \\
W_{81,50}^V = -k_{15}^M & W_{81,70}^V = k_{15}^M & W_{81,80}^V = k_{15}^M & W_{81,81}^V = k_{15}^M \\
W_{81,84}^V = k_{15}^M & W_{81,87}^V = -k_{15}^M & W_{82,28}^V = k_3^M & W_{82,32}^V = k_3^M \\
W_{82,80}^V = -k_3^M & W_{83,35}^V = -k_3^M & W_{84,35}^V = -\frac{1}{2}k_9^M + \frac{1}{2}k_{15}^M & W_{84,53}^V = -\frac{1}{2}k_9^M + \frac{1}{2}k_{15}^M
\end{array}$$

$$\begin{aligned}
W_{84,65}^V &= \frac{1}{2}k_9^M + \frac{1}{2}k_{15}^M & W_{84,68}^V &= \frac{1}{2}k_9^M + \frac{1}{2}k_{15}^M & W_{84,82}^V &= -\frac{1}{2}k_9^M - \frac{1}{2}k_{15}^M \\
W_{84,85}^V &= -\frac{1}{2}k_9^M - \frac{1}{2}k_{15}^M & W_{85,35}^V &= -k_{15}^M & W_{85,53}^V &= -k_{15}^M & W_{85,65}^V &= -k_{15}^M \\
W_{85,68}^V &= -k_{15}^M & W_{85,82}^V &= k_{15}^M & W_{85,85}^V &= k_{15}^M & W_{86,35}^V &= k_3^M
\end{aligned}$$

(all other elements are zero/null)

B.3 GRAVITY TERM SUBNETWORK

The gravity term subnetwork has nine preprocessed inputs (Subsection B.2.1) and contains only five unique coefficients (Subsection B.2.3).

B.3.1 Gravity Subnetwork Inputs

Denoting the row vector of gravity subnetwork preprocessed inputs as X^G , its elements are defined as follows.

$$\begin{aligned}
X_1^G &= c_2 & X_2^G &= s_2 & X_3^G &= s_{23} \\
X_4^G &= s_{23} c_5 & X_5^G &= c_{23} & X_6^G &= c_{23} c_4 s_5 \\
X_7^G &= s_{23} s_4 s_5 & X_8^G &= s_{23} c_4 c_5 & X_9^G &= c_{23} s_5
\end{aligned}$$

B.3.2 Gravity Subnetwork Weights Matrix

The gravity term can be found directly from

$$G = X^G W^G$$

where the gravity term weights matrix is defined as

$$W^G = \begin{bmatrix} 0 & k_4^G & 0 & 0 & 0 & 0 \\ 0 & k_1^G & 0 & 0 & 0 & 0 \\ 0 & k_5^G & k_5^G & 0 & 0 & 0 \\ 0 & k_2^G & k_2^G & 0 & 0 & 0 \\ 0 & k_3^G & k_3^G & 0 & 0 & 0 \\ 0 & k_2^G & k_2^G & 0 & 0 & 0 \\ 0 & 0 & 0 & -k_2^G & 0 & 0 \\ 0 & 0 & 0 & 0 & k_2^G & 0 \\ 0 & 0 & 0 & 0 & k_2^G & 0 \end{bmatrix}$$

B.3.3 Gravity Term Coefficients

The elements of the gravity term coefficients vector (k^G) are as follows.

$$k_1^G = {}^2r_y m_2 g$$

$$k_2^G = -{}^6r_z m_6 g$$

$$k_3^G = -a_3 m_4 g - a_3 m_5 g - a_3 m_6 g$$

$$k_4^G = -a_2 m_3 g - a_2 m_4 g - a_2 m_5 g - a_2 m_6 g - {}^2r_x m_2 g$$

$$k_5^G = {}^3r_y m_3 g - {}^4r_z m_4 g - d_4 m_4 g - d_4 m_5 g - d_4 m_6 g$$

B.3.4 Gravity Term Coefficient Values

The identified values and sensitivities for the gravity term coefficients are presented in Figs. B.3-1 & B.3-2.

Coefficient Name	Value Identified by the CSLC Network	Value Reported by Armstrong et al.	Output Sensitivity (Training Set)
k_1^G	-2.021	1.0246	0.1459
k_2^G	-0.004226	-0.028264	0.2554
k_3^G	-0.08723	0.24903	0.2816
k_4^G	-45.71	-37.250	0.07851
k_5^G	-11.66	-8.4577	0.1484

Figure B.3-1: The identified values and sensitivities for the gravity coefficients of PUMA A

Coefficient Name	Value Identified by the CSLC Network	Value Reported by Armstrong et al.	Output Sensitivity (Training Set)
k_1^G	-1.470	1.0246	0.1198
k_2^G	-0.01724	-0.028264	0.2337
k_3^G	0.9353	0.24903	0.2906
k_4^G	-46.85	-37.250	0.1070
k_5^G	-11.98	-8.4577	0.1300

Figure B.3-2: The identified values and sensitivities of the gravity coefficients of PUMA B.

B.4 VISCOUS FRICTION SUBNETWORK

The viscous friction torque contribution (τ^v), for a given joint, is simply equal to $\dot{q}_i k_i^v$. The values identified for k^v , and their sensitivities, are presented in Figs. B.4-1 & B.4-2.

Coeff. Name	Identified by CSLC Network	Reported by Leahy & Saridis	Value Obtained via Regression	Sensitivity (Training Set)
k_1^v	3.352	4.0	2.954	0.1840
k_2^v	6.660	3.5	7.875	0.01764
k_3^v	1.858	3.5	1.631	0.2295
k_4^v	0.3012	0.48	0.3398	0.5667
k_5^v	0.2174	0.55	0.2462	0.6007
k_6^v	0.3018	0.65	0.1360	0.6313

Figure B.4-1: Viscous friction coefficient values for PUMA A.

Coeff. Name	Identified by CSLC Network	Reported by Leahy & Saridis	Value Obtained via Regression	Sensitivity (Training Set)
k_1^v	6.097	4.0	5.631	0.2186
k_2^v	7.182	3.5	6.554	0.03708
k_3^v	3.006	3.5	3.013	0.2082
k_4^v	0.3536	0.48	0.3469	0.5551
k_5^v	0.3239	0.55	0.2315	0.5521
k_6^v	0.4020	0.65	0.2813	0.6141

Figure B.4-2: Viscous friction coefficient values for PUMA B.

Note that the coefficient values identified through regression are the optimal values (in terms of minimising the network error cost) for the corresponding training set, given the mass/velocity and gravity coefficients reported by [Armstrong et al. 1986].

B.5 DRY FRICTION SUBNETWORK

The dry friction torque contribution (τ^d), for a given joint, is simply equal to $\text{sgn}(\dot{q}_i) k_i^d$. The values identified for k^d are presented in Figs. B.5-1 & B.5-2.

Coefficient Name	Value Identified by CSLC Network	Value Reported by Leahy & Saridis	Value Obtained via Regression
k_1^d	9.169	5.355	9.756
k_2^d	9.778	6.138	9.374
k_3^d	5.392	3.519	5.750
k_4^d	1.047	0.963	0.9934
k_5^d	1.007	0.801	1.280
k_6^d	1.001	0.846	0.4987

Figure B.5-1: Identified dry friction coefficient values for PUMA A.

Coefficient Name	Value Identified by CSLC Network	Value Reported by Leahy & Saridis	Value Obtained via Regression
k_1^d	4.539	5.355	7.474
k_2^d	7.737	6.138	8.010
k_3^d	4.786	3.519	5.485
k_4^d	0.8837	0.963	0.9885
k_5^d	1.104	0.801	2.241
k_6^d	0.9971	0.846	1.135

Figure B.5-2: Dry friction coefficient values for PUMA B.

All dry friction coefficients have an output sensitivity of one sixth, for any given training set with non-zero joint velocities.

B.6 STATIC FRICTION VELOCITY THRESHOLDS

The joint velocity threshold magnitudes (which delimit the region in which static friction is considered significant) are presented in Figs. B.6-1 & B.6-2.

Joint	1	2	3	4	5	6
Threshold	0.16	0.083	0.12	0.78	0.82	1.6

Figure B.6-1: Static friction velocity threshold magnitudes for PUMA A.

Joint	1	2	3	4	5	6
Threshold	0.16	0.05	0.26	0.72	0.78	1.7

Figure B.6-2: Static friction velocity threshold magnitudes for PUMA B.

REFERENCES

- Armstrong, B., Khatib, O., and Burdick, K., "The explicit dynamic model and inertial parameters of the PUMA 560 arm," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 510-518, 1986.
- Leahy, M.B., Jr., and Saridis, G.N., "Compensation of industrial manipulator dynamics," *Int. J. Robotics Res.*, vol. 8, no. 4, pp. 73-84, 1989.

APPENDIX C

TECHNICAL SUMMARY

This appendix comprises a technical summary of the methodology employed in the identification of unknown coefficient values for a system of known algebraic form. As such, it is intended as a concise guide for those wishing to perform similar identification studies.

The specific application covered in this thesis, namely the identification of manipulator dynamics coefficients, is discussed first. Each step of the identification process is detailed in turn, complete with references to the pertinent equations and further information given in the main body of this thesis. The second section of this appendix discusses the application of the methodology developed in this thesis to the general class of problems which possess a known algebraic structure but unknown coefficients.

C.1 MANIPULATOR INVERSE DYNAMICS

The process of identifying a manipulator's dynamics coefficients can be broken down into seven principal stages. Most of these stages involve the manipulation of potentially huge algebraic constructs, which is best performed automatically through the use of computer software. The identification experiment detailed in this thesis made use of over 30 routines, written by the author for use with the MATLAB[®] technical computing language (version 4) and employing the additional Symbolic Math Toolbox (an implementation of MAPLE[®] V). This software was written for the general case, that is, it is applicable to the problem of identifying the dynamics coefficients of any given manipulator, and can be made available to interested parties upon request.

The seven principal stages for the identification of manipulator inverse dynamics coefficients are as follows;

- Allocation of coordinate frames
- Formation of Pseudo-Inertia Matrices and Homogeneous Transforms
- Determination of Inertia Matrix, Christoffel Symbols and Gravity Term

- Separation of Variable Inputs from Constant Coefficients
- Construction of CSLC Network
- Collection of Empirical Training Data
- Identification of Coefficients via Training of the CSLC Network

Each of these operations is described in detail in the following subsections.

C.1.1 Allocation of Coordinate frames

Choose a system of coordinate frames. One frame is required per degree of freedom possessed by the manipulator. A judicious choice of frames can result in a significant reduction in the algebraic complexity of the base homogeneous transforms, a recommended allocation method is that of the modified Denavit-Hartenberg convention (see Subsection 2.6.1).

Regardless of the allocation convention used, the resultant set of n coordinate frames can be uniquely summarised by $4n$ Denavit-Hartenberg parameters. If accurate data is available, measured values for some or all of the $3n$ constant D-H parameters may be used instead of undefined algebraic terms.

C.1.2 Formation of Pseudo-Inertia Matrices and Homogeneous Transforms

Form the pseudo-inertia matrices (I_i) using Eqn. 2.3-17. Alternatively, if the corresponding frames are known to be principal axes, the simplified form given in Eqn. 2.3-20 may be used. If accurate a priori knowledge of the link masses and/or centre of gravity coordinates is available, these values can be used in the pseudo-inertia matrices instead of the corresponding undefined algebraic representations. (It is not possible to accurately measure the inertia terms that appear in I_i)

Form the base homogeneous transforms (T_i) from the D-H parameters, using Eqns. 2.6-1 and 2.6-2.

C.1.3 Determination of Inertia Matrix, Christoffel Symbols and Gravity Term

Form the algebraic representations of the elements of the mass matrix (M_{ij}) from the pseudo-inertia matrices and base homogeneous transforms, using Eqns. 2.3-24 and 2.5-6.

Form the algebraic representations of the Christoffel symbols (c_{ijk}) from the elements of

the mass matrix, using Eqn. 2.4-5.

Form the algebraic representations of the elements of the gravity effect vector (G_i) from the pseudo-inertia matrices and base homogeneous transforms, using Eqns. 2.3-28 and 2.4-7.

C.1.4 Separation of Variable Inputs from Constant Coefficients

Factorise each element of the quantities determined in the previous step for each unique product of position dependent (variable) quantities they contain (see Section 5.1). This produces a summation of terms where each term consists of a product of position dependent quantities, or unity, factored by a group of constant parameters. For each of the mass, velocity and gravity terms, find the list of unique position dependent products that appear within the factorised elements of the mass matrix, Christoffel symbols and gravity effect vector respectively. Similarly, find the minimum set of coefficients which includes known multiples of all the groupings of constant parameters identified in the same quantities. To simplify identification, express the velocity term coefficients in terms of the mass term coefficients (see Subsection 5.1.2).

C.1.5 Construction of CSLC Network

The CSLC network for manipulator inverse dynamics comprises several distinct subnetworks (see Section 5.1).

Form the mass, velocity and gravity subnetworks' preprocessed input vectors from the corresponding list of unique position dependent products. For each of these subnetworks, the first computational layer comprises the unique elements of either the mass matrix, Christoffel symbols or gravity effect vector. Thus, for a given subnetwork, each element of the weights matrix connecting the preprocessed input layer to the first computational layer is defined as being equal to that coefficient (or multiple thereof) which factorises the associated input in order to produce a term within the corresponding unique element of the mass matrix, Christoffel tensor or gravity effect vector.

The algebraic form of the friction term is potentially the same for any given manipulator (see Subsection 2.5.1). Incorporate the viscous, dry and, if desired, static friction subnetworks into the CSLC using the simple subnetworks shown in Subsections 5.1.4 and/or 8.2.1.

C.1.6 Collection of Empirical Training Data

Devise a training trajectory with a high degree of excitation and low redundancy (such as that discussed in Subsection 7.4.1). Allow the manipulator to perform the training trajectory whilst recording the instantaneous joint positions, velocities and accelerations as well as the applied force/torques. If direct measurement of the applied torques is not possible, the relationship between demand torque and applied torque must be identified (see Subsection 7.2.3). If measurement of the joint velocities and/or accelerations is not possible, the positional (or velocity) data must be smoothed and differentiated. The recommended method for this is averaged spline smoothing (see Section 7.3).

Data which describes the inverse dynamics of a joint whilst it is experiencing backlash should be identified so as not to adversely affect identification of the non-backlash dynamics coefficients (see Subsection 7.2.4).

For the purpose of assessing network performance, it is useful to produce a comparison data set, in the same fashion as the training set, from a distinct trajectory.

C.1.7 Identification of Coefficients via Training of the CSLC Network

Train the CSLC network constructed for the given manipulator. Estimates of dynamics coefficients' values (if available) can be used for the initial network weights values in order to hasten training. Gradient descent learning algorithms (such as Backpropagation, see Subsection 3.4.2) will benefit from the CSLC network's sole error hypersurface minima and provide a rapid means of identifying approximate values for the dynamics coefficients. Fine tuning algorithms that do not rely on the error surface gradient (such as PERAL, see Section 6.2) are then recommended for the accurate identification of the coefficients.

The results of training can be assessed by comparing the recorded applied force/torques for a set of data not used in the network training process (the comparison trajectory data) against the force/torques predicted by the CSLC network.

C.2 OTHER SYSTEMS

The methodology used in this work to obtain accurate estimates of a manipulator's inverse dynamics coefficients is applicable to many other engineering systems. In all such

cases the overall process is identical and can be summarised as follows;

- Derivation of the symbolic form of the system equation.
- Simplification of the system equation via replacement of algebraic terms with numeric values, if reliable a priori knowledge is available.
- Separation of unknown constants and known variables.
- Generation of an adaptive network with inputs comprising the known variables and weights comprising the minimised set of unknown constants.
- Training of the formulated network with empirical data.

The results from such a network will generalise well if the symbolic system equation used is a good model of the system's true behaviour. Training will benefit from an error hypersurface with only one minima if the network outputs are linearly related to the network weights, which is achieved through the exclusive use of linear network activation functions.

