National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

Dynamic estimation for serial manipulators

Behram Kapadia

Department of Electrical Engineering
McGill University, Montreal

March 1995

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Masters of Engineering

March 1995

Abstract

Many industrial manipulators employed in industry today have serial rotational architectures; i.e. they are open-loop kinematic chains with rotating joints. As the tasks performed by these manipulators become more involved, there is a need for enhanced control. Model based control techniques for manipulators are based on the availability of a *dynamic model* that relates the generalised forces at the manipulator joints to the demand trajectory. Given an accurate dynamic model; model based control significantly improves the positioning and tracking capabilities of the manipulator.

In practice, the inertial parameters of a manipulator are often not known accurately and have to be calculated based on experimental data; this presents several problems due to inaccurate instrumentation as well as unavailability of a full order model. Since model based control is sensitive to model inaccuracies, dynamic estimation is an important issue in model based control.

This thesis considers the development of an accurate dynamic model for two industrial manipulators. The accuracy of available instrumentation for recording sensor-motion data required the application of practical methods for improving the estimation bounds. This thesis presents some methods for improving the estimation accuracy of the reduced order dynamic model, particularly with regard to the importance of a correct formulation for the base parametric set and the dependance of parameter estimation accuracy on the torque sensitivity.

The thesis also experimentally implements an algorithm for trajectory optimisation for estimation by the LMS adaptive law. Practical methods are suggested for carrying out the optimisation.

Results are presented based on experimentation conducted on an electric and a hydraulic manipulator; a brief description is also provided on the software that was developed for ths purpose. Identification results in one case were seen to closely approximate those previously determined by explicit measurements by Khatib et al. Finally, a comparison between ordinary PD control and feedforward control using the available model shows the promise of actual ability to implement model based control. However, a comparison of the results of the control performance between a direct drive manipulator and a geared manipulator indicate that unmodelled dynamics play a central role in increasing the error bounds of the identified parmeters.

Resume

Plusieurs manipulateurs industriels utilisés aujourd'hui sont conçus selon des architectures sérielles rotationelles, c'est-à-dire des chaînes cinématiques ouvertes utilisant des joints rotationels. Comme les tâches effectuées par ces robots deviennent de plus en plus complexes, on tend à utiliser des algorithmes de contrôle plus sophistiqués. Les algorithmes de contrôle basés sur un modèle du robot dépendent directement de la disponibilité d'un modèle dynamique exprimant les forces généralisées au niveau des joints du robot en fonction de la trajectoire demandée. Assumant un modèle dynamique précis, les techniques de contrôle basées sur un modèle du robot améliorent significativement les possibilités de positionnement et de poursuite du manipulateur.

En pratique, les paramètres inertiels d'un manipulateur ne sont pas connues avec précision et doivent être calculées à partir de données expérimentales. L'absence d'un modèle complet de même que l'imprécision des équipements de mesure causent certains problèmes. Puisque les algorithmes de contrôle basés sur un modèle du robot sont très sensibles à la précision de ce dernier, l'estimation dynamique du modèle devient un facteur déterminant pour ce type de contrôle.

Dans cette thèse, nous développerons un modèle dynamique précis de deux manipulateurs industriels. La précision des appareils de mesure disponibles pour enrégistrer les données de mouvement nous a contraint à utiliser des méthodes expérimentales pour améliorer les bornes de l'estimation. Nous suggérons quelques méthodes pour augmenter la précision de l'estimé dans le cas d'un modèle dynamique d'ordre réduit, particulièrement l'importance d'une formulation correcte de l'ensemble des paramètres de la base (du robot) et de la dépendance de la précision des paramètres estimés à la sensibilité au couple.

Aussi, dans cette thèse, nous implantons un algorithme d'optimisation de trajectoire, déjà existant dans la littérature, pour l'estimation utilisant la loi adaptative LMS. Nous suggérons diverses méthodes pour l'implantation.

Finalement, suite à une comparaison entre le contrôle proportionel-dérivée ordinaire et le contrôle *feedforward*, nous montrons qu'il est possible grâce au modèle développé ici, d'implanter un algorithme de contrôle basé sur un modèle du robot. Cependant, en comparant les résultats, on note une différence appréciable entre un robot utilisant un entraînement direct et un robot entraîné par l'entremise d'une boîte d'engrenages. On y voit que les

paramètres dynamiques non modélisés jouent un rôle crucial sur la borne de l'erreur des paramètres identifiés.

Acknowledgements

I would firstly like to express my gratitude to my thesis supervisor, Dr.L.K.Daneshmend. His guidance, broad knowledge of the subject and support was essential to the completion of this thesis.

I would also like to sincerely thank Dr.V.Hayward for the Lisp code by Izaguirre et al. for dynamic model generation. The dyn_model code for this thesis was partly written based on a procedure delineated by Khalil and Gautier and borrows methods from the above mentioned code.

The trajectory data and the consequent feedforward tests for the Sarcos manipulator were provided by Michel Doyon (at the Institut de Recherche d'Hydro-Québec). I would like to thank him for his help as well as for providing the Sarcos figure and for translating the abstract. I am very grateful to Robert Lucyshyn and Duncan Baird for the several hours they spent in helping me with the identification experiments and to Ayman Bedair for his help with the diagrams. I am very grateful for the facilities I could avail of at the McGill University Centre for Intelligent Machines for carrying out the identification experiments.

Finally, a sincere gratitude to my parents for their support that enabled me to pursue this program.

Notation

We give below, a list of symbols used in this thesis.

T	:the vector containing the joint torques
M	:the manipulator mass matrix with acceleration coefficients
V	:the velocity matrix with Centrifugal and Coriolis terms coefficients
F	:the friction coefficients
G	:the gravitational coefficients
Θ	:the joint angle vector
$\dot{\Theta}$	:the joint angular velocity vector
$\ddot{\Theta}$	:the joint acceleration vector
m_i	:the mass of the i^{th} link
$m\rho_i$	:the first moment of the i^{th} link
J_i	:the second moment of inertia of the i^{th} link
P	:the vector containing the dynamic parameters of the manipulator
H	:the inverse dynamic regressor matrix
ω_i	:the angular velocity of the i^{th} link
$\dot{\omega}_i$	:the angular acceleration of the i^{th} link
$\dot{v}_i$	:the acceleration of the origin of the frame attached to the i^{th} link
N_i	:the inertial moment due to the i^{th} link rotation
F_i	:the inertial force due to the motion of the i^{th} link
f_i	:the force exerted by the $(i+1)^{th}$ link on the i^{th} link
n_i	:the moment exerted by the $(i+1)^{th}$ link on the i^{th} link
$\hat{M}$	:the regressor matrix containing only the acceleration terms
$\hat{V}$	:the regressor matrix containing only the velocity terms
$\hat{G}$	:the regressor matrix containing only the gravity terms
$\hat{F}$	:the regressor matrix containing only the friction terms
U_H, S_H, V_H	:the matrices obtained by svd for the regressor matrix
C_T	:the coveariance of the motion noise
C_{P^a}	:the covariance of the *a priori* parameter vector
Γ_n	:the gain matrix in the LMS algorithm
$\tilde{\theta}$	:the parameter vector
R	:the regressor correlation matrix
C_{H_k}	:sensor noise correlation
J	:the cost function for trajectory optimisation for the LMS algorithm
μ_m	:the incremental gain in the optimisation procedure

C_k :the parametric error covariance matrix at the k^{th} iteration
K_k :the Kalman gain at the k^{th} iteration
Q_k :the covariance matrix of the process noise vector, w_k
R_k :the covariance matrix of the measurment noise

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Description

Industrial robots are generally controlled by simple linear feedback controllers, which are employed for their ease of implementation. A linear design based on stability and error analysis for linear feedback control systems is unable to consider the fluctuating relationship between the applied forces and the resultant motion changes; in some cases this relationship changes rapidly enough so that linear controllers designed for conservative margins produce large tracking errors for reasonably fast trajectories. Hence a controller design based on a linear manipulator model for even a restricted work space may be unacceptable for some industrial applications; the performance in most cases can be improved by employing more effective controllers.

One approach to achieving a controller with better tracking capabilities is to incorporate the dynamic model of the mechanical manipulator. The

model consists of explicit expressions relating the output trajectory to the actuator forces. A control algorthm based on the dynamic model can compensate the effects of inertial, centrifugal, Coriolis, gravity and frictional forces more effectively than a linear controller. The performance of a model based controller is related to the accuracy of the identified model and the order of the dynamic model incorporated. Unfortunately, existent model-based algorithms such as the computed torque method [1] are not robust to significant parameter bias. Consequently, it is necessary to identify the dynamic parameters as accurately as possible.

Given a dynamic model structure, an identification procedure can process the manipulator input-output (torque-motion) data to yield parameter estimates. In most cases, the data contains additive measurement noise; the system may itself be inherently noisy, in which case, *a priori* statistics are required to process the available data.

In the case of mechanical manipulators, the sources of noise are: the measurement (sensor) noise which leads to an error in the trajectory information and motion (process) noise which represents the error in the measured actuator force/torque. A manipulator cannot be completely modelled on account of the difficulty in deriving models for the frictional forces and forces due to the flexible modes. As selected models for dynamic estimation generally incorporate only the inertial forces, the contribution due to unmodelled dynamics of the manipulator leads to an error in identified parameters. The unmodelled dynamics contribute to the *systematic error*, so called since they can be eliminated by using a full-order dynamic model. Generally, systematic errors are also modelled as random vectors with known statistics. These

unmodelled dynamics present a major obstacle to accurate dynamic estimation.

Better excitation of the input trajectories for experimental data is the principal method of increasing the accuracy of the identification procedure. The choice of an experimental trajectory is generally based on minimising an error criterion which is a function of the trajectory. The selection of the trajectory depends on the employed identification procedure. The applicability of methods for generating suitable excitation trajectories is, however, limited because:

1. there is an unavoidable bias due to the unmodelled dynamics as they are correlated with the regressor; the correlations cannot be minimised as explicit expressions are unavailable.
2. actuator contraints limit the possible excitation.

This thesis studies available techniques in parametric identification applied to serial, rotational manipulators modelled as an open kinematic chain of rigid bodies. Experimental results are derived by applying two criteria for estimation; the batch least squares and the least mean square error criterion. Since the sensor data only consist of the position sensors and the force/torque (current/pressure) sensors, the effectiveness of identification methods is demonstrated for very noisy data. Workspace constraints, actuator limitations and particularly the unmodelled dynamics make dynamic estimation a difficult problem. Trajectory optimisation is conducted using appropriate error criteria. Finally, trajectory tracking employing a dynamic

model based on the identified parameters gives an indication of the accuracy of the identification experiment.

1.2 The manipulator model

In this section, we briefly describe the notation used to specify the manipulator architecture in this thesis. The type of manipulator we shall consider is an open-chain mechanism consisting of $n+1$ rotating rigid links, starting with the base link. In order to describe the motion of the manipulator, a kinematic model is required. A Cartesian coordinate frame $F_i(X_i, Y_i, Z_i)$ is attached to the i^{th} link. The modified Denavit-Hartenberg notation is used to assign the coordinate frames as this notation is well suited to the dynamic modelling of manipulators. Frame assignment by this notation is as follows:

- Z_i is along the i^{th} joint axis
- X_i is along the common perpendicular to Z_i and Z_{i+1}, directed from Z_i to Z_{i+1}.
  If Z_i and Z_{i+1} are parallel, X_i is directed from Z_i to Z_{i+1} such that O_i coincides with O_{i-1}

The following parameters then constitute the modified HD parameters.

- α_i is the angle between the Z_{i-1} and Z_i axes, measured about X_{i-1}.
- a_i is the distance between Z_{i-1} and Z_i.

- b_i is the Z_i coordinate of intersection of X_{i-1} and Z_i.
- θ_i is the angle between X_{i-1} and X_i, measured about Z_i.

The assignment is illustrated in fig. 1.1. For a rotational manipulator,

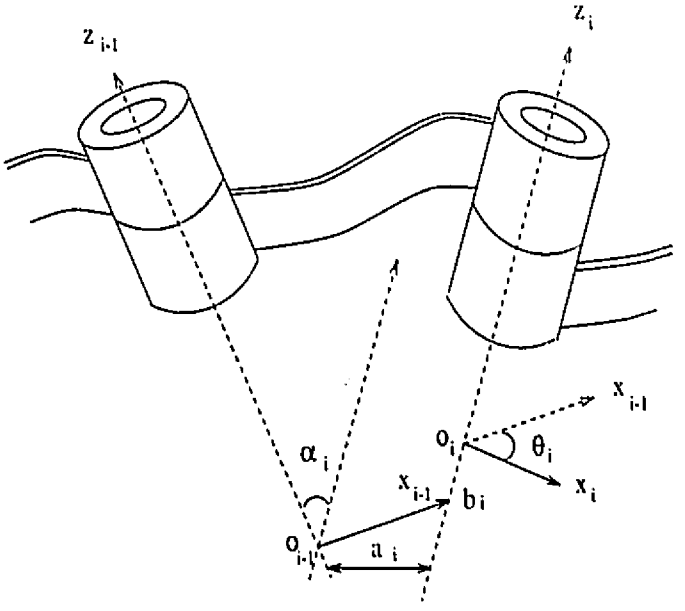

Figure 1.1: The modified DH notation

α_i,a_i and b_i are constant. The orientation of coordinates of the i^{th} frame, F_i, with respect to F_{i-1} coordinates is given by the matrix A_i^{i-1} where:

$$A_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \cos\alpha_i \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \\ \sin\alpha_i \sin\theta_i & \sin\alpha_i \cos\theta_i & \cos\alpha_i \end{bmatrix}$$

The position vector of O_i with respect to F_{i-1} is denoted by l_i and given by:

$$l_i = \begin{bmatrix} a_i \\ b_i \sin\alpha_i \\ -b_i \cos\alpha_i \end{bmatrix}$$

1.3 Dynamic behaviour of the manipulator

The manipulator inverse dynamics consist of the expressions for the joint torques due to the relative motion of the links. Due to the cross coupling forces; i.e. contributions to the joint torque due to the motion of a distal link, dynamic estimation based on joint motion data cannot be carried out for individual links. Also, in a practical manipulator, the friction and other unmodelled forces may exceed the contribution due to an inertial mode. It therefore becomes necessary to employ certain methods to eliminate the less significant modes and maximise the accuracy of estimating the remaining inertial parameters.

1.4 Overview of thesis

In chapter 2, we survey previous work in dynamic estimation for manipulators. Chapter 3 briefly states the Newton Euler algorithm for manipulator inverse dynamics and presents two theorems for obtaining the base parametric set of rotational manipulators. In chapter 4, the LS algorithms available for linear parametric estimation are studied; methods for improving the accuracy of the least squares estimation procedure incorporating a significance

analysis and column scaling criterion are developed.

Chapter 5 describes the least mean squares algorithm applied to dynamic estimation by Armstrong; the chapter is primarily concerned with the implementation of the multistage optimisation algorithm suggested by Armstrong for generating an exciting trajectory for the Puma 560 manipulator; practical methods are suggested for implementing the algorithm. The results of the procedure conducted for the 3 degree of freedom Puma 560 arm are presented.

In Chapter 6, we describe the two manipulators used for the estimation experiments and also summarize the software implementations of the estimation algorithms and algorithms for automatic model generation.

Chapter 7 presents the results of the analysis using recorded data for the two manipulators. Finally, in chapter 8, we conclude with a summary of inferences and suggestions for future work.

Chapter 2

Literature Survey

2.1 Introduction

The present chapter surveys the literature on the available system identification procedures and their application to the dynamic calibration problem. A high degree of correlated, unmodelled dynamics e.g. dissipative forces, flexible modes etc. and noisy position sensors lead to difficulties in parameter estimation when a high accuracy is required for a precise dynamic model. Least squares estimation is the most popular method for dynamic estimation and has been widely applied. Therefore the major part of this chapter deals with the literature in least squares estimation.

Model based control is the main objective of dynamic calibration; the accuracy required of estimated values is related to the tracking accuracy of a model based controller. The last section considers previous work on the robustness of model based controllers, namely the feedforward and computed

torque methods, in the presence of model uncertainties.

2.2 The system model

Parameter identification is generally based on the manipulator inverse dynamics formulation. The inverse dynamics of a manipulator are constituted by the expressions for the generalized inertial forces of the manipulator based on the desired state and the inertial parameter vector.

Employing the Euler-Lagrange equations ,[2], the manipulator inverse dynamics can be expressed by the matrix equation:

$$T = M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta)$$

where $\Theta, \dot{\Theta}, \ddot{\Theta}$ are the position, velocity and acceleration vectors respectively, M is the manipulator mass matrix, V denotes the contribution due to the centrifugal and Coriolis forces and G is the contribution due to the gravitational torque.

Luh, Walker and Paul [3] derive the recursive Newton-Euler algorithm for the manipulator inverse dynamics in terms of the link motion and inertial moments about its centre of mass. Employing the same recursive algorithm, Atkeson et al [4] derive linear expressions for the generalized forces in terms of the link moments about the joint axes. A linear model is preferred for purposes of estimation as fewer parameters are required for the estimation procedure, leading to improved problem conditioning. The system governing motion of a manipulator can therefore be given as:

$$H(\Theta, \dot{\Theta}, \ddot{\Theta})P = T_i + T_d$$

where H is the coefficient matrix whose elements are nonlinear functions of the manipulator state, P is the manipulator inertial vector given as:

$$P = [m_i, m\rho_i, J_i]$$

T_i is the N-dimensional vector of generalized forces and T_d represents the unmodelled dynamics.

2.3 The base parametric set

For accurate calibration, a full rank regressor must be incorporated in the estimation model. It is impossible to estimate all the link inertial parameters from the motion data due to the restricted degrees of freedom of each link on account of its constrained motion. A *base parametric set* constituted by a minimal, independant set of inertial paramters that generate the dynamic equations can be identified, given the kinematic model.

Numerical methods for identification of the base parametric set have been presented by Izaguirre et al. [5] and Sheu and Walker [6]. In [5], identification is based on generating random regressor equations and deriving a full rank regressor. Sheu and Walker [6] derive the base parametric set from the Lagrangian by investigating the basis functions generated by the Hamiltonian difference equations. The SVD algorithm is used to obtain the base parametric set. Gautier and Khalil [7] identify the parameters explicitly and give general regrouping relations for the inertial parameters from the basis functions.

Mayeda et al. [8] have derived an explicit base parametric set for a gen-

eral parallel, perpendicular manipulator. The inverse dynamic expressions are derived from the Euler Lagrange equations and the basis functions are evaluated. The coefficients of the basis functions then form the base parametric set as they generate a full rank regressor. The following theorem by Mayeda gives the basis coefficients derived for a parallel, perpendicular manipulator:

Theorem (Mayeda[8]). For a general parallel, perpendicular manipulator with rotational joints, the following constitute the regressor coefficients of the base parametric set:

$c_{J_j^z}$, $c_{R_j^z}$, $c_{R_j^y}$, $c_{J_i^x}$, $c_{J_i^{xz}}$, $c_{J_i^{xy}}$, $c_{J_i^{yz}}$

$$\forall 1 \leq j \leq N, \alpha_2 \leq i \leq N$$

where α_2 is the joint index of the second parallel cluster.

Since the manipulators considered for calibration in this thesis fall in this category, the theorem provides a convenient method for identifying the base coefficients.

In a later chapter, we try to develop the inverse dynamic equations suitable for base parametric evaluation employing the Newton-Euler algorithm; this develops a better intuitive understanding of the base parameters.

2.4 Estimation methods

Methods of linear system identification presented in the literature include batch least squares estimation, adaptive LMS procedures and stochastic

methods. Ideally, the problem of identification in the presence of noise is stochastic; however, with poor information regarding noise statistics, LS methods have found applicability in most cases.

2.4.1 Least squares procedures

The classical linear least squares method for parametric estimation has been widely applied in the case of dynamic calibration. The least squares problem is considered in detail in [9]. The standard linear least squares estimation procedure involves orthogonal decomposition techniques, the QR and SVD decomposition that employ the Householder transformations. Lawson and Hanson, [9], derive the perturbation bounds for the Least Squares problem; practical methods for improving the robustness of the estimation are also developed; chiefly among these are ridge regression and the method of weighted least squares. Atkeson et al. [4] estimate the load and link parameters of the PUMA600 and the MIT Serial Link Direct Drive Arm using least squares and the Newton-Euler equations. The methods of ridge regression and reduced parameter identification were observed to give near optimal results for estimation employing the pseudo-inverse. Sensor errors were found to be particularly significant when the position signal was differentiated twice for the acceleration. However, identification based on explicit integration of the dynamic equations was found to give worse results than the differentiation method.

Armstrong et al. [10] experimentally measured the inertial parameters of the PUMA560 manipulator after disassembling the links. The experimental

procedures were simplified by several heuristics; the results have been used for evaluating the accuracy of identification experiments conducted on the PUMA560 manipulator in the present thesis.

The adaptive algorithms for linear systems include the Recursive Least Squares (RLS), the Least Mean Squared Error (LMS) algorithm and optimal filtering algorithms (Kalman filter). In the latter case, implementation requires *a priori* noise statistics for the motion noise. Adaptive identification algorithms are considered in detail in [11] and [12]. Variance and convergence properties of the LMS algorithm are studied in [13]; expressions are also derived for the rate of parametric convergence.

The LMS algorithm was used for estimating the inertial parameters of rigid body manipulators by Craig [1] and Armstrong [14]. Sastry [15] derives the persistant excitation condition for the trajectory in order to obtain parameter convergence using the LMS algorithm. Armstrong [14] derives expressions for the steady state error expectation in terms of the sensor and motion noise statistics. A dimensionless quantity, *bias susceptibility* is introduced as an indicator of the robustness of the experiment to systematic error. Error minimisation, based on maximising the minimum singular value of the input correlation matrix was also considered by optimising the input trajectory based on the multistage optimisation technique by Bryson and Ho, [16]. Application of the method for trajectory optimisation of the Adept1 arm and the MIT/Asada Arm was seen to improve the bias susceptibility as well as to reduce the convergence time.

Gautier and Khalil [17] derive a method for estimating manipulator inertial parameters based on an energy method. The method obviates the need

for the acceleration data (measurement). A method to minimise the effect of noise and error modelling on the LS solution is presented and simulated. The constrained optimisation algorithm is based on a conjugate type method. The results were seen to improve when the number of equations increased but no more than twice the number of unknown parameters. The regressor condition was not sensitive to perturbations. As the demanded position and velocity are generally accurately tracked by a standard PID controller, the method is also feasable experimentally since the actual trajectory does not significantly differ from the demanded trajectory.

The LMS and RLS estimation methods can be derived as special cases of the discrete time Kalman filter; stability properties and expressions for error analyses conducted for the discrete time Kalman filter [18] [19] can be applied for improving the accuracy of the parametric identification given the availability of known noise statistics.

2.5 Application to model based control

The primary applicability of the identified dynamic model lies in its incorporation in model based control. Among the established methods of model based control are feedforward control and computed torque control. The experimental results of feedforward compensation have been presented in [20] and [21]. Tracking was observed to deteriorate significantly with poor knowledge of inertial parameters. The computed torque control law was proposed by Craig [1] and proved globally stable by Sastry [15]. Egeland [22] evaluated the robustness of this method by evaluating the stability of the linearised dy-

namic equations. The linearised dynamic equations give the evolution of the state error by the transition equation:

$$\frac{d}{dt}\begin{bmatrix} \Theta - \Theta_0 \\ \dot{\Theta} - \dot{\Theta}_0 \end{bmatrix} = (A_{nom} + \delta A)\begin{bmatrix} \Theta - \Theta_0 \\ \dot{\Theta} - \dot{\Theta}_0 \end{bmatrix}$$

where δA is a function of the model inaccuracies. Egeland derives inclusion regions for the modes of the perturbed system in terms of the model error using the block Gerschgorin theorem. A simulation shows the system eigenvalues to be very sensitive to inaccuracies in the inertial parameters as compared with the unmodelled dynamics. This result shows that model based control would only be effective in the presence of an accurate inertial dynamic model. This result shows the importance of accurately estimating the inertial parameters for model based control.

Chapter 3

Manipulator dynamics

3.1 Introduction

In this chapter, we first present the well known Newton Euler inverse dynamics algorithm. These equations are then employed in formulating rules for identifying the base parametric set of a general rotational manipulator.

3.2 The Newton Euler algorithm

Using this algorithm, we first calculate the velocities and accelerations of each link with reference to the base frame by a forward kinematic iteration and then derive the joint torques by an inverse iteration from the last link. The effect of gravity is simulated by accelerating the base link by $-g$. If the second moment of inertia about the joint axis is employed, the resulting expressions are linear in the link inertial parameters. The algorithm is stated

below:

$$\omega_0 = 0. \qquad \dot{\mathbf{v}}_0 = -\mathbf{g}$$

for $i = 1$ to N,

$$\omega_i = A_{i-1}^i \omega_{i-1} + \hat{\mathbf{e}}\dot{q}_i \tag{3.1}$$

$$\dot{\omega}_i = \frac{d}{dt}\omega_i$$

$$\hat{\mathbf{e}}\ddot{q}_i + A_{i-1}^i \dot{\omega}_{i-1} + A_{i-1}^i \omega_{i-1} \times \hat{\mathbf{e}}\dot{q}_i \tag{3.2}$$

$$\dot{\mathbf{v}}_i = A_{i-1}^i \dot{\mathbf{v}}_{i-1} + A_{i-1}^i \left(\dot{\omega}_{i-1} \times \mathbf{l}_{i-1} + \omega_{i-1} \times (\omega_{i-1} \times \mathbf{l}_{i-1})\right) \tag{3.3}$$

$$N_i = \frac{d}{dt}\left(J_i \omega_i\right)$$

$$= J_i \dot{\omega}_i + \omega_i \times J_i \omega_i \tag{3.4}$$

$$F_i = \left(m_i \dot{\mathbf{v}}_{\rho i}\right)_i$$

$$= m_i \dot{\mathbf{v}}_i + \dot{\omega}_i \times \mathbf{m}\rho_i + \omega_i \times (\omega_i \times \mathbf{m}\rho_i) \tag{3.5}$$

$\mathbf{f}_i$ = the force on link i by link $i+1$ represented in frame i.

$\mathbf{n}_i$ = the torque on link$i$by link $i+1$ represented in frame i.

for $i = N$ to 1,

$$\mathbf{f}_{i-1} = A_i^{i-1}\left(\mathbf{f}_i - m_i \dot{\mathbf{v}}_i - \dot{\omega}_i \times \mathbf{m}\rho_i - \omega_i \times (\omega_i \times \mathbf{m}\rho_i)\right) \tag{3.6}$$

$$\mathbf{n}_{i-1} = A_i^{i-1}\left(\mathbf{n}_i + \mathbf{l}_i \times \mathbf{f}_i - \mathbf{m}\rho_i \times \dot{\mathbf{v}}_i - J_i \dot{\omega}_i - \omega_i \times J_i \omega_i\right) \tag{3.7}$$

$$\Gamma_i = A_{i-1}^i \left(-\mathbf{n}_{i-1}\right).\hat{\mathbf{e}} \tag{3.8}$$

3.3 Base parametric identification

As mentioned in chapter 2; the minimal set of inertial parameters whose values can determine the dynamic model of the mechanical manipulator

uniquely, is termed the *base parametric set*. Since the relative motion of adjoining links is restricted to one degree of freedom, a given link may have a limited number of degrees of freedom with respect to the base link. The inertial parameters pertaining to these degrees of freedom do not affect the generalised forces and are redundant and non-identifiable from the trajectory data. Inertial quantities having degrees of freedom with reference to the base joint axis but not a distal joint axis cannot be individually identified but can be expressed as linear combinations of other link parameters. These are termed the dependant parameters. The base parametric set is only a property of the manipulator architecture.

3.3.1 An analytic evaluation

Based on the Newton Euler dynamical equations, we give below two theorems for identifying the base parameters of rotational manipulators. As far as the author is aware, the following two theorems are original contributions.

Theorem 1

For a rotational manipulator; the mass, m_{i+1} of the $(i+1)^{th}$ link is dependant on J_i and $m\rho_i$; the dependance is given by the following relations:

$$\mathbf{m}\rho_i^{aug} = \mathbf{m}\rho_i + m_{i+1}\mathbf{l}_{i+1}$$

$$\mathbf{J}_i^{aug} = \mathbf{J}_i + m_{i+1}[\mathbf{l}_i.\mathbf{l}_i I - \mathbf{l}_i\mathbf{l}_i^T]$$

Proof: From the manipulator dynamical equations of section 1, the total

torque contribution at joint axis i due to J_i, $\mathbf{m}\rho_i$ and m_{i+1} is given as:

$$\begin{aligned}
\mathbf{n}_i &= \frac{d}{dt}(\mathbf{J}_i\omega_i) + \mathbf{l}_{i+1} \times m_{i+1}\dot{\mathbf{v}}^i_{i+1} + \mathbf{m}\rho_i \times \dot{\mathbf{v}}_i \\
&= \mathbf{J}_i\dot{\omega}_i + \omega_i \times (J_i\omega_i) + m_{i+1}\mathbf{l}_i \times \dot{\mathbf{v}}_{i+1} + \mathbf{m}\rho_i \times \dot{\mathbf{v}}_i \\
&= \mathbf{J}_i\dot{\omega}_i + \omega_i \times (J_i\omega_i) + m_{i+1}(\mathbf{l}_i \times (\dot{\mathbf{v}}_i + \dot{\omega}_i \times \mathbf{l}_i + \omega_i \times (\omega_i \times \mathbf{l}_i))) \\
&= [J_i + m_{i+1}(\mathbf{l}_i.\mathbf{l}_i)\mathbf{I} - \mathbf{l}_i\mathbf{l}_i^T]\dot{\omega}_i + \\
&\quad \omega_i \times [J_i + m_{i+1}(\mathbf{l}_i.\mathbf{l}_i)\mathbf{I} - \mathbf{l}_i\mathbf{l}_i^T]\omega_i \\
&\quad + (\mathbf{m}\rho_i + m_{i+1}\mathbf{l}_i) \times \dot{\mathbf{v}}_i
\end{aligned}$$

Q.E.D

Hereafter, the first and second moments shall implicitly mean the augmented moments; since there is no method to isolate the contribution due to the masses alone.

Theorem 2

The base parameters of the j^{th} link of a rotational manipulator can be obtained from the basis function coefficients of the following expressions:

- C_s, $s = 1 \ldots N$ where C_s is the coefficient of $\ddot{q}_s$ in the following expressions:

$$\sum_{x=j+1}^{N}\sum_{k=1}^{x}[\hat{\mathbf{e}}.[\mathbf{l}_j \times [-\mathbf{m}\rho^j_x] + \mathbf{J}^j_x]\mathbf{e}^j_k]\ddot{q}_k + \sum_{k=1}^{j}\hat{\mathbf{e}}.\mathbf{J}^j_j\hat{\mathbf{e}}^j_k\ddot{q}_k +$$

$$\sum_{x=j}^{N}\sum_{y=1}^{x-1}\sum_{z=1}^{y}\hat{\mathbf{e}}.A^j_x[\mathbf{m}\rho_x \times [\dot{\omega}^j_y \times \mathbf{l}^j_y]]\ddot{q}_z$$

- $\hat{e}.\sum_{i_1=j}^{N} \mathrm{m}\rho_{i_1}^{j} \times \mathrm{g}_j$

where $\dot{\mathrm{v}}_{i_1}^{i_2}$ is the representation of v_{i_2} in frame i_1

Proof: By equation (3.8), the generalized torque at the j^{th} joint axis is given by :

$$\begin{aligned}
\Gamma_j &= A_{j-1}^{j}\left[-\mathrm{n}_{j-1}\right].\hat{\mathbf{e}} \\
&= \left[-\mathrm{n}_j - \mathrm{l}_j \times \mathrm{f}_j + \mathrm{m}\rho_j \times \dot{\mathrm{v}}_j + \mathbf{J}_j^j\dot{\omega}_j + \omega_j \times \mathbf{J}_j^j\omega_j\right].\hat{\mathbf{e}} \qquad (3.9) \\
&= \left[-\mathrm{n}_j - \mathrm{l}_j \times \left(\dot{\omega}_{j+1}^{j} \times \mathrm{m}\rho_{j+1}^{j}\right) + \mathrm{l}_j \times \left(\omega_{j+1}^{j} \times \left(\omega_{j+1}^{j} \times \mathrm{m}\rho_{j+1}^{j}\right)\right)\right. \\
&\quad \left. -\mathrm{l}_j \times \mathrm{f}_{j+1}^{j} + \mathrm{m}\rho_j \times \dot{\mathrm{v}}_j + \mathbf{J}_j\dot{\omega}_j + \omega_j \times \mathbf{J}_j\omega_j\right].\hat{\mathbf{e}} \qquad (3.10)
\end{aligned}$$

By Renaud [2], the velocity terms are obtained from the acceleration coefficients by employing the Christoffel symbols; hence the base paramters can be evaluated from the coefficients of the joint accelerations. Hence we isolate the acceleration torques.

$$\begin{aligned}
\Gamma_j^a &= \sum_{x=j+1}^{N} \hat{\mathbf{e}}.[\mathrm{l}_j\times][-\mathrm{m}\rho_x^j\times]\sum_{k=1}^{x}\hat{\mathbf{e}}_k^j\ddot{q}_k + \\
&\quad \sum_{x=j}^{N}\hat{\mathbf{e}}.A_x^j[\mathrm{m}\rho_x \times [\sum_{y=1}^{x-1}[\sum_{y=1}^{x-1}(\dot{\omega}_y^j \times \mathrm{l}_y^j)]] + \sum_{x=j}^{N}\hat{\mathbf{e}}.\mathbf{J}_x^j\sum_{k=1}^{x}\hat{\mathbf{e}}_k^j\ddot{q}_k \qquad (3.11) \\
&= \sum_{x=j+1}^{N}\sum_{k=1}^{x}\hat{\mathbf{e}}.[[\mathrm{l}_j\times][-\mathrm{m}\rho_x^j\times] + \mathbf{J}_x^j]\hat{\mathbf{e}}_k^j\ddot{q}_k + \hat{\mathbf{e}}.\mathbf{J}_j^j\sum_{k=1}^{j}\hat{\mathbf{e}}_k^j\ddot{q}_k + \\
&\quad \sum_{x=j}^{N}\sum_{y=1}^{x-1}\sum_{z=1}^{y}\hat{\mathbf{e}}.[\mathrm{m}\rho_x^j \times [\hat{\mathbf{e}}_z^j \times \mathrm{l}_y^j]]\ddot{q}_z \qquad (3.12)
\end{aligned}$$

Q.E.D

3.3.2 Numerical procedure

A numerical procedure can be used to identify the basis columns of the regressor. Considering n random values of position, velocity and acceleration; the regressor matrix can be calculated numerically using the Newton-Euler procedure, each coefficient being obtained by considering the torque contribution due to a unit parameter.

The regressor columns for the redundant parameters have null values; the dependancies between the remaining columns can be identified by the rank of the augmented matrix, $[H_{1\ldots k-1} : H_k]$. For a rank deficient matrix, the last column can be obtained as a linear combination of previous columns by an orthogonal basis B_k. Let

$$h_i = \sum_{j=1}^{k-1} \beta_{kj} c_j$$

$$\beta_k = A^{-1}\alpha_k \quad \alpha_{ij} = \langle h_i, \hat{Bk}_j \rangle$$

3.4 Conclusion

In this chapter, we presented the inverse dynamics algorithm and presented two new theorems for the identification of the *base parametric set*; the second theorem delineated two kinematic expressions whose base parameters constitute the base parameters of the dynamic model. The basis functions are only functions of the manipulator architecture and can be evaluated given the constant DH parameters. The advantage of employing the N-E equations for base parametric evaluation is that the reason for the dependancies is more apparent.

Chapter 4

Least Squares Estimation

4.1 Introduction

In this chapter, we present least squares identification procedures for the identification of the base parameters evaluated in chapter 3. All physical systems are noisy; in the case of manipulators the noise constribution also includes the unmodelled dynamics which lead to a systematic error in estimation. Section 2 describes the assumed models for the regressor and motion (process) noise. In section 3, the least squares procedures are described and perturbation bounds for the solution are given in terms of available error norms. The section also describes practical methods for improving the accuracy of the identification results.

Finally, in section 4, we state the procedure delineated in [23] for obtaining the manipulator state by optimal filtering techniques. This procedure was employed in the analysis as it presented more accurate results.

4.2 The noise models for the estimation problem

If position and motion data are available; the least squares estimation procedure can be biased on account of:

1. Sensor noise: this occurs mainly due to the noisy acceleration data which is obtained by twice differentiating the position vector. Sensor noise is modelled as a normally distributed random vector, with a zero mean for a calibrated system.

2. Unbiased Motion noise: this is the additive noise in the sensors connected to the actuators, e.g. the current input to the dc-motors controlling the joints. The motion noise is modelled as Gaussian white noise, uncorrelated with the regressor columns.

3. Unmodelled dynamics: the unmodelled dynamics are correlated with the regressor terms and are mainly responsible for bias in the estimation results.

Armstrong [14] presents models for the noise components as follows:

1. Sensor noise (Random)	$H_k = \hat{H}_k + \delta H_k$	$\delta H_k \epsilon N(\phi, C_{\overline{\phi}})$
2. Motion noise (Random)	$\tau_k = H_k' \theta^* + \overline{\tau}$	$\overline{\tau} \epsilon N(\overline{0}, \sigma_{\overline{\tau}}^2)$
3. Unmodelled	$\tau_k = H_k' \theta^* + \nu$	$\nu \epsilon L_\infty$

The uncorrelated motion noise does not bias the estimates; however, the systematic error and regressor noise due to the sensor noise are mainly responsible for parameter bias.

For a manipulator with revolute joints, the dynamic model can be represented as:

$$T = M(\Theta)\ddot{\Theta} + M(\Theta)\delta\ddot{\Theta} + V(\Theta,\dot{\Theta}) + F(\dot{\Theta}) + G(\Theta) + T_M + T_U$$

where

$$\delta\ddot{\Theta} = \ddot{\Theta}_{comp} - \ddot{\Theta}$$

represents the error in the calculated acceleration; the regressor noise may be assumed to consist only of the acceleration error. T_U represents the unmodelled dynamics and T_M denotes the motion noise. Augmenting the parameter vector P by the static and viscous friction terms, the model for estimation can be represented as:

$$T = \hat{M}(\Theta,\ddot{\Theta})P + \hat{M}(\Theta,\delta\ddot{\Theta})P + \hat{V}(\Theta,\dot{\Theta})P + \hat{F}(\dot{\Theta})P + \hat{G}(\Theta) + T_U + T_M$$

where $\hat{M}, \hat{V}, \hat{F}, \hat{G}$ are the regressors for the inertial, centrifugal and Coriolis, frictional and gravitational terms respectively. Let

$$\delta T \triangleq -T_U - T_M$$

$$H \triangleq \hat{M}(\Theta,\ddot{\Theta}) + \hat{V}(\Theta,\dot{\Theta}) + \hat{F}(\dot{\Theta}) + \hat{G}(\Theta)$$

$$\delta H \triangleq \hat{M}(\Theta,\delta\ddot{\Theta})$$

The estimation problem can be denoted as,

$$(H + \delta H)(P + \delta P) \cong T + \delta T$$

As mentioned earlier, the uncorrelated motion noise T_M does not contribute to estimation bias, however the regressor error and unmodelled dynamics contribute to parameter bias.

4.3 Improving the accuracy of the least squares problem

The least squares method used to estimate the base parametric set B_{min} from the singular value decomposition of the regressor $H + \delta H$ is:

$$B_{min} = V_{H+\delta H} S^{-1}_{H+\delta H} U^T_{H+\delta H} (T + \delta T)$$

A bound for the resulting parameter bias is given by Lawson, [9]

$$\| \delta B \| \leq \frac{\| S^{-1}_{H+\delta H} \| \left(\| \delta H \| \left(\| B^* \| + \| S^{-1}_{H+\delta H} \| \| T + \delta T - (H + \delta H) B_{min} \| \right) + \| \delta T \| \right)}{1 - \| \delta H S^{-1}_{H+\delta H} \|}$$

In terms of the matrix condition, $\kappa = \frac{\bar{\sigma}_{H+\delta H}}{\underline{\sigma}_{H+\delta H}}$; the inequality can be represented as:

$$\| \delta B \| \leq \frac{\kappa [\| \delta H \| \left(\| B^* \| + \| S^{-1}_{H+\delta H} \| \| T + \delta T - (H + \delta H) B_{min} \| \right) + \| \delta T \|]}{\| H + \delta H \| (1 - \| \delta H \| \| S^{-1} \|)} \tag{4.1}$$

From equation (4.1), the error can be reduced by minimising the condition κ. As the condition is a function of the regressor; an exciting trajectory, i.e. a trajectory that generates a regressor with a reduced condition reduces the parameter bias.

We shall apply the methods of ridge regression and significance analysis for improving the accuracy of the estimates.

1. **Ridge regression:** the procedure for ridge regression is described in [9]; it is applicable when the *a priori* covariance matrix, C_T, of the uncertainty in the data vector T and the *a priori* covariance C_{P^a} of the uncertainty in the *a priori* expected value of the parametric vector P^a are available. The LS problem is then represented [9] by the following equation:

$$\begin{bmatrix} C_T & \phi \\ \phi & C_{P^a} \end{bmatrix} \begin{bmatrix} H \\ I \end{bmatrix} P \cong \begin{bmatrix} C_T & \phi \\ \phi & C_{P^a} \end{bmatrix} \begin{bmatrix} T \\ P^a \end{bmatrix}$$

The procedure provides a combination of the weighted least squares and forced bias towards the expected value, P^a.

P^a and C^a are generally estimated based on available data on estimation conducted over previous trajectories; a measure of the covariance for the LS solution for a given $m \times n$ regressor matrix H_{traj} is given as:

$$\sigma^2 (H^T H)^{-1}$$

where

$$\sigma^2 = \frac{\| H_{traj} P - T_{traj} \|^2}{m - n}$$

In practice, the actual covariance is not available; a variable λ is introduced in the procedure as:

$$\begin{bmatrix} C_T & \phi \\ \phi & \lambda C_{P^a} \end{bmatrix} \begin{bmatrix} H \\ I \end{bmatrix} P \cong \begin{bmatrix} C_T & \phi \\ \phi & \lambda C_{P^a} \end{bmatrix} \begin{bmatrix} T \\ P^a \end{bmatrix} \qquad (4.2)$$

and the solution vector is examined for different choices of λ. A higher value of λ signifies a higher confidence in previous estimates.

2. **Column scaling:** Column scaling of the regressor is added to the identification procedure to improve the problem condition. Equation(4.1) gives an upper bound to the norm of the solution error; this quantity is not a useful indicator for evaluating individual parametric errors since the parameters influence generalised forces by different orders of magnitude. Schroer [24] suggests column scaling the Jacobian as a means of improving the condition for kinematic calibration.

   We will consider scaling the individual columns of the regressor matrix by the *experimental torque sensitivities* $[\frac{d}{dB_i} \mid T \mid]^{-1}$ which can be obtained to a first order of magnitude by the expression

$$\frac{(H + \delta H_c)^T T_c}{\sqrt{T_c^T T_c}}$$

   This procedure also improves the matrix condition and enables the error bound in equation (4.1) to be evaluated more effectively.

3. **Significance analysis:** In an experimental setup; where only a reduced order model is available; the effect of some inertial parameters may be smaller than the unmodelled dynamics; these parameters are insignificant for most purposes (e.g. for model based control). A significance analysis involves the deletion of insignificant parameters; this is equivalent to deleting the relevant columns in the regressor.The possible methods of significance analysis include:

   (a) **Sensitivity analysis:** Columns with low values of parametric sensitivities in the unscaled regressor matrix may be deleted; the

condition of the regressor decreases as a result of column deletion.

(b) **Eliminating near dependancies:** Considering the orthogonal decomposition, USV^T of the scaled regressor matrix; the regressor columns that are nearly dependant are determined from the columns of matrix $V_{H+\delta H}$ corresponding to the small singular values of the regressor. From [24]:

$$| (H + \delta H) v^i_{H+\delta H} | = s^i$$

the lower singular values constitute an almost dependant set of regressor columns.

Entries with largest absolute value in the vector v_i are hence considered nearly redundant and eliminated as they are most likely to be in error.

(c) **Eliminating the direction of weakest estimation:** The column v_i corresponding to the smallest singular value is the direction in which parameter estimation is most suceptible to an error due to the unmodelled dynamics. The results obtained by discarding the components in this direction are found to be closer to the actual parametric values. This is demonstrated for the Puma560 arm in chapter 7.

4.4 Considerations in the experimental procedure and analysis of the data

On account of sensor noise, (the joint angle data for the Puma560 was recorded using optical encoders), the backward difference procedure for obtaining the velocities and acceleration signals results in a large noise contribution, particularly so due to the small sampling interval i.e. 5 ms.

The problem of optimal filtering for estimating the manipulator state using position data has been considered in [23]. Since, under suitable assumptions, the knowledge that the output lies between two known levels (as is the case for any manipulator joint data) is equivalent to observing the output corrupted with Gaussian white noise, the state estimation problem is approximately equivalent to the Kalman filter.

The discrete signal model is given as:

$$x(k+1) = \begin{bmatrix} 1 & T_s & \frac{1}{2}T^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} \frac{1}{6}T_s^3 \\ \frac{1}{2}T_s^2 \\ T_s \end{bmatrix} \frac{d}{dt}\ddot{\theta}_{model}(k) + w(k) \qquad (4.3)$$

$$y(k) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} x(k) + e(k) \qquad (4.4)$$

with noise statistics:

$$Q = E(ww^T) = q \begin{bmatrix} \frac{1}{20}T_s^5 & \frac{1}{8}T_s^4 & \frac{1}{6}T_s^3 \\ \frac{1}{8}T_s^4 & \frac{1}{3}T_s^3 & \frac{1}{2}T_s^2 \\ \frac{1}{6}T_s^3 & \frac{1}{2}T_s^2 & T_s \end{bmatrix}$$

$$R = E(ee^T)$$

where w is the acceleration error and the signal $\frac{d}{dt}\ddot{\theta}_{model}$ represents the model acceleration rate. For an improvement with $\ddot{\theta}_{model}$, the noise intensity , $E(ww^T)$ must be less than that of the model acceleration. In the case of data logged using the available instrumentation, q was found to be of the order of 10^3 whereas the model acceleration rate was in the range $20 - 30$ in MKS units, so we chose to neglect this term.

As indicated in [23], the results are relatively insensitive to the choice of q. It is reasonable to use the square of the maximum slope of the acceleration signal.

Optimal filtering [23] for expected values based on the position data output has been shown to be relatively unaffected by inaccurate noise statistics. The system modelled by a double integrator driven by white noise, w, was shown to be a valid assumption for small values of the sampling time interval, T_s, [23].

The filter equations are as follows:

$$\hat{x}(k+1 \mid k) = A_d(I - KC)\hat{x}(k \mid k-1) + A_d K y(k) \tag{4.5}$$

where K is the Kalman gain.

$$K = PC^T R^{-1} \tag{4.6}$$

P can be obtained by solving the discrete Riccati equation:

$$P = (I + A_d T_s)(P - PC^T(CPC^T + R)^{-1}CP)(I + A_d T_s)^T + q\Gamma\Gamma^T T_s \tag{4.7}$$

In the rest of the chapter, the acceleration is calculated using the above filter since the final results were found to be better than employing any other

method employed for estimating the acceleration (low-pass filtering etc.). This is because the differentiated acceleration signal is adequately modelled as a Gaussian white noise process. The only way to improve the estimate would be to establish time varying statistics for the input noise based on actuator capabilities and the instantaneous manipulator pose; however, this was not considered at present.

4.5 Conclusions

In this chapter, we described noise models for the dynamic estimation problem that have been characteristically used for error analysis. We also present two methods of increasing the accuracy of the dynamic estimation problem. This is particularly important for an experiment where instrumentation with a limited accuracy is available.

Chapter 5

The LMS Method and Input Optimisation

5.1 Introduction

The convergence properties of the LMS adaptive filter have been studied by [13],[15],[1]. Based on assumptions regarding the motion and sensor noise and using the assumption of slow adaptation, Armstrong [14] derived expressions for the steady state bias in the estimated parameter vector and an approximate measure for the rate of convergence. The rate of convergence is an important factor in the stability of model based adaptive control schemes. Minimising the bias and maximising the rate of convergence were shown to be based on minimising the norm of the inverse of the input correlation matrix of the regressor. Accordingly, an optimal trajectory for identification purposes maximises the minimum singular value of the input correlation matrix

subject to maximum power and joint angle contraints. A multistage optimising algorithm, formulated by Bryson and Ho [16] has been implemented for trajectory optimisation of the Adept1 and MIT/Asada Direct Drive Arm by Armstrong [14].

In this chapter, the multistage optimisation technique [16] is applied to trajectory optimisation of the identification trajectory of the PUMA560 arm using the method employed by Armstrong. A method of choosing the initial trajectory is also formulated. Section 2 briefly states the results of error analysis in [14]. Section 3 describes the input optimisation algorithm for generating an exciting trajectory. Finally, section 4 describes the implementation for the 3 dof Puma manipulator and suggests methods for choosing an initial trajectory while obeying the manipulator contraints.

5.2 Convergence properties and input optimisation for the LMS algorithm

For the LMS parameter update law, [1],

$$\hat{P}_{k+1} = \hat{P}_k + \sum_n \Gamma_n H_k(\tau_k^n - H_k^n P_k) \tag{5.1}$$

the expectation of the parameter bias due to motion, sensor noise and unmodelled dynamics is given by the expression [14]:

$$E(\tilde{\theta}) = (R + C_{\bar{H}_k})^{-1} C_{\bar{\phi}} P^* - (R + C_{\bar{H}_k})^{-1} \sum_{k=0}^{K} H_k \nu_k \tag{5.2}$$

where

$$R = \frac{1}{K} \sum_{k=0}^{K} H_k^* H_k^{*\prime}$$

and

$$C_{\tilde{H}_k} = \frac{1}{K} E(\sum_{k=0}^{K} \tilde{H}_k \tilde{H}_k{}')$$

are the regressor correlation matrix and the sensor noise correlation matrix respectively.

The acceleration signal obtained by backward differentiation is very noisy; the high frequency components of the noise must be filtered before using the data in the calculations. Denoting the filtered acceleration signal by $\ddot{\Theta}_f$ and the actual acceleration as $\ddot{\Theta}_a$

$$C_{\tilde{H}_k} = \frac{1}{K} \sum_{k} \hat{M}_k(\Theta, \ddot{\Theta}_a - \ddot{\Theta}_f) \hat{M}_k(\Theta, \ddot{\Theta}_a - \ddot{\Theta}_f) \tag{5.3}$$

From expression (5.2), the expected parameter bias can be minimised by minimising the norm of the correlation matrix,

$$\| R + C_{\tilde{H}} \|^{-1} = \frac{1}{\underline{\sigma}(R + C_{\tilde{H}})^{-1}} \tag{5.4}$$

The evolution of the estimation error is given by the transition equation [14]:

$$\tilde{P}_{k+1} = (I - \Gamma \hat{H}_k \hat{H}_k') \tilde{P}_k - \Gamma \hat{H}_k (\tilde{H}_k' + \tilde{\tau}_k + \nu_k) \tag{5.5}$$

Assuming slow adaptation ($\overline{\sigma}(K\Gamma R) < 1$), the error can be shown to converge to zero with a maximum exponential time constant given by [14]

$$r_\tau^{\max} = \frac{T_s}{\underline{\sigma}(\Gamma R)} \tag{5.6}$$

Maximising the minimum singular value of the input correlation matrix therefore minimises the parameter bias and maximises the rate of convergence.

A sweep algorithm to optimize the identification trajectory is described by Bryson and Ho [16] and applied by Armstrong [14] to optimise the identification of the inertial parameters of the Adept1 and the MIT/Asada Direct Drive arm. The optimisation algorithm is described in the next section.
Given an optimizing trajectory $\ddot{\theta}_k^* \mid_{k=1...K}$, equations 5.1 and 5.2 can be used to estimate the parameter bias, given bounds on the unmodelled dynamics. The trajectory tracking error perturbs the singular values σ_i^*; the resulting perturbations in the singular values can be calculated using the following expression [9]:

$$\mid \sigma_i - \sigma_i^* \mid \leq \parallel \frac{1}{K} \sum_{k=1}^{K} \hat{M}(\theta_k, \ddot{\theta}_k^* - \ddot{\theta}_f)' \hat{M}(\theta_k, \ddot{\theta}_k^* - \ddot{\theta}_f) \parallel \qquad (5.7)$$

Since the trajectory tracking using conventional controllers for generating the demand trajectory is not very accurate, expression (5.7) provides a measure of the robustness of the input optimisation experiment to experimental errors.

5.3 Input Optimization

This section describes a multistage algorithm for trajectory optimisation for the LMS identification procedure developed by Bryson and Ho [16] and applied by Armstrong [14].

We define the following quantities:

$$\overline{X}_k \doteq [\overline{\theta}_k, \dot{\overline{\theta}}_k]^T$$

$$\overline{u}_k \doteq \ddot{\theta}_k^T$$

The optimization algorithm involves a gradient search for minimising the cost function $1/\underline{\sigma}(R)$, subject to the constraints relating the position and velocity to the acceleration.

$$f(\overline{x}_{k+1}, \overline{x}_k, \overline{u}_k) : \overline{x}_{k+1} - \begin{bmatrix} I & IT_s \\ \Phi & I \end{bmatrix} \overline{x}_k - \begin{bmatrix} \Phi \\ IT_s \end{bmatrix} \overline{u}_k = \underline{0}$$

The problem can be stated as:

$$\min_{\ddot{\theta}_k} \frac{1}{\underline{\sigma}(\frac{1}{K}\sum_{n,k} H_k^n H_k^{nT})} + \sum_k \ddot{\overline{\theta}}_k^T Q \ddot{\overline{\theta}}_k \doteq \min_{\ddot{\overline{\theta}}_k} F(\overline{x}, \overline{u}) \tag{5.8}$$

$$\varepsilon f(\overline{x}_{k+1}, \overline{x}_k, \overline{u}_k) = 0 \quad \forall k$$

The second term in the cost function limits the acceleration to within actuator capabilities. The relation between the maximum acceleration and the maximum torque depends on the state of the manipulator; hence Q must ideally be chosen as a function of k and $\overline{x}_k$; however, in practice, Q is chosen to be a constant diagonal matrix according to Brysons rules as incorporated by Armstrong [14]:

$$\frac{\| Q \|}{\max \ddot{\theta}^2} \approx \frac{| \frac{d}{d\ddot{q}}(1/\underline{\sigma}(R)) |}{\max \frac{1}{\underline{\sigma}(R)}^2} \tag{5.9}$$

Defining

$$J \triangleq F(\underline{x}, \underline{u}) + \sum_k \lambda_{k+1}^T f(\underline{x}_{k+1}, \underline{x}_k, \underline{u}_k)$$

the local optimum is given by the expressions:

$$\nabla_{\overline{x}_k} J = \nabla_{\overline{x}_k} F(\overline{x}, \overline{u}) + \overline{\lambda}_k^T - \overline{\lambda}_{k+1}^T A = \overline{0} \tag{5.10}$$

$$\nabla_{\overline{u}_k} J = \nabla_{\overline{u}_k} F(\overline{x}, \overline{u}) + 2\overline{u}^T Q - \overline{\lambda}_{k+1}^T B = \overline{0} \tag{5.11}$$

$$\nabla_{\overline{\lambda}_k} J = f(\overline{x}_{k+1}, \overline{x}_k, \overline{u}_k) = \overline{0} \quad \forall k \tag{5.12}$$

The terms in the partial derivatives $\nabla_{\bar{x}_k} J$ and $\nabla_{\bar{u}_k} J$ can be computed by the chain rule

$$\frac{\partial J}{\partial x_k^b} = \sum_i \sum_j \frac{\partial J}{\partial R_{ij}} \frac{\partial R_{ij}}{\partial x_k^b}$$

$$\frac{\partial J}{\partial u_k^b} = \sum_i \sum_j \frac{\partial J}{\partial R_{ij}} \frac{\partial R_{ij}}{\partial u_k^b}$$

Denoting

$$\alpha_{ij} \doteq \frac{\partial J}{\partial R_{ij}}$$

$$\frac{\partial J}{\partial x_k^b} = \sum_{i,j} \alpha_{ij} \frac{\partial}{\partial x_k^b} (\frac{1}{K} \sum_n H_k^n H_k^{nT})_{ij}$$

$$= \sum_{i,j} \alpha_{ij} \frac{1}{K} \sum_n (\frac{\partial}{\partial x_k^b} H_k^n H_k^{nT} + H_k^n \frac{\partial}{\partial x_k^b} H_k^{nT})_{ij}$$

$$\frac{\partial J}{\partial u_k^b} = \sum_{i,j} \alpha_{ij} \frac{1}{K} \sum_n (\frac{\partial}{\partial u_k^b} H_k^n H_k^{nT} + H_k^n \frac{\partial}{\partial u_k^b} H_k^{nT})_{ij}$$

The trajectory update equation is given as:

$$\bar{\bar{u}}_k^{m+1} = \bar{\bar{u}}_k^m - \mu_m \nabla_{u_k} J$$

where μ_m is the increment scaling value. The choice of μ_m depends on the magnitude $\| \nabla_{u_k} J \|$ and the maximum required resolution for $\ddot{u}_k$.

For implementation; μ_m was chosen on-line at the end of each iteration. The algorithm converges to a local minimum; hence the choice of the initial trajectory is important.

Since

$$\underline{\sigma}(R) \leq | \frac{1}{K} \sum_n H_k^n H_k^{n\prime} \hat{e}_m | = | \frac{1}{K} \sum_n H_k^n \tau_{\hat{p}_m} |$$

a heuristic principle may be followed to obtain an initial trajectory based on the argument that the minimum singular value is limited by the above norm.

Let $x \triangleq \min \underline{\sigma}(R)$. This value is based on the magnitude of the unmodelled dynamics, $\mid \nu_k \mid$ and on the maximum allowable error in the estimated parameters.

We choose points $\overline{x}_i, \overline{u}_i$ such that the power and joint angle constraints are satisfied and

$$\mid \sum_n H_k^n \tau_{\dot{P}_m} \mid > x \quad \forall m$$

The initial trajectory $\ddot{\theta}_k$ is generated by interpolation between values $\overline{x}_i, \overline{u}_i$ by the system equations

$$\overline{x}_{k+1} = A\overline{x}_k + BK(\overline{x}_i - \overline{x}_k)$$

and filtering the resulting trajectory.

5.4 Algorithm Implementation

The input optimisation algorithm was implemented for the PUMA 560 manipulator arm. Implementation was done in Matlab. To avoid instabilities in the computations, the partial derivatives were evaluated by explicit differentiation of the regressor terms using Maple. Matlab code for the optimisation algorithm is included in Appendix C.

5.4.1 Choice of μ_m

The choice of the incremental step for updating the trajectory must be based on the current magnitude of $dJ/d\overline{u}$. It is quite likely that a conservative choice of μ_m delays convergence whereas an overestimated value would take

the trajectory outside the current region of convergence. Since the average magnitude of the gradient decays with an increasing number of iterations, μ_m normally increases with each iteration. The algorithm is likely to converge rapidly using this approach.

5.4.2 Meeting the constraints

Since a manipulator typically has a constrained workspace, meeting the joint *and* the actuator constraint requirements is important for an identification experiment to be carried out. These constraints cannot be introduced *a priori*; however, the optimisation procedure may be terminated when these values are exceeded even before a local optimum is attained. The trajectory improvement may be significant even in this case.

5.4.3 Results

A frame assignment of the 3 dof PUMA 560 arm is shown in Figure 5.1. For the frame assignment as shown, the manipulator constraints are given in Table 5.1.

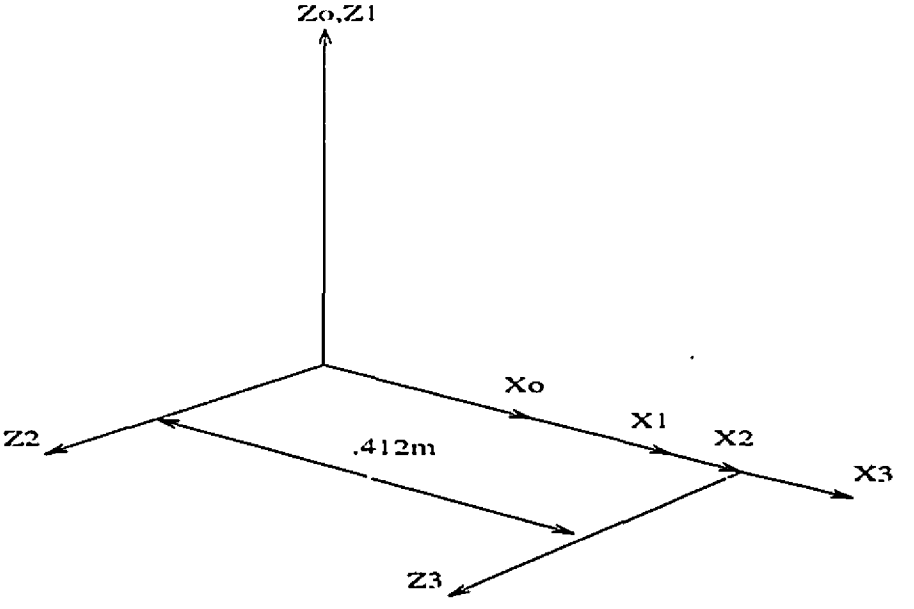

Figure 5.1: Frame assignment

Several instances of the initial trajectory were selected. In certain cases, the procedure caused a uniform scaling of the trajectories towards higher accelerations. Attempting optimization for all inertial parameters did not result in a significant improvement in J for even high acceleration trajectories; hence optimisation was conducted for the reduced regressor consisting of parameters with a high contribution to the torque; viz. $P_2, P_3, P_9, P_{10}, P_{11}, P_{13}, P_{14}, P_{15}$. The optimised trajectory, shown in Figure 5.2, was obtained in 15 iterations.

The improvement in the cost J was from an initial value of 0.0278 to a final value of 0.0024; an improvement (error reduction) by a factor of 11.5833.

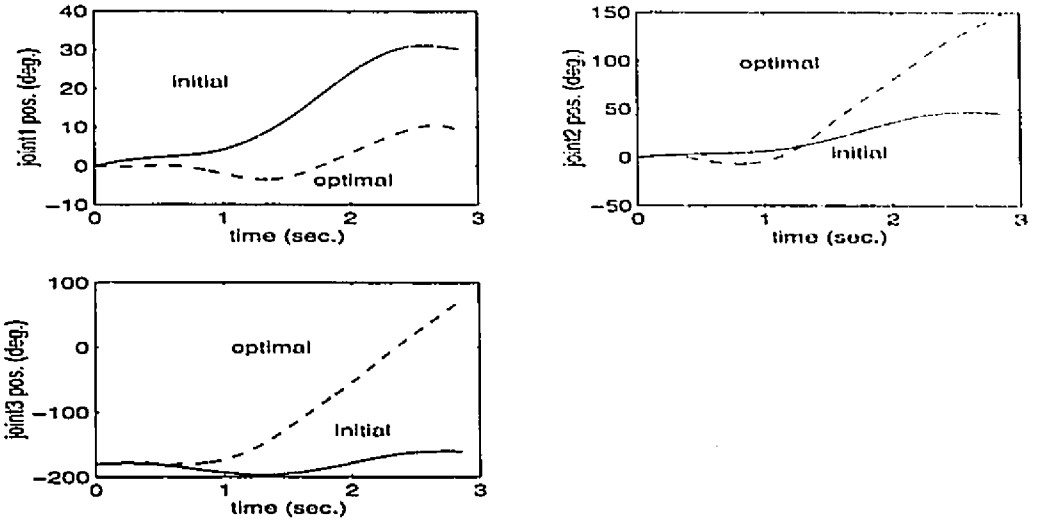

Figure 5.2: The Optimised trajectory for the 3 DOF Puma 560 arm

The final step in identification involves accurate tracking for the input trajectory; ordinary PID control may not be sufficiently accurate; for most optimal trajectories, the acceleration bandwidth of the resulting trajectory using PID control would be unacceptable for maintaining the required accuracy of the identification experiment. Feedforward or the computed torque procedure provide more accurate tracking; however, these control methods are based on the availability of the dynamic model; *a priori* values for the dynamics may be used in this case.

5.5 Conclusion

From the implementation procedure, we conclude that generating a trajectory that would be sufficiently exciting for all the modes of a given manipulator may easily exceed the capabilities of the manipulator; however, a significant reduction in the theoretical error bounds was obtained by conducting the optimisation procedure. The complete results of the experimentation will be shown in Chapter 6.

Joint	Joint angle Min(deg)	Joint angle Max(deg)	Max. Joint Torque (Nm)
1	-160	160	131.44
2	-35	215	228.57
3	-225	45	113.86

Table 5.1: Joint Constraints for the 3 dof Puma manipulator

Chapter 6

Experimental Procedures

6.1 Introduction

The estimaton procedures outlined in chapters 3 and 4 were carried out on trajectory data for two manipulators; the PUMA 560 arm and the SARCOS GRLA (General Robotic Large Arm). The present chapter briefly explains the software developed and the procedures adopted for accurately tracking the identification trajectories. The manipulators under consideration are also described. Finally, the experimental results for the tracking experiments are plotted. Analysis of the data is presented in the next chapter.

6.2 Software implementations

This section describes the code written for meeting the objectives of accurate identification and its consequent application to model based control.

6.2.1 Numerical methods for dependancy evaluation and LS estimation

Obtaining a full-rank regressor, as shown before, requires identification of a base parametric set. The base parametric set is numerically evaluated by generating the rank-deficient regressor and then obtaining the dependancies by orthogonal decomposition. Implementation is done using the Matlab programming language.

File *Estimation.m* contains routines for the LS estimation of the parameters. Significance analysis can be conducted based on the experimental torque sensitivity or on the singular value decomposition of the column scaled regressor. Finally, estimation over a trajectory can be based on a QR decomposition or previous estimates can be incorporated using the computed covariance and the ridge regression method. Input optimisation for the LMS method has been described in the previous chapter. The optimisation code in the Matlab language *mltstg.m* is listed in Appendix C.

6.2.2 Generating the dynamic model

In order to implement a model based controller, efficient code for the inverse dynamics based on the identified parameters and demand variables viz. joint angles, joint rates and accelerations, is required. The efficiency of the computation is crucial to implementation. Khalil et al. [25] developed an efficient algorithm for the full order dynamic model of a serial manipulator based on the Newton Euler algorithm. The algorithm develops intermediate variables for saving computations. The method, for a manipulator with N degrees of

freedom involves a maximum of $(105n - 92)$ multiplications and $(94n - 86)$ additions; this number is substantially reduced for a reduced basis set.

The algorithm has been implemented in Common Lisp and includes routines for eliminating all redundant operations. The program generates C code for full order dynamic compensation or only gravitational compensation. Program listings are included in Appendix B.

6.3 Trajectory generation and data acquisition for the Puma 560 and the Sarcos GRLA

6.3.1 The Puma 560 arm

The Puma 560 manipulator, is a 6 degree of freedom manipulator with a serial architecture as shown in Figure 6.1. The joints are actuated by DC motors with high gear ratios (>50:1). Position measurement is based on incremental encoders. The torque measurements are based on armature current readings. The frame assignment is according to the modified DH notation and is shown in Figure 6.2. The modified DH parameters for the manipulators are given in Table 6.1.

Tracking trajectories for identification and the implementation of model based control was carried out using the robot controller environment KALI,

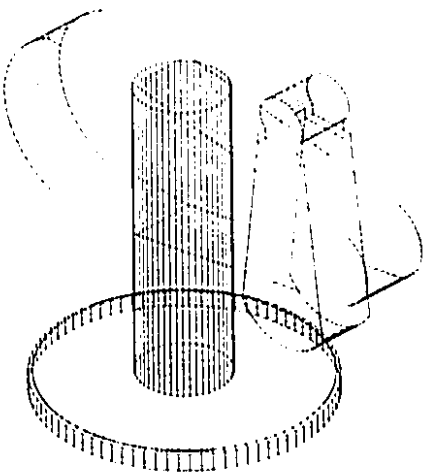

Figure 6.1: The Puma 560 arm

[26], implemented on a dual DSP processor board, the Challenger C30v. KALI provides velocity form pid control facilities allowing the user to modify the control gains in real time. Available techniques for selecting the PID gains are based on minimising a quadratic functional of the error evolution of the linearised system. The dynamic model for this purpose was generated using the estimated values over the batch LS estimates. Ordinary PID control gets unstable very fast beyond a range of trajectories. Moreover, acceleration and velocity tracking is not sufficiently accurate for identification over an optimal trajectory. For this reason, it is preferrable to choose optimal gains based on an approximate dynamic model. Methods for the same are delineated in [27].

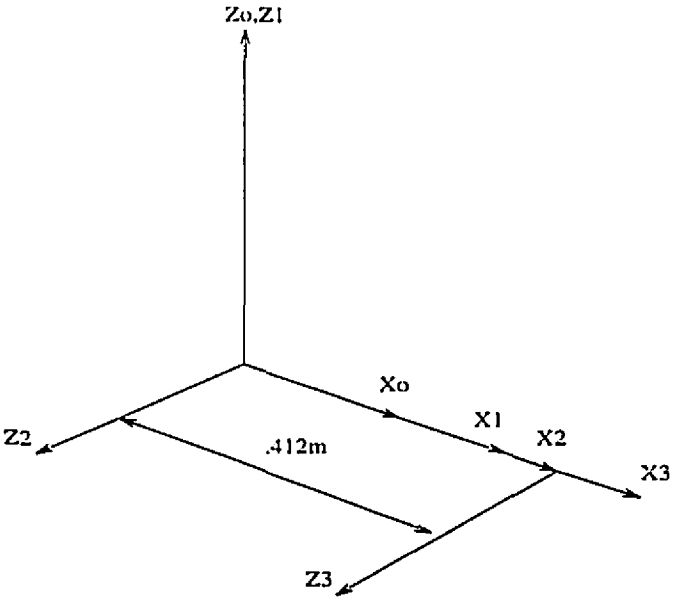

Figure 6.2: Frame assignment

6.3.2 The Sarcos GRLA

The Sarcos manipulator,shown in Figure 6.3 is a 7 DOF manipulator. The joints for the Sarcos are actuated by direct-drive hydraulic cylinders. Position measurement is based on LVDTs. Strain gauges are used for the torque measurements. The frame assignment and DH parameters are given in Figure 6.4 and Table 6.2 respectively. Estimation in this case was conducted for gravitational forces only, i.e. second moments of inertia were not identified. Experimental data were provided by the Institut de Recherche d'Hydro-Québec.

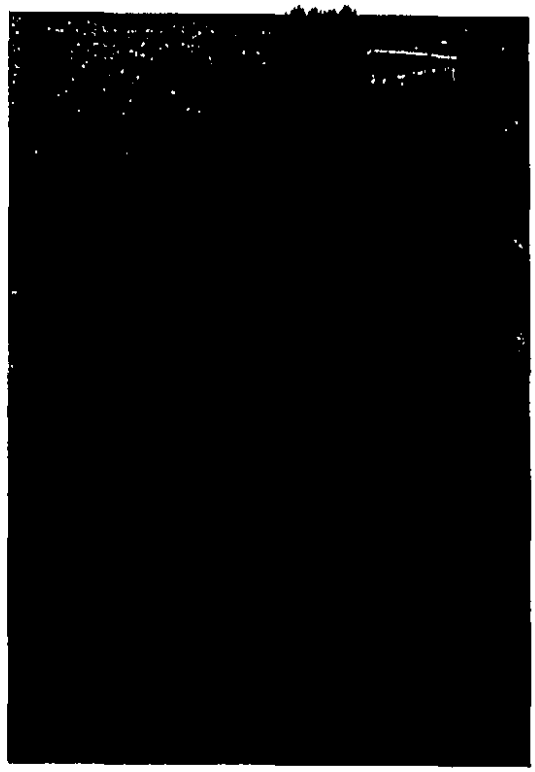

Figure 6.3: The SARCOS manipulator

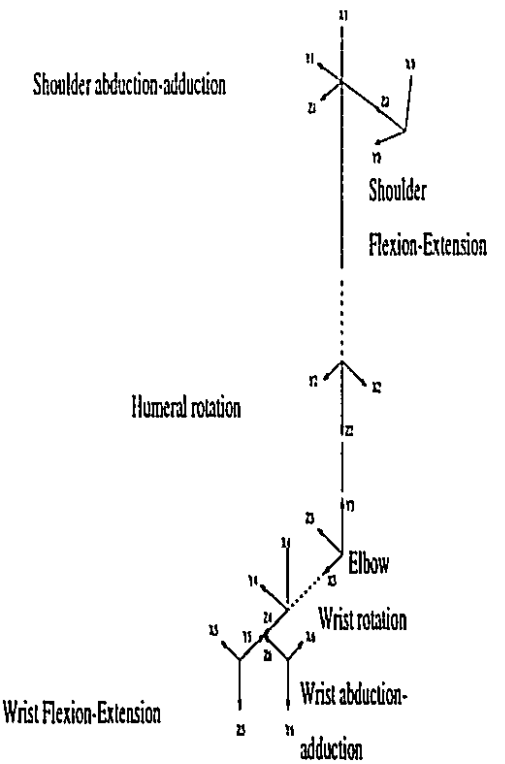

Figure 6.4: Frame assignment

Joint	α_i	d_i	b_i
1	90	0	0
2	0	0.432	0.149
3	-90	0.02	0

Table 6.1: DH parameters of the 3 dof Puma

Joint	α_i	d_i	b_i
1	90	0	0
2	90	0	0
3	-90	0	0.8788
4	90	0	0
5	-90	0	0.762
6	90	0	0
7	180	0	0

Table 6.2: DH parameters of the Sarcos GRLA

Chapter 7

Analysis of experimental data

7.1 Introduction

In this chapter, the least squares and LMS methods are applied to trajectory data obtained for identification trajectories for the PUMA560 arm and the Sarcos GRLA manipulators described in chapter 6. As indicated by Izaguirre et al., [5], and Atkeson et al., [4], a critical factor in the estimation accuracy is the acceleration noise which arises due to noisy position sensors. Results of the dynamic estimation procedure for the Puma 560 arm are given in section 2. Since estimation for the Puma was more involved than estimation for the Sarcos manipulator, we choose to group the results by the manipulators rather than the method. Finally, in section 3, we present the results of the estimation for the Sarcos manipulator and the performance of model based tracking using a dynamic model generated by the identified parameters. Since [1] tracking errors with computed torque and feedforward control [20]

increase with increasing inaccuracy of the identified parameters, the results of the model based control implementations serve as a good indicator of the accuracy of the identification experiment.

7.2 Dynamic estimation for the Puma 560 arm

7.2.1 Batch LS methods

Initially, estimation was conducted over ten identification trajectories. The cutoff frequency for the acceleration signal using a 4^{th} order Kalman filter was observed to be $\approx 1-2\ rad/s^2$ and large phase lags for higher frequencies; the frequency response for the velocity signal was much superior. Hence, the velocity and acceleration signals were obtained by filtering the calculated velocity by backward differentiation of the position signal. Filtering is therefore in two stages, firstly, for the position signal and next for the velocity and acceleration signals.

Although the dynamic parameters for two identical manipulators may differ; a standard is required for comparison, moreover, deviations are not very significant for the inertial parameters. A standard base parametric set for the present purpose can be obtained from the results in Khatib et al. [10]. It would be cumbersome to transform the quantities between the frame assignment we use and the one employed in [10]. It is preferrable to calculate the generalized torques for a series of trajectory points using the full order model in [10] and deriving the base parametric set using the regressors derived

using the current frame assignment (in Appendix A). This only requires a translation of joint position data for compatibility. This can be equivalently represented as :

$$H' P^{std} \approx T^{full-order}$$

Table 7.1 lists the parameters obtained using this method. In table 7.2, we show the effect on the condition of significance analysis based on the torque sensitivities, κ_1, and elimination of near dependancies based on the singular value decomposition of the scaled regressor, κ_2. The conditions are seen to be significantly reduced compared with the unscaled condition, κ_0.

Table 7.3 presents in three separate columns, the results of batch LS estimation over the ten identification trajectories. Columns 1 and 2 respectively show the results of batch LS and after succesive elimination in the two weakest directions of identification (ref. chapter 4). In the highlighted cases, when compared with the standard parameters, the influence on improving the accuracy of some identified parameters is significant. An improvement in the accuracy may be obtained if a minimum norm solution is sought using the ridge regression method provided accurate noise statistics are available; this solution can be located by plotting the norm of the residue against the solution norm with varying λ; [9]. However, regressor noise still causes large errors in certain parameters; for instance, the mismatch is especially pronounced for P_1.

7.2.2 The LMS method

As mentioned in chapter 4, input optimisation is practical for only a reduced set of significant parameters. The LMS identification trajectory generated in Chapter 4 was tracked by ordinary PD control; figure 7.1 shows the tracked trajectory; the results of the adaptive scheme are compared with the standard parameters in Table7.4. Convergence was not very satisfactory; by the error transition equation for the LMS algorithm given by Armstrong,

$$\tilde{\theta}_{(k+1)} = (I - \Gamma\hat{\phi}_k\hat{\phi}_k')\tilde{\theta}_k - \Gamma\hat{\phi}_k(\hat{\phi}_k'\theta^* + \tilde{\tau}_k + \nu) \qquad (7.1)$$

the maximal convergence rate is obtained along the eigenvector of the transition matrix corresponding to the smallest magnitude eigenvalue of the transition matrix, $(I - \Gamma\hat{\phi}_k\hat{\phi}_k')$. Accordingly, we increment the parameter at each iteration only along the direction of the most stable mode. The result of parameter evolution adopting this method over the identification trajectories is shown in Figures 7.2 and 7.3; the estimation results are recorded in Table 7.5.

Since the LMS algorithm was run over several trajectories, the discontinuities in the parameter evolution can be observed to be the result of a large percentage of the unmodelled dynamics due to stiction at the start of each trajectory. We could eliminate these points in the iterations; however, they have been retained as an indication of the parameter convergence.

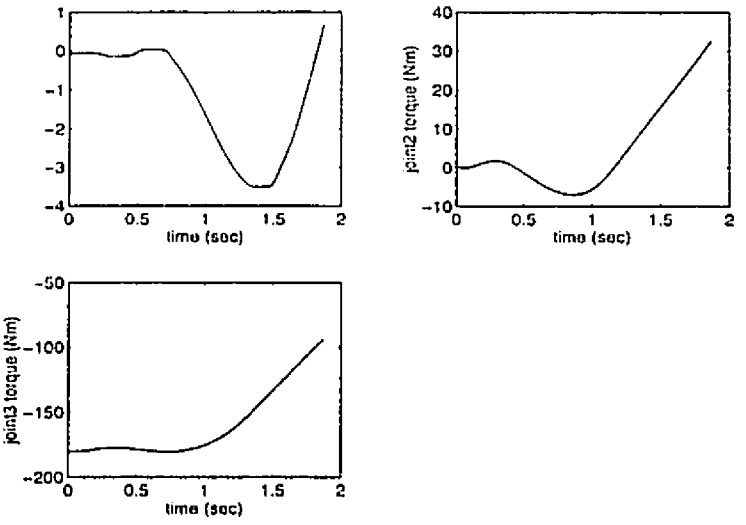

Figure 7.1: The Tracked Optimal Trajectory

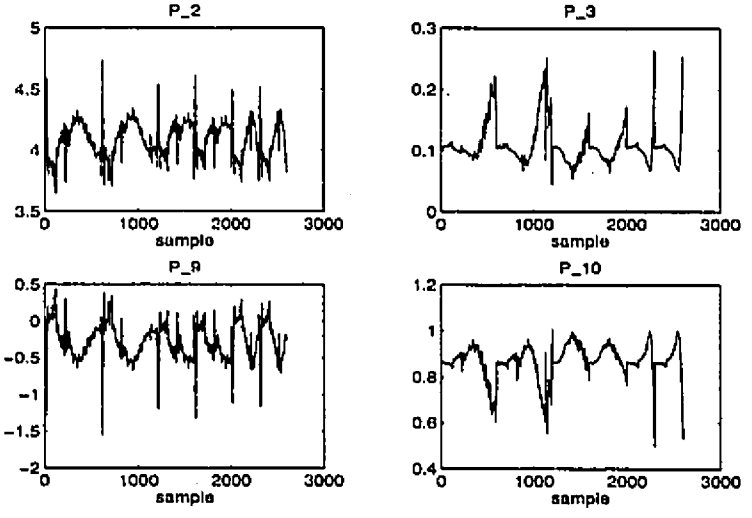

Figure 7.2: Parameter evolution

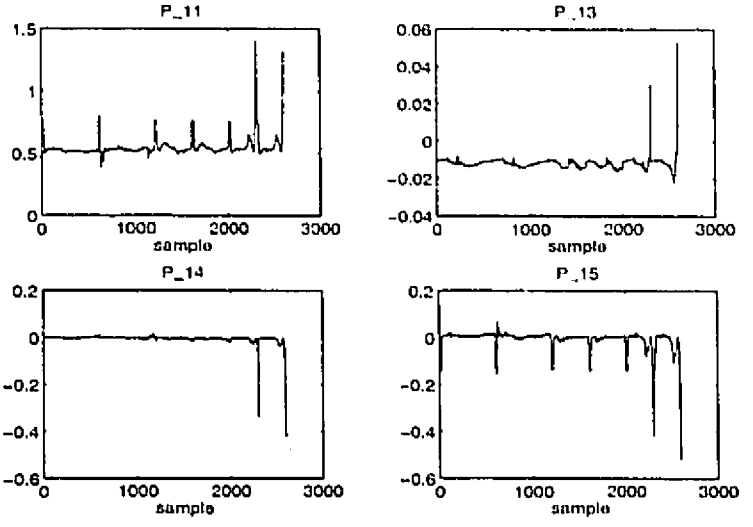

Figure 7.3: Parameter evolution

7.2.3 The model accuracy

In this section, we compare the recorded torques, the torques using Armstrongs parameters, the torques using the parameters obtained by elimination along the weakest directions and the abbreviated LMS model. The plots were obtained for eight different trajectories and are shown in figures 7.4 to 7.11. The X axis denotes the samples. The measured torque are indicated by solid lines, the dotted plots indicate the computed torques using LMS results along the convergent mode. The computed torques from the parametric set obtained using Khatib's full order model is used for the dashed plots. Finally, the solid bold plots indicate the LS results after elimination in the weakest direction of identification. The measured torque is characterized by a significant ripple not observed in the computed torques. The highest

deviation in the computed torque were observed when LMS results were employed. The LS results approached the measured values the closest. The bias using Khatib's parameters is especially pronounced for the third joint. For a mean square error criterion, the LS result is a better estimate compared to the explicit measured parameter set. However, their relative performance in a computed torque algorithm would depend on the trajectory employed.

7.3 Gravity compensation for the Sarcos manipulator

The kinematic model for the Sarcos manipulator was given in the last chapter. In this section, we consider estimation and feedforward control of the Sarcos manipulator employing gravitational compensation. Being a direct drive mechanism, the recorded torque for a given pose does not have a significant noise contribution. For this reason, dynamic estimation for this manipulator did not require incorporating practical methods for improving the accuracy of the estimation; a simple LS solution gives results with a very high accuracy.

7.3.1 Estimation results

The table of the inertial terms for gravity compensation and their estimated values from the data are given in Table 7.1. The standard deviations of the computed joint torque error are recorded in Table 7.2. Figure 7.1 compares the experimental torque values with the computed values. The results are

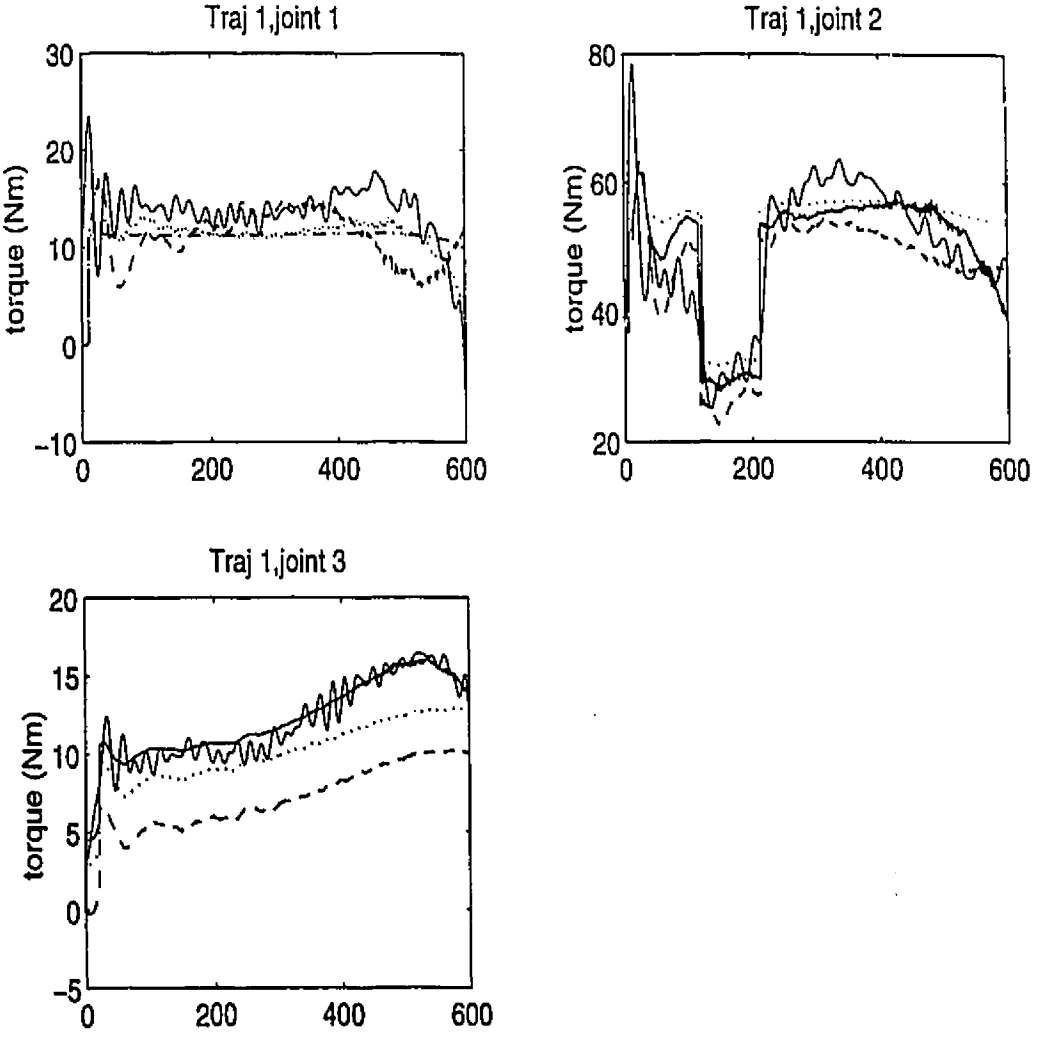

Figure 7.4: Model comparison - trajectory 1

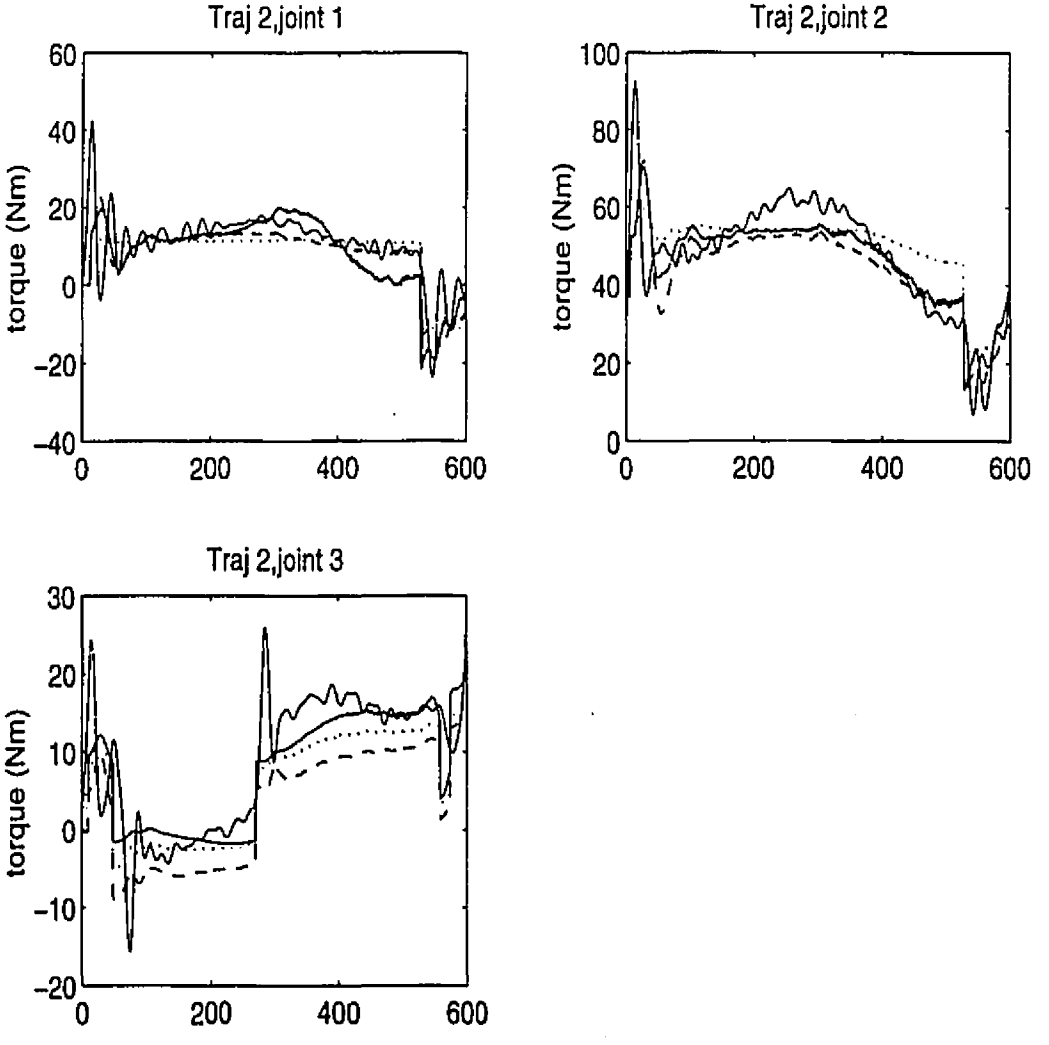

Figure 7.5: Model comparison - trajectory 2

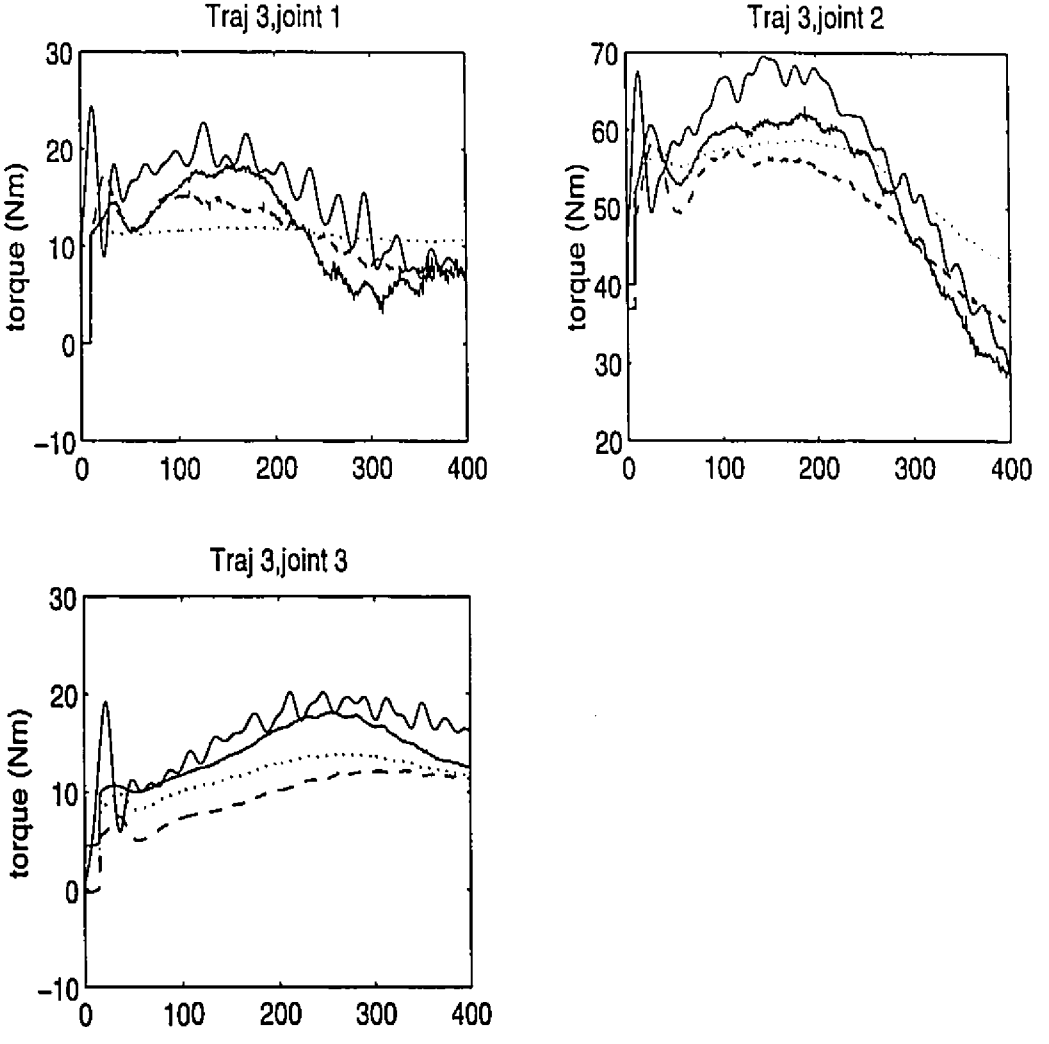

Figure 7.6: Model comparison - trajectory 3

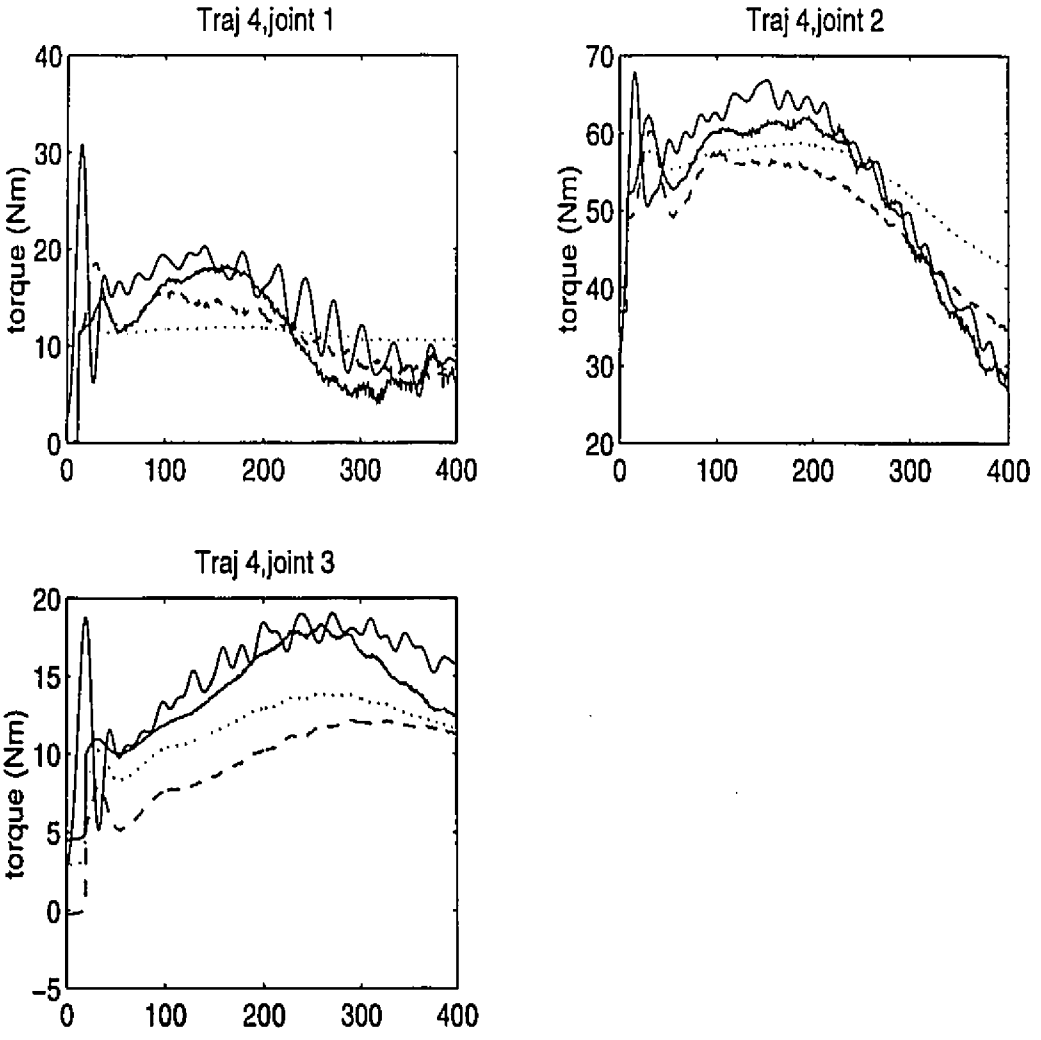

Figure 7.7: Model comparison - trajectory 4

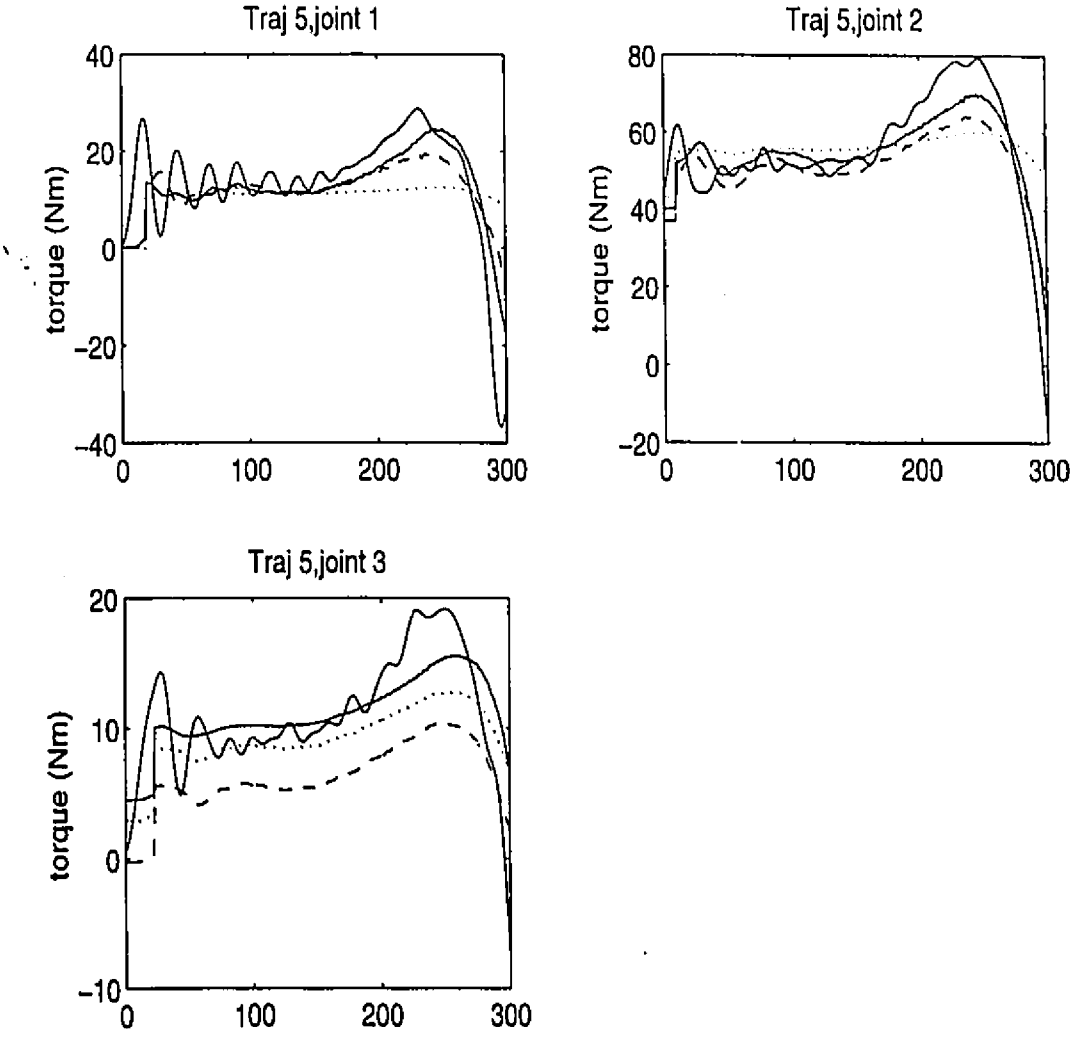

Figure 7.8: Model comparison - trajectory 5

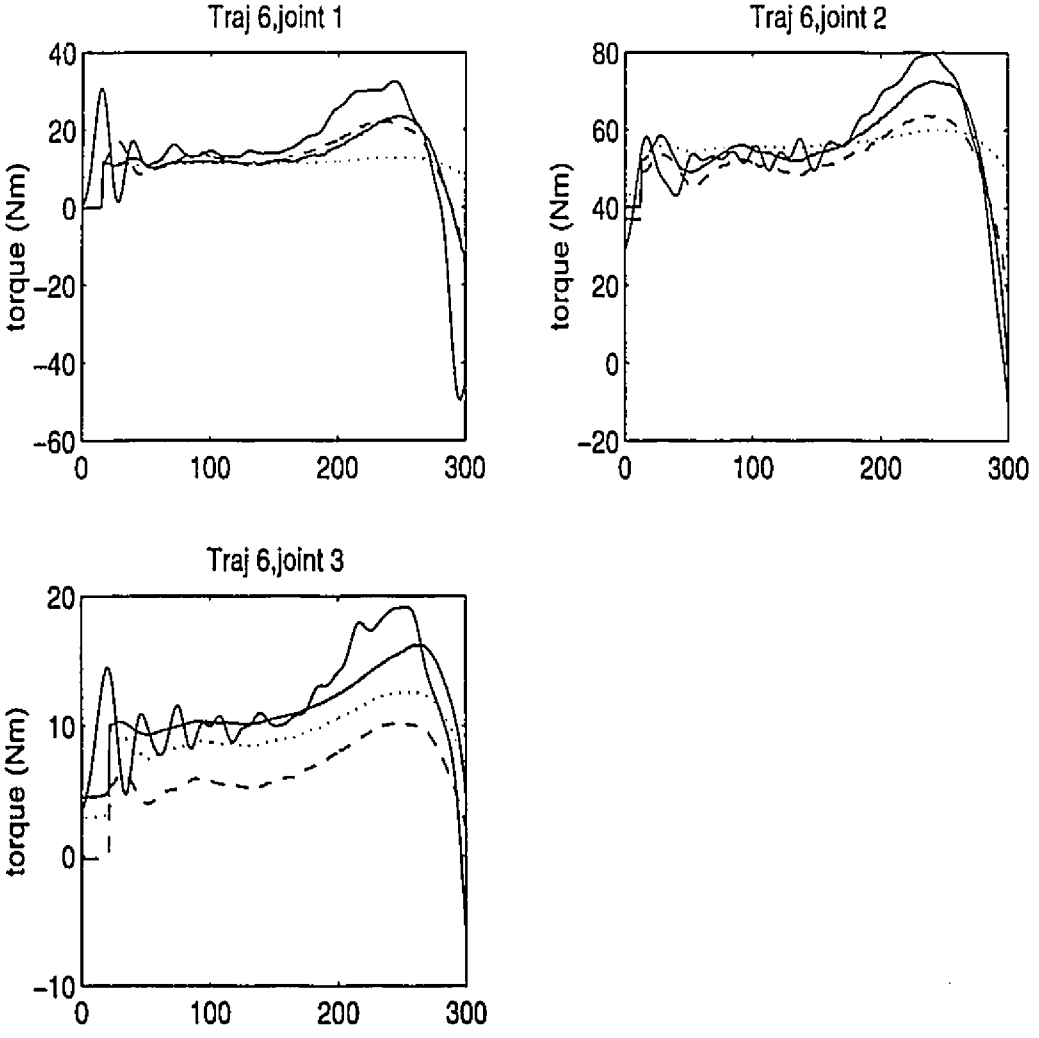

Figure 7.9: Model comparison - trajectory 6

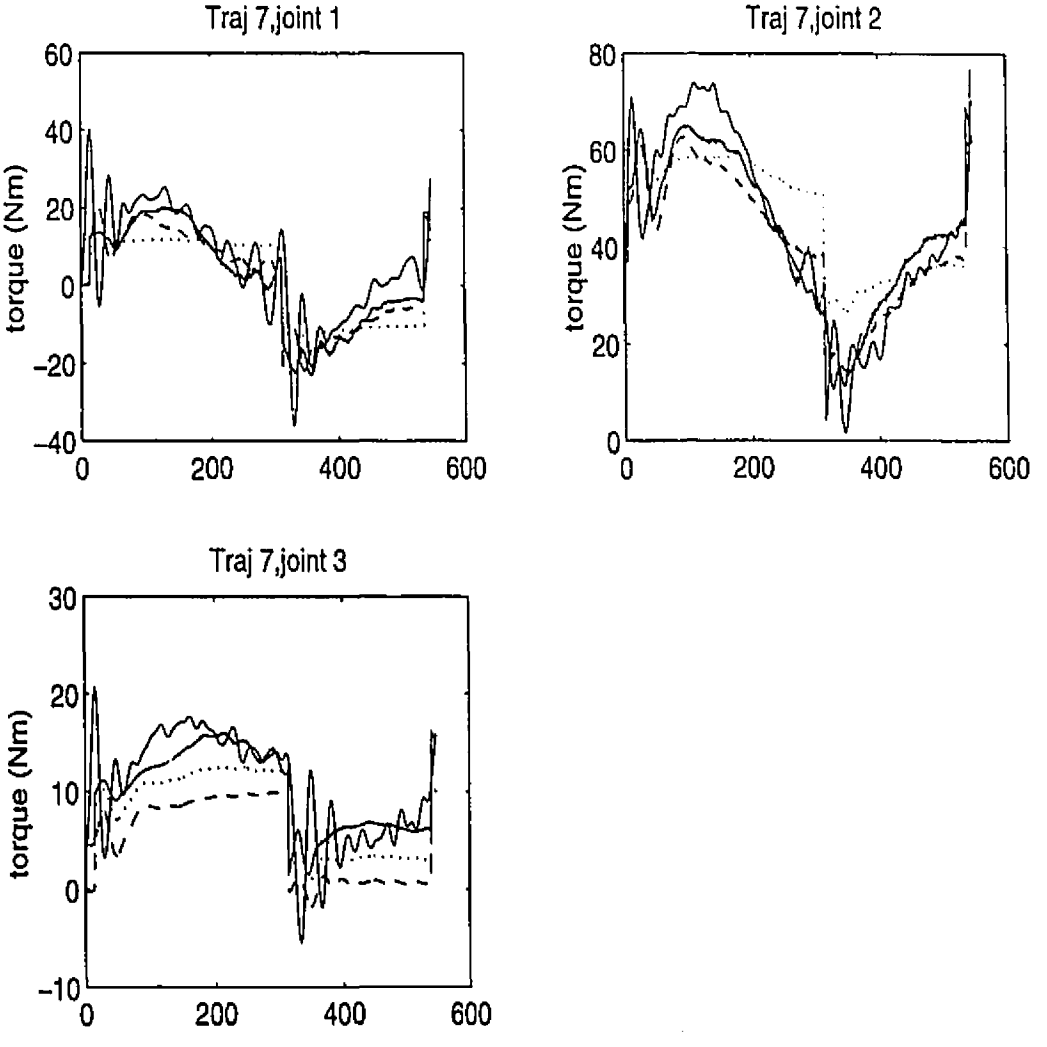

Figure 7.10: Model comparison - trajectory 7

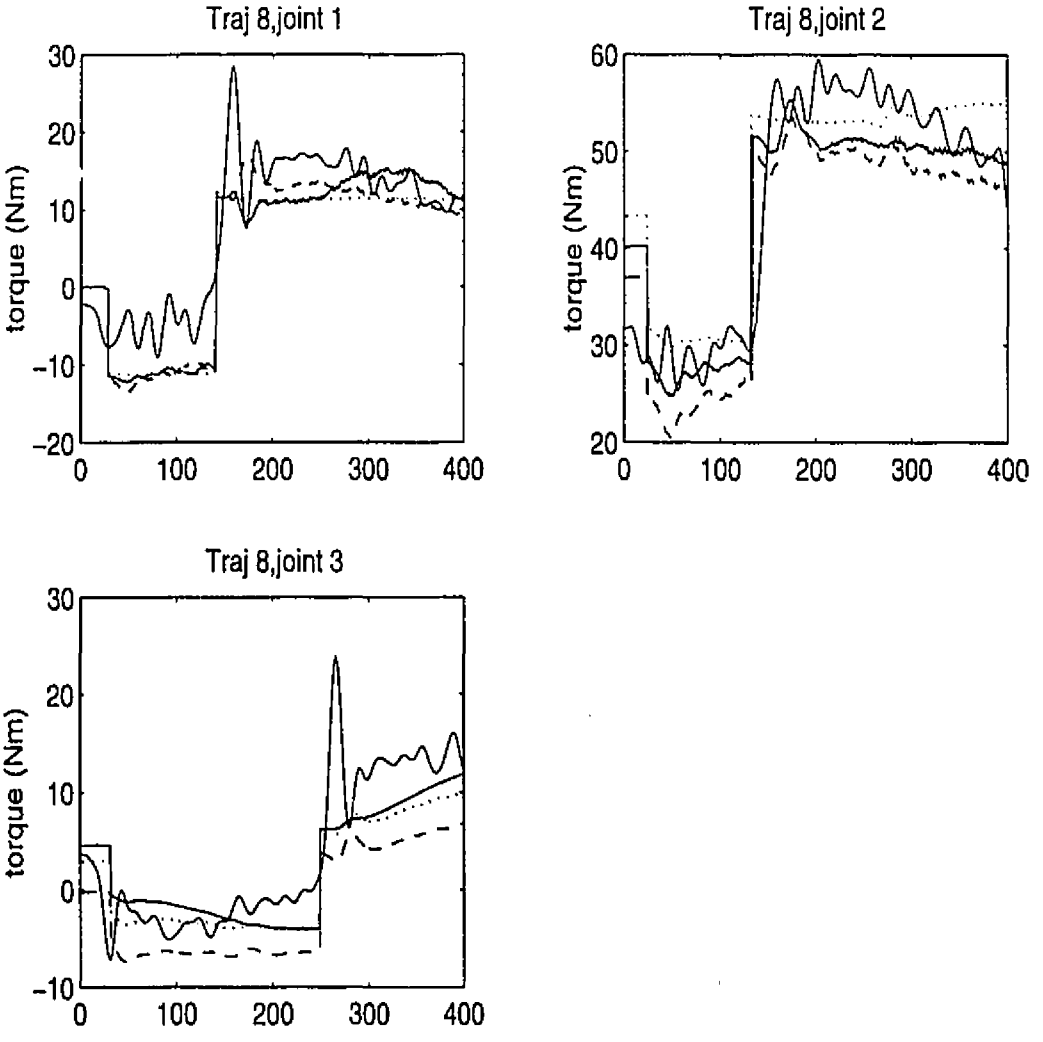

Figure 7.11: Model comparison - trajectory 8

seen to be quite accurate; the calculated error is less than 5 percent of the recorded torque for each joint.

7.3.2 Results of feedforward control

The dynamic model for gravity compensation was built using the dynmodel software. The C code listing is given in Appendix 1. The accuracy of the feedforward compensation was tested by observing the step response at joints 1 to 4. A demand step was applied to each joint; the step responses with and without gravitational compensation are shown in figures 2 and 3 respectively. The results demonstrate the accuracy of the dynamic model.

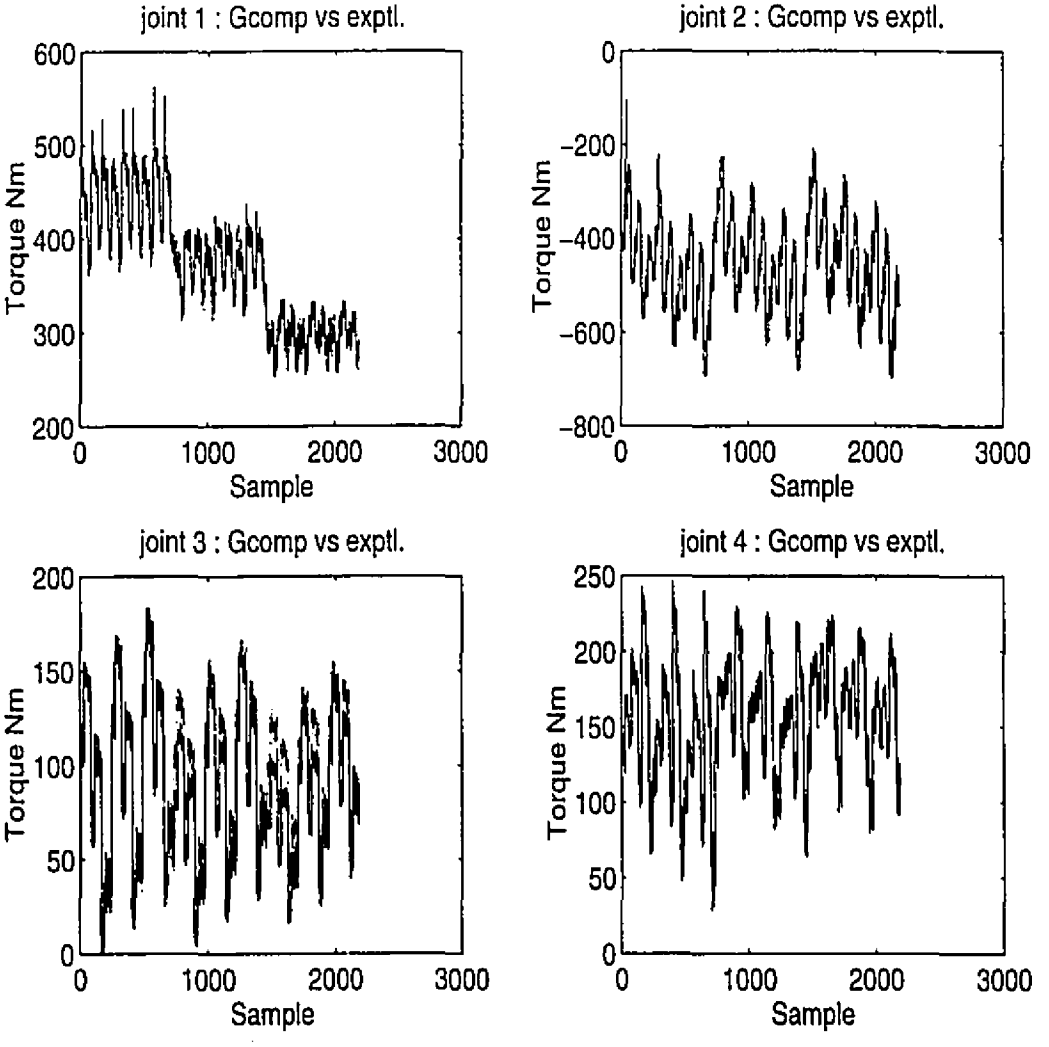

Figure 7.12: The recorded and computed torque values for the Sarcos GRLA

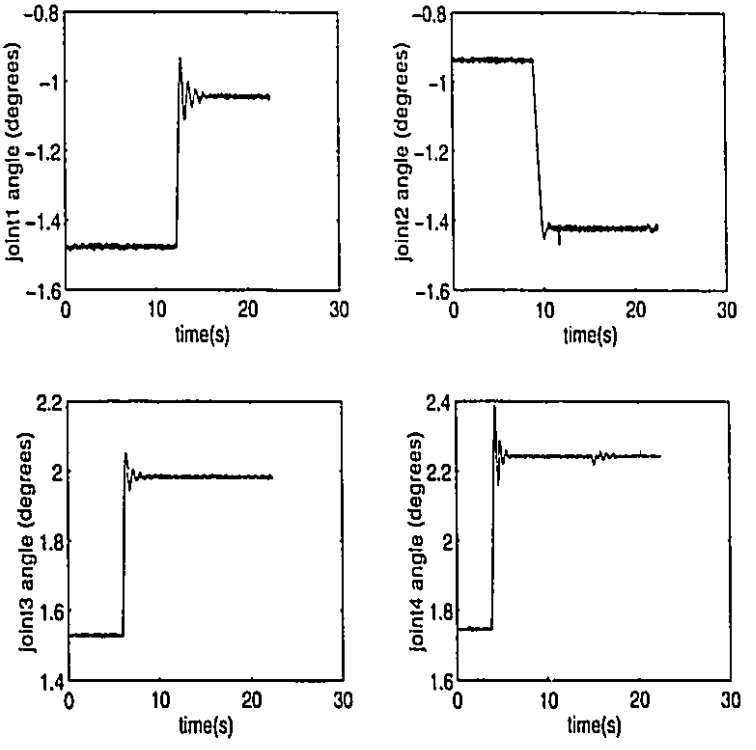

Figure 7.13: Joint step responses of the Sarcos manipulator with gravity compensation

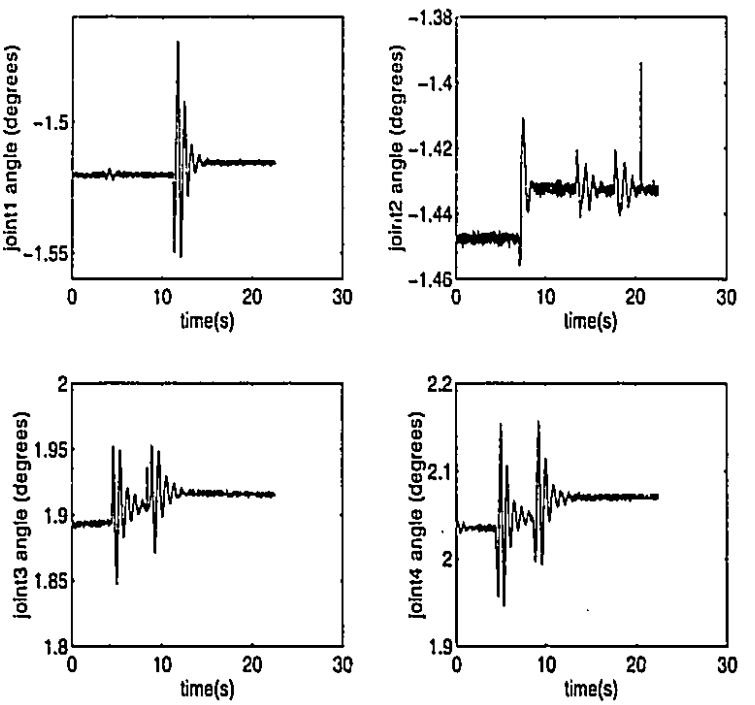

Figure 7.14: Joint step responses without gravity compensation

Par	Value
P_1	3.9495
P_2	3.7920
P_3	0.1040
P_4	5.6320
P_5	-1.3804
P_6	-.0070
P_7	-.6889
P_8	-.0227
P_9	.0254
P_{10}	.8632
P_{11}	.7491
P_{12}	.3006
P_{13}	-.0106
P_{14}	-.0038
P_{15}	-.1342

Table 7.1: Base parameters using the full order model of Khatib

Trajectory	Unscaled	Significance(σ_p)		Significance(svd)	
	κ_0	eliminated set	κ_1	eliminated set	κ_2
1	2499.5	1,5,7,12,13,14	92.1002	3,5,6,9,10	44.4784
2	1096.3	1,5,6,7,12	57.8807	5,9,10,12,17	84.54
3	502.3	7,14	498.5	2,3,4,9,10,17	127.0

Table 7.2: Effect of condition on the significance analysis

Parameter	Batch LS	Elimination	Standard(Khatib)
P_1	-2.33	-2.27	3.95
P_2	3.65	3.63	3.79
P_3	0.2882	**0.1726**	0.1040
P_4	2.1338	1.9362	5.6320
P_5	-21.4683	**-1.1323**	-1.3804
P_6	3.1672	**-0.7777**	-0.0070
P_7	9.6260	8.2690	-0.6889
P_8	2.5412	2.9666	-0.0227
P_9	-0.4547	-0.4591	0.0254
P_{10}	1.1508	1.1271	0.8632
P_{11}	0.0706	**0.1509**	0.7491
P_{12}	-0.3341	-5.1987	0.3006
P_{13}	-0.0859	2.1842	-0.0106
P_{14}	-1.7325	-1.7476	-0.0038
P_{15}	-0.4610	**-0.2702**	-0.1342
P_{16}	11.2301	11.2216	ϕ
P_{17}	11.8241	11.8651	ϕ
P_{18}	5.0680	5.0491	ϕ

Table 7.3: Batch LS and weakest direction elimination

Parameter	Computed	Standard(Khatib)
P_2	4.2884	3.7920
P_3	1.5080	0.1040
P_9	-0.4331	0.0254
P_{10}	6.0005	0.8632
P_{11}	1.4839	0.7491
P_{13}	-7.2153	-0.0106
P_{14}	-.5405	-0.0038
P_{15}	.4607	-0.1342
P_{16}	5.0596	ϕ
P_{17}	16.9753	ϕ
P_{18}	2.6616	ϕ

Table 7.4: Base parameters using the optimal input for the LMS method

Par	Value	Standard(Khatib)
P_2	4.1138	3.7920
P_3	0.0880	0.1040
P_9	-0.2991	0.0254
P_{10}	0.9029	0.8632
P_{11}	0.5202	0.7491
P_{13}	-0.0121	-0.0106
P_{14}	-0.0013	-0.0038
P_{15}	0.0076	-0.1342
P_{16}	11.2252	ϕ
P_{17}	11.6792	ϕ
P_{18}	5.0324	ϕ

Table 7.5: LMS results along the convergent mode

Inertial parameter	Estimated value
$m\rho_{x1}$	3.7467
$m\rho_{z1}$	3.7196
$m\rho_{x2}$	5.3212
$m\rho_{z2}$	78.096
$m\rho_{x3}$	-2.547
$m\rho_{z3}$	3.905
$m\rho_{x4}$	0.4624
$m\rho_{z4}$	23.1605
$m\rho_{x5}$	0.2526
$m\rho_{z5}$	0.616
$m\rho_{x6}$	-0.1174
$m\rho_{z6}$	-0.0052
$m\rho_{x7}$	-1.7239
$m\rho_{y7}$	-0.3668

Table 7.6: Estimation results for base gravity parameters of the Sarcos GRLA manipulator

Joint	σ_τ
1	12.6941
2	24.9719
3	9.1544
4	5.7428
5	1.3930
6	6.8802
7	1.8714

Table 7.7: Standard deviation of the joint torque error for the Sarcos GRLA

Chapter 8

Conclusions

In this chapter, we conclude the thesis with a summary of main results in section 1. Section 2 presents the scope for future work in this area.

8.1 Review of results

This thesis investigated estimation for two particular serial manipulators: an accurately calibrated direct drive manipulator and a geared manipulator in which sensor readings did not yield very accurate data. Unmodelled dynamics and motion noise are ubiquitous elements in dynamic calibration. Estimation accuracy reduces further due to an erroneous manipulator state. We review below some suggestions for improving the accuracy of the dynamic calibration procedure.

- for increasing the accuracy of the least squares estimation procedure, we considered the following:

- ridge regression: we suggest ridge regression only if the unmodelled dynamics are not very significant and an approximate *a priori* estimate of the parameter vector is available.
- column scaling: scaling the regressor columns by the experimental torque sensitivity suggested in Chapter 4 was found to reduce the regressor condition.
- significance analysis: two methods of significance analysis were considered; namely, eliminating the parameters with low values for the experimental torque sensitivities and eliminating the near dependant parameters in the regressor. These methods allow us to eliminate successively, the terms most insignificant and therefore most likely to be in error for the identification trajectory. It must of course be noted that different trajectories may have different sets of insignificant parameters.
- we found that the LS procedure conducted using the singular value decomposition gave increasingly accurate results when identification in the weakest direction was eliminated.

- LMS estimation: the LMS adaptive method was also found to generate accurate estimates for the inertial parameters; generation of exciting trajectories as a means to minimise parameter bias has been dealt with by Armstrong and was applied in this thesis. Our experience with the LMS procedure indicates that;

- the choice of an initial trajectory is important because it practically determines how far the optimisation would progress before actuator limitations were exceeded. It is also preferrable to start with the minimum number of parameters since optimisation becomes increasingly difficult as the number of parameters increases. We have suggested a method for selecting an initial trajectory in Chapter 5.
- it is necessary to choose appropriate values for the incremental scaling factor when updating the trajectory.
- eliminate estimation points likely to have large unmodelled dynamics; this is especially true of the initial part of the trajectory where static friction is overcome.
- at every iteration, only retain the incremental contribution of the eigenvectors corresponding to the stable modes. Applying the above procedures, we obtained a set of values for the significant inertial parameters in Table 7.4. The values were seen to closely approach the standard values.

The results for the PUMA 560 arm were found to closely match the values previously derived by Khatib for some of the base parameters. On comparing the results of the LS and the LMS methods with the measured torque signal and the torque computed using Khatibs parametric set; we found that the LS values provided the best approximation of the computed to the measured torque signal.

8.2 Suggestions for future work

The thesis concludes with some suggestions for future work:

- Explicit base parametric estimation: In chapter 3, we presented two theorems for evaluating the base parameters of a general rotational serial manpulator. In particular, the results of theorem 2 may be further analyzed to obtain explicit relations for the base parametric set in terms of the constant DH parameters alone. These could then be directly applied to the most general type of serial, rotational manipulator.

- Minimum variance estimators: The minimum variance estimation is a special form of the Kalman filter for linear, time-varying systems. Kalman filters and their applications are extensively dealt with in [18]. In order to apply Kalman filtering theory to the estimation problem, we must first model the inverse dynamics as a time varying system governed by the parameter state vector as shown below:

$$P_{k+1} = P_k + w_k$$

$$T_k = (H_k + \tilde{H}_k)' + T_{U_k} + T_{M_k}$$

where all quantities (which have been previously defined) are indexed by k. In order to incorporate an optimal filter, the covariance matrices of $[T_{M_k}]$ and $[w_k]$ are required. We shall denote these as Q_k and R_k respectively. Although the state vector $[P_k]$ should ideally remain unaltered between iterations, we assume the presence of a low energy state noise $[w_k]$ due to stability considerations [18].

The minimum error variance estimator is then given by the following equations:

$$P_{k+1} = P_k + K_k \left(Y_k - \left(H_k + \tilde{H}_k \right)' \hat{P}_k \right)$$

$$C_{k+1} = C_k - K_k \left(H_k + \tilde{H}_k \right)' C_k$$

$$K_k = C_k \left(H_k + \tilde{H}_k \right) \left[\left(H_k + \tilde{H}_k \right)' C_k \left(H_k + \tilde{H}_k \right) + R_\kappa \right]^{-1}$$

The unmodelled dynamics $[T_{U_k}]$ represent a nonwhite component in the measurement noise signal. Jazwinski derives the conditions for uniform, asymptotic stability of optimal filters. For an identification trajectory that generates a uniform, asymptoically stable filter, the parameter convergence is exponential. Jazwinski [28] has derived explicit relations for these time constants. Generating optimal trajectories that minimize these constants thus leading to rapid parameter convergence could form the topic of further research. Explicit expressions are also available [28] for the expected paramter bias due to $\tilde{H}_k$,T_{U_k} and δR_k.

As optimal filtering theoretically offers the most superior estimation for low values of the unmodelled dynamics, we suggest this as an alternative for future work.

Appendix A

Basis Coefficients of the 3 dof Puma 560 arm

In this appendix, we supply the basis coefficients of the Puma 560 arm. The choice of the basis coefficients is based on the results given by Mayeda et al. [8]. Coefficients were calculated by the Newton Euler method using the Maple symbolic computation package.

$C(J_{zz1})$

$$c_1 = \ddot{\theta}_1$$

$$c_2 = 0$$

$$c_3 = 0$$

$C(J_{zz2})$

$$
\begin{aligned}
c_1 &= 0 \\
c_2 &= \ddot{\theta}_2 \\
c_3 &= 0
\end{aligned}
$$

$C(m\rho_x^2)$

$$
\begin{aligned}
c_1 &= 0 \\
c_2 &= \cos(\theta_2)g \\
c_3 &= 0
\end{aligned}
$$

$C(m\rho_y^2)$

$$
\begin{aligned}
c_1 &= 0 \\
c_2 &= -\sin(\theta_2)g \\
c_3 &= 0
\end{aligned}
$$

$C(J_{xx2})$

$$
\begin{aligned}
c_1 &= \ddot{\theta}_1 - \cos(\theta_2)^2\ddot{\theta}_1 + 2\,\sin(\theta_2)\cos(\theta_2)\dot{\theta}_1\,\dot{\theta}_2 \\
c_2 &= -\cos(\theta_2)\dot{\theta}_1{}^2\sin(\theta_2) \\
c_3 &= 0
\end{aligned}
$$

$C(J_{xz2})$

$$
\begin{aligned}
c_1 &= \cos(\theta_2)\dot{\theta_2}^2 + \sin(\theta_2)\ddot{\theta_2} \\
c_2 &= \sin(\theta_2)\ddot{\theta_1} \\
c_3 &= 0
\end{aligned}
$$

$C(J_{xy2})$

$$
\begin{aligned}
c_1 &= 2\sin(\theta_2)\cos(\theta_2)\ddot{\theta_1} + 4\cos(\theta_2)^2\dot{\theta_1}\dot{\theta_2} - 2\dot{\theta_1}\dot{\theta_2} \\
c_2 &= \dot{\theta_1}^2 - 2\cos(\theta_2)^2\dot{\theta_1}^2 \\
c_3 &= 0
\end{aligned}
$$

$C(J_{yz2})$

$$
\begin{aligned}
c_1 &= \cos(\theta_2)\ddot{\theta_2} - \sin(\theta_2)\dot{\theta_2}^2 \\
c_2 &= \cos(\theta_2)\ddot{\theta_1} \\
c_3 &= 0
\end{aligned}
$$

$C(J_{zz3})$

$$
\begin{aligned}
c_1 &= 0 \\
c_2 &= \ddot{\theta_3} + \ddot{\theta_2} \\
c_3 &= \ddot{\theta_3} + \ddot{\theta_2}
\end{aligned}
$$

$C(m\rho_x^3)$

$$
\begin{aligned}
c_1 &= -4\cos(\theta_2)\cos(\theta_3)\sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 L - 2\cos(\theta_2)L\dot{\theta}_3\cos(\theta_3)\sin(\theta_2)\dot{\theta}_1 + \\
&\quad 2\sin(\theta_3)\dot{\theta}_1\dot{\theta}_2 L - 4L\sin(\theta_3)\cos(\theta_2)^2\dot{\theta}_1\dot{\theta}_2 + 2\cos(\theta_3)L\cos(\theta_2)^2\ddot{\theta}_1 \\
&\quad -2\sin(\theta_2)\sin(\theta_3)L\cos(\theta_2)\ddot{\theta}_1 - 2L\dot{\theta}_3\sin(\theta_3)\cos(\theta_2)^2\dot{\theta}_1 \\
c_2 &= 2\cos(\theta_3)\ddot{\theta}_2 L + \cos(\theta_3)\cos(\theta_2)g + \\
&\quad 2\cos(\theta_3)\sin(\theta_2)\dot{\theta}_1^{\,2}\cos(\theta_2)L + 2\sin(\theta_3)\cos(\theta_2)^2\dot{\theta}_1^{\,2}L - \\
&\quad \sin(\theta_3)\sin(\theta_2)g - L\sin(\theta_3)\dot{\theta}_3^{\,2} + L\cos(\theta_3)\ddot{\theta}_3 - \\
&\quad 2L\sin(\theta_3)\dot{\theta}_2\dot{\theta}_3 - L\dot{\theta}_1^{\,2}\sin(\theta_3) \\
c_3 &= -\sin(\theta_3)\sin(\theta_2)g + \cos(\theta_3)\cos(\theta_2)g + \sin(\theta_3)\cos(\theta_2)^2\dot{\theta}_1^{\,2}L + \\
&\quad \sin(\theta_3)\dot{\theta}_2^{\,2}L + \cos(\theta_3)\ddot{\theta}_2 L + \cos(\theta_3)\sin(\theta_2)\dot{\theta}_1^{\,2}\cos(\theta_2)L
\end{aligned}
$$

$C(m\rho_y^3)$

$$
\begin{aligned}
c_1 &= -2\sin(\theta_3)L\cos(\theta_2)^2\ddot{\theta}_1 + 2\cos(\theta_3)\dot{\theta}_1\dot{\theta}_2 L - \\
&\quad 4L\cos(\theta_3)\cos(\theta_2)^2\dot{\theta}_1\dot{\theta}_2 + 4\cos(\theta_2)\sin(\theta_3)\sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 L + \\
&\quad 2\cos(\theta_2)L\dot{\theta}_3\sin(\theta_3)\sin(\theta_2)\dot{\theta}_1 - \\
&\quad 2\sin(\theta_2)\cos(\theta_3)L\cos(\theta_2)\ddot{\theta}_1 - 2L\dot{\theta}_3\cos(\theta_3)\cos(\theta_2)^2\dot{\theta}_1 \\
c_2 &= -\sin(\theta_3)\cos(\theta_2)g - 2\sin(\theta_3)\ddot{\theta}_2 L - \\
&\quad 2\sin(\theta_3)\sin(\theta_2)\dot{\theta}_1^{\,2}\cos(\theta_2)L - \cos(\theta_3)\sin(\theta_2)g + 2\cos(\theta_3)\cos(\theta_2)^2\dot{\theta}_1^{\,2}L - \\
&\quad L\sin(\theta_3)\ddot{\theta}_3 - L\cos(\theta_3)\dot{\theta}_3^{\,2} - L\dot{\theta}_1^{\,2}\cos(\theta_3) - \\
&\quad 2L\cos(\theta_3)\dot{\theta}_2\,\dot{theta}_3 \\
c_3 &= -\cos(\theta_3)\sin(\theta_2)g - \sin(\theta_3)\cos(\theta_2)g + \cos(\theta_3)\cos(\theta_2)^2\dot{\theta}_1^{\,2}L +
\end{aligned}
$$

$$\cos(\theta_3)\dot{\theta}_2^{\,2} L - \sin(\theta_3)\ddot{\theta}_2 L -$$
$$\sin(\theta_3)\sin(\theta_2)\dot{\theta}_1^{\,2}\cos(\theta_2) L$$

$C(J_{xx3})$

$$\begin{aligned}
c_1 &= 2\sin(\theta_2)\cos(\theta_3)\sin(\theta_3)\cos(\theta_2)\ddot{\theta}_1 + \cos(\theta_2)^2\ddot{\theta}_1 - \\
&\quad 2\cos(\theta_3)^2\cos(\theta_2)^2\ddot{\theta}_1 + \cos(\theta_3)^2\ddot{\theta}_1 + 4\sin(\theta_2)\dot{\theta}_3\cos(\theta_3)^2\cos(\theta_2)\dot{\theta}_1 + \\
&\quad 4\sin(\theta_2)\cos(\theta_3)^2\cos(\theta_2)\dot{\theta}_1\dot{\theta}_2 - 2\sin(\theta_2)\cos(\theta_2)\dot{\theta}_1\dot{\theta}_2 - \\
&\quad 2\sin(\theta_2)\dot{\theta}_3\cos(\theta_2)\dot{\theta}_1 - 2\cos(\theta_3)\dot{\theta}_3\sin(\theta_3)\dot{\theta}_1 + \\
&\quad 4\sin(\theta_3)\dot{\theta}_3\cos(\theta_3)\cos(\theta_2)^2\dot{\theta}_1 + 4\sin(\theta_3)\cos(\theta_3)\cos(\theta_2)^2\dot{\theta}_1\dot{\theta}_2 - \\
&\quad 2\cos(\theta_3)\sin(\theta_3)\dot{\theta}_1\dot{\theta}_2 \\
c_2 &= -2\cos(\theta_3)^2\cos(\theta_2)\dot{\theta}_1^{\,2}\sin(\theta_2) + \cos(\theta_2)\dot{\theta}_1^{\,2}\sin(\theta_2) + \\
&\quad \sin(\theta_3)\dot{\theta}_1^{\,2}\cos(\theta_3) - 2\cos(\theta_3)\cos(\theta_2)^2\dot{\theta}_1^{\,2}\sin(\theta_3) \\
c_3 &= -2\cos(\theta_3)^2\cos(\theta_2)\dot{\theta}_1^{\,2}\sin(\theta_2) + \cos(\theta_2)\dot{\theta}_1^{\,2}\sin(\theta_2) + \\
&\quad \sin(\theta_3)\dot{\theta}_1^{\,2}\cos(\theta_3) - 2\cos(\theta_3)\cos(\theta_2)^2\dot{\theta}_1^{\,2}\sin(\theta_3)
\end{aligned}$$

$C(J_{zz3})$

$$\begin{aligned}
c_1 &= \cos(\theta_2)\sin(\theta_3)\ddot{\theta}_3 + \cos(\theta_2)\cos(\theta_3)\dot{\theta}_3^{\,2} + \\
&\quad \cos(\theta_2)\cos(\theta_3)\dot{\theta}_2^{\,2} - \sin(\theta_2)\sin(\theta_3)\dot{\theta}_3^{\,2} - \\
&\quad \sin(\theta_2)\sin(\theta_3)\dot{\theta}_2^{\,2} + \sin(\theta_2)\cos(\theta_3)\ddot{\theta}_3 + \sin(\theta_2)\cos(\theta_3)\ddot{\theta}_2 + \\
&\quad 2\cos(\theta_2)\cos(\theta_3)\dot{\theta}_2\dot{\theta}_3 + \cos(\theta_2)\sin(\theta_3)\ddot{\theta}_2 - \\
&\quad 2\sin(\theta_2)\sin(\theta_3)\dot{\theta}_2\dot{\theta}_3
\end{aligned}$$

$$c_2 = \sin(\theta_3)\cos(\theta_2)\ddot{\theta}_1 + \cos(\theta_3)\sin(\theta_2)\ddot{\theta}_1$$

$$c_3 = \sin(\theta_3)\cos(\theta_2)\ddot{\theta}_1 + \cos(\theta_3)\sin(\theta_2)\ddot{\theta}_1$$

$C(J_{xy3})$

$$\begin{aligned}
c_1 = {} & -4\,\dot{\theta}_3\,\dot{\theta}_1\,\cos(\theta_3)^2 - 4\,\dot{\theta}_3\,\cos(\theta_2)^2\dot{\theta}_1 - 2\,\sin(\theta_2)\cos(\theta_2)\ddot{\theta}_1 - 4\,\cos(\theta_2)^2\dot{\theta}_1\,\dot{\theta}_2 - \\
& 2\,\cos(\theta_3)\sin(\theta_3)\ddot{\theta}_1 - 8\,\sin(\theta_2)\cos(\theta_3)\sin(\theta_3)\cos(\theta_2)\dot{\theta}_1\,\dot{\theta}_2 - \\
& 8\,\sin(\theta_2)\cos(\theta_3)\dot{\theta}_3\,\sin(\theta_3)\cos(\theta_2)\dot{\theta}_1 + 4\,\sin(\theta_3)\cos(\theta_3)\cos(\theta_2)^2\ddot{\theta}_1 + \\
& 8\,\dot{\theta}_3\,\cos(\theta_3)^2\cos(\theta_2)^2\dot{\theta}_1 + 8\,\cos(\theta_3)^2\cos(\theta_2)^2\dot{\theta}_1\,\dot{\theta}_2 + \\
& 4\,\sin(\theta_2)\cos(\theta_3)^2\cos(\theta_2)\ddot{\theta}_1 + 2\,\dot{\theta}_1\,\dot{\theta}_2 - \\
& 4\,\dot{\theta}_1\,\dot{\theta}_2\,\cos(\theta_3)^2 + 2\,\dot{\theta}_3\,\dot{\theta}_1 \\
c_2 = {} & 2\,\cos(\theta_3)^2\dot{\theta}_1^{\,2} - 4\,\cos(\theta_3)^2\cos(\theta_2)^2\dot{\theta}_1^{\,2} + \\
& 4\,\cos(\theta_3)\sin(\theta_2)\dot{\theta}_1^{\,2}\sin(\theta_3)\cos(\theta_2) - \dot{\theta}_1^{\,2} + \\
& 2\,\cos(\theta_2)^2\dot{\theta}_1^{\,2} \\
c_3 = {} & 2\,\cos(\theta_3)^2\dot{\theta}_1^{\,2} - 4\,\cos(\theta_3)^2\cos(\theta_2)^2\dot{\theta}_1^{\,2} + \\
& 4\,\cos(\theta_3)\sin(\theta_2)\dot{\theta}_1^{\,2}\sin(\theta_3)\cos(\theta_2) - \dot{\theta}_1^{\,2} + \\
& 2\,\cos(\theta_2)^2\dot{\theta}_1^{\,2}
\end{aligned}$$

$C(J_{yz3})$

$$\begin{aligned}
c_1 = {} & -\sin(\theta_2)\cos(\theta_3)\dot{\theta}_3^{\,2} - \sin(\theta_2)\sin(\theta_3)\ddot{\theta}_2 - 2\,\sin(\theta_2)\cos(\theta_3)\dot{\theta}_2\,\dot{\theta}_3 - \\
& \cos(\theta_2)\sin(\theta_3)\dot{\theta}_3^{\,2} - 2\,\cos(\theta_2)\sin(\theta_3)\dot{\theta}_2\,\dot{\theta}_3 - \cos(\theta_2)\sin(\theta_3)\dot{\theta}_2^{\,2} - \\
& \sin(\theta_2)\sin(\theta_3)\ddot{\theta}_3 + \cos(\theta_2)\cos(\theta_3)\ddot{\theta}_2 + \cos(\theta_2)\cos(\theta_3)\ddot{\theta}_3 - \sin(\theta_2)\cos(\theta_3)\dot{\theta}_2^{\,2}
\end{aligned}$$

$$c_2 = -\sin(\theta_3)\sin(\theta_2)\ddot{\theta}_1 + \cos(\theta_3)\cos(\theta_2)\ddot{\theta}_1$$

$$c_3 = -\sin(\theta_3)\sin(\theta_2)\ddot{\theta}_1 + \cos(\theta_3)\cos(\theta_2)\ddot{\theta}_1$$

Appendix B

The dyn_model lisp code

In this appendix, we list the LISP code written for generating a dynamic model for the manipulator inverse dynamics as efficiently as possible by the method of construting local variables per Khalil et al. [25] and saving redundant computations by identifying zero terms and eliminating them in the output code. The software has been coded in Common Lisp; however, some ideas for functions and macros were drawn from the Symbolics Lisp code initially written by Izaguirre et al. [5] for generating their model. Accordingly, we shall preserve the name of the functions that were used by them although they may be slightly modified.

The input format for the base parameters is also identical to the above mentioned paper. The reader is referred to this paper for the required input format. We give as an example, the input file for the base parameters for the PUMA 560 arm.

```
number"link 3
M 0 0 0
SIGMA 0 0 0
AL 90 0 -90
TH 0 0 0
```

```
R 0 .149 0
D 0 .432 .02
XX 0 0 - 1939918
XY 0 .27590585 0
XZ 0 0 - .01917035
YY 0.023046964 0 0
YZ 0 -.013731773 .01250796
ZZ 0 0 0
MX 0 -.03555448 0
MY 0 .03011657 0
MZ 0 0 .01999423
IA 0 0 0
FS .115261 -.068686146 .09904939
FV .14671 -.214025 .0926147
GX 0
GY 0
GZ 9.81
stop
```

Once the code is loaded, the following steps generate dynamic model.

```
(load 'macros.lisp)
(load 'sun.lisp)
(load 'newdefuns.lisp)
(load 'load.lisp)
(reader 'input'file.ex)
(load 'def1.lisp)
(fill-defs)
(transf)
(or2OR)
(out-recur)
(in-recur)
(load 'print.lisp)
(printoC 'output'file.c)
```

A dynamic model can be generated only for gravity compensation by the function call $(only-grav)$ after $(fill-defs)$. The code listing is given below:

MACROS.LISP : the macro definitions

```
(defmacro aset (val array &optional index1 index2 index3 index4 index5)
`(cond   ((null ,index2 ) (setf (aref ,array ,index1) ,val))
        ((null ,index3 ) (setf (aref ,array ,index1 ,index2) ,val))
        ((null ,index4 ) (setf (aref ,array ,index1 ,index2 ,index3) ,val))
        (t               (setf (aref ,array ,index1 ,index2 ,index3 ,index4) ,val))
 )
)
```

```
(defmacro for (index from init to fin do &optional arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9
                                    arg10 arg11 arg12 arg13 arg14 arg15 arg16 arg17 arg18
                                    arg19 arg20 arg21 arg22 arg23 arg24 arg25 arg26
                                    arg27 arg28 arg29 arg30 arg31 arg32 arg33 arg34 arg35
                                    arg36 arg37 arg38 arg39 arg40 arg41 arg42 arg43 arg44
                                    arg45 arg46 arg47 arg48 arg49 arg50 arg51 arg52 arg53
                                    arg54 arg55 arg56 arg57 arg58 arg59 arg60 arg61 arg62
                                    arg63 arg64 arg65 arg66 arg67 arg68 arg69 arg70 arg71
                                    arg72 arg73 arg74 arg75 arg76 arg77 arg78 arg79 arg80
                                    arg81 arg82 arg83 arg84 arg85 arg86 arg87 arg88 arg89
                                    arg90 arg91 arg92)
'(cond ((< ,fin ,init) (do ((,index ,init)) ((= ,index (1+ ,fin)))
,arg1 ,arg2 ,arg3 ,arg4 ,arg5 ,arg6 ,arg7 ,arg8 ,arg9 ,arg10 ,arg11 ,arg12 ,arg13 ,arg14
,arg15 ,arg16 ,arg17 ,arg18 ,arg19 ,arg20 ,arg21 ,arg22 ,arg23 ,arg24
,arg25 ,arg26 ,arg27 ,arg28 ,arg29 ,arg30 ,arg31 ,arg32 ,arg33 ,arg34 ,arg35 ,arg36 ,arg37
,arg38 ,arg39 ,arg40 ,arg41 ,arg42 ,arg43 ,arg44 ,arg45 ,arg46 ,arg47 ,arg48 ,arg49 ,arg50
,arg51 ,arg52 ,arg53 ,arg54 ,arg55 ,arg56 ,arg57 ,arg58 ,arg59 ,arg60 ,arg61 ,arg62 ,arg63
,arg64 ,arg65 ,arg66 ,arg67 ,arg68 ,arg69 ,arg70 ,arg71 ,arg72 ,arg73 ,arg74 ,arg75 ,arg76
,arg77 ,arg78 ,arg79 ,arg80 ,arg81 ,arg82 ,arg83 ,arg84 ,arg85 ,arg86 ,arg87 ,arg88 ,arg89
,arg90 ,arg91 ,arg92
(incf ,index)))
      ((> ,fin ,init) (do ((,index ,init)) ((= ,index (1- ,fin)))
,arg1 ,arg2 ,arg3 ,arg4 ,arg5 ,arg6 ,arg7 ,arg8 ,arg9 ,arg10 ,arg11 ,arg12 ,arg13 ,arg14
,arg15 ,arg16 ,arg17 ,arg18 ,arg19 ,arg20 ,arg21 ,arg22 ,arg23 ,arg24
,arg25 ,arg26 ,arg27 ,arg28 ,arg29 ,arg30 ,arg31 ,arg32 ,arg33 ,arg34 ,arg35 ,arg36 ,arg37
,arg38 ,arg39 ,arg40 ,arg41 ,arg42 ,arg43 ,arg44 ,arg45 ,arg46 ,arg47 ,arg48 ,arg49 ,arg50
,arg51 ,arg52 ,arg53 ,arg54 ,arg55 ,arg56 ,arg57 ,arg58 ,arg59 ,arg60 ,arg61 ,arg62 ,arg63
,arg64 ,arg65 ,arg66 ,arg67 ,arg68 ,arg69 ,arg70 ,arg71 ,arg72 ,arg73 ,arg74 ,arg75 ,arg76
,arg77 ,arg78 ,arg79 ,arg80 ,arg81 ,arg82 ,arg83 ,arg84 ,arg85 ,arg86 ,arg87 ,arg88 ,arg89
,arg90 ,arg91 ,arg92
(decf ,index)))
)
)

(defmacro my-if (test success &optional arg1 arg2 arg3 arg4 arg5 arg6 arg7 arg8 arg9 arg10
                                arg11 arg12 arg13 arg14 arg15 arg16 arg17 arg18 arg19)
          '(cond
           (,test ,success)
           (t (progn() ,arg1 ,arg2 ,arg3 ,arg4 ,arg5 ,arg6 ,arg7 ,arg8 ,arg9 ,arg10
                    ,arg11 ,arg12 ,arg13 ,arg14 ,arg15 ,arg16 ,arg17 ,arg18 ,arg19))
           )
)

FUNCTIONS.LISP : liste the functions

(defun explode'moi (name)
   (explode-help (symbol-name name) 0))

(defun explode (name)
```

```
(cond ((numberp name) '(name)) (t (explode-help (symbol-name name) 0))))
)

(defun explode-help (str ind)
(declare (fixnum ind)
 (string str))
   (cond
      ((>= ind (length str)) NIL)
      (t (cons (intern (string (char str ind)))
            (explode-help str (1+ ind))))))

(defun implode'moi (list)
 (do (
     (str (make-string (length list)))
     (lis list (cdr lis))
     (loc 0 (1+ loc))
     )
     ((null lis) (intern str))
     (declare (string str) (fixnum loc))
     (setf (char str loc) (char (symbol-name (car lis)) 0))
 )
)

;redefining implode'moi for convenience and lack of elegance ...

(defun implode'moi (list)
     (setq n (length list))
     (setq str (make-string n))
     (do ((i -1))
        ((= i  (- n 1)))
        (incf i)
        (cond
        ((stringp (car list)) (setf (char str i) (char (car list) 0)))
        ((atom    (car list)) (setf (char str i) (char (string (car list)) 0)))
         )
      (setq list (cdr list))
      )
(intern str)
)

; NEWDEFUNS.LISP : attempts to eliminate zero parameters terms, this is done by assuming that
; parameters enter the expressions only in multiplications so the ssm routine is modified to
; lookup the #define list and check if the symbols is there and also if it is zero, if zero
; the output of ssm is set to zero.

(defun mm (i j x)
(declare (special A))
(if (listp x) (setq x (make-array 4 :initial-contents (append '(nil) x))))
(setq ttemp (make-array 4))
(setf (aref ttemp 1) (ssa (ssm(aref A i j 1 1)(aref x 1))
                  (ssa (ssm(aref A i j 1 2)(aref x 2))(ssm(aref A i j 1 3)(aref x 3)))))
```

```
(setf (aref ttemp 2) (sss (ssm(aref A i j 2 1)(aref x 1))
                          (sss (ssm(aref A i j 2 2)(aref x 2))(ssm(aref A i j 2 3)(aref x 3)))))
(setf (aref ttemp 3) (sss (ssm(aref A i j 3 1)(aref x 1))
                          (sss (ssm(aref A i j 3 2)(aref x 2))(ssm(aref A i j 3 3)(aref x 3)))))
ttemp
)


(defun sss (x y)
(setq defined (mapcar 'car def-list))
(setq temp nil)
(my-if (and(numberp x)(zerop x)(numberp y)(zerop y))
(setq temp 0))

(my-if (and (and (symbolp x)(memberp x defined)(zerop (value-of x)))
         (and (symbolp y)(memberp y defined)(zerop (value-of y))))
      (setq temp 0)
      (my-if (and (symbolp x)(memberp x defined)(zerop (value-of x)))
      (setq temp (sss 0 y))
      (my-if (and (symbolp y)(memberp y defined)(zerop (value-of y)))
      (setq temp (sss 0 x)))
      )
)

(my-if (and(numberp x)(numberp y))
(setq temp (+ x y))
(setq temp (with-output-to-string (temp)
(my-if (numberp x) (if (zerop x)(format temp "(~A)" y)
                       (format temp "((~F)+~A)" x y))
               (my-if (numberp y) (if (zerop y)(format temp "(~A)" x)
                                      (format temp "((~F)+~A)" y x))
                              (format temp "(~A+~A)" x y)
               )
)
))
)
(if (stringp temp)(setq temp (make-symbol temp)))
temp
)

(defun sss (x y)
(setq temp nil)
(my-if (and(numberp x)(zerop x)(numberp y)(zerop y))
(setq temp 0))
(my-if (and (and (symbolp x)(memberp x defined)(zerop (value-of x)))
         (and (symbolp y)(memberp y defined)(zerop (value-of y))))
      (setq temp 0)
      (my-if (and (symbolp x)(memberp x defined)(zerop (value-of x)))
      (setq temp (sss 0 y))
      (my-if (and (symbolp y)(memberp y defined)(zerop (value-of y)))
```

```
      (setq temp (== x 0)))
      )
)
(my-if (and(numberp x)(numberp y))
(setq temp (- x y))
(setq temp (with-output-to-string (temp)
(my-if (numberp x) (if (zerop x) (format temp "(-~A)" y)
                         (format temp "(~F-~A)" x y))
              (my-if (numberp y)(if (zerop y)(format temp "~A" x)
                                    (format temp "(~A-(~F))" x y))
                         (format temp "(~A-~A)" x y)
              )
)
))
)
(if (stringp temp) (setq temp (make-symbol temp)))
temp
)

(defun sam (x y)
(setq temp nil)
(setq defined (mapcar 'car def-list))
(my-if (or (and (numberp x)(zerop x)) (and (numberp y)(zerop y)) )
      (setq temp 0)

;the added statement for examining zero-parameter terms
(my-if (or (and (symbolp x)(memberp x defined)(zerop (value-of x)))
         (and (symbolp y)(memberp y defined)(zerop (value-of y))))
      (setq temp 0)

      (my-if (and (numberp x)(numberp y))
            (setq temp (* x y))
(setq temp (with-output-to-string (temp)
            (my-if (numberp x)
(my-if (= x 1)(format temp "~A" y)
(my-if (= x -1)(format temp "(-~A)" y)
            (format temp "((~F)*~A)" x y)
))
            (my-if (numberp y)
(my-if (= y 1)(format temp "~A" x)
(my-if (= y -1)(format temp "(-~A)" x)
            (format temp "((~F)*~A)" y x)
))
            (format temp "(~A*~A)" x y)
            )
            )
))
      )
)
)
```

```
(if (stringp temp) (setq temp (make-symbol temp)))
temp
)

; value-of : to return the value corresponding to the given symbol
(defun value-of (x)
(setq temp def-list)
(setq n (length def-list))
(for i from 1 to n do
(if (equal (caar temp) x)
(setq ans (cadar temp))
)
(setq temp (cdr temp))
)
ans
)

(defun vva (x y)
(setq vva'temp (make-array 4))
(if (listp x) (setq x (make-array 4 :initial-contents (append '(nil) x))))
(if (listp y) (setq y (make-array 4 :initial-contents (append '(nil) y))))
(aset (ssa (aref x 1)(aref y 1)) vva'temp 1)
(aset (ssa (aref x 2)(aref y 2)) vva'temp 2)
(aset (ssa (aref x 3)(aref y 3)) vva'temp 3)
vva'temp
)

(defun svm (x y)
(setq ttemp (make-array 4))
(if (listp y) (setq y (make-array 4 :initial-contents (append '(nil) y))))
(aset (ssm x (aref y 1)) ttemp 1)
(aset (ssm x (aref y 2)) ttemp 2)
(aset (ssm x (aref y 3)) ttemp 3)
ttemp
)

(defun vsm (x y)
(setq tttemp (make-array 4))
(if (listp x) (setq x (make-array 4 :initial-contents (append '(nil) x))))
(aset (ssm y (aref x 1)) tttemp 1)
(aset (ssm y (aref x 2)) tttemp 2)
(aset (ssm y (aref x 3)) tttemp 3)
tttemp
)

;(defun ssa (x y)
;(setq temp nil)
;(my-if (and(numberp x)(zerop x)(numberp y)(zerop y))
;(setq temp 0))
;(my-if (and(numberp x)(numberp y))
```

```
,(setq temp (- x y)))
,(setq temp (with-output-to-string (temp)
,(my-if (numberp x) (format temp "(~F-~A)" x y)
.               (my-if (numberp y) (format temp "((~F)-~A)" y x)
.                       (format temp "(~A-~A)" x y)
.               )
,)
,))
,(setq temp (make-symbol temp))
,temp
;)

(defun cr (x y)
(setq cr'temp (make-array 4))
(if (listp x) (setq x (make-array 4 :initial-contents (append '(nil) x))))
(if (listp y) (setq y (make-array 4 :initial-contents (append '(nil) y))))
(aset (sss (ssm (aref x 2)(aref y 3)) (ssm (aref y 2)(aref x 3))) cr'temp 1)
(aset (sss (ssm (aref x 3)(aref y 1)) (ssm (aref y 3)(aref x 1))) cr'temp 2)
(aset (sss (ssm (aref x 1)(aref y 2)) (ssm (aref y 1)(aref x 2))) cr'temp 3)
cr'temp
)

(defun dp (x y)
(setq dp'temp (make-array 4))
(if (listp x) (setq x (make-array 4 :initial-contents (append '(nil) x))))
(if (listp y) (setq y (make-array 4 :initial-contents (append '(nil) y))))
(aset (ssm (aref x 1)(aref y 1)) dp'temp 1)
(aset (ssm (aref x 2)(aref y 2)) dp'temp 2)
(aset (ssm (aref x 3)(aref y 3)) dp'temp 3)
(setq dp'temp (sss (aref dp'temp 1) (sss (aref dp'temp 2)(aref dp'temp 3))))
dp'temp
)

(defun memberp (x y)
(if (equal (member x y) NIL)
(setq temp NIL)
(setq temp T)
)
temp
)

; reader reads the indicated file and generates the appropriate arrays with indicated values
; where not indicated, the values are assumed zero
(defun reader (file)
    (declare (special in number'link sigma r al th d m mX mY mZ XX XY XZ YZ YY ZZ IA Fs Fv GX GY GZ))
    (setq in (open file :direction :input))
    (setq temp (read in))
    (if (equal temp 'number'link) (setq number'link (read in))
       (and (error "the first argument should be number'link")(break)))
    (setq sigma (make-array (1+ number'link)))
    (setq r (make-array (1+ number'link)))
```

```
    (setq al (make-array (1+ number'link)))
    (setq th (make-array (1+ number'link)))
    (setq d (make-array (1+ number'link)))
    (setq m (make-array (1+ number'link)))
    (setq mX (make-array (1+ number'link)))
    (setq mY (make-array (1+ number'link)))
    (setq mZ (make-array (1+ number'link)))
    (setq XX (make-array (1+ number'link)))
    (setq XY (make-array (1+ number'link)))
    (setq XZ (make-array (1+ number'link)))
    (setq YZ (make-array (1+ number'link)))
    (setq YY (make-array (1+ number'link)))
    (setq ZZ (make-array (1+ number'link)))
    (setq IA (make-array (1+ number'link)))
    (setq Fs (make-array (1+ number'link)))
    (setq Fv (make-array (1+ number'link)))
    (loop
        (setq temp (read in))
    (cond ((equal temp 'sigma)(fill sigma))
        ((equal temp 'r    )(fill r    ))
        ((equal temp 'al   )(fill al   ))
        ((equal temp 'th   )(fill th   ))
        ((equal temp 'd    )(fill d    ))
        ((equal temp 'm    )(fill m    ))
        ((equal temp 'mX   )(fill mX   ))
        ((equal temp 'mY   )(fill mY   ))
        ((equal temp 'mZ   )(fill mZ   ))
        ((equal temp 'XX   )(fill XX   ))
        ((equal temp 'XY   )(fill XY   ))
        ((equal temp 'XZ   )(fill XZ   ))
        ((equal temp 'YZ   )(fill YZ   ))
        ((equal temp 'YY   )(fill YY   ))
        ((equal temp 'ZZ   )(fill ZZ   ))
        ((equal temp 'IA   )(fill IA   ))
        ((equal temp 'Fs   )(fill Fs   ))
        ((equal temp 'Fv   )(fill Fv   ))
        ((equal temp 'gx   )(setq gx (read in)))
        ((equal temp 'gy   )(setq gy (read in)))
        ((equal temp 'gz   )(setq gz (read in)))
        ((equal temp 'stop )(return nil))
        (t (and (error "something amiss")(break)))
    ))
(close in)
)

(defun fill (list)
(for i from 1 to number'link do
      (setf (aref list i) (read in))
)
)
```

PRINT.LISP : prints the dyn'model C code statements in the named output C file

```
(defun printoC (str)
(declare (special stmnts def-list float-decl stmnts))
(setq out (open str :direction :output))

; first the #define 's are taken care of first
(setq defines def-list)
(loop
(cond ((null defines) (return nil))
    (t
        (if (zerop (value-of (caar defines)))
           ()
                (progn()
                (princ "#define " out)
                (princ (caar defines) out)
                (princ " " out)
                (princ (cadar defines) out)
                (princ #"newline out)
                )
        )
    )
)
(setq defines (cdr defines))
)

; next the floating point declarations - but before these, the intermediated syntax ...

(princ "float *dyn'robot(float *Q,float *QD,float *QDD)" out)
(princ #"newline out)
(princ "-" out)
(princ #"newline out)

(setq floats float-decl)
                                    (setq i 0)
(princ "float " out)
(loop
(cond ((null floats) (return nil))
    ((= (length floats) 1)
                (princ (car floats) out))
    (t          (progn()
                (princ (car floats) out)
                (princ ", " out)
                                    (incf i)
                (if (= i 10) (progn (princ #"newline out) (setq i 0)))
                )
    )
)
(setq floats (cdr floats))
)
```

```
(princ ";" out)
(princ #"newline out)

(setq statements stmnts)
(loop
(cond ((null statements) (return nil))
    (t              (progn()
                    (princ (caar statements) out)
                    (princ " = " out)
                    (princ (cadar statements) out)
                    (princ ";" out)
                    (princ #"newline out)
                    )
    )
)
(setq statements (cdr statements))
)

(princ "~" out)
(close out)
)
```

We give an example output file used to compensate the gravity terms of the Sarcos GRLA after parametric identification.

```
#define GY -6.93567
#define GZ 6.93567
#define U1 1
#define mX1 3.7467000000000001
#define mZ1 3.7195999999999998
#define U2 1
#define mX2 5.3212000000000002
#define mZ2 78.006199999999996
#define R3 0.87880000000000003
#define U3 -1
#define mX3 -2.5465
#define mZ3 3.9049
#define L0023 -0.87880000000000003
#define U4 1
#define mX4 0.46240000000000003
#define mZ4 23.160499999999999
#define R5 0.76200000000000001
#define U5 -1
#define mX5 0.25259999999999999
#define mZ5 0.61610000000000009
#define L0025 -0.76200000000000001
#define U6 1
#define mX6 -0.1174
```

```
#define mZ6 -0.005200000000000006
#define L7 -1
#define mX7 -1.7239
#define mY7 -0.3668000000000001
float *dyn_robot(float *Q,float *QD,float *QDD)
{
float A111, A121, A123, A113, A211, A221, A223, A213, A311, A321,
A323, A313, A411, A421, A423, A413, A511, A521, A523, A513,
A611, A621, A623, A613, A711, A721, A712, A722, GAM[7], W11,
W21, W31, PWS11, PWS21, PWS31, WP11, WP21, WP31, DV111, DV121,
DV131, DV221, DV231, DV331, U111, U121, U131, U211, U221, U231,
U311, U321, U331, VP11, VP21, VP31, F11, F21, F31, N11,
N21, N31, W12, W22, W32, PWS12, PWS22, PWS32, WP12, WP22,
WP32, DV112, DV122, DV132, DV222, DV232, DV332, U112, U122, U132,
U212, U222, U232, U312, U322, U332, VP12, VP22, VP32, F12,
F22, F32, N12, N22, N32, W13, W23, W33, PWS13, PWS23,
PWS33, WP13, WP23, WP33, DV113, DV123, DV133, DV223, DV233, DV333,
U113, U123, U133, U213, U223, U233, U313, U323, U333, VP13,
VP23, VP33, F13, F23, F33, N13, N23, N33, W14, W24,
W34, PWS14, PWS24, PWS34, WP14, WP24, WP34, DV114, DV124, DV134,
DV224, DV234, DV334, U114, U124, U134, U214, U224, U234, U314,
U324, U334, VP14, VP24, VP34, F14, F24, F34, N14, N24,
N34, W15, W25, W35, PWS15, PWS25, PWS35, WP15, WP25, WP35,
DV115, DV125, DV135, DV225, DV235, DV335, U115, U125, U135, U215,
U225, U235, U315, U325, U335, VP15, VP25, VP35, F15, F25,
F35, N15, N25, N35, W16, W26, W36, PWS16, PWS26, PWS36,
WP16, WP26, WP36, DV116, DV126, DV136, DV226, DV236, DV336, U116,
U126, U136, U216, U226, U236, U316, U326, U336, VP16, VP26,
VP36, F16, F26, F36, N16, N26, N36, W17, W27, W37,
PWS17, PWS27, PWS37, WP17, WP27, WP37, DV117, DV127, DV137, DV227,
DV237, DV337, U117, U127, U137, U217, U227, U237, U317, U327,
U337, VP17, VP27, VP37, F17, F27, F37, N17, N27, N37,
E17, E27, E37, M17, M27, M37, E16, E26, E36, M16,
M26, M36, E15, E25, E35, M15, M25, M35, E14, E24,
E34, M14, M24, M34, E13, E23, E33, M13, M23, M33,
E12, E22, E32, M12, M22, M32, E11, E21, E31, M11,
M21, M31;
A111 = cos(Q[1]);
A121 = sin(Q[1]);
A123 = (-cos(Q[1]));
A113 = sin(Q[1]);
A211 = cos(Q[2]);
A221 = sin(Q[2]);
A223 = (-cos(Q[2]));
A213 = sin(Q[2]);
A311 = cos(Q[3]);
A321 = sin(Q[3]);
A323 = cos(Q[3]);
A313 = (-sin(Q[3]));
A411 = cos(Q[4]);
```

```
A421 = sin(Q[4]);
A423 = (-cos(Q[4]));
A413 = sin(Q[4]);
A511 = cos(Q[5]);
A521 = sin(Q[5]);
A523 = cos(Q[5]);
A513 = (-sin(Q[5]));
A611 = cos(Q[6]);
A621 = sin(Q[6]);
A623 = (-cos(Q[6]));
A613 = sin(Q[6]);
A711 = cos(Q[7]);
A721 = sin(Q[7]);
A712 = sin(Q[7]);
A722 = (-cos(Q[7]));
W11 = 0;
W21 = 0;
W31 = 0;
PWS11 = 0;
PWS21 = 0;
PWS31 = 0;
WP11 = ((A111*PWS11)+((A121*PWS21)));
WP21 = ((PWS31));
WP31 = ((A113*PWS11)+((A123*PWS21)));
DV111 = -W11*W11;
DV121 = W11*W21;
DV131 = W11*W31;
DV221 = -W21*W21;
DV231 = W21*W31;
DV331 = -W31*W31;
U111 = DV331+DV221;
U121 = DV121-WP31;
U131 = DV131+WP21;
U211 = DV121+WP31;
U221 = DV111+DV331;
U231 = I V231-WP11;
U311 = DV131-WP21;
U321 = DV231+WP11;
U331 = DV111+DV221;
VP11 = ((((A121*GY))));
VP21 = (((GZ)));
VP31 = ((((A123*GY))));
F11 = (((U111*mX1)+((U131*mZ1))));
F21 = (((U211*mX1)+((U231*mZ1))));
F31 = (((U311*mX1)+((U331*mZ1))));
N11 = 0;
N21 = 0;
N31 = 0;
W12 = ((A211*(W11))+((A221*(W21))));
W22 = (((W31)));
```

```
W32 = ((A213*(W11))+((A223*(W21))));
PWS12 = (WP11);
PWS22 = (WP21);
PWS32 = (WP31);
WP12 = ((A211*PWS12)+((A221*PWS22)));
WP22 = ((PWS32));
WP32 = ((A213*PWS12)+((A223*PWS22)));
DV112 = -W12*W12;
DV122 = W12*W22;
DV132 = W12*W32;
DV222 = -W22*W22;
DV232 = W22*W32;
DV332 = -W32*W32;
U112 = DV332+DV222;
U122 = DV122-WP32;
U132 = DV132+WP22;
U212 = DV122+WP32;
U222 = DV112+DV332;
U232 = DV232-WP12;
U312 = DV132-WP22;
U322 = DV232+WP12;
U332 = DV112+DV222;
VP12 = (((A211*VP11)+((A221*VP21))));
VP22 = (((VP31)));
VP32 = (((A213*VP11)+((A223*VP21))));
F12 = (((U112*mX2)+((U132*mZ2))));
F22 = (((U212*mX2)+((U232*mZ2))));
F32 = (((U312*mX2)+((U332*mZ2))));
N12 = 0;
N22 = 0;
N32 = 0;
W13 = ((A311*(W12))+((A321*(W22))));
W23 = (((-(W32))));
W33 = ((A313*(W12))+((A323*(W22))));
PWS13 = (WP12);
PWS23 = (WP22);
PWS33 = (WP32);
WP13 = ((A311*PWS13)+((A321*PWS23)));
WP23 = (((-PWS33)));
WP33 = ((A313*PWS13)+((A323*PWS23)));
DV113 = -W13*W13;
DV123 = W13*W23;
DV133 = W13*W33;
DV223 = -W23*W23;
DV233 = W23*W33;
DV333 = -W33*W33;
U113 = DV333+DV223;
U123 = DV123-WP33;
U133 = DV133+WP23;
U213 = DV123+WP33;
```

```
U223 = DV113+DV333;
U233 = DV233-WP13;
U313 = DV133-WP23;
U323 = DV233+WP13;
U333 = DV113+DV223;
VP13 = (((A311*VP12)+((A321*VP22)))+((((U123*L0023))));
VP23 = ((((-VP32)))+((((U223*L0023))));
VP33 = (((A313*VP12)+((A323*VP22)))+((((U323*L0023))));
F13 = ((((U113*mX3)+((U133*mZ3))));
F23 = ((((U213*mX3)+((U233*mZ3))));
F33 = ((((U313*mX3)+((U333*mZ3))));
N13 = 0;
N23 = 0;
N33 = 0;
W14 = ((A411*(W13))+((A421*(W23))));
W24 = (((W33)));
W34 = ((A413*(W13))+((A423*(W23))));
PWS14 = (WP13);
PWS24 = (WP23);
PWS34 = (WP33);
WP14 = ((A411*PWS14)+((A421*PWS24)));
WP24 = ((PWS34));
WP34 = ((A413*PWS14)+((A423*PWS24)));
DV114 = -W14*W14;
DV124 = W14*W24;
DV134 = W14*W34;
DV224 = -W24*W24;
DV234 = W24*W34;
DV334 = -W34*W34;
U114 = DV334+DV224;
U124 = DV124-WP34;
U134 = DV134+WP24;
U214 = DV124+WP34;
U224 = DV114+DV334;
U234 = DV234-WP14;
U314 = DV134-WP24;
U324 = DV234+WP14;
U334 = DV114+DV224;
VP14 = (((A411*VP13)+((A421*VP23))));
VP24 = (((VP33)));
VP34 = (((A413*VP13)+((A423*VP23))));
F14 = ((((U114*mX4)+((U134*mZ4))));
F24 = ((((U214*mX4)+((U234*mZ4))));
F34 = ((((U314*mX4)+((U334*mZ4))));
N14 = 0;
N24 = 0;
N34 = 0;
W15 = ((A511*(W14))+((A521*(W24))));
W25 = (((-(W34))));
W35 = ((A513*(W14))+((A523*(W24))));
```

```
PWS15 = (WP14);
PWS25 = (WP24);
PWS35 = (WP34);
WP15 = ((A511*PWS15)+((A521*PWS25)));
WP25 = (((-PWS35)));
WP35 = ((A513*PWS15)+((A523*PWS25)));
DV115 = -W15*W15;
DV125 = W15*W25;
DV135 = W15*W35;
DV225 = -W25*W25,
DV235 = W25*W35;
DV335 = -W35*W35;
U115 = DV335+DV225;
U125 = DV125-WP35;
U135 = DV135+WP25;
U215 = DV125+WP35;
U225 = DV115+DV335;
U235 = DV235-WP15;
U315 = DV135-WP25;
U325 = DV235+WP15;
U335 = DV115+DV225;
VP15 = (((A511*VP14)+((A521*VP24)))+(((U125*L0025))));
VP25 = ((((-VP34)))+(((U225*L0025))));
VP35 = (((A513*VP14)+((A523*VP24)))+(((U325*L0025))));
F15 = (((U115*mX5)+((U135*mZ5))));
F25 = (((U215*mX5)+((U235*mZ5))));
F35 = (((U315*mX5)+((U335*mZ5))));
N15 = 0;
N25 = 0;
N35 = 0;
W16 = ((A611*(W15))+((A621*(W25))));
W26 = (((W35)));
W36 = ((A613*(W15))+((A623*(W25))));
PWS16 = (WP15);
PWS26 = (WP25);
PWS36 = (WP35);
WP16 = ((A611*PWS16)+((A621*PWS26)));
WP26 = ((PWS36));
WP36 = ((A613*PWS16)+((A623*PWS26)));
DV116 = -W16*W16;
DV126 = W16*W26;
DV136 = W16*W36;
DV226 = -W26*W26;
DV236 = W26*W36;
DV336 = -W36*W36;
U116 = DV336+DV226;
U126 = DV126-WP36;
U136 = DV136+WP26;
U216 = DV126+WP36;
U226 = DV116+DV336;
```

```
U236 = DV236-WP16;
U316 = DV136-WP26;
U326 = DV236+WP16;
U336 = DV116+DV226;
VP16 = (((A611*VP15)+((A621*VP25))));
VP26 = (((VP35)));
VP36 = (((A613*VP15)+((A623*VP25))));
F16 = (((U116*mX6)+((U136*mZ6))));
F26 = (((U216*mX6)+((U236*mZ6))));
F36 = (((U316*mX6)+((U336*mZ6))));
N16 = 0;
N26 = 0;
N36 = 0;
W17 = ((A711*(W16))+((A721*(W26))));
W27 = ((A712*(W16))+((A722*(W26))));
W37 = (((-(W36))));
PWS17 = (WP16);
PWS27 = (WP26);
PWS37 = (WP36);
WP17 = ((A711*PWS17)+((A721*PWS27)));
WP27 = ((A712*PWS17)+((A722*PWS27)));
WP37 = (((-PWS37)));
DV117 = -W17*W17;
DV127 = W17*W27;
DV137 = W17*W37;
DV227 = -W27*W27;
DV237 = W27*W37;
DV337 = -W37*W37;
U117 = DV337+DV227;
U127 = DV127-WP37;
U137 = DV137+WP27;
U217 = DV127+WP37;
U227 = DV117+DV337;
U237 = DV237-WP17;
U317 = DV137-WP27;
U327 = DV237+WP17;
U337 = DV117+DV227;
VP17 = (((A711*VP16)+((A721*VP26))));
VP27 = (((A712*VP16)+((A722*VP26))));
VP37 = ((((-VP36))));
F17 = (((U117*mX7)+((U127*mY7))));
F27 = (((U217*mX7)+((U227*mY7))));
F37 = (((U317*mX7)+((U327*mY7))));
N17 = 0;
N27 = 0;
N37 = 0;
E17 = F17;
E27 = F27;
E37 = F37;
M17 = (N17+((mY7*VP37)));
```

```
M27 = (N27+((-(VP37*mX7))));
M37 = (N37+(((mX7*VP27)-(VP17*mY7))));
GAM[7] = ((((-M37)))+(0));
E16 = (((A711*E17)+((A712*E27)))+F16);
E26 = (((A721*E17)+((A722*E27)))+F26);
E36 = ((((-E37)))+F36);
M16 = ((N16+((-(VP26*mZ6))))+((A711*M17)+((A712*M27))));
M26 = ((N26+(((mZ6*VP16)-(VP36*mX6))))+((A721*M17)+((A722*M27))));
M36 = ((N36+((mX6*VP26)))+(((-M37))));
GAM[6] = (((M26))+(0));
E15 = (((A611*E16)+((A613*E36)))+F15);
E25 = (((A621*E16)+((A623*E36)))+F25);
E35 = (((E26))+F35);
M15 = ((N15+((L0025*E35)+(-(VP25*mZ5))))+((A611*M16)+((A613*M36))));
M25 = ((N25+(((mZ5*VP15)-(VP35*mX5))))+((A621*M16)+((A623*M36))));
M35 = ((N35+((-(E15*L0025))+(mX5*VP25)))+((M26)));
GAM[5] = ((((-M25)))+(0));
E14 = (((A511*E15)+((A513*E35)))+F14);
E24 = (((A521*E15)+((A523*E35)))+F24);
E34 = ((((-E25)))+F34);
M14 = ((N14+((-(VP24*mZ4))))+((A511*M15)+((A513*M35))));
M24 = ((N24+(((mZ4*VP14)-(VP34*mX4))))+((A521*M15)+((A523*M35))));
M34 = ((N34+((mX4*VP24)))+(((-M25))));
GAM[4] = (((M24))+(0));
E13 = (((A411*E14)+((A413*E34)))+F13);
E23 = (((A421*E14)+((A423*E34)))+F25);
E33 = (((E24))+F33);
M13 = ((N13+((L0023*E33)+(-(VP23*mZ3))))+((A411*M14)+((A413*M34))));
M23 = ((N23+(((mZ3*VP13)-(VP33*mX3))))+((A421*M14)+((A423*M34))));
M33 = ((N33+((-(E13*L0023))+(mX3*VP23)))+((M24)));
GAM[3] = ((((-M23)))+(0));
E12 = (((A311*E13)+((A313*E33)))+F12);
E22 = (((A321*E13)+((A323*E33)))+F22);
E32 = ((((-E23)))+F32);
M12 = ((N12+((-(VP22*mZ2))))+((A311*M13)+((A313*M33))));
M22 = ((N22+(((mZ2*VP12)-(VP32*mX2))))+((A321*M13)+((A323*M33))));
M32 = ((N32+((mX2*VP22)))+(((-M23))));
GAM[2] = (((M22))+(0));
E11 = (((A211*E12)+((A213*E32)))+F11);
E21 = (((A221*E12)+((A223*E32)))+F21);
E31 = (((E22))+F31);
M11 = ((N11+((-(VP21*mZ1))))+((A211*M12)+((A213*M32))));
M21 = ((N21+(((mZ1*VP11)-(VP31*mX1))))+((A221*M12)+((A223*M32))));
M31 = ((N31+((mX1*VP21)))+((M22)));
GAM[1] = (((M21))+(0));
```

Appendix C

Multistage optimization code

The algorithm, initially applied by Armstrong to the dynamic estimation problem, has been implemented here in Matlab. The partial derivatives were evaluated explicitly by functions *diff_th1_D1* etc. The user would have to provide these functions in Matlab before running the program.

```
load init'traj   % the initial acceleration vector
u=init'traj;
x0=[0;0;-pi;0;0;0];
K=length(init'traj);
Ts=0.005;
A=[eye(3),Ts*eye(3);zeros(3,3),eye(3)];
B=[zeros(3,3);Ts*eye(3)];

for m=1:M,

X=ltitr(A,B,u',x0);
x=X';

th1=x(1,:)';
th2=x(2,:)';
th3=x(3,:)';
th1d=x(4,:)';
th2d=x(5,:)';
th3d=x(6,:)';
th1dd=u(1,:)';
th2dd=u(2,:)';
th3dd=u(3,:)';
```

```
[D1,D2,D3]=CALC_D(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);

R1=D1'*D1;
R2=D2'*D2;
R3=D3'*D3;
R=(1/K)*(R1+R2+R3);
J=1/min(eig(R));

for i=1:11,
for j=1:11,
incr=zeros(11,11);
incr(i,j)=0.1;
alpha(11*(i-1)+j) = 10*( (1/min(eig(R+incr))) - J );
end
end

alpha=real(alpha);

% column matrices

d'th1'D1 = diff'th1'D1(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th1'D2 = diff'th1'D2(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th1'D3 = diff'th1'D3(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);

d'th2'D1 = diff'th2'D1(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th2'D2 = diff'th2'D2(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th2'D3 = diff'th2'D3(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);

d'th3'D1 = diff'th3'D1(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th3'D2 = diff'th3'D2(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th3'D3 = diff'th3'D3(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);

d'th1d'D1 = diff'th1d'D1(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th1d'D2 = diff'th1d'D2(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th1d'D3 = diff'th1d'D3(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);

d'th2d'D1 = diff'th2d'D1(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th2d'D2 = diff'th2d'D2(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th2d'D3 = diff'th2d'D3(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);

d'th3d'D1 = diff'th3d'D1(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th3d'D2 = diff'th3d'D2(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th3d'D3 = diff'th3d'D3(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);

d'th1dd'D1 = diff'th1dd'D1(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th1dd'D2 = diff'th1dd'D2(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th1dd'D3 = diff'th1dd'D3(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);

d'th2dd'D1 = diff'th2dd'D1(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th2dd'D2 = diff'th2dd'D2(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
```

```
d'th2dd'D3 = diff'th2dd'D3(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd),

d'th3dd'D1 = diff'th3dd'D1(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th3dd'D2 = diff'th3dd'D2(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);
d'th3dd'D3 = diff'th3dd'D3(th1,th2,th3,th1d,th2d,th3d,th1dd,th2dd,th3dd);

for k=1:K,

del'x1'F(k) = alpha  * (kron(d'th1'D1(k,:)',D1(k,:)')+kron(D1(k,:)',d'th1'D1(k,:)')+  ...
                kron(d'th1'D2(k,:)',D2(k,:)')+kron(D2(k,:)',d'th1'D2(k,:)')+ ...
                kron(d'th1'D3(k,:)',D3(k,:)')+kron(D3(k,:)',d'th1'D3(k,:)'));

del'x2'F(k) = alpha  * (kron(d'th2'D1(k,:)',D1(k,:)')+kron(D1(k,:)',d'th2'D1(k,:)')+ ...
                kron(d'th2'D2(k,:)',D2(k,:)')+kron(D2(k,:)',d'th2'D2(k,:)')+ ...
                kron(d'th2'D3(k,:)',D3(k,:)')+kron(D3(k,:)',d'th2'D3(k,:)'));

del'x3'F(k) = alpha  * (kron(d'th3'D1(k,:)',D1(k,:)')+kron(D1(k,:)',d'th3'D1(k,:)')+ ...
                kron(d'th3'D2(k,:)',D2(k,:)')+kron(D2(k,:)',d'th3'D2(k,:)')+ ...
                kron(d'th3'D3(k,:)',D3(k,:)')+kron(D3(k,:)',d'th3'D3(k,:)'));

del'x4'F(k) = alpha  * (kron(d'th1d'D1(k,:)',D1(k,:)')+kron(D1(k,:)',d'th1d'D1(k,:)')+ ...
                kron(d'th1d'D2(k,:)',D2(k,:)')+kron(D2(k,:)',d'th1d'D2(k,:)')+ ...
                kron(d'th1d'D3(k,:)',D3(k,:)')+kron(D3(k,:)',d'th1d'D3(k,:)'));

del'x5'F(k) = alpha  * (kron(d'th2d'D1(k,:)',D1(k,:)')+kron(D1(k,:)',d'th2d'D1(k,:)')+ ...
                kron(d'th2d'D2(k,:)',D2(k,:)')+kron(D2(k,:)',d'th2d'D2(k,:)')+ ...
                kron(d'th2d'D3(k,:)',D3(k,:)')+kron(D3(k,:)',d'th2d'D3(k,:)'));

del'x6'F(k) = alpha  * (kron(d'th3d'D1(k,:)',D1(k,:)')+kron(D1(k,:)',d'th3d'D1(k,:)')+ ...
                kron(d'th3d'D2(k,:)',D2(k,:)')+kron(D2(k,:)',d'th3d'D2(k,:)')+ ...
                kron(d'th3d'D3(k,:)',D3(k,:)')+kron(D3(k,:)',d'th3d'D3(k,:)'));

del'u1'F(k) = alpha  * (kron(d'th1dd'D1(k,:)',D1(k,:)')+kron(D1(k,:)',d'th1dd'D1(k,:)')+ ...
                kron(d'th1dd'D2(k,:)',D2(k,:)')+kron(D2(k,:)',d'th1dd'D2(k,:)')+ ...
                kron(d'th1dd'D3(k,:)',D3(k,:)')+kron(D3(k,:)',d'th1dd'D3(k,:)'));

del'u2'F(k) = alpha  * (kron(d'th2dd'D1(k,:)',D1(k,:)')+kron(D1(k,:)',d'th2dd'D1(k,:)')+ ...
                kron(d'th2dd'D2(k,:)',D2(k,:)')+kron(D2(k,:)',d'th2dd'D2(k,:)')+ ...
                kron(d'th2dd'D3(k,:)',D3(k,:)')+kron(D3(k,:)',d'th2dd'D3(k,:)'));

del'u3'F(k) = alpha  * (kron(d'th3dd'D1(k,:)',D1(k,:)')+kron(D1(k,:)',d'th3dd'D1(k,:)')+ ...
                kron(d'th3dd'D2(k,:)',D2(k,:)')+kron(D2(k,:)',d'th3dd'D2(k,:)')+ ...
                kron(d'th3dd'D3(k,:)',D3(k,:)')+kron(D3(k,:)',d'th3dd'D3(k,:)'));

end

del'x'F = [del'x1'F;del'x2'F;del'x3'F;del'x4'F;del'x5'F;del'x6'F];
del'u'F = [del'u1'F;del'u2'F;del'u3'F];

del'x'F=flipud(del'x'F');
```

```
Lambda0=zeros(6,1);
Lambda=lutr(A,eye(6),del'x'F,Lambda0);

% Q assignment according to Brysons laws ... norm(Q) = (accn.limit).(del'u'F)/(Tolerable'J)^2
q=.4*(norm(del'u'F,'fro'))/(3*K*(.3^2));
Q=[q 0 0;0 q 0;0 0 q];


del'u'J = del'u'F ...
  + 2*(u'*Q)' - (Lambda*B)';

mu'cv = .1/max([max(abs(del'u'J(1,:))),max(abs(del'u'J(2,:))), ...
max(abs(del'u'J(3,:)))]);

u = u - mu'cv*del'u'J;

if m==1,
save -ascii u1 u
elseif m==2,
save -ascii u2 u
elseif m==3,
save -ascii u3 u
elseif m==4,
save -ascii u4 u
elseif m==5,
save -ascii u5 u
elseif m==6,
save -ascii u6 u
elseif m==7,
save -ascii u7 u
elseif m==8,
save -ascii u8 u
elseif m==9,
save -ascii u9 u
elseif m==10,
save -ascii u10 u

end
end
```

Bibliography

[1] Craig J.J. *Adaptive Control of Mechanical Manipulators.* Addison-Wesley Publishing Co., 1988.

[2] M. Renaud. An efficient iterative analytical procedure for obtaining a robot manipulator dynamic model. In *Robotic Research: The First International Symposium.* Cambridge MA:MIT Press, 1984.

[3] R.P.Paul J.Luh, Y.S.Walker. On-line compuatational scheme for mechanical manipulators. *Journal Dyn.Sys.Meas.Contr.*, AC-25, 1980.

[4] C.Atkeson C.An J.M.Hollerbach. Estimation of inertial parametersof manipulator loads and links. *Int. J. Rob.Res.*, 5(3), 1986.

[5] A.Izaguirre et al. A new computational structure for real-time dynamics. *Int. J. of Rob. Res.*, 11(4), 1992.

[6] Sheu S. and Walker M. Basis sets for manipulator inertial parameters. *IEEE Intl. Conf. on Robotics and Automation*, 1989.

[7] Gautier M. and Khalil W. Identification of the minimum inertial parameters of robots. *IEEE Intl. Conf. on Robotics and Automation*, 1989.

[8] H.Mayeda K.Yoshida K.Osuka. Base parameters of manipulator dynamic models. *IEEE Transactions on Robotics and Automation*, 6(3), 1990.

[9] Lawson.Ch.L. Hanson.R.J. *Solving Least Squares Problems.* Englewood Cliffs: Prentice-Hall, 1974.

[10] B.Armstrong O.Khatib and J.Burdick. The explicit dynamics model and inertial parameters of the puma 560 arm. *Proc. 1986 IEEE Int. Conf. Robot. Automat.*, 1986.

[11] J.M. Mendel. *Discrete Techniques of Parameter Estimation.* New York: Marcel Dekker, Inc., 1973.

[12] Ljung L. *System Identification: Theory for the User.* Prentice Hall Inc. Englewood Cliffs N.J., 1987.

[13] Widrow B. and Stearns S.D. *Adaptive Signal Processing.* Prentice Hall Inc. Englewood Cliffs N.J., 1985.

[14] B.Armstrong. On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics. *Intl.J.Rob.Res.*, 8(6), 1989.

[15] Sastry S.S. Model-reference adaptive control—stability, parameter convergence and robustness. *IMAJ.Math.Control Inform.*, 1, 1984.

[16] A.E.Bryson and Y.Ho. *Applied Optimal Control.* New York:Hemisphere Publishing Co., 1975.

[17] Gautier M. and Khalil W. Exciting trajectories for the identification of base inertial parameters of robots. *The Intl. Journal of Robotics Research*, 11(4), 1992.

[18] B.D.O.Anderson J.B.Moore. *Optimal Filtering*. Prentice-Hall, 1979.

[19] A.H.Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[20] Atkeson C. Hollerbach J. An, C. Experimental determination of the effect of feedforward control on trajectory tracking errors. In *Proc. 1986 IEEE Int. Conf. Robot. Automat.*, 1986.

[21] Valavanis K. Leahy M. and Saridis G. The effects of dynamic models on robot control. In *Proc. 1986 IEEE Int. Conf. Robot. Automat.*, 1986.

[22] Egeland O. On the robustness of the computed torque technique in manipulator control. In *1986 IEEE Int. Conf. on Robotics and Automation*, 1986.

[23] P.E. Belanger. Estimation of angular veocity and acceleration from shaft encoder measurements. Technical Report TR-CIM-91-1, McGill Research Centre for Intelligent Machines, 1991.

[24] Schroer. Theory of kinematic modelling and numerical procedures for robot calibration. In *Robot Calibration*. Chapman and Hall, 1993.

[25] M.Gautier W.Khalil, J.F.Kleinfinger. Reducing the computational burden of the dynamic models of robots. *IEEE Transactions on Robotics Research*, 1986.

[26] Anthony Topper. A computing architecture for a multiple robot controller. *Masters Thesis, Dept. of Elec. Engg., McGill University*, 1991.

[27] Wredenhagen G. F. Performance factors for fine end-point position control in robots. *PH.D. Thesis Dept. of Elec. Engg. McGill University*, 1994.

[28] Jazwinski A.H. *Stochastic processes and filtering theory.* New York: Academic Press, 1970.