Terminal sliding mode control for rigid robotic manipulators with uncertain dynamics

Nicola Ritter
Edith Cowan University

Terminal Sliding Mode Control for Rigid Robotic Manipulators with Uncertain Dynamics

Nicola Ritter
Grad.Dip.Comp. Dip.Ed. B.Sc. M.A.C.S.

A thesis submitted in partial fulfilment of the requirements for the award of Master of Science at the Faculty of Science, Technology and Engineering, Edith Cowan University.

November 8, 1995

Abstract

This thesis presents two new adaptive control laws that use the terminal sliding mode technique for the tracking problem of rigid robotic manipulators with nonlinearities, dynamic couplings and uncertain parameters.

The first law provides a robust scheme which uses several properties of rigid robotic manipulators and adaptively adjusts seven uncertain parameter bounds. The law ensures finite time error convergence to the system origin and is simple to implement. The second law treats the manipulator as a partially known system. The known dynamics are used to build a nominal control law and the effects of unknown system dynamics are compensated for by use of a sliding mode compensator. The resulting control law is robust, asymptotically convergent, has finite time convergence to the sliding mode and allows for bounded external disturbances. It is easy to implement and requires no bounds on system parameters, adaptively adjusting only three bounds on system uncertainties.

Both laws are extended to include a reduction of chattering by use of the boundary layer technique. They are tested via application to a two-link robot simulated using MatLab.

Declaration

I certify that this thesis does not incorporate, without acknowledgment, any material previously submitted for a degree or diploma in any institution of higher education and that, to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where due reference is made in the text.

N. J. Ritter

November 8, 1995

Acknowledgments

I thank my supervisor Dr. Zhihong Man for his help, direction and encouragement and my Chairperson of Department Dr. Jim Millar for his provision of an excellent working environment.

I would also like to thank my husband Dr. James Cooper for his unstinting support, morale boosting and advice. My children, Steven and Christine, deserve thanks for their patience during the long haul and for the times that I was absent when I should not have been.

There are many others who have provided help and encouragement when I needed it. In particular Dr. Paul Maj, Andrew Mehnert and my sister Dr Leonora Ritter are deserving of mention.

Contents

List of Figures

List of Tables

1 Introduction

This introduction is divided up into several sections. Sections 1.1 and 1.2 give definitions of symbols and concepts used, but not defined elsewhere in this thesis. Section 1.3 introduces the robot control problem.

1.1 Symbols

The following symbols will be used throughout this thesis.

$$\mathrm{sgn}(s) \triangleq \begin{cases} 1 & s > 0 \\ 0 & s = 0 \\ -1 & s < 0 \end{cases}$$

$\mathbf{0}$	$\triangleq$	the matrix or vector where all elements are zero
$\mathbf{I}$	$\triangleq$	the identity matrix
$\mathbf{R}^{n}$	$\triangleq$	Euclidean n-space: the set of all ordered n-tuples where n is a positive number and an n-tuple is a sequence of n real numbers $(a1, a2, ..., a_n)$
$\mathbf{R}^{m \times n}$	$\triangleq$	Euclidean $m \times n$ space.
x	$\triangleq$	a vector
X	$\triangleq$	a matrix
x^T, X^T	$\triangleq$	the transposes of x, X

$\det(A)$	$\triangleq$	the determinant of matrix X
	$\triangleq$	$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$
	$=$	$a_{11}a_{22} - a_{12}a_{21}$
$\lambda(X)$	$\triangleq$	an eigenvalue of matrix X (see definition on page 3)
$\lVert x\rVert$	$\triangleq$	the Euclidean norm of x $\lVert x\rVert = \left(\sum_{j=1}^{m} x_j^2\right)^{\frac{1}{2}}$ (note: $\lVert x\rVert^2 = x^T x$)
$\lVert X\rVert$	$\triangleq$	the induced Euclidean norm of X (Spong and Vidyasagar 1987) $\lVert X\rVert = (\lambda_{max}(X^T X))^{\frac{1}{2}}$
$\dot{x}, \dot{X}$	$\triangleq$	the first order derivatives of x, X: the derivative of each element of the vector or matrix
$\ddot{x}, \ddot{X}$	$\triangleq$	the second order derivatives of x, X
$X = \operatorname{diag}(x_i)$	$\triangleq$	$\begin{bmatrix} x_1 & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & x_i & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & x_n \end{bmatrix}$ (all elements outside the diagonal are zero)
$y^{(n)}$	$\triangleq$	the nth order derivative of the variable y

1.2 Definitions

The following terms are used in this thesis.

A **symmetric matrix** is one where the matrix is equal to its transpose:

$$M = M^T = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1i} \\ m_{12} & m_{22} & & m_{2i} \\ \vdots & & \ddots & \vdots \\ m_{1i} & m_{2i} & \cdots & m_{ii} \end{bmatrix}$$

If A is a matrix then x is an **eigenvector** of A

$$\text{if} \quad x \neq 0 \quad \text{and} \quad \exists\, \lambda \in \mathbf{R} : Ax^T = \lambda x^T \tag{1.1}$$

(Damiano and Little 1988).

Note:

- The scalar λ is called an **eigenvalue** of A.
- If A is an $n \times n$ matrix, then there are exactly n eigenvectors of A (they may not all be distinct).
- $\lambda_{min}(A)$ is the minimum eigenvalue of the matrix A, and $\lambda_{max}(A)$ the maximum eigenvalue.
- $Re\lambda(A)$ is defined as the real part of the eigenvalue λ of matrix A.
- The eigenvalues of a matrix A are roots of the following equation:

$$\det(A - \lambda\mathbf{I}) = 0$$

A **positive definite** matrix is one where all its eigenvalues are positive.

A **reference model** of a system provides a desired output response for the system to follow. Figures 1.1 and 1.2 show block diagrams of a robot and its reference model.

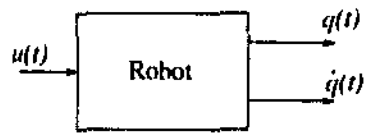

Figure 1.1: An n-link rigid robotic manipulator, where $u(t)$ is the control input vector to the robot and $q(t)$ is the vector of joint values (Man, Paplinski, and Wu 1994).

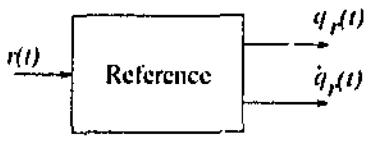

Figure 1.2: A reference model of a robot manipulator, where $r(t)$ is the reference input vector and $q_r(t)$ is the vector of reference signals for the joints' angular signal vector q to follow (Man, Paplinski, and Wu 1994).

Tracking is the task of making a rigid robotic manipulator follow a required trajectory, given by a reference model.

Feedback is the current output of the system, fed back into the control of the system to allow that control to be adjusted for current errors.

The **set point regulation problem** can be described as follows. Given an initial position and velocity of the manipulator, p_0 and v_0, and a desired position and velocity, p_d and v_d, find a control law such that $p(t) \to p_d$ and $v(t) \to 0$ (Young 1978).

Perturbations or disturbances are a combination of input disturbances, friction, actuator dynamics and joint flexibility (Ortega and Spong 1988).

A **Lyapunov function** can be defined as follows:
Let Ω be an open region in $\mathbf{R}^m$ containing the origin. A function V is a Lyapunov function on Ω if and only if:

1. $V(x)$ has a continuous derivative.
2. $V(0) = 0$.
3. $V(x) > 0 \quad \text{for} \quad x \neq 0$.

(Schilling 1990).

A state x_e is an **equilibrium state** (or equilibrium point) of the system if once the system state x is equal to x_e, it remains equal to x_e for all future time.

Stability in the sense of Lyapunov: The origin is stable for a system s whose state is $x(t)$, if $\forall\epsilon > 0$, $\exists\delta > 0$, such that $\|x(t_0)\| < \delta \;\Rightarrow\; \|x(t_0)\| < \epsilon$, $\forall t \geqslant t_0$. Further, it can be considered to be asymptotically stable if, in addition to the above, $x(t) \to 0$ as $t \to \infty$ (Stonier nd). Diagrams illustrating stability can be found in figure 1.3.

Lyapunov's Second Method: Let $x_e = 0$ be an equilibrium point of a system S associated with a constant input $r(t)$, and let V be a Lyapunov function on the domain Ω where $\Omega = x : V(x) < \rho$ for some $\rho > 0$. Then x_e is asymptotically stable with domain of attraction Ω, if, along solutions of S:

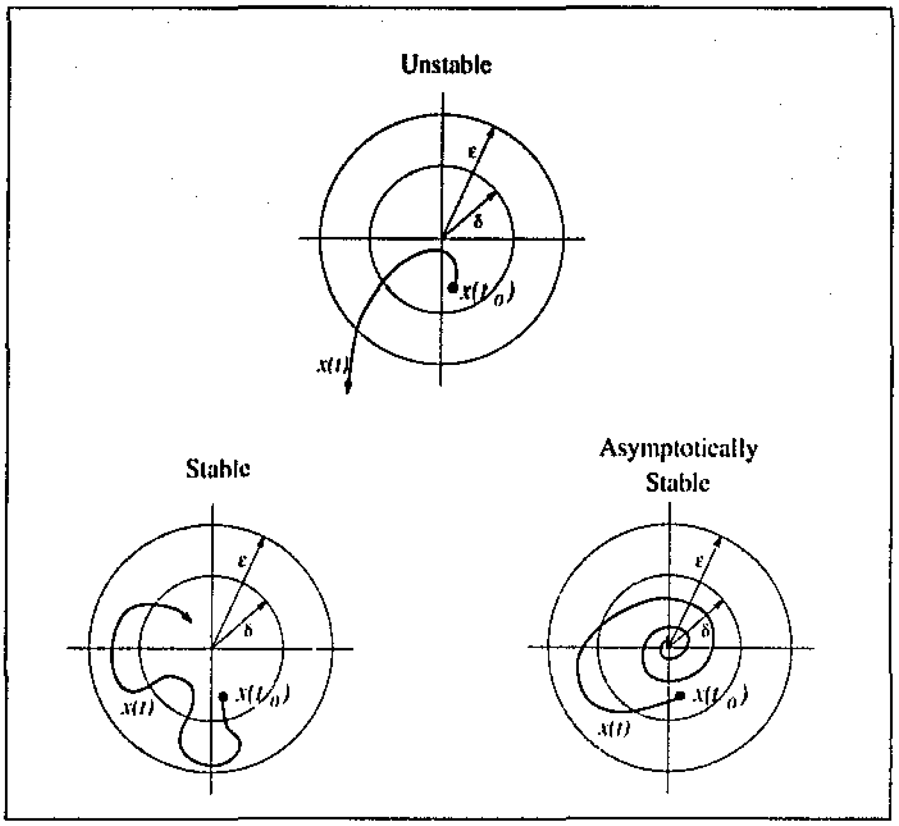

Figure 1.3: Stability in the sense of Lyapunov (Stonier nd).

1. $\dot{V}(x(t)) \leqslant 0$
2. $\dot{V}(x(t)) = 0 \Rightarrow x(t) = 0$

(Schilling 1990)

Robustness can be defined as the ability of a controller to deal with changes in parameters (Grimm, Becker, and Frank 1988) (such as inertia, load mass, etc.) and small perturbations (Ortega and Spong 1988) without the system becoming unstable.

Robust controllers are those that provide a 'good enough' control for the controlled system with uncertainties in the system model (Abdallah, Dawson, Dorato, and Jamishidi 1991).

Servo-mechanism theory is based on the internal model principle which states that a controller must have information about the rest of its environment if it is going to be robust.

Globally convergent controllers are those that achieve tracking for all required trajectories and for all different initial conditions (Ortega and Spong 1988).

Adaptive controllers are those that attempt to adjust themselves dynamically so as to continually improve the accuracy of the control and eventually attain zero error (Abdallah, Dawson, Dorato, and Jamishidi 1991).

A **nominal system** is one that only takes into account known parts of the system.

A **nominal controller** is one designed to control a nominal system.

1.3 The Robot Control Problem

The robot control problem can be described as the task of designing a control law that will produce accurate tracking. A thorough description of conventional methods of controller design for tracking can be found in Luh (1983), which details fully, with examples, the major design elements of traditional robotics control. The 'simplest' way of achieving this control for tracking is by using the inverse kinematic expressions to calculate the joint-space trajectory from the tool-configuration trajectory. The derivative of this can then be found which gives the velocity at which each joint must be driven (Schilling 1990). One disadvantage of this method is that it must be possible to control the speed of the joints, which is not always possible. Another problem is that it is a computationally intensive method and too time consuming for practical use.

A more useful approach to control is to regulate the torque being applied to each joint. This has its own difficulties due to the complex and non-linear nature of the equation of motion of the arm, which are affected by inertia, gravity, centrifugal and Coriolis forces. This can be seen in the standard expression of motion of a rigid robotic manipulator :

$$u(t) = M(q)\ddot{q} + f(q, \dot{q})\dot{q} + g(q) \tag{1.2}$$

where $q(t) \in \mathbf{R}^n$ is the vector of joint angular positions, n is the number of joints in the manipulator, $u(t) \in \mathbf{R}^n$ is the input vector of joint torques, $M(q) \in \mathbf{R}^{n \times n}$ is a symmetric positive-definite inertia matrix, $f(q, \dot{q}) \in \mathbf{R}^n$ is the vector of combined centrifugal and Coriolis torques and $g(q) \in \mathbf{R}^n$ is the vector of torques due to gravity. A detailed description of the development of this expression can be found in Schilling (1990).

To complicate matters further, these elements (M, f and g) are influenced by external disturbances, the current manipulator load, the interactions of joints upon each other (which increase as the speed of the manipulator increases (Bailey and Arapostathis 1987)) and the magnitude of static and dynamic frictional forces. For this reason most uses of robotic systems to date have been for well-defined repetitive tasks (such as welding) that do not require high speed. A thorough discussion of the problems and requirements of robotic tracking control laws can be found in Qu, Dawson, and Dorsey (1992).

Early work to deal with these problems required detailed models of the system as well as knowledge of the manipulator load and were costly to implement. Later work relied on the elimination of interactions and nonlinearities when the manipulator was designed, however this proved to be both costly and limiting (Morgan and Ozguner 1985). Freund (1982) solves the control problem by direct decoupling design methods for particular classes of robots, however this also relies on having a detailed model of the system.

More successful approaches (for example Ioannou and Tsakalis (1986), Singh (1986) and Spong and Ortega (1990)) use feedback loops and adaptive mechanisms that adjust the control law as the robot moves. Changes in parameters and external disturbances are passed back to the control law and allow it to change dynamically. Figure 1.4 shows a block diagram of an adaptive control mechanism.

One disadvantage of this kind of system is that the tracking error cannot be guaranteed to converge to zero (Man and Palaniswami 1994), although the boundedness (Singh 1986) and the stability (Ioannou and Tsakalis 1986) of the system can be guaranteed. A second problem is the time it takes for the adaption to take effect: there is a period at the start of the robot's motion where the tracking lacks precision (Kosuge and Furuta 1988). In later work the term adaptive has come to mean controllers that specifically estimate and update the values of controller parameters (Ortega and Spong 1988), such as the bounds on the inertia matrix, and the Coriolis, centrifugal and gravitational forces.

More recently much research has gone into the field of variable structure systems

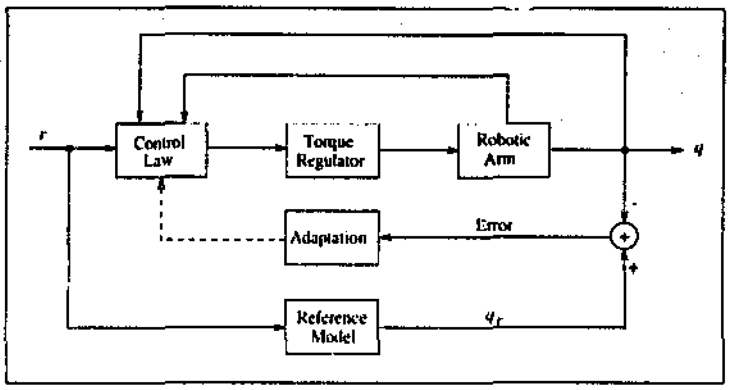

Figure 1.4: Block diagram of an adaptive control mechanism. (From Shilling, 1990, p. 290)

(VSS), which do not require exact modeling, nor prior information about the load (Morgan and Ozguner 1985). VSS are those systems where there is a discontinuity in the feedback control, which is altered in nature as the state moves across a hyperplane (or manifold) of discontinuity. Thus it can be considered as being made up of several subsystems, each of which operate in their own separate sections of subspace (Dorling and Zinober 1986). An example of a VSS can be seen in figure 1.5, which shows the two substructures as well as the combined structure. An advantage of this kind of system is that, whilst each of the individual states may not be asymptotically stable, the combination can be (Utkin 1977) and the combined or *equivalent* system is different from either of its parts and is of a lower order of complexity (Dorling and Zinober 1986).

When the subsystems are designed so that all movement is directed toward the discontinuity manifold there develops a *sliding mode*, where the system crosses and re-crosses this boundary, as it slides towards the origin. Figure 1.6 shows a VSS in two dimensions and its path as it traverses the sliding mode. Thus the motion

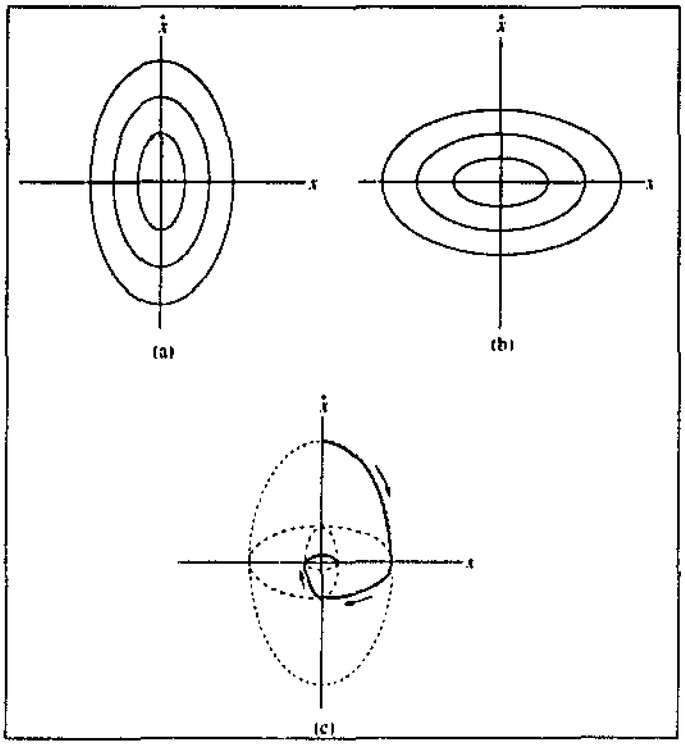

Figure 1.5: A variable structure system. Figures (a) and (b) show the two separate systems and figure (c) shows the combined system (Utkin 1977).

of a VSS can be considered in two parts, a *reaching* phase which provides a rapid convergence to the switching hyperplane or manifold and the sliding mode where a force is applied to the system if it starts to deviate from the manifold, thereby ensuring its return to it (Bailey and Arapostathis 1987). A terminal sliding mode is one that forces convergence to equilibrium in finite time (Vnekataraman and Gulati 1992).

Once in the sliding mode, the system can be considered to be robust as it is not affected by external disturbances, alterations in the system parameters or individual characteristics of parts of the system (Morgan and Ozguner 1985) and to reach the sliding mode no detailed model of the system is required by the designer (Hashimoto,

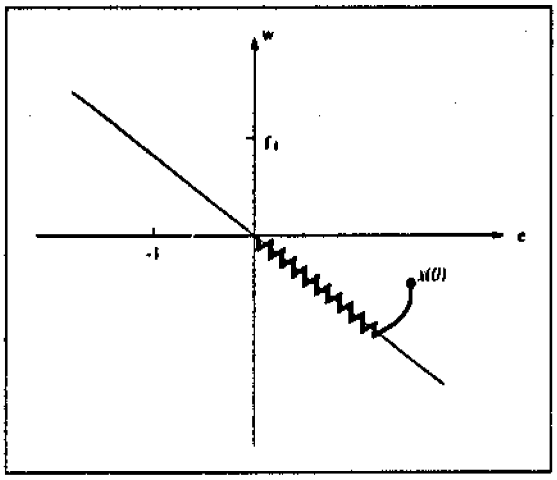

Figure 1.6: Sliding mode solution to control, in two dimensions (Schilling 1990).

Maruyama, and Harashima 1987).

One view of the problem has thus become that of designing a variable structure control law that pushes the system into the sliding mode as quickly as possible and then ensures that it stays there. Once in the sliding mode there is a need to reduce the chattering across the switching hyperplane and to ensure that tracking errors converge to zero in a finite time. It is also a requirement that a sliding mode will result no matter what the initial conditions and that the system be stable and unaffected by disturbances during the movement to the sliding mode.

Finally it is most useful if required prior knowledge of the system, its parameter bounds and expected external disturbances, be kept to a minimum. This has involved designs that combine the two methods of variable structure and adaptive systems.

A further extension of robot control work looks at the system that includes external disturbances. Thus the expression above becomes

$$u = M(q)\ddot{q} + f(q,\dot{q})\dot{q} + g(q) + d(t), \tag{1.3}$$

where $d(t) \in \mathbf{R}^n$ is the vector of bounded disturbances.

This thesis provides two new control laws that are based on expressions 1.2 and 1.3. Neither law attempts to deal with frictional forces as it has been shown in Leung, Zhou, and Su (1991) that this can be successfully compensated for. There is no examination of the problems encountered when the robot is constrained in some way, such as the requirement that the arm remain on a particular plane, recent research that covers this area can be found in Yao, Chan, and Wang (1992), Yao, Chan, and Wang (1994a) and Yao, Chan, and Wang (1994b).

n	$\triangleq$	the number of joints/links in the robot.
$u(t)$	$\triangleq$	the control input for the rigid robotic manipulator.
$r(t)$	$\triangleq$	the reference input.
$q(t)$	$\triangleq$	the vector of joint angular positions.
$q_r(t)$	$\triangleq$	the reference vector of joint angular positions.
$M(q)$	$\triangleq$	the inertia matrix.
$f(q, \dot{q})$	$\triangleq$	the vector of Coriolis and centrifugal forces.
$g(q)$	$\triangleq$	the vector of gravitational forces.
$\varepsilon(t)$	$\triangleq$	the position or tracking error, generally defined as $\varepsilon(t) = q - q_r$
$x(t)$	$\triangleq$	the state vector of the rigid robotic manipulator.
$x_r(t)$	$\triangleq$	the state vector of the reference model.
$e(t)$	$\triangleq$	the state error vector, generally defined as $e = x - x_r$.
$s(t)$	$\triangleq$	the vector representing the sliding mode manifold.
$d(t)$	$\triangleq$	the vector of bounded disturbances.
ϕ	$\triangleq$	the regressor matrix.

Table 1: Standard usage for variables.

2 Literature Review

Throughout this survey I have tried to be consistent in my use of variables, translating the work of the various papers into the standard usage shown in Table 2.

The major work of this thesis uses a combination of two methods of robot control, namely variable structure control and adaptive control. For this reason this review concentrates on these two areas. Section 2.1 covers adaptive control techniques, 2.2 covers variable structure systems, and 2.3 looks at combinations of the two. Section 2.4 then looks briefly at other available methods and 2.5 examines various methods of reducing the chattering that is inherent within variable structure systems.

2.1 Adaptive Control

Adaptive control applies to any system where the control signal is altered to take into account changing conditions in the plant parameters and disturbances in the environment. Commonly the last 'value' of the plant is used in combination with the last expected or reference value to provide an additional control term to that of the control law originally designed to drive the plant.

One of the early adaptive techniques for rigid robotic manipulators is that of Dubowsky and DesForges (1979), however their design is complex and unsystematic, requiring a difficult stability analysis.

A better design in Balestrino, Maria, and Sciavicco (1983) defines a state vector for the system as

$$x = \begin{bmatrix} q^T \\ \dot{q}^T \end{bmatrix} \tag{2.4}$$

and a state reference model as follows

$$\dot{x}_r = A_r x_r + B_r r \tag{2.5}$$

where

$$x_r = \begin{bmatrix} q_r^T \\ \dot{q}_r^T \end{bmatrix} \tag{2.6}$$

is a state vector for the reference model, $r(t)$ is the reference vector of joint torques,

q_r is the vector of reference joint angular positions,

$$A_r = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ A_{r1} & A_{r2} \end{bmatrix}, \tag{2.7}$$

and

$$B_r = \begin{bmatrix} \mathbf{0} \\ B_{r1} \end{bmatrix}, \tag{2.8}$$

where $A_{r1} = \mathrm{diag}(a_{r1i})$, $A_{r2} = \mathrm{diag}(a_{r2i})$ and $B_{r1} = \mathrm{diag}(b_{r1i})$ are constant $n \times n$ matrices chosen such that the reference model of expression 2.5 is stable.

Defining the error as

$$e = x_r - x, \tag{2.9}$$

where x is the state of the actual rigid robotic manipulator, defined as in expression 2.4, the input to the manipulator is chosen in Balestrino *et al.* (1983) as

$$u = \Phi(u, x, t)x - kx + \Psi(v, r, t)u + k_r r. \tag{2.10}$$

where v is a linear compensator, Φ and Ψ are matrices generated by the adaption mechanism and k, k_r are feedback and feedforward terms. This input is shown to be stable if the adaption mechanisms are designed appropriately, however the control law cannot guarantee the convergence of the tracking error to zero.

A further advance in adaptive methods is made in Craig, Hsu, and Sastry (1986).

The approach of Craig *et al.* (1986) does not require the detailed information of the rigid robotic manipulator of previous methods, but instead requires knowledge only of the bounds on the system parameters. However, it also requires the measurement of the joint acceleration which is not always possible and the inversion of the estimate of the inertia matrix which is computationally expensive (Spong and Ortega 1990). Another problem is that the system responds poorly when a load is picked up, leading to a large immediate deviation in the estimated parameters.

Hsu, Bodson, Sastry, and Paden (1987) remove these problems by filtering the control input using a proper transfer function such as $L(s) = \frac{b}{s+a}$ and using bounds on the inertia matrix rather than the matrix itself. However the response of the system is too slow for high speed robotic movement. Middleton and Goodwin (1988) also improve on the work of Craig *et al.*, using a least-squares estimation procedure for the parameters. They prove global convergence of the error and boundedness of the torque inputs and their derivatives (q, $\dot{q}$ and $\ddot{q}$), however they provide no simulation or experimental work to back up the proposed control law and hence no empirical evidence of the speed at which the convergence takes place.

In Singh (1986) a control law is developed that guarantees the ultimate boundedness of the error and ensures that the tracking error falls within an arbitrarily small bound. As with Craig *et al.* there is only the assumption of boundedness of system parameters, i.e. bounds on the possible values of M, f and g of the equation of motion given in expression 1.2.

Amestegui, Ortega, and Ibarra (1987) divide adaptive methods into two distinct types. The first of these, *adaptive computed torque*, uses the standard equation of motion, but replaces the known parameters with estimates:

$$u(t) = \hat{M}(q)\tilde{\ddot{q}} + \hat{f}(q,\dot{q})\dot{q} + \hat{g}(q), \tag{2.11}$$

where

$$\tilde{\ddot{q}} = K_r\dot{q} + K_p q - K_p q_r \tag{2.12}$$

and K_r and K_p are $n \times n$ matrices designed to ensure that the estimates of the parameters are equal to the parameters themselves under ideal conditions.

The second method, called an *adaptive compensator*, uses an available model, designated with the subscript 0, plus a compensating signal designed to deal with inaccuracies in the model and external disturbances. This gives the expression

$$u(t) = M_0(q)\tilde{\ddot{q}} + f_0(q,\dot{q})\dot{q} + g_0(q) + w, \tag{2.13}$$

where w is the compensator.

Amestegui *et al.* (1987) go on to show, via simulations, that neither of these two models give significant improvements over standard computed torque methods when applied to tracking.

Another attempt is made to remove the needs of acceleration measurement and matrix inversion in Sadegh and Horowitz (1987), however the controller design presented in this paper is computationally expensive and therefore difficult to implement in real situations. It also does not guarantee uniform asymptotic stability.

A summary and comparison of the results of Craig *et al.* (1986), Middleton and Goodwin (1988), Amestegui *et al.* (1987) and Sadegh and Horowitz (1987) (amongst others) can be found in Ortega and Spong (1988).

Miyasato and Oshima (1989) produce excellent results for tracking using a control law defined as

$$u = K_1 x + K_2 r + k_3 + \Theta_1 x + \Theta_2 r + \Theta_3, \tag{2.14}$$

where x is the state vector defined in expression 2.4, r is the reference vector of joint torques, $K_1 \in \mathbf{R}^{n\times n}$, $K_2 \in \mathbf{R}^{n\times n}$ and $k_3 \in \mathbf{R}^{n}$ are time-invariant functions of x, and $\Theta_1 \in \mathbf{R}^{n\times n}$, $\Theta_2 \in \mathbf{R}^{n\times n}$ and $\Theta_3 \in \mathbf{R}^{n}$ are updated with adaptive laws in seven variables. The paper proves that the control system is uniformly bounded and that the error converges to a region around zero that can be made arbitrarily small. The major disadvantage of the system is the computational expense if used for real-time fast moving rigid robotic manipulators.

2.2 Variable Structure Control

The first important survey in English on variable structure systems is that of Utkin (1977), in which he describes the concept of variable control in a general form, not applying it to rigid robotic manipulators. Prior to this most work was published by Russians in their own language.

The application of VSS to robotics is examined in Young (1978). This paper applies

VSS control theory to the set point regulation problem. Whilst it does remove the problems of interactions between joints, the method used involves a very complex analysis of stability. However, a significant result of this paper is the proof that a sliding mode exists if

$$s_i \dot{s}_i < 0 \qquad 1 \leqslant i \leqslant n, \tag{2.15}$$

where s_i is the switching manifold vector for the ith joint.

In Slotine and Sastry (1983) (and extended in Slotine (1984)) VSS theory is applied to the problem of robot tracking along with the idea of a sliding surface that varies with time. A single input nth-order time-varying linear control system is considered, and defined by

$$u = x_1^n + a_{n-1}(t)x_1^{n-1} + a_{n-2}(t)x_1^{n-2} + \cdots + a_0(t)x_1, \tag{2.16}$$

where x is defined as a state variable for the input:

$$x(t) = \begin{bmatrix} x_1(t) & \dot{x}_1(t) & \ddot{x}_1(t) & \cdots & x_1{}^{(n)}(t) \end{bmatrix}^T. \tag{2.17}$$

Using a similar definition for the state reference model $x_r(t)$, the error is defined as

$$e(t) = x(t) - x_r(t). \tag{2.18}$$

Finally Slotine and Sastry (1983) define the sliding (switching) surface as

$$s(x,t) = ce = 0, \tag{2.19}$$

where c is a vector of the form $\begin{bmatrix} c_1 & \cdots & c_{n-1} & 1 \end{bmatrix}$.

The control law developed to ensure that the trajectory slides along the sliding surface described above is

$$u = \beta^T(x)x + \sum_{i=1} n - 1k_i(x,t)\tilde{x}_{i+1} - k_n \mathrm{sgn}(s), \tag{2.20}$$

where k_i are suitably selected gain vectors. As can be seen, it is only a general expression for the control law and to formulate an actual control law it is necessary to have detailed information about the actual robot and the path it is to follow. A major disadvantage to the control laws proposed in Slotine and Sastry (1983) and Slotine (1984) is that they require the calculation of the inverse of the inertia matrix, making the law unrealistic for real-time robot control.

A full discussion of the choice of vector c (or matrix C) can be found in Dorling and Zinober (1986). Other research building on the concepts described in Slotine and Sastry and Slotine can be found in Hashimoto, Maruyama, and Harashima (1987), Bailey and Arapostathis (1987) and Kosuge and Furuta (1988) amongst others, with incremental increases in the efficiency and accuracy of the tracking. One difference between Kosuge and Furuta (1988) and the others is the use of task space rather than joint space.

Shoureshi, Momot, and Roesler (1990) split the system into a nominal system that can be linearized and a non-linear system that contains the uncertainties and for which a sliding mode control law is devised. Thus the control law becomes

$$u = u_1 + u_0, \tag{2.21}$$

where u_1 is the nominal input and u_0 is the sliding mode compensator. This is a very useful way to consider the problem and allows the best use of what prior knowledge is available for the system, whilst still taking into account the uncertainties of the system as a whole. Unfortunately the actual control law presented in this paper is too specific to the robot used in the experiments (the GE-P50) to be of general use. A further problem is that the bounds on the system parameters are required and are found via experimentation which means that they might not be absolute upper bounds. The problems of estimating these bounds are discussed in detail in Grimm (1990).

Wijesoma and Richards (1990), as well as providing a unifying approach for Slotine and Li (1987), Bailey and Arapostathis (1987) and Yeung and Chen (1988), also splits the system into two subsystems in a similar manner to Shoureshi *et al.* (1990). Furthermore it extrapolates the developed nominal plus sliding mode system into task space as did Kosuge and Furuta (1988). Chern and Wu (1992) also base their control law on the method of splitting the system into one of nominal control plus compensator, but this results in a complicated control law that requires calculation of three separate sliding modes for each joint of the robot.

Song and Gao (1991) uses a variable structure system to deal with the extra problem of external disturbances, as described by expression 1.3. The resulting control law, as for Shoureshi *et al.*, requires knowledge of the bounds on the system, namely the upper and lower bounds on the uncertain parts of M, f, g and d in expression 1.3. Whilst the control law itself produces excellent results, the requirement of knowing these eight bounds is restrictive. The paper also develops a controller based on the fact that there exists a vector α and a function ϕ such that

$$M(q,p)\ddot{q}_r + f(q,\dot{q},p)\dot{q}_r + g(q,p) = \phi(q,\dot{q},\dot{q}_r,\ddot{q}_r)\alpha(p), \tag{2.22}$$

where ϕ is known as the regressor and is not dependent on unknown parameters of the rigid robotic manipulator(Slotine and Li 1987). This leads to a simpler design than the previous one, however it requires the bounds on α, which may be even harder to calculate or estimate than those of M, f, g and d.

Stepanenko and Su (1992) redefine this property of motion as

$$M(q,p)\ddot{q} + f(q,\dot{q},p)\dot{q} + g(q,p) = \phi(r,\dot{r},\ddot{r})p. \tag{2.23}$$

The control law they use to solve this description of the problem is then given as

$$u = \phi(r,\dot{r},\ddot{r})\psi - K_d s - f(s,e), \tag{2.24}$$

where s is the switching vector, e the tracking error, ψ is defined as

$$\psi = -\beta_i \text{sgn}(() \sum_{j=1}^{n} s_j \phi_{ij}(r,\dot{r},\ddot{r})) \quad 1 \leqslant i \leqslant m, \tag{2.25}$$

β_i are the system bounds, K_d is a positive symmetric gain matrix and $f(s,e)$ is a non-linear feedback term. The paper proves that the system is robust with respect to unmodelled dynamics and measurement noise and since the regressor is a function of the trajectory alone, most of the computationally expensive calculation can be done off-line. However, there is no proof that the reaching time is finite or fast, or that the error converges to zero, and there is no experimental or simulation evidence to show the actual effectiveness of the control law proposed.

A much simpler control law is developed in Man and Palaniswami (1992) in which they prove asymptotic error convergence. Using the description of the system from expressions 2.4 to 2.8 and combining this with a state error defined by

$$e = \begin{bmatrix} \varepsilon \\ \dot{\varepsilon} \end{bmatrix}, \tag{2.26}$$

they develop an error differential expression

$$\dot{e} = A_r e + B_e h(q, \dot{q}, u, r), \tag{2.27}$$

where

$$B_e = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}, \tag{2.28}$$

and

$$h(q, \dot{q}, u, r) = M(q)^{-1}(u - f(q, \dot{q})\dot{q} - g(q)) - A_{r1}q - A_{r2}\dot{q} - B_{r1}r. \tag{2.29}$$

(Man and Palaniswami 1992) define the switching manifold vector similarly to Leung *et al.* (1991) in expression 2.73 as

$$s = Ce, \tag{2.30}$$

$$C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}, \tag{2.31}$$

the difference being that C_1 and C_2 are defined as diagonal matrices where

$$Re\lambda(-C_2^{-1}C_1) < 0. \tag{2.32}$$

The following assumptions are then used:

Assumption 2.1 *r, q_r and $\dot{q}_r$ are measurable,*

Assumption 2.2 $\lambda_{min}(C_2 M(q)^{-1} C_2^T) \geqslant a_1$,

Assumption 2.3 $\|M(q)^{-1}\| \leqslant a_2$,

Assumption 2.4 $\|f(q,\dot{q})\dot{q} + g(q)\| < b_1 + b_2\|q\| + b_3\|\dot{q}\|^2$,

where a_1, a_2, b_1, b_2 and b_3 are positive numbers.

The control law is then described by

$$u = \begin{cases} -\frac{C_2^T s}{a_1\|s\|}\, w & \|s\| \neq 0 \\ 0 & \|s\| = 0 \end{cases}, \tag{2.33}$$

where

$$\begin{aligned} w &= \|C_2 A_{r1} q_r\| + \|C_2 A_{r2} \dot{q}_r\| + \|C_2 B_{r1} r\| \\ &\quad + a_2\|C_2\|(b_1 + b_2\|q\| + b_3\|\dot{q}\|^2) + \|C_1\|\dot{e}. \end{aligned} \tag{2.34}$$

The simulation results provided in the paper show excellent tracking and error convergence, however the law requires relatively high control inputs to maintain the system in the sliding mode. The advantage of this law over other laws is that only the five bounds of assumptions 2.2 to 2.4 need to be found and not both bounds on every system parameter.

Man and Palaniswami (1993) propose a different variable structure scheme, based on the variable structure adaptive scheme proposed in Leung *et al.* (1991) (see page 37. In this paper they simplify the control law of Leung *et al.* to

$$u = \Theta_1 x + \Theta_2 r + \Theta_3 e, \tag{2.35}$$

where Θ_i are variable structure controller gain matrices which, unlike Leung, Zhou, and Su, contain no adaptive terms.

Another paper with work on similar lines to that of Man and Palaniswami (1992) is Hanmandlu and Pandian (1993), however this paper does not prove convergence or robustness and fails to reference most of the more recent work on the tracking control problem. An improvement on the work of Man and Palaniswami (1992) is made in Man, Paplinski, and Wu (1994) which reduces the required control inputs. Using the definition of error from Morgan and Ozguner (1985), i.e.

$$\varepsilon = q - q_r, \tag{2.36}$$

this paper defines the switching manifold as

$$s = C\hat{e}, \tag{2.37}$$

where

$$C = \begin{bmatrix} C_1 & C_2 \end{bmatrix} = \begin{bmatrix} c_{11} & & & 1 & & \\ & \ddots & & & \ddots & \\ & & c_{nn} & & & 1 \end{bmatrix}, \quad c_{ii} > 0, \quad 1 \leqslant i \leqslant n \tag{2.38}$$

and

$$\hat{e} = \begin{bmatrix} \bar{\varepsilon} \\ \dot{\varepsilon} \end{bmatrix}, \tag{2.39}$$

where

$$\bar{\varepsilon} = \begin{bmatrix} \varepsilon_1^{\frac{p_1}{p_2}} & \dots & \varepsilon_n^{\frac{p_1}{p_2}} \end{bmatrix}^T \tag{2.40}$$

and

$$p_1 = (2m+1), \quad m = 0,1,2,\ldots \text{ and} \tag{2.41}$$

$$p_2 = (2m+1), \quad m = 1,2,\ldots \tag{2.41'}$$

Note that this is somewhat different from that described in Man *et al.* (1994). Dr Zhihong Man requests this alteration as it is a better description of the criteria for p_1 and p_2.

The ith element of s in expression 2.37 can be written in the form

$$s_i = c_{ii}\varepsilon_i^{\frac{p_1}{p_2}} + \dot{\varepsilon}_i,$$

thus the terminal sliding mode can be described by

$$s_i = c_{ii}\varepsilon_i^{\frac{p_1}{p_2}} + \dot{\varepsilon}_i = 0.$$

It is shown in this paper that the relaxation time for the system above is found with

$$\begin{aligned} t_i &= c_{ii}^{-1}\int_{\varepsilon_i(0)}^{\varepsilon_i \to 0} \frac{d\varepsilon_i}{\varepsilon_i^{\frac{p_1}{p_2}}} \\ &= \frac{|\varepsilon_i(0)|^{1-\frac{p_1}{p_2}}}{c_{ii}(1-\frac{p_1}{p_2})}, \end{aligned} \tag{2.42}$$

which implies that the output tracking error will converge to the sliding mode in finite time.

Using the same four assumptions as for Man and Palaniswami (1992) (assumptions 2.1 to 2.4), the control law they develop is

$$u = \begin{cases} -\frac{s}{a_1\|s\|}\,w & \|s\| \neq 0 \\ \\ 0 & \|s\| = 0 \end{cases}, \tag{2.43}$$

where

$$w = \|A_{r1}q_r\| + \|A_{r2}\dot{q}_r\| + \|B_{r1}r\| + a_2(b_1 + b_2\|q\| + b_3\|\dot{q}\|^2) + \|C_1\|\dot{\bar{\varepsilon}} \tag{2.44}$$

and $\bar{\varepsilon}$ is defined as in expression 2.40, and hence

$$\dot{\bar{\varepsilon}} = \mathrm{diag}(\tfrac{p_1}{p_2}\varepsilon^{\frac{p_1}{p_2}-1})\dot{\varepsilon}. \tag{2.45}$$

This paper proves error convergence as well as finite convergence to the sliding mode, however it does not include external disturbances and requires the bounds a_1, a_2, b_1, b_2 and b_3 to be known.

In Man and Palaniswami (1994) a control method is devised based on a similar tactic to Shoureshi *et al.* (1990) — dividing the system into a nominal part and an unknown part. The equation of motion used is derived from that of expression 1.2 and defined as

$$u(t) = M(q)\ddot{q} + h(q, \dot{q}), \tag{2.46}$$

where $h(q, \dot{q}) \in \mathbf{R}^{n\times 1}$ is the vector of combined Coriolis, centrifugal and gravitational torques.

They then consider the robotic manipulator described in expression 2.46 to have some known and some unknown parts defining M and h, in a similar manner to Kim, Lee, Park, and Youn (1993), as

$$M(q) = \Delta M(q) + M_0(q) \tag{2.47}$$

and

$$h(q,\dot{q}) = \Delta h(q,\dot{q}) + h_0(q,\dot{q}), \tag{2.48}$$

where $M_0(q)$ and $h_0(q,\dot{q})$ are the known parts and $\Delta M(q)$ and $\Delta h(q,\dot{q})$ are the unknown parts of the inertia matrix and combined Coriolis, centrifugal and gravitational forces.

Using expressions 2.47 and 2.48 expression 2.46 is rewritten as

$$M_0(q)\ddot{q} + h_0(q,\dot{q}) = u(t) + \rho(t), \tag{2.49}$$

where

$$\rho(t) = -\Delta M(q)\ddot{q} - \Delta h(q,\dot{q}) + d(t). \tag{2.50}$$

It is further stated that

$$\|\rho(t)\| < b_0 + b_1\|q\| + b_2\|\dot{q}\|^2. \tag{2.51}$$

Using the control law as in expression 2.21, the nominal system is then defined by

$$u_1 = M_0(q)\ddot{q} + h_0(q,\dot{q}). \tag{2.52}$$

Defining the state error, e, as in expression 2.72, the paper defines a linearized error system as

$$\dot{e} = Ae + Bv, \tag{2.53}$$

where

$$A = \begin{bmatrix} 0 & \mathbf{I} \\ 0 & 0 \end{bmatrix}, \tag{2.54}$$

$$B = \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} \tag{2.55}$$

and

$$v = M_0(q)^{-1}(u_1 - h_0(q,\dot{q})) - \ddot{q}_r. \tag{2.56}$$

The error dynamics described in expression 2.53 for the nominal system 2.52 are then considered to be stabilised using the following nominal feedback control law

(Abdallah, Dawson, Dorato, and Jamishidi 1991)

$$u_1 = h_0(q,\dot{q}) + M_0(q)(Ke + \ddot{q}_r), \tag{2.57}$$

where $K = \begin{bmatrix} -K_1 & -K_2 \end{bmatrix}$ $(K_1, K_2 \in \mathbf{R}^{n \times n})$, and matrix K is designed such that

$$A_k = A + BK, \tag{2.58}$$

is an asymptotically stable matrix. It is further stated that the error dynamics can be written in the form

$$\dot{e} = A_k e + BM_0(q)^{-1}u_0 + BM_0(q)^{-1}\rho(t). \tag{2.59}$$

The sliding hyperplane used in this paper is the same as for Man and Palaniswami (1992) and is defined in expressions 2.30 to 2.32. The final control law for the unknown portion of the system is given by the variable structure system

$$u_0 = \begin{cases} \frac{(s^T C_2 M_0(q)^{-1})^T}{\|s^T C_2 M_0(q)^{-1})^T\|^2} w & \|s\| \neq 0 \\ 0 & \|s\| = 0 \end{cases}, \tag{2.60}$$

where

$$w = -s^T C A_k e - \|s\| \|C_2 M_0(q)^{-1}\| (b_0 + b_1\|q\| + b_2\|\dot{q}\|^2). \tag{2.61}$$

This paper proves error convergence, but suffers from the problem of all non-adaptive systems in that the three parameter bounds b_0, b_1 and b_2 must be known. A similar scheme is designed in Man and Palaniswami (1995).

2.3 Combined Variable Structure and Adaptive Systems

One of the first combined laws for tracking is based on the work of Young (1978) and is given in Morgan and Ozguner (1985). Morgan and Ozguner do not claim that they combine the two methodologies, however if one considers expression 2.65 that they derive, this clearly fits the definition of adaptive laws as given on page 15.

The equation of motion is simplified by Morgan and Ozguner to

$$u_i = (m_i l_i^2 - J_i)\ddot{q}_i + f_i \quad 1 \leqslant i \leqslant n, \tag{2.62}$$

where m_i is the lumped equivalent mass of the ith link, l_i its length, J_i its moment of inertia about the centre of gravity and f_i is an uncertainty parameter that includes the gravitational torques and all non-linear interactions between the joints. Defining the position error as

$$\varepsilon = q - q_r, \tag{2.63}$$

where q_r is the reference input, they give the switching manifold variable as

$$s = C\varepsilon + \dot{q}. \tag{2.64}$$

They then assume that the disturbance f_i is measurable and use the previously sampled value to calculate the next control input. Thus the input torque required to maintain tracking at sampling time k, is given by

$$\tau_i(k) = (m_i l_i^2) - J_i)\ddot{q}_i(k) + f_i(k-1) \quad 1 \leqslant i \leqslant n. \tag{2.65}$$

This method of calculation suffers several major disadvantages. Firstly it is based on the assumption that there are only small changes in the disturbances from one sampling moment to the next. Secondly it requires the control of the acceleration of the joints, $\ddot{q}$. Finally detailed knowledge of the system (the lumped mass and the joint lengths) is required. If these are unavailable or inaccurate then the uncertainty factor $\dot{f}$ may fail to cope with the changes in disturbance. Mitra, Gupta, and Moinuddin (1989) refine the work of Morgan and Ozguner (1985) by shortening the time taken to reach the sliding mode and reducing the chattering once there, however the original problems are not dealt with.

Al-Abbass and Ozguner (1985) devise a combined system of variable structure and adaptive control. This is based on the model following adaptive system of Balestrino, Maria, and Sciavicco (1983) combined with the variable structure system of Young (1978). However, it works on the problem in a decentralised manner, considering the control law as being made up of n subsystems, which do not communicate with each other. The model for the state variable is then given as

$$\dot{x}_i = A_i x_i + B_i u_i + \sum_{\substack{j \neq i \\ j=1}}^{n} A_{ij} x_j \qquad 1 \leqslant i \leqslant n, \tag{2.66}$$

where the summation term represents the interactions between the subsystems. The reference model of expression 2.5 then becomes

$$\dot{x}_{mi} = A_{mi} x_{mi} + B_{mi} r_i \qquad 1 \leqslant i \leqslant n, \tag{2.67}$$

with $A_{mi} \in \mathbf{R}^{n \times n}$ and $B_{mi} \in \mathbf{R}^{n}$ Finally the sliding manifold is defined in a similarly decentralised manner as

$$s_i = C_i e_i \qquad 1 \leqslant i \leqslant n, \tag{2.68}$$

where

$$e_i = x_{mi} - x_i$$

and

$$\dot{e}_i = A_{mi} e_i + (A_{mi} - A_i) x_i + B_{mi} r_i - B_i u_i - \sum_{\substack{j \neq i \\ j=1}}^{n} A_{ij} x_j \qquad 1 \leqslant i \leqslant n.$$

In this paper a proof of the stability and error convergence of the control law

$$u_i = k_{ei} e_i + k_{pi} x_i + k_{ri} r_i + \gamma_i, \tag{2.69}$$

is presented, where k_{ei}, k_{pi}, k_{ri} and γ_i are appropriately designed feedback gains. One disadvantage of this system is the more complex calculations that are required, as each subsystem (joint) requires a different calculation of the feedback gains. These gains (as given in Al-Abbass and Ozguner (1985)) are themselves complicated to calculate, hence the processing load on a real-time system would be inappropriate for fast, accurate robot tracking.

An interesting variation of the use of combined adaptive and variable structure methods is given in Slotine and Li (1987) and further developed in Kelly and Ortega (1988). In these papers, a sliding mode is used to ensure accurate position tracking and an adaption method used to give accuracy of velocity of the joints compared to the reference.

Further work on the combination of adaptive and variable structure techniques is presented in Hsu and Costa (1989), in which they prove finite convergence to the sliding mode. Narendra and Boskovic (1990) give a useful summary of the difference between direct, indirect and variable structure adaptive systems, and combinations of these.

An important improvement on the work in previous papers is found in Leung, Zhou, and Su (1991). Using the state variable x as defined in expression 2.4, they rewrite expression 1.2 as

$$\begin{aligned} \dot{x} &= \begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} \begin{bmatrix} \dot{q} \\ M(q)^{-1}(-f(q,\dot{q})\dot{q} - g(q)) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} M(q)^{-1}u(t) \\ &= \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ A_1 & A_2 \end{bmatrix} x + \begin{bmatrix} \mathbf{0} \\ B_1 \end{bmatrix} u \\ &= Ax + Bu. \end{aligned} \tag{2.70}$$

Defining the state reference model as in expression 2.4 and the error as

$$\varepsilon = q - q_r, \tag{2.71}$$

the paper then defines state error as

$$e = \begin{bmatrix} \varepsilon \\ \dot{\varepsilon} \end{bmatrix}. \tag{2.72}$$

Note that this is the same as the state error given in expression 2.18, expressed in a slightly different manner. The sliding surface is then defined by a similar expression to 2.19, but with C defined as a matrix. Thus the sliding mode variable now becomes

$$s = Ce = 0, \tag{2.73}$$

where

$$C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}, \tag{2.74}$$

where $C_1, C_2 \in \mathbf{R}^{n \times n}$.

Substituting expression 2.72 into expression 2.73 results in

$$\dot{\varepsilon} = -C_2^{-1} C_1 \varepsilon. \tag{2.75}$$

This implies that the the transient response of the tracking error depends solely on the eigenvector structure of $C_1 C_2$.

Leung *et al.* (1991) then define the control law in a similar manner to expression 2.14, but include an error term

$$u = K_1 x + K_2 r + k_3 e + \Theta_1 x + \Theta_2 r + \Theta_3 e, \tag{2.76}$$

where the variable structure systems Θ_i involve five parameters that are adaptively updated. However a major drawback of the proposed law is that the system will fail upon some conditions, with the adaptive values tending to infinity as time tends to infinity (Man and Palaniswami 1993).

In much the same manner as Leung *et al.*, Fu (1992) expects the bounds of the system parameters to be known and uses the adaptive method only to estimate two compensation terms of the control law itself, which is an improvement over the five estimations required in Leung *et al.*.

Stepanenko and Su (1992) also use an adaptive method to update parameters. Using the variable structure control law (described on page 23) in expression 2.24 the method is extended to the adaptive form by the use of an adaptive formula for the bounds β_i of expression 2.25. However, the basic method used for the control suffers from several problems, as detailed in the discussion on its variable structure on page 24. Another hybrid control law is given in Qu, Dawson, and Dorsey (1992), however the experimental results show that the tracking error is relatively slow to converge to zero. Park, Jiang, Hesketh, and Clements (1994) also detail a combined adaptive and sliding mode control law, but do not prove asymptotic error convergence nor convergence to the sliding mode. An extension of this work into task space can be found in Jiang, Park, Clements, and Hesketh (1994).

In Su and Leung (1993) they derive their combined adaptive and variable method from the property of motion defined in expression 2.22 on page 23. This regressor based method sets the control law as

$$u = \phi(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\psi - K_d s, \tag{2.77}$$

where K_d is a positive definite design matrix, the switching manifold s is that described in expressions 2.73 and 2.74, with $C_2 = \mathbf{I}$, and ψ is defined as

$$\psi_i = -\hat{\eta}_i \mathrm{sgn}(() \sum_{j=1}^{n} s_j \phi_{ji}(q, \dot{q}, \dot{q}_r, \ddot{q}_r), \quad 1 \leqslant i \leqslant m, \tag{2.78}$$

The $\hat{\eta}_i$ are estimates that are then adaptively updated as the robot moves. This law gives asymptotic error convergence. The paper also proves the convergence when uncertainties are included (as per expression 1.3). The main problem with the law is that the number of adaptions necessary is dependent on the number of links in the rigid robotic manipulator, hence the complexity increases with the complexity of the robot. However,, the method of adaption and the method of proving that the adaption gives error convergence are useful tools and are emulated in Chapters 3 and 4. An extension of this work can be found in Su and Stepanenko (1993). In this paper the research is based entirely on the inclusion of disturbances and develops the control law with adaptive estimates for ψ as well as three control gain vectors, however the problem common to all regressor based methods, that of computational intensity, still remains.

2.4 Other Methods

There have been several other quite different approaches to designing control laws. Desa and Roth (1985) use servo-mechanism theory combined with the standard method of splitting up the system into nominal and unknown parts. The disadvantages of the method they develop are several fold. Firstly it is a method, not an actual control law, and hence needs to be carried further for every rigid robotic manipulator for which it is to be implemented. Secondly it only provides fast, accurate tracking where the task is clearly defined and preferably repetitive, this is because it is only fast if the kinematic and dynamic calculations can be done off-line.

Spong and Vidyasagar (1987) devise a method that uses stable factorization in the design of a linear compensator that is to deal with the inaccuracies in the model used to design the control law.

Abdallah, Dawson, Dorato, and Jamishidi (1991) provide an excellent summary of the four most common methods in use for solving the robot tracking problem in a robust manner. The linear-multivariable approach, passivity based approaches, variable structure approach and robust adaptive approach are described are described with simplicity and clarity.

2.5 Chattering

One of the biggest problems with VSS methods is the discontinuity at the sliding manifold formed by s, the sliding mode vector. Physical control devices have properties (such as switching delays) that interact with the change of control structure at the sliding manifold and cause the system to *chatter* back and forth across this discontinuity. The chattering is undesirable because it may cause unmodelled high-frequency dynamics (Slotine and Sastry 1983).

One method devised to deal with chattering is that of a variable sliding mode devised by Al-Abbass and Ozguner (1985). This requires that the elements of the matrix

C_i (in expression 2.68) be allowed to vary in time. Whilst this is a good method in principle, the complexities of calculation require too great an overhead in terms of processing for real-time application.

Furuta (1990) is one of very few authors who tackle the chattering problem by approaching the system as discrete rather than continuous from the start. The method they present can eliminate the chattering but does not give a very fast response, which is essential when the rigid robotic manipulator is moving at high speed.

Shoureshi, Momot, and Roesler (1990) deal with the problem by considering the characteristics of the actual robot used for the experiments. It is shown that the particular robot in use (a GE-P50) has enough friction at the joints to act as a low pass filter, thus reducing the chattering. Furthermore the DC motors in use at each joint have enough bandwidth to be unaffected by the chattering problem. These two chatter mitigating factors are claimed to apply to most industrial robots, however this means that the design is based on a characteristic of rigid robotic manipulators that may not always hold true.

However, the most popular method for solving this problem is via a boundary layer around the sliding surface, inside which the discontinuous function is approximated with a continuous one. This removes the chattering, but can no longer ensure convergence of the tracking error to zero, only to an arbitrarily small boundary of the region near the sliding mode manifold.

The ultimate boundedness of such a continuous system is proven guaranteed in Corless and Leitmann (1981) for any control p that conforms to the specification provided by

$$\begin{aligned} p(x,t) &= -\tfrac{\|\mu(x,t)\|}{\mu(x,t)}\rho(x,t) && \text{if } \|\mu(x,t)\| > \delta \\ p(x,t) &\leqslant \rho(x,t) && \text{if } \|\mu(x,t)\| \leqslant \delta, \end{aligned} \tag{2.79}$$

where $\rho(x,t)$ is a bound on the system uncertainties and μ is a function dependent on the particular system involved.

The concept of a boundary layer as applied to the control of rigid robotic manipulators is developed in Slotine and Sastry (1983), in this paper it is stated (without proof) that any interpolation between the two edges of the boundary layer that is a continuous function will suffice to remove the chattering.

In Slotine (1984) the boundary layer method is refined to allow for the varying of parameter uncertainties over time. This is called *suction control* and adds complexity for the sake of a small gain in accuracy. Sun, Sun, and Gu (1991) use a boundary layer to solve the chattering problem for a system that is considered discrete from the start in much the same way as does Furuta (1990).

Glatzl, Murphy, Wen, and Kopacek (1993) produce experimental evidence that suggests that the boundary layer is a poor way to reduce chattering, however as the control law used is that of Young (1978) which has been much improved upon, there is doubt as to the validity of the results.

Most control laws are fractional and it seems that the easiest way to create the boundary layer is to substitute for some part of the denominator the size of the boundary layer itself. For example, in Man and Palaniswami (1992) the nonboundary layer control law is

$$u = \begin{cases} -\frac{C_2^T s}{a_1\|s\|} w & \|s\| \neq 0 \\ \\ 0 & \|s\| = 0 \end{cases}, \tag{2.80}$$

and thus the boundary layer law is defined as

$$u = \begin{cases} -\frac{C_2^T s}{a_1\|s\|} w & \|s\| \geqslant \delta \\ \\ -\frac{C_2^T s}{a_1\delta} w & \|s\| < \delta \end{cases}, \tag{2.81}$$

Other papers that use a similar techuique to produce the boundary layer are Man and Palaniswami (1993), Man and Palaniswami (1994) and Man and Palaniswami (1995). The work in Chapters 3 and 4 uses the boundary layer to remove the problems of chattering across the switching manifold, altering the control law in a similar manner to that above.

3 An Adaptive Variable Structure Control Law

3.1 Introduction

This chapter introduces a new control law which does not require the prior knowledge of the bounds on system uncertainties. Instead it uses an adaptive method to estimate these bounds at run time. The controller ensures both asymptotic error convergence and robustness. The sliding mode manifold is that proposed by Man, Paplinski, and Wu (1994) and described on page 27 with expressions 2.36 to 2.42, the state description used is from Balestrino, Maria, and Sciavicco (1983) and given in expressions 2.4 to 2.8 on page 15 along with its extension by Man and Palaniswami (1992) given on page 24 in expressions 2.27 to 2.32. The state error is that defined by Leung, Zhou, and Su (1991), in expressions 2.71 to 2.72 on page 36.

To design this control law, it is necessary to make some assumptions about robotic manipulators and the bounds to the parametric uncertainties. These are based on the characteristics of industrial robots.

Assumption 3.1 $\|M(q)\| \leqslant k_1\ , \quad k_1 > 0.$

Assumption 3.2 $\|\dot{M}(q)\| < k_2 + k_3\|q\| + k_4\|\dot{q}\|\ , \quad k_2, k_2, k_3 > 0.$

Assumption 3.3 $\|f(q,\dot{q})\dot{q} + g(q)\| < k_5 + k_6\|q\| + k_7\|\dot{q}\|^2\ , \quad k_5, k_6, k_7 > 0$

The assumptions above are widely used in the literature on the control of rigid robotic manipulators. Assumption 3.1 can be found in Grimm (1990) and 3.3 in Man, Paplinski, and Wu (1994). Dr Zhihong Man has stated in a personal communication, that Assumption 3.2 can be proved for any rigid robotic manipulator but that there is no generic proof for all systems due to the differences in the characteristics between rigid robotic manipulators. A proof for the two-link robot simulated in sections 3.4 and 4.4 is given in Appendix A.

In the rest of this chapter, section 3.2 describes the new controller and section 3.3 describes the boundary layer control law. Section 3.4 gives the data and results for a simulation that applies the two controllers to a two link rigid robotic manipulator, using the mathematics program MatLab. Finally section 3.5 gives some conclusions.

3.2 Controller Design

The assumptions 3.1 to 3.3 cause problems in the design of robot control. As pointed out by Grimm (1990), they must be evaluated for every different robot and too conservative an estimate of the bounds leads to robustness that is too restrictive. This section develops a method by which these upper bounds can be estimated adaptively as the rigid robotic manipulator is in motion.

Let $\hat{k}_i$, $1 \leqslant i \leqslant 7$ be the estimates of k_i, $1 \leqslant i \leqslant 7$ in Assumptions 3.1 to 3.3 and let these estimates be adaptively updated using the following laws:

$$\dot{\hat{k}}_1 = \eta_1 \|s\| \| - A_{r1} q_r - A_{r2} \dot{q}_r - B_{r1} r + C_1 \dot{\hat{\varepsilon}} \|, \tag{3.2}$$

$$\dot{\hat{k}}_2 = \frac{1}{2}\eta_2\|s\|^2, \tag{3.3}$$

$$\dot{\hat{k}}_3 = \frac{1}{2}\eta_3\|s\|^2\|q\|, \tag{3.4}$$

$$\dot{\hat{k}}_4 = \frac{1}{2}\eta_4\|s\|^2\|\dot{q}\|, \tag{3.5}$$

$$\dot{\hat{k}}_5 = \eta_5\|s\|, \tag{3.6}$$

$$\dot{\hat{k}}_6 = \eta_6\|s\|\|q\|, \tag{3.7}$$

$$\dot{\hat{k}}_7 = \eta_7\|s\|\|\dot{q}\|^2, \tag{3.8}$$

where $\tilde{z}$ and $\dot{\tilde{z}}$ are given in expressions 2.40 and 2.45 respectively, and η_i for $1 \leqslant i \leqslant 7$ are arbitrary positive numbers and $\hat{k}_i$ for $1 \leqslant i \leqslant 7$ have arbitrary initial values.

Parameters η_i for $1 \leqslant i \leqslant 7$ in expressions 3.2 to 3.8 are called adaptive constants. They are properly chosen to adjust the convergence rate of adaptive laws in expressions 3.2 to 3.8. Generally, the larger the paramters η_i are, the faster the convergence of the adaptive gainss. However in practical situations, the values of parameters η_i can not be very large because this may cause high control gains which are undesirable.

Using these adaptive laws it is possible to pose the following theorem:

Theorem 3.1 *Consider the error dynamics of expression 2.27. If the control input vector is designed such that*

$$u = \begin{cases} -\frac{s}{\|s\|}w & \|s\| \neq 0, \varepsilon_i \neq 0 \\ & \\ 0 & otherwise \end{cases}, \tag{3.9}$$

where

$$\begin{aligned} w &= \| -A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{e}\|\hat{k}_1 \\ &+\tfrac{1}{2}\|s\|(\hat{k}_2 + \hat{k}_3\|q\| + \hat{k}_4\|\dot{q}\|) + (\hat{k}_5 + \hat{k}_6\|q\| + \hat{k}_7\|\dot{q}\|^2) \end{aligned}, \tag{3.10}$$

then the output tracking error vector converges to zero in a finite time.

Proof: Consider the Lyapunov function (Su and Leung 1993)

$$V = \frac{1}{2}s^T M(q)s + \frac{1}{2}\sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i^2. \tag{3.11}$$

where $\tilde{k}_i = k_i - \hat{k}_i, \quad 1 \leqslant i \leqslant 7.$

Differentiating V with respect to time we get

$$\dot{V} = \frac{1}{2}\dot{s}^T M(q)s + \frac{1}{2}s^T\dot{M}(q)s + \frac{1}{2}s^T M(q)\dot{s} - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i.$$

Since $\dot{s}^T M(q)s$ is scalar it follows that it must be equal to its own transpose, thus

$$\begin{aligned}\dot{s}^T M(q)s &= \left[\dot{s}^T M(q)s\right]^T \\ &= s^T M(q)^T \dot{s} \\ &= s^T M(q)\dot{s} \qquad \text{(Since } M(q) \text{ is symmetric).}\end{aligned}$$

Thus the derivative of V can be written as

$$\dot{V} = s^T M(q)\dot{s} + \frac{1}{2}s^T \dot{M}(q)s - \sum_{i=1}^{7} \eta_i^{-1}\tilde{k}_i\dot{\tilde{k}}_i. \tag{3.12}$$

In Man, Paplinski, and Wu (1994) it was noted that expression 2.37 could be written as

$$s = Ce + C_1(\hat{\varepsilon} - \varepsilon). \tag{3.13}$$

Differentiating this with respect to time we get

$$\dot{s} = C\dot{e} + C_1\dot{\hat{\varepsilon}} - C_1\dot{\varepsilon}.$$

and substituting this into expression 3.12 for $\dot{V}$ we have

$$\dot{V} = s^T M(q)\left[C\dot{e} + C_1\dot{\hat{\varepsilon}} - C_1\dot{\varepsilon}\right] + \frac{1}{2}s^T \dot{M}(q)s - \sum_{i=1}^{7} \eta_i^{-1}\tilde{k}_i\dot{\tilde{k}}_i.$$

Substituting for $\dot{e}$ from expression 2.27

$$\begin{aligned}\dot{V} &= s^T M(q)\left[CA_r e + CB_e h(q,\dot{q},u,r) + C_1\dot{\bar{\varepsilon}} - C_1\dot{\varepsilon}\right] \\ &\quad + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\tilde{k}}_i.\end{aligned}$$

Looking at the definitions for C and B_e (expressions 2.38 and 2.28), it can be seen that $CB_e = \mathbf{I}$. Thus we now have

$$\dot{V} = s^T M(q)\left[CA_r e + h(q,\dot{q},u,r) + C_1\dot{\bar{\varepsilon}} - C_1\dot{\varepsilon}\right] + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\tilde{k}}_i.$$

Substituting for h from expression 2.29

$$\begin{aligned}\dot{V} &= s^T M(q)\left[\, CA_r e + M(q)^{-1}(u - f(q,\dot{q})\dot{q} - g(q))\right. \\ &\quad \left. - A_{r1}q - A_{r2}\dot{q} - B_{r1}r + C_1\dot{\bar{\varepsilon}} - C_1\dot{\varepsilon}\,\right] + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\tilde{k}}_i.\end{aligned} \qquad (3.14)$$

Using the definitions of C, A_r and e (expressions 2.38, 2.7 and 2.72), we can simplify the term $CA_r e$ as follows:

$$\begin{aligned}CA_r e &= \begin{bmatrix} C_1 & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{0} & \mathbf{I} \\ A_{r1} & A_{r2}\end{bmatrix}\begin{bmatrix}\varepsilon \\ \dot{\varepsilon}\end{bmatrix} \\ &= \begin{bmatrix} A_{r1} & C_1 + A_{r2}\end{bmatrix}\begin{bmatrix}\varepsilon \\ \dot{\varepsilon}\end{bmatrix} \\ &= A_{r1}\varepsilon + (C_1 + A_{r2})\dot{\varepsilon},\end{aligned} \qquad (3.15)$$

and substituting this into expression 3.14 we have

$$\begin{aligned}
\dot{V} &= s^T M(q)\,[\,A_{r1}\varepsilon + (C_1 + A_{r2})\dot{\varepsilon} + M(q)^{-1}(u - f(q,\dot{q})\dot{q} - g(q)) \\
&\quad -A_{r1}q - A_{r2}\dot{q} - B_{r1}r + C_1\dot{\hat{\varepsilon}} - C_1\dot{\varepsilon}\,] + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i \\
&= s^T M(q)\,[\,A_{r1}\varepsilon + C_1\dot{\varepsilon} + A_{r2}\dot{\varepsilon} + M(q)^{-1}(u - f(q,\dot{q})\dot{q} - g(q)) \\
&\quad -A_{r1}q - A_{r2}\dot{q} - B_{r1}r + C_1\dot{\hat{\varepsilon}} - C_1\dot{\varepsilon}\,] + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i \\
&= s^T M(q)\,[\,A_{r1}\varepsilon + A_{r2}\dot{\varepsilon} + M(q)^{-1}(u - f(q,\dot{q})\dot{q} - g(q)) \\
&\quad -A_{r1}q - A_{r2}\dot{q} - B_{r1}r + C_1\dot{\hat{\varepsilon}}\,] + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i \\
&= s^T M(q)\,[\,A_{r1}\varepsilon - A_{r1}q + A_{r2}\dot{\varepsilon} - A_{r2}\dot{q} + M(q)^{-1}(u - f(q,\dot{q})\dot{q} - g(q)) \\
&\quad -B_{r1}r + C_1\dot{\hat{\varepsilon}}\,] + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i \\
&= s^T M(q)\,[\,A_{r1}(\varepsilon - q) + A_{r2}(\dot{\varepsilon} - \dot{q}) + M(q)^{-1}(u - f(q,\dot{q})\dot{q} - g(q)) \\
&\quad -B_{r1}r + C_1\dot{\hat{\varepsilon}}\,] + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i .
\end{aligned}$$

Substituting for ε from expression 2.36 gives

$$\begin{aligned}
\dot{V} &= s^T M(q)\,[\,-A_{r1}q_r - A_{r2}\dot{q}_r + M(q)^{-1}(u - f(q,\dot{q})\dot{q} - g(q)) \\
&\quad -B_{r1}r + C_1\dot{\hat{\varepsilon}}\,] + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i \\
&= s^T M(q)\,[\,-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\hat{\varepsilon}} + M(q)^{-1}u \\
&\quad -M(q)^{-1}(f(q,\dot{q})\dot{q} + g(q))\,] + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i \\
&= s^T M(q)\,(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\hat{\varepsilon}}) \\
&\quad +s^T u - s^T(f(q,\dot{q})\dot{q} + g(q)) + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i .
\end{aligned}$$

Substituting for u from expression 3.9 (assuming $\|s\| \neq 0$)

$$\begin{aligned}
\dot{V} &= s^T M(q)\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\right) \\
&\quad + s^T\left(-\tfrac{s}{\|s\|}w\right) - s^T(f(q,\dot{q})\dot{q} + g(q)) \\
&\quad + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i \\
&= s^T M(q)\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\right) \\
&\quad - \|s\|w - s^T(f(q,\dot{q})\dot{q} + g(q)) \\
&\quad + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i.
\end{aligned}$$

Substituting for w from expression 3.10

$$\begin{aligned}
\dot{V} &= s^T M(q)\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\right) \\
&\quad - \|s\|\| - A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\|\hat{k}_1 - \tfrac{1}{2}\|s\|^2(\hat{k}_2 + \hat{k}_3\|q\| + \hat{k}_4\|\dot{q}\|) \\
&\quad - \|s\|(\hat{k}_5 + \hat{k}_6\|q\| + \hat{k}_7\|\dot{q}\|^2) - s^T(f(q,\dot{q})\dot{q} + g(q)) \\
&\quad + \tfrac{1}{2}s^T\dot{M}(q)s - \sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i.
\end{aligned}$$

Rearranging this expression and splitting $\sum_{i=1}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i$ into its component parts we can write the expression for $\dot{V}$ as three terms

$$\begin{aligned}
\dot{V} &= \Big[s^T M(q)\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\right) \\
&\quad - \|s\|\| - A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\|\hat{k}_1 - \eta_1^{-1}\tilde{k}_1\dot{\hat{k}}_1\Big] \\
&\quad + \left[\tfrac{1}{2}s^T\dot{M}(q)s - \tfrac{1}{2}\|s\|^2(\hat{k}_2 + \hat{k}_3\|q\| + \hat{k}_4\|\dot{q}\|) - \sum_{i=2}^{4}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i\right]
\end{aligned}$$

$$
\begin{aligned}
&+\left[-s^T(f(q,\dot{q})\dot{q}+g(q)) - \|s\|(\hat{k}_5+\hat{k}_6\|q\|+\hat{k}_7\|\dot{q}\|^2) - \sum_{i=5}^{7}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i\right] \\
\therefore\ \dot{V} &= T1+T2+T3.
\end{aligned}
$$

These three terms will now be considered separately.

Substituting for $\dot{\hat{k}}_1$ from expression 3.2 into the first term, we have

$$
\begin{aligned}
T1 &= s^T M(q)\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\right) \\
&\quad -\|s\|\| - A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\|\hat{k}_1 - \eta_1^{-1}\tilde{k}_1\dot{\hat{k}}_1
\end{aligned}
$$

$$
\begin{aligned}
\therefore\ T1 &= [s^T M(q)\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\right) \\
&\quad -\|s\|\| - A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\|\hat{k}_1 \\
&\quad -\tilde{k}_1\|s\|\| - A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\|,
\end{aligned}
$$

then substituting $\tilde{k}_1 = k_1 - \hat{k}_1$ and simplifying we get

$$
\begin{aligned}
T1 &= s^T M(q)\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\right) \\
&\quad -\|s\|\| - A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\|k_1.
\end{aligned}
$$

Since $s^T M(q)\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\right)$ is scalar, it must be less than or equal to its own norm,

$$
\begin{aligned}
T1 &\leqslant \|s^T M(q)\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\right)\| \\
&\quad -\|s\|\| - A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\varepsilon}\|k_1
\end{aligned}
$$

$$\begin{aligned}
\therefore\ T1 &\leqslant \|s\|\|M(q)\|\|\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\bar{\varepsilon}}\right)\| \\
&\quad -\|s\|\|-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\bar{\varepsilon}}\|k_1 \\
&\leqslant \left(\|M(q)\| - k_1\right)\left(\|s\|\|\left(-A_{r1}q_r - A_{r2}\dot{q}_r - B_{r1}r + C_1\dot{\bar{\varepsilon}}\right)\|\right) \\
&\leqslant 0 \qquad\qquad\qquad \text{(From Assumption 3.1).}
\end{aligned} \tag{3.16}$$

Considering the second term, and substituting for $\dot{\hat{k}}_2$, $\dot{\hat{k}}_3$ and $\dot{\hat{k}}_4$ (from expressions 3.3, 3.4 and 3.5) we get

$$T2 = \tfrac{1}{2}s^T\dot{M}(q)s - \tfrac{1}{2}\|s\|^2(\hat{k}_2 + \hat{k}_3\|q\| + \hat{k}_4\|\dot{q}\|) - \sum_{i=2}^{4}\eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i$$

$$\begin{aligned}
\therefore\ T2 &= \tfrac{1}{2}s^T\dot{M}(q)s - \tfrac{1}{2}\|s\|^2(\hat{k}_2 + \hat{k}_3\|q\| + \hat{k}_4\|\dot{q}\|) \\
&\quad -\eta_2^{-1}\tilde{k}_2\dot{\hat{k}}_2 - \eta_3^{-1}\tilde{k}_3\dot{\hat{k}}_3 - \eta_4^{-1}\tilde{k}_4\dot{\hat{k}}_4 \\
&= \tfrac{1}{2}s^T\dot{M}(q)s - \tfrac{1}{2}\|s\|^2(\hat{k}_2 + \hat{k}_3\|q\| + \hat{k}_4\|\dot{q}\|) \\
&\quad -\eta_2^{-1}\tilde{k}_2(\tfrac{1}{2}\eta_2\|s\|^2) - \eta_3^{-1}\tilde{k}_3(\tfrac{1}{2}\eta_3\|s\|^2\|q\|) \\
&\quad -\eta_4^{-1}\tilde{k}_4(\tfrac{1}{2}\eta_4\|s\|^2\|\dot{q}\|) \qquad \text{(Subs. for } \dot{\hat{k}}_i\text{)} \\
&= \tfrac{1}{2}s^T\dot{M}(q)s - \tfrac{1}{2}\|s\|^2(k_2 + k_3\|q\| + k_4\|\dot{q}\|) \quad \text{(Subs. for } \tilde{k}_i\text{)} \\
&< \tfrac{1}{2}((k_2 + k_3\|q\| + k_4\|\dot{q}\|) - \dot{M}(q))\|s\|^2 \\
&< 0 \qquad\qquad\qquad \text{(From Assumption 3.2).}
\end{aligned} \tag{3.17}$$

Finally, considering term 3, and performing similar substitutions we get

$$\begin{aligned}
T3 &= -s^T(f(q,\dot{q})\dot{q} + g(q)) - \|s\|(\hat{k}_5 + \hat{k}_6\|q\| + \hat{k}_7\|\dot{q}\|^2) - \sum_{i=5}^{7} \eta_i^{-1}\tilde{k}_i\dot{\hat{k}}_i \\
&= -s^T(f(q,\dot{q})\dot{q} + g(q)) - \|s\|(k_5 + k_6\|q\| + k_7\|\dot{q}\|^2) \\
&\leqslant \| - s^T(f(q,\dot{q})\dot{q} + g(q))\| - \|s\|(k_5 + k_6\|q\| + k_7\|\dot{q}\|^2) \\
&\leqslant \| - s^T\|\|(f(q,\dot{q})\dot{q} + g(q))\| - \|s\|(k_5 + k_6\|q\| + k_7\|\dot{q}\|^2) \\
&\leqslant \|s\|(\|(f(q,\dot{q})\dot{q} + g(q))\| - (k_5 + k_6\|q\| + k_7\|\dot{q}\|^2)) \\
&< 0 \qquad\qquad \text{(From Assumption 3.3)}.
\end{aligned} \tag{3.18}$$

Using expressions 3.16–3.18 we have

$$\dot{V} < 0 \qquad\qquad \|s\| \neq 0 \tag{3.19}$$

Using Lyapunov's second method (see page 5), expression 3.19 means that the switching plane variable vector s converges to zero in a finite time and on the terminal sliding mode (Vnekataraman and Gulati 1992), represented by expression 2.37, if the error dynamics satisfy expression 2.2 then the output tracking error converges to zero in finite time.

It is easy to see that, using the control law in expression 3.9, the control signal is bounded in error space. At the point $\varepsilon_i = 0$ and $\|s\| \neq 0$, the control input $u = 0$ can cause the error ε_i to move away from 0, however another part of the variable structure control law will then drive the sliding variable vector s to the terminal sliding mode where $s = 0$. The desired error dynamics can then be obatined on the terminal sliding mode.

This result differs from those of previous research (Slotine and Sastry 1983, Young 1978, Yeung and Chen 1988, Leung, Zhou, and Su 1991, Man and Palaniswami 1993, Man and Palaniswami 1994 amongst others) in that the output tracking error is guaranteed to be driven to terminal sliding mode in a finite time and once in that mode, driven to the system origin in a finite time (Man, Paplinski, and Wu 1994). This is achieved without prior knowledge of the bounds on the system uncertainties as defined in Assumptions 3.1 to 3.3, as required by Man, Paplinski, and Wu (1994).

3.3 Reduction of Chattering

The control law described in expression 3.9 has a discontinuity when $\|s\| = 0$ which leads to chattering. This problem can be solved by using a boundary layer around the discontinuity, giving the following control law

$$u = \begin{cases} -\frac{s}{\|s\|}w & \|s\| \geqslant \delta \\ \\ -\frac{s}{\delta}w & \|s\| < \delta \end{cases} , \qquad (3.20)$$

where $\delta > 0$ and w is as defined by expression 3.10. However when using the control law of expression 3.20, the output tracking error can not converge to zero, only to an arbitrarily small boundary region around zero.

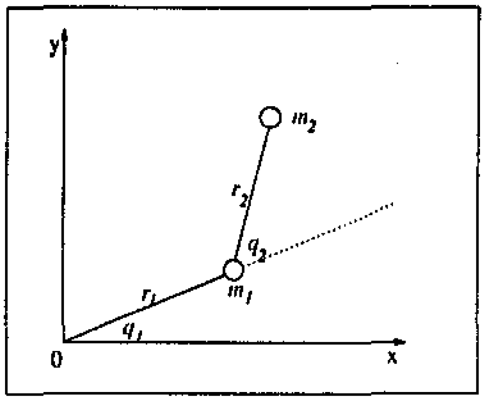

Figure 3.7: A two-link rigid robotic manipulator, where q_i is the position of each joint, in radians, m_i is the mass of the joint, in kg, and r_i is the radius of the joint in metres.

3.4 Simulation

The simulation is for a two link rigid robotic manipulator, as shown in figure 3.7 and was performed using the mathematics software package MatLab running on an IBM RS6000. It involved calculating the actual position of the simulated robot using inverse kinematics and comparing that to the required reference position.

Using m_i for mass, r_i for the joint length, J_i for the joint's moment of inertia, q_i for the joint's angular position (i = 1,2) and G for the acceleration due to gravity, the dynamic expressions for such a robot (as described in expression 1.2) are as follows (Young 1978) The inertia matrix is

$$M(q) = \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{bmatrix},$$

where

$$\begin{aligned}
\mu_{11} &= (m_1 + m_2)r_1^2 + m_2 r_2^2 + 2m_2 r_1 r_2 \cos(q_2) + J_1, \\
\mu_{12} &= m_2 r_2^2 + m_2 r_1 r_2 \cos(q_2), \\
\mu_{21} &= \mu_{12}, \\
\mu_{22} &= m_2 r_2^2 + J_2.
\end{aligned}$$

The vector of Coriolis and centrifugal forces is

$$f(q, \dot{q}) = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix},$$

where

$$\begin{aligned}
f_1 &= m_2 r_1 r_2 \sin(q_2)\dot{q}_1^2 + 2\dot{q}_1, \\
f_2 &= -m_2 r_1 r_2 \sin(q_2)\dot{q}_2^2.
\end{aligned}$$

The vector of gravitational forces is

$$g(q) = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix},$$

where

$$\begin{aligned}
g_1 &= -((m_1 + m_2)r_1 \cos(q_2)) + m_2 r_2 \cos(q_1 + q_2)))G \\
g_2 &= -m_2 r_2 \cos(q_1 + q_2) * G
\end{aligned}$$

and G is acceleration due to gravity.

The actual parameter values are set as

$$m_1 = 0.5 \text{ kg}, \quad m_2 = 1.5 \text{ kg}$$

$$r_1 = 1\text{m}, \quad r_2 = 0.8\text{m}$$

$$J_1 = 5 \text{ kg.m}, \quad J_2 = 5 \text{ kg.m}$$

The parameters for the reference model given by expression 2.5 are taken from Man, Paplinski, and Wu (1994):

$$A_r = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -4 & 0 & -5 & 0 \\ 0 & -4 & 0 & -5 \end{bmatrix},$$

$$B_r = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and}$$

$$r(t) = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \qquad \forall t > 0.$$

The desired reference signals are also taken from Man, Paplinski, and Wu (1994) and are given by

$$q_{r1} = 1.25 - \frac{5}{3}e^{-t} + \frac{5}{12}e^{-4t} + \frac{8}{30}e^{-t} - \frac{2}{30}e^{-4t}$$

and

$$q_{r2} = 1.25 + \frac{8}{3}e^{-t} - \frac{2}{3}e^{-4t} - \frac{5}{3}e^{-t} + \frac{5}{12}e^{-4t},$$

which gives initial values of the reference model as $q_{r1} = 0.2$ and $q_{r2} = 2.0$.

The values for p_1 and p_2 in expression 2.40 have been chosen as $p_1 = 3$ and $p_2 = 5$, which is consistent with the requirements of expression 2.41.

The Runge-Kutta method with sampling interval of 0.01s is used to solve the non-linear differential expressions numerically. The MatLab code for the simulation can be found in appendix B.

In figures 3.8, 3.11, 3.14, 3.17, 3.20 and 3.23, that show both the reference model and the robot, the reference model is shown with a solid line and the robot movement with a dotted line.

Initially the starting values for $\hat{k}_i$ were chosen as 0.5 and the values for η_i set to 10. The robot was assumed to start from the same position as the reference model. The output tracking of joints 1 and 2 is shown in figure 3.8. From these graphs it is clear that the estimates for k_i quickly adjust so that the position of the robot closely tracks that of the reference model. The tracking error for this choice of initial values can be seen in figure 3.9 and the control input in figure 3.10.

The effect of choosing a different initial value for $\hat{k}_i$ can be seen in figures 3.11 to 3.13. For this choice the tracking is so accurate that the dotted line showing the path of the robot simulator cannot be distinguished from the solid line of the reference path.

Looking at the graphs of the tracking error (figure 3.12), it can be seen that it remains very small indeed, in fact in the region of 0 ± 0.001 radians. This is an excellent response from the control law.

Figure 3.13 shows the control input for both joints. The effect of the chattering is clearly visible in the fast switching required from positive to negative input values. Despite this however, the results are good in terms of the relatively low absolute values of the control input required to achieve the tracking.

Figures 3.14 to 3.16 show the effect of having the robot's initial position differ from the reference model. The graphs indicate that the control law still gives a very good result, because the tracking converges in approximately 1 second, and the control input still has a reasonable bound.

Graphs shown in figures 3.17 to 3.19 give a 'worst case' situation where the robot does not have the same initial value as the reference model, and the initial values of $\hat{k}_i$ have been chosen poorly. It can be seen that the robot still only takes 2 seconds to commence tracking along the reference path. In real terms it should be possible to estimate reasonable starting values for $\hat{k}_i$ in much the same way as bounds estimates have always been estimated, ensuring that the initial values are not out by a factor of 10 as are those used in the simulation that gives these graphs.

Changing the value of η_i for $1 \leqslant i \leqslant 7$ can be seen in figures 3.20 to 3.22. Whilst the higher values of η_i may produce faster convergence of the adaptive laws, it can be seen by comparing figures 3.16 and 3.22 that the control input has increased significantly, wh·-h is undesirable.

The final graphs for the unbounded control law are shown in figures 3.23 to 3.25. These graphs show the results of starting values where $\varepsilon_1 = 0$ and $\varepsilon_2 \neq 0$ and hence $\|s\| \neq 0$. The control law can be seen to deal with this situation.

For the boundary layer simulations, the value of δ in expression 3.20 has been set to 0.06. The first graphs, shown in figures 3.26 to 3.28 show the situation where the robot and reference start at the same place and the 'good' initial value of $\hat{k}_i = 5$, for $1 \leqslant i \leqslant 7$ was used. It is clear from the graphs, that the tracking is not as accurate as when no boundary layer is used. The graphs for output error tracking show that it no longer oscillates around zero, instead tending towards values that are small, in the order of $< \pm 0.005$, but not zero. This value can be made arbitrarily small by reducing the chosen value of δ. The advantage of using a boundary layer can be seen in figure 3.28 where the control inputs now follow a smooth path instead of chattering.

The final figures (3.29 to 3.31) show the boundary layer control law for the 'worst case'. In can be seen that the tracking still becomes close to accurate after 2 seconds, and the control input no longer chatters.

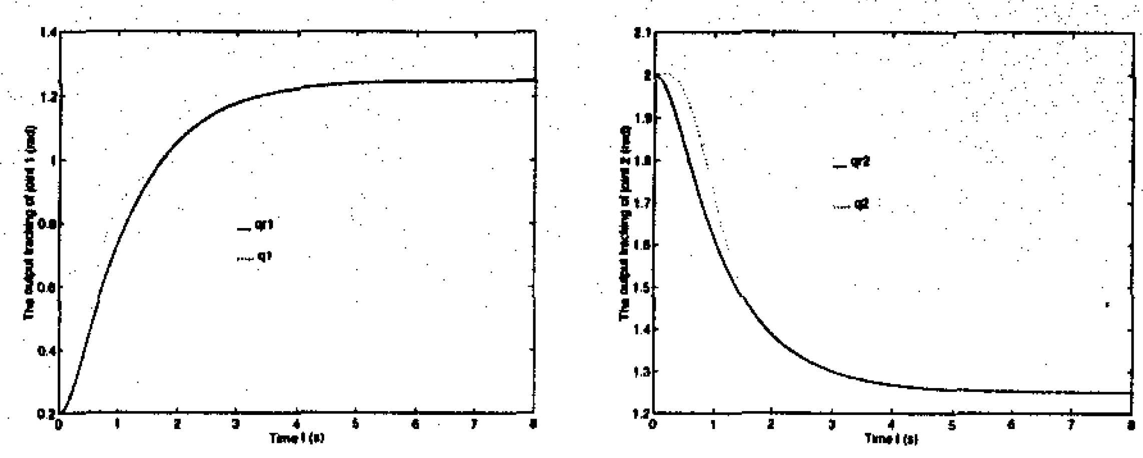

Figure 3.8: Simulation 1: Output tracking where reference and joint values coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 0.5$, $1 \leqslant i \leqslant 7$.

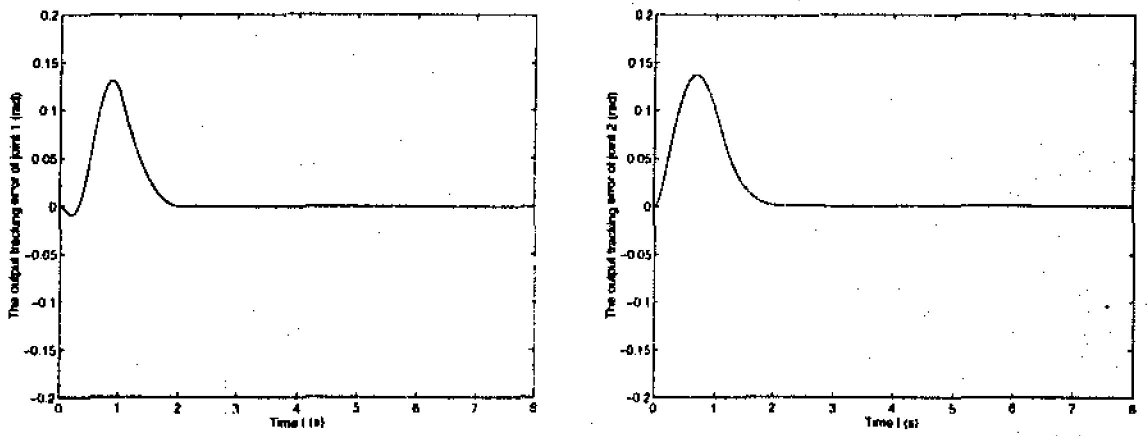

Figure 3.9: Simulation 1: Position errors where reference and joint values coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 0.5$, $1 \leqslant i \leqslant 7$.

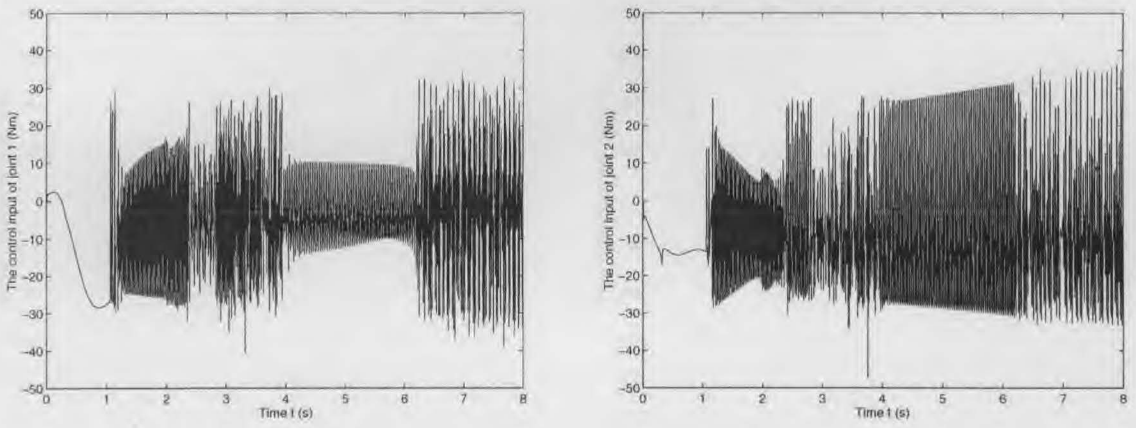

Figure 3.10: Simulation 1: Control input where reference and joint values coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 0.5$, $1 \leqslant i \leqslant 7$.

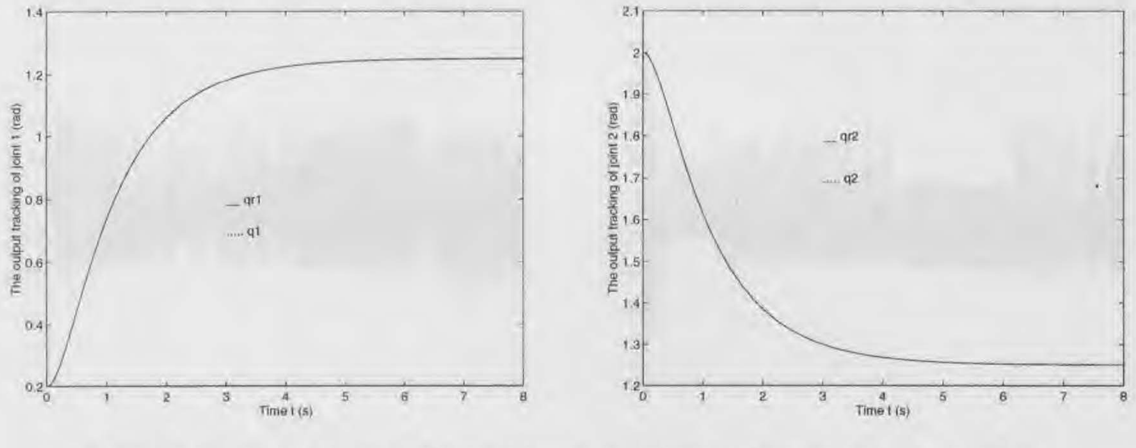

Figure 3.11: Simulation 1: Output tracking where reference and joint values coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

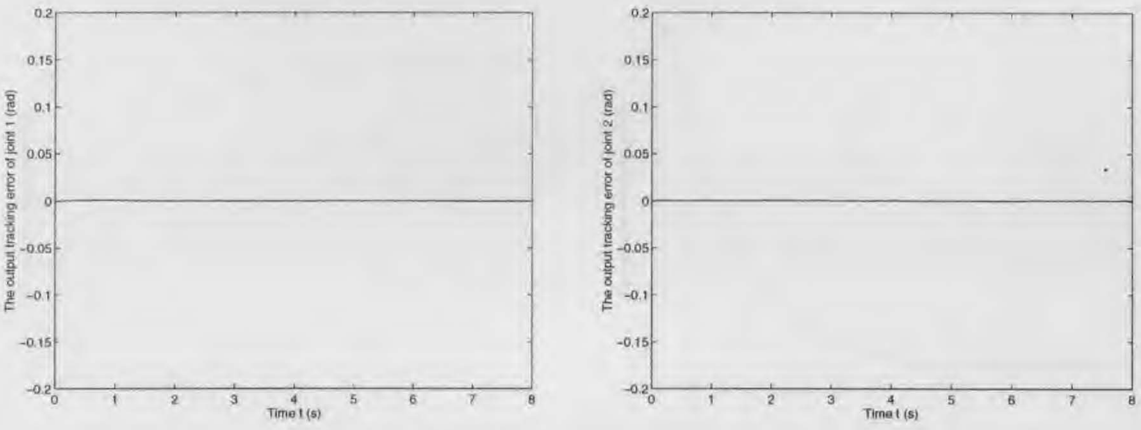

Figure 3.12: Simulation 1: Position errors where reference and joint values coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

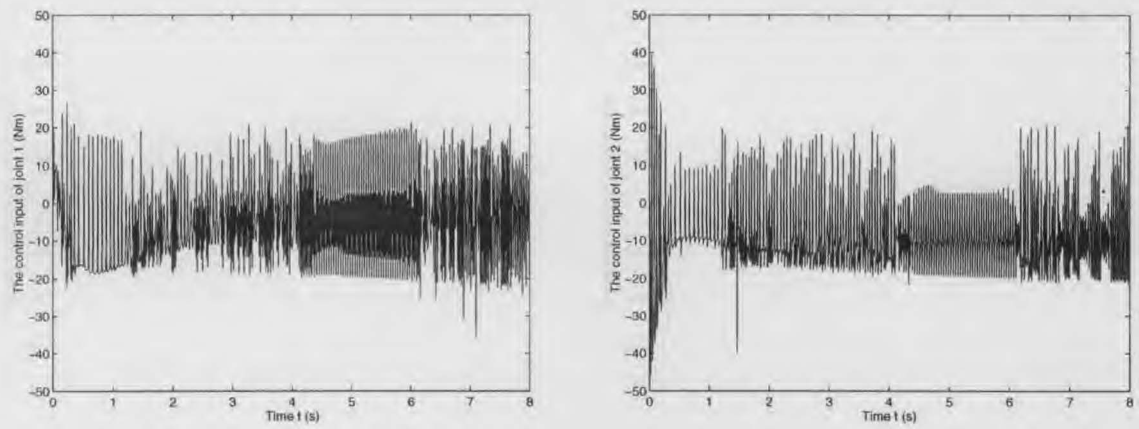

Figure 3.13: Simulation 1: Control input where reference and joint values coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

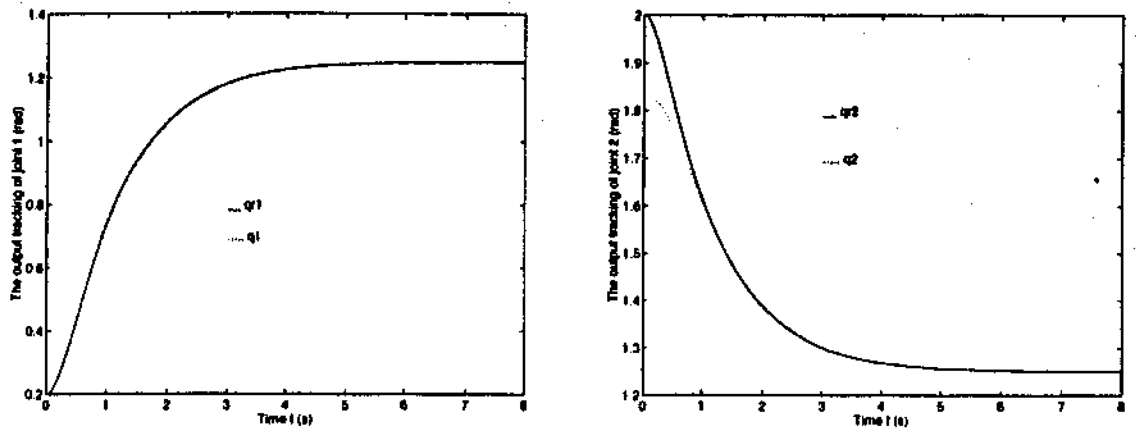

Figure 3.14: Simulation 1: Output tracking where reference and joint values do not coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

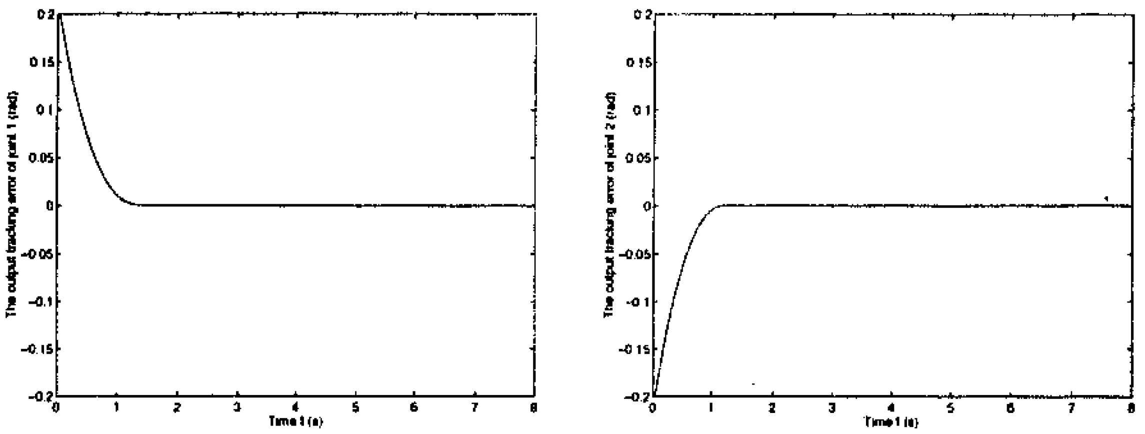

Figure 3.15: Simulation 1: Position errors where reference and joint values do not coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

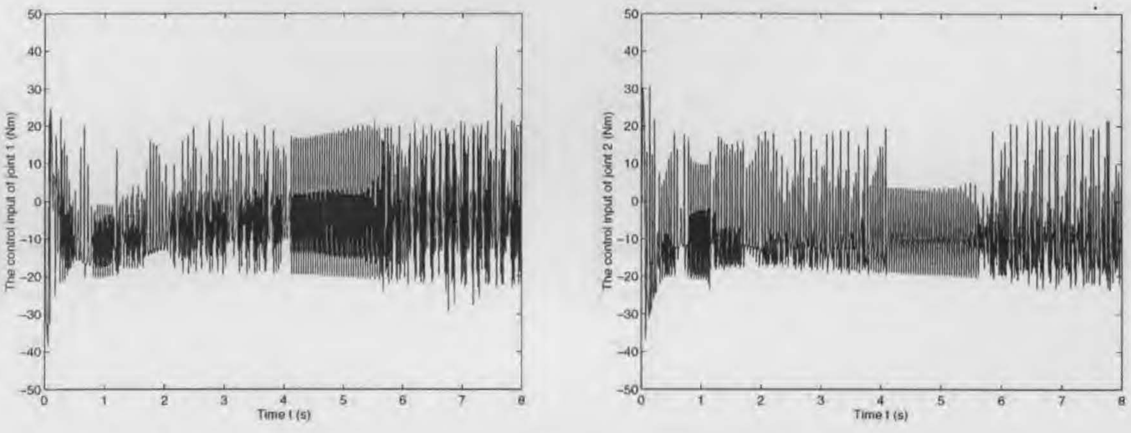

Figure 3.16: Simulation 1: Control input where reference and joint values do not coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

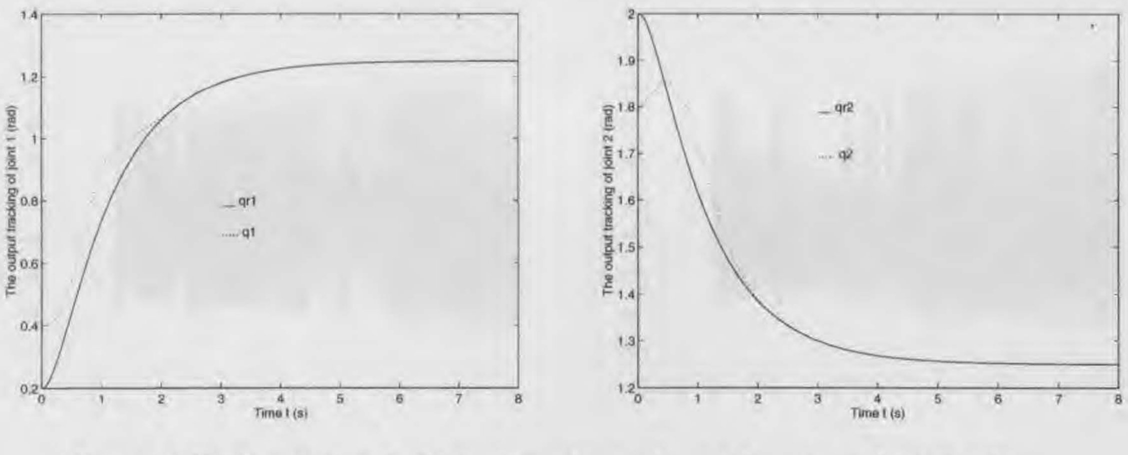

Figure 3.17: Simulation 1: Output tracking where reference and joint values do not coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = .5$, $1 \leqslant i \leqslant 7$.

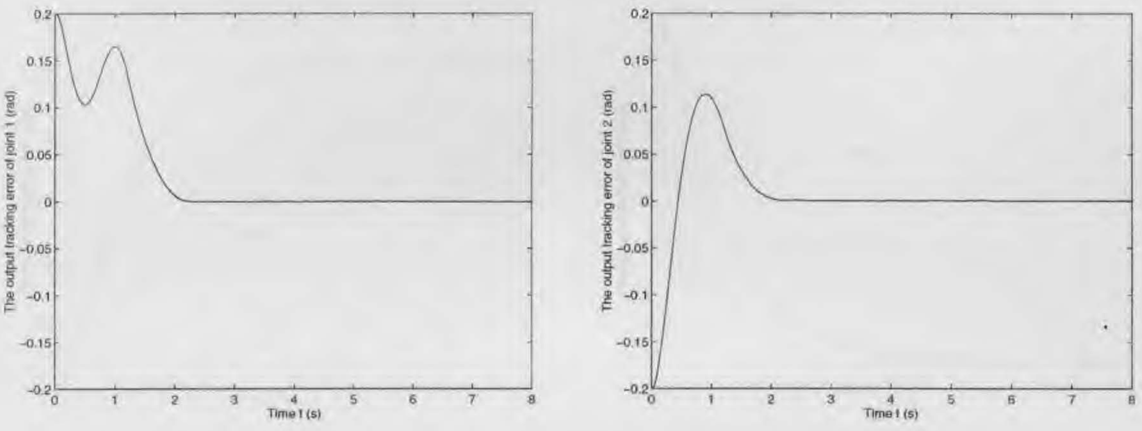

Figure 3.18: Simulation 1: Position errors where reference and joint values do not coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = .5$, $1 \leqslant i \leqslant 7$.

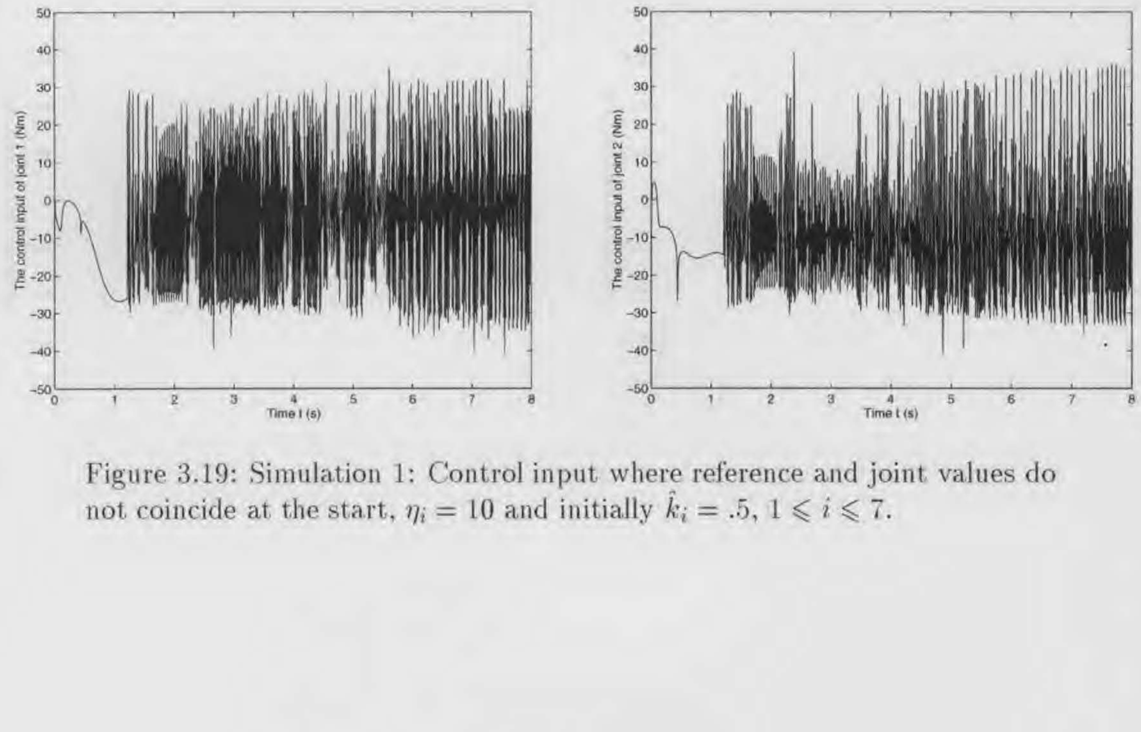

Figure 3.19: Simulation 1: Control input where reference and joint values do not coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = .5$, $1 \leqslant i \leqslant 7$.

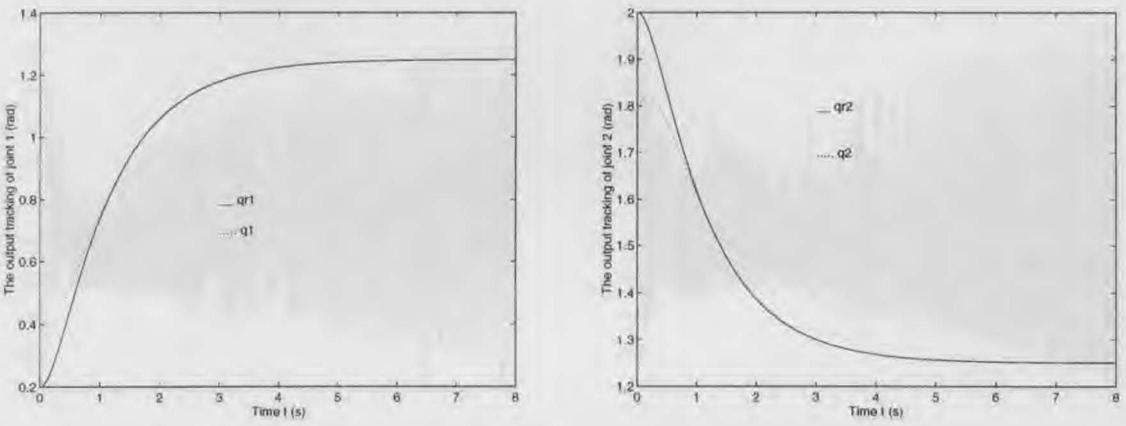

Figure 3.20: Simulation 1: Output tracking where reference and joint values do not coincide at the start, $\eta_i = 30$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

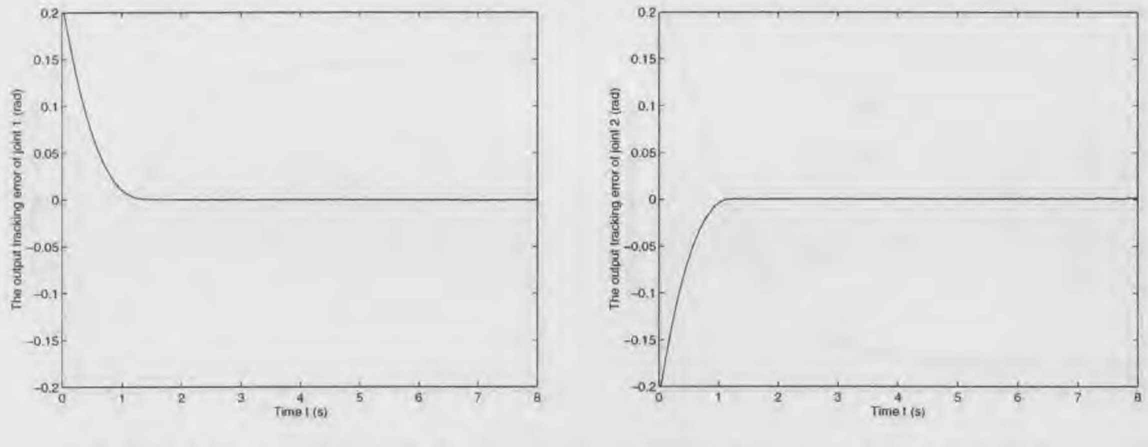

Figure 3.21: Simulation 1: Position errors where reference and joint values do not coincide at the start, $\eta_i = 30$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

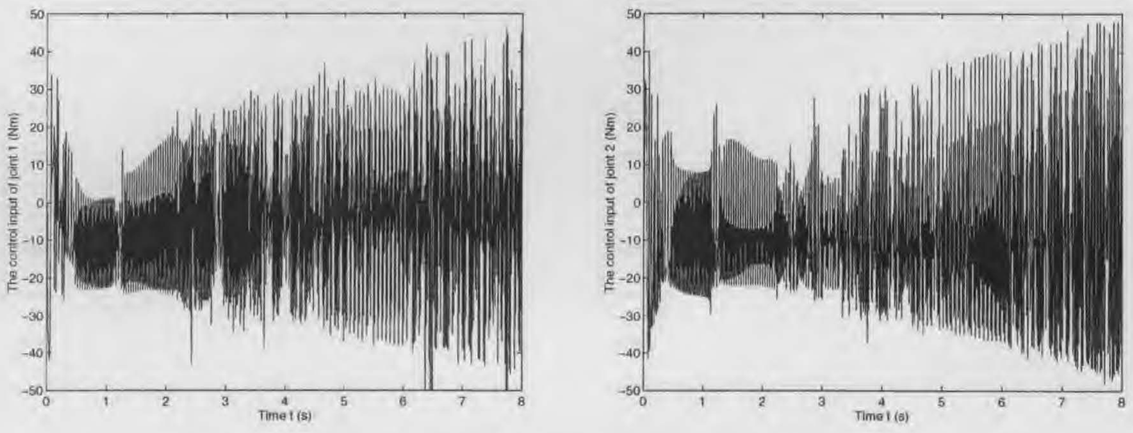

Figure 3.22: Simulation 1: Control input where reference and joint values do not coincide at the start, $\eta_i = 30$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

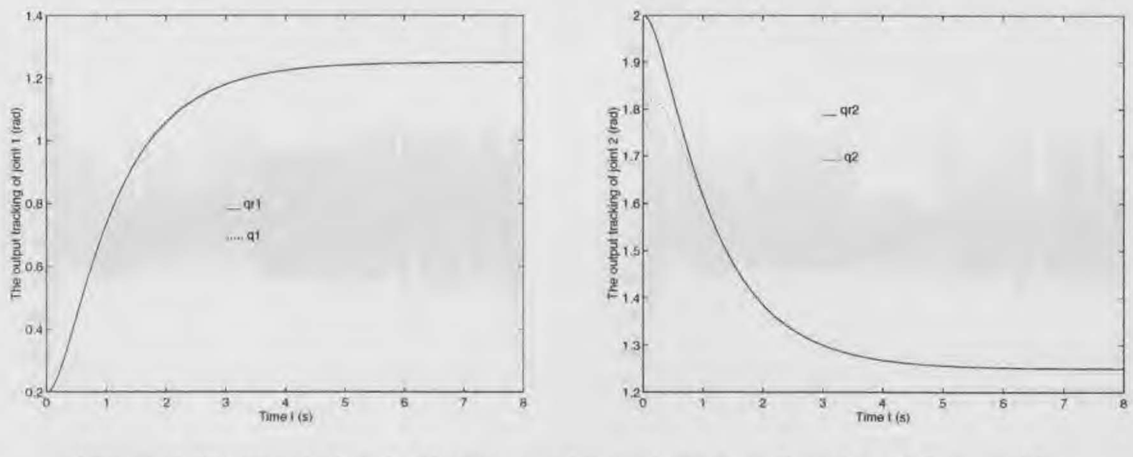

Figure 3.23: Simulation 1: Output tracking where $\varepsilon_1 = 0$ and $\varepsilon_2 \neq 0$, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

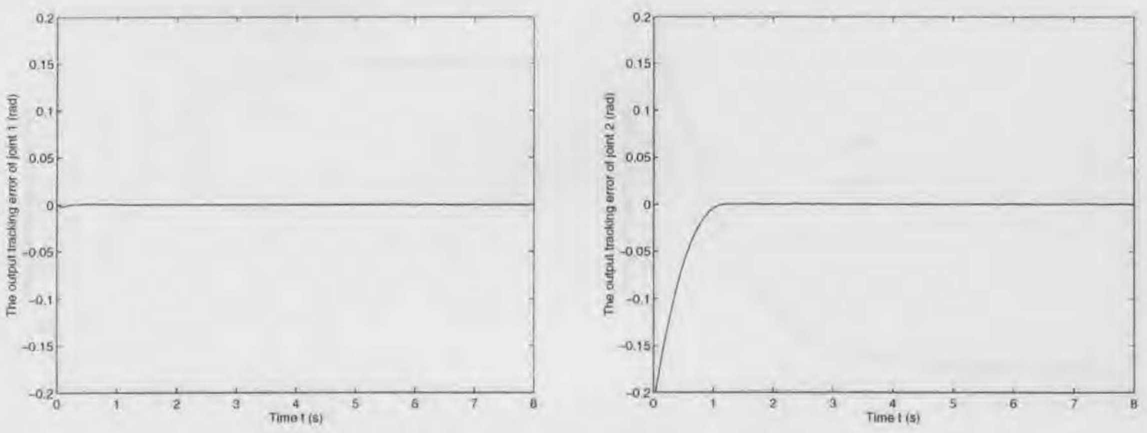

Figure 3.24: Simulation 1: Position errors where $\varepsilon_1 = 0$ and $\varepsilon_2 \neq 0$, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

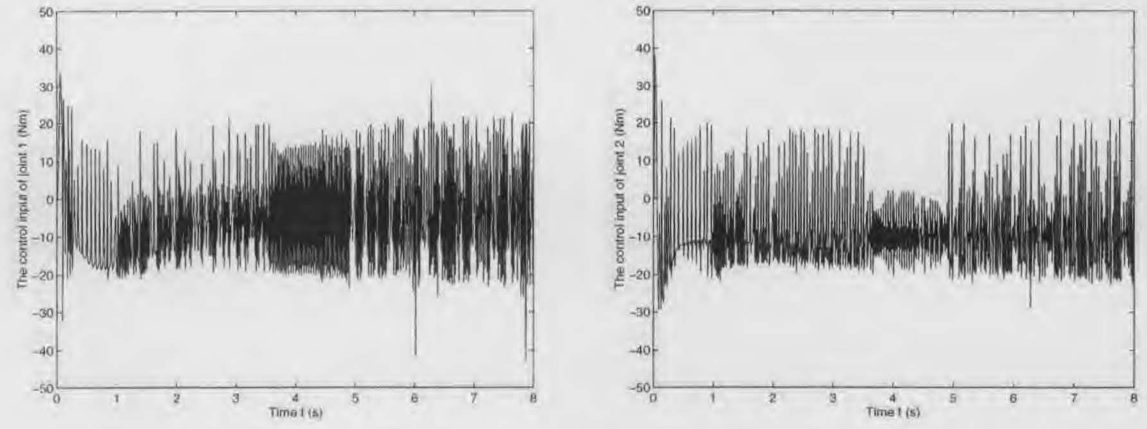

Figure 3.25: Simulation 1: Control input where $\varepsilon_1 = 0$ and $\varepsilon_2 \neq 0$, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

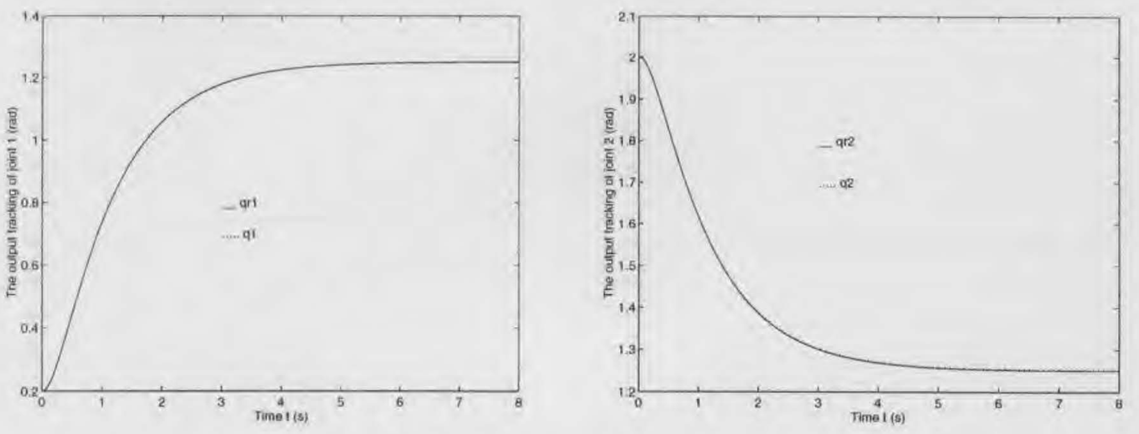

Figure 3.26: Simulation 1: Output tracking using a boundary layer, reference and joint values coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

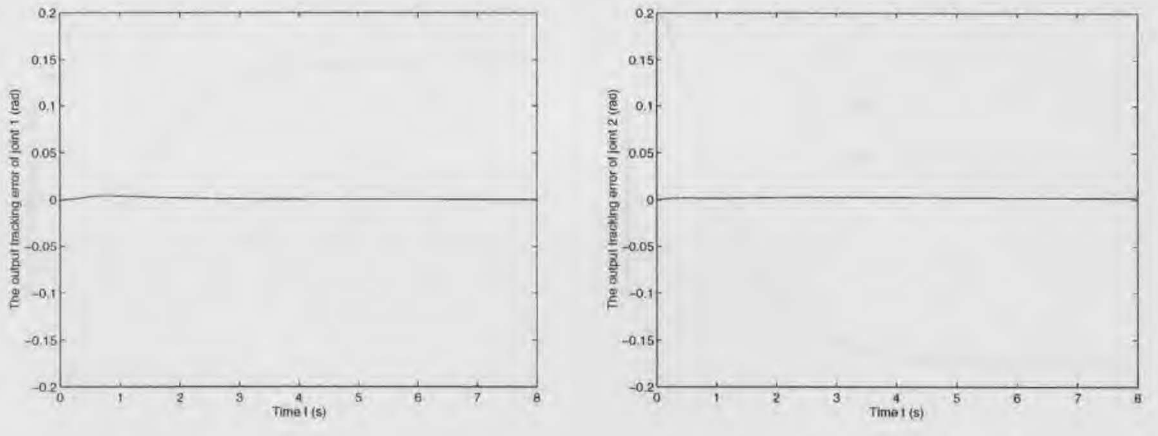

Figure 3.27: Simulation 1: Position errors using a boundary layer, reference and joint values coincide at the start, $\eta_i = 10$ and initially $\hat{k}_i = 5$, $1 \leqslant i \leqslant 7$.

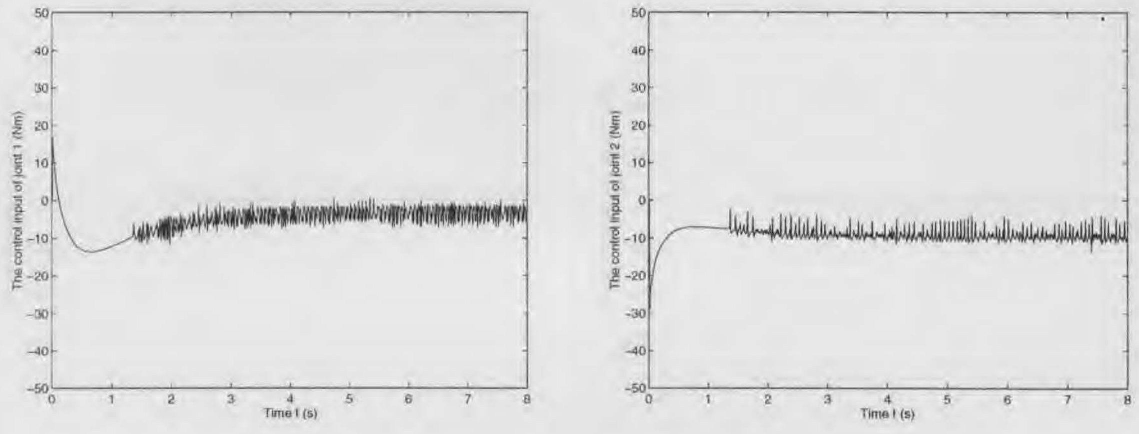

Figure 4.34: Simulation 2: Control input where reference and joint values coincide at the start and $\hat{b}_i = 0.5$, $0 \leqslant i \leqslant 2$.

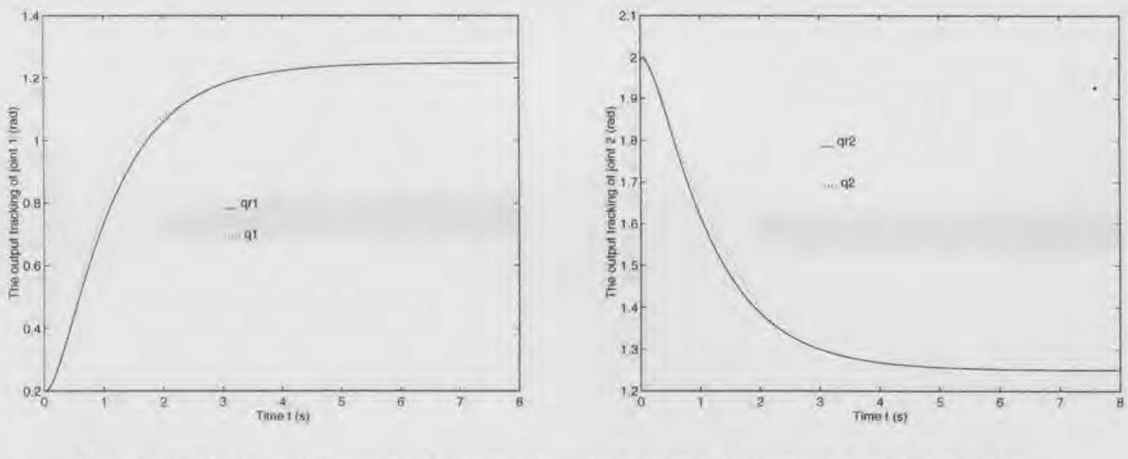

Figure 4.35: Simulation 2: Output tracking where reference and joint values coincide at the start and $\hat{b}_i = 0.05$, $0 \leqslant i \leqslant 2$.

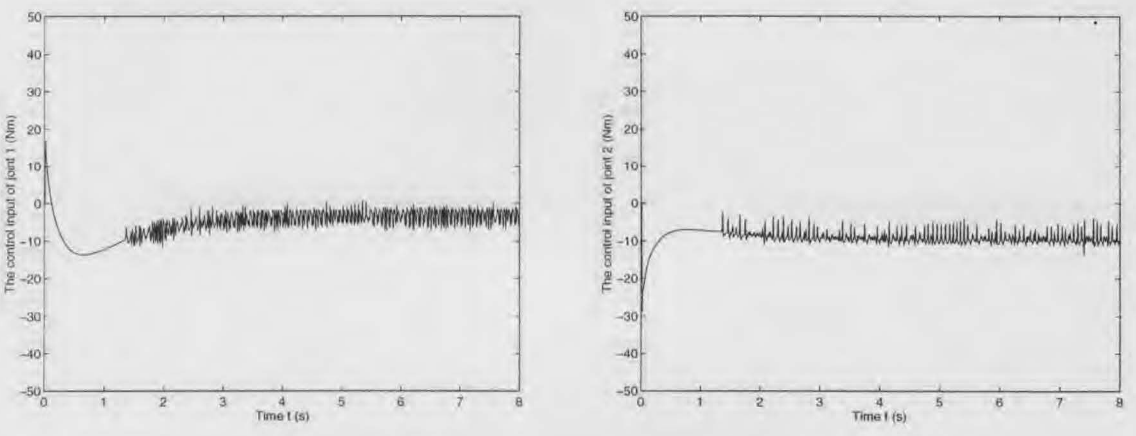

Figure 4.34: Simulation 2: Control input where reference and joint values coincide at the start and $\hat{b}_i = 0.5$, $0 \leqslant i \leqslant 2$.

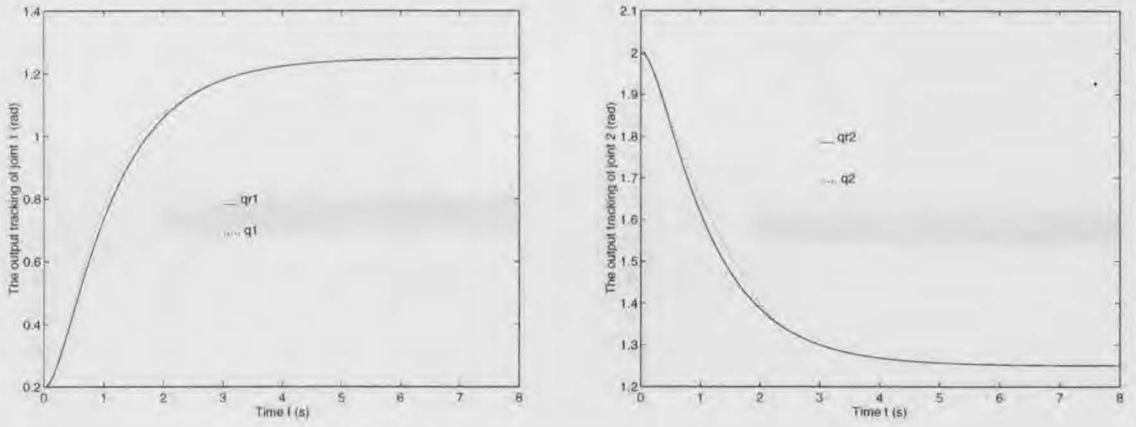

Figure 4.35: Simulation 2: Output tracking where reference and joint values coincide at the start and $\hat{b}_i = 0.05$, $0 \leqslant i \leqslant 2$.

3.5 Conclusion

In this chapter there has been developed a robust control law that guarantees finite time error convergence as well as finite convergence to the sliding variables. The control law does not require prior knowledge of the system bounds, instead using adaptive laws to estimate these bounds as the robot is in motion. Use of the boundary layer technique removes the problem of chattering, but can no longer guarantee absolute tracking, however the errors can be reduced to an arbitrarily small value by choice of the value of δ in expression 3.20.

4 A Control Scheme Utilizing Known Dynamics

4.1 Introduction

In this chapter there will be developed a robust tracking scheme that, unlike the one in the previous chapter, will deal with bounded disturbances as well as with uncertainties in parameters. As with the other scheme, it is robust and provides asymptotic error convergence in finite time.

The system is treated as a partially known system. The known dynamics are linearized and a nominal feedback controller is designed to stabilise the nominal system. The effects of the unknown dynamics are compensated for by use of a sliding mode adaptive compensator that estimates parameter bounds on-line. Thus we will be considering the control input in two parts (Shoureshi, Momot, and Roesler 1990) with

$$u = u_1 + u_0, \tag{4.2}$$

where u_1 is the nominal input and u_0 is the compensator that will be designed to deal with uncertainties.

The control law in this chapter is based on the expression of motion as defined in Kim, Lee, Park, and Youn (1993), the control law of Man, Paplinski, and Wu (1994), and the adaptive methods and method of proof used in Su and Leung (1993).

Equation 1.2 for the motion of a rigid robotic manipulator can be written in the following form (Man, Paplinski, and Wu 1994)

$$M(q)\ddot{q} + h(q,\dot{q}) = u(t), \tag{4.3}$$

where $h(q,\dot{q}) \in \mathbf{R}^{n \times 1}$ is the vector of combined Coriolis, centrifugal and gravitational torques.

If we then include disturbances in the expression we can define the robot motion as

$$M(q)\ddot{q} + h(q,\dot{q}) = u(t) + d(t), \tag{4.4}$$

where $d(t) \in \mathbf{R}^{n \times 1}$ is the vector of bounded disturbances (Spong and Vidyasagar 1987).

The assumptions required for the controller presented in this chapter are as follows:

Assumption 4.1 $\|M(q)\| < \alpha_0 , \quad \alpha_0 > 0.$

Assumption 4.2 $\|h(q,\dot{q})\| < \beta_0 + \beta_1\|q\| + \beta_2\|\dot{q}\|^2 , \quad \beta_0, \beta_1, \beta_2 > 0.$

Assumption 4.3 $d(t) < d_1 , \quad d_1 > 0.$

Assumption 4.4 *The form of the control input vector $u(t)$ is chosen such that its norm satisfies the following inequality:*
$\|u(t)\| < \lambda_0 + \lambda_1\|q\| + \lambda_2\|\dot{q}\|^2 , \quad \lambda_0, \lambda_1, \lambda_2 > 0.$

The first three of these assumptions are reasonable in terms of the characteristics of rigid robotic manipulators. The first of them has been used in Grimm (1990), and Shoureshi, Momot, and Roesler (1990), the second was used by Man and Palaniswami (1994) and the third in Kim, Lee, Park, and Youn (1993) and Park and Lee (1993). It will be seen that the controller designed in the next section satisfies assumption 4.4.

In the rest of the chapter, section 4.2 describes the new controller, section 4.3 describes the control law when a boundary layer is formed and section 4.4 describes the simulation used to test this controller. Section 4.5 then presents some conclusions.

4.2 Controller Design

The robotic manipulator described in expression 4.3 can be considered to have some known and some unknown parts, therefore we may define M and h with

$$M(q) = \Delta M(q) + M_0(q) \tag{4.5}$$

and

$$h(q,\dot{q}) = \Delta h(q,\dot{q}) + h_0(q,\dot{q}), \tag{4.6}$$

where $M_0(q)$ and $h_0(q,\dot{q})$ are the known parts and $\Delta M(q)$ and $\Delta h(q,\dot{q})$ are the unknown parts (Kim, Lee, Park, and Youn 1993).

Using expressions 4.5 and 4.6 we can rewrite expression 4.4 as

$$M_0(q)\ddot{q} + h_0(q, \dot{q}) = u(t) + \rho(t), \tag{4.7}$$

where

$$\rho(t) = -\Delta M(q)\ddot{q} - \Delta h(q, \dot{q}) + d(t). \tag{4.8}$$

Thus $\rho(t)$ contains all the system uncertainties.

As stated before, the system is to be considered as two separate systems, the nominal or known system with a control law designed using linearization and the unknown system designed as a sliding mode compensator. The nominal system is defined in Man and Palaniswami (1994) (see pages 31 to 32) and stabilised using

$$u_1 = h_0(q, \dot{q}) + M_0(q)(Ke + \ddot{q}_r), cnom \tag{4.9}$$

where $K = \begin{bmatrix} -K_1 & -K_2 \end{bmatrix}$ ($K_1, K_2 \in \mathbf{R}^{"\times"}$), and matrix K is designed such that

$$A_k = A + BK, \tag{4.10}$$

is an asymptotically stable matrix.

To design the variable structure compensator, the following assumptions will be used:

Assumption 4.5 *$M_0(q)$ is invertible for all q.*

Assumption 4.6 *The nominal system of expression 2.52 is stabilizable.*

Assumption 4.7 $\|(\mathbf{I} + \Delta M(q) M_0(q)^{-1})^{-1}\| < \alpha_1, \qquad \alpha_1 > 0.$

Assumption 4.8 $\|\Delta M(q) M_0(q)\| < \alpha_2, \qquad \alpha_2 > 0.$

Assumption 4.9 $\|h_0(q,\dot{q})\| < \alpha_3 + \alpha_4\|q\| + \alpha_5\|\dot{q}\|^2, \qquad \alpha_3, \alpha_4, \alpha_5 > 0.$

Assumption 4.10 $\|\Delta h(q,\dot{q})\| < \alpha_6 + \alpha_7\|q\| + \alpha_8\|\dot{q}\|^2, \qquad \alpha_6, \alpha_7, \alpha_8 > 0.$

The first two of these assumptions have been used by many researchers, for instance Grimm (1990), Man and Palaniswami (1994) and Shoureshi, Momot, and Roesler (1990). Assumptions 4.7 to 4.10 can be derived from assumptions 4.1 and 4.2.

The following lemma can now be proved:

Lemma 4.1 *The norm of the system uncertainties $\rho(t)$ in expression 4.7 satisfies the following inequality:*

$$\|\rho(t)\| < b_0 + b_1\|q\| + b_2\|\dot{q}\|^2. \tag{4.11}$$

Proof: Expression 4.7 can be rearranged to give

$$\ddot{q} = M_0(q)^{-1}(u(t) + \rho(t) - h_0(q,\dot{q})). \tag{4.12}$$

Substituting this into expression 4.8 we have

$$\begin{aligned}
\rho(t) &= -\Delta M(q)\ddot{q} - \Delta h(q,\dot{q}) + d(t) \\
&= -\Delta M(q)\left[M_0(q)^{-1}(u(t) + \rho(t) - h_0(q,\dot{q}))\right] - \Delta h(q,\dot{q}) + d(t) \\
&= -\Delta M(q)M_0(q)^{-1}u(t) - \Delta M(q)M_0(q)^{-1}\rho(t) + \Delta M(q)M_0(q)^{-1}h_0(q,\dot{q}) \\
&\quad -\Delta h(q,\dot{q}) + d(t),
\end{aligned}$$

rearranging this gives

$$\begin{aligned}
\rho(t) + \Delta M(q)M_0(q)^{-1}\rho(t) &= -\Delta M(q)M_0(q)^{-1}u(t) \\
&\quad +\Delta M(q)M_0(q)^{-1}h_0(q,\dot{q}) - \Delta h(q,\dot{q}) + d(t) \\
\therefore\ \rho(t)\left[\mathbf{I} + \Delta M(q)M_0(q)^{-1}\right] &= -\Delta M(q)M_0(q)^{-1}u(t) \\
&\quad +\Delta M(q)M_0(q)^{-1}h_0(q,\dot{q}) - \Delta h(q,\dot{q}) + d(t).
\end{aligned}$$

Thus $\rho(t)$ can be written as

$$\begin{aligned}
\rho(t) &= -\left(\mathbf{I} + \Delta M(q)M_0(q)^{-1}\right)^{-1}\Delta M(q)M_0(q)^{-1}u(t) \\
&\quad +\left(\mathbf{I} + \Delta M(q)M_0(q)^{-1}\right)^{-1}\Delta M(q)M_0(q)^{-1}h_0(q,\dot{q}) \\
&\quad -\left(\mathbf{I} + \Delta M(q)M_0(q)^{-1}\right)^{-1}\Delta h(q,\dot{q}) \\
&\quad +\left(\mathbf{I} + \Delta M(q)M_0(q)^{-1}\right)^{-1}d(t) \\
\therefore\ \|\rho(t)\| &= \| -\left(\mathbf{I} + \Delta M(q)M_0(q)^{-1}\right)^{-1}\Delta M(q)M_0(q)^{-1}u(t) \\
&\quad +\left(\mathbf{I} + \Delta M(q)M_0(q)^{-1}\right)^{-1}\Delta M(q)M_0(q)^{-1}h_0(q,\dot{q}) \\
&\quad -\left(\mathbf{I} + \Delta M(q)M_0(q)^{-1}\right)^{-1}\Delta h(q,\dot{q}) \\
&\quad +\left(\mathbf{I} + \Delta M(q)M_0(q)^{-1}\right)^{-1}d(t)\|
\end{aligned}$$

$$
\begin{aligned}
\therefore\ \|\rho(t)\| \leqslant\ & \|(\mathbf{I}+\Delta M(q)M_0(q)^{-1})^{-1}\Delta M(q)M_0(q)^{-1}u(t)\| \\
& +\|(\mathbf{I}+\Delta M(q)M_0(q)^{-1})^{-1}\Delta M(q)M_0(q)^{-1}h_0(q,\dot{q})\| \\
& +\|(\mathbf{I}+\Delta M(q)M_0(q)^{-1})^{-1}\Delta h(q,\dot{q})\| \\
& +\|(\mathbf{I}+\Delta M(q)M_0(q)^{-1})^{-1}d(t)\| \\
\leqslant\ & \|(\mathbf{I}+\Delta M(q)M_0(q)^{-1})^{-1}\|\|\Delta M(q)M_0(q)^{-1}\|\|u(t)\| \\
& +\|(\mathbf{I}+\Delta M(q)M_0(q)^{-1})^{-1}\|\|\Delta M(q)M_0(q)^{-1}\|\|h_0(q,\dot{q})\| \\
& +\|(\mathbf{I}+\Delta M(q)M_0(q)^{-1})^{-1}\|\|\Delta h(q,\dot{q})\| \\
& +\|(\mathbf{I}+\Delta M(q)M_0(q)^{-1})^{-1}\|\|d(t)\|
\end{aligned}
$$

Substituting from assumptions 4.3, 4.4 and 4.7 to 4.10, we have

$$
\begin{aligned}
\|\rho(t)\| \leqslant\ & \alpha_1\alpha_2(\lambda_0+\lambda_1\|q\|+\lambda_2\|\dot{q}\|^2) \\
& +\alpha_1\alpha_2(\alpha_3+\alpha_4\|q\|+\alpha_5\|\dot{q}\|^2) \\
& +\alpha_1(\alpha_6+\alpha_7\|q\|+\alpha_8\|\dot{q}\|^2) \\
& +\alpha_1 d_1 \\
\therefore\ \|\rho(t)\| \leqslant\ & b_0+b_1\|q\|+b_2\|\dot{q}\|^2,
\end{aligned}
$$

where

$b_0 = \alpha_1\alpha_2\lambda_0+\alpha_1\alpha_2\alpha_3+\alpha_1\alpha_6+\alpha_1 d_1,$

$b_1 = \alpha_1\alpha_2\lambda_1+\alpha_1\alpha_2\alpha_4+\alpha_1\alpha_7,$

$b_2 = \alpha_1\alpha_2\lambda_2+\alpha_1\alpha_2\alpha_5+\alpha_1\alpha_8.$

Remark 4.1 *In Man (1993) a method was developed for estimating the parameters of expression 4.11 off-line. However this was achieved by neglecting some system dynamics and did not include disturbances, which makes the estimates conservative and hence not very useful for practical purposes.*

The compensator u_0 can now be designed using the sliding mode technique, with adaptive estimation of the uncertainty bounds of expression 4.8 so that the requirement of the prior knowledge of system uncertainties is not necessary.

Let $\hat{b}_0$, $\hat{b}_1$ and $\hat{b}_2$ be the estimates of b_0, b_1 and b_2 of expression 4.11. The new adaptive laws for these estimates will then be defined as

$$\dot{\hat{b}}_0 = \kappa_0 \|CBM_0(q)^{-1}\| \|s\|, \tag{4.13}$$

$$\dot{\hat{b}}_1 = \kappa_1 \|CBM_0(q)^{-1}\| \|s\| \|q\| \tag{4.14}$$

and

$$\dot{\hat{b}}_2 = \kappa_2 \|CBM_0(q)^{-1}\| \|s\| \|\dot{q}\|^2, \tag{4.15}$$

where κ_i for $0 \leqslant i \leqslant 2$ are arbitrary positive numbers, $\hat{b}_i$ have arbitrary initial values, C is as defined in expression 2.38, and B in expression 2.55.

Parameters κ_i for $0 \leqslant i \leqslant 2$ in expressions 4.13 to 4.15 are called adaptive constants. They act to adjust the convergence rate of the adaptive laws in expressions 4.13 to 4.15. Faster convergence of the adaptive gains is achieved by increasing the value of these parameters, however, in practice, the speed of convergence is limited by the large size of the control gains that are generated by high values of κ_i.

To design the compensator u_0, to guarantee error convergence in finite time, we use the terminal switching plane variable vector described in expressions 2.36 to 2.42 on pages 27 to 28.

Theorem 4.1 *Consider the error dynamics of expression 2.59*

$$\dot{e} = A_k e + BM_0(q)^{-1}u_0 + BM_0(q)^{-1}\rho(t). \tag{4.16}$$

for the robotic manipulator system of expression 4.4 with assumptions 4.1 to 4.5. If the compensator u_0 is designed such that

$$u_0 = \begin{cases} \frac{\left(s^T CBM_0(q)^{-1}\right)^T}{\|s^T CBM_0(q)^{-1}\|^2} w & CBM_0(q)^{-1} \neq 0, \varepsilon_i \neq 0 \\ \mathbf{0} & otherwise \end{cases} \tag{4.17}$$

where

$$w = -s^T CA_k e - \|s\| \|CBM_0(q)^{-1}\| \left(\hat{b}_0 + \hat{b}_1\|q\| + \hat{b}_2\|\dot{q}\|^2\right) - s^T C_1 \dot{\bar{\varepsilon}} - s^T C_1 \dot{\varepsilon} \tag{4.18}$$

and $\hat{b}_i$ for $0 \leqslant i \leqslant 2$ are updated by the adaptive laws in expressions 4.13 to 4.15, then the output tracking error, $\varepsilon(t)$, will converge to zero in a finite time.

Proof: Using the Lyapunov function (Su and Leung 1993)

$$V = \frac{1}{2}s^T s + \frac{1}{2}\sum_{i=0}^{2} \kappa_i^{-1}\tilde{b}_i^2 \tag{4.19}$$

where

$$\tilde{b}_i = b_i - \hat{b}_i, \tag{4.20}$$

and differentiating V with respect to time, we have

$$\dot{V} = \frac{1}{2}s^T\dot{s} + \frac{1}{2}\dot{s}^T s - \sum_{i=0}^{2} \kappa_i^{-1}\tilde{b}_i\dot{\hat{b}}_i.$$

Since $\dot{s}^T s$ is scalar, it must be equal to its own transpose, therefore

$$\begin{aligned}\dot{V} &= \tfrac{1}{2}s^T\dot{s} + \tfrac{1}{2}\dot{s}^T s - \sum_{i=0}^{2} \kappa_i^{-1}\tilde{b}_i\dot{\hat{b}}_i \\ &= \tfrac{1}{2}s^T\dot{s} + \tfrac{1}{2}(\dot{s}^T s)^T - \sum_{i=0}^{2} \kappa_i^{-1}\tilde{b}_i\dot{\hat{b}}_i \\ &= \tfrac{1}{2}s^T\dot{s} + \tfrac{1}{2}s^T\dot{s} - \sum_{i=0}^{2} \kappa_i^{-1}\tilde{b}_i\dot{\hat{b}}_i \\ &= s^T\dot{s} - \sum_{i=0}^{2} \kappa_i^{-1}\tilde{b}_i\dot{\hat{b}}_i.\end{aligned} \tag{4.21}$$

Substituting for $\dot{s}$ from expression 3.2

$$\dot{V} = s^T(C\dot{e} + C_1\dot{\hat{\varepsilon}} - C_1\dot{\varepsilon}) - \sum_{i=0}^{2} \kappa_i^{-1}\tilde{b}_i\dot{\hat{b}}_i,$$

and then substituting $\dot{e}$ from expression 2.59 we have

$$\begin{aligned}\dot{V} &= s^T(CA_ke + CBM_0(q)^{-1}u_0 + CBM_0(q)^{-1}\rho(t) + C_1\dot{\hat{\varepsilon}} - C_1\dot{\varepsilon}) \\ &\quad - \sum_{i=0}^{2} \kappa_i^{-1}\tilde{b}_i\dot{\hat{b}}_i \\ &= s^TCA_ke + s^TCBM_0(q)^{-1}u_0 + s^TCBM_0(q)^{-1}\rho(t) + s^TC_1\dot{\hat{\varepsilon}} \\ &\quad - s^TC_1\dot{\varepsilon} - \sum_{i=0}^{2} \kappa_i^{-1}\tilde{b}_i\dot{\hat{b}}_i.\end{aligned}$$

Now consider the control law defined for the compensator in expression 4.17. Assuming $CBM_0(q)^{-1} \neq 0$, we have

$$u_0 = \frac{\left(s^T CBM_0(q)^{-1}\right)^T}{\|s^T CBM_0(q)^{-1}\|^2} w.$$

Rearranging this expression we get

$$w = \frac{\|s^T CBM_0(q)^{-1}\|^2}{\left(s^T CBM_0(q)^{-1}\right)^T} u_0.$$

Since $s^T CBM_0(q)^{-1}$ is a $1 \times n$ vector

$$\|s^T CBM_0(q)^{-1}\|^2 = \left[s^T CBM_0(q)^{-1}\right] \left[s^T CBM_0(q)^{-1}\right]^T.$$

therefore

$$w = s^T CBM_0(q)^{-1} u_0 \qquad\qquad s^T \neq 0.$$

Substituting this into 4.2 we get

$$\begin{aligned} \dot{V} &= s^T CA_k e + w + s^T CBM_0(q)^{-1} \rho(t) + s^T C_1 \dot{\hat{\varepsilon}} \\ &\quad - s^T C_1 \dot{\varepsilon} - \sum_{i=0}^{2} \kappa_i^{-1} \tilde{b}_i \dot{\hat{b}}_i. \end{aligned}$$

and substituting w from expression 4.18 we have

$$\begin{aligned}
\dot{V} &= s^T C A_k e + \Big\{ -s^T C A_k e - \|s\| \|CBM_0(q)^{-1}\| \left(\hat{b}_0 + \hat{b}_1 \|q\| + \hat{b}_2 \|\dot{q}\|^2 \right) \\
&\quad - s^T C_1 \dot{\hat{\varepsilon}} + s^T C_1 \dot{\varepsilon} \Big\} + s^T C B M_0(q)^{-1} \rho(t) + s^T C_1 \dot{\hat{\varepsilon}} - s^T C_1 \dot{\varepsilon} \\
&\quad - \sum_{i=0}^{2} \kappa_i^{-1} \tilde{b}_i \dot{\hat{b}}_i \\
&= -\|s\| \|CBM_0(q)^{-1}\| \left(\hat{b}_0 + \hat{b}_1 \|q\| + \hat{b}_2 \|\dot{q}\|^2 \right) + s^T C B M_0(q)^{-1} \rho(t) \\
&\quad - \sum_{i=0}^{2} \kappa_i^{-1} \tilde{b}_i \dot{\hat{b}}_i .
\end{aligned}$$

Substituting for $\tilde{b}_i$ from expression 4.20

$$\begin{aligned}
\dot{V} &= -\|s\| \|CBM_0(q)^{-1}\| \left(\hat{b}_0 + \hat{b}_1 \|q\| + \hat{b}_2 \|\dot{q}\|^2 \right) + s^T C B M_0(q)^{-1} \rho(t) \\
&\quad -\kappa_0^{-1} (b_0 - \hat{b}_0) \dot{\hat{b}}_0 - \kappa_1^{-1} (b_0 - \hat{b}_1) \dot{\hat{b}}_1 - \kappa_2^{-1} (b_0 - \hat{b}_2) \dot{\hat{b}}_2 ,
\end{aligned}$$

and then substituting the adaptive laws defined in expressions 4.13 to 4.15, we get

$$\begin{aligned}
\dot{V} &= -\|s\| \|CBM_0(q)^{-1}\| \left(\hat{b}_0 + \hat{b}_1 \|q\| + \hat{b}_2 \|\dot{q}\|^2 \right) + s^T C B M_0(q)^{-1} \rho(t) \\
&\quad -\kappa_0^{-1} (b_0 - \hat{b}_0) \kappa_0 \|CBM_0(q)^{-1}\| \|s\| \\
&\quad -\kappa_1^{-1} (b_0 - \hat{b}_1) \kappa_1 \|CBM_0(q)^{-1}\| \|s\| \|q\| \\
&\quad -\kappa_2^{-1} (b_0 - \hat{b}_2) \kappa_2 \|CBM_0(q)^{-1}\| \|s\| \|\dot{q}\|^2 \\
&= -\|s\| \|CBM_0(q)^{-1}\| \left(\hat{b}_0 + \hat{b}_1 \|q\| + \hat{b}_2 \|\dot{q}\|^2 \right) + s^T C B M_0(q)^{-1} \rho(t) \\
&\quad -b_0 \|CBM_0(q)^{-1}\| \|s\| + \hat{b}_0 \|CBM_0(q)^{-1}\| \|s\| \\
&\quad -b_1 \|CBM_0(q)^{-1}\| \|s\| \|q\| + \hat{b}_1 \|CBM_0(q)^{-1}\| \|s\| \|q\| \\
&\quad -b_2 \|CBM_0(q)^{-1}\| \|s\| \|\dot{q}\|^2 + \hat{b}_2 \|CBM_0(q)^{-1}\| \|s\| \|\dot{q}\|^2
\end{aligned}$$

$$\begin{aligned}
\therefore\ \dot{V} &= -\|s\|\|CBM_0(q)^{-1}\|\left(\hat{b}_0+\hat{b}_1\|q\|+\hat{b}_2\|\dot{q}\|^2\right)+s^TCBM_0(q)^{-1}\rho(t)\\
&\quad -\|s\|\|CBM_0(q)^{-1}\|\left(b_0+b_1\|q\|+b_2\|\dot{q}\|^2\right)\\
&\quad +\|s\|\|CBM_0(q)^{-1}\|\left(\hat{b}_0+\hat{b}_1\|q\|+\hat{b}_2\|\dot{q}\|^2\right)\\
&= -\|s\|\|CBM_0(q)^{-1}\|\left(b_0+b_1\|q\|+b_2\|\dot{q}\|^2\right)+s^TCBM_0(q)^{-1}\rho(t)\\
&\leqslant -\|s\|\|CBM_0(q)^{-1}\|\left(b_0+b_1\|q\|+b_2\|\dot{q}\|^2\right)+\|s\|\|CBM_0(q)^{-1}\|\|\rho(t)\|\\
&= -\|s\|\|CBM_0(q)^{-1}\|\left\{\left(b_0+b_1\|q\|+b_2\|\dot{q}\|^2\right)-\|\rho(t)\|\right\}.
\end{aligned}$$

Using Lemma 4.1 we can now state that

$$\dot{V} < 0 \quad \|s\| \neq 0. \tag{4.22}$$

Using Lyapunov's second method, expression 4.22 means that the switching plane variable vector s converges to zero in a finite time and, on the sliding mode, the output tracking error converges to zero in a finite time.

The control law given in expression 4.17 is bounded in error space. When it is in the condition given by $CBM_0(q)^{-1} \neq 0$ when $\varepsilon_i = 0$, it is possible that a control input of $u = 0$ will drive ε_i away from 0. In this case a different part of the variable structure controll law will drive the vector s to the terminal sliding mode, where the desired error dynamics will then be reached.

This result is an improvement over those in Young (1978), Freund (1982), Spong and Vidyasagar (1987), Al-Abbass and Ozguner (1985), Morgan and Ozguner (1985), Shoureshi, Momot, and Roesler (1990), Fu (1992), Dorling and Zinober (1986), Man and Palaniswami (1994), Man and Palaniswami (1993) and Man and Palaniswami (1995) because it ensures that the output tracking error can be driven into the terminal sliding mode in a finite time and also that the error dynamics can reach the system origin in a finite time once on the terminal sliding mode. It is also a more practical scheme because only three parameters of the upper bounds of the system uncertainties need to be estimated and this holds true for any n-link rigid robotic manipulator.

In practical terms the success of the proposed law can be explained thus: when the output tracking errors are large, the estimates $\hat{b}_i$ where $i = 0, 1, 2$, can be automatically increased using the laws in expressions 4.13 to 4.15. This increases the control gain which pushes the error towards zero. Since these expressions are estimates in Lyapunov sense of the upper bounds of the system uncertainty represented by $\rho(t)$, it is not necessary for them to converge to their true values. They simply need to be adjusted until the switching plane vector s converges to zero, at which point they become constants that ensure that the error dynamics remain on the terminal sliding mode.

The final control law for the system is therefore the sum of the nominal control law plus the adaptive sliding mode compensator:

$$\begin{aligned} u(t) &= u_1 + u_0 \\ &= h_0(q, \dot{q}) + M_0(q)(Ke + \ddot{q}_r) + \begin{cases} \frac{\left(s^T CBM_0(q)^{-1}\right)^T}{\|s^T CBM_0(q)^{-1}\|^2} w & CBM_0(q)^{-1} \neq 0 \\ 0 & CBM_0(q)^{-1} = 0, \end{cases} \end{aligned} \tag{4.23}$$

where $K = \begin{bmatrix} -K_1 & -K_2 \end{bmatrix}$ ($K_1, K_2 \in \mathbf{R}^{n \times n}$), with matrix K designed such that

$$A_k = A + BK, \tag{4.24}$$

is an asymptotically stable matrix and

$$w = -s^T C A_k e - \|s\| \|CBM_0(q)^{-1}\| \left(\hat{b}_0 + \hat{b}_1\|q\| + \hat{b}_2\|\dot{q}\|^2\right) - s^T C_1 \ddot{\xi} - s^T C_1 \dot{\varepsilon}. \tag{4.25}$$

4.3 Reducing Chattering

The compensator control law described in 4.17 is discontinuous across the line $CBM_0(q)^{-1} = 0$ and this causes chattering to occur. If this control law is replaced by the control law described in 4.26 below, chattering is eliminated, but the error can no longer remain exactly zero. It can, however, be reduced to an arbitrarily small amount.

$$u_0 = \begin{cases} \dfrac{\left(s^T CBM_0(q)^{-1}\right)^T}{\|s^T CBM_0(q)^{-1}\|^2}\, w & CBM_0(q)^{-1} \geqslant \delta \\[2ex] \dfrac{\left(s^T CBM_0(q)^{-1}\right)^T}{\delta^2}\, w & CBM_0(q)^{-1} < \delta, \end{cases} \tag{4.26}$$

where δ is a positive number and w is as defined in expression 4.18. The nominal system control law does not change.

4.4 Simulation

The simulation was performed using the same manipulator as that described in Section 3.4 and shown in figure 3.7. The MatLab code can be found in Appendix C.

To simulate system uncertainties, the nominal values of the masses of the two links are assumed to be

$$\hat{m}_1 = 0.4\text{kg} \quad \text{and} \quad \hat{m}_2 = 1.2\text{kg}$$

instead of the 'correct' values of 0.5kg and 1.5kg respectively.

If we let the desired error dynamics of the closed loop nominal system have the following form

$$\ddot{\varepsilon}_i + 5\dot{\varepsilon}_i + 4\varepsilon_i = 0 \quad i = 1,2$$

then, using the pole placement method for expression 2.58, the feedback matrix K can be designed as

$$K = \begin{bmatrix} -4 & 0 & -5 & 0 \\ 0 & -4 & 0 & -5 \end{bmatrix}$$

The values of the constants κ_i $(i = 0, 1, 2)$ were arbitrarily set to 10.

The initial value chosen for the starting point of $\hat{b}_i$, $0 \leqslant i \leqslant 2$ was 0.5. From the graphs in figures 4.32 to 4.34 it can be seen that this was a good choice. It can be seen clearly that the adaptive estimates of the bounds of system uncertainties deal with the incorrect nominal values of the masses of the two joints, since the actual position and reference position are indistinguishable from each other. The errors in the two joints clearly converge to zero and the control inputs are very low, although the chattering effect can be seen in the fast switching from positive to negative values.

The control law was also tested with a poor starting value for $\hat{b}_i$, 0.05. It can be seen from the graphs (in figures 4.35 to 4.37) that the adaptive scheme now takes longer to produce accurate tracking, however error convergence is still achieved in approximately 2 seconds.

Figures 4.38 to 4.40 show the control law's response to an initial robot position of $q_1 = 0.4$ and $q_2 = 1.8$, instead of being equal to the initial reference model values. As can be seen accurate tracking is achieved within 2 seconds of the start.

The results of the final unbounded simulation are shown in figures 4.41 to 4.43. These show the 'worst-case' scenario of using an incorrect starting value for $\hat{b}_i$ as well as non-matching starting points for the robot and the reference model. Despite these problems, the tracking error converges to zero in under 2 seconds.

For the boundary layer simulation, δ from expression 4.26 is set to 0.015. Using $\hat{b}_i = 0.5$ for $0 \leqslant i \leqslant 2$, and the same initial position for the robot and the reference model, the simulation shows that the chattering caused by the discontinuity in the control law has now been removed (see figure 4.46). The non-zero error can be seen in figures 4.44 and 4.45, where the path of the robot and the reference model no longer exactly coincide and the output tracking errors no longer converge to zero, instead being kept within a boundary layer near zero.

Finally the simulation has been performed for the 'worst case' (as defined above) while using a boundary layer (see figures 4.47 to 4.49). Again it can be seen that the error converges to a region around zero in around 2 seconds and the chattering has been removed.

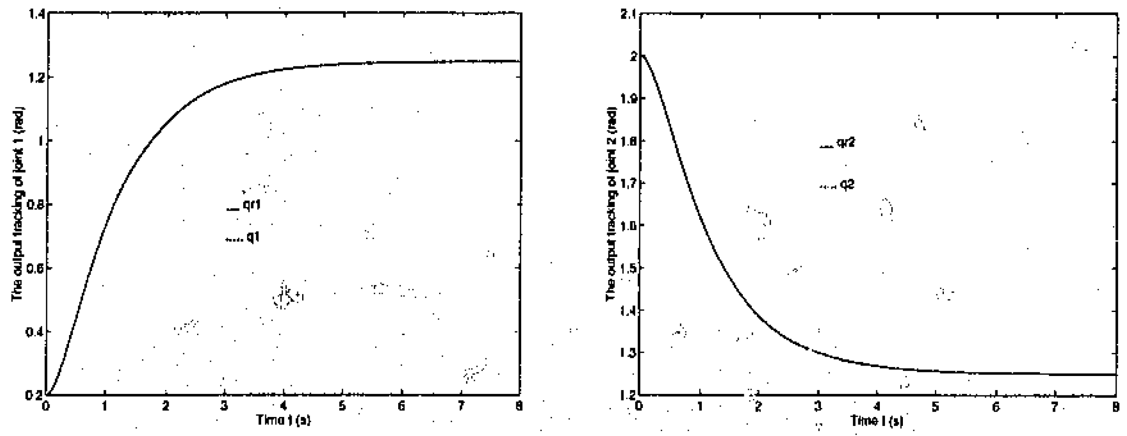

Figure 4.32: Simulation 2: Output tracking where reference and joint values coincide at the start and $\hat{b}_i = 0.5$, $0 \leqslant i \leqslant 2$.

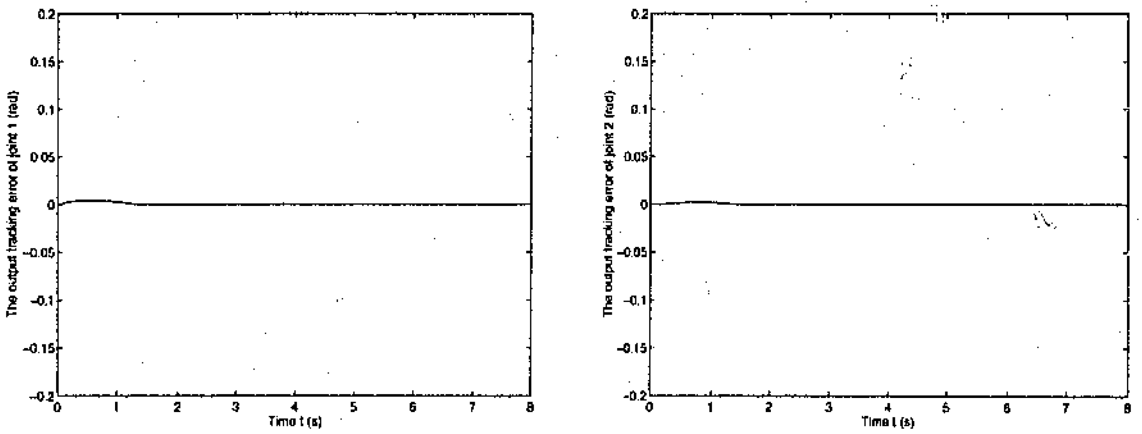

Figure 4.33: Simulation 2: Position errors where reference and joint values coincide at the start and $\hat{b}_i = 0.5$, $0 \leqslant i \leqslant 2$.

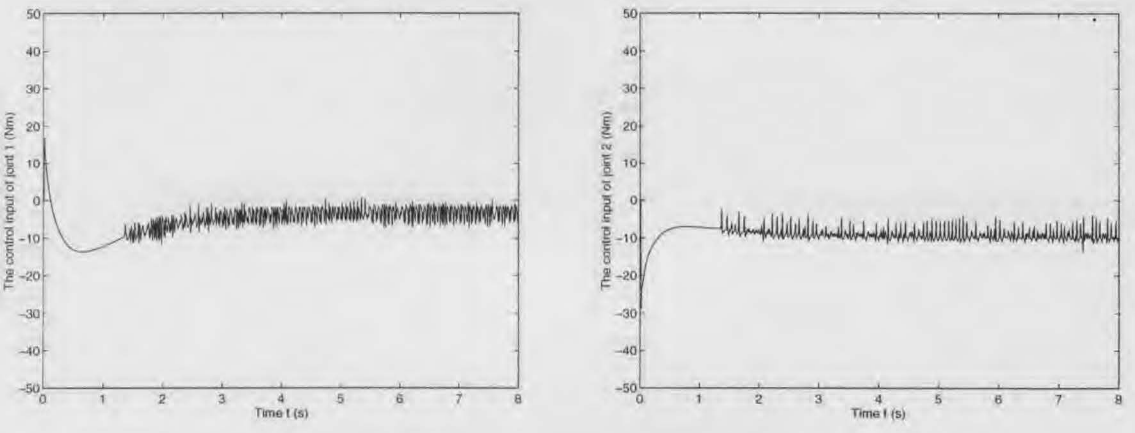

Figure 4.34: Simulation 2: Control input where reference and joint values coincide at the start and $\hat{b}_i = 0.5$, $0 \leqslant i \leqslant 2$.

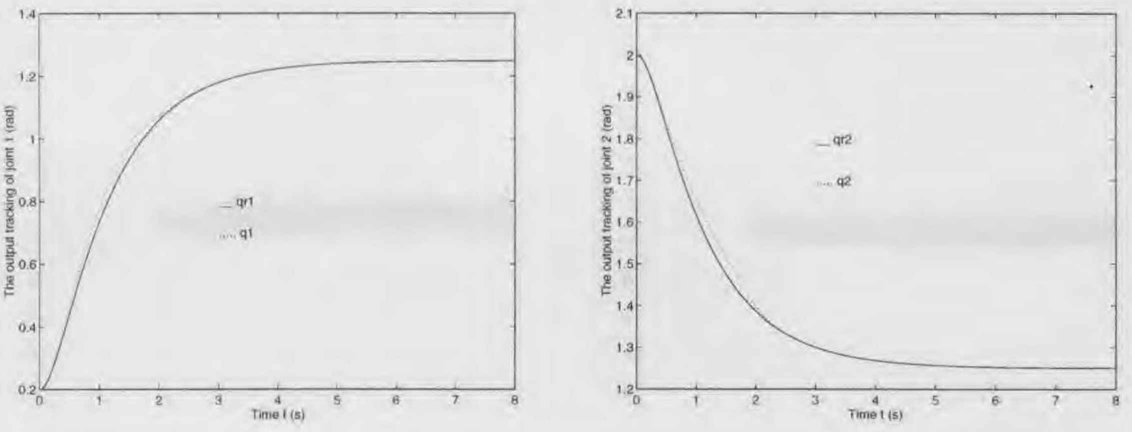

Figure 4.35: Simulation 2: Output tracking where reference and joint values coincide at the start and $\hat{b}_i = 0.05$, $0 \leqslant i \leqslant 2$.

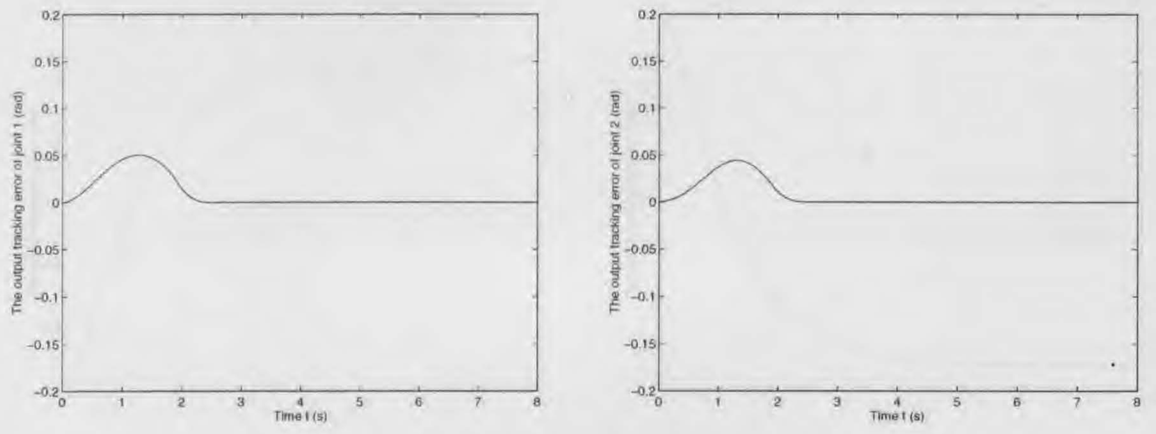

Figure 4.36: Simulation 2: Position errors where reference and joint values coincide at the start and $\hat{b}_i = 0.05$, $0 \leqslant i \leqslant 2$.

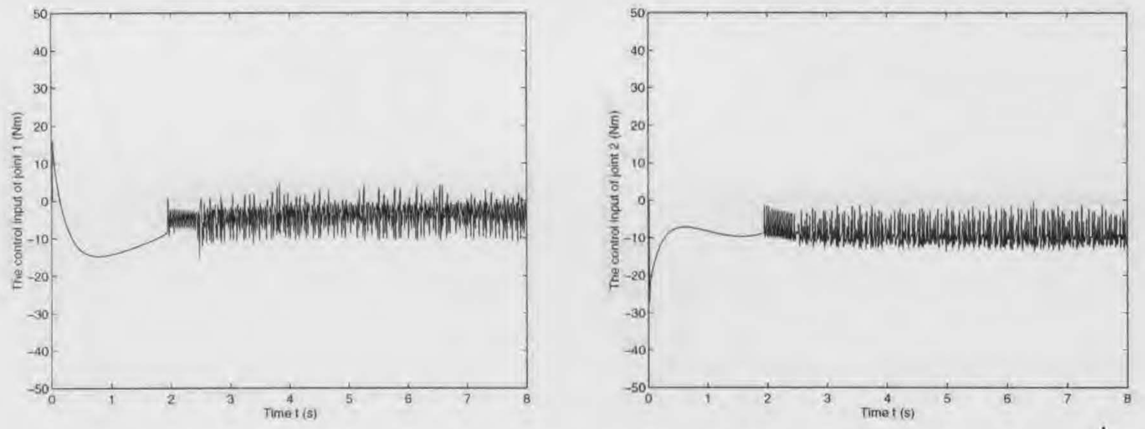

Figure 4.37: Simulation 2: Control input where reference and joint values coincide at the start and $\hat{b}_i = 0.05$, $0 \leqslant i \leqslant 2$.

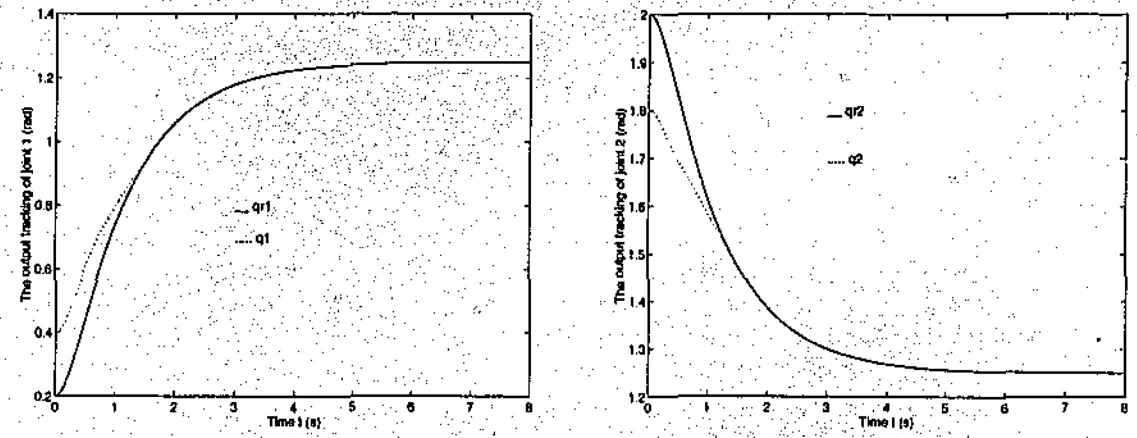

Figure 4.38: Simulation 2: Output tracking where reference and joint values do not coincide at the start and $\hat{b}_i = .5,\ 0 \leqslant i \leqslant 2$.

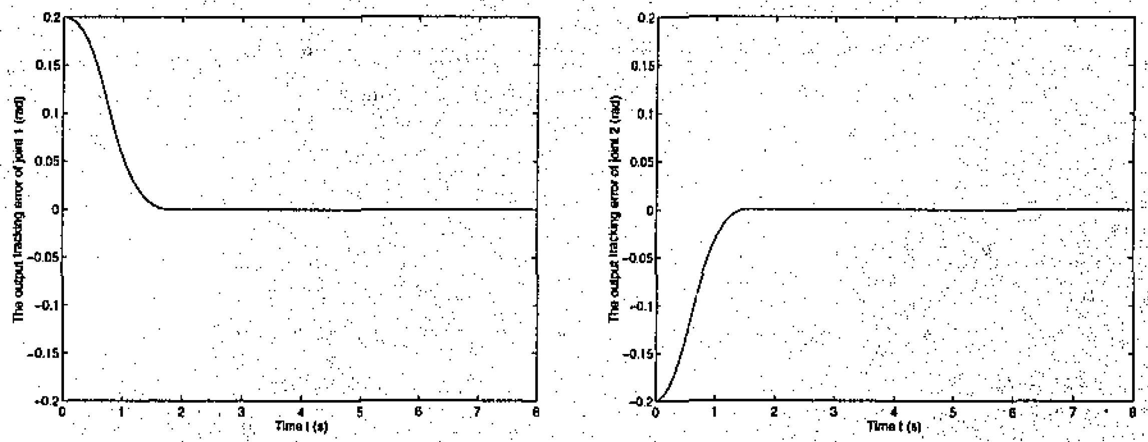

Figure 4.39: Simulation 2: Position errors where reference and joint values do not coincide at the start and $\hat{b}_i = .5,\ 0 \leqslant i \leqslant 2$.

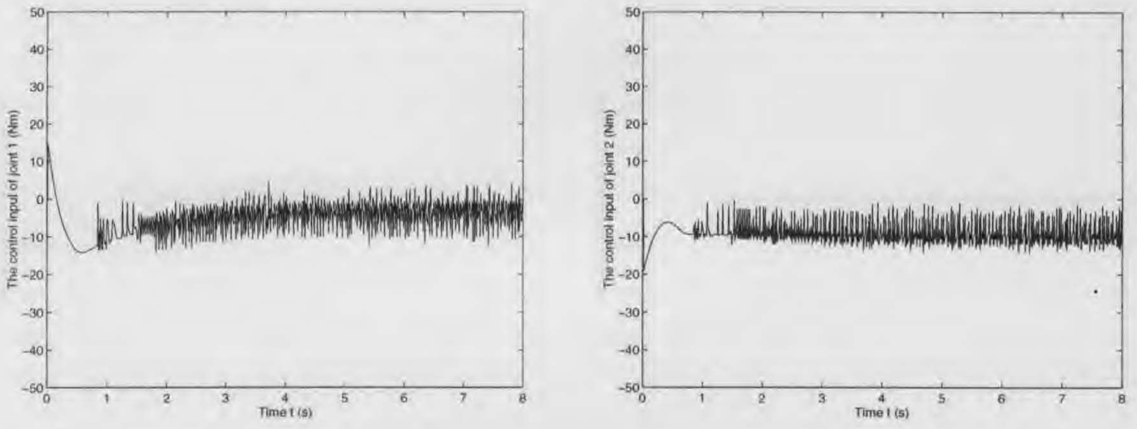

Figure 4.40: Simulation 2: Control input where reference and joint values do not coincide at the start and $\hat{b}_i = .5$, $0 \leqslant i \leqslant 2$.

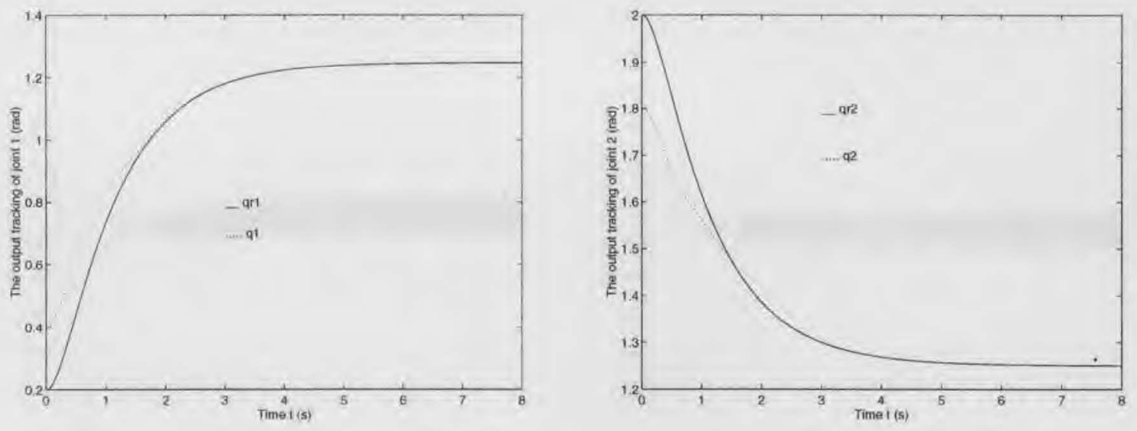

Figure 4.41: Simulation 2: Output tracking where reference and joint values do not coincide at the start and $\hat{b}_i = .05$, $0 \leqslant i \leqslant 2$.

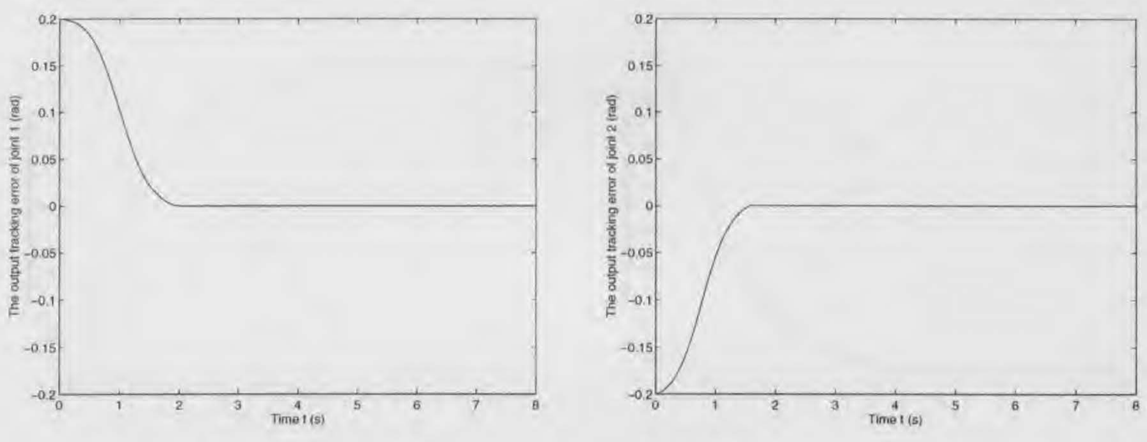

Figure 4.42: Simulation 2: Position errors where reference and joint values do not coincide at the start and $\hat{b}_i = .05$, $0 \leqslant i \leqslant 2$.

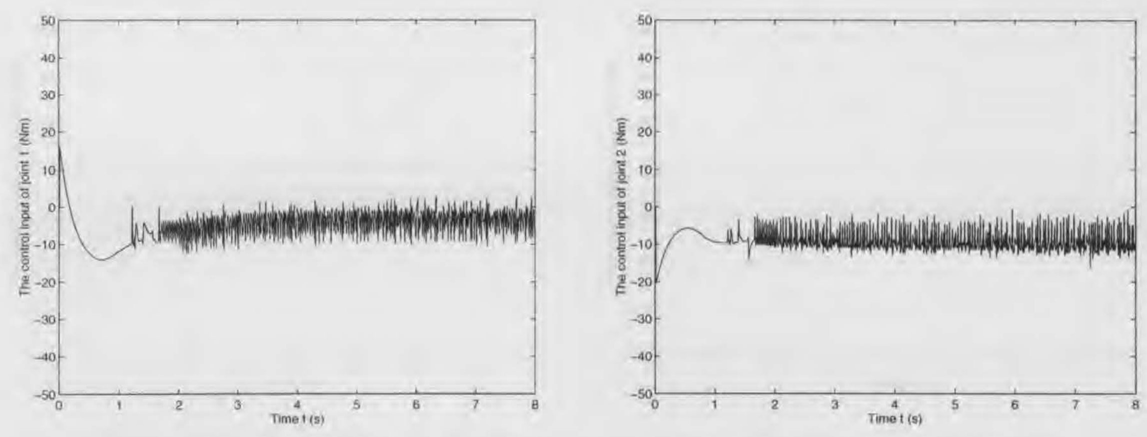

Figure 4.43: Simulation 2: Control input where reference and joint values do not coincide at the start and $\hat{b}_i = .05$, $0 \leqslant i \leqslant 2$.

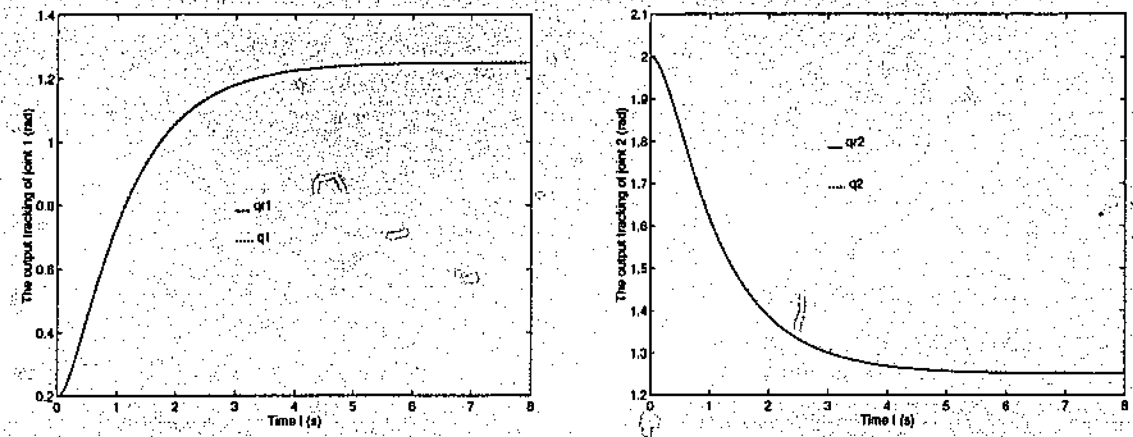

Figure 4.44: Simulation 2: Output tracking where a boundary layer is used, reference and joint values coincide at the start and $\hat{b}_i = .5,\ 0 \leqslant i \leqslant 2$.

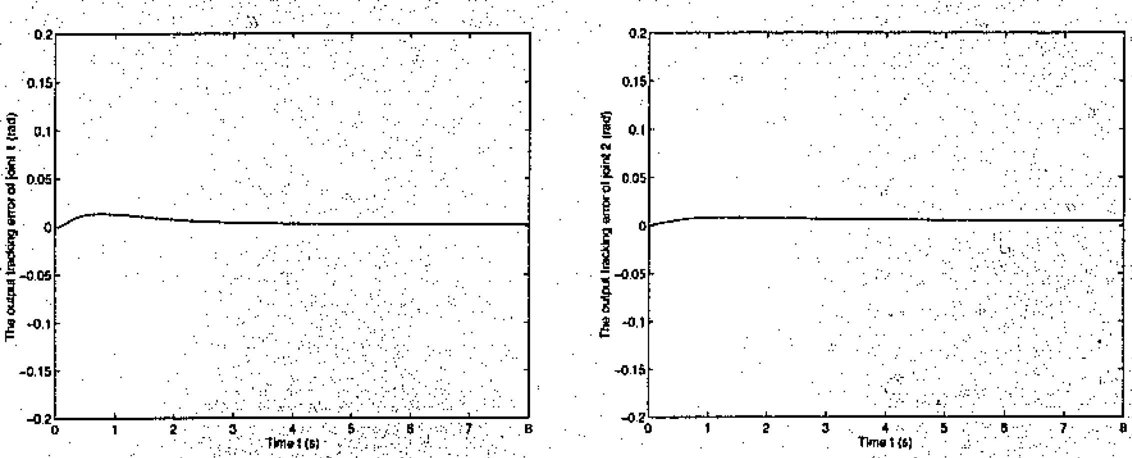

Figure 4.45: Simulation 2: Position errors where a boundary layer is used, reference and joint values coincide at the start and $\hat{b}_i = .5,\ 0 \leqslant i \leqslant 2$.

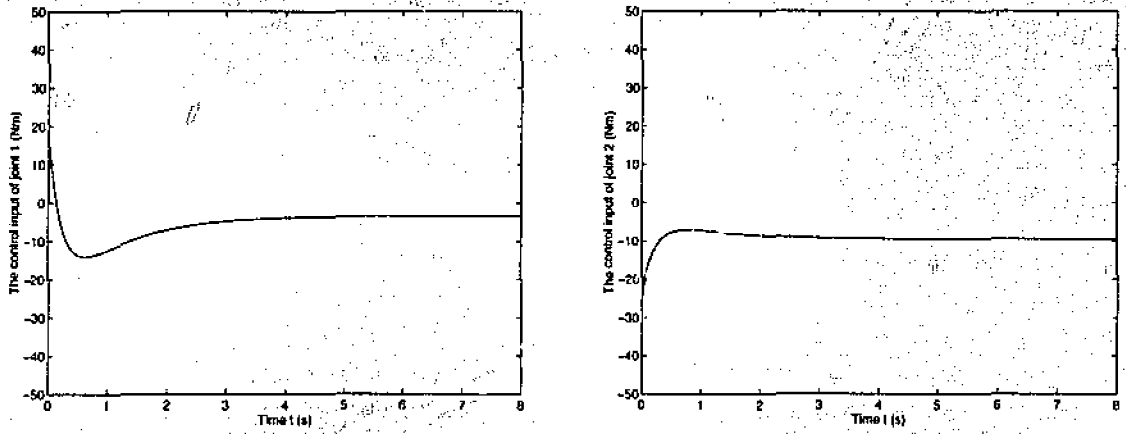

Figure 4.46: Simulation 2: Control input where a boundary layer is used, reference and joint values coincide at the start and $\hat{b}_i = .5$, $0 \leqslant i \leqslant 2$.

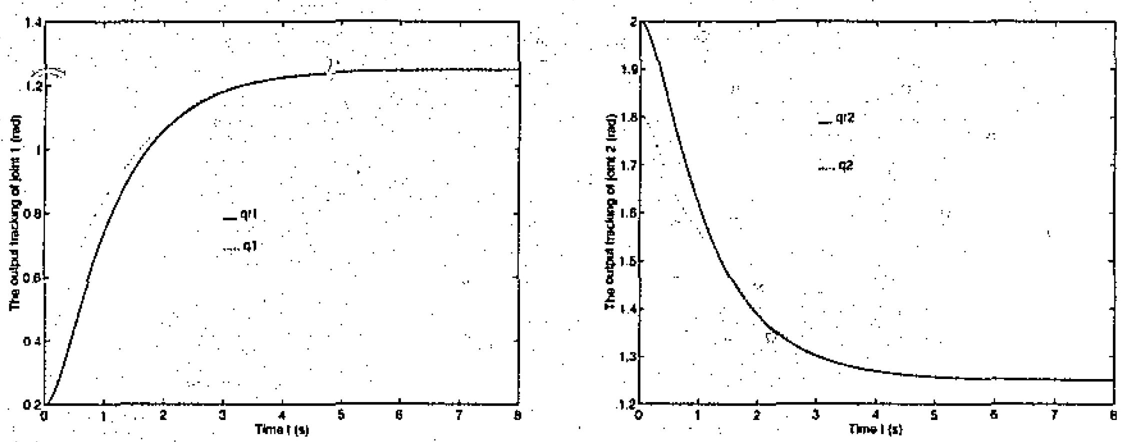

Figure 4.47: Simulation 2: Output tracking where a boundary layer is used, reference and joint values do not coincide at the start and $\hat{b}_i = .05$, $0 \leqslant i \leqslant 2$.

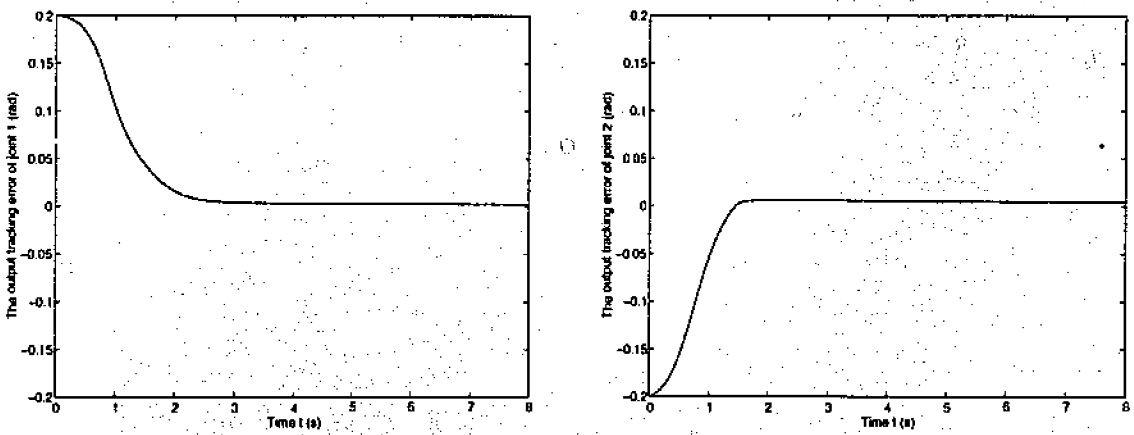

Figure 4.48: Simulation 2: Position errors where a boundary layer is used, reference and joint values do not coincide at the start and $\hat{b}_i = .05$, $0 \leqslant i \leqslant 2$.

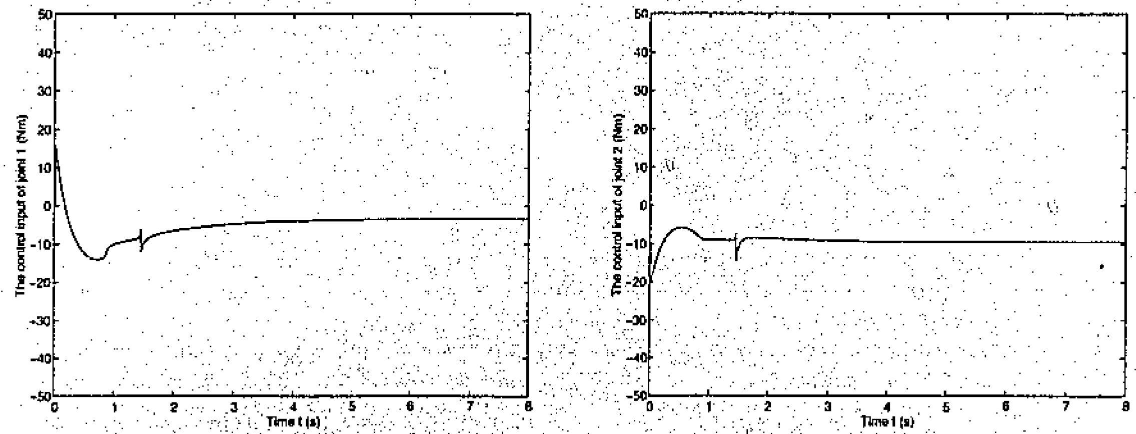

Figure 4.49: Simulation 2: Control input where a boundary layer is used, reference and joint values do not coincide at the start and $\hat{b}_i = .05$, $0 \leqslant i \leqslant 2$.

4.5 Conclusion

In this chapter there has been developed a robust control law that guarantees finite time error convergence. It assumes both system uncertainties and disturbances and is based on a control law made up of a nominal input for the known part of the system and a sliding mode compensator for the unknown part of the system. The results indicate that the control law gives excellent tracking. The boundary layer technique is used to remove the problem of chattering in the control input, however this can no longer ensure convergence of the tracking error to zero, only to an arbitrarily small boundary around zero.

5 Conclusion

This thesis presents two new control laws for rigid robotic manipulators with uncertain dynamics. The first law, described in Chapter 3 is based on that of (Man, Paplinski, and Wu 1994) which gives a robust variable structure control law with proven asymptotic error convergence and finite convergence to the sliding mode. The improvement that chapter 3 details is the use of adaptive techniques that adjust the seven system parameters required for the control law so that no prior knowledge of uncertain parameters is required. Simulations show the accuracy of the tracking produced with this control law. A boundary layer controller is then used to reduce the chattering effect caused by the variable structure nature of the controller and simulations are again used to show its effectiveness both for tracking and for removing the chattering. The convergence of the error to zero can no longer be guaranteed, however it can be confined to an arbitrarily small boundary around zero.

The second law, detailed in Chapter 4, includes the problem of external disturbances to the system and utilizes known information about the system by dividing it up into known and unknown parts. The known (nominal) part is then stabilized with a standard feedback controller, whilst the unknown part is dealt with by use of an adaptive variable structure compensator. The compensator only requires three unknown parameters to be adaptively adjusted and is simple to implement. The resulting control law is robust and has finite time error convergence. A boundary layer compensator is implemented to solve the problem of chattering and simulation results are given showing the effectiveness of both the original and boundary layer control laws.

The two controllers are improvements over prior methods as they do not require bounds on system uncertainties, ensure that the reaching phase of the system is completed in finite time, and are simple to implement.

References

Abdallah, A., Dawson, D., Dorato, P., and Jamishidi, M. (1991). Survey of robust control for rigid robots. *IEEE Control Systems Magazine 11*(2), 24–30.

Al-Abbass, F. and Ozguner, U. (1985). Decentralised model reference adaptive system using a variable structure control. In *Proceedings of the 24th Conference on Decision and Control*, pp. 1473–1478.

Amestegui, M., Ortega, R., and Ibarra, J. (1987). Adaptive linearizing-decoupling robotic control: A comparative study of different parametrizations. In *Proceedings of the 5th Yale Workshop on Applications of Adaptive Systems Theory*, pp. 74–80.

Bailey, E. and Arapostathis, A. (1987). Simple sliding mode control scheme applied to robot manipulator. *International Journal of Control 45*(4), 1197–1209.

Balestrino, A., Maria, G. D., and Sciavicco, L. (1983). An adaptive model following control for robotic manipulators. *Journal of Dynamic Systems, Measurement and Control 105*, 143–151.

Chern, T.-L. and Wu, Y.-C. (1992, March). Integral variable structure control approach for robot manipulators. *IEE Proceedings-D (Control Theory and Applications) 139*(2), 161–166.

Corless, M. and Leitmann, G. (1981). Continous state feedback guaranteeing uniform ultimate boundedness for uncertain dynamic system. *IEEE Transactions on Automatic Control 26*(5), 1139–1144.

Craig, J., Hsu, P., and Sastry, S. (1986). Adaptive control of mechanical manipulators. In *IEEE International Conference on Robotics and Automation*, pp. 190–195. IEEE.

Damiano, D. and Little, J. (1988). *A Course in Linear Algebra*. New York: Academic Press.

Desa, S. and Roth, B. (1985). Synthesis of control systems for manipulators using multivariable robust servo mechanism theory. *International Journal of Robotics Research 4*, 18–34.

Dorling, C. and Zinober, A. (1986). Two approaches to hyperplane design in multivariable variable structure control systems. *International Journal of Control 44*(1), 65–82.

Dubowsky, S. and DesForges, D. (1979). The application of model-referenced adaptive control to robot manipulators. *ASME Journal of Dynamic Systems, Measurement and Control 101*, 193–200.

Fraleigh, J. and Beauregard, R. (1991, August). *Linear Algebra* (2 ed.). Addison-Wesley Publishing Company.

Freund, E. (1982). Fast nonlinear control with arbitrary pole-placement for industry robots and manipulators. *International Journal of Robotics Research 1*, 65–78.

Fu, L. (1992). Robust adaptive decentralised control of robot manipulators. *IEEE Transactions on Automatic Control 37*(1), 106–110.

Furuta, K. (1990). Sliding mode control of a discrete system. *Systems and Control Letters 14*, 145–152.

Glatzl, A., Murphy, S. H., Wen, J. T., and Kopacek, P. (1993). Discrete implementation and adaptation of sliding mode control for robot manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Volume 1, pp. 539–544. IEEE.

Grimm, W. (1990). Robot nonlinearity bounds evaluation techniques for robust control. *International Journal of Adaptive Control and Signal Processing 4*, 501–522.

Grimm, W., Becker, N., and Frank, P. (1988). Robust stability design framework for robot manipulator control. In *IEEE International Conference on Robotics Automation*, pp. 1042–1047. IEEE.

Hanmandlu, M. and Pandian, S. (1993, December). A model-based sliding mode controller for robot manipulators. In *Proceedings of the 32nd Conference on Decision and Control*, pp. 2155–2156. IEEE.

Hashimoto, H., Maruyama, K., and Harashima, F. (1987). A microprocessor-based robot manipulator using sliding mode. *IEEE Trans. Ind. Eng. 34*(1), 11–18.

Hsu, L. and Costa, R. R. (1989). Variable structure reference adaptive control using only input and output measurements. *International Journal of Control 49*(2), 399–416.

Hsu, P., Bodson, M., Sastry, S., and Paden, B. (1987, March). Adaptive identification and control for manipulators without using joint accelerations. In *IEEE Conference on Robotics Automation*, Raleigh, NC, pp. 1210–1215. IEEE.

Ioannou, P. and Tsakalis, K. (1986). A robust direct adaptive controller. *IEEE Transactions on Automatic Control 31*(11), 1033–1043.

Jiang, Y., Park, J., Clements, D., and Hesketh, T. (1994). Adaptive control of robot manipulators in task space. In *Proceedings of the 1994 IEEE SOUTHEASTCON*, pp. 147–151. IEEE.

Kelly, R. and Ortega, R. (1988, March). Adaptive control of robot manipulators : An input-output approach. In *Proceedings of the IEEE Conference on Robotics Automation*, Philadelphia, pp. 699–703. IEEE.

Kim, J.-J., Lee, J.-J., Park, K.-B., and Youn, M.-J. (1993, January). Design of new time-varying sliding surface for robot manipulator using variable structure controller. *Electronics Letters 29*(2), 195–196.

Kosuge, K. and Furuta, K. (1988). Motion control of a robot arm using a variable structure system. In *Proceedings of the U.S.A.-Japan Symposium on Flexible Automation*, pp. 637–643.

Leung, T.-P., Zhou, Q.-J., and Su, C.-Y. (1991, Mar). An adaptive variable structure model following control design for robot manipulators. *IEEE Transactions on Automatic Control 36*(3), 347–352.

Luh, J. (1983). Conventional controller design for industrial robots - a tutorial. *IEEE Transactions on Systems, Man and Cybernetics 13*(3), 298–316.

Man, Z. (1993). *Robust Variable Structure Control for Rigid Robotic Manipulators*. Ph. D. thesis, Department of Electrical and Electronic Engineering, The University of Melbourne, Australia.

Man, Z. and Palaniswami, M. (1992). A robust tracking controller for robotic manipulators. In *1992 IEEE Region 10 International Conference*, Volume 2, pp. 953–7. IEEE.

Man, Z. and Palaniswami, M. (1993). An improved variable structure model-following control for robitic manipulators. *International Journal of Adaptive Control and Signal Processing 7*, 539–562.

Man, Z. and Palaniswami, M. (1994, January). Robust tracking control for rigid robotic manipulators. *IEEE Transactions on Automatic Control 39*(1), 154–159.

Man, Z. and Palaniswami, M. (1995). A robust tracking control scheme for rigid robotic manipulators with uncertain dynamics. *International Journal of Computers and Electrical Engineering 21*(3), 211–220.

Man, Z., Paplinski, A., and Wu, H. R. (1994, December). A robust mimo terminal sliding mode control scheme for rigid robotic manipulators. *IEEE Transactions on Automatic Control 39*(12), 2464–2469.

Middleton, R. and Goodwin, G. (1988). Adaptive computed torque control for rigid link manipulations. *Systems & Control Letters 10*, 9–16.

Mitra, R., Gupta, I., and Moinuddin (1989, November). Performance comparison of variable structure control techniques applied to robot manipulators. In *Fourth IEEE Region 10 International Conference*, pp. 523–526. IEEE.

Miyasato, Y. and Oshima, Y. (1989). Non-linear adaptive control for robotic manipulators with continuous control inputs. *International Journal of Control 49*(2), 545–559.

Morgan, R. G. and Ozguner, U. (1985, March). A decentralized variable structure control algorithm for robotic manipulators. *IEEE Journal of Robotics and Automation 1*(1), 57–65.

Narendra, K. and Boskovic, J. (1990). A combined direct, inderect and variable structure method for robust adaptive control. In *1990 Automatic Control Conference*, pp. 543–548.

Ortega, R. and Spong, M. (1988). Adaptive motion control of rigid robots: A tutorial. In *Proceedings IEEE Conf. Dec. & Control*, pp. 1575–1584. IEEE.

Park, J., Jiang, Y., Hesketh, T., and Clements, D. (1994). Trajectory control of manipulators using adaptive sliding mode control. In *Proceedings of the 1994 IEEE SOUTHEASTCON*, pp. 142–146. IEEE.

Park, K.-B. and Lee, J.-J. (1993). Variable structure controller for robot manipulators using time varying sliding surface. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Volume 1, pp. 89–93. IEEE.

Qu, Z., Dawson, D., and Dorsey, J. (1992). Exponentially stable trajectory following of robotic manipulators under a class of adaptive control. *Automatica 28*(3), 579–586.

Sadegh, N. and Horowitz, R. (1987). Stability analysis of an adaptive controller for robotic manipulators. In *Proceedings of the IEEE International Conference on Robotics Automation*, Raleigh, NC. IEEE.

Schilling, R. J. (1990). *Fundamentals of Robotics*. New Jersey: Prentice Hall.

Shoureshi, R., Momot, M., and Roesler, M. (1990). Robust control for manipulators with uncertainties. *Automatica 26*(2), 353–359.

Singh, S. N. (1986). Ultimate boundedness control of uncertain robotic systems. *International Journal of Systems Science 17*, 859–863.

Slotine, J.-J. (1984). Sliding controller design for non-linear systems. *International Journal of Control 40*(2), 421–434.

Slotine, J.-J. and Li, W. (1987). On the adaptive control of robotic manipulators. *International Journal of Robotics Research 6*(3), 49–59.

Slotine, J.-J. E. and Sastry, S. (1983). Tracking control of nonlinear system using sliding surface with application to robotic manipulators. *International Journal of Control 38*(2), 465–492.

Song, Y. and Gao, W. (1991). Path tracking control of robot manipulator via the vss approach. *International Journal of Systems Science 22*(1), 151–163.

Spong, M. and Vidyasagar, M. (1987). Robust linear compensator design for nonlinear robotic manipulators. *IEEE Journal of Robotics and Automation 3*(4), 345–351.

Spong, M. W. and Ortega, R. (1990, Jan). On adaptive inverse dynamics control of rigid robots. *IEEE Transactions on Automatic Control 35*(1), 92–95.

Stepanenko, Y. and Su, C.-Y. (1992, November). Regressor based sliding mode control of robotic manipulators. In *Proceedings of the 31st Conference on Decision and Control*, Volume 2, pp. 1410–1416. IEEE.

Stonier, R. (n.d.). 84713 Techniques in Control Theory: Lecture Notes.

Su, C.-Y. and Leung, T.-P. (1993, April). A sliding mode controller with bound estimation for robot manipulators. *IEEE Transactions on Automatic Control 9*(2), 208–214.

Su, C.-Y. and Stepanenko, Y. (1993). A new class of adaptive control laws for rigid robots. In *Proceedings of the IECON '93. International Conference on Industrial Electronics, Control and Instrumentation*, Volume 3, pp. 2144–8. IEEE.

Sun, F., Sun, Z., and Gu, W. (1991). The discrete-time controller design based on sliding mode for manipulators. In *International Conference on Control*, Volume 2, pp. 1123–8. IEE.

Utkin, V. (1977, April). Variable structure systems with sliding modes. *IEEE Transactions on Automatic Control 22*(2), 212–220.

Vnekataraman and Gulati (1992). Control of nonlinear systems using terminal sliding modes. In *Proceedings of the 1992 American Control Conference*, Volume 1, pp. 891–3. IEEE.

Wijesoma, S. and Richards, R. (1990). Robust trajectory following of robot using computed torque structure with vss. *International Journal of Control 52*(4), 935–962.

Yao, B., Chan, S., and Wang, D. (1992, December). Robust motion and force control of robot manipulators in the presence of environmental constraint uncertainties. In *PDC31*, Arizona, pp. 1875–1880.

Yao, B., Chan, S., and Wang, D. (1994a). Variable structure adaptive motion and force control of robot manipulators. *Automatica 30*(9), 1473–1477.

Yao, B., Chan, S., and Wang, D. (1994b). Vsc motion and force control of robot manipulators in the presence of environmental constraint uncertainties. *Journal of Robotic Systems 11*(6), 503–515.

Yeung, K. and Chen, Y. (1988). A new controller design for manipulators using the theory of variable structure systems. *IEEE Transactions on Automatic Control 33*(2), 200–206.

Young, K.-K. D. (1978, Feb). Controller design for a manipulator using theory of variable structure systems. *IEEE Transactions on Systems, Man and Cybernetics 8*(2), 101–109.

A Proof of Assumption 3.2

In chapter 4 - A Control Scheme Utilizing Known Dynamics, it was required that the norm of the derivative of the inertia matrix be bounded in the following manner:

$$\|\dot{M}(q)\| < k_2 + k_3\|q\| + k_4\|\dot{q}\| \,, \quad k_2, k_2, k_3 > 0. \tag{A.2}$$

As stated page 44, there is no generic proof for all systems, however it can be proved for any particular rigid robotic manipulator devised.

Below is the proof for the two link rigid robotic manipulator used in the simulations described in section 3.4. A diagram of the manipulator appears on page 56.

The inertia matrix for such a robot is described by (Young 1978)

$$M(q) = \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{12} & \mu_{22}, \end{bmatrix}$$

where

$$\mu_{11}(q_2) = (m_1 + m_2)r_1^2 + m_2 r_2^2 + 2m_2 r_1 r_2 \cos(q_2) + J_1$$

$$\mu_{12}(q_2) = m_2 r_2^2 + m_2 r_1 r_2 \cos(q_2)$$

$$\mu_{22} = m_2 r_2^2 + J_2.$$

Thus $\dot{M}(q)$ can be calculated as

$$\dot{M}(q) = \begin{bmatrix} \dot{\mu}_{11} & \dot{\mu}_{12} \\ \dot{\mu}_{12} & \dot{\mu}_{22}, \end{bmatrix}$$

where

$$\dot{\mu}_{11}(q_2) = -2m_2r_1r_2\sin(q_2)\dot{q}_2$$

$$\dot{\mu}_{12}(q_2) = -m_2r_1r_2\sin(q_2)\dot{q}_2$$

$$\dot{\mu}_{22} = 0.$$

The norm of the matrix $\dot{M}$ is defined as (Spong and Vidyasagar 1987)

$$\|\dot{M}(q)\| = (\lambda_{max}(\dot{M}^T\dot{M}))^{\frac{1}{2}}$$

Using $\alpha = -m_2r_1r_2\sin(q_2)\dot{q}_2$, the eigenvalues of $\dot{M}^T\dot{M}$ can now be calculated as shown on page 3. A full description of how to calculate eigenvalues can be found in Fraleigh and Beauregard (1991).

Firstly calculate $\dot{M}^T\dot{M}$:

$$\begin{aligned}
\dot{M}^T\dot{M} &= \begin{bmatrix} 2\alpha & \alpha \\ \alpha & 0 \end{bmatrix} \begin{bmatrix} 2\alpha & \alpha \\ \alpha & 0 \end{bmatrix} \\
&= \begin{bmatrix} 5\alpha^2 & 2\alpha^2 \\ 2\alpha^2 & \alpha^2 \end{bmatrix}
\end{aligned}$$

Finding the eigenvalues:

$$\begin{aligned}
\det(\dot{M}^T\dot{M} - \lambda \mathrm{I}) &= 0 \\
\therefore \begin{vmatrix} 5\alpha^2 - \lambda & 2\alpha^2 \\ 2\alpha^2 & \alpha^2 - \lambda \end{vmatrix} &= 0 \\
\therefore (5\alpha^2 - \lambda)(\alpha^2 - \lambda) - (2\alpha^2)(2\alpha^2) &= 0 \\
\therefore 5\alpha^4 - 6\alpha^2\lambda + \lambda^2 - 4\alpha^4 &= 0 \\
\therefore \lambda^2 - 6\alpha^2\lambda + \alpha^4 &= 0
\end{aligned}$$

Solving this quadratic expression we get

$$\lambda = (3 \pm 2\sqrt{2})\alpha^2.$$

Therefore

$$\begin{aligned}
\|\dot{M}(q)\| &= (\lambda_{max}(\dot{M}^T\dot{M}))^{\frac{1}{2}} \\
&= ((3+2\sqrt{2})\alpha^2)^{\frac{1}{2}} \\
&= (3+2\sqrt{2})^{\frac{1}{2}}|\alpha| \\
&= (3+2\sqrt{2})^{\frac{1}{2}} m_2 r_1 r_2 \sin(q_2)\dot{q}_2 \quad \text{(subs. } \alpha) \\
&= k\sin(q_2)\dot{q}_2, \quad \text{(where } k = (3+2\sqrt{2})^{\frac{1}{2}} m_2 r_1 r_2 \geqslant 0) \\
&= k\sin(q_2)[0 \;\; 1]\dot{q} \\
&\leqslant k[0 \;\; 1]\dot{q}, \quad \text{(since } \sin\theta \leqslant 1, \; \forall\theta) \\
&\leqslant k\|[0 \;\; 1]\dot{q}\| \\
&\leqslant k\|[0 \;\; 1]\|\|\dot{q}\| \\
&= k\|\dot{q}\| \\
&\leqslant k_2 + k_3\|q\| + k_4\|\dot{q}\|
\end{aligned}$$

B Simulation 1

```
%----------------------------------------------------------------------
% Copyright Nicola Ritter (1995)

% This MATLAB program performs the simulations for Chapter 3 of
% my Masters Thesis.
%----------------------------------------------------------------------

%----------------------------------------------------------------------
% It is worth noting (for those that are possibly confused) that in
% Matlab:

% POINT (1)
%
% [] brackets are used to define matrices
% e.g.  identity = [1 0; 0 1]
% defines a two dimensional identity matrix
% and () brackets are used to access particular elements of the
% matrix
% e.g.  identity(1)(1)
% refers to the value of row 1, column 1 in the matrix identity.

% POINT (2)
%
% `...'  means that an expression is continued on the following
% line.
%----------------------------------------------------------------------

%----------------------------------------------------------------------
% Define some constants
%----------------------------------------------------------------------
TRUE = 1;
FALSE = 0;

%----------------------------------------------------------------------
% Perform initialisation of time series data
%----------------------------------------------------------------------

ts_ref_joint1 = [];
ts_ref_joint2 = [];
ts_joint1 = [];
```

```
ts_joint2 = [];
ts_count = [];
ts_error1 = [];
ts_error2 = [];
ts_control_input1 = [];
ts_control_input2 = [];

%-----------------------------------------------------------------
% Perform the joint initialisations:  mass, radius and moment of
% inertia
%-----------------------------------------------------------------

% Define the links:
mass1 = 0.5;
mass2 = 1.5;

radius1 = 1.0;
radius2 = 0.8;

inertia1 = 5;
inertia2 = 5;

%-----------------------------------------------------------------
% Perform all other initialisations
%-----------------------------------------------------------------

% Set the value for gravity
gravity = 9.81;

% Define the error constant:
p1 = 3;
p2 = 5;
pconstant = p1/p2;

% Set the initial position for the reference model
qr1 = 0.2;
qr2 = 2.0;

% Get the initial position for the robot
stemp1 = 'Enter initial position for';
stemp2 = sprintf('%s joint 1 (reference = %0.1f):  ',stemp1,qr1);
q1 = input (stemp2);
```

```
stemp2 = sprintf('%s joint 2 (reference = %0.1f):  ',stemp1,qr2);
q2 = input (stemp2);

% Define the starting values of the state vector
% i.e.  the joint values and their derivatives.
state = [q1; q2; 0; 0];

% Define the starting values of the reference state
% i.e.  the reference joint values and their derivatives.
ref_state = [qr1; qr2; 0; 0];

% Set the sliding mode matrix C as, C = [C1 C2]
C = [1 0 1 0; 0 1 0 1];
C1 = [1 0; 0 1];
C2 = [1 0; 0 1];

% The reference input is to be 5 for each joint
ref_input = [5;5];

% Describe the parts of Ar
Ar1 = [-4 0; 0 -4];
Ar2 = [-5 0; 0 -5];

% Define parts of Br
B1 = [1 0; 0 1];

% Define the constants used in the adaptive equations for
% the estimates of the parameter bounds k1 to k7

kkvalue = input('Enter initial value for kk1 to kk7:  ');

kk1 = kkvalue;
kk2 = kkvalue;
kk3 = kkvalue;
kk4 = kkvalue;
kk5 = kkvalue;
kk6 = kkvalue;
kk7 = kkvalue;

% Define the starting values for the estimates of the parameter
% bounds k1 to k7
```

```
kvalue = input('Enter initial value for k1 to k7:  ');

k1 = kvalue;
k2 = kvalue;
k3 = kvalue;
k4 = kvalue;
k5 = kvalue;
k6 = kvalue;
k7 = kvalue;

% Ask the user for bounded or unbounded calculations
bounded = 3;
while ((bounded ~= TRUE) & (bounded ~= FALSE))
  stemp1 = sprintf('(%d) Unbounded or (%d) Bounded simulation?:  '...
                    ,FALSE, TRUE);
  bounded = input(stemp1);
end % while

% Set heading for graphs and boundary layer size
if (bounded == FALSE),
  delta = 0;
else
  delta = input('Enter boundary layer size:  ');
end;

% Set the time intervals
interval = 0.01;

%------------------------------------------------------------------
% Initialise variables before the loop starts
%------------------------------------------------------------------

% Initialise switching plane variable to zero
spvar = [0; 0];
norm_spvar = 0;

% Initialise the control input
control_input = [0; 0];

% Initialise the error
error = state - ref_state;
```

```
% Initialise errtilde to be a 1 * 4 vector
errtilde = [0; 0; 0; 0];

% Initialise epstilde_deriv to be a 1 * 2 vector
epstilde_deriv = [0; 0];

%----------------------------------------------------------------
% Loop to produce the values over 800 time intervals
%----------------------------------------------------------------

for count = 1:800,

  %---------------------------------------------------
  % Define time series variables for graphing purposes
  %---------------------------------------------------

  % The time intervals measured in seconds
  ts_count(count) = count * interval;

  % For each joint, define a time series function of the
  % control input
  ts_control_input1(count) = control_input(1);
  ts_control_input2(count) = control_input(2);

  % Define time series functions of the joint positions
  ts_joint1(count) = state(1);
  ts_joint2(count) = state(2);

  % Define time series functions of the reference joint positions
  ts_ref_joint1(count) = ref_state(1);
  ts_ref_joint2(count) = ref_state(2);

  % Set up a time series of the error for each joint
  ts_error1(count) = error(1);
  ts_error2(count) = error(2);

  %---------------------------------------------------
  % Calculate the next positions of the joints
  %---------------------------------------------------

  % Calculate the inertia matrix M(q)
  m11 = (mass1 + mass2) * radius1 * radius1 ...
```

```
        + mass2 * radius2 * radius2 ...
        + 2 * mass2 * radius1 * radius2 * cos(state(2)) ...
        + inertia1;

m12 = mass2 * radius2 * radius2 ...
        + mass2 * radius1 * radius2 * cos(state(2));

m21 = m12;

m22 = mass2 * radius2 * radius2 + inertia2;

Mmatrix = [m11 m12; m21 m22];

% Calculate Coriolis and centrifugal forces vector, f.
temp = mass2 * radius1 * radius2 * sin (state(2));

f1 = temp * (state(3) * state(3) ...
             + 2 * state(3) * state(4));

f2 = temp * state(4) * state(4);

fvector = [f1;f2];

% Calculate gravity vector, g.
g1 = - ( (mass1 + mass2) * radius1 * cos (state(2)) ...
         + mass2 * radius2 * cos (state(1) ...
                                  + state(2))) ...
       * gravity;

g2 = - mass2 * radius2 * cos (state(1) + state(2)) ...
     * gravity;

gvector = [g1; g2];

% Split the state up into the joint values
% and the derivatives
joint = [state(1); state(2)];
joint_deriv = [state(3); state(4)];

% Calculate the new derivatives
joint_deriv = joint_deriv + interval * ((Mmatrix\fvector) ...
                 + (Mmatrix\gvector) + (Mmatrix\control_input));
```

```
norm_joint_deriv = norm (joint_deriv, 'fro');

% Calculate the new joint positions
joint = joint + interval * joint_deriv;
norm_joint = norm (joint, 'fro');

% Set the new state
state(1) = joint(1);
state(2) = joint(2);
state(3) = joint_deriv(1);
state(4) = joint_deriv(2);

%-----------------------------------------------------
% Calculate the next position of the reference state
% using the predetermined reference formulae.
%-----------------------------------------------------
ref_state(1) = 1.25 - (5/3) * exp(-interval * count) ...
                 + (5/12) * exp(-4 * interval * count) ...
                 + (8/30) * exp(-interval * count) ...
                 - (2/30) * exp(-4 * interval * count);

ref_state(2) = 1.25 + (8/3) * exp(-interval * count) ...
                   - (2/3) * exp(-4 * interval * count) ...
                   - (5/3) * exp(-interval * count) ...
                   + (5/12) * exp(-4 * interval * count);

ref_state(3) = (5/3) * exp(-interval * count) ...
                 - (5/3) * exp(-4 * interval * count) ...
                 - (8/30) * exp(-interval * count) ...
                 + (8/30) * exp(-4 * interval * count);

ref_state(4) = - (8/3) * exp(-interval * count) ...
                 + (8/3) * exp(-4 * interval * count) ...
                 + (5/3) * exp(-interval * count) ...
                 - (5/3) * exp(-4 * interval * count);


% Set up a variable for reference joint positions and derivatives
ref_joint = [ref_state(1); ref_state(2)];
ref_joint_deriv = [ref_state(3); ref_state(4)];

%-----------------------------------------------------
```

```
% Calculate the error and the switching manifold
% vector.
%-----------------------------------------------

% Calculate the error
error = state - ref_state;

% Calculate errtilde
errtilde(1) = sign(error(1)) * abs(error(1))^pconstant;
errtilde(2) = sign(error(2)) * abs(error(2))^pconstant;
errtilde(3) = error(3);
errtilde(4) = error(4);

% Calculate the switching manifold vector
spvar = C * errtilde;

% Calculate the norm of the switching manifold vector
norm_spvar = norm (spvar, 'fro');

%-----------------------------------------------
% Calculate the control input for the next iteration.
%-----------------------------------------------

if ((norm_spvar == 0) & (bounded == FALSE)) | ...
  (~(norm_spvar == 0) & (error(1) == 0 | error(2) == 0))
  control_input = [0; 0];
else
  %-----------------------------------------
  % Calculate the new estimates of the
  % parameter bounds k1 to k7
  %-----------------------------------------

  % Calculate the derivative of epstilde
  epstilde_deriv(1) ...
    = pconstant * sign(error(1)) * abs(error(1))^(pconstant-1)...
      * error(3); ...
  epstilde_deriv(2) ...
    = pconstant * sign(error(2)) * abs(error(2))^(pconstant-1)...
      * error(4);

  % Calculate a temporary value used for k1 and for the
  % w factor in the control input formula.
```

```
    temp_matrix = -Ar1 * ref_joint ...
                  - Ar2 * ref_joint_deriv ...
                  + C1 * epstilde_deriv - B1 * ref_input;
    norm_temp_matrix = norm (temp_matrix, 'fro');

    % Calculate the new values of the estimates k1 to k7
    k1 = k1 + interval * kk1 * norm_spvar * norm_temp_matrix;
    k2 = k2 + 0.5 * interval * kk2 * norm_spvar^2;
    k3 = k3 + 0.5 * interval * kk3 * norm_spvar^2 * norm_joint;
    k4 = k4 + 0.5 * interval * kk4 * norm_spvar^2 ...
              * norm_joint_deriv;
    k5 = k5 + interval * kk5 * norm_spvar;
    k6 = k6 + interval * kk6 * norm_spvar * norm_joint;
    k7 = k7 + interval * kk7 * norm_spvar * norm_joint_deriv^2;

    %-----------------------------------------
    % Calculate the w factor used in the
    % calculation of the control input required
    % for the next iteration of the loop
    %-----------------------------------------

    wfactor = k1 * norm_temp_matrix ...
            + 0.5 * norm_spvar ...
                  * (k2 + k3 * norm_joint ...
                      + k4 * norm_joint_deriv) ...
            + k5 + k6 * norm_joint ...
            + k7 * norm_joint_deriv * norm_joint_deriv;

    %-----------------------------------------
    % Calculate the new control input
    %-----------------------------------------
    if ((bounded == FALSE) | ...
        ((bounded == TRUE) & (norm_spvar >= delta))),
      control_input = - (spvar / norm_spvar) * wfactor;
    else
      control_input = - (spvar / delta) * wfactor;
    end %if
  end %if

  %---------------------------------------------------
  % Output count, so that 'time' can be seen to pass
  %---------------------------------------------------
  count
end % for loop

%-----------------------------------------------------------------
% Plot the results stored in the time series variables,
% and save to file
%-----------------------------------------------------------------

figure(1);
plot (ts_count, ts_ref_joint1, 'c-', ts_count, ts_joint1, 'y:');
ylabel('The output tracking of joint 1 (rad)');
xlabel('Time t (s)');
text(3,0.8,'_ qr1');
text(3,0.7,'.....  q1');
print figure1;

figure(2);
plot (ts_count, ts_ref_joint2, 'c-', ts_count, ts_joint2, 'y:');
ylabel('The output tracking of joint 2 (rad)');
xlabel('Time t (s)');
text(3,1.8,'_ qr2');
text(3,1.7,'.....  q2');
print figure2;

figure(3);
plot (ts_count, ts_error1);
axis([0 8 -0.2 0.2]);
ylabel('The output tracking error of joint 1 (rad)');
xlabel('Time t (s)');
print figure3;

figure(4);
plot (ts_count, ts_error2);
axis([0 8 -0.2 0.2]);
ylabel('The output tracking error of joint 2 (rad)');
xlabel('Time t (s)');
print figure4;

figure(5);
plot (ts_count, ts_control_input1);
axis([0 8 -50 50]);
ylabel('The control input of joint 1 (Nm)');
```

```
xlabel('Time t (s)');
print figure5;

figure(6);
plot (ts_count, ts_control_input2);
axis([0 8 -50 50]);
ylabel('The control input of joint 2 (Nm)');
xlabel('Time t (s)');
print figure6;

%-------------------------------------------------------------------
% End of program
%-------------------------------------------------------------------
```

C Simulation 2

```
%-----------------------------------------------------------------
% Copyright Nicola Ritter (1995)

% This program forms the simulation of chapter 4 of my Masters
% thesis.
%-----------------------------------------------------------------

%-------------------------------------------------------------------
% It is worth noting (for those that are possibly confused) that in
% Matlab:

% POINT (1)
%
% [] brackets are used to define matrices
% e.g.  identity = [1 0; 0 1]
% defines a two dimensional identity matrix
% and () brackets are used to access particular elements of the
% matrix
% e.g.  identity(1)(1)
% refers to the value of row 1, column 1 in the matrix identity.

% POINT (2)
%
% `...'  means that an expression is continued on the following
% line.
%-------------------------------------------------------------------

%-------------------------------------------------------------------
% Define some constants
%-------------------------------------------------------------------

TRUE = 1;
FALSE = 0;

%-----------------------------------------------------------------
% Perform initialisation of time series data
%-----------------------------------------------------------------

ts_ref_joint1 = [];
ts_ref_joint2 = [];
```

```
ts_joint1 = [];
ts_joint2 = [];
ts_count = [];
ts_error1 = [];
ts_error2 = [];
ts_control_input1 = [];
ts_control_input2 = [];

%-----------------------------------------------------------------
% Perform the joint initialisations:  mass, radius and moment of
% inertia
%-----------------------------------------------------------------

% Define the links:
mass1 = 0.5;
mass2 = 1.5;

radius1 = 1.0;
radius2 = 0.8;

inertia1 = 5;
inertia2 = 5;

% Define the nominal (estimated) value for the masses
n_mass1 = 0.4;
n_mass2 = 1.2;

%-----------------------------------------------------------------
% Perform all other initialisations
%-----------------------------------------------------------------

% Set the value for gravity
gravity = 9.81;

% Define the 'p' error constant:
p1 = 3;
p2 = 5;
pconstant = p1/p2;

% Set the initial position for the reference model
qr1 = 0.2;
qr2 = 2.0;
```

```
% Get the initial position for the robot
stemp1 = 'Enter initial position for';
stemp2 = sprintf('%s joint 1 (reference = %0.1f):  ',stemp1,qr1);
q1 = input (stemp2);
stemp2 = sprintf('%s joint 2 (reference = %0.1f):  ',stemp1,qr2);
q2 = input (stemp2);

% Define the starting values of the state vector
% i.e.  the joint values and their derivatives.
state = [q1; q2; 0; 0];

% Define the starting values of the reference state
% i.e.  the reference joint values and their derivatives.
ref_state = [qr1; qr2; 0; 0];

% The sliding mode uses C as defined below
% and C = [C1 C2]
C = [1 0 1 0; 0 1 0 1];
C1 = [1 0; 0 1];
C2 = [1 0; 0 1];

% The reference input is to be 5 for each joint
ref_input = [5;5];

% Describe the feedback matrix
K = [-4 0 -5 0; 0 -4 0 -5];

% Describe the constants for the linearized nominal system
% Ae = A + B * K;
A = [0 0 1 0; 0 0 0 1; 0 0 0 0; 0 0 0 0];
B = [0 0; 0 0; 1 0; 0 1];

% Define the constants used in the adaptive equations for
% the estimates of the parameter bounds b0 to b2
bk0 = 10;
bk1 = 10;
bk2 = 10;

% Define the starting values for the estimates of the parameter
% bounds b0 to b2
```

```
bvalue = input('Enter initial value for b0 to b2:  ');

b0 = bvalue;
b1 = bvalue;
b2 = bvalue;

% Ask the user for bounded or unbounded calculations
bounded = 3;
while ((bounded ~= TRUE) & (bounded ~= FALSE))
  stemp1 = sprintf('(%d) Unbounded or (%d) Bounded simulation?:  '...
                   ,FALSE, TRUE);
  bounded = input(stemp1);
end % while

% Set boundary layer size
if (bounded == FALSE),
  delta = 0;
else
  delta = input('Enter boundary layer size:  ');
end;

% Set the time intervals
interval = 0.01;

%-------------------------------------------------------------------
% Initialise variables before the loop starts
%-------------------------------------------------------------------

% Initialise the switching manifold vector and its norm to zero
spvar = [0; 0];
norm_spvar = 0;

% Initialise control inputs to zero
nominal_input = [0; 0];
compensator = [0; 0];
control_input = [0; 0];

% Initialise error
error = state - ref_state;

% Initialise ref_joint_2deriv to be a 1 * 2 vector
ref_joint_2deriv = [0; 0];
```

```
% Initialise epstilde_deriv to be a 1 * 2 vector
epstilde_deriv = [0; 0];

%----------------------------------------------------------------
% Loop to produce the values over time intervals
%----------------------------------------------------------------

for count = 1:800,

  %--------------------------------------------------
  % Define time series variables for graphing purposes
  %--------------------------------------------------

  % Set up a time series variable for graphing purposes,
  % measured in seconds
  ts_count(count) = count * interval;

  % For each joint, define a time series function of
  % the control input
  ts_control_input1(count) = control_input(1);
  ts_control_input2(count) = control_input(2);

  % Define time series functions of the joint positions
  ts_joint1(count) = state(1);
  ts_joint2(count) = state(2);

  % Define time series functions for the reference joint
  % positions
  ts_ref_joint1(count) = ref_state(1);
  ts_ref_joint2(count) = ref_state(2);

  % Set up a time series of the error for each joint
  ts_error1(count) = error(1);
  ts_error2(count) = error(2);

  %--------------------------------------------------
  % Calculate the current actual position of the robot
  % joints.
  %--------------------------------------------------

  % Split the state up into the joint values and the derivatives
% Calculate the inertia matrix M(q)
m11 = (mass1 + mass2) * radius1 * radius1 ...
       + mass2 * radius2 * radius2 ...
       + 2 * mass2 * radius1 * radius2 * cos(state(2)) ...
       + inertia1;

m12 = mass2 * radius2 * radius2 ...
       + mass2 * radius1 * radius2 * cos(state(2));

m21 = m12;

m22 = mass2 * radius2 * radius2 + inertia2;

Mmatrix = [m11 m12; m21 m22];

% Calculate Coriolis and centrifugal forces vector, f.
temp = mass2 * radius1 * radius2 * sin (state(2));

f1 = temp * (state(3) * state(3) ...
              + 2 * state(3) * state(4));

f2 = temp * state(4) * state(4);

fvector = [f1;f2];

% Calculate gravity vector, g.
g1 = - ( (mass1 + mass2) * radius1 * cos (state(2)) ...
         + mass2 * radius2 * cos (state(1) ...
                                  + state(2)) ...
       * gravity;

g2 = - mass2 * radius2 * cos (state(1) + state(2)) ...
     * gravity;

gvector = [g1; g2];

% Split the state up into the joint values
% and the derivatives
joint = [state(1); state(2)];
joint_deriv = [state(3); state(4)];

% Calculate the new derivatives
```

```
joint_deriv = joint_deriv + interval * ((Mmatrix\fvector) + ...
                (Mmatrix\gvector) + (Mmatrix\control_input));
norm_joint_deriv = norm (joint_deriv, 'fro');

% Calculate the new joint positions
joint = joint + interval * joint_deriv;
norm_joint = norm (joint, 'fro');

% Set the new state
state(1) = joint(1);
state(2) = joint(2);
state(3) = joint_deriv(1);
state(4) = joint_deriv(2);

%-----------------------------------------------------
% Calculate the new reference positions for the joints
% and the reference position derivatives.
%-----------------------------------------------------

ref_state(1) = 1.25 - (5/3) * exp(-interval * count) ...
                       + (5/12) * exp(-4 * interval * count) ...
                       + (0.8/3) * exp(-interval * count) ...
                       - (0.2/3) * exp(-4 * interval * count);

ref_state(2) = 1.25 + (8/3) * exp(-interval * count) ...
                       - (2/3) * exp(-4 * interval * count) ...
                       - (5/3) * exp(-interval * count) ...
                       + (5/12) * exp(-4 * interval * count);

ref_state(3) = (5/3) * exp(-interval * count) ...
                 - (5/3) * exp(-4 * interval * count) ...
                 - (0.8/3) * exp(-interval * count) ...
                 + (0.8/3) * exp(-4 * interval * count);

ref_state(4) = - (8/3) * exp(-interval * count) ...
                 + (8/3) * exp(-4 * interval * count) ...
                 + (5/3) * exp(-interval * count) ...
                 - (5/3) * exp(-4 * interval * count);

ref_joint_2deriv(1) ...
               = - (5/3) * exp(-interval * count) ...
                 + (20/3) * exp(-4 * interval * count) ...
```

```
                  + (0.8/3) * exp(-interval * count) ...
                  - (3.2/3) * exp(-4 * interval * count);

ref_joint_2deriv(2) ...
                = + (8/3) * exp(-interval * count) ...
                  - (32/3) * exp(-4 * interval * count) ...
                  - (5/3) * exp(-interval * count) ...
                  + (20/3) * exp(-4 * interval * count);

% Set up a variable for reference joint positions and derivatives
ref_joint = [ref_state(1); ref_state(2)];
ref_joint_deriv = [ref_state(3); ref_state(4)];

%-----------------------------------------------------
% Calculate the error values
%-----------------------------------------------------

% Calculate exact error
error = state - ref_state;

% Split the exact error into joint_error
% and derivatives of joint_error
joint_error = [error(1); error(2)];
joint_error_deriv = [error(3); error(4)];

% Calculate the norm of joint_error
norm_joint_error = norm (joint_error, 'fro');

% Calculate errtilde
errtilde = [sign(error(1)) * abs(error(1))^pconstant; ...
            sign(error(2)) * abs(error(2))^pconstant; ...
            error(3); ...
            error(4)];

% Calculate the derivative of epstilde
epstilde_deriv(1) ...
  = pconstant * sign(error(1)) * abs(error(1))^(pconstant-1)...
    * error(3); ...
epstilde_deriv(2) ...
  = pconstant * sign(error(2)) * abs(error(2))^(pconstant-1)...
    * error(4);
```

```
%-----------------------------------------------------
% Calculate the switching manifold vector
%-----------------------------------------------------

% Calculate the switching manifold vector
spvar = C * errtilde;

% Calculate the norm of the switching manifold vector
norm_spvar = norm (spvar, 'fro');

%-----------------------------------------------------
% Calculate the nominal part of the control input
%-----------------------------------------------------

% Calculate the nominal inertia matrix
n_m11 = (n_mass1 + n_mass2) * radius1 * radius1 ...
         + n_mass2 * radius2 * radius2 ...
         + 2 * n_mass2 * radius1 * radius2 * cos(state(2)) ...
         + inertia1;

n_m12 = n_mass2 * radius2 * radius2 ...
         + n_mass2 * radius1 * radius2 * cos(state(2));

n_m21 = n_m12;

n_m22 = n_mass2 * radius2 * radius2 + inertia2;

n_Mmatrix = [n_m11 n_m12; n_m21 n_m22];

% Calculate Coriolis and centrifugal forces nominal matrix
temp = n_mass2 * radius1 * radius2 * sin (state(2));

n_f1 = temp * (state(3) * state(3) + 2 * state(3) * state(4));

n_f2 = - temp * state(4) * state(4);

n_fvector = [n_f1; n_f2];

% Calculate nominal gravity matrix
n_g1 = - ((n_mass1 + n_mass2) * radius1 * cos (state(2)) + ...
         n_mass2 * radius2 * cos (state(1) + state(2))) ...
         * gravity;
```

```
n_g2 = - n_mass2 * radius2 * cos (state(1) + state(2)) * gravity;

n_gvector = [n_g1; n_g2];

% Calculate combined Coriolis, centrifugal and gravity matrix
n_hvector = n_fvector + n_gvector;

% Calculate the nominal control input
nominal_input = n_Mmatrix * K * error ...
                   - n_hvector + n_Mmatrix * ref_joint_2deriv;

%-----------------------------------------------------
% Calculate the compensator
%-----------------------------------------------------

% Calculate an interim value and its norm
interim = spvar' * C * B / n_Mmatrix;
norm_interim = norm(interim, 'fro');

if ((norm_interim == 0) & (bounded == FALSE)) ...
  | (~(norm_interim == 0) & (error(1) == 0 | error(2) == 0)),
  compensator = [0; 0];
else

  % Calculate a temporary value
  b_temp = C * B / n_Mmatrix;
  norm_b_temp = norm (b_temp, 'fro');
  b_value = norm_spvar * norm_b_temp;

  % Calculate the new estimates for the parameter bounds
  b0 = b0 + interval * bk0 * b_value;
  b1 = b1 + interval * bk1 * b_value * norm_joint;
  b2 = b2 + interval * bk2 * b_value * norm_joint_deriv^2;

  % Calculate terms of the w factor used to calculate the
  % compensator

  fterm1 = -spvar' * C * (A+B*K) * error;

  fterm2 = -b_value ...
          * (b0 + b1 * norm_joint + b2 * norm_joint_deriv^2);
```

```
      fterm3 = -spvar' * C1 * epstilde_deriv;

      fterm4 = spvar' * C1 * joint_error_deriv;

      wfactor = fterm1 + fterm2 + fterm3 + fterm4;

      if ((bounded == FALSE) | ...
          ((bounded == TRUE) & (norm_interim >= delta))),
        compensator = interim' / norm_interim^2 * wfactor;
      else
        compensator = interim' / delta^2 * wfactor;
      end; %if

    end %if

    %--------------------------------------------------
    % Calculate the control input from its two parts
    %--------------------------------------------------

    control_input = nominal_input + compensator;

    %--------------------------------------------------
    % Output count, so that 'time' can be seen to pass
    %--------------------------------------------------

    count

  end % for loop

  %------------------------------------------------------------
  % Plot the results stored in the time series variables,
  % and save to file
  %------------------------------------------------------------

  figure(1);
  plot (ts_count, ts_ref_joint1, 'c-', ts_count, ts_joint1, 'y:');
  ylabel('The output tracking of joint 1 (rad)');
  xlabel('Time t (s)');
  text(3,0.8,'_ qr1');
  text(3,0.7,'.....  q1');
  print figure1;
```

```
figure(2);
plot (ts_count, ts_ref_joint2, 'c-', ts_count, ts_joint2, 'y:');
ylabel('The output tracking of joint 2 (rad)');
xlabel('Time t (s)');
text(3,1.8,'_ qr2');
text(3,1.7,'.....  q2');
print figure2;

figure(3);
plot (ts_count, ts_error1);
axis([0 8 -0.2 0.2]);
ylabel('The output tracking error of joint 1 (rad)');
xlabel('Time t (s)');
print figure3;

figure(4);
plot (ts_count, ts_error2);
axis([0 8 -0.2 0.2]);
ylabel('The output tracking error of joint 2 (rad)');
xlabel('Time t (s)');
print figure4;

figure(5);
plot (ts_count, ts_control_input1);
axis([0 8 -50 50]);
ylabel('The control input of joint 1 (Nm)');
xlabel('Time t (s)');
print figure5;

figure(6);
plot (ts_count, ts_control_input2);
axis([0 8 -50 50]);
ylabel('The control input of joint 2 (Nm)');
xlabel('Time t (s)');
print figure6;

%------------------------------------------------------------------
% End of program
%------------------------------------------------------------------
```