Self-tuning Position and Force Control of a Hydraulic Manipulator

Andrew C. Clegg

Thesis submitted
for the
Degree of Doctor of Philosophy

Heriot-Watt University

Department of Computing and Electrical Engineering

November 2000

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that the copyright rests with its author and that no quotation from the thesis and no information derived from it may be published without the prior written consent of the author or the University (as may be appropriate).

Table of Contents

List of Figures

List of Tables

Principal Abbreviations

ADC	Analogue to Digital Convertor
ARMAX	AutoRegressive Moving Average Exogenous
ATI	Assurance Technologies Inc.
BUD-RLS	Bierman U-D Factorisation Recursive Least Squares
DAC	Digital to Analogue Convertor
DOF	Degrees Of Freedom
DSP	Digital Signal Processor
EASY-RLS	Simplified Matrix Inversion Lemma Recursive Least Squares
flops	floating point operations
GPP	Generalised Pole Placement
GUI	Graphical User Interface
LIRMM	Laboratoire d'Informatique, de Robotique et de Microélectric de Montpellier
LQG	Linear Quadratic Gaussian
LSI	Loughborough Sound Images
MIL-RLS	Matrix Inversion Lemma Recursive Least Squares
MIMO	Multi-Input Multi-Output
MRAC	Model Reference Adaptive Controller
MV	Minimum Variance
PP	Pole Placement
RLS	Recursive Least Squares
RMS	Root Mean Square
ROV	Remotely Operated Vehicle
SEL	Slingsby Engineering Limited
SISO	Single-Input Single-Output
VSC	Variable Structure Control
VSC-HF	Variable Structure Control - High Frequency

Acknowledgements

Firstly, I would like to thank my supervisors Dr. Matt Dunnigan and Prof. Dave Lane for their support and enthusiasm throughout the years. I would particularly like to thank Dave for establishing (and continuing long past my departure) an excellent research environment in the Ocean Systems Laboratory, which provided the motivation and emphasis for my work. From Matt I gained an excellent understanding of control engineering, a subject that I now feel at least competent in. I am indebted to both of them for their many suggestions and corrections to this thesis.

I would also like to thank all of those people I worked with during the course of this research, in particular Andrew Quinn, Phil Knightbridge, Alistair Houstin, and the staff of the Mechanical Workshop who helped keep the robots going. This work was undertaken under several research projects, namely TUUV (Technology for Unmanned Underwater Vehicles) and UNION (Underwater Intelligent Operation and Navigation). I must extend my gratitude to those companies that sponsored this research work. I hope you found it worthwhile.

Thanks go to all of my family, friends and colleagues (both past and present). Finally, I will be forever grateful to Lorna without whose love and support this work would still only be "nearly there". Thank you.

Abstract

Robotic systems for use in hazardous and unstructured environments are primarily teleoperated. This imposes a high workload on the remote human operator, severely limiting the efficiency of these systems. An important goal in robotics research is to allow the operator to interact with the robot at a much higher level than at present, thereby increasing the system's effectiveness. One prerequisite for this is the accurate and automatic control of the robot.

This thesis presents a dynamic model of a hydraulically actuated manipulator, typical of the robots used in the subsea domain. The model provides an insight into how the manipulator behaves and its associated nonlinearities. It is also used for simulation purposes and is validated experimentally.

Adaptive control of the manipulator is proposed as these strategies can automatically accommodate wide changes in operating conditions, such as payload and manipulator configuration. The adaptive scheme used is a self-tuning pole placement controller, and is initially applied to the independent control of the experimental manipulator's joint angles. This demonstrates the feasibility of such a controller, and its associated benefits over conventional fixed gain controllers.

To realise complex constrained motion tasks, a hybrid position/force controller is then considered. Here the end-effector positions and forces are controlled simultaneously in orthogonal directions in the Cartesian workspace. A fixed gain hybrid position/force controller is developed to demonstrate the capabilities that such a scheme provides.

A multivariable self-tuning pole placement controller is then applied to the hybrid position/force control problem. Results are presented showing the ability of the controller to cope with varying operating conditions, and its consequent benefits over an equivalent fixed gain controller.

Chapter 1

Introduction

1.1 Introduction

Robot evolution is driven by two distinct requirements. The first, and what is often considered the primary use for robots, is increasing productivity, whether on a automotive production line or for automated fruit picking. The second rationale for using robots is safety. Robots can perform tasks that would otherwise be too hazardous for humans, for example in the space, nuclear or subsea domains. It is this latter reason that has provided the motivation for the work in this thesis.

Generally, robots used for increasing productivity exhibit little or no intelligence. They are taught exactly how to perform a task and repeat that sequence of commands a fixed number of times. Thus, in manufacturing, robotic workcells are designed to be structured to eliminate the occurrence of unexpected events which the robot cannot cope with. Conversely, robots that operate in hazardous environments cannot be pre-programmed as the workspace is often unstructured and subject to changes. Consequently, the robot must be able to react safely to any unpredictable incidents.

To meet the requirements for operation in hazardous environments, robots are currently *teleoperated* where a human operator controls every aspect of the robot from a remote location [1.1, 1.2]. Cameras mounted at the remote worksite allow the operator to see what is in the vicinity of the robot, enabling the required task to be carried out. Movements of the robot are achieved using a *master-slave* arrangement, where the remote

slave robot follows any motions that the operator makes with a suitable *master* input device.

There are many problems associated with teleoperation, the main ones being :-

- The two-dimensional image from the camera and lack of depth cues in the scene impedes the operator in visualising the three-dimensional workspace [1.1].
- There are instances where the visibility may be reduced, either by object occlusion or turbidity, in which case the operator effectively works blindfolded.
- The video image provides no tactile information to the operator, making tasks that involve contact difficult to perform.
- In certain situations there may be delays between the robot and operator sites, for example when teleoperating space robots. These delays can lead to severe control problems, since the operator is acting on the basis of information that may be a few seconds old [1.2].

These problems result in a high workload for the operator and severely limit the overall effectiveness of these systems. However, the reason that this type of system is employed so widely is that there is no viable alternative means of coordinating and controlling the robot in these unstructured and hazardous environments.

1.2 Subsea Teleoperated Robots

Divers have traditionally performed underwater tasks, at great risk to themselves and financial cost to the offshore oil and gas industries. Additionally, the depths to which divers can operate is strictly limited, restricting the potential for subsea exploitation [1.3]. Consequently, there is much interest in subsea robotic systems to reduce and even eliminate the need for divers for subsea intervention work.

A typical underwater robot comprises one or more manipulators mounted on an unmanned Remotely Operated Vehicle (ROV) which is teleoperated from on board a surface ship. The two sites are connected by a tether, used to pass power, telemetry and video images. The manipulators used are almost exclusively hydraulically actuated (as opposed to electrically powered robots that dominate production lines) due to their mechanical robustness and large power to weight ratio.

The problems associated with teleoperation, highlighted above, are especially relevant to these subsea robotic systems, as the operator has to teleoperate *both* the ROV and manipulator. This is compounded by the poor quality of underwater images, almost total lack of depth cues and frequent loss of visibility [1.4]. Consequently, the operators of these systems are only capable of performing relatively simple tasks and can only work for short lengths of time before becoming mentally fatigued.

To provide the operator with more workspace feedback, force reflecting systems are available where the forces being exerted on the remote slave robot can be 'felt' by the operator through an active master mechanism. However, these systems have only limited effectiveness as they actually place additional burden on the operator who is now part of the control loop [1.5]. Furthermore, the forces cannot be controlled to a specified level, relying on the operator to adjust the pressure that he, or she, is feeling.

Therefore, removing the operator from the loop is a major goal of subsea robotics research.

1.3 Robot Teleassistance

At present, full autonomous operation of robots remains a distant goal which requires input from all aspects of robot technology, including control, artificial intelligence and workspace sensing. An intermediate goal is that of achieving *teleassistance* [1.4, 1.6], where the operator supervises the robot and computers automatically realise low level tasks

such as obstacle avoidance and trajectory following. Therefore, the operator interacts at a higher level than with teleoperation, alleviating the problems highlighted above and improving the efficiency and capabilities of these robots.

Various functional components are required to realise teleassistance [1.7], and these are shown schematically in Figure 1.1. The *task planning* function provides the operator with an interface to direct and supervise the actions of the robot. The level of sophistication embedded in the task planner will govern how reliant the robotic system is upon the operator. For instance, implementing primitive actions, such as 'move', 'grasp' and 'align', would require little intelligence, however, automatically sequencing these actions to perform a complete task autonomously is rather more difficult [1.8].

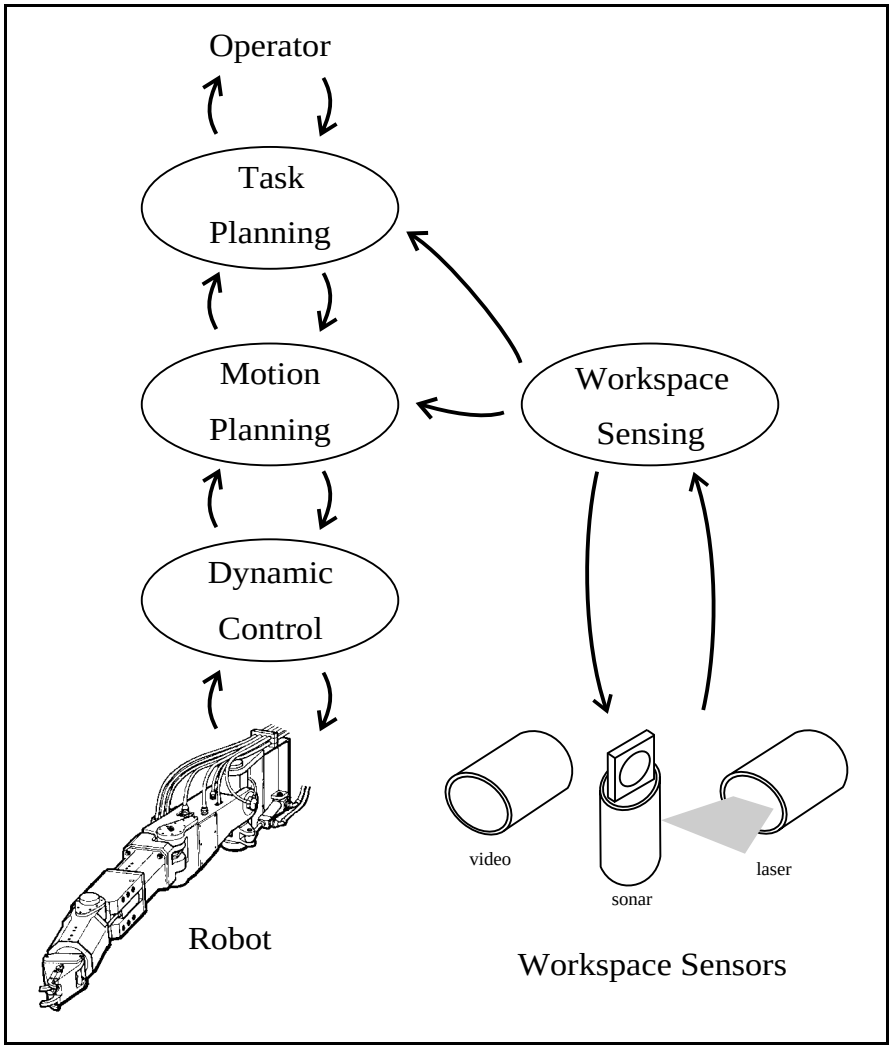

Figure 1.1 Teleassistance Functional Architecture

Motion planning, workspace sensing and dynamic control are fundamental to teleassistance, since they remove the burden of control from the operator. The *motion planning* function determines the robot motions required to meet the commanded actions of the task planner, avoiding problematic robot configurations and obstacles [1.9]. Knowledge about target objects and obstacles in the vicinity of the robot is acquired by the *workspace sensing* sub-system [1.10].

The *dynamic control* function realises the required robot motions to achieve the specified actions, and so interfaces directly with the robot. The controlled variables can be positions, velocities or interaction forces, the latter being used to realise tasks such as grinding and assembly operations.

This thesis investigates the dynamic control of a typical subsea robot in the context of teleassistance, as defined by Figure 1.1. The task planning, motion planning and workspace sensing problems are not addressed here. It should also be noted that this work is concerned with the operation of a single manipulator[†], rather than the ROV/manipulator system as a whole, and hence assumes that the manipulator is on a stationary base. This assumption may seem restrictive, but parallel research is addressing the problems of ROV stabilisation [1.12] and coupled control of the manipulator and ROV [1.13].

Though the work presented in this thesis is focused on this specific application area, the conclusions drawn could be applied to many other robotic systems.

1.4 Robot Control

When manipulators are employed offshore, tasks are performed under unknown and changing conditions, for example variations in payload. Currently, remote manipulators are

[†] The term *robot* conventionally refers to any re-programmable, multi-functional mechatronic device [1.11], though any such generalised definition is open to much misinterpretation. The term *manipulator* is more specific in that it refers to a robot that comprises of a set of links connected together by powered joints. Throughout the remainder of this thesis these two terms are used synonymously, pertaining to a serial link (i.e. connected in a chain) manipulator.

equipped with relatively simple controllers and are incapable of coping with these changes. However, this is adequate for teleoperated systems because the operator can manually correct for any errors arising from the poor control by looking at the video image and adjusting the master arm accordingly.

It is therefore apparent that if teleassistance is to be realised and the operator removed from the loop, then accurate and automatic control of the robot is needed. Furthermore, the controllers used should be able to maintain accuracy in the presence of the uncertain and changing conditions associated with subsea tasks. It is these fundamental requirements that have motivated the work in this thesis.

Such controllers could also be used to automatically realise more complex tasks. For instance, control in Cartesian space is feasible, allowing trajectories defined in the three-dimensional workspace to be followed automatically. This is problematic to achieve using teleoperation, since operators find it difficult to move the spatially correspondent master arm along a predefined spatial trajectory with any degree of accuracy.

Another beneficial type of control is the control of the forces and torques being exerted by the manipulator on its environment. This can facilitate tasks that would otherwise be problematic to complete without some form of tactile feedback. For example, when mating a connector and socket under teleoperation, it is difficult to align the two parts with sufficient positional accuracy and the tendency is to for them to jam. This type of task can be readily achieved using force control, which can automatically align the manipulator holding the connector as it is inserted, increasing the probability of successful completion [1.14]. This task could also be achieved using a force reflecting teleoperation system, but as described earlier these schemes are far from ideal.

Regarding the uncertain and changing conditions, the simple fixed gain controllers currently used can only be tuned to work well under one particular set of operating conditions and degradation occurs once these change. So, to provide an adequate stability

margin over a wide range of conditions, these fixed gain controllers are tuned for the worst circumstances likely to be encountered. Consequently, sub-optimal operation is manifest for the majority of situations encountered during a typical task.

Therefore, it follows that a control scheme which can accommodate the unknown and changing conditions would be beneficial to subsea manipulation systems. This can be achieved by employing an advanced controller such as a robust controller which is insensitive to plant variations, or an adaptive controller which can automatically take such changes into account [1.15].

Therefore, the dynamic control of a subsea robot is required to provide :-

- Accurate control under unknown and changing conditions, enabling the operator to be removed from the loop and thereby improving the efficiency of such systems.
- Enhanced capabilities, enabling the robot to automatically perform complex tasks such as simultaneous control of Cartesian positions and forces, allowing automatic trajectory following and part mating.

This is just one, albeit important, area of research that is required to place more intelligence at the remote manipulator, leading to semi-autonomous and perhaps even fully autonomous robots. For subsea robotics, this is a significant step towards cutting the tether between the ROV and the surface.

A final motivation for this work is that many advanced control schemes proposed by the robotics research community, have only been applied to simulations or specialised laboratory robots. These often bear little resemblance to real industrial robots, and particularly hydraulically powered manipulators, to which these controllers must eventually be applied.

1.5 Thesis Organisation

This thesis opens with an introduction to the field of robot control, then the manipulator used is described and the control schemes proposed are subsequently developed. Both simulation and practical results are presented and compared to results obtained from conventional fixed gain controllers. A more detailed description of the individual chapters follows.

Chapter Two presents a broad overview of robot control research, discussing the various approaches that have been proposed in the context of this application. The concepts of *independent joint control* and *multivariable Cartesian control* of robots are introduced. The requirements for suitable controllers are highlighted and self-tuning pole placement schemes are proposed.

A mathematical model of the manipulator used in this work is derived in Chapter Three. The model covers both kinematic and dynamic aspects of the manipulator, as well as detailed mathematical analysis of the actuators used on this particular robot. This model provides an insight into the operation of the hydraulic manipulator and is used during the simulation phase which precedes the practical implementation of the controllers.

The theory of self-tuning controllers is discussed in Chapter Four. Initially, single-input single-output (SISO) controllers are derived and discussed in the framework of independent joint control. The theory is then extended to multi-input multi-output (MIMO) systems, which is applicable to the multivariable control of Cartesian positions and forces.

Chapter Five presents the results of the SISO self-tuning controller applied to the experimental manipulator, and comparisons are made with the results from a fixed gain controller. Real-time implementation and operational issues are discussed.

A hybrid position/force control scheme is then developed to perform simultaneous Cartesian position and force control. Results from the application of a fixed gain hybrid position/force controller to the experimental manipulator are presented in Chapter Six. The

limitations of this scheme under typical operating conditions are demonstrated.

Chapter Seven presents the results of a MIMO self-tuning hybrid position/force controller. The benefits that it provides over an equivalent fixed gain scheme are illustrated. Furthermore, a brief comparison between this and another form of advanced controller, specifically a robust variable structure controller, is also given.

The final chapter of this thesis, Chapter Eight, summarises the work presented and draws relevant conclusions. The author's contributions to the field of robot control are discussed, together with suggestions for areas of future work.

Chapter 2

Manipulator Control Strategies

2.1 Introduction

The introductory chapter highlighted the need for accurate manipulator control to realise both teleassisted and fully autonomous systems. Currently, offshore manipulators are teleoperated and rely on the human operator to monitor and correct for inaccuracies in the control. Advanced control techniques will improve the performance of such systems, allowing the operator to be removed from the control loop and placed in a more supervisory role, thereby increasing the productivity of the system as a whole. Furthermore, advanced control strategies can automatically realise more complex tasks, such as trajectory following and the mating of parts, that are difficult to achieve under teleoperation. This would enhance the range of tasks that the manipulator could achieve.

Any control system must accommodate the wide variations in manipulator dynamics, which arise due to changes in payload, acceleration and configuration. This requirement is particularly important for manipulators operating in unstructured environments. These systems have unknown and changing operating conditions, implying that the manipulator's motions and appropriate control actions cannot be determined a priori. The control of such systems is of great interest and it is the application of advanced control schemes to improve the absolute accuracy of such manipulators that is the motivation behind this thesis.

For manipulators that operate in well defined environments, such as manufacturing

workcells, improved controller performance is not so crucial. This is because the variations mentioned above, though still present, are consistent and are easily accommodated by the sequence of predetermined commands that they execute. Consequently such robots are more concerned with repeatability, rather than absolute accuracy. Nevertheless, advanced control schemes can provide additional functionality and some examples of this are also discussed within this thesis.

This chapter starts by describing the problems associated with manipulator control. The different control schemes that have found use are then reviewed, first from the perspective of the required action of the controller, and secondly looking at the various control techniques available. The self-tuning controller developed in this thesis is placed in context of this review, and previously proposed self-tuning manipulator controllers are described in detail.

2.2 The Manipulator Control Problem

The requirement for manipulator control primarily stems from the fact that motion of the manipulator is provided by actuators at each joint that generate forces or torques. If an actuator can directly execute a desired trajectory, as in the case of a stepper motor, open loop control will suffice. These systems are rarely used due to their limited physical capabilities, and so some form of control algorithm is invariably needed.

Manipulator control has been the subject of many years research, and continues to attract much attention. The reason for this is the many difficult challenges posed by these systems, for instance :-

- the highly nonlinear dynamics of both manipulator and actuator, including inertia, gravitational, Coriolis and centrifugal effects, friction, mechanical flexibility, backlash, hysteresis and actuator geometry.

- accurate control is required over a wide range of operating conditions.
- there is cross-coupling between neighbouring inputs and outputs of the system.
- the system dynamic parameters are time varying, for example due to changes in payload, configuration, speed of motion and component wear.

These problems are compounded by the subsea application considered here :-

- the unstructured workspace requires a reactive system, so tasks cannot be predefined and the control action cannot be determined a priori. This precludes certain types of control.
- operating conditions are unknown and time varying. Uncertainties in environmental parameters, such as contact stiffness, are important.
- offshore manipulators are crude when compared to typical laboratory robots, and generally sacrifice performance and accuracy for mechanical robustness. In addition, limited instrumentation is available on these manipulators.
- the manipulator used here utilises direct drive actuators rather than a gear or belt transmission as is common on most manipulators. Gearing reduces the inertia and disturbances as seen by the actuator, by as much as 100:1. Consequently, direct drive manipulators experience a much greater variation in dynamics than those that use transmissions.
- subsea robots are almost exclusively hydraulically actuated and the large payload capacity of these robots exacerbates the nonlinearities of the system. For instance, the Slingsby (SEL) TA9 manipulator used in this study can handle payloads of up to 80 kg, about 2.5 times its own weight. Further, the actuator itself is highly nonlinear due to compliance, leakage and nonlinear flow of the hydraulic fluid.
- additionally, the hydraulic direct drive actuator used here does not have the same

benefits of minimal friction and backlash as does a typical electrical direct drive actuator.

There are many different manipulator control techniques available, with the particular application and the manipulator itself determining which is most appropriate. For example, force measurements are generally noisy so controllers that use signal derivatives should be avoided for force control. Similarly, a controller for a robot with a gearbox would have a different set of specifications to one for a direct drive manipulator, since the gearing decreases the inertial effects seen by the actuators by as much as 100:1.

2.3 Manipulator Control Schemes

Manipulator controllers can be classified into two broad categories, namely joint space control schemes and Cartesian space control schemes. Joint space schemes have control loops local to each joint of the manipulator, whereas Cartesian space schemes have control loops acting on Cartesian space variables.

This classification can also be applied to controllers that perform constrained motion control, that is where the manipulator is in contact with an object and the contact force or torque is controlled. However, there are additional categories of control strategy associated with force control, which are distinct to those for unconstrained motions.

This section introduces the various control structures used for both unconstrained and constrained motions, with the controller itself being regarded as a "black box". The various control techniques used for manipulator control will be described in Section 2.4.

2.3.1 Joint Space Control Schemes

A joint space control scheme uses individual controllers operating at each joint of

the manipulator. These control the angle, velocity or torque of each joint[†] independently of the other joints in the manipulator, resulting in a simple single-input single-output (SISO) controller. These schemes work well when the robot is moving slowly, where any coupling between neighbouring joints is minimal. However, at high speed these interactions can be significant and act as disturbances on the independent controllers, with consequent degradation in performance.

Manipulation tasks performed by robots are generally specified in Cartesian space, whether it be relative to the end-effector or with reference to some global coordinate system. A transformation is therefore required to translate the desired Cartesian space motions into appropriate joint space motions. This transformation is referred to as the *inverse kinematics* of the manipulator, and is a nonlinear function of joint angles and link lengths. The resulting controller structure is shown in Figure 2.1a, where desired Cartesian positions, ${}^{C}X_{d}$, are transformed into desired joint angles, θ_{d}, which are then controlled by the independent joint space controllers. This structure can also be applied to control of velocities, in which case the required transformation is the *inverse Jacobian*.

The solution to the inverse kinematics (and Jacobian) is complex for all but the simplest of manipulators. Indeed, depending upon the manipulator configuration, there may be no, multiple or infinite solutions [1.8]. The solution to the inverse kinematics and Jacobian are described in Chapter Three, however the above mentioned problems associated with these inverses are not addressed in this thesis.

2.3.2 Cartesian Space Control Schemes

In Cartesian space control schemes the Cartesian variables, either position or velocity, are controlled directly. This can be achieved in two ways, either by transforming

[†] Throughout this thesis joints are referred to as revolute, with references being made to joint angles, angular velocities and torques. However, this discussion also pertains to linear joints, which are described by linear positions, linear velocities and forces.

the errors in Cartesian space into errors in joint space and then using joint space controllers, or by controlling directly in Cartesian space. The former is an extension of the joint space control schemes and is shown in Figure 2.1b, whereas the latter treats the robot as a single multi-input multi-output (MIMO) system, as shown in Figure 2.1c.

Both methods use the *forward kinematics* in the feedback path to determine the position of the manipulator's end-effector from the measured joint angles. Alternatively, for a Cartesian velocity controller, the *Jacobian* would be used in the feedback path. These transformations are again nonlinear functions of joint angles and link lengths, but are much less problematic than their associated inverse functions.

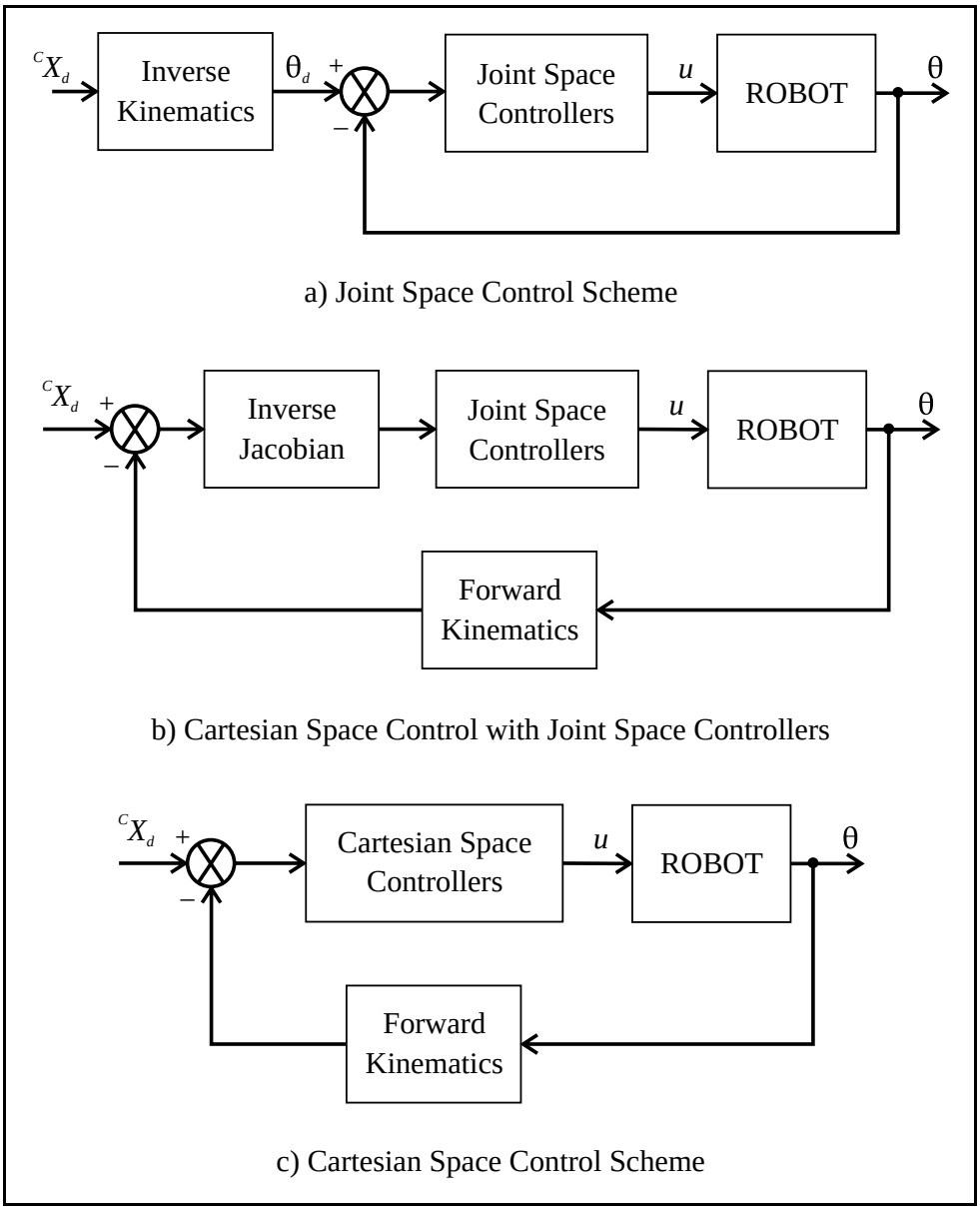

Figure 2.1 Joint Space and Cartesian Space Control Schemes

The controller structure of Figure 2.1b uses the *inverse Jacobian* to transform the errors in Cartesian space into joint space. These joint space errors are then acted on by the independent SISO joint space controllers, which again have the limitation that the performance can degrade if coupling becomes significant. Additionally, since the inverse Jacobian varies with the manipulator configuration, the gain and hence the response of the controlled system changes as the robot moves throughout the workspace [2.1]. The SISO joint controllers must accommodate these variations to maintain closed loop performance. One advantage of this approach is that the resulting controller implementation is still relatively simple.

A similar scheme to the one just discussed, utilises independent SISO workspace controllers, with the *Jacobian transpose* transforming the outputs of the controllers into joint space commands [2.1]. This scheme has exactly the same features and limitations as the one shown in Fig 2.1b.

The controller structure shown in Figure 2.1c uses a MIMO controller that operates directly on the Cartesian space errors. These schemes have the advantage that they can compensate for any coupling between joints, giving improved control when the manipulator is moving quickly.

Cartesian space controllers are preferable to joint space controllers when complex manipulations are to be performed, since both the task and desired performance criterion are naturally specified in the Cartesian frame of reference. Furthermore, some complex manipulations are difficult to realise using purely independent joint level controllers, for example the control of the contact forces during constrained motions. However, these benefits are at the expense of increased controller complexity.

2.3.3 Constrained Motion Control Schemes

Many tasks require the control of contact forces, for example grinding and assembly

operations. These tasks cannot be achieved by controlling the position of the manipulator since the high rigidity of the robot will produce large and potentially catastrophic forces for even the smallest of positional errors. Therefore, control of end-effector forces is essential for these operations, which leads to an increase in effective positional accuracy of the manipulator.

One simple way to alleviate this conflict is to use *passive compliance*, by introducing a mechanically soft device that reduces the effective stiffness of the robot. One such mechanism used for peg-in-hole insertions is called *remote centre compliance*. Passive compliance devices are usually specific to a particular task, and the forces are not directly controlled and some positional accuracy is lost.

Direct control of the force exerted by a manipulator is termed *active compliance*, and since this is programmed rather than a physical mechanism, the characteristics can be changed to suit different operations. One of the simplest ways to achieve active compliance is to reduce, or "soften" the gains of the position control loops, thereby creating a manipulator that exhibits an appropriate stiffness.

There are many different approaches to realise active compliance, and attempts to categorise the various schemes is the subject of many review papers [2.2, 2.3]. The most natural division between the strategies is the distinction between *explicit force control* and *implicit force control*.

The fundamental difference between these two approaches is that explicit force control directly controls the force, whereas implicit force control regulates the force via an inner position or velocity loop. Schematic diagrams of these schemes are shown in Figure 2.2, and since specific controllers can be derived in either Cartesian or joint space, the required transformations are omitted in the interest of clarity and generality.

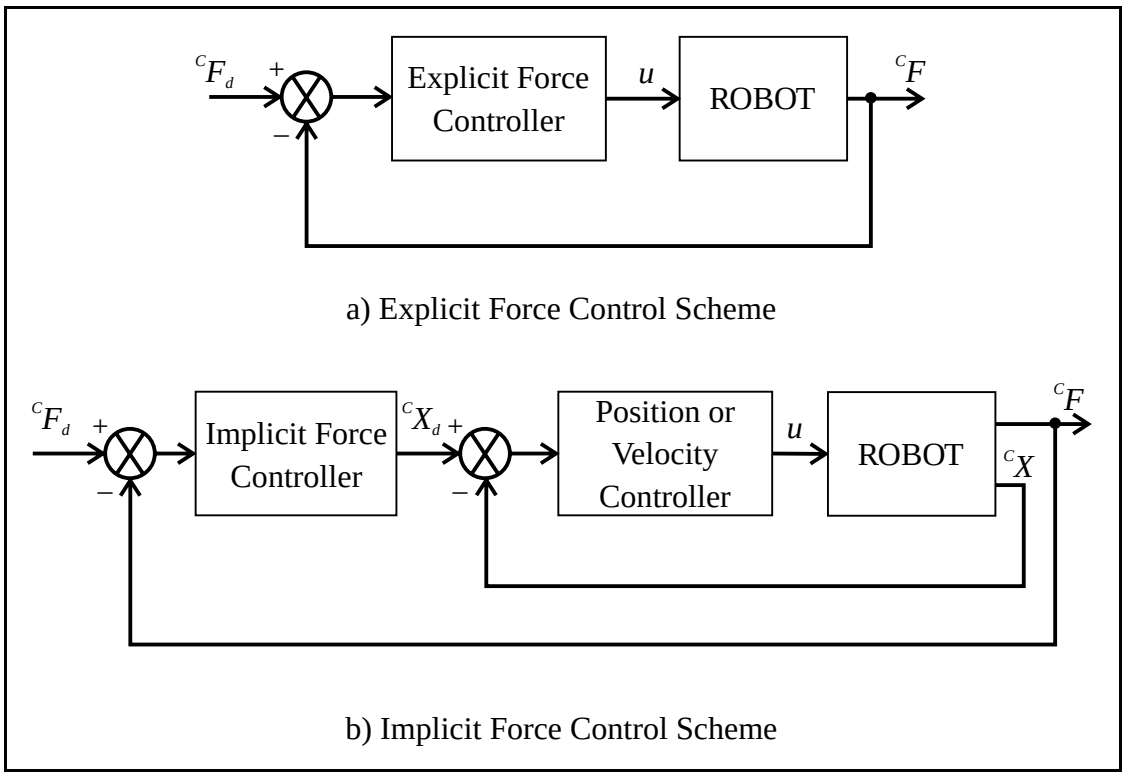

Figure 2.2 Explicit and Implicit Force Control Schemes

Explicit Force Control

Explicit force control uses the error between desired and measured forces to directly compute the control signals applied to the actuators, Figure 2.2a. The controller can be formulated in either Cartesian or joint space, with the latter using the *Jacobian transpose* to transform the Cartesian force errors into joint space errors, which are then acted upon by independent joint space controllers. These two approaches follow the discussion of the previous section, corresponding to Figures 2.1c and 2.1b respectively.

Implicit Force Control

Implicit force control uses an outer force control loop which surrounds an inner position or velocity loop which provides the control to the actuators, Figure 2.2b. The principle behind this is that the force controller determines suitable positions or velocities that would realise the desired forces. These are used as the set-point for the inner controller, which causes the robot to move and generate the required force. The inner controller can

be formulated in either Cartesian space or joint space, as described in the previous section, with all three schemes shown in Figure 2.1 being applicable. Again, the *Jacobian transpose* provides the transformation from Cartesian force errors to joint space errors where required.

The idea of implicit force control was first developed by Salisbury [2.4], where the force is regulated by an inner loop that controls the position of the robot in the direction of the constraint surface. The gain of the outer force loop determines the effective stiffness of the robot in the specific direction, and hence this method of force control is referred to as *stiffness control*.

An ideal position controller has infinite stiffness since it completely rejects force disturbances, whereas an ideal force controller has zero stiffness since it maintains a contact force, irrespective of position. The concept of stiffness control allows the stiffness of the manipulator to be specified by the designer between these two extremes.

This is achieved by adjusting the desired positions, ${}^{C}X_{d}$, that generate the desired forces, ${}^{C}F_{d}$, using the following expression in the outer controller :-

$$ {}^{C}F_{d} = K_{p}({}^{C}X_{d} - {}^{C}X_{e}) \tag{2.1} $$

where K_p is controller gain, and can be interpreted as stiffness of manipulator, ${}^{C}X_{e}$ is the position of the constraint surface. Therefore, the outer force controller should ideally contain a model of the interaction between the robot and environment, though variants of stiffness control have been suggested where this requirement is relaxed. Some reported stiffness control schemes do not use a force reference, and merely maintain a prescribed relationship between force and position, regardless of absolute values [2.5].

Whitney [2.6] proposed a similar implicit force control scheme, referred to as *damping control*, which utilised an inner velocity control loop to provide better stability than stiffness control. A practical extension of this was reported by Youcef-Toumi [2.7].

Both of these implicit schemes were generalised in the work on *impedance control*

developed by Hogan [2.8], which used both position and velocity inner loops. The outer controller is then designed so that the robot behaves like a combination of a spring and a damper, termed the target impedance. This again governs how compliant the robot appears, and is set so as to realise the required task. Recently this work has been revisited [2.9] and equivalence between implicit and explicit force control has been shown for certain impedance control formulations.

Implicit controllers tend to be more robust to parameter variations than explicit schemes. However, Stokić [2.10] reported that they are slower and less accurate due to determination of the equivalent positions/velocities corresponding to the desired forces, this being due to inaccurate kinematic models and disturbances such as friction.

Few tasks can be achieved using the control of end-effector forces alone, and simultaneous control of end-effector position is also needed to realise practical tasks, such as the mating of parts and sliding motions. Forces are controlled in constrained directions, while positions are controlled in the orthogonal unconstrained directions.

An implicit force control scheme can realise simultaneous position/force control by using suitable stiffness in the appropriate directions, high stiffness in position controlled directions and low stiffness in those that are force controlled. This is implemented by specifying appropriate values for the diagonal elements of K_p (a 6×6 matrix for the full Cartesian space problem) in Equation 2.1.

Another approach for simultaneous position/force control was first suggested by Paul [2.11]. He partitioned the manipulator's actuators into two sets; one to control the position over a surface and the other to control the force normal to the surface. This is simple for Cartesian robots whose orientation coincide with the constraint surface. However it is too simplistic for use with a generalised manipulator operating over a generalised surface.

A more comprehensive scheme was proposed by Raibert [2.12], in which the errors in the position and force sub-spaces are controlled by two independent controllers. The outputs from these controllers are then summed, giving the actuator drive signal which represents that particular joint's contribution to satisfying both the position and force commands. This control strategy, referred to as *hybrid position/force control*, is shown in Figure 2.3. A Cartesian velocity controller can be used to augment or replace the position controller to provide improved stability.

The key element within the hybrid position/force controller is the *compliance selection matrix*, $S_C = \text{diag}[s_1, s_2, .., s_n]$, where n is the number of degrees of freedom (DOF) of the manipulator. This determines which directions are to be position controlled ($s_i = 0$) and which are under force control ($s_i = 1$), where $i \in \{1, 2, .., n\}$.

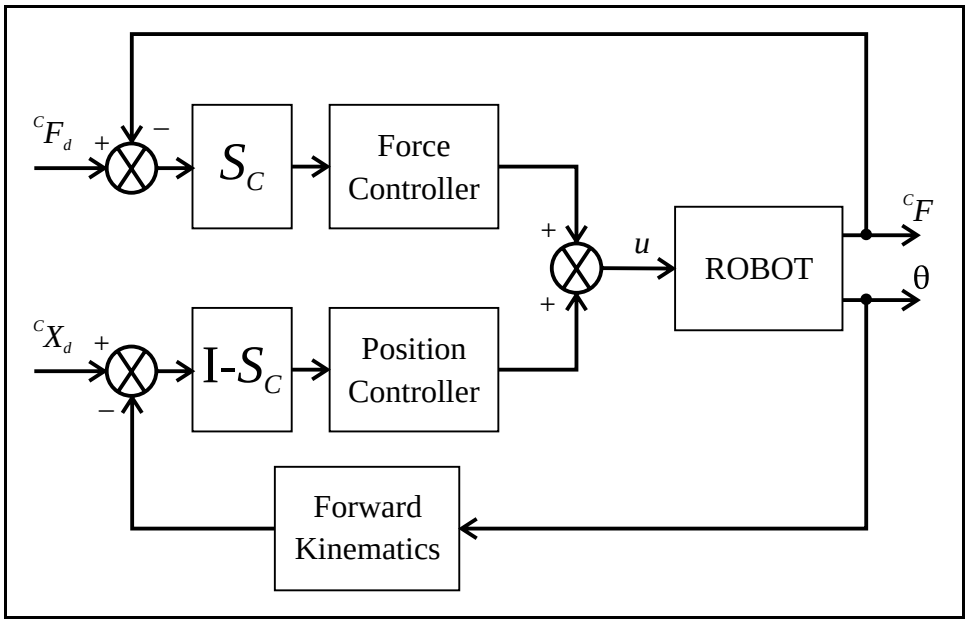

Figure 2.3 Hybrid Position/Force Control [2.12]

The original formulation [2.12] did not prescribe a particular control law, rather it was presented as a control architecture. It is essentially an extension of a Cartesian position controller that allows force control in orthogonal constrained directions. The position controller can be designed using any one of the schemes given in Figure 2.1, and similarly the force controller can be either explicit or implicit. Thus, one important advantage of

having separate position and force controllers is that they can be developed separately, using techniques appropriate to that mode of control.

Raibert [2.12] illustrated the concept using an example with fixed gain joint space PID controllers for the position controller (cf. Figure 2.1b), and explicit force control implemented using fixed gain joint space PI controllers. Many other control schemes have been applied to this general framework. For instance, Zhang [2.13] formed the position errors in joint space using the inverse kinematics of the manipulator, resulting in the structure given in Figure 2.1a. A further technique is the Parallel Approach, presented by Chiaverini [2.14], where the force and position controllers act on the full-dimensional space without using the selection matrices. Conflicts between the position and force controllers are managed by a rule based priority strategy. Hybrid position/force controllers have also been reported with both implicit force control [2.15, 2.16, 2.17] and explicit force control [2.18, 2.19].

The hybrid position/force controller has been shown to be unstable for certain manipulator configurations [2.20, 2.21], even when the manipulator Jacobian is well conditioned. However, Fisher [2.22] corrected an inappropriate transformation, the use of the inverse Jacobian, solving the instabilities reported in the earlier papers. Many papers have been written addressing problems of stability and the different implementations that are possible within the structure of hybrid position/force control, and a good summary of this work is presented in [2.23].

2.4 Methods for Robot Control

There is an extensive body of research on the application of advanced control to robotics, and it is the subject of many books [2.24, 2.25, 2.26], yet virtually all present day industrial robots are controlled using simple fixed gain PID controllers. Whilst this is primarily due to their simplicity, a lack of understanding about and scepticism towards

advanced controllers also exists. It has already been noted that advanced controllers can offer improved control, in terms of accuracy and speed, over a wider range of operating conditions. Furthermore, advanced actions can be realised using strategies such as Cartesian space control and hybrid position/force control, thereby widening the range of tasks that can be automated.

Any advanced controller must be implemented on a digital computer due to its complexity. Indeed this is often the required implementation as sensors and the interfaces to other sub-systems, such as vision and planning functions, are often computer based.

The previous discussion introduced the various structures that are available for robot control, with the actual controller being treated as a "black box". This section looks at the different control techniques that can be employed within these structures, and are grouped into well know categories. Distinctions between independent SISO joint space controllers and MIMO Cartesian space controllers will be highlighted where appropriate.

2.4.1 Model Based Control Techniques

Fixed gain PID controllers are widely employed for manipulator control. A proportional-derivative controller is the ideal structure to control a pure inertia, since the resulting closed loop system is second order with pole locations determined by the controller gains. However, as shall be shown in Chapter Three, centrifugal, Coriolis, gravitational and friction effects are also present in the manipulator dynamics. An integral term is often used to remove steady state errors caused by these terms, but the dynamic effects are more difficult to compensate for. Furthermore, such fixed gain controllers are only tuned for one particular set of conditions, and if these change the control action will degrade. To guarantee stability the controller is often tuned for the worst possible situation, and hence the system will have a slow, sub-optimal response for most conditions.

Whiting [2.27] successfully extended a PID control scheme to cope with

nonlinearities arising from payload variations in an electrohydraulic actuator. Another application of PID control to a hydraulic actuator was made by Liu [2.28], who developed an optimal tuning method to meet a set of defined performance and stability requirements.

If a controller is designed with knowledge of the system dynamics, then variations in operating conditions can be accommodated and the system response maintained. These methods are referred to as *model based controllers*, and can range from simple gravitational compensation schemes to feedback linearisation of the full manipulator dynamics. Clearly the suitability of a model based controller is dependent upon how well the system under control is known.

An ideal model based controller consists of the inverse of the system dynamics, used as a pre-compensator to the actual system. The control inputs required to meet the desired positions, velocities and accelerations can then be calculated directly from the inverse system model. Thus, the system is driven open loop with perfect cancellation between the inverse dynamics and the real system.

Obviously this is impractical as no real system is known perfectly, and any unmodelled effects will not be compensated. Feedback is used to alleviate this, and can be introduced by augmenting the open loop model based controller with a classical, usually fixed gain PID, feedback controller. These two controllers are often referred to as the *primary controller* for the model based part, and the *secondary controller* which maintains set-point tracking in the presence of modelling errors and unmodelled disturbances. This approach is shown in Figure 2.4a for a SISO joint angle controller, and is termed a *feedforward model based controller*. This strategy is equally applicable to Cartesian position, velocity or force control, providing a suitable model of the system exists.

The primary controller is designed using any available knowledge of the system under control. Primary controllers which contain a complete robot model were first proposed by Paul [2.29] and Bejczy [2.30], and are referred to as *computed torque*

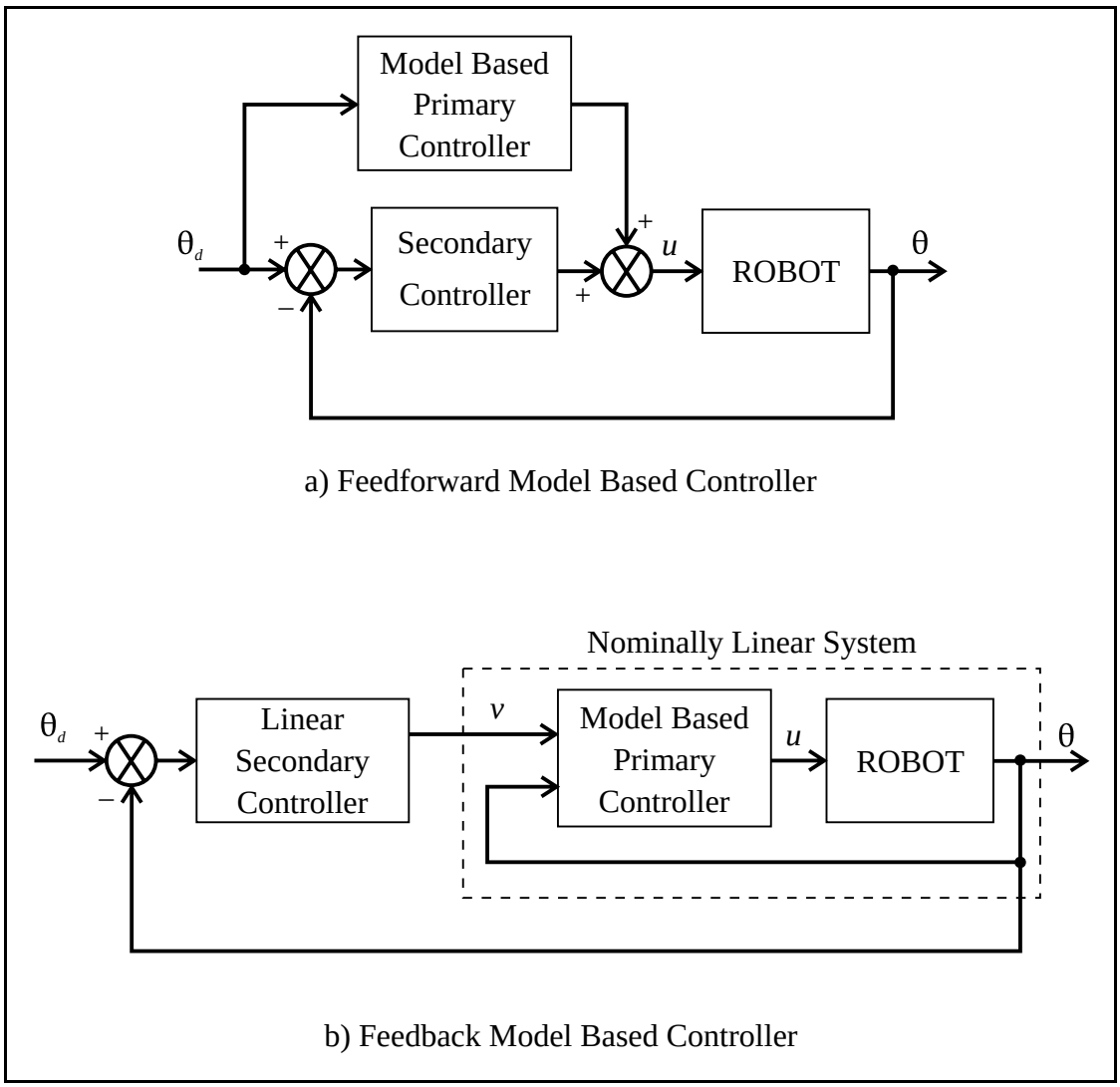

Figure 2.4 Model Based Controllers

controllers. This name arises since the primary controller computes the torque required to follow the desired positions, velocities and accelerations from the full manipulator model. Consequently these schemes are computationally intensive as the model equations involve many complex trigonometric functions.

Simpler schemes exist which use only part of the manipulator model in the primary controller, such as compensation for gravitational and kinematic effects [2.31]. The resulting controller is more practical since the gravitational part of the model is relatively simple and its parameters are often well known. This reduces the amount of integral action required in the secondary controller, since the primary controller provides the "holding torque" that maintains the robot's position in the presence of gravity. This decreases the

tendency for limit cycles in the system output, which arise from interactions between the integral action and friction present at the joints of the robot.

Other feedforward control strategies have been proposed, such as using the inertia terms of the robot dynamics. Again this is relatively simple and can be applied either to the joints independently, or to the robot as a whole thereby compensating for any coupling between joints. Force control signals are commonly applied in this way, using the Jacobian transpose to determine the joint torques required to realise a specific end-effector force.

Another way of reducing the computational burden of these model based schemes is to use a linearised model of the system under control. This can take the form of a state-space controller designed to position the poles of the closed loop system, or to optimise some performance criterion. However, the linearised model quickly becomes inappropriate as the manipulator moves throughout its workspace, and hence degrades the control. This approach may be effective if deviations from the linearisation point are small, or alternatively if different linearised models are used as the robot moves along its trajectory.

An alternative approach, which has attracted much theoretical work, uses model based feedback to linearise and decouple the manipulator. This method, referred to as a *feedback model based controller*, is shown in Figure 2.4b, again for a SISO joint angle controller for clarity. Here, the inner primary controller is designed using the inverse system dynamics to give an ideally decoupled and linearised system. For a manipulator, the combined primary controller and system can ideally be reduced to a set of decoupled double integrators :-

$$\boldsymbol{v} = \ddot{\theta} \tag{2.2}$$

where v is the input to the *nominally linear system*. Therefore, the primary controller can be viewed as an input transformation that moves the problem from choosing desired torque inputs, which is difficult, to choosing acceleration inputs, which is easier.

It is then a simple task to design a secondary controller that regulates the nominally linear system, giving the required closed loop system response. The secondary controller also compensates for errors in the model based primary controller, to ensure set point tracking and disturbance rejection. This approach is also occasionally referred to as computed torque control, but differs from previous law since it uses feedback rather than feedforward.

Feedback linearisation is usually performed in joint space, however it can be applied using a manipulator model expressed in Cartesian space. This was first proposed by Khatib [2.32] and is known as the *operational space formulation*. This results in a nominally linear and decoupled Cartesian space system :-

$$\boldsymbol{v} = {}^{C}\ddot{\boldsymbol{X}} \tag{2.3}$$

The secondary controller is therefore designed in Cartesian space, with the associated advantages discussed in Section 2.3.2. This method has since been extended to constrained motion control [2.33], yielding a controller that can achieve simultaneous control of positions and forces. However, the transformation from joint space to Cartesian space exacerbates the already complicated dynamic equations, and problems can arise at singularities in the workspace due to ill-conditioning. The use of a pseudo-inverse for the Jacobian and kinematics can alleviate these problems somewhat.

The main drawback with model based control is that it is computationally intensive for all but the simplest of cases. Much research has been carried out to reduce this computational burden, for example a simplified model of a PUMA robot has been derived that uses 10% of the complete model calculations, yet is accurate to within 1% of the full model [2.34].

The use of simplified models, as in the case of gravitational compensation, alleviates this but then only provides marginal improvements over conventional fixed gain

controllers. The inverse static actuator characteristics can be used within a feedback controller to linearise any nonlinearities associated with the actuator. Other schemes compute the manipulator model terms off-line, which are then used in a large look-up table [2.35]. However, these are best suited to robots performing repetitive tasks where a priori knowledge of the trajectory is available. Another approach to ease computational expense is to express the model in *configuration space* where the model parameters are functions of the manipulator position only [2.33]. The computational burden of all model based schemes can be reduced by using a background process, operating at a reduced sample rate from the main control loop, to calculate the model terms. However, any form of discretisation of the calculated model will also lead to inaccuracies.

2.4.2 Optimal Control Methods

The aim of optimal control is the minimisation of a suitable performance criterion of the system under control. For a robot the most useful performance criterion is the minimisation of position or force errors, though the time to complete a task [2.36] or the demand on the actuators can also be used.

Optimal control utilises a linear model of the manipulator dynamics. A suitable performance index is then minimised, subject to the constraints imposed by the model and bounds on the control input. The linear approximations used result in a *linear quadratic optimal controller*. Since the nonlinear dynamic model is not used, the response is sub-optimal away from the operating point.

As mentioned in the previous section, an optimal controller can be used for the secondary controller of a model based scheme. There is also a class of optimal controllers that utilise a low order linear model of the robot which is determined on-line, rather than a fixed predetermined model. These controllers will be discussed in the context of adaptive control methods in Section 2.4.4.

However, optimal controllers are usually too complex to be used with manipulators with more than four degrees of freedom [2.25]. Further, a priori knowledge of the desired trajectory is often required, and this is not available in this particular application.

2.4.3 Robust Control Strategies

Robust control techniques were initially devised to address the problem of poorly known system dynamics, and they are therefore insensitive to modelling errors and variations in the system under control. Robust controllers have been used in the secondary controller part of model based schemes, to cope with the presence of uncertainties in the model based primary controller.

One nonlinear robust control technique [2.37], which utilises the Second Method of Lyapunov, guarantees the stability of the closed loop system providing the errors in the model are bound within a known range. The resulting control law is a discontinuous switching function and, due to the discrete implementation, the control signal rapidly alternates between different values. This phenomenon is known as *chattering* and is problematic since excessive activity of the control signal can cause heating and rapid wear within the actuators. Another problem is that the high frequency content of the signal can excite unmodelled dynamics of the manipulator, such as flexibility.

Another robust method, termed *variable structure control* (VSC) or *sliding mode control*, is similar to the Lyapunov method in that it uses a discontinuous switching function [2.38, 2.39]. This drives the system rapidly onto a *switching line* or *sliding surface*, defined in the state-space of the system, as shown in Figure 2.5a. After this initial *reaching phase*, the system response is then governed entirely by the equation of the line, called the *sliding mode*, see Figure 2.5b. The system then remains on the sliding surface and is insensitive to disturbances and system variations, hence providing robustness.

The theory behind VSC is based entirely on continuous time systems, and a discrete

time implementation is an approximation of this. To ensure the stability of a discrete VSC, high sample rates are required to prevent the system moving away from the sliding surface during sample intervals. The requirement of high sample rates counteracts one of the main advantages of VSC, namely their low computational requirements.

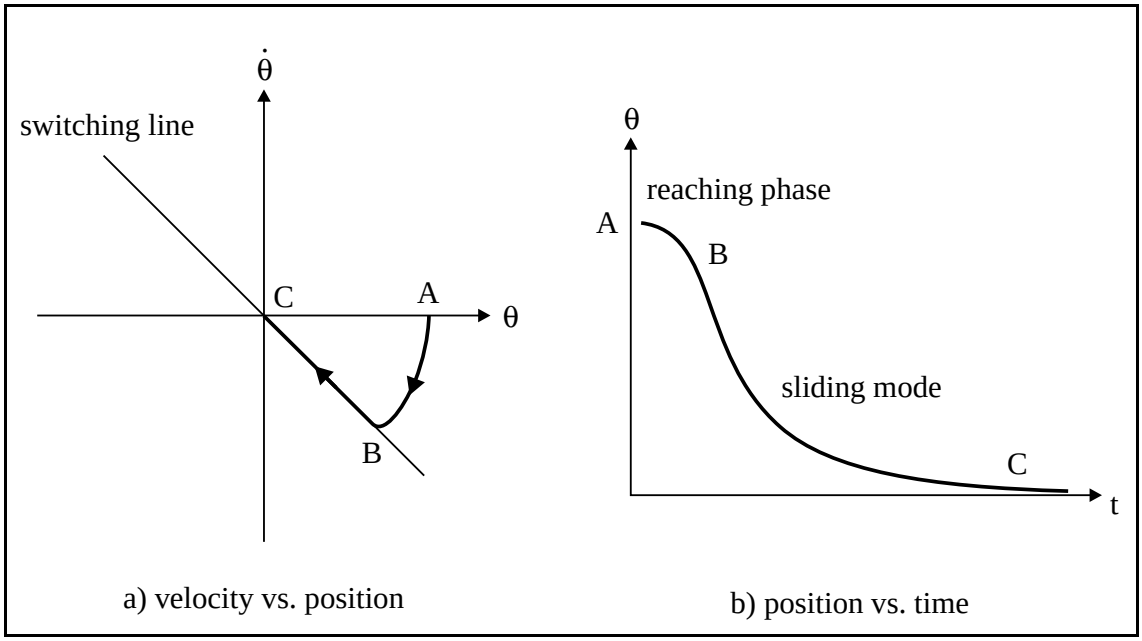

Figure 2.5 Variable Structure Control (From Reay [2.38])

The problem of chattering is present to an even greater degree with VSC and several approaches have been proposed to reduce this. One technique is to split the control signal into continuous and discrete components [2.38], another involves using a finite width boundary layer either side of the sliding surface [2.39]. A recently proposed VSC reduced chattering by increasing the switching frequency beyond the bandwidth of the system, using dedicated hardware circuits [2.40]. Another utilised fuzzy tuning rules to achieve the same objective [2.41].

Another difficulty with VSC is that the derivative of the error signal is required to realise a first order sliding surface, and this can be problematic if signals are noisy. Further problems may arise with VSC if the initial state of the system under control is far from the sliding surface, since during the reaching phase the system dynamics are undefined and it

may never reach the surface. Despite these problems, application of VSC to robots now forms an extensive body of work, ranging from SISO control [2.39], to multivariable robot control [2.38] and hybrid position/force control [2.40, 2.42].

The methods just discussed are classed as nonlinear robust control since the resulting control law is a nonlinear function. Linear robust controllers, based on the H_2 and H_∞ design methodologies, have also been applied to manipulator control [2.43, 2.44]. These methods result in a highly robust system, but this is often at the expense of conservative performance. These controllers need careful selection of their cost weightings and again, signal derivatives are usually required. Another form of linear robust control, termed Quantitative Feedback Theory, has found application to the force control of hydraulic actuators [2.45]. This method has the benefit that the order of the resulting controller can be restricted, thereby yielding a more practical controller. Robust controllers that incorporate an adaptation mechanism have also been proposed [2.37, 2.46]. A good survey of robust robot control techniques and applications is given in [2.47], which covers all of the major categories of robust control mentioned above.

2.4.4 Adaptive Controllers

The controllers discussed previously have constant parameters, and are designed to be stable even when there are variations in the system under control. An alternative approach, termed *adaptive control* [2.48, 2.49], automatically adjusts the controller gains as the system changes, as shown in Figure 2.6. The controller therefore acts to maintain the closed loop system response in the presence of variations in the system.

The simplest form of adaptive controller is called a *gain scheduling controller*, where the gains are adjusted on the basis of a suitable system parameter. A gain scheduling force controller is presented in [2.50], where the gain is a function of the load applied to a hydraulic leg, embodying the changes that are occurring in the underlying system. However,

such schemes can only adapt to variations in the system which are known a priori and manifest themselves in a measured system variable.

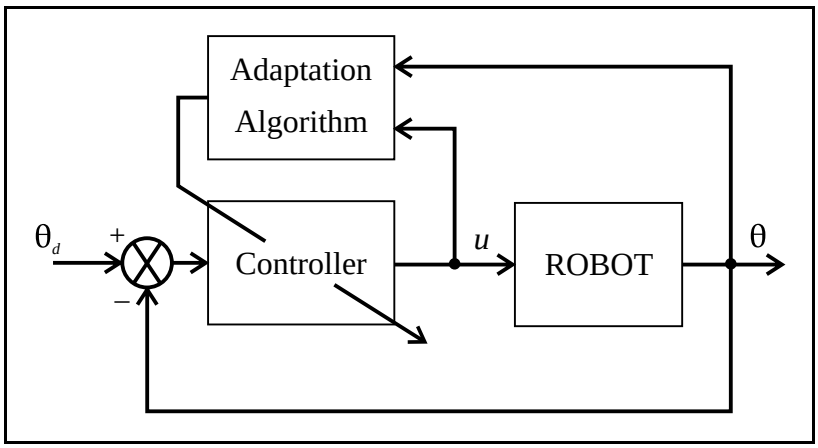

Figure 2.6 Generalised Adaptive Controller

Model based adaptive schemes have been proposed [2.51] where those coefficients of the robot model that are not well known or are changing, are updated automatically. This is achieved using a *system identification* algorithm, which uses past input and output values of the system to estimate the parameters, for example payload mass. The nonlinear equations of motion of the robot are expressed as a linear function of joint outputs and model parameters. These parameters are estimated using a Lyapunov function candidate approach, and they converge to their true values providing certain constraints are met. This method requires measurement of the joint angles, velocities and accelerations which can be problematic due to noise.

Variants of this scheme have been proposed that alleviate the need for acceleration measurements, one notable method being that of Slotine and Li [2.52] which also does not require the inversion of the inertia matrix of the robot model. This particular area of research has seen much work and is still active, addressing issues of convergence, stability and computational burden. Another proposed scheme [2.53] used a combination of direct and indirect adaptive controllers, to improve the disturbance rejection of force control. However, these model based adaptive controllers are generally only practical if the number

of estimated parameters is restricted. The problem becomes complex if the full manipulator model is to be estimated.

Therefore, simpler adaptive schemes have been investigated, which are applied on an independent joint basis, or that use a low order linear approximation of the robot model. One such scheme is the *model reference adaptive controller* (MRAC), as proposed by Dubowsky [2.54]. Here, a reference model is specified and an adaptation algorithm adjusts the controller gains so that the output of the actual system follows that of the reference model. The advantage of these controllers is that they require only a moderate number of computations, and do not contain any of the complex mathematical dynamic models used in previous methods. Analysis of the stability of such systems is difficult, though many successful laboratory implementations have been reported [2.55].

Another class of adaptive control, referred to as *self-tuning control*, utilises a low order linear approximation of the robot model, whose parameters are estimated on-line from past input and output values using a system identification algorithm. The model structure used can be either SISO or MIMO, and have joint space or Cartesian space outputs, as appropriate. The use of a linear autoregressive model allows the use of efficient recursive identification algorithms, such as *recursive least squares* (RLS) [2.56]. The controller parameters are then designed based on this linear model, so that the closed loop system meets some prescribed performance criterion. Any variations in the dynamics of the system will be tracked by the identification algorithm, and hence automatically accommodated by the controller.

There are several ways of designing a self-tuning controller [2.57], namely linear quadratic Gaussian control, pole placement control and using multi-stage predictive control. A *linear quadratic Gaussian* (LQG) controller essentially optimises the output of the linearised model [2.58], and is designed along the same lines as the optimal controllers described in Section 2.4.2. A variant of this, called the *minimum variance controller*, has

been widely applied to robot control [2.59, 2.60], and differs from LQG in that the optimisation has no penalty on the demand to the actuators. These minimum variance controllers suffer problems when the system to be controlled is nonminimum phase, that is when fractional time delays are present, as is usually the case in digital implementations.

The use of *pole placement (PP) controllers* alleviates this restriction, and can also be used when the delays are present, which again can be problematic for minimum variance controllers. Pole placement schemes work by automatically adjusting the gains so that the poles of the closed loop system are placed at some specified location [2.61, 2.62]. Consequently, the system response remains constant irrespective of changes in the underlying system. This type of self-tuning controller has found fewer applications to robot control compared with minimum variance controllers, the main reason for this being the significantly higher computational expense [2.63]. However, pole placement is more intuitive for the designer as it resembles a classical control design method, rather than using weighting variables which require careful selection [2.26] and do not have a clear physical meaning. Pole placement controllers also yields smoother control signals.

Multi-stage predictive controllers also use an optimisation criterion, but avoid the problems associated with minimum variance control by using predicted future values in the minimised cost function. One particular type of predictive controller is the *generalised pole placement* (GPP) controller, which incorporates a pole placement procedure to determine its weighting factors automatically [2.64]. These predictive control schemes can be computationally intensive depending upon how many future predicted values are required, as these are used in a matrix inversion. Also, a priori knowledge of the desired trajectory is required to achieve optimal control. However, if this is not known and assumed to be equal to the current set-point, the resulting control is sub-optimal. Nevertheless, predictive control has seen successfully application to both position [2.65] and force control [2.66] of hydraulic actuators.

Adaptive controllers have been applied to most manipulator control problems, including hybrid position/force control [2.67, 2.18, 2.19]. These schemes overcome the inadequacies of fixed gain controllers which cannot take into account of changing operating conditions and unknown dynamics. However, the controllers reported have only been verified in theory or simulation, or applied to specialised electrically actuated robots. Little work has been published regarding adaptive force control of industrial hydraulic manipulators [2.50, 2.68].

2.4.5 Other Control Schemes

The field of robot control contains many diverse solutions, with the established approaches being described in the sections above. Indeed, much reported work blurs these boundaries by combining different control methods within the proposed controller. More novel approaches are now being explored, using the concepts of neural networks, learning control and fuzzy logic.

Neural networks, based on a "bottom up" approach to artificial intelligence, have shown the ability to deliver simple yet powerful solutions to problems that have proved difficult for conventional computing. Neural networks have been applied to robot control, where the network learns the characteristics of the robot by adjusting its own weightings [2.69]. The neural network forms a nonlinear model of the manipulator [2.70] which can then be used within any of the aforementioned model based schemes, such as computed torque control and adaptive control [2.71]. A neural network needs to be trained prior to use, using a predefined set of learning data. This can be time consuming and essentially prevents it being used for unstructured tasks. This is an active area of research, and recently a neural network based force controller has been proposed [2.72].

Learning controllers work on a similar principle, in that the system learns the behaviour of the robot, it is the implementation that differentiates it from a neural

controller. In a neural network the information about the system is distributed across many synaptic weightings, whereas in a learning controller it is stored as independent quantities. Again this area has seen much recent work, particularly for implicit force control [2.73], including application to robotic de-burring [2.74]. However, the need to train these systems limits their applicability to repetitive tasks, also practical issues have yet to be addressed and few experimental results have been presented.

Controllers based on fuzzy logic use heuristic and qualitative rules [2.75], rather than the algebraic and differential equations of traditional controllers. These have been combined with adaptive [2.76] and neural schemes to enable these rules to adapt to changes in the system. Many forms of fuzzy robot controller have already been proposed, including compliance control for insertion tasks [2.77]. However, these studies are predominantly restricted to simulation studies.

2.5 Self-tuning Pole Placement Robot Controllers

The various control laws described above continue to be the subject of a vast number of research papers. Many subtle variations of the general schemes have been developed to address potential disadvantages with a particular type of control, though these generally sacrifice some of the associated benefits. When selecting which control method to use it is impossible to look at every form of controller published. Hence, the choice is often made on the basis of the most appropriate general controller for a specific application.

This section describes the choice of a suitable control law for use with subsea manipulators, the key features of which were introduced in Chapter One. The controller selected is a self-tuning pole placement controller and the specific controller developed in this thesis is then placed in the context of previously proposed self-tuning controllers, which are described in some detail.

2.5.1 Selection of Self-tuning Pole Placement for Robot Control

The problems encountered when controlling a typical subsea robot have been highlighted in Section 2.2. The most important ones are the requirement of reactivity, and the large variations in dynamics that arise due to the fact that it is a direct drive hydraulic manipulator. Constraints imposed by the physical manipulator also restrict the choice of controller that can be applied.

A model of the Slingsby TA9 manipulator used in this study can be derived from general assembly drawings, physical measurements and appropriate assumptions, and this is the subject of Chapter Three. However, this model is used solely for the purpose of simulation, as no model based parameters are used within the proposed controller. The reason a model based scheme was not used was because the hydraulic actuator dynamics further complicate the already computationally intensive manipulator model. Furthermore, typical subsea tasks are subject to unpredictable variations in payload and end-effector forces, and since this has a large influence on the robot dynamics, any model based scheme would quickly degrade.

The optimal and robust control methods were deemed unsuitable for this particular application for two reasons. The first reason is the requirement of signal derivatives. The Slingsby TA9 has analogue joint angle sensors which are noisy, thus any attempt to approximate the joint velocity from successive joint positions yields an unusable signal. Secondly, the rapid switching of the control signal would be problematic for a hydraulically actuated robot. Although the switching frequency can be made to be above the bandwidth of the hydraulic actuators, it is comparable to that of the servovalves that regulate the flow of hydraulic fluid. Servovalves are precision devices and would be susceptible to wear and even seizure when switched at such high frequencies.

Controllers using neural networks, fuzzy logic or learning tend to be based around traditional schemes, with the extensions providing the ability to automatically train the

controller. However, this would be of little benefit in this application, since subsea tasks are generally not repetitive.

An adaptive control scheme was deemed most suitable for this application, as it can automatically accommodate the wide variations in manipulator dynamics. Model based adaptive schemes were precluded for the reasons given above. Self-tuning controllers and model reference adaptive controllers are technically closely related [2.57], and it is the underlying philosophy that distinguishes them. Consequently, a self-tuning controller was chosen in preference to a MRAC scheme, simply because it is more intuitive.

Of the different forms of self-tuning controller described in Section 2.4.4, a pole placement scheme was selected. While it is certainly true that pole placement controllers are more computationally intensive than other approaches [2.40], there is scope for simplification to realise practical implementations. Furthermore, microprocessors are now providing vast computing power, and computationally efficient pole placement algorithms have been developed.

One of the main reasons for selecting a pole placement controller is that it can control a nonminimum phase system, which is typical when using digital control [2.78]. Other reasons include the smoother control, and the lack of weighting values that make other self-tuning approaches more difficult to design.

2.5.2 The Proposed Self-tuning Pole Placement Robot Controllers

The low order model at the heart of a self-tuning controller, enables the application of a self-tuner within any of the manipulator control schemes outlined in Section 2.3, be it independent SISO joint control or explicit MIMO force control. The low order model is simply constructed using the appropriate input and output signals, and the controller is then designed accordingly. The experimental manipulator cannot provide good velocity measurements due to noisy sensors, so the proposed controllers only operate on position

and force signals.

This thesis demonstrates the feasibility of applying self-tuning pole placement control to a Slingsby TA9 subsea hydraulic robot by first employing it in an independent SISO joint angle controller. This will show what benefits such a self-tuning scheme provides over the standard fixed gain controllers used on the manipulator.

This is then extended to the MIMO hybrid position/force control problem, where simultaneous control of forces and positions in Cartesian space is required. This will explore the benefits of using the more complex Cartesian space controllers, which should provide improvements over the SISO joint space controllers, as discussed in Section 2.3.2. The hybrid position/force control problem also allows investigation of the suitability of self-tuning controllers for force control.

An explicit force controller is used in preference to an implicit controller, since the model of the robot/environment interaction is not well known and will change from task to task. This relationship is required in an implicit controller. Also inner velocity loops, often required to improve the stability of implicit schemes, are not viable for this robot. Further, the large unmodelled disturbances present in typical underwater tasks could be problematic for implicit schemes.

A detailed discussion of the specific self-tuning controllers used in this thesis will be presented in Chapter Four. Stability and robustness analysis of these particular controllers is beyond the scope of this work. However, the performance of the proposed self-tuning controllers will be compared to conventional fixed gain controllers, for many different operating conditions of the experimental manipulator. The benefits of the proposed schemes will be presented from a practical perspective.

2.5.3 Previously Proposed Self-tuning Robot Controllers

There have been many different types of self-tuning control applied to manipulators,

for instance both SISO and MIMO controllers have found use, as have LQG, minimum variance and pole placement schemes. Also, the particular manipulator to which they have been applied has varied widely. Some are experimental while most are simulated, and the majority use ideal or electrical drives, with only a few being hydraulically actuated. A summary of the various SISO self-tuning robot controllers reported is given in Table 2.1, the actual controller designs used are discussed in more detail in Chapter Four.

The first reported application of a self-tuning robot controller was by Koivo [2.59], in which an LQG controller was applied to a simulated electrical robot. Lelic [2.64] reported the first application of a self-tuning controller to an experimental laboratory robot, this manipulator was powered by harmonic drives and had an inner velocity feedback loop. A GPP controller was used which performed better when a priori knowledge of the reference trajectory was available. Pole placement schemes were first proposed by Karam [2.79] and Broome [2.80], but both were implemented in simulation only.

The first application of a pole placement scheme to a hydraulic mechanism was reported by Finney [2.78], who used an experimental linear hydraulic ram similar to those that are used to actuate robot joints. Plummer [2.81] extended this work by including a forgetting factor and a novel covariance management algorithm to increase the reliability of the practical system identifier. The use of variable forgetting factors and numerically robust system identification algorithms were investigated by Ozsoy [2.82], however since the system identification was carried out off-line, it cannot strictly be regarded as a self-tuning control scheme.

The work of Sepheri [2.60] used a simulated hydraulic manipulator, and investigated the effect of augmenting a minimum variance controller with feedforward model based compensation of the actuator dynamics. Ananthakrishnan [2.68] reported the application of pole placement control to an experimental industrial hydraulic manipulator. It is referred to as a MIMO scheme, though it clearly uses independent SISO joint

	Controller Type	Exp. or Sim.	Controlled Variable	*n* DOF	Robot Type	Notes:
Koivo [2.59]	LQG	Sim.	joint velocity	6	Lab. Electrical	Stanford/JPL arm.
Lelic [2.64]	GPP	Exp.	joint angle	1	Lab. Electrical	Uses internal velocity feedback loop.
Karam [2.79]	PP	Sim.	joint angle	6	Ind. Electrical	Kuka IR161/15.
Broome [2.80]	PP	Sim.	joint angle	2	Small Ideal	Puma 560 with ideal torque sources for joints 2 and 3.
Finney [2.78]	PP	Exp.	linear position	1	Hydraulic Ram	Single linear hydraulic actuator.
Plummer [2.81]	PP	Exp.	linear position	1	Hydraulic Ram	Single linear hydraulic actuator.
Sepheri [2.60]	MV	Sim.	joint velocity	2	Small Hydraulic	Rotary hydraulic actuators.
Ananthakrishnan [2.68]	PP	Exp	joint angle	4	Ind. Hydraulic	Positech CC1A. Rotary/linear actuators
Eun [2.84]	PP	Sim	force/position	3	Small Ideal	Hybrid Position/Force Controller
Koivo [2.26]	PP, LQG	Sim.	joint angle	3	Small Ideal	Ideal torque source actuators.
Wang [2.85, 2.86]	PP	Exp	contact force	3	Ind. Electrical	Puma 560 with passive compliance.
Clegg [1.13, 2.87]	PP	Exp.	joint angle	1	Subsea Hydraulic	TA9. Rotary and linear hydraulic actuators.

Table 2.1 Different SISO Self-tuning Controllers used for Robot Control

Notes:

1) The controller types are linear quadratic Gaussian (LQG); minimum variance (MV); generalised pole placement (GPP); pole placement (PP).

2) *n* is the number of joints to which the independent controllers were applied.

controllers. This work was extended in [2.83] which demonstrated the superiority of a pole placement controller over a MRAC scheme, as well as its ability to cope with significant joint friction. Another application of SISO pole placement controllers to a MIMO robot control problem was reported by Eun [2.84]. The independent SISO controllers were applied to the hybrid position/force control problem of a simulated 3 DOF ideal manipulator and demonstrated good performance.

Koivo [2.26] presented a series of examples comparing both LQG and pole placement controllers, and also looked at the use of feedforward compensation of Coriolis and gravitational terms. Wang [2.85] reported the use of SISO self-tuning pole placement controllers for the outer loop of an implicit impedance force controller for a PUMA robot. However, the results presented, and those in a follow up paper [2.86], are inconclusive probably due to the low sample rates of the VAL-2 controller dedicated to the PUMA.

The initial work presented in this thesis [1.13, 2.87] complements the work that has gone before. Only Ananthakrishnan has applied pole placement control to an experimental hydraulic robot, and one which is more sophisticated than the Slingsby TA9 used here. Also, the work here investigates the performance benefits offered by these controllers over conventional fixed gain PID schemes for a wide range of conditions.

The SISO scheme initially developed here formed the basis for the extension of the self-tuning pole placement controller to the MIMO hybrid position/force control problem. Fewer MIMO self-tuning schemes have been proposed in the literature, and a summary of these are given in Table 2.2.

A MIMO decoupling pole placement controller was applied by Plummer [2.88] to an experimental 2 DOF electrohydraulic actuator system. However, the multivariable system identification used was applied off-line, and again it cannot be regarded as a self-tuning scheme. Nevertheless, it did demonstrate the benefits of multivariable control for such a system.

	Controller Type	Exp. or Sim.	Controlled Variables	*n* DOF	Robot Type	Notes:
Koivo [2.59]	LQG	Sim.	joint velocities	6	Lab. Electrical	Stanford/JPL arm.
Koivo [2.63]	LQG	Sim.	Cart. velocities and positions	3	Lab. Electrical	Stanford/JPL arm.
Koivo [2.89, 2.26]	LQG	Sim.	Cart. velocity and force	2	Small Ideal	Stanford/JPL arm with ideal torque sources for joints 1 and 3.
Ozsoy [2.90]	MV	Sim.	Cart. velocity and force derivative	2	Small Ideal	
Clegg [2.40 and Chapter Seven]	PP	Sim.	Cart. position and force	2	Subsea Hydraulic	TA9 joints 1 and 3. Linear hydraulic actuators.

Table 2.2 Different MIMO Self-tuning Controllers used for Robot Control

Notes:

1) The controller types are linear, quadratic Gaussian (LQG); minimum variance (MV); pole placement (PP).

2) *n* is the number of DOF to which the MIMO controller was applied.

Much of the MIMO self-tuning work has been carried out by Koivo, who was first to apply such a controller to a manipulator [2.59]. This initial work still operated in joint space, controlling the joint angle velocities, but used the multivariable structure to regulate the coupling between joints. This work was extended to a MIMO Cartesian controller [2.63], where Cartesian velocities were controlled. A more complex scheme was introduced in the same paper which controlled both Cartesian velocities and positions. A force-position-velocity scheme was then developed [2.89] and extended [2.26]. These schemes proposed by Koivo essentially model the system with Cartesian velocity outputs, and only use force errors within the minimised cost function in the LQG.

Another self-tuning MIMO controller was reported by Ozsoy [2.90]. This scheme used the Cartesian velocity and force derivative as the estimated model outputs and hence the controlled variables, for the simulated 2 DOF ideal manipulator. This model was coupled to a minimum variance controller, to give a self-tuning hybrid velocity/force system. However, the results presented were brief and did not clearly demonstrate the effectiveness of the developed system.

The self-tuning hybrid position/force controller developed in this thesis uses a MIMO pole placement scheme [2.40], which has not been previously applied. This controller goes further than Koivo's constrained motion controller, in that it explicitly uses the force measurements in the estimated process model. Furthermore, the simulation model to which the control scheme is applied includes realistic hydraulic actuator dynamics, whereas Koivo's work assumed ideal torque sources. This thesis also describes the implementation of the MIMO self-tuning hybrid position/force controller on the experimental Slingsby TA9 manipulator.

2.6 Summary

This chapter described the main approaches to the problem of manipulator control.

It first considered the distinction between joint space and Cartesian space control schemes and then discussed methods of controlling constrained motions, namely implicit and explicit force control. The concept of hybrid position/force control was introduced.

The many different control techniques that have found application to manipulator control were then presented. The merits and drawbacks of each particular method were discussed, and instances of successful applications of each technique were highlighted.

A self-tuning pole placement controller was selected for use in this application, and the reasons for this choice were discussed. The controllers developed here were then placed in the context of previously proposed self-tuning manipulator controllers, showing how they complement previously reported work and their own novel features.

Chapter 3

Manipulator, Actuator and Contact Modelling

3.1 Introduction

In this chapter mathematical models of the manipulator and actuators are developed, this includes modelling of any contacts made between the robot and objects or surfaces that may be within the workspace of the robot. The need for modelling stems from the requirement to understand how a particular system behaves, and the derivation of these mathematical representations provides the necessary insight. The models can then be used to develop realistic simulations, allowing complex control strategies to be tried in the comparative simplicity and safety of simulation. The models can also be used within *model based controllers*, where full or partial knowledge of the model is incorporated in the control design. The modelling process also forces strict definitions upon the system, this being particularly relevant to the assignment of coordinate systems along the length of the manipulator.

Firstly, the manipulator used in the experimental work throughout this thesis is introduced, with a description of its salient features. Then the kinematic model of the manipulator is detailed, this being developed for a reduced number of *degrees of freedom* (DOF). The kinematic model embraces the motion of the manipulator without any regard for the forces which cause the movement, and hence comprises solely of the geometrical representation of the robot. The *forward kinematics* gives the Cartesian position and orientation of the manipulator end-effector from the joint angles and link lengths,

transforming from *joint space* to *Cartesian space*. The reverse transformation is referred to as the *inverse kinematics*. One further important kinematic transformation is the *Jacobian* which relates Cartesian velocities to joint space velocities and also forces in Cartesian space to those in joint space.

The dynamic equations of the manipulator are then presented, these being an extension of the kinematics to include the forces that cause the motion of the robot links. This embodies the rigid body dynamics of the robot links, the dynamics of the hydraulic actuators and contact dynamics. The complete robot dynamic model is presented in a *closed form* with the hydraulic actuator model being derived such that it is applicable to a manipulator with an arbitrary number of joints.

For the purpose of simulation, the complete model is implemented for use in the MATLAB/SIMULINK package which enables controllers of varying complexity to be readily constructed and tested. Validation of this model against experimental data is presented in the later chapters of this thesis.

3.2 Experimental Manipulator

The manipulator used in this study is a right-handed Slingsby (SEL) TA9 hydraulic underwater manipulator that is located in the Ocean Systems Laboratory at Heriot-Watt University. This manipulator, shown in Figure 3.1, is in widespread use in the offshore industry of the North Sea, and is primitive when compared to the specialised laboratory robots to which advanced control strategies are usually applied. The manipulator is one of a pair mounted inside a water-tight tank in which they can operate submerged.

This six degree of freedom (DOF) robot is 1.53 m in length, has a mass of 36 kg and a rated maximum payload of 80 kg. An external hydraulic pump provides hydraulic fluid at a nominal pressure of 175×10^5 N m^{-2} (2500 psi). There are six revolute joints, four of which are actuated by linear rams acting about pivots and two by rotary hydraulic actuators.

The claw open/close is also operated by a linear ram. The flow of hydraulic fluid to these actuators is regulated by MOOG E777-006 electrohydraulic servovalves, which are described in subsequent sections. Joint angle sensing is achieved using linear potentiometers, the wiper voltage being proportional to the joint angle. Simple calibration of the robot determines the relationship between the wiper voltage and joint angle.

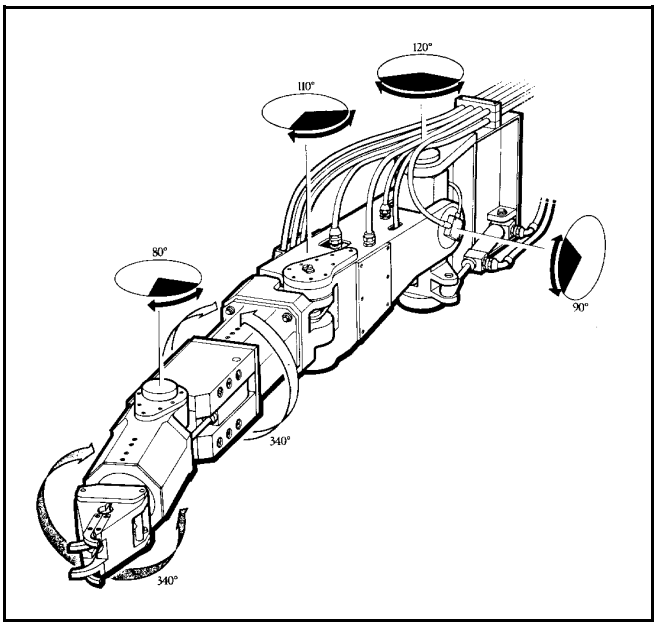

Figure 3.1 Slingsby TA9 Underwater Manipulator

The manipulator used in this study has been equipped with a six axis force/torque sensor (Assurance Technologies Inc Model 150/600), which has been marinised for use in the laboratory test tank. This sensor is mounted between the claw rotate joint and the wrist joint, 257 mm from the tip of the end-effector. The raw strain gauge signals from the sensor are converted into force/torque data by the ATI *transducer controller*, located remotely from the manipulator. This factory calibrated unit outputs the data as analogue voltages.

3.3 Kinematic Manipulator Model

For the purposes of this study it was decided to limit the manipulator used in the experiments and simulations to two DOF, namely the shoulder slew and elbow joints. This enabled the experimentation to proceed without being impeded by the complications of a

full six DOF system, requiring determination of the complete kinematics and dynamics. The use of the shoulder and elbow joints still provided a large workspace for the Cartesian position and force control investigation. The control schemes and conclusions developed here for the restricted manipulator can be extended to the full TA9, and this has been borne in mind throughout the work.

The restricted manipulator is confined to operation in the horizontal plane, providing the unused joints are frozen[†]. A plan view of the manipulator is given in Figure 3.2. The wrist joint introduces a third, albeit fixed, joint angle into the kinematic analysis since it cannot be set to 0°, due to mechanical limits of the TA9.

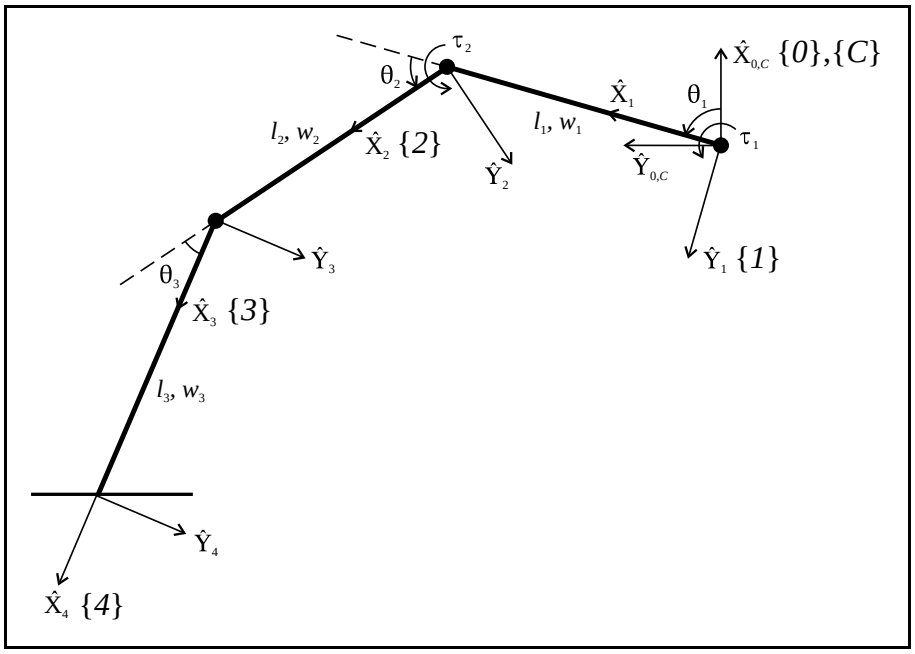

Figure 3.2 Plan View of Restricted TA9 Configuration

To describe the location of the links of the manipulator, coordinate frames $\{i\}$ are attached to each successive link i, these frames being assigned using the modified Denavit-Hartenberg notation [3.1, 2.1]. Essentially, frame $\{i\}$ has its origin coincident with the particular joint, with the $\hat{Z}_i$ axis aligned with the axis of the joint (out of the paper in the

[†] The four joints not used are set to fixed values using conventional high gain independent joint angle controllers, thereby not playing a part in the control schemes. The shoulder up/joint is set to 5°, the elbow to 175°, the wrist to 14.48° (its mechanical limit), the claw rotate to 0°. These joint angles allow the restricted TA9 to operate in the horizontal plane.

case of Figure 3.2). The $\hat{X}_i$ axis is in the direction of the next joint and $\hat{Y}_i$ is formed by the right-hand rule to complete the frame $\{i\}$. The *constraint frame*, $\{C\}$, is the coordinate system in which Cartesian positions and forces are specified, here it is coincident with the frame $\{0\}$.

The link lengths and widths of the manipulator were measured and found to be l_1 = 0.452 m, l_2 = 0.522 m, l_3 = 0.558 m, w_1 = 0.17 m, w_2 = 0.14 m and w_3 = 0.12 m. The limits of the joint angles, as defined on Figure 3.2, were measured and found to be :-

shoulder slew, θ_1 = 66.95° to 182.45°

elbow pivot, θ_2 = 0.57° to 102.32°

wrist pivot, θ_3 = 14.48° (fixed)

These link lengths and joint angles were measured on the experimental arm and confirmed using the assembly drawings of the robot. It should be noted that these joint angle ranges differ from the nominal ranges provided by the manufacturer, as given in Figure 3.1. The values were validated by commanding the robot to move in a square under Cartesian control and observing the relative position of the corners, which matched closely with the distances commanded.

The forward kinematics of a manipulator are derived using the homogeneous transformations relating successive frames along the length of the robot. This procedure is detailed in [2.1] and yields the following expression for this particular arrangement :-

$$ {}^{C}X = \begin{bmatrix} l_1c_1 + l_2c_{12} + l_3c_{123} \\ l_1s_1 + l_2s_{12} + l_3s_{123} \end{bmatrix} \tag{3.1} $$

where $c_1 = \cos(\theta_1)$, $s_1 = \sin(\theta_1)$, $c_{12} = \cos(\theta_1+\theta_2)$ etc. The forward kinematics of Equation 3.1 specifies the unique mapping from joint space, $\Theta = [\theta_1\ \theta_2]^T$, to Cartesian space, giving

the end-effector position specified in frame $\{C\}$, ${}^{C}X = [{}^{C}x\ {}^{C}y]^{T}$. The inverse kinematics, providing the reverse mapping, is often problematic to solve as no, multiple or even infinite solutions may exist [1.8]. This is not addressed in this thesis.

The Jacobian of a manipulator is similarly derived from the homogeneous frame transformations, again detailed in [2.1], and for this particular manipulator is given by Equation 3.2.

$$ {}^{C}J = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} - l_3 s_{123} & -l_2 s_{12} - l_3 s_{123} \\ l_1 c_1 + l_2 c_{12} + l_3 c_{123} & l_2 c_{12} + l_3 c_{123} \end{bmatrix} \tag{3.2} $$

By definition, the Jacobian transforms the joint space angular velocities to the Cartesian velocity of the manipulator end-effector, as shown by Equation 3.3. It is also used to relate the torques at the joints, $\mathbf{T} = [\tau_1\ \tau_2]^{T}$, to forces that are acting upon the tip of the robot specified in frame $\{C\}$, ${}^{C}F = [{}^{C}F_x\ {}^{C}F_y]^{T}$, as defined by Equation 3.4.

$$ {}^{C}\dot{X} = {}^{C}J\,\dot{\Theta} \tag{3.3} $$

$$ \mathbf{T} = {}^{C}J^{T}\,{}^{C}F \tag{3.4} $$

There are also kinematic properties associated with the actuator, but these are inextricably linked to the actuator dynamics, which will be covered in the subsequent sections.

3.4 Dynamic Manipulator Model

It is more usual for the dynamic model of a robot to consist only of the rigid body dynamics of the links. Here, however, the dynamics of the hydraulic actuators used on the TA9 are also considered and augment the standard rigid body dynamics, resulting in a highly nonlinear system. The hydraulic system can be separated into two parts; an

electrohydraulic servovalve which regulates the flow of hydraulic fluid, and an actuator that generates movement at the joint. The dynamics of these hydraulic components are first formulated and their static characteristics are examined, enabling some of the nonlinearities to be visualised. This actuator model is then embodied in the traditional rigid body manipulator model. The overall model is derived such that it is applicable to robots with an arbitrary number of hydraulically actuated joints.

The values of the parameters used to model the restricted TA9 manipulator in the following sections are given in Appendix A.2.

3.4.1 Modelling of Electrohydraulic Servovalves

Electrohydraulic servovalves are used to regulate the flow of hydraulic fluid, enabling low power electrical signals (less than 1 W) to precisely control high power hydraulic components (which may be rated up to 7000 W). In robotic systems they regulate the fluid flow into either rotary or linear hydraulic actuators, which provide motion at the joints of the manipulator.

There are many different types of servovalve [3.2], but only *two-stage four-way* servovalves, which are specified on the TA9 manipulator, will be considered here. These regulate the flow of hydraulic fluid using the displacement of a central spool, which partially opens each port through which fluid can flow, as indicated in Figure 3.3. The type of valve shown is termed a four-way servovalve since it has four hydraulic connections; one for the supply pressure, P_s, another for the exhaust, P_e, and two regulated control ports, one feeding the hydraulic system, P_1, and the other being the corresponding return, P_2.

Motion of the spool is provided by means of a *torque motor* which converts an electrical input into a small angular deflection. This deflection drives a *nozzle-flapper* valve (not shown on Figure 3.3), which in turn actuates the central spool of the main four-way valve. Thus these servovalves are termed two-stage and this results in both high precision

and high gain. Generally, the torque motors are actuated by two coils and when driven by a suitable current source, the time constant of the circuit is insignificant when compared to the rest of the system.

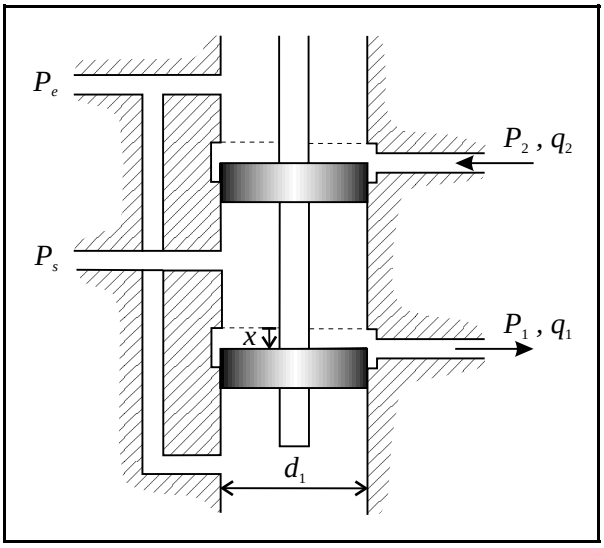

Figure 3.3 Servovalve Schematic Diagram

The general dynamic analysis of this type of servovalve yields a sixth-order expression [3.3]. However, the following first-order transfer function can be used with little loss of accuracy for frequencies up to 200 Hz [3.4] :-

$$\frac{x(s)}{i(s)} = \frac{K_i}{\tau_i s + 1} \tag{3.5}$$

where x is the spool displacement (m) and i is the coil input current (mA). Further, for static or low frequency analysis where the dynamics of the servovalve can be completely ignored, and the gain of the servovalve spool is simply K_i.

When the spool, of internal diameter d_1, is displaced by a distance x, two annular orifices are formed, each of diameter d_1 and width x, giving an orifice area of $\pi d_1 x$. The assumption that the valve openings can be treated as orifices is valid since x is small compared to d_1, which has the implication that the power available in the pressure drop across the orifice is converted entirely to kinetic energy of the fluid. Therefore, using

standard fluid dynamics analysis [3.2] the expression for volumetric flowrate through these orifices, q_1 and q_2, can be found to be :-

$$q_1 = q_2 = \frac{C_d \pi d_1}{\sqrt{\rho}} x \sqrt{P_s - \mathrm{sgn}(x) P_m} \tag{3.6}$$

where ρ is the density of the hydraulic fluid assumed constant and sgn(x) is the signum function[‡]. The constant C_d accounts for the fact that the jet formed from the flow through the orifice is smaller than the actual orifice, due to turbulent flow. Typically this has a value in the range of 0.5 to 0.6, here it is taken to be 0.5 so as to match the rated flowrate of these specific servovalves. The servovalve is assumed symmetrical, which allows the pressure drop across each orifice to be equated, that is, P_s - P_1 = P_2 - P_e. Also, the exhaust pressure is assumed negligible (i.e. P_e = 0) and the load pressure is defined as P_m = P_1 - P_2. Using this definition of P_m and the assumptions of servovalve symmetry and negligible exhaust pressure, it can be shown that :-

$$\begin{aligned} \dot{P}_1 &= \tfrac{1}{2} \dot{P}_m \\ \dot{P}_2 &= -\tfrac{1}{2} \dot{P}_m \end{aligned} \tag{3.7}$$

Other factors that introduce further nonlinear variations in flowrate are temperature, degree of lap [3.2], internal leakage, mechanical resolution, hysteresis and drifting of the input signal. These characteristics are not considered in the development of this model.

3.4.2 Linear Hydraulic Actuators

There are two broad categories of actuator used in hydraulic robots, linear actuators and rotary actuators. Only linear actuators are investigated here, since similar analysis can

---

‡

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

be applied to rotational actuators in complete analogy with that presented here. A schematic diagram of a linear hydraulic actuator is shown in Figure 3.4.

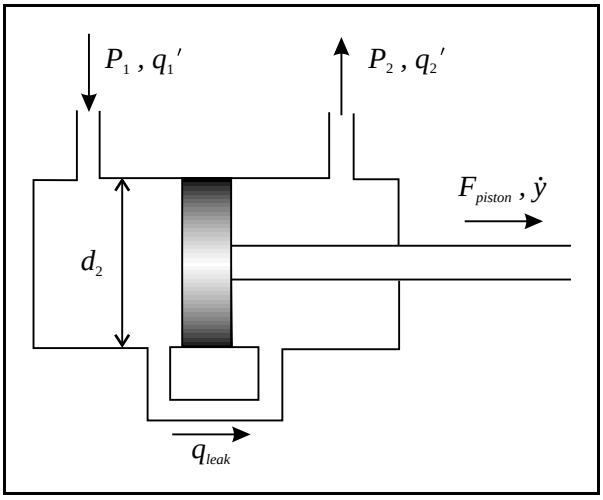

Figure 3.4 Linear Hydraulic Actuator

As fluid enters the left-hand side of the ram, the pressure across the piston increases, which gives rise to a force, F_{piston}, which in turn produces linear motion of the ram, $\dot{y}$. This motion causes fluid to leave the right-hand chamber until equilibrium is reached. The expression for the force, F_{piston}, is simply :-

$$F_{piston} = A_2 P_m \tag{3.8}$$

where A_2 is the piston area. Often, the cross-sectional area of the connecting rod inside the cylinder is significant, and then the piston area should be taken to be the average area on both sides of the ram.

A small leakage flow across the piston, q_{leak}, may exist and is shown as the separate flow path in Figure 3.4. This is often deliberately introduced by designers to improve overall system stability. This flow is taken to be proportional to the pressure drop across the piston :-

$$q_{leak} = k_{leak} P_m \tag{3.9}$$

Since there is no external leakage, the flowrates entering and leaving the ram are equal, and can be equated to the volume displaced by the piston motion and leakage across it, thus :-

$$q_1' = q_2' = A_2\dot{y} + q_{leak} \tag{3.10}$$

These flows are regulated by a servovalve, as described in the preceding section, therefore the electrical input signal to the servovalve governs the motion of the piston. Often, long hoses connect the actuator to the servovalve and as the hydraulic fluid flows between them, the flowrate is diminished due to compression of the fluid. This reduction is proportional to both the volume into which the fluid is flowing and the rate of change of pressure which gives [3.2, 3.4] :-

$$\begin{aligned} q_1' &= q_1 - \frac{V_1}{\beta}\dot{P}_1 \\ q_2 &= q_2' - \frac{V_2}{\beta}\dot{P}_2 \end{aligned} \tag{3.11}$$

where V_1 and V_2 are the volumes of fluid on the respective sides of the piston, including that in the connecting hoses. The *bulk modulus*, β_0, of a fluid is a measure of its compressibility, and for a typical hydraulic fluid is taken to be 17×10^8 N m^{-2}. The *effective bulk modulus*, β, used in Equation 3.11, includes additional effects such as hose wall flexibility, the presence of air entrapped in the fluid and changes in fluid temperature, and is consequently lower than β_0.

Dilation of the walls of the enclosing vessel can reduce the bulk modulus by 20% for thin walled aluminium cylinders, and can be up to 50% for flexible hoses, which are often used in robotic applications. The presence of air has a more dramatic effect upon the bulk modulus since gases are of the order of 100 times more compressible than fluids. Typically, if the fluid contains 0.1% air by volume, then the bulk modulus is reduced by

10%, with 1% air entrapped this increases to a 50% reduction. The reduction with fluid temperature is shown in Table 3.1 [3.5] :-

Temperature (°C)	Bulk Modulus (β, $\times 10^8$ N m^{-2})
-50	22.0
0	19.0
50	15.5
100	12.5
150	9.5

Table 3.1 Variation of Bulk Modulus with Fluid Temperature

Since all of these effects manifest themselves as changes in the effective bulk modulus, it is easy to investigate how performance is degraded under such extreme operating conditions.

Equations 3.7 and 3.9 to 3.11 can then be combined to give :-

$$q_1 = q_2 = A_2\dot{y} + k_{leak}P_m + \frac{V_t}{4\beta}\dot{P}_m \tag{3.12}$$

where V_t is the total volume of fluid in the piston and connecting hoses, and is used to represent the average values of V_1 and V_2. This expression illustrates that the flow leaving the servovalve forms three components; the flow which provides motion of the ram, the leakage flow and the flow due to fluid compressibility. Equating this to Equation 3.6, the expression for flowrate through the servovalve, and rearranging yields :-

$$\dot{P}_m = \frac{4\beta}{V_t}\left[\frac{C_d \pi d_1}{\sqrt{\rho}}\, x\sqrt{P_s - \mathrm{sgn}(x)P_m} - A_2\dot{y} - k_{leak}P_m\right] \tag{3.13}$$

3.4.3 Linear Hydraulic Actuators Acting About a Pivot

Linear hydraulic actuators used in robotic applications can drive either prismatic joints (linear motion) or revolute joints (rotation about a pivot) as used in the Slingsby TA9 manipulator. The subsequent analysis will concern linear rams actuating revolute joints, as shown in Figure 3.5, since the prismatic joint case is a subset of that presented.

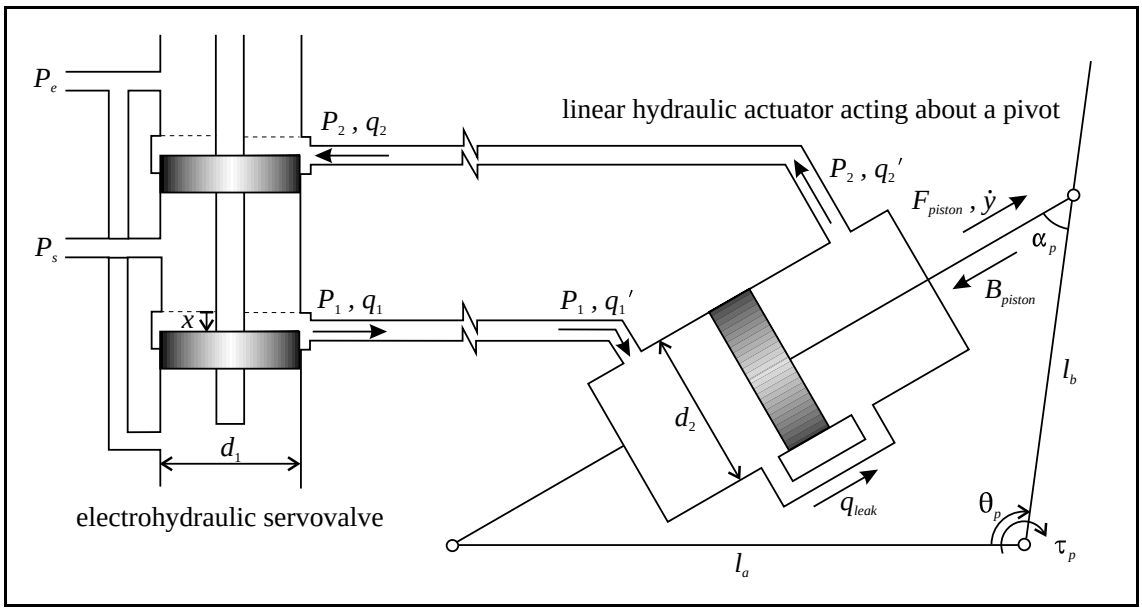

Figure 3.5 Linear Hydraulic Actuator acting about a Pivot

Taking the overall piston length as *y*, the following expressions can be derived by applying the cosine and sine rules respectively to the geometry of the actuator shown in Figure 3.5 :-

$$y = \sqrt{l_a^2 + l_b^2 - 2 l_a l_b \cos(\theta_p)} \tag{3.14}$$

$$y = l_a \frac{\sin(\theta_p)}{\sin(\alpha_p)} \tag{3.15}$$

An expression for $\dot{y}$ can then be derived by differentiating Equation 3.14 with respect to time, giving :-

$$\dot{y} = l_a l_b \sin(\theta_p) \sqrt{l_a^2 + l_b^2 - 2 l_a l_b \cos(\theta_p)}\ \dot{\theta}_p \tag{3.16}$$

This expression relates the piston velocity to the piston angle and angular velocity, for use in Equation 3.13. The torque about the joint, τ_p, is related to the force produced by the piston and the friction present in the ram, B_{piston}, by :-

$$\tau_p = (F_{piston} - B_{piston}) \sin(\alpha_p) l_b \tag{3.17}$$

which, using Equations 3.8, 3.14 and 3.15, can be re-written as :-

$$\tau_p = (A_2 P_m - B_{piston}) J_p(\theta_p) \tag{3.18}$$

where :-

$$J_p(\theta_p) = l_a l_b \sin(\theta_p) \sqrt{l_a^2 + l_b^2 - 2 l_a l_b \cos(\theta_p)} \tag{3.19}$$

The manipulator joint angle, θ, as defined according to the modified Denavit-Hartenberg conventions, see Figure 3.2, may not coincide with the angle made by the piston and pivots, θ_p. Also, due to the physical arrangement of the actuator, the joint angle may be in the opposite sense to θ_p, therefore Equation 3.20 is used to transform from one angle to the other :-

$$\theta = k_{\theta dir} \theta_p + \theta_{poffset} \tag{3.20}$$

where $k_{\theta dir}$ is defined to be either +1 or -1 depending upon the relative sense of the angles, and $\theta_{poffset}$ is determined from the physical configuration of the actuator and link to align the two angles. Since the joint torque is defined in the same direction as the joint angle, the following relation also applies :-

$$\tau = k_{\theta dir} \tau_p \tag{3.21}$$

and therefore :-

$$\tau = k_{\theta dir}(A_2 P_m - B_{piston}) J_p(\theta_p) \tag{3.22}$$

The friction present in the ram, B_{piston}, is modelled here having both viscous and Coulomb friction components. The viscous friction is proportional to the velocity of the ram, and Coulomb friction is constant except for a sign dependency on the ram velocity [3.6]. Therefore :-

$$B_{piston} = c_p \operatorname{sgn}(\dot{y}) + v_p \dot{y} \tag{3.23}$$

Friction is also present in the joint itself and is again composed of both viscous and Coulomb friction. Therefore :-

$$B_{joint} = c_j \operatorname{sgn}(\dot{\theta}) + v_j \dot{\theta} \tag{3.24}$$

This joint friction is embodied in the closed form dynamic model of the complete robot, as described later in Section 3.4.5. For certain actuators on the TA9, a positive input signal produces motion in the joint in the opposite sense to that indicated on Figure 3.5, this being the result of polarity changes in the servovalve solenoid, spool and/or connecting hoses. The model can be generalised to accommodate this, by setting the sign of K_i in Equation 3.5 appropriately.

3.4.4 Static Characteristics of Linear Hydraulic Actuators

To visualise the nonlinear expressions that model a hydraulic actuator, the static characteristics shall now be investigated. During steady state conditions, a small flow through the servovalve is maintained due to the leakage flow across the piston ram. Setting the derivative and velocity dependent terms, including friction, in Equations 3.5, 3.13 and 3.18 to zero gives :-

$$\frac{C_d \pi d_1}{\sqrt{\rho}} K_i i \sqrt{P_s - \mathrm{sgn}(K_i i) P_m} - k_{leak} P_m = 0 \tag{3.25}$$

$$\tau_p = A_2 P_m J_p(\theta_p) \tag{3.26}$$

These relationships give the nonlinear dependency that the torque about the joint has upon both the input current, i, and the piston angle, θ_p, under static conditions. Only positive values of i will be considered here, as the results are symmetrical about i = 0 mA. These expressions are plotted in Figure 3.6a, with the piston angle being transformed to the joint angle using Equation 3.20 for the purpose of comparison. These particular characteristics are of the shoulder slew actuator of the TA9 and clearly illustrate the nonlinearities present in these types of actuator. The other actuators of the TA9 have similar characteristics. Figure 3.6b shows the equivalent characteristics of the often assumed ideal torque source, which produces a torque proportional to the input signal and is independent of the configuration of the manipulator.

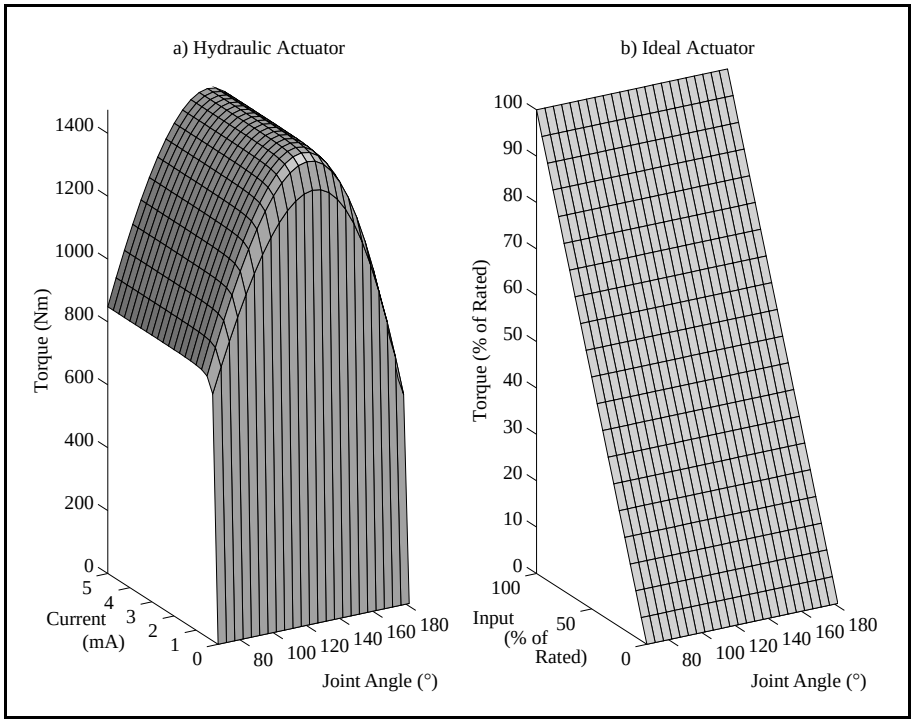

Figure 3.6 Static Characteristics for Hydraulic and Ideal Actuators

In many robot force control strategies the Jacobian is used to transform force errors in Cartesian space to joint torque errors, Equation 3.4. Since the control signal for the TA9 manipulator is servovalve current, the controllers used must be able to accommodate the nonlinearities shown in Figure 3.6a. It must be stressed that these characteristics do not include any dynamic effects which will introduce further nonlinearities. They do clearly demonstrate the need for the nonlinear model, to enable realistic simulations to be produced.

3.4.5 Hydraulically Actuated Robot Model

A generalised direct drive manipulator with an arbitrary number of joints is conveniently represented by the following *closed form* dynamic model [2.1] :-

$$\mathbf{T} = M(\Theta)\ddot{\Theta} + V(\Theta,\dot{\Theta}) + G(\Theta) + B(\dot{\Theta}) + {}^{C}J^{T}(\Theta)\,{}^{C}F \tag{3.27}$$

where $\mathbf{T}$ is the vector of joint torques, Θ is the vector of joint angles, M is the inertia matrix of the manipulator links, V is the Coriolis and centrifugal effects matrix, G is the gravity matrix, B is the vector of friction acting at each joint, ${}^{C}J$ is the Jacobian of the manipulator and ${}^{C}F$ is the vector of forces and torques at the end-effector. The joint torques, $\mathbf{T}$, and joint friction, B, for the hydraulically actuated manipulator are simply obtained by stacking Equations 3.22 and 3.24 respectively for each joint of robot. This enures that the hydraulically actuator model can be applied to manipulators with an arbitrary number of joints.

The closed form model matrices for the restricted TA9 are given in Appendix A.4, these being derived using the conventional recursive Newton-Euler dynamic equations. Often these are formulated with the mass of the link being concentrated at a single point at the distal end. However, here each link was assumed to be a homogeneous rectangular mass, with its centre of mass being halfway along its principal axes.

Rearranging Equation 3.27 as :-

$$\ddot{\Theta} = M(\Theta)^{-1}[\mathbf{T} - V(\Theta,\dot{\Theta}) - G(\Theta) - B(\dot{\Theta}) - {}^{C}J^{T}(\Theta)\ {}^{C}F] \tag{3.28}$$

enables the motion of the manipulator to be simulated by simply integrating Equation 3.28 twice with respect to time. Similarly, the load pressure, P_m, of each actuator required to determine T, is obtained by integrating Equation 3.13 for each joint of the robot.

Hydrodynamic effects associated with manipulator motion in water, such as drag and buoyancy, are complicated to determine and are only significant when the robot is moving at high speed. It has been shown experimentally that operation in water or in air has little consequence to typical motions [1.2]. These hydrodynamic effects are therefore left unmodelled.

3.5 Environment Model

When the manipulator end-effector is in contact with a surface or object it is modelled either as a *soft contact* or a *hard contact*, often termed *compliant motion* and *constrained motion* respectively. A soft contact involves translational motion into the surface, with the force being proportional to the displacement into the surface, ${}^{C}\Delta X$. The constant of proportionality is termed the *environmental stiffness*, K_E, and effectively models the contact as a spring, thus :-

$${}^{C}F = K_E\,{}^{C}\Delta X \tag{3.29}$$

Other soft contact models that can be used have the behaviour of dampers and/or inertias, however the spring model is the most prevalent in robotics research due to its simplicity [2.3]. Hard contacts, on the other hand, involve strict mathematical constraints which must be satisfied, with the end-effector and environmental surface maintaining their shapes regardless of the magnitude of force exerted. Whilst contact is maintained, this

constraint imposes limits on certain joint variables of the manipulator. To model this behaviour it is required to reform the robot model with a reduced number of independent joint variables. This is problematic and beyond the scope of this thesis.

Sliding motions across the constraint surface are assumed frictionless, this situation is approached experimentally with the use of a ball transfer unit tool to minimise the sliding friction.

The transition from free space motion to constrained motion involves the study of rigid body impacts. This is beyond the scope of this work and consequently the simulations are initialised with the robot in contact with the environment. In the case of practical experiments any such transients are allowed to decay and so do not effect the results.

3.6 Hydraulic Manipulator Model Realisation

The actuator model, consisting of Equations 3.5, 3.13, 3.16, 3.19, 3.22 and 3.23, can be conveniently written in a few lines of MATLAB[††] code, as shown below. This code can be directly applied to manipulators with an arbitrary number of joints, due to MATLAB's handling of vector calculations. The model may also be constructed using the graphical user interface that SIMULINK provides, as shown in Figure 3.7.

Either form of the hydraulic actuator model can be easily incorporated into the overall manipulator dynamic model for subsequent simulation. The simulation results

```
% hydraulic actuator dynamics
jp = la.*lb.*sin(theta-theta_poffset)./ sqrt(la.*la+lb.*lb-2*la.*lb.*cos(theta-theta_poffset));
y_dot = jp.*theta_dot;
pm_dot = 4*beta*( kv.*x.*sqrt(ps-sign(x).*pm) - A2.*y_dot - kleak.*pm )./Vt;
x_dot = (ki.*servovalve_current - x)./taui;
piston_fric = Fram(:,1).*y_dot + Fram(:,2).*sign(y_dot);
tau = (A2.*pm - piston_fric).*jp;
```

MATLAB code for hydraulic actuator model

[††] The software versions used throughout this thesis are MATLAB version 5.3 and SIMULINK version 3.

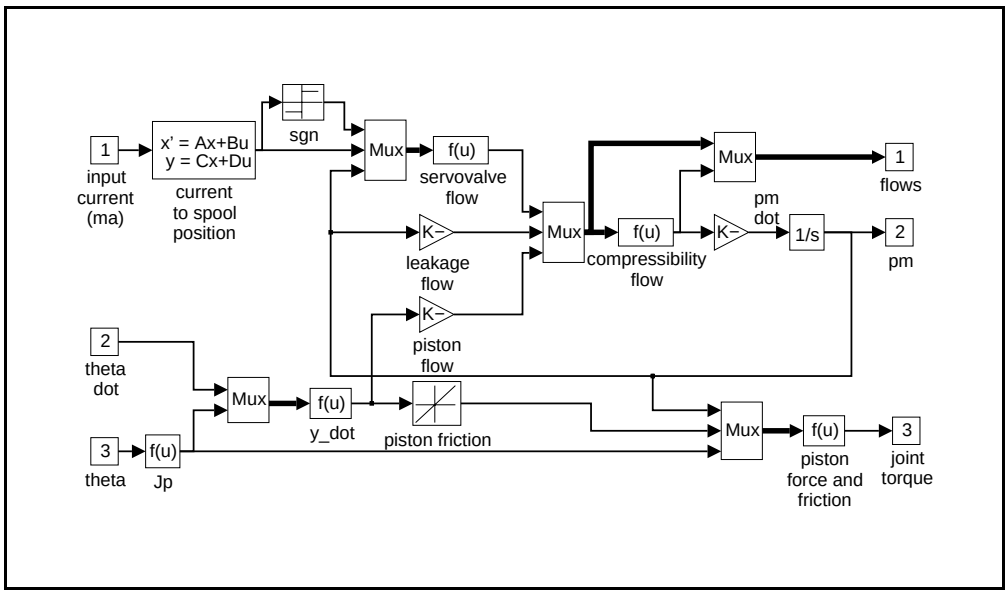

Figure 3.7 SIMULINK Graphical Model of a Hydraulic Actuator

presented in the following chapters were generated using the coded version embodied in an S-Function. This form was used since it produces a more compact, versatile and rigorous model than the graphical version, since a copy of Figure 3.7 is needed for each joint of the robot in the graphical model.

The values of the various parameters used to model the restricted TA9 manipulator are given in Appendix A.2. Furthermore, the complete set of MATLAB files that constitute the SIMULINK model, as used in the simulation work presented later, are given in Appendix A.5.

3.6.1 Model Initial Conditions

For the model to start in a state of equilibrium the initial conditions of the states, integrators and inputs need to be set correctly. The initial joint velocities, $\dot{\Theta}\big|_{t=0}$, and accelerations, $\ddot{\Theta}\big|_{t=0}$, are set to zero so that the manipulator is not moving, and the joint angles are initialised with the desired starting joint angle, $\Theta\big|_{t=0}$. The friction terms are zero since there is no motion. Therefore, from Equation 3.28, the joint torques must equal the gravitational term and the effect of any initial end-effector forces, ${}^{C}F\big|_{t=0}$:-

$$\mathbf{T}|_{t=0} = G(\Theta|_{t=0}) + {}^{C}J^{T}(\Theta|_{t=0})\ {}^{C}F|_{t=0} \tag{3.30}$$

Similarly, the load pressure at each actuator to provide this torque is derived from Equation 3.22, as :-

$$P_m|_{t=0} = \frac{\tau|_{t=0}}{k_{\theta dir} A_2 J_p(\theta_p|_{t=0})} \tag{3.31}$$

The value of P_m, required to maintain an equilibrium, is provided by an initial servovalve flow, which is derived from Equation 3.12 as :-

$$q_1|_{t=0} = q_2|_{t=0} = k_{leak} P_m|_{t=0} \tag{3.32}$$

The spool position to give this flow is obtained from Equation 3.6, and ultimately the applied current for equilibrium is derived from the servovalve spool gain, K_i.

3.6.2 Controller Model Realisation

The controllers used within the simulation work presented later were implemented using appropriate discrete time controllers, as implemented on the experimental system. The controller model was augmented with blocks to mimic the effect of sampling and quantisation on the analogue signals read in or output by the ADC and DAC boards respectively. These effects are easily modelled in SIMULINK as quantisation and zero-order-hold blocks are provided within its standard library. Reference signals within the controllers were also subject to zero-order-hold blocks to create an entirely discrete controller. Further modifications to the controller model were made by incorporating anti-aliasing filters, again to match those present in the experimental setup.

3.6.3 Model Integration Algorithm

MATLAB offers a variety of integration algorithms, each type providing different

performance in terms of speed and accuracy for different types of system being simulated. The Runge-Kutta fifth-order integration method has been found to be the optimum for this type of system, especially when a digital controller is utilised, as is the case here. This algorithm outperforms the others when the system is highly nonlinear and/or discontinuous, which is particularly true for this model with the inclusion of the contact model. This algorithm performs well for all types of continuous, discrete and/or mixed systems. It does not work well when the system has both fast and slow dynamics, however this can be alleviated with suitable minimum time step sizes and tolerances.

The Runge-Kutta algorithm takes a variable time step size which is limited to within a specified range, the step size used is chosen such that the required accuracy is maintained. For this system, suitable values of minimum and maximum step size were found to be 1×10^{-5} and 0.1 seconds respectively.

The accuracy of the simulation is controlled using the tolerance parameter which specifies the relative error of the integration at each step, a value of 1×10^{-6} was used. The smaller this parameter is, the more steps the integration method will take, resulting in a more accurate, but slower simulation.

3.7 Summary

This chapter has described the kinematic and dynamic models of the Slingsby TA9 hydraulic manipulator used throughout this thesis. This modelling has enabled the controllers presented in subsequent chapters to be tried and tested on a realistic simulation before being applied to the experimental manipulator. The work presented in this chapter also provides a thorough understanding as to how hydraulic manipulators function, and the implications that this has upon position and force control will be explored in later chapters.

The experimental manipulator, introduced in Section 3.2, was restricted to operation in two degrees of freedom for the purpose of this thesis. This enabled the implementation

to be simplified whilst maintaining the validity of the conclusions drawn from the simulations and experiments. The kinematic model, Jacobian and frame definitions (see Figure 3.2) of this restricted manipulator were presented in Section 3.3 and are used in deriving the position and force controllers.

Section 3.4 derived the nonlinear mathematical dynamic model of a hydraulically actuated robot, including models of the servovalve, fluid compressibility and the linear ram acting about a pivot to actuate a revolute joint. The static characteristics of the actuator model were visualised and compared to that of the often assumed ideal actuator in Section 3.4.3. The hydraulic actuator model was then incorporated into a generalised direct drive manipulator model, yielding the overall robot model. A model of the contact between the robot and a surface was then presented, and used with the robot model in the simulations.

The implementation of these mathematical representations of the robot and environment in the MATLAB/SIMULINK simulation package was then discussed in Section 3.5. This overall system model was used to obtain the simulation results presented in the following chapters.

Chapter 4

Self-tuning Controller Theory for Robot Applications

4.1 Introduction

Currently, the control of manipulators is primarily achieved using fixed gain schemes, this being due to their relative simplicity. For example on the Slingsby TA9, analogue proportional only controllers are used as standard. Only the simplest of the advanced controllers described in Chapter Two, are finding actual implementation. However, the advent of high power, low cost microprocessors has made these more advanced controllers feasible, with the associated advantages described in Chapter Two.

Self-tuning controllers are one particular type of advanced control scheme that are particularly suitable for robotic applications. One of the main problems faced by designers of robot controllers is the wide variations in dynamics that are encountered during typical operations. These variations arise from changes in the configuration of the robot, payload and acceleration, and are highly nonlinear as illustrated by the equations developed in Chapter Three. A self-tuning controller is able to accommodate these changes in dynamic behaviour by tracking the changes and adjusting its gains accordingly.

In this chapter the basic self-tuning pole placement controller is reviewed, first for a single-input single-output (SISO) system, then for the multi-input multi-output (MIMO) case. The SISO controller is applicable to independent joint control, whereas the MIMO controller is suitable for the multivariable problems of Cartesian and hybrid control. The application of these controllers to robot control is discussed, and a multivariable self-tuning

pole placement controller is derived for the hybrid position/force control problem.

Emphasis is placed on the practical operating aspects of these control schemes, including numerically robust and efficient algorithms.

4.2 Self-tuning Pole Placement Controllers

The papers of Wellstead [2.61] and Åstrom [2.62] were amongst the first to exploit the idea of a self-tuning pole placement controller, though the concept of a "self-tuning" system was first proposed by Kalman in the late 1950s. The theory developed here follows these original formulations, and also includes additional concepts relevant to practical implementation.

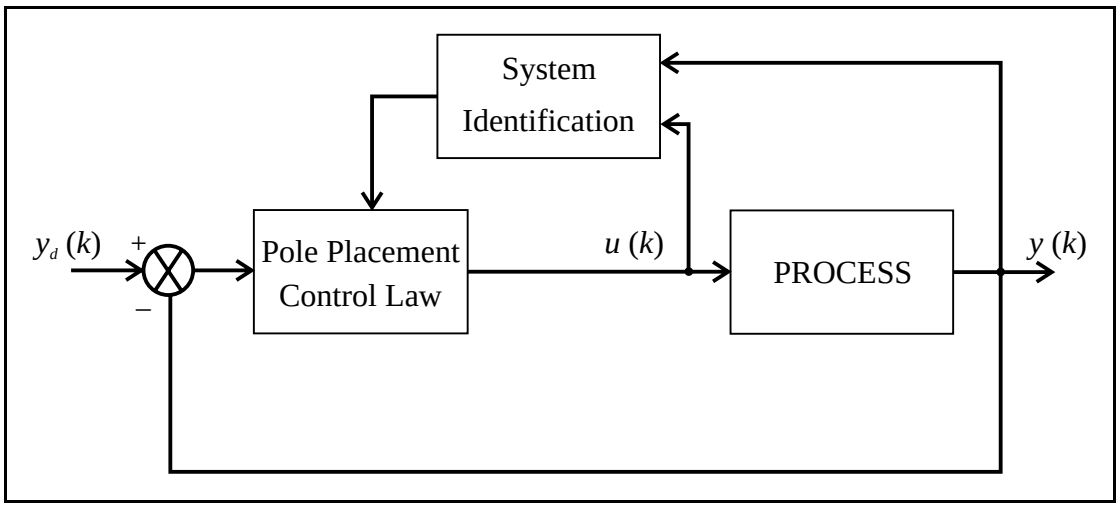

Figure 4.1 Self-tuning Controller Functional Structure

Self-tuning pole placement controllers work on the principle of continually adjusting the controller gains, such that the closed loop system poles remain constant, irrespective of changes in the system's dynamics. The controller has three components; a low-order linear model of the process under control, an on-line system identifier and a pole placement controller. Figure 4.1 shows these fundamental components.

The system identification block uses a recursive algorithm to fit past values of inputs and outputs to a low-order linear model of the process. The identification mechanism

has a *forgetting factor* to allow the linear model to adapt as the system changes with variations in operating conditions. The controller parameters are then determined using this low-order linear model so that the characteristic polynomial of the closed loop system matches some user specified polynomial.

Though the process under control is a continuous time system, the controller is discrete due to its implementation on a digital computer. Hence, the system input and output, $u(k)$ and $y(k)$ respectively, are defined as discrete time signals, where k is the sample number[†]. The desired system output, or reference signal, is denoted $y_d(k)$.

The three elements that constitute the self-tuning controller will now be discussed in detail.

4.2.1 The SISO Process Model

The process model used in self-tuning controllers is generally represented by a low-order difference equation, where the output, $y(k)$, is written in terms of past values of itself and the input, $u(k)$. Therefore, the process can be conveniently expressed as the following ARMAX (AutoRegressive Moving Average Exogenous input) model, where $e(k)$ represents the error due to modelling :-

$$\left[1 + a(z^{-1})\right] y(k) = b(z^{-1})\, u(k) + d_0 + e(k) \tag{4.1}$$

The terms $a(z^{-1})$ and $b(z^{-1})$ are polynomials of the form :-

$$\begin{aligned} a(z^{-1}) &= a_1 z^{-1} + \ldots + a_{n_a} z^{-n_a} \\ b(z^{-1}) &= b_1 z^{-1} + \ldots + b_{n_b} z^{-n_b} \end{aligned} \tag{4.2}$$

and z^{-i} is the backward shift operator, defined by :-

$$z^{-i} y(k) = y(k-i) \tag{4.3}$$

[†] The sample number takes integer values (0, 1, 2, 3, ...) such that the discrete time signal at sample k corresponds to the continuous signal at time $k\tau_s$, where τ_s is the sample period of the digital system.

and d_0 represents a constant unmeasurable drift disturbance, which in the general case is also a polynomial in z^{-1}. However, in this application only a single term is considered, $n_d = 1$, and therefore the total number of model parameters, n_Φ, is defined as :-

$$n_\Phi = n_a + n_b + n_d \tag{4.4}$$

The model represented by Equation 4.1 is a simplified version of a generalised linear process model [2.57]. Not included here are terms used to model noise sources, measurable disturbances and any higher order unmeasurable disturbances, as they were not deemed necessary for this particular application.

The choice of model order and structure of Equation 4.1 should reflect the underlying physical system, so that the modelling error is as small as possible. However, with nonlinear systems it is difficult to determine an appropriate representation. In such cases it is then prudent to use a simplified model which captures the dominant dynamics of the system. One means of empirically determining the model order is to look at the *loss function* [4.1]. Here, the modelling errors for different orders and structures are observed, and a low modelling error indicates which is most appropriate.

4.2.2 SISO System Identification

When the process under control is unknown, the parameters of the model (Equation 4.1) need to be estimated, and this is the purpose of the *on-line* system identification block in Figure 4.1. The system identifier fits the a, b and d_0 parameters using past values of the measured inputs and outputs of the process, so as to minimise a suitable error criterion.

There are several different procedures available to achieve the on-line identification, namely recursive least-squares, recursive instrumental variables, recursive maximum likelihood and extended least-squares. A survey of these methods is given in [2.56]. A

recursive least-squares (RLS) algorithm was selected for this application for the following reasons :-

- parameter estimates converge quickly, allowing fast adaptation under unknown and changing conditions.
- requires relatively small computational effort, which is crucial for real-time control of robots, which require high sample rates.
- the high signal-to-noise ratios typical of robotic applications, enable RLS to maintain high quality parameter estimates. The other methods outperform RLS when there is a low signal-to-noise ratio.

Many forms of RLS exist, and the conventional *matrix inversion lemma* (MIL-RLS) algorithm is developed initially. A numerically robust algorithm, using *Bierman U-D factorisation* (BUD-RLS), is then described. It is this latter method which is used in practice, to remove problems caused by accumulation of rounding errors over many iterations.

To formulate the RLS identification algorithm, the process model, Equation 4.1, is formulated in terms of the parameters to be estimated. A convenient expression is[†] :-

$$y(k) = \Phi^{T}(k)\,\psi(k) + e(k) \tag{4.5}$$

where $\Phi(k)$ is the $n_{\Phi}\times 1$ vector of the estimated model parameters, and $\psi(k)$ is the $n_{\Phi}\times 1$ *regression vector* which is made up of past measured inputs and outputs of the process. More specifically :-

[†] The conventional expression used for $y(k)$ when developing the RLS algorithm is $y(k) = \psi^{T}(k)\Phi(k)+e(k)$. The alternative equation, presented above, is identical for a SISO process and also extends to the MIMO case, whereas the conventional expression does not. This will be discussed further in the subsequent sections on MIMO self-tuning.

$$\Phi^{\mathrm{T}}(k) = \left[-a_1, \ldots, -a_{n_a}, b_1, \ldots, b_{n_b}, d_0 \right] \tag{4.6}$$

$$\psi^{\mathrm{T}}(k) = \left[y(k-1), \ldots, y(k-n_a), u(k-1), \ldots, u(k-n_b), 1 \right] \tag{4.7}$$

The standard MIL-RLS algorithm provides a means of iteratively updating the estimated parameters, $\Phi(k)$, at each sample instant, using the past input and output data contained within the regression vector, $\psi(k)$. Furthermore, if any of the model parameters are fixed to zero, then $\Phi(k)$ and $\psi(k)$ can be re-defined accordingly to reduce the number of estimated parameters, n_Φ, and hence overall computational burden. The MIL-RLS algorithm works by minimising the sum of the squares of the modelling errors, and its derivation is given in [2.57], the result of which can be summarised as follows :-

At sample instant *k*,

Step 1: Read system output, $y(k)$, and form $\psi(k)$ using past input and output values.

Step 2: Calculate the *a priori prediction error*, $\epsilon(k)$, as the difference between the actual output and the predicted model output, using parameter estimates from the previous sample, $\Phi(k\text{-}1)$. Therefore :-

$$\epsilon(k) = y(k) - \Phi^{\mathrm{T}}(k-1)\,\psi(k) \tag{4.8}$$

Step 3: Form $L(k)$, the *Kalman gain vector*, as :-

$$L(k) = \frac{P(k-1)\,\psi(k)}{\left[\lambda + \psi^{\mathrm{T}}(k) P(k-1)\,\psi(k) \right]} \tag{4.9}$$

and, calculate the symmetric *covariance matrix*, $P(k)$, using :-

$$P(k) = \lambda^{-1}\left[\mathrm{I} - L(k)\,\psi^{\mathrm{T}}(k) \right] P(k-1) \tag{4.10}$$

Step 4: Update the parameter estimates, $\Phi(k)$:-

$$\Phi(k) = \Phi(k-1) + L(k)\epsilon(k) \tag{4.11}$$

Step 5: Wait for next sample, $k \leftarrow k+1$, and then loop back to Step 1.

The term λ introduced in the above algorithm is called the *forgetting factor*, and is defined as a number between 0 and 1. This acts as a bias, effectively reducing the influence of older data which may no longer be relevant to the model. This allows the parameter estimates to track changes in the process, allowing time varying and nonlinear systems to be identified.

Practical values of λ are usually in the range 0.95 to 1, and the smaller the value used, the faster the estimates converge. However, decreasing λ increases the sensitivity of the estimation procedure to noise.

A further problem associated with the use of forgetting factors, is that it opposes the tendency of the RLS algorithm to decrease the covariance matrix, $P(k)$, as the estimates become more accurate. If little or no new information is brought into the estimator, then Equation 4.10 reduces to [4.2] :-

$$P(k) \simeq \lambda^{-1} P(k-1) \tag{4.12}$$

and hence the inclusion of a forgetting factor acts to increase $P(k)$. If this occurs over a long period then the covariance matrix can become large, causing loss of parameter identifiability and destabilising the estimation process [2.57]. This process is called *covariance blow-up* or *estimator wind-up*. There are several approaches to prevent this occurring and is discussed in Section 4.2.4 in the context of operational issues.

The standard MIL-RLS algorithm, Equations 4.8 to 4.11, is not suitable for all situations, such as cases which require numerical robustness, or where computational power is limited. Many alternative algorithms exist, and are often used in preference to the

standard MIL-RLS.

Rounding errors, due to finite computer word length, can accumulate over long periods of estimator operation. These errors affect the accuracy of the estimates, and more importantly can cause the covariance matrix to become negative semi-definite, causing divergence of parameter estimates. It is therefore good engineering practice to use a numerically robust RLS for any practical implementation. One widely used numerically robust approach is called the *Bierman U-D factorisation* (BUD-RLS) algorithm [4.3], and is detailed in Appendix B. This uses the symmetric property of the covariance matrix to allow the factorisation of $P(k)$ as :-

$$P(k) = U(k)D(k)U^{T}(k) \tag{4.13}$$

where $U(k)$ is an upper triangular matrix and $D(k)$ is a diagonal matrix. This is equivalent to calculating $P(k)$ at twice the precision of the standard RLS algorithm, and ensures that the covariance matrix remains positive definite [2.82].

Numerically efficient RLS algorithms have also been developed and are especially important where large numbers of parameters are being estimated, however these generally sacrifice numerical accuracy or stability for speed. It has been proposed [2.79] that computational requirements can be reduced by calculating only the diagonal elements of $P(k)$ in Equation 4.10. Another approach would be to update the covariance matrix less frequently, whilst maintaining the update rate of the rest of the identification process [2.63]. The symmetry of the covariance matrix can also be exploited to increase speed of computation.

4.2.3 SISO Pole Placement Controller

With the system under control represented by a low-order linear equation, the controller can be designed using the model parameters, to meet a specified closed loop

performance criterion. This results in a closed loop system which is unaffected by changes in the underlying process, since the controller gains are automatically adjusted to maintain the performance criterion. With a self-tuning pole placement controller the user specified criterion is the *desired characteristic polynomial* which locates the closed loop system poles, and hence dictates the response of the system.

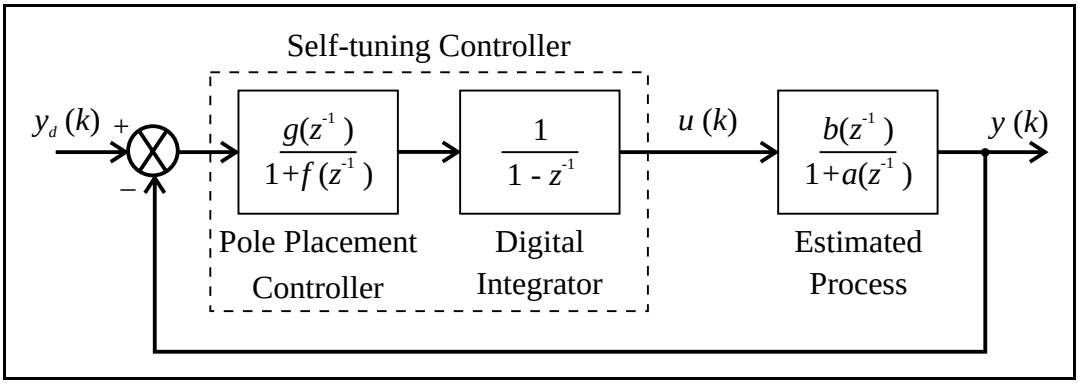

Figure 4.2 Incremental Self-tuning Controller Structure

Using the process model, Equation 4.1, the system block diagram can be redrawn to include the controller, as shown in Figure 4.2. The controllers developed in this thesis are used for reference tracking, rather than the problem of regulation, and hence incorporate a digital integrator. This type of controller is termed an *incremental controller*, since it generates the *change* in control signal and the *digital integrator* forms the actual control signal. Incremental controllers are widely used in industrial applications as they ensure zero steady-state error in the presence of offsets and unmodelled disturbances, and allow bumpless transfer between different controllers [2.57].

The controller polynomials, $f(z^{-1})$ and $g(z^{-1})$, are defined as :-

$$\begin{aligned} f(z^{-1}) &= f_1 z^{-1} + \ldots + f_{n_f} z^{-n_f} \\ g(z^{-1}) &= g_0 + g_1 z^{-1} + \ldots + g_{n_g} z^{-n_g} \end{aligned} \tag{4.14}$$

and the closed loop transfer function of the controlled system shown in Figure 4.2 is :-

$$y(k) = \frac{b(z^{-1})g(z^{-1})}{(1+a(z^{-1}))(1-z^{-1})(1+f(z^{-1})) + b(z^{-1})g(z^{-1})} y_d(k) \tag{4.15}$$

The desired closed loop system pole locations are defined by the polynomial $t(z^{-1})$. The controller polynomials, $f(z^{-1})$ and $g(z^{-1})$, that realise these desired poles are obtained by equating the denominator of Equation 4.15, the *characteristic equation*, to $t(z^{-1})$:-

$$t(z^{-1}) = (1 + a(z^{-1}))(1 - z^{-1})(1 + f(z^{-1})) + b(z^{-1})g(z^{-1}) \tag{4.16}$$

where $t(z^{-1})$ is defined as :-

$$t(z^{-1}) = 1 + t_1 z^{-1} + \ldots + t_{n_t} z^{-n_t} \tag{4.17}$$

Equation 4.16 is solved for the controller polynomials by equating like powers of z, resulting in a set of simultaneous linear equations which yields expressions for f and g in terms of a, b and t. For a unique solution of Equation 4.16 to exist, the orders of the f, g and t polynomials must meet the following constraints :-

$$\begin{aligned} n_f &= n_b - 1 \\ n_g &= n_a \\ n_t &\leq n_a + n_b \end{aligned} \tag{4.18}$$

For the majority of applications first and second order $t(z^{-1})$ polynomials are sufficient to provide the desired system response.

The set of simultaneous equations can be represented as a matrix expression, though for the general case this representation becomes inconvenient. Explicit solutions of the controller polynomials for the different model orders used throughout this thesis are presented in Appendix C. These solutions include a controller for use with a process model order of $n_a = 2$, $n_b = 1$, which corresponds directly to a self-tuning PID.

The controller output, $u(k)$, is calculated at each sample instant from the controller polynomials and the digital integrator, according to the structure given in Figure 4.2. Since the polynomials $f(z^{-1})$ and $g(z^{-1})$ are dependent upon the parameter estimates, $\Phi(k)$, this calculation is performed after Step 4 in the RLS algorithm given in the previous section.

The inclusion of the disturbance parameter, d_0, in the process model does not alter the equations that define the incremental controller. With such a term included, the overall closed loop equation (Equation 4.15) becomes :-

$$y(k) = \frac{b(z^{-1})g(z^{-1})}{t(z^{-1})} y_d(k) + \frac{(1-z^{-1})(1+f(z^{-1}))}{t(z^{-1})} d_0 \qquad (4.19)$$

The second term on the right hand side of Equation 4.19 governs the system response to any disturbance, and this has poles given by $t(z^{-1})$ as required. Further, the effect of the $(1-z^{-1})$ term, introduced by the digital integrator, is to exactly cancel any disturbances at low frequencies, since $z = 1$ in the steady state.

Without integral action, a non-zero controller output is required to compensate for the disturbance, d_0. This also enables the controller to reject time varying disturbances, whereas other techniques often rely on the disturbance being constant [2.57].

The closed loop transient response is dominated by the poles of the system, specified by $t(z^{-1})$, however the system zeros, given by $b(z^{-1})g(z^{-1})$ in Equations 4.15 and 4.19, also contribute. Direct manipulation of the system zeros is possible [2.62], however this is complicated and results in an unstable system when used with a nonminimum phase process [2.78]. If the transient response is unacceptable due to the effect of the system zeros, one possible solution is to use the modified control law shown in Figure 4.3.

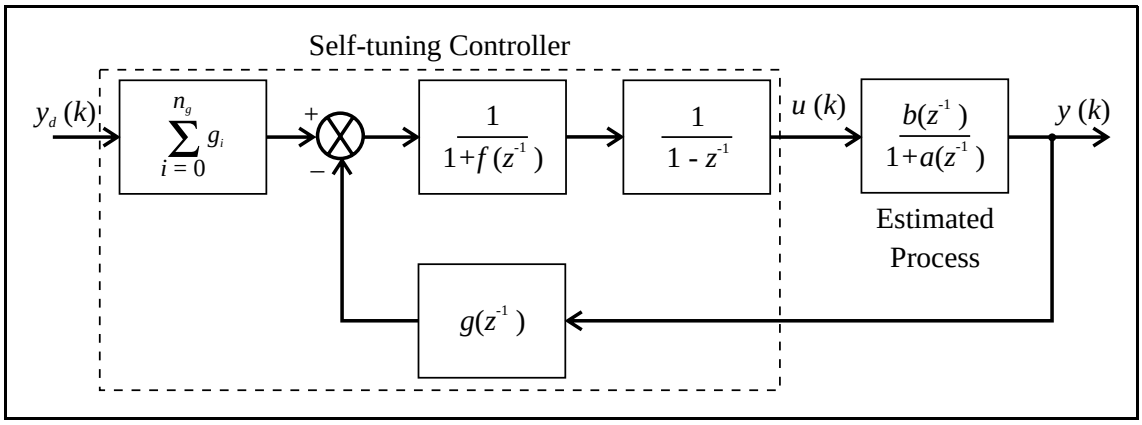

Figure 4.3 Alternative Incremental Self-tuning Controller Structure

This removes the contribution of the $g(z^{-1})$ polynomial to the closed loop system zeros, resulting in the following closed loop transfer function (cf. Equation 4.15) :-

$$y(k) = \frac{b(z^{-1}) \sum_{i=0}^{n_g} g_i}{t(z^{-1})} y_d(k) \tag{4.20}$$

The presence of a feedback loop can cause identifiability problems in the estimation part of the self-tuning control system. This problem stems from the fact that with a closed loop system the input, $u(k)$, is dependent upon the output, $y(k)$, which can lead to the covariance matrix becoming ill-conditioned due to linear dependency among the rows of $P(k)$. The problem is alleviated by using time varying or nonlinear controllers, or by using a *persistently exciting* signal as discussed in the following section.

The self-tuning controller design equations (Appendix C) can also be used to tune the gains of a corresponding fixed gain controller. RLS identification is used to obtain the parameter estimates of the process when operating under typical conditions. These parameter estimates are then used in the design equations to give a set of constant controller gains, so as to meet the specified performance criterion. Obviously, the resulting fixed gain controller will not maintain the desired response when the operating conditions vary, as the adaptability of the controller is no longer present, nevertheless it will produce the required response under the nominal conditions. This form of off-line design removes the need for troublesome manual tuning of a fixed gain controller, which is often problematic.

4.2.4 Operational Issues of Self-tuning Controllers

To ensure that a self-tuning controller will reliably work, there are many aspects of the system that require careful consideration before the controller is used. Firstly, there is the correct initialisation of the controller, and secondly there is the need to ensure that the

various components of the controller continue to function properly.

To initialise the estimator, suitable values for $\psi(0)$, $\Phi(0)$ and $P(0)$ (or $U(0)$ and $D(0)$ in the case of the BUD-RLS) are required. The usual way to attain the initial regression vector, $\psi(0)$, is to allow it to fill with system data for the required number of samples, before the estimator is turned on. The initial parameter estimates, $\Phi(0)$, can be specified if known, from either knowledge of the process or a priori identification. However, if little or no knowledge of the process is available, then an alternative is to assume that the process acts as an integrator with unity gain, so suitable initial model parameters would be :-

$$a_i = \begin{cases} -1 & i = 1 \\ 0 & i = 2 \ldots n_a \end{cases}$$
$$b_i = \begin{cases} \tau_s & i = 1 \\ 0 & i = 2 \ldots n_b \end{cases} \qquad (4.21)$$

where τ_s is the sample period of the controller.

The value used for $P(0)$ should reflect the uncertainty in the initial parameter estimates. When $\Phi(0)$ is well known a small value of $P(0)$ is used, typically $P(0) = I$. On the other hand if there is no prior knowledge then a higher value, e.g. $P(0) = 100I$, allows the estimates to converge quickly to the true values. For the BUD-RLS algorithm this uncertainty is similarly reflected in $D(0)$, whilst $U(0)$ is initialised to the identity matrix.

The rate of convergence is also dependent upon the forgetting factor, λ, which is set to reflect the variability of the system parameters during typical use. However, a *variable forgetting factor* can be employed when $\Phi(0)$ is not well known. An initial low value of λ allows the estimates to converge quickly, and is then increased with time to the required value.

Often when self-tuning controllers are presented, no a priori knowledge is utilised, even though this assumption can be met under most circumstances. In such cases there is usually a period where the system is controlled using a default, usually fixed gain

controller. This allows the parameter estimates to converge before the self-tuning controller is started. However, during this convergence period the control is poor due to the use of the default control and this degradation defeats the purpose of using an advanced controller to a certain extent.

The alternative is to obtain initial estimates of the process model that are closer to the true values, enabling the self-tuning controller to be introduced sooner. As mentioned earlier, these initial estimates can be obtained from an analytic model of the process, or from off-line estimation using previous system responses. The former involves linearisation of the model about a set point, whilst off-line estimation simply utilises the same identification algorithms described earlier.

To obtain good estimates, either on-line or off-line, the excitation signal needs to meet the requirement of *persistent excitation*. This ensures that the dominant process modes are excited, enabling the estimates to converge to their true values. Suitable input signals are square wave, pseudo-random binary noise or white noise, with the former being most practical for real implementations. The frequency content of the signal should be sufficient to excite the major modes of the process. For a square wave a rule of thumb is to use a period approximately six times the major time constant of the system [2.57].

The requirement of persistent excitation also needs to be maintained during the operation of the self-tuning controller. If this cannot be met, then the eigenvalues of the covariance matrix will tend towards zero or become large, leading to destabilisation of the estimator and the controlled system. This is suppressed by using one or more of the *covariance management techniques* described below :-

Constant Trace Algorithms: These work by automatically bounding the size of $P(k)$ within the algorithm.

Directional Forgetting: Here, a forgetting factor is used to update only those parameters

for which new information is available.

Variable Forgetting: The forgetting factor can also be specified as a function of the prediction error, $\epsilon(k)$, such that when the error increases the forgetting factor decreases allowing the model to adapt.

Estimator Dead-zone: This method suspends the estimation process during periods of low excitation. By monitoring the prediction error the estimator can be re-started when necessary.

Covariance Resetting: Periodic or heuristically determined resetting of $P(k)$. This method, however, has been reported to give oscillatory parameter estimates [2.83].

Dither Signals: The requirement for persistent excitation of the process can be met by super-imposing a dither signal (e.g. a square wave) on the input. This can degrade the steady state output of the system, and may exceed the required accuracy of the system.

This functionality is lumped under the term *jacketing software,* as it surrounds the self-tuning controller to ensure the integrity of the overall system. Monitoring functions may also be included, for example control limit checking. If the controller output reaches a saturation limit, then this may be an indication that there is a problem, and the control could be either reset, or switched to a backup controller.

Finally, the choice of sample period, τ_s, is important for the correct operation of a self-tuning controller. As an initial guide it should be about between 0.25 and 0.1 times the dominant process time constant.

4.3 Application of SISO Self-tuning Controllers to Robot Control

Self-tuning control is particularly suitable for robot control since the large changes in robot dynamics that occur during typical operations, can be tracked and compensated automatically. This section considers the SISO problem of controlling the position of

individual joints. Cartesian position control and hybrid position/force control require a MIMO controller, and will be discussed later. Self-tuning controllers have been applied to the control of the individual joints of manipulators [2.79, 2.59, 1.13], with each joint process assumed to be independent of the others. This assumption is valid when the manipulator is moving slowly, further, if the coupling between joints does become significant then it is simply treated as a disturbance.

Each joint is modelled as a discrete time difference equation, in the form of Equation 4.1, with the input corresponding to the actuator input and the output taken as either the joint angle or the joint angular velocity. Tachometers, that give the angular velocity measurements, are generally found only on specialised laboratory robots. The manipulator used here, the Slingsby TA9, only has joint angle sensors, and so only positional outputs are considered in the following discussion.

The model order used for self-tuning control of manipulators is often chosen so as to result in a simple control scheme, for example the self-tuning PID [2.79]. However, it is preferable to reflect the underlying physical system as closely as possible. The nonlinear expression representing the dynamics of a generalised *n* DOF manipulator, given by Equation 3.27, can be linearised about a set point and then discretised using Euler's method [2.59]. This results in the following multi-input multi-output discrete time autoregressive model :-

$$\left[\mathrm{I} + A_1 z^{-1} + A_2 z^{-2}\right]\Theta(k) = \left[B_1 z^{-1} + B_2 z^{-2}\right]\mathrm{T}(k) + D_0 + E(k) \qquad (4.22)$$

where $\Theta(k)$ are the outputs (a $n\times1$ vector of joint angles) and $\mathrm{T}(k)$ are the inputs (a $n\times1$ vector of joint torques), A_i and B_i are $n\times n$ matrices, and D_0 and $E(k)$ are $n\times1$ vectors, with A_i, B_i, and D_0 constituting the MIMO model and $E(k)$ representing the modelling errors (cf. Equation 4.1).

This MIMO model can be simplified by neglecting the coupling between the joints

of the manipulator, represented by the off-diagonal terms of the A and B matrices. Therefore, treating each joint as a separate SISO system, Equation 4.22 becomes :-

$$\left[1 + a_1 z^{-1} + a_2 z^{-2}\right]\theta(k) = \left[b_1 z^{-1} + b_2 z^{-2}\right]\tau(k) + d_0 + e(k) \tag{4.23}$$

where $\theta(k)$ and $\tau(k)$ are the joint angle and joint torque respectively for each joint. This model corresponds to that given in Equation 4.1 with $n_a = n_b = 2$. The scalars a_i and b_i model the effects of inertia and damping of the link, and d_0 can be interpreted as modelling the effects of gravity on the links and offsets within the actuation mechanism. To represent a n DOF manipulator requires n of these independent equations, with any coupling between joints being treated as a disturbance to the SISO model, also embodied in d_0.

This representation assumes that the actuator acts a pure torque source, however this is not appropriate for hydraulic actuators, as used by the Slingsby TA9. These actuators are primarily rate driven, where the *velocity* of the actuator is proportional to the applied servovalve voltage, $v(k)$. Hence, these actuators are often approximated by an integrator [2.6], which can be modelled by introducing an extra pole into the joint model, increasing n_a to 3 :-

$$\left[1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}\right]\theta(k) = \left[b_1 z^{-1} + b_2 z^{-2}\right]v(k) + d_0 + e(k) \tag{4.24}$$

The SISO joint model of Equation 4.24 is used in this thesis, together with simpler models that yield self-tuning PI or PID controller structures for the purposes of comparison. The controllers that correspond to these models are derived in Appendix C.

Many different model orders and structures have been proposed in the literature, when self-tuning controllers have been applied to the independent control of manipulator joints. The model used is dependent upon the manipulator instrumentation and actuation mechanism, although as mentioned earlier it is sometimes chosen to provide a certain order

of controller. The SISO joint models used in this thesis [1.13] and those used in previously proposed self-tuning robot controllers, as discussed in Section 2.5.3, are summarised in Table 4.1.

When using any form of controller with a robotic arm, a sample rate of typically 50 Hz and upwards is required, depending on the particular manipulator. The computational complexity of self-tuning controllers is often cited as the reason for their exclusion from this application. However, the use of fast and inexpensive microprocessors coupled with the efficient estimation algorithms described in Section 4.2.2, have removed this restriction for all but a few cases.

The *Bierman U-D factorisation* (BUD-RLS) algorithm should be employed to eliminate problems associated with numerical round-off, and to ensure that the covariance matrix remains in good condition. The use of a priori estimates for the initial system parameters has been shown to improve the initial response of a self-tuning controller [2.63], and so should be used whenever possible.

A forgetting factor is required to allow the controller to adapt to changes in the robot dynamics as it performs tasks, with forgetting factor set according to the anticipated variability of the system parameters. The inclusion of a forgetting factor motivates the need for *covariance management* as discussed earlier in this chapter. This should also alleviate problems caused by not fully satisfying the criterion of persistent excitation, which may result from the trajectory required to achieve the task. In a practical installation, a simple backup controller should become automatically available if problems with the self-tuning controller are detected.

4.4 Multivariable Self-tuning Control

The development of the self-tuning controller in Section 4.2 pertained to a single-input single-output (SISO) system. In this section the work will be extended to the multi-

	n_a	n_b	n_d	n_Φ	n DOF	Input	Output	Notes:
Koivo [2.59]	2	2	1	4	6	motor voltage	joint velocity	$a_1 = 0$.
Lelic [2.64]	2	2	1	5	1	motor voltage	joint angle	Uses internal velocity feedback loop.
Karam [2.79]	2	1	0	3	6	motor current	joint angle	To give self-tuning PID.
Broome [2.80]	2	2	0	4	2	joint torque	joint angle	Ideal torque source actuators.
Finney [2.78]	3	2	0	5	1	servovalve voltage	linear position	linear hydraulic actuator acting in a horizontal plane, hence $n_d = 0$.
Plummer [2.81]	3	3	0	6	1	servovalve current	linear position	
Sepheri [2.60]	3	3	1	7	2	servovalve spool posn.	joint velocity	Rotary hydraulic actuators.
Sepheri [2.60]	2	2	1	4	2	differential actuator pressure	joint velocity	$a_1 = 0$. Cascaded with a classical controller, which controls the hydraulic actuator.
Ananthakrishnan [2.68]	2	2	0	4	4	servovalve voltage	joint angle	Rotary and linear hydraulic actuators.
Eun [2.84]	2	2	1	5	3	joint torque	force/position	Puma 560 with ideal torque sources.
Koivo [2.26]	2	1	1	4	3	joint torque	joint angle	Ideal torque source actuators.
Wang [2.85, 2.86]	2	2	0	4	3	end-effector posn.	contact force	Puma 560 with passive compliance.
Clegg [1.13, 2.87]	3	2	1	6	1	servovalve voltage	joint angle	TA9. Rotary and linear hydraulic actuators.
Clegg [1.13, 2.87]	2	1	0/1	4	1	servovalve voltage	joint angle	As above. To give self-tuning PID.

Table 4.1 Different Model Orders and Structures used for SISO Self-tuning Control of Manipulator Joints

Notes:

1) n is the number of joints to which the independent controllers were applied.

2) The details of the specific self-tuning controllers used are given in Chapter Two.

input multi-output case (MIMO) for a system with n inputs and n outputs, denoted by the vectors $U(k)$ and $Y(k)$ respectively. This type of scheme can be used to control the entire robot as a single system, including the coupling between joints. The initial extension of the SISO self-tuner to a MIMO system was proposed by Prager [4.4] and the theory developed here follows the original formulation.

The MIMO controller is structurally identical to the SISO scheme, with a MIMO model of the process, an on-line MIMO system identifier and a MIMO pole placement controller (cf. Figure 4.1). However, the use of matrices requires careful consideration of the order in which they appear, since matrix arithmetic is not commutative. This results in an additional step when deriving the controller polynomials which will be detailed later.

The three components of the multivariable self-tuning controller will now be discussed in turn.

4.4.1 The MIMO Process Model

The low-order linear model of a n-input, n-output process can be represented as :-

$$\left[\mathrm{I} + A(z^{-1})\right] Y(k) = B(z^{-1})\, U(k) + D_0 + E(k) \tag{4.25}$$

which compares directly with Equation 4.1, where the MIMO equivalents of the SISO parameters are designated by their uppercase equivalents. $A(z^{-1})$ and $B(z^{-1})$ are polynomial matrices in the backward shift operator z^{-i}, and are of the form :-

$$\begin{aligned} A(z^{-1}) &= A_1 z^{-1} + \ldots + A_{n_a} z^{-n_a} \\ B(z^{-1}) &= B_1 z^{-1} + \ldots + B_{n_b} z^{-n_b} \end{aligned} \tag{4.26}$$

where A_i and B_i are $n \times n$ matrix coefficients. Similarly, D_0 and $E(k)$ are $n \times 1$ vectors representing the constant unmeasurable drift disturbances and modelling errors. Therefore the total number of model parameters for the MIMO model, n_Φ, is :-

$$n_{\Psi} = (n_a + n_b)n^2 + n_d n \tag{4.27}$$

The diagonal elements in the A_i and B_i matrices reflect the relationship between the nth input and nth output, whereas the off-diagonal terms model the coupling between the different inputs and outputs. Again the model order and structure should be chosen to reflect the continuous time process being modelled. However, the number of parameters to be estimated, Equation 4.27, may quickly become excessive and often only low order models may be feasible.

4.4.2 MIMO System Identification

MIMO system identification is a simple extension of the SISO RLS algorithm (covered in Section 4.2.2) which iteratively estimates the model parameters using past values of the actual system inputs and outputs. Again, the process model, Equation 4.25, is formulated in terms of the parameters to be estimated (cf. Equation 4.5) :-

$$Y(k) = \Phi^{\mathrm{T}}(k)\,\psi(k) + E(k) \tag{4.28}$$

however now, the estimated model parameter matrix, $\Phi(k)$, is a $(n_a n+n_b n+n_d)\times n$ matrix and the *regression vector*, $\psi(k)$, is a $(n_a n+n_b n+n_d)\times 1$ vector. More specifically :-

$$\Phi^{\mathrm{T}}(k) = \left[-A_1, \ldots, -A_{n_a}, B_1, \ldots, B_{n_b}, D_0 \right] \tag{4.29}$$

$$\psi^{\mathrm{T}}(k) = \left[Y^{\mathrm{T}}(k-1), \ldots, Y^{\mathrm{T}}(k-n_a), U^{\mathrm{T}}(k-1), \ldots, U^{\mathrm{T}}(k-n_b), 1 \right] \tag{4.30}$$

The SISO MIL-RLS algorithm, Equations 4.8 to 4.11, is extended to the MIMO case simply by modifying Step 4 to accommodate the new structure of $\Phi(k)$. The remaining steps are the same as the SISO case, with the a priori prediction error, $\mathrm{E}(k)$, the Kalman gain vector, $L(k)$, and the covariance matrix, $P(k)$, sized appropriately. Step 4 becomes :-

Step 4: Update the parameter estimates, $\Phi(k)$ with $i = 1, 2, ..., n$:-

$$\Phi_i(k) = \Phi_i(k-1) + L(k)\mathrm{E}_i(k) \tag{4.31}$$

where $\Phi_i(k)$ is the ith *column* of $\Phi(k)$, and $\mathrm{E}_i(k)$ is the ith element of the a priori prediction error vector.

This modification can also be applied to Step 7 of the numerically robust BUD-RLS algorithm (see Appendix B) to extend it to the MIMO case, with the other terms sized accordingly.

The operational issues of the SISO RLS estimator, presented in Section 4.2.4, are equally applicable to the MIMO case. Correct initialisation is important, as is the use of covariance management and jacketing software to maintain persistent excitation of the system and prevent covariance blow-up. However, the use of a single forgetting factor, λ, may be inappropriate for a MIMO process, and more complex schemes may be required to allow different adaptation rates for different parts of the system.

The estimation algorithm for the multivariable case does not involve much extra computation compared to a SISO system with the same number of parameters. This is because the columns of $\Phi(k)$ are updated using the same covariance matrix, $P(k)$. However, MIMO process models generally have many more parameters than an equivalent number of independent SISO models. For example, the n-input n-output MIMO model can be replaced by n SISO models, and from Equations 4.27 and 4.4 respectively :-

Number of parameters for MIMO representation = $(n_a n + n_d n + n_d)n$,

Number of parameters for n SISO models = $(n_a + n_d + n_d)n$.

The extra terms in the MIMO case represent the coupling between different inputs

and outputs. With so many parameters to be estimated, any terms which can be fixed to zero will significantly reduce the computational requirements of such an algorithm. The use of this technique in robot control will be discussed in Section 4.5.

The system identification part of a self-tuning controller is the most computationally intensive, and this increases exponentially with the number of inputs/outputs.

4.4.3 MIMO Pole Placement Controller

The multivariable self-tuning pole placement controller originally proposed by Prager [4.4] was derived for the regulator problem. Here, this is extended to provide reference tracking, incorporating a digital integrator to form an *incremental controller*, as for the SISO controller. The structure of the controller is identical to the SISO controller, shown in Figure 4.2, with the MIMO parameters denoted by their uppercase equivalents :-

$$\begin{aligned} F(z^{-1}) &= F_1 z^{-1} + \ldots + F_{n_f} z^{-n_f} \\ G(z^{-1}) &= G_0 + G_1 z^{-1} + \ldots + G_{n_g} z^{-n_g} \end{aligned} \tag{4.32}$$

where F_i and G_i are $n \times n$ matrix coefficients.

Due to problems associated with the non-commutivity of the matrix polynomials, the controller must initially be cast in the following form :-

$$U(k) = G(z^{-1})\left[\mathrm{I} + F(z^{-1})\right]^{-1}\left[\mathrm{I} - \mathrm{I}z^{-1}\right]^{-1}(Y_d(k) - Y(k)) \tag{4.33}$$

Using this formulation, the overall closed loop transfer function of the controlled system can be derived to give the following, which corresponds to that for the SISO case (cf. Equation 4.15) :-

$$Y(k) = B(z^{-1})G(z^{-1})\left[(\mathrm{I} + A(z^{-1}))(\mathrm{I} - \mathrm{I}z^{-1})(\mathrm{I} + F(z^{-1})) + B(z^{-1})G(z^{-1})\right]^{-1} Y_d(k) \tag{4.34}$$

The closed loop system poles are set to some specified values, defined by the roots

of the polynomial matrix $T(z^{-1})$, by solving the following equation for the controller polynomials, $F(z^{-1})$ and $G(z^{-1})$:-

$$T(z^{-1}) = (\mathrm{I} + A(z^{-1}))(\mathrm{I} - \mathrm{I}z^{-1})(\mathrm{I} + F(z^{-1})) + B(z^{-1})G(z^{-1}) \tag{4.35}$$

where :-

$$T(z^{-1}) = \mathrm{I} + T_1 z^{-1} + \ldots + T_{n_t} z^{-n_t} \tag{4.36}$$

and T_i are $n \times n$ matrix coefficients.

Equation 4.34 is solved by equating like powers of z, resulting in a set of simultaneous linear equations which can be solved to give expressions for F and G in terms of A, B and T. For a unique solution to exist, the orders of the F, G and T matrix polynomials must meet the constraints of Equation 4.18. The solution to Equation 4.35 can be represented as a matrix expression, though again for the general case this becomes inconvenient. Specific solutions for the different model orders used in this thesis are presented in Appendix C, and have a similar form to the equivalent SISO solutions.

The controller given by Equation 4.33, is not directly implementable due to matrix non-commutivity [4.4], so it must be transformed into a realisable form. One method to achieve this involves using the *pseudo-commutivity* relation :-

$$[\mathrm{I} + \tilde{F}(z^{-1})]G(z^{-1}) = \tilde{G}(z^{-1})[\mathrm{I} + F(z^{-1})] \tag{4.37}$$

which, when substituted into Equation 4.33, gives the following *realisable* controller :-

$$\left[\mathrm{I} + \tilde{F}(z^{-1})\right]\left[\mathrm{I} - \mathrm{I}z^{-1}\right]U(k) = \tilde{G}(z^{-1})\,(Y_d(k) - Y(k)) \tag{4.38}$$

The solution to Equation 4.37 again yields a set of simultaneous equations specific to the polynomial orders used, and the explicit solutions particular to this thesis are given in Appendix C. Simpler approaches to obtain a realisable controller do exist [4.5], however

these can be problematic when nonminimum phase dynamics are present, so the generalised solution is used here.

The controller polynomials are calculated at each sample instant after Step 4 of the RLS algorithm, once $\Phi(k)$ has been updated. Then the pseudo-commutivity transformation, Equation 4.37, is applied and the controller output, $U(k)$, is calculated using Equation 4.38.

Similar analysis to that for the SISO controller, shows that the MIMO incremental controller equations are not altered by inclusion of the disturbance parameters, D_0, in the process model. Also, if the transient response is unacceptable due to the effect of the system zeros, a modified control law (cf. Figure 4.3) can be used to reduce the effect of the $G(z^{-1})$ polynomial.

4.5 MIMO Self-tuning Control of Robots

The MIMO self-tuning controller described in the previous sections can be applied to the control of a robot manipulator in a number of ways. The process input is taken as the vector of joint torques, $\Gamma(k)$, or actuation signals, for example a vector of actuator voltages.

The process output can take a variety of forms depending upon the required control. Firstly, the output can be taken as a vector of joint space variables [2.59], either joint positions or velocities, corresponding to the robot model of Equation 4.22. The diagonal elements of the matrices relate each joint output to its own actuation signal, and conversely the off-diagonal terms represent the coupling between the joints.

An alternative approach is to take the process output as a vector of end-effector Cartesian coordinates [2.63], again either positions or velocities. This is known as *Cartesian control* and only requires the forward kinematics rather than the complicated inverse kinematics, as shown in Figure 2.1.

The Cartesian control strategy can be extended to the case where the manipulator is constrained by a surface, namely the well known *hybrid position/force control* problem

[2.89, 2.40]. This uses a process output vector consisting of forces and torques in the constrained directions, and positions and orientation in the orthogonal directions. These Cartesian space control schemes are advantageous since manipulation tasks naturally decompose into this frame of reference, both in terms of desired outputs and control performance criterion.

The model order and structure used by the self-tuning controller should reflect the underlying continuous time system. The extension to the Cartesian control schemes does not introduce any additional dynamics, since the forward kinematics involves trigonometric functions, and the interaction of the robot with the environment is approximated by the expression of Equation 3.29. Hence the model order derived for the SISO systems is equally applicable to these MIMO cases.

A summary of the MIMO robot models used in this thesis [2.40] and those used in previously proposed self-tuning MIMO robot controllers, is given in Table 4.2. These will now be described in more detail.

Koivo's initial work on MIMO self-tuning robot controllers [2.59] still operated in joint space, controlling the joint angle velocities. The natural decoupling of the manipulator was used to divide the control into two 3 DOF systems. This reduced the complexity of the multivariable model to a manageable level, with the A_2 matrix being partitioned into two 3×3 matrices. The A_1 matrix was set to zero, and the B_i matrices were diagonal, which reduced the number of model parameters to 36 for the 6 DOF system. However, this MIMO controller did not show any improvements over the use of equivalent SISO controllers.

The work was extended to a MIMO Cartesian controller in [2.63], where Cartesian velocities were used as the system output for a 3 DOF robot. A low model order was used ($n_a = n_b = n_d = 1$), and by setting A_1 to be diagonal this resulted in only 15 parameters to estimated. The rationale for making A_1 diagonal, was that the Cartesian outputs are orthogonal, and hence independent of the outputs in the other directions.

	n_a	n_b	n_d	n DOF	n_Φ	Model Input	Model Output	Notes:
Koivo [2.59]	2	2	1	6	36	motor voltages	joint velocities	$A_1 = 0$, A_2 partitioned into two 3×3 matrices to match coupling of robot, B_i = diagonal.
Koivo [2.63]	1	1	1	3	15	motor voltages	Cart. velocities	A_1 = diagonal.
Koivo [2.63]	1	1	1	3	24	motor voltages	Cart. positions and velocities	A_1 = diagonal. Extended to control of *both* Cartesian position and velocity.
Koivo [2.89, 2.26]	1	1	1	2	8	joint torques	Cart. velocities	A_1 = diagonal. Model used within a MIMO position-velocity-force controller.
Ozsoy [2.90]	2	2	0	2	16	joint torques	Cart. velocities and force derivatives	Small ideal manipulator
Clegg [2.40 and Chapter Seven]	3	2	1	2	22	servovalve voltages	Cart. position and force	Linear hydraulic actuators.

Table 4.2 Different Model Orders and Structures used for MIMO Self-tuning Control of Manipulators

Notes:

1) *n* is the number of DOF to which the MIMO controller was applied.

2) The details of the specific self-tuning controllers used are given in Chapter Two.

Koivo also investigated the application of the same low order model structure to achieve a type of self-tuning hybrid position/force control [2.26]. In this scheme, the force and position errors are embedded within the LQG controller design criterion, but are not specifically used within the MIMO model which uses the Cartesian velocities of the robot. This was extended by Ozsoy [2.90], who used a MIMO square-root parameter estimation algorithm together with a minimum variance controller. The estimated model used $n_a = 2$, $n_b = 2$ and 2 DOF, giving a total of 16 parameters to be estimated.

The self-tuning hybrid position force controller presented in Chapter Seven of this thesis [2.40] extends this further, employing a MIMO process model which explicitly uses the end-effector forces and positions as the system outputs. The controller is applied to the 2 DOF hydraulic manipulator described in Chapter Three. In this case, the system has two servovalve input voltages, $v_1(k)$ and $v_2(k)$, and the output consists of the Cartesian position along the ${}^{C}y$-axis, ${}^{C}y(k)$, and the force along the ${}^{C}x$-axis, ${}^{C}F_x(k)$.

With this manipulator, the coupling between the orthogonal force and position controlled directions was significant enough to require the inclusion of the off-diagonal elements in the model matrices. This resulted in a model with 22 terms to be estimated. This increases rapidly when extended to robots with more DOFs, for example a 6 DOF robot would require $n_\Phi = 186$ parameters. The actual computational requirements will be quantified for this scheme, when the experimental implementation is discussed later.

4.6 Summary

This chapter has introduced the concept of system identification and self-tuning control, including the extension to multivariable systems. The application of both SISO and MIMO self-tuning controllers to various robot control problems has been discussed, including the approaches proposed in previous work.

The structure of the controllers have been described in detail, including presentation

of both the conventional MIL-RLS and numerically robust BUD-RLS system identification algorithms. The important issues concerning initialisation and practical operation of these controllers have also been covered.

The derivation of both SISO and MIMO pole placement controllers has been presented, with the specific solutions used in this thesis being given in Appendix C. The use of an incremental controller was introduced, together with an alternative structure to alter the influence of the controller zeros, if required.

Subsequent chapters will detail the specific implementations of the self-tuning controllers discussed here, with application to both real and simulated manipulators. An independent SISO self-tuning joint angle controller is presented in Chapter Five, and compared to a conventional fixed gain controller. The results for a fixed gain hybrid position/force controller are given in Chapter Six, which is then compared with a self-tuning MIMO controller in Chapter Seven.

Chapter 5

SISO Self-tuning Pole Placement Joint Angle Control

5.1 Introduction

This chapter presents a SISO self-tuning pole placement controller to control the joint angles of a typical hydraulic underwater manipulator. This work demonstrates the feasibility and benefits of using self-tuning control over conventional fixed gain PID controllers, under a variety of operating conditions. This forms the preparatory work to the practical development of a MIMO self-tuning hybrid position/force controller discussed in Chapter Seven. Consequently, the operational issues of using self-tuning control with such a manipulator are studied extensively. The SISO self-tuning controllers developed here are applied within the structure shown in Figure 2.1a, with the advantages and disadvantages of such a scheme having being discussed in Chapter Two.

The experimental setup is described first, then the procedures involved in operating a self-tuning pole placement controller with this robot are detailed. This includes investigating the different algorithms and operational issues needed for reliable system identification. The performance of this controller is compared directly with a conventional fixed gain controller under a variety of operating conditions, examining both transient and steady-state performance.

5.2 Experimental Setup

The experimental robot used in this work is a Slingsby TA9 hydraulic manipulator,

which is widely used by the offshore industry in the North Sea. This robot has already been described in Section 3.2 and is shown in Figure 3.1. The analogue proportional controllers, that are employed as standard, have been replaced by purpose built interfacing circuitry and digital controllers.

The analogue signals from the robot comprise the potentiometer outputs located at each joint, and the force/torque sensor outputs. These signals pass through dedicated signal conditioning hardware which provides adjustable gains and level shifting to ensure that the full dynamic range of the analogue to digital convertors (ADCs) is used. Anti-aliasing filters and buffering are also included as appropriate in this signal conditioning. The potentiometer output voltages are assumed linearly proportional to the joint angles, and are routinely calibrated by driving each joint to its mechanical end-stops, and reading the corresponding voltages at the ADC. These voltages are then used to determine the linear relationship between voltage and joint angle. No such calibration procedure is required for the force/torque sensor as it is a factory calibrated unit.

The controller outputs, from the digital to analogue convertors (DACs), are converted by a voltage to current amplifier into an appropriate driving signal for the electrohydraulic servovalves. The gains of these current amplifiers are adjusted so that full scale DAC output produces full scale servovalve current. The ADC and DAC signal conditioning circuitry is duplicated for each potentiometer and servovalve respectively. The TA9 manipulator has 7 joints and when coupled with the six DOF force/torque sensor gives a total of 7 inputs and 13 outputs.

The real-time control algorithms are implemented on a Texas Instruments TMS320C30 32-bit floating point digital signal processor (DSP), hosted by a proprietary Loughborough Sound Images (LSI) PC card. This is connected to a 32 channel 12 bit ADC board and a 16 channel 12 bit DAC board, which interface directly with the signal conditioning circuitry. These cards are hosted by a 486 PC, which provides power to all

three boards and a common dual-ported memory area that allows the PC to access the digital controllers.

The control algorithms are programmed in 'C', with the coding and compilation being carried out on the PC. The resulting object code is downloaded to the DSP via the dual ported memory, using LSI run-time library routines. The use of a high level language such as 'C', allows complex controller functions to be developed efficiently and the resulting assembler code produced by the DSP's compiler is highly efficient.

Although there is only one DSP, the requirement for an individual controller for each of the joints is met by operating the controllers sequentially. That is, the control for each joint is performed successively, and therefore the controller sample rate is determined by the number of joints being controlled and the time between successive joints. For example, to control 7 joints at an individual sample rate of 50 Hz, requires successive joints to be updated every 2.86 ms (350 Hz). In this system, this controller update rate is determined by a clock on the ADC card, which synchronises the reading of joint angle values in the main control loop. Synchronisation between the PC and DSP is also maintained at this sample rate, so that the data accessed by the PC remains current.

The PC supervises the operation of the controllers running on the DSP, storing data for off-line analysis and providing a graphical user interface (GUI) that allows access to the various controller parameters. The controller reference values can be set via the keyboard, or the standard master arms which also interface to the DSP (via the signal conditioning and ADC boards) to realise master slave control. The PC allows the user to record joint angles and use them later as controller references, providing a *teach and play* facility.

The supervisory PC also enables higher level systems, such as motion planning and task planning, to interface to the low level controllers as shown in Figure 1.1. These high level systems are implemented on UNIX systems and communicate with the PC over a LAN ethernet link using TCP/IP. A *trajectory interpolator* is used to smooth the values

generated by the motion planner, since this operates at a much lower update rate than the controller. This interpolation is performed on-line by the DSP which fits a fifth-order polynomial to the discrete values [5.1]. The independent joint angle controllers developed in this thesis have been successfully integrated with an on-line motion planner [1.9] that solves the inverse kinematics (see Figure 2.1a). The resulting system was capable of real-time trajectory following and obstacle avoidance [1.4], however, this work is not presented in this thesis.

5.3 Practical SISO System Identification

The first stage in designing a self-tuning controller is the development of the system identification block, shown in Figure 4.1. This section presents practical RLS system identification of the joints of the TA9 manipulator, and investigates the operational issues described in Section 4.2.4. Selection of suitable model orders and structures have been discussed at length in Section 4.3, and the ideas introduced there are reinforced with experimental results.

The two manipulator joints used in this work are the forearm rotate joint, driven by a rotary hydraulic actuator, and the elbow joint which uses a linear actuator acting about a pivot. All seven actuators of the manipulator are under control during this work, each using a fixed gain PI controller of the form given by Equation C.3. These are implemented on the DSP and are sequentially controlled at a sample rate of 51 Hz (τ_s = 19.6 ms) for each joint, which is used throughout this chapter.

Persistent excitation of the system identifier is required to ensure that good system estimates are obtained, as discussed in Section 4.2.4. This is achieved by commanding the joint to execute a series of steps. The response of the forearm rotate joint stepping between 210° and 170° with a period of 8 s, is shown in Figure 5.1. The PI controller used has gains

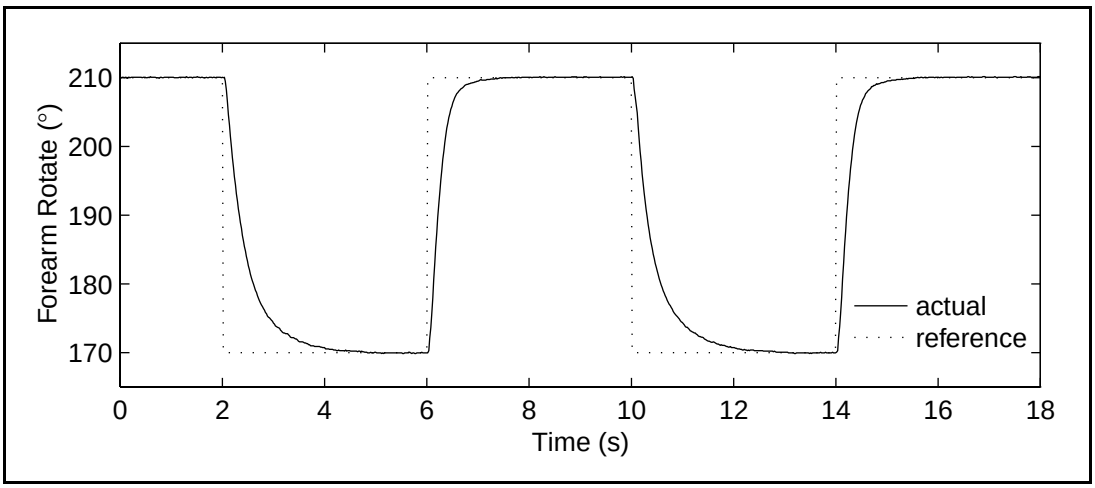

Figure 5.1 Response of Forearm Rotate Joint under Fixed Gain PI Control

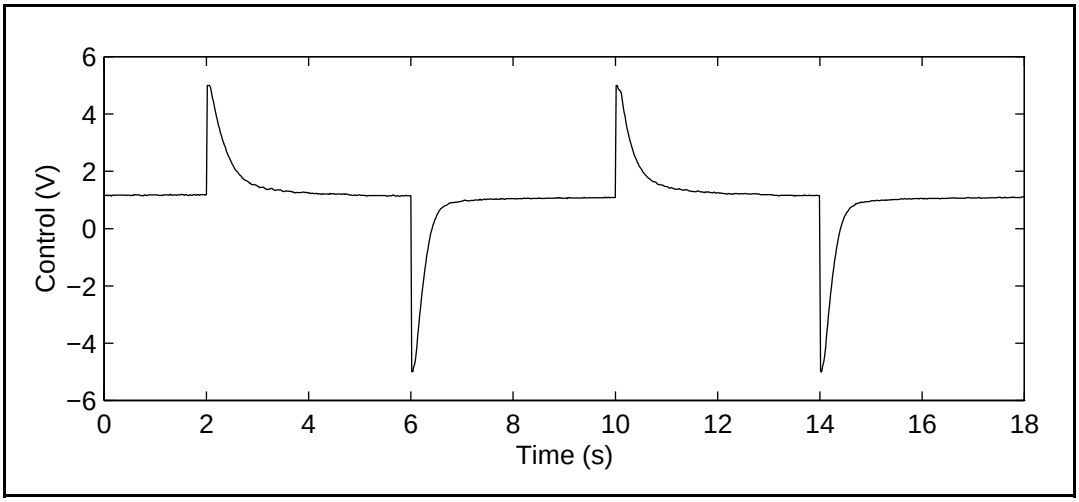

Figure 5.2 Fixed Gain PI Controller Output

of $k_p = 15.0$ and $k_i = 20.0$[†], which were obtained using a priori system estimates as described later in Section 5.4. The corresponding controller output is shown in Figure 5.2

The RLS algorithm described in Chapter Four, generates parameter estimates from past values of the system output and input, the joint angle (represented as a potentiometer voltage) and servovalve input respectively. To facilitate the investigation of the effect of different RLS operating parameters, this initial system identification is performed off-line. This removes effects attributable to differences between experimental runs, as identical sequences of input and output data are used. The off-line RLS algorithm was implemented

[†] The controllers were developed to act on joint angle errors represented by potentiometer voltages. The reason for this is that for certain joints the direction of motion caused by a positive control signal does not correspond to a positive joint angle (as defined by the Denavit-Hartenberg convention described in Section 3.3 and modelled by $k_{\theta dir}$ in Equations 3.20 and 3.21). The use of potentiometer voltages within the controller removes this inconsistency.

as an m-file in MATLAB matching the on-line RLS algorithm programmed in 'C' running on the DSP.

Initially, the standard Matrix Inversion Lemma RLS algorithm (MIL-RLS) is employed, together with a model order that represents the physical system, $n_a = 3$, $n_b = 2$ and $n_d = 1$. The six parameter estimates resulting from the forearm rotate joint response of Figures 5.1 and 5.2, are shown in Figure 5.3. No prior knowledge of the system model is assumed, with the initial parameter estimates, $\Phi(0)$, specified to represent an integrator, cf. Equation 4.21. The use of a priori estimates is investigated in the next section. The initial value for the covariance matrix, $P(0)$, is set to 100I to reflect this uncertainty, and a forgetting factor, λ, of 0.995 is used. Under these conditions, the parameter estimates vary as shown in Figure 5.3.

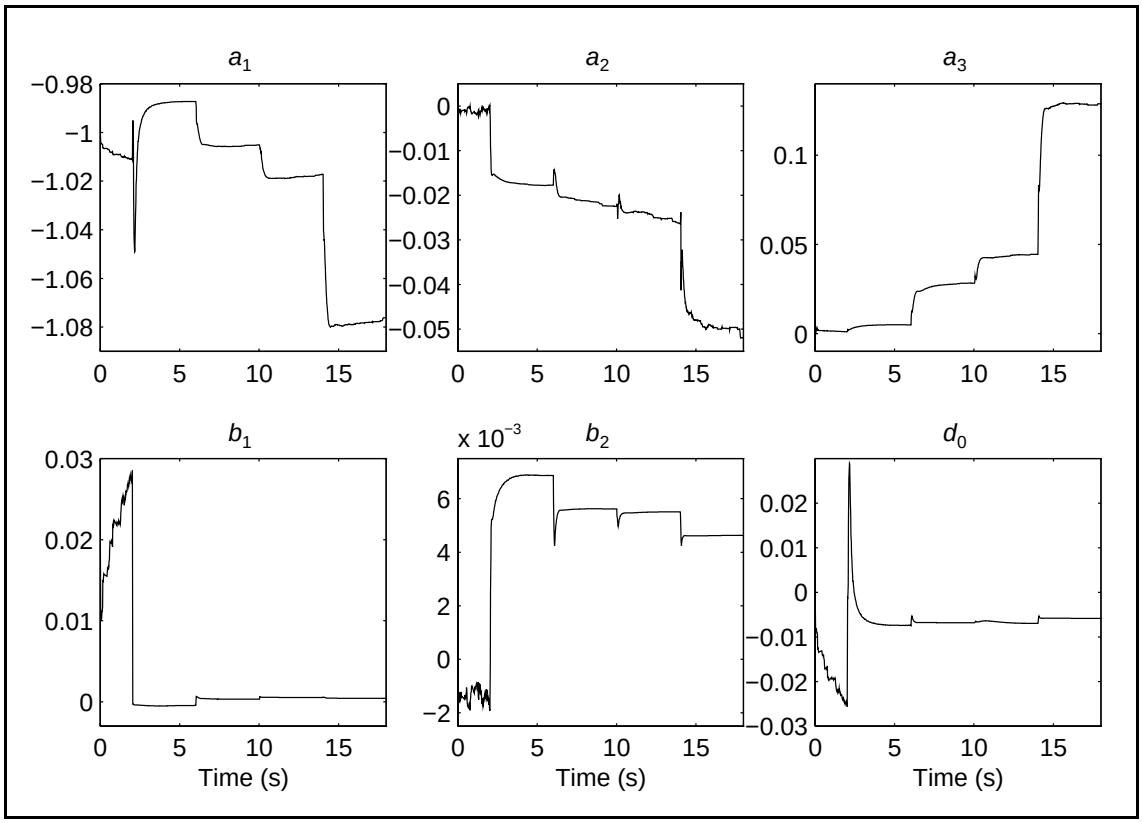

Figure 5.3 RLS Parameter Estimates for Forearm Rotate Joint

These parameter estimates show a relatively large degree of variation during the experiment. However, it is more useful to examine the poles, zeros and gain of the

estimated model as these provide a better insight into the underlying physical process. The discrete time poles, zeros and gain corresponding to the parameter estimates and ARMAX model given by Equations 4.1 and 4.2, are shown in Figure 5.4.

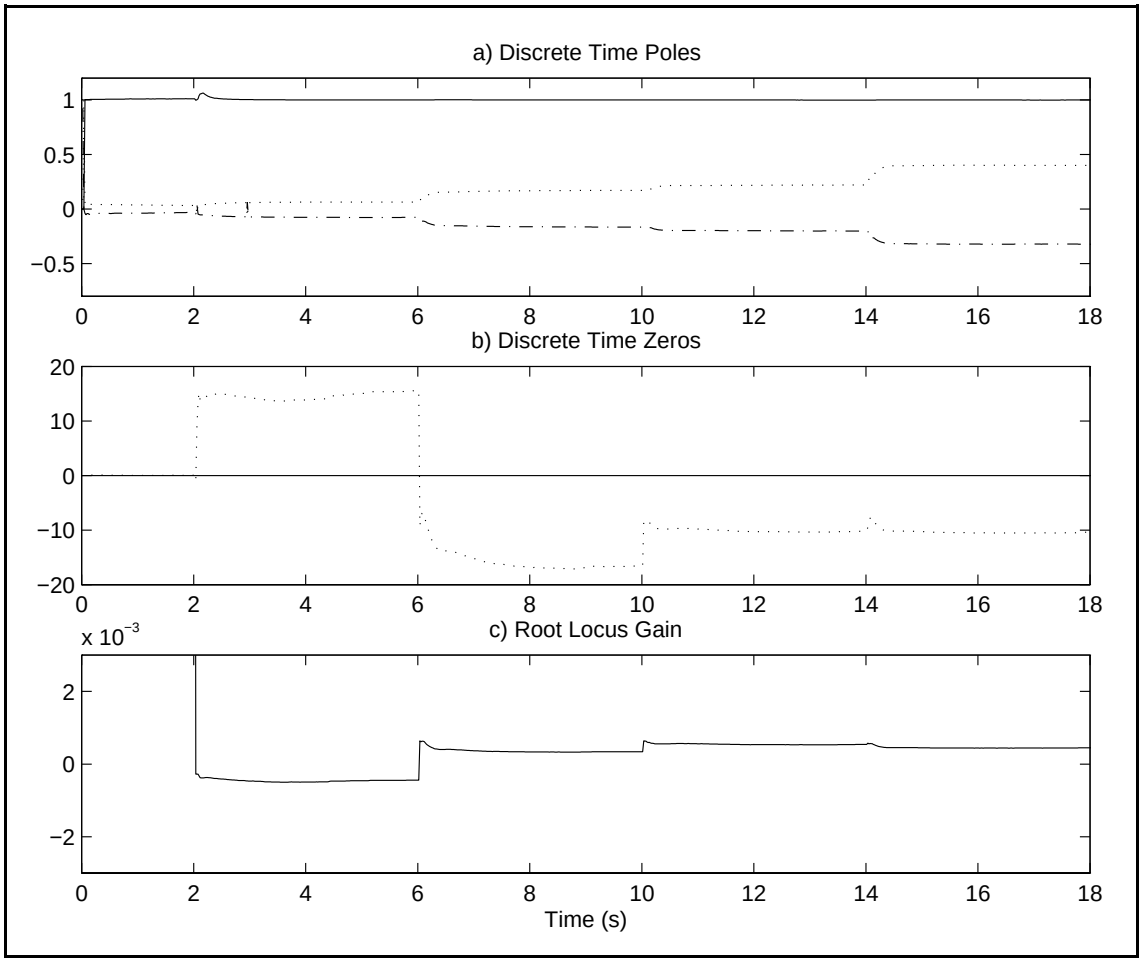

Figure 5.4 RLS Estimates of Poles, Zeros and Gain for Forearm Rotate Joint

As can be seen the poles converge faster and more smoothly than the individual polynomial coefficients that they are derived from. Also, the integrating nature of the system is clear, with one of the poles quickly attaining the value 1.0, corresponding to a discrete time integrator. The two remaining poles are located inside the unit circle which describes the area of stability for discrete time systems. The zeros are less important in terms of system response, one being located at the origin, due to the structure of the ARMAX model used, Equation 4.1, and the other converging to -10.5. The root locus gain converges in a similar fashion, to 0.44×10^{-3}.

The parameter estimates shown in Figure 5.3, and also the poles, zeros and gain illustrated in Figure 5.4, have a regular pattern of variation after the initial period of convergence. These regular variations are due to changes in the dynamics of the robot and specifically arise from the steps acting both with, and then, against gravity. These variations are tracked by the RLS algorithm due to the forgetting factor which reduces the influence of older data, allowing such time varying and nonlinear systems to be identified.

However, using a forgetting factor of 0.995 means that data that is 600 samples (or 11.8 seconds) old has a weighting of 5% of that of the current data [2.56]. This implies that any changes in the dynamics of robot, such as those caused by steps in different directions, will still have some significance within the parameter estimates some 10-12 seconds after the change occurred. This explains why the parameter estimates do not converge to similar values for subsequent steps in the same direction, since the parameter estimates are to some extent averaging the changing dynamics over steps in both directions.

The difference between the estimated model output and the actual system output is the *a priori prediction error*, $\epsilon(k)$, defined by Equation 4.8. This is shown in Figure 5.5, and indicates that the model output follows the actual system output closely. The mean of the a priori prediction error is close to zero, implying that there is no bias within the parameter estimates. The root mean square (RMS) of the a priori prediction error is another useful

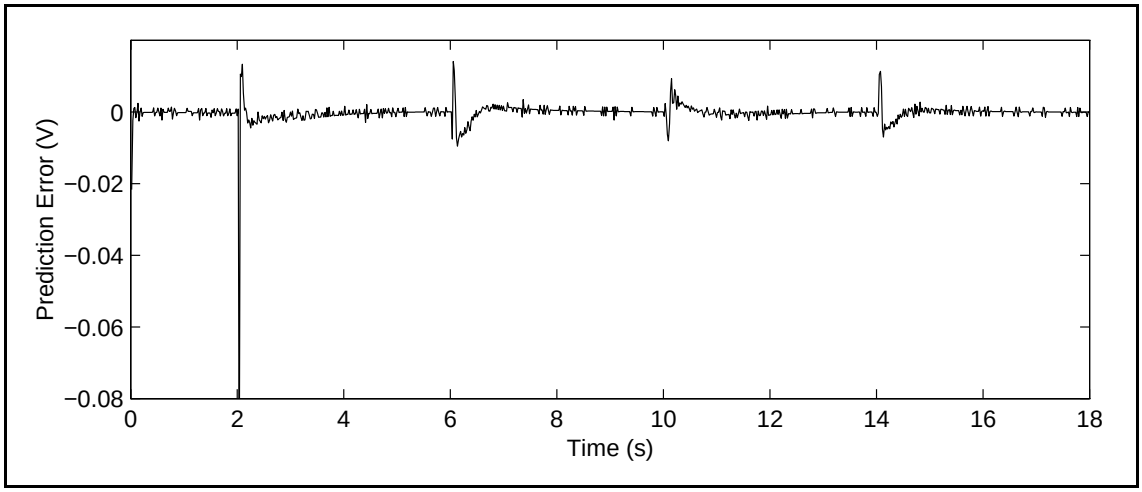

Figure 5.5 Prediction Error for RLS System Identification

performance measure for the identification process, the smaller the error the better is the estimation. For this particular system identification experiment the RMS a priori prediction error is 1.72×10^{-3} V, neglecting the first three seconds of data since the estimates are undergoing initial convergence.

5.3.1 Operational Issues for Practical System Identification

The regression vector, ψ, which holds past system input and output data, is usually allowed to fill up before the identification algorithm is started. However, for this particular system, it was found that the parameter estimates exhibited better convergence if the regression vector was initialised using data from the previous sample only (cf. Equation 4.7) :-

$$\psi^{T}(0) = [\, y(-1), 0, 0, u(-1), 0, 1 \,] \tag{5.1}$$

The parameter estimates shown in Figure 5.3 were derived using this method of initialisation.

The effect of these two different methods of initialisation on this particular system is shown in Figure 5.6, which gives the a_1 parameter estimate for both cases (cf. a_1 in Figure 5.3). The two parameter estimates converge to different values and are also more irregular when $\psi(0)$ is fully filled. The other parameter estimates exhibit similar behaviour. When the estimator is coupled with an appropriate pole placement controller, stability problems have been observed when $\psi(0)$ is fully filled, which results from the higher and more erratic controller gains generated.

The effect of different forgetting factors, λ, on the a_1 parameter estimate is shown in Figure 5.7. There is little difference between $\lambda = 0.995$ (cf. Figure 5.3) and $\lambda = 0.99$, however with $\lambda = 0.95$, the parameter estimate becomes erratic due to the increased sensitivity to noise [2.57]. Similar effects are seen with the other parameter estimates.

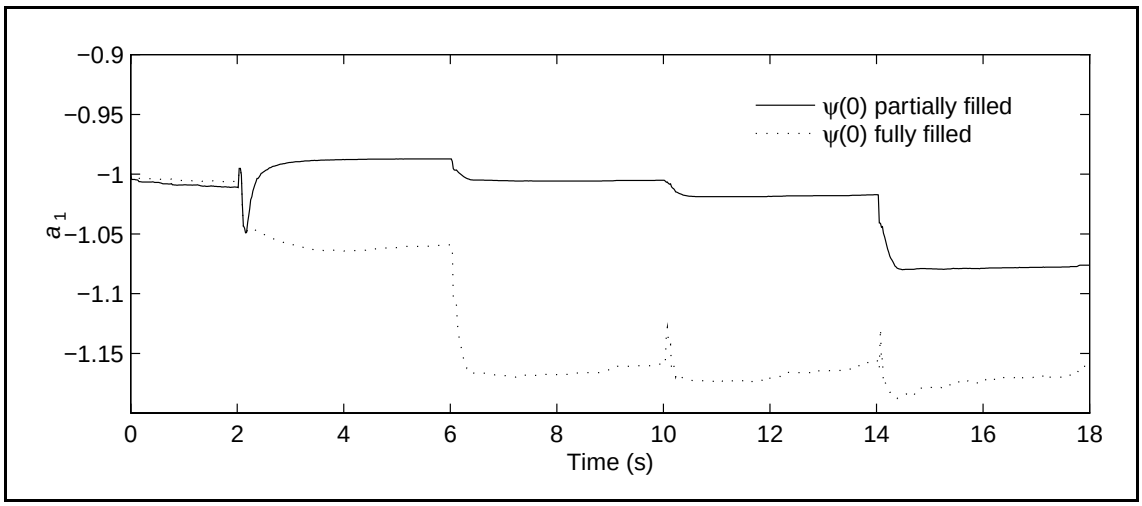

Figure 5.6 Effect of using Different $\psi(0)$ on the a_1 Parameter Estimate

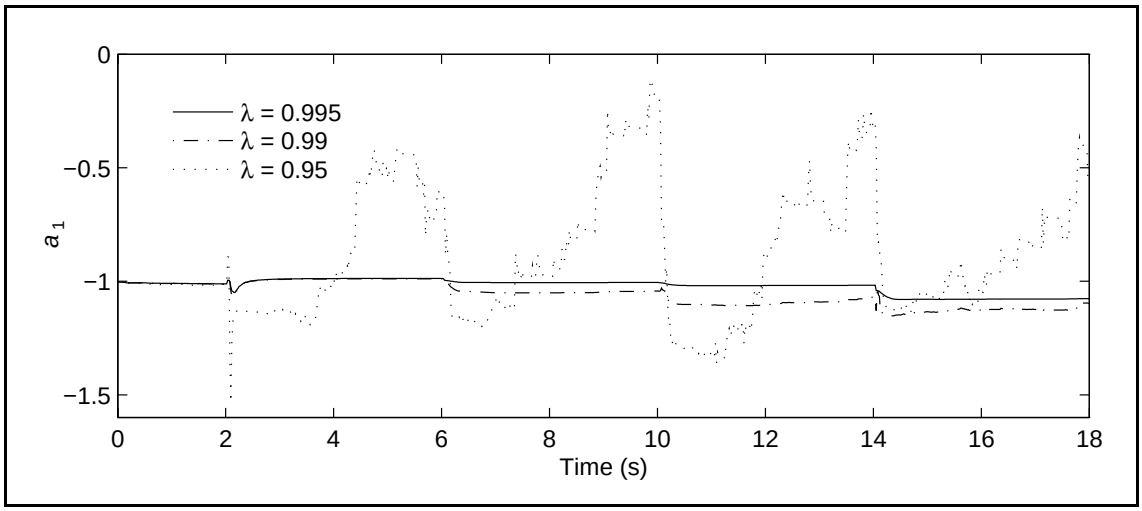

Figure 5.7 Effect of using Different λ on the a_1 Parameter Estimate

The identification results shown in Figure 5.3 do not use a priori knowledge about the system being identified, with the initial parameter estimates corresponding to a simple integrator, Equation 4.21. The time taken for the parameters to converge can be reduced by using a priori initial parameter estimates obtained from previous identification runs. Figure 5.8 shows the same parameters being identified as in Figure 5.3, but with $\Phi(0)$ equal to the average of the parameter estimates in Figure 5.3 from 3 s to 18 s. The initial covariance matrix, $P(0)$, is set to I accordingly. The axis limits used in Figure 5.8 are the same as those in Figure 5.3 to allow direct comparison of the plots. As expected the parameter estimates are much smoother during the initial three seconds than those obtained without the use of a priori parameter estimates. After this, the two sets of estimates follow similar trajectories.

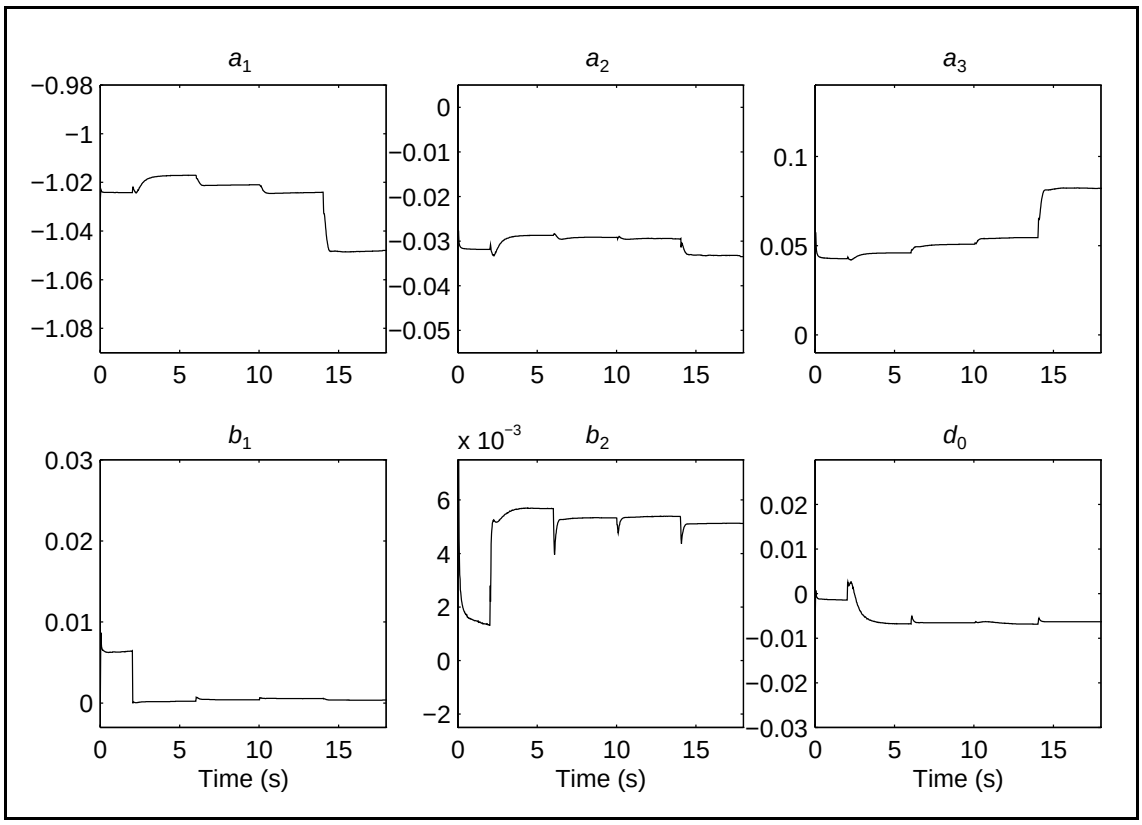

Figure 5.8 Effect of using a priori Parameter Estimates

5.3.2 Effect of Different Identification Algorithms

Many different forms of recursive least squares (RLS) algorithm exist, and three particular types have been investigated in this study :-

- Matrix Inversion Lemma (MIL-RLS) : this is the traditional recursive least squares algorithm, and was presented in Section 4.2.2, Equations 4.8 to 4.11.
- Bierman U-D Factorisation (BUD-RLS) : a numerically robust recursive least squares algorithm, Appendix B, Equations B.2 to B.8. This prevents rounding errors from affecting the parameter estimates, and also ensures that the covariance matrix remains positive definite, which is a requirement for convergence.
- Simplified Matrix Inversion Lemma (EASY-RLS) : this is identical to the MIL-RLS but with only the diagonal elements of the covariance matrix, $P(k)$, being updated, thereby reducing the computational requirements, as proposed in [2.79].

When operating as stand-alone algorithms, different identification methods can produce slightly different results [2.82]. Here, the MIL-RLS and BUD-RLS estimates are almost identical, both with RMS a priori prediction errors of 1.72×10^{-3} V. The estimates from the EASY-RLS algorithm converge to slightly different values, giving an RMS prediction error of 3.03×10^{-3} V. The difference in performance between these algorithms is more marked when coupled with the self-tuning controller, as discussed in Section 5.5.2.

The relative computational effort required by each algorithm can be determined using MATLAB's flops command, which measures the number of floating point operations performed. The algorithms running under MATLAB have specifically been written to exploit the vectorisation techniques available to increase their speed and simplicity. However, these algorithms cannot be applied directly to the DSP which is programmed in 'C' code. Therefore Table 5.1 shows the number of floating point operations (flops) for the three different algorithms using both the vectorised MATLAB code as well as the 'C' code equivalent. The execution time for each algorithm is also given, which was determined by averaging the time taken for 960 iterations of the algorithm on a 166 MHz Pentium PC. The model structure used was the same as used previously, that is $n_a = 3$, $n_b = 2$ and $n_d = 1$.

Identification Algorithm	flops	execution time (ms)
MIL-RLS	764	2.03
EASY-RLS	764	2.38
BUD-RLS	1082	3.57
MIL-RLS ('C' code equivalent)	871	31.68
EASY-RLS ('C' code equivalent)	211	10.60
BUD-RLS ('C' code equivalent)	180	10.32

Table 5.1 System Identification Computational Requirements

Looking at the 'C' code equivalent algorithms first, the EASY-RLS clearly has a much smaller computational requirement than the standard MIL-RLS algorithm. However,

the BUD-RLS has the lowest requirement. This is at odds with [2.57] which states that equally efficiently coded MIL-RLS and BUD-RLS algorithms will take approximately the same number of computations. The execution times for the compiled 'C' code running on the DSP vary in proportion to those values in Table 5.1, showing that assembler optimisations are consistent across the three algorithms.

The MATLAB specific algorithms use more floating point operations than their 'C' code equivalents, however the vectorised nature of these algorithms means that they do execute significantly faster. It should also be noted that the BUD-RLS algorithm is slower than both the MIL-RLS and EASY-RLS schemes. This arises since it cannot be vectorised to the same degree as the other algorithms.

The number of estimated parameters, n_Φ, has a nonlinear influence on the number of floating point operations. For both the BUD-RLS and EASY-RLS algorithms this is approximately proportional to n_Φ^2, and for the MIL-RLS it is roughly proportional to n_Φ^3. The reason for these differences can be directly attributable to the 'C' code implementation of each algorithm, and specifically the number of nested for-next loops present.

5.3.3 Effect of Different Model Orders

The model order and structure used in the above examples is based on the linearisation of the underlying physical system discussed in Section 4.3. However, lower order models may work just as well with the benefit of lower computational effort, alternatively higher order models may offer improved accuracy. One way of determining the most effective model order is to look at how the RMS a priori prediction error varies with n_a, n_b and n_d. This is shown in Figure 5.9 for models up to fifth order, $n_a = n_b = 5$, with Figures 5.9a and 5.9b corresponding to $n_d = 0$ and $n_d = 1$ respectively.

The various models all follow the actual system output closely, but it is clear from Figure 5.9 that the higher order models provide better tracking of the system. This is as to

be expected since the higher order terms allow higher order dynamics, and noise to a certain extent, to be captured in the model and hence provide the marginal improvements shown. Comparing Figures 5.9a and 5.9b, the inclusion of an offset term in the model, n_d = 1, reduces the modelling error significantly. This justifies the argument in Section 4.3 for including this term in the linearised manipulator model.

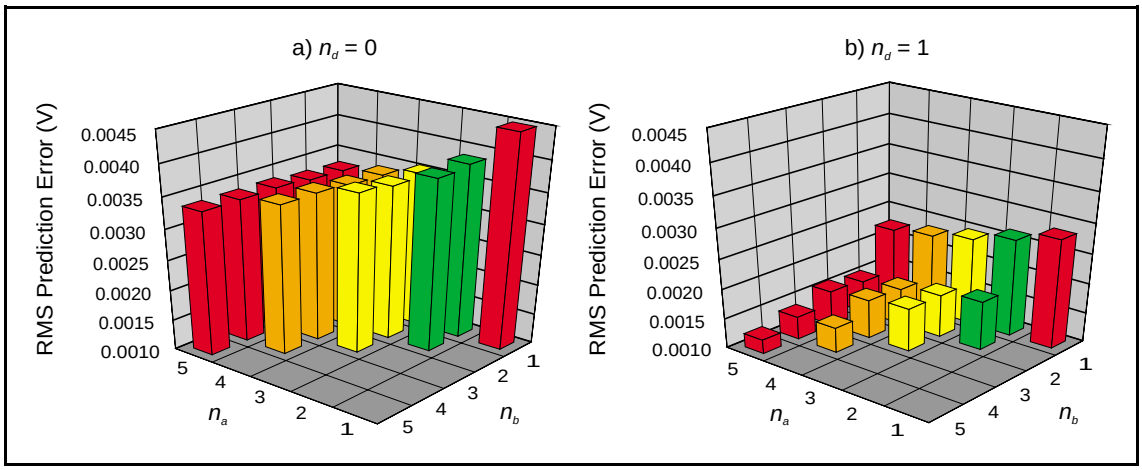

Figure 5.9 Effect of Model Order on Prediction Error

The relationship between model order and number of calculations has already been discussed, and therefore there is an obvious trade-off between prediction error and computational requirement. Consequently the work here uses model orders corresponding to the underlying system (n_a = 3, n_b = 2 and n_d = 1) and also those corresponding to self-tuning PI and PID controllers.

5.4 Off-line PID Tuning

Self-tuning controllers use the parameter estimates generated by the identification algorithm to determine their gains so as to meet some predefined performance criterion. Therefore, the gains are automatically adjusted to accommodate any changes in system dynamics that arise from variations in the operating conditions. Another way in which the parameter estimates can be used to design a controller is to determine the *average* controller gains from a series of identification results. These values are then used in a *fixed gain*

controller, which has the advantage of computational simplicity but cannot cope with changes in dynamics. This procedure provides a simple means of tuning fixed gain controllers, with the desired polynomial, $t(z^{-1})$, determining the system response (when operating under typical conditions).

This procedure was followed using model orders of $n_a = 2$, $n_b = 1$, $n_d = 0$, corresponding to a self-tuning PID controller, and $n_a = 1$, $n_b = 1$, $n_d = 0$, matching a self-tuning PI controller, Appendix C.2. As in the previous work, the forearm rotate joint was stepped between 210° and 170° with a period of 8 s, but the gains used were $k_p = 10.0$ and $k_i = 10.0$ and the experiment continued until t = 80 s. The system identification was configured as described in Section 5.3, with $\lambda = 0.995$, no prior knowledge of parameter estimates assumed, and $\psi(0)$ filled with data from the previous sample only (Equation 5.1).

The parameter estimates obtained from both model orders are then used to determine the corresponding controller gains, as detailed in Appendix C.2, using Equations C.7 and C.8. The desired polynomial was specified to give two poles at 0.95, so :-

$$t(z^{-1}) = 1 - 1.9z^{-1} + 0.9025z^{-1} \tag{5.2}$$

Figure 5.10 shows the resulting proportional gains for both the PI and PID controllers, with the integral and derivative gains having similar variations. It can be seen

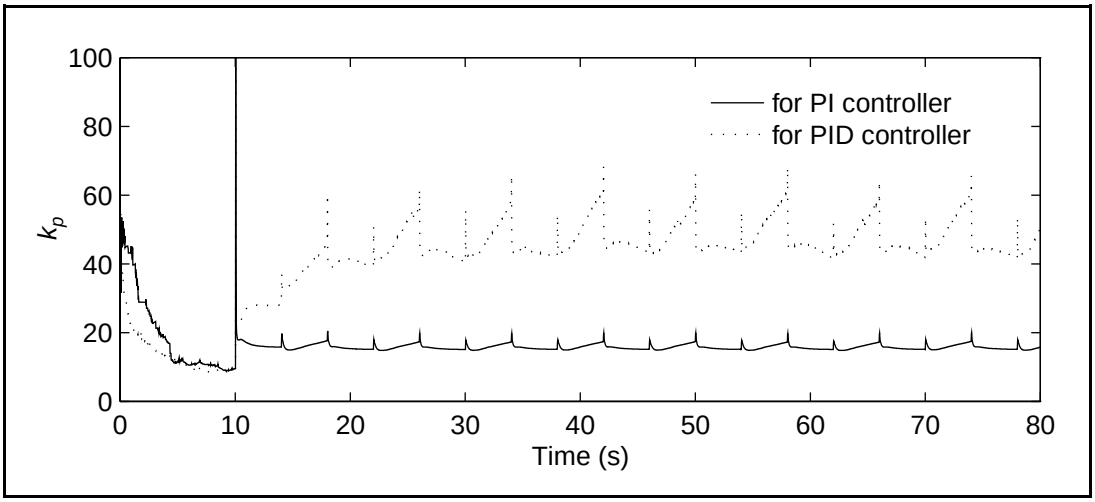

Figure 5.10 Estimated Proportional Gains for PI and PID Controllers

that the required controller gains vary during this simple movement, which is due to the changes in system dynamics as discussed earlier. The fixed gain controller terms are obtained by averaging the estimated gains between 20 s and 80 s, to neglect the period of initial convergence, and are given in Table 5.2.

	k_p	k_i	k_d
PI controller	15.73	20.38	-
PID controller	46.63	61.15	6.71

Table 5.2 Controller Gains Obtained by System Identification

The response of both fixed gain PI and PID controllers using these gains for steps in the forearm rotate joint are shown in Figures 5.11a and 5.11b respectively (cf. Figure 5.1). The *desired response* shown in these plots corresponds to that specified by the desired polynomial of Equation 5.2. The joint angle attempts to follow the desired response in both cases, with the PID controller providing better tracking. This is reflected in the RMS errors between desired and actual joint angle over the 18 s period of this test, which for the PI controller was 6.4° whereas the PID error was 2.9°.

The gains obtained for the PID controller are high, and place a high demand on the

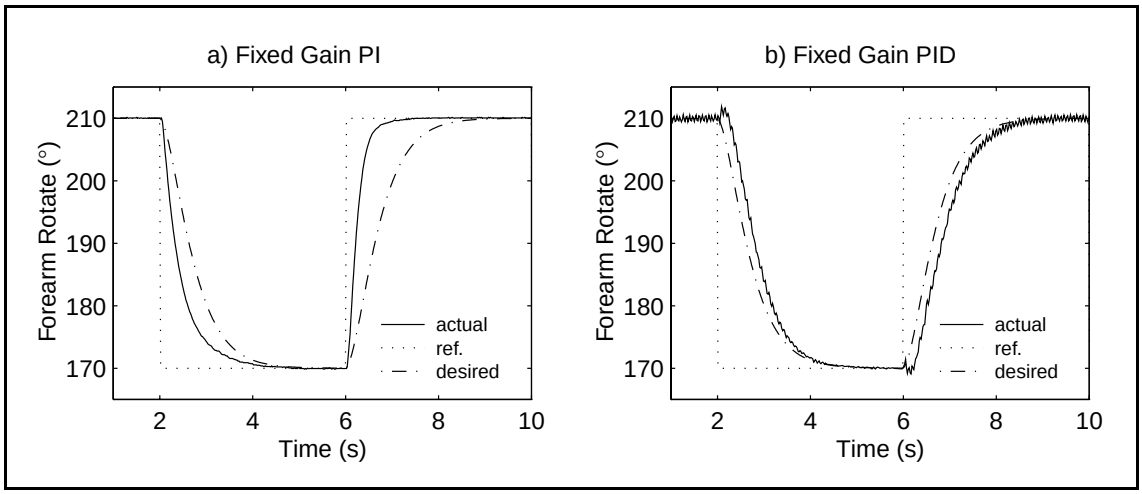

Figure 5.11 Response of Fixed Gain PI and PID Controllers

actuators since the controller output signal readily saturates at its limits. This is impractical as servovalve wear and failure is greatly increased under these conditions. Therefore, the fixed gain PI controller is used as the benchmark to which the subsequent self-tuning joint controllers are compared.

5.5 SISO Self-tuning Pole Placement Joint Angle Control

This section discusses the operation and performance of self-tuning joint angle controllers for a variety of different operating conditions, showing benefits over the fixed gain controllers derived in the previous section. The operational issues of RLS system identification explored in Section 5.3 will be investigated again to look at the consequence of use within a self-tuning controller.

The self-tuning controller developed here operates on the forearm rotate joint stepping from 210° to 170°, as used in the earlier system identification work. The initial model order used corresponds to the physical system, with $n_a = 3$, $n_b = 2$ and $n_d = 1$, and the Matrix Inversion Lemma RLS algorithm (MIL-RLS) is employed, with a forgetting factor, $\lambda = 0.995$. The initialisation of the RLS is as described previously; $\Phi(0)$ is assumed to be an integrator, $P(0) = 100\mathrm{I}$, and $\psi(0)$ is partially filled with data from the previous sample.

The system identifier is started at 0 s, and the step from 210° to 170° in the forearm rotate is executed at 12 s. Proportional gain control is used for the first 10 samples (0.196 s) after the step is commanded, as this ensures that the self-tuning controller is well-behaved, which is switched on at 12.2 s. This switching is facilitated by the use of incremental controllers which allow bumpless transfer between different control strategies, as mentioned in Section 4.2.3.

The response of the forearm rotate joint using this self-tuning controller is shown

in Figure 5.12, together with the desired response given by Equation 5.2[†], where higher order terms in $t(z^{-1})$ are set to zero. It can be seen that the controlled joint angle follows the desired response closely, with the RMS error between them being 0.59°, showing significant improvement over the fixed gain controllers used in the previous section. The controller output is shown in Figure 5.13.

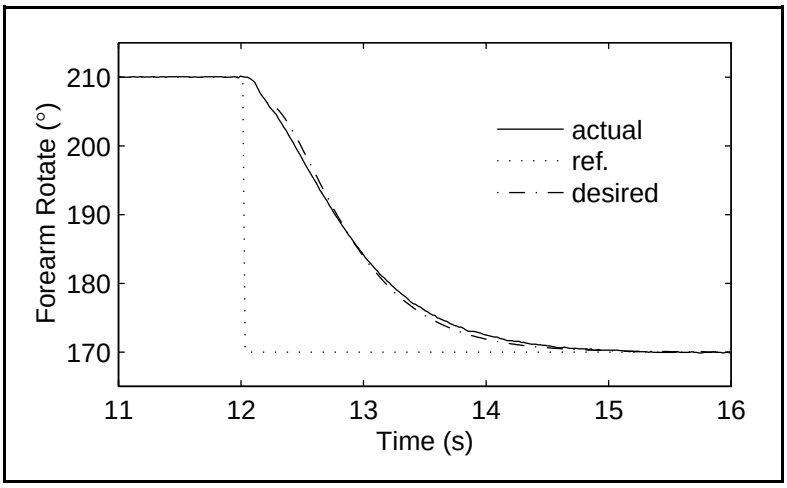

Figure 5.12 Response of Self-tuning Controller

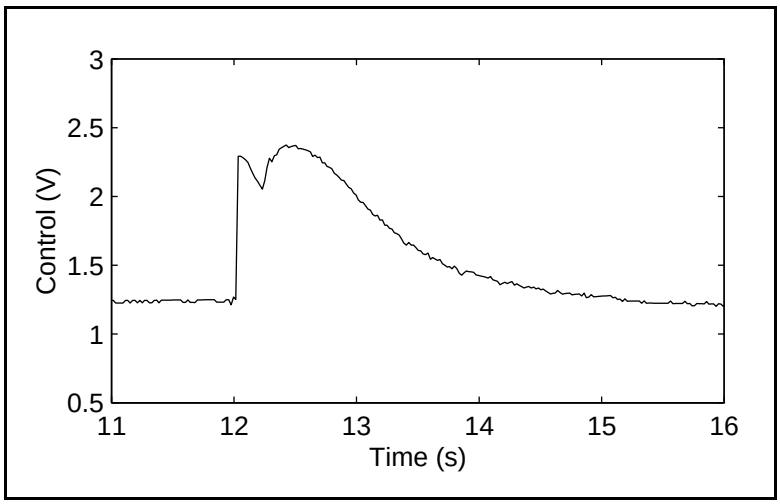

Figure 5.13 Self-tuning Controller Output

The use of step changes in the reference generally results in large control signals, which is potentially problematic for self-tuning controllers. However, step responses provide a good means of quantifying controller performance. In practice smoother

[†] The desired response, and hence error measurements, commence at 12.2 s since they are only applicable to the portion of the response under self-tuning control.

trajectories would be more appropriate, which are less demanding on the controllers being used. Furthermore, when performing a task the robot may be motionless for long periods of time, this may lead to problems due to lack of persistent excitation. The covariance management techniques described in Section 4.2.4 can be used to alleviate this. For the purpose of the experimentation in this thesis no covariance management is employed.

Figure 5.14 shows the response of the forearm rotate joint for a variety of different desired polynomials, including an underdamped one, due to the specification of complex conjugate poles. Again, the period of proportional only control can be clearly seen on the plots as it is identical for each response. The joint follows the desired responses closely in all four cases, but some degradation was observed for d) which was due to the desired response requiring a slew rate faster than the manipulator could attain. Similarly, specifying two poles at 0.75 proved too fast for the manipulator to achieve, with the controller output saturating. These results do show the ease with which the designer can change the system response through specifying appropriate pole locations.

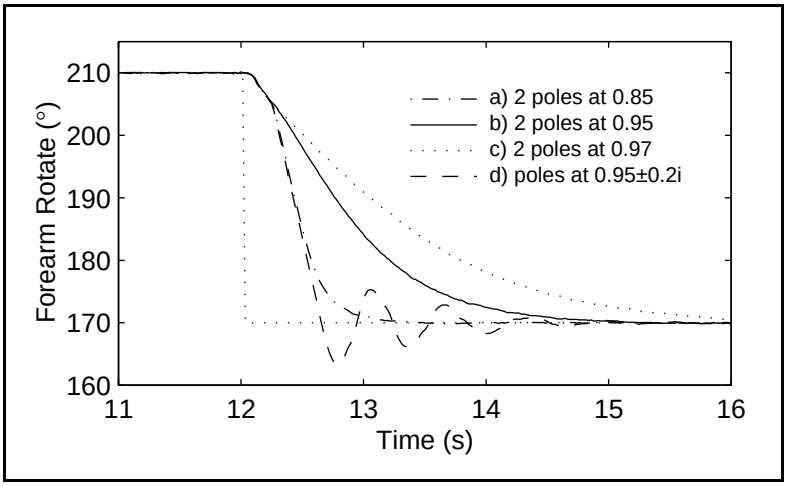

Figure 5.14 Response for Different Desired Polynomials

5.5.1 Effect of Different Model Orders

Here, the effect of using different model orders in the self-tuning controller is investigated, in terms of both controller performance and computational complexity of the

resulting controller algorithm. The test conditions and operating procedures of the self-tuning controller are the same as those used in the previous example. Table 5.3 gives the RMS error between desired and actual responses for several different model orders, which is calculated as previously detailed.

Controller Type	n_a	n_b	n_d	RMS error(°)
Self-tuning PI (see Appendix C.2)	1	1	0	1.122*
	1	1	1	1.126*
Self-tuning PID (see Appendix C.2)	2	1	0	1.233
	2	1	1	1.119*
Self-tuning pole placement based on physical model	3	2	1	0.594

Table 5.3 Self-tuning Controller Errors for Different Model Orders

The results marked with an asterisk (*) in above table were obtained by starting the self-tuning controller after 20 samples of proportional only control, rather than 10 samples as used earlier. Though these results are complicated by these slightly different procedures used, it is clear that the model representing the underlying physical system is the best. Furthermore, there is little difference in performance between the self-tuning PI and PID controllers.

The number of floating point operations required for the various controllers used are given in Table 5.4. These figures were determined using MATLAB's flops command, and applies to the 'C' code implementation of the controllers rather than the vectorised MATLAB code, as described in Section 5.3.2. Clearly, the computational burden of a self-tuning controller increases with model order, however, the number of computations required by the controller is secondary when compared to that required by the identification algorithm (cf. Section 5.3.2 for $n_a = 3$, $n_b = 2$, $n_d = 1$).

It should be noted that non-incremental self-tuning controllers were experimented

with for control of the forearm rotate joint, but the responses obtained did not follow the desired response and quickly became unstable.

Controller Type	n_a	n_b	n_d	flops
Fixed Gain PI	-	-	-	7
Fixed Gain PID	-	-	-	12
Self-tuning PI	1	1	0 or 1	12
Self-tuning PID	2	1	0 or 1	19
Self-tuning	3	2	0 or 1	81

Table 5.4 Self-tuning Controller Computational Requirements

5.5.2 Effect of Different Identification Algorithms

The effect of using different system identification algorithms within the self-tuning controller are now explored. The test conditions and operating procedures of the self-tuning controller are the same as those used in the previous examples, with the model order used being $n_a = 3$, $n_b = 2$ and $n_d = 1$.

The algorithms investigated are the MIL-RLS, BUD-RLS and EASY-RLS routines discussed in Section 5.3.2. The tests are performed over an extended period since the algorithms have implications in terms of numerical robustness, as well as dynamic accuracy. Therefore, the forearm rotate joint is stepped between 210° and 170° with a period of 8 s as before, but allowed to carry on until t = 98 s. The proportional only control, used prior to starting the self-tuner, is used only during the first step, since the self-tuning controller copes with subsequent step changes once it is has been initiated correctly.

The response of the forearm rotate joint under self-tuning control using these three different identification algorithms is given in Figure 5.15, for the period t = 18 s to t = 98 s. The MIL-RLS algorithm follows the desired response well, but does degrade at certain times and large spikes are evident in the response. These spikes probably arise due to the

covariance matrix becoming ill-conditioned. The BUD-RLS system follows the desired response closely with no perturbations, showing improved robustness over the standard RLS routine. Figure 5.15c shows the response when using the EASY-RLS algorithm and the desired response is not followed closely, with some overshoot present. Table 5.5 gives the RMS and peak errors between desired and actual responses for the period 18 s to 98 s, and these reflect the previous discussion.

The long term accuracy is significantly better for the BUD-RLS system identification, whereas the MIL-RLS follows the desired response well but the spurious spikes demonstrate its shortcoming. As can be seem from Figure 5.15c the performance of the EASY-RLS algorithm is poor, as it does not follow the desired polynomial.

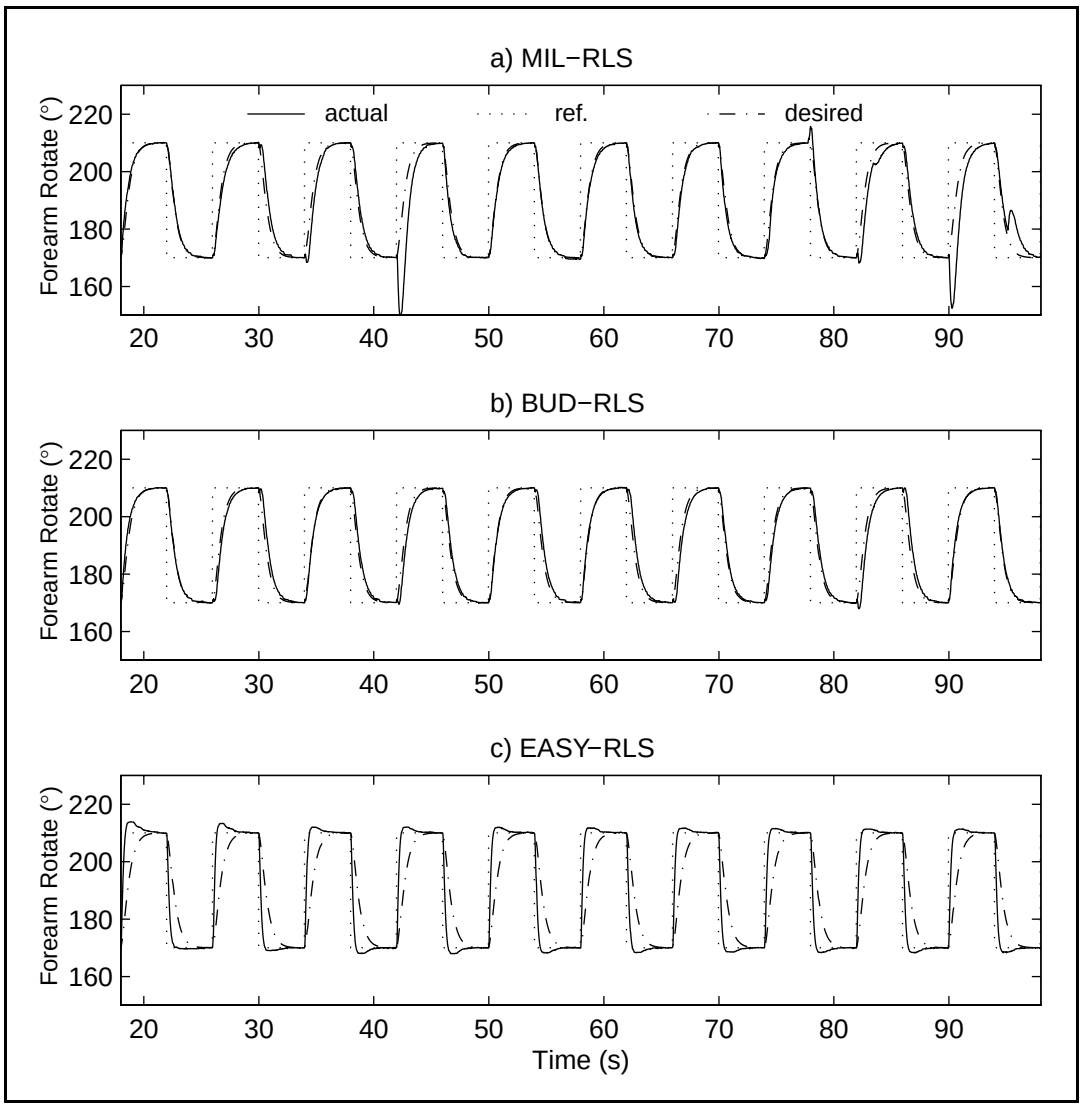

Figure 5.15 Effect of Using Different Estimation Algorithms

RLS Algorithm	RMS Error (°)	Peak Error (°)
MIL-RLS	4.625	33.115
BUD-RLS	2.314	11.090
EASY-RLS	10.067	28.703

Table 5.5 Controller Errors for Different RLS Algorithms

5.5.3 Effect of Different Operating Conditions

The results discussed in this section look at the ability of the self-tuning controller to adapt to changing conditions. Several different operating conditions are tried, using both fixed gain and self-tuning controllers to quantify the benefits of using these more advanced schemes. The operation of the self-tuning controllers is the same as described in Section 5.5, specifically $\lambda = 0.995$, $t(z^{-1})$ as given by Equation 5.2, $\Phi(0)$ assumed to be an integrator (i.e. no a priori knowledge assumed), $P(0) = 100I$, $\psi(0)$ is filled with data from one previous sample, and the MIL-RLS algorithm is used.

Three different controllers are studied :-

- the self-tuning pole placement controller introduced in Section 5.5, using a model order of $n_a = 3$, $n_b = 2$ and $n_d = 1$.
- a self-tuning PID controller with $n_a = 2$, $n_b = 1$ and $n_d = 0$.
- the fixed gain PI controller using gains derived from off-line system identification, as described in Section 5.4.

The different operating conditions on the forearm rotate joint are as follows :-

a) step from 210° to 170°, as used in the initial work in Section 5.5.

b) step from 210° to 170° with a 28 kg payload.

c) step from 170° to 210°, which acts in the direction of gravity as opposed to a) which acts against gravity.

d) step from 330° to 290°.

The response of the higher order self-tuning controller to these four conditions is shown in Figure 5.16, where the responses for c) and d) have been shifted to allow comparison on the same graph. These responses are virtually identical, all following the desired response closely, indicating that the self-tuning controller is able to adapt to and accommodate these changes in manipulator dynamics.

Figure 5.17 shows the results when using the self-tuning PID controller. Clearly there is some degradation when compared to the higher order self-tuner, but the responses are still reasonably close to the desired response. It should be noted that the controller used for response c) uses a disturbance term ($n_d = 1$), as the control was poor without one. The effect of using a standard fixed gain PI controller is given in Figure 5.18, and the degradation that arises from the changing manipulator dynamics is apparent. The response under condition c) is the one that is significantly different from the rest, and is attributable to the effects of gravity acting in a different direction in this case.

To quantify the performance of the three different controllers, the RMS error between the actual and desired responses are calculated for each of the four conditions and then averaged, giving :-

- self-tuning controller using $n_a = 3$, $n_b = 2$ and $n_d = 1$, has an average error of 0.88°.
- self-tuning PID controller has an average error of 1.29°.
- fixed gain PI controller has an average error of 5.81°.

Further tests were carried out with the manipulator submerged, to determine if

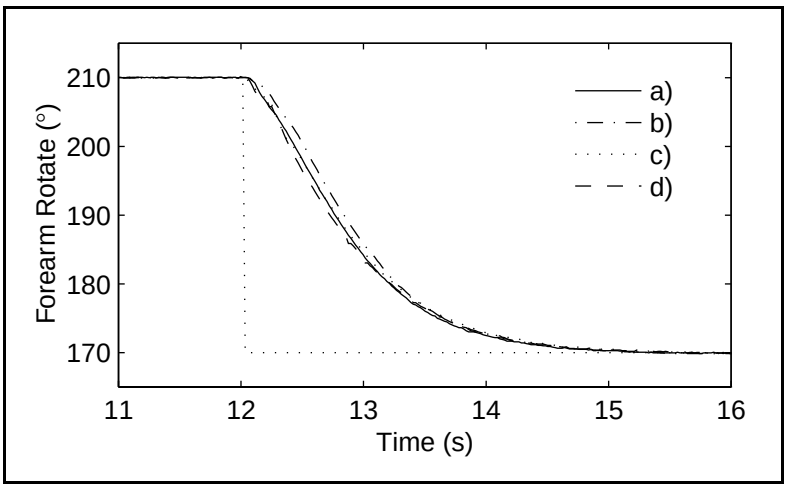

Figure 5.16 Response of Self-tuning Controller (n_a = 3, n_b = 2, n_d = 1) under Conditions a) to d)

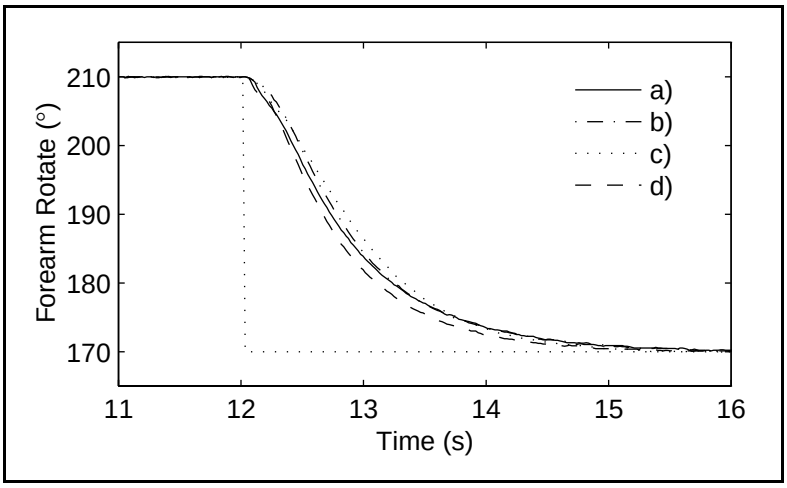

Figure 5.17 Response of Self-tuning PID Controller under Conditions a) to d)

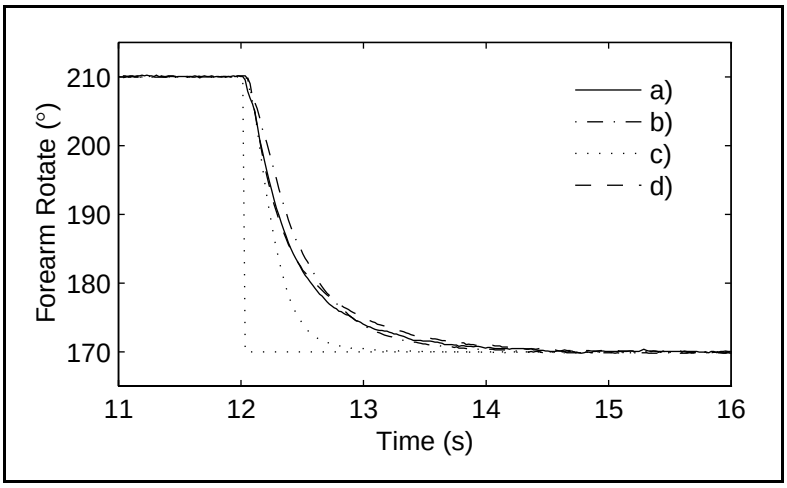

Figure 5.18 Response of Fixed Gain PI Controller under Conditions a) to d)

hydrodynamic effects would influence the controller response. However, little difference was observed between operation in air and fully submerged operation [1.15]. This confirms claims made in [1.2] that the hydrodynamic effects of typical subsea manipulators are insignificant when the manipulator is moving at low velocities. This justifies the omission of these hydrodynamic effects from the dynamic manipulator model developed in Chapter Three.

The different conditions examined above embrace only a small range of the TA9s capabilities, this being due to the limited scope of laboratory tests. However, improvements in performance should be even more significant at the extremes of the TA9's operational range, such as payload and operating temperature, that are experienced during actual offshore operation.

5.5.4 Static Accuracy of Self-tuning Controllers

The previous work looked at the dynamic performance of the manipulator joint when responding to a step change command. This section explores the effect of self-tuning control on static accuracy.

With the forearm rotate joint held constant at 210°, the fixed gain PI controller used previously gave an RMS error of 0.13° over a period of 30 s. The self-tuning controller developed shows improvement on this, giving an RMS error of 0.098°.

This test was repeated for the elbow joint, since this exhibits poor static accuracy due to limit cycles which arise from the combination of controller integral action and mechanical stiction in the hydraulic piston [5.2]. Pistons are used on four of the six joints of the TA9, but this effect is not observed on the forearm rotate joint as it uses a rotary vane actuator. These limit cycles are shown in Figure 5.19a, which shows the elbow being held constant at 40° under fixed gain PI control (using the devised gains of k_p = 7.0 and k_i = 2.0) giving an RMS error of 0.69°. The response when self-tuning control is used is given in

Figure 5.19b, and shows a significant improvement in reducing the limit cycles, resulting in an RMS error of 0.14°.

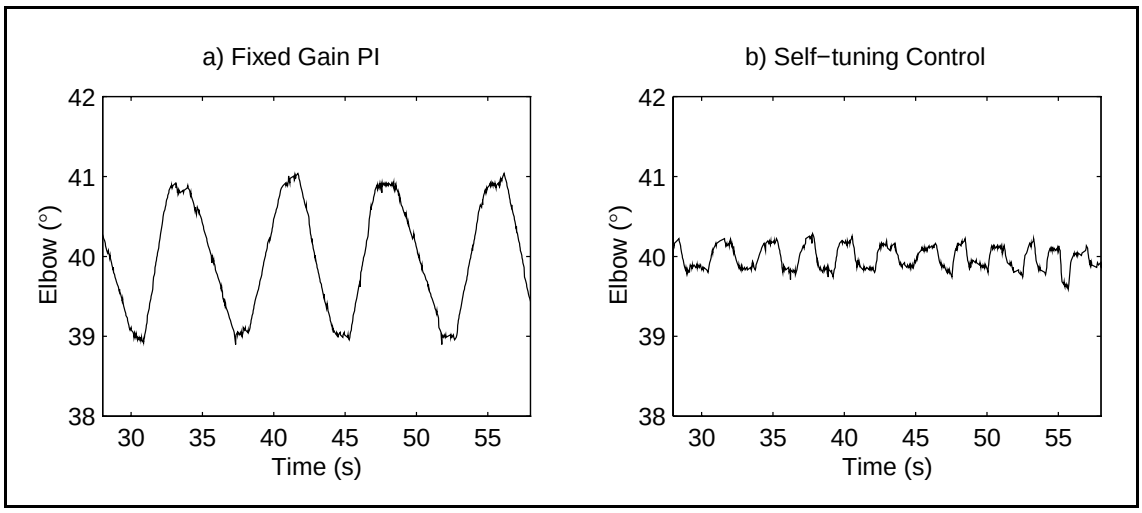

Figure 5.19 Static Response of Elbow Joint under Fixed Gain and Self-tuning Control

An interesting effect is that these limit cycles grow in size as the robot heats up, increasing considerably after about 30 minutes of operation, reducing the robots static accuracy dramatically. Tests were carried out that eliminated heating of electrical components from causing this effect. It was eventually deemed that the likely cause was a reduction in bulk modulus of the hydraulic fluid used in the manipulator as its temperature increases, Table 3.1. This makes the hydraulic fluid more compressible, making the manipulator responses become more oscillatory [5.3]. The seals and other mechanical components in the hydraulic system may also contribute to this degradation as they heat up. The effect that gives rise to these limit cycles is not included in the dynamic model developed in Chapter Three.

5.5.5 Effect of Coupling Between Joints

This section investigates the effect that coupling between joints has on the performance of fixed gain and self-tuning control schemes. It has already been stated that for these independent joint controllers, coupling is treated simply as a disturbance which

must be rejected.

The elbow joint is used, and as in the previous section is held constant at 40°, but now rapid movements in the neighbouring wrist joint are commanded to produce coupling. These movements are a 40° peak-to-peak amplitude sinusoid with a frequency of 1.7 Hz, and also a 40° amplitude square wave at 0.45 Hz.

The response of the elbow under fixed gain PI control actually shows some improvement in static accuracy under these conditions of coupling, with smaller limit cycles giving an RMS error of 0.51°. This reduction is attributable to the small forces produced by the coupling, which decreases stiction at the joint as it is never perfectly motionless.

The same test performed using self-tuning control also showed improvements, but this is probably as a result of improved persistent excitation for the RLS estimation algorithm. Small perturbations that were observed sporadically in joint angle in the purely static case, are removed altogether as the coupling provides some, albeit small, excitation to the joint. This yielded an RMS error of 0.15°. Similar effects were also observed when these tests were repeated on the forearm rotate joint.

5.6 Summary

This chapter has presented a series of results that explored the operational aspects of self-tuning controllers when applied to the independent joint control of a typical offshore hydraulic manipulator. The benefits of using self-tuning controllers over conventional fixed gain controllers have been shown, in terms of both dynamic and static accuracy.

First, the experimental setup was described, detailing how the digital controllers are interfaced to the robot. The links with the higher level system, such as motion and task planners, were also briefly described.

The operational aspects of practical system identification were then investigated.

This encompassed different initialisation strategies, choice of forgetting factor, and the effect of different model orders. The investigation demonstrated the suitability of using the linear model structure determined from the underlying physical system, presented in Chapter Four. Three alternative RLS algorithms were also tested, in terms of accuracy and computational requirement.

The use of off-line system identification to obtain appropriate gains for fixed gain PI and PID controllers was then discussed. The resulting fixed gain controllers were then used as the benchmark for comparison with the self-tuning controllers.

The operation of self-tuning controllers was discussed, again looking at proper initialisation, effects of different model orders and the different RLS algorithms. It was found that for such practical applications the Bierman U-D Factorisation RLS algorithm should be used to ensure good behaviour.

The performance of the self-tuning controllers was compared to the fixed gain controllers, over a series of different conditions. This included investigating the effect of different manipulator configurations and different payloads. Self-tuning control was also examined in terms of static accuracy, ability to cope with coupling between joints and the effects of hydrodynamics. These results demonstrated the ability of adaptive control strategies to cope with unknown and changing conditions.

Chapter 6

Fixed Gain Hybrid Position/Force Control

6.1 Introduction

The introductory chapter of this thesis highlighted the need for the automation of certain subsea intervention tasks, requiring both automatic and accurate position control of underwater manipulators. Furthermore, control of the forces arising from interactions between the robot and objects in the workplace greatly increases the capabilities and efficiency of such systems. For example, tasks such as the mating of subsea connectors and NDT weld inspection can be realised using automatic force control. A hybrid position/force control scheme (discussed in Section 2.3.3) was chosen to facilitate such tasks, as this strategy allows the simultaneous control of force and position in orthogonal directions in the workspace.

Much of the previously reported work on hybrid position/force control has focused on theoretical issues such as robustness and performance, with little consideration for the practical aspects of the problem. Where experiments have been performed, they have almost exclusively used specialised laboratory robots which have very different characteristics to the manipulators used by the offshore industry. Consequently, if hybrid position/force controllers are to be successfully employed offshore the limitations imposed by hydraulic actuator technology and real-time implementation need to be addressed.

Therefore, the main objective of the work presented in this chapter is to establish the technical feasibility of applying a practical hybrid position/force controller to a typical

offshore manipulator. Fixed gain PID controllers are used to give a relatively simple realisation, allowing practicalities to be addressed prior to the implementation of more advanced controllers [1.5]. The limitations of this fixed gain system when operating under unknown and changing conditions are examined. Furthermore, this system forms the benchmark against which the self-tuning hybrid position/force controller developed in Chapter Seven is compared. Although the fixed gain schemes have limitations, they still significantly enhance the capabilities of these offshore manipulators.

This chapter starts by reintroducing the hybrid control scheme first discussed in Section 2.3.3, defining the specific controller used and how it is implemented on the TA9 manipulator described in the previous chapter. Results from the experimental hybrid position/force controller are then presented for a variety of different conditions to investigate the performance of the system. The chapter concludes with a brief look at the realisation of some typical offshore tasks using the developed fixed gain hybrid position/force control scheme.

6.2 Practical Fixed Gain Hybrid Position/Force Control

The hybrid position/force controller implemented uses fixed gain PID controllers within the generalised framework of Figure 2.3, and as presented in the example in the definitive paper by Raibert and Craig [2.12]. Such fixed gain schemes do have limitations, yet this form of hybrid position/force controller provides a simple realisation that significantly enhances the capabilities of current offshore manipulators.

The force controller and position controller are realised as two separate controllers which enables them to be designed independently to reflect these two different control problems. The force control loop is realised as an explicit force controller (see Figure 2.2a), with the Cartesian force errors being transformed into the joint space by the *Jacobian transpose*, and then acted upon by PID controllers. The Cartesian position controller is

similarly realised using PID controllers operating in the joint space, with the Cartesian position errors being transformed into joint space errors by the *inverse Jacobian*, see Figure 2.1b. The relative merits and drawbacks of these particular force and position control schemes have been discussed in Sections 2.3.3 and 2.3.2 respectively. The resulting hybrid position/force controller is shown in Figure 6.1 (cf. Figure 2.3).

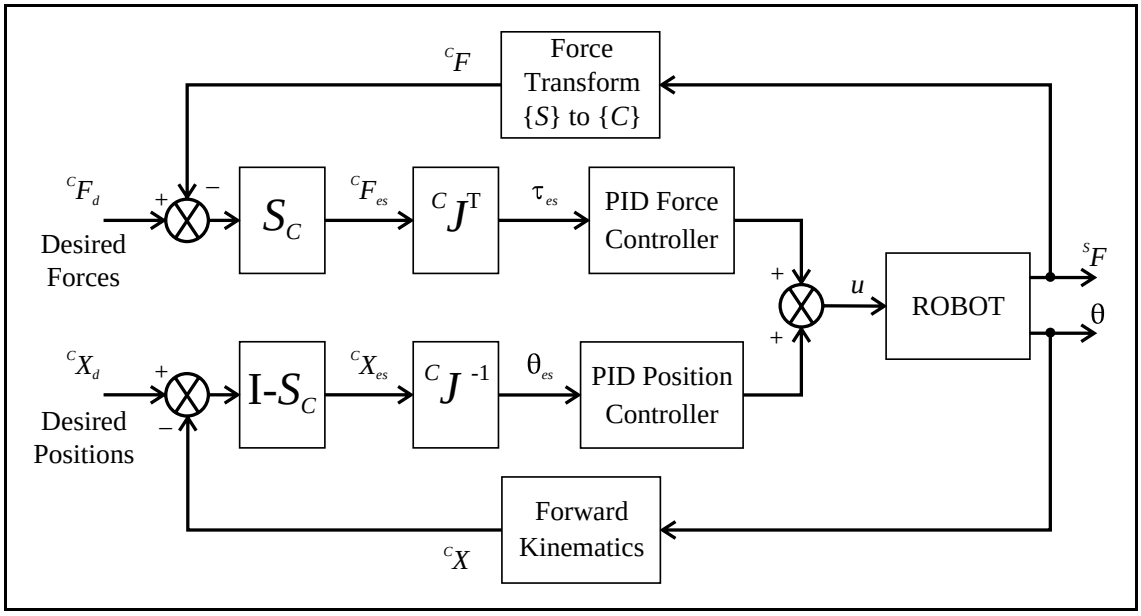

Figure 6.1 Fixed Gain Hybrid Position/Force Control using Joint Space PID Controllers

As described previously, the compliance selection matrix, S_C = diag[s_1, s_2, .., s_n], determines which directions in the constraint frame, {C}, are position controlled (s_i = 0) and which are under force control (s_i = 1), where $i \in \{1, 2, .., n\}$. Therefore, the errors formed by the selection matrices, ${}^{C}X_{es}$ and ${}^{C}F_{es}$, are zero in the constrained and unconstrained directions respectively.

The position and force errors in the constraint frame, {C}, are both transformed into the joint space of the manipulator using the Jacobian, ${}^{C}J$, which is a function of the current joint angles and manipulator link lengths. By definition the Jacobian relates the joint angle velocities to the Cartesian end-effector velocities, as specified by Equation 3.3. This relationship is linearised and inverted to map small Cartesian position errors, ${}^{C}X_{es}$, to joint angle errors, θ_{es} :-

$$\theta_{es} = {}^{C}J^{-1}\,{}^{C}X_{es} \tag{6.1}$$

Similarly, the force errors, ${}^{C}F_{es}$, are transformed to joint torque errors, τ_{es}, by the Jacobian transpose (Equation 3.4) which appears in the hybrid control scheme as :-

$$\tau_{es} = {}^{C}J^{T}\,{}^{C}F_{es} \tag{6.2}$$

These errors in the joint space are then acted upon by independent PID controllers, with *n* controllers for the position errors and *n* for the force errors. The outputs from each position and force controller for each joint are then summed and applied to the appropriate actuator. Consequently each actuator contributes to both the position and force responses, and it is this that leads to coupling between the two control loops.

Two further transformations are shown in Figure 6.1; the *forward kinematics,* given by Equation 3.1, and a force transformation that converts forces measured in the *sensor frame*, ${}^{S}F$, to the constraint frame, ${}^{C}F$. The generalised transformation of forces and torques from sensor frame to constraint frame, is given by [2.1] :-

$${}^{C}F = \begin{bmatrix} {}^{C}_{S}R & 0 \\ {}^{C}P_{SORG} \times {}^{C}_{S}R & {}^{C}_{S}R \end{bmatrix} {}^{S}F \tag{6.3}$$

where ${}^{C}_{S}R$ is the rotation matrix that describes the rotation of $\{S\}$ relative to $\{C\}$ and ${}^{C}P_{SORG}$ is the position vector which locates the origin of $\{S\}$, usually located near the end-effector of the manipulator, relative to $\{C\}$.

6.3 Hybrid Position/Force Control Applied to the Restricted TA9

The fixed gain PID hybrid position/force controller described above is applied to the right-handed restricted TA9 manipulator, described in Section 3.3 and shown in Figure 3.2, which yields a two DOF manipulator acting in an horizontal plane. This enabled the

controllers to be developed and practical problems to be addressed without undue complexity associated with a full six DOF system. This restricted configuration is realised on the TA9 by holding four of its joints at constant angles. A mechanical limit on the wrist joint introduces a third, albeit fixed, joint angle, θ_3, into the kinematic analysis as discussed in Section 3.3.

6.3.1 Coordinate Systems and Transformations Used

The forward kinematics and Jacobian of this particular arrangement have already been defined by Equations 3.1 and 3.2 respectively. The constraint frame, $\{C\}$, was specified to coincide with the base frame of the manipulator, as shown in Figure 3.2. The location of the constraining surface is also shown in Figure 3.2 and is parallel to the *y*-axis of $\{C\}$. The selection matrix is therefore defined as S_C = diag[1, 0] for this particular arrangement.

The six axis force/torque sensor (ATI Model 150/600) is mounted between the claw rotate joint and the wrist joint. The forces and torques measured by this sensor are defined in the sensor frame, $\{S\}$, which is located 257 mm from the tip of the end-effector. The transformation of these forces into the constraint frame is accomplished in two stages. Firstly, the transformation from $\{S\}$ to frame $\{4\}$ is performed within the ATI transducer controller as it is independent of the manipulator's joint angles. The transformation uses the general expression of Equation 6.3 since both rotations and translations are required to move from $\{S\}$ to $\{4\}$[†].

The second transformation, from frame $\{4\}$ to $\{C\}$, is performed by the hybrid position/force controller software since it requires the current joint angles. This

[†] This transformation is specified within the ATI transducer controller in terms of XYZ Euler Angles which determines the specific order of rotations used to define the transformation. If the correct order is not observed then the transformation will be incorrectly specified. The ATI transducer controller manual [6.1] gives details of how to configure this particular transformation and also the general operating procedures. The specific command used within the ATI transducer controller is given in Appendix A.4.

transformation can be simplified since no torques are involved in this 2 DOF configuration and Equation 6.3 becomes :-

$$ {}^{C}F = \begin{bmatrix} c_{123} & -s_{123} \\ s_{123} & c_{123} \end{bmatrix} {}^{4}F \tag{6.4} $$

The expressions for the kinematics, Jacobian and force transformations can be reformulated to exploit the fact that θ_3 is constant and reduces the number of trigonometric calculations which are computationally intensive for the DSP. Further, due to the sparse nature of the selection matrix only certain expressions common to both the Jacobian and the inverse Jacobian need to be calculated, further minimising the number of calculations. Obviously these simplifications are specific to this particular implementation and particular constraint frame. If the constraint frame is altered then appropriate modifications to the software would be needed.

6.3.2 Experimental Setup

The experimental robot and ancillary hardware have already been described in Section 5.2. For the purposes of hybrid position/force control the arrangement was modified slightly to incorporate the six axis force/torque sensor and constraining environment.

The force/torque sensor has been briefly described in Section 3.2. The analogue sensor outputs are interfaced to the DSP (which runs the control algorithm) via the same 12 bit ADCs that are used for the joint angle measurements. The architecture of the resulting multi-loop controller implementation is described in the following section.

The constraining surface used in the experiments was a vertically mounted 5 mm thick steel plate, which can be seen in Figure 6.2. This plate was secured at its lower edge to a rigid beam that ran parallel to the ${}^{C}y$-axis at a distance of approximately 1060 mm along the negative ${}^{C}x$-axis. The stiffness of this test environment could be altered by using

plates of different thicknesses or materials, or by commanding the manipulator to contact the plate at different distances from the fixed edge. Further, this stiffness could be determined from the Young's modulus and dimensions of the plate. It should be remembered that with a hydraulic robot there is little chance of damaging the manipulator using such stiff environments since it has some inherent compliance due to the compressibility of the hydraulic fluid used in the actuators.

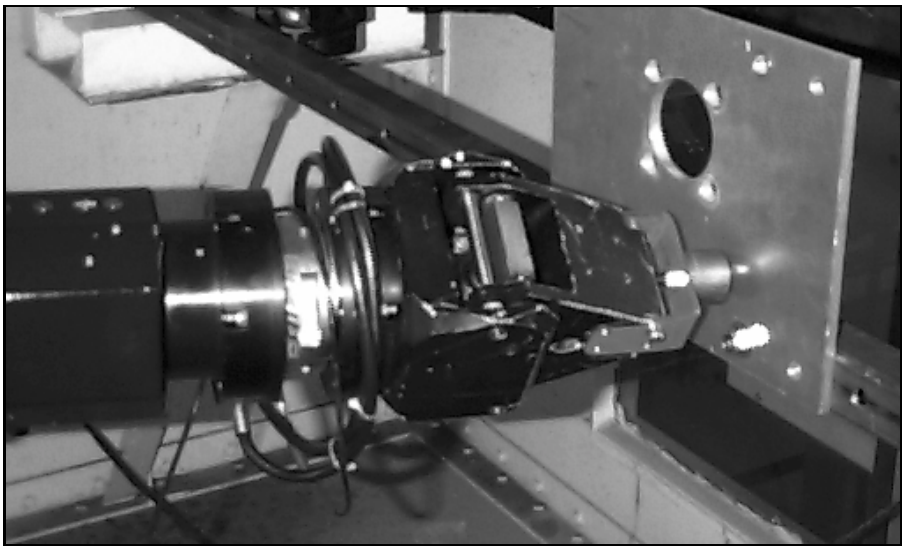

Figure 6.2 TA9 Manipulator Performing Hybrid Position/Force Task

The manipulator's end-effector was modified by adding a ball transfer unit, which can also be seen in Figure 6.2, allowing the end-effector to slide in any direction and minimising friction present during sliding contacts. Any friction that is present would appear as a disturbance in the control system and consequently degrade performance. Sensors or tools being deployed using these schemes should therefore be designed to minimise friction whilst sliding.

One constraint upon the use of this 'tool' is that it cannot accommodate contact angles of greater than 20° from the normal of the surface. This constraint is typical of the sensors and tools that are deployed during offshore tasks, for example NDT sensors generally have to be held perpendicular to the weld being examined. However, it should be

noted that this constraint is not a function of the hybrid position/force control system developed, which can accommodate any contact angle subject to proper controller tuning. Finally, as the ball transfer unit is not marinised these tests were performed out of water. However, the immersion of the manipulator in water would not have a significant effect upon the system's performance as hydrodynamic effects are not pronounced at the low velocities used here [1.2].

The weight of the claw and tool is sensed by the force/torque sensor and this effect must be removed from the sensor data before being used in the control algorithm. This is achieved by zeroing the sensor reading just prior to making contact, the procedure for which is described in Section 6.3.4. The effect of the claw's weight on the sensor measurements varies with its orientation within the gravitational field and this should be taken into account as the robot traverses the workspace. However, since this particular experimental setup is confined to a horizontal plane no such compensation is required.

It should also be noted that the sensor measures the reaction force rather than the force that the manipulator is applying to the environment, hence the force measurements need to be inverted.

One peculiarity of the TA9 manipulator is that the elbow actuator drives that joint in the opposite direction to that defined by the kinematics, where conventions specify that a positive input signal produces an increase in joint angle. This was accommodated by inverting the control signal to the elbow actuator within the hybrid controller. This compensation was not required for the SISO joint angle controllers since those controllers use the raw potentiometer voltages, whose directions followed the conventions.

The implemented hybrid position/force controller, incorporating the modifications specific to this configuration described above, is shown in Figure 6.3.

6.3.3 Controller Implementation

The realisation of the MIMO hybrid position/force controller using the single DSP will now be discussed. Three of the seven actuators of the TA9 manipulator (the wrist, claw rotate and claw open/close) are driven open loop, with a constant voltage being used to maintain them at their end-stops. Two of the four remaining joints (the shoulder up/down and forearm rotate) are controlled by SISO joint angle controllers that simply maintain these joint angles constant during the experiments. The remaining two joints (the shoulder slew and elbow) are used in the hybrid position/force control scheme given in Figure 6.3.

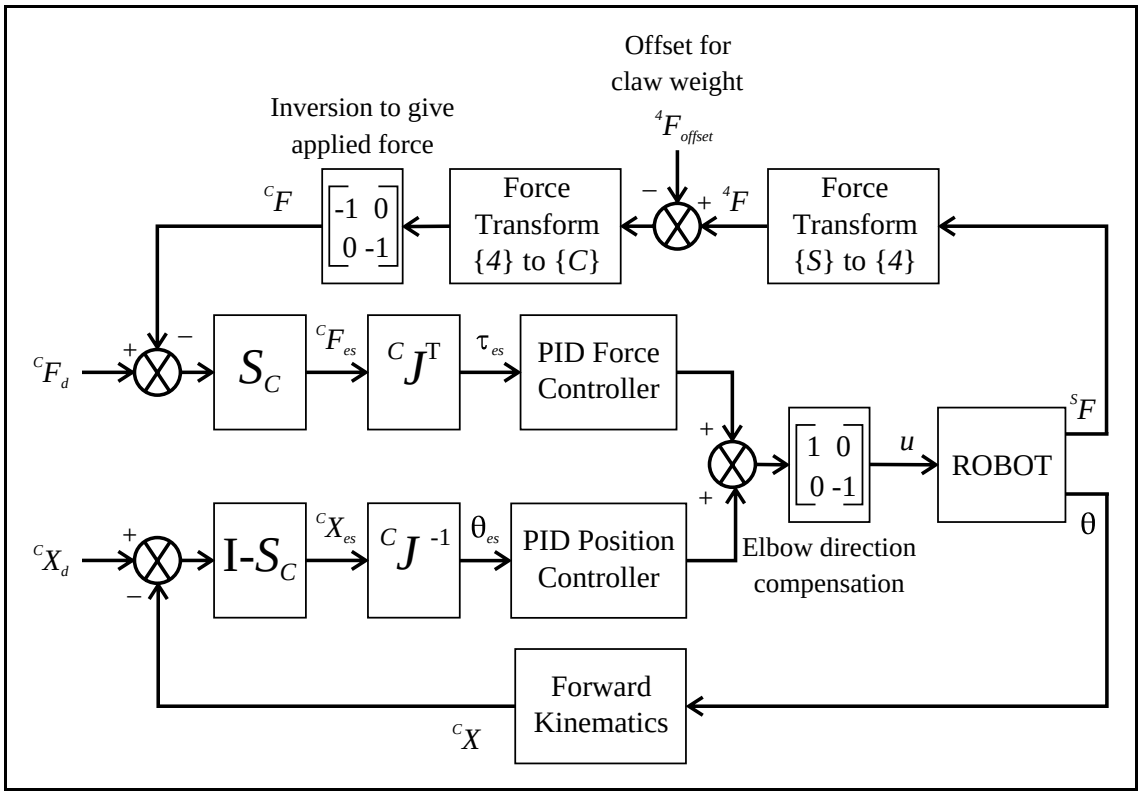

Figure 6.3 Experimental 2 DOF Fixed Gain Hybrid Position/Force Control

This system was realised on the DSP using the sequential structure described in Section 5.2, where the control for each joint is performed successively. The two SISO joint angle controllers were implemented as described previously. However, the two joints under hybrid control each consisted of two independent PID controllers, one for the force loop and one for the position loop. The outputs from each were then summed to produce the

actuator drive signal. An additional 'read-cycle' was required during which the force sensor data was acquired, but no control action taken. Thus five cycles were performed before the sequence was repeated, and therefore to realise the sample rate of 50 Hz an update rate of 4 ms for each successive cycle was required.

The calculations for the Jacobian, kinematics and transformations were performed during the control period that the elbow joint angle is read. Therefore the shoulder slew angle used in the calculations is not exactly coincident with the elbow angle due to the sequential structure of the controller, but this has a negligible effect.

The fixed gain PID controllers used are implemented in the incremental form (see Equation C.3) as this allows a smooth transition from one mode of control to another. This is especially important since it would be difficult to start the manipulator directly under the hybrid position/force control scheme. The PID controllers used in the hybrid controller have the form :-

$$\Delta i_p(k) = K_{Pp}[\theta_{es}(k) - \theta_{es}(k-1)] + K_{Ip}\tau_s\theta_{es}(k) + \frac{K_{Dp}}{\tau_s}[\theta_{es}(k) - 2\theta_{es}(k-1) + \theta_{es}(k-2)] \qquad (6.5)$$

$$\Delta i_f(k) = K_{Pf}[\tau_{es}(k) - \tau_{es}(k-1)] + K_{If}\tau_s\tau_{es}(k) + \frac{K_{Df}}{\tau_s}[\tau_{es}(k) - 2\tau_{es}(k-1) + \tau_{es}(k-2)] \qquad (6.6)$$

$$i(k) = i(k-1) + \Delta i_p(k) + \Delta i_f(k) \qquad (6.7)$$

where θ_{es}, τ_{es} = transformed position and force errors (Equations 6.1 and 6.2)

τ_s = sample period

$\Delta i_p(k)$ = incremental servovalve current from position loop at sample k

$\Delta i_f(k)$ = incremental servovalve current from force loop at sample k

$i(k)$ = summed servovalve current at sample instant k

K_{Pp}, K_{Pf} = proportional gains for position and force loops respectively

K_{Ip}, K_{If} = integral gains for position and force loops respectively

K_{Dp}, K_{Df} = derivative gains for position and force loops respectively

The controller output voltages are converted into servovalve currents by the signal conditioning circuitry. These currents produce the required changes in the joint angles and joint torques to give the desired Cartesian positions and forces. Since the joint torques cannot be explicitly commanded, the static nonlinear relationship between input current and joint torque, shown in Figure 3.6, augments the nonlinear robot dynamics presented in Chapter Three. However, under normal conditions the manipulator operates in the steep linear region of these characteristics and it is only when the commanded force is close to the limit of the robot's capability that this relationship becomes nonlinear. Consequently the PID controllers are tuned to operate in the linear part of these characteristics and if higher torques are commanded then degradation of the response will occur.

The fixed gain PID controllers were tuned manually to give fast and smooth responses to step inputs in both Cartesian position and force. The hybrid position/force controller structure allows independent gains to be used for both the position and force controlled directions, allowing the different dynamics of each to be accommodated.

6.3.4 Operation of Experimental Hybrid Position/Force Controller

The controllers have been developed to allow smooth transitions between unconstrained (independent SISO joint angle control) and constrained (MIMO hybrid position/force control) manipulator motions. This was achieved with the use of incremental PID controllers which provides bumpless transfer between controllers.

A hybrid task is started by first bringing the end-effector of the manipulator close to the initial contact point using the unconstrained joint angle controllers. The hybrid controller is then started using a keypress or command from a higher level planning system. This automatically zeroes the force/torque sensor readings to remove the effect of the claw's

weight. Then a force is commanded in the direction of the surface to be contacted, causing the end-effector to move towards the surface until contact is made and the desired force attained. The hybrid position/force control experiments are started once any transients due to the impact have decayed and a stable contact established, which is typically within one second.

The desired Cartesian positions and forces can be adjusted by simple keypresses, or can follow a pre-programmed sequence of events. Higher level planning systems can also be integrated with the hybrid scheme, with the desired positions and forces being received via TCP/IP communication. The forces, joint angles and control signals for each controller are recorded and stored at every sample instant by the PC which hosts the DSP. This enables the controller performance for the various experiments to be analysed off-line. Access to the trajectory interpolation and teach and play facility, as described in Section 5.2, are also available for use in the hybrid position/force controller.

6.3.5 Development Process for the Hybrid Position/Force Controller

A discussion on the process followed during the development of the hybrid position/force controller is considered to be useful at this point. It shows the evolution of the scheme from a simple single degree of freedom scheme, such as those presented in the previous chapter, through separate Cartesian force and position control schemes, which were then combined to give the final hybrid position/force control scheme.

This development process commenced with an initial undertaking to close a single force control loop around one of the TA9's joints. This was first attempted with the claw rotate joint, controlling the torque exerted on a fixed bar gripped in the jaws of the manipulator. Open loop tests were initially carried out to investigate how the torque exerted varied with applied control voltage. The control loop was then closed with a PID controller and experiments were conducted to investigate the effect of the three controller terms. This

demonstrated that, as would be expected, an integral term was required to removed steady state errors in the torque, and that too much integral gain destabilised the control loop. Derivative gain did help to speed up the response slightly, however due to the noisy nature of the force signal it was deemed better to omit derivative action.

The next stage in the development of the hybrid scheme was the realisation of a two DOF fixed gain static force controller. This essentially involved the implementation of the upper half of Figure 6.3 and allowed the following to be accomplished :-

- implementation of the MIMO control system within the framework described in Section 6.3.3 and integration with the force sensor and acquisition hardware.
- validation of the coordinate frames, transformations and Jacobian used.
- development of the operational procedure for starting these force control schemes, as described previously.
- tuning of the force control loop parameters to give a fast response with minimal overshoot. The resulting gains were K_{Pf} = 0.002, K_{If} = 0.0004 and K_{Df} = 0, and demonstrated that integral gain was required to remove steady state error.

A free space Cartesian position controller was then developed, corresponding to the lower half of Figure 6.3, which allowed the following to be achieved :-

- implementation of the MIMO control system within the same framework as the two DOF static force controller.
- validation of the controller calculations and operational procedures.
- appropriate controller gains were determined to be K_{Pp} = 20, K_{Ip} = 1.5 and K_{Dp} = 0 in the locality of ${}^{C}x$ = -900 mm and ${}^{C}y$ = 900 mm. These gave a fast response with minimal overshoot and an acceptable level of coupling between the two orthogonal

directions. It was observed that too much integral action caused excessive overshoot and too little resulted in long settling times.

- validation of forward kinematics through physical measurement of the end-effector location as the manipulator is commanded to move specific distances.
- variation of controller response as the manipulator moves throughout its workspace, away from the operating point at which its controllers were tuned.

Once these two control schemes had been proven separately, it was then simply a case of combining them, together with the appropriate selection matrices, to form the hybrid force/position controller of Figure 6.3. The tuned controllers derived for the independent force and position schemes worked well with an acceptable level of coupling (as shown in the next section) and did not require any re-tuning.

A further set of experiments carried out were open loop force control tests on the shoulder slew joint within the framework of the two DOF fixed gain static force control scheme. These results were used to determine the constant of proportionality of leakage across the hydraulic piston (k_{leak}) used in the simulation model, as described in Section 3.4.2. The open loop response was also used to validate the response of the simulation model. This is described in more detail in Section 7.3.1.

6.4 Experimental Hybrid Position/Force Control Results

This section presents the results of the hybrid position/force controller described in the previous section. A simple task, encompassing both force and position responses, was used to demonstrate the hybrid position/force control.

The hybrid position/force controller used those gains determined from the separate Cartesian force and position controllers, namely $K_{Pf} = 0.002$, $K_{If} = 0.0004$ and $K_{Df} = 0$ for the force loops and $K_{Pp} = 20$, $K_{Ip} = 1.5$ and $K_{Dp} = 0$ for the position loops. Re-tuning of the

PID gains under the hybrid scheme was attempted, however the aforementioned gains proved to be optimum for the nominal operating conditions.

Any deviation from these nominal conditions will result in a degradation of the control. To demonstrate this the basic task was repeated for various operating conditions, including changes in contact position in the manipulator workspace and changes in the level of applied force. The previous chapter demonstrated that significant variations in the dynamic response of the robot occurs when it moves either with or against gravity. This effect could not be examined here since the manipulator is confined to the horizontal plane. Consequently, if such a scheme is extended to a full six DOF system wider variations in robot dynamics would be observed. The same basic task was also repeated to investigate the effect of different trajectories for the reference signals.

All Cartesian force and position responses presented in the following sections are defined in the constraint frame, $\{C\}$, unless otherwise specified. Furthermore, the sample rate used within the controllers is 50 Hz, again unless otherwise stated.

6.4.1 Experimental Tests Performed

The basic task used in these experiments consisted of first bringing the end-effector close (within 2 cm) to the environment, located at ${}^{C}x$ = -1063 mm as shown in Figure 3.2. The hybrid control scheme was then started and a desired force commanded in the direction of the environment. This caused the manipulator to come into contact with the environment and exert the desired force. Then 10 seconds after the hybrid scheme was started, the desired force was increased by 50 N. A further 10 seconds later the position of the end-effector was commanded to move 100 mm in the negative ${}^{C}y$ direction. Although simple, this task does encompass both the force and position responses.

The data from each control loop was captured at the sample rate (50 Hz) and stored by the PC for off-line analysis. The performance of each controller was quantified using the

following measures :-

- the RMS position error during the time 5 s to the end of the experiment (32 s). The first five seconds were not used since the end-effector contact was being established.
- the RMS force error during the time 5 s to 20 s. This period only includes the step in desired force, as the effect of the position step on the force control is more effectively measured by the peak-to-peak force error.
- the peak-to-peak force error in force during the time 20 s to 25 s. This quantifies the disturbance on the force control resulting from the step in desired position.

6.4.2 Principal Results at Nominal Conditions

The experimental results presented in this section are for the operating conditions for which the PID controllers were tuned. Figure 6.4 shows the force and position responses for the full sequence of the task and Figure 6.5 shows the responses in the orthogonal, and hence uncontrolled directions. The corresponding control signals are shown in Figure 6.6.

Prior to the start of the task the end-effector of the manipulator is positioned close to the constraining environment using independent joint angle control. Figure 6.5 shows that at the start of the task, t = 0 s, the end-effector is about 1 cm away from the plate (${}^{C}x$ = -1063 mm).

The transition from independent joint angle control to hybrid position/force control is made at t = 1 s, this being indicated by the measured forces being zeroed to remove sensor offsets due to the weight of the claw and tool. At the same instant the desired *y*-position is set to the current *y*-position, ${}^{C}y$ = 870 mm. At t = 2 s the desired force is set to ${}^{C}F_{xd}$ = -100 N, as shown in Figure 6.4, and this causes the end-effector to move towards the environment, shown by the plot of ${}^{C}x$ in Figure 6.5. Contact is made at t = 3 s, where

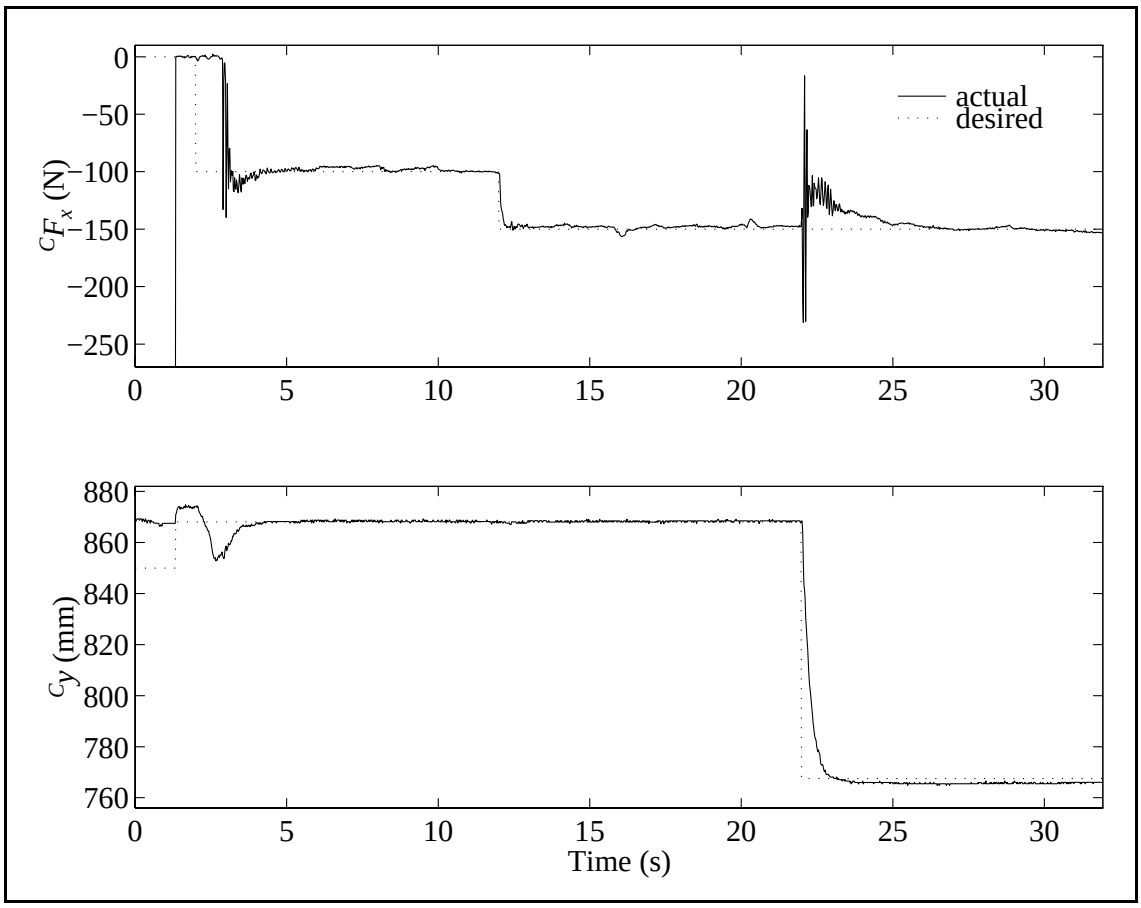

Figure 6.4 Hybrid Position/Force Control Results at Nominal Conditions

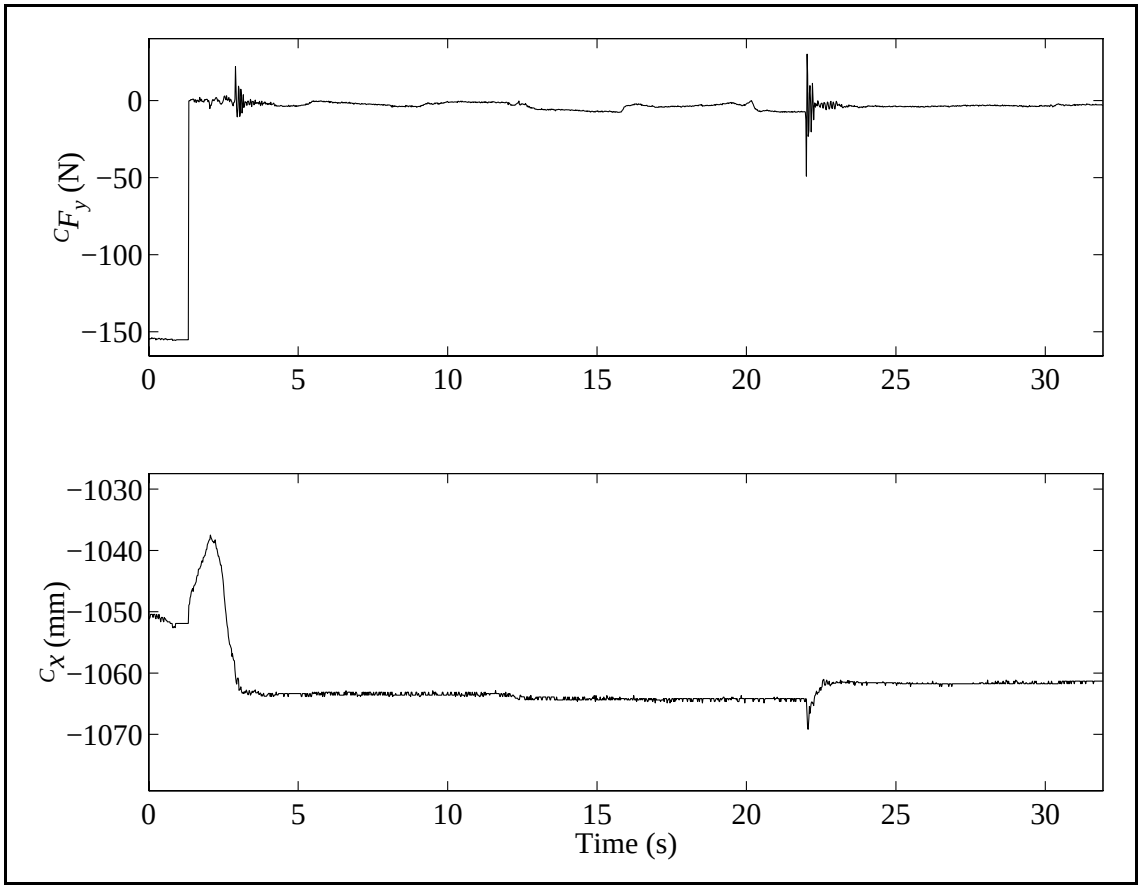

Figure 6.5 Orthogonal Position and Force Measurements

${}^{C}x$ attains a value of -1063 mm corresponding to the location of the constraining surface. The transition from free space motion to constrained motion is made and the force stabilises to the commanded value of -100 N by t = 4 s. The size of this initial commanded force determines the speed at which the manipulator approaches the environment, which in turn governs the size and duration of the impact. In all of the experiments carried out care was taken to allow a stable initial contact to be established prior to the experiment commencing. The *y*-position, which deviates slightly during the contact phase, is controlled to the previously defined desired position during this time.

The force reference is stepped to -150 N at t = 12 s, and then the desired *y*-position is decreased by 100 mm at t = 22 s. The experiment is stopped at t = 32 s. The responses of both the position and force are good, being both fast and well damped.

From these results it is clear that the step in the commanded force has an insignificant effect on the position control, however the position step degrades the force control significantly. Indeed the larger the position step the bigger is the disturbance on the force control loop. This implies that small and/or slow changes in position should be employed to minimise these disturbances. This idea is examined in Section 6.4.4.

The control signals, shown in Figure 6.6, indicate that the effort needed for the increase in force from 100 N to 150 N is small, whereas the motion of the manipulator requires much more effort. It is this increased effort during the movement phase of the task that creates the large disturbance in the force control. The performance measures for the controller response at these nominal conditions are :-

RMS position error, from 5 s to 32 s = 7.743 mm

RMS force error, from 5 s to 20 s = 4.047 N

peak-to-peak force error, from 20 s to 25 s = 214.7 N

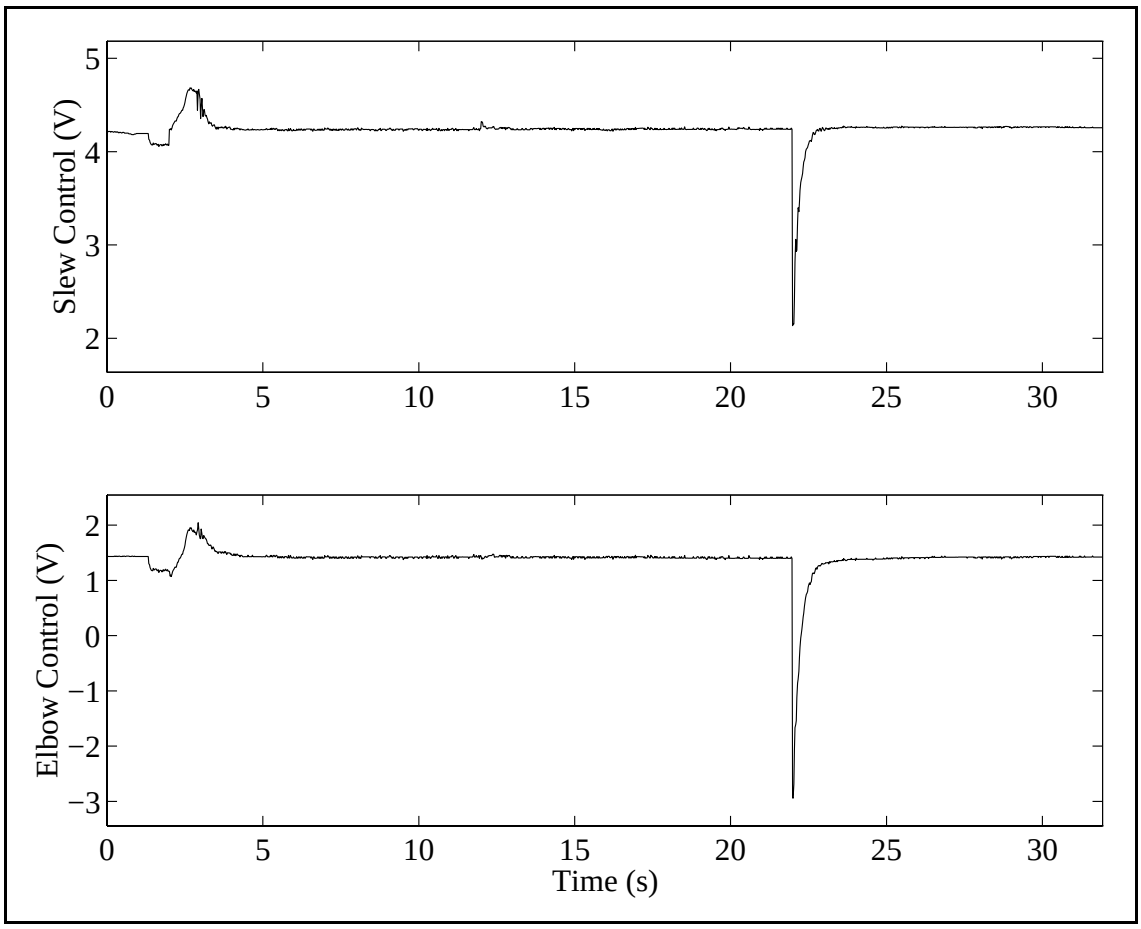

Figure 6.6 Control Signals for Hybrid Position/Force Control Task

6.4.3 Effect of Different Contact Positions

The same task was repeated for different points of contact along the ${}^{C}x$ = -1063 mm axis, the specific points being ${}^{C}y$ = 462 mm, 645 mm and 1015 mm, representing a good spread though the workspace of the manipulator. The RMS errors in both position and force for these locations and the nominal result (${}^{C}y$ = 870 mm) are plotted in Figure 6.7.

As the contact position changes, the variation in the response of the position controllers is minimal, as shown by Figure 6.7a. A more significant variation with respect to contact position can be seen in the RMS force error of Figure 6.7b. Here it is evident that the PID controllers tuned for the contact at 870 mm degrade as the contact is moved away from this point. Similar degradation away from the nominal point is also observed in the third performance measure, the disturbance in the force during the motion phase of the task.

The above differences in how the force and position loops degrade with respect to

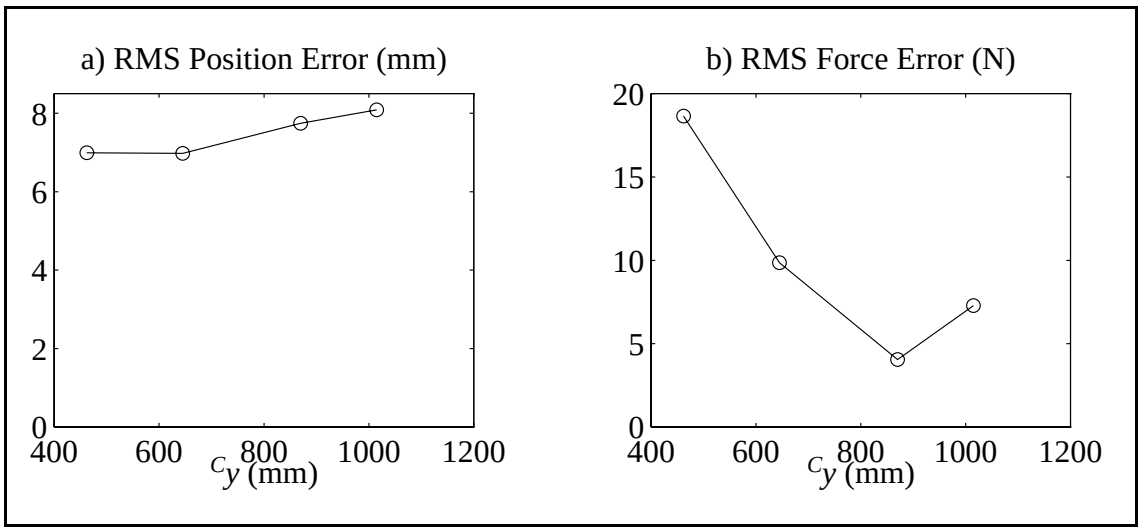

Figure 6.7 RMS Position and Force Errors for Different Contact Positions

the contact position arises from how the Jacobian varies throughout the manipulator's workspace. Figures 6.8 and 6.9 show how the elements of the $J^T(\theta)$ and $J^{-1}(\theta)$ matrices vary with respect to the joint angles θ_1 and θ_2. The elements of the $J^T(\theta)$ matrix, used in the force loop change significantly between the different configurations used, whilst the elements of the $J^{-1}(\theta)$ matrix, used in the position loop, remain relatively constant.

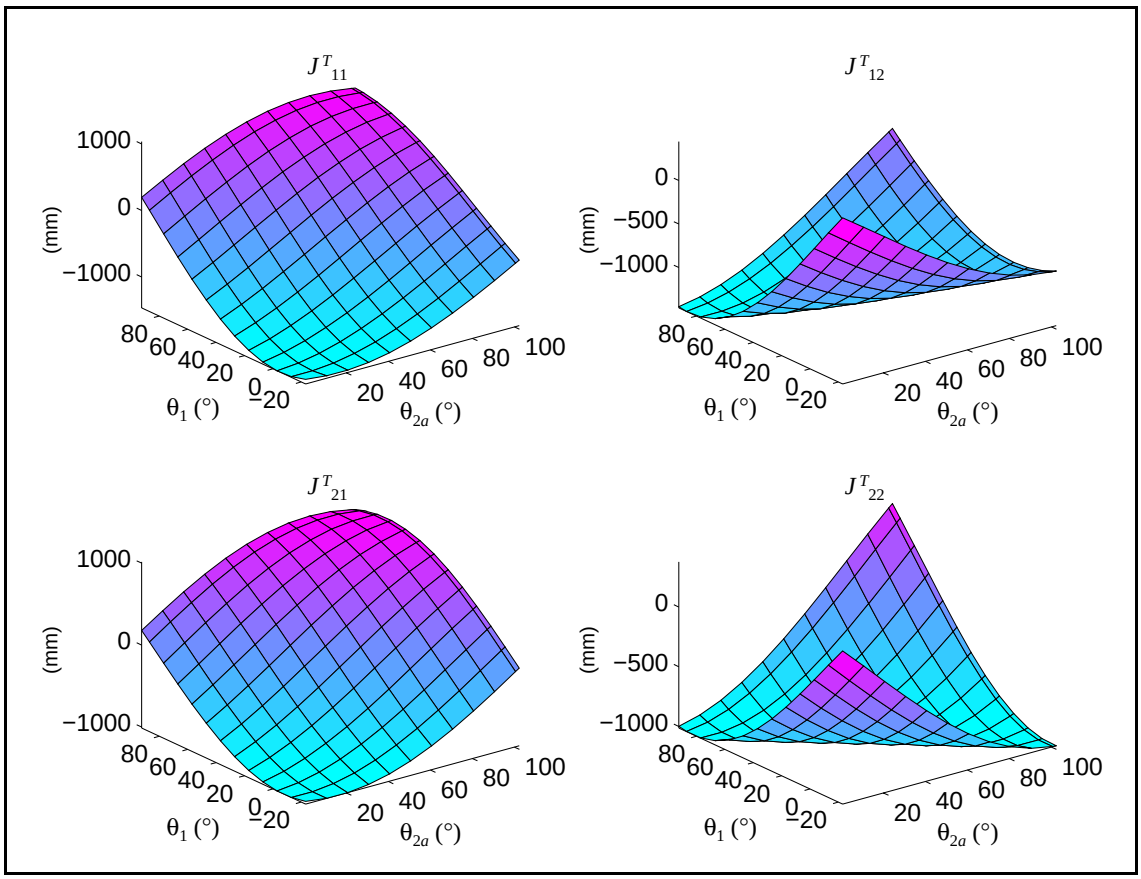

Figure 6.8 Elements of the $J^T(\theta)$ Matrix

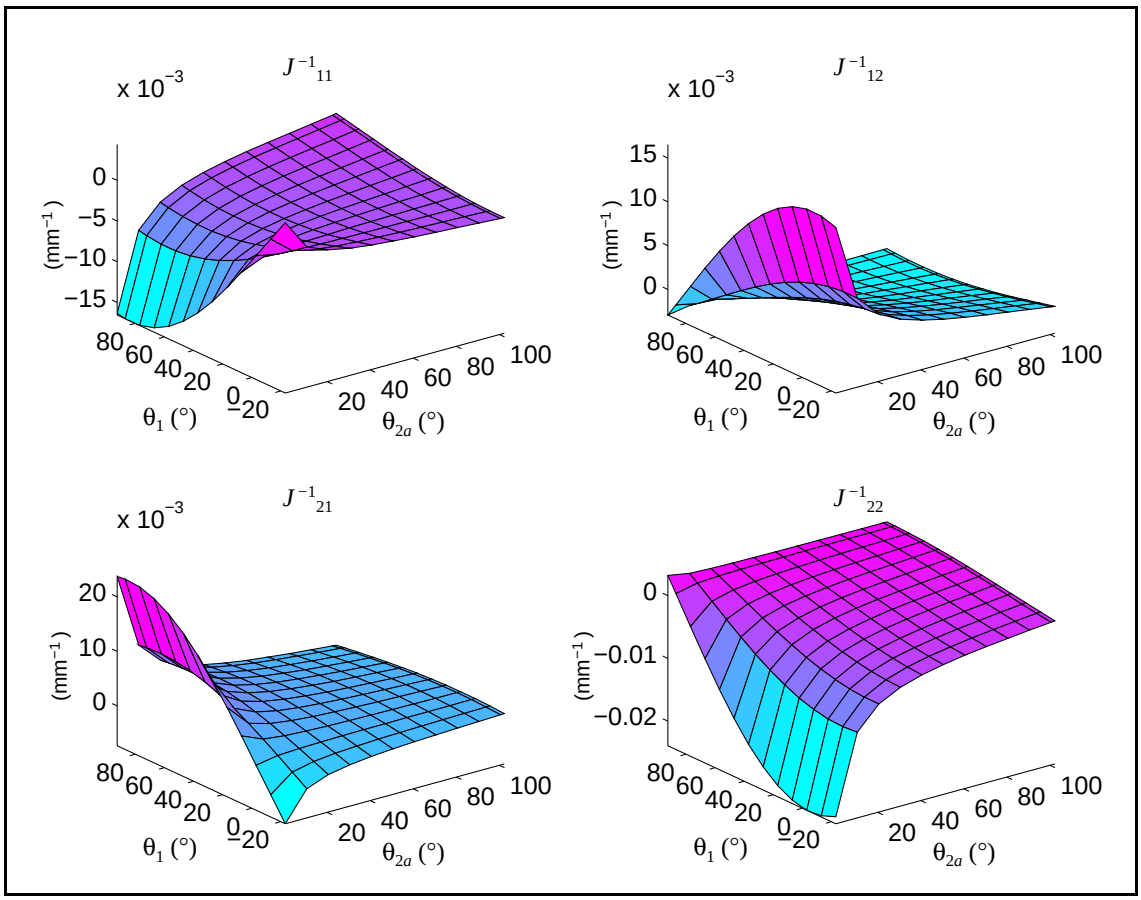

Figure 6.9 Elements of the $J^{-1}(\theta)$ Matrix

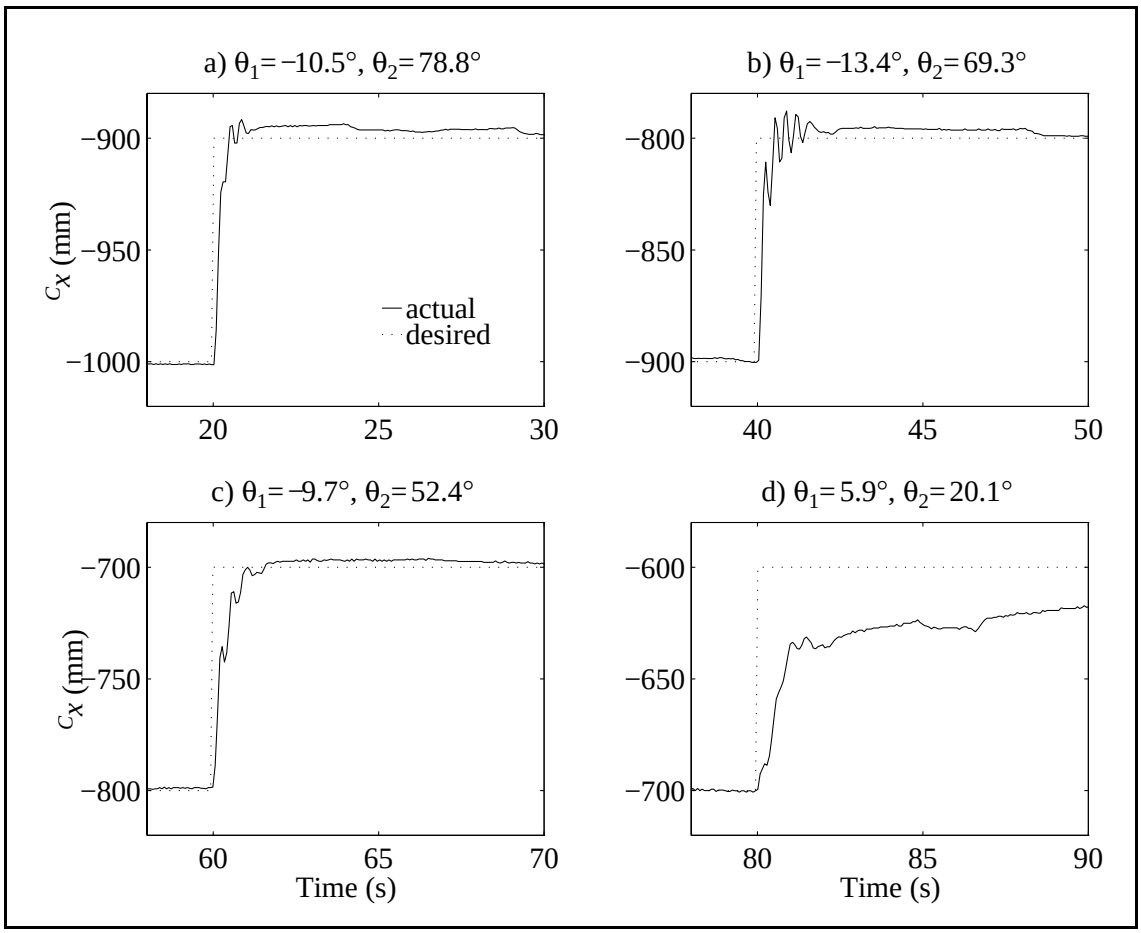

Figure 6.10 Variation of Position Control Loop Response for 2 DOF Cartesian Controller

Variations in the position response would be observed at those points in the workspace where there is significant variation in the elements of $J^{-1}(\theta)$, that is for $\theta_2 < 20^\circ$ which is when the manipulator is close to full extension. Experimental constraints prevented these regions being explored with the fixed gain hybrid position/force control scheme. Nevertheless, such variations in the position response were observed with the two DOF Cartesian free space position control scheme as the manipulator approaches the edge of its workspace and becomes fully extended. These variations, shown in Figure 6.10, would equally apply to a hybrid position/force controller, and illustrate the main limitation of fixed gain controllers whose gains can only be tuned for one point in the workspace.

6.4.4 Effect of Using a Modified Position Reference Trajectory

One of the limitations of this hybrid controller is the large disturbances in the force control loop when a step change in the position is commanded. The results presented in this section show the effect of using a smoother position reference trajectory to reduce the force disturbances. Figure 6.11 shows the effect of a ramped position reference, where the position reference took 2 seconds to achieve the 100 mm motion. The reduction in disturbance on the force is apparent from the plots.

The performance measures for these are given below, and when compared to those presented in Section 6.4.2 clearly show the benefits of using such a ramped position reference. An additional benefit is that the control signals are smaller and hence place lower demands on the actuators of the robot.

RMS position error, from 5 s to 32 s = 3.768 mm

RMS force error, from 5 s to 20 s = 7.777 N

peak-to-peak force error, from 20 s to 25 s = 62.1 N

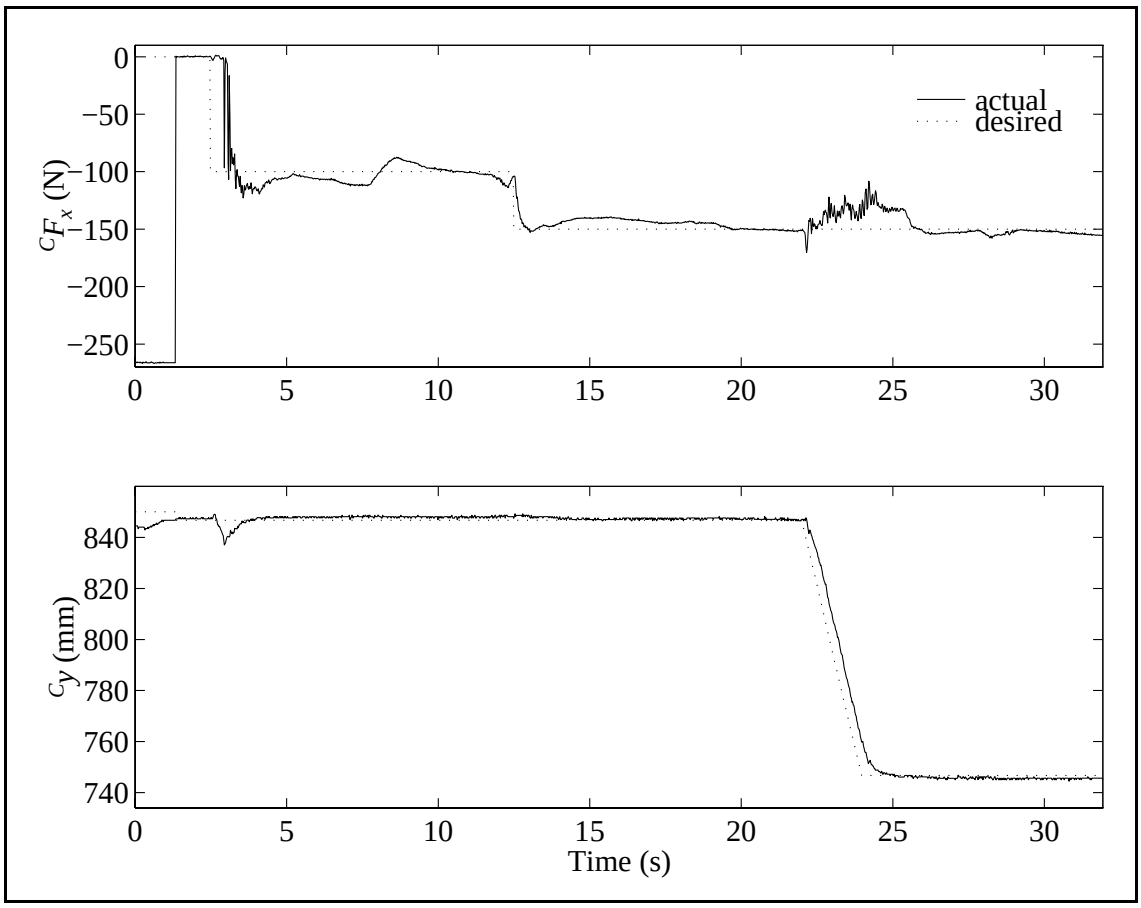

Figure 6.11 Hybrid Position/Force Control Results with Ramped Position Reference

It was observed that as the duration of the ramp is increased the force disturbances decrease as expected. A further experiment was carried out using a sixth-order polynomial to create a continuously differentiable function for the desired position trajectory (i.e. with smooth velocity and acceleration profiles) [5.1]. This did not result in significant improvement in performance of the system compared to the first-order case. The effect of using smoothed force reference trajectories was not investigated as the position loop does not suffer from disturbances from step changes in the force.

It should be noted that in any practical implementation of the hybrid position/force controller the position references employed should be ramped rather than stepped.

6.4.5 Effect of Different Levels of Commanded Force

To determine if the system response varied with respect to the size of applied force,

the task was repeated with the initial commanded force being varied between -50 N and -250 N. The size of the force step remained at 50 N.

Any changes in system response proved difficult to measure since the variations due to these changing operating conditions are small compared to the repeatability of the experiments. Some general trends can be observed however; the RMS force error during the step in commanded force increases with increasing initial commanded force, whilst the peak-to-peak disturbance in the force during the motion phase of the task decreases.

With the range of forces used, the manipulator is operating in a linear region of its torque/current characteristics (Figure 3.6). However, the use of much larger forces, causing the robot to operate in the nonlinear portion of the characteristics, could not be tested due to saturation of the force/torque sensor.

6.4.6 Effect of Different Sampling Rates

The results presented above use a sample rate of 50 Hz and this section describes some brief observations when different sample rates are used. At the higher sample rate of 100 Hz the response of the system was similar to that for the default case. However, the response of the force control loop degraded at sample rates of 25 Hz and 16 Hz (the lowest sample rate possible due to hardware timer limitations) with significant deviation from the reference being observed. No such degradation was observed in the response of the position loop at these lower sample rates.

The above findings justify the selection of 50 Hz as the sample rate used within the hybrid position/force controller.

6.5 Practical Manipulation Tasks

The fixed gain hybrid position/force controllers described in this chapter can be used to realise many different practical intervention tasks. For example, NDT inspection

of a weld on a subsea structure would require the implementation of a full six DOF hybrid position/force control scheme. This would use the same basic principles as the system developed here, but with the extension that curved constraining surfaces and operation outside the horizontal plane would need to be accommodated.

Other tasks that can be realised using these force and position control schemes include automatic insertions, guarded moves and automatic grasp alignment. These three particular behaviours have been implemented on the TA9 using the controller structures developed in this chapter and they will now be described in detail.

The specific task of mating a typical underwater connector and socket, shown in Figure 6.12, was chosen to demonstrate the extent to which this form of automated force control could assist the human operator. Under teleoperation these mating tasks are problematic as there is insufficient positional accuracy to allow the parts to be aligned precisely and jamming frequently occurs.

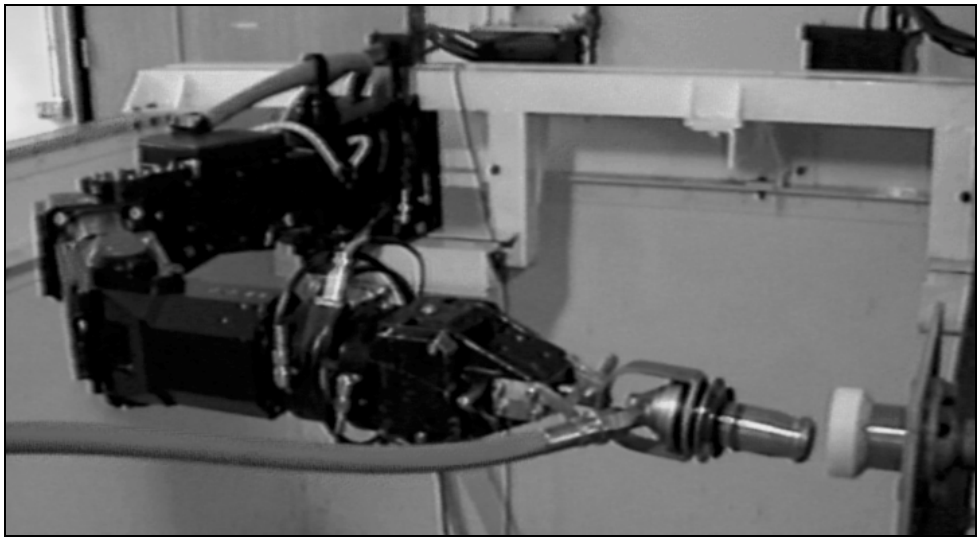

Figure 6.12 Automated Insertion of a Subsea Connector

Manual insertions of a typical underwater connector and socket were carried out using master-slave teleoperation to determine the level of forces generated during an insertion [1.4]. The forces and torques generated were large as any slight change in the

position reference from the master arm caused large forces to be developed. Figure 6.13 shows the forces and torques measured during a typical teleoperated insertion.

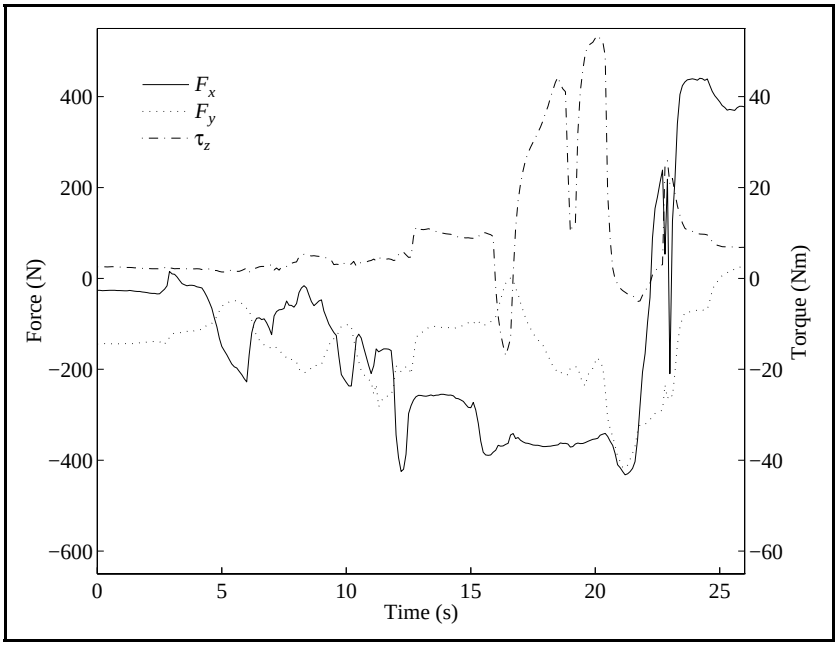

Figure 6.13 Forces and Torques During a Teleoperated Insertion

The particular insertion task examined here is confined to a horizontal plane as in the case of the hybrid position/force controllers described previously. However, an additional degree of freedom (the wrist joint) is required to control the torque at the tip of the connector, τ_{zd}. This results in a three DOF system, compared to the two DOF hybrid position/force controller described previously. Manual teleoperated insertions were performed using three DOF control from the master arm to demonstrate that it was physically possible to insert the connector when restricting the TA9 movements to just three joints.

The automatic insertion scheme developed here, shown in Figure 6.14, has distinct position and force controllers. Position control is used to bring the connector, held by the manipulator, close to and in *approximate* alignment with the socket. This positioning can be realised through the operator or some higher level task/motion planning system [1.4]. Once close to the socket the manipulator is switched from position control to force control

so that the insertion can then commence. The use of incremental PID controllers within this structure ensures that this transition is smooth with minimal transient demands on the actuators.

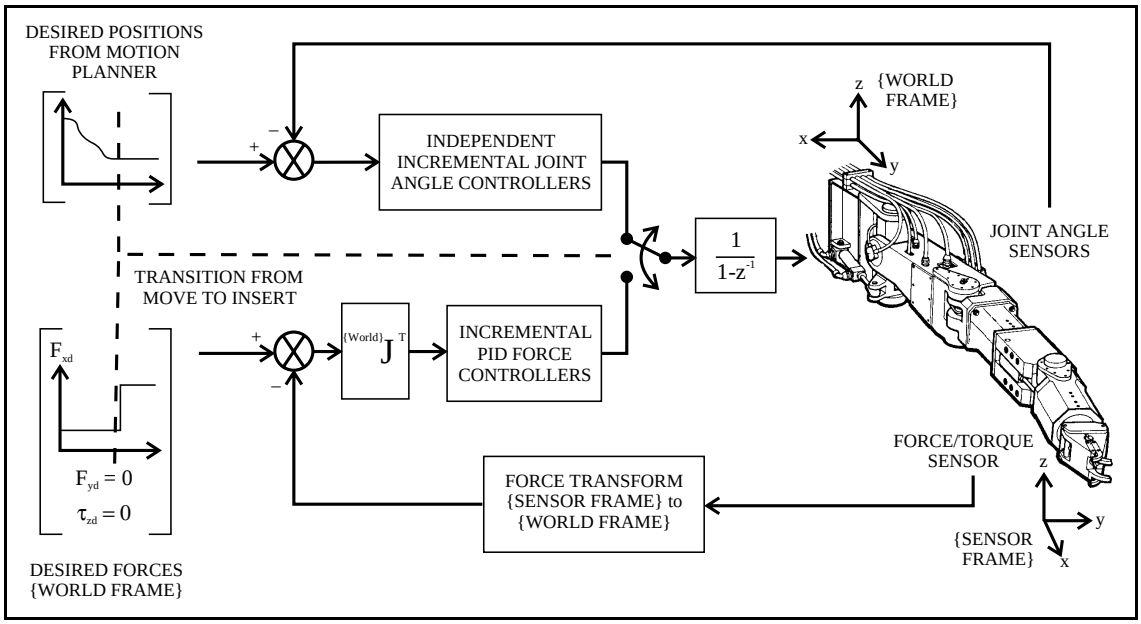

Figure 6.14 Controller Structure for Insertion Task

The insertion is realised by commanding a force, F_{xd} = 600 N, in the direction of the socket which pushes the connector into the socket. Although large, a step change of 600 N is well within the capability of the robot and is necessary to overcome the latching/locking mechanisms within this typical subsea connector. The reference values for the remaining controlled directions are set to zero, F_{yd} = 0 and τ_{zd} = 0, making the robot actively compliant in these directions. This compliance corrects for any misalignment between the two parts and prevents them from jamming. As the connector is being pushed into the socket the manipulator can be physically seen to be adjusting to the correct alignment [6.2].

The force and torque responses during a typical automatic insertion are shown in Figure 6.15. The latching mechanism within the subsea connector and socket caused considerable disturbances during the insertion which are evident from the responses shown. However, the automatic control scheme is able to reliably perform the mating task despite

these large disturbances. When compared to manual insertions this automatic system improves the reliability of mating and also reduces the possibility of damage to the connector, socket and manipulator.

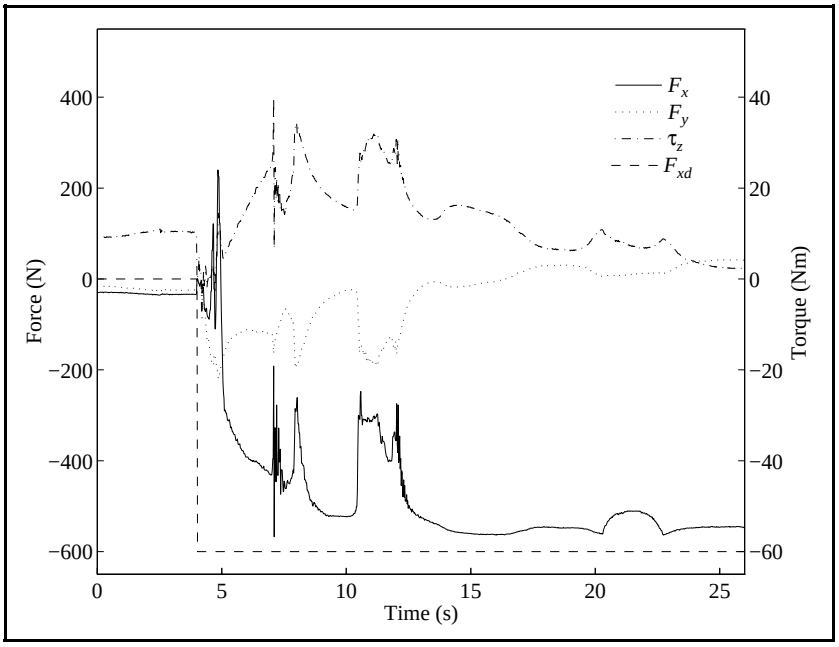

Figure 6.15 Forces and Torques During an Automatic Insertion

It should be noted that this task could also have been realised using a hybrid position/force scheme, but the simple division between position and force controlled phases, shown in Figure 6.14, proved adequate for this task.

Guarded moves and automatic grasp alignment can be implemented using similar techniques to that used for the automatic insertion. Guarded moves are a means of reducing the effect of unexpected collisions between the manipulator and its workspace. This is achieved by continually monitoring the end-effector forces during unconstrained motions. If the forces exceed a predefined value (indicating a contact) a force controller is invoked that acts to quickly reduce the contact force to a safe level. This behaviour can be used to improve the safety of subsea teleoperated robots as the workspace is often poorly known.

Similarly, automatic grasp alignment is achieved using the forces and torques generated by any misalignment between the robot and an object that it is attempting to

grasp. These forces and torques can be minimised by a force controller which has the action of automatically bringing the robot gripper into alignment with the object. This capability has been used to automate the grasping of connector handles and valves [6.2] and can significantly reduce the incidence of breakages since the forces exerted by the robot are now automatically controlled.

These automatic force control schemes can be of great benefit to the operators of robot manipulators, providing such systems with enhanced operational capabilities. Many low level *atomic actions* such as "touch", "tighten", "align", "slide" and "insert" can be implemented within this framework by a specific set of force, torque and position references. These atomic actions could then be used by operators to perform tasks and hence reduce the time and effort required and improve the reliability of operations. These commands could also form the primitive behaviours required by high level task planning software to realise complex behaviours autonomously.

6.6 Summary

This chapter has demonstrated that it is possible to obtain good results from a fixed gain hybrid position/force control scheme implemented on an industrial hydraulically actuated manipulator. These highly satisfactory practical results were achieved despite the inherent problems associated with the TA9 manipulator, including :-

- the joint angles used in the position control loop are obtained from noisy and low accuracy analogue potentiometers.
- ideally the joint angle velocities should be used, but such instrumentation is not available on this industrial manipulator.
- there is considerable stiction present in the hydraulic pistons and joints which degrades control performance.

Furthermore, the manipulator is a nonlinear system, and although the dynamic relationship between servovalve input currents and joint angles and joint torques has been modelled in Chapter Three, it is complex and there is coupling between joints. The use of independent PID controllers assumes a decoupled and linearised plant, nevertheless it has been shown that such a scheme still performs well when controlling this nonlinear and coupled system.

The fixed gain PID controllers can only be tuned for one particular set of operating conditions. Changes in these operating conditions are often unknown and/or unpredictable and have been shown to degrade the performance of the fixed gain controllers. If the robot is required to work over a wide range of operating conditions then the controllers need to be de-tuned and a poor control response accepted. Alternatively, an advanced control strategy could be employed to cope with these variations. In the next chapter of this thesis a self-tuning hybrid position/force controller will be developed that can automatically adapt to unknown and changing operating conditions.

Nevertheless, this chapter has demonstrated that even fixed gain hybrid position/force control can significantly widen the range of tasks that can be carried out by typical offshore manipulators. Task such as automatic insertions and grasp alignment can be realised in the framework of hybrid position/force control. This can therefore provide a suite of low level atomic actions that can be accessed by the operator or a high level planning system. Such enhanced capabilities represent a significant step forward for offshore teleoperated manipulation systems.

Chapter 7

Self-tuning Hybrid Position/Force Control

7.1 Introduction

The preceding chapter showed that the fixed gain hybrid position/force controller developed for the TA9 manipulator performed well when operating under those conditions that it was tuned for. However, the controller's performance degraded away from these nominal conditions. This degradation became particularly severe as the manipulator approached the limits of its operation, in terms of workspace, payload and applied force. Furthermore, these manipulators operate in environments where the nominal conditions are not well known. These difficulties highlight the need for a control strategy that can adapt to both changing and unknown conditions.

Chapter Five demonstrated that a SISO self-tuning controller can successfully provide performance improvements over a fixed gain controller for joint angle control. In this chapter a similar, but multivariable, enhancement is made to the hybrid position/force control scheme so that it can better accommodate unknown and changing operating conditions.

This chapter opens with an extension of the SISO self-tuning controller to the multivariable hybrid position/force control problem. The performance of the MIMO self-tuning control scheme is investigated through simulation, using the model of the TA9 manipulator developed in Chapter Three. Results are then presented that illustrate the controller's ability to cope with a wide range of unknown and changing operating

conditions, including different environmental stiffnesses and different positions in the workspace. These are compared with similar results for a fixed gain hybrid position/force controller to demonstrate the benefits of the self-tuning scheme.

A brief comparison of this self-tuning hybrid position/force controller is made with another form of advanced control, specifically a robust hybrid controller using Variable Structure Control (VSC). The performance and practicalities of the two control strategies are discussed.

Finally, the implementation of this multivariable self-tuning hybrid position/force controller on the actual TA9 manipulator is discussed, together with some preliminary results.

7.2 Self-tuning Pole Placement Hybrid Position/Force Control

Hybrid position/force control can be realised using the multivariable self-tuning pole placement controller framework proposed by Prager [4.4] and described in detail in Section 4.4. As mentioned previously, such a multivariable scheme controls the entire robot, including the coupling between the position and force controlled directions, as a single system. This self-tuning strategy identifies a time variant low-order linear MIMO model of the robot/environment. This is then used to continually adjust the controller gains such that the poles of the closed loop system match some user specified values, irrespective of changes in the dynamics of the system as conditions change.

The robot/environment system has inputs corresponding to the actuator input voltages and the outputs are taken as the Cartesian positions and orthogonal forces to be controlled. Hence, for the 2 DOF manipulator configuration described in Section 6.3, the system has two inputs, $v_1(k)$ and $v_2(k)$, the slew and elbow actuator voltages respectively. The outputs consist of the Cartesian position along the ${}^{C}y$-axis, ${}^{C}y(k)$, and the force along the ${}^{C}x$-axis, ${}^{C}F_x(k)$, where k is the sample number in the discrete time sequence. The

structure of this controller is shown in Figure 7.1 (cf. Figure 4.1 for the generalised self-tuning controller and Figure 6.1 for the fixed gain hybrid position/force controller).

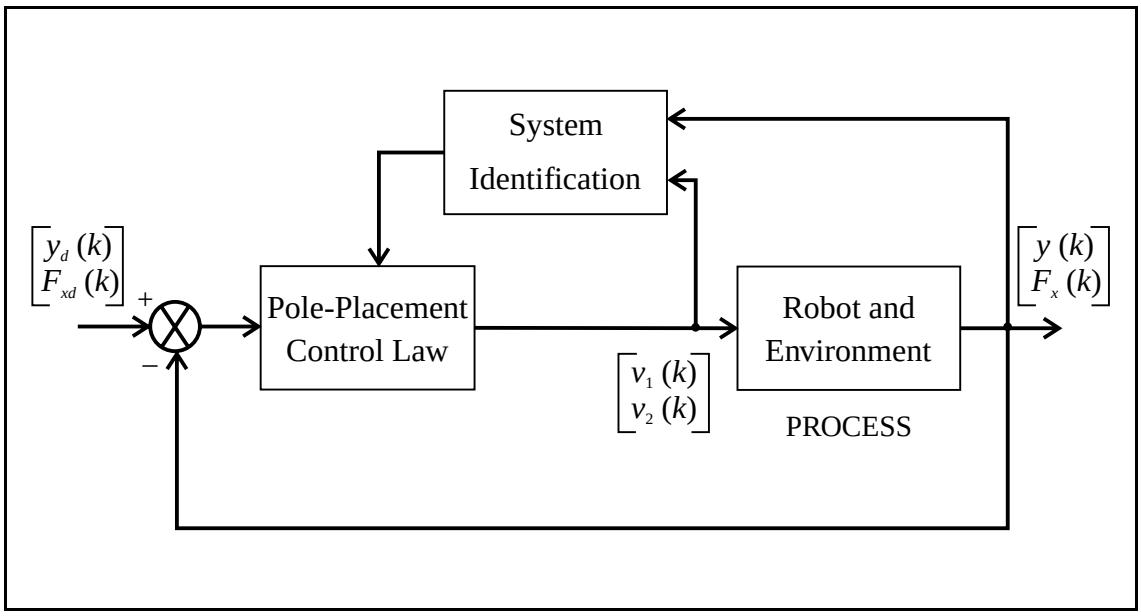

Figure 7.1 Self-tuning Pole Placement Hybrid Position/Force Controller

The generalised low-order linear process model given in Equation 4.25 becomes

$$\left[\mathrm{I} + A(z^{-1})\right]\begin{bmatrix} {}^{C}y(k) \\ {}^{C}F_x(k) \end{bmatrix} = B(z^{-1})\begin{bmatrix} v_1(k) \\ v_2(k) \end{bmatrix} + D_0 + E(k) \tag{7.1}$$

where $A(z^{-1})$ and $B(z^{-1})$ are polynomial matrices as described in Section 4.4.1, which have n_a and n_b 2×2 matrix coefficients respectively. Similarly, D_0 and $E(k)$ are 2×1 vectors representing the drift disturbances and modelling errors respectively. Therefore the total number of model parameters for the MIMO model, n_Φ, is $4(n_a+n_b)+2$ (cf. Equation 4.27).

The model structure and polynomial order, n_a and n_b, should be chosen to best reflect the actual physical process, so that the modelling error is as small as possible. This choice is discussed in Section 7.4.2.

The system identification component of Figure 7.1 uses recursive least squares (RLS) to iteratively estimate the A and B matrix polynomials using past values of the system's inputs and outputs. As with the SISO system, the RLS algorithm is formulated

using Bierman U-D factorisation to ensure numerical stability, which is easily extended to the multivariable case, as described in Section 4.4.2. A forgetting factor, λ, is utilised to allow the model to adapt to time varying changes in the process dynamics. The operational issues associated with the system identification component, including choice of λ, are discussed in the Section 7.4.1.

With the robot/environment represented by the low-order model of Equation 7.1, the system block diagram can be redrawn so that the controller can be defined as a function of the model parameters, as shown in Figure 7.2. As with the SISO joint angle controller, this multivariable self-tuning pole placement controller incorporates a digital integrator, yielding a MIMO incremental controller.

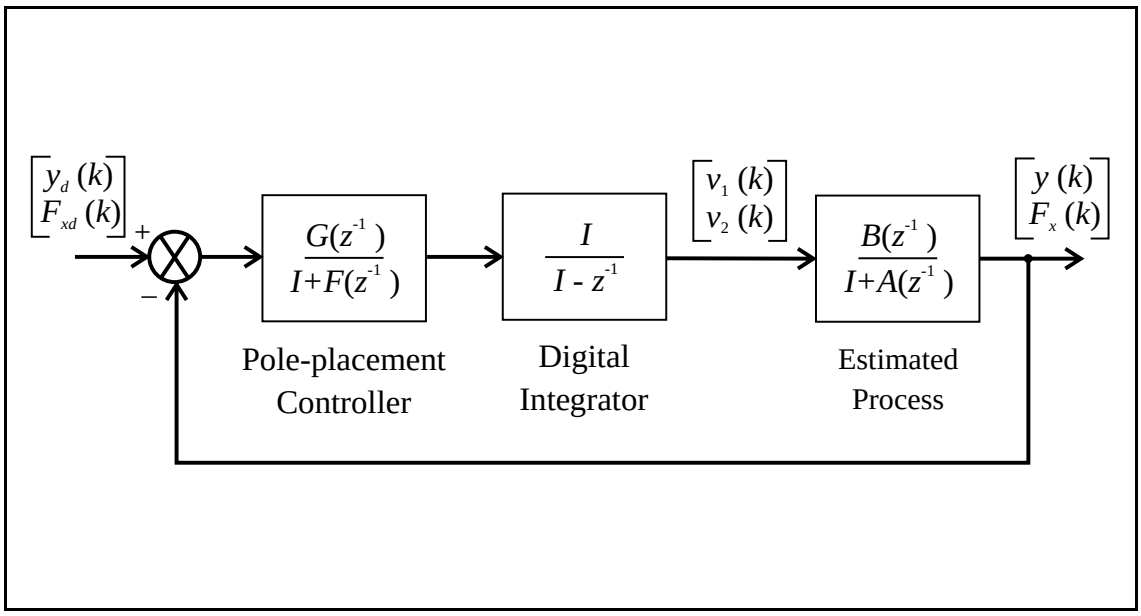

Figure 7.2 MIMO Self-tuning Controller Configuration

The controller matrix polynomials, $F(z^{-1})$ and $G(z^{-1})$, are as defined in Section 4.4.3, with n_f and n_g+1 2×2 matrix coefficients respectively. These controller polynomials are calculated on-line such that the poles of the closed loop system are equal to values defined by the user specified polynomial $T(z^{-1})$. This involves the solution of a set of linear simultaneous equations, as described in Section 4.4.3, which define n_f and n_g in terms of the model orders n_a and n_b. The specific solutions to these linear simultaneous equations

for the model orders used in this thesis are given in Appendix C.

7.3 Utilisation of Simulations

The work in this Chapter is largely developed using the high fidelity nonlinear model of the TA9 manipulator that was developed in Chapter Three. This model was realised within the SIMULINK simulation environment as described in Section 3.6, and the various files and parameters used are presented in Appendix A.5. It must be stressed that this model is only for the purpose of simulating the dynamics of the manipulator, and is not used within the controller part of the simulation.

The use of a simulation allows the development of such advanced control strategies to be carried out without the additional complexity of a practical, real-time implementation. There are several other benefits that a simulation provides over practical experimentation, specifically :-

- it allows the controllers to be developed and tested much more efficiently.
- simulations allow controller performance to be investigated under conditions that would be difficult to explore with practical experiments, for example, changes in hydraulic fluid compressibility.
- simulations do not suffer from problems of repeatability, as practical experiments can. This is particularly important for force control as small differences in positioning can result in large changes in the forces observed.
- simulations also allow controllers developed by different research teams to be compared using a common system. This is demonstrated in Section 7.6.

However, the practical implementation must always be borne in mind when the controllers are being developed, as it is the practical realisation that is the fundamental goal

of this work. A practical implementation of the self-tuning hybrid position/force controller is described in Section 7.7.

7.3.1 Simulation Model Validation

The hydraulic manipulator model used within the simulation must resemble the real manipulator for the simulation results to be valid. The model parameters (e.g. link lengths, angle limits and offset angles) were set to match the physical dimensions and specifications of the actual manipulator, actuators and environment. The various model parameters used are given in Appendix A.2. Validation against experimental data was required since not all of the parameters were known accurately, and also to confirm the model formulation and assumptions.

The kinematic parameters were initially validated through Cartesian control experiments, comparing the actual motion of the end-effector to the expected motion. Since this is purely a function of the manipulator link lengths and joint angles, this enabled them to be validated in isolation.

The piston leakage parameter, k_{leak}, introduced in Section 3.4.2 could not be accurately obtained from any manufacturer's specification. Therefore, an open loop force control experiment was devised to measure it. This test was carried out by having the manipulator in contact with the environment under open loop force control, i.e. a fixed current being supplied to the servovalves. The servovalve current was then increased by a small known amount, causing an increase in contact force as shown in Figure 7.3a.

The size of the step in the contact force is related to k_{leak} and other known model parameters by means of expressions given in [3.4]. For the shoulder slew actuator k_{leak} was found to be 8.476×10^{-14} m^5 N^{-1} s^{-1}. The values for the other actuators were estimated by assuming the leakage to be proportional to the internal circumference of the piston.

With this value for k_{leak}, the simulated open loop force response to the step in

servovalve current was faster than that observed experimentally. By reducing the value of bulk modulus, β, from the ideal value of 17×10^8 N m^{-2} to a more practical value of 7×10^8 N m^{-2}, the rise time of the simulation matched the experimental result, as shown in Figure 7.3b. The small perturbations that can be seen in the experimental response are due to unmodelled effects which are only apparent when under open loop control.

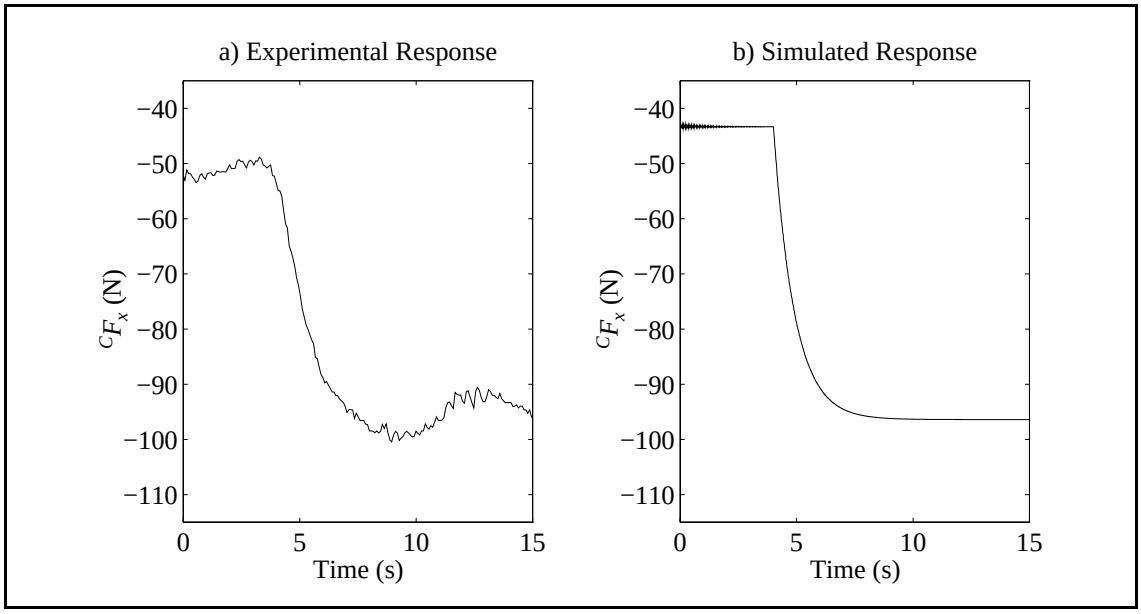

Figure 7.3 Experimental Determination of k_{leak}

The manipulator and environment model was linked to a fixed gain PID hybrid position/force controller. This controller was configured to match, as closely as possible, the experimental controller developed in Chapter Six, including using the same coordinate systems and transformations. The gains used also matched those used in the experimental setup, specifically $K_{Pf} = 0.002$, $K_{If} = 0.0004$ and $K_{Df} = 0$ for the force loops and $K_{Pp} = 20$, $K_{Ip} = 1.5$ and $K_{Dp} = 0$ for the position loops. The stiffness of the simulated environment was specified as 7×10^5 N m^{-1}, representing the steel end-effector contacting the steel plate, as used in the experimental setup. The sample rate was also set to match the 50 Hz used in the experiments, however the control signals applied to each joint were calculated concurrently, and not sequentially as in the experimental implementation.

The simulations were initialised with a stable contact established by setting the

initial controller outputs to match the equilibrium voltages, as described in Section 3.5. This therefore corresponds to the practical results presented in Chapter Six, neglecting the initial impact transients.

The top-level SIMULINK block diagram for this hybrid position/force controller is shown in Figure A.1 in Appendix A. The integration algorithm used within the simulation is as described in Section 3.6.

The response of the simulated system is shown in Figure 7.4 and compares well with the experimental results given in Figure 6.4. The simulated force response has a larger disturbance response than the experimental result arising from the manipulator motion along the ${}^{C}y$-axis. However, these short transients are probably masked in the practical system by filters within the force/torque transducer controller. The response times of the force control and position control show good correspondence between simulation and experimental results.

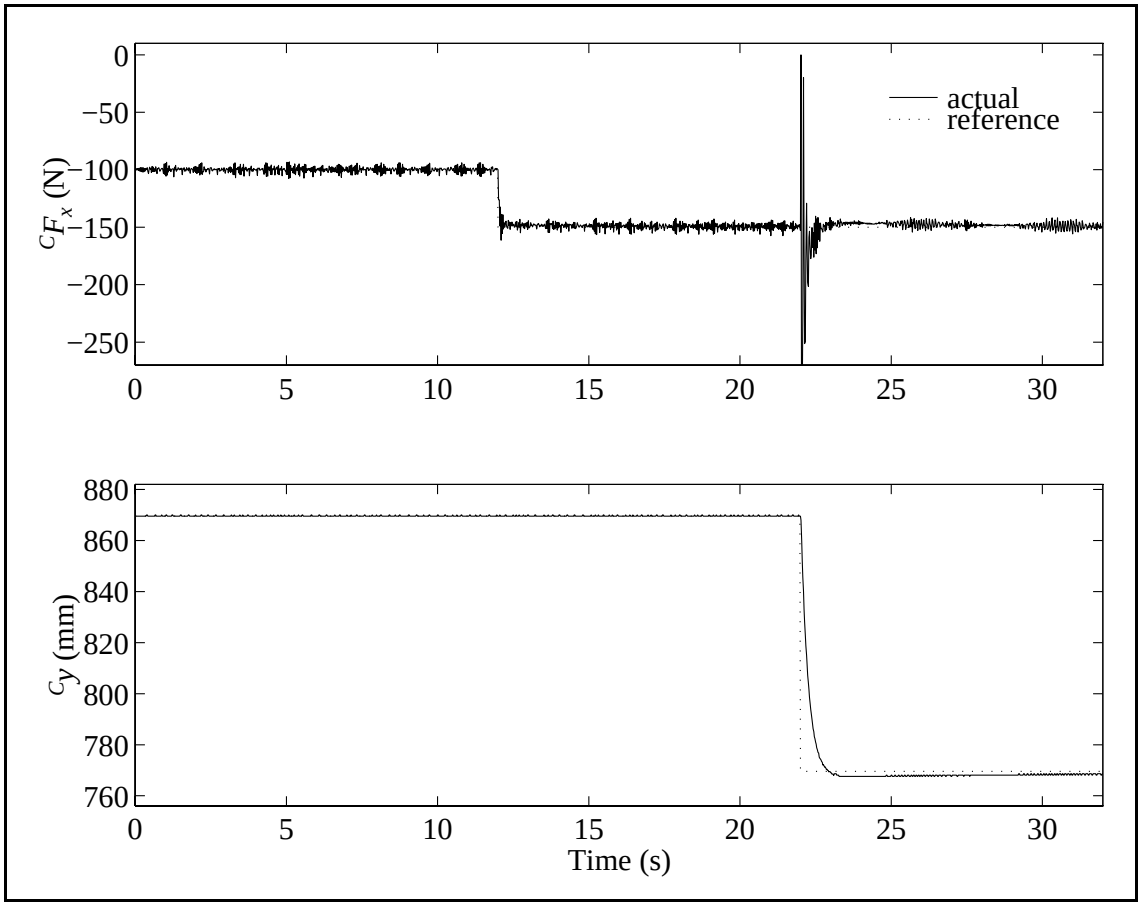

Figure 7.4 Simulated Fixed Gain Hybrid Position/Force Control Results

7.4 Self-tuning Hybrid Position/Force Control Development

As with the experimental hybrid position/force controller, a staged development of the self-tuning controller took place to ensure that the system was not burdened by unnecessary complexity at an early stage. This commenced with the development of a self-tuning force controller for a single link manipulator. The ability of this simple controller to cope with changing operating conditions was explored.

The satisfactory outcome of this SISO self-tuning force control paved the way for the development of the MIMO self-tuning hybrid/position force control scheme. This was developed firstly in terms of the MIMO system identifier, then appropriate model structures were established and finally the overall self-tuning controller was then established. These three stages will now be discussed in more detail.

7.4.1 MIMO System Identification Operation

The operation of the MIMO system identifier drew heavily upon the findings for the SISO system identification, presented in Section 5.3.1. Specifically, the operational benefits of using the Bierman UD RLS (BUD-RLS) algorithm over the standard Matrix Inversion Lemma RLS (MIL-RLS) identified for the SISO system in Section 5.3.2, meant that the BUD-RLS algorithm was used exclusively for the MIMO system identification.

As with the SISO system there is a trade-off between how fast the identifier can track system changes and its sensitivity to noise, governed by the forgetting factor, λ. The effect of different forgetting factors on the parameter estimates was similar to that presented in Figure 5.7 for the SISO case. A value of $\lambda = 0.99$ was found to be most effective for this system. With $\lambda = 0.995$, the system identification was not sufficiently quick, and for $\lambda = 0.95$, it proved too sensitive, causing large parameter changes even for small changes in both position and force.

The initial model parameter estimates, $\Phi(0)$, used within the MIMO system

identification were critical to its success. For the SISO case a simple integrator was assumed, however here it was crucial to start with initial parameter estimates that better matched the physical system. The initial model parameters used were obtained from off-line identification of data acquired when the manipulator was executing a series of steps in both position and force controlled directions. It should be noted that these initial parameter estimates are a function of the sample period used, since the model is defined in discrete time.

The use of initial parameter estimates that closely match the system should be reflected in the initial value of covariance matrix, $P(0)$. However, it was found that to enable adaptation over a wide range of unknown operating conditions, a value of $P(0) = 100I$ should be used. This matched that used for the SISO case. The use of a priori determined initial parameter estimates also meant that the regression vector, ψ, should be completely filled with past data before the identifier is started, rather than just being initialised with data from the previous sample only (Equation 5.1), as in the SISO case.

As mentioned previously, for system identification to generate well conditioned estimates, it requires the system to be persistently excited, since under steady state conditions there is little new information for the identification algorithm. The fact that this MIMO system has cross coupling actually helps maintain persistent excitation, since even if only one input is being changed, the coupling helps to excite other parts of the system.

No covariance management techniques were employed in the MIMO system identification algorithm. However, it is recognised that they should be used for practical implementations to safeguard the identification process, especially when the manipulator is not moving or when the inputs are changing smoothly or slowly.

7.4.2 MIMO Model Order and Structure Selection

As mentioned earlier, the choice of model structure and its order should ideally

reflect the nature of the underlying continuous time process. However, the number of parameters to be estimated, given by Equation 4.27, may quickly become excessive and often only low order models may be feasible for practical implementations.

As with the SISO process, an empirical method of determining the most appropriate model order is to examine the a priori prediction errors for different values of n_a and n_b and simply take the one with the smallest error. Table 7.1 gives the RMS a priori prediction error for both the force and position outputs of the MIMO model, for a sequence of open loop steps in position and force. The mean of these errors were close to zero, indicating that there was no bias in the parameter estimates generated.

n_a	n_b	n_d	RMS position error (mm)	RMS force error (N)
1	1	0	0.0368	0.239
1	1	1	0.0364	0.236
2	1	0	0.0194	0.124
2	1	1	0.0192	0.122
3	2	1	0.0012	0.013

Table 7.1 RMS A Priori Prediction Errors for Different Model Orders

Table 7.1 shows that with this process even low order models gave low modelling errors and could track the process output well. However, when coupled to the appropriate self-tuning controller they could not accommodate wide changes in operating conditions. This may have arisen due to limitations with the corresponding controller, which become increasingly sophisticated as the estimated model order increases. A suitable model order was found to be $n_a = 3$, $n_b = 2$, matching the model order used in the earlier experimental SISO joint angle control work, presented in Chapter Five.

The MIMO model structure proposed by Koivo [2.89] for a hybrid position/force controller, discussed in Section 4.5, used a simplified structure where the off-diagonal

terms in the *A* matrices were set to zero to reflect the orthogonality of his system. This reduced the computational requirements of the system identification algorithm employed. However, this structure proved unsuitable for the system considered here, due to the large degree of cross coupling between position and force axes. Hence the off-diagonal terms were included in the model, and the cross coupling was sufficient for all of the model parameters to be readily identified. The resulting model contained 22 parameters for the 2 DOF manipulator system.

7.4.3 Self-tuning Hybrid Position/Force Control Operation

The initial parameter estimates of the model are critical to the successful operation of the self-tuning hybrid position/force controller. If the model parameters do not match the process, then it is prudent to use a fixed gain controller until the estimated model has converged and the prediction error is small. Once the model has converged for the current conditions, the self-tuner can track the changes in the physical system as they occur. Clearly the closer the initial estimates are to the real system, the sooner the self-tuning controller can be used. The initial parameter estimates were chosen, using a priori off-line identification, to match the process at some prescribed nominal condition.

A fixed gain controller was used for the first 20 samples (0.2 s) following the first commanded step, after which the self-tuning controller was used. This short time proved sufficient to allow the controller to operate over the wide range of initial conditions considered here, as demonstrated in Section 7.5. The use of incremental controllers facilitated the switching between the fixed gain and self-tuning control strategies. The sample rate used throughout this work was 100 Hz.

To further extend the range of initial conditions that the controller can successfully accommodate, the length of time under fixed gain control could be increased, allowing more time for the model to converge to appropriate values. Most previous reports of

self-tuning control have used a fixed gain controller for a number of repeated steps, before the self-tuning controller was switched on. If acceptable, this would allow operation over an even wider range of conditions.

The self-tuning controller is computationally intensive. Around 6100 floating point operations are required for the 2×2 system identification algorithm (with $n_a = 3$, $n_b = 2$ and $n_d = 1$) and approximately 1200 floating point operations for the corresponding self-tuning controller. This should be compared to the 46 floating point operations required for the equivalent fixed gain PID hybrid position/force controller developed in Chapter Six. This large number of floating point operations stems from the large matrices operated upon during both the identification and control parts of the scheme. Although the controller is computationally intensive, given the power of currently available digital signal processors, it is feasible to implement this on the DSP system available at the sample rates required, as demonstrated in Section 7.7. This aspect is also discussed further in Section 7.6.2, where the self-tuning scheme is compared to another form of advanced control.

7.5 Self-tuning Hybrid Position/Force Control Results

This section presents results from the self-tuning hybrid position/force controller developed in the previous sections. As with the fixed gain hybrid position/force controller, simple tasks involving both position and force commands were used to investigate the controller performance. Results are presented for a wide variety of operating conditions, demonstrating the controller's ability to cope with both unknown initial conditions and changing conditions. This includes changes in environmental stiffness, contact position, level of applied force and hydraulic fluid compressibility. Further, these results are compared to those obtained from an equivalent fixed gain controller to highlight the benefits that the self-tuning scheme provides.

The self-tuning hybrid position/force controller directly replaced the fixed gain

controller used in the validated simulation that was described in Section 7.3. The Cartesian position and force responses presented here are defined in the constraint frame, $\{C\}$, matching the convention used in the earlier work.

The system identification algorithm and its initialisation is as discussed in Section 7.4.1. The model structure used had $n_a = 3$, $n_b = 2$ and $n_d = 1$, with the corresponding controller matrix polynomials being as defined in Appendix C. The polynomial, $T(z^{-1})$, defining the desired closed loop poles, was specified to give two discrete time poles at 0.96 for the position loop and 0.95 for the force loop, giving rise times of approximately 1.7 s and 1.5 s respectively.

7.5.1 Simulation Tests Performed

The simple task used to investigate the performance of the controller was similar to the one used with the fixed gain hybrid position/force controller in the preceding chapter. The simulations were initialised with the manipulator in equilibrium, exerting zero force (${}^{C}F_{xd} = 0$ N) on the environment which was defined as being along the ${}^{C}x = -1050$ mm axis. Consequently, the procedure used previously to bring the manipulator into steady contact with the environment was not required.

The basic hybrid position/force task used consisted of a step in commanded force to ${}^{C}F_{xd} = -100$ N at t = 0.1 s, followed by a 50 mm move along the negative ${}^{C}y$-axis at t = 2.0 s. The simulation was stopped at t = 5.0 s. Although simple, this task does encompass both position and force responses as well as the cross coupling between them. Some alternative, more complex tasks were also used, consisting of repeated steps in commanded position and/or force, to investigate the ability of the system to adapt to changing conditions. These tasks will be described when introduced.

The nominal conditions used in these results were, an environmental stiffness, K_E, specified as 1×10^4 N m^{-1}, a hydraulic fluid compressibility (bulk modulus, β) of 7×10^8

N m^{-2}, an initial contact position of ${}^{C}x$ = -1050 mm, ${}^{C}y$ = 870 mm, corresponding to shoulder slew and elbow angles of 98.3° and 51.0° respectively.

The operation of the self-tuning controller was as discussed in Section 7.4.3, where a fixed gain controller was used for the first 0.2 s after the first step was commanded. The self-tuning controller was then utilised from t = 0.3 s onwards. The fixed gain controller used was a pole placement law, using a model with $n_a = 1$, $n_b = 1$, and $n_d = 0$, operating at 100 Hz. The fixed model parameters for this were obtained through a priori off-line identification, in the same way as used to obtain the initial parameter estimates for the system identifier. Furthermore, the pole placement controller used the same desired closed loop poles as the self-tuning controller.

7.5.2 Principal Results at Nominal Conditions

The response of the self-tuning hybrid position/force controller performing the default task under the nominal conditions is shown in Figure 7.5. The desired response is also shown, but the controlled system follows the desired responses so closely that the two are almost coincident. The RMS error[†] between the actual and desired force response was 0.46 N, and for the position control loop it was 0.32 mm. The responses also show that there was little interaction between the position and force controlled directions, which was a problem for the fixed gain PID controller as reported in Chapter Six. The small bump seen on the position loop was entirely due to the short period of fixed gain control used at the start of the task. Furthermore, since the same desired closed loop poles are used in both fixed gain and self-tuning controllers, there is no obvious distinction in their responses under these default conditions. The corresponding controller outputs are given in Figure 7.6.

The response of the fixed gain PID hybrid position/force controller to the same task

[†] These errors are only defined when the self-tuning controller is operational, that is from t = 0.3 s onwards.

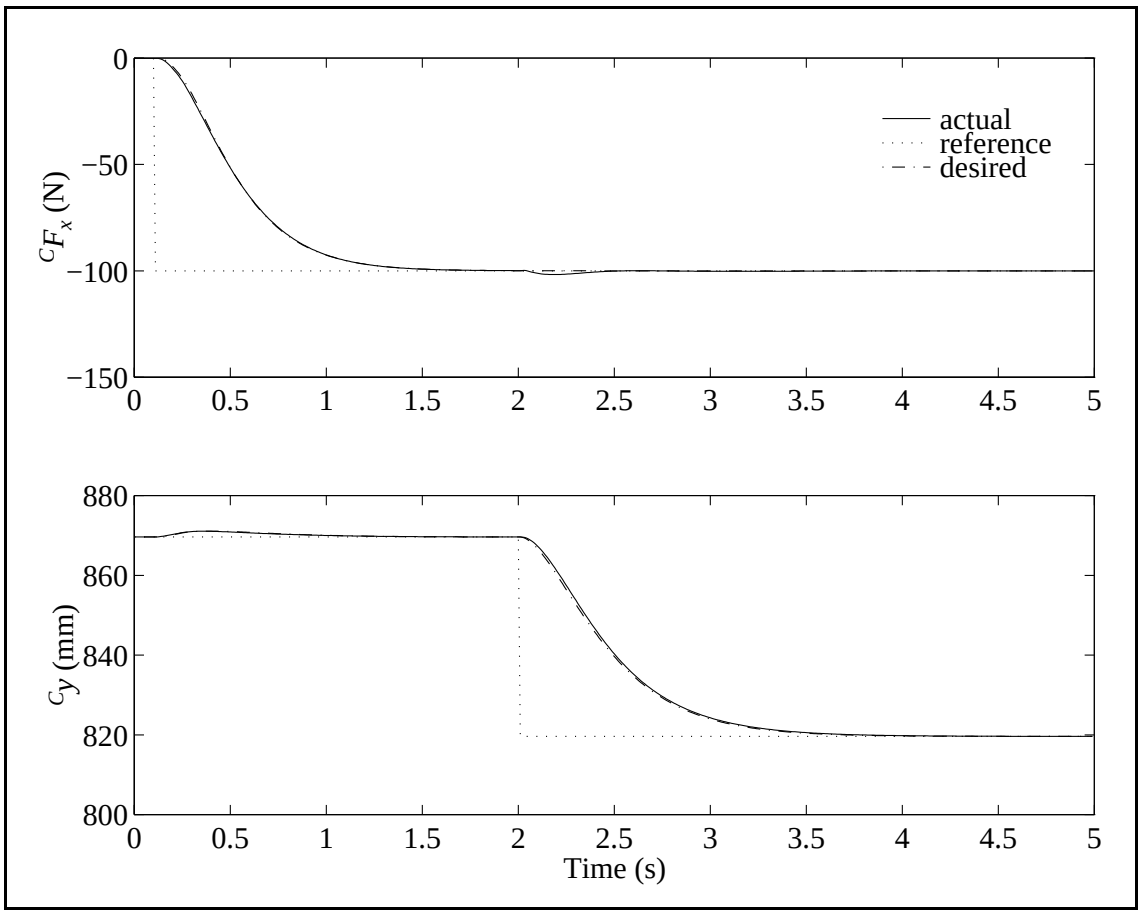

Figure 7.5 Self-tuning Hybrid Position/Force Control Results at Nominal Conditions

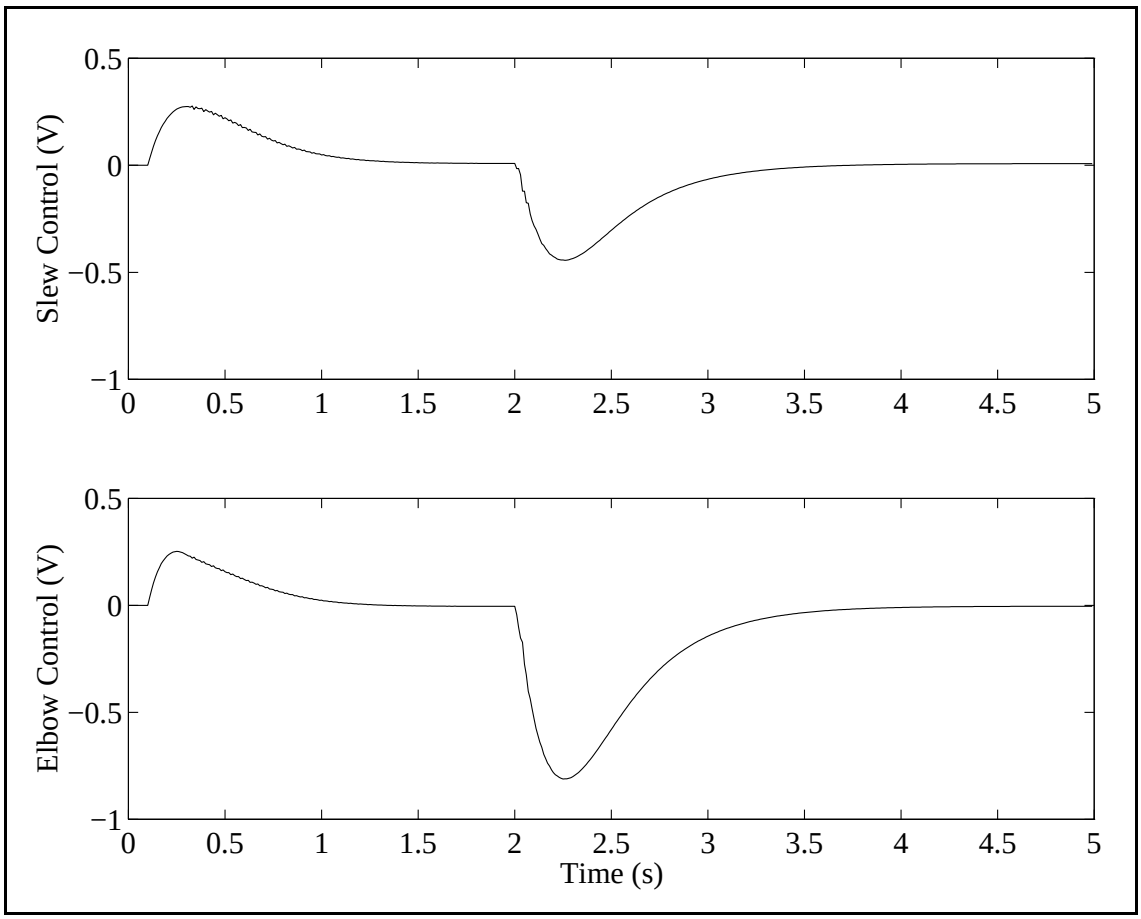

Figure 7.6 Control Signals for Self-tuning Hybrid Position/Force Controller

and conditions is shown in Figure 7.7. The responses have degraded from those presented earlier, with small oscillations now present. This has arisen since the operating conditions have changed from those it was configured for. Therefore, a direct comparison with the self-tuning controller cannot be made, since the two controllers have different tuning specifications.

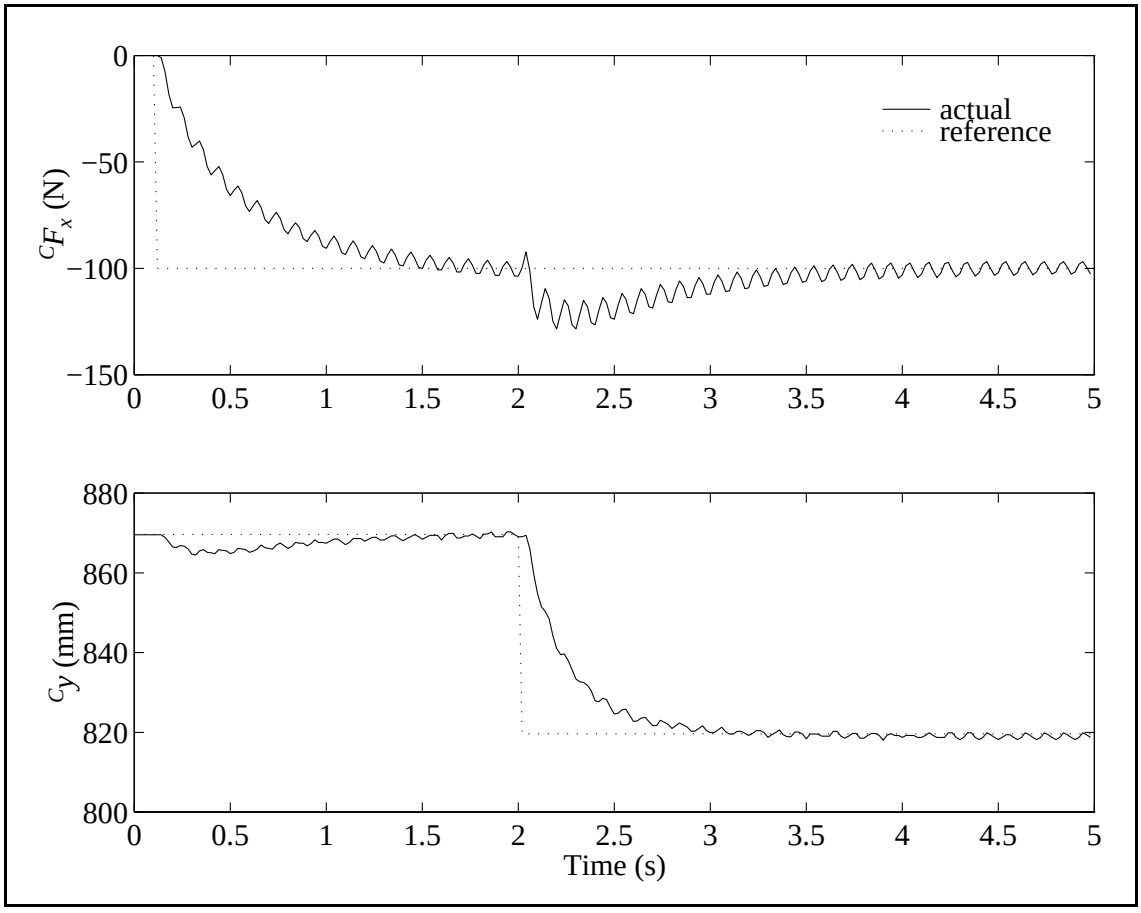

Figure 7.7 Fixed Gain PID Hybrid Position/Force Control Results at Nominal Conditions

A better choice of fixed gain controller for comparison purposes is the fixed gain pole placement (PP) controller employed during the first 0.3 s of the task prior to the self-tuning controller. This used the same desired polynomial matrix as the self-tuning controller, and so a direct comparison could be made.

The results for this fixed gain pole placement controller are given in Figure 7.8 and also follow the desired response. The RMS error between the actual and desired force response was 4.9 N, and for the position control loop it was 2.1 mm (cf. 0.46 N and 0.32

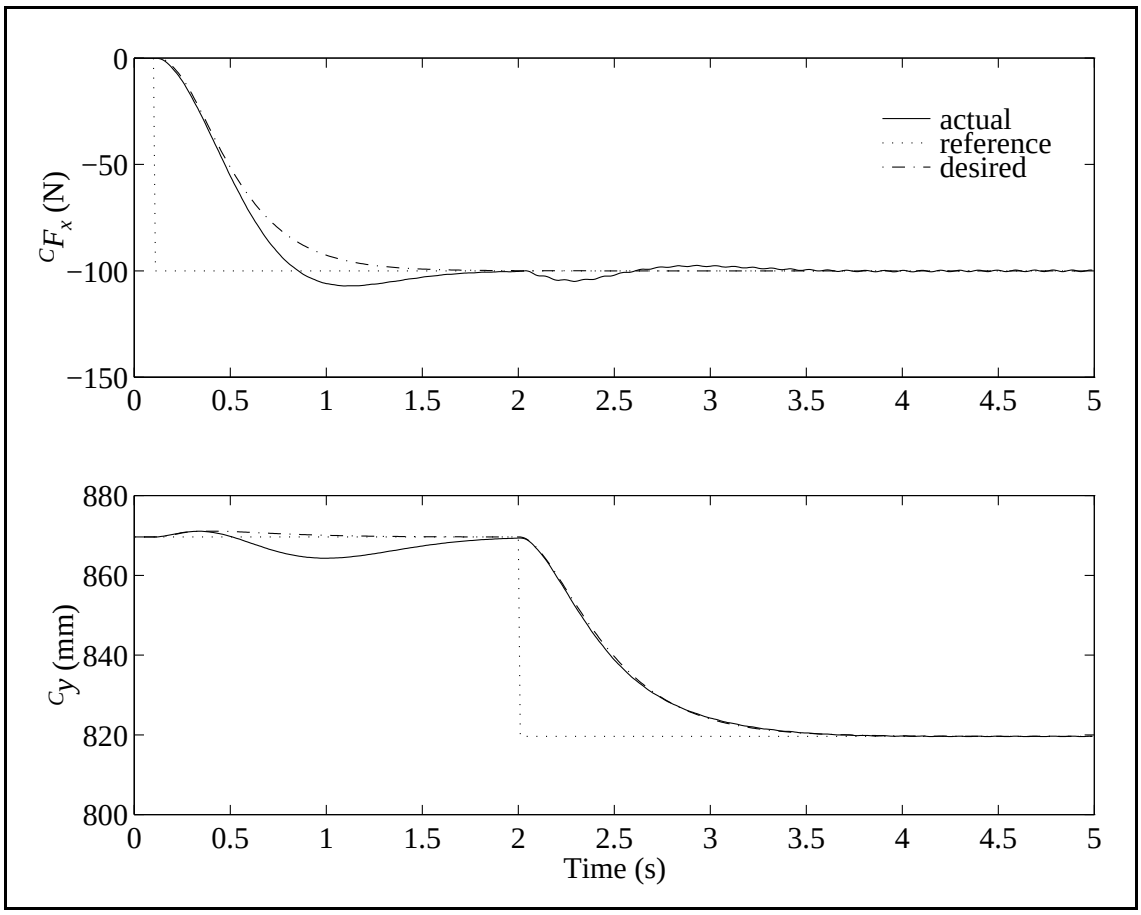

Figure 7.8 Fixed Gain PP Hybrid Position/Force Control Results at Nominal Conditions

mm for the same errors for the self-tuning controller).

7.5.3 Effect of Different Environmental Stiffnesses

The self-tuning controller needs to be able to cope with both unknown conditions and time varying conditions. These two issues were investigated separately, through two different types of task. Firstly, the ability to cope with unknown conditions was explored by repeating the simple task under a variety of different conditions from those it was configured for. Secondly, to investigate adaptation to time varying conditions, a longer task was used, where the nominal operating conditions were used at start up, but were then altered as time progressed. This section explores the ability of the self-tuning controller to accommodate unknown and changing environmental stiffnesses.

Figure 7.9 shows the response of the self-tuning hybrid position/force controller

when started under a wide variety of different stiffnesses, ranging from 5×10^2 N m^{-1} to 1×10^5 N m^{-1}, including the nominal stiffness of 1×10^4 N m^{-1}. This represents a useful and practical range. Some degradation in the response was present at the extremes of this range, including a significant correction at t = 1.3 s for the 5×10^2 N m^{-1} case, seen more clearly on the position loop response. The self-tuning controller could not cope with stiffnesses outside (both lower and higher) this range. However, it should be noted that the controller could probably cope with a wider range of unknown initial conditions by using the fixed gain controller for longer than 0.2 s. The coupling between the force and position loops was expected to increase as the stiffness increased, however, this effect was only minor.

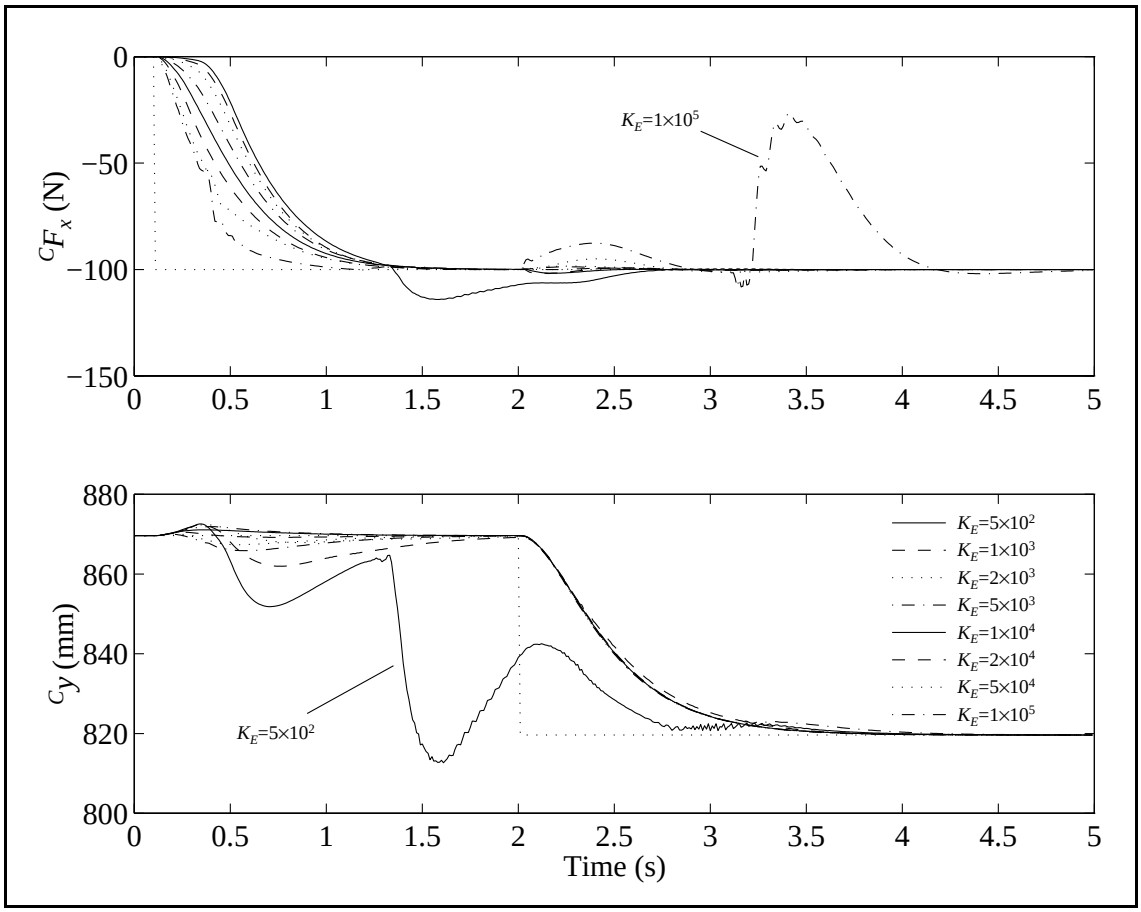

Figure 7.9 Self-tuning Hybrid Position/Force Control Results for Different Stiffnesses

The response of the fixed gain pole placement controller to the same set of stiffnesses were also obtained. For stiffnesses of 5×10^2 N m^{-1} and 1×10^5 N m^{-1} the responses were not stable, and so are not shown in Figure 7.10. For those stiffnesses that are stable,

there was still considerable degradation under this fixed gain scheme, with the lower stiffnesses resulting in a much slower force response. The larger stiffnesses, specifically that of 1×10^5 N m^{-1}, produced a highly oscillatory force response.

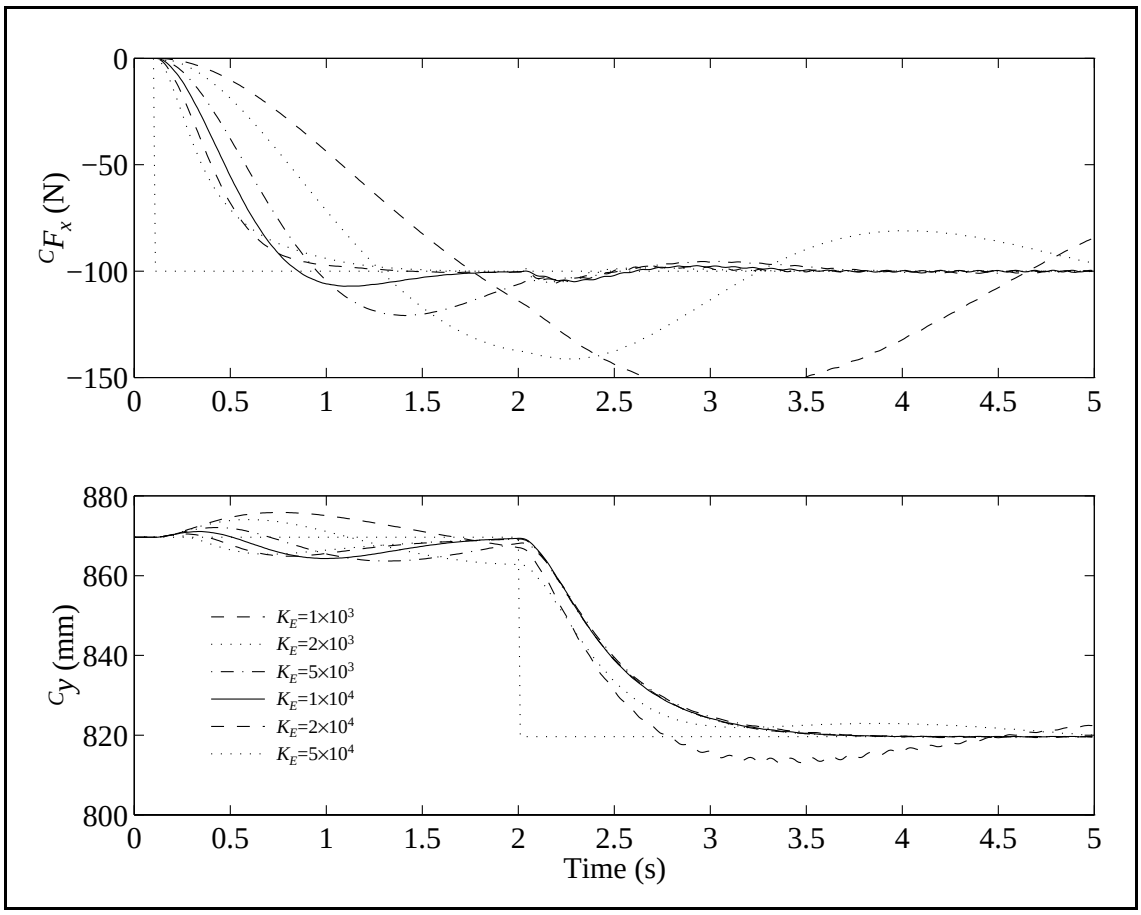

Figure 7.10 Fixed Gain PP Hybrid Position/Force Control Results for Different Stiffnesses

The performance improvement that the self-tuning controller provides, when compared to the fixed gain pole placement scheme, is quantified in Table 7.2. This lists the RMS errors between the actual and desired responses for the various stiffnesses used. The nominal conditions, the conditions at which both controllers were setup, give the smallest errors. The degradation of the control away from these nominal conditions is apparent, as is the improvement that the self-tuner provides over the fixed gain scheme.

The ability of the self-tuning controller to accommodate changing conditions was investigated using a sequence of force steps, stepping between -50 N and -150 N, while the environmental stiffness was gradually changed as time progressed. Figure 7.11 shows the

Environmental Stiffness (N m^{-1})	Self-tuning Hybrid Control		Fixed Gain Pole Placement	
	RMS Force Error (N)	RMS Position Error (mm)	RMS Force Error (N)	RMS Position Error (mm)
5×10^{2}	9.33	19.10	-	-
1×10^{3}	6.66	2.87	37.23	5.18
2×10^{3}	5.32	0.87	24.03	3.76
5×10^{3}	2.72	0.35	9.08	2.40
1×10^{4} (nom.)	0.46	0.32	4.93	2.08
2×10^{4}	3.13	0.65	5.22	1.96
5×10^{4}	6.06	1.12	5.96	1.91
1×10^{5}	21.88	1.72	-	-

Table 7.2 RMS Force and Position Errors for Different Stiffnesses

response of both controllers as the stiffness was increased logarithmically from 1×10^{4} N m^{-1} to 3×10^{5} N m^{-1} at t = 20 s, when the task was stopped. Figure 7.12 shows both responses as the stiffness was decreased logarithmically from 1×10^{4} N m^{-1} to 6×10^{2} N m^{-1} at t = 20 s. The position reference was maintained constant, at ${}^{C}y_{d}$ = 870 mm, and is not shown as it was subject to little variation.

Figure 7.11a shows that the self-tuner can adapt to these changing conditions, and maintains the system's response to closely match the desired response throughout the task, except for a small correction at t = 8 s. The response of the fixed gain pole placement controller, given in Figure 7.11b, has instabilities which occur at t = 15 s, corresponding to a stiffness of 1×10^{5} N m^{-1}. Figure 7.12 shows the self-tuning controller coping with the decreasing stiffness, whereas the fixed gain controller was unable to do so, with its response becoming slower as the stiffness decreases. Again a correction is present in the self-tuning controller response, at t = 14 s.

It might be expected that the changing environmental stiffness used in these results could be shown to correspond to changes within certain model parameters. However,

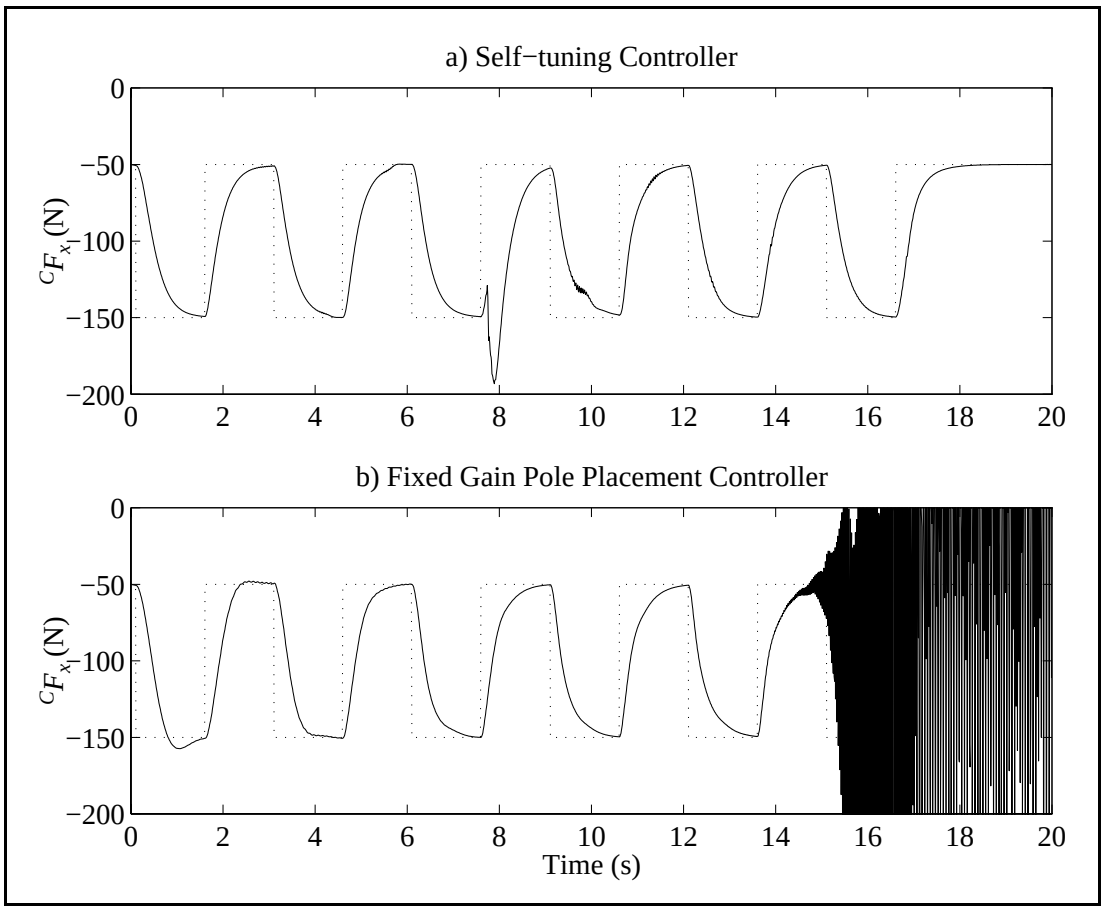

Figure 7.11 Hybrid Position/Force Control Results for Increasing Stiffnesses

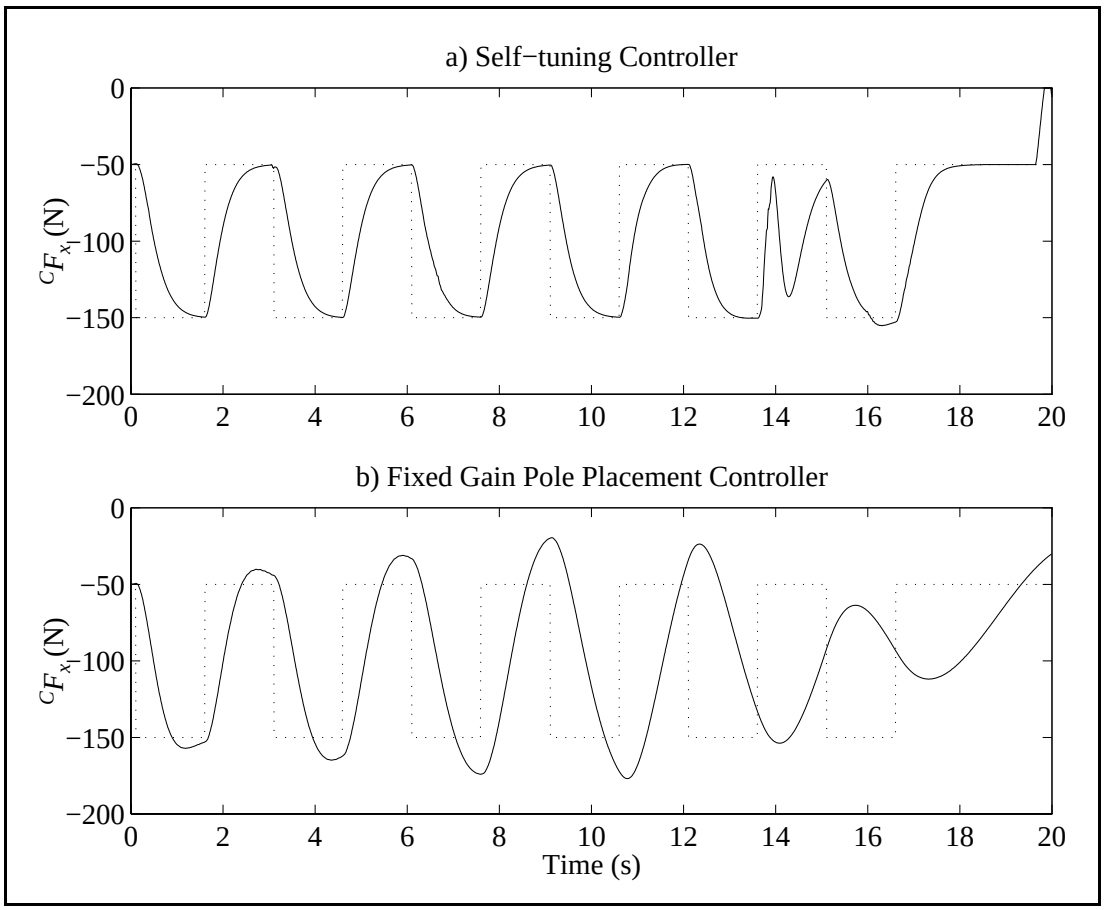

Figure 7.12 Hybrid Position/Force Control Results for Decreasing Stiffnesses

analysis of the estimated parameters did not reveal any clear corresponding trend. This may be due to the sequence of steps exciting the parameter estimation so much so that it masked these changes.

The range of stiffnesses that the self-tuning controller can cope with is large, equating to a 200 times increase in stiffness. Furthermore, these tests show that the self-tuning controller can adapt to a wider range of changing stiffnesses than it can accommodate as unknown initial conditions.

7.5.4 Effect of Different Contact Positions

A similar series of tests, examining the ability of the system to cope with both unknown initial conditions and changing conditions, were carried out for different contact positions. The contact position, ${}^{C}y$, was varied along the ${}^{C}x$ = -1050 mm axis, which encompasses a particularly wide range of manipulator dynamics, as discussed in Section 6.4.3 for the fixed gain hybrid position/force controller.

As in the previous section, the controller response to unknown initial conditions will be explored first. Figure 7.13 shows the self-tuning controller responses for three different initial contact positions, specifically ${}^{C}y|_{t=0}$ = 870 mm (the nominal condition), ${}^{C}y|_{t=0}$ = 370 mm and ${}^{C}y|_{t=0}$ = -130 mm. The position response plots have been scaled so that the three plots can be compared on the same graph. Two further initial conditions were tested, ${}^{C}y|_{t=0}$ = 1070 mm and ${}^{C}y|_{t=0}$ = -630 mm, however the self-tuning controller could not cope with such large differences from the nominal conditions that it had been setup to operate at.

The fixed gain pole placement hybrid position/force controller was operated at the same five different initial contact positions. As with the self-tuning controller, the fixed gain scheme did not work correctly for ${}^{C}y|_{t=0}$ = 1070 mm, as the force control could not maintain contact with the environment. This point is close to the edge of manipulator's workspace, and hence control is particularly difficult as the Jacobian is approaching a

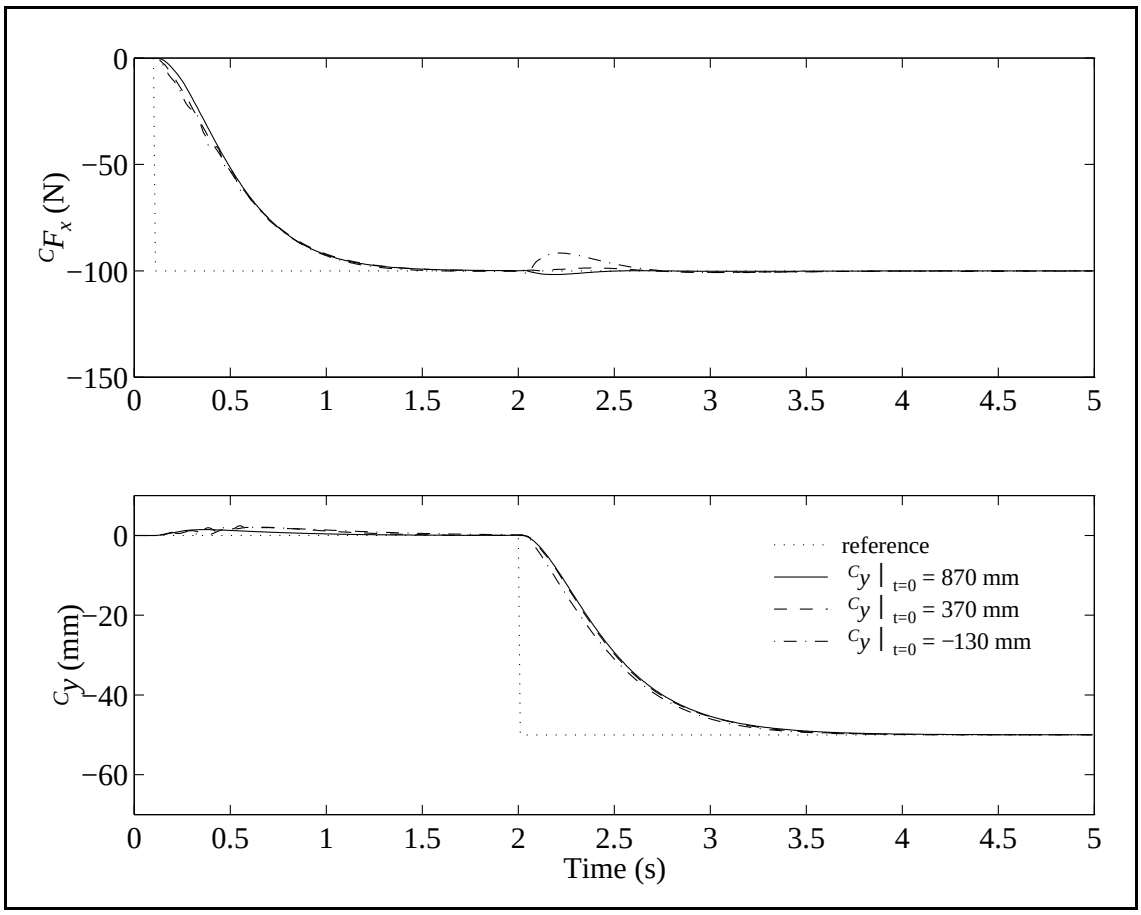

Figure 7.13 Self-tuning Controller Results for Different Contact Positions

singularity [2.1]. The responses for $^{C}y|_{t=0}$ = 370 mm and more notably for $^{C}y|_{t=0}$ = -130 mm, have significant oscillations in both the position and force responses. However, at the next point along the ^{C}x-axis, $^{C}y|_{t=0}$ = -630 mm, the controller responses actually improve. This implies that there is a central region along the ^{C}x = -1050 mm axis, that has significantly worse responses than those either side of it. Figure 7.14 shows the various controller responses, except for $^{C}y|_{t=0}$ = -130 mm and $^{C}y|_{t=0}$ = 1070 mm. The various effects identified here are more clearly seen in the results that follow.

The RMS errors between the actual and desired responses for both the self-tuning and fixed gain controllers for the various contact positions are given in Table 7.3. The self-tuning responses that did work followed the desired response closely, as indicated by the small errors. Those responses away from the nominal conditions suffered some degradation, but this was only minor. The fixed gain pole placement controller did not perform as well

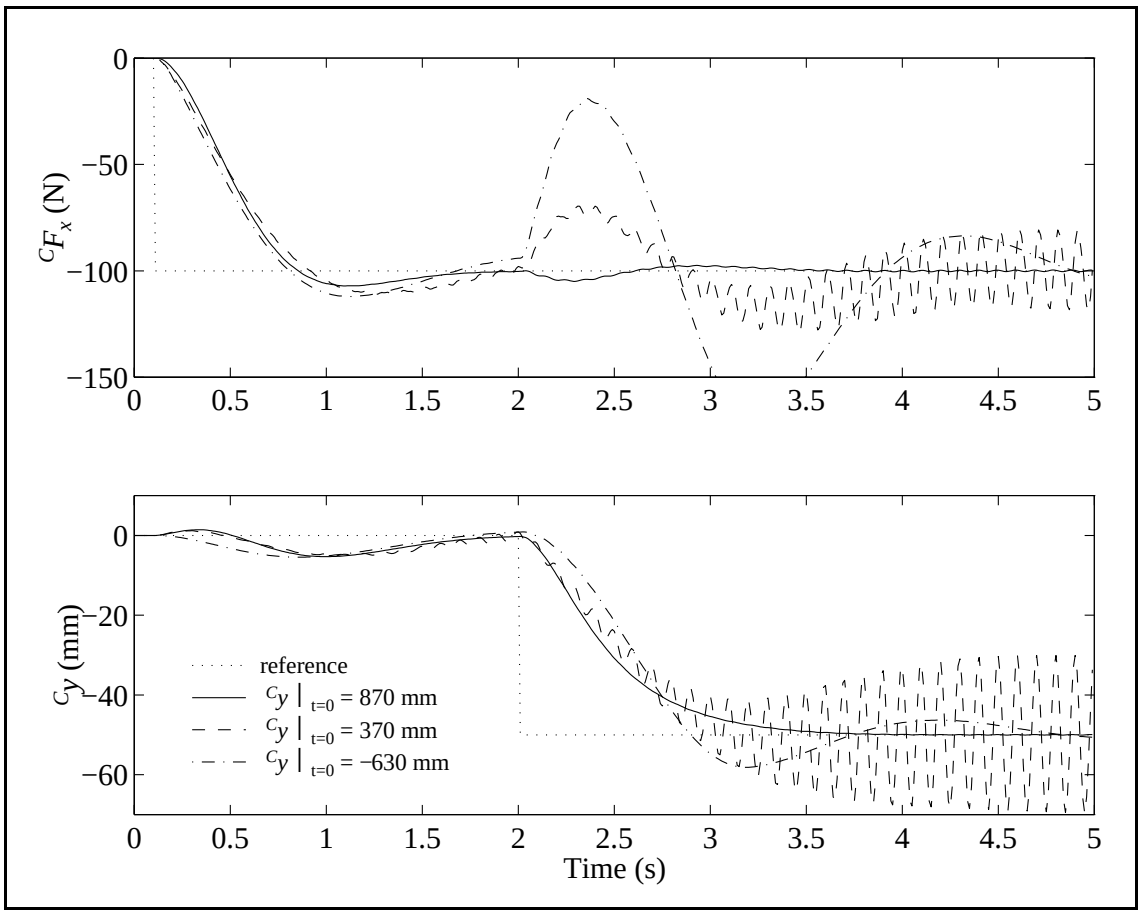

Figure 7.14 Fixed Gain PP Controller Results for Different Contact Positions

as the self-tuning controller, suffering significantly more degradation. This behaviour away from the nominal conditions is similar to that noted in the experimental results presented in Section 6.4.3. The degradation within the central region, as mentioned earlier, can be more clearly seen in Table 7.3. The errors for $^{C}y|_{t=0}$ = 370 mm and $^{C}y|_{t=0}$ = -130 mm become increasingly worse, however, those for $^{C}y|_{t=0}$ = -630 mm do show some improvement.

The self-tuning controller was then examined in terms of its ability to cope with changes in contact position. This was carried out using a sequence of steps in commanded contact position $^{C}y_d$, both increasing and decreasing away from the nominal condition.

Figure 7.15 shows the response of both controllers to four successive +50 mm steps in contact position, resulting in a final position of ^{C}y = 1070 mm, as used as an initial contact position previously. The plots show that the self-tuning controller copes with this well, adapting to the changing conditions and maintaining the system's response to match

Initial Contact Position, ${}^{C}y\|_{t=0}$ (mm)	Self-tuning Hybrid Control		Fixed Gain Pole Placement	
	RMS Force Error (N)	RMS Position Error (mm)	RMS Force Error (N)	RMS Position Error (mm)
1070	-	-	103.89	18.44
870 (nom.)	0.46	0.32	4.93	2.08
370	1.10	0.38	13.62	8.13
-130	2.40	0.63	75.07	15.56
-630	-	-	33.65	5.00

Table 7.3 RMS Force and Position Errors for Different Contact Positions

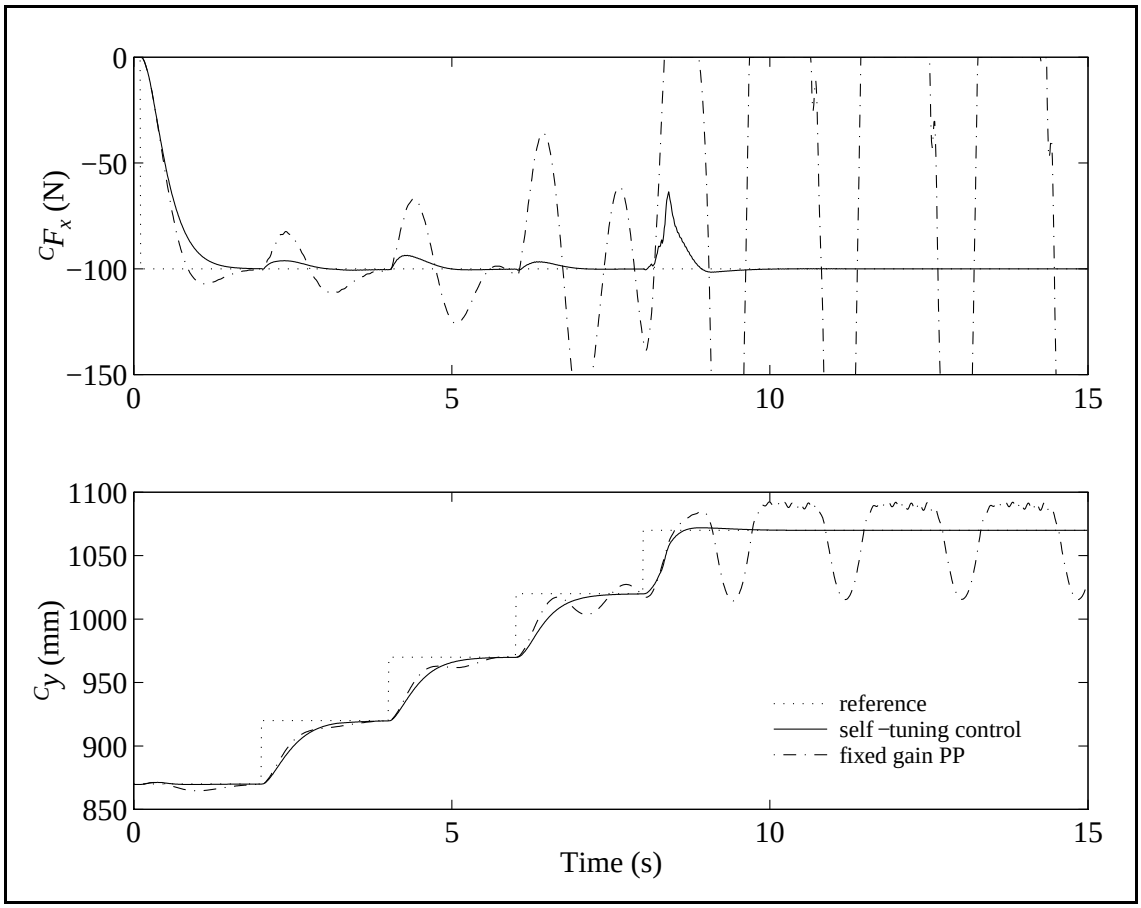

Figure 7.15 Hybrid Position/Force Control Results for Increasing Contact Position

the desired response throughout the task. This is particularly relevant since the final step was a condition that the self-tuning controller could not cope with when used as an initial contact position. The degradation of the fixed gain controller is evident from the response, particularly for the last step, where both position and force responses became unstable and

contact could not be maintained. As mentioned previously this step takes the manipulator close to the boundary of its workspace, and indeed the boundary constraint at ${}^{C}y$ = 1099 mm, can be seen in the plot as a saturation on the fixed gain position response.

The responses of the self-tuning and fixed gain controllers to decreasing steps in commanded position are shown in Figures 7.16 and 7.17 respectively. These cover a much wider range of positions than the previous test since the manipulator's workspace extends much further in that direction. The steps used are of the same magnitude (50 mm) and duration (2 s). Again the self-tuning system coped with the changing conditions well, maintaining a consistent response for both position and force. The fixed gain controller did not cope with this task as well, and the oscillatory behaviour of the system over the central portion of the workspace, as mentioned previously, can be clearly seen in both the force and position responses. The response did become more stable towards the end of the task, but the degraded response accounts for about 65% of the range of positions. This central region of poor control for the fixed gain controller was not identified with the experimental system, since the range of positions explored was not sufficiently wide.

The results in Figures 7.15 to 7.17 show that the self-tuning system copes well over a wide range of contact positions, whereas the fixed gain controller was unable to do so. These tests again reinforce the view that the self-tuning controller can adapt to a wider range of changing conditions, than it can accommodate as unknown initial conditions.

7.5.5 Effect of Different Levels of Commanded Force

The experimental work presented in Section 6.4.5 investigated the effects of different levels of commanded force upon the response of the hybrid position/force controller. The level of applied force alters the manipulator's dynamics due to the nonlinear torque/current characteristics shown in Figure 3.6. However, the range of forces used in Section 6.4.5 was restricted due to sensor saturation and any variations in the controller

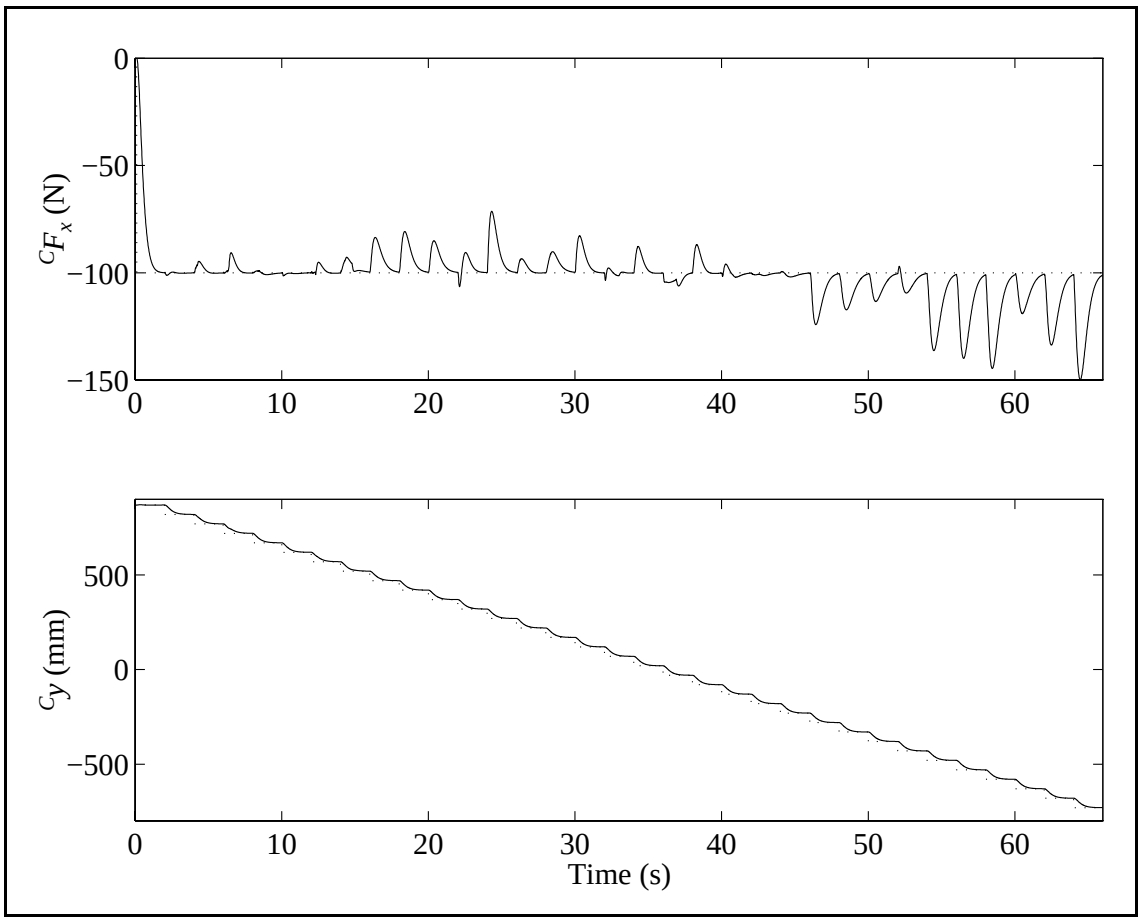

Figure 7.16 Self-tuning Controller Results for Decreasing Contact Position

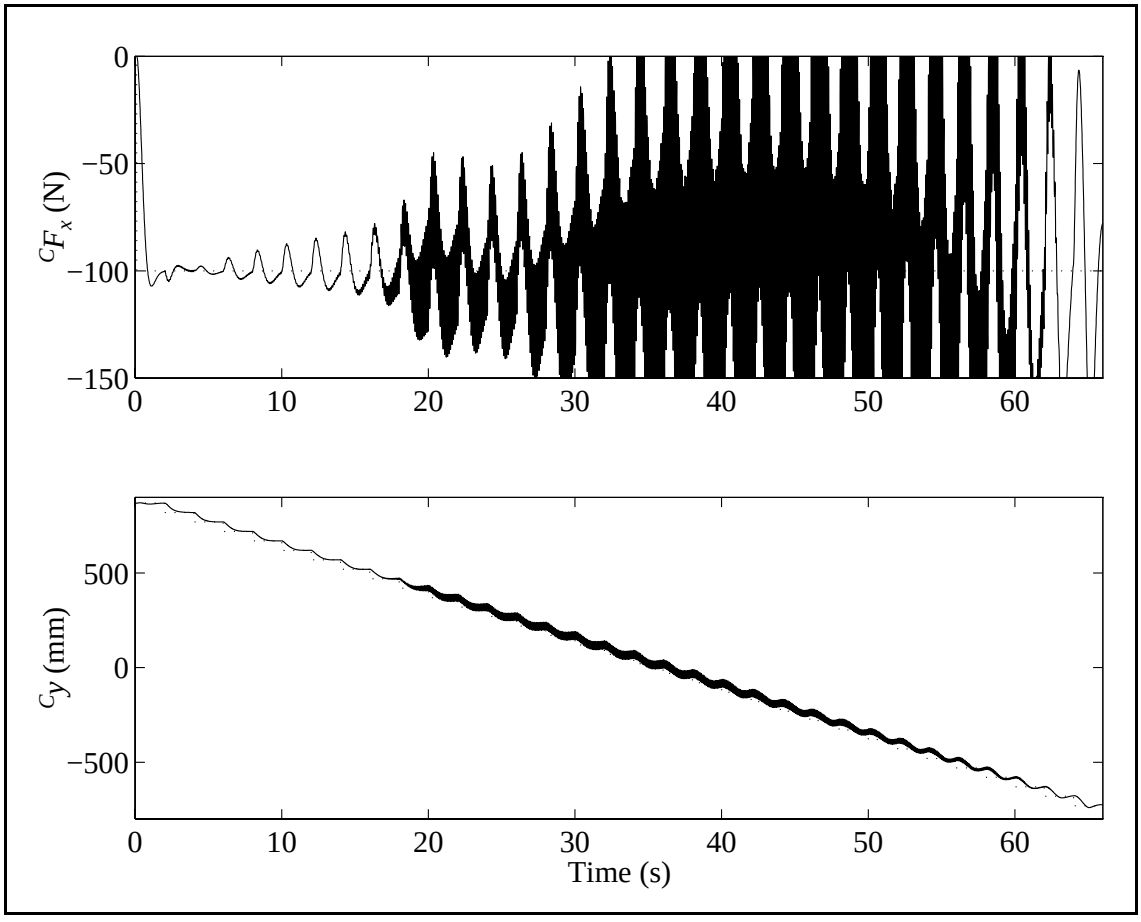

Figure 7.17 Fixed Gain PP Controller Results for Decreasing Contact Position

response were not pronounced. This section re-examines this, looking at the controller response over a much wider range of applied forces.

Figure 7.18 shows the response of both the self-tuning hybrid position/force controller and the fixed gain pole placement scheme to a series of steps in commanded force, starting at 0 N and increasing to -1350 N, stepping -150 N every 1 s. The self-tuning controller has a consistent response throughout this task, and there is minimal disturbance on the position loop. The fixed gain controller response shows degradation in both force and position control as the level of applied force increases.

These results again demonstrate the improvements that the self-tuner provides over the fixed gain hybrid position/force controller.

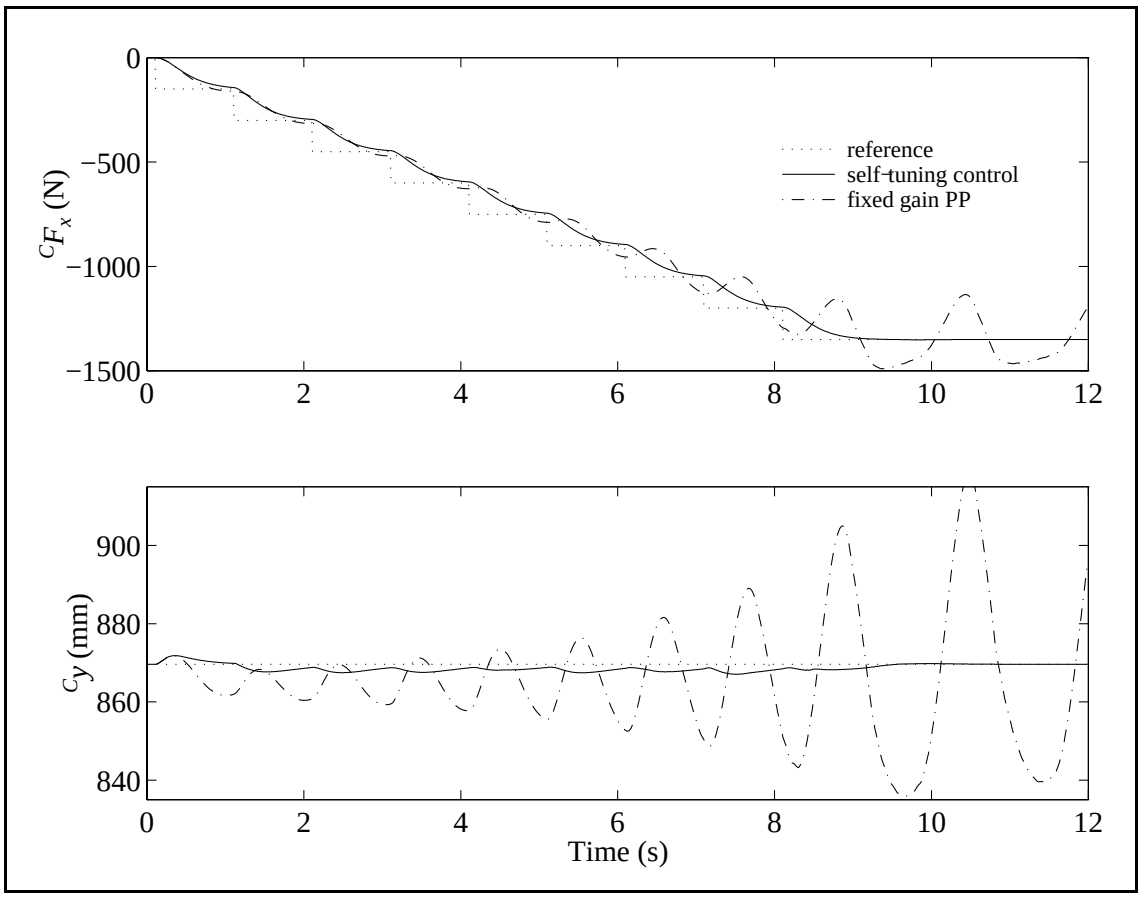

Figure 7.18 Hybrid Position/Force Control Results for Increasing Applied Force

7.5.6 Effect of Different Hydraulic Fluid Compressibility

Another factor that influences the manipulator's dynamics is the compressibility of

the hydraulic fluid, specified as the effective bulk modulus, β. This parameter changes with temperature, entrapped air and hose wall flexibility, as discussed in Section 3.4.2. It is difficult to carry out experiments that investigate this factor, as it is problematic to quantify and influence the compressibility of the fluid. However, the use of simulations allow this to be investigated.

The self-tuning and fixed gain hybrid position/force controllers were operated with different values of β, specifically 17×10^8 N m^{-2}, 7×10^8 N m^{-2} (the nominal conditions) and 3×10^8 N m^{-2}, representing a practical range of bulk modulus. The controllers were operated using these conditions both as unknown initial conditions and as time varying conditions, similar to those tests performed to investigate the effect of different stiffnesses.

Both the self-tuning controller and fixed gain pole placement controller coped well with these variations, with no noticeable degradation being observed in either. However, it should be noted that the original fixed gain PID hybrid position/force controller did start to exhibit oscillations for a bulk modulus of 3×10^8 N m^{-2}.

7.5.7 Effect of Using a Modified Reference Trajectories

The use of a ramped, rather than stepped, position reference was shown to be beneficial for the experimental fixed gain PID hybrid position/force controller in Section 6.4.4. This section extends this to the self-tuning hybrid position/force controller.

A ramped position reference was used, which took 0.5 s to realise the 50 mm movement in the basic task described in Section 7.5.1. This did not yield the same benefits in reducing the coupling between the force and position loops as presented in Section 6.4.4, since the amount of coupling present when using the self-tuning controller (see Figure 7.5) was already small. One benefit that a ramped position signal did provide was that it generated smaller actuator signals.

As mentioned previously, any practical implementation of a hybrid position/force

controller should utilise ramped position references. However, this will inject less coupling so it may have a detrimental effect upon the self-tuning controller due to the lower levels of signal excitation. The employment of a covariance management technique should prevent this from being problematic in practice.

7.6 Comparison Between Self-tuning and Robust Control Schemes

Within the UNION project, under which this work was funded, it was required to compare a robust hybrid position/force controller, developed by LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectric de Montpellier), and the self-tuning hybrid control scheme presented here. These are two different approaches to coping with the same problem of plant uncertainties and variations, providing an interesting and useful comparison [7.1]. Such comparisons are seldom available since the models and control tasks used by one research group are rarely the same as those used by others. This work was intended to address this and allow a direct comparison.

7.6.1 Description of VSC-HF Controller

The robust hybrid position/force controller developed at LIRMM was based on Variable Structure Control (VSC), a robust control scheme which does not require knowledge of the robot's dynamic parameters. The system is forced to behave according to a predefined equation, represented in state-space by a sliding surface. This equation includes a nonlinearity (sign function) which induces switching in the state variables (position and velocity of the manipulator). To eliminate this chattering phenomenon, the frequency of these oscillations was increased far beyond the bandwidth of the controlled system, by the introduction of an additional high frequency (-HF) element.

The VSC-HF controller demonstrated good robustness for Cartesian position control, however it was not very robust for force control with respect to variations in the

stiffness of the environment. To cope with this, the concept of a Virtual Environment was used to minimise the influence of the real stiffness.

A more detailed description of the scheme developed can be found in [7.1].

7.6.2 Comparison of Results

The two controllers were compared under a variety of different operating conditions, similar to those presented in Section 7.5. Both performed well throughout the range of conditions studied, with the VSC-HF controller performing better at different positions in the workspace, while the self-tuning controller coped with changes in stiffnesses better. Both proved superior to the conventional fixed gain controllers that had previously been investigated [1.5]. The main findings from the study were :-

- both controllers had difficulty in operating at the extreme edge of the manipulator's workspace (cf. Section 7.5.4).
- the VSC-HF controller coped with an initial contact position of ${}^{C}y|_{t=0}$ = -630 mm, whereas, as mentioned in Section 7.5.4, the self-tuning controller could not.
- the self-tuning controller could accommodate a wider range of environmental stiffnesses than the VSC-HF scheme, which exhibited significant degradation at both the upper and lower extremes of stiffnesses investigated. Figure 7.19 shows the response of both controllers at two different stiffnesses, specifically 5×10^{3} N m^{-1} and 5×10^{4} N m^{-1}, either side of the nominal stiffness used.
- the self-tuning controller also demonstrated greater robustness to variations in oil compressibility, as demonstrated in Figure 7.20.

The comparison also examined issues such as computational complexity and system commissioning, which can be as important as controller performance when selecting which

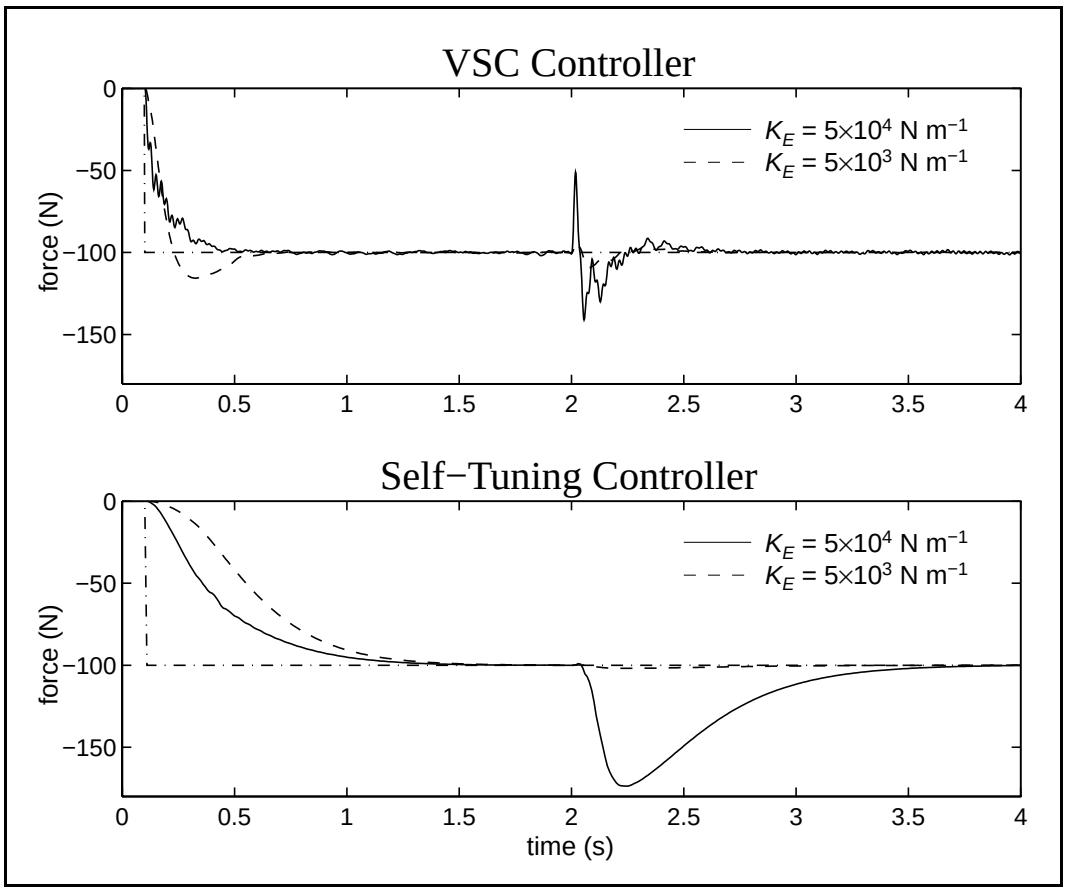

Figure 7.19 Effect of Different Stiffnesses on Self-tuning and VSC-HF Controllers

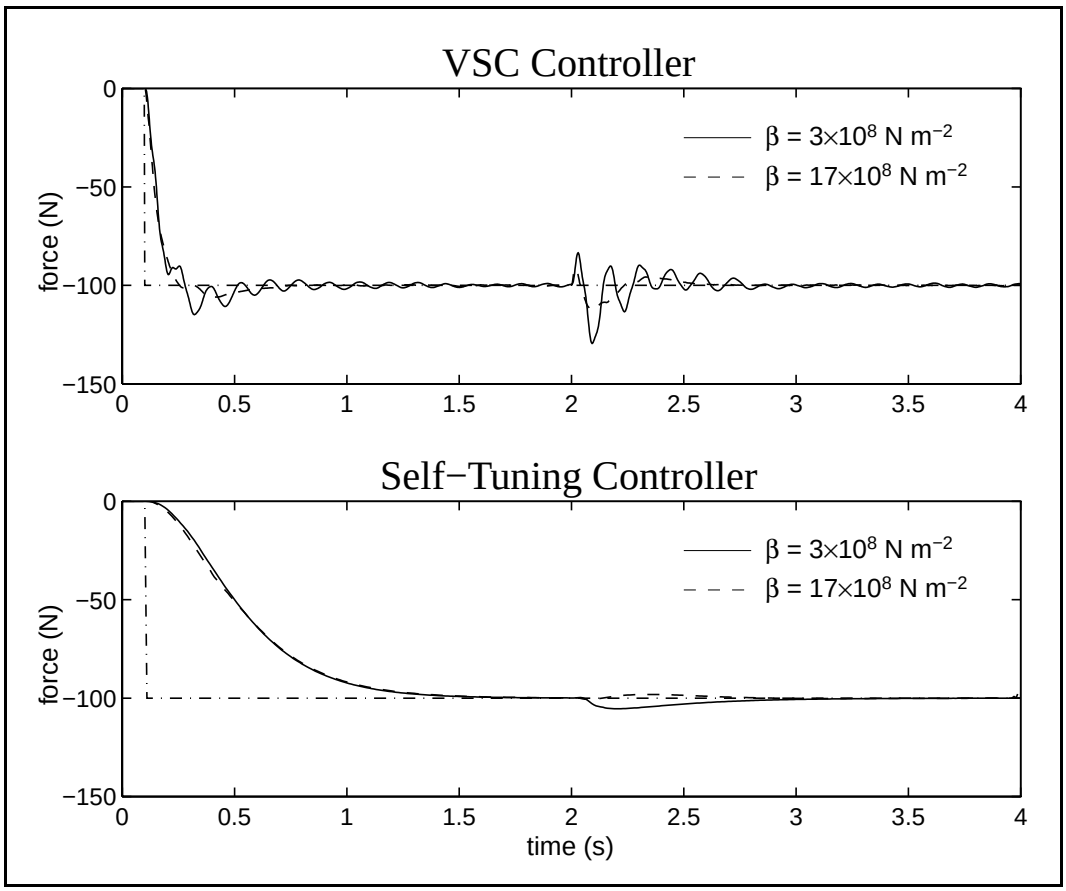

Figure 7.20 Effect of Oil Compressibility on Self-tuning and VSC-HF Controllers

control scheme to use.

In terms of computational effort required the self-tuning controller is significantly more expensive than the VSC-HF scheme. The VSC-HF controller required approximately 120 floating point operations, whereas the self-tuning controller required around 7300. However, since the VSC-HF controller requires a sample rate that was five times faster than the adaptive scheme, the ratio of required computing power becomes approximately 10:1. Both schemes are within the capabilities of current fast digital signal processors.

Both the VSC-HF and self-tuning controllers require correct initialisation before they can operate with a given system. For the VSC-HF, the gain for the position control can be set easily. However, without the virtual environment, any variation in the environmental stiffness requires adjustment to the force control gains. The introduction of the virtual environment alleviates this, simplifying implementation with minimal computational expense. The self-tuning scheme can also be used to facilitate the commissioning process, in that off-line analysis of previously recorded data can be used to initialise the parameter estimates. The self-tuning controller provides some additional flexibility, in that the desired responses are user specified and so can be changed as required for the given task.

One major practical drawback of the VSC-HF controller is that the control signal switches at a high frequency, which would place considerable stress on the hydraulic servovalves. The control signal from the self-tuning controller is much smoother (as shown in Figure 7.6) placing less demand, and thus potential wear, on the system.

A more detailed presentation of the various results can be found in [7.1].

7.7 Experimental Self-tuning Hybrid Position/Force Control

The self-tuning hybrid position/force controller developed in this chapter was applied to the experimental subsea hydraulic TA9 manipulator, restricted to two DOF acting in a horizontal plane, as described in Section 6.3. The practical implementation

followed the same staged development, as the simulation based system. Firstly, the system identification algorithm was implemented, then its operation was explored. The self-tuning controller was then implemented, initially for unconstrained Cartesian position control and then for the constrained hybrid position/force control problem.

The system identifier and self-tuning controller algorithms were integrated within the experimental fixed gain hybrid position/force controller that was described in Section 6.3.3. This enabled the same previously developed transformations and coordinate system to be utilised. The MIMO self-tuning controller was integrated into the sequential DSP operation, in the same way as the fixed gain hybrid position/force controller.

7.7.1 Practical MIMO System Identification

The MIMO Bierman UD factorisation RLS system identification algorithm was implemented on the DSP. The five steps of the algorithm, as listed in Section 4.2.2, were split over different read cycles in the sequential controller operation, allowing the computational burden to be distributed. Specifically, the Jacobian, kinematics and force transformations were evaluated as soon as the correct data was available. Steps 2 and 3 of the RLS algorithm were then carried out at the next read cycle, and step 4 was calculated during another. The specific sequence used depends upon the order in which the outputs are read and when the inputs are required. This distributed evaluation gave execution times of 22.7 ms for the MIL-RLS, and 2.86 ms for the BUD-RLS algorithms running on the DSP, cf. Table 5.1. This indicated that the DSP had sufficient computing power to achieve the target 100 Hz sample rate when using the BUD-RLS algorithm.

The various operational issues associated with the system identifier were then explored. Firstly, test signals were applied to the force and position references, whilst under fixed gain control, and the responses captured. Off-line identification, using the same algorithm implemented as MATLAB code, was then carried out on this data to examine the

effects of different operational factors. This was also used to verify the correct operation of the on-line system identifier, which was found to operate correctly.

As with the simulated system, the parameter estimates were initially set to match an integrator, but subsequent tests utilised initial parameters estimates obtained from previous off-line identifications. Different values of forgetting factor (between $\lambda = 0.999$ and 0.99) were tried, as were different values of $P(0)$ and different methods of initialising the regression vector. A typical set of estimates obtained are shown in Figure 7.21 for different values of forgetting factor.

The experimental MIMO system identifier could produce converged parameter estimates, however the main observation was that it was more sensitive to the initial parameter estimates than the simulation version had been. Furthermore, the increased sensitivity of this system is also apparent when Figure 7.21 is compared to Figure 5.7, which shows a similar plot for the experimental SISO system identifier. The effect of including or omitting the dc offset in the manipulator control signals was also investigated, and found to have little effect on the parameter estimates.

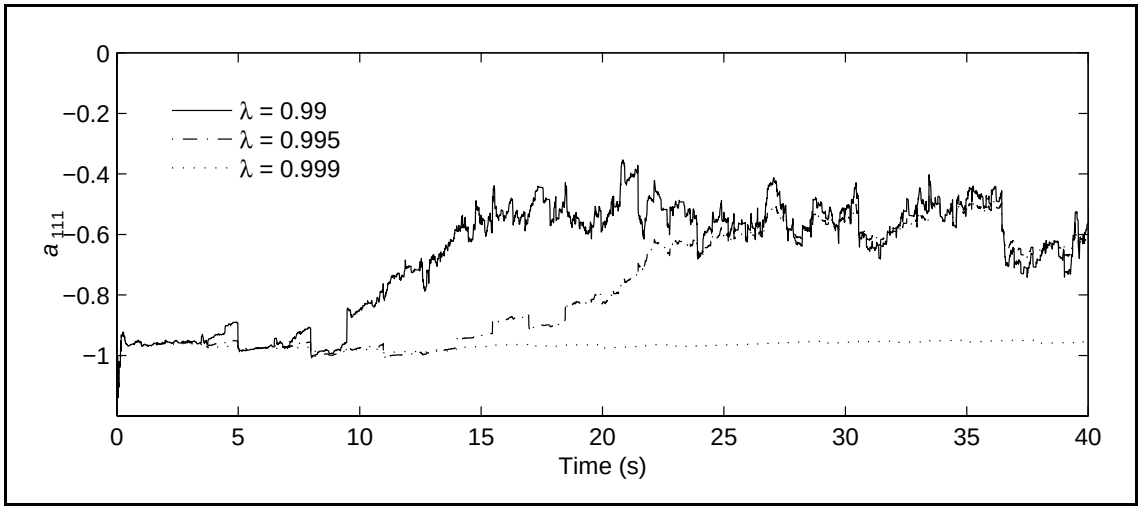

Figure 7.21 Effect of Different λ on the a_{111} Parameter Estimate

Nevertheless even with the increased sensitivity of the system identifier, the mean a priori prediction error was still close to zero indicating unbiased estimates. Its RMS value

was also small indicating that the model was able to track the system well. For a typical system identification test, mean a priori prediction errors of 0.0075 mm for the position loop and 0.022 N for the force loop were obtained, with the RMS error being 0.4 mm and 1.5 N respectively.

As with the simulation study no covariance management was employed within the system identification algorithm. This may have helped to make the system less sensitive.

7.7.2 Development of the Self-tuning Hybrid Position/Force Controller

The implementation of the experimental self-tuning controller did not result in an operational hybrid position/force controller. The self-tuning controller was successfully implemented within the sequential DSP operation and linked to the system identification algorithm. The controller proved to be realisable at the desired 100 Hz sample rate.

The controller was initially tested by configuring it to use the fixed parameter estimates used to initialise the system identification. This yielded a fixed gain pole placement controller, similar to the one used in Section 7.5, but using higher order models and hence higher order controllers. These fixed gain pole placement controllers produced responses that corresponded to the user specified desired polynomial, $T(z^{-1})$. This verified that the self-tuning controllers were implemented correctly, and it also allowed a few key operational issues associated with the controller to be explored.

Firstly, the choice of $T(z^{-1})$ was crucial to the correct operation of the controller. It was observed that specifying the desired poles at 0.97 and below, produced a response that was too oscillatory and that quickly became unstable. Further, poles at 0.99 and above proved too slow and control was poor. Good correlation between the controller response and the desired response was obtained when using desired poles at 0.98. Secondly, it was observed that if more, or indeed fewer, than two desired poles (as used throughout this work) were used, then the fixed gain pole placement controller would not work correctly.

The effect of using the alternative incremental self-tuning controller structure, shown in Figure 4.3, was also investigated. This is illustrated in Figure 7.22, which shows the response of a fixed gain pole placement controller with the usual structure, as in Figure 4.2, and the response of the alternative controller structure. Clearly, this latter form is better suited to this application, as it has less overshoot and follows the desired response much more closely.

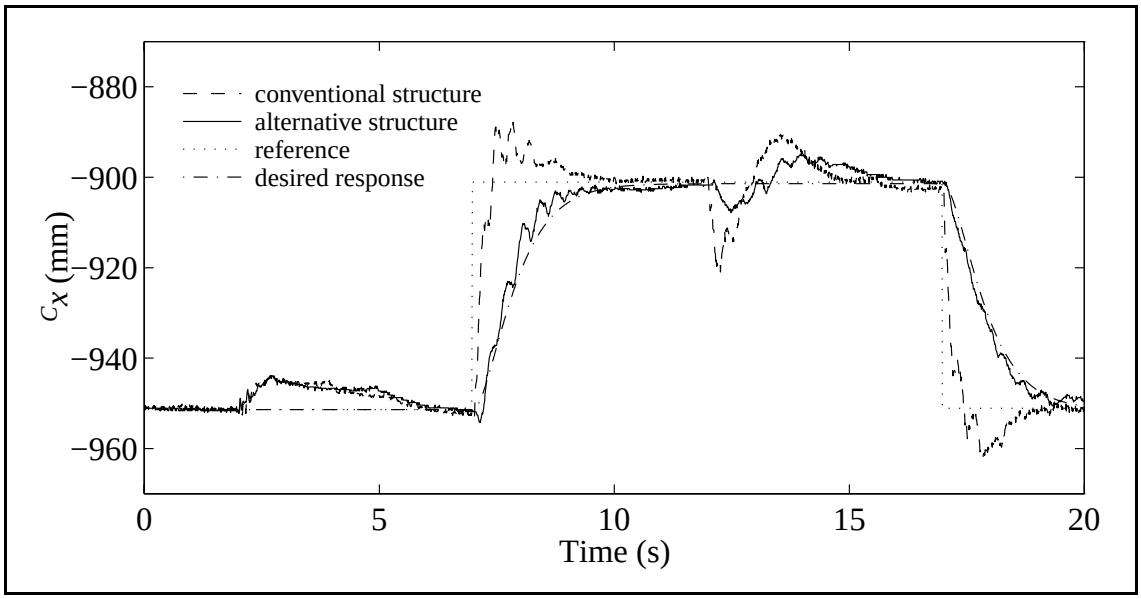

Figure 7.22 Response of Different Experimental Controller Structures

Unfortunately, the experimental self-tuning hybrid position/force controller could not be made to operate successfully. It is clear from the above discussion that the experimental implementation was far more sensitive to its initialisation and operation than was the simulation based controller. However, there are many modifications and alternative approaches that could be employed to give a working experimental self-tuning hybrid position/force controller. Some of these are discussed in Section 8.3.

7.8 Summary

This chapter has explored the self-tuning hybrid position/force controller, in terms of its development, operation, benefits and experimental implementation. The bulk of the

work presented in this chapter utilised a simulation of the manipulator. The need and validity of the simulations was considered and every effort was made to ensure that there was close correspondence with the experimental hydraulic manipulator.

There are many parameters within the robotic system that can change during a typical subsea task, such as contact position, environmental stiffness, applied force and hydraulic fluid compressibility. Variations in all of these were explored, looking at the ability of the controller to cope with them as unknown initial conditions and as time varying conditions. The self-tuning hybrid position/force controller was able to accommodate a wide range of the conditions examined, demonstrating significant benefit over the fixed gain controller used for comparison purposes. The range of unknown initial conditions could be extended by allowing the system to remain under fixed gain control for longer, allowing the system identifier to converge to more appropriate parameter estimates.

It should be recalled that Chapter Five demonstrated that the greatest changes in manipulator dynamics occur when the robot is moving with or against gravity. This effect was not examined here since the manipulator model, as in the practical system, was confined to the horizontal plane. Therefore larger variations in manipulator dynamics would be seen for the full 6 DOF system, and hence the self-tuning controller may be expected to provide even greater scope for improvement.

The comparison between the self-tuning controller and the VSC-HF controller developed at LIRMM proved interesting, identifying the strengths and weaknesses of both of these advanced control techniques. It is unusual to be able to compare two techniques so directly and the findings are valuable.

Although the benefits of the self-tuning hybrid position/force controller were proved in simulation, attempts to translate this to the experimental manipulator were not successful. The experimental investigation highlighted some practical issues with the system, however there are many more that can be explored to realise a working system.

Chapter 8

Conclusions

8.1 Summary

This thesis has demonstrated the benefits that both SISO and MIMO self-tuning control provides over the generally used fixed gain control for both constrained and unconstrained manipulator control problems. The benefits highlighted included; extending the range of operating conditions that can be accommodated; reduced a priori knowledge of the operating conditions required; reduced coupling for both SISO and MIMO systems; more repeatable and easier defined controller performance. These controllers also enable increasingly complex tasks to be carried out with greater reliability and speed than is currently achievable.

It should also be noted that limitations in this investigation, such as not exploring the manipulator's full payload capacity and confining the manipulator to the horizontal plane, suggests that the full 6 DOF manipulator could experience an even wider range of conditions than those considered here. Hence, the self-tuning controllers developed may provide even greater scope for improvement when applied to a manipulator in actual use.

However, these benefits are at the expense of increased controller complexity and more problematic initialisation. These disadvantages are the prime obstacles as to why the self-tuning controllers developed here will not find immediate use in the field. Nevertheless, this work does demonstrate the increased effectiveness and capabilities of underwater manipulation systems provided by these advanced controllers. It therefore

represents a significant step forward for offshore teleoperated manipulation systems, and may even contribute to bringing fully autonomous operation one step nearer.

The key feature of the controllers developed in this thesis was the focus on practical issues pertaining to a typical subsea hydraulically actuated manipulator. The specific problems associated with the TA9 manipulator that had to be accommodated included :-

- joint angles were obtained from noisy, low accuracy analogue potentiometers.
- ideally joint angle velocities should be used in the control system, but such instrumentation was not available on this industrial manipulator.
- considerable stiction was present in the hydraulic pistons and joints which acts to degrade control performance.

The practicalities considered applied not only to the selection and design of the controllers, but also explored real operational issues associated with the system identification and pole placement components of the self-tuning controller. Though the MIMO self-tuning hybrid position/force controller was not operational on the experimental system, its development was still constrained by what could be practically implemented. This work has not revealed any fundamental reason for a practical system being unfeasible, and suggestions for further work, discussed in Section 8.3, may yield a workable solution.

The development of the kinematic and dynamic models representing the TA9 hydraulic manipulator enabled the various controllers to be tried and tested on a realistic simulation before being applied to the experimental manipulator. This also provided a thorough understanding as to how hydraulic manipulators function, and the implications that this has upon position and force control. The validity of the models was paramount and every effort was made to ensure that there was close correspondence with the experimental hydraulic manipulator.

8.2 Author's Contributions

The most significant contribution to the subject of robot control that this thesis makes is that the advanced control schemes developed are ultimately applied to an industrial hydraulic manipulator, typical of those used in the offshore oil and gas industries. Therefore, the control strategies have been developed within the constraints of the real robot and do not rely on instrumentation that would only be found on laboratory manipulators, or on a simulated idealised model as is often used in much reported work. The controllers have been developed to allow the manipulator to operate in unknown and changing environments, and also to extend the capabilities of the manipulator by realising Cartesian position and force control.

The self-tuning pole placement controllers developed in this thesis are novel in both their application and their structure. Firstly, this is the first instance of such a SISO self-tuning pole placement scheme being used with an offshore hydraulic manipulator. Secondly, the application of a MIMO self-tuning pole placement scheme to the hybrid position/force control problem has not been reported before. The use of an incremental controller structure also has a degree of originality. The controllers developed here were placed in the context of previously proposed self-tuning manipulator controllers, summarised in Tables 2.1, 2.2, 4.1 and 4.2, illustrating how they complement previously reported work and their novel features.

8.3 Suggestions for Future Work

There are two main areas where the work presented in this thesis can be usefully extended; firstly, enabling the MIMO self-tuning hybrid position/force controller to work with the real hydraulic manipulator, and secondly, simplifying controller initialisation. Indeed controller modifications that are better suited to the practical application, may also yield a controller that is easier to initialise.

The main suggestions for further work are :-

- to investigate the various covariance management methods mentioned in Section 4.2.4, to ensure the long term correct functioning of the system identification algorithm. The use of a variable forgetting factor has been shown to offer some improvement in this regard [2.82]. A simple but possibly useful idea to explore, would be the use of different values of forgetting factor for the force and position parts of the MIMO system identifier.
- the square-root parameter estimation algorithm has some reported advantages [2.90] over the standard RLS algorithm. It would be useful to compare it to the Bierman UD Factorisation (BUD-RLS) algorithm used in this thesis.
- to realise the identified model and self-tuning controllers using the δ-operator [2.57]. This gives a better approximation of the underlying continuous time system than the backward shift operator, z^{-1}, used in Equation 4.22. Furthermore, it can also remove numerical problems which can arise when using the backward shift operator when sampling rates are too fast.
- the solution to the pole assignment problem (Equations 4.16 and 4.35 for the SISO and MIMO cases respectively) can be solved using more robust algorithms, such as Kucera's method [2.57]. These could be explored to see if they offer any benefits for this particular application.
- the benefits and effects of using the alternative self-tuning controller structure given in Figure 4.3, and demonstrated in Figure 7.22, should be explored further.
- there are certain elements within the manipulator system that are well known and that knowledge should be utilised explicitly within the self-tuning controller. For example, this could involve incorporating the inverse static actuator characteristics or Jacobian into the controller explicitly, so that these well known elements do not

have to be identified.

- restricting the *A* matrices in the MIMO identified process model to be diagonal (as used by Koivo [2.63]) was attempted in simulation, but this did not work as well as using the full matrix. However, this could prove beneficial for the practical self-tuning hybrid position/force controller.
- for the MIMO self-tuning controllers, the polynomial $T(z^{-1})$ representing the desired closed loop poles, were confined to being diagonal. It may be useful to investigate the effect of specifying the off-diagonal values as well.

Several complications with the development of a fully deployable system were noted but not investigated. These could also form areas of further work :-

- the extension of the hybrid position/force controller to a full 6 DOF system, including operation outside of the horizontal plane considered here. The ability of the system to follow curved surfaces could also be explored, as could its ability to cope with typical tools that may exhibit friction.
- using the developed force controllers within a commercial teleoperated manipulator system to increase its capabilities, for example providing guarded moves and automatic insertions.
- the extension of hybrid position/force control to a non-square system, where the number of actuators does not equal the number of controlled directions. This would not only exhibit the problem of multiple kinematic solutions, but would also require the extension of the MIMO self-tuning controller to such a system.
- the use of position, rather than velocity based measurements for manipulator control is known to be less ideal, since the manipulator is a rate driven system. It may be useful to investigate the benefits of using velocity signals, obtained directly from

tachogenerators, as this may identify sufficient benefits to justify their inclusion in future generations of subsea manipulators.

Finally, the comparison between the self-tuning and VSC-HF robust hybrid position/force controllers presented in Section 7.6, highlighted that both provided benefits, albeit in different areas. Therefore, one promising avenue of work would be to combine these two types of control, to give a system with the benefits of both. An initially interesting combination may be to use a robust controller for the position control part of a hybrid controller, and a self-tuning controller for the force control, as it has been demonstrated that these are the strong points of each.

Appendix A

TA9 Manipulator Model Parameters and Simulation Files

A.1 Introduction

This appendix gives the various parameters used within the TA9 manipulator model developed in this thesis. These parameters were obtained from specifications, physical measurements and estimation. Some of these parameters were also used directly in the experimental control scheme implemented on the actual manipulator, such controller gains, link lengths and angle limits.

Also given are the closed form model matrices for the restricted TA9 and the various files that constitute the SIMULINK simulation model.

A.2 TA9 Model Parameters

The manipulator kinematic and inertial parameters, obtained from Slingsby drawings and physical measurements :-

l_1 = 0.452 m

l_2 = 0.522 m

l_3 = 0.558 m (including length of ball transfer unit tool used)

w_1 = 0.17 m

w_2 = 0.14 m

w_3 = 0.12 m

m_1 = 18.49 kg

m_2 = 9.70 kg

m_3 = 7.84 kg

v_{j1} = 20 N m s rad^{-1}

v_{j2} = 20 N m s rad^{-1}

θ_1 range = 66.95° to 182.45°

θ_2 range = 0.57° to 102.32°

θ_3 = 14.48° (fixed wrist offset angle)

The hydraulic actuator parameters are :-

V_{t1} = 1.83×10^{-4} m^3

V_{t2} = 1.75×10^{-4} m^3

$A_{2(1)}$ = 10.67×10^{-4} m^2

$A_{2(2)}$ = 7.72×10^{-4} m^2

l_{a1} = 0.337 m

l_{a2} = 0.337 m

l_{b1} = 0.079 m

l_{b2} = 0.08 m

k_{leak1} = 8.476×10^{-14} m^5 N^{-1} s^{-1}

k_{leak2} = 7.417×10^{-14} m^5 N^{-1} s^{-1}

v_{p1} = 30 N m^{-1} s

v_{p2} = 30 N m^{-1} s

$\theta_{poffset1}$ = 204.11°

$\theta_{poffset2}$ = 21.80°

It should be noted that only viscous friction was used in the simulations. Coulomb friction was considered, however it made little difference to the controller response and made the simulation run excessively slowly, and hence was not used in the simulations.

The MOOG E777 series of flow control servovalves, used in the TA9 manipulator, are low cost equivalents of the E050-31. The following data was obtained from MOOG E777-006 (PDS E777 1.88) technical specifications :-

Signal:	±10 ma (parallel coils), ±5 ma (series coils)
Resistance:	1000 Ω per coil
Leakage:	< 4% rated flow at rated pressure
Flow:	6.35×10^{-5} m^3 s^{-1} (at 2000 psi load pressure)
	11.00×10^{-5} m^3 s^{-1} (no-load rated flow)
Max Pressure:	2.07×10^{7} N m^{-2} (3000 psi)
Resolution:	< 1.0% rated signal
Hysteresis:	< 3.0% rated signal
Rise time:	0.0035 s (at rated pressure, 3000 psi)
(to 90% of output)	0.0045 s (at 2000 psi)
	0.0060 s (at 1000 psi)
-3 dB point:	> 200 Hz (at rated pressure, 3000 psi)

These correspond to the following parameters for the first order approximation of the servovalve (±5 ma rated current, series connected coils) :-

$$K_v = 6.32\times10^{-5} = \frac{C_d \pi d_1}{\sqrt{\rho}}$$

K_i = 7.60×10^{-5} m mA^{-1}

τ_i = 0.0017 s

Hydraulic fluid parameters :-

density, ρ = 870 kg m^{-3}

bulk modulus, β = 7×10^8 N m^{-2}

supply pressure, P_s = 175×10^5 N m^{-2} (2500 psi)

Environment parameters for the hybrid position/force controller :-

K_E, environmental stiffness = 1×10^4 N m^{-1}

PID controller settings (simulation and practical) for the hybrid position/force control :-

K_{Pp} = 20.0

K_{Ip} = 1.5

K_{Dp} = 0

K_{Pf} = 0.002

K_{If} = 0.0004

K_{Df} = 0

τ_s, sample period = 0.01 s

Controller output limit = 8.188 V

Controller output quantisation level = 4.0×10^{-3} V

shoulder slew quantisation level = 6.39×10^{-4} °

elbow quantisation level = 5.11×10^{-4} °

force quantisation level = 0.326 N

A.3 Useful Conversions

Some useful conversions between metric, imperial and American units for pressure and flow :-

1 lb = 4.44822 N

1 in^2 = 6.4516×10^{-4} m^2

1 psi = 6894.76 N m^{-2}

1 bar = 1×10^5 N m^{-2}

1 gpm = 6.31×10^{-5} m^3 s^{-1}

1 lpm = 1.67×10^{-5} m^3 s^{-1}

1 cis = 1.64×10^{-5} m^3 s^{-1}

Note, gpm = gallons per minute, lpm = litres per minute, lb = pounds, in = inches, psi = pounds per square inch (lb in^{-2}), cis = cubic inches per second (in^3 s^{-1}).

A.4 TA9 Dynamic and Kinematic Model Matrices

This section gives the closed form model matrices for the restricted 2 DOF TA9 manipulator, derived using the conventional recursive Newton-Euler dynamic equations [2.1]. A generalised direct drive manipulator with an arbitrary number of joints is represented by the following *closed form* dynamic model, Equation 3.27 :-

$$\mathbf{T} = M(\Theta)\ddot{\Theta} + V(\Theta,\dot{\Theta}) + G(\Theta) + B(\dot{\Theta}) + {}^{C}J^{T}(\Theta)\,{}^{C}F$$

where $\mathbf{T}$ is the vector of joint torques, Θ is the vector of joint angles, M is the inertia matrix of the manipulator links, V is the Coriolis and centrifugal effects matrix, G is the gravity matrix, B is the vector of friction acting at each joint, ${}^{C}J$ is the Jacobian of the manipulator and ${}^{C}F$ is the vector of forces and torques at the end-effector.

The manipulator model developed assumed that each link was a homogeneous rectangular mass, with its centre of mass being halfway along its principal axes. The specific matrices used in the above model, are defined as follows :-

$$M(\Theta) = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

$$V(\Theta,\dot{\Theta}) = \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

$$G(\Theta) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$B(\dot{\Theta}) = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

where :-

$$M_{11} = \tfrac{1}{3}m_2 l_2^2 + \tfrac{1}{12}m_2 w_2^2 + \tfrac{1}{3}m_3 l_3^2 + \tfrac{1}{12}m_3 w_3^2 + l_2 m_3 (l_2 + l_3 c_3) + \tfrac{1}{3}m_1 l_1^2 + \tfrac{1}{12}m_1 w_1^2 + l_1^2 (m_2 + m_3) + l_1 l_3 m_3 c_2 c_3 + l_1 l_2 (2m_3 + m_2) c_2 - l_1 l_3 m_3 s_2 s_3$$

$$M_{12} = M_{21} = \tfrac{1}{3}m_2 l_2^2 + \tfrac{1}{12}m_2 w_2^2 + \tfrac{1}{3}m_3 l_3^2 + \tfrac{1}{12}m_3 w_3^2 + l_2 m_3 (l_2 + l_3 c_3) + \tfrac{1}{2}l_1 l_3 m_3 c_2 c_3 + l_1 l_2 (m_3 + \tfrac{1}{2}m_2) c_2 - \tfrac{1}{2}l_1 l_3 m_3 s_2 s_3$$

$$M_{22} = \tfrac{1}{3}m_2 l_2^2 + \tfrac{1}{12}m_2 w_2^2 + \tfrac{1}{3}m_3 l_3^2 + \tfrac{1}{12}m_3 w_3^2 + l_2 m_3 (l_2 + l_3 c_3)$$

$$V_1 = \left(\tfrac{1}{2}l_1 l_3 m_3 c_2 c_3 + l_1 l_2 (m_3 + \tfrac{1}{2}m_2) c_2 - \tfrac{1}{2}l_1 l_3 m_3 s_2 s_3\right)\left(\dot{\theta}_2^2 + 2\dot{\theta}_1 \dot{\theta}_2\right)$$

$$V_2 = -\left(\tfrac{1}{2}l_1 l_3 m_3 c_2 c_3 + l_1 l_2 (m_3 + \tfrac{1}{2}m_2) c_2 - \tfrac{1}{2}l_1 l_3 m_3 s_2 s_3\right)\dot{\theta}_1^2$$

$$B_1 = v_{j1}\dot{\theta}_1$$

$$B_2 = v_{j2}\dot{\theta}_2$$

and $c_1 = \cos(\theta_1)$, $s_1 = \sin(\theta_1)$, $c_{12} = \cos(\theta_1+\theta_2)$ etc. and the definitions of T, Θ and ${}^{C}F$ are as specified in Chapter Three. Furthermore, the forward kinematics of the manipulator is given by Equation 3.1 and the Jacobian, ${}^{C}J$, is given by Equation 3.2.

Two specific force transformations were used in the hybrid position/force control scheme. The first, the transformation from the sensor frame, $\{S\}$, to end-effector frame $\{4\}$, was defined within the ATI transducer controller as :-

```
SLIDER      0       0       1012  900    -300   900
```

following the conventions of rotations and units used within that device [6.1]. The second transformation, from frame $\{4\}$ to the constraint frame, $\{C\}$, is given by Equation 6.4.

A.5 MATLAB/SIMULINK Simulation and Modelling Files

This section gives the MATLAB M-files used to model the TA9 hydraulic manipulator, using the relationships developed in Chapter Three. The manipulator model represents the should slew and elbow joints, therefore restricting the model to operation in the horizontal plane, as detailed in Section 3.3. The files representing this model have been developed to be generic, and hence can easily be extended to more joints of the robot.

Figure A.1 shows the SIMULINK block diagram of the model, HYBRIDST.MDL. This is primarily a series of blocks representing the controller and manipulator model, together with blocks that generate the controller reference signals and those that save the model's outputs to the MATLAB workspace.

The files used to execute the simulation are listed at the end of this appendix. SIM2DATA.M is the file that initialised the various parameters used in the model, matching those defined in Section A.2, and this needs to be executed before a simulation is run. The manipulator model and controller are implemented as SIMULINK s-functions, specifically by the files LNK2_05H.M and RLSCONT2.M (for the MIMO self-tuning

hybrid position force controller) and CONTS.M (for the fixed gain PI hybrid position/force controller). The file SIMULATE.M was simply a convenient way to initialise, execute the simulation and store and display the results.

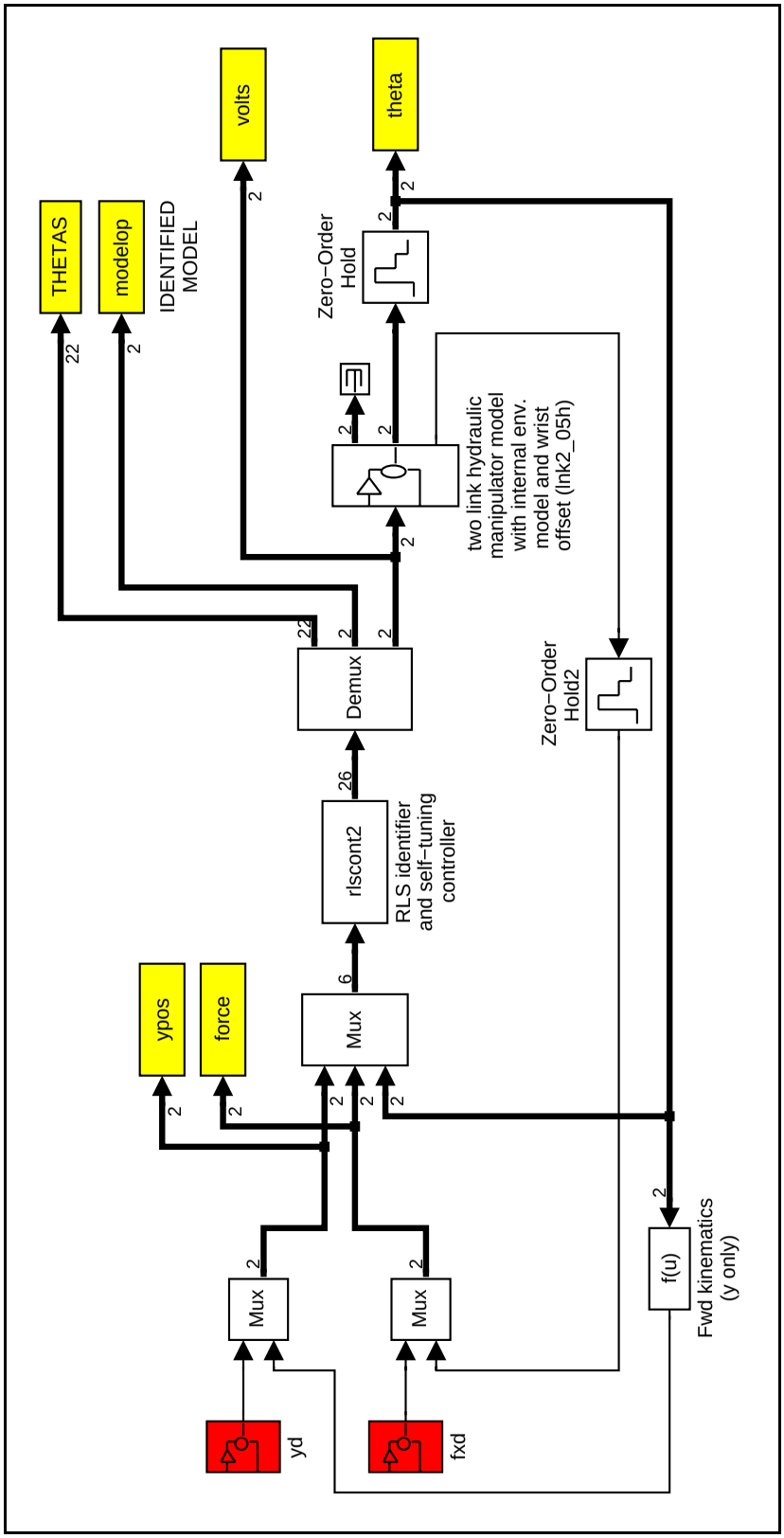

Figure A.1 SIMULINK Diagram for Self-tuning Hybrid Position/Force Controller

SIM2DATA.M

```
% two hydraulic link dynamics model simulation parameters - based on
% Slingsby Engineering Ltd (SEL) TA9 shoulder slew and elbow joints
% with 14.48 deg offset in wrist. NOTE : all angles etc are in RADIANS
global joints pm_ini x_ini KE ground theta_ini theta_dot_ini;
global F Fram A2 theta_poffset ki taui ps K KH;
global samp_per v_ini ypos_ini fxee_ini;
global KP KI KPP KPI;
global na nb nd nt lambda;
global THETA T1 T2 T3 T4 T5 FIXG0 FIXG1;
global PSI P U D;
model = 'hybridst'
joints = 2;
%
% link parameters
%
L = [ 0.452; 0.522; 0.558 ];                    % from SEL data
M = [ 18.49; 9.70; 7.84 ];                      % from SEL data
H = [ 0.17; 0.14; 0.12 ];                       % from SEL data
wrist_offset = 14.48*pi/180;                    % 14.48 deg
g = 0.0;
F = [ 20.0  0.0
      20.0  0.0 ];                              % estimated
theta_lim = [ 3.1844 1.1685
              1.7858 0.0099 ];                  % from SEL data
%
% hydraulic actuator parameters - shoulder slew & elbow
%
Vt = [ 1.83e-4; 1.75e-4 ];                      % estimated from SEL drawings
A2 = [ 10.67e-4; 7.72e-4 ];             % from SEL drawings
la = [ 0.337; 0.337 ];                          % from SEL drawings
lb = [ 0.079; 0.080 ];                          % from SEL drawings
theta_poffset = [ 3.5622; -0.3805 ];            % from SEL drawings
Fram = [ 30.0  0.0
         30.0  0.0 ];                           % [ from Traa ; not-known ]
kleak = [ 8.476e-14; 7.417e-14 ];               % measured
ps = 175e5;                                     % from MOOG data sheets
beta = 7e8;                                     % from Stringer - ideal = 17e8
%
% servovalve parameters - for MOOG E777-006 (series connected coils)
%
kv = [ 6.32e-5; 6.32e-5 ];                      % from MOOG data
ki = [ -7.60e-5; -7.60e-5 ]*0.6104;             % from MOOG data and servovalve amps (8.192V -> 5mA)
taui = [ 0.0017; 0.0017 ];                      % from MOOG data
%
% internal model constants - assuming homogeneous mass distribution
%
K = [ 0.5*L(3)*M(3)*g*cos(wrist_offset)+L(2)*g*(0.5*M(2)+M(3))
    -0.5*L(3)*M(3)*g*sin(wrist_offset)
    L(1)*g*(0.5*M(1)+M(2)+M(3))
    -0.5*L(1)*L(3)*M(3)*sin(wrist_offset)
    0.5*L(1)*L(3)*M(3)*cos(wrist_offset)+L(1)*L(2)*(0.5*M(2)+M(3))
    M(1)*(4*L(1)^2+H(1)^2)/12+M(2)*(4*L(2)^2+H(2)^2)/12+M(3)*(4*L(3)^2+H(3)^2)/12+L(2)*M(3)*(L(2)
        +L(3)*cos(wrist_offset))+(L(1)^2)*(M(2)+M(3))
    M(2)*(4*L(2)^2+H(2)^2)/12+M(3)*(4*L(3)^2+H(3)^2)/12+L(2)*M(3)*(L(2)+L(3)*cos(wrist_offset))
    -L(3)*cos(wrist_offset)-L(2)
    -L(3)*sin(wrist_offset)
    -L(1) ];
KH = [ 4*beta*kv./Vt 4*beta*A2./Vt la.*la+lb.*lb la.*lb 4*beta*kleak./Vt ];
%
% initial conditions - for equilibrium at ypos ~= 0.87m, xpos ~= -1.05m
%
theta_ini = [ 1.7157 ; 0.8901 ];
theta_dot_ini = [ 0.0; 0.0 ];
ci1 = cos(theta_ini(1));
si1 = sin(theta_ini(1));
ci12 = cos(theta_ini(1)+theta_ini(2));
si12 = sin(theta_ini(1)+theta_ini(2));
xpos_ini = -K(10)*ci1-K(8)*ci12+K(9)*si12;
```

```
ypos_ini = -K(10)*si1-K(8)*si12-K(9)*ci12;
fxee_ini = 0.0;
% torque for equilibrium
tmp = K(1)*ci12+K(2)*si12+fxee_ini*(K(8)*si12+K(9)*ci12);
t_ini = [ tmp+K(3)*ci1+K(10)*fxee_ini*si1
          tmp ];
pm_ini= t_ini.*sqrt(KH(:,3)-2*KH(:,4).*cos(theta_ini-theta_poffset))
./(A2.*KH(:,4).*sin(theta_ini-theta_poffset));
x_ini = kleak.*pm_ini./(kv.*sqrt(ps-sign(pm_ini).*pm_ini));
v_ini = x_ini./ki;
%
% position and force step parameters
%
sim_duration = 5;
yd_time = [ 0 2.0 2.0 sim_duration ];
yd = [ ypos_ini ypos_ini ypos_ini-0.05 ypos_ini-0.05 ];
fxd = [ fxee_ini fxee_ini fxee_ini-100 fxee_ini-100 ];
fxd_time = [ 0 0.1 0.1 sim_duration ];
%
% environment parameters
%
KE = 1e4;
ground = xpos_ini - fxee_ini/KE;
%
% analogue to digital and digital to analogue convertor parameters
%
theta_quant = [ 6.39e-4;
                5.11e-4 ];          % 12 bit (slew range=150deg, elbow=120deg)
force_quant = 0.326;                % 12 bit
dac_quant = 4.0e-3;                 % 12 bit
theta_wc = 2*pi*19.4;               % 19.4 hz anti-aliasing filters
%
% controller parameters
%
samp_per = 0.01
control_limit = [ 8.188; 8.188 ];
control_int_ini = [ 0 ; 0; v_ini ];
%
% PID Controllers
%
KP = [ 0.0020 ; 0.0020 ];
KI = [ 0.0004 ; 0.0004 ]*samp_per;
KPP = [ 20.0 ; 20.0 ];
KPI = [ 1.5 ; 1.5 ]*samp_per;
%
% Fixed Gain Pole Placement Controller
%
TFIX1 = [ -1.92 0; 0 -1.9 ];
TFIX2 = [ 0.9216 0; 0 0.9025 ];
load ttt
a1 = -TTT(1:2,:)';
b1 = TTT(3:4,:)';
FIXG0 = b1\(TFIX1+eye(2)-a1);
FIXG1 = b1\(TFIX2+a1);
%
% multivariable self-tuning controller parameters
%
na = 3; nb = 2; nd = 1;
nt = na*joints+joints*nb+nd;
lambda = 0.99;
PSI = [ zeros(joints*(na+nb),1); ones(nd,1) ];
P = 100*eye(nt);
U = eye(nt);
D = 100*ones(nt,1);
%THETA = [ 1.0 0.0; 0.0 1.0; zeros(joints*(na-1),2); samp_per samp_per; samp_per samp_per;
          zeros(joints*(nb-1)+nd,2) ];
THETA = [  0.7265   -0.0335
           0.0004    2.5233
          -0.2010   -0.0001
          -0.0008   -2.3199
           0.4746    0.0656
```

```
              0.0003    0.7961
             -0.0012   -0.2737
              0.0006    0.6865
             -0.0004   -2.5779
              0.0019    0.8732
              0.0000   -0.0526 ];
T1 = [ -1.92 0; 0 -1.9 ];
T2 = [ 0.9216 0; 0 0.9025 ];
T3 = [ 0 0; 0 0 ];
T4 = [ 0 0; 0 0 ];
T5 = [ 0 0; 0 0 ];
%
% simulation parameters
%
no_pts = sim_duration/samp_per+1;
tolerance = 1e-6;
minstep = 1e-5;
maxstep = 0.1;
```

lnk2_05.m

```
function [sys, x0, str, ts] = lnk2_05h(t,x,u,flag)
% s-function for the two-link manipulator dynamic model with environment
% modelled internally as a stiffness KE - with offset in last link
%
% Arm modelled having a homogeneous mass distribution, with viscous and
% coulomb friction and actuated by a linear hydraulic actuator acting about
% a pivot. This includes the full non-linear servovalve model, oil
% compressibility, leakage and friction associated with the hydraulic ram.
%
% u = [i_in1; i_in2]
%   = [ servovalve input currents (ma)]
% x = [theta1_dot;theta2_dot;theta1;theta2;pm1;pm2;spool_pos1;spool_pos2]
%   = [ joint velocities ; joint angles ; load pressures ; spool positions]
% y = [theta1_dot; theta2_dot; theta1; theta2]
% theta_ddot = I1 \ [ [tau1 ; tau2] - V - G - I2 - fric]
%
% [tau1; tau2] = joint torques = (a2*pm - piston_fric)*jp(theta)
% V = Coriolis Matrix
% G = manipulator Gravity Matrix
% fric = joint friction matrix
% I1 = inertial matrix
% I2 = interaction matrix
%
% andy clegg       heriot-watt university                                  10/8/95
%
global K KH F Fram theta_ini theta_dot_ini pm_ini ps joints A2;
global theta_poffset ki taui x_ini KE ground;
if abs(flag) == 1
        % return state derivatives
        c1 = cos(x(3));
        s1 = sin(x(3));
        c2 = cos(x(4));
        s2 = sin(x(4));
        c12 = cos(x(3)+x(4));
        s12 = sin(x(3)+x(4));
        tmp = K(1)*c12+K(2)*s12;
        G = [ tmp+K(3)*c1 ; tmp ];
        tmp = K(4)*c2-K(5)*s2;
        V = [ tmp*x(2)*(x(2)+2*x(1)) ; -tmp*x(1)*x(1) ];
        tmp = K(5)*c2+K(4)*s2;
        I1 = [ K(6)+2*tmp K(7)+tmp ; K(7)+tmp K(7) ];
        fxee = (-K(10)*c1-K(8)*c12+K(9)*s12-ground)*KE;
        if (fxee<0)
                tmp = fxee*(K(8)*s12+K(9)*c12);
                I2 = [ tmp+fxee*K(10)*s1 ; tmp ];
        else
                I2 = [ 0.0 ; 0.0 ];
        end
```

```
        jp = KH(:,4).*sin(x(3:4)-theta_poffset)./sqrt(KH(:,3)-2*KH(:,4).*cos(x(3:4)-theta_poffset));
        y_dot = jp.*x(1:2);
        tau = (A2.*x(5:6) - Fram(:,1).*y_dot).*jp;
        theta_ddot = I1\(tau - V - I2 - F(:,1).*x(1:joints));
% to ensure that sqrt doesn't return an imaginary number
        pms = ps-sign(x(7:8)).*x(5:6);
        if (pms(1) <= 0)
                pms(1) = 0.0;
        end
        if (pms(2) <= 0)
                pms(2) = 0.0;
        end
        pm_dot = KH(:,1).*x(7:8).*sqrt(pms) - KH(:,2).*y_dot - KH(:,5).*x(5:6);
        x_dot = (ki.*u(1:2) - x(7:8))./taui;
        sys = [ theta_ddot ; x(1:2) ; pm_dot ; x_dot ];
elseif flag == 3
        % return system outputs
        fxee = (-K(10)*cos(x(3))-K(8)*cos(x(3)+x(4))+K(9)*sin(x(3)+x(4))-ground)*KE;
        if (fxee<0)
                sys = [ x(1:4) ; fxee ];
        else
                sys = [ x(1:4) ; 0.0 ];
        end
elseif flag == 0
        % return sizes and initial conditions
        sys = [ 4*joints; 0; 2*joints+1; joints; 0; 0; 1 ];
        x0 = [ theta_dot_ini ; theta_ini ; pm_ini; x_ini ];
        ts = [ 0.0; 0.0 ];
else
        % return nothing else - continuous system
        sys = [];
end
```

CONTS.M

```
function [sys, x0, str, ts] = conts(t,x,u,flag)
% this s-function performs PI hybrid position/force control
%
% u = [yd(k); y(k); fxd(k); fx(k); theta1(k); theta2(k)]
global samp_per v_ini KP KI KPP KPI K joints nt;
if flag == 2
        % PI control
        ijacden = K(10)*(K(8)*sin(u(6))+K(9)*cos(u(6)));
        jac12 = K(8)*sin(u(5)+u(6))+K(9)*cos(u(5)+u(6));
        jac11 = K(10)*sin(u(5))+jac12;
        ijac12 = -jac12/ijacden;
        ijac22 = jac11/ijacden;

        erry1 = ijac12*(u(1)-u(2));
        erry2 = ijac22*(u(2)-u(1));
        errfx1 = jac11*(u(3)-u(4));
        errfx2 = jac12*(u(4)-u(3));

        deltauy1 = KPI(1)*erry1 + KPP(1)*(erry1-x(1));
        deltaufx1 = KI(1)*errfx1 + KP(1)*(errfx1-x(3));
        controlslew = [ x(5)+deltauy1+deltaufx1 ];
        deltauy2 = KPI(2)*erry2 + KPP(2)*(erry2-x(2));
        deltaufx2 = KI(2)*errfx2 + KP(2)*(errfx2-x(4));
        controlelbow = [ x(6)+deltauy2+deltaufx2 ];

        sys = [ erry1; erry2; errfx1; errfx2; controlslew; controlelbow ];

elseif flag == 3
        % return the next output is.
        % PI control
        ijacden = K(10)*(K(8)*sin(u(6))+K(9)*cos(u(6)));
        jac12 = K(8)*sin(u(5)+u(6))+K(9)*cos(u(5)+u(6));
        jac11 = K(10)*sin(u(5))+jac12;
        ijac12 = -jac12/ijacden;
```

```
        ijac22 = jac11/ijacden;

        erry1 = ijac12*(u(1)-u(2));
        erry2 = ijac22*(u(2)-u(1));
        errfx1 = jac11*(u(3)-u(4));
        errfx2 = jac12*(u(4)-u(3));

        deltauy1 = KPI(1)*erry1 + KPP(1)*(erry1-x(1));
        deltaufx1 = KI(1)*errfx1 + KP(1)*(errfx1-x(3));
        controlslew = [ x(5)+deltauy1+deltaufx1 ];
        deltauy2 = KPI(2)*erry2 + KPP(2)*(erry2-x(2));
        deltaufx2 = KI(2)*errfx2 + KP(2)*(errfx2-x(4));
        controlelbow = [ x(6)+deltauy2+deltaufx2 ];

        sys = [ zeros(nt,1); zeros(nt,1); 0; 0; controlslew; controlelbow ];

elseif flag == 0
        % returns the sizes vector and initial conditions.
        sys = [ 0; 6; (nt+2)*joints; 6; 0; 1; 1 ];
        x0 = [ 0; 0; 0; 0; v_ini ];
        ts = [ samp_per 0.00 ];
else
        % Flags not considered here are treated as unimportant.
        sys = [];
end
```

RLSCONT2.M

```
function [sys, x0, str, ts] = rlscont2(t,x,u,flag)
% this s-function performs the rls identification and self-tuning control
% the global variables L, PSI, P, THETA
% u = [yd(k); y(k); fxd(k); fx(k); theta1(k); theta2(k)]
% lambda = forgetting factor
% na = no. of A co-efficients
% nb = no. of B co-efficients
% nd = no. of disturbance co-efficients
% nt = na+nb+nd
global PSI P U D
global THETA T1 T2 T3 T4 T5 FIXG0 FIXG1;
global na nb nd nt samp_per lambda v_ini ypos_ini fxee_ini K joints;
if flag == 2
        % perform check on sample hit due to bug in Simulink - see release notes
        if abs(round(t/samp_per)-(t/samp_per)) < 1e5*eps
                if (t>0.3)
                        % multivariable self-tuning control
                        a1 = -THETA(1:2,:)';
                        a2 = -THETA(3:4,:)';
                        a3 = -THETA(5:6,:)';
                        b1 = THETA(7:8,:)';
                        b2 = THETA(9:10,:)';
                        x1 = b1/b2;
                        x2 = x1*x1;
                        x3 = x2*x1;
                        x4 = x3*x1;
                        xf = eye(2)+x1*(eye(2)-a1)-x2*(a1-a2)+x3*(a2-a3)-x4*a3;
                        xx = T1+eye(2)-a1-x1*(T2-a2+a1)+x2*(T3-a3+a2)-x3*(T4+a3)+ x4*T5;
                        F1 = xf\xx;
                        G0 = b1\(T1-F1+eye(2)-a1);
                        G1 = b1\(T2-a2-a1*F1+a1+F1-b2*G0);
                        G2 = b1\(T3-a3-a2*F1+a2+a1*F1-b2*G1);
                        G3 = b1\(T4-a3*F1+a3+a2*F1-b2*G2);
                        % transorm like in the Prager paper.
                        GG0 = G0;
                        xx = G3*F1-G2*F1*F1+G1*F1*F1*F1-G0*F1*F1*F1*F1;
                        xf = G3-G2*F1+G1*F1*F1-G0*F1*F1*F1;
                        FF1 = xx/xf;
                        GG1 = G1+FF1*G0-GG0*F1;
                        GG2 = G2+FF1*G1-GG1*F1;
                        GG3 = G3+FF1*G2-GG2*F1;
```

```
                deltau = (GG0+GG1+GG2+GG3)*u(1:2:3)-(GG0*u(2:2:4)
                        +GG1*x(1:2)+GG2*x(3:4)+GG3*x(5:6))-FF1*x(7:8);
                control = [ x(9:10)+deltau ];
        else
                % fixed gain na=1,nb=1 (PI) controller
                deltau = (FIXG0+FIXG1)*u(1:2:3)-(FIXG0*u(2:2:4)+ FIXG1*x(1:2));
                control = [ x(9:10)+deltau ];
        end
        sys = [u(2:2:4); x(1:4); deltau; control ];
else
        sys = [];
end

elseif flag == 3
% return the next output is.
if (t >=0.08)
        % system identification : Bierman UD Factorisation
        F = U'*PSI;
        G = D.*F;
        budbeta = lambda+cumsum(F.*G);
        oldbudbeta = [lambda; budbeta(1:nt-1)];
        tmp = cumsum(diag(G)*U');
        L = tmp(nt,:)'/budbeta(nt);
        D = D.*oldbudbeta./(budbeta*lambda);
        U = U - (diag(F./oldbudbeta)*[zeros(1,nt); tmp(1:nt-1,:)])';
        % system identification : Matrix Inversion Lemma
        %L = P*PSI/(lambda+PSI'*P*PSI);
        %P = (eye(nt)-L*PSI')*P/lambda;
        THETA(:,1) = THETA(:,1)+L*(u(2)-PSI'*THETA(:,1));
        THETA(:,2) = THETA(:,2)+L*(u(4)-PSI'*THETA(:,2));
        model_op = [ THETA(:,1)'*PSI ; THETA(:,2)'*PSI ];
else
        model_op = [ 0; 0 ];
end

if (t>0.3)
        % multivariable self-tuning control
        a1 = -THETA(1:2,:)';
        a2 = -THETA(3:4,:)';
        a3 = -THETA(5:6,:)';
        b1 = THETA(7:8,:)';
        b2 = THETA(9:10,:)';
        x1 = b1/b2;
        x2 = x1*x1;
        x3 = x2*x1;
        x4 = x3*x1;
        xf = eye(2)+x1*(eye(2)-a1)-x2*(a1-a2)+x3*(a2-a3)-x4*a3;
        xx = T1+eye(2)-a1-x1*(T2-a2+a1)+x2*(T3-a3+a2)-x3*(T4+a3)+x4*T5;
        F1 = xf\xx;
        G0 = b1\(T1-F1+eye(2)-a1);
        G1 = b1\(T2-a2-a1*F1+a1+F1-b2*G0);
        G2 = b1\(T3-a3-a2*F1+a2+a1*F1-b2*G1);
        G3 = b1\(T4-a3*F1+a3+a2*F1-b2*G2);
        % transorm like in the Prager paper.
        GG0 = G0;
        xx = G3*F1-G2*F1*F1+G1*F1*F1*F1-G0*F1*F1*F1*F1;
        xf = G3-G2*F1+G1*F1*F1-G0*F1*F1*F1;
        FF1 = xx/xf;
        GG1 = G1+FF1*G0-GG0*F1;
        GG2 = G2+FF1*G1-GG1*F1;
        GG3 = G3+FF1*G2-GG2*F1;
        deltau = (GG0+GG1+GG2+GG3)*u(1:2:3)-(GG0*u(2:2:4)+GG1*x(1:2)
                        +GG2*x(3:4)+GG3*x(5:6))-FF1*x(7:8);
        control = [ x(9:10)+deltau ];
else
        % fixed gain na=1,nb=1 (PI) controller
        deltau = (FIXG0+FIXG1)*u(1:2:3) - (FIXG0*u(2:2:4)+FIXG1*x(1:2));
        control = [ x(9:10)+deltau ];
end

PSI(1:nt-nd) = [ u(2); u(4); PSI(1:(na-1)*joints); control; PSI(na*joints+1:joints*(na+nb-1)) ];
```

```
        sys = [ THETA(:,1); THETA(:,2); model_op; control ];

elseif flag == 0
        % returns the sizes vector and initial conditions.
        sys = [ 0; 10; (nt+2)*joints; 6; 0; 1; 1 ];
        x0 = [ ypos_ini; fxee_ini; ypos_ini; fxee_ini; ypos_ini;f xee_ini; 0; 0; v_ini ];
        ts = [ samp_per 0.00 ];
else
        % Flags not considered here are treated as unimportant.
        sys = [];
end
```

SIMULATE.M

```
clc; clear all;
% load simulation models and data
sim2data;
% start simulation
tic;
rk45(model,sim_duration,[],[tolerance,minstep,maxstep,1,0,2]);
toc;

delete(gcf);
if (exist('force') == 1) & (exist('ypos') == 1)
        subplot(211),plot(ypos)
        axis([0 sim_duration/samp_per min(min(ypos))-0.01 max(max(ypos))+0.01])
        subplot(212),plot(force)
        axis([0 sim_duration/samp_per min(min(force))-20 0])
end
save results ypos force volts theta THETAS modelop lambda
```

Appendix B

Bierman U/D Factorisation Algorithm

B.1 Introduction

A widely used numerically robust version of the RLS method is the *Bierman U-D factorisation* (BUD-RLS) algorithm [4.3]. This uses the symmetric property of the covariance matrix to allow the factorisation of $P(k)$ as :-

$$P(k) = U(k)D(k)U^{T}(k) \tag{B.1}$$

where $U(k)$ is an upper triangular matrix and $D(k)$ is a diagonal matrix. The BUD-RLS algorithm then computes $U(k)$ and $D(k)$ from $U(k\text{-}1)$ and $D(k\text{-}1)$ respectively. This is equivalent to calculating $P(k)$ at twice the precision of the standard RLS algorithm. This method also ensures that the covariance matrix remains positive definite, which is a requirement for parameter convergence.

B.2 BUD-RLS Algorithm

This section presents the algorithm for the BUD-RLS, and uses the same parameters and nomenclature as introduced in Section 4.2.2 for the MIL-RLS algorithm. The proof of the BUD-RLS algorithm is given in [4.3].

At sample instant k,

Step 1: Read system output, $y(k)$, and form $\psi(k)$ using past input and output values.

Step 2: Calculate the *a priori prediction error*, $\epsilon(k)$, as for the MIL-RLS :-

$$\epsilon(k) = y(k) - \theta^{T}(k-1)\,\psi(k) \tag{B.2}$$

Step 3: Form the vectors *f* and *g* :-

$$\begin{aligned} f &= U^{T}(k-1)\,\psi(k) \\ g &= D(k-1)\,f \end{aligned} \tag{B.3}$$

and set $\lambda_0 = \lambda$, the *forgetting factor*.

Repeat steps 4 and 5 with $j = 1, 2, ..., n_{\psi}$, where n_{ψ} is the number of rows in the regression vector, $\psi(k)$.

Step 4: Calculate :-

$$\begin{aligned} \lambda_j &= \lambda_{j-1} + f_j\, g_j \\ D_j(k) &= (\lambda_j\, \lambda)^{-1}\, \lambda_{j-1} D_j(k-1) \\ \nu_j &= g_j \end{aligned} \tag{B.4}$$

where λ_j is a scalar, and f_j, g_j, D_j and ν_j are elements of the relevant vector.

Step 5: When $j > 1$, calculate the following with $i = 1, 2, ..., j-1$:-

$$\begin{aligned} U_{ij}(k) &= U_{ij}(k-1) - \nu_i\, f_j / \lambda_{j-1} \\ \nu_i &= \nu_i + U_{ij}(k-1)\, \nu_j \end{aligned} \tag{B.5}$$

where U_{ij} is the appropriate element of the *U* matrix.

Step 6: Calculate the new *Kalman gain vector*, $L(k)$:-

$$L(k) = \nu / \lambda_{n_{\psi}} \tag{B.6}$$

Step 7: Update the parameter estimates, $\Phi(k)$:-

$$\Phi(k) = \Phi(k-1) + L(k)\,\epsilon(k) \tag{B.7}$$

Step 8: Wait for next sample, $k \leftarrow k+1$, and then loop back to Step 1.

Implementation of the above BUD-RLS algorithm generally requires fewer computations than an equally efficient coded MIL-RLS algorithm [4.3]. A MIMO version of this algorithm is realised by modifying Step 7, see Section 4.4.2 :-

Step 7: Update the parameter estimates, $\Phi(k)$ with $i = 1, 2, ..., n$:-

$$\Phi_i(k) = \Phi_i(k-1) + L(k)\mathrm{E}_i(k) \tag{B.8}$$

where $\Phi_i(k)$ is the ith *column* of $\Phi(k)$, and $\mathrm{E}_i(k)$ of the ith element of the a priori prediction error vector.

A version of the BUD-RLS algorithm is given below for the MATLAB programming language. The highly vectorised nature of the language results in compact and efficient code that is capable of both SISO and MIMO system identification.

```
% this code performs steps 2 to 7 of BUD-RLS identification algorithm
% uses the global variables PSI, PHI, U, D
% PSI = regression vector, PHI = parameter estimates
% y = y(k) - system output
% n = no. of inputs/outputs
% lambda = forgetting factor
% na = no. of A coefficients, nb = no. of B coefficients
% nd = no. of disturbance coefficients
% npsi = no. of rows of PSI
%
F = U'*PSI;
G = D.*F;
budbeta = lambda+cumsum(F.*G);
oldbudbeta = [lambda; budbeta(1:npsi-1)];
D = D.*oldbudbeta./(budbeta*lambda);
tmp = cumsum(diag(G)*U');
U = U - (diag(F./oldbudbeta)*[zeros(1,npsi); tmp(1:npsi-1,:)])';
L = tmp(npsi,:)'/budbeta(npsi);
for i = 1:n
      PHI(:,i) = PHI(:,i)+L*(y(i)-PSI'*PHI(:,i));
end
```

MATLAB code for Steps 2 to 7 of the BUD-RLS algorithm

Appendix C

Explicit Solutions for Self-tuning Pole Placement Controllers

C.1 Introduction

This appendix presents the solution to the design equations for both SISO and MIMO self-tuning pole placement incremental controllers, as described in Chapter Four. The controller polynomials are found from the solution of Equations 4.16 and 4.35, for SISO and MIMO cases respectively.

The controller designs presented here are specific to the model orders used throughout this thesis. A process model of $n_a = 2$ and $n_b = 1$, results in an incremental controller with the same structure as a PID controller, and the gains of this controller are derived in terms of the estimated model parameters. The higher order model corresponding to the underlying physical system, $n_a = 3$, $n_b = 2$ and $n_d = 1$, yields a different set of design equations that are also derived. It should be recalled that the inclusion of the offset parameter, d_0, does not alter the controller design.

C.2 Self-tuning PID Controller Design

The conventional expression for a continuous time PID controller acting on the error, e, is :-

$$u = k_p e + k_i \int e\, dt + k_d \dot{e} \tag{C.1}$$

with proportional gain k_p, integral gain k_i and derivative gain k_d. This can be cast into the

following discrete time representation :-

$$u(k) = k_p e(k) + k_i \tau_s \sum_{j=0}^{k} e(j) + \frac{k_d}{\tau_s}[e(k) - e(k-1)] \qquad \text{(C.2)}$$

where the sampling period, τ_s, is sufficiently short for this approximation to hold. This may then be expressed in an *incremental* form (cf. Figure 4.2) as :-

$$u(k) = u(k-1) + k_p[e(k) - e(k-1)] + k_i \tau_s e(k) + \frac{k_d}{\tau_s}[e(k) - 2e(k-1) + e(k-2)] \qquad \text{(C.3)}$$

This conforms to the controller introduced in Section 4.2.3, expressed in terms of polynomials $f(z^{-1})$ and $g(z^{-1})$, with orders of $n_f = 0$ and $n_g = 2$:-

$$u(k) = u(k-1) + g_0 e(k) + g_1 e(k-1) + g_2 e(k-2) \qquad \text{(C.4)}$$

From Equation 4.18, this form of PID controller corresponds to a model order of $n_a = 2$ and $n_b = 1$. These controller parameters are obtained in terms of the model parameters by solving Equation 4.16, which for this particular model becomes :-

$$(1 + t_1 z^{-1} + t_2 z^{-2} + t_3 z^{-3}) = (1 + a_1 z^{-1} + a_2 z^{-2})(1 - z^{-1}) + b_1 z^{-1}(g_0 + g_1 z^{-1} + g_2 z^{-2}) \qquad \text{(C.5)}$$

Equating like powers of z :-

$$\begin{aligned} z^{-1}&: \quad t_1 = a_1 - 1 + b_1 g_0 \\ z^{-2}&: \quad t_2 = a_2 - a_1 + b_1 g_1 \\ z^{-3}&: \quad t_3 = -a_2 + b_1 g_2 \end{aligned} \qquad \text{(C.6)}$$

Which gives :-

$$g_0 = \frac{t_1 - a_1 + 1}{b_1}$$
$$g_1 = \frac{t_2 - a_2 + a_1}{b_1} \qquad \text{(C.7)}$$
$$g_2 = \frac{t_3 + a_2}{b_1}$$

This set of equations defines the SISO self-tuning PID incremental controller. The correlation between these coefficients and the original PID controller gains is found by comparing Equations C.3 and C.4 :-

$$k_p = -g_1 - 2g_2$$
$$k_i \tau_s = g_0 + g_1 + g_2 \qquad \text{(C.8)}$$
$$\frac{k_d}{\tau_s} = g_2$$

A self-tuning PI controller can be formulated in an identical manner. Since no derivative gain is present, $e(k\text{-}2)$ does not appear in the controller (cf. Equation C.3). Therefore, $n_f = 0$ and $n_g = 1$, which, from Equation 4.18, corresponds to a process model of $n_a = 1$ and $n_b = 1$.

C.3 Self-tuning Controller Design for $n_a = 3$, $n_b = 2$

The model that corresponds to the underlying physical system, given by Equation 4.24, has polynomial orders $n_a = 3$, $n_b = 2$ and $n_d = 1$. For a unique solution of Equation 4.16 to exist, the required controller polynomial orders are $n_f = 1$, $n_g = 3$ and $n_t \leq 5$, from Equation 4.18.

Equation 4.16 is solved by equating like powers of z, and this results in five simultaneous linear equations in five unknowns, which can be conveniently represented by the following matrix expression :-

$$\begin{bmatrix} 1 & b_1 & 0 & 0 & 0 \\ a_1-1 & b_2 & b_1 & 0 & 0 \\ a_2-a_1 & 0 & b_2 & b_1 & 0 \\ a_3-a_2 & 0 & 0 & b_2 & b_1 \\ -a_3 & 0 & 0 & 0 & b_2 \end{bmatrix} \begin{bmatrix} f_1 \\ g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} t_1+1-a_1 \\ t_2+a_1-a_2 \\ t_3+a_2-a_3 \\ t_4+a_3 \\ t_5 \end{bmatrix} \qquad \text{(C.9)}$$

The special banded structure of the left hand matrix in Equation C.9 is typical of *Sylvester matrices*. The solution to Equation C.9 can be obtained by pre-multiplying both sides by the inverse of this 5×5 matrix, yielding expressions for $f(z^{-1})$ and $g(z^{-1})$ in terms of $a(z^{-1})$, $b(z^{-1})$ and $t(z^{-1})$. This matrix inversion can become ill-conditioned under certain situations, and robust methods for solving the polynomial identities exist [2.57].

Here, the solution to the set of five simultaneous equations is derived explicitly, since it is simpler than determining the matrix inversion due to the sparse nature of the matrix. The resulting equations that define the controller are :-

$$\begin{aligned} f_1 &= \frac{x_x}{x_f} \\ g_0 &= \frac{t_1+1-a_1-f_1}{b_1} \\ g_1 &= \frac{t_2+a_1-a_2-(a_1-1)f_1-b_2g_0}{b_1} \\ g_2 &= \frac{t_3+a_2-a_3-(a_2-a_1)f_1-b_2g_1}{b_1} \\ g_3 &= \frac{t_4+a_3-(a_3-a_2)f_1-b_2g_2}{b_1} \end{aligned} \qquad \text{(C.10)}$$

where :-

$$\begin{aligned} x &= \frac{b_1}{b_2} \\ x_f &= 1-x(a_1-1)+x^2(a_2-a_1)-x^3(a_3-a_2)+x^4(-a_3) \\ x_x &= t_1+1-a_1-x(t_2+a_1-a_2)+x^2(t_3+a_2-a_3)-x^3(t_4+a_3)+x^4t_5 \end{aligned} \qquad \text{(C.11)}$$

C.4 MIMO Self-tuning PID Controller Design

The structure of a MIMO self-tuning pole placement controller is identical to that of a SISO controller, except that the coefficients of the controller polynomials are now $n \times n$ matrices, rather than scalars. However, due to the non-commutivity of matrices, the order in which calculations are performed in the derivation is now important.

The equation that defines the controller, Equation 4.35, is identical to that for a SISO controller. Further, the structure of the SISO PID incremental controller given by Equation C.3 is also applicable to a MIMO system; the controller outputs and errors simply become $n \times 1$ vectors and the gains become $n \times n$ matrices. Therefore, it follows that the order of the model and controller polynomials match those of the SISO PID, $n_f = 0$, $n_g = 2$, $n_a = 2$, $n_b = 1$. Equation 4.35 becomes (cf. Equation C.5) :-

$$\begin{aligned}(\mathrm{I} + T_1 z^{-1} + T_2 z^{-2} + T_3 z^{-3}) = (\mathrm{I} + A_1 z^{-1} + A_2 z^{-2})(\mathrm{I} - \mathrm{I} z^{-1}) \\ + B_1 z^{-1}(G_0 + G_1 z^{-1} + G_2 z^{-2})\end{aligned} \tag{C.12}$$

The solution to this gives expressions for the controller polynomials in terms of the model parameters. Equating like powers of z gives :-

$$\begin{aligned} z^{-1}: &\qquad T_1 = A_1 - \mathrm{I} + B_1 G_0 \\ z^{-2}: &\qquad T_2 = A_2 - A_1 + B_1 G_1 \\ z^{-3}: &\qquad T_3 = -A_2 + B_1 G_2 \end{aligned} \tag{C.13}$$

which is the MIMO equivalent of Equation C.6, this is then rearranged to give :-

$$\begin{aligned} G_0 &= B_1^{-1}(T_1 - A_1 + \mathrm{I}) \\ G_1 &= B_1^{-1}(T_2 - A_2 + A_1) \\ G_2 &= B_1^{-1}(T_3 + A_2) \end{aligned} \tag{C.14}$$

This set of equations define the MIMO self-tuning PID incremental controller, and are identical to the SISO case, Equation C.7, except that the order of matrix calculations

must be preserved. The off-diagonal elements of these $n \times n$ matrices control the interactions between different process inputs and outputs. The correlation between these coefficients and the PID controller gains can be similarly obtained (cf. Equation C.8).

A MIMO self-tuning PI controller can also be formulated using the same procedure, with $n_f = 0$, $n_g = 1$, $n_a = 1$ and $n_b = 1$.

C.5 MIMO Self-tuning Controller Design for $n_a = 3$, $n_b = 2$

The controller design for the MIMO model that corresponds to the underlying physical system, $n_a = 3$, $n_b = 2$ and $n_d = 1$, follows the same procedure as the SISO case. For a unique solution of Equation 4.35 to exist, the required controller polynomial orders are $n_f = 1$, $n_g = 3$ and $n_t \leq 5$, from Equation 4.18.

Equating like powers of z in Equation 4.35, results in five simultaneous linear equations in five unknowns represented by the following matrix expression[†] :-

$$\begin{bmatrix} \mathrm{I} & B_1 & 0 & 0 & 0 \\ A_1 - \mathrm{I} & B_2 & B_1 & 0 & 0 \\ A_2 - A_1 & 0 & B_2 & B_1 & 0 \\ A_3 - A_2 & 0 & 0 & B_2 & B_1 \\ -A_3 & 0 & 0 & 0 & B_2 \end{bmatrix} \begin{bmatrix} F_1 \\ G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix} = \begin{bmatrix} T_1 + \mathrm{I} - A_1 \\ T_2 + A_1 - A_2 \\ T_3 + A_2 - A_3 \\ T_4 + A_3 \\ T_5 \end{bmatrix} \qquad \text{(C.15)}$$

This matches the equivalent SISO expression (cf. Equation C.9), with the elements now being $n \times n$ matrices rather than scalars. Again, expressions for $F(z^{-1})$ and $G(z^{-1})$ in terms of $A(z^{-1})$, $B(z^{-1})$ and $T(z^{-1})$ can be found be inverting the matrix on the left of Equation C.15. However, the explicit solution to the simultaneous equations is used as it is much less computationally intensive than performing the inversion of a $5n \times 5n$ matrix. The resulting equations that define the controller match those of the SISO controller (cf. Equations C.10

[†] Equation C.15 conforms to the generalised solution of the MIMO self-tuning pole placement controller design equations presented by Prager [4.4]. To correspond to the controller proposed by Prager, the incremental controller presented here must have the $(\mathrm{I}\text{-}\mathrm{I}z^{-1})$ term embodied in the $F(z^{-1})$ controller polynomial.

and C.11) :-

$$
\begin{aligned}
F_1 &= X_f^{-1} X_x \\
G_0 &= B_1^{-1}(T_1 + \mathrm{I} - A_1 - F_1) \\
G_1 &= B_1^{-1}(T_2 + A_1 - A_2 - (A_1 - \mathrm{I})F_1 - B_2 G_0) \\
G_2 &= B_1^{-1}(T_3 + A_2 - A_3 - (A_2 - A_1)F_1 - B_2 G_1) \\
G_3 &= B_1^{-1}(T_4 + A_3 - (A_3 - A_2)F_1 - B_2 G_2)
\end{aligned}
\qquad \text{(C.16)}
$$

where :-

$$
\begin{aligned}
X &= B_1 B_2^{-1} \\
X_f &= \mathrm{I} - X(A_1 - \mathrm{I}) + X^2(A_2 - A_1) - X^3(A_3 - A_2) + X^4(-A_3) \\
X_x &= T_1 + \mathrm{I} - A_1 - X(T_2 + A_1 - A_2) + X^2(T_3 + A_2 - A_3) - X^3(T_4 + A_3) + X^4 T_5
\end{aligned}
\qquad \text{(C.17)}
$$

This solution is applicable to systems with an arbitrary number of inputs and outputs, however, the number of controller calculations required will grow exponentially as *n* increases.

C.6 Pseudo-Commutivity Transformation

A MIMO self-tuning controller is designed in terms of $F(z^{-1})$ and $G(z^{-1})$ using the procedures described above. However, the resulting controller is not directly implementable due to matrix non-commutivity, as discussed in Section 4.4.3. The following *pseudo-commutivity* relation is used to transform the controller polynomials to yield a realisable MIMO self-tuning controller [4.4], given by Equation 4.38 :-

$$
[\mathrm{I} + \tilde{F}(z^{-1})]\, G(z^{-1}) = \tilde{G}(z^{-1})\, [\mathrm{I} + F(z^{-1})] \qquad \text{(C.18)}
$$

The MIMO incremental PID controller derived in Section C.4, does not have a $F(z^{-1})$ term, and so this transformation is superfluous. The MIMO controller presented in Section

C.5 has polynomial orders of $n_f = 1$ and $n_g = 3$, and the specific transformation for this case will now be derived. Equation C.18 becomes :-

$$(\mathrm{I}+\tilde{F}_1 z^{-1})(G_0+G_1 z^{-1}+G_2 z^{-2}+G_3 z^{-3}) = (\tilde{G}_0+\tilde{G}_1 z^{-1}+\tilde{G}_2 z^{-2}+\tilde{G}_3 z^{-3})(\mathrm{I}+F_1 z^{-1}) \quad \text{(C.19)}$$

This is solved by equating like powers of z, which gives the following set of simultaneous equations :-

$$\begin{aligned}
z^0 &: \quad G_0 = \tilde{G}_0 \\
z^{-1} &: \quad G_1+\tilde{F}_1 G_0 = \tilde{G}_1+\tilde{G}_0 F_1 \\
z^{-2} &: \quad G_2+\tilde{F}_1 G_1 = \tilde{G}_2+\tilde{G}_1 F_1 \\
z^{-3} &: \quad G_3+\tilde{F}_1 G_2 = \tilde{G}_3+\tilde{G}_2 F_1 \\
z^{-4} &: \quad \tilde{F}_1 G_3 = \tilde{G}_3 F_1
\end{aligned} \quad \text{(C.20)}$$

The solution to these simultaneous equations gives the required pseudo-commutivity transformation for this particular controller, which is :-

$$\begin{aligned}
\tilde{F}_1 &= X F_1 X^{-1} \\
\tilde{G}_0 &= G_0 \\
\tilde{G}_1 &= G_1+\tilde{F}_1 G_0-\tilde{G}_0 F_1 \\
\tilde{G}_2 &= G_2+\tilde{F}_1 G_1-\tilde{G}_1 F_1 \\
\tilde{G}_3 &= G_3+\tilde{F}_1 G_2-\tilde{G}_2 F_1
\end{aligned} \quad \text{(C.21)}$$

where :-

$$X = [(G_0 F_1-G_1) F_1+G_2] F_1-G_3 \quad \text{(C.22)}$$

This result is applicable to controllers with or without the incremental term, $(\mathrm{I}-\mathrm{I}z^{-1})$, as it is cancelled when introduced into the above derivation.

List of References

Chapter 1

[1.1] A.K.Bejczy, "Virtual Reality in Robotics", IEEE Conf. on Emerging Technologies and Factory Automation, New York, USA, Vol.1, pp.7-15,1996.

[1.2] C.P.Sayers, D.R.Yoerger, R.P.Paul and J.S.Lisiewicz, "A Manipulator Work Package for Teleoperation from Unmanned Untethered Vehicles - Current Feasibility and Future Applications", 6th IARP Workshop on Underwater Robotics, Toulon, France, 27-29 March 1996.

[1.3] D.Broome, T.Larkum and M.Hall, "Subsea Weld Inspection using an Advanced Underwater Robotic Manipulator", Proc. of IEEE Oceans '95, San Diego, USA, 9-12 October, 1995.

[1.4] D.M.Lane, M.W.Dunnigan, A.W.Quinn and A.C.Clegg, "Motion Planning and Contact Control for a Tele-Assisted Hydraulic Underwater Robot", Journal of Autonomous Robots, Special Issue on Autonomous Underwater Robots, Vol.3, No.3, pp. 233-51, 1996.

[1.5] M.W.Dunnigan, D.M.Lane, A.C.Clegg and I.Edwards, "Hybrid Position/Force Control of a Hydraulic Underwater Manipulator", Proc. of the IEE: Part D, Vol. 143, No.2, pp. 145-51, 1996.

[1.6] P.K.Pook and D.H.Ballard, "Remote Teleassistance", IEEE Int. Conf. on Robotics and Automation, pp. 944-9, Nagoya, Japan, 1995.

[1.7] D.M.Lane, M.W.Dunnigan, P.J.Knightbridge, A.W.Quinn and A.C.Clegg, "Dual Manipulator Collaboration: Issues in Architecture, Planning and Control", 7th Int. Symp. on Unmanned, Untethered Submersible Technology, University of New Hampshire, USA, September 1991.

[1.8] D.M.Lane and P.J.Knightbridge, "Task Planning and World Modelling for Supervisory Control in Unstructured Environments", Proc. of Intelligent Robotic Systems Symp., Grenoble, France, July 1994.

[1.9] A.W.Quinn, "Motion Planning for Manipulators using Distributed Search", Ph.D Thesis, Heriot-Watt University, 1993.

[1.10] C.S.Reid, "Integration of Acoustic and Visual Data for Subsea Robotics", Ph.D Thesis, Heriot-Watt University, May 1992.

[1.11] M.P.Groover, M.Weiss, R.Nagel and N.Odrey, *Industrial Robotics : Technology, Programming and Applications*, McGraw-Hill, 1st ed., 1986.

[1.12] M.Perrier and C.Canudas de Wit, "Robust Nonlinear Control for Subsea Robots", 6th IARP Workshop on Underwater Robotics, Toulon, France, 27-29 March 1996.

[1.13] M.W.Dunnigan, "An Investigation of the Dynamic Coupling between a Manipulator and an Underwater Vehicle", Ph.D Thesis, Heriot-Watt University, 1994.

[1.14] A.C.Clegg, M.W.Dunnigan, D.M.Lane, "Force Control and Modelling of Hydraulic Underwater Manipulators", Int. Workshop on Advanced Robotics and Intelligent Machines, University of Salford, UK, pp. 1-8, 5-6 April 1995.

[1.15] M.W.Dunnigan, A.C.Clegg and D.M.Lane, "Self-tuning Control of a 7-Function Hydraulically Powered Underwater Manipulator", 2nd Int. Conf. on Automation, Robotics and Computer Vision, Singapore, Vol.3, pp. RO3.1.1-5, 16-18 September 1992.

Chapter 2

[2.1] J.J.Craig, *Introduction to Robotics : Mechanics and Control*, Addison-Wesley, 2nd ed., 1989.

[2.2] D.E.Whitney, "Historical Perspectives and State of the Art in Robot Force Control", Int. Journal of Robotics Research, Vol. 6, pp. 3-14, 1987.

[2.3] S.P.Patarinski and R.G.Botev, "Robot Force Control : A Review", Mechatronics, Vol.3, No.4, pp.377-98, 1993.

[2.4] J.K.Salisbury, "Active Stiffness Control of a Manipulator in Cartesian Coordinates", Proc. of 19th IEEE Conf. on Decision and Control, pp. 95-100, Albuquerque, USA, 1980.

[2.5] P. Rocco, G. Ferretti and G.Magnani, "Implicit Force Control for Industrial Robots in Contact with Stiff Surfaces", Automatica, Vol. 33, No. 11, pp. 2041-7, 1997.

[2.6] D.E.Whitney, "Force Feedback of Manipulator Fine Motions", Trans. of the ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 99, No. 2, pp. 91-7, 1977.

[2.7] K.Youcef-Toumi and D.Li, "Force Control of Direct-Drive Manipulators for Surface following", IEEE Int. Conf. on Robotics and Automation, pp.2055-60, 1987.

[2.8] N.Hogan, "Impedance Control: An Approach to Manipulation: Part 1 - Theory, Part 2 - Implementation, Part 3 - Applications", Trans. of the ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 107, pp. 1-24, June 1985.

[2.9] B.Heinrichs and N.Sepehri, "Relationship of Position-Based Impedance Control to Explicit Force Control: Theory and Experiments", Proc. of the American Control Conference, USA, pp. 2072-6, 1999.

[2.10] D.M.Stokić, M.K.Vukobratović and D.M.Šurdilović, "An Adaptive Control Scheme for Manipulation Robots with Implicit Force Control", 5th Int. Conf. on

Advanced Robotics, pp. 1505-8, Pisa, Italy, June 1991.

[2.11] R.P.Paul and B.E.Shimano, "Compliance and Control", Proc. of the Joint Automatic Control Conf., pp. 694-9, Purdue, USA, 1976.

[2.12] M.H.Raibert and J.J.Craig, "Hybrid Position/Force Control of Manipulators", Trans. of the ASME Journal of Dynamic Systems, Measurement and Control, Vol. 102, pp. 126-33, 1981.

[2.13] H.Zhang and R.P.Paul, "Hybrid Control of Robot Manipulators", IEEE Int. Conf. on Robotics and Automation, pp. 602-7, USA, 1985.

[2.14] S.Chiaverini and L.Sciavicco, "The Parallel Approach to Force/Position Control of Robotic Manipulators", IEEE Trans. on Robotics and Automation, Vol. 9, No. 4, pp. 361-73, August 1993.

[2.15] V.Perdereau and M.Drouin, "A New Scheme for Hybrid Force-Position Control", Robotica, Vol. 11, pp. 453-64, 1993.

[2.16] D.M.Stokić, "Constrained Motion Control of Manipulation Robots - a Contribution", Robotica, Vol. 9, pp.157-63, 1991.

[2.17] B.Bluethmann, S.Ananthakrishnan, J.Scheerer, T.N.Faddis and R.B.Greenway, "Experiments in Dextrous Hybrid Force and Position Control of a Master/Slave Electrohydraulic Manipulator", IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Pittsburgh, USA, Vol. 3, pp.27-32, August 1995.

[2.18] D.P.Stotten and S.P.Hodgson, "The Application of the Minimal Control Synthesis Algorithm to the Hybrid Control of a Class 1 Manipulator", Int. Journal of Control, Vol. 56, No. 3, pp. 499-513, 1992.

[2.19] R.Lozano and B.Brogliato, "Adaptive Hybrid Force-Position Control for Redundant Manipulators", IEEE Trans on Automatics Control, Vol. 37, No. 10, pp. 1501-5, October 1992.

[2.20] C.H.An and J.M.Hollerbach, "Kinematic Stability Issues in Force control of

Manipulators", IEEE Int. Conf. on Robotics and Automation, pp. 897-903, USA, 1987.

[2.21] H.Zhang, "Kinematic Stability of Robot Manipulators under Force Control", IEEE Int. Conf. on Robotics and Automation, pp.80-5, USA, 1989.

[2.22] W.D.Fisher and M.S.Mujtaba, "Hybrid Position/Force Control : A Correct Formulation", Int. Journal of Robotics Research, Vol. 11, No. 4, pp. 299-311, August 1992.

[2.23] M.Vukobratović and R.Stojić, "Historical Perspective of Hybrid Control in Robotics: Beginnings, Evolution, Criticism and Trends", Mechanism and Machine Theory, Vol. 30, No. 4, pp.519-32, 1995.

[2.24] M.W.Spong and M.Vidyasagar, *Robot Dynamics and Control*, Wiley, 1989.

[2.25] K.S.Fu, R.C.Gonzalez and C.S.G.Lee, *Robotics : Control, Sensing, Vision and Intelligence*, McGraw-Hill, 1987.

[2.26] A.J.Koivo, *Fundamentals for Control of Robotic Manipulators*, Wiley, 1989.

[2.27] I.M.Whiting, "Tools for the Implementation of Enhanced PID Controllers and their use in Electro-Hydraulic Servo Applications", IEE Colloq. on "Getting the Best Out of PID in Machine Control", Digest No.:96/287, London, UK, pp.5.1-4, 24 October 1996.

[2.28] G.P.Liu and S.Daley, "Optimal Tuning PID Controller Design in the Frequency Domain with Application to a Rotary Hydraulic System", IFAC Journal on Control Engineering Practice, Vol. 7, No. 7, pp. 821-30, 1999.

[2.29] R.P.Paul, "Modelling, Trajectory Calculations, and Servoing of a Computer Controlled Arm", Technical Report AIM-177, Stanford University Artificial Intelligence Laboratory, Stanford, CA, USA, 1972.

[2.30] A.Bejczy, "Robot Arm Dynamics and Control", Jet Propulsion Laboratory Technical Memo 33-669, Pasadena, CA, USA, 1974.

[2.31] J.Zhou, "Experimental Evaluations of a Kinematic Compensation Control Method for Hydraulic Robot Manipulators", IFAC Journal of Control Engineering Practice, Vol. 3, No. 5, pp. 675-84, 1995.

[2.32] O.Khatib, "Commande Dynamique dans l'Espace Operationnel des Robots Manipulateurs en Presence d'Obstacles", Docteur Ingenieur Thesis, L'Ecole Nationale Superieure de l'Areonautique et de l'Espace, Toulouse, France, 1980.

[2.33] O.Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators : The Operational Space Formulation", IEEE Journal of Robotics and Automation, Vol. RA-3, pp. 43-53, 1987.

[2.34] B.Armstrong, O.Khatib and J.Burdick, "The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm", IEEE Int. Conf. on Robotics and Automation, Philadelphia, USA, 1988.

[2.35] M.Raibert, "Mechanical Arm Control Using a State Space Memory", SME Paper MS77-750, 1977.

[2.36] M.E.Khan and B.Roth, "The Near-Minimum Time Control of Open-Loop Articulated Kinematic Chains", Trans. of the ASME Journal of Dynamic Systems, Measurement and Control, Vol. 93, No. 3, pp. 164-72, 1971.

[2.37] B.S.Chen, Y.C.Chang and T.C.Lee, "Adaptive Control in Robotic Systems with H_∞ Tracking Performance", Automatica, Vol. 33, No.2, pp. 227-34, 1997.

[2.38] D.S.Reay, "Variable Structure Control of Industrial Robots", Ph.D Thesis, Cambridge University, 1988.

[2.39] J.J.Slotine, "The Robust Control of Robot Manipulators", Int. Journal of Robotics Research, Vol. 4, No. 2, pp. 46-64, 1986.

[2.40] A.C.Clegg, L.Cellier, P.Dauchez, D.M.Lane and M.W.Dunnigan, "Comparison of Robust and Adaptive Hybrid Position/Force Control Schemes for Hydraulic Manipulators", 6th IARP Workshop on Underwater Robotics, Toulon, France, 27-

29 March 1996.

[2.41] Q.P.Ha, D.C.Rye and H.F.Durrant-Whyte, "Fuzzy Moving Sliding Mode Control with Application to Robotic Manipulators", Automatica, Vol. 35, No. 4, pp. 607-16, 1999.

[2.42] R.R.Y.Zhen and A.A.Goldenberg, "Variable Structure Hybrid Control of Manipulators in Unconstrained and Constrained Motion", Trans. of the ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 118, No. 2, pp. 327-32, 1996.

[2.43] B.Boulet and V.Hayward, "Robust Control of a Robot Joint With Hydraulic Actuator Redundancy", Technical Report TR-CIM-93-7, Centre for Intelligent Machines, McGill University, Quebec, Canada, 1993.

[2.44] B.S.Chen and Y.C.Chang, "Nonlinear Mixed H_2/H_∞ Control for Robust Tracking Design of Robotic Systems", Int. Journal of Control, Vol. 67, No. 6, pp. 837-57, 1997.

[2.45] N.Niksefat and N.Sepehri, "Robust Force Controller Design for a Hydraulic Actuator Based on Experimental Input-Output Data", Proc. of American Control Conference, Piscataway, USA, pp. 3718-22, 1999.

[2.46] Y.Stepanenko and J.Yuan, "Robust Adaptive Control of a Class of Nonlinear Mechanical Systems with Unbounded and Fast Varying Uncertainties," Automatica, Vol. 28, No. 2, pp. 265-76, 1992.

[2.47] H.G.Sage, M.F.de Mathelin and E.Ostertag, "Robust Control of Robot Manipulators: A Survey", Int. Journal of Control, Vol. 72, No. 16, pp. 1498-1522, 1999.

[2.48] F.M.Hughes, "Self-tuning and Adaptive Control - A Review of Some Basic Techniques", Trans. of the Inst. Measurement and Control, Vol. 8, No. 2, pp. 100-110, 1986.

[2.49] T.C.Hsia, "Adaptive Control: A Review", IEEE Int. Conf. on Robotics and Automation, Philadelphia, USA, 1986.

[2.50] H.Lehtinen, "Force Based Motion Control of a Walking Machine", Ph.D Thesis, Technical Research Centre of Finland, 1994.

[2.51] J.J.Craig, P.Hsu and S.S.Sastry, "Adaptive Control of Mechanical Manipulators", Int. Journal of Robotics Research, Vol. 6, No. 2, pp. 16-28, 1987.

[2.52] J.J.Slotine and W.Li, "On the Adaptive Control of Robot Manipulators", Int. Journal of Robotics Research, Vol. 6, No. 3, pp. 49-59, 1987.

[2.53] H.Yu and S.Lloyd, "Combined Direct and Indirect Adaptive Control of Constrained Robots", Int. Journal of Control, Vol. 68, No. 5, pp. 955-70, 1997.

[2.54] S.Dubowsky and D.T.DesForges, "The Application of Model Referenced Adaptive Control to Robotic Manipulators", Trans. of the ASME Journal of Dynamic Systems, Measurement and Control, Vol. 101, pp. 193-200, 1979.

[2.55] F.G.de Almeida, "Model Reference Adaptive Control of a 2 Axes Hydraulic Manipulator", Ph.D Thesis, Bath University, 1993.

[2.56] K.J.Hunt, "A Survey of Recursive Identification Algorithms", Trans. of the Inst. Measurement and Control, Vol. 8, No. 5, pp. 273-8, 1986.

[2.57] P.E.Wellstead and M.B.Zarrop, *Self-tuning Systems*, Chichester, England: Wiley, 1991.

[2.58] K.J.Åström, U.Borisson, L.Ljung and B.Wittenmark, "Theory and Application of Self-tuning Regulators", Automatica, Vol. 13, pp. 457-76, 1977.

[2.59] A.J.Koivo and T.H.Guo, "Adaptive Linear Controller for Robotic Manipulators", IEEE Trans. on Automatic Control, Vol. AC-28, No. 2, pp. 162-71, 1983.

[2.60] N.Sepehri, G.A.M.Dumont, P.D.Lawrence and F.Sassani, "Cascade Control of Hydraulically Actuated Manipulators", Robotica, Vol. 8, pp.207-16, 1990.

[2.61] P.E.Wellstead, D.Prager and P.Zanker, "Pole Assignment Self-tuning Regulator",

Proc. of the IEE: Part D, Vol. 126, pp.781-7, 1979.

[2.62] K.J.Åström and B.Wittenmark, "Self-tuning Controllers Based on Pole-Zero Placement", Proc. of the IEE: Part D, Vol. 127, No. 3, pp. 120-30, 1980.

[2.63] A.J.Koivo, "Self-tuning Manipulator Control in Cartesian Base Coordinate System", Trans. of the ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 107, No. 4, pp. 316-23, 1985.

[2.64] M.A.Lelic and P.E.Wellstead, "Generalised Pole Placement Self-tuning Controller: Part 2 Application to Robot Manipulator Control", Int. Journal of Control, Vol. 46, No. 2, pp. 569-601, 1987.

[2.65] N.Sepehri and G.Wu, "Experimental Evaluation of Generalized Predictive Control Applied to a Hydraulic Actuator", Robotica, Vol. 16, No. 4, pp. 463-74, 1998.

[2.66] G.Wu, N. Sepehri and K.Ziaei, "Design of a Hydraulic Force Control System Using a Generalised Predictive Control Algorithm", Proc. of the IEE: Part D, Vol. 145, No.5, pp. 428-36, 1998.

[2.67] H.Seraji, "Adaptive Force and Position Control of Manipulators", Journal of Robotic Systems, Vol. 4, No. 4, pp. 551-78, 1987.

[2.68] S.Ananthakrishnan and R.M.Fullmer, "Experimental Adaptive Control of a Four Axis Hydraulically Actuated Robot", Int. Journal of Robotics and Automation, Acta Press, Vol. 6, No. 4, 1991.

[2.69] K.Warwick, "Neural Net System for Adaptive Robot Control", in *Expert Systems and Robotics*, T.Jordanides and B.Torbey (editors), Springer-Verlag, Computer Systems Science Series, 71, pp. 601-8, 1991.

[2.70] S.He and N.Sepehri, "Modelling and Prediction of Hydraulic Servo Actuators with Neural Networks", Proc. of the 1999 American Control Conference, USA, pp. 3708-12, 1999.

[2.71] R.M.Sanner and J.E.Slotine, "Structurally Dynamic Wavelet Networks for Adaptive

Control of Robotic Systems", Int. Journal of Control, Vol. 70, No. 3, pp. 405-21, 1998.

[2.72] M.Tokita and T.Fukuda, "Force Control of Robotic Manipulator using Neural Networks", IEEE Int. Conf. on Robotics and Automation, pp. 581-6, Nagoya, Japan, 1995.

[2.73] C.C.Cheah and D.Wang, "Learning Impedance Control for Robotic Manipulators", IEEE Int. Conf. on Robotics and Automation, pp. 2150-5, Nagoya, Japan, 1995.

[2.74] M.Vukobratovic and D.Katic, "Stabilizing Position/Force Control of Robots Interacting with Environment by Learning Connectionist Structures", Automatica, Vol. 32, No. 12, pp. 1733-9, 1996.

[2.75] N.Sepehri, T.Corbet and P.D.Lawrence, "Fuzzy Control System Techniques and Their Application in Hydraulically Actuated Industrial Robots", in *Fuzzy Theory Systems: Techniques and Applications*. Academic Press, USA, pp. 577-608, 1999.

[2.76] F.Y.Hsu and L.C.Fu, "A New Design of Adaptive Fuzzy Hybrid Force/Position Controller for Robot Manipulators", IEEE Int. Conf. on Robotics and Automation, pp. 863-8, Nagoya, Japan, 1995.

[2.77] F.Naghdy and N.P.Nguyen, "Fuzzy Logic Compliance Control of the Peg in Hole Insertion", IFAC Journal of Control Engineering Practice, Vol. 6, No. 12, pp. 1459-74, 1998.

[2.78] J.M.Finney, A. de Pennington, M.S.Bloor and G.S.Gill, "A Pole-assignment Controller for an Electrohydraulic Cylinder Drive", Trans. of the ASME, Journal of Dynamic Systems, Measurement and Control, Vol. 107, pp. 145-50, 1985.

[2.79] K.Z.Karam and K.Warwick, "A Microprocessor Based Adaptive Controller for Robotic Manipulators", IEE Colloq. on Advances in Robot Control, London, November 1989.

[2.80] D.Broome and Q.Wang, "Adaptive Control of Underwater Robotic Manipulators",

5th Int. Conf. on Advanced Robotics, Pisa, Italy, June 1991.

[2.81] A.R.Plummer and N.D.Vaughan, "Robust Adaptive Control for Hydraulic Servosystems", Trans ASME Journal of Dynamic Systems, Measurement and Control, Vol. 118, No. 2, pp. 237-44, June 1996.

[2.82] C.Ozsoy, A.Kural and A.Kuzucu, "Pole-placement position control of a hydraulic robot arm", Proc. of the Inst. of Mechanical Engineers, Journal of Systems and Control Engineering, Vol. 208, No. I3, pp. 149-56, 1994.

[2.83] R.R.Fullmer and S.Ananthakrishnan, "Experimental Adaptive Controllers for a Class of Electrohydraulic Servosystems", Int. Journal of Adaptive Control and Signal Processing, Vol. 7, No. 2, pp.151-62, 1993.

[2.84] T.Eun and H.S.Cho, "On the Implementation of an Adaptive Hybrid Position/Force Control for Manipulators",Proc. of the Inst. of Mechanical Engineers, Journal of Mechanical Engineering Science, Vol. 201, No. C6, pp. 403-12, 1987.

[2.85] Q.Wang, D.R.Broome and I.T.W.Daycock, "Adaptive Force Control of the Puma 560 Industrial Robot Manipulator", IEE Int. Conf. Control '91, Edinburgh, UK, March 1991.

[2.86] D.R.Broome, Q.Wang and A.R.Greig, "Adaptive Compliant Control for an Inspection Robot System", Proc. of the IEE: Part D, Vol. 140, No. 3, pp. 191-7, 1993.

[2.87] A.C.Clegg, A.W.Quinn, M.W.Dunnigan and D.M.Lane, "Practical On-line Motion Planning and Dynamic Control for Robotic Manipulators", IEE Colloq. on "Advances in Practical Robot Controllers", Digest No.:1993/225, London, UK, pp.9.1-4, 26 November 1993.

[2.88] A.R.Plummer and N.D.Vaughan, "Decoupling pole-placement control, with application to a multi-channel electro-hydraulic servosystem", IFAC Journal on Control Engineering Practice, Vol. 5, No. 3, pp. 313-23, 1997.

[2.89] A.J.Koivo, "Force-Position-Velocity Control with Self-Tuning for Robotic Manipulators", IEEE Int. Conf. on Robotics and Automation, San Francisco, USA, pp. 1563-8, April 1986.

[2.90] C.Ozsoy and T.Sisman, "MIMO and SISO self-tuning hybrid position/force control of robotic manipulators", Journal of Robotic Systems, Vol. 13, No. 1, pp. 41-51, Jan. 1996.

Chapter 3

[3.1] R.S.Hartenberg and J.Denavit, "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices", Journal of Applied Mechanics, Vol.77, pp.215-21, June 1955.

[3.2] J.D.Stringer, *Hydraulic Systems Analysis*, MacMillan Press, 1982.

[3.3] W.J.Thayer, "Transfer Functions for MOOG Servovalves", MOOG Technical Bulletin 103, MOOG Controls, New York, 1965.

[3.4] A.C.Clegg, "The Mechanics and Modelling of Hydraulically Actuated Manipulators", Research Memo RM/94/8, Dept. of Computing & Electrical Eng., Heriot-Watt University, November 1994.

[3.5] M.Guillon, *Hydraulic Servo Systems Analysis and Design*, Butterworth, 1969.

[3.6] M.R.Elhami and D.J.Brookfield, "Sequential Identification of Coulomb and Viscous Friction in Robot Drives", Automatica, Vol. 33, No. 3, pp. 393-401, 1997.

Chapter 4

[4.1] R.Isserman, "Practical Aspects of Process Identification", Automatica, Vol. 16, pp. 575-87, 1980.

[4.2] B.Wittenmark and K.J.Åström, "Practical Issues in the Implementation of Self-tuning Control", Automatica, Vol. 20, No.5, pp. 595-605, 1984.

[4.3] C.L.Thornton and G.J.Bierman, "Filtering and Error Analysis via the UDU^T

Covariance Factorisation", IEEE Trans. on Automatic Control, Vol. AC-23, No. 5, pp. 901-7, 1978.

[4.4] D.L.Prager and P.E.Wellstead, "Multivariable Pole-Assignment Self-tuning Regulators", Proc. of the IEE: Part D, Vol. 128, No. 1, pp.9-18, 1981.

[4.5] A.S.Morris and F.Mahieddine, "Microcomputer Implementation of a MIMO Pole-Assignment Self-tuning Regulator", Electronics Letters, Vol. 22, No. 1, pp.56-8, 1986.

Chapter 5

[5.1] A.C.Clegg, "Real Time Trajectory Interpolation for Robotic Manipulators", Research Memo RM/92/2, Dept. of Computing & Electrical Eng., Heriot-Watt University, February 1992.

[5.2] W.T.Townsend and J.K.Salisbury, "The Effect of Coulomb Friction and Stiction on Force Control", IEEE Int. Conf. on Robotics and Automation, pp. 883-9, USA, 1987.

[5.3] A.C.Clegg, M.W.Dunnigan and D.M.Lane, "Modelling of Hydraulically Actuated Manipulators", 7th Int. Conf. on Advanced Robotics, Sant Feliu de Guixols, Spain, Vol.1, pp.351-5, 20-22 September 1995.

Chapter 6

[6.1] "Installation and Operation Manual for Force/Torque Sensors", Manual PN 9610-05-1001-8, Assurance Technologies Ltd., 1993.

[6.2] D.M.Lane, M.W.Dunnigan, P.J.Knightbridge, A.W.Quinn and A.C.Clegg, "A Supervisory Control System for the Automation of Subsea Tasks", IEEE Int. Conf. on Robotics and Automation, Nagoya, Japan, 1995. ISBN 0-7803-2378-5 (PAL)

Chapter 7

[7.1] D.M.Lane, M.W.Dunnigan, A.C.Clegg, P.Dauchez and L.Cellier, "A Comparison between Robust and Adaptive Hybrid Position/Force Control Schemes for Hydraulic Underwater Manipulators", Trans. of the Inst. Measurement and Control, Vol. 19, No. 2, 1997.

Bibliography

The following papers and reports have arisen from work in this thesis.

D.M.Lane, M.W.Dunnigan, A.C.Clegg, P.Dauchez and L.Cellier, "A Comparison between Robust and Adaptive Hybrid Position/Force Control Schemes for Hydraulic Manipulators", Trans. of The Institute of Measurement and Control, Vol.19, No.2, pp.107-16, 1997.

Abstract: Tele-operated hydraulic underwater manipulators are commonly used to perform remote underwater intervention tasks such as weld inspection or mating of connectors. Automation of these tasks to use tele-assistance requires a suitable hybrid position/force control scheme, to specify simultaneously the robot motion and contact forces. Classical linear control does not allow for the highly non-linear and time varying robot dynamics in this situation. Adequate control performance requires more advanced controllers. This paper presents and compares two different advanced hybrid control algorithms. The first is based on a modified Variable Structure Control (VSC-HF) with a virtual environment, and the second uses a multivariable self-tuning adaptive controller. A direct comparison of the two proposed control schemes is performed in simulation, using a model of the dynamics of a hydraulic underwater manipulator (a Slingsby TA9) in contact with a surface. These comparisons look at the performance of the controllers under a wide variety of operating conditions, including different environment stiffnesses, positions of the robot and

dynamic parameters. Conclusions are drawn based on the relative performance of each controller and on the practicalities of the proposed schemes.

D.M.Lane, M.W.Dunnigan, A.W.Quinn and A.C.Clegg, "Motion Planning and Contact Control for a Tele-Assisted Hydraulic Underwater Robot", Journal of Autonomous Robots, Vol. 3, No. 3, pp 233-51, July 1996.

Abstract: Implementing tele-assistance or supervisory control for autonomous subsea robots requires atomic actions that can be called from high level task planners or mission managers. This paper reports on the design and implementation of a particular atomic action for the case of a subsea robot carrying out tasks in contact with the surrounding environment. Subsea vehicles equipped with manipulators can have upward of 1 degrees of freedom (DOF), with degenerate and redundant inverse kinematics. Distributed local motion planning is presented as a means to specify the motion of each robot DOF given a goal point or trajectory. Results are presented to show the effectiveness of the distributed versus non-distributed approach, a means to deal with local minima difficulties, and the performance for a trajectory following with and without saturated joint angles on a robot arm. Consideration is also given to the modelling of hydraulic underwater robots and to the resulting design of hybrid position/force control strategies. A model for a hydraulically actuated robot is developed, taking into account the electrohydraulic servovalve, the bulk modulus of oil, piston area, friction, hose compliance and other arm parameters. Open and closed-loop control results are reported for simulated and real systems. Finally, the use of distributed motion planning and sequential position/force control of a Slingsby TA9 hydraulic underwater manipulator is described, to implement an atomic action for tele-assistance. The specific task of automatically positioning and inserting a Tronic subsea mateable connector is illustrated, with results showing the contact conditions during

insertion.

M.W.Dunnigan, D.M.Lane, A.C.Clegg and I.Edwards, "Hybrid Position/Force Control of a Hydraulic Underwater Manipulator", IEE Proceedings on Control Theory and Applications, Vol.143, No.2, pp.145-51, March 1996.

Abstract: Current generation underwater Remotely Operated Vehicle's equipped with robotic manipulators are teleoperated and consequently place a large workload burden on the human operator. A greater degree of automation could improve the efficiency and accuracy with which underwater tasks are carried out. A hybrid position/force control scheme is proposed to control an industrial hydraulic underwater manipulator, modified to include a force/torque sensor, for tasks such as weld inspection. Modelling of the hydraulic actuation mechanism is performed and the resulting model is easily incorporated in the standard robot dynamic equations. An experimental facility using a PC-based Digital Signal Processor is used to produce practical hybrid position/force results for a Slingsby TA9 hydraulic arm sliding across a planar surface in different parts of its work envelope. Good general agreement is obtained between the simulated and practical systems. A fixed gain control strategy is shown to work well, provided it is tuned for the arm configuration when sliding occurs.

A.C.Clegg, L.Cellier, P.Dauchez, D.M.Lane and M.W.Dunnigan, "Comparison of Robust and Adaptive Hybrid Position/Force Control Schemes for Hydraulic Manipulators", 6th IARP Workshop on Underwater Robotics, Toulon, France, 27-29 March 1996.

Abstract: In order to perform a large class of underwater tasks with a manipulator, such as weld inspection or mating of connectors, a robust or adaptive hybrid position/force

control scheme is necessary. This paper presents two different hybrid control algorithms, the first one is based on a modified Variable Structure Control (VSC-HF) with a virtual environment and the second uses a multivariable self-tuning adaptive controller. A direct comparison of the two proposed control schemes, using a model of a hydraulic underwater manipulator (a Slingsby TA9), is performed in simulation. These comparisons look at the performance of the controllers under a wide variety of operating conditions, including different environment stiffnesses, positions of the robot and dynamic parameters. Conclusions are drawn based on the relative performance of each controller and on the practicalities of the proposed schemes.

D.M.Lane, M.W.Dunnigan, P.J.Knightbridge, A.W.Quinn and A.C.Clegg, "A Supervisory Control System for the Automation of Subsea Tasks", IEEE Int. Conf. on Robotics and Automation, Nagoya, Japan, 1995. ISBN 0-7803-2378-5 (PAL), 0-7803-2377-7 (NSTC).

A.C.Clegg, M.W.Dunnigan and D.M.Lane, "Modelling of Hydraulically Actuated Manipulators", 7th Int. Conf. on Advanced Robotics, Sant Feliu de Guixols, Spain, Vol.1, pp.351-5, 20-22 September 1995.

Abstract: Much of the currently published research on hydraulically actuated manipulators makes use of simple linearised models of the hydraulic components within the robotic mechanism. This paper presents a complete nonlinear model of a hydraulically actuated robot. This will enable more realistic simulations to be developed, yielding results and conclusions that can be more reliably appreciated by users of actual industrial hydraulic manipulators. The nonlinear model developed is based on the shoulder up/down joint of a Slingsby TA9 underwater hydraulic manipulator. This consists of an electrohydraulic servovalve and a linear ram that operates about the pivot of a revolute joint. Other forms

of actuation, such as rotary actuators driving revolute joints, are simpler and thus require only a subset of the model derived here. The model is presented, together with system responses, in a form for use in the MATLAB/SIMULINK modelling and simulation package and is directly applicable to robots with an arbitrary number of hydraulically actuated joints.

A.C.Clegg, M.W.Dunnigan and D.M.Lane, "Force Control and Modelling of Hydraulic Underwater Manipulators", Int. Workshop on Advanced Robotics and Intelligent Machines, Salford, UK, pp.1-8, 5-6 April 1995.

A.C.Clegg, A.W.Quinn, M.W.Dunnigan and D.M.Lane, "Practical On-line Motion Planning and Dynamic Control for Robotic Manipulators", IEE Collo. on "Advances in Practical Robot Controllers", Digest No.:1993/225, London, UK, pp.9.1-4, 26 November 1993.

M.W.Dunnigan, A.C.Clegg and D.M.Lane, "Self-Tuning Control of a 7-Function Hydraulically Powered Underwater Manipulator", 2nd Int. Conf. on Automation, Robotics and Computer Vision, Singapore, Vol.3, pp.RO3.1.1-5, 16-18 September 1992.

Abstract: This paper is concerned with reporting current work on the low-level joint angle control of hydraulically powered underwater manipulators. The self-tuning controller has been chosen in preference to other adaptive schemes because of its conceptual simplicity and the wide variety of design methods offering a trade-off between complexity and performance ability. The selection of model type and order in the estimation block and the corresponding accuracy has been investigated for an independent joint autoregressive model. The controller design methods investigated have been of the deterministic variety

and have comprised of the PI, PID and incremental pole-placement controllers. These adaptive controllers are compared to a fixed gain PID controller on the forearm rotate joint and the transient response improvement is shown in a variety of experimental scenarios.

A.C.Clegg, "The Mechanics and Modelling of Hydraulically Actuated Manipulators", Research Memo RM/94/8, Dept. of Computing & Electrical Eng., Heriot-Watt University, 40 pages, November 1994.

Abstract: This research memo describes how a typical hydraulic actuator is used to drive the joints of a robotic manipulator. The hydraulic system can be separated into two parts, the electrohydraulic servovalve (discussed in Section 2) which regulates the flow of hydraulic fluid, and the actuation mechanism (covered in Section 3) which generates movement at the joint. The mechanics of operation are detailed, together with common assumptions that are made to simplify the analysis of such systems. Much of the currently published research uses a simple linearised model of the servovalve and actuator, both this and the full nonlinear model are detailed here, enabling more realistic simulations to be produced. A full nonlinear model of a typical hydraulically actuated robot joint is developed, based on the shoulder up/down joint of a Slingsby TA9 underwater manipulator. This particular joint consists of a linear hydraulic ram which operates about the pivot of a revolute joint. Other forms of actuation, such as rotary actuators driving revolute joints, are simpler than the type reported on, and thus require only a subset of the model derived here. The model is presented in Section 4, in a form for use in the MATLAB/SIMULINK modelling and simulation package. It can easily be extended to multiple jointed robots, and to the situation where the robot is in contact with the environment.

A.C.Clegg, "Real Time Trajectory Interpolation for Robotic Manipulators", Research

Memo RM/92/2, Dept. of Computing & Electrical Eng., Heriot-Watt University, 18 pages, February 1992.

Abstract: This research memo details a technique used to interpolate, in real time, between joint angle values at variable time intervals, maintaining continuous first and second order time derivatives of the joint angles throughout. This technique has been applied successfully to the interface between the motion planner and the joint angle controller for a TA9 hydraulic manipulator. The motion of the robot arm is now smooth, and passes through all points stipulated by the motion planner maintaining the required temporal intervals.

D.M.Lane, M.W.Dunnigan, P.J.Knightbridge, A.W.Quinn and A.C.Clegg, "Dual Manipulator Collaboration: Issues in Architecture, Planning and Control", 7th Int. Symp. on Unmanned, Untethered Submersible Technology, University of New Hampshire, USA, September 1991.

Abstract: This paper presents some early ideas and results from work on the automated use of coordinated manipulators in the underwater environment. The specific goals is to construct and operate a two manipulator system exhibiting collaborative behaviour using closed loop computer control with higher level activities implementing planning and a task level operator interface. The system would be deployed from a ROV or A-ROV to carry out general inspection and intervention work in typical oil exploration and production tasks. We have employed a functional decomposition to obtain both the system architecture and the structure of the project. Functional modules are investigating self-tuning adaptive control, a multi-agent approach to kinematic guidance and a reactive planner mixing task planning and execution for error recovery. Collaborative behaviour is to be achieved

through co-operation of functional modules dictating task and motion activities. Implementation is proceeding using a network of SUN workstations, PCs and DSP chips and a pair of Slingsby TA-9 7 function hydraulic arms. A companion paper by Chantler details work on sensing for the system.