1989

Experimental adaptive control of a hydraulic robot

S. Ananthakrishnan
Iowa State University

Order Number 9003498

Experimental adaptive control of a hydraulic robot

Ananthakrishnan, S., Ph.D.

Iowa State University, 1989

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

Experimental adaptive control of a hydraulic robot

by

S. Ananthakrishnan

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1989

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express sincere thanks to my major professor, Dr. Rees Fullmer, for his guidance and encouragement throughout this research. I would also like to thank Prof. Bion Pierson, Prof. Jerry Hall, Prof. Jim Bernard, Prof. Donald Flugrad, Prof. Steve Skaar, and Prof. Hal Brockman for serving as members of the thesis committee. My thanks to Mr. Kirk Bigelow and Mr. David McMurrin for their help in building the resolver-to-digital converter.

I thank my wife for her constant encouragement and support during the course of this work. My parents have always been a source of moral support and inspiration in my life. I thank them for guiding me throughout the course of my education.

I dedicate this work to the fond memory of Gus.

ABSTRACT

An adaptive position controller is investigated to compensate for the effects of both the actuator and linkage dynamics in a four axis hydraulic robot. A simulation study was done, prior to the experimental implementation, in order to compare and contrast different adaptive control algorithms for this system. The dynamics of the linkage were developed using Lagrangian techniques. The equations of motion of the robot were generated with a computer program which uses symbolic manipulation. The equations describing the dynamics of the hydraulic actuator were developed using a lumped parameter, control volume analysis.

Implicit and explicit deterministic autoregressive moving average (DARMA) model based adaptive control algorithms were first simulated for a single axis hydraulic servo system. Least-squares identification algorithms with or without covariance resetting or forgetting factors were combined with model reference, pole assignment, and weighted one-step-ahead control algorithms.

The model reference adaptive controller which offered an advantage in the design process was then applied to the entire four axis model. Simulation results indicated a satisfactory response for this multiple-input, multiple-output system.

The results of this discrete-time, model-reference adaptive controller (MRAC) were compared with Craig's continuous-time, state- variable-based, adaptive con-

troller. Simulation results showed that the position response between the two were comparable despite the discretization and unit delay incorporated in our model.

The real-time implementation was then carried out using the DARMA model-based adaptive controllers. The experimental setup consisted of the robot, hydraulic power unit, and the Masscomp control computer. The motion of each axis was controlled by an electrohydraulic servovalve. Geared resolvers were mounted on each axis of the robot to provide the position feedback signal. The supply of hydraulic fluid, at a constant pressure, to the actuators was provided by the hydraulic power unit.

As a prelude to experimental control, identification testing was carried out. First, frequency response tests were performed to obtain the approximate continuous-time model of the plant. Recursive least-squares identification was then performed to determine the first order discrete-time model of the axis. Comparisons were made between the discretized continuous-time plant and the discrete-time model developed using recursive estimation.

Higher order models were later identified for several sets of open loop input-output data to obtain the most representative second, and third order discrete-time models. These average models were used to compare the least-squares estimation error between the models.

A model reference adaptive controller with a first order model for identification and control was first used for single axis testing. Experimental studies were conducted to see the effect of initial parameter estimates, choice of reference model, size of moves, and covariance modification on the position and control response as

well as parameter convergence.

The open loop data indicated a minimum phase plant, while closed- loop identification indicated a nonminimum phase behavior. This was due to the data sampling and computational latency of the control computer. Therefore, the MRAC could not be implemented for higher order models. Further, in using first order model for MRAC, the controller had to be based on a model with no time delay.

The pole assignment adaptive controller (PAAC) which does not require zero cancellation as in MRAC was chosen to overcome this difficulty. A second order DARMA model was used for identification and control. The position response of the PAAC was well damped with no overshoot after the initial transient period of identification. This alternative approach was used in further single axis testing of each axis.

After successful single-axis tests on the individual axis of the robot, two, three, and four-axis PAAC were implemented on the robot. In all the multiple-axis testing, well damped performance was achieved as desired after an initial transient period on all the axes.

NOMENCLATURE

Subscript i refers to axis number of the robot (i = 2 or 3 for translational axes; i = 1 or 4 for rotary axes), while subscripts a and b refer to opposite sides of an actuator.

$A(q^{-1})$ = matrix polynomial in q^{-1}

A_{a_i,b_i} = piston area of link i $(in.^2)$

A_{a_i} = vane actuator area in $(in.^2)$

A_m = reference matrix in state form

A_n = homogeneous transformation matrix

A_p = unknown plant coefficient matrix

$B(q^{-1})$ = matrix polynomial in q^{-1}

B_i = viscous friction force coefficient $(lb - s/in.)$

B_{1_i} = viscous friction torque coefficient $(in. - lb - s)$

B_m = reference input matrix in state form

B_p = unknown plant input coefficient matrix

B_{mm} = effective bulk modulus (psi)

$\bar{B}$ = adaptive gain

C_d = servovalve discharge coefficient

C_{f_i} = coulombic friction force of link i (lb)

C_{f1_i} = coulombic friction torque of link i ($in. - lb$)

$e(t)$ = state error vector

$F(q^{-1})$ = matrix polynomial in q^{-1}

F_i = actuator force for axis i (lb)

$G(q^{-1})$ = matrix polynomial in q^{-1}

$grav$ = gravity field vector

$I_{\bar{x}\bar{x}_i}$ = inertia of link i about x axis ($in. - lb - s^2$)

$J, \bar{J}$ = integral performance indices

J_i = pseudo inertia matrix of link i

$J_N(\theta)$ = performance index for identification

K_{p_i} = piston leakage coefficient for link i ($in.^5/lb - s$)

K_s = spool valve gain ($in./V$)

M_{a_i} = vane actuator moment arm (in.)

m_i = mass of link i ($lb - s^2/in.$)

P = positive definite matrix

$P()$ = covariance matrix

$\bar{P}$ = parameter vector in Craig's approach

P_{a_i,b_i} = piston pressure (psi)

Q_{a_i,b_i} = volume flow rate for link i ($in.^3/s$)

q^{-1} = back shift operator

q_1 = base rotation (rad)

q_2 = vertical translation (in.)

q_3 =horizontal translation (in.)

q_4 = wrist rotation (rad)

$R(s), r(t)$ = reference input in Laplace and time domain

T_i = actuator torque for axis i $(in. - lb)$

$U(t)$ = MIMO input vector

$u(t)$ = SISO input

u_i = servovalve driving voltage (V)

V = Lyapunov function

VO_{a_i,b_i} = actuator volume for translational link i $(in.^3)$

VP_{a_i,b_i} = swept actuator volume for rotational link i $(in.^3/rad.)$

W = servovalve area gradient $(in.^2/in.)$

$Y(t)$ = MIMO output vector

$y(t)$ = SISO output

y_m = state vector of reference model

$\bar{y}_i$ = command displacement of ith axis

$\dot{\bar{y}}_i$ = slew rate of the ith axis

y_p = state vector of plant

$\bar{z}_i$ = center-of-mass offset position for link i (in.)

$\bar{\alpha}(q^{-1})$ = matrix polynomial of the predictor in q^{-1}

$\bar{\beta}(q^{-1})$ = matrix polynomial of the predictor in q^{-1}

ω_{n_i} = natural frequency of the ith axis reference model (rad/s)

$\hat{\theta}$ = estimate of parameters

$\phi()$ = regression vector of y( ) and u( )

ζ = damping coefficient

ρ = fluid density $(lb - s^2/in.^4)$

1 INTRODUCTION

1.1 Introduction to Adaptive Control of Robots

Electrohydraulic robots constitute a small but important class of mechanical manipulators. In many instances, their high load capacity and direct drive actuators make for an attractive alternative to electromechanically actuated robots. Electrohydraulic servo driven systems combine the versatility and the precision available from electrical measurement and signal processing techniques with the rapid response and load capacity of hydraulic cylinder drives. This type of drive provides robust, fast, and accurate movement.

Adaptive control algorithms used in robotic applications usually take into account the linkage dynamics. In this research, an adaptive control approach is proposed to compensate for the effects of both the hydraulic actuator and linkage dynamics in a four-axis robot.

The nonlinearities of the robot can cause the linearly controlled response behavior to vary over the range of motion of the robot. In addition, the controller must deal with both voltage and flow saturation in the servovalve, which determines the maximum speed, or slew rate, of an axis.

To account for these nonlinearities, the lead-lag controllers must be designed

in a conservative fashion to ensure a non-oscillatory response over the range of motion of the manipulator. This compromise can result in impaired performance due to an increased response time of the robot. In this work, an attempt is made to increase the controlled bandwidth of the robot over a wide range of motions while still maintaining a suitable position response. To accomplish this, strategies that directly or indirectly incorporate a model of the system in the control algorithm are considered.

Previous work in both robotics and hydraulics indicate that adequate control designs can be based on linear approximations of the nonlinear behavior. However, the performance of these constant coefficient controllers cannot be assured when applied to nonlinear systems over the entire configuration space of the robot.

The next step is to combine online parameter estimation with online control. Controllers that use coefficients determined by concurrent parameter identification methods in the control law constitute a class of adaptive control algorithms.

The goal of this work is to evaluate the performance of these adaptive control approaches on the hydraulic robot in terms of robustness and performance characteristics. The evaluation includes numerical simulation over the range of motion of the robot, followed by experimental verification.

1.2 Overview of the Research

The major objectives of this research are

1. To investigate adaptive control of multiple-axis hydraulic robot manipulator using a digital controller.

2. To use adaptive position controller to compensate for the effect of actuator and linkage dynamics.

3. To improve the controlled response of the robot over a wide range of motion and maintain a well damped response.

4. To implement deterministic auto-regressive moving average (DARMA) model based controllers.

The robot used in this study is a four-axis electrohydraulic manipulator built by Positech Inc. A four-way servovalve with a critically centered spool is used for controlling the flow across the piston that generates the actuating forces for the linear axes, and the rotary axes use similarly configured vane type actuators. Position measurement is achieved by geared resolvers on each axis.

The dynamics of the hydraulic actuator were modeled using equations describing fluid flow into each side of the piston and a lumped parameter control volume analysis which takes into account compressibility, leakage, and actuator flows.

The force or torque equlibirium equations included the viscous and coulomb friction forces. The dynamics of the linkage were derived using Lagrange's equation of motion. The generation of the dynamic equation of a multiple axis manipulator is complicated. A systematic procedure which uses homogeneous transformation matrices for kinematics and combinations of derivatives of these transformations, pseudo inertia matrix, position vector and gravity force vector for dynamics was used. Since this procedure is in a form convenient for automation, the generation of equations of motion was automated using a symbolic manipulation language called

MACSYMA. The dynamics of the hydraulic actuator combined with the linkage dynamics resulted in a sixteenth order system.

Discrete time adaptive control algorithms were used since they are suitable with the sampled-data behavior of the digital controller. Algorithms based on a deterministic autoregressive moving average (DARMA) model of the plant were used in developing the digital adaptive control algorithms.

The nonlinear differential equations describing the four-axis robot were approximated by a linear, time-varying difference equation with unknown parameters. The resulting DARMA model of the system was used in direct adaptive control algorithms, such as the pole assignment adaptive control algorithm, or converted into a predictor form for indirect adaptive control algorithms, such as the model reference adaptive control (MRAC) algorithm.

Several adaptive control algorithms were investigated in this study, including the weighted one-step-ahead, model reference and pole assignment discrete-time adaptive controllers, and Craig's continuous time adaptive controller. These methods require design parameters to be selected for their implementation. The value of the weighting factor λ in the weighted one-step-ahead controller was found using discrete time root locus analysis. The reference model for the model reference adaptive controller and the desired pole locations for the pole assignment adaptive controller were chosen to provide a critically damped response. The proportional and velocity gains used in Craig's approach were selected in a similar manner. The frequency of the reference model was selected based on the command displacement and slew rate of the axis.

Least-squares-based identification routines were used to estimate the unknown coefficients in the control laws. The identification routine was used only when the robot was in motion to ensure that the parameter estimation algorithm had a persistently exciting signal. In addition, the identification routine uses the actual saturated voltage to the servovalve rather than the analytically determined controller output. In practice, the least-squares algorithm has very fast initial convergence rate, but the algorithm gain reduces dramatically after a few iterations. The covariance matrix was prevented from converging to zero by using a forgetting factor or frequently resetting the covariance matrix to a large initial value.

The proposed adaptive control algorithms were tested initially by simulation using a computer program that allowed the numerical integration of a continuous nonlinear system under the control of a multiple-input, multiple-output, sampled-data controller. Different adaptive control algorithms were implemented with the same generic program by changing only the control algorithm. The aim of the simulation study was to compare and contrast the effectiveness of various adaptive control algorithms as a prelude to experimental testing. Comparisons were made between continuous adaptive control algorithms developed by Craig [10] and the DARMA model based adaptive control schemes used here.

Since adaptive control programs are computationally taxing, the effect of order reduction of the control and identification algorithm was studied. Resolver accuracy plays an important role in the performance of adaptive controllers in real time. A study was conducted to see the effect of resolver noise in the position response and performance of the controller.

Experimental work was carried out using a Masscomp 5450 data acquisition and control computer and a Positech CC-1A robot. The implementation of the proposed adaptive control schemes on the robot was preceeded by real-time testing of these algorithms using an analog computer to represent the linearized dynamics of the robot.

Open loop identification testing was carried out by applying sinusoidal and square wave input voltages to the robot over a range of frequencies. Frequency response plots showing the magnitude and phase relationship between the input voltage and the resulting position were obtained from the sinusoidal response. The approximate open loop transfer function was obtained from the frequency response curves. Least-squares identification tests on the open loop data were used to determine the discrete-time model coefficients. First, second, and third order models were estimated from the open loop data. The error between different order models was compared to determine the tradeoff between complexity and utility of the models.

After successful open loop identification and frequency response tests, simple nonadaptive controllers were designed for implementation. Proportional and model reference controllers were successfully applied to the horizontal axis of the robot. Subsequently, the identification algorithm was added to the model reference controller for model reference adaptive control of the robot. This was followed by experimental testing of the pole assignment adaptive controller. Several adaptive control studies were done to see the effect of initial conditions, covariance modification, and choice of reference models.

After the successful horizontal axis implementation of the adaptive controllers, the pole assignment adaptive controller was implemented on each of the other axes of the robot. In each of the single axis cases, the pole assignment adaptive controller provided the desired well damped position response.

The pole assignment adaptive controller was then extended to the multiple axis case. At first, a two-axis pole assignment adaptive controller was implemented on the vertical and horizontal axes of the robot. Subsequently, all four axes of the robot were adaptively controlled using a closed-loop, pole assignment algorithm. The multiple axis cases also provided a well damped response with very small steady state errors dictated by the resolution of the resolver.

In Chapter Two, different methods of adaptive control are briefly described. The literature in adaptive control of manipulators is also reviewed. Chapter Three discusses the development of the mathematical model of the system. A detailed description of the different adaptive control approaches that were used in this research is given in Chapter Four. The results of simulation studies are discussed in Chapter Five. An overview of the experimental set-up and the results of single and multiple axis experimental testing is presented in Chapter Six. Conclusions and suggestions for further work are given in Chapter Seven. The Appendices contain the different programs developed for simulation and experimental control.

2 METHODS OF ADAPTIVE CONTROL

2.1 Different Methods of Adaptive Control

The underlying idea behind most adaptive control schemes is to combine a parameter estimator with an on-line control algorithm to control a process whose model has unknown and possibly changing parameters. Depending on the configuration and mathematical description adaptive schemes differ. In this section, some of the adaptive techniques which have come to wide use in manipulator control are described.

2.1.1 The MIT Rule

The earliest of the schemes used in adaptive control design was called the MIT rule [40]. This method is based on using a gradient search algorithm to update parameters, which are used in turn to change the gain of the controller. Here, a performance index based on desired and actual output is minimized with respect to the parameters b. Consider the generalized adaptive control system shown in Figure 2.1. The performance index is given by

$$\bar{J} = \int_0^T \| y_m(t) - y_p(t) \|^2 dt \tag{2.1}$$

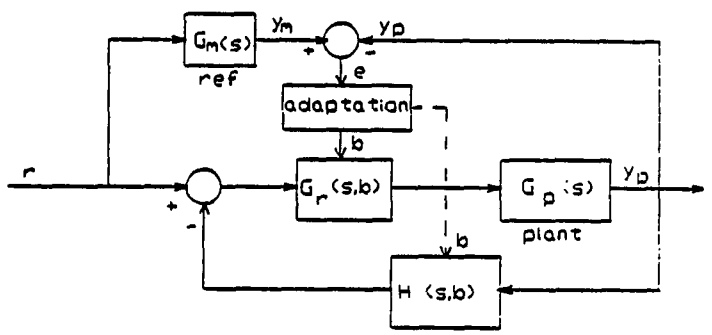

Figure 2.1: Configuration of MRAC used in MIT rule

For a linear system, the following relationships can be obtained from the block diagram:

$$y_m(s) = G_m(s)R(s) \tag{2.2}$$

$$y_p(s) = \frac{G_p(s)G_r(s,b)}{1 + G_p(s)G_r(s,b)H(s,b)}R(s) \tag{2.3}$$

$$y_p(s) = G(s,b)R(s) \tag{2.4}$$

Now using a gradient search algorithm, the estimate of the parameters can be obtained as

$$\dot{b}(t) = -\bar{B}\frac{\partial \bar{J}}{\partial b} \tag{2.5}$$

$\bar{B}$ is an arbitrary constant called the adaptive gain. Defining $e(t) = y_m(t) - y_p(t)$,

the above equation can be rewritten as

$$\dot{b}(t) = -\bar{B}\frac{\partial e}{\partial b}e(t) \tag{2.6}$$

This equation for generating the estimate of the parameters is called the MIT rule. This method of adaptive control has very slow parameter convergence rate, and does not address the problem of stability. The method is typically used in adaptive controllers which require only gain adjustment.

2.1.2 Lyapunov MRAC

Since stability is an important issue in adaptive control design, adaptive control methods based on the Lyapunov stability approach have come into use. In this approach, a differential equation that describes the error between the reference and actual output is first developed. The parameter adjusting scheme is then selected to cause the derivative of the Lyapunov function, which depends on the error and parameter variables, to be negative definite, hence insuring stability via Lyapunov's second theorem.

Consider the following configuration of a model reference adaptive control (MRAC) system shown in Figure 2.2. A Lyapunov scheme can be developed for this system as described by Parks [32]. The equation for the desired reference model in state space representation is of the form:

$$\dot{y}_m = A_m y_m - B_m r \tag{2.7}$$

The reference model is assumed to be stable. The state equation for the plant is

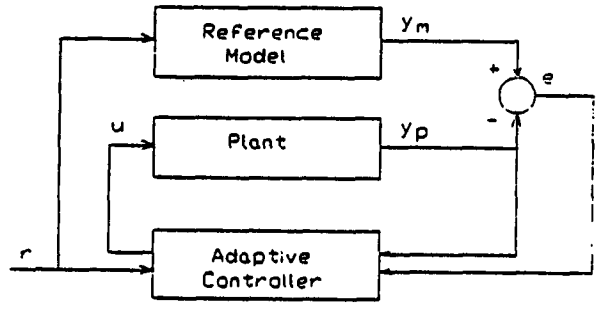

Figure 2.2: Block Diagram of MRAC used in Lyapunov Method

given by

$$\dot{y}_p = A_p(t)y_p + B_p(t)u \tag{2.8}$$

where y_m and y_p are n dimensional state vectors, and r and u are m dimensional input and control vectors. A_p and B_p contain the unknown coefficients.

Defining the error e as $e = y_m - y_p$, the differential equation for the error can be written as

$$\dot{e} = A_m e + f \tag{2.9}$$

where

$$f = (A_m - A_p)y_p + B_m r - B_p u \tag{2.10}$$

The objective is to manipulate f so that the error goes to zero as time goes to infinity. In order to achieve this, a positive definite Lyapunov function V is introduced as follows:

$$V = e^T P e + h(\phi, \psi) \tag{2.11}$$

where ϕ, ψ are parameter vectors which are related to A_p, B_p, etc. The time derivative of V can be obtained as

$$\dot{V} = -e^T Q e + 2e^T P f + \dot{h} \tag{2.12}$$

where

$$-Q = A_m^T P + P A_m \tag{2.13}$$

Now with any $Q^T = Q > 0$, it follows that $P = P^T$ is a unique solution, provided A_M is a stable matrix [7]. The parameter adjustment scheme that ensures a negative definite $\dot{V}$ can be obtained by setting

$$2e^T P f - \dot{h} = 0 \tag{2.14}$$

To explain the above step further, consider a candidate Lyapunov function

$$V = e^T P e + \sum_{i=1}^{n} \sum_{j=1}^{n} \bar{\mu}_{ij} \bar{\alpha}_{ij}^2 - \sum_{i=1}^{n} \sum_{j=1}^{r} \bar{\nu}_{ij} \bar{\beta}_{ij}^2 \tag{2.15}$$

where $\bar{\mu}_{ij}$ and $\bar{\nu}_{ij}$ are real positive constants and $\bar{\alpha}_{ij}$ and $\bar{\beta}_{ij}$ are additional state variables defined by the following equation

$$A_m - A_p = [\bar{\alpha}_{ij}], \quad i, j = 1, \ldots, n \tag{2.16}$$

$$B_m - B_p = [\bar{\beta}_{ij}], \quad i = 1, \ldots, n; j = 1, \ldots, r \tag{2.17}$$

The time derivative of the above specific Lyapunov function is

$$\begin{aligned}\dot{V} &= -e^T Q e + 2\sum_{i=1}^{n}\sum_{j=1}^{n}(\bar{\mu}_{ij}\dot{\bar{\alpha}}_{ij} + y_{pj}e^T p_i)\bar{\alpha}_{ij} \\ &\quad +2\sum_{i=1}^{n}\sum_{j=1}^{r}(\bar{\nu}_{ij}\dot{\bar{\beta}}_{ij} + u_j e^T p_i)\bar{\beta}_{ij} \qquad (2.18)\end{aligned}$$

where p_i is the ith element of P, y_{pj} is the jth element of y_p, and u_j is the jth element of u. If $\bar{\alpha}_{ij}$ and $\bar{\beta}_{ij}$ are so chosen such that the second and third term in the above equation vanish, we obtain the following differential equations to be solved for the control laws:

$$\dot{\bar{\alpha}}_{ij} = -y_{pj}e^T p_i/\bar{\mu}_{ij}, \quad i,j = 1,....n \qquad (2.19)$$

$$\dot{\bar{\beta}}_{ij} = -u_j e^T p_i/\bar{\nu}_{ij}, \quad i = 1,...,n; j = 1,....r \qquad (2.20)$$

This scheme in which the parameter adjusting equations are used to assure that the differential equation describing the error is asymptotically stable is called the MRAC using Lyapunov technique. The advantage of this method over the MIT rule is that stability of the system is assured. However, the choice of a suitable Lyapunov function poses difficulties. The method also requires knowledge of the entire state vector for implementation. This can result in exhaustive and complicated experimental setup even for a simple system.

2.1.3 Hyperstable MRAC

While the Lyapunov approach alleviated problems of stability found in the MIT rule, the need for finding a suitable Lyapunov function poses difficulties. In the hyperstability method due to Landau [26], the search for suitable Lyapunov

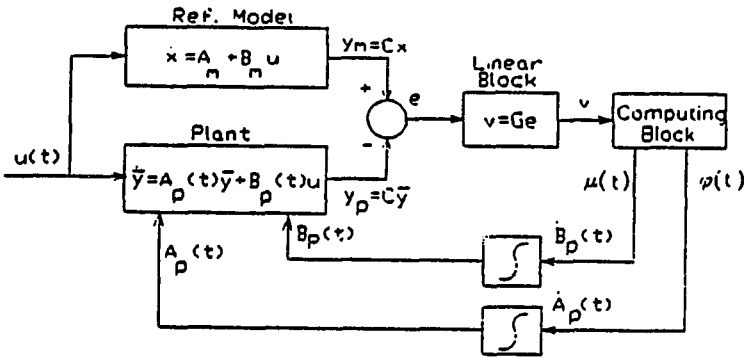

Figure 2.3: Block Diagram of Hyperstable MRAC

function is replaced by relatively straightforward positivity and integral inequality conditions (based on Popov's stability theory [34]). Further, this approach offers greater flexibility in the choice of adaptive laws.

The generalized model reference adaptive control system used in the hyperstability approach is shown in Figure 2.3 .

This system can be described by the following equations [7], [26].

$$\dot{x} = A_m x + B_m u \tag{2.21}$$

$$y_m = Cx \tag{2.22}$$

$$\dot{\bar{y}} = A_p(t)\bar{y} + B_p(t)u \tag{2.23}$$

$$y_p = C\bar{y} \tag{2.24}$$

$$\epsilon = y_m - y_p \tag{2.25}$$

$$e = C(x - y) \tag{2.26}$$

$$e = Cz \tag{2.27}$$

$$v = Ge \tag{2.28}$$

$$\dot{A}_p(t) = \phi(v(\tau), t) \qquad \tau \leq t \tag{2.29}$$

$$\dot{B}_p(t) = \mu(v(\tau), t) \qquad \tau \leq t \tag{2.30}$$

where A_m and B_m are time invariant matrices of dimension (n x n) and (n x m) respectively. A_p(t) and B_p(t) are possibly time varying matrices describing the plant dynamics. C is an (r x n) constant output matrix and G is a constant matrix of order r. Functions ϕ and μ represent the nonlinear dependence between the elements of $\dot{A}_p$(t) and $\dot{B}_p$(t) and the values of v in the interval $\tau \leq t$.

In order to determine the hyperstability criterion for this system, we use the following differential equation in z :

$$\dot{z} = A_m z + \omega_1 I \tag{2.31}$$

where

$$\omega_1 I = [A_m - A_p(t)]\bar{y} + [B_m - B_p(t)]u \tag{2.32}$$

Now from the differential equations relating $\dot{A}_p$ to ϕ and $\dot{B}_p$ to μ, we can define

$$\begin{aligned} \omega(t) &= -\omega_1(t) \\ &= F(v(\tau), t) \qquad \tau \leq t \end{aligned} \tag{2.33}$$

The above two equations combined with

$$v = GCz \tag{2.34}$$

when used in Popov's integral inequality result in the following condition to be satisfied for a hyperstable system.

$$\int_0^t \omega^T(\tau)v(\tau)d\tau \geq -\gamma_0^2 \tag{2.35}$$

This method replaces the search for suitable Lyapunov function by positivity and integral inequality constraints. However, the method is mathematically more complicated than the previous method. Implementation of the above approach also requires access to the full state vectors of the reference model and the adjustable system [7]. This is a significant difficulty because some of the states may be inaccessible. Even when all the states are available, there is increased complexity of the experimental setup and the need for additional instrumentation.

2.1.4 Self-Tuning Regulators

Self-tuning regulators (STR) by Åström and Wittenmark [4], [6] represent an important class of adaptive controllers. Typically a stochastic autoregressive moving average model with auxiliary input (ARMAX) is used to describe the plant dynamics under noisy conditions [4]. In this method, the design procedure specifies a set of desired controller parameters as functions of unknown parameters of the plant. These plant parameters are identified concurrently with the controller algorithm. A regulator with this property is called self tuning, since it automatically tunes the controller to the desired performance.

2.1.5 DARMA model based STR

Goodwin and Sin [21] summarized the STR for the discrete-time case, based on a deterministic autoregressive moving average (DARMA) model of the system. The algorithms are relatively simple and are applicable to multiple- input multiple-output systems with rather general assumptions. There are two methods of self tuning adaptive control which are used in this work. The first is called a direct approach, based on minimum prediction error controllers. Here one thinks of the model as providing a way to predicting the future outputs of a system based on past outputs and past and present inputs. The control action at the present instant of time that would bring the future output to the desired value is calculated at each sample instant. An example of this is one-step-ahead adaptive control, where for a system with delay d, the control action attempts to bring the controlled value at a future time to the desired reference value in one step.

The second approach is called indirect, since the evaluation of the control law is indirectly achieved via the system model. An example of this is the closed-loop pole assignment adaptive controller, where the estimated system parameters are used to generate a new set of variables used in the feedback control law. The control laws generated using the direct or indirect approach, when combined with the parameter estimation schemes such as the least-squares method, result in a class of adaptive control algorithms. These algorithms are used in the simulation and subsequent experimental testing of this project.

For the following conditions,

i) the system is linear and time-invariant,

ii) the delay d of the DARMA model is known,

iii) the zeros of the model (for minimum prediction error controllers) lie inside the unit circle, and

iv) the system input is persistently exciting,

it can be shown [21] that the closed loop system is stable and the output tracking error asymptotically goes to zero. An interesting observation about the convergence of the estimation scheme is that a stable controller can be achieved even if the parameters do not necessarily converge to the true values.

The advantage of these adaptive control methods is that they are conceptually simple. Further, since they are developed in the discrete domain, they are convenient to apply using digital computers. These methods also take into account practical implementation aspects such as delay. These algorithms are based on input and output behavior, and require less setup effort than the state-space methods such as Lyapunov and hyperstability approach, which require knowledge of the entire state at each control step. The mathematical details of the DARMA model based adaptive controllers are presented in Chapter Four.

2.2 Applications of Adaptive Control

Until the early 1970s, adaptive control applications were based on analog realization [24]. These were often not successful because of hardware problems. The renewed interest in this area is partially due to the advent of relatively inexpensive microcomputers, which has made the technology cost effective. However, among

the wide variety of proposed applications in the technical literature, although many have been tested by computer simulation, only a few have yet been tested experimentally.

Some of the areas in which adaptive control has been shown to be useful include load frequency control in an interconnected power system [1], control of motor torque to maintain constant tension in the web when inertia of paper winding wheel changes as paper is wound in a paper mill [7], adaptive autopilot for ship steering [6], and control of processes under load disturbance, temperature and flow variations.

Adaptive control techniques have also been used for positioning of optical tracking telescope [18]. In this case, parameter variations are caused by changes in the moment of inertia of the horizontal plane (moment of inertia depends on the vertical orientation of the telescope), and by the variations of the bearing friction which depend on the angular speed. This application is similar to adaptive control of robot arms where inertia changes can occur due to coupling. Simulation results presented in reference [18] indicate that the position controller worked well under adaptive model-following control.

Porter and Tatnall [35] evaluated the performance of a multi-variable adaptive controller synthesized by the Lyapunov method described earlier. Computer simulation results were presented for simple plant models. The Lyapunov function consisted of a positive definite term dependent on the error and a second quadratic term dependent on the adjustable parameters. It was pointed out that practical implementation of the control laws by this method would require access to the

entire state vector. In a subsequent work, Porter and Tatnall [36] evaluated the performance of the adaptive controller based on Lyapunov method for a hydraulic servo mechanism. The experimental work was carried out using an analog computer. The adaptive gain was generated by the analog computer by defining a differential equation for the gain which depended on the reference state, estimated state, and estimated velocity of the state. It was noted that extensive equipment were required even for this modest system. The authors also indicated difficulties in implementation due to saturation of analog amplifiers.

A semi-automatic scheme using a combination of an on-line fixed gain digital controller and off-line identification algorithm was developed for an electrohydraulic cylinder drive by Finney et al [14]. The computations were expected to take 0.5-2.0 seconds every sampling interval. The coefficients of the on-line, fixed-coefficient controller were changed every 3 seconds using the values calculated from off-line identification.. Drift in the estimated numerator and denominator coefficients of the discrete time plant was noted, even after the substantial initial transient had disappeared. A square root least squares algorithm was used for off-line identification. The paper indicated difficulties in estimating numerator coefficients, which were siginificantly smaller than the denominator coefficients. The desired closed-loop poles were assumed to be combination of first order and underdamped second order terms, and they were restricted to a small domain within the unit circle. The positional accuracy of the proposed semi-automatic scheme and fixed controller was not reported.

2.3 Adaptive Control of Manipulators

In this section, we deal with the application of adaptive control of robotic manipulators. Most of today's industrial robots use a control system whose design is based on treating each joint of the robot arm as simple servomechanism. Such modeling neglects the dynamic coupling and configuration of the entire arm mechanism. This can result in impaired performance due to poor accuracy during motion, restricting the robots use to only limited velocity tasks [10], [27].

In robotics, adaptive control methods can be used to maintain good performance over a wide range of motions and payloads. Several adaptive control schemes have been developed for robot manipulators in the recent years. However, experimental verification of the effectiveness of these techniques is still in the research stage.

Dubowsky and Des Forges [12] proposed a model reference adaptive controller with a linear, second order reference model for each degree of freedom. The manipulator was controlled by tuning the position and velocity gains. The adaptation algorithm was based on the steepest descent method. The coupling among joints and the nonlinear terms in the manipulator equations of motion were neglected in the control design. The results obtained in the simulation study demonstrated that MRAC techniques are suitable to develop control algorithms which will maintain high performance over a wide range of system motions and payloads.

A discrete time investigation of model reference adaptive control was also developed by Dubowsky [13] based on the earlier work of Dubowsky and Des Forges.

Horowitz and Tomizuka's [22] work on adaptive control of mechanical manip-

ulators extended Dubowsky's efforts by using a hyperstability approach, with the coupling among the joints and the nonlinear terms in the manipulator equations of motion explicitly considered in the nonlinear block. The simulation study concluded that the manipulator control system with adaptive controller is insensitive to variations of manipulator configurations and payload. The implementation of this approach required access to the full state vector of both the reference model and of the adjustable system. This poses significant difficulty in implementation.

Tomizuka et al. [38], have implemented both continuous and discrete time adaptive controllers on a laboratory test stand which emulates one axis of a direct drive robot as well as a Toshiba TSR-500V industrial robot. Their results indicate that implementation of adaptive control schemes is feasible for control of direct drive robot arms. It was also pointed out that nonlinear frictional forces arising from gearing are detrimental to performance if not properly compensated. This gives an indication that actuator dynamics should be considered when developing adaptive control algorithms.

Lee et al. [27] reported on an adaptive control approach based on a linear perturbation equation in the vicinity of a desired trajectory. Both combined feedback and feedforward components were computed. The feedforward components provide the nominal torques which compensate for the linkage dynamics, while the feedback component computes the perturbation torques using recursive least squares identification and one step optimal control. Results of this simulation study indicate that the adaptive controller performed better than a Proportional plus Integral controller under various load conditions. However, the authors mentioned that the physical

implementation of the proposed adaptive control requires further investigation of actuator dynamics, frictional forces and so on.

Hsia [23] mentioned that one of the major problems encountered in the adaptive control of indirect drive robots using gear boxes is the gear backlash. He mentioned difficulties encountered in implementing adaptive controllers because of frictional forces in the actuators and gear backlash.

Arimoto and Takegaki [3] developed adaptive control algorithms based on local parameter optimization for a robot described by a linear, time varying model derived from linearization around the desired trajectory. This method assures stability of the error system, but simulations using a nonlinear model resulted in significant position errors.

Craig's [9] approach to adaptive control of robotic manipulators has a structure similar to the computed torque servo [10], but in addition has an adaptive element. After sufficient on-line learning, the control algorithm decouples and linearizes the manipulator so that each joint behaves as an independent second order system with fixed dynamics. Since the torques required to move the manipulator are computed numerically using the dynamic equations, it is difficult to solve the equations in real time.

Craig [10] describes the results of an experimental implementation of the non-linear, model-based adaptive controller. An Adept One robot employing direct drive actuators was used in the experiment. The trajectories were created by smoothly interpolating a set of prespecified positions. The algorithm was written in C using 32-bit integer arithmetic on two Motorola 68000 processors. One processor receives

the desired trajectory from the Adept controller and implements the computed torque servo, while the second processor runs the adaptive control algorithm and updates the first processor's parameter values at the rate of 250 Hz. Prior to implementing the adaptive algorithm on the two link manipulator, reasonable initial estimates of the parameters to be identified were obtained. Further, the parameter estimates were bound within a prespecified limit. If the parameters exceed these values, these bounds would act as saturation limits. In the experiments, the adaption was enabled after the system had been operating certain time, varying from 7.5 secs to 23 secs. Results of parameter estimates based on adaptation using continuous-time modelling were presented. The parameter estimates showed some variations in their values even after sufficient adaptation. It was also pointed out that the adaptive controller did not outperform the Adept's fixed controller. It was concluded by the author that with more engineering effort spent on careful implementation, the adaptive controller should outperform the fixed controller.

Koivo and Guo [25] used an approach to manipulator control which was based on defining time series difference equations for the manipulator system. An autoregressive model expression was used to model the robot, and the parameters of the model were estimated using a recursive least squares algorithm with a forgetting factor. A one-step-ahead adaptive controller was used. This method is based on the same principle used in the self-tuning approach used in this research. However, implementation aspects, such as controller saturation and difficulty in incorporating one-step-ahead controller were not reported by the authors.

Efficient methods for digital control of robotic manipulators based on non-

adaptive pole placement and gain scheduling were discussed by Norcross et al. [31]. Computer simulation results were presented for sampling rates of between 100 Hz to 200 Hz. An integral control mode was used for elimination of disturbances and steady state errors. An example of computation performed on a three- link arm subjected to different reference trajectories showed that the steady state errors were between 7mm to 37 mm, while the accuracy in tracking a circular path of 700 mm diameter depended on the sampling interval.

Stoten [37] developed discrete time algorithms based on MRAC techniques combined with Popov's hyperstability theory. Simulations were carried out to study the effect of system nonlinearities, coupling and parameter variations on the performance of the manipulator. Computer simulations were carried out on a two link arm with a discrete-time control sampling rate of 100 Hz. The reference models were chosen so that a critically damped step response could be obtained. Position and control responses for step changes of 90 degrees on each link show that good performance was achieved with hyperstable MRAC.

Leininger [28] used a pole placement self-tuning algorithm due to Wellstead [39] for closed-loop control of multiple degree-of-freedom manipulator. Computer simulations were carried out for a manipulator with six degrees of freedom. The method demonstrated that the pole placement adaptive controller was effective in position control of the robot. Comparison studies between the position errors of fixed gain and adaptive controllers were not reported.

In summary, several approaches have been suggested for adaptive control of robotic manipulators. Although there has been a good deal of published works

on adaptive control of manipulators, only a few implementations are reported. In many of the practical implementations of the schemes, it was pointed out that neglecting actuator dynamics has a detrimental effect on the performance of the adaptive controller.

Very little work is reported in the area of adaptive control of hydraulic systems. The reported works on hydraulic servosystems and robotics referred to in this research indicate that adaptive control of hydraulic systems and electrohydraulic servo driven robots needs further investigation and research.

3 DEVELOPMENT OF THE SYSTEM MODEL

3.1 Description of the Positech Robot

The robot used in this study is a Positech CC-1A hydraulic material- handling manipulator capable of handling loads of up to 250 pounds in a cylindrical work space. The kinematic configuration of this robot includes two prismatic axes and two rotary axes, as shown in Figure 3.1 . Pistons are used as actuators for the linear axes while rotary vane actuators are used for angular motions. The motion of each axis is controlled by an electrohydraulic servovalve which regulates the flow of hydraulic fluid into both sides of the actuator. The supply of hydraulic pressure and flow to the servovalves is provided by a remotely situated fifteen horsepower hydraulic power unit (HPU), capable of supplying up to fifteen gallons per minute (GPM) at 1700 pounds per square inch (psi). The solenoid actuated gripper on the robot was not included in the model. Geared resolvers are mounted on each axis of the robot to provide a position feedback signal. A commercially available computer (Masscomp 5450) designed for real-time data acquisition was used to perform the closed-loop control of the robot.

3.2 Dynamics of the Hydraulic Actuator

The dynamic model of the system to be controlled includes the dynamics of the hydraulic components as well as the linkage dynamics of the robot. The hydraulic system equations were developed by following classical hydraulic modeling approaches [30]. The model of the linear actuator is presented here. The model of the rotary actuator is similar in form, but has properly scaled torques and angular displacements replacing forces and positions. The form of the hydraulic model for each axis consists of equations describing the servovalve, the fluid flow into each side of the piston, and the force equilibrium equation.

The form of the flow equations for the servovalve are based on a four-way valve with a critically centered spool [11] as shown in Figure 3.2. The driving electronics and the torque motor dynamics of the servovalve were modeled as a DC gain due to their high natural frequency when compared to the natural frequencies of the rest of the components of the system. The flow into each side of the piston chamber (identified by subscripts a and b) is modeled below.

$$Q_{a_i} = \begin{cases} C_d W K_s u_i \sqrt{2(P_s - P_{a_i})/\rho} & if\ u_i > 0 \\ \\ C_d W K_s u_i \sqrt{2P_{a_i}/\rho} & if\ u_i < 0 \end{cases} \tag{3.1}$$

$$Q_{b_i} = \begin{cases} -C_d W K_s u_i \sqrt{2P_{b_i}/\rho} & if\ u_i > 0 \\ \\ -C_d W K_s u_i \sqrt{2(P_s - P_{b_i})/\rho} & if\ u_i < 0 \end{cases} \tag{3.2}$$

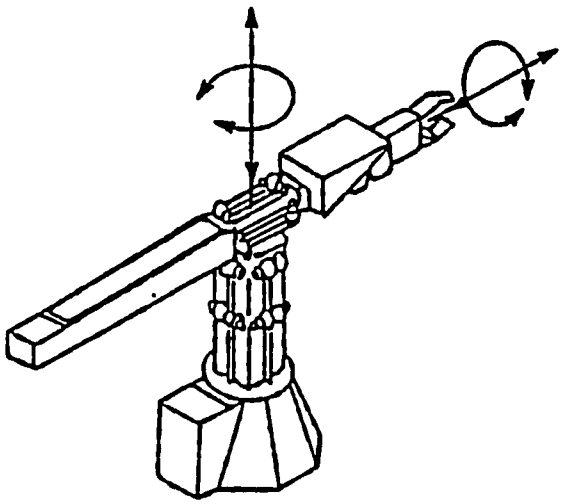

Figure 3.1: Schematic of the Positech Robot

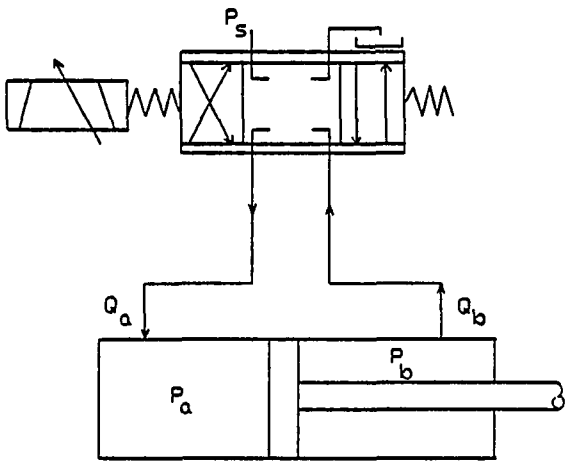

Figure 3.2: Schematic of the Servovalve

where u_i is the input voltage to the servovalve.

A simple lumped parameter control volume analysis for each side of the actuator shows that the flow into one-half of the cylinder supplies compressibility, leakage, and actuator flows. The leakage flow includes an orifice intentionally added by the manufacturer to enhance stability by viscous damping. The continuity equations for each side of the linear axes are

$$Q_{a_i} - \frac{(VO_{a_i} + A_{a_i} q_i)}{B_{mm}} \dot{P}_{a_i} - K_{pa_i}(P_{a_i} - P_{b_i}) = A_{a_i} \dot{q}_i \tag{3.3}$$

$$Q_{b_i} - \frac{(VO_{b_i} - A_{b_i} q_i)}{B_{mm}} \dot{P}_{b_i} - K_{pb_i}(P_{b_i} - P_{a_i}) = A_{b_i} \dot{q}_i \tag{3.4}$$

while for the rotary axis hydraulics, the terms $A_{a_i}\dot{q}_i$, $A_{b_i}\dot{q}_i$ in equations (3.3) and (3.4) are replaced by $VP_{a_i}\dot{q}_i$, $VP_{b_i}\dot{q}_i$, repectively. It should be noted here that the cross-sectional area across each side of the piston is different.

The force available to accelerate an axis is equal to the difference in pressure-times-area across each piston, opposed by both viscous and coulombic frictional forces. A similar expression holds for the torques about the rotary axes.

$$F_i = P_{a_i} A_{a_i} - P_{b_i} A_{b_i} - B_i \dot{q}_i - C_{f_i} Sign(\dot{q}_i) \tag{3.5}$$

$$T_i = P_{a_i} A_{a_i} M_{a_i} - P_{b_i} A_{a_i} M_{a_i} - B_{1_i} \dot{q}_i - C_{f1_i} Sign(\dot{q}_i) \tag{3.6}$$

3.3 Kinematics and Dynamics of the Robot

The equations for the linkage dynamics were developed for the robot by use of a computer program that uses a symbolic language MACSYMA. The results were checked by hand calculation.

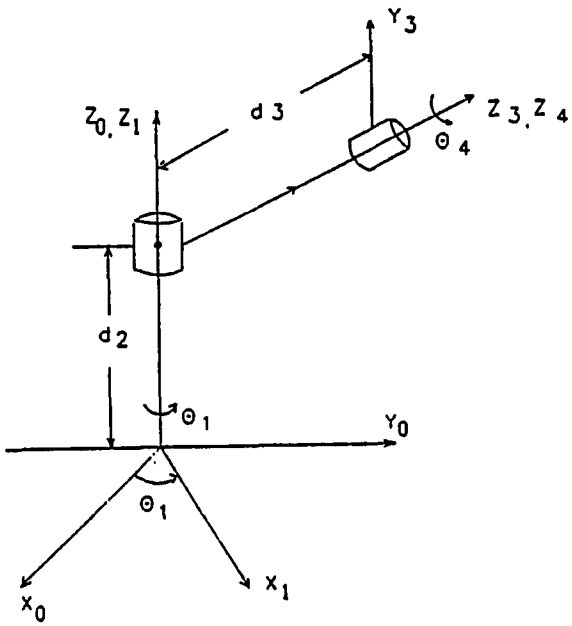

Figure 3.3: Schematic of the Kinematic Configuration

3.3.1 Kinematics

The linkage kinematics can be described using homogeneous coordinate transformations [33]. Figure 3.3 shows the kinematic configuration of the robot. Based on the Denavit-Hartenberg method, a coordinate frame is assigned to each link, and the transformation from the coordinate frame of link n-1 to that of link n is expressed by a series of simple translation and rotation transformations.

i) rotate about z_{n-1} axis an angle θ_n to line up z_{n-1} axis with axis z_n

ii) translate along z_{n-1} axis a distance d_n

iii) translate along the new x_n vector equal to x_{n-1} a length a_n

iv) rotate about x_n an angle α_n

This can be represented as a product of four homogeneous transformations

relating the coordinate frame of link n to the coordinate frame of link n-1. This relationship is

$$A_n = Rot(z,\theta)Trans(0,0,d)Trans(a,0,0)Rot(x,\alpha) \tag{3.7}$$

or equivalently,

$$A_n = \begin{bmatrix} C\theta_n & -S\theta_n C\alpha_n & S\theta_n S\alpha_n & aC\theta_n \\ S\theta_n & C\theta_n C\alpha_n & -C\theta_n S\alpha_n & aS\theta_n \\ 0 & S\alpha_n & C\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

The four parameters θ_n, d_n, a_n, and α_n are the geometric parameters associated with link n. Among these, all are constants except θ_n for revolute joints and d_n for prismatic joints. The transformation from the coordinate frame of link $i-1, i \leq n$ to that of link n is given by

$$^{i-1}T_n = A_i A_{i-1} \ldots A_n \tag{3.9}$$

^0T_i refers to the position and orientation of link i with reference to the base coordinate frame.

3.3.2 Dynamics

The dynamics of the robot are developed using Lagrange's equations of motion. The Lagrangian L is defined as

$$L = K - P \tag{3.10}$$

where K is kinetic energy of the system and P is potential energy of the system.

The dynamic equations, in terms of the coordinates used to express kinetic and potential energy are given by

$$F_i = \frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} \tag{3.11}$$

where q_i are the coordinates in which the kinetic and potential energy are expressed, $\dot{q}_i$ is the corresponding velocity, and F_i the corresponding force or torque, depending on whether q_i is linear or angular coordinate.

After substituting the kinetic energy K and the potential energy P by appropriate matrix representations, the equations of motion for a n degree-of-freedom manipulator have been shown [33] to be

$$F_i = \sum_{j=1}^{n} D_{ij}\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} D_{ijk}\dot{q}_j\dot{q}_k + D_i - I_{a_i}\ddot{q}_i \tag{3.12}$$

where

$$D_{ij} = \sum_{p=max(ij)}^{n} Trace(\frac{\partial T_p}{\partial q_j} J_p \frac{\partial T_p^T}{\partial q_i}) \tag{3.13}$$

$$D_{ijk} = \sum_{p=max(ijk)}^{n} Trace(\frac{\partial^2 T_p}{\partial q_j \partial q_k} J_p \frac{\partial T_p^T}{\partial q_i}) \tag{3.14}$$

$$D_i = \sum_{p=i}^{n} -m_p g^T \frac{\partial T_p}{\partial q_i} {}^p\bar{r}_p \tag{3.15}$$

Here terms of the form D_{ij} represent the coupling inertia between joints i and j. D_{ijk} represents the Coriolis forces at joint i due to velocities at joints j and k. D_i represents gravity effects at link i. T_i is a matrix representing the transformation

between the coordinate frames of link i and the base coordinate frame. ${}^{i}r_i$ is the position vector of the mass center of link i. J_i is the pseudo inertia matrix of link i described with respect to the coordinate frame of link i and is given by

$$J_i = \begin{bmatrix} \frac{-Ixx_i+Iyy_i+Izz_i}{2} & Ixy_i & Ixz_i & m_i x_i \\ Ixy_i & \frac{Ixx_i-Iyy_i+Izz_i}{2} & Iyz_i & m_i y_i \\ Ixz_i & Iyz_i & \frac{Ixx_i+Iyy_i-Izz_i}{2} & m_i z_i \\ m_i x_i & m_i y_i & m_i z_i & m_i \end{bmatrix} \quad (3.16)$$

The gravitational field vector $grav$ is given by

$$grav = [g_x \quad g_y \quad g_z \quad 0]^T \quad (3.17)$$

where g_x, g_y, and g_z are functions of the joint angle θ.

3.3.3 Automatic Manipulator Dynamics Generation

Manual manipulation of manipulator matrix equations is time consuming, error prone, and tedious. The generation of the equations of motion by hand requires several vector and matrix manipulations, and the generated equations may consist of hundreds of terms. Automatic generation of equations of motion using a computer is desirable even for simple manipulators.

A computer program developed by Leu and Hemati [29], is modified to automatically generate the equations of motion. Similar generation schemes have been developed earlier by Fullmer [17]. The program uses MACSYMA, a LISP based computer algebra system devoted to the manipulation of algebraic expressions including variables, integrals, derivatives, functions, and matrices.

The manipulator kinematics is described using homogeneous coordinate transformations based on the Denavit-Hartenberg method. The dynamic equations of motion are generated using Lagrange techniques. The program is capable of generating the equations of motion for any combination of prismatic and revolute joints and for any number of degrees of freedom. It was found on examination that the published paper had an error in generating the D_{ij} terms. This is described and corrected for in the program in Appendix 9.1. In order to verify the modified program, several standard manipulator configurations such as the two-degree-of-freedom pendulum, the three degree-of-freedom modified Stanford arm, and the six degree-of-freedom Stanford arm were tested with the modified program and compared with Paul [33]. The link geometric parameters were generated using a tabulation method given in [33]. The user entries required to run the program, the listing of the actual code used and the procedure to use the program are given in Appendix 9.1.

The equations describing the linkage dynamics generated using MACSYMA are

$$\begin{aligned}
\ddot{q}_1 &= \Big[q_3^2(m_4 - m_3) + 2q_3 m_3 \bar{z}_3 + 2q_3 m_4 \bar{z}_4 - I_{\bar{z}\bar{z}_1} - I_{\bar{y}\bar{y}_3} + I_{\bar{y}\bar{y}_2} \\
&\quad + I_{\bar{x}\bar{x}_4} + cos^2 q_4 (I_{\bar{y}\bar{y}_4} - I_{\bar{x}\bar{x}_4})\Big]^{-1} \\
&\quad \Big[T_1 - 2\dot{q}_1\dot{q}_3 m_3 \bar{z}_3 - 2\dot{q}_1\dot{q}_3 m_4 \bar{z}_4 - 2\dot{q}_1\dot{q}_3 q_3 (m_4 - m_3) \\
&\quad + 2\dot{q}_1\dot{q}_4 cos q_4 sin q_4 (I_{\bar{y}\bar{y}_4} - I_{\bar{x}\bar{x}_4})\Big] \qquad (3.18) \\
\ddot{q}_2 &= \Big[m_4 + m_3 + m_2\Big]^{-1}\Big[F_2 - (m_4 + m_3 + m_2) grav\Big] \qquad (3.19)
\end{aligned}$$

$$\ddot{q}_3 = \left[m_4 + m_3\right]^{-1}\left[F_3 + (m_4 + m_3)\dot{q}_1^2 q_3 + \dot{q}_1^2(m_3\bar{z}_3 + m_4\bar{z}_4)\right] \quad (3.20)$$

$$\ddot{q}_4 = \left[I_{\bar{z}\bar{z}_4}\right]^{-1}\left[T_4 - \dot{q}_1^2 cosq_4 sinq_4(I_{\bar{y}\bar{y}_4} - I_{\bar{x}\bar{x}_4})\right] \quad (3.21)$$

Notice that the dynamic equations are relatively decoupled because of the kinematic design of the robot.

3.4 Dynamics of the System

The complete dynamic model of the robot consists of appropriate versions of the actuator dynamics equations along with the coupled linkage dynamics. This results in a sixteenth order nonlinear model of the robot. The numerical value of the coefficients used in describing the dynamics of the hydraulic actuator as well as the linkage dynamics were found previously by Foley et al.[15] and are listed in the Appendix 9.3.

3.5 Linearized Model of Single Axis

An approximate linearized model may be obtained for each axis if the analysis is restricted to small perturbations around a chosen operating point. Valve flow equations (3.1) and (3.2) relate the flow rate with the two independent variables; spool motion and cylinder pressure. Thus the nonlinear flow equations can be linearized about any desired operating point (x_0, P_0) using Taylor series expansion

$$Q_v \approx Q_{v,0} + \frac{\partial Q_v}{\partial x_v}\Big|_{o.p.}(x_v - x_0) + \frac{\partial Q_v}{\partial P_c}\Big|_{o.p.}(P_c - P_0) \quad (3.22)$$

The above equation can be rewritten as

$$Q_v \approx Q_0 + C_x x_v + C_p P_c \quad (3.23)$$

The coefficients C_x and C_p are called the flow gain and pressure coefficient respectively. These coefficients can be evaluated numerically about a typical operating point.

To linearize equations (3.3) and (3.4) which describe the lumped parameter control volume analysis for each side of the actuator, the volumes $VO_a + A_a q_i$ and $VO_b - A_b q_i$ are assumed to be constant at VO_a and VO_b. This is a good approximation for small changes in q_i. Substituting the linearized version of the flow equations in the lumped parameter control volume equations, the differential equations describing the left and right chamber pressure can be obtained. The coulomb friction term in the force equation can be approximated by an equivalent viscous term in order to linearize the force equation.

The transfer function relating the velocity q_i and the control input u_i can be obtained from the four first order differential equations for P_a, P_b, q_i, and $\dot{q}_i$ as

$$\frac{q_i}{u_i} = \frac{n1\ s + n2}{s(s^3 + d1\ s^2 + d2\ s + d3)} \tag{3.24}$$

where $C_{1a} = \frac{V_{oa}}{B_{mm}}$, $C_{1b} = \frac{V_{ob}}{B_{mm}}$, and

$$n1 = \frac{A_a C_{x_a}}{C_{1_a} m} - \frac{A_b C_{x_b}}{C_{1_b} m} \tag{3.25}$$

$$n2 = \frac{(A_a - A_b) K_{p_b}}{C_{1_a} C_{1_b} m} C_{x_a} + \frac{(A_b - A_a) K_{p_a}}{C_{1_a} C_{1_b} m} C_{x_b} \tag{3.26}$$

and

$$d1 = \frac{K_{p_a}}{C_{1_a}} + \frac{K_{p_b}}{C_{1_b}} + \frac{B_{eq}}{m} \tag{3.27}$$

$$d2 = \frac{K_{p_a} B_{eq}}{C_{1_a} m} + \frac{K_{p_b} B_{eq}}{C_{1_b} m} + \frac{A_b^2}{C_{1_b} m} + \frac{A_a^2}{C_{1_a} m} \tag{3.28}$$

$$\begin{aligned} d3 &= \frac{K_{p_a} A_b^2}{C_{1_a} C_{1_b} m} - \frac{K_{p_a} A_a A_b}{C_{1_a} C_{1_b} m} \\ &- \frac{K_{p_b} A_a^2}{C_{1_a} C_{1_b} m} - \frac{K_{p_b} A_a A_b}{C_{1_a} C_{1_b} m} \end{aligned} \tag{3.29}$$

Notice the open loop pole at the origin, indicating integrator behavior of the open loop system.

The open-loop transfer function for the horizontal axis of the robot can be obtained from the above equation using the parameters obtained either from the blue print specification or through previous identification. The transfer function relating the input servo voltage and output displacement when the leakage coefficient is assumed to be negligible (resulting in $n_2 = 0$, and $d_3 = 0$) is given by

$$\frac{q_i}{u_i} = \frac{33.186 * K_a}{s\,(0.0001\ \ s^2 - 0.013442\ \ s - 2.4995)} \tag{3.30}$$

where K_a is the gain of the voltage to current converter.

The discrete-time representation [16] of the above transfer function at a sampling period of 0.024 secs is given by

$$\frac{y(t)}{u(t)} = \frac{0.0296(z - 0.4980)(z + 0.2755)}{(z - 1)(z - 0.19 \pm 0.057i)} \tag{3.31}$$

3.5.1 Effects of Controller Latency

The computer used for real-time control experiments was found to have a nonsimultaneous digital-to-analog (DA) and analog-to-digital (AD) conversion response.

This led to a time delay, τ, in the controller. A system composed of a time delay and a continuous time state space system can be represented by [5]

$$\dot{x} = \bar{A}x(t) + \bar{B}u(t-\tau) \tag{3.32}$$

where the time delay τ is assumed to be smaller than the sampling period h. Integration of the above equation over one sampling period gives

$$x(kh+h) = e^{\bar{A}h}x(kh) + \int_{kh}^{kh+h} e^{\bar{A}(kh+h-s')}\bar{B}u(s'-\tau)ds' \tag{3.33}$$

where u(t) and $u(t-\tau)$ are piecewise constant control signals. The sampled system obtained above can be written as

$$x(kh+h) = \bar{\Phi}x(kh) - \bar{\Gamma}_0 u(kh) + \bar{\Gamma}_1 u(kh-h) \tag{3.34}$$

where

$$\bar{\Phi} = e^{\bar{A}h} \tag{3.35}$$

$$\bar{\Gamma}_1 = e^{\bar{A}(h-\tau)} \int_0^{\tau} e^{\bar{A}s} ds \bar{B} \tag{3.36}$$

$$\bar{\Gamma}_0 = \int_0^{h-\tau} e^{\bar{A}s} ds B \tag{3.37}$$

The transfer function for the system with delay τ in terms of the shift operator q is given by

$$\bar{H}(q) = \bar{C}(qI - \bar{\Phi})^{-1}(\bar{\Gamma}_0 - \bar{\Gamma}_1 q^{-1}) \tag{3.38}$$

Thus the effect of system delay is to add an additional term in the numerator of the transfer function, while leaving the denominator polynomial unaltered.

Consider the simplified transfer function for the horizontal axis where the higher frequency effects are assumed negligible,

$$G(s) = \frac{33.186\ K_a e^{-\tau\ s}}{2.4995\ s} \tag{3.39}$$

The terms in the discrete-time transfer function can be obtained as

$$\bar{\Phi} = 1 \tag{3.40}$$

$$\bar{\Gamma}_0 = \int_0^{h-\tau} K_a ds \tag{3.41}$$

$$\bar{\Gamma}_0 = 33.186\ K_a(h-\tau) \tag{3.42}$$

$$\bar{\Gamma}_1 = e^{0(h-\tau)} \int_0^{\tau} K_a ds \tag{3.43}$$

$$\bar{\Gamma}_1 = 33.186\ K_a \tau \tag{3.44}$$

Substituting these values in the transfer function yield the following discrete-time transfer function

$$H(q^{-1}) = \frac{33.186\ K_a[(h-\tau)q^{-1} - \tau q^{-2}]}{1-q^{-1}} \tag{3.45}$$

where $\tau < h$. Notice that the time delay causes a nonminimum phase system for $\frac{\tau}{h-\tau} > 1$.

4 ADAPTIVE CONTROL ALGORITHMS FOR DARMA MODELED SYSTEMS

4.1 Description of the DARMA model

In this section, a discrete-time multiple-input multiple-output (MIMO) model in which the current output vector Y(t) is expressed as a linear combination of past outputs, Y(t-j), and past inputs U(t-j-d) is introduced. This will be used to represent the robot dynamics in discrete time.

$$A_0Y(t) = -\sum_{j=1}^{n1} A_jY(t-j) + \sum_{j=0}^{m1} B_jU(t-j-d) \quad t \geq 0 \tag{4.1}$$

A_0 is square and nonsingular and d represents a time delay. Y(t) and U(t) are vectors of dimension m and r respectively. The term in the right hand side of the above equation which depends on the past values of Y is called the autoregressive component, and the term which depends on U, the moving average component [21]. The complete model of equation (4.1), is referred to as the Deterministic Auto-Regressive Moving Average (DARMA) model. Figure 4.1 shows the general configuration of the DARMA based discrete-time adaptive control system. The variable t used in the above equation is an integer representing the sampling period. U(t) represents the control value at the present sampling instant and U(t-1) represents control at the previous sampling instant. For our problem, Y consists of

the four robot positions (m=4) and U consists of the four command voltages (r=4). A fourth order model was chosen for identification and control, consistent with the order of the continuous time system.

The DARMA model can be rewritten as

$$A(q^{-1})Y(t) = B(q^{-1})U(t) \quad t \geq 0 \tag{4.2}$$

where the backward shift operator q^{-1} is defined as:

$$q^{-1}Y(t-j) = Y(t-j-1) \tag{4.3}$$

$A(q^{-1})$ and $B(q^{-1})$ are matrix polynomials in q^{-1} when used in MIMO case and scalar polynomials when used in SISO case.

$$A(q^{-1}) = A_0 + A_1 q^{-1} - .... - A_{n1} q^{-n1} \quad A_0 \ \ nonsingular \tag{4.4}$$

$$\begin{aligned} B(q^{-1}) &= (B_0 - B_1 q^{-1} - .... - B_{m1} q^{-m1}) q^{-d} \\ &= q^{-d} B'(q^{-1}) \end{aligned} \tag{4.5}$$

In general, equation (4.2) can be normalized by multiplying both sides by A_0^{-1}. With $A_0 = I$, the DARMA model can be expressed as

$$Y(t) = \Phi(t-1)^T \theta_0 \quad t \geq 0 \tag{4.6}$$

where θ_0 is an (p x 1) vector of parameters in $A(q^{-1})$ and $B(q^{-1})$ and $\Phi(t-1)^T$ is a (m x p) matrix containing past values of the output and input vectors and Y(t) is an (m x 1) output vector. The model in equation (4.6) is in the general

form convenient for use in the parameter estimation schemes. For a single-input single-output system, the form of $\phi(t-1)^T$ and θ_0 are

$$\phi(t-1)^T = [-y(t-1), -y(t-2), \ldots\ldots, u(t-1), u(t-2), \ldots\ldots] \tag{4.7}$$

$$\theta_0^T = [a_1, a_2, \ldots\ldots, b_1, b_2 \ldots\ldots] \tag{4.8}$$

4.2 Predictor DARMA Representation

Discrete time adaptive control schemes which combine concurrent parameter estimation with control are described in this section. Both direct and indirect adaptive control schemes are considered. The algorithms used follow references [5], [19], [20], and [21].

For the indirect adaptive control schemes, the input-output description of the plant as given in equation (4.2) can be used directly. But, for the direct adaptive control schemes (minimum prediction error controllers) the control law is obtained by trivial manipulation of the following predictor form obtained from equation (4.2):

$$Y(t-d) = \bar{\alpha}(q^{-1})Y(t) + \bar{\beta}(q^{-1})U(t) \tag{4.9}$$

where d is the delay chosen so that the leading coefficient of $B'(q^{-1})$ in equation (4.2) is nonzero. $\bar{\alpha}$ and $\bar{\beta}$ are defined by:

$$\bar{\alpha}(q^{-1}) = G(q^{-1}) \tag{4.10}$$

$$\bar{\beta}(q^{-1}) = F(q^{-1})B'(q^{-1}) \tag{4.11}$$

where $F(q^{-1})$ and $G(q^{-1})$ are unique polynomials of the form,

$$F(q^{-1}) = I + F_1 q^{-1} + \ldots. + F_{d-1} q^{-d-1} \tag{4.12}$$

$$G(q^{-1}) = G_0 + G_1 q^{-1} - \ldots. - G_{n1-1} q^{-n1-1} \tag{4.13}$$

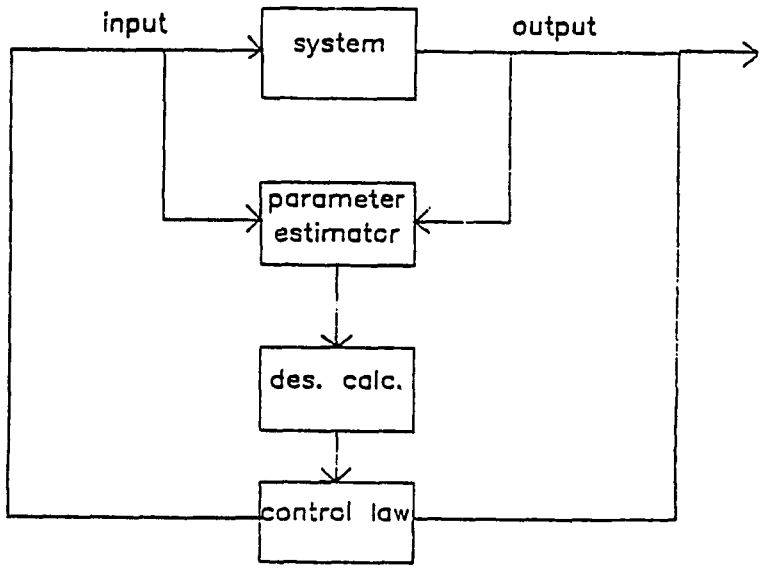

Figure 4.1: Block Diagram of Adaptive Control System

satisfying the following diphontine equation:

$$I = F(q^{-1})A(q^{-1}) + q^{-d}G(q^{-1}) \tag{4.14}$$

The coefficients of the polynomial equation for $F(q^{-1})$ and $G(q^{-1})$ that would satisfy the above equation are presented in Goodwin and Sin [21] without proof. They are derived here.

Proof: The DARMA model of the plant is given by

$$A(q^{-1})Y(t) = B(q^{-1})U(t)$$

Premultiplying either side of the above equation with $F(q^{-1})$, we obtain

$$F(q^{-1})A(q^{-1})Y(t) = F(q^{-1})q^{-d}B'(q^{-1})U(t) \tag{4.15}$$

Now using the relation (equation 4.14) $F(q^{-1})A(q^{-1}) = I - q^{-d}G(q^{-1})$ in the above equation we obtain the predictor

$$Y(t) = q^{-d}G(q^{-1})Y(t) + F(q^{-1})q^{-d}B'(q^{-1})U(t) \tag{4.16}$$

or

$$Y(t+d) = \bar{\alpha}(q^{-1})Y(t) + \bar{\beta}(q^{-1})U(t) \tag{4.17}$$

The values of $F(q^{-1})$ and $G(q^{-1})$ that satisfy

$$I = F(q^{-1})A(q^{-1}) + q^{-d}G(q^{-1})$$

are computed as follows:

We expand the above relation, keeping in mind that the delay d is less than n1, as follows:

$$\begin{aligned} I &= (I + F_1 q^{-1} - \ldots - F_{d-1}q^{-d+1}) \\ &\quad (I + A_1 q^{-1} + \ldots + A_{d-1}q^{-d+1} + A_d q^{-d} + \ldots + A_{n1}q^{-n1}) \\ &\quad + q^{-d}(G_0 + G_1 q^{-1} + \ldots + G_{n1-1}q^{-n1-1}) \end{aligned} \tag{4.18}$$

The result of this expansion is

$$\begin{aligned} I &= I + (F_1 + A_1)q^{-1} + (F_2 + A_2 + F_1 A_1)q^{-2} + \ldots + \\ &\quad (F_{d-1} + A_{d-1} + A_{d-2}F_1 + \ldots + A_1 F_{d-2})q^{-d-1} + \\ &\quad q^{-d}(G_0 + A_d) + (G_1 + F_1 A_d + A_{d+1})q^{-1} + \\ &\quad (G_2 + F_2 A_d + F_1 A_{d-1} + A_{d+2})q^{-2} + \ldots \end{aligned} \tag{4.19}$$

In order to satisfy above equation, the coefficients of q^{-1}, q^{-2},..., should vanish. This requirement defines the recursive relations for $F(q^{-1})$ and $G(q^{-1})$ as follows:

$$F_i = -\sum_{j=0}^{i-1} F_j A_{i-j}; \quad i = 1, ..., d-1 \tag{4.20}$$

$$G_i = -\sum_{j=0}^{d-1} F_j A_{i+d-j}; \quad i = 0, ..., n1-1 \tag{4.21}$$

It should be noted that for the case of unit delay, (q^{-1}), the term $F(q^{-1})$ reduces to the identity matrix.

4.3 Methods of Parameter Estimation

Consider the following quadratic cost function described for a single-input single-output DARMA model of the system:

$$J_N(\theta) = \frac{1}{2}\sum_{t=1}^{N}(y(t) - o(t-1)^T\theta)^2 - \frac{1}{2}(\theta - \hat{\theta}(0))^T P_0^{-1}(\theta - \hat{\theta}(0)) \tag{4.22}$$

Here the cost represents the sum of the squares of the model errors $e(t) = y(t) - o(t-1)^T\theta$, which is the difference between the actual observation y(t) and the value predicted by the model with the parameter vector θ. The second term accounts for possible errors in the initial parameter estimates.

The minimization of the above performance index results in the Recursive Least Squares Algorithm [21] which follows:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P(t-2)o(t-1)}{1 + o(t-1)^T P(t-2)o(t-1)}(y(t) - o(t-1)^T\hat{\theta}(t-1)) \tag{4.23}$$

$$t \geq 1$$

$$P(t-1) = P(t-2) - \frac{P(t-2)o(t-1)o(t-1)^T P(t-2)}{1 + o(t-1)^T P(t-2)o(t-1)} \tag{4.24}$$

with $\hat{\theta}(0)$ given and P(-1) is any positive definite matrix P_0. P_0 can be seen as a measure of confidence in the initial estimate $\hat{\theta}(0)$. Typically a diagonal matrix with identical elements whose numerical values vary from 10 to 10000 is used for P_0.

This is the well known recursive least-squares algorithm. In practice, the ordinary least-squares algorithm has very fast initial convergence rate, but the algorithm gain reduces dramatically when the covariance matrix P gets small after a few iterations.

Schemes in which the ordinary least-squares algorithm is modified are often used. One scheme in which the covariance matrix is reset to $\bar{P}_0$ at specific intervals is called the least squares-algorithm with covariance resetting. This procedure will revitalize the identification algorithm and is useful in maintaining an overall fast convergence rate. The least squares algorithm with covariance resetting is similar to ordinary least squares algorithm with equation (4.24) replaced by equation (4.25) at specific times t_i. Let $Z_s = t_1\ t_2\ t_3\ ....$ be the times at which resetting occurs; then for $t \ni Z_s$ an ordinary sequential least-squares update is used. At times $t = t_i \in Z_s$, $P(t_i - 1)$ is reset as follows:

$$P(t_i - 1) = k_i I \tag{4.25}$$

Another method of covariance modification which has been used to achieve parameter convergence is the least-squares algorithm with exponential weighting of data. Here, in the basic least squares algorithm, a factor $\bar{\lambda}$, called the forgetting factor, is incorporated in such a way that the most recent value of the covariance matrix P is given more weight when compared to the past values. While $\bar{\lambda}$ can take any value between 0 and 1 ($0 < \bar{\lambda} \leq 1$), typical values of $\bar{\lambda}$ range from 0.95 to

0.99. The cost function for the ordinary least squares as given by equation (4.22) is modified to include a term $\bar{\lambda}^{N-t}$ within the summation. This yields the following parameter and covariance update scheme.

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P(t-2)\phi(t-1)}{\bar{\lambda} + \phi(t-1)^T P(t-2)\phi(t-1)}[y(t) - \phi(t-1)^T\hat{\theta}(t-1)] \quad (4.26)$$

$$t \geq 1$$

$$P(t-1) = \frac{1}{\bar{\lambda}}[P(t-2) - \frac{P(t-2)\phi(t-1)\phi(t-1)^T P(t-2)}{\bar{\lambda} + \phi(t-1)^T P(t-2)\phi(t-1)}] \quad (4.27)$$

4.4 Methods of Adaptive Control

This section deals with the design of discrete-time adaptive control algorithms. The approach used is to combine a particular estimation technique with any control law. Both direct and indirect adaptive control algorithms are developed. Weighted one-step-ahead and model reference adaptive controllers fall under the direct category, since the control law parameters are simply the model parameters in the predictor form. Closed-loop pole assignment adaptive controller is an example of indirect algorithm, since the evaluation of the control law is indirectly achieved by first identifying the system model, and then using these values to calculate a certainty-equivalence controller.

4.4.1 Weighted One-Step-Ahead Adaptive Control

For the model of the single-input single-output system defined by the predictor form, the control law minimizing the following performance index:

$$J(t+d) = \frac{1}{2}(y(t+d) - y^*(t+d))^2 + \frac{\lambda}{2}u(t)^2 \quad (4.28)$$

at each time instant, with estimates of parameters used in the control law, is called the Weighted One-Step-Ahead Adaptive Control law. Notice that λ differs from $\bar{\lambda}$ used for least-squares identification with covariance forgetting.

This adaptive control law has been shown [21] to be expressed in the form

$$u^*(t) = \phi(t)^T \hat{\theta}(t) \tag{4.29}$$

where

$$\begin{aligned} \phi(t)^T = [y^*(t+d),\ -y(t),\ \ldots,\ -y(t-n+1),\ -u(t-1), \ldots, \\ \ldots\ -u(t-m-d+1)] \end{aligned} \tag{4.30}$$

The values of $\hat{\alpha}$ and $\hat{\beta}$ that are used in $\hat{\theta}$ are estimated using one of the parameter estimation schemes described previously. The effective discrete-time pole location can be found by selecting a value of λ. The value of the control weighting factor λ in the performance index is found using a discrete time root locus analysis on the unit circle.

4.4.2 Model Reference Adaptive Control

The model reference adaptive control schemes developed in this section follow the same general principle used in other DARMA-based methods discussed previously. However, the method of implementation is different from the MRAC method using either the Lyapunov or Hyperstability approaches. Figure 4.2 shows the configuration of the predictor-based MRAC system used in this work. In this section, we again use the single-input single-output versions of the DARMA model

of the system as given in equation (4.2). In addition, the following reference model assumptions are made.

1) The desired output $y^*(t)$ for a reference input r(t) satisfies the reference model expressed as follows:

$$E(q^{-1})y^*(t) = q^{-d'}gH(q^{-1})r(t) \tag{4.31}$$

with the associated transfer function $G(z^{-1}) = z^{-d'}H(z^{-1})g/E(z^{-1})$, where g is a constant gain and

$$H(z^{-1}) = h_0 - h_1 z^{-1} + ... + h_l z^{-l}, \quad h_0 = 1 \tag{4.32}$$

$$E(z^{-1}) = \epsilon_0 + \epsilon_1 z^{-1} - .... + \epsilon_l z^{-l}, \quad \epsilon_0 = 1 \tag{4.33}$$

2) $E(z^{-1})$ is stable

3) d' = d

Previously, y(t + d) was predicted and the control law was chosen so as to set it equal to $y^*(t+d)$, since the objective was to achieve $y(t+d) = y^*(t-d)$. Here, the prediction is $E(q^{-1})$y(t) and the control value is chosen so as to set it equal to $q^{-d}H(q^{-1})gr(t)$, since the objective is to achieve $E(q^{-1})y(t) = q^{-d}H(q^{-1})gr(t)$. The predictor form in equation (4.9) for a SISO system becomes

$$E(q^{-1})y(t+d) = \alpha(q^{-1})y(t) + \beta(q^{-1})u(t) \tag{4.34}$$

where

$$E(q^{-1}) = F(q^{-1})A(q^{-1}) + q^{-d}G(q^{-1}) \tag{4.35}$$

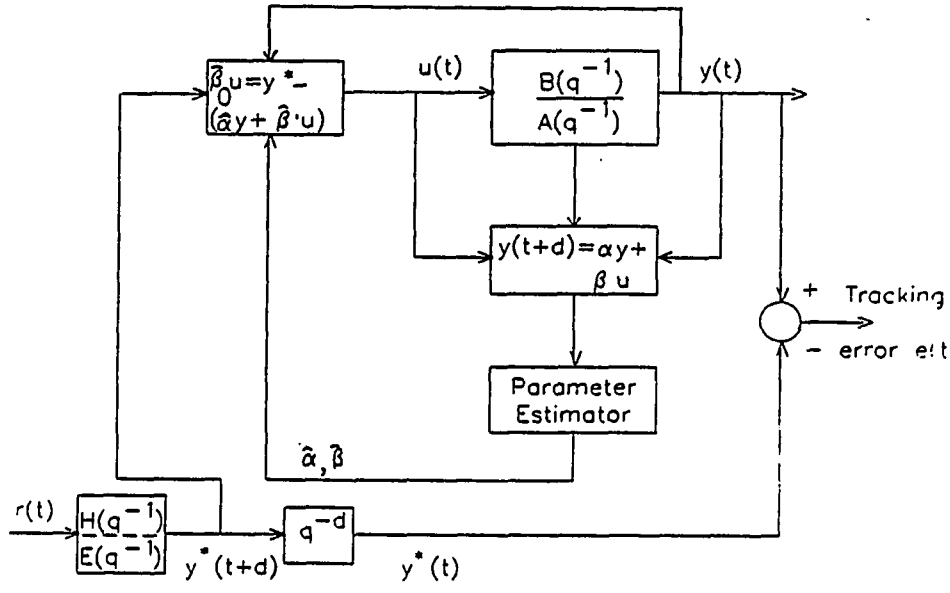

Figure 4.2: Block Diagram of MRAC System

Here $F(q^{-1})$, $A(q^{-1})$, $G(q^{-1})$, $\alpha(q^{-1})$, and $\beta(q^{-1})$ are polynomials in q^{-1}. The model reference adaptive control law can now be obtained as a generalization of the one-step-ahead adaptive controller as [21]

$$u^*(t) = \frac{1}{\hat{\theta}_{n+1}(t)}[\hat{\theta}_1 y(t) \ldots \hat{\theta}_n y(t-n+1) - \hat{\theta}_{n+2} u(t-1) - \ldots - \hat{\theta}_{n+m+d} u(t-m-d+1) + r_a(t)] \tag{4.36}$$

where $r_a(t) = gH(q^{-1})r(t)$. $\hat{\theta}$ is the parameter vector estimated using one of the parameter estimation schemes.

The advantage of this model reference control algorithm is that the control law parameters are simply the parameters in the one-step-ahead predictor. They can be estimated directly, making the control design very simple.

In the following derivation, it is shown that solving for the closed loop transfer

function of the model reference controller results in pole-zero cancellation.

Consider the predictor form of single-input single-output DARMA model given in equation (4.34). The model reference control law can be obtained by substituting for $\alpha(q^{-1})$ and $\beta(q^{-1})$ as

$$u(t) = \frac{gH(q^{-1})r(t) - G(q^{-1})y(t)}{F(q^{-1})B'(q^{-1})} \tag{4.37}$$

Substituting the above control law in the DARMA model

$$A(q^{-1})y(t) = q^{-d}B'(q^{-1})u(t)$$

results in a closed loop system whose transfer function is

$$CLTF = \frac{B'(q^{-1})q^{-d}gH(q^{-1})r(t)}{B'(q^{-1})[F(q^{-1})A(q^{-1}) + q^{-d}G(q^{-1})]} \tag{4.38}$$

Notice that both the numerator and denominator have a common factor $B'(q^{-1})$. Further the second term $(F(q^{-1})A(q^{-1}) + q^{-d}G(q^{-1}))$ is nothing but the observer $E(q^{-1})$ as defined by the Diphontine equation (4.14).

Thus, the model reference control scheme can be thought of as a special case of pole assignment control in which the observer is chosen to have dynamics given by $E(q^{-1})$ and the closed loop poles are assigned to $B'(q^{-1})$. This pole-zero cancellation restricts the use of the algorithm to only sytems with stable $B'(q^{-1})$ [21].

4.4.3 Pole Assignment Adaptive Control

The pole assignment adaptive control (PAAC) method differs from model reference and one-step-ahead adaptive control in that this is an indirect adaptive

controller which uses the DARMA model directly. The objective of this method is to select a controller which will assign specific roots (poles) to the closed-loop characteristic equation. Figure 4.3 shows the block diagram of the pole assignment adaptive control system.

The control input u(t) in equation (4.45) at each time instant is determined by solving the following pole assignment equation

$$\hat{A}(t,q^{-1})\hat{L}(t,q^{-1}) + \hat{B}(t,q^{-1})\hat{P}(t,q^{-1}) = A^*(q^{-1}) \tag{4.39}$$

where $k = max(n1, m1)$, $A^*(q^{-1})$ is a polynomial of order $2k-1$, and $\hat{A}(t,q^{-1})$ and $\hat{B}(t,q^{-1})$ are estimated plant polynomials in q^{-1}. The dominant roots of the polynomial $A^*(q^{-1})$ are chosen based on the reference model selection scheme discussed in Section 4.6. The rest of the poles were assigned to well damped values. $\hat{L}(t,q^{-1})$ and $\hat{P}(t,q^{-1})$ are unique polynomials of order $k-1$ obtained by solving the pole assignment equation. It should be noted that the minimal degrees for L,P, and A^* are $m1, n1-1, n1-m1$ respectively. If a fourth order model is used for the single-input single-output system, then L and P would each be a (4 x 1) column vector.

$\hat{A}(t,q^{-1})$ and $\hat{B}(t,q^{-1})$ are polynomials in q^{-1} describing the unknown plant dynamics as follows

$$\hat{A}(t,q^{-1}) = \hat{a}_0 - \hat{a}_1(t)q^{-1} + ... - \hat{a}_k(t)q^{-k} \quad \hat{a}_0 = 1 \tag{4.40}$$

$$\hat{B}(t,q^{-1}) = \hat{b}_1(t) + ... + \hat{b}_k(t)q^{-k} \tag{4.41}$$

The coefficients of the above polynomials can be obtained by recursive estimation using least squares algorithms. The coefficients of the polynomials $\hat{L}$ and $\hat{P}$

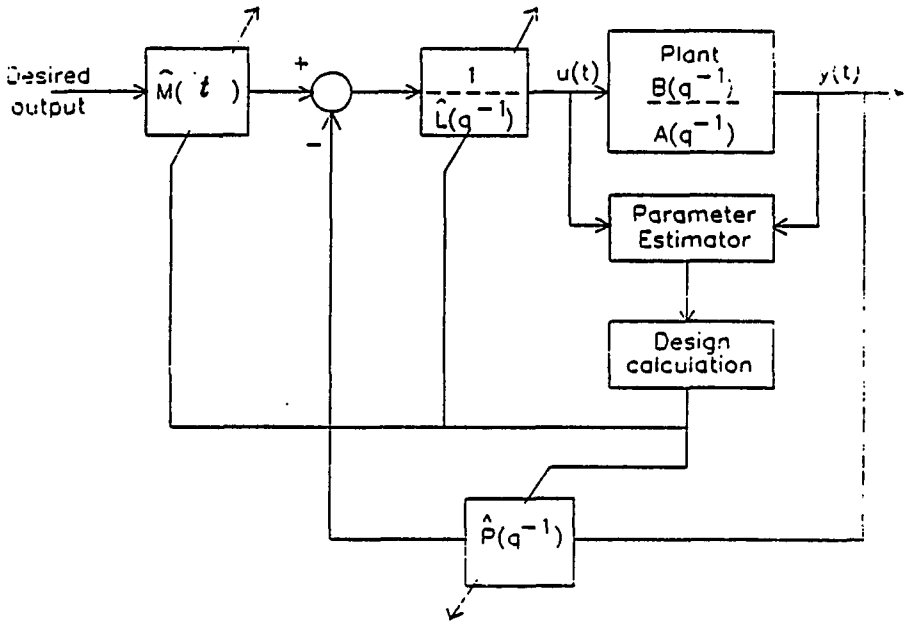

Figure 4.3: Block Diagram of PAAC System

are then obtained by solving the following linear equation

$$M_e \begin{bmatrix} \hat{l}_0 \\ \cdot \\ \hat{l}_{k-1} \\ \hat{p}_0 \\ \cdot \\ \hat{p}_{k-1} \end{bmatrix} = \begin{bmatrix} a_0^* \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ a_{2k-1}^* \end{bmatrix} \tag{4.42}$$

where

$$M_e = \begin{bmatrix} \hat{a}_0 & & & & & \\ \hat{a}_1 & \hat{a}_0 & & \hat{b}_0 & & \\ . & \hat{a}_1 & . & \hat{b}_1 & \hat{b}_0 & \\ \hat{a}_k & . & . & . & . & . \\ & \hat{a}_k & . & & . & \\ & & \hat{a}_k & & & . \end{bmatrix} \tag{4.43}$$

The feedback control law can be obtained as a solution of

$$\hat{L}(t, q^{-1})u(t) = \hat{M}(t)y^*(t) - \hat{P}(t, q^{-1})y(t) \tag{4.44}$$

where the scalar $\hat{M}()$ is a gain term obtained from

$$\hat{M} = \frac{1 - a_1^* - a_2^* - \ldots}{\hat{b}_1 + \hat{b}_2 + \ldots} \tag{4.45}$$

where the values of a_i^* are the coefficients of A^*. As can be seen from this algorithm, the pole assignment controller overcomes the restrictive minimum phase assumptions on the plant zeros of the MRAC scheme. However, this method requires additional matrix computations for estimates of $\hat{L}$ and $\hat{P}$ which add to the complexity of real time implementation.

4.5 Craig's Approach to Adaptive Control

Since the adaptive control approach used in this research is only one among several possible approaches, a comparison of the proposed approach and an existing approach would be useful. In this section, Craig's approach [9] to adaptive control

of manipulators is described based on Lyapunov MRAC. In this method, the overall adaptive control system maintains the structure of the computed torque servocontroller, but in addition has an adaptive element. Figure 4.4 shows the structure of the adaptive controller used by Craig.

The manipulator equation of motion for a general n-axis robot can be written as

$$T = M(\bar{q})\ddot{\bar{q}} - Q(\bar{q}, \dot{\bar{q}}) \tag{4.46}$$

with $\bar{q}$ representing the generalized coordinates of the robot. The above equation is a compact form of the equations of motion described in Chapter Three. The notation used here agrees with Craig [10].

To control the manipulator, the following control law is proposed.

$$T = \hat{M}(\bar{q})\ddot{\bar{q}}^* + \hat{Q}(\bar{q}, \dot{\bar{q}}) \tag{4.47}$$

with $\hat{M}(\bar{q})$ and $\hat{Q}(\bar{q}, \dot{\bar{q}})$ the estimates of $M(\bar{q})$ and $Q(\bar{q}, \dot{\bar{q}})$ and

$$\ddot{\bar{q}}^* = \ddot{\bar{q}}_d + K_v\dot{E} + K_pE \tag{4.48}$$

where $\bar{q}_d$ is the desired trajectory.

The servo error E in the above equation is defined as

$$E = \bar{q}_d - \bar{q} \tag{4.49}$$

K_v and K_p are diagonal gain matrices of order n representing velocity and position controller gain.

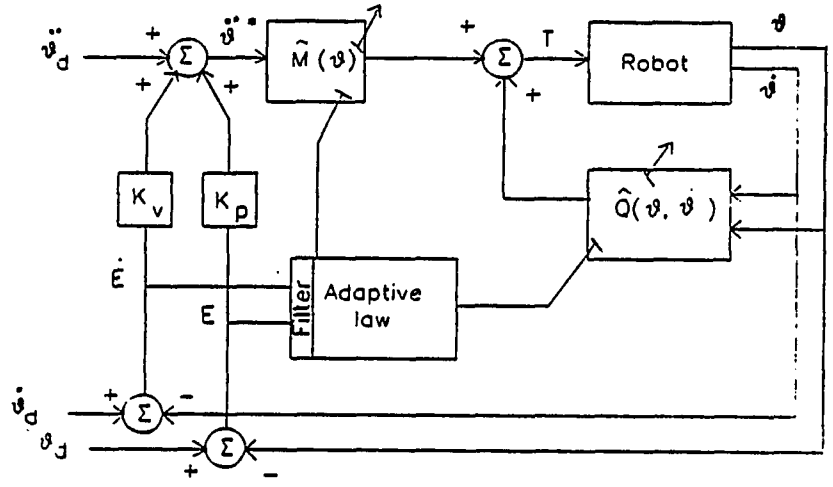

Figure 4.4: Block Diagram of Craig's Adaptive Control System

The adaptive law will compute the change in parameter estimates as a function of the filtered servo error signal E_1, which is obtained from

$$E_1 = \dot{E} + \Psi E \tag{4.50}$$

where Ψ is a diagonal matrix of order n with positive constants. The elements of the matrix Ψ are chosen such that the transfer function

$$TF = \frac{s + \psi_j}{s^2 + k_{vj}s + k_{pj}} \tag{4.51}$$

is strictly positive real. k_{vj} and k_{pj} are the elements in the K_v and K_p matrices.

Define Φ as an (r x 1) vector of parameter errors $\bar{P} - \hat{\bar{P}}$, where $\bar{P}$ describes the r system parameters and $\hat{\bar{P}}$ are the respective parameter estimates obtained by

solving the following parameter update differential equation

$$\dot{\hat{P}} = \Gamma W^T \hat{M}^{-1} E_1 \tag{4.52}$$

Γ is a diagonal matrix of order r with each element γ_i greater than zero. W is a (n x r) matrix of functions which depends on $\bar{q}$, $\dot{\bar{q}}$, and $\ddot{\bar{q}}$ based on dynamic equations.

The parameters of the robot such as the link length and piston areas are fairly well known from the blueprint specifications. The parameters which are usually estimated are friction coefficients, masses, and inertia. The advantage of this method is that the unknown parameters are physical values of the system. However, this method does not take into effect the discretization and delay which exists in real-time digital control implementation of the control algorithm. Further, the dynamics of the system have to be solved at each instant of time to generate the computed torque required to actuate the system. Finally, the actuator must be capable of directly applying the required torque.

4.6 Method of Model Selection

The methods of adaptive control used require either the specification of reference model or the desired closed-loop poles. The reference model for the model reference adaptive controller was based on second order, linear, time invariant dynamics. The desired closed loop poles for the pole assignment adaptive controller and the gains K_v and K_p for Craig's approach were also selected in a similar manner.

The desired reference model was chosen to have a damping ratio of unity . The natural frequency of the model was selected based on the command displacement

and the maximum speed of the axis often referred to as the 'slew rate'. By this procedure, fast response could be achieved by taking into account the controller saturation in the design.

Consider the reference model for each axis of the robot given by the following differential equation:

$$\ddot{y}_i + 2\zeta\omega_{n_i}\dot{y}_i + \omega_{n_i}^2 y_i = \omega_{n_i}^2 \bar{y}_i \qquad (4.53)$$
$$i = 1, \ldots 4$$

For a step input with ($\zeta = 1$) and magnitude $\bar{y}_i$, the response y_i is

$$y_i(t) = \left[\frac{1}{\omega_{n_i}^2}\left(1 - e^{-\omega_{n_i} t}\right) - \frac{1}{\omega_{n_i}} t e^{-\omega_{n_i} t}\right]\bar{y}_i \omega_{n_i}^2 \qquad (4.54)$$

Differentiating the above expression results in $\dot{y}_i$ as

$$\dot{y}_i(t) = \bar{y}_i\left(t e^{-\omega_{n_i} t}\right)\omega_{n_i}^2 \qquad (4.55)$$

From the above expression, the maximum velocity can be found to occur at time t^*. Setting the maximum velocity to the slew rate, ω_{n_i} and t^* can calculated from the following expressions:

$$t^* = \frac{1}{\omega_{n_i}} \qquad (4.56)$$

$$\omega_{n_i} = \frac{\dot{\bar{y}}_i}{\bar{y}_i}.e \qquad (4.57)$$

Hence the frequency of the reference model is a function of the command displacement for a given slew rate. Note that robotic moves are typically preprogrammed, with $\bar{y}_i$ known. A reference natural frequency selected based on the

above procedure is expected to cause the servovalve to barely enter into saturation. Overestimating the reference natural frequency can cause the servo voltage to remain saturated for a certain length of time. Previous tests allowing the servo voltage to saturate did not pose control problems. Therefore, the model reference natural frequency was typically overestimated for large moves to take advantage of the flow capability of the hydraulics.

5 SIMULATION RESULTS

This section describes the results of a simulation study [2] designed to compare and contrast the effectiveness of different adaptive control algorithms as a prelude to experimental testing. An adaptive control simulation program was developed that allowed the numerical integration of a continuous-time nonlinear system model under the control of a multiple-input multiple-output sampled-data controller as shown in Figure 5.1. A fourth order Runge-Kutta with fixed step size was used to generate the continuous time dynamics. The integration step-size and sampling interval were selectable with integration step-size an order of magnitude smaller than the sampling period. Several different estimation routines, namely, least-squares, least-squares with covariance resetting, least-squares with covariance forgetting, and least- squares with covariance addition were developed so that one has a choice of subroutines for recursive identification. The adaptive control laws could also be incorporated through a separate subroutine. Features such as plotting, printing of the data on the screen, and storing of the data in a file were included. The programs used in the simulation are listed in Appendices 9.3 and 9.4.

At first, the recursive least-squares identification routines were tested separately using input-output sequences generated from a known DARMA model. The parameters converged to the values of the DARMA model indicating the correct-

ness of the identification routines. Simple discrete-time control laws with known coefficients such as the model reference controller were then tested. Finally, the identification algorithm was combined with on-line control to generate the coefficients of the controller, which led to the simulation of adaptive control of the robot.

5.1 Single Degree of Freedom Simulation Results

At first, the proposed adaptive control algorithms were applied to a single-axis model of the robot. Versions of the least-squares algorithm with or without covariance modification were combined with weighted one-step-ahead, model reference, and pole assignment adaptive control in order to compare and contrast their relative performances. The model of the system consisting of the linkage and actuator dynamics developed in Chapter Three was used to simulate the dynamics of the robot.

Figure 5.2 shows the position response of the vertical axis of the robot to a 10-in., square wave command when used with a model reference adaptive controller and covariance resetting parameter identification routine. The reference model chosen was a second- order critically damped system with a natural frequency of 3 rad/sec. The sampling rate of the control and identification algorithm was 50 Hz. In this simulation, the reference model natural frequency was chosen for a command displacement of 10-in., corresponding to displacement encountered during the first half cycle of motion. Notice that a desirable response was seen, even though the controller input was driven into saturation. This study also indicates that driving the control voltage into saturation for extended period of time does not have any

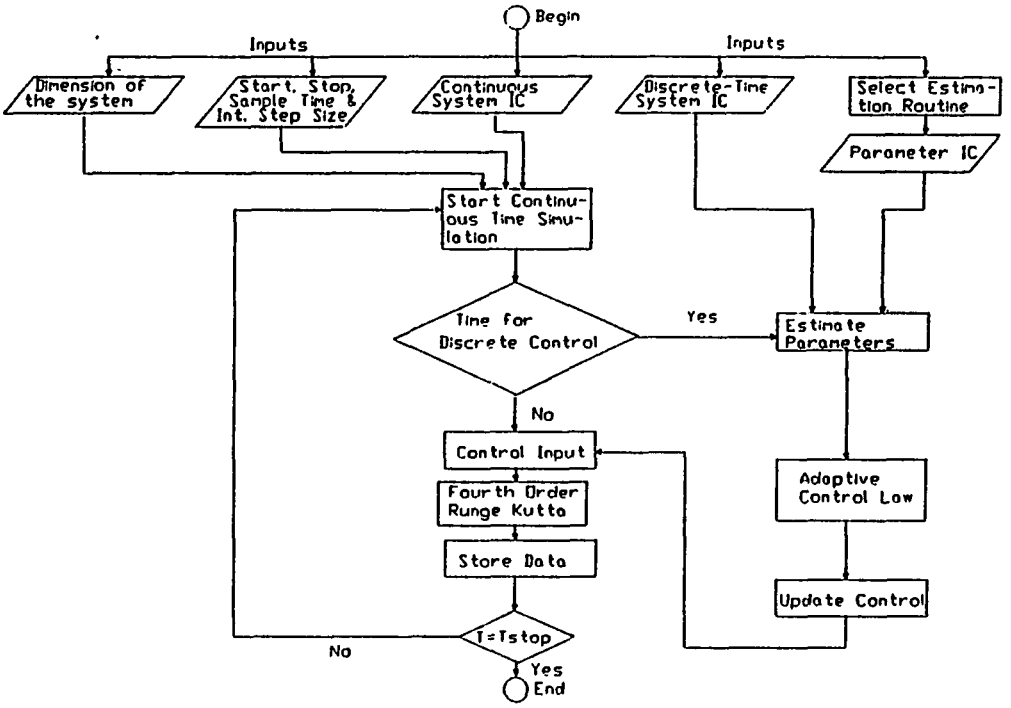

Figure 5.1: Schematic of the Simulation Program

detrimental effect on the position response.

Similar results were obtained for both the weighted one-step-ahead and the pole assignment algorithms. A critically damped system with a natural frequency of ω_n = 3 rad/sec was used for the reference model of a 10-in. move. The lower frequency poles for the pole assignment algorithm were chosen to correspond with those of the reference model. Higher frequency poles were assigned to well damped values (eg. z=0.5). The weighting parameter for the weighted one-step-ahead approach was found by a root locus analysis on the unit circle. The value of the weighting parameter λ was obtained by equating $\hat{\beta}_0/\lambda$ to the gain. The numerical value of the weighting factor was 0.0066. Figure 5.3 shows the comparison of the position response of the vertical axis to a 10-in.,square wave command when used with model

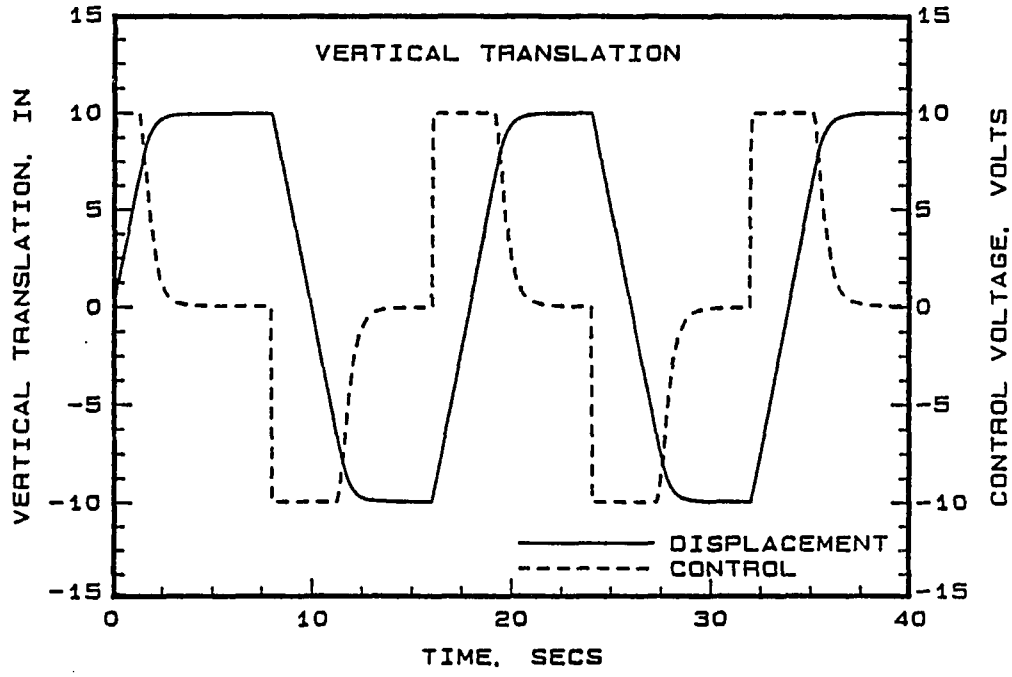

Figure 5.2: Position Response of MRAC System

reference and weighted one-step-ahead adaptive controller (WOSAAC). It can be seen that a well damped response was obtained in both the cases. The weighting factor was chosen to approximately match the critically damped roots of the MRAC. Further, the small difference in the responses is due to the one-step-ahead nature of WOSAAC which tries to match the desired value at each step as opposed to the MRAC which tries to match the desired value through the prespecified reference model.

The change in cylinder pressure produces a force on the mass that causes displacement of an axis. The deviation from nominal piston pressures should be minimized, since this indicative of smoother axis motion. A comparison of piston pressures under identical displacements was carried out for MRAC, pole assignment

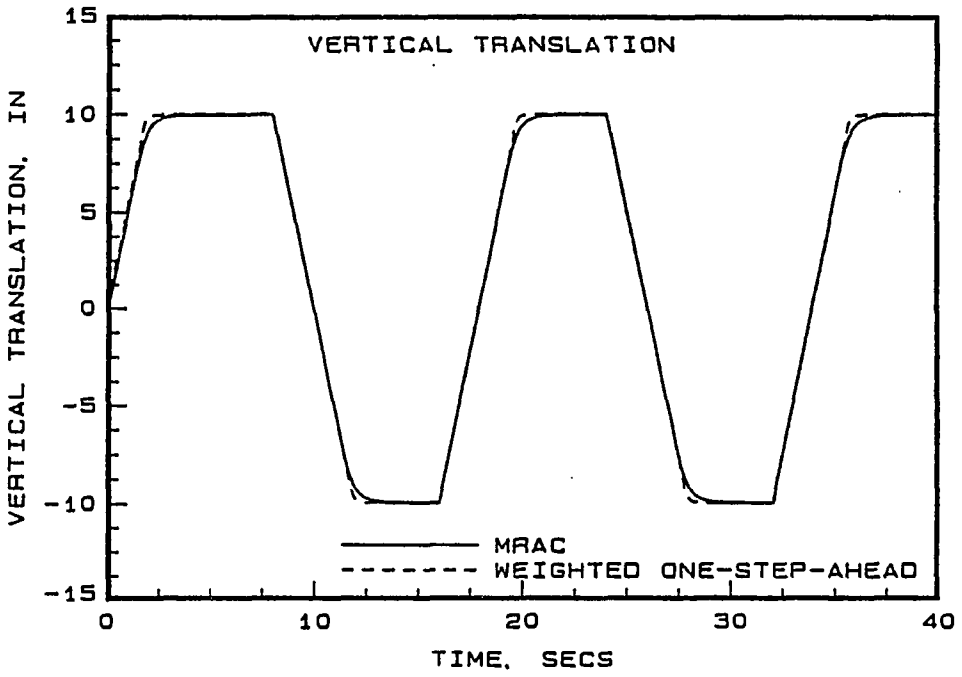

Figure 5.3: Comparison of Position Responses of MRAC and WOSAAC System

and one-step-ahead adaptive controllers. It was observed that the model reference controller provided the lowest fluctuation of chamber pressure from the nominal values.

5.1.1 Comparison of Control and Identification

The MRAC offered an intuitive advantage in the design process because the determination of a desirable reference model was fairly obvious. Determining the weighting coefficient for the one-step-ahead algorithm or the pole location of the higher bandwidth poles of the pole assignment algorithm proved to add additional complexity to the design. In addition, the pole assignment algorithm included a matrix inversion (equation 4.26) to calculate the pole and zero estimates at each

sampling instant that added to the computational burden of the algorithm.

The basic least-squares proved effective if a good initial guess of parameters was available. However, if a bad initial guess of the parameters was combined with a low initial covariance, the parameter estimates would converge to incorrect values. Covariance modification schemes such as resetting were shown to help in such cases.

When parameter identification based on a linear model of the robot was performed on the linearized plant dynamics, the parameters converged to the linear model coefficients. Figure 5.4 shows the parameter estimates of the leading denominator cofficients $\hat{a}_1$ and $\hat{a}_2$ for a model reference adaptive controller using a covariance resetting least squares identification algorithm for a 10-in., vertical displacement move. It can be seen that the parameters converged to the linearized values within 0.75 seconds and varied very little after this initial transient period, even with frequent reinitialization of the covariance matrix with covariance resetting algorithm.

Servovalve saturation proved to be the most noticeable nonlinear characteristic, occuring at +/-10V and representing full spool travel on the servovalves. This condition placed an inherent upper limit on the speed of travel of an axis. In order to avoid saturation voltages for a considerable length of time, the choice of reference model was based on the slew rate of an axis and the displacement command as discussed in Chapter Four. This method allowed the selection of natural frequency of the reference model so as to limit the controller saturation during the motion.

Figure 5.5 shows that the reference model with 3 rad/sec natural frequency designed to avoid saturation for a large displacement (10 in.), move provides a slug-

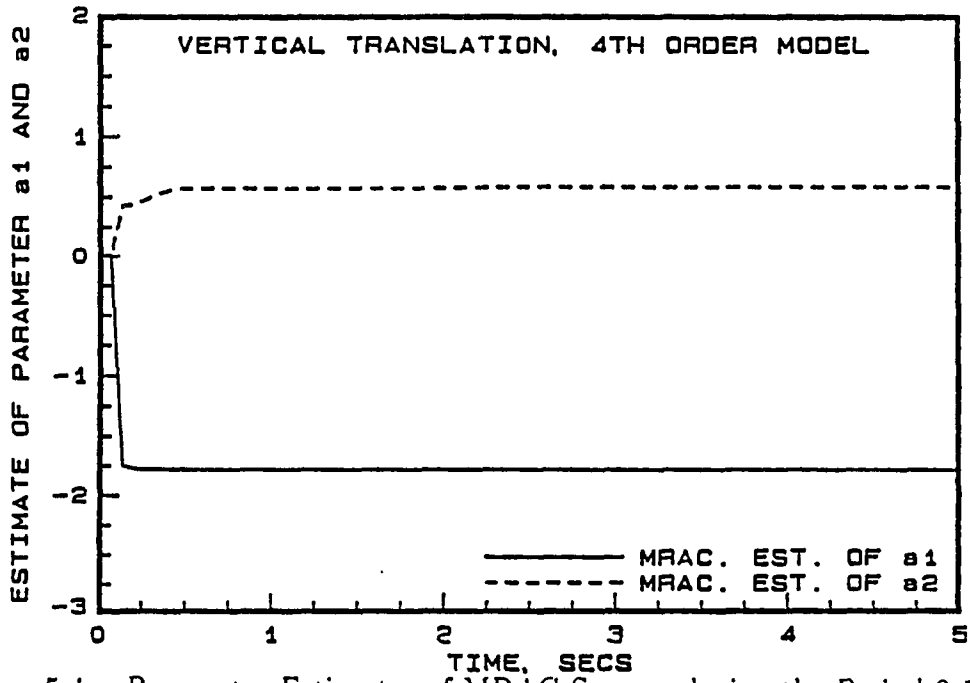

Figure 5.4: Parameter Estimates of MRAC System during the Period 0-5 sec

response for a 1 in move. The 30 rad/sec reference model is more appropriate for the 1-in. move, as indicated in the figure.

The slew rate was determined by the maximum flow capacity of the flow control valve. Basing the maximum velocity on the servovalve flow rate overestimates the slew rate actually encountered because the pressure drop across the servovalve is not used in defining the flow limitation. One way of looking at controller saturation is that it gurantees that the controller is taking advantage of the maximum power capability available. The simulation studies indicated that the controller was able to work quite well with the levels of saturation selected by this method. So, this displacement dependent approach used to select the reference natural frequency model was retained.

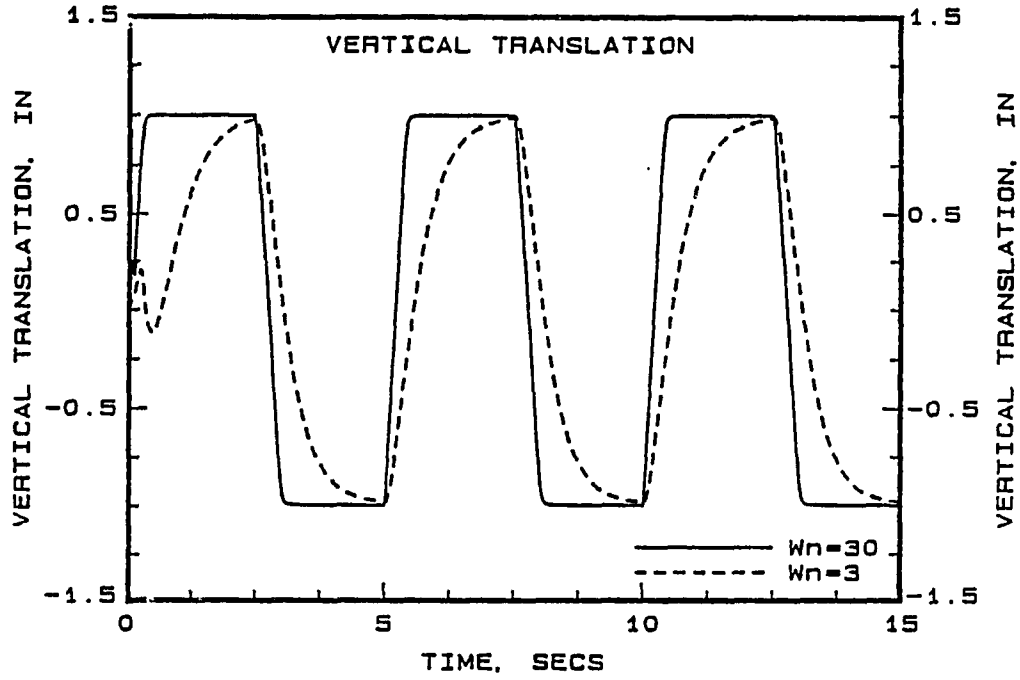

Figure 5.5: Position Responses of MRAC using Different Ref. Models

5.2 Multiple Degree of Freedom Results

From the results of single-axis simulation, it appears that the model reference controller, using a displacement-dependent reference model along with least-squares parameter identification algorithm, can be used for the adaptive control of the entire robot.

The general input-output representation of a multiple-input multiple-output DARMA model given in equation (4.2) can be used to represent the four-input four-output robot dynamics. The input-output relation in the bakward shift operator q^{-1} for the base rotational axis of the robot is given by the following equation:

$$(1 - a_{1_{11}}q^{-1} + a_{1_{12}}q^{-2} - a_{1_{13}}q^{-3})y_1(t) \quad +$$

$$
\begin{aligned}
&(a_{1_{21}}q^{-1} + a_{1_{22}}q^{-2} + a_{1_{23}}q^{-3})y_2(t) \quad + \\
&(a_{1_{31}}q^{-1} + a_{1_{32}}q^{-2} + a_{1_{33}}q^{-3})y_3(t) \quad + \\
&(a_{1_{41}}q^{-1} + a_{1_{42}}q^{-2} + a_{1_{43}}q^{-3})y_4(t) \quad = \\
&(b_{1_{11}}q^{-1} + b_{1_{12}}q^{-2} + b_{1_{13}}q^{-3})u_1(t)
\end{aligned} \tag{5.1}
$$

Similar relations can be written for the vertical translation, horizontal translation, and wrist rotation. The above equation, when combined with the backward shift operator notation for the rest of the axes, yields the following matrix representation for the input-output relation of the robot.

$$[I + A_1q^{-1} + A_2q^{-2} + A_3q^{-3}]Y(t) = [B_1q^{-1} + B_2q^{-2} + B_3q^{-3}]U(t) \tag{5.2}$$

where A_1, A_2,...., are all (4 x 4) matrices of parameters. The above equation can be rewritten in the form:

$$
\begin{aligned}
Y(t) \;=\; & -A_1Y(t-1) - A_2Y(t-2) - A_3Y(t-3) + B_1U(t-1) \\
& +B_2U(t-2) + B_3U(t-3)
\end{aligned} \tag{5.3}
$$

The controller form of the above equation is

$$
\begin{aligned}
U(t) \;=\; & B_1^{-1}[Y(t+1)^* - A_1Y(t) + A_2Y(t-1) + A_3Y(t-2) \\
& -B_2U(t-1) - B_3U(t-2)]
\end{aligned} \tag{5.4}
$$

where $Y(t+1)^*$ is the desired output of the system.

The reference model for each axis of the robot is given by the following equation:

$$\ddot{y}_i + 2\zeta\omega_{n_i}\dot{y}_i + \omega_{n_i}^2 y_i = \omega_{n_i}^2 \bar{y}_i \tag{5.5}$$

$$i = 1, .., 4$$

where ζ was assumed to be unity, corresponding to critically damped behavior and $\omega_{n_i}^2$ was selectable.

The coefficients used in defining the dynamics of the robot are listed in the Appendix 9.3. The base rotational axis and wrist are actuated by rotary vane type actutators while the vertical and horizontal axes are actuated by linear pistons. The vertical and horizontal axes were given a 10-in. square wave displacement command and the base and wrist axes were given a 1 radian square wave. command. Figures 5.6 - 5.9 indicate the simultaneous response of all four axes of the robot when model reference adaptive control is used. The adaptive control system response is fast and has no overshoot. The control voltage is driven into saturation for a short interval of time, taking advantage of the slew rate capability of the servodrives. The multiple axis simulation results indicate that it is possible to implement MRAC for the entire four axes of the robot.

In the most general case, the discrete-time linear dynamic model has sixty unknown coefficients, which makes for an extremely complicated identification and control problem. However, by taking advantage of the relatively decoupled nature of the dynamics, the identification algorithm reduced the number of coefficients to be identified to fifty-two. Even with this reduction, 3968 floating point multiplications were required per sampling period for both control and identification. This would amount to 40 msecs of computation at each sampling instant. Note that the identification algorithms for each axis are independent and can be computed in parallel.

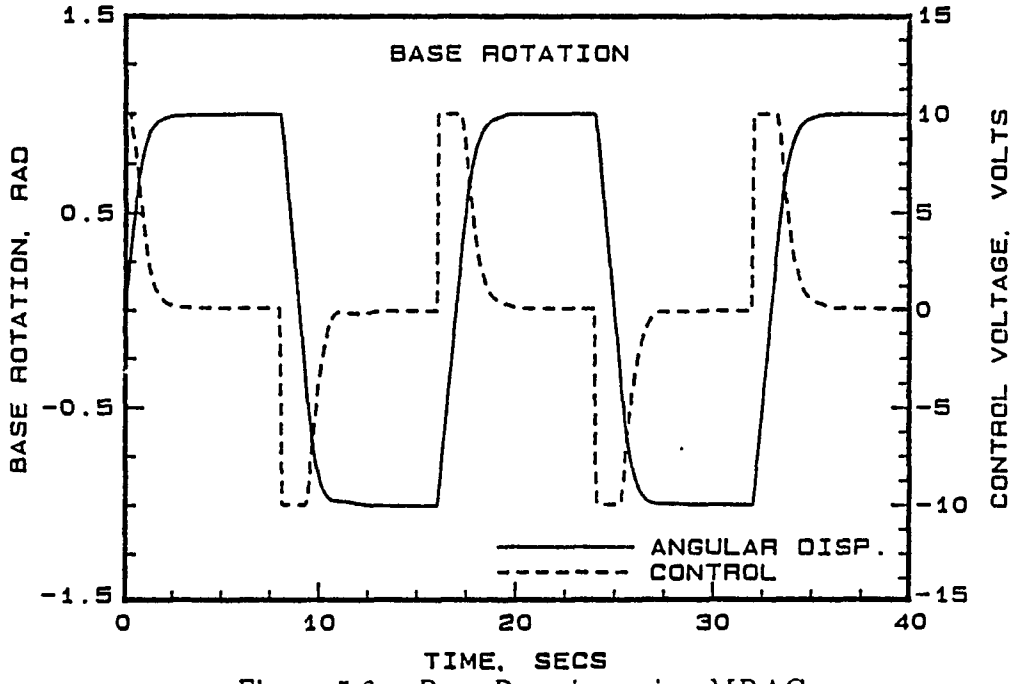

Figure 5.6: Base Rotation using MRAC

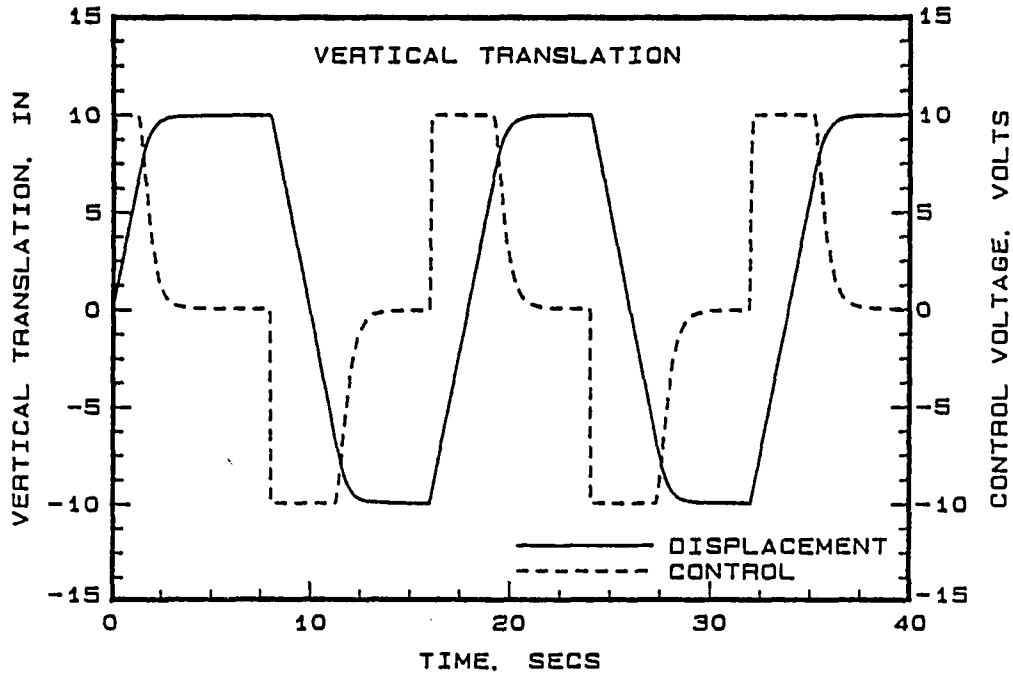

Figure 5.7: Vertical Translation using MRAC

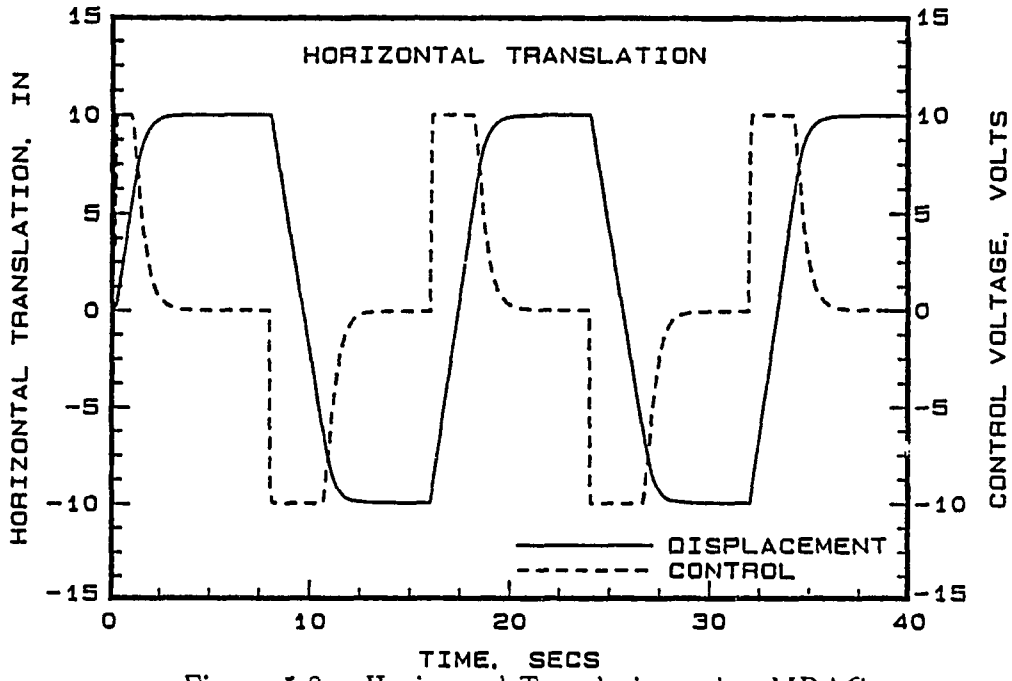

Figure 5.8: Horizontal Translation using MRAC

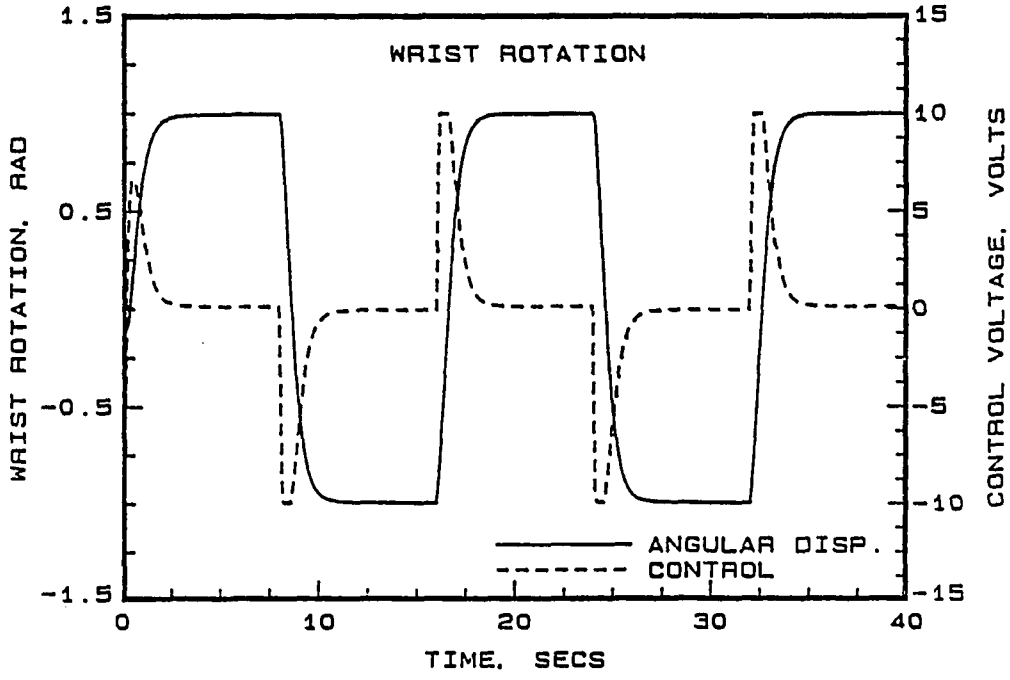

Figure 5.9: Wrist Rotation using MRAC

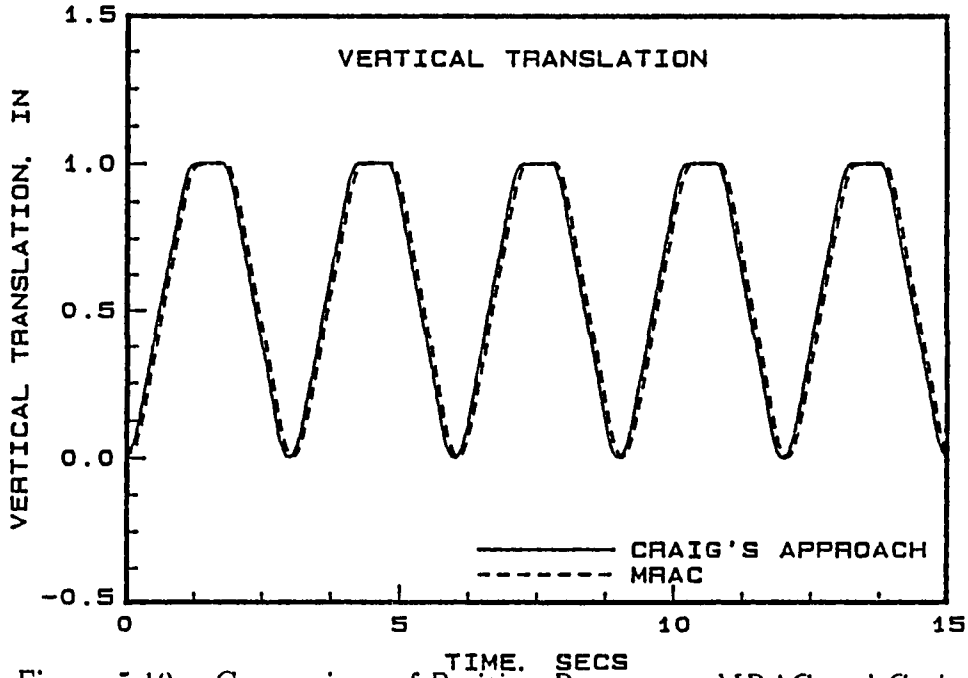

Figure 5.10: Comparison of Position Responses, MRAC and Craig

5.3 Comparison with Craig's Approach

In this section, the adaptive control algorithm based on continuous- time modelling developed by Craig is compared with the discrete time MRAC. Figure 5.10 shows the comparison of displacement response between discrete-time MRAC and Criag's approach. The details of the computer program used in simulating Craig's approach is given in Appendix 9.4 .

In this comparison, the command input trajectory used was an s-shaped curve which consists of a ramp-constant-ramp velocity profile. The reason for using this profile is because the Craig's approach requires the first two derivatives of command displacement to define the desired trajectories. Craig's approach uses a zero delay model while the discrete-time MRAC allows unit delay in the real-time digital con-

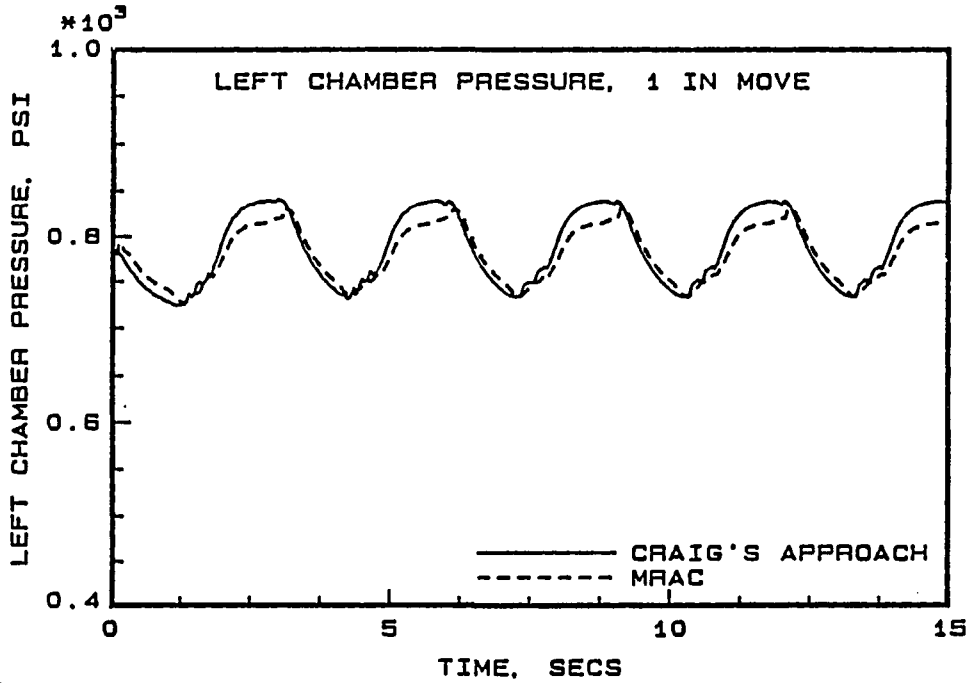

Figure 5.11: Comparison of Left Chamber Pressure, Craig and MRAC

troller for computation. The proportional and velocity gains, namely, K_v and K_p, of Craig's method were chosen from a reference model used in the discrete time MRAC. The initial guesses on the discrete-time parameters were obtained from the continuous- time initial guesses by Z-transform techniques. It can be seen from the figure that the two responses are comparable despite the discretization and delay in the discrete-time MRAC.

Figure 5.11 shows the comparison of left chamber pressure between the discrete-time MRAC and Craig's approach. The plot shows that the deviation from the nominal pressure of 776 psi is about the same for both the methods.

One of the interesting features of Craig's method is that the parameters estimated by the adaptive law are physical parameters of the robot such as the mass,

inertia, viscous friction coefficient, and coulomb friction coefficient, and so on. So, if we have reasonable initial guesses on these parameters, we can directly use them in the identification scheme, as opposed to finding the discrete time equivalent of the parameters. Figure 5.12 shows the parameter estimate of the mass of the robot. Figure 5.13 shows the rapid convergence of transient behavior of the mass estimate from 0 to 3 seconds. It can be seen that the mass estimate which was initially assumed to be 0.5 $lb - s^2/in$ converges to its actual value of 1.05 $lb - s^2/in$ after an initial transient.

Figures 5.14 and 5.15 show the estimate of viscous and coulombic friction coefficients whose initial values were assumed to be 60 lb-s/in and 25 lb respectively. It can be seen that the two friction coefficients also converged to their actual values as indicated in the figure. The input required for the Craig's approach is force or torque and so the actuator dynamics were linearized to derive a relationship between input servo-voltage and force.

In designing an adaptive controller using Craig's approach, we need to solve the dynamic equations of motion with estimated parameters to generate the computed torque to drive the robot. The estimated parameters are obtained by solving the parameter update differential equation. Whereas, in discrete-time MRAC, the parameter estimates are directly used in the control law. This is a significant advantage of DARMA model based controllers compared to Craig's approach.

5.4 Effect of Order Reduction of the System

Adaptive control algorithms used in this research are based on a discrete-time input-output model of the system described using a DARMA model (equation 4.2). Since adaptive controllers of this form require an online identification to be combined with online control, every effort should be made to reduce the computational burden on the real-time control computer. One way of doing this is to study the effect of reducing the total number of parameters used for identification and control on the response of the system.

Simulation tests were conducted to examine the effect of reducing the discrete-time model for identification and control from fourth order to third order. Figures 5.16 and 5.17 show a comparison of the position and control response of a fourth and third order model used for identification and control of the vertical axis of the robot. The results show that there is no noticeable difference between the position and control response of the two controllers. In fact, adaptive controller based on the first order model of the plant showed no significant changes in the position response as compared with the third order model. This indicated that the quadratic lag term contributed to complex root pair in the far left half of the s-plane. The higher order roots are in the far left of the s-plane and do not contribute much to the dynamics of the system.

The number of floating point multiplications for the single-axis fourth order model was 216. For a third order model, this reduces to 126.

5.5 Effect of Measurement Discretization

The position of the robot has to be converted to digital values to be read by the digital control computer. This is done with a 12 bit resolver-to-digital converter (R/D converter). This analog-to-digital conversion process is accurate to within 10 to 11 bits, resulting in a resolver resolution of $360/2^{10}$ or $360/2^{11}$ degrees. The corresponding linear axis resolution is 0.0024 or 0.0012 inches. The resulting position error might not only affect the response, but also the identification and control algorithm. Hence, a simulation study was proposed in which the continuous-time system generates the analog equivalent of the position signal, while the position input to the identification and control algorithm is rounded off to 10 bit resolution. A fourth order discrete-time model was used for identification and control.

Figures 5.18 and 5.19 show the result of this study in terms of the comparison of the position and control responses. Though the 10 bit resolution caused slightly higher steady state errors, the performance of the position adaptive controller was still acceptable. The control response exhibited small fluctuations around the zero voltage level and the steady state position errors were around 0.0036 inches as opposed to around 0.001 inches when noise was not introduced. This study helped us in determining that a 12 bit resolver-to- digital converter is more than adequate to measure the position of the robot.

5.6 Summary of Simulation Results

A simulation study was performed to evaluate the performance of direct and indirect discrete-time adaptive control algorithms for the control of a four axis hy-

draulic robot as a prelude to experimental testing. Weighted one-step-ahead, model reference, and pole assignment adaptive control algorithms were investigated for single axis control of the robot. While all the three algorithms provided satisfactory results when designed to provide critical damping, the model reference controller when coupled with least- squares-based identification algorithm was chosen for application to the four-axis simulation because of its ease in design and lower pressure fluctuations in the hydraulics.

A linear model of the hydraulic components was found to be adequate. Parameter estimates based on a linear model assumption tended to converge fast, and not vary, after the initial transient period. The linear parameter model used for the linkage dynamics worked very well. The identification routine was only active during a motion command and used the saturated voltages of the controller in the estimation of the parameters. The nonlinear effects were minimal.

The primary nonlinearity encountered was the saturation of the servovalve voltage, which placed an upper limit on the axis velocity. Saturation was properly taken into account in the design of the controller by using saturated values for identification and control. In the MRAC case, saturation was dealt with by using the displacement of the move to determine the natural frequency of the reference model. This allowed the axis to limit the saturation. Increasing this frequency by a constant factor allowed the axis to run in a saturated condition with a stable, well behaved response.

Although the MRAC is computationally more efficient than the pole assignment adaptive control method, the computational overhead of MRAC is still high,

Table 5.1: Number of Floating Point Multiplications for Single Axis

Model Order	MRAC	Pole Assignment
First Order	18	28
Second Order	60	96
Third Order	126	204
Fourth Order	216	352

primarily because of the number of terms that need to be identified in the coupled dynamics. Every effort should be made to find the smallest order system which will provide a compromise between position response and computational load on the control computer. A comparison of the number of floating multiplications required for different order models used for MRAC and pole assignment is shown in Table 5.1 .

A 12 bit resolver to digital converter seems to be adequate for experimental purposes since a 10 bit resolver to digital converter did not significantly affect the position and control response of the system.

A comparison study done between the proposed adaptive control algorithm and Craig's continuous time based approach to adaptive control showed that the proposed discrete-time adaptive controller performed in a comparable manner despite discretization and delay incorporated in our model for realistic real time implementation goals. Further, Craig's approach is more complicated than the algorithms used here, since it requires desired joint velocities and accelerations in addition to desired position and also has a component which requires the linkage dynamics at each control interval.

According to this simulation study, a model reference control algorithm, with

displacement-dependent reference natural frequency combined with least squares identification with or without covariance modification has been identified as providing an improvement over the existing control of a four-axis robot. However, it should be kept in mind that the MRAC approach requires discrete-time zero cancellation, and is not implementable if the system is non-minimum phase. In such a case, the pole assignment adaptive control will be used.

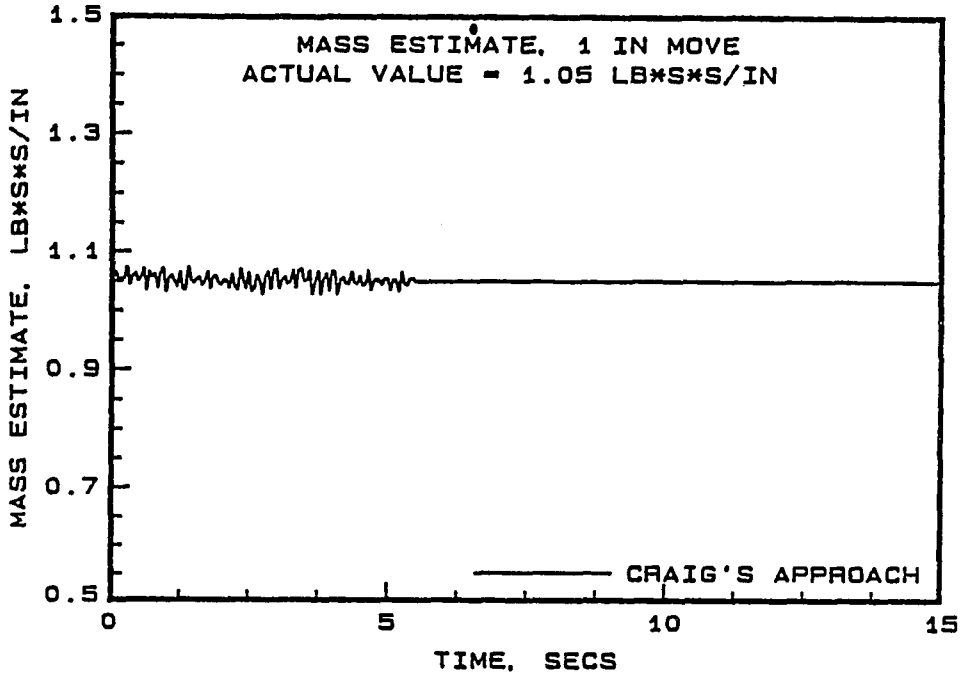

Figure 5.12: Mass Estimate using Craig's Approach

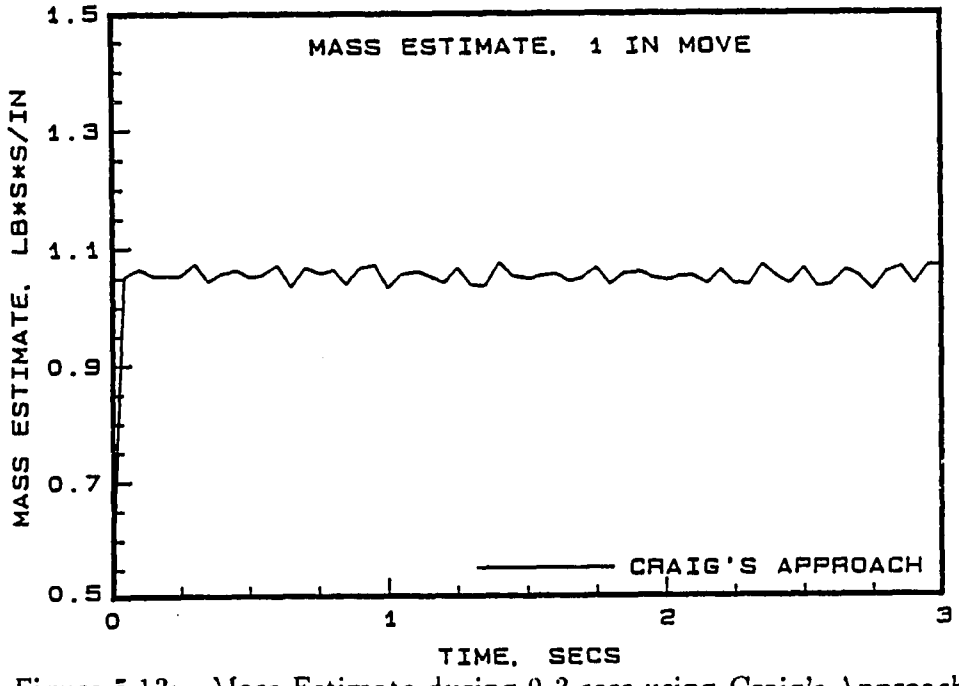

Figure 5.13: Mass Estimate during 0-3 secs using Craig's Approach

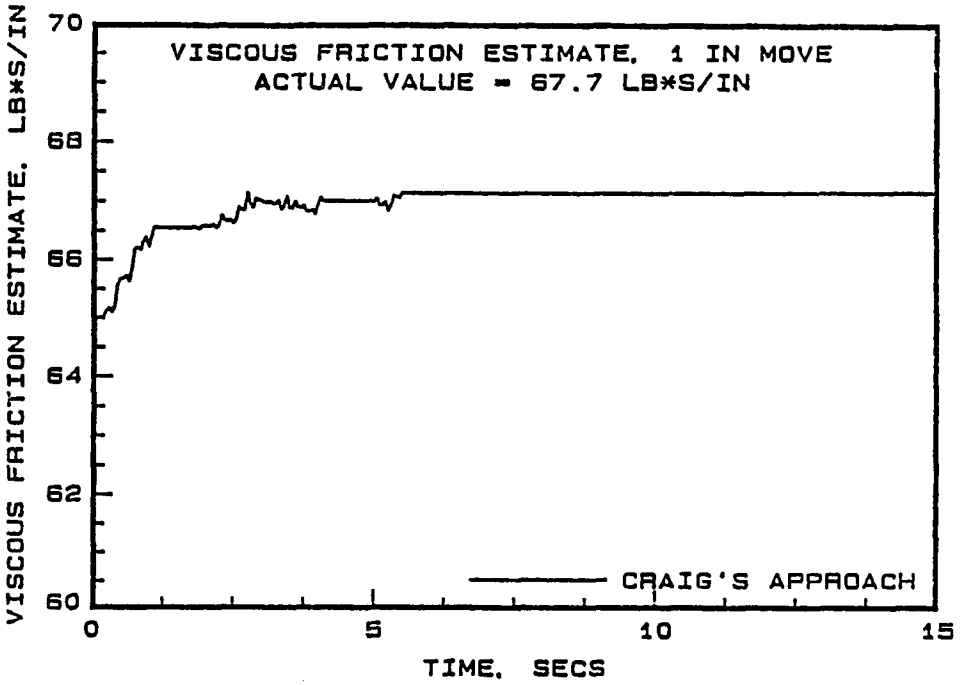

Figure 5.14: Estimate of Viscous Friction using Craig's Approach

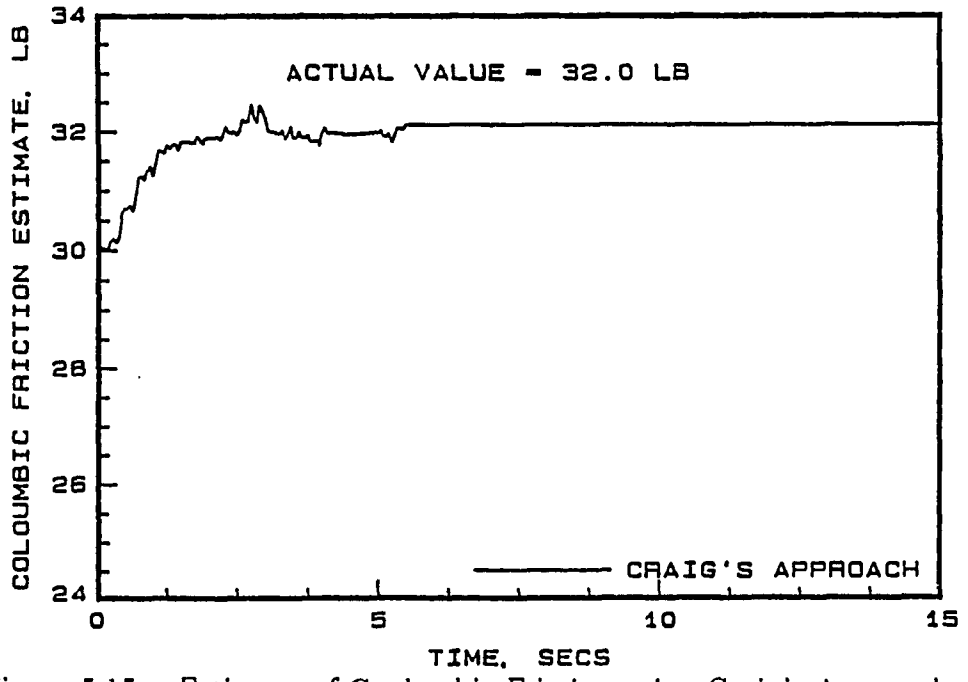

Figure 5.15: Estimate of Coulombic Friction using Craig's Approach

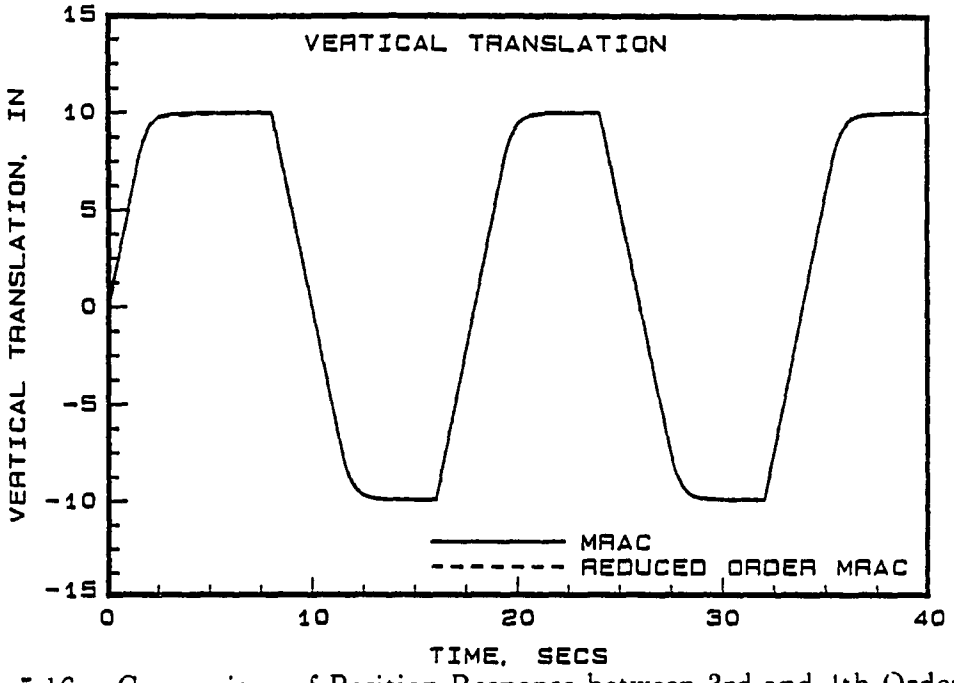

Figure 5.16: Comparison of Position Response between 3rd and 4th Order Model

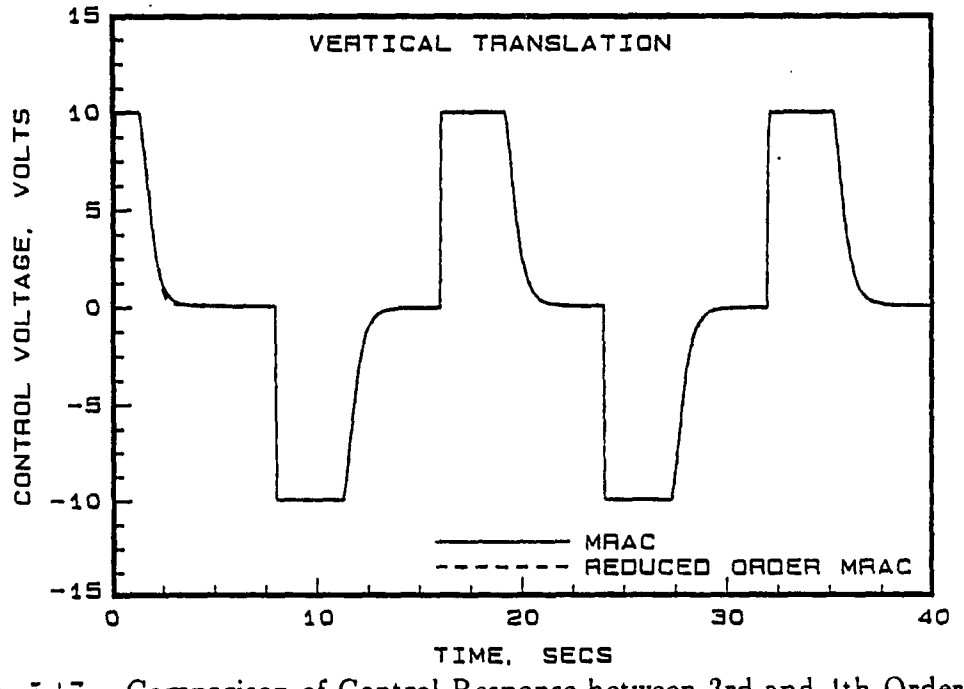

Figure 5.17: Comparison of Control Response between 3rd and 4th Order Model

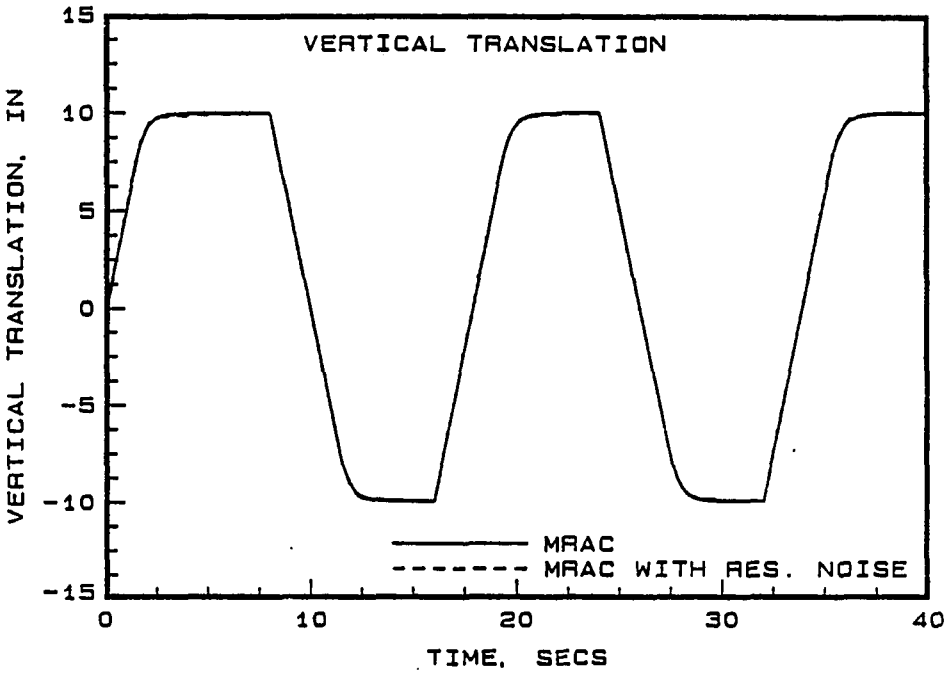

Figure 5.18: Comparison of Position with and without Resolver Noise

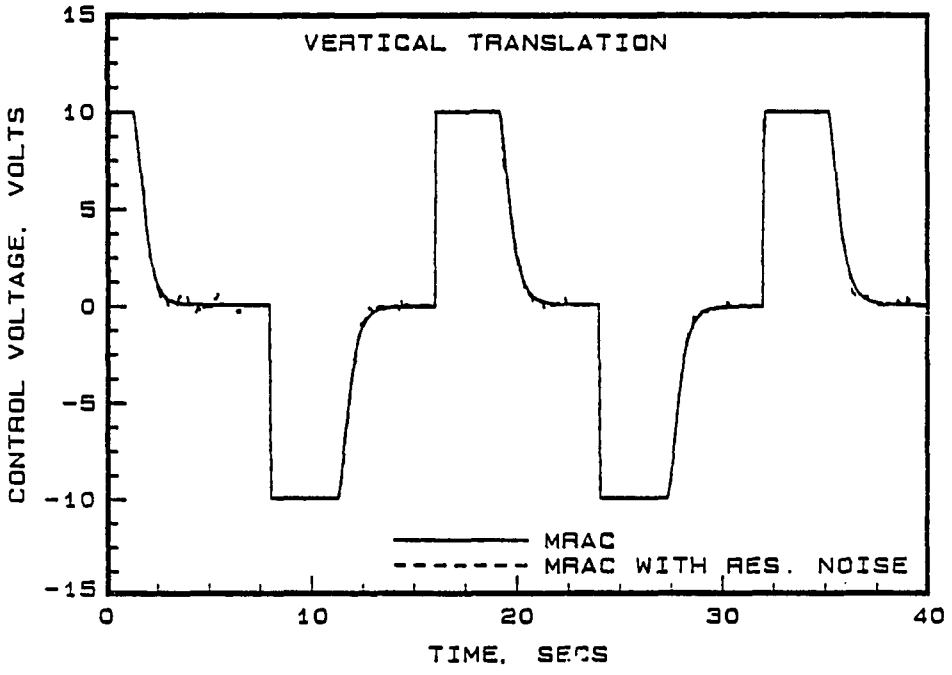

Figure 5.19: Comparison of Control with and without Resolver Noise

6 EXPERIMENTAL TESTING

6.1 Experimental Overview

This chapter deals with the implementation aspects of single adaptive controllers using the DARMA based approach described in Chapter Four.

The experimental setup consisted of the robot, hydraulic power unit, resolver-based position sensor, and the Masscomp control computer. The motion of each axis was controlled by an electrohydraulic servovalve which regulated the flow of hydraulic fluid into both sides of the actuator. Geared resolvers were mounted on each axis of the robot to provide the position feedback signal. The supply of hydraulic fluid, at a constant pressure, to the actuators was provided by the hydraulic power unit.

As a prelude to experimental control, identification testing was carried out. First, frequency response tests were performed to obtain the approximate continuous time model of the plant. Recursive least-squares identification was then used to determine the first-order discrete-time model of the axis. Comparisons were made between the discretized continuous time plant and the discrete-time model developed using recursive estimation.

The parameters of higher order models were identified for several sets of open-

loop input-output data to obtain the most representative second and third order discrete-time models. Least-squares estimation errors were compared between the models. This comparison provided a tradeoff between model complexity and the desired accuracy.

After modelling the discrete-time open loop plant, simple control configurations were developed. A proportional controller was first designed and successfully tested. Before implementing different adaptive control schemes, the model reference digital controller, whose reference model was based on slew rate, was experimentally tested. The resulting position response was well behaved as predicted by the reference model.

A model reference adaptive controller with a first order model for identification and control was first used for single axis testing. Initial testing of the control algorithms was done using an analog computer which simulated the dynamics of the robot. Subsequently, the model reference adaptive controller was used to control the robot. Studies were conducted to see the effect of choice of reference model, initial parameter estimates, size of the move, and covariance modification on the position and control response as well as parameter convergence.

The open-loop data at the sampling rate used for control (24 msec), indicated a minimum phase plant for first order as well as higher order models. However, the closed loop data identification indicated a nonminimum phase behavior for models higher than order one. This was due to a 12 msec delay between data sampling and analog voltage output from the control computer. Because of this latency, MRAC could not be implemented for higher order models, since this method requires the

discrete-time zeros to be minimum phase.

The pole assignment adaptive controller developed in Chapter Four assigns the closed loop poles to a specified polynomial $A^*(q^{-1})$ and does not require zero cancelling as does the MRAC. So, for adaptive control using higher order discrete-time models with nonminimum phase zeros, the pole assignment adaptive controller was implemented. The closed loop poles were assigned to a critically damped second order reference model whose natural frequency was selected based on slew rate and well damped poles.

At first, a closed loop pole assignment adaptive controller using a second-order DARMA model was applied to the horizontal axis of the robot. Subsequently, the pole assignment adaptive controller was implemented on each of the other three axes.

Since the pole assignment adaptive controller provided the best overall response in single axis testing, it was selected for closed loop control of the multiple axis. The multiple axis testing requires the control computer to properly sequence the inputs and outputs. Therefore, initial multiple axis testing was done using an analog computer. Then, the pole assignment adaptive controller was implemented on the vertical and horizontal axes of the robot. This was followed by simultaneous pole assignment adaptive control of the entire four axes of the robot.

6.2 Description of the System Hardware

Figure 6.1 shows the general configuration of the robot, the controller, and the resolver-to-digital converter (RDC). Motion control of the robot was achieved by

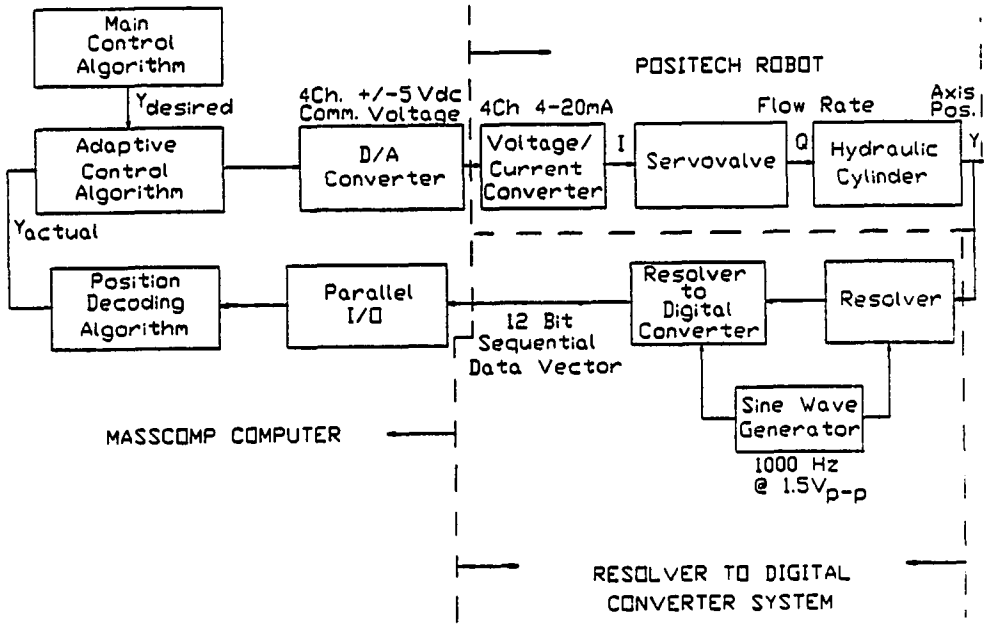

Figure 6.1: Configuration of the Controller, Robot, and R/D Converter

means of electrohydraulic servovalves which control the flow of hydraulic fluid into either side of the piston for linear motion or into either side of the vane for rotary motion. The hydraulic power unit (HPU) which supplies hydraulic fluid to each axis of the robot, was actuated by switches in a remote control panel. The panel consists of switches for starting the hydraulic pump and releasing the hydraulic fluid to the robot. A safety stop that shuts off the HPU in the event of an emergency was also included.

The controller used in this experimental work is an UNIX based real time data acquisition system built by Masscomp Inc. The computer consists of a high speed, 12-bit digital-to-analog converter used for sending analog control voltages to the four channel voltage-to-current converter. The voltage-to-current converter

in turn generates the current that drives the servovalves mounted on each axis of the robot. The position of the robot measured by the resolvers is received by the control program through the RDC and the Parallel I/O port.

The resolvers resemble small motors and are essentially rotary transformers designed so the coefficient of coupling between rotor and stator varies with shaft angle. When a rotor winding is excited with an ac reference signal, stator windings produce ac voltage outputs that vary in amplitude according to the sine and cosine of shaft position.

The stator signals from a resolver are routed to a specialized type of analog-to-digital converter system refered to as a resolver-to-digital (R/D) converter. The R/D converter used in this experiment is a multiplexed, four channel 160B100 series Computer Sciences Inc., module. Each resolver is connected to a separate input channel of the R/D converter. The module contains one reference processor that is shared by all the four resolvers. The outputs from each resolver are two voltages proportional to the sine and cosine of the input angle. These signals are sampled by the dual sample/hold circuits during the negative peaks of the reference wave form. The sampled dc outputs are multiplexed together to the central processor input. The multiplexer select lines determine which of these outputs will be processed.

After the conversion cycle is initiated by the start convert pulse (SC), the 12-bit successive approximation register is reset. The solid state control transformer (SSCT) performs the trigonometric computation $\theta - \phi \approx sin\theta cos o - cos\theta sin o$. This value is fed through a comparator which sets the parallel binary angle o contained in the register equal to the selected resolver angle θ.

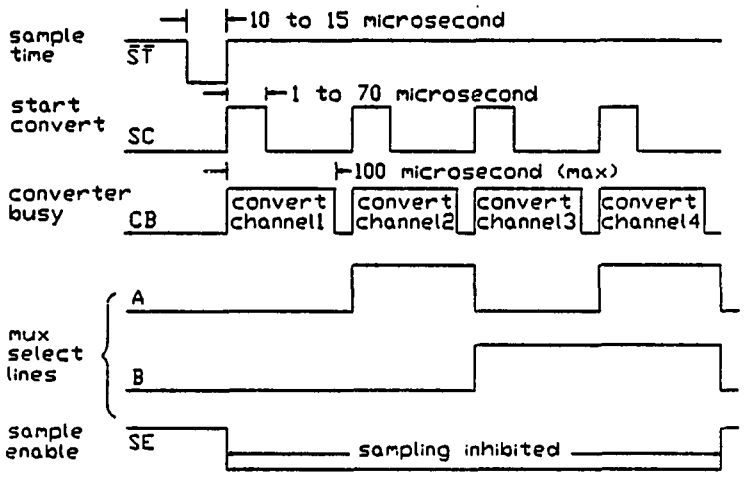

Figure 6.2: Timing Diagram for the R/D Converter

The digital position output from the RDC is transmitted to the Masscomp via a 16-bit Parallel I/O port. The digital position information is decoded and converted to engineering units in the control program.

In order to accomplish the above sequence of tasks at each sampling instant, proper timing and control of the R/D converter, and handshaking between R/D converter and Masscomp is required. Figure 6.2 shows the timing diagram of the control lines that was required for successful R/D conversion of all four resolvers.

The timing and control of the R/D converter requires the generation of start convert (SC) pulses 1 to 70 microsecond in width to produce the converter busy (CB) pulses of width less than 100 microseconds. In order to achieve this, a monostable multivibrator triggered by the sample time pulses generated by the R/D converter

was designed and built at Iowa State University. In addition, an interlock circuit was also fabricated to avoid the following two situations when the sample time can interfere with the conversion:

1) When the converter is busy (CB=1), the sample and hold (S/H) pulse must be inhibited by setting SE to logic 0 until the conversion is complete.

2) When the sampler is busy ($\bar{ST} = 0$), the SC pulse must be delayed until $\bar{ST}$ goes to logic 1.

Nand gates were used in the design of these interlock circuits. The rotor input to the resolvers at $1.5V_{p-p}$ and 1000 Hz was generated by an ISU built sine-wave oscillator. The reference input of $12V_{p-p}$ at 1000 Hz for the RDC was obtained by amplifying the rotor input. Figures 6.3 and 6.4 show the circuits that were designed and built at ISU for R/D conversion.

6.3 Real Time Programming

The programming was done in FORTRAN 77 using a real-time UNIX operating system in the Masscomp 5450 computer. Closed loop adaptive control of the robot consisted of a combination of software and hardware interface at each sampling instant.

The digital control loop performed the following tasks:

1) Position data were received by the parallel I/O port of the Masscomp using the RDC.

2) The position and control data were converted to physical units

3) Command input and the resulting position output of the robot were used to

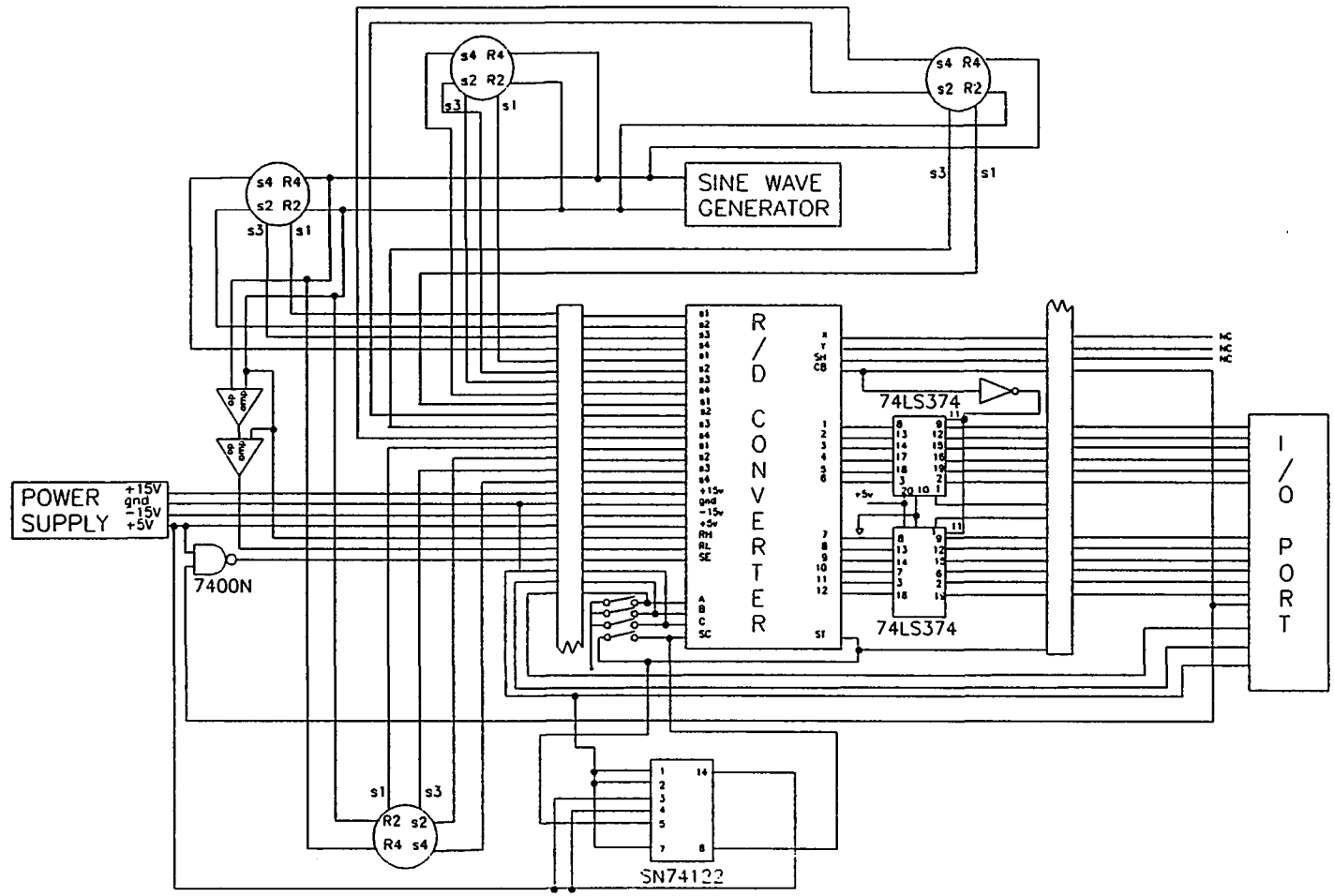

Figure 6.3: Circuit Diagram of the R/D Converter

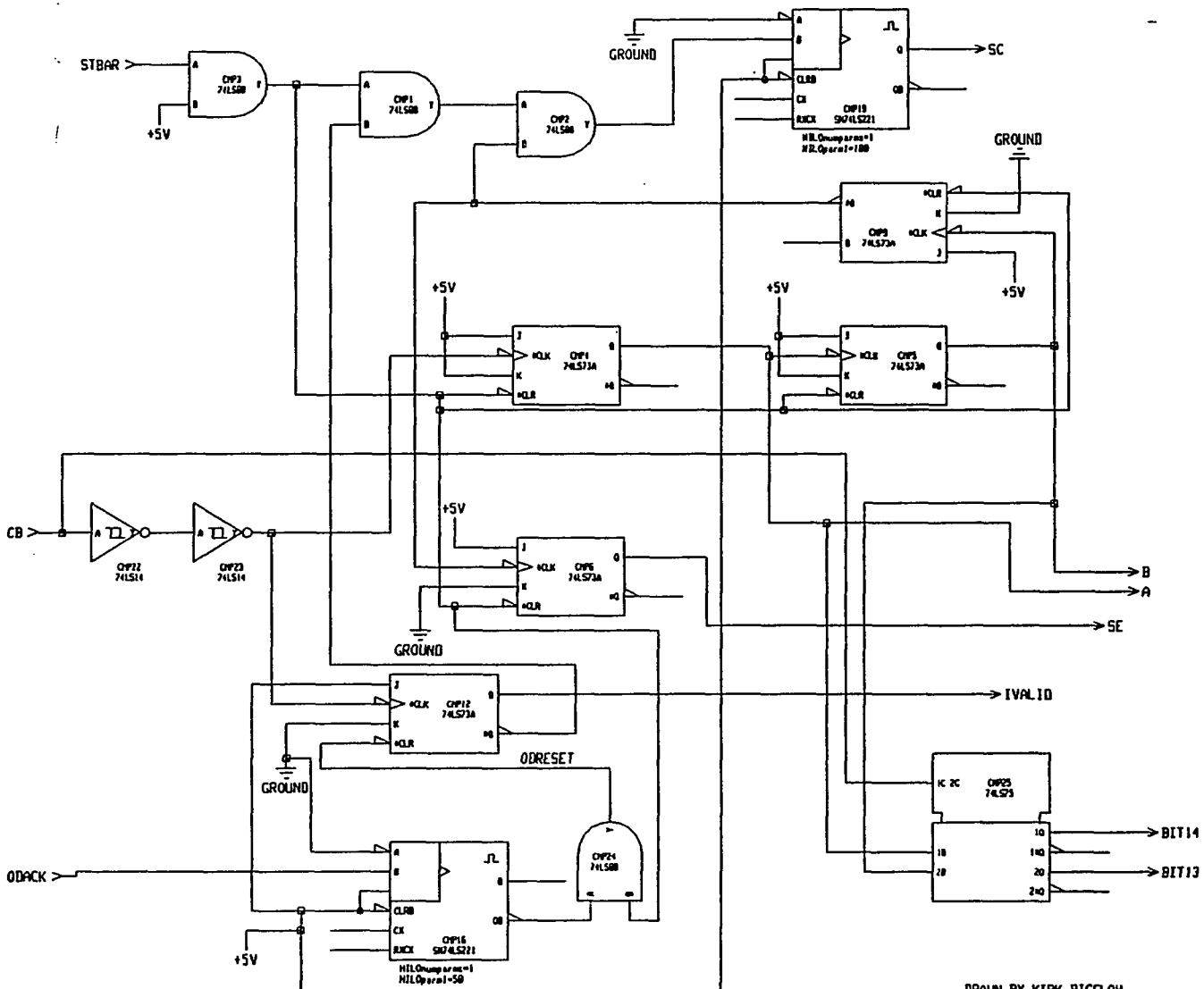

Figure 6.4: Additional Circuit of the R/D Converter for Four Axis Testing

identify the discrete-time model of the system.

4) The discrete-time model was used to derive a digital controller.

5) The digital controller was used to generate a control signal for the robot.

6) The generated control voltage was sent out through the digital-to-analog ports.

This process was repeated at each sampling instant. Model parameters were estimated using the least-squares algorithm with or without covariance modification. Controller synthesis was based on DARMA model of the system. The programming was made modular by using subroutines for identification, data conversion, data storage, matrix multiplication and so on.

The real time input-output data transfers are accomplished using the Data Acquisition and Control Processor (DACP). The DACP does not simply process one data part at a time, but rather stores the data items into a small holding area called a DACP buffer. It processes each item in the buffer, and then proceeds to fill and process more buffers of data until the transfer finishes. The buffer management scheme is designed to handle fast input transfers or fast output transfers as in a data acquisition setup. However, when input transfers (e.g., position information from RDC) are used in software (e.g., control algorithm) to then generate output transfers (e.g., voltage to the servovalve) as in a digital control setup, a control time delay is experienced. The effect of this time delay or latency is to alter the open loop discrete-time zeros of the plant as described in Chapter Three.

The data transfer latency was found by applying a triangular waveform into the analog-to-digital input port, and then returning that signal through a digital-

to-analog port. The resulting output was compared with the original input at various sampling instants. The output was delayed when compared with the input, indicating a 12 msec time delay.

6.4 Identification Testing

As a first step towards the implementation of digital controllers, open-loop identification tests were carried out. Open-loop frequency response tests, using sine wave inputs, were carried out for the horizontal axis of the robot. An approximate continuous-time transfer function of the open loop plant was obtained from the frequency response plots. The DARMA model which defines the discrete-time input-output relationship of the plant was obtained by recursive least-squares estimation of the open-loop data. The number of unknown coefficients of the autoregressive and moving average components were varied to fit different order discrete-time models for the plant. These models were then used in studying the effect of model complexity on the mean-square-error between the estimated output and the actual output.

6.4.1 Frequency Response

The open loop frequency response tests were conducted for the horizontal axis of the robot using an external function generator to provide the driving voltage for the servovalve. The command voltage and resulting position were sampled at a rate of 1000 Hz. The sinusoidal input from the function generator was varied from 0.1 Hz to 10 Hz to obtain the amplitude ratio and phase shift of the system. The upper

Table 6.1: Open Loop Magnitude and Phase at Different Frequencies

Frequency (r/sec)	Amp. Ratio (in/volt)	Phase (deg.)	S.D. of Phase (deg.)
1.40	0.8213	-93.14	1.71
2.64	0.4993	-93.10	3.59
3.82	0.3556	-90.09	3.86
5.08	0.2468	-87.21	4.16
10.27	0.1261	-88.40	4.08
12.84	0.0984	-87.10	2.97
25.13	0.0706	-93.52	2.63
43.71	0.0440	-88.52	3.71
66.88	0.0252	-94.52	3.83

limit of the exciting frequency was dictated by a pronounced structural resonant behavior of the robot at 10 Hz. The amplitude ratio and phase shift at different frequencies were calculated at several time intervals in order to obtain an estimate of their mean and standard deviation as shown in Table 6.1 .

Figures 6.5 and 6.6 show the comparison of the model used for simulation and experiment of the amplitude ratio and phase shift. It can be seen that the frequency response plot shows only the dominant root behavior, namely, the integrator, in the frequency range used for open loop testing. In the simulation plot, the roll off due to the second order term occured after 100 rad/sec, which could not be identified in experiment due to the structural resonant behavior around 67.28 rad/sec. The discrepancy of 6 db between the experimental and simulation amplitude plots was found to be due to smaller gain values used for the voltage-to-current converter used in the simulation model. The gain term in the simulation was then modified for comparing simulation and experimental position responses. The phase shift comparison between experiment and simulation indicated that the phase shift did

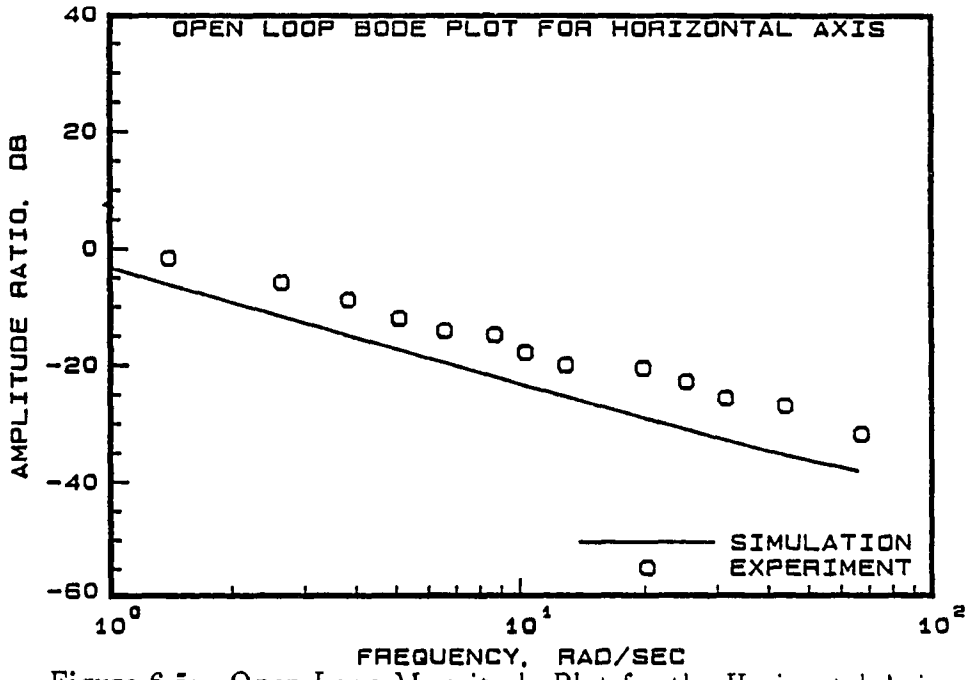

Figure 6.5: Open Loop Magnitude Plot for the Horizontal Axis

did not deviate much from -90 degrees in the frequency range used for testing.

6.4.2 Recursive Least Squares Estimation

The open-loop frequency response analysis indicated that the dominant root of the system is the integrator. This suggested that the discrete-time identification tests on the open loop data should be able to identify a pole at unity. Further, the gain of the system can be obtained from the discrete-time numerator polynomial, since the sampling rate is known. For the purpose of identification, the robot was excited with both sine and square wave input signals at frequencies in the range 0.2 to 0.5 Hz. Since servovalves can have a drift, the open-loop data was identified using least- squares estimation taking the drift into account. The resulting first-order identification results consists of three parameters, namely, the discrete-time

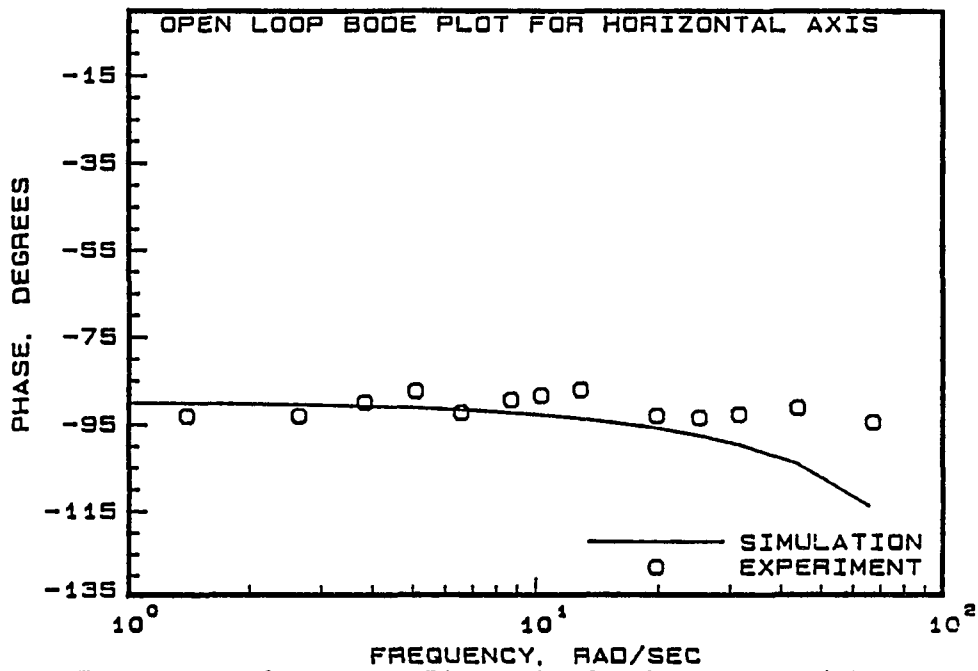

Figure 6.6: Open Loop Phase Plot for the Horizontal Axis

pole, the discrete-time gain, and the drift parameter as shown in the following equation.

$$y(t) - a\, y(t-1) = b\, u(t-1) + d \tag{6.1}$$

In all the identification tests using the first order model with drift, the pole at the unit circle was estimated accurately to within ± 0.0009. The numerator coefficient as estimated by the recursive least squares identification algorithm can be used to determine the gain of the plant by dividing it by the sampling rate used for open loop tests. This gain value was comparable with the gain found from the open loop Bode plots, as well as from the response to proportional control as shown in Table 6.2 .

It was also found that the drift term was more than an order of magnitude

Table 6.2: Comparison of Forward Path Gain Using Different Methods

Method	Forward Path Gain
Open Loop Freq. Res.	1.6312
Open Loop Estimation	1.5914
Cl. Loop Prop. Cont.	1.6000
Modified Simulation	1.6709

smaller than the smallest numerator coefficient, indicating that the drift can be neglected while identifying the discrete-time coefficients with input-output data.

6.4.3 Effect of Higher Order Terms

After successful identification tests on the first-order model, the identification tests were carried out using higher order models for the discrete-time system. It was found from the open-loop frequency tests that only the zero on the unit circle exhibited dominant behavior, while the higher frequency quadratic lag term was hard to identify due to the structural resonance of the robot. So as a next step, the higher order roots were estimated based on offline open-loop input-output data obtained when the robot was excited with a square and sine wave input at frequencies between 0.2 to 0.5 Hz. Results of the identification tests on several data sets showed an average behavior of the first, second and third order discrete-time models. Tables 6.3 through 6.6 show the identification results obtained from closed-loop control data as well as open-loop data sampled at the frequency of closed-loop control. Table 6.3 shows the values of the A and B coefficients when open-loop data sampled

Table 6.3: Parameter Estimates of Open Loop Data at 42 Hz

Model	A_1	A_2	A_3	B_1	B_2	B_3
First Order	-0.9994	–	–	0.0382	–	–
Second Order	-1.2844	0.2844	–	0.0439	-0.0190	–
Third Order	-1.1260	-0.1019	0.2885	0.0298	-0.0046	-0.0076

Table 6.4: Std. Dev. of Parameter Estimates of Open Loop Data at 42 Hz

Model	A_1	A_2	A_3	B_1	B_2	B_3
First Order	$\mp$9.3E-4	–	–	$\pm$5.28E-3	–	–
Second Order	$\mp$0.0862	$\pm$0.0858	–	$\pm$0.0139	$\mp$0.0142	–
Third Order	$\mp$0.0945	$\mp$0.1202	$\pm$0.2074	$\pm$0.0150	$\mp$0.0072	$\mp$0.0115

at 42 Hz. was used for identification. Table 6.4 shows the standard deviation of the estimates obtained in Table 6.3 over four sets of data. Similarly, Table 6.5 shows the value of A and B coefficients when closed-loop data with controller latency was used for identification. Table 6.6 shows the standard deviation of the estimates obtained in Table 6.5 over six sets of data.

Table 6.5: Parameter Estimates of Closed Loop Data at 42 Hz

Model	A_1	A_2	A_3	B_1	B_2	B_3
First Order	-1.000	–	–	0.0403	–	–
Second Order	-1.1509	0.1502	–	0.0108	0.0298	–
Third Order	-1.3185	0.1721	0.1457	0.0013	0.0452	-0.0240

Table 6.6: Std. Dev. of Parameter Estimates of Closed Loop Data at 42 Hz

Model	A_1	A_2	A_3	B_1	B_2	B_3
First Order	$\mp$1.3E-3	–	–	$\pm$9.7E-4	–	–
Second Order	$\mp$0.0237	$\pm$0.0234	–	$\pm$0.0039	$\pm$0.0016	–
Third Order	$\mp$0.0273	$\pm$0.0671	$\pm$0.0724	$\pm$8E-4	$\pm$2E-3	$\mp$5E-3

From this table it can be infered that:

i) higher order models have higher standard deviations in the parameter estimates when compared to the first order model.

ii) the percentage standard deviation from nominal value is higher for the B (control parameters). This can be due to the fact that the B parameters are about two orders of magnitude smaller than the coefficients of the discrete-time poles.

iii) the zeros of the open loop discrete-time data showed a minimum phase behavior while the zeros of the closed loop discrete-time data exhibited a nonminimum phase behavior. This is due to the sampling latency introduced by the hardware while performing closed-loop control.

iv) the higher order roots of the open-loop plant are in the far left of the s-plane, which confirms the difficulty in identifying them using frequency response tests.

In order to see the effect of the higher order terms, goodness-of-fit tests were conducted on several data sets using the models described in Table 6.7 for different order discrete-time models. The goodness of fit tests on nine different data sets showed that the sum-of-squared-errors for increasingly complex models was

Table 6.7: Mean Square Error for Different Order Models

Model	Mean Square Error
First Order	7.5E-5
Second Order	4.2E-5
Third Order	2.8E-5

smaller than the first order model. Parameter estimates converged rapidly after an initial transient period in all the discrete-time models developed using recursive least squares estimation.

Using the average open loop model, the discrete-time transfer function was developed as follows.

$$OLTF = \frac{0.0298(z + 0.59)(z - 0.43)}{(z - 0.9991)(z + 0.42)(z - 0.54)} \tag{6.2}$$

Comparing the above transfer function with the open loop transfer function in equation (3.32), it can be seen that the terms corresponding to the open-loop gain and pole at unity are estimated accurately. The higher order terms do not compare as well.

6.5 Single Axis Experimental Control

The experimental testing of the control algorithms was done initially with the horizontal axis of the robot. The controllers were implemented in increasing order of difficulty, ranging from proportional to pole assignment adaptive controller. In this section, the control law implementation results are discussed in detail.

6.5.1 Proportional and Model Reference Control

As preliminary tests , the proportional and model reference control laws were first implemented. The gain of the proportional controller was varied from 1 to 5 and the position response was recorded. The proportional controller with a gain of 1 exhibited a sluggish response, while a higher proportional gain of 5 leads to hydraulic shock due to large pressure fluctuations. Figure 6.7 shows the position response of the robot when a gain of 1 was used for the proportional controller. The jerk of the robot with higher proportional gain is due to rapid acceleration of the robot arm from the starting position when a step input is applied. A time constant of 0.625 secs was obtained experimentally when a proportinal gain of 1 was used which compared well with the estimated time constant of 0.598 secs.

The model reference controller was then implemented using the first order discrete-time model developed in the identification testing. The reference model for the model reference controller was selected such that the closed loop system exhibited a critically damped response. Figure 6.8 shows the position and control response of the model reference controller. It can be seen that the position response has a small overshoot. This is due to the fact that the controller had to be based on a system with no time delay to avoid nonminimum phase zero cancellation. This is further discussed in section (6.5.2). Since the position response to a step input is achieved while following the reference model, the model reference controller results in a smoother axis motion. So, from the stand point of compromise between axis speed and smooth response, the model reference controller was superior to proportional control.

6.5.2 Model Reference Adaptive Control

After successful implementation of nonadaptive controllers, adaptive controllers were implemented. Figure 6.9 shows the comparison of the horizontal axis position response under model reference adaptive control using different reference models. A first order plant model was used for adaptive control. It can be seen that the faster reference-model based adaptive controller converged to the desired value prior to the slower reference model-based adaptive controller. The faster reference model was chosen according to the reference model selection scheme which was derived previously in Chapter Four. This choice of reference model provides a fast response while driving the controller into saturation for a short interval of time. This combination leads to a fast response and smooth axis motion over wide range of displacements given the limitations on the system.

After selecting the reference model for different displacement moves, the model reference adaptive control law was then used to track a ± 5 inch square wave position command. Figures 6.10, 6.11, and 6.12 show the overall response to model reference adaptive controller, response during the initial transient period, and response during the negative half of the first cycle. As expected, under reasonable initial guess on parameters, -0.9 for the discrete-time pole, 0.03 for the control parameter, and 100I for the covariance, the position and control responses converge after an initial transient period of one cycle. As anticipated, the worst position control behavior occurs during the initial identification phase. After the first half cycle, it can be seen from the Figure 6.12 that the adaptive controller tries to match the reference input of -5-in. As can be seen in the Figure 6.10, the position response converges

to the desired value in the subsequent cycles.

The MRAC response showed an overshoot prior to settling down due to the numerator dynamics introduced by the time delay. This can be explained alternatively as follows. The MRAC requires discrete-time zero cancellation. The time delay in the system introduces a discrete-time zero on the unit circle as shown below:

$$y(t) = y(t-1) + K^{\cdot}\, u(t-1) + K^{\cdot}\, u(t-2) \tag{6.3}$$

Since we cannot cancel this zero, the control law had to be based on the following system equation with no time delay.

$$y(t) = y(t-1) + K^{\cdot}1\, u(t-1) \tag{6.4}$$

The above explanation was verified by computer simulation.

The model reference adaptive controller uses estimates of parameters based on recursive least-squares identification at each sampling instant. Therefore, the time variation of the parameter estimates and their convergence was studied. Figure 6.13 shows the estimate of the denominator coefficient $\hat{a}_1$ when a large move using MRAC was applied to the horizontal axis of the robot. This parameter corresponds to a pure integrator indicating a discrete pole at unity. The identified coefficient converged to this value, as expected. The small variations of the gain parameter $\hat{b}_1$ during each half cycle correspond to the gain changes caused by the unequal area across each side of the piston. Figure 6.14 shows the gain parameter along with the position response and command displacement when MRAC was applied for a 10-in. move. It can be seen that a large change in the parameter occurs during the initial transient period. Thereafter the changes in the gain coefficient were due to

different areas across the piston. The value of the coefficient converged close to the previously identified value.

Adaptive control systems based on DARMA models can be stable even if the parameters do not converge to their true values [21]. The least-squares algorithm has fast initial convergence, but the convergence rate decreases after some iterations. If reasonable initial guesses of parameters and covariance were used, convergence can be accomplished using the least- squares algorithm alone. However, when there is poor knowledge of parameters, one can attempt to achieve convergence by using a least-squares algorithm with covariance resetting.

A study was done to see the effect of covariance resetting on the performance of first order MRAC. Figure 6.15 shows the comparison of position responses for a one inch square wave command with and without covariance resetting when poor initial guesses of parameters and covariances were used. Notice the very large initial error. Figure 6.16 shows the comparison of the same position response during a 3 sec interval. It can be seen that for the case when the covariances were reset, the response was faster. This can be explained as follows:

The model reference adaptive controller tries to set the closed loop poles to the open loop zeros. Now let $B'(q^{-1})$ be the nominal or the true numerator or zero polynomial. If the actual value to which the poles converge is $B'_n(q^{-1})$, then it can be shown that the closed loop transfer function is

$$CLTF = \frac{B'(q^{-1})q^{-d}gH(q^{-1})r(t)}{B'_n(q^{-1})E(q^{-1})} \tag{6.5}$$

In the experimental results discussed above, the true numerator polynomial,

$B(q^{-1})$, is much smaller than $B'_n(q^{-1})$ when the covariances are not reset. This essentially reduces the gain of the system and results in a slower response.

Figure 6.17 shows the comparison of gain parameter estimates with and without covariance resetting. It can be seen that with covariance resetting, the estimated parameter tends to the true value, while in the case without covariance resetting, the control parameter did not converge to the expected value of around 0.040.

Since higher order identification tests lead to nonminimum phase zeros in the open loop case due to computational latency problems, it is not possible to implement a stable model reference adaptive controller for higher order discrete-time models. In order to show this, the controller was designed such that the initial few cycles would be under proportional control, during which time online identification will be carried out to identify the plant parameters. The controller would then be switched to model reference adaptive controller, which should lead to instability of the axis. The reason for this as indicated earlier is that the MRAC is based on setting the closed-loop poles to open- loop zeros. Figures 6.18 and 6.19 show the position response and estimate of parameters $\hat{b}1$ and $\hat{b}2$ under combined proportional and higher order MRAC. It can be seen that the position response exhibited unstable oscillatory behavior after switching to MRAC after 60 secs. The parameter estimates that follow show that the $\hat{b}1$ coefficient is less than the $\hat{b}2$ coefficient, indicating a nonminimum phase plant behavior.

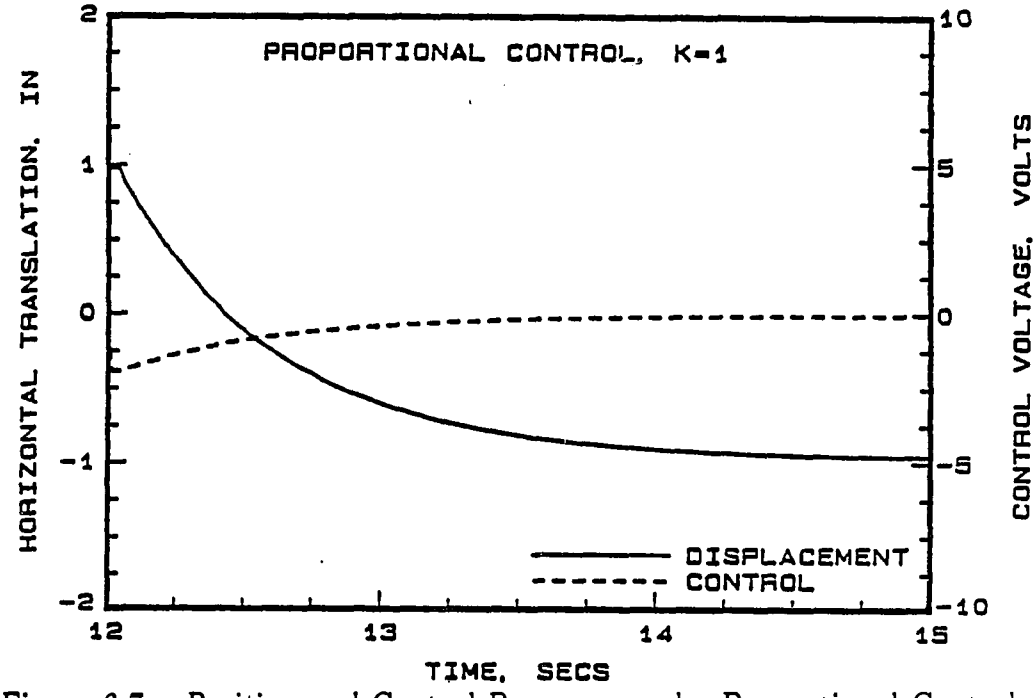

Figure 6.7: Position and Control Response under Proportional Control

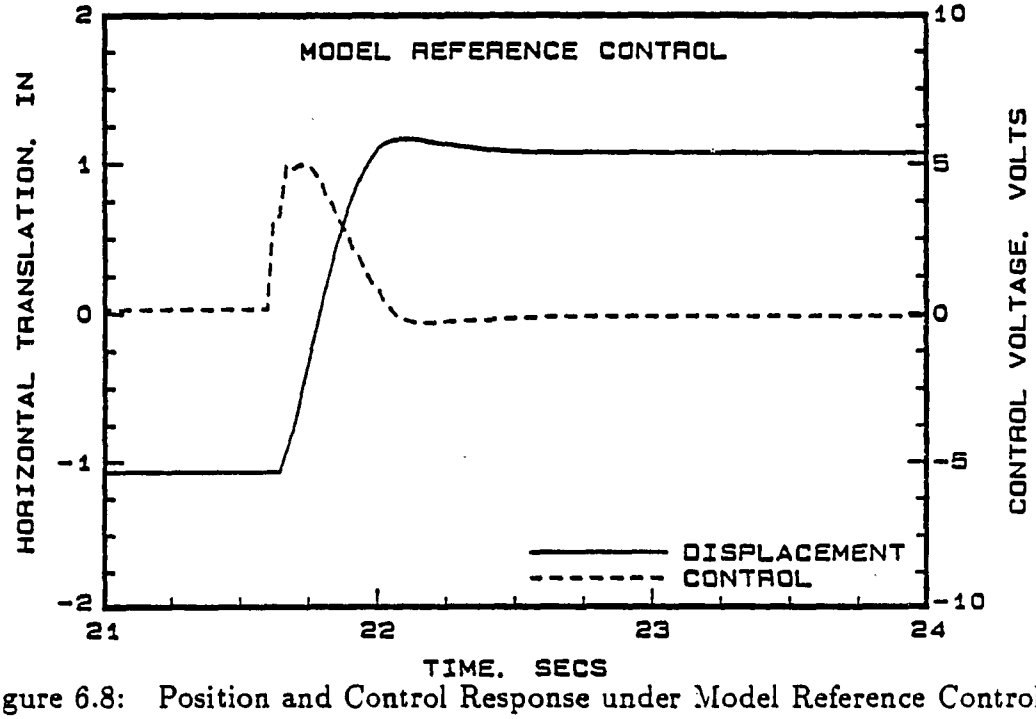

Figure 6.8: Position and Control Response under Model Reference Control

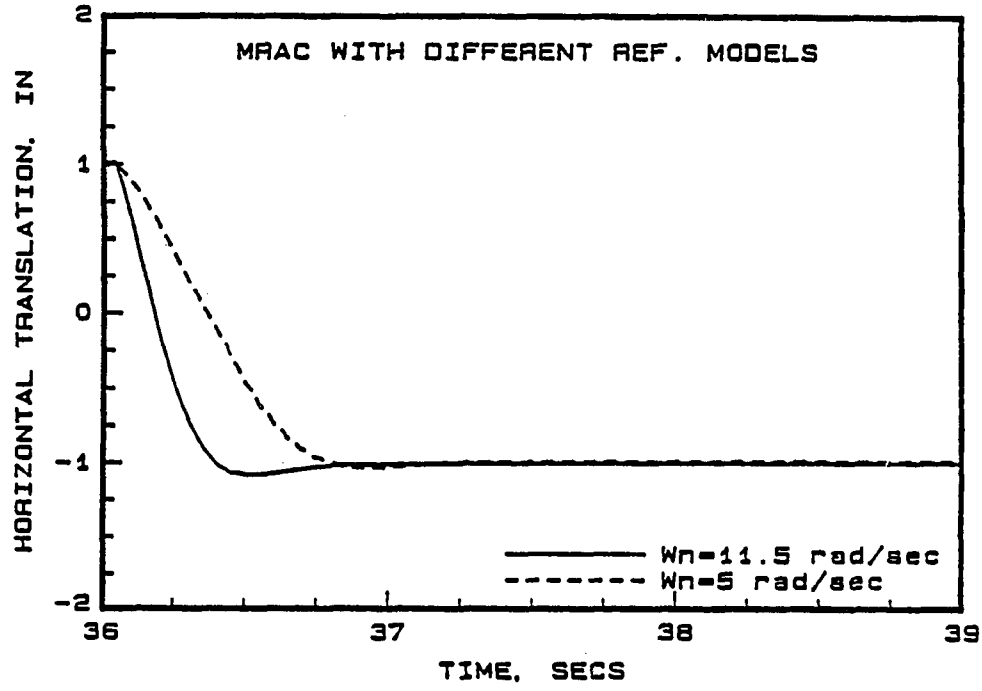

Figure 6.9: Position Response using MRAC with Different Ref. Models

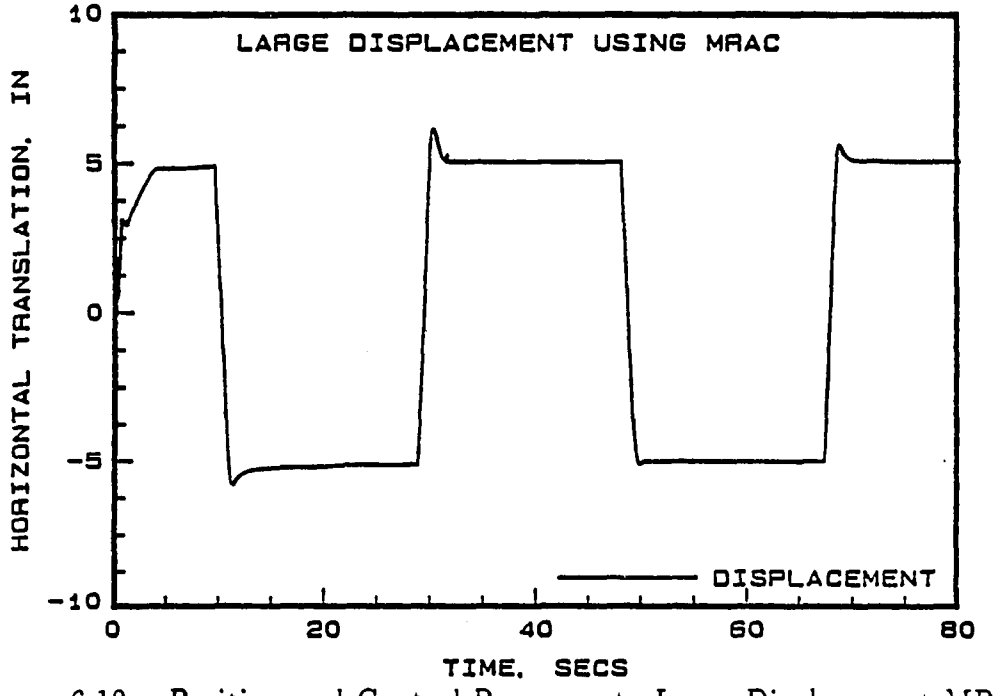

Figure 6.10: Position and Control Response to Large Displacement MRAC

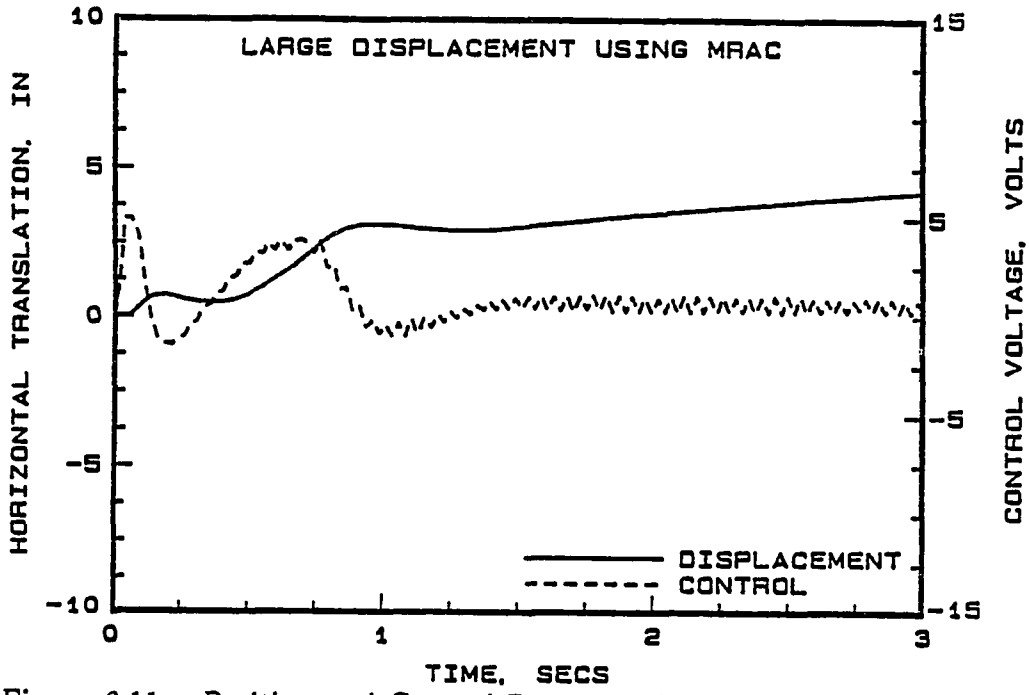

Figure 6.11: Position and Control Response during the Period 0 to 3 secs

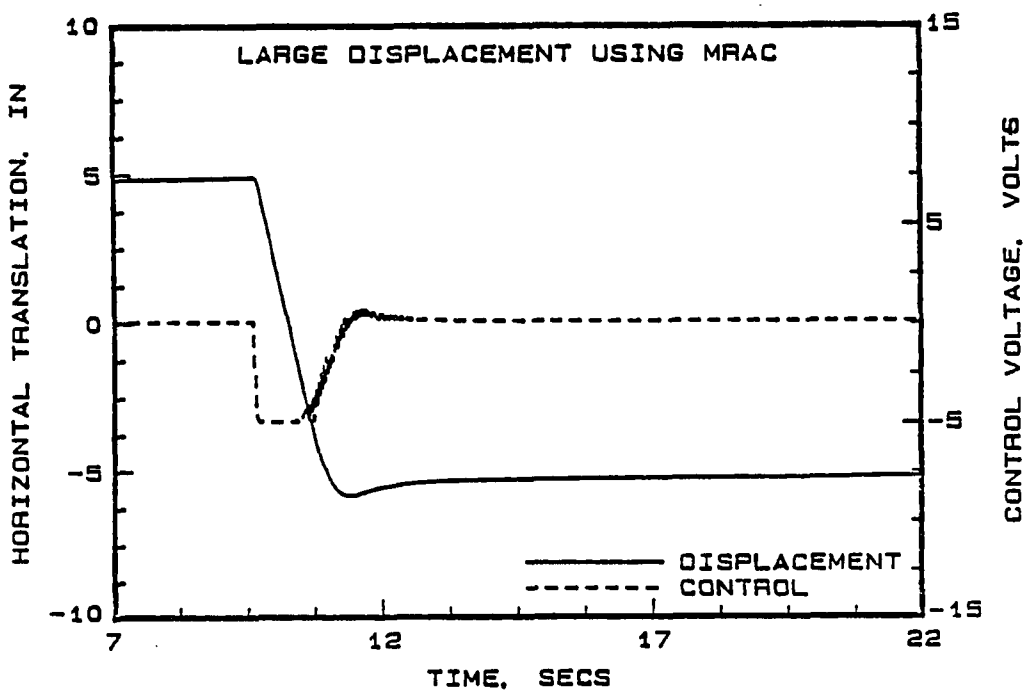

Figure 6.12: Position and Control Response during the Period 7 to 22 secs

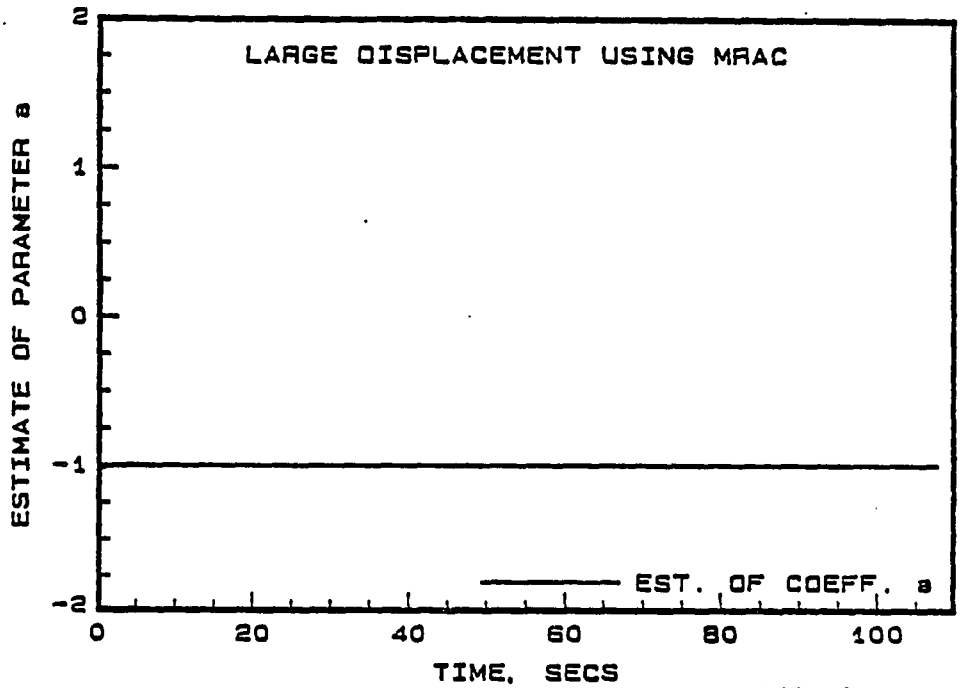

Figure 6.13: Estimate of Parameter $\hat{a}_1$ using MRAC

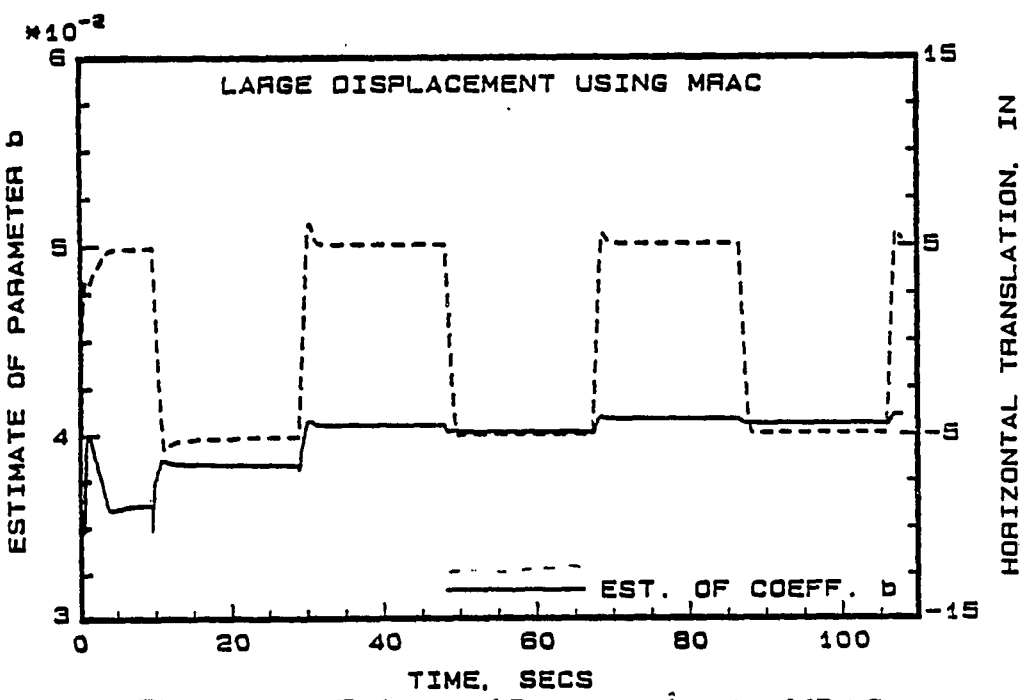

Figure 6.14: Estimate of Parameter $\hat{b}_1$ using MRAC

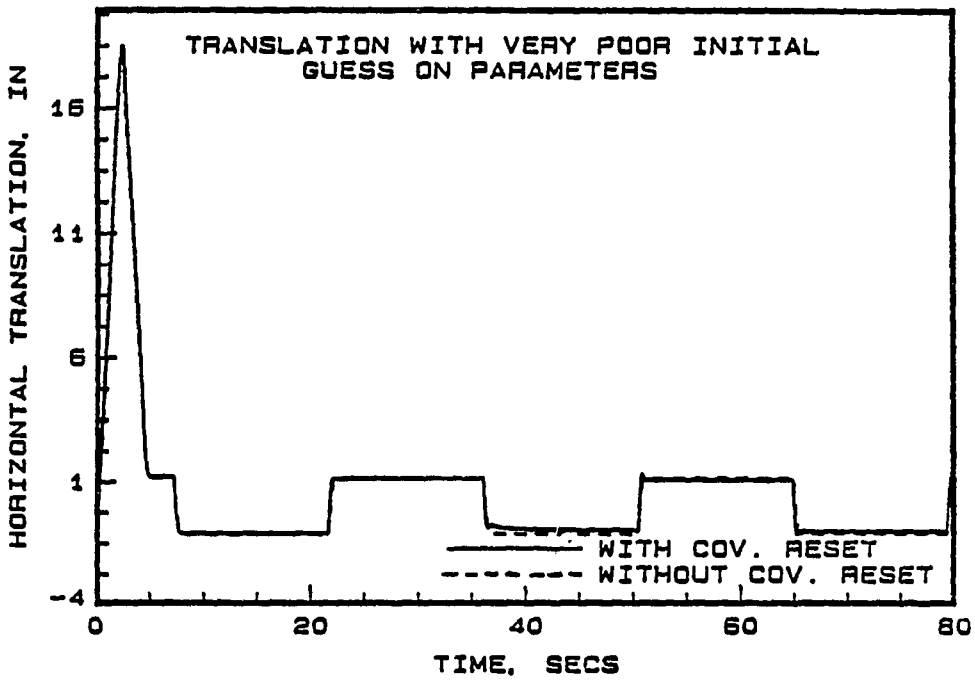

Figure 6.15: Effect of Cov. Reset on MRAC Position Response

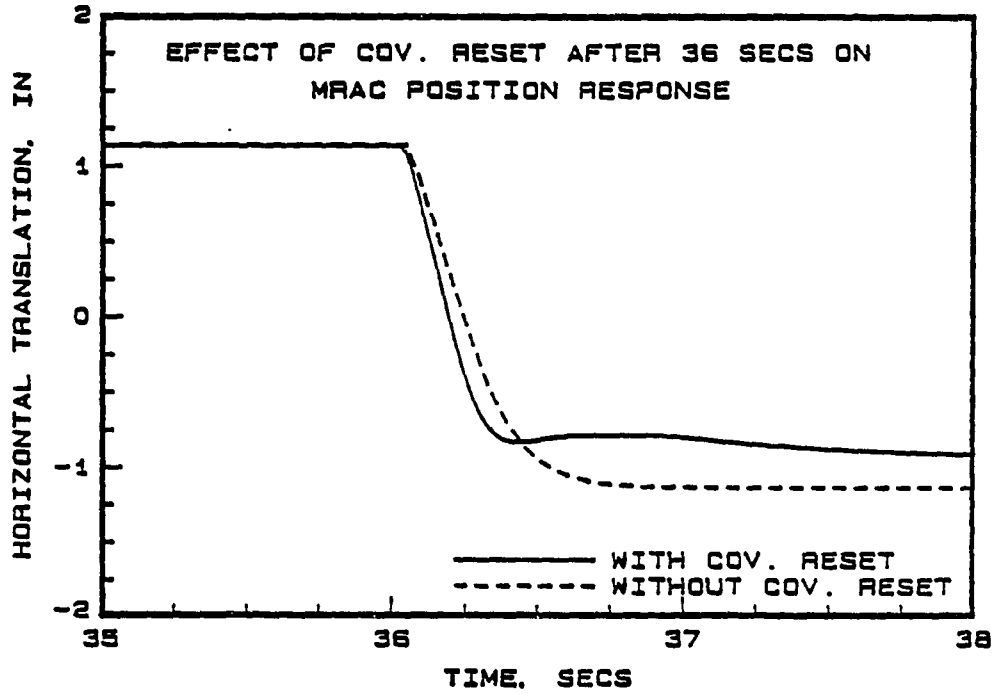

Figure 6.16: Effect of Cov. Reset on MRAC 3 sec duration

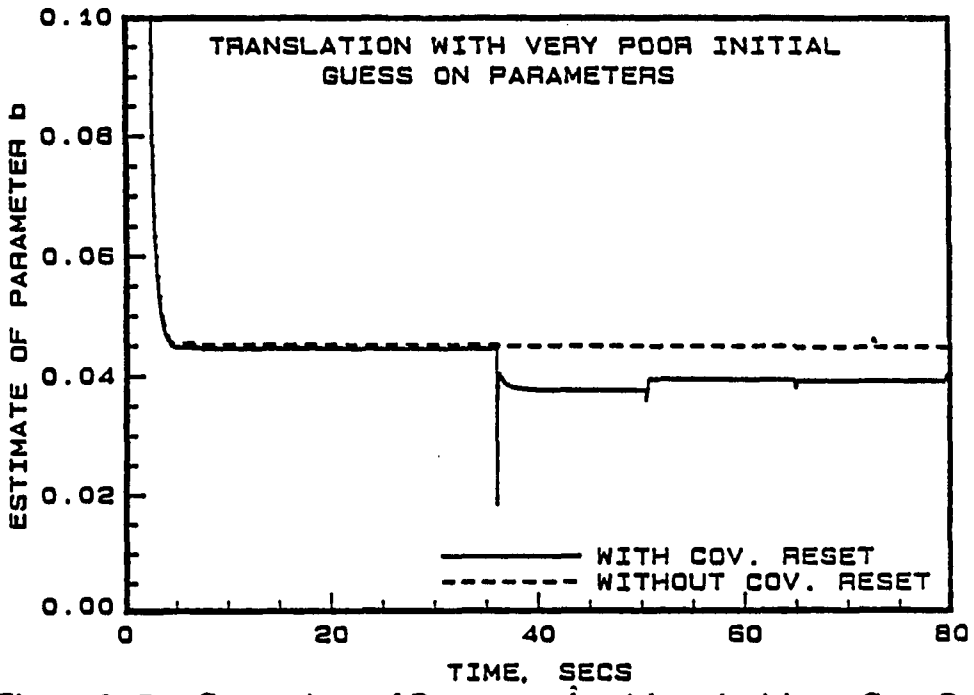

Figure 6.17: Comparison of Parameter $\hat{b}_1$ with and without Cov. Reset

6.5.3 Comparison of Experimental MRAC with Simulation

A comparison study is done in this section between the experimental results and the computer simulation results for model reference adaptive controller. In the open-loop frequency response testing, it was determined that there was a difference of about 6 db between the experimental and simulation gain values. This difference was attributed to underestimating the gain of the voltage-to-current converter block of the servosystem. Initial comparison studies conducted between the computer simulation and experimental results indicated this difference. The gain of the voltage-to-current converter in the computer model was modified to take into

account this discrepancy. Figure 6.20 shows a comparison between the modified simulated and actual position response. This comparison shows agreement between simulation and experiment, indicating the good evaluation of parameters in our simulation studies.

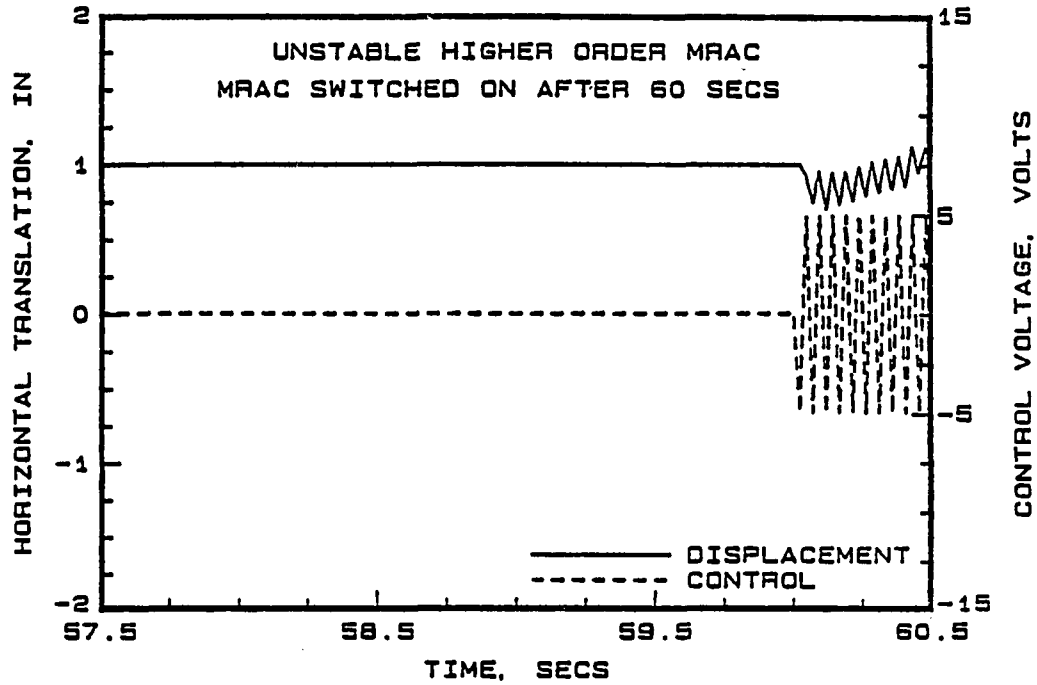

Figure 6.18: Position Response under Proportional and Higher Order MRAC

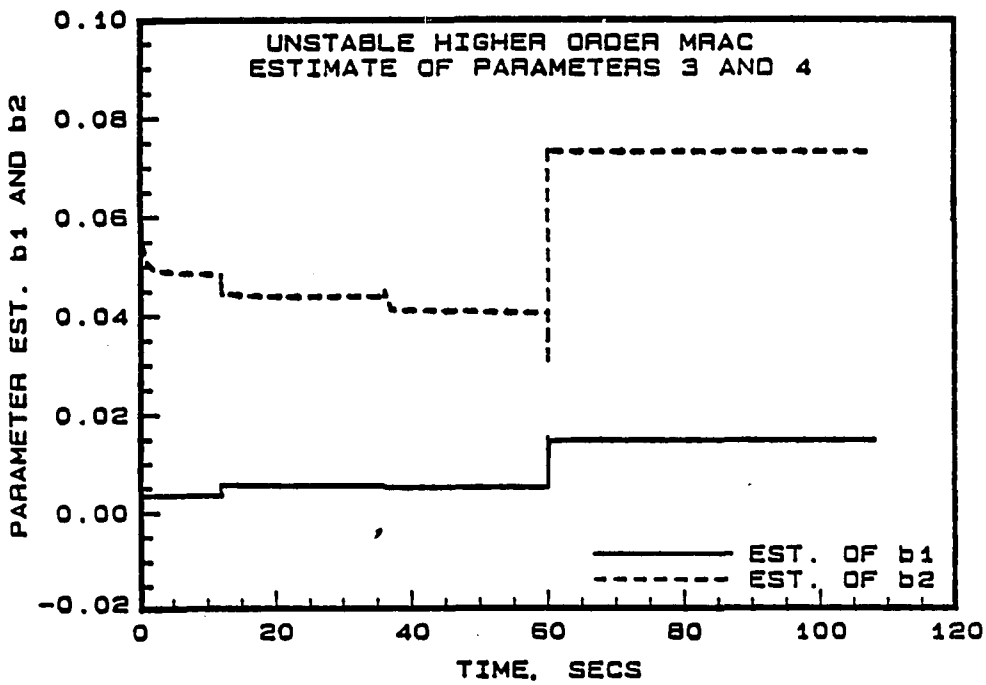

Figure 6.19: Control Parameters under Proportional and Higher Order MRAC

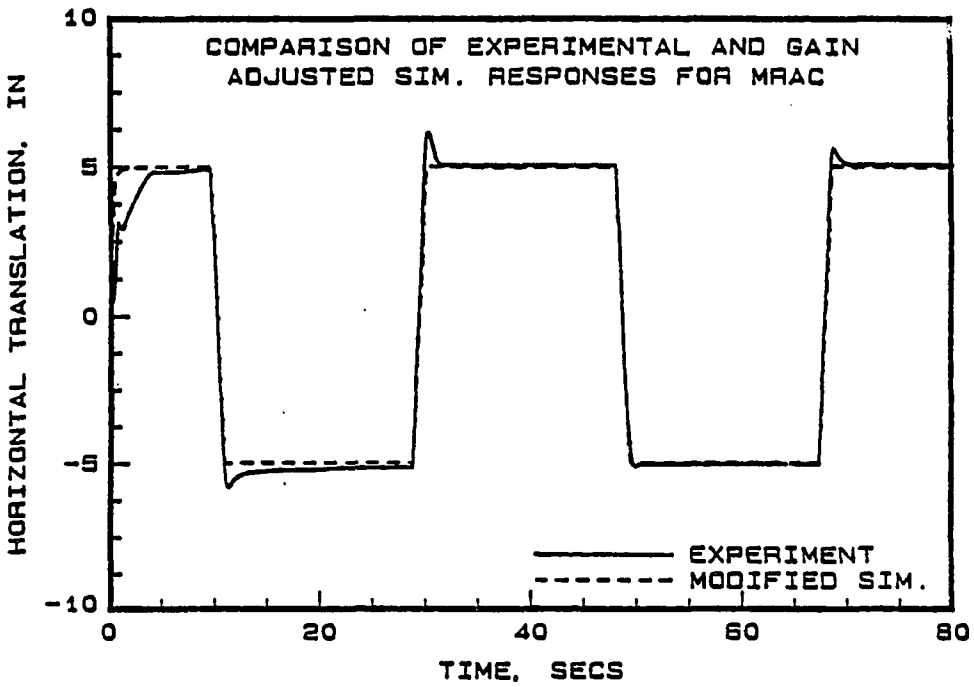

Figure 6.20: Comparison of MRAC Position Response, Experiment and Simulation

6.5.4 Pole Assignment Adaptive Control

As pointed out earlier, the control laws based on the one-step-ahead principle (MRAC) require restrictive assumptions on the zeros of the system. The pole assignment adaptive controller (PAAC) , which overcomes this difficulty, was described in Chapter Four. In this section, the implementation of a pole assignment adaptive controller to the horizontal axis of the robot is discussed. A second order DARMA model was used for identification and control.

The pole assignment adaptive controller synthesis procedure equates the closed-loop characteristic equation of the system with a user defined polynomial $A^*(q^{-1})$, and then solves for the unknown controller polynomials $L(q^{-1})$ and $P(q^{-1})$ based on real-time parameter estimates. These polynomials are used in the control law (equation 4.45), and the resulting control signal is applied to the servovalve for closed-loop control.

The desired pole set $A^*(q^{-1})$ may contain up to three poles. Specifying these values requires careful consideration. In this experiment, the dominant poles were selected based on the model selection scheme described earlier. The third pole was assigned to well damped value (z=0.5).

Figure 6.21 shows the position and control response of the horizontal axis of the robot to a square wave reference command using a pole assignment adaptive controller. As can be seen, the worst behavior occurs during the initial identification phase. After this transient period, the position response was well damped with no overshoot, as desired. Figure 6.22 shows the position and control response between 21 to 24 secs. It can be seen that the position and control response are well damped.

Figure 6.23 shows the estimates of the denominator coefficients of the DARMA model. The parameters converge after the first cycle, and change very little after this period. From these parameter estimates, the open-loop pole at the unit circle corresponding to the pure integrator can be extracted. Figure 6.24 shows the parameter estimates of the numerator coefficients of the DARMA model. As expected, the estimates show a nonminimum phase behavior, with $\hat{b}_2$ greater than $\hat{b}_1$. The numerator parameters converged very well after an initial transient period, even though they were significantly smaller than the denominator ones.

6.5.5 Single Axis Adaptive Control of Other Axes

Based on the experience from applying adaptive control algorithms to the horizontal axis of the robot, the pole assignment adaptive controller was applied to the vertical translational axis and base rotational axis of the robot without change.

Figure 6.25 shows the position response using pole assignment adaptive control on the vertical axis of the robot. The vertical axis of the robot rests on the base of the robot. The upward motion of the robot was the negative displacement from the base rest position. A series of steps were applied in the negative direction, and then the step inputs were reversed in direction in the same order. Figure 6.26 shows the response during 28 to 31 secs. It can be seen that the response is well damped with no overshoot, corresponding to the specified dominant poles of the pole assignment controller.

In order to see whether the gravitational forces cause a drift in the response, the closed loop input-output data was used to identify this drift parameter in addtion to

the DARMA model coefficients. The identification from four sets of data indicated an average drift coefficient of 0.000459. This value was more than two orders of magnitude smaller than the smallest control parameter coefficient, indicating that the drift coefficient was negligible. Further evidence to this fact was the position response over a period of time which indicated no change in the robots vertical position after reaching the desired value until the next step input is applied. The gravitational forces were offset by the differential area across the piston.

As a next step, the base rotational axis was considered for adaptive control. Since the base has vane actuators as opposed to the linear actuators used in the horizontal and vertical axes, a simple proportional controller was first designed for the base. The steady state errors were very large. Figure 6.27 shows a significant steady state error when a proportional controller with a gain of two was used for control. The poor response of the controller was attributed to the stick/slip frictional nature of the servovalve actuator, resulting in dead band behavior. Figure 6.28 shows the plot of input voltage versus the base velocity. The plot shows a dead band during which the velocity is zero when the input voltage is between -0.25 and +0.5 volts. Further, the plot shows the hysteresis due to this nonlinear effect. Initial tests using a pole assignment adaptive controller indicated significantly less steady state error compared to proportional control, but the position response had an overshoot, despite specifying critically damped reference model. However, when the reference model was chosen to be overdamped, the overshoot in the position response was eliminated. Additionally the PAAC had a significantly smaller steady state error than the proportional controller, indicating an improved response. Fig-

ure 6.29 show the base rotational response to pole assignment adaptive control. As can be seen, the position response is well damped with no overshoot after the initial transient period when the parameters have still not converged. Figure 6.30 shows the comparison between the command displacement and position response for the base when PAAC was used. It can be seen that the PAAC tracks the command with very little steady state errors.

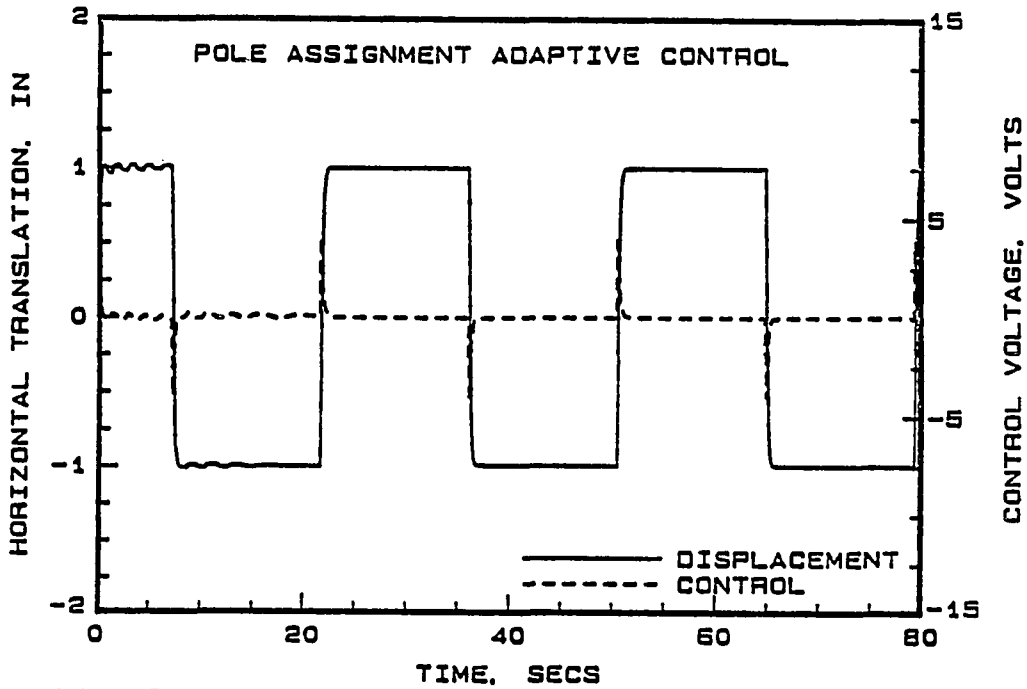

Figure 6.21: Position and Control Response to Pole Assignment Adaptive Control

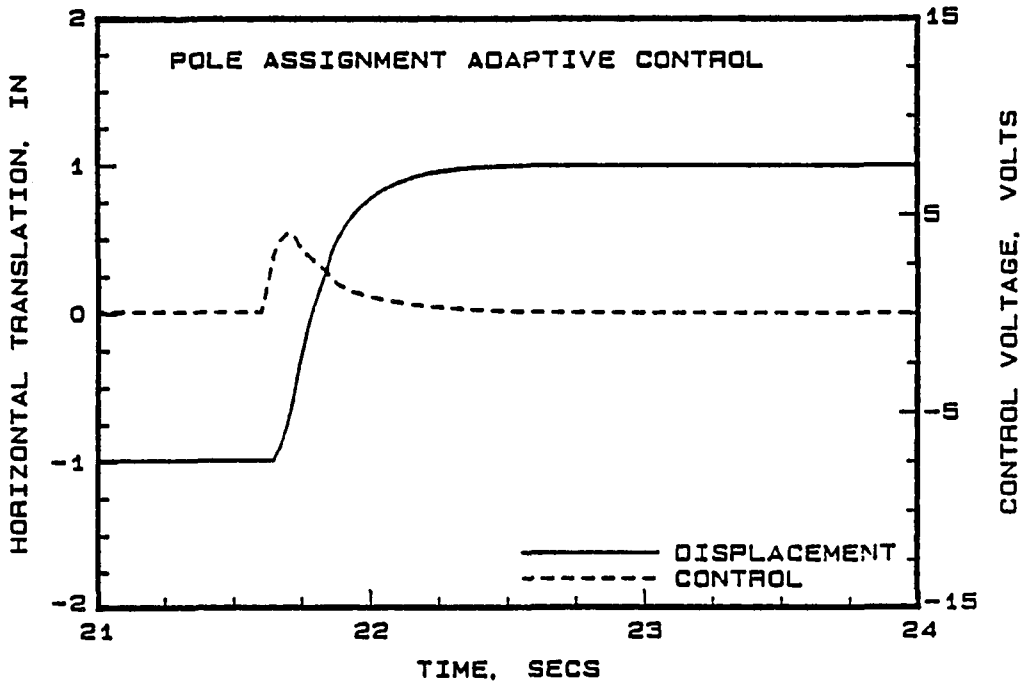

Figure 6.22: Position and Control Response during 21-24 secs using PAAC

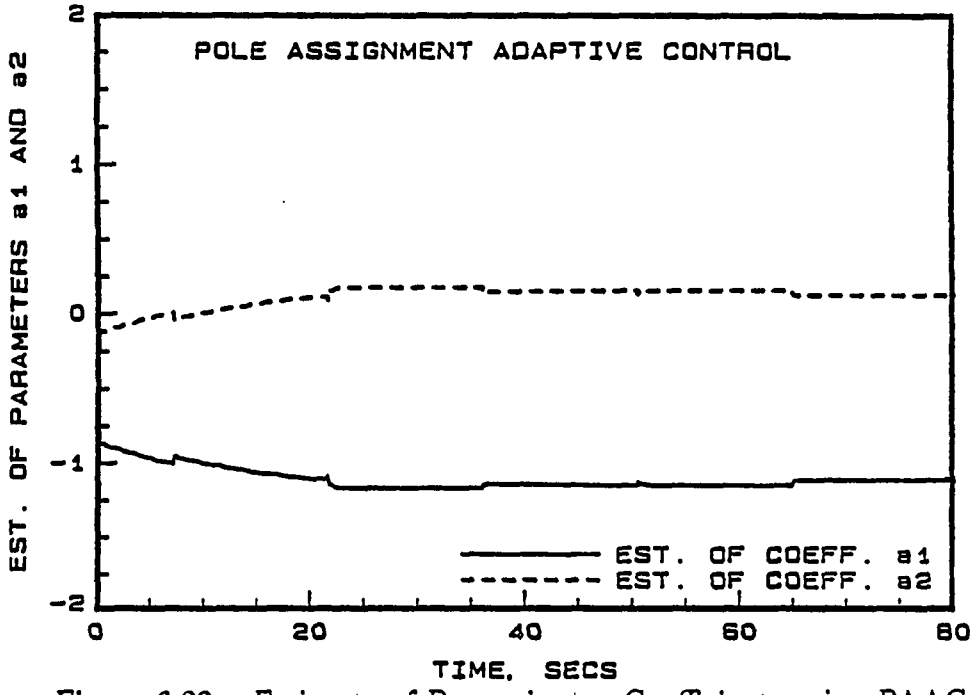

Figure 6.23: Estimate of Denominator Coefficients using PAAC

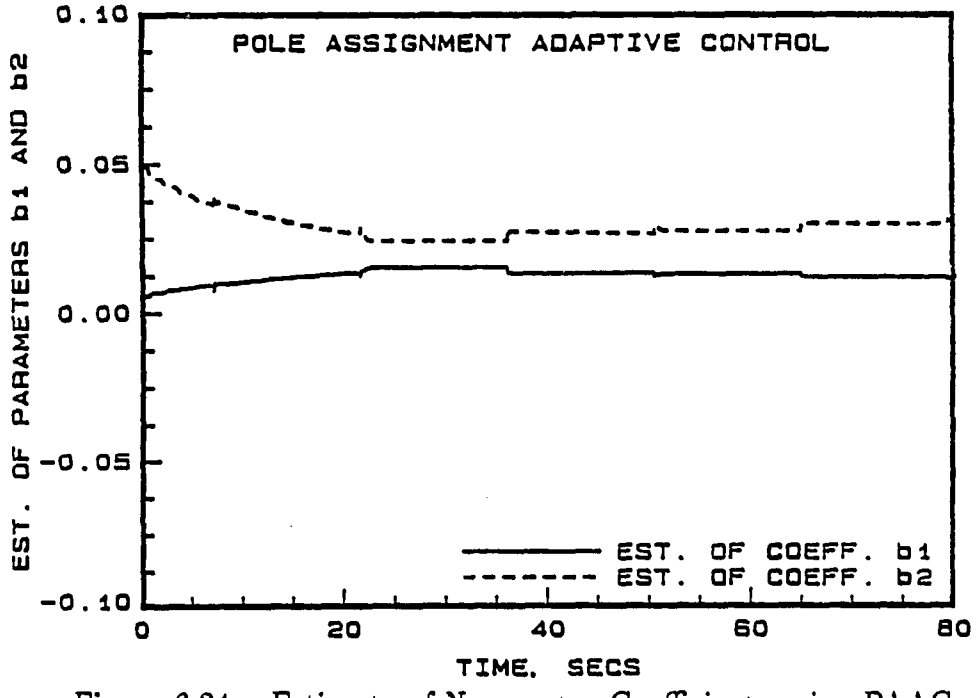

Figure 6.24: Estimate of Numerator Coefficients using PAAC

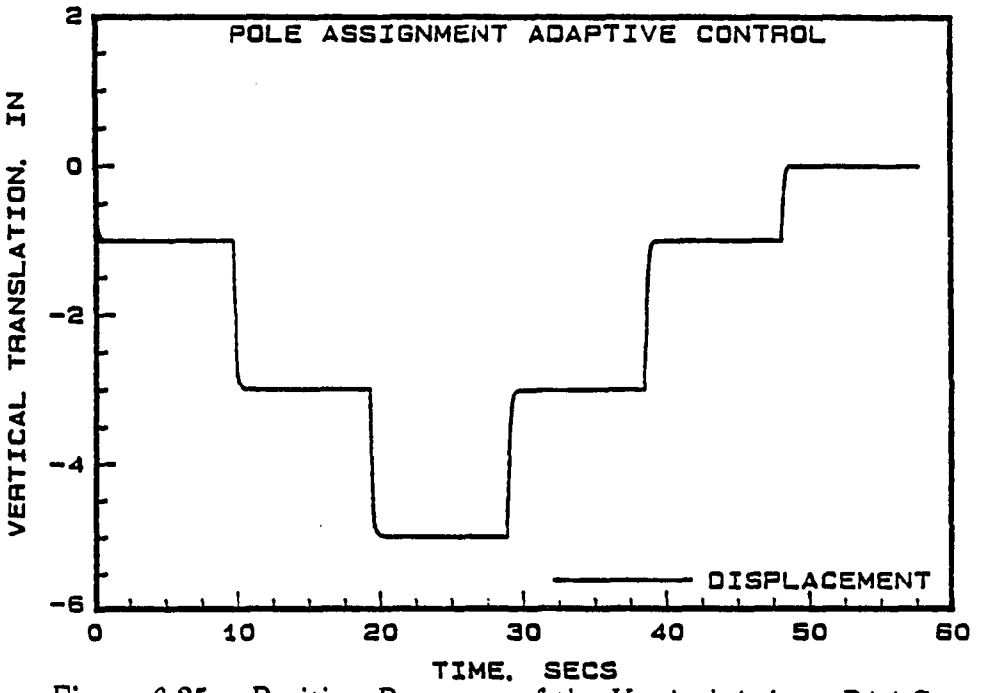

Figure 6.25: Position Response of the Vertical Axis to PAAC

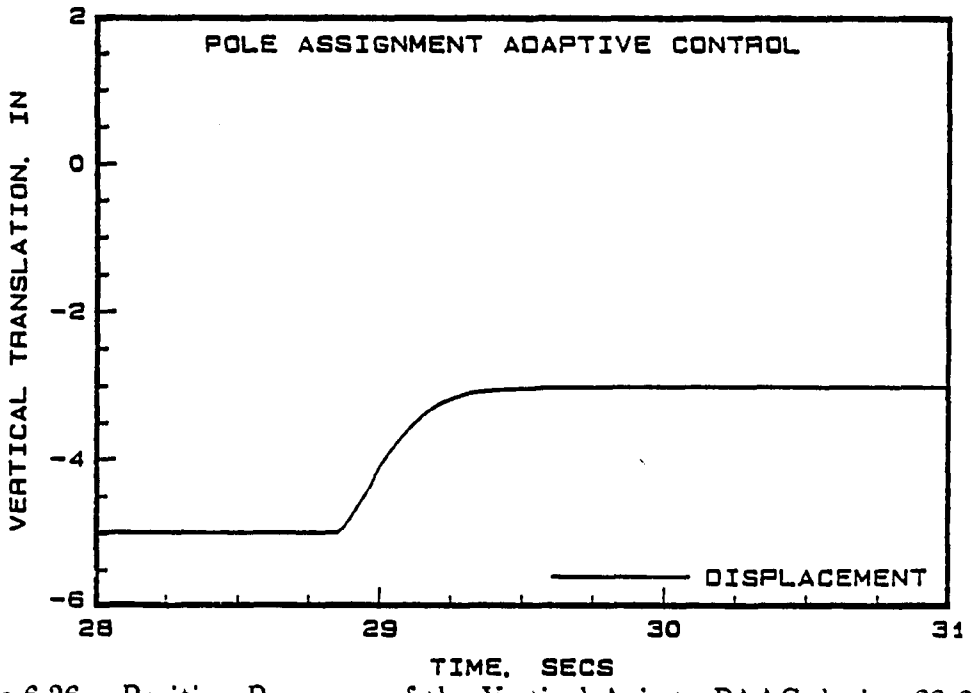

Figure 6.26: Position Response of the Vertical Axis to PAAC during 28-31 secs

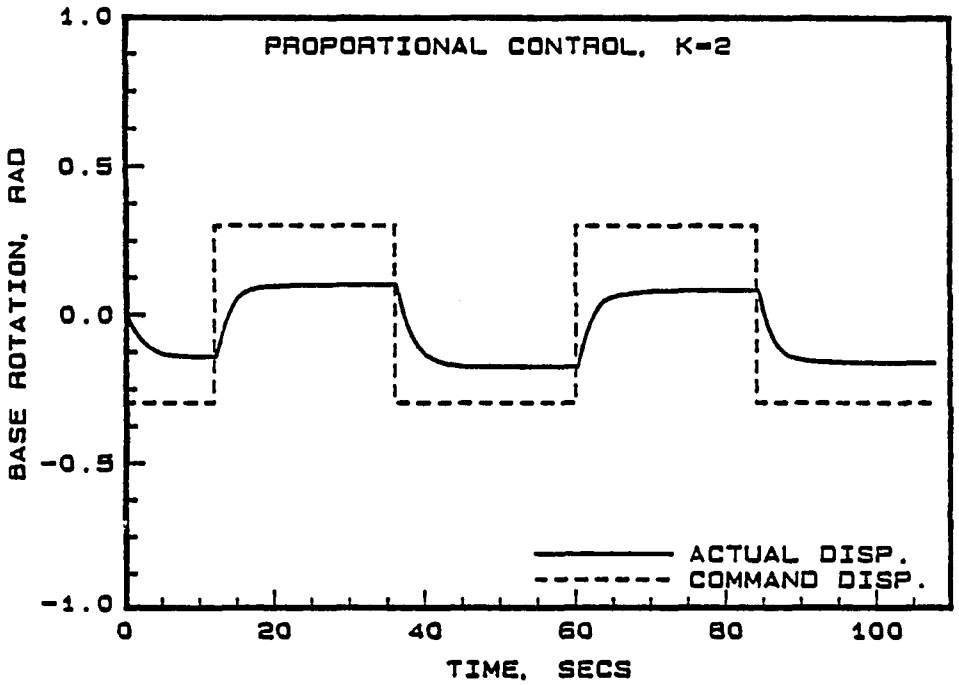

Figure 6.27: Position Response of Base Rotation Axis to Proportional Control

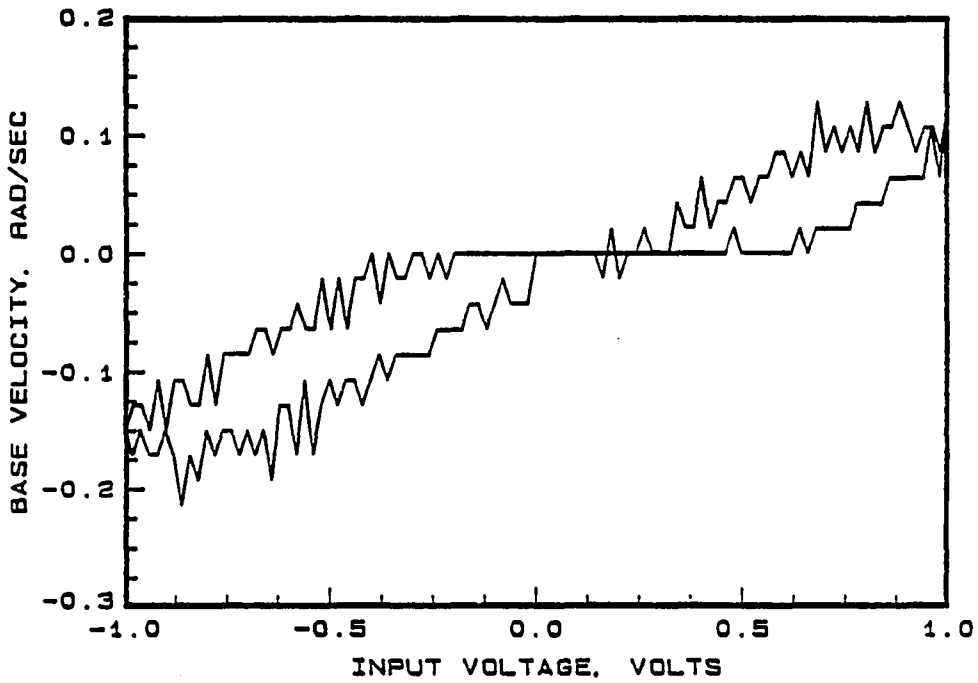

Figure 6.28: Hysteresis Plot for the Base Rotation Axis

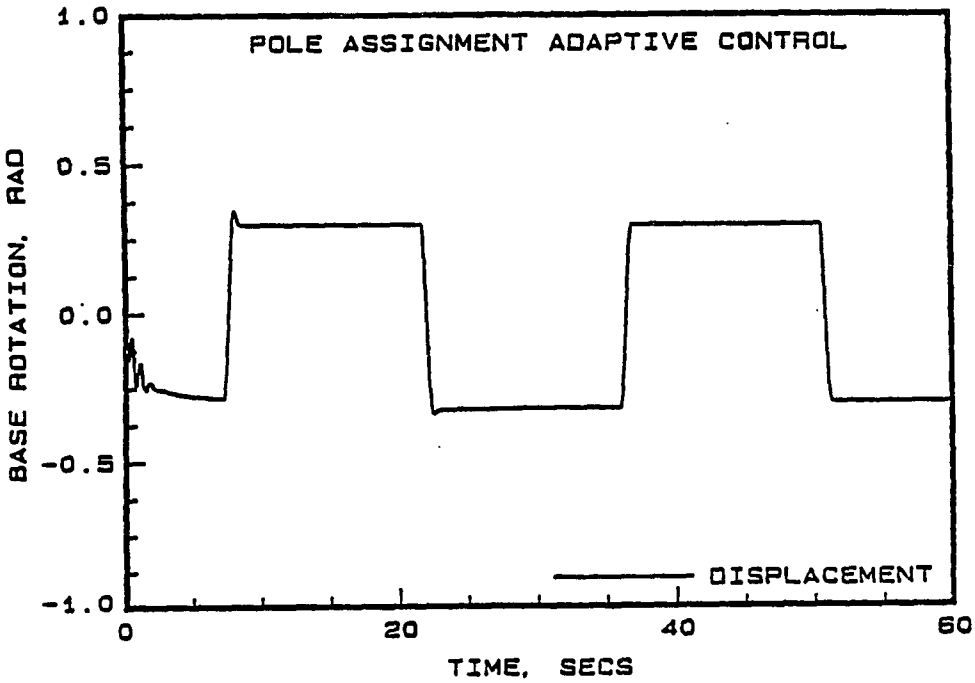

Figure 6.29: Position Response of Base Rotation Axis to PAAC

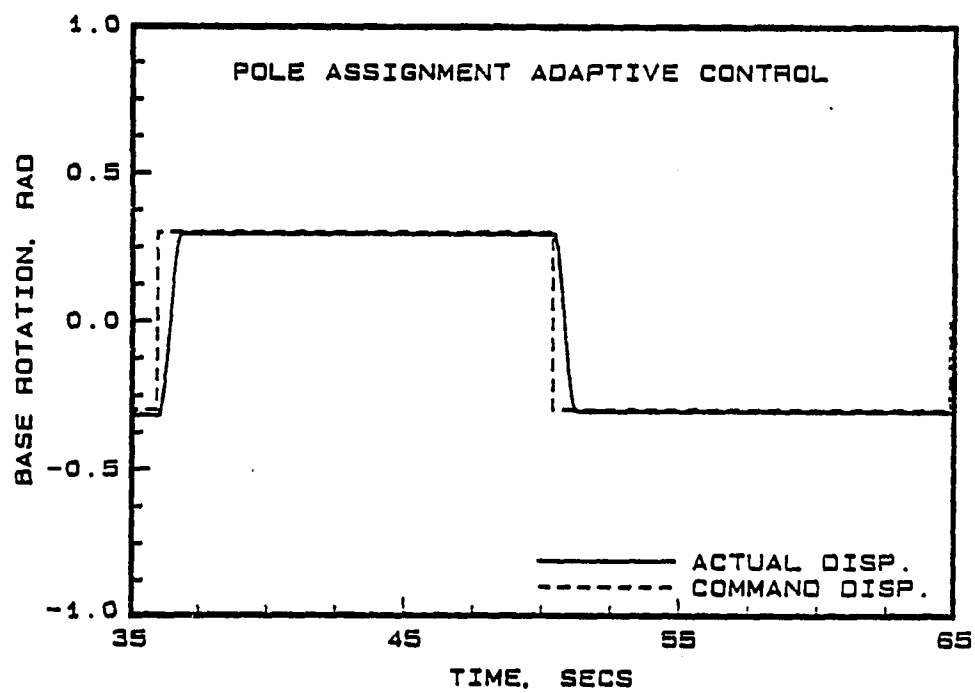

Figure 6.30: Command and Position Response of Base Rotation Axis to PAAC

6.5.6 Summary of Single Axis Results

Open-loop frequency response tests conducted on the horizontal axis of the robot identified the integrator. However, the identification of the higher order roots corresponding to the quadratic lag term was difficult since the robot could not be excited beyond 67.28 rad/sec due to structural resonance. The open loop gain was also determined from the frequency response plots.

Recursive least-squares estimation of the open-loop data provided a discrete-time representation of the plant transfer function. Comparisons made between the open-loop gain obtained by estimation and frequency response showed close agreement.

Higher order models were identified with the open-loop data. A comparison of the fit error between first, second, and third order models showed that the fit error was smaller for more complex models.

Simple proportional and model reference controllers were implemented on the robot prior to adaptive control of the horizontal axis. The model reference controller exhibited an overshoot because of the time delay in the plant transfer function.

The MRAC response also showed an overshoot prior to settling down. This was due to time delay in the system and a control law which had to be based on a system with no time delay. The steady state error using MRAC was smaller than model reference and proportional controller.

The pole assignment adaptive controller which does not require zero cancellation was implemented on the horizontal axis of the robot. A second order model was used for the discrete-time plant. Results indicated well damped position re-

sponse corresponding to the reference model. In addition to good position response, smaller steady state errors were obtained using pole assignment adaptive control than MRAC.

An average steady state error of 0.017 inch occurred when a proportional controller with gain of unity was used on the horizontal axis of the robot. When the MRAC was used, the average steady state error was around 0.007 inch. The steady state error further reduced to 0.0035 inch when PAAC was used.

As a next step, the pole assignment adaptive control was applied to the vertical axis and the base. While the vertical axis position response was well damped with no overshoot, the base had overshoot in the response when a critically damped reference model was used. This was due to the highly nonlinear nature of the friction acting on the base. A slightly overdamped model was used for the base to obtain position response with no overshoot.

The fixed gain controller performed poorly for the base axis rotation with steady state error of 0.349 radians with a 0.600 radian command. The PAAC provided a significant improvement over the proportional controller, even in the presence of strong nonlinearities resulting in steady state error of 0.010 radians.

The same pole assignment adaptive algorithm was successfully applied to each of the four separate axes. This indicated the excellent ability of the adaptive controller to adapt to different dynamic conditions.

6.6 Multiple Axis Experimental Testing

The results of the single-axis adaptive control tests indicated that all axes of the robot can be controlled adaptively. But, prior to the implementation of multiple-axis adaptive controller, the following hardware modifications were required:

1) Multiple channel D/A voltage conversion was required.

2) The output of the four resolvers had to be properly sequenced through the resolver-to-digital converter (RDC) and the parallel I/O ports.

The four channel D/A voltage conversion was accomplished through modifications in the software. The four channel resolver-to-digital conversion was achieved through modifications in hardware and software. The circuit diagram in Figure 6.4, includes the additional hardware built for receiving four channels of resolver data. Further, software changes were made to ensure that the controller received the correct resolver data for each axis.

At first, a simultaneous pole assignment adaptive control of the vertical and horizontal axes of the robot was tested. Figures 6.31 and 6.32 show the position response of two axis adaptive controller. The response was well damped with no overshoot as desired.

A decoupled pole assignment scheme was used to adaptively control all the four axes of the robot. Figures 6.33 through 6.36 show the position response of all four axes of the robot under simultaneous four-axis adaptive control. The results indicate that all four axes have well damped responses. Figures 6.37 and 6.38 show the transient control and position response of the base and wrist axes. These responses correspond to the period during which the identification was initialized. Figures 6.39

through 6.42 show the position response of the four-axis pole assignment adaptive controller after the parameters had converged. The plots show that in all cases the position response was well damped with no overshoot. Figures 6.43 and 6.44 show the parameter estimates of base and wrist denominator coefficients which indicate good convergence after the initial transient period.

The tests using the four-axis pole assignment adaptive control were repeated nine times. In all cases, the responses were comparable, indicating repeatability between experiments.

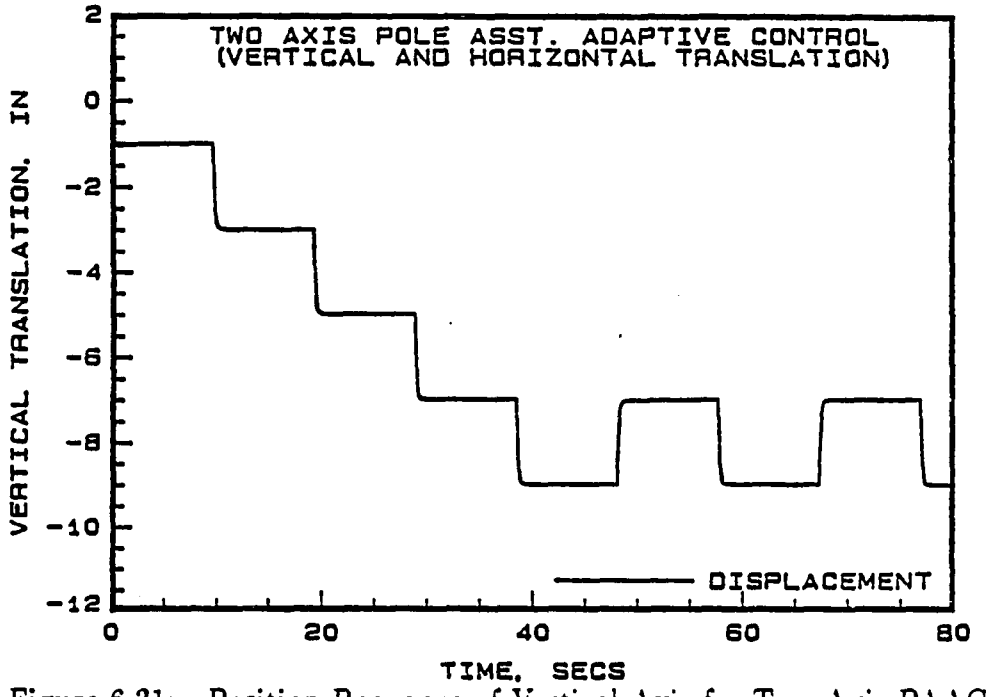

Figure 6.31: Position Response of Vertical Axis for Two Axis PAAC

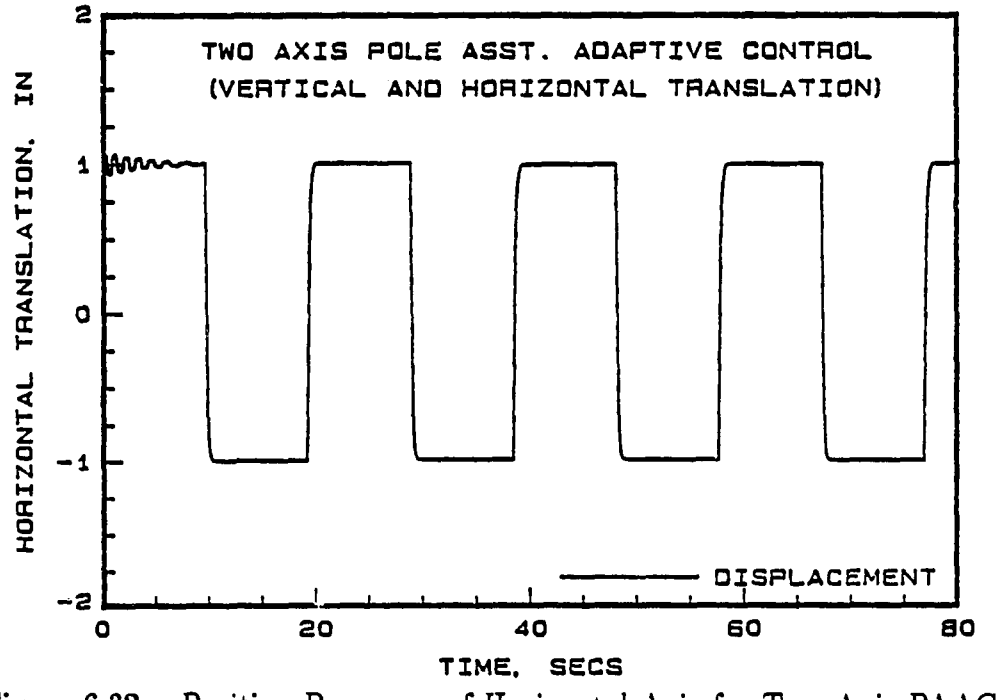

Figure 6.32: Position Response of Horizontal Axis for Two Axis PAAC

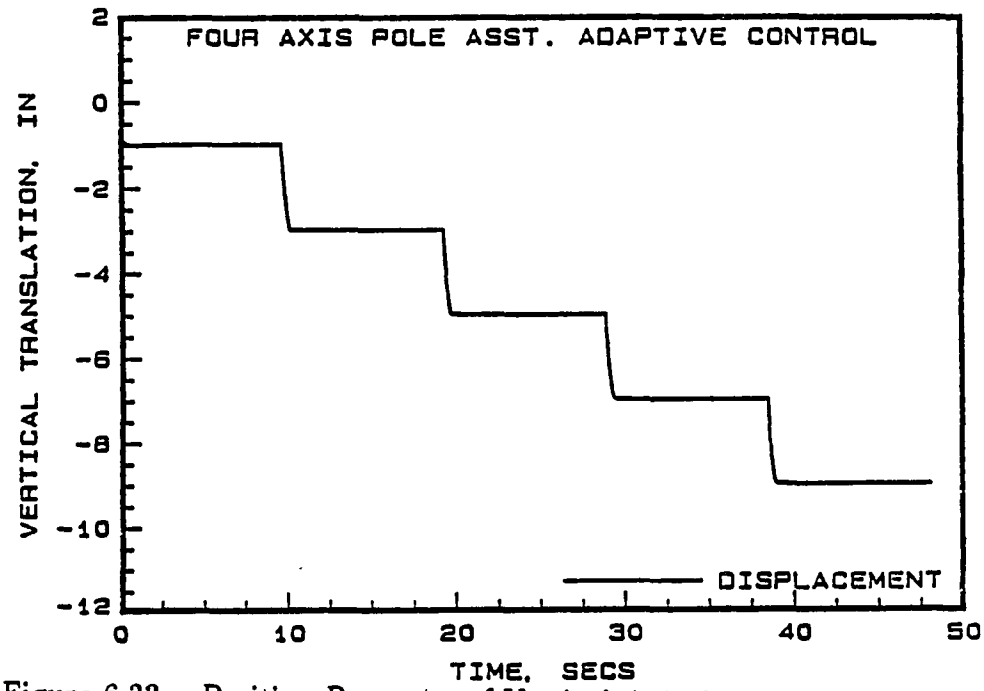

Figure 6.33: Position Response of Vertical Axis for Four Axis PAAC

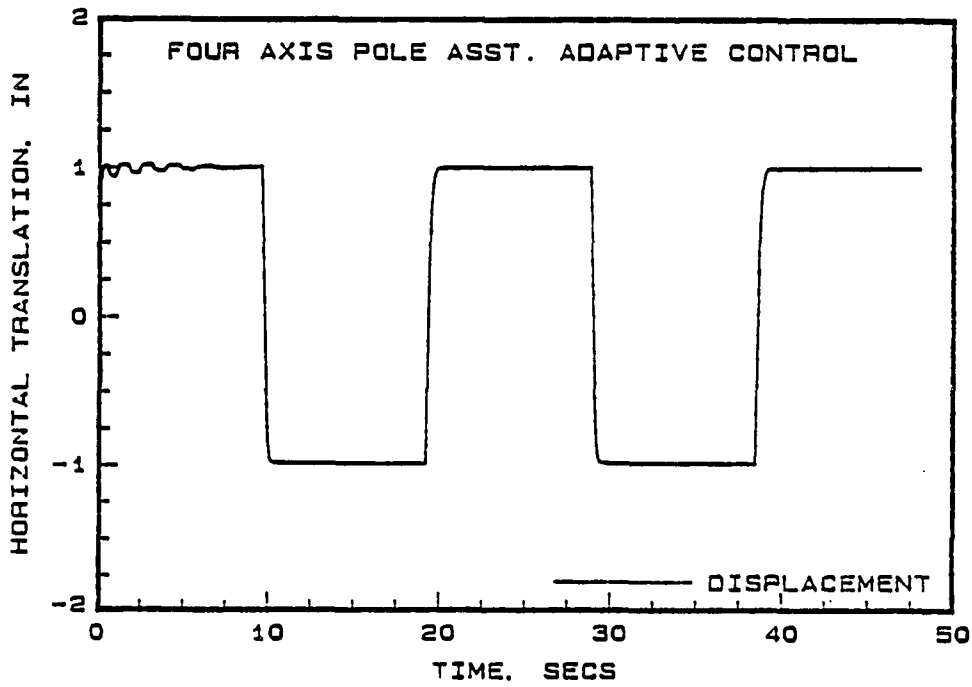

Figure 6.34: Position Response of Horizontal Axis for Four Axis PAAC

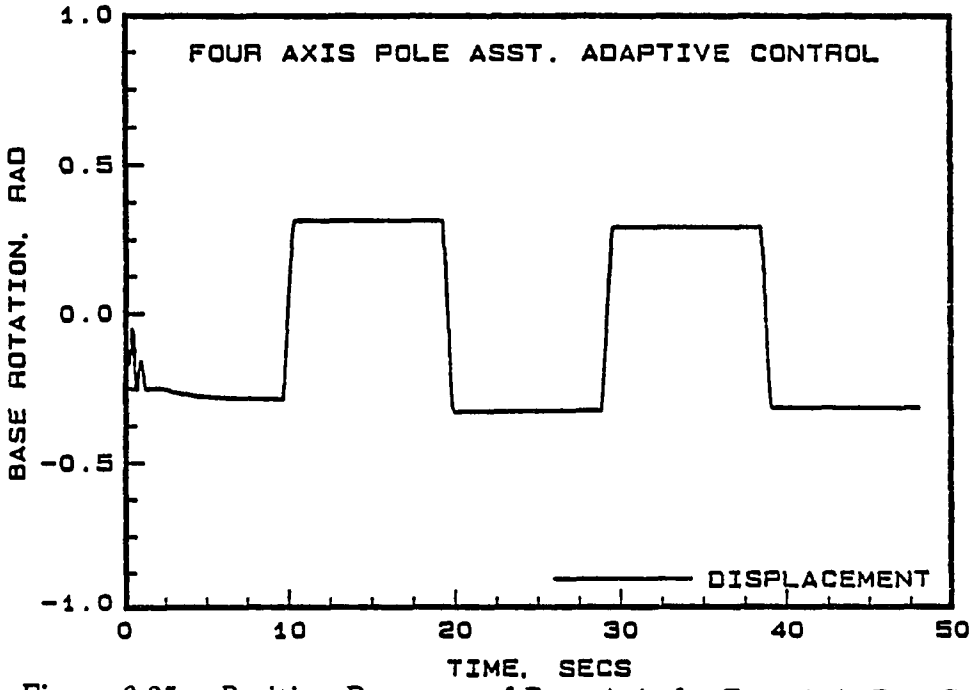

Figure 6.35: Position Response of Base Axis for Four Axis PAAC

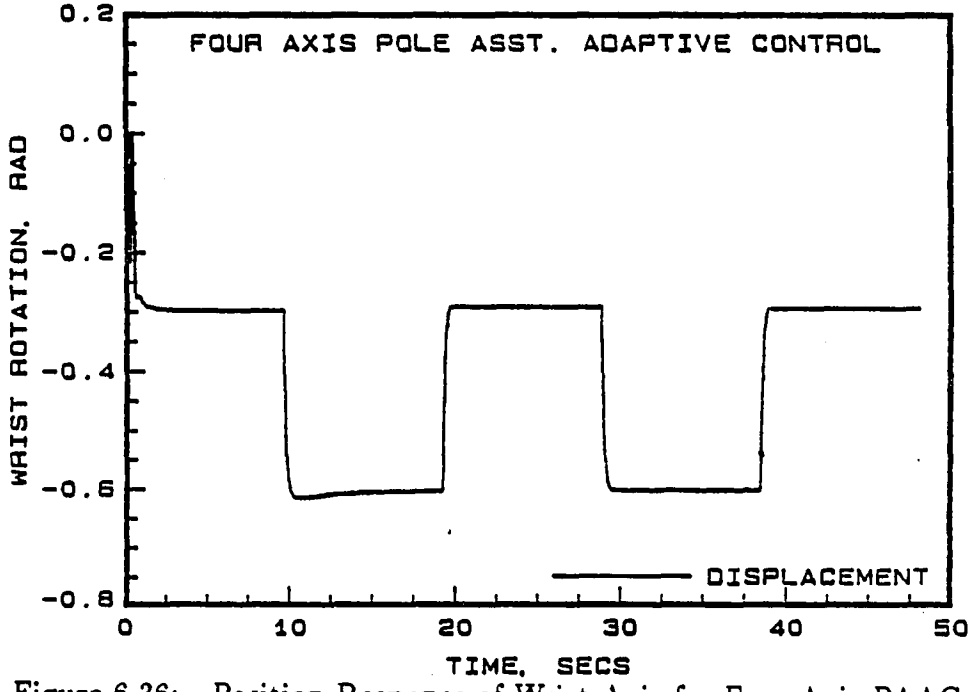

Figure 6.36: Position Response of Wrist Axis for Four Axis PAAC

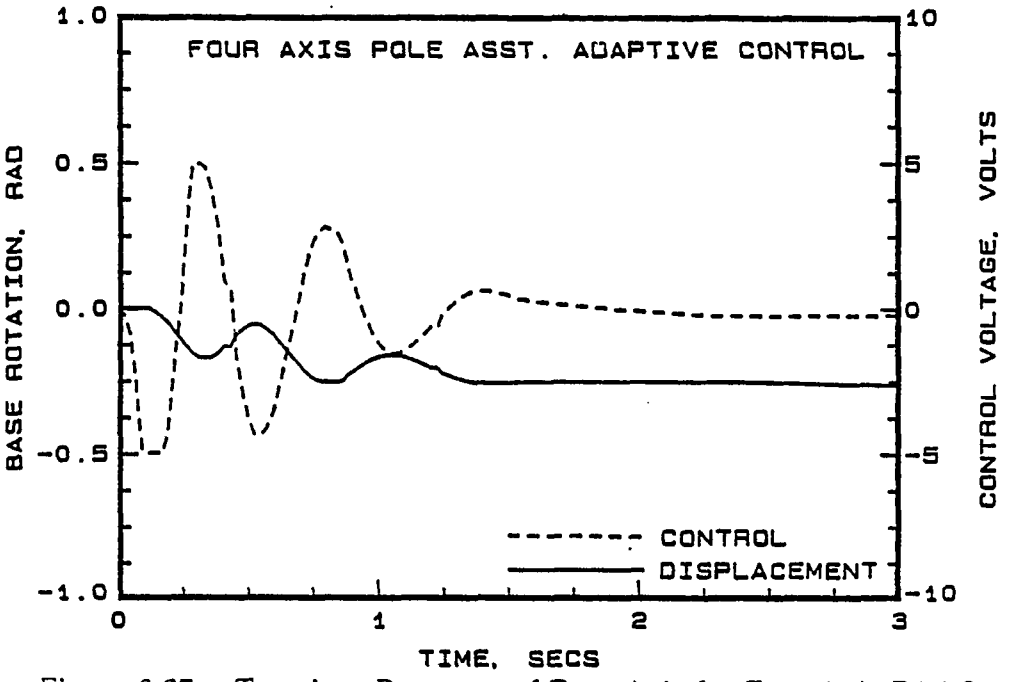

Figure 6.37: Transient Response of Base Axis for Four Axis PAAC

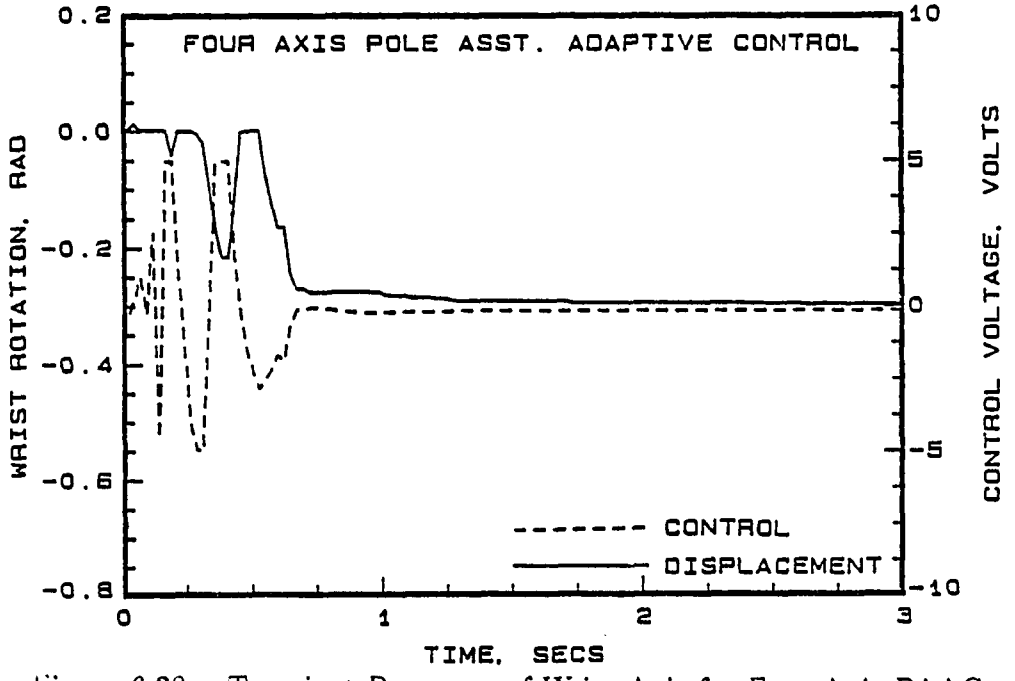

Figure 6.38: Transient Response of Wrist Axis for Four Axis PAAC

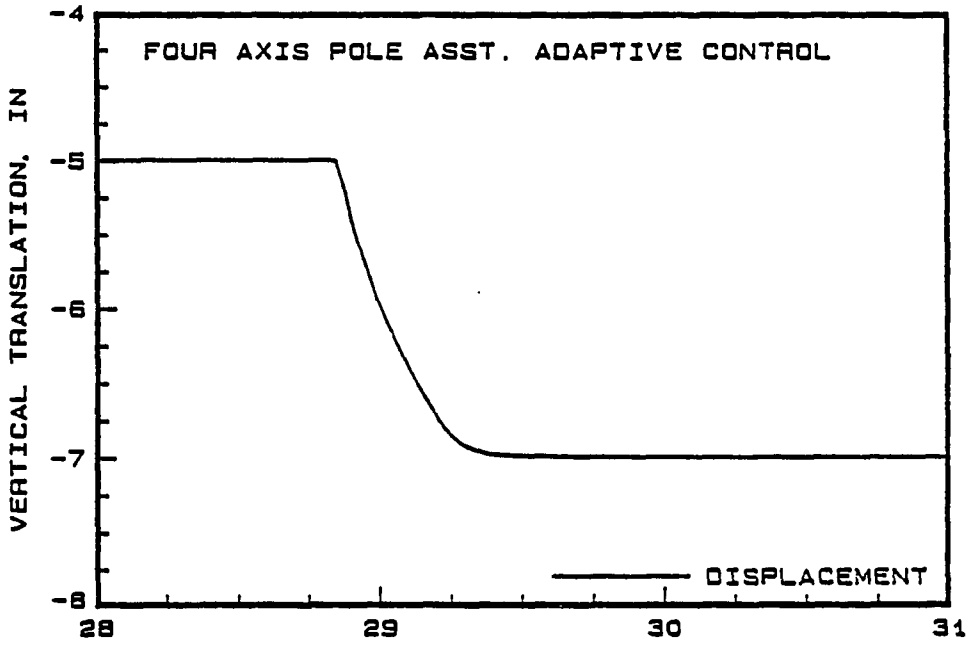

Figure 6.39: Position Response of Vertical Axis between 28-31 secs

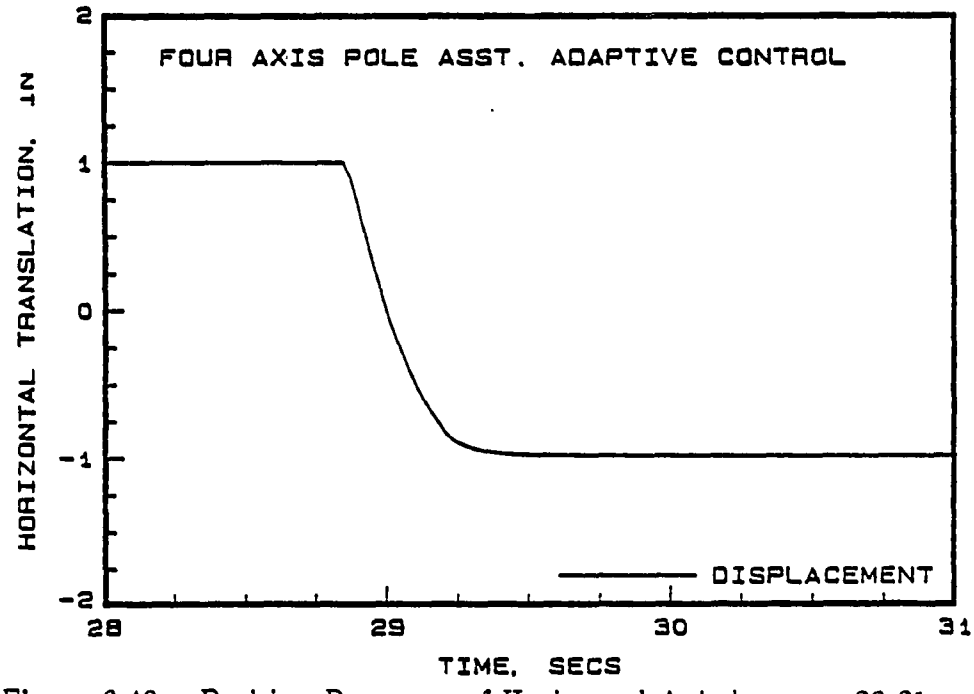

Figure 6.40: Position Response of Horizontal Axis between 28-31 secs

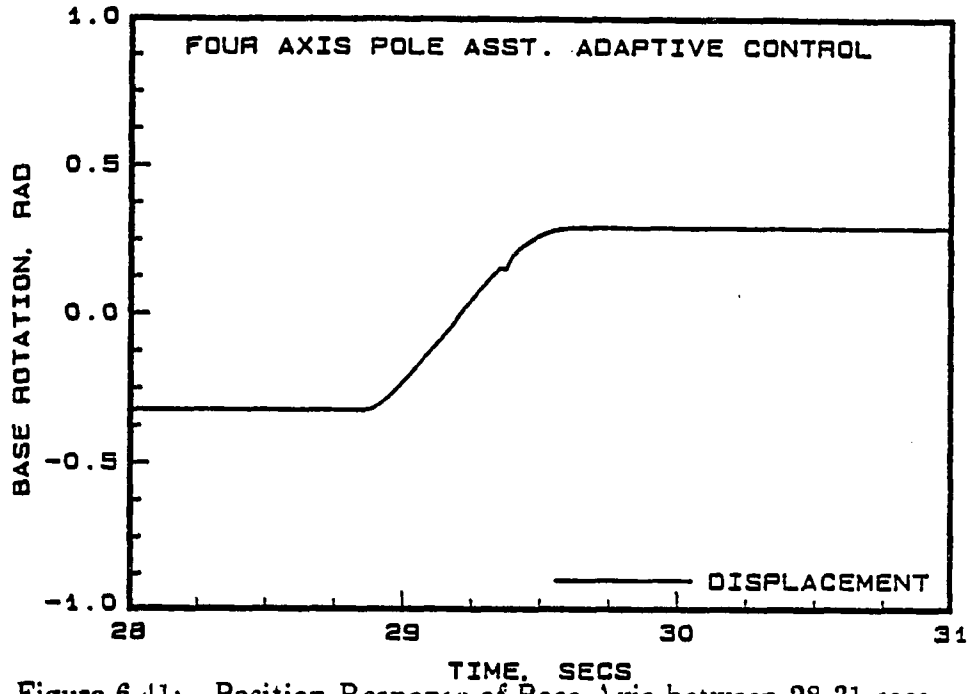

Figure 6.41: Position Response of Base Axis between 28-31 secs

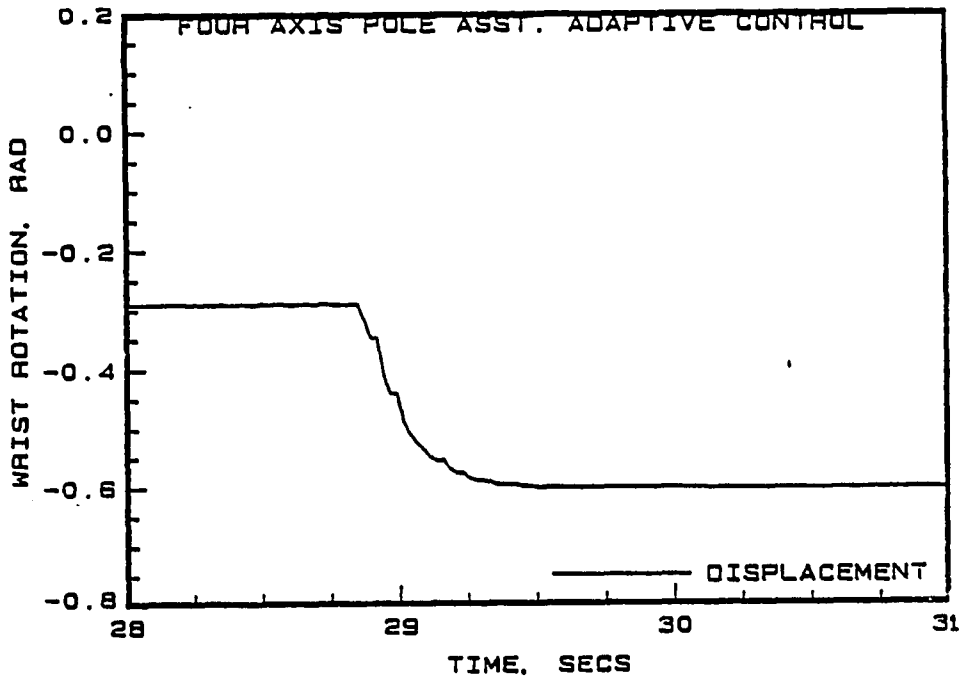

Figure 6.42: Position Response of Wrist Axis between 28-31 secs

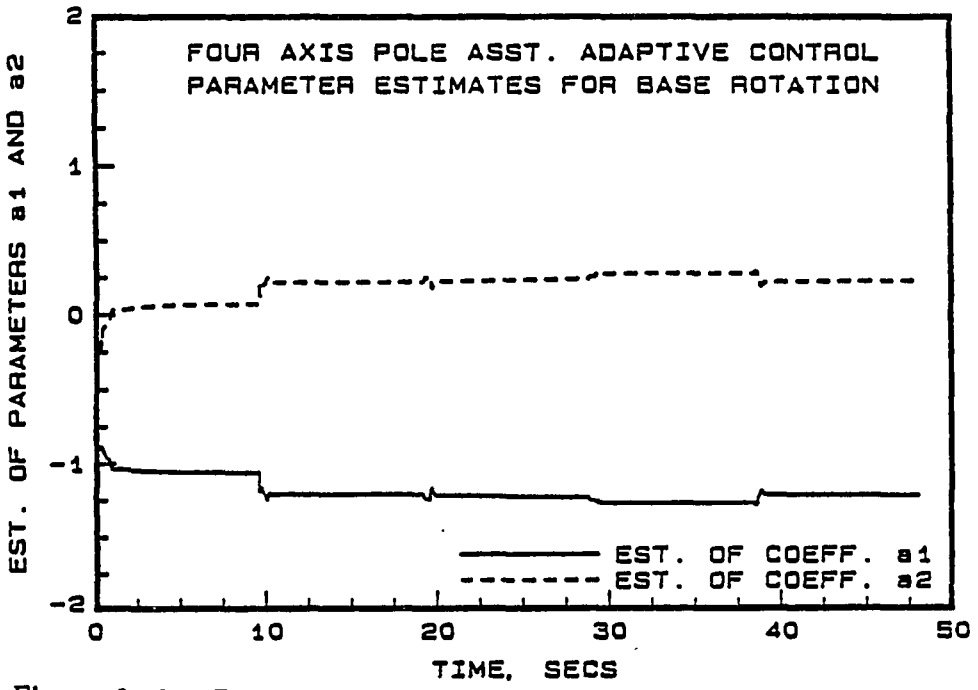

Figure 6.43: Parameters of Base Rotation Axis for Four Axis PAAC

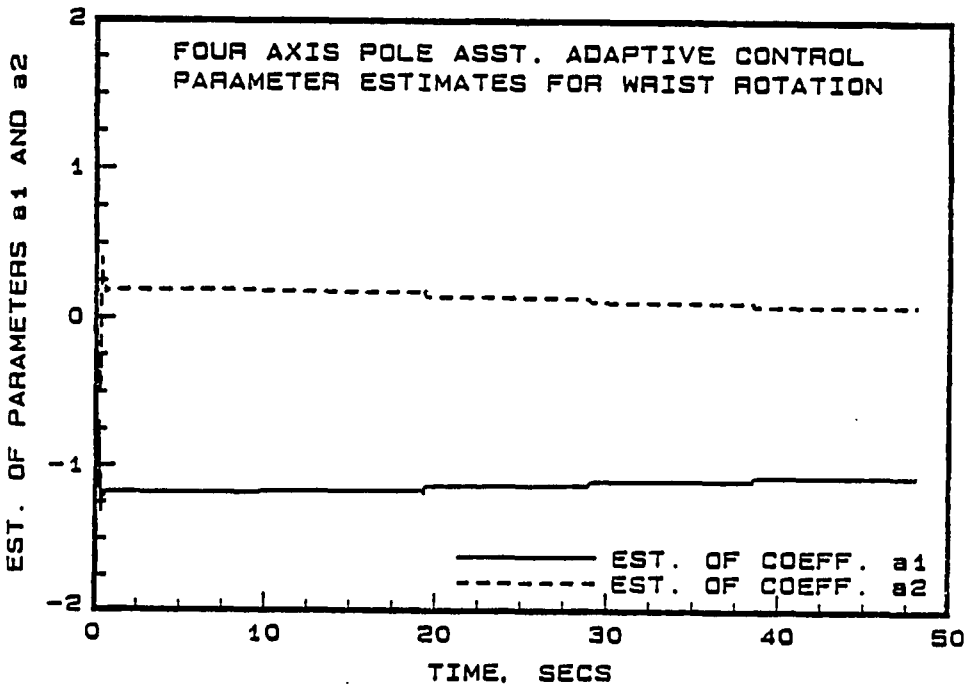

Figure 6.44: Parameters of Wrist Axis for Four Axis PAAC

7 CONCLUSIONS

The performance of DARMA model based direct and indirect discrete-time adaptive control algorithms for a four axis hydraulic robot was evaluated by simulation studies and experimental testing. Weighted one-step-ahead, model reference, and pole assignment adaptive control algorithms were investigated for single axis control of the robot. While all three algorithms provided satisfactory results when designed to provide critical damping, the model reference controller coupled with least squares based identification algorithm was chosen for application to the four-axis simulation because of its ease in design and lower pressure fluctuations in the hydraulics.

A linear model of the hydraulic components was found to be adequate. Parameter estimates based on a linear model assumption tended to converge fast, and not vary, after the initial transient period. This indicated that the nonlinear effects were minimal. The linear parameter model used for the linkage dynamics worked very well. The identification routine was only active during a motion command and used the saturated voltages of the controller in the estimation of the parameters.

The primary nonlinearity encountered was the saturation of the servovalve voltage, which placed an upper limit on the axis velocity. Saturation was effectively taken into account in the design of the controller by using saturated values for

identification and control. In the MRAC case, saturation was dealt with by using the displacement of the move to determine the natural frequency of the reference model. This allowed the axis to limit the saturation. Increasing this frequency by a constant factor allowed the axis to run in a saturated condition with a stable, well behaved response.

A 12 bit resolver to digital converter was adequate for experimental purposes, since a 10 bit resolver to digital converter did not significantly affect the position and control response of the system.

A comparison study done between the proposed adaptive control algorithm and Craig's approach to adaptive control showed that the proposed discrete-time adaptive controller performed in a comparable manner, despite the discretization and delay incorporated in our model for realistic real-time implementation goals. Further, Craig's approach is more complicated than the algorithms used here, since it requires desired joint velocities and accelerations in addition to desired position and also has a component which requires the linkage dynamics at each control interval.

Experimental open loop frequency response tests conducted on the horizontal axis of the robot identified the integrator. However, identification of the higher order roots corresponding to the quadratic lag term was difficult since the robot could not be excited beyond 67.28 rad/sec due to structural resonance. The open loop gain was also determined from the frequency response plots.

Recursive least squares estimation of the open-loop experimental data provided a discrete-time representation of the plant transfer function. Comparisons made

between the open loop gain obtained by estimation and frequency response showed close agreement. Comparison of the fit error between first, second, and third order models showed that the fit error was smaller for more complex models.

The model reference controller exhibited an overshoot due to the time delay introduced by the control computer. As with the non-adaptive model reference controller, the MRAC response also showed an overshoot prior to settling down.

In order to overcome the difficulties encountered in implementing the MRAC, the pole assignment adaptive controller was implemented. Results of PAAC applied to the horizontal axis of the robot indicated well damped position response with smaller steady state errors. An identical PAAC applied to the vertical axis of the robot indicated that the position response was well damped.

However, the PAAC when implemented on the base had an overshoot in the position response due to the highly nonlinear nature of the servovalve/actuator of the base. The desired closed loop poles were chosen to be slightly overdamped for the base to obtain a well damped position response. The adaptive controller provided significant improvement over the proportional controller for the base rotation.

The same pole assignment adaptive control algorithm was successfully applied to each of the four axes. This indicated that the adaptive controller worked well under different dynamic conditions.

The steady state position errors of MRAC and pole assignment adaptive controllers were smaller than the proportional controller indicating an improved accuracy over fixed controller. Table 7.1 shows the comparison of steady state position errors using different types of controllers.

Table 7.1: Comparison of Average Steady State Position Errors

Type of Control	Vertical (in.)	Horizontal (in.)	Base (rad.)	Wrist (rad.)
Horizontal Axis Proportional (K=1)	—	0.017	—	—
Horizontal Axis MRAC	—	0.007	—	—
Horizontal Axis PAAC	—	0.0035	—	—
Base Axis Proportional (K=2)	—	—	0.349	—
Base Axis PAAC	—	—	0.010	—
Multiple Axis PAAC	0.0026	0.0043	0.015	0.005

The results of simultaneous four-axis adaptive control indicated that the DARMA model based self-tuning regulator was sufficient despite the nonlinearities of the actuator and the robot.

In conclusion, DARMA model based adaptive controllers were shown to be implementable for a four axis hydraulic robot. Discrete-time MRAC is computationally simpler than a pole assignment adaptive controller. However, factors such as time delay limits the performance of MRAC. Therefore, from a practical standpoint, pole assignment adaptive controller provides the best overall performance.

8 BIBLIOGRAPHY

[1] Agathoklis, P., and Hamza, M. M. 1984. Comparison of three algorithms for load frequency control. Elect. Power Syst. Res., 7:165-172.

[2] Ananthakrishnan, S., and Fullmer, R. R. 1988. Adaptive control algorithms for a hydraulically actuated robot. Proceedings of the 11th IASTED Conference in Robotics and Automation, Santa Barbara, California.

[3] Arimoto, S., and Takegaki, M. 1981. An adaptive method for trajectory control of manipulators. Proc. 8th IFAC World Congress, Kyoto, Japan.

[4] Åström, K. J., and Wittenmark, B. 1973. On self tuning regulators. Automatica, 9:185-199.

[5] Åström, K. J., and Wittenmark, B. 1984. Computer-Controlled Systems. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[6] Åström, K. J., and Wittenmark, B. 1989. Adaptive Control. Addison-Wesley Co., Reading, Massachusetts.

[7] Chalam, V. V. 1987. Adaptive Control Systems. Marcel Dekker, Inc., New York.

[8] Choi, Y. K., Chung, M. J., and Bien, Z. 1986. An adaptive control scheme for robot manipulators. Int. J. Control, 44:1185-1191.

[9] Craig, J. J., Hsu, P., and Sastry, S. 1986. Adaptive control of Mechanical Manipulators. IEEE Conference on Robotics and Automation, San Francisco, California.

[10] Craig, J. J. 1988. Adaptive Control of Mechanical Manipulators. Addison-Wesley Co., Reading, Massachusetts.

[11] Doeblin, O. E. 1985. Control System Principles And Design. John Wiley & Sons, Inc., New York.

[12] Dubowsky, S., and Des Forges, D. T. 1979. The application of model reference adaptive control to robotic manipulators. ASME Journal of Dynamic Systems, Measurement, and Control, 101:225-232.

[13] Dubowsky, S. 1981. On the adaptive control of robotic manipulators: The discrete time case. Proceedings of the Joint Automatic Control Conference, Charlottesvile, Virginia.

[14] Finney, J. M.,de Pennington, A., Bloor, M. S., and Gill, G.S. 1985. A pole assignment controller for an electrohydraulic cylinder drive. ASME Journal of Dynamic Systems, Measurement, and Control, 107:145-150.

[15] Foley, E. D., Fullmer, R. R., Ananthakrishnan, S. 1987. Dynamic modeling and experimental coefficient identification of a four axis hydraulic robot. Proceedings of the 5th International Conference on Systems Engineering, Dayton, Ohio.

[16] Franklin, G. F. and Powell, J. D. 1980. Digital Control of Dynamic Systems. Addison-Wesley Co., Reading, Mass.

[17] Fullmer, R. R., Meek, S.G., and Jacobsen, S.C. 1985. Generation of the 7 degree-of-freedom controller equations for a prosthetic arm. Second Conf. of Applied Motion Control, Minneapolis, Minnesota.

[18] Gilbart, J. W., and Winston, G. C. 1974. Adaptive compensation for an optical tracking telescope. Automatica, 10:125-131.

[19] Goodwin, G. C., Ramadge, P. J., and Caines, P. E. 1980. Discrete-time multivariable adaptive control. IEEE Trans. Auto. Control, AC-25:449-456.

[20] Goodwin, G. C., Ramadge, P. J., and Caines, P. E. 1980. Generalization of results on multivariable adaptive control. IEEE Trans. Auto. Control. AC-25:1241-1245.

[21] Goodwin, G. C., and Sin, K. S. 1984. Adaptive Filtering Prediction and Control. Prentice Hall, Inc., Englewood Cliffs, New Jersey.

[22] Horowitz, R., and Tomizuka, M. 1986. An adaptive control scheme for mechanical manipulators - compensation of nonlinearity and decoupling control. ASME Journal of Dynamic Systems, Measurement, and Control 108:127-135.

[23] Hsia, T.C. 1988. University of California, Davis, California. Personal Communication during the 1988 IASTED Robotics and Automation Conference, Santa Barbara, California.

[24] Kalman, R.E. 1958. Design of self optimizing control system. Trans. ASME 80:468-478.

[25] Koivo, A. J., and Guo, T. H. 1983. Adaptive linear controller for robotic manipulators. IEEE Transactions on Automatic Control, AC-28, No. 2:233-242.

[26] Landau, Y. D. 1979. Adaptive Control : The Model Reference Approach. Marcel Dekker Inc., New York.

[27] Lee, C. S. G., Chung, M. J., and Lee, B. H. 1984. An approach of adaptive control for robot manipulators. Journal of Robotic Systems 1, No 1:27-57.

[28] Leininger, G. G. 1984. Self-tuning adaptive control of manipulators. In Advanced Software in Robotics. A. Danthine and M. Géradin, eds. Elsevier science publishers B. V., North Holland.

[29] Leu, M. C., and Hemati, N. 1986. Automated symbolic derivation of dynamic equations of motion for robotic manipulators. ASME Journal of Dynamic Systems Measurement, and Control. 108:172-179.

[30] Meritt, H. E. 1967. Hydraulic Control Systems. Wiley, New York.

[31] Norcross, R. J., Wang, J. C., McInnis, B. C., and Shieh, L. S. 1986. Pole placement methods for multivariable control of robotic manipulators. ASME Journal of Dynamic Systems, Measurements, and Control 108:340-345.

[32] Parks, P. C. 1966. Lyapunov redesign of model reference adaptive control. IEEE Trans. Auto. Control. AC-11:362-367.

[33] Paul, R. P. 1981. Robot Manipulators: Mathematics, Programming, and Control. The MIT Press, Cambridge, Massachussets.

[34] Popov, V. M. 1973. Hyperstability of Control Systems. Springer-Verlag, Berlin.

[35] Porter, B., and Tatnall, M. L. 1969. Performance characteristics of multivariable model-reference adaptive systemS synthesized by Liapunov's direct method. Int. J. Control, 10, No. 3:241-257.

[36] Porter, B., and Tatnall, M. L. 1970. Performance characteristics of an adaptive hydraulic servomechanism. Int. J. Control, 11, No. (5):41-757.

[37] Stoten, D. P. 1982. Discrete adaptive control of a manipulator arm. Optimal Control Applications & Methods, 3:423-433.

[38] Tomizuka, M., Horowitz, R., Anwar, G., and Jia, Y. L. 1988. Implementation of adaptive techniques for motion control of robotic manipulators. ASME Journal of Dynamic Systems, Measurement, and Control 110:62-69.

[39] Wellstead, P. E., Edmunds, J. M., Prager, D., and Zanker, P. 1979. Self-tuning pole/zero assignment regulators. Int. J. Control, 30, No. 1:1-26.

[40] Whitaker, H. P., and Rediess, H. A. 1970. A new model performance index for engineering design of flight control systems. AIAA J. Aircraft 7, No. 5:542-549.

9 APPENDIX

9.1 Automatic Generation of Dynamics using MACSYMA

The following user entries are required to run this program.

1) number of degrees of freedom

2) Type of joint associated with each link

3) Geometric parameters θ_i, d_i, a_i, and α_i

4) Link pseudo inertia matrix J_i

5) Link mass center position vector ${}^i r_i$ and,

6) Gravitational field vector g

The program listing for automatic generation of dynamic equations of motion of manipulators is given below.

```
WRITEFILE(ROBO_DYN);
" ************** HOW TO EXECUTE THIS PROGRAM  ****************
   AT THE DOLLAR SIGN PROMPT IN VAX TYPE.....
   MACSYMA      THEN AT THE FIRST PROMPT IN MACSYMA TYPE...
   BATCH(ROBO_DYNAM)SEMICOLON                                "$
"  TO GET OUT OF MACSYMA TYPE.....
      QUIT()SEMICOLON                                        "$
"                                                            "$
"  THIS PROGRAM DERIVES THE EQUATIONS OF MOTION OF A MANIPULATOR
```

```
   LINK USING THE LAGRANGIAN FORMULATION AND USING THE COMPUTER
   ALGEBRA PROGRAM CALLED *** MACSYMA ***                          "$
"                                                                  "$
"            BY   M.C. LEU
                & N. HEMATI                                        "$
"                                                                  "$
"


     MODIFIED BY S.ANANTHAKRISHNAN
                                                                   "$
"    INPUT=======>
                DOF : NO. OF DEGREES OF FREEDOM
                JOINT : THE TYPE OF JOINT ( 0 FOR REVOLUTE,
                        1 FOR PRISMATIC/TRANSLATIONAL              "$
"               D,A,ALPHA : LINK GEOMETRIC PARAMETERS
                R : LINK MASS CENTER POSITION VECTOR
                M : LINK MASS                                      "$
"               MOM : LINK PSEUDO INERTIA MATRIX
                LNK : LINK NUMBER
                GF : GRAVITATIONAL FIELD VECTOR                    "$
"    OUTPUT======>
                F[I] : GENERALIZED FORCE AT JOINT I
NOW... GENERATE THE T MATRICES                                     "$

   (
   PRINT(" ENTER THE NUMBER OF DEGREES OF FREEDOM"),DOF:READ(),
      FOR I THRU DOF DO(
        PRINT("TYPE 0 IF JOINT IS REVOLUTE AND 1 IF JOINT IS
        PRISMATIC"),
        PRINT(" "),JOINT:READ(),
        IF JOINT=0 THEN (
           PRINT("INPUT THE PARAMETERS OF THE REVOLUTE
           JOINT:D,A,ALPHA"),
           PRINT(" "),D[I]:READ(),AD[I]:READ(),ALF[I]:READ(),
           PRINT(" "),
           A[I]:MATRIX([COS(Q[I]),-SIN(Q[I])*COS(ALF[I]),
                       SIN(Q[I])*SIN(ALF[I]),AD[I]*COS(Q[I])],
                       [SIN(Q[I]),COS(Q[I])*COS(ALF[I]),
                       -COS(Q[I])*SIN(ALF[I]),AD[I]*SIN(Q[I])],
```

```
                        [0,SIN(ALF[I]),COS(ALF[I]),D[I]],
                        [0,0,0,1]))
        ELSE(PRINT("INPUT THE PARAMETERS OF THE PRISMATIC
             JOINT:THETA,A,ALPHA"),
             PRINT(" "),TH[I]:READ(),AD[I]:READ(),ALF[I]:READ(),
             PRINT(" "),
             A[I]:MATRIX([COS(TH[I]),-SIN(TH[I])*COS(ALF[I]),
                         SIN(TH[I])*SIN(ALF[I]),AD[I]*COS(TH[I])],
                         [SIN(TH[I]),COS(TH[I])*COS(ALF[I]),
                         -COS(TH[I])*SIN(ALF[I]),AD[I]*SIN(TH[I])],
                         [0,SIN(ALF[I]),COS(ALF[I]),Q[I]],
                         [0,0,0,1]))),
FOR I THRU DOF DO(
      IF I=1 THEN T[I]:A[I] ELSE T[I]:T[I-1].A[I]));
" TAKE THE FIRST DERIVATIVE OF THE
  T MATRICES W.R.T. THE JOINT VARIABLES. "$

  (FOR I THRU DOF DO(
               FOR J THRU DOF DO(
                 IF I>=J THEN(U[I,J] : DIFF(T[I],Q[J]))))));

" TAKE THE SECOND DERIVATIVES OF THE T MATRICES "$
   (FOR I THRU DOF DO(
                FOR J THRU DOF DO(
                   IF I>=J THEN(
                      FOR K:J THRU DOF DO(
                         IF I>= K THEN(W[I,J,K] :
                              DIFF(U[I,J],Q[K]))))))));

" INPUT THE MASS PROPERTIES "$

  (FOR I THRU DOF DO(
                PRINT("ENTER THE INERTIA MATRIX FOR LINK NO. ",I),
                MOM[I]:ENTERMATRIX(4,4),PRINT(" "),
                PRINT("ENTER THE CENTER OF MASS VECTOR FOR
                LINK NO. ",I),
                R[I]:ENTERMATRIX(4,1),PRINT(" ")),
               PRINT("ENTER THE GRAVITY FIELD VECTOR"),
               GF:ENTERMATRIX(4,1));
```

```
" DERIVE THE DI TERMS "$

    (PRINT("  "),PRINT("ENTER THE LINK NO. "),LNK:READ(),
                DI : 0,I:LNK,
                FOR PPI:I THRU DOF DO (
                    DI : DI + (((-M[PPI]*TRANSPOSE(GF)).U[PPI,I]).
                    R[PPI])),
                DD[I] : DI);

" DERVIE THE DIJ TERMS "$

  (I:LNK,
                  FOR J THRU DOF DO(
                   (IF J<I  THEN
                     MAXIJ : I
                      ELSE
                       MAXIJ : J),TRAC : 0,
                        FOR P:MAXIJ THRU DOF DO (
                        JTQI[P] : MOM[P].TRANSPOSE(U[P,I]),
                        FOR L THRU 4 DO (
                            TRAC1 : TRAC + ROW(U[P,J],L).
                            COL(JTQI[P],L),
                            TRAC : TRAC1)),
                    DIJ[I,J] : TRAC));


" DERIVE THE DIJK TERMS "$

  (I:LNK,
                FOR J THRU DOF DO (
                 FOR K:J THRU DOF DO (
                     DK[I,J,K] : 0,
                     IF J=I AND I>=K THEN
                      DK[I,J,K] : 0
                     ELSE(IF (J<I AND J<K AND K>=I) OR
                         (J>=I OR J>=K) THEN(
                        IF I>K THEN
                           (IF I>J THEN MAXIJK:I ELSE MAXIJK:J)
```

```
                    ELSE MAXIJK:K,
                    TRACEP : 0,
                    FOR PP:MAXIJK THRU DOF DO (
                        JTPI[PP] : MOM[PP].TRANSPOSE(U[PP,I]),
                        FOR LL THRU 4 DO (
                            TRACEP : TRACEP +
                                     ROW(W[PP,J,K],LL).
                               COL(JTPI[PP],LL))),
                    DK[I,J,K] : TRACEP)))));

" COLLECT THE DI, DIJ, DIJK TERMS TO OBTAIN
  THE EQUATIONS OF MOTION OF LINK I        "$

 (TRMDIJ : 0.,TRMDIJK : 0.,
        FOR J THRU DOF DO(
               TRMDIJ : TRMDIJ + DIJ[LNK,J]*DDQ[J],


             FOR K:J THRU DOF DO(
                  IF K=J THEN
                   TRMDIJK : TRMDIJK + DK[LNK,J,K]*DQ[J]*DQ[K]
                  ELSE IF J<LNK AND J<K THEN
                    (IF K>=LNK THEN
                      TRMDIJK : TRMDIJK +2*DK[LNK,J,K]*DQ[J]*DQ[K]
                     ELSE
                      TRMDIJK : TRMDIJK+2*DK[LNK,J,K]*DQ[J]*DQ[K])
                  ELSE
                      TRMDIJK : TRMDIJK +
                                2*DK[LNK,J,K]*DQ[J]*DQ[K])),
               F[LNK] : TRMDIJ + IA[LNK]*DDQ[LNK] + TRMDIJK +
                        DD[LNK]);

  F1[LNK] : TRIGSIMP( F[LNK] );

  F2[LNK] : TRIGREDUCE( F1[LNK] );
```

9.2 Generation of Equation of Motion for Base Rotation

```
(d2)                                robo_dyn

(c3) " ************** HOW TO EXECUTE THIS PROGRAM  ****************
   AT THE DOLLAR SIGN PROMPT IN VAX TYPE.....
   MACSYMA      THEN AT THE FIRST PROMPT IN MACSYMA TYPE...
   BATCH(ROBO_DYNAM)SEMICOLON                                    "$

(c4) "  TO GET OUT OF MACSYMA TYPE.....
      QUIT()SEMICOLON                                            "$

(c5) "
                                                "$

(c6) "  THIS PROGRAM DERIVES THE EQUATIONS OF MOTION OF A MANIPULATOR
   LINK USING THE LAGRANGIAN FORMULATION AND USING THE COMPUTER
   ALGEBRA PROGRAM CALLED *** MACSYMA ***                        "$

(c7) "
                                                "$

(c8) "            BY   M.C. LEU
               & N. HEMATI                                      "$

(c9) "
                                                "$

(c10) "

     MODIFIED BY S.ANANTHAKRISHNAN
                                                                 "$

(c11) "    INPUT=======>
               DOF : NO. OF DEGREES OF FREEDOM
               JOINT : THE TYPE OF JOINT ( 0 FOR REVOLUTE,
                       1 FOR PRISMATIC/TRANSLATIONAL             "$
```

```
(c12) "                D,A,ALPHA : LINK GEOMETRIC PARAMETERS
                R : LINK MASS CENTER POSITION VECTOR
                M : LINK MASS                                    "$

(c13) "                MOM : LINK PSEUDO INERTIA MATRIX
                LNK : LINK NUMBER
                GF : GRAVITATIONAL FIELD VECTOR                  "$

(c14) "    OUTPUT======>

                                      sin(q ) (- izz4 + iyy4 + ixx4)
                                           1
(d29)ddq(- sin(q )(- q sin(q )m4z4 - ------------------------------)
        1        1     3    1                        2

 - q sin(q )(- sin(q )m4z4 - cos(q )sin(q ) m4 y4 - q  sin(q ) m4)
    3     1         1             1      4           3      1

                                     cos(q ) (- izz4 + iyy4 + ixx4)
                                          1
 - cos(q ) (- q  cos(q ) m4 z4 - ------------------------------)
        1      3      1                        2

 - q cos(q )(- cos(q )m4 z4 + sin(q ) sin(q )m4y4 - q  cos(q ) m4)
    3     1         1              1       4           3      1

                                    cos(q ) sin(q ) (izz4 - iyy4 + ixx4)
                                         1       4
- cos(q)sin(q)(- q sin(q)m4y4 - ------------------------------------)
       1     4    3     1                          2

                      sin(q ) sin(q ) (izz4 - iyy4 + ixx4)
                           1       4
+ sin(q )sin(q )(------------------------------------ - q cos(q)m4y4)
       1      4                    2                      3     1

                                 2        2
                              sin (q ) cos (q ) (izz4 + iyy4 - ixx4)
  2    2        2    2             1        4
```

$$+q_3 \sin(q_1) m3 + q_3 \cos(q_1) m3 + \frac{\cos^2(q_1)\cos^2(q_4)(izz4 + iyy4 - ixx4) \quad \sin^2(q_1)(izz3 + iyy3 - ixx3)}{2 \qquad\qquad 2}$$

$$+ \frac{\cos^2(q_1)(izz3 + iyy3 - ixx3) \quad \sin^2(q_1)(- izz3 + iyy3 + ixx3)}{2 \qquad\qquad 2}$$

$$+ \frac{\cos^2(q_1)(- izz3 + iyy3 + ixx3) \quad \sin^2(q_1)(izz2 + iyy2 - ixx2)}{2 \qquad\qquad 2}$$

$$+ \frac{\cos^2(q_1)(izz2 + iyy2 - ixx2) \quad \sin^2(q_1)(- izz2 + iyy2 + ixx2)}{2 \qquad\qquad 2}$$

$$+ \frac{\cos^2(q_1)(- izz2 + iyy2 + ixx2) \quad \sin^2(q_1)(izz1 + iyy1 - ixx1)}{2 \qquad\qquad 2}$$

$$+ \frac{\cos^2(q_1)(izz1 + iyy1 - ixx1) \quad \sin^2(q_1)(izz1 - iyy1 + ixx1)}{2 \qquad\qquad 2}$$

```
     2
  cos (q ) (izz1 - iyy1 + ixx1)
        1
+ ----------------------------) + 2 dq  dq
                2                     1   3

(- sin(q ) (- sin(q ) m4 z4 - cos(q ) sin(q ) m4 y4 - q  sin(q ) m4)
        1          1               1       4           3      1

- cos(q ) (- cos(q ) m4 z4 + sin(q ) sin(q ) m4 y4 - q  cos(q ) m4)
       1          1               1       4           3      1

        2                 2
+ q  sin (q ) m3 + q  cos (q ) m3) + ddq
   3       1        3       1           3

(cos(q ) (- sin(q ) m4 z4 - cos(q ) sin(q ) m4 y4 - q  sin(q ) m4)
      1          1               1       4           3      1

- sin(q ) (- cos(q ) m4 z4 + sin(q ) sin(q ) m4 y4 - q  cos(q ) m4))
       1          1               1       4           3      1

    2
+ dq  (sin(q ) sin(q ) (- q  sin(q ) m4 y4
    4       1       4      3      1

  cos(q ) sin(q ) (izz4 - iyy4 + ixx4)
       1       4
- -----------------------------------)
                   2

                   sin(q ) sin(q ) (izz4 - iyy4 + ixx4)
                        1       4
+cos(q) sin(q)(---------------------------------- -q cos(q)m4 y4))
      1      4                    2                   3    1

+ ddq  (- sin(q ) cos(q ) (- q  sin(q ) m4 y4
     4         1       4      3      1
```

```
  cos(q ) sin(q ) (izz4 - iyy4 + ixx4)
       1       4
- ------------------------------------)
                   2

                   sin(q ) sin(q ) (izz4 - iyy4 + ixx4)
                        1       4
-cos(q)cos(q) (------------------------------------ -q cos(q)m4y4))
     1      4                    2                    3     1

+ 2 dq  dq  (- cos(q ) cos(q ) (- q  sin(q ) m4 y4
      1   4         1       4      3      1

  cos(q ) sin(q ) (izz4 - iyy4 + ixx4)
       1       4
- ------------------------------------)
                   2

                   sin(q ) sin(q ) (izz4 - iyy4 + ixx4)
                        1       4
+sin(q)cos(q) (------------------------------------ - q cos(q)m4y4)
      1     4                    2                     3     1

     2
  sin (q ) cos(q ) sin(q ) (izz4 + iyy4 - ixx4)
        1       4       4
- ------------------------------------------
                        2

     2
  cos (q ) cos(q ) sin(q ) (izz4 + iyy4 - ixx4)
        1       4       4
- ------------------------------------------) + ddq  ia
                        2                            1   1
```

```
(c30) F1[LNK] : TRIGSIMP( F[LNK] );

Batching the file /macsyma/share/trgsmp.mac
```

```
(c30) /*-*-macsyma-*-*/

eval_when([translate],transcompile:true,compgrind:true)$

(c30) eval_when([translate,batch,demo],
          matchdeclare(a,true))$

(c30) defrule(trigrule1,tan(a),sin(a)/cos(a))$

(c30) defrule(trigrule2,sec(a),1/cos(a))$

(c30) defrule(trigrule3,csc(a),1/sin(a))$

(c30) defrule(trigrule4,cot(a),cos(a)/sin(a))$

(c30) defrule(htrigrule1,tanh(a),sinh(a)/cosh(a))$

(c30) defrule(htrigrule2,sech(a),1/cosh(a))$

(c30) defrule(htrigrule3,csch(a),1/sinh(a))$

(c30) defrule(htrigrule4,coth(a),cosh(a)/sinh(a))$

(c30) TRIGSIMP(x):=trigsimp3(radcan(apply1(x,trigrule1,
    trigrule2,trigrule3,
    trigrule4,htrigrule1,htrigrule2,htrigrule3,htrigrule4)))$

(c30) TRIGSIMP3(EXPN) :=
   (EXPN: TOTALDISREP(EXPN),
   RATSIMP(TRIGSIMP1(NUM(EXPN))/TRIGSIMP1(DENOM(EXPN)))) $

(c30) define_variable(bestlength,0,fixnum)$

%/macsyma/transl/trmode.o being loaded.

(c30) define_variable(trylength,0,fixnum)$

(c30) TRIGSIMP1(EXPN) := BLOCK(
   [LISTOFTRIGSQ, BESTLENGTH:999999, TRYLENGTH:TRYLENGTH],
```

```
   LISTOFTRIGSQ: LISTOFTRIGSQQ(EXPN),
   IF LISTOFTRIGSQ#[] THEN IMPROVE(EXPN, LISTOFTRIGSQ),
   EXPN) $

(c30) IMPROVE(SUBSOFAR, LISTOFTRIGSQ) :=
   IF LISTOFTRIGSQ=[] THEN (
      TRYLENGTH: EXPNLENGTH(SUBSOFAR),
      IF TRYLENGTH<BESTLENGTH THEN (
         EXPN: SUBSOFAR,
         BESTLENGTH: TRYLENGTH))
   ELSE (IMPROVE(SUBSOFAR, REST(LISTOFTRIGSQ)),
      FOR ALT IN FIRST(LISTOFTRIGSQ) DO
         IMPROVE(RATSUBST(
            IF INPART(ALT,0)='SIN THEN 1-COS(INPART(ALT,1))^2
            ELSE IF PIECE='COS THEN 1-SIN(INPART(ALT,1))^2
            ELSE IF PIECE='SINH THEN COSH(INPART(ALT,1))^2-1
            ELSE 1+SINH(INPART(ALT,1))^2,
               ALT^2, SUBSOFAR), REST(LISTOFTRIGSQ))) $

(c30) LISTOFTRIGSQQ(EXPN) :=
   IF ATOM(EXPN) THEN []
   ELSE BLOCK([INFLAG, ANS],
      IF INPART(EXPN,0)="^" AND INTEGERP(INPART(EXPN,2))
         AND PIECE>=2 THEN
         IF ATOM(EXPN:INPART(EXPN,1)) THEN RETURN([])
         ELSE IF MEMBER(INPART(EXPN,0),'[SIN,COS,SINH,COSH])
            THEN RETURN([[EXPN]]),
   INFLAG:TRUE,
      ANS:[],
      FOR ARG IN EXPN DO
         ANS: SPECIALUNION(LISTOFTRIGSQQ(ARG), ANS),
      RETURN(ANS)) $

(c30) SPECIALUNION(LIST1,LIST2) :=
   IF LIST1=[] THEN LIST2
   ELSE IF LIST2=[] THEN LIST1
   ELSE BLOCK([ALTERNATES],
      ALTERNATES: FIRST(LIST1),
      FOR ALT IN ALTERNATES DO LIST2:
```

```
         IF INPART(ALT,0)='SIN THEN UPDATE(ALT,'COS)
         ELSE IF PIECE='COS THEN UPDATE(ALT,'SIN)
         ELSE IF PIECE='SINH THEN UPDATE(ALT,'COSH)
         ELSE UPDATE(ALT,'SINH),
      RETURN(SPECIALUNION(REST(LIST1),LIST2))) $

(c30) UPDATE(FORM, COMPLEMENT) := BLOCK(
   [ANS],
   COMPLEMENT: APPLY(COMPLEMENT,[INPART(FORM,1)]),
   ANS: FOR ELEMENT IN LIST2 DO
      IF MEMBER(FORM, ELEMENT) THEN RETURN('FOUND)
      ELSE IF MEMBER(COMPLEMENT,ELEMENT) THEN RETURN(
         CONS([FORM,COMPLEMENT], DELETE(ELEMENT,LIST2))),
   IF ANS='FOUND
   THEN LIST2
   ELSE IF ANS='DONE
        THEN CONS([FORM],LIST2)
        ELSE ANS) $

(c30) EXPNLENGTH(EXPR) :=
   IF ATOM(EXPR)
   THEN 1
   ELSE 1 + ARGSLENGTH(ARGS(EXPR))$

(c30) ARGSLENGTH(ARGS) :=
  APPLY("+",MAP('EXPNLENGTH,ARGS))$

Batching done.
```

$$(d30)\ (2\ ddq_1\ q_3 + 2\ dq_1\ dq_3)\ m4\ z4 + ((-\ q_3\ dq_4^2 - ddq_3)\ \sin(q_4)$$

$$+\ q_3\ ddq_4\ \cos(q_4))\ m4\ y4 + (ddq_1\ q_3^2 + 2\ dq_1\ dq_3\ q_3)\ m4$$

$$+\ (ddq\ q^2 + 2\ dq\ dq\ q)\ m3 + ddq\ izz1$$

```
      1  3       1   3  3          1

           2
 + (ddq  cos (q ) - 2 dq  dq  cos(q ) sin(q )) iyy4+ddq iyy3+ddq iyy2
      1       4        1   4      4       4           1        1

                                          2
 + (2 dq  dq  cos(q ) sin(q ) - ddq  cos (q ) + ddq ) ixx4 + ddq  ia
       1   4      4       4       1       4       1            1   1

(c31) F2[LNK] : TRIGREDUCE( F1[LNK] );

                                                   2
(d31) 2 ddq  q  m4 z4 + 2 dq  dq  m4 z4 - q  dq  sin(q ) m4 y4
          1  3             1   3          3   4      4

                                                  2
 - ddq  sin(q ) m4 y4 + q  ddq  cos(q )m4y4 +ddq q  m4+2dq dq q  m4
     3      4           3    4      4          1 3       1  3 3

        2
 + ddq  q  m3 + 2 dq  dq  q  m3 + ddq  izz1 - dq  dq  sin(2 q ) iyy4
     1  3          1   3  3         1          1   4        4

   ddq  cos(2 q ) iyy4   ddq  iyy4
      1        4            1
 + ------------------- + --------- + ddq  iyy3 + ddq  iyy2
            2                2          1           1

                              ddq  cos(2 q ) ixx4   ddq  ixx4
                                 1        4            1
 + dq  dq  sin(2 q ) ixx4 - ------------------- + ---------+ddq  ia
    1   4        4                   2                 2        1   1


(d32)                                BATCH DONE

(c33)
Break Entering lisp:
```

<1>:

9.3 Program Listing for Multiple Axis Discrete-time Adaptive Control

```
C          DOUBLE PRECISION RUNKA KUTTA ROUTINE WITH
C           PROVISIONS FOR DISCRETE ADAPTIVE CONTROL


C...DEFINITIONS
        IMPLICIT REAL*8 (a-H,o-z)
        REAL*8 DELTAT
        INTEGER*4 NPLOT
        DIMENSION Y(30),DY(30),PA(30)
        DIMENSION Y1(30),RK0(30),RK1(30),RK2(30),YLAST(30)
        VIRTUAL DATA(512,31)

        COMMON/FCTCOM/PA
        COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1  NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
        COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
        COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1  ILOOP
        COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
        COMMON/CONTIN/N,NP
        COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1  PCOV4(30,30)
        COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
        COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
        COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
        COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,NPARAM2
     1   ,NPARAM3,NPARAM4

C...NP IS NUMBER OF PARAMETERS TO BE INPUT BY USER,
C...NPARAM IS NUMBER OF PARAMETERS TO BE IDENTIFIED


C...INITIAL INPUTS
        TYPE *,' INPUT THE DIMENSION OF THE SYSTEM'
        ACCEPT *,N
        TYPE *,' INPUT THE NUMBER OF PARAMETERS USED'
        ACCEPT *,NP
```

```
      CALL FCT(T,Y,DY)

998   TYPE *,'-----------------------------------------------------
     1 ------'
      TYPE *,'A DISCRETE TIME ADAPTIVE CONTROL SYSTEM WITH A
     1 CONTINUOUS'
      TYPE *,'PLANT, USING A DOUBLE PRECISION RUNGA KUTTA
     1 NUMERICAL '
      TYPE *,'INTEGRATION PROGRAM'
      TYPE *,'-----------------------------------------------------
     1 ----'


C...TIMING INPUTS
      TYPE *,' INPUT THE STARTING TIME IN SECONDS:'
      ACCEPT *,TSTART
      TYPE *,' INPUT THE FINAL TIME IN SECONDS:'
      ACCEPT *,TSTOP
      TYPE *,' INPUT THE DISCRETE CONTROL TIME STEP SIZE IN
     1 SECONDS:'
      ACCEPT *,CNTLDT
      TYPE *,' INPUT THE NUMBER OF CALLS TO RK FOR EACH CONTROL
     1 STEP'
      ACCEPT *,INTCTL

      DELTAT=CNTLDT/FLOAT(INTCTL) ! CONTINUOUS TIME SAMPLING RATE

      TSAMPL=(TSTOP-TSTART)/512.      ! DATA STORAGE SAMPLING RATE

      TYPE *,' PRESENT DATA STORAGE PERIOD IS ',TSAMPL
      TYPE *,' INPUT [1 FOR LARGER PERIOD/ 0 TO KEEP STORAGE
     1 PERIOD]'
      ACCEPT *,JUNK
      IF(JUNK.EQ.1) THEN
              TYPE *,' INPUT STORAGE PERIOD (MUST BE LARGER!) '
              ACCEPT *,TTTT
              IF(TTTT.GE.TSAMPL) TSAMPL=TTTT
              TYPE *,' NEW STORAGE PERIOD IS',TSAMPL
```

```
        ENDIF
        NPLOT=((TSTOP-TSTART)/DELTAT)
        TYPE *,' NUMBER OF STEPS EVALUATED =',NPLOT


C...CONTINUOUS TIME PARAMETER INPUTS
        IF(NP.GE.1) THEN
                TYPE *,' INPUT VALUES OF THE PARAMETERS'
                DO 2222 I=1,NP
                TYPE *,' INPUT PARAMETER ',I
2222            ACCEPT *,PA(I)
        ENDIF


C...CONTINUOUS TIME INITIAL VALUES
        TYPE *,' INPUT INITIAL VALUES OF THE VARIABLES'
        DO 2221 I=1,N
        TYPE 2555,I
2555    FORMAT(' INPUT Y(',I2,'):')
2221    ACCEPT *,Y(I)



C...DISCRETE TIME INITIAL VALUES
        CALL INITOV

C...SET UP RECURSIVE ESTIMATION ROUTINE
        CALL SETUP


C...INITIALIZING STORAGE ARRAY
        TDATA=TSTART
        DO 98 I=1,30
98      DATA(1,I)=Y(I)
        DATA(1,31)=TSTART
        JDATA=2

        ICNTRL=INTCTL           ! FIRST STEP ALWAYS CALLS TO CONTROL
        ILOOP=1         ! FIRST CALL INITIALIZES ESTIMATION ROUTINES
```

```
C...START CONTINUOUS SYSTEM SIMULATION
        DO 1 T=TSTART,TSTOP,DELTAT
        ICNTRL= ICNTRL+1
        IF(ICNTRL.GE.INTCTL) THEN       ! TIME FOR DISCRETE CONTROL
                ICNTRL=0
                CALL MOVBAK(T,Y)
                CALL CONTRL(T,Y)
C                       CONTRL IS THE CONTROL ROUTINE, PASSES BACK
C                       CONTROL VALUES THROUGH THE PARAM VECTOR
C                   THESE VALUES CAN BE PLOTTED BY STORING IN Y(?)
        ILOOP=100.
        ENDIF

C...CLASSIC RUNGA KUTTA FROM HERE!
        T1=T+DELTAT/2.0D0
        T2=T+DELTAT

        CALL FCT(T,Y,DY)

        DO 2 I=1,N
        RK0(I)=DY(I)*DELTAT
2       Y1(I)=Y(I)+RK0(I)/2.0D0
        CALL FCT(T1,Y1,DY)

        DO 3 I=1,N
        RK1(I)=DY(I)*DELTAT
3       Y1(I)=Y(I)+RK1(I)/2.0D0
        CALL FCT(T1,Y1,DY)

        DO 4 I=1,N
        RK2(I)=DY(I)*DELTAT
4       Y1(I)=Y(I)+RK2(I)
        CALL FCT(T2,Y1,DY)

        DO 5 I=1,N
5       DY(I)=DY(I)*DELTAT
```

```
C  RK3 AND DY ARE THE SAME

        DO 6 I=1,N
6       Y(I)=Y(I)+(RK0(I)+DY(I)+2.0D0*(RK1(I)+RK2(I)))/6.0D0

C...NEW RESULT IS Y(I) AT T+DELTAT
C...CALL FCT ONE LAST TIME TO GET NON-INTEGRATED VALUES
        CALL FCT(T,Y,DY)
C...END OF CLASSIC RUNGA KUTTA


C...STORE DATA IN ARRAY
        IF(JDATA.GT.512) GO TO 1
        IF(NPLOT.LT.512) THEN
                DO 10 I=1,30
10              DATA(JDATA,I)=Y(I)
                DATA(JDATA,31)=T+DELTAT
                JDATA=JDATA+1
        ELSE
                IF(T.GT.TDATA) THEN
                        DO 22 I=1,30
22                      DATA(JDATA,I)=Y(I)
                        TDATA=TDATA+TSAMPL
                        DATA(JDATA,31)=T+DELTAT
                        JDATA=JDATA+1
                ENDIF
        ENDIF


C...NEXT CONTINUOUS TIME STEP INCREMENT!
1       CONTINUE


C...OUTPUT RESULTS
999     TYPE *,' INPUT:'
        TYPE *,'        0) FOR PLOT OF DATA'
        TYPE *,'        1) FOR PRINOUT OF DATA'
        TYPE *,'        2) TO RUN PROGRAM AGAIN'
        TYPE *,'        3) TO STORE DATA IN A FILE'
```

```
      ACCEPT *,RJUNK
      JUNK=INT(RJUNK+.0001)
      IF(JUNK.LT.0.OR.JUNK.GT.3) GO TO 999
      IF(JUNK.EQ.1) THEN
              CALL PRINT(NPLOT,DATA)
      ENDIF
      IF(JUNK.EQ.0) THEN
              CALL PLOT (NPLOT,DATA)
      ENDIF
      IF(JUNK.EQ.3) THEN
              CALL WRITE(NPLOT,DATA)
      ENDIF
      IF(JUNK.EQ.2) GO TO 998


      GO TO 999
      END




      SUBROUTINE SETUP
C     SET UP ESTIMATION ROUTINES
      IMPLICIT REAL*8 (a-H,o-z)

      COMMON/FCTCOM/PA
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4
```

```
C... SELECTS THE PARAMETER ESTIMATE ROUTINE
1999    TYPE *,' ------------------------------------------------'
        TYPE *,' '
        TYPE *,' SELECT ESTIMATION ALGORITHM:'
        TYPE *,' 2) OR 3) FOR MIMO SYSTEM    '
        TYPE *,' '
        TYPE *,'        1) LEAST SQUARES'
        TYPE *,'        2) LEAST SQUARES WITH COVARIANCE RESETTING'
        TYPE *,'        3) LEAST SQUARES WITH COVARIANCE FORGETTING'
        TYPE *,'        4) LEAST SQUARES WITH COVARIANCE ADDITION'
        TYPE *,'        5) PROJECTION ALGORITHM'
        TYPE *,'        6) ORTHOGONAL PROJECTION ALGORITHM'
        TYPE *,'        7) SPECIFIC ESTIMATION ROUTINE'
        TYPE *,' '
        TYPE *,' ------------------------------------------------'
        ACCEPT *,ISEL
        IF(ISEL.LT.1.OR.ISEL.GT.7) GO TO 1999
        IEST=ISEL
        IF(IEST.EQ.7) THEN
                CALL INITID  ! PREPROGRAMMED RECURSIVE IDENT METHOD
        ELSE
                TYPE *,'  '
                type *,'          -1     -1     -1    '
                TYPE *,' Yi = -A(q ) Y + q  B( q ) U'
                TYPE *,' INPUT THE ESTIMATED ORDER OF POLY A'
                ACCEPT *,NA
                TYPE *,' INPUT THE ESTIMATED ORDER OF POLY B'
                ACCEPT *,NB
                TYPE *,' INPUT THE ESTIMATED DELAY IN B'
                ACCEPT *,ND

                NPARAM1=NA+NA+NA+NA+NB
                NPARAM2=NA+NB
                NPARAM3=NA+NA+NB
                NPARAM4=NA+NA+NB
```

```
          TYPE *,'  INPUT INITIAL ESTIMATES FOR A PARAMETERS:'
          TYPE*, '  AXIS 1                                   '
          DO 101 I=1,NA
          TYPE *,' A  PARAM1(',I,')='
101       ACCEPT *,PARAM1(I)

          DO 102 I=1,NA
          TYPE *,' A  PARAM1(',NA+I,')='
102       ACCEPT *,PARAM1(NA+I)

          DO 103 I=1,NA
          TYPE *,' A  PARAM1(',NA+NA+I,')='
103       ACCEPT *,PARAM1(NA+NA+I)

          DO 104 I=1,NA
          TYPE *,' A  PARAM1(',NA+NA+NA+I,')='
104       ACCEPT *,PARAM1(NA+NA+NA+I)

          DO 105 I=1,NB
          INA=I+NA+NA+NA+NA
          TYPE *,' B  PARAM1(',INA,')='
105       ACCEPT *,PARAM1(INA)
          TYPE *,'  '

          TYPE *,'  INPUT INITIAL ESTIMATES FOR A PARAMETERS:'
          TYPE*, '  AXIS 2                                   '
          DO 201 I=1,NA
          TYPE *,' A  PARAM2(',I,')='
201       ACCEPT *,PARAM2(I)


          DO 205 I=1,NB
          INA=I+NA
          TYPE *,' B  PARAM2(',INA,')='
205       ACCEPT *,PARAM2(INA)
          TYPE *,'  '

          TYPE *,'  INPUT INITIAL ESTIMATES FOR A PARAMETERS:'
          TYPE*, '  AXIS 3                                   '
```

```
              DO 301 I=1,NA
              TYPE *,' A  PARAM3(',I,')='
301           ACCEPT *,PARAM3(I)

              DO 302 I=1,NA
              TYPE *,' A  PARAM3(',NA+I,')='
302           ACCEPT *,PARAM3(NA+I)


              DO 305 I=1,NB
              INA=I+NA+NA
              TYPE *,' B  PARAM3(',INA,')='
305           ACCEPT *,PARAM3(INA)
              TYPE *,'  '

              TYPE *,'  INPUT INITIAL ESTIMATES FOR A PARAMETERS:'
              TYPE*, '  AXIS 4                                   '
              DO 401 I=1,NA
              TYPE *,' A  PARAM4(',I,')='
401           ACCEPT *,PARAM4(I)

              DO 402 I=1,NA
              TYPE *,' A  PARAM4(',NA+I,')='
402           ACCEPT *,PARAM4(NA+I)


              DO 405 I=1,NB
              INA=I+NA+NA
              TYPE *,' B  PARAM4(',INA,')='
405           ACCEPT *,PARAM4(INA)
              TYPE *,'  '


              IF(IEST.EQ.1) THEN
                      TYPE *,' INPUT INITIAL COVARIANCE MAGNITUDE '
                      ACCEPT *,COV
              ENDIF

              IF(IEST.EQ.2)THEN
```

```
                TYPE *,' INPUT INITIAL COVARIANCE MAGNITUDE '
                ACCEPT *,COV
                TYPE *,' INPUT NUMBER OF RESET SAMPLES '
                ACCEPT *,IRESET
        ENDIF

        IF(IEST.EQ.3) THEN
                TYPE *,' INPUT INITIAL COVARIANCE MAGNITUDE '
                ACCEPT *,COV
                TYPE *,' INPUT FORGETTING FACTOR'
                ACCEPT *,FORGET
        ENDIF

        IF(IEST.EQ.4) THEN
                TYPE *,' INPUT INITIAL COVARIANCE MAGNITUDE '
                ACCEPT *,COV
                TYPE *,' INPUT NOISE COVARIANCE MAGNITUDE '
                ACCEPT *,COVNOI
        ENDIF

        TYPE *,'  '
        TYPE *,'-----------------------------------------
     1  ----'


    ENDIF
    RETURN
    END



    SUBROUTINE LSTSQR
C...PERFORMS THE LEAST SQUARES ALGORITHM FOR ARIMA PROCESS
C...IDENTIFICATION
    IMPLICIT REAL*8 (a-H,o-z)

    COMMON/FCTCOM/PA
    COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
   1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
```

```
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4


C...INITIALIZATION OF COVARIANCE MATRIX
      IMAX=NPARAM
      IF(ILOOP.EQ.1) THEN
              DO 1 J=1,IMAX
              DO 2 K=1,IMAX
2             PCOV(J,K)=0.
1             PCOV(J,J)=COV
      ENDIF



      sum=1.
      do 200 j=1,imax
      store(j)=0
      do 201 k=1,imax
201   store(j)=store(j)+PCOV(j,k)*phi(k)
200   sum=sum+store(j)*phi(j)

c...note store = PCOV(t-2) * Phi (t-1)



      do 202 j=1,imax
202   param(j)=param(j)+store(j)*error/sum

C
```

```
C
C


c...updating matrix p

        do 203 j=1,imax
        do 203 k=1,imax
203     PCOV(j,k)=PCOV(j,k)-store(j)*store(k)/sum

        CALL TRACE

        RETURN
        END





        SUBROUTINE LSTRES
C...PERFORMS THE LEAST SQUARES ALGORITHM FOR ARIMA PROCESS
C...IDENTIFICATION
C...WITH COVARIANCE RESETTING
        IMPLICIT REAL*8 (a-H,o-z)

        COMMON/FCTCOM/PA
        COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1  NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
        COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
        COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1  ILOOP
        COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
        COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1  PCOV4(30,30)
        COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
        COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
        COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
        COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
```

```
     1  NPARAM2,NPARAM3,NPARAM4


C...RE-INITIALIZATION OF COVARIANCE MATRIX
        IMAX1=NPARAM1
        IMAX2=NPARAM2
        IMAX3=NPARAM3
        IMAX4=NPARAM4


        IF(ILOOP.EQ.1) ICOUNT=IRESET
        IF(ICOUNT.GE.IRESET) THEN


                DO 101 J=1,IMAX1
                DO 102 K=1,IMAX1
102             PCOV1(J,K)=0.
101             PCOV1(J,J)=COV
                ICOUNT=0

                DO 201 J=1,IMAX2
                DO 202 K=1,IMAX2
202             PCOV2(J,K)=0.
201             PCOV2(J,J)=COV
                ICOUNT=0

                DO 301 J=1,IMAX3
                DO 302 K=1,IMAX3
302             PCOV3(J,K)=0.
301             PCOV3(J,J)=COV
                ICOUNT=0

                DO 401 J=1,IMAX4
                DO 402 K=1,IMAX4
402             PCOV4(J,K)=0.
401             PCOV4(J,J)=COV
                ICOUNT=0
```

```
        ENDIF
        ICOUNT=ICOUNT+1

C...    calculating denominator term

        sum=1.
        do 1200 j=1,imax1
        store(j)=0
        do 1201 k=1,imax1
1201    store(j)=store(j)+PCOV1(j,k)*phiM1(k)
1200    sum=sum+store(j)*phiM1(j)

c...note store = PCOV(t-2) * Phi (t-1)

        do 1202 j=1,imax1
1202    param1(j)=param1(j)+store(j)*error1/sum

c...updating matrix p

        do 1203 j=1,imax1
        do 1203 k=1,imax1
1203    PCOV1(j,k)=PCOV1(j,k)-store(j)*store(k)/sum


C...    calculating denominator term

        sum=1.
        do 2200 j=1,imax2
        store(j)=0
        do 2201 k=1,imax2
2201    store(j)=store(j)+PCOV2(j,k)*phiM2(k)
2200    sum=sum+store(j)*phiM2(j)

c...note store = PCOV(t-2) * Phi (t-1)

        do 2202 j=1,imax2
2202    param2(j)=param2(j)+store(j)*error2/sum
```

```
c...updating matrix p

        do 2203 j=1,imax2
        do 2203 k=1,imax2
2203    PCOV2(j,k)=PCOV2(j,k)-store(j)*store(k)/sum


C...    calculating denominator term

        sum=1.
        do 3200 j=1,imax3
        store(j)=0
        do 3201 k=1,imax3
3201    store(j)=store(j)+PCOV3(j,k)*phiM3(k)
3200    sum=sum+store(j)*phiM3(j)

c...note store = PCOV(t-2) * Phi (t-1)

        do 3202 j=1,imax3
3202    param3(j)=param3(j)+store(j)*error3/sum

c...updating matrix p

        do 3203 j=1,imax3
        do 3203 k=1,imax3
3203    PCOV3(j,k)=PCOV3(j,k)-store(j)*store(k)/sum


C...    calculating denominator term

        sum=1.
        do 4200 j=1,imax4
        store(j)=0
        do 4201 k=1,imax4
4201    store(j)=store(j)+PCOV4(j,k)*phiM4(k)
4200    sum=sum+store(j)*phiM4(j)

c...note store = PCOV(t-2) * Phi (t-1)
```

```
      do 4202 j=1,imax4
4202  param4(j)=param4(j)+store(j)*error4/sum

c...updating matrix p

      do 4203 j=1,imax4
      do 4203 k=1,imax4
4203  PCOV4(j,k)=PCOV4(j,k)-store(j)*store(k)/sum

      CALL TRACE
      RETURN
      END



      SUBROUTINE LSTFOR
C...PERFORMS THE LEAST SQUARES ALGORITHM FOR ARIMA PROCESS
C...IDENTIFICATION
      IMPLICIT REAL*8 (a-H,o-z)

      COMMON/FCTCOM/PA
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4
```

```
C...INITIALIZATION OF COVARIANCE MATRIX
        IMAX1=NPARAM1
        IMAX2=NPARAM2
        IMAX3=NPARAM3
        IMAX4=NPARAM4

        IF(ILOOP.EQ.1) THEN

                DO 101 J=1,IMAX1
                DO 102 K=1,IMAX1
102             PCOV1(J,K)=0.
101             PCOV1(J,J)=COV

                DO 201 J=1,IMAX2
                DO 202 K=1,IMAX2
202             PCOV2(J,K)=0.
201             PCOV2(J,J)=COV

                DO 301 J=1,IMAX3
                DO 302 K=1,IMAX3
302             PCOV3(J,K)=0.
301             PCOV3(J,J)=COV

                DO 401 J=1,IMAX4
                DO 402 K=1,IMAX4
402             PCOV4(J,K)=0.
401             PCOV4(J,J)=COV

        ENDIF

C...    calculating denominator term

        sum=FORGET
        do 1200 j=1,imax1
        store(j)=0
        do 1201 k=1,imax1
1201    store(j)=store(j)+PCOV1(j,k)*phiM1(k)
1200    sum=sum+store(j)*phiM1(j)
```

```
c...note store = PCOV(t-2) * Phi (t-1)

        do 1202 j=1,imax1
1202    param1(j)=param1(j)+store(j)*error1/sum

c...updating matrix p

        do 1203 j=1,imax1
        do 1203 k=1,imax1
1203    PCOV1(j,k)=(PCOV1(j,k)-store(j)*store(k)/sum)/FORGET



C...    calculating denominator term

        sum=FORGET
        do 2200 j=1,imax2
        store(j)=0
        do 2201 k=1,imax2
2201    store(j)=store(j)+PCOV2(j,k)*phiM2(k)
2200    sum=sum+store(j)*phiM2(j)

c...note store = PCOV(t-2) * Phi (t-1)

        do 2202 j=1,imax2
2202    param2(j)=param2(j)+store(j)*error2/sum

c...updating matrix p

        do 2203 j=1,imax2
        do 2203 k=1,imax2
2203    PCOV2(j,k)=(PCOV2(j,k)-store(j)*store(k)/sum)/FORGET



C...    calculating denominator term

        sum=FORGET
```

```
        do 3200 j=1,imax3
        store(j)=0
        do 3201 k=1,imax3
3201    store(j)=store(j)+PCOV3(j,k)*phiM3(k)
3200    sum=sum+store(j)*phiM3(j)

c...note store = PCOV(t-2) * Phi (t-1)

        do 3202 j=1,imax3
3202    param3(j)=param3(j)+store(j)*error3/sum

c...updating matrix p

        do 3203 j=1,imax3
        do 3203 k=1,imax3
3203    PCOV3(j,k)=(PCOV3(j,k)-store(j)*store(k)/sum)/FORGET



C...    calculating denominator term

        sum=FORGET
        do 4200 j=1,imax4
        store(j)=0
        do 4201 k=1,imax4
4201    store(j)=store(j)+PCOV4(j,k)*phiM4(k)
4200    sum=sum+store(j)*phiM4(j)

c...note store = PCOV(t-2) * Phi (t-1)

        do 4202 j=1,imax4
4202    param4(j)=param4(j)+store(j)*error4/sum

c...updating matrix p

        do 4203 j=1,imax4
        do 4203 k=1,imax4
4203    PCOV4(j,k)=(PCOV4(j,k)-store(j)*store(k)/sum)/FORGET
```

```
      CALL TRACE
      RETURN
      END


      SUBROUTINE LSTADD
C...PERFORMS THE LEAST SQUARES ALGORITHM FOR ARIMA PROCESS
C...IDENTIFICATION
C...WITH NOISE COVARIANCE ADDITION ,INITIALIZATION OF COVARIANCE
C...MATRIX
      IMPLICIT REAL*8 (a-H,o-z)

      COMMON/FCTCOM/PA
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4



      IMAX=NPARAM
```

```
        IF(ILOOP.EQ.1) THEN
                DO 1 J=1,IMAX
                DO 2 K=1,IMAX
2               PCOV(J,K)=0.
1               PCOV(J,J)=COV+COVNOI
        ENDIF

C...    calculating denominator term

        sum=1.
        do 200 j=1,imax
        store(j)=0
        do 201 k=1,imax
201     store(j)=store(j)+PCOV(j,k)*phi(k)
200     sum=sum+store(j)*phi(j)

c...note store = PCOV(t-2) * Phi (t-1)

        do 202 j=1,imax
202     param(j)=param(j)+store(j)*error/sum

c...updating matrix p

        do 203 j=1,imax
        do 203 k=1,imax
203     PCOV(j,k)=PCOV(j,k)-store(j)*store(k)/sum
        DO 204 J=1,IMAX
204     PCOV(J,J)=PCOV(J,J)+COVNOI

        CALL TRACE
        RETURN
        END



        SUBROUTINE ORTHPR
C...PERFORMS THE ORTHOGONAL PROJECTION ALGORITHM
        IMPLICIT REAL*8 (a-H,o-z)
```

```
      COMMON/FCTCOM/PA
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4



      imax=NPARAM
C... INITIAL COVARIANCE MATRIX IS IDENTITY
      IF(ILOOP.EQ.1) THEN
              DO 111 J=1,imax
              DO 111 I=1,imax
              PCOV(I,J)=0.
111           PCOV(J,J)=1.
              type *,' initializing covariance'
      ENDIF

C...  calculating denominator term

      sum=0.
      do 200 j=1,imax
      store(j)=0
      do 201 k=1,imax
201   store(j)=store(j)+pCOV(j,k)*phi(k)
200   sum=sum+store(j)*phi(j)

C....BUG OUT IF SUM IS TOO SMALL
      IF(SUM.eq.0.0) RETURN
```

```
c...note store = PCOV(t-2) * Phi (t-1)

        do 202 j=1,imax
202     param(j)=param(j)+store(j)*error/sum

c...updating matrix p

        do 203 j=1,imax
        do 203 k=1,imax
203     pCOV(j,k)=pCOV(j,k)-store(j)*store(k)/sum


        CALL TRACE
        RETURN
        END

        SUBROUTINE PROJCT
C...PERFORMS THE CLASSICAL PROJECTION ALGORITHM
        IMPLICIT REAL*8 (a-H,o-z)

        COMMON/FCTCOM/PA
        COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1  NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
        COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
        COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1  ILOOP
        COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
        COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1  PCOV4(30,30)
        COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
        COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
        COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
        COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1  NPARAM2,NPARAM3,NPARAM4


        imax=NPARAM
```

```
      PHISQR=0.
      DO 1 I=1,4
1     PHISQR=PHI(I)*PHI(I)
      if(phisqr.eq.0.0) return    ! bad input data. ignore it
      DO 104 J=1,imax
      PARAM(J)=PARAM(J)+PHI(J)*ERROR/PHISQR
104   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TRACE
      IMPLICIT REAL*8 (a-H,o-z)
      COMMON/FCTCOM/PA
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4
```

```
      SUM=0.
      DO 101 I=1,NPARAM1
101   SUM=SUM+PCOV1(I,I)
      TRACEP1=SUM
```

```
      SUM=0.
      DO 201 I=1,NPARAM2
201   SUM=SUM+PCOV2(I,I)
      TRACEP2=SUM


      SUM=0.
      DO 301 I=1,NPARAM3
301   SUM=SUM+PCOV3(I,I)
      TRACEP3=SUM


      SUM=0.
      DO 401 I=1,NPARAM4
401   SUM=SUM+PCOV4(I,I)
      TRACEP4=SUM


      RETURN
      END


      SUBROUTINE MOVBAK(T,Y)
      IMPLICIT REAL*8 (a-H,o-z)
      DIMENSION Y(30),PA(30)

      COMMON/FCTCOM/PA
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/CONTIN/N,NP
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
```

```
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4


C     RECURSIVELY INCREMENT OLD DATA

      DO 1 I=1,30
      DO 1 J=9,1,-1
1     YI(I,J+1)=YI(I,J)        ! MOVE BACK OLD SAMPLED DATA
C                                     YI(1,1)=Y1(I)
C                                     YI(1,2)=Y1(I-1)
C                                     YI(1,3)=Y1(I-2) ETC.

      DO 2 I=1,30
      DO 2 J=9,1,-1
2     UI(I,J+1)=UI(I,J)  ! MOVE BACK OLD CONTROLLER OUTPUT DATA

      DO 4 I=1,30
      DO 4 J=9,1,-1
4     EI(I,J+1)=EI(I,J)        ! MOVE BACK OLD CONTROLLER INPUT DATA

      DO 3 I=1,30
3     YI(I,1)=Y(I)             ! STORE NEW YI

      RETURN
      END




      SUBROUTINE IDENT
C     CALL ESTIMATION ROUTINES FROM CONTRL

      IMPLICIT REAL*8 (a-H,o-z)
      COMMON/FCTCOM/PA
```

```
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4


C...PARAM(1) THROUGH PARAM(NA) ARE THE A PARAMETERS,
C...PARAM(NA+1)->PARAM(NPARAM) ARE THE B PARAMETERS.

C...PREDICTED YI OUTPUT FROM MODEL

      SUM=0.
      DO 102 J=1,NA
102   SUM=SUM-PARAM1(J)*YI(NOUTPT1,J+1)
      DO 103 J=1,NA
103   SUM=SUM-PARAM1(NA+J)*YI(NOUTPT2,J+1)
      DO 104 J=1,NA
104   SUM=SUM-PARAM1(NA+NA+J)*YI(NOUTPT3,J+1)
      DO 105 J=1,NA
105   SUM=SUM-PARAM1(NA+NA+NA+J)*YI(NOUTPT4,J+1)

      DO 106 J=1,NB
106   SUM=SUM+PARAM1(NA+NA+NA+NA+J)*UI(NINPUT1,J+ND)
      YIEST1=SUM

C...SET UP RECURSIVE ESTIMATE

      DO 107 J=1,NA
107   PHIM1(J)=-YI(NOUTPT1,J+1)
      DO 108 J=1,NA
```

```
108     PHIM1(NA+J)=-YI(NOUTPT2,J+1)
        DO 109 J=1,NA
109     PHIM1(NA+NA+J)=-YI(NOUTPT3,J+1)
        DO 110 J=1,NA
110     PHIM1(NA+NA+NA+J)=-YI(NOUTPT4,J+1)

        DO 111 J=1,NB
111     PHIM1(J+NA+NA+NA+NA)=UI(NINPUT1,J+ND)

        ERROR1=YI(NOUTPT1,1)-YIEST1



C...PREDICTED YI OUTPUT FROM MODEL

        SUM=0.
        DO 202 J=1,NA
202     SUM=SUM-PARAM2(J)*YI(NOUTPT2,J+1)

        DO 206 J=1,NB
206     SUM=SUM+PARAM2(NA+J)*UI(NINPUT2,J+ND)
        YIEST2=SUM

C...SET UP RECURSIVE ESTIMATE

        DO 207 J=1,NA
207     PHIM2(J)=-YI(NOUTPT2,J+1)

        DO 211 J=1,NB
211     PHIM2(J+NA)=UI(NINPUT2,J+ND)

        ERROR2=YI(NOUTPT2,1)-YIEST2



C...PREDICTED YI OUTPUT FROM MODEL

        SUM=0.
        DO 302 J=1,NA
```

```
302     SUM=SUM-PARAM3(J)*YI(NOUTPT1,J+1)
        DO 303 J=1,NA
303     SUM=SUM-PARAM3(NA+J)*YI(NOUTPT3,J+1)

        DO 306 J=1,NB
306     SUM=SUM+PARAM3(NA+NA+J)*UI(NINPUT3,J+ND)
        YIEST3=SUM

C...SET UP RECURSIVE ESTIMATE

        DO 307 J=1,NA
307     PHIM3(J)=-YI(NOUTPT1,J+1)
        DO 308 J=1,NA
308     PHIM3(NA+J)=-YI(NOUTPT3,J+1)

        DO 311 J=1,NB
311     PHIM3(J+NA+NA)=UI(NINPUT3,J+ND)

        ERROR3=YI(NOUTPT3,1)-YIEST3


C...PREDICTED YI OUTPUT FROM MODEL

        SUM=0.
        DO 402 J=1,NA
402     SUM=SUM-PARAM4(J)*YI(NOUTPT1,J+1)
        DO 403 J=1,NA
403     SUM=SUM-PARAM4(NA+J)*YI(NOUTPT4,J+1)

        DO 406 J=1,NB
406     SUM=SUM+PARAM4(NA+NA+J)*UI(NINPUT4,J+ND)
        YIEST4=SUM

C...SET UP RECURSIVE ESTIMATE

        DO 407 J=1,NA
407     PHIM4(J)=-YI(NOUTPT1,J+1)
        DO 408 J=1,NA
```

```
408     PHIM4(NA+J)=-YI(NOUTPT4,J+1)

        DO 411 J=1,NB
411     PHIM4(J+NA+NA)=UI(NINPUT4,J+ND)

        ERROR4=YI(NOUTPT4,1)-YIEST4




        GO TO (11,12,13,14,15,16,17) IEST
        TYPE *,' ERROR IN PAREST'

11      CALL LSTSQR
        GO TO 8
12      CALL LSTRES
        GO TO 8
13      CALL LSTFOR
        GO TO 8
14      CALL LSTADD
        GO TO 8
15      CALL PROJCT
        GO TO 8
16      CALL ORTHPR
        GO TO 8
17      CALL LSTRES
8       RETURN
        END



        SUBROUTINE PLOT (NPLOT,DATA)
        IMPLICIT REAL*8 (a-H,o-z)
        INTEGER*4 NPLOT
C       IMPLICIT REAL*8 (a-C,T,Y-z)
        REAL*8 YMIN(30),YMAX(30),TSTART,TSTOP
        DIMENSION TPLOT(512),YPLOT(512),IPLOT(30)
```

```
      VIRTUAL DATA(512,31)
      CHARACTER*2 CHROUT
      CHARACTER*35 PLTLBL,SIDE,BOTTOM
      COMMON/CONTIN/N,NP
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4


      NMAX=512
      IF(NPLOT.LT.512) NMAX=NPLOT

      TYPE *,' LABELS FOR THE PLOT? [NO=0 / 1=YES]'
      ACCEPT *,JUNK
      IF (JUNK.EQ.1) THEN
              PLTLBL='                                    '
              SIDE='                                    '
              BOTTOM='                                    '
              TYPE *,' INPUT TITLE FOR THE PLOT:'
              TYPE *,'-----------------------------------'
              ACCEPT 993,pltlbl
              PLTLBL='                                    '
              TYPE *,' INPUT BOTTOM LABEL FOR THE PLOT:'
              TYPE *,'-----------------------------------'
              ACCEPT 993,BOTTOM
              PLTLBL='                                    '
              TYPE *,' INPUT SIDE LABEL FOR THE PLOT:'
              TYPE *,'-----------------------------------'
              ACCEPT 993,SIDE
993           FORMAT(A)
      ENDIF
C     TYPE *,' INPUT THE NUMBER OF VARIABLES TO PLOT
C     SIMULTANEOUSLY:'
C     ACCEPT *,ISIMUL
      ISIMUL=1
```

```
C       IF(ISIMUL.LT.1.OR.ISIMUL.GT.5) ISIMUL=1
c       DO 9999 I=1,ISIMUL

        TYPE *,' TO PLOT RESPONSE OF Y(I), INPUT I:  '
        ACCEPT *,REALI
        I=1
9999    IPLOT(I)=INT(REALI+.00001)
        IF(IPLOT(I).GT.30.OR.IPLOT(I).LT.1) IPLOT(I)=1
        TYPE *,' PLOTTING RESPONSE OF VARIABLE ',IPLOT(I)
        I=IPLOT(I)


        PMAX=DATA(1,I)
        PMIN=DATA(1,I)
        DO 777 J=1,NMAX
        IF(PMAX.LT.DATA(J,I)) PMAX=DATA(J,I)
777     IF(PMIN.GT.DATA(J,I)) PMIN=DATA(J,I)


C       IF(ISIMUL.GT.1) THEN
C               DO 9998 I=1,ISIMUL
C               TYPE 9997,I,YMAX(IPLOT(I)),YMIN(IPLOT(I))
C9997        FORMAT(' FOR PLOT ',I3,', YMAX=',F12.5,' YMIN=',F12.4)
C9998           CONTINUE
C               IF(PMAX.LT.YMAX(IPLOT(I))) PMAX=YMAX(IPLOT(I))
C               IF(PMIN.GT.YMIN(IPLOT(I))) PMIN=YMIN(IPLOT(I))
C       ENDIF

        TSTART=DATA(1,31)
        TSTOP=DATA(NMAX,31)

        TMAX=TSTOP
        TMIN=TSTART

C-----------------------------------------------------------------
C               SETTING PLOT WINDOW
C-----------------------------------------------------------------

        TYPE *,' DEFAULT PLOT:'
```

```
      TYPE *,'                    Y MIN:  ',PMIN
      TYPE *,'                    Y MAX:  ',PMAX
      TYPE *,'                    T MIN:  ',TMIN
      TYPE *,'                    T MAX:  ',TMAX
1     TYPE *,'  '
      TYPE *,' USE THESE VALUES FOR THE PLOT WINDOW, OR INPUT
    1 NEW ONES? '
      TYPE *,' [USE THESE:0/INPUT NEW:1]'
      ACCEPT *,INN
      IF(INN.NE.1.AND.INN.NE.0) THEN
              TYPE *,' TRY AGAIN WITH CORRECT INPUT '
              GO TO 1
      ENDIF
      IF(INN.EQ.1) THEN
2     TYPE *,' THE CURRENT PLOT WINDOW IS:'
      TYPE *,'                    Y MIN:  ',PMIN
      TYPE *,'                    Y MAX:  ',PMAX
      TYPE *,'                    T MIN:  ',TMIN
      TYPE *,'                    T MAX:  ',TMAX
      TYPE *,'  '
      TYPE *,' INPUT:'
      TYPE *,'        0: IF THESE ARE ACCEPTABLE VALUES'
      TYPE *,'        1: TO CHANGE Y MIN'
      TYPE *,'        2: TO CHANGE Y MAX'
      TYPE *,'        3: TO CHANGE T MIN'
      TYPE *,'        4: TO CHANGE T MAX'
      TYPE *,'  '
      ACCEPT *,RINN
      INN=INT(RINN+.00001)
      IF(INN.EQ.0) GO TO 5
      IF(INN.LT.0.OR.INN.GE.5) THEN
              TYPE *,' USE THE CORRECT INPUT. TRY AGAIN.'
              GO TO 2
      ENDIF
      IF(INN.EQ.1) THEN
              TYPE *,' INPUT NEW VALUE OF Y MIN FOR THE PLOT:'
              ACCEPT *,PMIN
      ENDIF
      IF(INN.EQ.2) THEN
```

```
            TYPE *,' INPUT NEW VALUE OF Y MAX FOR THE PLOT:'
            ACCEPT *,PMAX
      ENDIF
      IF(INN.EQ.3) THEN
3           TYPE *,' INPUT NEW VALUE OF T MIN FOR THE PLOT:'
            ACCEPT *,TTTMIN
            IF(TTTMIN.LT.TSTART) THEN
            TYPE *,' T MIN IS NOT WITHIN DATA POINTS. TRY AGAIN.'
            GO TO 3
            ENDIF
            IF(TTTMIN.GT.TMAX) THEN
        TYPE *,' T MIN CANNOT BE GREATER THAN T MAX. TRY AGAIN'
                    GO TO 3
            ENDIF
            TMIN=TTTMIN
      ENDIF
      IF(INN.EQ.4) THEN
4           TYPE *,' INPUT NEW VALUE OF T MAX FOR THE PLOT:'
            ACCEPT *,TTTMAX
            IF(TTTMAX.GT.TSTOP) THEN
          TYPE *,' T MAX IS NOT WITHIN DATA POINTS. TRY AGAIN.'
                    GO TO 4
            ENDIF
            IF(TTTMAX.LT.TMIN) THEN
          TYPE *,' TMAX CANNOT BE LESS THAN T MIN. TRY AGAIN.'
                    GO TO 4
            ENDIF
      TMAX=TTTMAX
      ENDIF
      GO TO 2
      ENDIF


C-----------------------------------------------------------------
C           INITIALIZING THE GRAPHICS AND FRAME
C-----------------------------------------------------------------

5     DELTAP=PMAX-PMIN
```

```
        YPMAX=1.15*DELTAP+PMIN          ! MAKE WINDOW COORD.
        YPMIN=PMIN-.15*DELTAP

        TPMIN=-.1*(TMAX-TMIN)+TMIN
        TPMAX=1.07*(TMAX-TMIN)+TMIN

2000    TYPE *,' TERMINAL TYPE: [ VT240=0 / 4107 =1]'
        ACCEPT *,JUNK
        IF(JUNK.NE.1.AND.JUNK.NE.0) GO TO 2000
        YGDU = 100.0
        IF(JUNK.EQ.0) THEN
         CALL BAUDRT(9600)
         CALL GRSTRT(240,1)
         CALL NEWPAG
         XGDU = 131.2
        ELSE
         CALL BAUDRT(9600)
         CALL GRSTRT(4107,1)
         CALL NEWPAG
         XGDU = 133.34
        ENDIF


      CALL WINDOW(tpmin,tpmax,ypmin,ypmax)
        CALL TXICUR(8)
      CALL MOVE(TPMIN,ypMIN)
      CALL DRAW(TPMIN,YPMAX)
      CALL DRAW(TPMAX,YPMAX)
      CALL DRAW(TPMAX,YPMIN)
      CALL DRAW(TPMIN,YPMIN)

C-----------------------------------------------------------------
C               DRAWING THE AXIS
C-----------------------------------------------------------------
      call move(tmin,pmax)
        CALL DRAW (TMIN,PMAX)           ! DRAW VERTICAL AXIS
        CALL DRAW (TMIN,PMIN)           !
```

```
      IF(PMAX.GT.0.0.AND.PMIN.LT.0) THEN
                                    ! HORIZONTAL AXIS BETWEEN
              ZERO=0.
              CALL MOVE (TMAX,ZERO)   ! DRAW HORIZONTAL AXIS
              CALL DRAW (TMIN,ZERO)   !
      ELSE
              ZERO=PMIN
              CALL MOVE (TMAX,ZERO)   ! DRAW HORIZONTAL AXIS
              CALL DRAW (TMIN,ZERO)   !
      ENDIF


      CENTER=.25*(TMAX-TMIN)+TMIN
      TOP=1.1*(DELTAP)+PMIN


     CALL MOVE (CENTER,TOP)
     CALL TEXT(35,PLTLBL)

C-----------------------------------------------------------------
C                       AXIS TIME TICKS
C-----------------------------------------------------------------
      DT=TMAX-TMIN
      IF(DT.LT.1.0) THEN
              I1=INT(LOG10(TMAX-TMIN))-1
            ! MAGNITUDE SCALE OF VERTICAL AXIS
      ELSE
              I1=INT(LOG10(TMAX-TMIN))
            ! MAGNITUDE SCALE OF VERTICAL AXIS
      ENDIF

      EXPI1=10.0**I1                ! TEN TO THE I1 POWER
      DT1=(TMAX-TMIN)/EXPI1         ! SINGLE DIGIT SCALING
      TMIN1=TMIN/EXPI1
      TMAX1=TMAX/EXPI1
      ISTART=INT(TMIN1)          ! STARTING POINT FOR LARGE TICKS
      ISTOP=INT(TMAX1)           ! STOPING POINT FOR LARGE TICKS
      DCHARY=(PMAX-PMIN)/25.
      DCHART=(TMAX-TMIN)/60.
```

```
      DPTICK=.02*DELTAP+ZERO      ! DETERMINE BIG TOP TICK HEIGTH
      DPTIC2=-.02*DELTAP+ZERO    ! DETERMINE BIG BOTTOM TICK HEIGTH
      DO 10 I=ISTART,ISTOP
              TMARK=(FLOAT(I))*EXPI1          ! FIND TIME TICK MARK
              CALL MOVE (TMARK,DPTICK)     ! MOVE TO TICK LOCATION
              CALL DRAW (TMARK,DPTIC2)        ! DRAW TICK

              ENCODE(2,100,CHROUT) I      ! CHANGE I TO CHARACTER
100   FORMAT(I2)
              PDTIME=ZERO-DCHARY
              TMARK=TMARK-DCHART*3./2.   ! CENTER NUMB. UNDER TICK
              CALL MOVE (TMARK,PDTIME)        ! MOVE TO TEXT OUTPUT
              CALL TEXT(2,CHROUT)        ! OUTPUT TIME TICK TEXT

10    CONTINUE
      IF(I1.NE.0) THEN
              TMARK=TMAX+DCHART/2.
              CALL MOVE (TMARK,PDTIME) ! MOVE TO LOCATION FOR TEXT
              CALL TEXT(3,'x10')     ! OUTPUTTING TIME AXIS SCALING
              ENCODE(2,100,CHROUT) I1 ! CHANGE I1 TO CHARACTER
              CALL MOVE(TMARK,PDTIME) ! GO BACK TO TEXT LOCATION
              DT=2.*DCHART+TMARK
              DY=DCHARY+PDTIME
              CALL MOVE(DT,DY)      ! RELATIVE MOVE TO TEXT OUTPUT
              CALL TEXT(2,CHROUT)   ! OUTPUT SECOND SCALING TEXT
      ENDIF

      DPTICK=.01*DELTAP+ZERO    ! DETERMINE LITTLE TOP TICK HEIGTH
      DPTIC2=-.01*DELTAP+ZERO
                              ! DETERMINE LITTLE  BOTTOM TICK HEIGTH
      EXPI2=10.0**(I1-1)
      DO 11 I=ISTART-1,ISTOP+1         ! LITTLE TICK MARKS

      ILTICK=10                ! SCALING THE NUMBER OF TICK MARKS
      MULTI=1
      ILTEST=ISTOP-ISTART+2            ! PER MAJOR DIVISION
      IF(ILTEST.GT.8) THEN
              ILTICK=2
```

```
            MULTI=5
        ENDIF
        IF(ILTEST.GT.4.AND.ILTEST.LE.8) THEN
            ILICK=5
            MULTI=2
        ENDIF
        DO 11 J=1,ILTICK
        TMARK=FLOAT(I)*EXPI1+FLOAT(J*MULTI)*EXPI2
              ! DETERMINE LITTLE TICK LOCATION
        IF(TMARK.GT.TMIN.AND.TMARK.LT.TMAX) THEN
C                                ! DRAW TICKS WITHIN TIME AXIS
            CALL MOVE(TMARK,DPTICK) ! MOVE TO TICK LOCATION
            CALL DRAW (TMARK,DPTIC2)        ! DRAW TICK
        ENDIF
11      CONTINUE

C       TMARK=(TMAX-TMIN)*.95+TMIN
        PDTIME=ZERO-DELTAP*0.10
        CALL MOVE(TMARK,PDTIME)         ! MOVE TO TEXT OUTPUT
        CALL TEXT(10,'TIME (SEC)')              ! OUTPUT TEXT


C-----------------------------------------------------------------
C                       VERTICAL AXIS TICKS
C-----------------------------------------------------------------
        DP=PMAX-PMIN
        IF(DP.LT.1.0) THEN
            I1=INT(LOG10(PMAX-PMIN))-1
        ! MAGNITUDE SCALE OF VERTICAL AXIS
        ELSE
            I1=INT(LOG10(PMAX-PMIN))
        ! MAGNITUDE SCALE OF VERTICAL AXIS
        ENDIF
        EXPI1=10.0**I1                  ! TEN TO THE I1 POWER
        DP1=(PMAX-PMIN)/EXPI1           ! SINGLE DIGIT SCALING
        PMIN1=PMIN/EXPI1
        PMAX1=PMAX/EXPI1
        ISTART=INT(PMIN1)       ! STARTING POINT FOR LARGE TICKS
        ISTOP=INT(PMAX1)        ! STOPING POINT FOR LARGE TICKS
```

```
      DELTAT=TMAX-TMIN
      ZERO=TMIN
      DTTICK=.01*DELTAT+ZERO      ! DETERMINE BIG TOP TICK HEIGTH
      DTTIC2=-.01*DELTAT+ZERO     ! DETERMINE BIG BOTTOM TICK HEIGTH
      DO 110 I=ISTART,ISTOP
              PMARK=(FLOAT(I))*EXPI1          ! FIND TIME TICK MARK
              CALL MOVE(DTTICK,PMARK) ! MOVE TO TICK LOCATION
              CALL DRAW(DTTIC2,PMARK) ! DRAW TICK

              ENCODE(2,100,CHROUT) I     ! CHANGE I TO CHARACTER
              TPD=ZERO-DELTAT*0.05
              CALL MOVE(TPD,PMARK)            ! MOVE TO TEXT OUTPUT
              DPTEXT=DCHARY/2.+PMARK  ! MOVE NUMBER TEXT TO ALIGN
              CALL MOVE(TPD,DPTEXT)           ! WITH TICK MARK
              CALL TEXT(2,CHROUT)     ! OUTPUT TIME TICK TEXT
110   CONTINUE
      IF(I1.NE.0) THEN
              TPD=ZERO-DCHART*5.
              YYY=PMAX+.05*DELTAP
              CALL MOVE(TPD,YYY)        ! MOVE TO TOP OF Y AXIS
              CALL TEXT(3,'x10')     ! OUTPUTTING TIME AXIS SCALING
              ENCODE(2,100,CHROUT) I1  ! CHANGE I1 TO CHARACTER
              CALL MOVE(TPD,YYY)    ! MOVE TO TOP OF Y AXIS
              DT=2.*DCHART+TPD
              DY=DCHARY+YYY
              CALL MOVE(DT,DY)     ! RELATIVE MOVE TO TEXT OUTPUT
              CALL TEXT(2,CHROUT) ! OUTPUT SECOND SCALING TEXT
      ENDIF


      DTTICK=.005*DELTAT+ZERO  ! DETERMINE LITTLE TOP TICK HEIGTH
      DTTIC2=-.005*DELTAT+ZERO
                ! DETERMINE LITTLE  BOTTOM TICK HEIGTH
      EXPI2=10.0**(I1-1)
      DO 111 I=ISTART-1,ISTOP+1       ! LITTLE TICK MARKS

      ILTICK=10        ! SCALING THE NUMBER OF TICK MARKS
```

```
      MULTI=1
      ILTEST=ISTOP-ISTART+2           ! PER MAJOR DIVISION
      IF(ILTEST.GT.8) THEN
              ILTICK=2
              MULTI=5
      ENDIF
      IF(ILTEST.GE.5.AND.ILTEST.LE.8) THEN
              ILICK=5
              MULTI=2
      ENDIF

      DO 111 J=1,ILTICK
      PMARK=FLOAT(I)*EXPI1+FLOAT(MULTI*J)*EXPI2
         ! DETERMINE LITTLE TICK LOCATION
      IF(PMARK.GT.PMIN.AND.PMARK.LT.PMAX) THEN
C                ! DRAW TICKS WITHIN TIME AXIS
              CALL MOVE(DTTICK,PMARK) ! MOVE TO TICK LOCATION
              CALL DRAW(DTTIC2,PMARK) ! DRAW TICK
      ENDIF
111   CONTINUE

      PMARK=(PMAX-PMIN)*.4+PMIN
      TDTIME=ZERO-DELTAT*0.07
      CALL TXANGL(90.0)          ! ROTATE CHARACTERS 90 DEGREES
      CALL MOVE(TDTIME,PMARK)                  ! MOVE TO TEXT OUTPUT
      CALL TEXT(35,SIDE)              ! OUTPUT TEXT
      CALL TXANGL(0.0)   ! ROTATE CHARACTERS BACK TO 0 DEGREES


C-----------------------------------------------------------------
C             DRAWING THE DATA
C-----------------------------------------------------------------
      IF (NMAX.GT.50) THEN
              DO 12 J=1,ISIMUL
              DO 13 I=1,NMAX
              TPLOT(I)=DATA(I,31)
              YPLOT(I)=DATA(I,IPLOT(J))
              IF(YPLOT(I).GT.PMAX) YPLOT(I)=PMAX
              IF(YPLOT(I).LT.PMIN) YPLOT(I)=PMIN
```

```
                IF(TPLOT(I).GT.TMAX) TPLOT(I)=TMAX
                IF(TPLOT(I).LT.TMIN) TPLOT(I)=TMIN
13              CONTINUE
                CALL MOVE(TPLOT(1),YPLOT(1))
                DO 1222 I=1,NMAX
1222            CALL DRAW(TPLOT(I),YPLOT(I))     ! DRAW ARRAY
12              CONTINUE
        ELSE
C               break into discrete linear plots!
                DO 912 J=1,ISIMUL
                KDATA=0
                DO 913 IJKL=1,NMAX
                IJ1=IJKL+1
                DT=(DATA(IJ1,31)-DATA(IJKL,31))/10.
                DO 913 K=1,N
                KDATA=KDATA+1
                TPLOT(KDATA)=DATA(Ijkl,31)+FLOAT(K-1)*DT
                YPLOT(KDATA)=DATA(Ijkl,IPLOT(J))
c               type *,kdata,tplot(kdata),yplot(kdata)
                IF(YPLOT(KDATA).GT.PMAX) YPLOT(KDATA)=PMAX
                IF(YPLOT(KDATA).LT.PMIN) YPLOT(KDATA)=PMIN
                IF(TPLOT(KDATA).GT.TMAX) TPLOT(KDATA)=TMAX
                IF(TPLOT(KDATA).LT.TMIN) TPLOT(KDATA)=TMIN
913             CONTINUE
                NMAX10=10*NMAX
                CALL MOVE(TPLOT(1),YPLOT(1))
                DO 2222 I=1,NMAX
2222            CALL DRAW(TPLOT(I),YPLOT(I))     ! DRAW ARRAY
912             CONTINUE
        ENDIF
        CALL GRSTOP

        RETURN
        END

        SUBROUTINE PRINT(NPLOT,DATA)
        IMPLICIT REAL*8 (a-H,o-z)
        INTEGER*4 NPLOT
        VIRTUAL DATA(512,31)
```

```
      NMAX=NPLOT
      IF(NMAX.GT.512) NMAX=512
      type *,' INPUT THE TWO VARIABLES TO BE PRINTED OUT:'
111   TYPE *,' INPUT I FOR THE FIRST VARIABLE Y(I) [I=1 TO 10]'
      ACCEPT *,I1
      IF(I1.GT.30.OR.I1.LT.1) THEN
              TYPE *,' I MUST BE BETWEEN 1 AND 30. TRY AGAIN'
              GO TO 111
      ENDIF
112   TYPE *,' INPUT I FOR THE SECOND VARIABLE Y(I) [I=1 TO 30]'
      ACCEPT *,I2
      IF(I2.GT.10.OR.I2.LT.1) THEN
              TYPE *,' I MUST BE BETWEEN 1 AND 30. TRY AGAIN'
              GO TO 112
      ENDIF

      TYPE *,'  '
      TYPE *,'  '
      TYPE 113,I1,I2
113   FORMAT(10X,'TIME',12X,'Y',I2,12X,'Y',I2)

      NTYPE=1
      IF(NMAX.GT.20) NTYPE=INT((NMAX/20.))
      DO 1 I=1,NMAX,NTYPE
1     TYPE 2,DATA(I,11),DATA(I,I1),DATA(I,I2)
2     FORMAT(3F15.5)
      RETURN
      END


      SUBROUTINE WRITE(NPLOT,DATA)
      IMPLICIT REAL*8 (a-H,o-z)
      INTEGER*4 NPLOT
      VIRTUAL DATA(512,31)
      CHARACTER*13 FILEN

11    TYPE*,' INPUT THE VARIABLE Y(I) [I=1 TO 30] YOU WANT IN A
     1 FILE'
      ACCEPT *,I
```

```
      IF(I.GT.30.OR.I.LT.1) THEN
      TYPE*,' I MUST BE BETWEEN 1 AND 30. TRY AGAIN '
      GO TO 11
      ENDIF

      TYPE*,' INPUT NAME OF FILE TO WRITE DATA TO... '
      ACCEPT 3,FILEN
3     FORMAT(A13)
C.......LOCATE SPACE IN INPUT
      LOCS=INDEX(FILEN,' ')-1
      TYPE*,'FILENAME=',FILEN
      OPEN(UNIT=1,FILE=FILEN( :LOCS),TYPE='NEW')

       NMAX=NPLOT
       IF(NMAX.GT.512)NMAX=512

       DO 4 J=1,NMAX
         WRITE(1,2) DATA(J,31),DATA(J,I)
4      CONTINUE

2     FORMAT(F10.5,1X,F10.5)
22    FORMAT(I5)
      CLOSE(UNIT=1,DISPOSE='SAVE')
      RETURN
      END


C-----------------------------------------------------------------

C                       CONTINUOUS PLANT MODEL FOR
C           DISCRETE ADAPTIVE CONTROL OF CONTINUOUS SYSTEMS

C                       USE ONLY WITH RKADAPT

C-----------------------------------------------------------------


      SUBROUTINE FCT(T,Y,DY)
```

```
      IMPLICIT REAL*8 (a-H,o-z)
      DIMENSION Y(30),DY(30),PA(30)
      COMMON/FCTCOM/PA
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/CONTIN/N,NP
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4

C...NP IS NUMBER OF PARAMETERS TO BE INPUT BY USER,
C...NPARAM IS NUMBER OF PARAMETERS TO BE IDENTIFIED


C
C       INERTIAS AND MASSES FOR AXES

C       AXIS 1  (BASE ROTATION)

        EMASS1=PA(1)

        EIXX1=PA(2)
        EIYY1=PA(3)
        EIZZ1=PA(4)

C       AXIS 2  (VERITCAL TRANSLATION)

        EMASS2=PA(5)

        EIXX2=PA(6)
        EIYY2=PA(7)
```

```
      EIZZ2=PA(8)

C     AXIS 3  (HORIZONTAL TRANSLATION)

      EMASS3=PA(9)

      EIXX3=PA(10)
      EIYY3=PA(11)
      EIZZ3=PA(12)


C     AXIS 4  (WRIST ROTATION)

      EMASS4=PA(13)

      EIXX4=PA(14)
      EIYY4=PA(15)
      EIZZ4=PA(16)

C
C

C     AXIS 1  (BASE ROTATION) - CONSTANTS AND EQUATION OF MOTION
C



      PS1=1650.0D0
      C1A1=1.52D0
      C1B1=1.31D0
      BMA1=100000.0D0
      VOA1=64.9368D0
      VOB1=64.9368D0
      AA1=5.30D0
      AB1=5.30D0
      VPA1=18.603D0
      VPB1=18.603D0
      BMB1=100000.0D0
      B11=59.0D0
```

```
      CF1=53.0D0
      EKPA1=0.0411D0
      EKPB1=0.0411D0
      EMARM1=3.510D0
      EM1=0.6101D0
      G1=386.4D0
C
C
C
      XV1=0.05D0*UI(1,1)
C
      SW11=XV1
C
      IF(SW11.LT.0.0D0) THEN
      SHUT11=0.0D0
      SHUT21=1.0D0
      ELSE IF(SW11.EQ.0.0D0) THEN
      SHUT11=0.0D0
      SHUT21=0.0D0
      ELSE IF(SW11.GT.0.0D0) THEN
      SHUT11=1.0D0
      SHUT21=0.0D0
      ENDIF
C
      QA1=C1A1*XV1*DSQRT(PS1-Y(1))*SHUT11 +
     1     C1A1*XV1*DSQRT(Y(1))*SHUT21
C
      QB1=-C1B1*XV1*DSQRT(Y(2))*SHUT11 -
     1      C1B1*XV1*DSQRT(PS1-Y(2))*SHUT21
C
C
C
      F1=(Y(1)*AA1 - Y(2)*AB1)*EMARM1-B11*Y(4)-
     1    CF1*((Y(4)+0.00001D0)/(DABS(Y(4)) + 0.00001D0))
C
      DY(1)=(BMA1/(VOA1 + VPA1*Y(3)))*
     1    (QA1-VPA1*Y(4)-EKPA1*(Y(1)-Y(2)))
C
      DY(2)=(BMB1/(VOB1 - VPB1*Y(3)))*
```

```
     1     (QB1+VPB1*Y(4)-EKPB1*(Y(2)-Y(1)))
C
        DY(3)=Y(4)
C
        DY(4)=(1.D0/(Y(11)*Y(11)*EM1 + EIZZ1+EIYY3+EIYY2+EIXX4+
     1       DCOS(Y(15))*DCOS(Y(15))*(EIYY4-EIXX4)))*
     2       (F1-2.D0*Y(4)*Y(12)*Y(11)*EM1
     3     +2.D0*Y(4)*Y(16)*DCOS(Y(15))*DSIN(Y(15))*(EIYY4-EIXX4))
C
C



C       AXIS 2  (VERTICAL TRANSLATION) - CONSTANTS AND EQUATION
C                                        OF MOTION



        PS2=1650.0D0
        C1A2=1.46D0
        C1B2=1.28D0
        BMA2=100000.0D0
        VOA2=97.0D0
        VOB2=78.0D0
        AA2=4.04D0
        AB2=3.25D0
        BMB2=100000.0D0
        B12=67.7D0
        CF2=32.0D0
        EKPA2=0.0327D0
        EKPB2=0.0281D0
        EM2=1.05D0
        G2=386.4D0
C
C
C
        XV2=0.05D0*UI(2,1)
C
        SW12=XV2
```

```
C
      IF(SW12.LT.0.0D0) THEN
      SHUT12=0.0D0
      SHUT22=1.0D0
      ELSE IF(SW12.EQ.0.0D0) THEN
      SHUT12=0.0D0
      SHUT22=0.0D0
      ELSE IF(SW12.GT.0.0D0) THEN
      SHUT12=1.0D0
      SHUT22=0.0D0
      ENDIF
C
      QA2=C1A2*XV2*DSQRT(PS2-Y(5))*SHUT12 +
     1     C1A2*XV2*DSQRT(Y(5))*SHUT22
C
      QB2=-C1B2*XV2*DSQRT(Y(6))*SHUT12 -
     1      C1B2*XV2*DSQRT(PS2-Y(6))*SHUT22
C
C
C
      F2=Y(5)*AA2 - Y(6)*AB2 - B12*Y(8) - CF2*((Y(8) + 0.00001D0)/
     1   (DABS(Y(8)) + 0.00001D0))
C
      DY(5)=(BMA2/(VOA2 + AA2*Y(7)))*
     1       (QA2-AA2*Y(8)-EKPA2*(Y(5)-Y(6)))
C
      DY(6)=(BMB2/(VOB2 - AB2*Y(7)))*
     1       (QB2+AB2*Y(8)-EKPB2*(Y(6)-Y(5)))
C
      DY(7)=Y(8)
C
      DY(8)=F2/EM2 - G2

C


C     AXIS 3  (HORIZONTAL TRANSLATION) - CONSTANTS AND EQUATION
C                                      OF MOTION
```

```
      PS3=1650.0D0
      C1A3=1.46D0
      C1B3=1.28D0
      BMA3=100000.0D0
      VOA3=113.0D0
      VOB3=85.0D0
      AA3=3.14D0
      AB3=2.35D0
      BMB3=100000.0D0
      B13=67.7D0
      CF3=32.0D0
      EKPA3=0.000327D0
      EKPB3=0.000281D0
      EM3=0.6101D0
      G3=386.4D0
C
C
C
      XV3=0.05D0*UI(3,1)
C
      SW13=XV3
C
      IF(SW13.LT.0.0D0) THEN
      SHUT13=0.0D0
      SHUT23=1.0D0
      ELSE IF(SW13.EQ.0.0D0) THEN
      SHUT13=0.0D0
      SHUT23=0.0D0
      ELSE IF(SW13.GT.0.0D0) THEN
      SHUT13=1.0D0
      SHUT23=0.0D0
      ENDIF
C
      QA3=C1A3*XV3*DSQRT(PS3-Y(9))*SHUT13+
     1     C1A3*XV3*DSQRT(Y(9))*SHUT23
C
      QB3=-C1B3*XV3*DSQRT(Y(10))*SHUT13-
```

```
     1       C1B3*XV3*DSQRT(PS3-Y(10))*SHUT23
C
C
C
        F3=Y(9)*AA3 - Y(10)*AB3 - B13*Y(12) - CF3*
     1     ((Y(12) + 0.00001D0)/(DABS(Y(12)) + 0.00001D0))
C
        DY(9)=(BMA3/(VOA3 + AA3*Y(11)))*
     1        (QA3-AA3*Y(12)-EKPA3*(Y(9)-Y(10)))
C
        DY(10)=(BMB3/(VOB3 - AB3*Y(11)))*
     1        (QB3+AB3*Y(12)-EKPB3*(Y(10)-Y(9)))
C
        DY(11)=Y(12)
C
        DY(12)=F3/EM3 + Y(4)*Y(4)*Y(11)

C
C


C       AXIS 4  (WRIST ROTATION) - CONSTANTS AND EQUATION OF MOTION
C



        PS4=1650.0D0
        C1A4=1.42D0
        C1B4=1.25D0
        BMA4=100000.0D0
        VOA4=51.3674D0
        VOB4=51.3674D0
        AA4=3.37D0
        AB4=3.37D0
        VPA4=10.511D0
        VPB4=10.511D0
        BMB4=100000.0D0
        B14=51.6D0
        CF4=42.3D0
```

```
      EKPA4=0.0300D0
      EKPB4=0.0300D0
      EMARM4=3.12D0
      EM4=0.0D0
      G4=386.4D0
C
C
C
      XV4=0.05D0*UI(4,1)
C
      SW14=XV4
C
      IF(SW14.LT.0.0D0) THEN
      SHUT14=0.0D0
      SHUT24=1.0D0
      ELSE IF(SW14.EQ.0.0D0) THEN
      SHUT14=0.0D0
      SHUT24=0.0D0
      ELSE IF(SW14.GT.0.0D0) THEN
      SHUT14=1.0D0
      SHUT24=0.0D0
      ENDIF
C
      QA4=C1A4*XV4*DSQRT(PS4-Y(13))*SHUT14+
     1     C1A4*XV4*DSQRT(Y(13))*SHUT24
C
      QB4=-C1B4*XV4*DSQRT(Y(14))*SHUT14-
     1      C1B4*XV4*DSQRT(PS4-Y(14))*SHUT24
C
C
C
      F4=(Y(13)*AA4 - Y(14)*AB4)*EMARM4-B14*Y(16)-CF4*
     1    ((Y(16)+0.00001D0)/(DABS(Y(16)) + 0.00001D0))
C
      DY(13)=(BMA4/(VOA4+VPA4*Y(15)))*
     1    (QA4-VPA4*Y(16)-EKPA4*(Y(13)-Y(14)))
C
      DY(14)=(BMB4/(VOB4-VPB4*Y(15)))*
     1    (QB4+VPB4*Y(16)-EKPB4*(Y(14)-Y(13)))
```

```
C
      DY(15)=Y(16)
C
      DY(16)=(1.D0/EIZZ4)*(F4 - Y(4)*Y(4)*DCOS(Y(15))*DCOS(Y(15))*
     1        (EIYY4-EIXX4))
C
C


C     THESE ARE DUMMY STATEMENTS TO PLOT THE CONTROL EFFORT FOR
C     EACH OF THE AXIS.

      DY(17)=0.D0
      Y(17)=PARAM3(5)

      DY(18)=0.D0
      Y(18)=PARAM3(6)

      DY(19)=0.D0
      Y(19)=PARAM3(7)

      DY(20)=0.D0
      Y(20)=PARAM3(8)

C
C

      RETURN
      END


      SUBROUTINE CONTRL(T,Y)
      IMPLICIT REAL*8 (a-H,o-z)
      DIMENSION Y(30),PA(30)
      COMMON/FCTCOM/PA
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
```

```
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4


C     TAKES INPUT EI, OUTPUTS CONTROL VARIABLE UI

c     YI(1,1) = Y1 (I)
c     YI(1,2) = Y1 (I-1)
c     YI(1,3) = Y1 (I-2)       .ETC
c     YI(1,4) = Y1 (I-3)
C     YI(3,1) = Y3 (I)
C

      CALL IDENT                        ! CALL IDENTIFICATION ROUTINE


      DO 111 I=1,10
       TI=I*8.0D0
       IF(T.GT.(TI-8.0D0).AND.T.LE.TI) RINPUT1=((-1.D0)**(I+1))*1.0
111   CONTINUE

      DO 112 I=1,10
       TI=I*8.0D0
       IF(T.GT.(TI-8.0D0).AND.T.LE.TI)
     1 RINPUT2=((-1.D0)**(I+1))*10.0
112   CONTINUE


C     MODEL REFERENCE ADAPTIVE CONTROLLER  -  AXIS 1

      SUMMM1=PARAM1(1)*YI(3,1)+PARAM1(2)*YI(3,2)+PARAM1(3)*YI(3,3)
```

```
     1        + PARAM1(4)*YI(3,4)+PARAM1(5)*YI(7,1)+PARAM1(6)*YI(7,2)
     2    + PARAM1(7)*YI(7,3)+PARAM1(8)*YI(7,4)+PARAM1(9)*YI(11,1)
     3  + PARAM1(10)*YI(11,2)+PARAM1(11)*YI(11,3)+PARAM1(12)*YI(11,4)
     4  +PARAM1(13)*YI(15,1)+PARAM1(14)*YI(15,2)+PARAM1(15)*YI(15,3)
     5     +PARAM1(16)*YI(15,4)-PARAM1(18)*UI(1,2)-PARAM1(19)*UI(1,3)
     6      -PARAM1(20)*UI(1,4)+ 1.883529D0*YI(3,1)
     7      -0.88692D0*YI(3,2) + (0.00172959D0+0.001661775D0)*RINPUT1


        UI(1,1)=SUMMM1/PARAM1(17)


C       MODEL REFERENCE ADAPTIVE CONTROLLER  -  AXIS 2

        SUMMM2=PARAM2(1)*YI(7,1)+PARAM2(2)*YI(7,2)+PARAM2(3)*YI(7,3)
     1        + PARAM2(4)*YI(7,4)-PARAM2(6)*UI(2,2)
     2     -PARAM2(7)*UI(2,3)-PARAM2(8)*UI(2,4)+ 1.883529D0*YI(7,1)
     3   -0.88692D0*YI(7,2) + (0.00172959D0+0.001661775D0)*RINPUT2

        UI(2,1)=SUMMM2/PARAM2(5)


C       MODEL REFERENCE ADAPTIVE CONTROLLER  -  AXIS 3

        SUMMM3=PARAM3(1)*YI(3,1)+PARAM3(2)*YI(3,2)+PARAM3(3)*YI(3,3)
     1   + PARAM3(4)*YI(3,4)+PARAM3(5)*YI(11,1)+PARAM3(6)*YI(11,2)
     2   +PARAM3(7)*YI(11,3)+PARAM3(8)*YI(11,4)-PARAM3(10)*UI(3,2)
     3 -PARAM3(11)*UI(3,3)-PARAM3(12)*UI(3,4)+ 1.883529D0*YI(11,1)
     4 -0.88692D0*YI(11,2) + (0.00172959D0+0.001661775D0)*RINPUT2

        UI(3,1)=SUMMM3/PARAM3(9)


C       MODEL REFERENCE ADAPTIVE CONTROLLER  -  AXIS 4
```

```
      SUMMM4=PARAM4(1)*YI(3,1)+PARAM4(2)*YI(3,2)+PARAM4(3)*YI(3,3)
     1  + PARAM4(4)*YI(3,4)+PARAM4(5)*YI(15,1)+PARAM4(6)*YI(15,2)
     2 +PARAM4(7)*YI(15,3)+PARAM4(8)*YI(15,4)-PARAM4(10)*UI(4,2)
     3 -PARAM4(11)*UI(4,3)-PARAM4(12)*UI(4,4)+ 1.883529D0*YI(15,1)
     4 -0.88692D0*YI(15,2)+ (0.00172959D0+0.001661775D0)*RINPUT1

      UI(4,1)=SUMMM4/PARAM4(9)


C     TO SPECIFY THE LIMITS ON THE CONTROLLER INPUTS
      UMIN=-10.0D0
      UMAX=10.0D0

      IF(UI(1,1).GE.UMIN.OR.UI(1,1).LE.UMAX) UI(1,1)=UI(1,1)
      IF(UI(1,1).LT.UMIN) UI(1,1)=UMIN
      IF(UI(1,1).GT.UMAX) UI(1,1)=UMAX

      IF(UI(2,1).GE.UMIN.OR.UI(2,1).LE.UMAX) UI(2,1)=UI(2,1)
      IF(UI(2,1).LT.UMIN) UI(2,1)=UMIN
      IF(UI(2,1).GT.UMAX) UI(2,1)=UMAX

      IF(UI(3,1).GE.UMIN.OR.UI(3,1).LE.UMAX) UI(3,1)=UI(3,1)
      IF(UI(3,1).LT.UMIN) UI(3,1)=UMIN
      IF(UI(3,1).GT.UMAX) UI(3,1)=UMAX

      IF(UI(4,1).GE.UMIN.OR.UI(4,1).LE.UMAX) UI(4,1)=UI(4,1)
      IF(UI(4,1).LT.UMIN) UI(4,1)=UMIN
      IF(UI(4,1).GT.UMAX) UI(4,1)=UMAX

C     END OF CONTROL LIMIT SPECIFICATION
C
C


      RETURN
      END
```

```
      SUBROUTINE INITOV
      IMPLICIT REAL*8 (a-H,o-z)
      DIMENSION Y(30),PA(30)
      COMMON/FCTCOM/PA
      COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1 NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
      COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
      COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1 ILOOP
      COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
      COMMON/CONTIN/N,NP
      COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1 PCOV4(30,30)
      COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
      COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
      COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
      COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1 NPARAM2,NPARAM3,NPARAM4

C     all initialization of the yi's, ui's and ei's for the
C     discrete
c     system must be included here

      NINPUT1=1        ! INPUT 1 FOR MIMO ESTIMATION FROM UI(1, )
      NOUTPT1=3        ! OUTPUT 3 FOR MIMO ESTIMATION FROM YI(3, )

      NINPUT2=2        ! INPUT 2 FOR MIMO ESTIMATION FROM UI(2, )
      NOUTPT2=7        ! OUTPUT 7 FOR MIMO ESTIMATION FROM YI(7, )

      NINPUT3=3        ! INPUT 3 FOR MIMO ESTIMATION FROM UI(3, )
      NOUTPT3=11       ! OUTPUT 11 FOR MIMO ESTIMATION FROM YI(11, )

      NINPUT4=4        ! INPUT 4 FOR MIMO ESTIMATION FROM UI(4, )
      NOUTPT4=15       ! OUTPUT 15 FOR MIMO ESTIMATION FROM YI(15, )

      DO 1 I=1,30
      DO 1 J=1,10
```

```
1       YI(I,J)=0.
        DO 2 I=1,30
        DO 2 J=1,10
        UI(I,J)=0.
2       EI(I,J)=0.
C.........put initial values here!
        return
        end



        SUBROUTINE INITID

C...SETS UP A CANNED ESTIMATION PROCEDURE

        IMPLICIT REAL*8 (a-H,o-z)
        COMMON/FCTCOM/PA
        COMMON/OLDVAL/YI(30,10),UI(30,10),EI(30,10),NINPUT1,NOUTPT1,
     1  NINPUT2,NOUTPT2,NINPUT3,NOUTPT3,NINPUT4,NOUTPT4
        COMMON/RECURS/PHI(30),PCOV(30,30),STORE(30),ERROR,SUM
        COMMON/ESTMAT/COV,COVNOI,IEST,IRESET,FORGET,PARAM(30),ICOUNT,
     1  ILOOP
        COMMON/TRACE1/TRACEP,NPARAM,NA,NB,ND
        COMMON/MULCOV/PCOV1(30,30),PCOV2(30,30),PCOV3(30,30),
     1  PCOV4(30,30)
        COMMON/MULPHI/PHIM1(30),PHIM2(30),PHIM3(30),PHIM4(30)
        COMMON/MULPAR/PARAM1(30),PARAM2(30),PARAM3(30),PARAM4(30)
        COMMON/MULERR/ERROR1,ERROR2,ERROR3,ERROR4
        COMMON/TRACEM/TRACEP1,TRACEP2,TRACEP3,TRACEP4,NPARAM1,
     1  NPARAM2,NPARAM3,NPARAM4

C...LEAST SQUARES
        IEST=1
        COV=10D4
        NA=4
        NB=4
        ND=1            ! NO DELAY  (ND=0 IMPLIES FEEDFORWARD!!!!!!)
        NPARAM=NA+NB
C...INITIAL GUESS AT PARAMETERS
```

```
PARAM(1)=0.0D0          ! ACTUAL VALUE  -1.9491
PARAM(2)=0.0D0          ! ACTUAL VALUE  .9412
PARAM(3)=0.0D0          ! ACTUAL VALUE  .0078
PARAM(4)=0.0D0          ! ACTUAL VALUE  .0002
PARAM(5)=0.1D0          ! ACTUAL VALUE  .0329
PARAM(6)=0.0D0          ! ACTUAL VALUE  -.0216
PARAM(7)=0.0D0          ! ACTUAL VALUE  -.0096
PARAM(8)=0.0D0          ! ACTUAL VALUE  -0.0001

RETURN
END
```

9.4 Program Listing for Craig's Approach to Adaptive Control

```
C     The purpose of this FCT routine is to perform an adaptive
c     control of one dof robot
c
c     Definition of variables:
c     ------------------------
c
c          Y(1) - velocity of vertical translation
c          Y(2) - position of vertical translation
c          Y(3) - estimate of mass M1
c          Y(4) - estimate of K1
c          Y(5) - estimate of V1
c          Y(6) - error in joint 1
c          Y(7) - error in joint 1 derivative
c          Y(8) - TH(1)
c          Y(9) - left chamber pressure
c          Y(10)- right chamber pressure
c
c     Definition of parameters:
c     -------------------------
c
c          PA(1) - if > 1.0 include damping at joints
c          PA(2) - filter time constant PSI
c          PA(3) - Gamma Matrix weighting coefficient for masses
c          PA(4) - KV, feedback velocity gains
c          PA(5) - KP, feedback position gains
c          PA(6) - Gamma Matrix weighting coefficient for damping
c          PA(7) - if > 1.0 include parameter reset
c          PA(8) - if =0.0 and PA(2)=1.0, no filter
c          PA(9) - Gamma Matrix weighting coefficient for dyn. fric.

           SUBROUTINE FCT(T,Y,DERY)

           REAL *8 T,Y(10),DERY(10),TI,TT,TTT
           REAL *8 LHS(1,1),RHS(1),ACCEL(1,1)
           REAL *8 M1,G
           REAL *8 V1,K1
           REAL *8 PA(20)
```

```
      REAL *8 ST,CT,S2T,C2T,S4T,C4T,S6T,C6T
      REAL *8 TH(1),THD(1),THDD(1),A1,A2,B1,B2
      REAL *8 E(1),ED(1),THDDS(1),KV,KP
      REAL *8 PSI, E1(1)

      REAL *8 MHAT(1,1),QHAT(1,1),ACC(1)
      REAL *8 WT(3,1)
      REAL *8 GAM(3,3),MHI(1,1),DETM
      REAL *8 TMP1(1,1),TMP2(3,1),PHATD(3,1),TMP3(1,1)
      REAL *8 TOR(1,1),DELMA,DELDA,LOW(3),HI(3)
      REAL *8 TSTAR,ATC,XMAX,VMAX
      REAL *8 PS,C1A,C1B,BMA,VOA,VOB,AA,AB,BMB,EKPA,EKPB
      REAL *8 QA,QB,SW11,SHUT11,SHUT21

      COMMON/FCTCOM/PA

C     Input robot parameters

      G=386.4
      M1=1.05

C     Input damping terms

      IF(PA(1).GT.1.0) THEN

       V1=67.7
       K1=32.0

      ELSE

       V1=0.D0
       K1=0.D0

      ENDIF

C     Compute the desired joint angles and derivatives

     TSTAR=PA(11)
```

```
      ATC=PA(12)
      VMAX=PA(13)
      XMAX=PA(14)
      TTT=PA(15)


      DO 1099 I=0,5

        IF(T.GT.((2.*TSTAR+TTT)*I).
     1   AND.T.LT.((2.*TSTAR+TTT)*(I+1))) THEN

         TT=T-(2.*TSTAR+TTT)*I

      IF(TT.GT.0.0.AND.TT.LT.ATC)TH(1)=((1.D0)**I)*
     1 (VMAX*TT*TT*0.5D0)/(ATC)
      IF(TT.GE.ATC.AND.TT.LT.(TSTAR-ATC))TH(1)=
     1 ((1.D0)**I)*(-0.5D0*ATC*VMAX+VMAX*TT)
      IF(TT.GE.(TSTAR-ATC).AND.TT.LT.TSTAR)TH(1)=
     1 ((1.D0)**I)*(-(VMAX*TT*TT*0.5D0)/ATC
     2 + (VMAX*(TSTAR-ATC)*TT)/ATC + VMAX*TT-ATC*0.5*VMAX-
     3 (VMAX*0.5D0*(TSTAR-ATC)*(TSTAR-ATC))/ATC)

      IF(TT.GE.TSTAR.AND.TT.LT.(TSTAR+TTT))TH(1)=
     1 ((1.D0)**I)*XMAX

      IF(TT.GE.(TSTAR+TTT).AND.TT.LT.(TSTAR+ATC+TTT))TH(1)=
     1 ((1.D0)**I)*(XMAX-(VMAX*0.5D0/ATC)*
     2  (TT-TSTAR-TTT)*(TT-TSTAR-TTT))

      IF(TT.GE.(TSTAR+ATC+TTT).AND.TT.LT.
     1 (2.D0*TSTAR-ATC+TTT))TH(1)=
     2 ((1.D0)**I)*(XMAX +
     3 0.5D0*VMAX*ATC - VMAX*(TT-TSTAR-TTT))

      IF(TT.GE.(2.D0*TSTAR-ATC+TTT).AND.TT.LT.
     1 (2.D0*TSTAR+TTT))TH(1)=
     2 ((1.D0)**I)*(XMAX +
     3 VMAX*ATC - VMAX*TSTAR+(VMAX*(TT-TTT)*(TT-TTT)
     4 *0.5D0/ATC)-(2.D0*VMAX*TSTAR
```

```
     5    *(TT-TTT)/ATC) + (2.DO*VMAX*TSTAR*TSTAR/ATC))

C          IF(TT.GE.(2.*TSTAR+TTT).AND.TT.LT.(2.*TSTAR+2.*TTT))
C     1    TH(1)=0.0

          IF(TT.GT.0.0.AND.TT.LT.ATC)THD(1)=
     1     ((1.DO)**I)*(VMAX*TT/ATC)
          IF(TT.GE.ATC.AND.TT.LT.(TSTAR-ATC))THD(1)=
     1     ((1.DO)**I)*(VMAX)
          IF(TT.GE.(TSTAR-ATC).AND.TT.LT.TSTAR)THD(1)=
     1     ((1.DO)**I)*(VMAX- (VMAX/ATC)*(TT-TSTAR+ATC))

          IF(TT.GE.TSTAR.AND.TT.LT.(TSTAR+TTT))THD(1)=0.0


          IF(TT.GE.(TSTAR+TTT).AND.TT.LT.(TSTAR+ATC+TTT))THD(1)=
     1     ((1.DO)**I)*((-VMAX/ATC)*(TT-TSTAR-TTT))

          IF(TT.GE.(TSTAR+ATC+TTT).AND.TT.LT.
     1     (2.DO*TSTAR-ATC+TTT))THD(1)=
     2     ((1.DO)**I)*(-VMAX)

          IF(TT.GE.(2.DO*TSTAR-ATC+TTT).AND.TT.
     1     LT.(2.DO*TSTAR+TTT))THD(1)=
     2     ((1.DO)**I)*(-VMAX + (VMAX/ATC)*(TT-2.*TSTAR+ATC-TTT))

C         IF(TT.GE.(2.*TSTAR+TTT).AND.TT.LT.(2.*TSTAR+2.*TTT))
C     1     THD(1)=0.0


          IF(TT.GT.0.0.AND.TT.LT.ATC)THDD(1)=
     1     ((1.DO)**I)*(VMAX/ATC)
          IF(TT.GE.ATC.AND.TT.LT.(TSTAR-ATC))THDD(1)=
     1     ((1.DO)**I)*(0.0)
          IF(TT.GE.(TSTAR-ATC).AND.TT.LT.TSTAR)THDD(1)=
     1     ((1.DO)**I)*(- (VMAX/ATC))

          IF(TT.GE.TSTAR.AND.TT.LT.(TSTAR+TTT))THDD(1)=0.0
```

```
          IF(TT.GE.(TSTAR+TTT).AND.TT.LT.(TSTAR+ATC+TTT))THDD(1)=
     1     ((1.D0)**I)*((-VMAX/ATC))

          IF(TT.GE.(TSTAR+ATC+TTT).AND.TT.
     1     LT.(2.D0*TSTAR-ATC+TTT))THDD(1)=
     2     ((1.D0)**I)*(0.0)

          IF(TT.GE.(2.D0*TSTAR-ATC+TTT).AND.TT.
     1     LT.(2.D0*TSTAR+TTT))THDD(1)=
     2     ((1.D0)**I)*((VMAX/ATC))

C          IF(TT.GE.(2.*TSTAR+TTT).AND.TT.LT.(2.*TSTAR+2.*TTT))
C     1     THDD(1)=0.0


          ENDIF
1099        CONTINUE


C      Compute errors and derivative of errors

           E(1)=TH(1)-Y(2)
           ED(1)=THD(1)-Y(1)

           DERY(6)=0.0
              Y(6)=E(1)
           DERY(7)=0.0
              Y(7)=ED(1)

           DERY(8)=0.0
              Y(8)=TH(1)
C           DERY(9)=0.0
C              Y(9)=THD(1)
C           DERY(10)=0.0
C              Y(10)=THDD(1)

C      Compute input accelerations
```

```
      KV=PA(4)
      KP=PA(5)

      THDDS(1)=THDD(1) + KV*ED(1) + KP*E(1)

C     Compute the filtered error

      PSI=PA(2)
      E1(1)=PA(8)*ED(1) + PSI*E(1)

C     Form MHAT and QHAT

      CALL HAT(Y,MHAT,QHAT)

C     Form the input torque vector TOR

      TMP3(1,1)=MHAT(1,1)*THDDS(1)
      TOR(1,1)=TMP3(1,1) - QHAT(1,1)

C     Generate the actuator dynamics

      PS=1650.D0
      C1A=1.46D0
      C1B=1.28D0
      BMA=100000.0D0
      VOA=97.0D0
      VOB=78.0D0
      AA=4.04D0
      AB=3.25D0
      BMB=100000.0D0
      EKPA=0.0327D0
      EKPB=0.0281D0


      XV=(TOR(1,1) + 25.244D0*Y(1))/1128.7D0

      IF(XV.GT.0.5D0)XV=0.5D0
      IF(XV.LT.-0.5D0)XV=-0.5D0
      IF(XV.LT.0.5D0.AND.XV.GT.-0.5D0)XV=XV
```

```
      SW11=XV

      IF(SW11.LT.0.0D0) THEN
      SHUT11=0.0D0
      SHUT21=1.0D0
      ELSE IF(SW11.EQ.0.0D0) THEN
      SHUT11=0.0D0
      SHUT21=0.0D0
      ELSE IF(SW11.GT.0.0D0) THEN
      SHUT11=1.0D0
      SHUT21=0.0D0
      ENDIF
C
      QA=C1A*XV*DSQRT(PS-Y(9))*SHUT11 +
     1   C1A*XV*DSQRT(Y(9))*SHUT21
C
      QB=-C1B*XV*DSQRT(Y(10))*SHUT11 -
     1    C1B*XV*DSQRT(PS-Y(10))*SHUT21
C
      DERY(9)=(BMA/(VOA + AA*Y(2)))*
     1         (QA - AA*Y(1) - EKPA*(Y(9)-Y(10)))
C
      DERY(10)=(BMB/(VOB - AB*Y(2)))*
     1          (QB + AB*Y(1) - EKPB*(Y(10)-Y(9)))



C     Compute the nonlinear mass

      LHS(1,1)=M1

C     Compute the nonlinear force

      RHS(1)=-M1*G -V1*Y(1) -K1*DSIGN(1.D0,Y(1)) +TOR(1,1)
     1        + Y(9)*AA - Y(10)*AB

C     Compute the acceleration terms
```

```
      ACCEL(1,1)=RHS(1)/LHS(1,1)

C     Replace ACCEL with DERY and velocity terms

      DERY(1)=ACCEL(1,1)
      DERY(2)=Y(1)

C     Form the WT matrix

      WT(1,1)=ACCEL(1,1) + G
      WT(2,1)=DSIGN(1.D0,Y(1))
      WT(3,1)=Y(1)

C     Form the Gamma Matrix

      DO 30 I=1,3
        DO 30 J=1,3

30        GAM(I,J)=0.D0

      GAM(1,1)=PA(3)*(DABS(E(1)))
      GAM(2,2)=PA(6)*(DABS(E(1)))
      GAM(3,3)=PA(9)*(DABS(E(1)))

C     Compute the inverse of MHAT

      MHI(1,1)=1.0/MHAT(1,1)

C     Form the parameter update vector

      CALL ALMUL(MHI,E1,TMP1,1,1,1)
      CALL ALMUL(WT,TMP1,TMP2,3,1,1)
      CALL ALMUL(GAM,TMP2,PHATD,3,3,1)

      DO 50 I=1,3

        DERY(I+2)=PHATD(I,1)
```

```
50          CONTINUE


            IF (PA(7).GT.1.0) THEN

C       Reset parameters

            DELMA=0.025
            DELDA=1.0

C       Lower bounds

            LOW(1)=1.05
            LOW(2)=30.0
            LOW(3)=65.0

C       Upper bounds

            HI(1)=1.05
            HI(2)=34.0
            HI(3)=70.0

C       Reset

            IF(Y(3).LE.(LOW(1)-DELMA)) THEN

             DERY(3)=0.0
                Y(3)=LOW(1)
             PRINT*,' PARA # ',1,' SATURATED AT T= ', T

            ELSE IF(Y(3).GE.(HI(1)+DELMA)) THEN

             DERY(3)=0.0
                Y(3)=HI(1)
             PRINT*,' PARA # ',1,' SATURATED AT T= ', T

            ENDIF

            IF(Y(4).LE.(LOW(2)-DELDA)) THEN
```

```
      DERY(4)=0.0
         Y(4)=LOW(2)
      PRINT*,' PARA # ',2,' SATURATED AT T= ', T

     ELSE IF(Y(4).GE.(HI(2)+DELDA)) THEN

      DERY(4)=0.0
         Y(4)=HI(2)
      PRINT*,' PARA # ',2,' SATURATED AT T= ', T

     ENDIF

     IF(Y(5).LE.(LOW(3)-DELDA)) THEN

      DERY(5)=0.0
         Y(5)=LOW(3)
      PRINT*,' PARA # ',3,' SATURATED AT T= ', T

     ELSE IF(Y(5).GE.(HI(3)+DELDA)) THEN

      DERY(5)=0.0
         Y(5)=HI(3)
      PRINT*,' PARA # ',3,' SATURATED AT T= ', T

     ENDIF
    ENDIF


    RETURN
    END

    SUBROUTINE HAT(Y,MHAT,QHAT)

C       This subroutine is used to compute an approximation to the
c       mass matrices and right hand side vector.
c       The unknown parameters are:
c
c          Y3 - M1
```

```
C         Y4 - K1
C         Y5 - V1
C

          REAL *8 Y(10),MHAT(1,1),QHAT(1,1)
          REAL *8 M1,K1,V1,G,AA,AB

C      Convert Ys for ease of use

          M1=Y(3)
          K1=Y(4)
          V1=Y(5)
          G=386.4D0
          AA=4.04D0
          AB=3.25D0

C      Compute the nonlinear mass MHAT

           MHAT(1,1)=M1

C      Compute the nonlinear force QHAT

           QHAT(1,1)=-M1*G -V1*Y(1) -K1*DSIGN(1.0D0,Y(1))
     1                + Y(9)*AA - Y(10)*AB

          RETURN
          END


          SUBROUTINE ALMUL(ARRAYA,ARRAYB,ARRAYC,M,N,NC)

          INTEGER M,N,NC,I,J,K

          REAL *8 ARRAYA(M,N),ARRAYB(N,NC),ARRAYC(M,NC)
          REAL *8 SUM

          DO 60 I=1,M
          DO 60 J=1,NC
```

```
            SUM=0.0

            DO 70 K=1,N

               SUM=SUM+ARRAYA(I,K)*ARRAYB(K,J)
70          CONTINUE
            ARRAYC(I,J)=SUM

60       CONTINUE

         RETURN
         END

         SUBROUTINE MATSUB(ARRAYA,ARRAYB,ARRAYC,M,N)

         INTEGER I,J,M,N

         REAL *8 ARRAYA(M,N),ARRAYB(M,N),ARRAYC(M,N)

         DO 20 I=1,M
            DO 10 J=1,N

               ARRAYC(I,J)=ARRAYA(I,J)-ARRAYB(I,J)

10          CONTINUE
20       CONTINUE

         RETURN
         END
```

9.5 Program Listing for Experimental Four Axis Pole Assignment Adaptive Control

```
      Program vhbwpole4.f
c...Designed for testing of four axis pole assignment adaptive
```

```
c   control of positech robot. 24msec rate (approx)
c
        character SccsId*(*)
        parameter (SccsId = ' @(#)anadin.f
                  6.1 (MASSCOMP) 9/5/87 ')


*  Standard parameter definitions:
        include '/usr/include/mr.f'

*  Local parameters (note explicit typing):
        integer EXREAD, EXWRIT, NEARFRQ, SQUARE, LOW
        parameter (EXREAD = 1)
        parameter (EXWRIT = 0)
        parameter (NEARFRQ = 0)
        parameter (SQUARE = 4)
        parameter (LOW = 0)

        character clockdev*(*)
        character addev*(*)
        character dadev*(*)
        character digdev*(*)

        parameter (clockdev = '/dev/dacp0/efclk3')
*  Both the AD12F and the EF12M A/D device have the same
*  device pathname so no problem here.

        parameter (addev = '/dev/dacp0/adf0')
        parameter (dadev = '/dev/dacp0/daf0')
        parameter (digdev= '/dev/dacp0/pdi0')

        integer adpn, clkpn, fchan, nchans, incr, gain, samples
        integer dapn, cvsync, digpn
        integer*2 resolver1, idataold1,resolver2,idataold2
        integer*2 resolver3, idataold3,resolver4,idataold4


        integer byteslocked
        integer status(2)
```

```
      real position1(5000), control1(5000),position2(5000)
      real control2(5000),position3(5000),control3(5000)
      real position4(5000),control4(5000)

      real param11(5000),param12(5000),param21(5000)
      real param22(5000),param31(5000),param32(5000)
      real param41(5000),param42(5000),param43(5000)
      real param44(5000)

      integer*2 ui(48),yi(12)
      real trigfreq, fret, wret

      dimension theta1(10),phi1(1,4),u1(5000),y1(5000),p1(4,4)
      dimension r(4,1),pnnt(1,1),c(10,10),pp(4,4),phit(4,1)
      dimension pt(4,1),ptt(4,4),uii(4,4),con(1,1),pn(4,4)

      dimension theta2(10),phi2(1,4),u2(5000),y2(5000),p2(4,4)

      dimension theta3(10),phi3(1,4),u3(5000),y3(5000),p3(4,4)

      dimension theta4(10),phi4(1,4),u4(5000),y4(5000),p4(4,4)

*  Force long-word alignment of data buffer:
      common yi
      common /daocom/ui

*  Init path numbers to -1 so system will assign them:
      data dapn /-1/
      data clkpn /-1/
      data digpn /-1/
      data adpn /-1/


*  Lock this process's text and data segments into main memory:
      call mrlock(0, 0, byteslocked)
      print *, byteslocked, ' bytes locked in memory'

*  Open paths to AD12F and CK10 clock or EF12M devices:
      call mropen(digpn, digdev, EXREAD)
```

```
      print *, digdev, ' opened, pathno=', digpn
      call mropen(dapn, dadev, EXWRIT)
      print *, dadev, ' opened, pathno=', dapn
      call mropen(clkpn, clockdev, EXREAD)
      print *, clockdev, ' opened, pathno=', clkpn

*  Set up clock to generate 12 100kHz commands to the D/A device:
      trigfreq =100000.
      print *, 'Setting clock to ', trigfreq, 'Hz'
      call mrclk1(clkpn, NEARFRQ, trigfreq, fret, SQUARE, 0.0,
     1wret, LOW)
      print *, 'Clock set up freq=', fret, ' width=', wret


*  Output 12 samples from D/A channel 1 for each loop.
*  mradxin arms the clock so we don't have to.
      fchan = 0
      nchans = 4
      incr = 1
      gain = 1
      samples = 12
      sample2=2
      cvsync=1


*       This is a Recursive least squares estimation program
*       to identify the system using input and output information.
*
*
*        print *,' input the no. of measurements '
*       read(5,*)n
*
*        print *,' input the no. of parameters '
*        read(5,*)nr
       nr=4
*
       print *,' input the initial guess for parameters v'
       do 1012 i=1,nr
        read(5,*)theta1(i)
```

```
1012   continue

       print *,' input the initial guess for parameters h'
       do 1081 i=1,nr
        read(5,*)theta2(i)
1081   continue

       print *,' input the initial guess for parameters b'
       do 1090 i=1,nr
        read(5,*)theta3(i)
1090   continue

       print *,' input the initial guess for parameters w'
       do 1099 i=1,nr
        read(5,*)theta4(i)
1099   continue


*
       print *,' input the covariance of P matrix v'
       read(5,*)cov1

       print *,' input the covariance of P matrix h'
       read(5,*)cov2

       print *,' input the covariance of P matrix b'
       read(5,*)cov3

       print *,' input the covariance of P matrix w'
       read(5,*)cov4

*
       do 1013 i=1,nr
        do 2013 j=1,nr
          p1(i,j)=0.0
          p2(i,j)=0.0
          p3(i,j)=0.0
          p4(i,j)=0.0
2013     continue
```

```
 1013   continue
*
        do 2019 i=1,nr
         p1(i,i)=cov1
         p2(i,i)=cov2
         p3(i,i)=cov3
         p4(i,i)=cov4
 2019   continue
*
*
*       to perform recursive identification
*
*        nn=nr/2
*
        nn=2
       print *,' number of a parameters: nn=',nn

        do 5010 i=1,nn
         y1(i)=0.0
         y2(i)=0.0
         y3(i)=0.0
         y4(i)=0.0
 5010   continue

        do 5020 i=1,nn
         u1(i)=0.0
         u2(i)=0.0
         u3(i)=0.0
         u4(i)=0.0
5020    continue

        print *,' transfer beginning '
*  simulated closed loop output

        sum=0
        iloop=1
        k=nn

        call mrpeimod(digpn,0,1,1,1,1,-1,-1)
```

```
      call mrpdxin (digpn, samples, yi)
      call mrevwt  (digpn, status, 5000)

      param11(iloop)=theta1(1)
      param12(iloop)=theta1(2)
      param21(iloop)=theta2(1)
      param22(iloop)=theta2(2)
      param31(iloop)=theta3(1)
      param32(iloop)=theta3(2)
      param41(iloop)=theta4(1)
      param42(iloop)=theta4(2)

c...yset is the desired position
      yseth = 1.0
      ysetv1 = -1.0
      ysetv2 = -2.0
      ysetv3 = -3.0
      ysetv4 = -4.0
      ysetv5 = -5.0
      ysetv6 = -6.0
      ysetv7 = -7.0
      ysetv8 = -8.0
      ysetv9 = -9.0
      ysetb =-0.30
      ysetw1 =-0.30
      ysetw2 =-0.60
      ysetw3 =-0.30
      ysetw4 =-0.60
      ysetw5 =-0.30

      do 401 ijkl=1,5
      yseth=((-1.0)**(ijkl+1))*1.0

      if(ijkl.eq.1)ysetv=ysetv1
      if(ijkl.eq.2)ysetv=ysetv3
      if(ijkl.eq.3)ysetv=ysetv5
      if(ijkl.eq.4)ysetv=ysetv7
      if(ijkl.eq.5)ysetv=ysetv9
      if(ijkl.eq.6)ysetv=ysetv7
```

```
        if(ijkl.eq.7)ysetv=ysetv9
        if(ijkl.eq.8)ysetv=ysetv7
        if(ijkl.eq.9)ysetv=ysetv9
        ysetb=((-1.0)**(ijkl+1))*(-0.30)
        if(ijkl.eq.1)ysetw=ysetw1
        if(ijkl.eq.2)ysetw=ysetw2
        if(ijkl.eq.3)ysetw=ysetw3
        if(ijkl.eq.4)ysetw=ysetw4
        if(ijkl.eq.5)ysetw=ysetw5

*        print*,'yset=',yset

        do 400 kount=1,400
        iloop=iloop+1
*       call mrpdxin (digpn, samples, yi)
*       call mrevwt (digpn, status, 5000)

*        call mradxin(adpn,clkpn,-1,fchan,nchans,incr,gain,
*     1samples,yi)
*        call mrevwt(adpn,status,50)
*        yp=(yi(2)/409.6)
*        if(k.le.1)yp=0.0

*   control subroutine goes here

c...yset is the desired position
        call mrpdxin (digpn, samples, yi)
        call mrevwt (digpn, status, 5000)

*   control subroutine goes here

        flag = 0
        flag1 = 0
        flag2 = 0
        flag3 = 0
        flag4 = 0
        do 467 kplace =5,12
        if (flag.ne.4) then
```

```
            if (yi(kplace).le.4096) then
*               print*,yi(kplace)
               if (flag1.eq.0) then
                  resolver1 = yi(kplace)
                  flag1 = 1
*                  kplace1 = kplace
               endif

            else if (yi(kplace).ge.24576) then
                    if (flag4.eq.0) then
                       resolver4 = yi(kplace) - 24576
                       flag4 = 1
                    endif

                 else if (yi(kplace).ge.16384) then
                         if (flag2.eq.0) then
                             resolver2 = yi(kplace) - 16384
                             flag2 = 1
                         endif

                      else if (yi(kplace).ge.8192) then
                              if (flag3.eq.0) then
                                  resolver3 = yi(kplace) - 8192
                                  flag3 = 1
                              endif

            endif
             flag = flag1 + flag2 + flag3 + flag4
         endif
 467     continue

         if(kount.eq.1.and.ijkl.eq.1) idataold1=resolver1
         if(kount.eq.1.and.ijkl.eq.1) idataold2=resolver2
         if(kount.eq.1.and.ijkl.eq.1) idataold3=resolver3
         if(kount.eq.1.and.ijkl.eq.1) idataold4=resolver4

         call inchesv(resolver1,idataold1,sum1,yp1)
         call inchesh(resolver2,idataold2,sum2,yp2)
         call radiansb(resolver3,idataold3,sum3,yp3)
```

```
      call radiansw(resolver4,idataold4,sum4,yp4)

      y1(k+1)=yp1
      y2(k+1)=yp2
      y3(k+1)=yp3
      y4(k+1)=yp4


       call ident(k,nn,nr,p1,y1,u1,theta1)
       call ident(k,nn,nr,p2,y2,u2,theta2)
       call ident(k,nn,nr,p3,y3,u3,theta3)
       call ident(k,nn,nr,p4,y4,u4,theta4)


       ea11=theta1(1)
       ea12=theta1(2)
       eb10=theta1(3)
       eb11=theta1(4)

       el10=1.0

       el11=(eb11*eb11*(-2.0176-ea11) -eb10*eb11*(1.3346-ea12)
     1 +eb10*eb10*
     2 (-0.2879))/(eb11*(eb11-(eb10*ea11)) +eb10*eb10*ea12)

       ep10=(-2.0176 - ea11 - el11)/(eb10)

       ep11=(-0.2879 - (ea12*el11))/(eb11)

       eh10=(1-2.0176+1.3346-0.2879)/(eb10 + eb11)

       u1(k+1)=(eh10*ysetv -(ep10*y1(k+1)) -(ep11*y1(k))
     1 -(el11*u1(k)))


       ea21=theta2(1)
       ea22=theta2(2)
       eb20=theta2(3)
       eb21=theta2(4)
```

```
      el20=1.0

      el21=(eb21*eb21*(-2.0176-ea21) -eb20*eb21*(1.3346-ea22)
1     +eb20*eb20*
2     (-0.2879))/(eb21*(eb21-(eb20*ea21)) +eb20*eb20*ea22)

      ep20=(-2.0176 - ea21 - el21)/(eb20)

      ep21=(-0.2879 - (ea22*el21))/(eb21)

      eh20=(1-2.0176+1.3346-0.2879)/(eb20 + eb21)

      u2(k+1)=(eh20*yseth -(ep20*y2(k+1)) -(ep21*y2(k))
1    -(el21*u2(k)))

      ea31=theta3(1)
      ea32=theta3(2)
      eb30=theta3(3)
      eb31=theta3(4)

      el30=1.0

      el31=(eb31*eb31*(-1.7857-ea31) -eb30*eb31*(0.97435-ea32)
1     +eb30*eb30*
2     (-0.16575))/(eb31*(eb31-(eb30*ea31)) +eb30*eb30*ea32)

      ep30=(-1.7857 - ea31 - el31)/(eb30)

      ep31=(-0.16575 - (ea32*el31))/(eb31)

      eh30=(1-1.7857+0.97435-0.16575)/(eb30 + eb31)

      u3(k+1)=(eh30*ysetb -(ep30*y3(k+1)) -(ep31*y3(k))
1    -(el31*u3(k)))

      ea41=theta4(1)
      ea42=theta4(2)
      eb40=theta4(3)
```

```
      eb41=theta4(4)

      el40=1.0

      el41=(eb41*eb41*(-1.8176-ea41) -eb40*eb41*(1.03108-ea42)
     1 +eb40*eb40*
     2 (-0.17274))/(eb41*(eb41-(eb40*ea41)) +eb40*eb40*ea42)

      ep40=(-1.8176 - ea41 - el41)/(eb40)

      ep41=(-0.17274 - (ea42*el41))/(eb41)

      eh40=(1-1.8176+1.03108-0.17274)/(eb40 + eb41)

      u4(k+1)=(eh40*ysetw -(ep40*y4(k+1)) -(ep41*y4(k))
     1 -(el41*u4(k)))


      if(u1(k+1).gt.(2040/409.6))u1(k+1)=(2040/409.6)
      if(u1(k+1).lt.(-2040/409.6))u1(k+1)=(-2040/409.6)

      if(u2(k+1).gt.(2040/409.6))u2(k+1)=(2040/409.6)
      if(u2(k+1).lt.(-2040/409.6))u2(k+1)=(-2040/409.6)

      if(u3(k+1).gt.(2040/409.6))u3(k+1)=(2040/409.6)
      if(u3(k+1).lt.(-2040/409.6))u3(k+1)=(-2040/409.6)

      if(u4(k+1).gt.(2040/409.6))u4(k+1)=(2040/409.6)
      if(u4(k+1).lt.(-2040/409.6))u4(k+1)=(-2040/409.6)


*       print *,k,u(k),y(k),theta(1),theta(2)

*   ucontrol should be control value in volts
      ucontrol1=u1(k+1)
      ucontrol2=u2(k+1)
      ucontrol3=u3(k+1)
      ucontrol4=u4(k+1)
```

```
* proportional control for checking:
*        ucontrol=(yset-yp)
*        print*,'ucontrol,yp',ucontrol,yp

*   5 volts=2047 bits
*   -5 volts=-2048 bits
        ubits1=409.6*ucontrol1
        ubits2=409.6*ucontrol2
        ubits3=409.6*ucontrol3
        ubits4=409.6*ucontrol4


*   da output wraps around, so include saturation here
        if(ubits1.gt.2040) ubits1=2040
        if(ubits1.lt.-2040) ubits1=-2040

        if(ubits2.gt.2040) ubits2=2040
        if(ubits2.lt.-2040) ubits2=-2040

        if(ubits3.gt.2040) ubits3=2040
        if(ubits3.lt.-2040) ubits3=-2040

        if(ubits4.gt.2040) ubits4=2040
        if(ubits4.lt.-2040) ubits4=-2040


        do 200 i=0,11
         ii=nchans*i
        ui(ii+1)=int(ubits1)
        ui(ii+2)=int(ubits2)
        ui(ii+3)=int(ubits3)
        ui(ii+4)=int(ubits4)
 200    continue
        call mrdaxout(dapn, clkpn, -1, 0, nchans, incr,
&               samples, ui, cvsync)
        call mrevwt(dapn, status, 5000)

*   position in inches; control in volts
```

```
        position1(iloop)=yp1
        position2(iloop)=yp2
        position3(iloop)=yp3
        position4(iloop)=yp4
        control1(iloop)=ucontrol1
        control2(iloop)=ucontrol2
        control3(iloop)=ucontrol3
        control4(iloop)=ucontrol4
        param11(iloop)=theta1(1)
        param12(iloop)=theta1(2)
        param21(iloop)=theta2(1)
        param22(iloop)=theta2(2)
        param31(iloop)=theta3(1)
        param32(iloop)=theta3(2)
        param41(iloop)=theta4(1)
        param42(iloop)=theta4(2)
        k=k+1
400     continue
401     continue
        print *,' end of control loop; HYDRAULICS MUST BE OFF'

*  Close devices:
        call mrclosall
        call store(position1,control1,position2,control2,position3,
     1             control3,position4,control4)
        call store1(param11,param12,param21,param22,param31,param32,
     1              param41,param42)
*
*       stop
      end



        subroutine inchesh(resolver,idataold,sum,rinches)

        integer*2 resolver,ibits,idataold,rev

        ibits=resolver

        idiff=ibits-idataold
```

```
if(idiff.lt.3600.and.idiff.gt.-3600) then
        sum=sum + (idiff*2.473695)/4095
endif

if(idiff.gt.3600) then
        idiff=(ibits-4095-idataold)
        sum=sum + (idiff*2.473695)/4095
endif

if(idiff.lt.-3600) then
        idiff=(ibits-(idataold-4095))
        sum=sum + (idiff*2.473695)/4095
endif

idataold=ibits

rinches=-sum

return
end

subroutine inchesv(resolver,idataold,sum,rinches)

integer*2 resolver,ibits,idataold,rev

ibits=resolver

idiff=ibits-idataold

if(idiff.lt.3600.and.idiff.gt.-3600) then
  sum = sum + (idiff*2.473695)/4095
endif

if(idiff.gt.3600) then
  idiff=(ibits-4095-idataold)
  sum = sum + (idiff*2.473695)/4095
endif
```

```
if(idiff.lt.-3600) then
  idiff=(ibits-(idataold-4095))
  sum = sum + (idiff*2.473695)/4095
endif

idataold=ibits

rinches=sum

return
end

subroutine radiansb(resolver,idataold,sum,radian)

integer*2 resolver,ibits,idataold,rev

ibits=resolver

idiff=ibits-idataold

if(idiff.lt.3600.and.idiff.gt.-3600) then
  sum=sum + (idiff*2.1038629)/4095
endif

if(idiff.gt.3600) then
  idiff=(ibits-4095-idataold)
  sum=sum + (idiff*2.1038629)/4095
endif

if(idiff.lt.-3600) then
  idiff=(ibits-(idataold-4095))
  sum=sum + (idiff*2.1038629)/4095
endif

idataold=ibits

radian=sum

return
```

```
      end

      subroutine radiansw(resolver,idataold,sum,radian)

      integer*2 resolver,ibits,idataold,rev

      ibits=resolver

      idiff=ibits-idataold

      if(idiff.lt.3600.and.idiff.gt.-3600) then
        sum=sum + (idiff*6.1956484)/4095
      endif

      if(idiff.gt.3600) then
        idiff=(ibits-4095-idataold)
        sum=sum + (idiff*6.1956484)/4095
      endif

      if(idiff.lt.-3600) then
        idiff=(ibits-(idataold-4095))
        sum=sum + (idiff*6.1956484)/4095
      endif

      idataold=ibits

      radian=sum

      return
      end



      subroutine store(position1,control1,position2,control2,
     1                 position3,control3,position4,control4)
      real position1(5000),control1(5000),position2(5000)
      real control2(5000),position3(5000),control3(5000)
      real position4(5000),control4(5000)
```

```
open (8, FORM='unformatted', FILE='position1.001')
rewind(8)
write(8) position1
close(8)

open (8, FORM='unformatted', FILE='control1.001')
rewind(8)
write(8) control1
close(8)

open (8, FORM='unformatted', FILE='position2.001')
rewind(8)
write(8) position2
close(8)

open (8, FORM='unformatted', FILE='control2.001')
rewind(8)
write(8) control2
close(8)

open (8, FORM='unformatted', FILE='position3.001')
rewind(8)
write(8) position3
close(8)

open (8, FORM='unformatted', FILE='control3.001')
rewind(8)
write(8) control3
close(8)

open (8, FORM='unformatted', FILE='position4.001')
rewind(8)
write(8) position4
close(8)

open (8, FORM='unformatted', FILE='control4.001')
rewind(8)
write(8) control4
```

```
      close(8)

      return
      end

      subroutine store1(param11,param12,param21,param22,param31,
     1                  param32,param41,param42)
      real param11(5000),param12(5000),param21(5000)
      real param22(5000),param31(5000),param32(5000)
      real param41(5000),param42(5000)

      open (8, FORM='unformatted',FILE='param11.001')
      rewind(8)
      write(8) param11
      close(8)

      open (8, FORM='unformatted',FILE='param12.001')
      rewind(8)
      write(8) param12
      close(8)

      open (8, FORM='unformatted',FILE='param21.001')
      rewind(8)
      write(8) param21
      close(8)

      open (8, FORM='unformatted',FILE='param22.001')
      rewind(8)
      write(8) param22
      close(8)

      open (8, FORM='unformatted',FILE='param31.001')
      rewind(8)
      write(8) param31
      close(8)

      open (8, FORM='unformatted',FILE='param32.001')
      rewind(8)
      write(8) param32
```

```
      close(8)

      open (8, FORM='unformatted',FILE='param41.001')
      rewind(8)
      write(8) param41
      close(8)

      open (8, FORM='unformatted',FILE='param42.001')
      rewind(8)
      write(8) param42
      close(8)



      return
      end


       subroutine almul(arraya,arrayb,arrayc,m,n,nc)
       integer m,n,nc,i,j,k
       real arraya(m,n),arrayb(n,nc),arrayc(m,nc)
       real sum
*
       do 60 i=1,m
       do 60 j=1,nc
         sum=0.0
         do 70 k=1,n
          sum=sum+arraya(i,k)*arrayb(k,j)
 70      continue
         arrayc(i,j)=sum
60     continue
*
       return
       end
       subroutine trans(arraya,arrayb,m,n)
       integer i,j,m,n
       real arraya(m,n),arrayb(n,m)
*
       do 20 i=1,m
```

```
          do 10 j=1,n
           arrayb(j,i)=arraya(i,j)
 10       continue
 20      continue
*
         return
         end

        subroutine ident(k,nn,nr,p,y,u,theta)
        dimension theta(10),phi(1,4),u(5000),y(5000),p(4,4)
        dimension r(4,1),pnnt(1,1),c(10,10),pp(4,4),phit(4,1)
        dimension pt(4,1),ptt(4,4),uii(4,4),con(1,1),pn(4,4)


         do 2000 i=1,nn
          phi(1,i)=-y(k+1-i)
 2000     continue
         do 2022 i=1,nn
          phi(1,i+nn)=u(k+1-i)
 2022    continue
*
         call trans(phi,phit,1,nr)
         call almul(p,phit,pt,nr,nr,1)
*
         call almul(phi,pt,con(1,1),1,nr,1)
*
         do 2002 iij=1,nr
 2002    r(iij,1)=pt(iij,1)/(1+con(1,1))
*
         call almul(r,phi,ptt,nr,1,nr)
*
         do 2003 i1=1,nr
         do 2003 i2=1,nr
         uii(i1,i2)=-ptt(i1,i2)
 2003    if(i1.eq.i2)uii(i1,i2)=1.0-ptt(i1,i2)
*
         call almul(uii,p,pn,nr,nr,nr)
*
         do 2005 il=1,nr
```

```
      do 2005 ik=1,nr
 2005  p(il,ik)=pn(il,ik)
*
      call almul(phi,theta,pnnt(1,1),1,nr,1)
*
      do 2004 it=1,nr
 2004 theta(it)=theta(it)+r(it,1)*(y(k+1)-pnnt(1,1))
*

      return
      end
```