A SYNTHESIS METHOD FOR ADAPTIVE CONTROL OF NONLINEAR SYSTEMS WITH APPLICATION TO A ONE DEGREE-OF-FREEDOM MOTION SIMULATOR

By

AMOS BURSHTEIN

Bachelor of Science
in Mechanical Engineering
Israel Institute of Technology
Haifa, Israel
1962

Master of Science
in Civil Engineering
University of Minnesota
Minneapolis, Minnesota
1971

Submitted to the faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements
for the degree of
DOCTOR OF PHILOSOPHY
December, 1984

A SYNTHESIS METHOD FOR ADAPTIVE CONTROL OF NONLINEAR SYSTEMS WITH APPLICATION TO A ONE DEGREE-OF-FREEDOM MOTION SIMULATOR

Thesis Approved:

Karl N. Reid
Thesis Adviser

Robert L. Swain

James R. Rowland

Gary E. Young

Norman N. Durham
Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my gratitude to the people who assisted me in this work and to those who supported me during my studies at Oklahoma State University.

Professor Karl N. Reid, my major advisor, encouraged me to resume graduate studies, laid aside in 1971 and taken up afresh in 1980. He provided the valuable guidance and direction which brought this thesis to a conclusion.

Professor Robert L. Swaim gave freely of his time to solve several fundamental problems.

Professor J. R. Rowland and Professor G. E. Young both contributed helpful suggestions and constructive criticism.

Burtek, Inc., Tulsa, Oklahoma made this research possible by providing me flexible work hours and permission to use the company computer. Fellow employees at Burtek who offered much support and encouragement are too numerous to mention. I am especially indebted to Mr. A. Nick, System Department Manager, Mr.M. Westerby and Mr. B. Whatcott for their full support and encouragement, to Mr. A. Catchpole for his computer support, to Mr. M. Pilditch for providing the plotting routine, to Mr. L. S. Ma for his help in the hardware tests, and to Dr. R. Prasad for his supervision and proof-reading of the word-processing task.

Finally I want to thank my wife Nili for her devotion, encouragement and help; and my children, Iris, Eran and Tali for the many sacrifices they made during the last four years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

A	An nxn System Matrix
a_i (i=1,4)	Orifice Area (in section 4.1)
A_i (i=1,2)	Actuator Area (in section 4.1)
B	An nxm Distribution Matrix of Inputs
g	Gravitational Acceleration (381 in/sec^2)
I	Control Input to Servo Valve (ma)
J	Performance Cost Function
K	Gain Matrix
P_i (i=i,2)	Pressure (psi)
P_v	Pressure Drop Across Valve (psi)
P_m	Load Pressure (psi)
p	A Co-state Vector in Optimal Control Theory
Q	A Symmetric Positive Weighting Matrix
Q_i (i=1,2)	Volumetric Flow (in^3/sec)
R	A Symmetric Positive Weighting Matrix
r (or R)	System Input Vector
T,t	Time Variable
U or u	Total Control Vector
U_i (i=1,m)	A Component of the Control Vector
u*	Optimal Control
V_i (i=1,2)	Volume of Actuator Chamber
X	Total Space Vector

x_i (i=1,n)	A Component of the State Vector
$\hat{X}$	Estimated Model State Vector
$\dot{X}_L$	A Velocity Limit of the Actuator
X_L	A Position Corresponding to $\dot{X}_L$
X_{max}	Maximum Allowed Displacement of Actuator
Z	Measurement Vector
α	A Scalar Parameter
β	Bulk Modulus of Fluid (Lb/in^2)
ε	A Scalar Parameter or Error
ρ	Density of Fluid
ϕ	Analytic Vector Function (Section 1.3.2.1) or System Parameter Vector Elsewhere
$\hat{\phi}$	Estimated Parameter Vector
ω_n	Undamped Natural Frequency of a System (rad/sec)

CHAPTER I

INTRODUCTION AND BACKROUND

Classical control system design is generally a trial-and-error process in which various methods of analysis are used iteratively to determine the design parameters of an "acceptable" system. A new and direct approach to the synthesis of complex, integrated control systems has been made feasible by advances in digital computers.

The application of modern control theory to the synthesis of systems that include electro-hydraulic servo components has been limited. Hydraulic subsystems have typically been designed separately from the rest of the system, without regard to impact on overall system performance.

This thesis discusses a technique for the design of on-line adaptive controllers for nonlinear systems which incorporate electro-hydraulic drives. Special emphasis is given to the development of an integrated control system for a one degree of freedom motion platform in which the highly non-linear equations of motion form a part of the plant model.

1.1 Motion Platform

A "motion platform" provides motion and force infor-

mation to the trainee in an aircraft training simulator. The operating envelope of the ground-based motion platform is limited by economic constraints and technical considerations. Usually the system is capable of providing moderate "g" cues in the frequency band of 0.5 Hz to 4.0 Hz. It is also capable of providing useful acceleration information for maneuvers with periods of 2 seconds or more by tilting the simulator. This system is very limited in evoking the physiological effects due to high-g maneuvers. On the other hand, it is quite useful for providing vibration or short duration alerting cues. A detailed discussion on the tasks of the motion platform and how these tasks are related to the human perceptual system can be found in Reference 1. A typical, constrained, motion platform response to aircraft dynamics inputs is delineated in Figure 1. The control provides the higher frequency "onset" portion of the acceleration profiles, then returns the cockpit to a neutral position, so as to be able to display the next "onset" cue. This process is known as "washing out" the motion cues. Washout levels should be constrained to values below the trainee's indifference threshold (from Reference 1: 2 degrees/sec and 0.1g for the semicircular canals and otoliths, respectively).

The common motion platform is driven hydraulically and is controlled via a conventional electro-hydraulic servo system. Signal inputs from aircraft equations of motion

(accelerations or rates or positions according to the specific configuration of the simulator) are processed via filters and directed to the servo controller. A typical block diagram of a motion platform control system is shown in Figure 2.

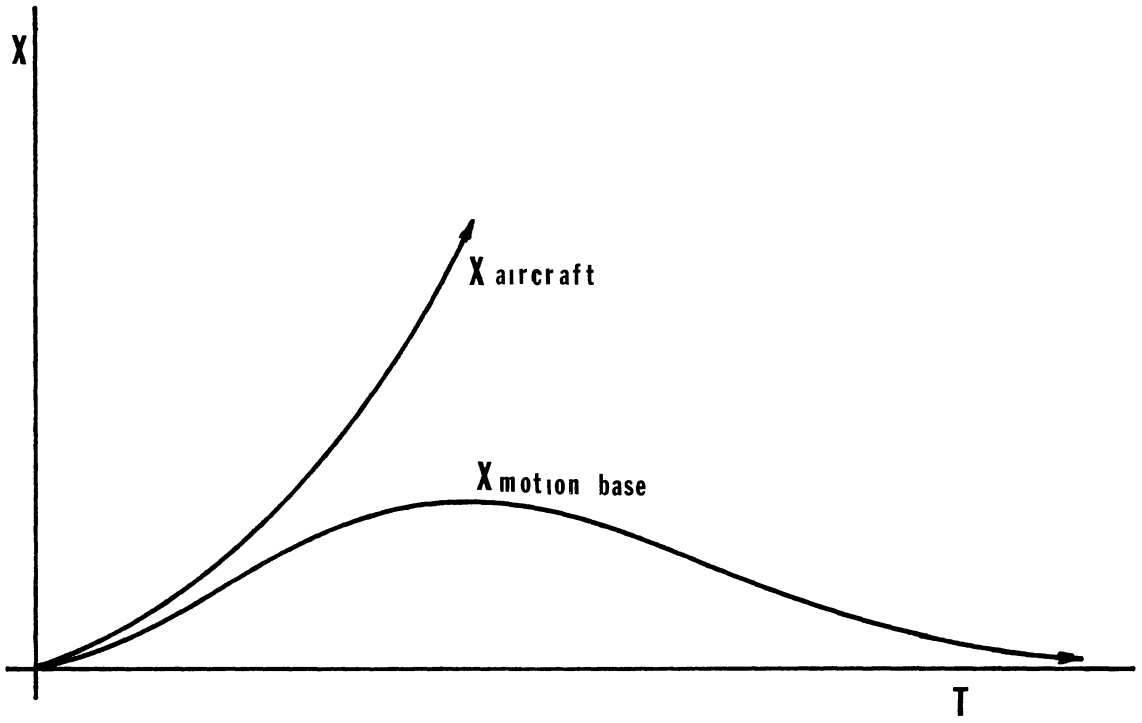

Figure 1. Typical System Response with Washout

New, modern approaches to the design of a motion platform control system (which omit the model of the hydraulic

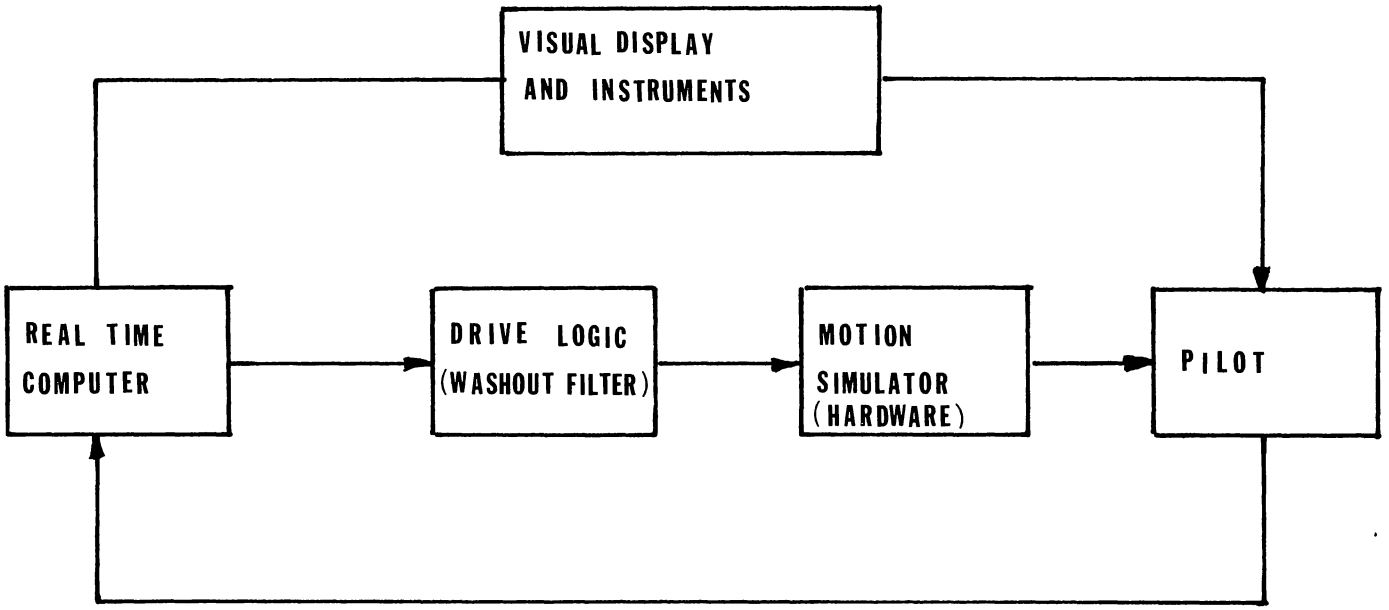

Figure 2. Block Diagram of Moving Base Simulator
(From Ref.11)

system) are documented in References 2-11. A military standard of motion system requirement is given in Reference 12.

1.2 The Role of Digital Computers in Closed Loop Electrohydraulic Control Systems

R.H. Maskrey (13) explored possible applications of microcomputer technology for expanding the utilization of electrohydraulic servos. Possible uses of computer techniques can be categorized into six areas:

a. Closing the loop. Using the computer as a closed-loop servo-controller.

b. Pre-loop processing. Processing command information ahead of a conventional closed-loop servo.

c. Peripheral processing. Where information is processed both ahead of and subsequent to a conventional closed-loop servo.

d. Adaptive control. Using intelligence to modify the basic closed-loop control process.

e. "Smart" Redundancy. Improving a redundant servo with the addition of higher level thinking.

f. Improved time-optimal control. Enhancing bang-bang control.

Areas (b), (c) and (d) above are related to this

thesis. These areas are discussed in more detail below.

1.2.1 Pre-loop Processing

Computers are in use today as pre-loop processors or as command generators for conventional analog servos. This approach is commonly used in motion platform control. The computer is used effectively to perform interface functions, to interpret keyboard readings, to process rate signal inputs, or to monitor interactions between subsystems. The computer outputs drive a high performance servo.

1.2.2 Peripheral Processing

"Peripheral Processing" involves the use of a computer to handle related information both ahead of and after a conventional closed loop. This terminology should not be confused with the computer world's use of satellite computers to do ''peripheral''data processing. An example of peripheral processing which uses a large computer with a six degrees of freedom motion simulator is given in Reference 2. As seen in Figure 3, the computer processes the information of the flight simulation, manipulates the centroid transformation, washout system and actuator extension transformation. The computer also works with output information generated as a result of the servo operation, performs inverse transformation and predicts the position limits. Based upon these calculations, the computer is able to use

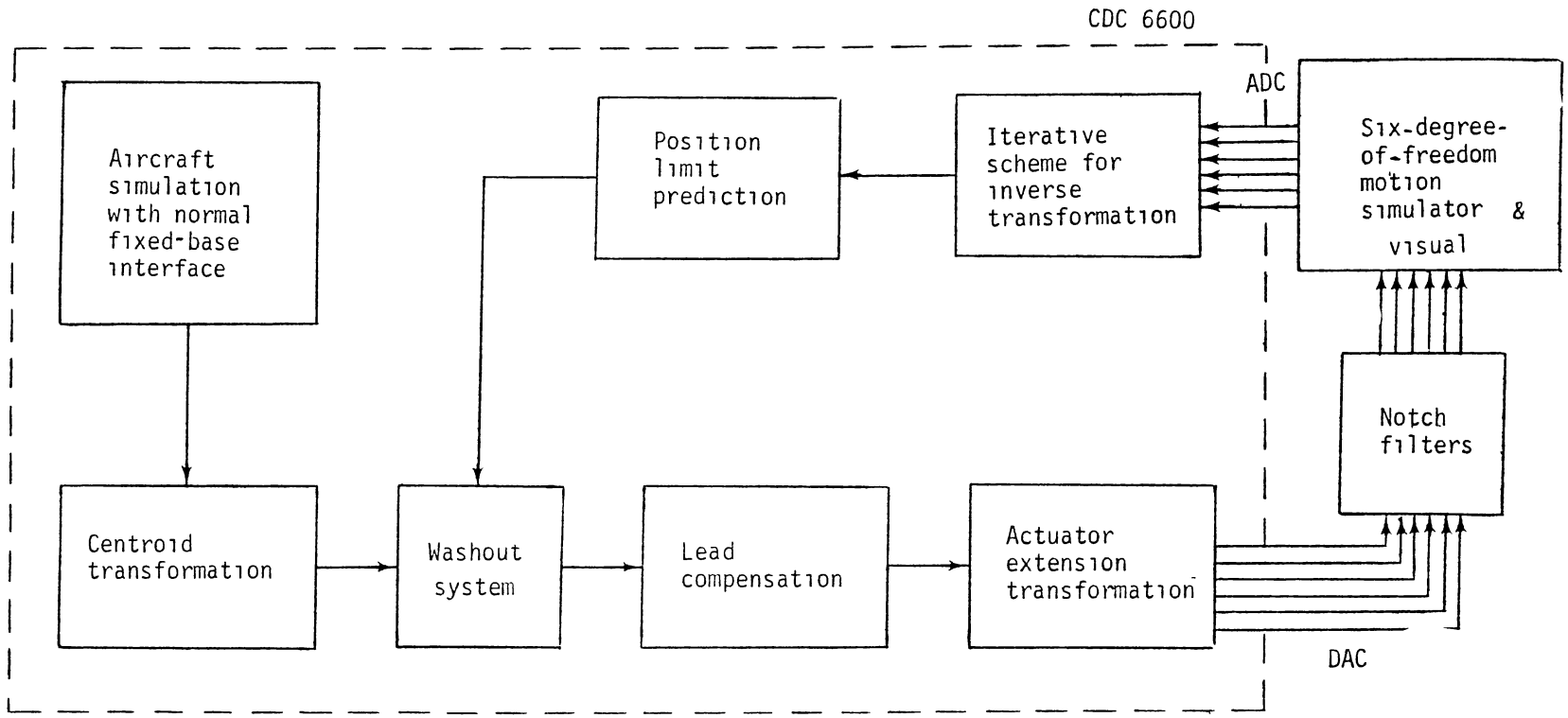

Figure 3. Major Blocks of a Real-Time Motion Simulator (From Ref.2)

the output imformation to improve the next cycle. Other cases of peripheral processing have been discussed by R. L.Kosut (6) and W.R.Sturgeon (10). In these cases, optimal control theory is used to generate an optimal controller (i.e.,linearized model with constant feedback gains) based on information about the platform dynamics. But, in each of these cases, the traditional hydraulic servo system remains in the loop, untouched.

1.2.3 Adaptive Control

If there is a "best" application of digital computers to electrohydraulic control, then adaptive control may be it. It has long been held that adaptive control requires the capabilities of a computer. A self adaptive control system is one that is able to adjust itself to provide optimal or, perhaps, consistent control in the presence of identifiable changes. Typically, adaptive control should be used where constant performance is desired even though the following changes occur.

a. Changes in the physical process being controlled with respect to time.
b. Changes that occur in the power availability, such as changes in hydraulic supply pressure due to other demands.

c. Changes in loading conditions, such as load inertia changes because of normal operation.

d. Changes of gain with amplitude.

As a result of these changes, it may be desirable to control a variety of conditions based upon historical and predicted performance of the system. These conditions could include limits, gains, boundaries for nonlinearities and even control modes.

1.3 Control System Synthesis

The design of an automatic control system generally involves the selection of additional components which usually have adjustable parameters such that the overall system meets a desired performance specification. For example, this performance specification may be formulated (performance index) in terms of the minimization of an error criterion, a settling time, an energy constraint, or, it may be simply required that the response be stable.

The performance index as frequently used in control system design provides a quantitative measure of system performance and is normally chosen to emphasize important system characteristics. This quantitative measure is very important for parameter identification, state estimation, and for the design of optimal and suboptimal control systems.

1.3.1 Feedback Control Synthesis for Deterministic Linear Systems

The equations governing the behaviour of dynamic systems of the type considered in this thesis are usually non-linear. Even in cases where a linear approximation is justified, its range of validity is likely to be limited. In this thesis nonlinear equations of motion will be used exclusively. However, in order to start the iterative computation in defining the control laws, there is a need to start with a stable control law before the first iteration is executed. This control law is calculated for the linearized system operating around a steady-state equilibrium position.

Generally, the control law for the linearized system can be calculated (provided the system is controllable) by using one of the numerous methods of classical or optimal control.

In this thesis, the excess pole specification method of C.W. Merriam (14) is used to generate the linear system control law.

1.3.2 Control Synthesis for Nonlinear Systems

General approaches for controller synthesis similar to those available for linear systems do not exist at the present time. A few of the proposed approaches to solve the above problem are reviewed in this section.

1.3.2.1 A Functional Expansion Approach to the Solution of Nonlinear Feedback Problems. Methods for synthesizing suboptimal feedback control laws for nonlinear systems based on a functional expansion technique have been proposed (for example, see References 18 and 19). These methods apply to controllable dynamical systems modelled by

$$\dot{X} = AX + \varepsilon\Phi(X) + BU \qquad X(0) = X_o \qquad (1.3.2.1\text{-}1)$$

where X, an n-vector, is the state;

U, an m-vector, is the control;

X_o is the initial state;

Φ is an analytic vector function;

A and B are constant matrices, and ε is a scalar parameter.

W.L.Garrard (18) suggested that the optimization problem is the determination of a feedback control which minimizes the index of performance

$$J = \tfrac{1}{2}\int_o^\infty (X'QX + U'RU)dt \qquad (1.3.2.1\text{-}2)$$

Q and R are symmetric positive definite matrices. The optimal control U* minimizes the scalar function

$$H(X,p,U,\varepsilon) = p'(AX + \varepsilon\Phi(X) + BU) + \tfrac{1}{2}(X'QX + U'RU) \qquad (1.3.2.1\text{-}3)$$

and is given by

$$U^* = -R^{-1}B'p \qquad (1.3.2.1\text{-}4)$$

where the costate equations are:

$$\dot{p} = -\frac{\partial H}{\partial X} = -A'p - QX - \varepsilon\Phi'_x p \qquad (1.3.2.1\text{-}5)$$

$$\lim_{t \to \infty} p(t) = 0$$

Garrard proposed the block diagram of Figure 4 for control. Garrard noted that this scheme allows the designer to easily calculate a second order approximation to the optimal control. He also stressed that this control is only useful in some domain of asymptotic stability surrounding the origin.

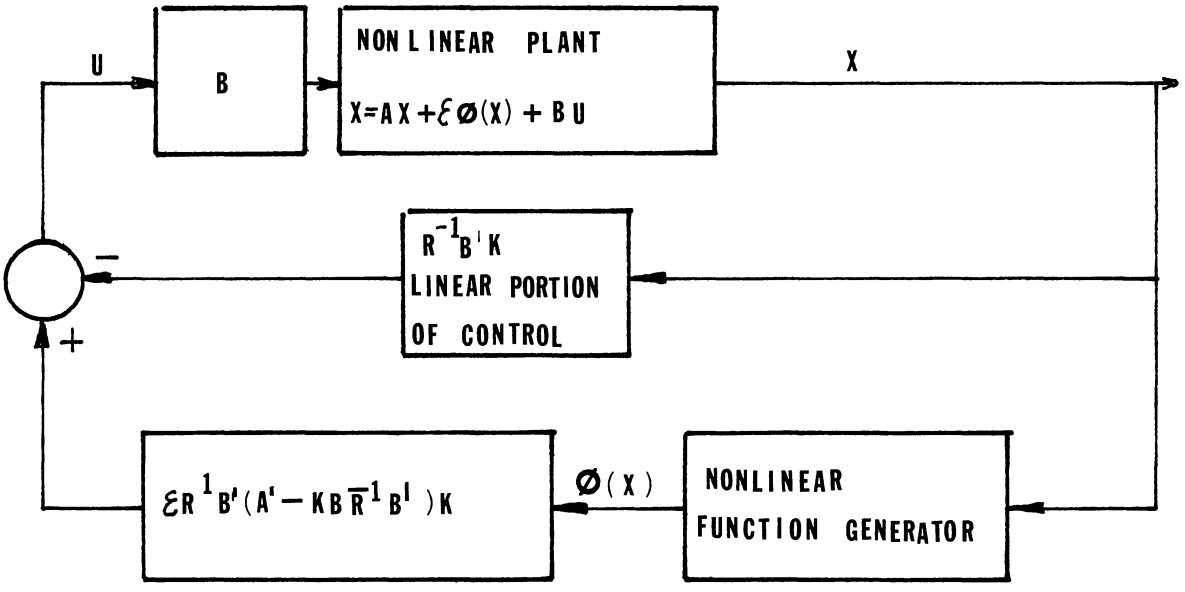

Figure 4. An Implementation of the Suboptimal Control (From Ref 18)

1.3.2.2 Optimal Control Law. The optimal controller synthesis problem can be solved utilizing Pontryagin's Maximum Principle (see also Reference 15).

Consider a controlled dynamic process given by a set of first-order differential equations:

$$\dot{x}_i = f_i(X,\phi,U,T) \qquad (1.3.2.2\text{-}1)$$

$$i = 1,2 \ldots n$$

where, x = states, ϕ = system parameters, U = control, t = time; and a performance index

$$J = \int_o^T L(X,U,r,t)dt \qquad (1.3.2.2\text{-}2)$$

which is to be minimized. The integral L(X,U,r,t) is the cost function, and represents a measure of instantaneous change from the ideal performance. The maximum principle requires that the optimal control, U* which minimizes J will maximize the scalar Hamiltonian function:

$$H(X,U,p,t) = \sum_{i=1}^{n} p_i f_i(X,U,\phi,t) + L(X,U,r,t) \qquad (1.3.2.2\text{-}3)$$

where vector p, called the costate vector, is given by:

$$\dot{p}_i = -\frac{\partial H(X,U,p,t)}{\partial x_i} \qquad (1.3.2.2\text{-}4)$$

$$i = 1,2 \ldots n$$

and

$$\dot{x}_i = \frac{\partial H(X,U,p,t)}{\partial p_i} \qquad (1.3.2.2\text{-}5)$$

The optimal trajectory x*(t) and p*(t) are found by solving the above equations with 2n boundary conditions. The optimal control U*(t) is obtained in terms of x, p and t as:

$$U^* = U(x^*,p^*,t)$$

The individual optimal control solution is dependent upon the particular choice of performance index.

The need for the solution of two point boundary value problems (TPBVP) has attracted a lot of attention in recent years. Nonlinear TPBVP's are encountered in solving optimal control problems via the maximum principle. Numerical solution of the optimal control problems may be obtained using iterative or non-iterative techniques. These methods suffer from the disadvantage however, that they yield the optimal control in the open loop form. In some cases, the problem under consideration may be sensitive to small perturbations in the initial costate and, as a result, convergence to an optimal solution may be slow if convergence occurs at all.

1.3.2.3 Direct Control Algorithm. Although a controller designed by the optimal control approach is optimal in the sense of satisfying the necessary conditions for optimality, the resultant control system is open loop and,

as such, may be very sensitive to environmental disturbances. Thus, it seems appropriate to consider methods for designing a closed-loop control system to overcome such problems. In their paper (20) H. Kaufman and R. Teavassos suggested the following method. Suppose the controller is constrained to be of the form:

$$U(t) = h[X(t),t] = KX(t) \qquad (1.3.2.3\text{-}1)$$

where K is a gain matrix whose elements must be determined. The problem is to find a finite number of constants K_i, which minimize:

$$J = \Phi[x(t_f),t_f] + \int_{t_o}^{t_f} L[x(t),h(x(t),t)]dt \qquad (1.3.2.3\text{-}2)$$

subject to the dynamic constraint given by

$$\dot{x}(t) = f[x(t),h(x(t),t),t] \qquad (1.3.2.3\text{-}3)$$

Kaufman and Teavassos developed and evaluated control computation algorithms that employ a significant degree of parallelism. Their work involved the evaluation of two approaches: optimal control and direct suboptimal control. They recommended the direct estimation algorithm should be used only if no appreciable noise exist. (No solution was given for this problem). On the other hand, if process noise is excessive, then the indirect method (optimal, two

point boundary value problem in which the process noise is regarded as control) should be considered.

With regard to the control algorithms, Kaufman and Teavassos recommended the use of the direct gain optimization procedure in view of their findings that near optimal response was obtained without an excessive amount of computation. Also, they concluded that this procedure should be seriously considered because the equations of motion of a highly nonlinear system can be utilized directlly in the control system design process.

1.4 Algorithm for Unconstrained Minimization

From the material discussed in previous sections it is evident that an efficient algorithm for minimization of a nonlinear function, called a performance index, must be employed. The optimization problem will be of the form:

minimize $f(x)$, subject to $x \in \Omega$

where f is a real-valued function and Ω, the feasible set, is a subset of E^n. In the completely unconstrained case, $\Omega = E^n$.

Usually, optimization problems have constraints, like the differential constraints ($\dot{x} = f(x,t)$) of the dynamic system; however, some of the most powerful and convenient methods of solving constrained problems involve the conversion of the problem to one of unconstrained minimization. A comprehensive discussion of nonlinear optimization can be found in References 22 and 23. A short overview of this subject is given in Appendix A, including a description of Davidon's optimization algorithm (without line search) that was used in this research.

1.4.1 Parallel Algorithms for Reducing Computation Time

The computer industry has developed parallel computers which are capable of performing the same set of instructions on many data sets simultaneously. Basically, a parallel

computer can be viewed as a set of processing elements, each of which has its own local memory and a repertoire of arithmetic and logical instructions. The role of the central processor is to coordinate the efforts of each processing element while its local memory is used for temporarily storing intermediate results.

Due to the availability of parallel processors, a number of nonlinear estimation and control algorithms (for parallel processing) have been proposed. Applications of quasi-Newton algorithms (for parallel processing) are reviewed in Reference 20, while work on square-root algorithms for optimal estimation (linear systems) is reported in Reference 26.

1.5 Motivation for the Research

Electrohydraulic subsystems normally are designed separately from the rest of the system, and then experimentally tuned to give the "best" performance for a specific limited operational envelope. References 1-11 ignored the role of the electrohydraulic servo dynamics in the control system design. R.L.Kosut (6) recommended the inclusion of actuator dynamics in the system model when the controller is designed analytically.

There is a need for an integrated control system design scheme that includes the hydraulic subsystem dynamics in the overall plant model. This subsystem can be modelled in its

natural state, that is using variables such as pressures, velocities and displacements, that are readily available for measurement.

The electrohydraulic servo system is highly nonlinear, with parameters that can only be assumed e.g bulk modulus of fluid, valve gain which changes near saturation, is also different for pressure port and return port if the actuator is unsymmetrical. A procedure is needed to determine a control law which is adaptive in nature, so that performance is consistent regardless of the nonlinearities and drifts in parameter values.

Another important design consideration is the washout motion for the specific application. If the computer is used for on-line control, it may perform decisions based upon state measurement and known hardware limitations and generate the inputs required for washout motion.

1.6 Research Objectives and Contributions

The objectives of this research were as follows:

1. To develop computationally efficient procedures for on-line identification and control of nonlinear dynamical systems with time varying inputs.
2. To formulate a washout strategy for a one degree of freedom motion platform.

3. To develop a state-space model of the motion platform system that will include the nonlinear actuator dynamics.
4. To simulate the proposed control system off-line for various inputs and modes of operation (e.g. follower mode, washout mode) and to evaluate the results.

The principal results from conducting this research are:

1. A strategy has been developed for the adaptive control of nonlinear systems using preselected integration intervals. (Objective 1)
2. The Q-N optimization method without line search has been shown to be an efficient method for solving the dynamical optimization problem (nonlinear identification and control). (Objective 2)
3. A braking/washout procedure can produce software generated inputs to the system in such a way that cues will not be attenuated unless physical constraints of the actuator dictates this attenuation. (Objective 3)
4. It has been demonstrated that the inclusion of the nonlinear hydraulic subsystem dynamics in the plant model produces a control system design with improved overall system performance, compared with a design which assumes the hydraulic subsystem dynamics are negligible. (Objectives 3 and 4).

CHAPTER II

METHOD AND PROCEDURES

In this chapter, an adaptive control algorithm is given, which can be used to solve nonlinear identification and control problems using digital computers. Section 2.1 presents the proposed control method, Section 2.2 presents the identification procedure and Section 2.3 describes the approach chosen for the calculation of inputs to the washout motion.

The direct identification and the direct control methods were chosen in lieu of optimal control and estimation approaches for the reasons discussed in Section 1.3.2.3. Also, Kaufman and Teavassos (20), showed that for a sample system, one iteration of the indirect (optimal) control algorithm required significantly more time to execute compared with the direct gain optimization procedure. This observation was also valid for the indirect and direct estimation algorithms.

2.1 Nonlinear System Control Algorithm

This section discusses the development of a numerical method for synthesizing the controller for a nonlinear dynamical system. The approach is a suboptimal, direct

search for the closed-loop gains that will minimize an error function criteria.

2.1.1 Problem Statement

The physical process to be controlled is assumed to be an n-th order continuous-time system:

$$\begin{aligned}
\dot{x}_1 &= f_1[x_1(t),x_2(t), \ldots x_n(t),t] \\
\dot{x}_2 &= f_2[x_1(t),x_2(t), \ldots x_n(t),t] \\
&\vdots \\
\dot{x}_n &= f_n[x_1(t),x_2(t), \ldots x_n(t),t]
\end{aligned} \qquad (2.1.1\text{-}1)$$

which is often expressed in vector form

$$\dot{x}(t) = f[x(t),t] \qquad (2.1.1\text{-}2)$$

where $x(t) \in R^n$ is the state vector of the system.

The controller to be constrained is:

$$U(t) = K(t)[r(t) - x(t)] \qquad (2.1.1\text{-}3)$$

where,

$U(t) \in R^m$ is the control vector, $m<n$

$K(t)$ is an $m \times n$ gain matrix

$r(t) \in R^n$ is the input vector signal

The problem is to calculate the values of the gain matrix elements k_{ij} that will minimize the quadratic error function:

$$J = \int_{t_o}^{t_o+\Delta t} [x(t) - r(t)]^T [Q][x(t) - r(t)]dt \qquad (2.1.1\text{-}4a)$$

Or, for finite steps L of integration,

$$J = \sum_{j=1}^{L} [x(j) - r(j)]^T [Q][x(j) - r(j)] \qquad (2.1.1\text{-}4b)$$

Subject to the dynamic constraints,

$$\dot{x} = f[x(t),U(t),\hat{\phi},t] \qquad (2.1.1\text{-}5)$$
$$x(t_o) = z(t_o)$$

z(t) is the plant state vector (measurement data), and $\hat{\phi}$ is the parameter vector of the nonlinear system.

2.1.2 Control Algorithm

The solution of the above problem requires the definition of the input to the system, r(t), and the weighting matrix [Q] for the quadratic error function defined in the equation (2.1.1-4).

Since $x(t_o)$ and $r(t_o)$ are known, but r(t) is, in general time dependent, there is a need to extrapolate r(t) using past information. The approach used is as follows:

Extract from storage the last 4 reference input values of the previous cycle (see also Figure 5). Calculate the first, second and third order time derivatives by Newton's backward difference formula:

$$\begin{aligned} r'(i) &= (3r_i - 4r_{i-1} + r_{i-2})/2\Delta h \\ r''(i) &= (r_i - 2r_{i-1} + r_{i-2})/\Delta h^2 \\ r'''(i) &= (r_i - 3r_{i-1} + 3r_{i-2} - r_{i-3})/\Delta h^3 \end{aligned} \qquad (2.1.2\text{-}1)$$

Using these values, the input to the system can be predicted by using a Taylor Series Expansion:

$$r(t) = r(t_i+h) = r(t_i)+hr'(t_i)+(\frac{h^2}{2})r''(t_i) + (\frac{h^3}{6})r'''(t_i) \qquad (2.1.2\text{-}2a)$$

or

$$r(i+k) = r(i)+k\Delta hr'(i)+ \frac{(K\Delta h)^2}{2} r''(i) + \frac{(K\Delta h)^3}{6} r'''(i) \qquad (2.1.2\text{-}2b)$$

Using this procedure, the integration should be performed over a fraction of the update interval.

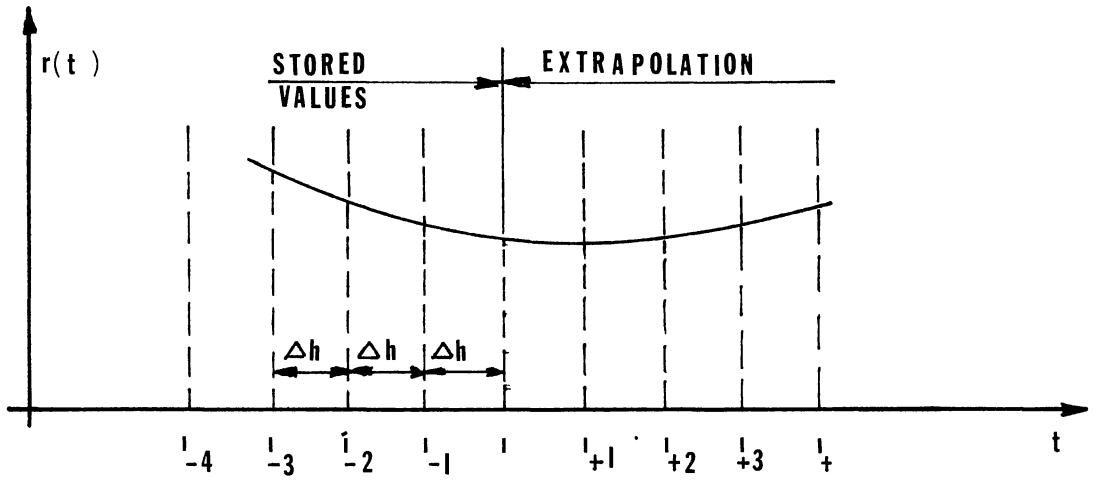

Figure 5. Reference input extrapolation for Control Law Synthesis.

Differentiating the quadratic error function defined in (2.11-4)

$$\frac{dJ}{dt} = [x(t) - r(t)]^T[Q][x(t) - r(t)]$$

and rearranging for a diagonal matrix Q

$$\frac{dJ}{dt} = \sum_{i=1}^{n} Q_i(x_i - r_i)^2 \qquad (2.1.2\text{-}3)$$

If the following values are set

$$Q_i = 1.0 \quad (i = k)$$
$$Q_i = 0.0 \quad (i = k), \qquad (2.1.2\text{-}4)$$

then a specific mode of control is obtained. If the subscript k corresponds to the velocity (x(k) = velocity), then the system will follow the velocity input only. If the subscript k corresponds to the position input, a position servo will be obtained. This mode selection for control is very flexible because it does not require a different algorithm for each mode of operation selected.

2.1.3 Implementation Steps

The control synthesis procedure can be summarized in the following steps.

Step (0). Off-line Initialization

Linearize the open-loop system equations of motion with nominal system parameters around a steady-state equilibrium point. Calculate $K_{last} = K(0)$ for the linearized deterministic system using any of the known methods so that the forward integration of Equation 2.1.1-5 is stable.

Step (1). On-line

Apply the input to the system. Record the input signal for the last four grid points. Calculate the first, second and third time derivatives for the input signal at the last point of the measured data.

Step (2). On-line

Given K_{last} and the initial state x(i) as well as r(i+j) from extrapolation, use a nonlinear minimization procedure to update K so that the cost function J in Equation 2.1.1-4b will be minimized. x(i+j), j=1,L can be calculated by the integration of Equation 2.1.1-5 forward in time over the time interval (t_i, t_{i+L}) from the initial state x(i).

Step (3). On-line

If $|J^{n+1} - J^n| < \epsilon$, where n denotes the number of optimization iterations performed, or if the allowed number of optimization iterations was reached, accept the new value of K and update the gain coefficient as follows:

$$K_{new} = K_{last} + \alpha(K - K_{last}) \qquad (2.1.3\text{-}1)$$

$$0.0 < \alpha < 1.0$$

Then return to step (1).

2.2 Nonlinear System Parameter Identification

This section discusses the development of a numerical method for parameter identification of nonlinear dynamical system. The approach taken was a direct search for the parameters by minimization of a specific cost function. This specific cost function is a measure of the deviation of the

response between the plant states (measurements) and the assumed model (unknown parameters) states.

2.2.1 Problem Statement

The nonlinear dynamical system model and the measured variables may be represented by the state-vector equations:

$$\dot{x} = f[x(t),U(t),\phi(t),t] \qquad (2.2.1\text{-}1)$$

$$z(t) = h[x(t),t] + w(t),$$

where $x(t) \in R^n$ is the state vector.

$z(t) \in R^L$ is the measurement vector.

$U(t) \in R^m$ is the control vector.

$\phi(t) \in R^k$ is the parameter vector.

$w(t) \in R^L$ is a zero-mean white noise sequence.

Assuming that all the states are accessible for measurement, the problem reduces to searching for the values of the parameters that will minimize the error function:

$$J = \int_{t_o-\Delta t}^{t_o} [z(t) - \hat{x}(t)]^T[Q][z(t) - \hat{x}(t)]dt \qquad (2.2.1\text{-}2)$$

subject to the dynamic constraints of the assumed plant model:

$$\dot{\hat{x}}(t) = f(\hat{x}(t),U(t),\hat{\phi}(t),t) \qquad (2.2.1\text{-}3)$$

where $\hat{x}(t_o - \Delta t) = Z(t_o - \Delta t)$, or, for finite steps L of integration,

$$J = \sum_{j=1}^{L} [Z(j) - \hat{x}(j)]^T [Q][Z(j) - \hat{x}(j)]dt \qquad (2.2.1\text{-}4)$$

Working directly with the nonlinear system equations of motion rather than trying to linearize the system may render faster system identification. Usually only a few parameters of the nonlinear system need identification. If the system is represented locally as $\dot{x} = [A]X + [B]U$, then the number of parameters to be identified will grow. If there are n states and m inputs, then the [A] matrix with n x n elements and the [B] matrix with n x m elements need identification. For example, if n = 4, m = 1 then 4 x 4 + 4 x 1 = 20 parameters need to be identified. Figure 6 depicts how the sampling and integration intervals are related. The identification process is applied only during a fraction of the interval between updates, thereby conserving function evaluation time.

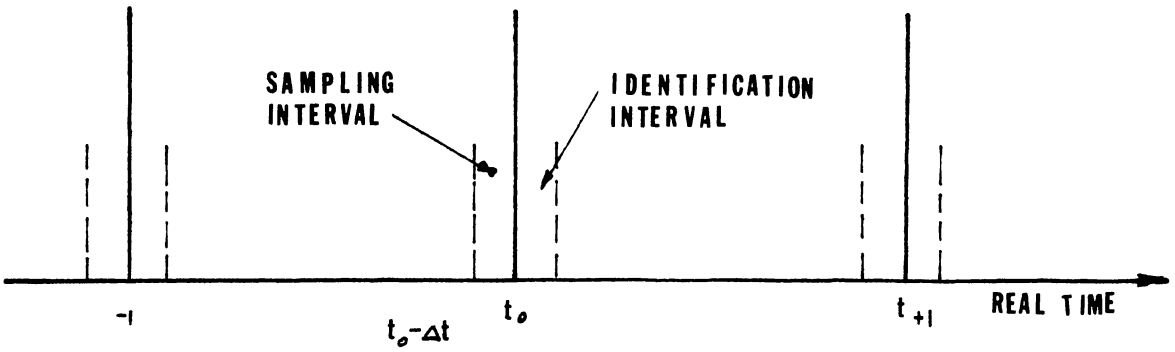

Figure 6. Integration and Sampling Interval for the Identification Process.

2.2.2 Implementation Steps

Step (0). Off-line Initialization

Develop the nonlinear model of the system:

$$\dot{\hat{x}}(t) = f[\hat{x}(t),\hat{U}(t),\hat{\phi}(t),t] \qquad (2.2.2\text{-}1)$$

where $\hat{U}(t) = K[r(t) - \hat{x}(t)]$

Calculate and initialize the nominal vector of system parameters $\hat{\phi}$.

Step (1). On-line

Apply the input to the system and record the input and measurements of the response (state values) during the measurement data window.

Step (2). On-line

Given $\hat{\phi}_{last}$ and the initial state $\hat{x}_i = z_i$, use the nonlinear minimization procedure to update $\hat{\phi}$ so that the cost function J in Equation 2.2.1-4 is minimized. The values of $\hat{x}(i)$ in the cost function are obtained by forward integration of Equation 2.2.1-3.

Step (3). On-line

If $|J^{n+1} - J^n| < \epsilon$, where n denotes the number of optimization iterations performed, or if the allowed number of optimization iterations is exceeded, accept the new values of $\hat{\phi}$ and update the system parameters as follows:

$$\hat{\phi}_{new} = \hat{\phi}_{last} + \alpha(\hat{\phi} - \hat{\phi}_{last}) \qquad (2.2.2\text{-}2)$$

$$0.0 < \alpha < 1.0$$

Then return to step (1).

2.3 Washout Algorithm

The braking/washout procedure described in this section is related to a special mode of operation for the motion platform and involves the generation of a particular input to the control system.

2.3.1 Problem Statement

All motion simulators have limits on the amount of movement in each degree of freedom. The design of a system

which will transmit motion cues to a pilot while keeping the movement of the simulator within its constraints is the major task faced by the designer. After the initial cue has been transmitted, another function of the system is to return the simulator to its neutral position without the pilot being aware of the movement. This effect is termed "washout". This technique of keeping the simulator near its neutral position, maximizes the movement allowable for subsequent cues.

2.3.2 Algorithm

The signal input to the system should be switched from the follower mode to the washout mode in the translational degree-of-freedom at a safe point, so that the physical limit of the actuator is not reached.

$$\tfrac{1}{2} M\dot{X}_L^2 = \int_{X_L}^{X_{max}} F dx = \int_{X_L}^{X_{max}} M\ddot{X}\, dx \,, \qquad (2.3.2\text{-}1)$$

where:

$\dot{X}_L$ is the velocity limit of the actuator at some position limit X_L,

X_L is the position corresponding to the velocity limit $\dot{X}_L$, and

X_{max} is the allowed maximum displacement of the actuator from neutral position.

If the maximum acceleration for washout is a constant α, then

$$\tfrac{1}{2}\,\dot{X}_L^{\,2} = \alpha(X_{max} - X_L) \quad , \qquad (2.3.2\text{-}2)$$

or

$$\dot{X}_L = \sqrt{2\alpha(X_{max} - X_L)} \qquad (2.3.2\text{-}3)$$

will define the switching point. Once the control mode is switched to washout, a command signal is generated to bring the motion base to the origin with acceleration level $\ddot{x} \leq \alpha$. When the base reaches the origin, the system is able to accept more inputs from the aircraft dynamics as produced by the real-time computations.

2.3.3 Implementation Steps (On-line)

Note: A "washout flag" is a switch that transfers the control system to work in washout mode when "True" and to follower mode when "False".

Step(1).

Evaluate state measurements of the system. If ABS(x)<ϵ, then clear the washout flag. If $\dot{x} \geq \dot{x}_L$, set the washout flag. Here x is the translational displacement, $\dot{x}$ is the translational velocity and $\dot{x}_L$ is the velocity limit ($\dot{x}_L = \sqrt{2\alpha(x_{max} - x)}$).

If the washout flag is set, go to step (2). Otherwise, branch to the normal follower mode.

Step (2).

In the washout mode there are two regions of operation:

a) If the maximum displacement allowed is not reached, the system is in the brake mode.

b) If the maximum displacement allowed is exceeded, the system is approaching the steady-state position.

Step (2a).

The reference (acceleration) input to the system is:

$$r = r + K(\dot{x}-\dot{x}_L), \quad |r| \leq \alpha \qquad (2.3.3\text{-}1)$$

Step (2b).

If $\dot{x} > \sqrt{2\alpha x}$ - the system moves too fast towards the origin, then

$$r = r + K(\dot{x} - \sqrt{2\alpha x}) \qquad (2.3.3\text{-}2)$$

Otherwise,

$$r = r + K(\alpha - r) \qquad (2.3.3\text{-}3)$$

Step 3.

An updated input r in the washout mode has been established; use it as a reference input to the system.

The Direct Identification method and the Direct Gain-Optimization method are known to offer the possibility of nearly-optimal control systems.

The contributions made in this chapter are related to the proposed algorithms. Integration intervals of the control algorithm (Section 2.1.2) and of the identification algorithm (Section 2.2.1) are related to a fraction of the update intervals. The updated values of gains and parameters are a function of previous results obtained during the iteration process and the new results obtained in the current iteration. The input to the system which is used for gain optimization, is extrapolated forward in time, using memorized data, rather than providing only a fixed value.

CHAPTER III

APPLICATION MODEL STUDIES

3.0 General Considerations

Before a detailed analysis, synthesis and simulation of the application system can be undertaken, it is important that the system configuration and desired mode of operation be understood. So that this research will produce qualitative conclusions concerning the advantages of the method, the physical configuration and its dynamical characteristics should be typical of those now in use. Similar consideration should also be given to the hydraulic servomechanism and the measured system variables used to control it.

This section is concerned with a description of the selected one degree-of-freedom motion platform model, its lumped-parameter representation and the desired performance under controlled conditions.

For this investigation,a translational degree-of-freedom motion platform is considered. A recommended operational envelope, (from Reference 1), is given in Figure 7. This envelope is typical for the "heave axis" performance of motion platforms currently in service.

The motion platform system should work in two modes of operation: a follower mode and a washout mode. In the

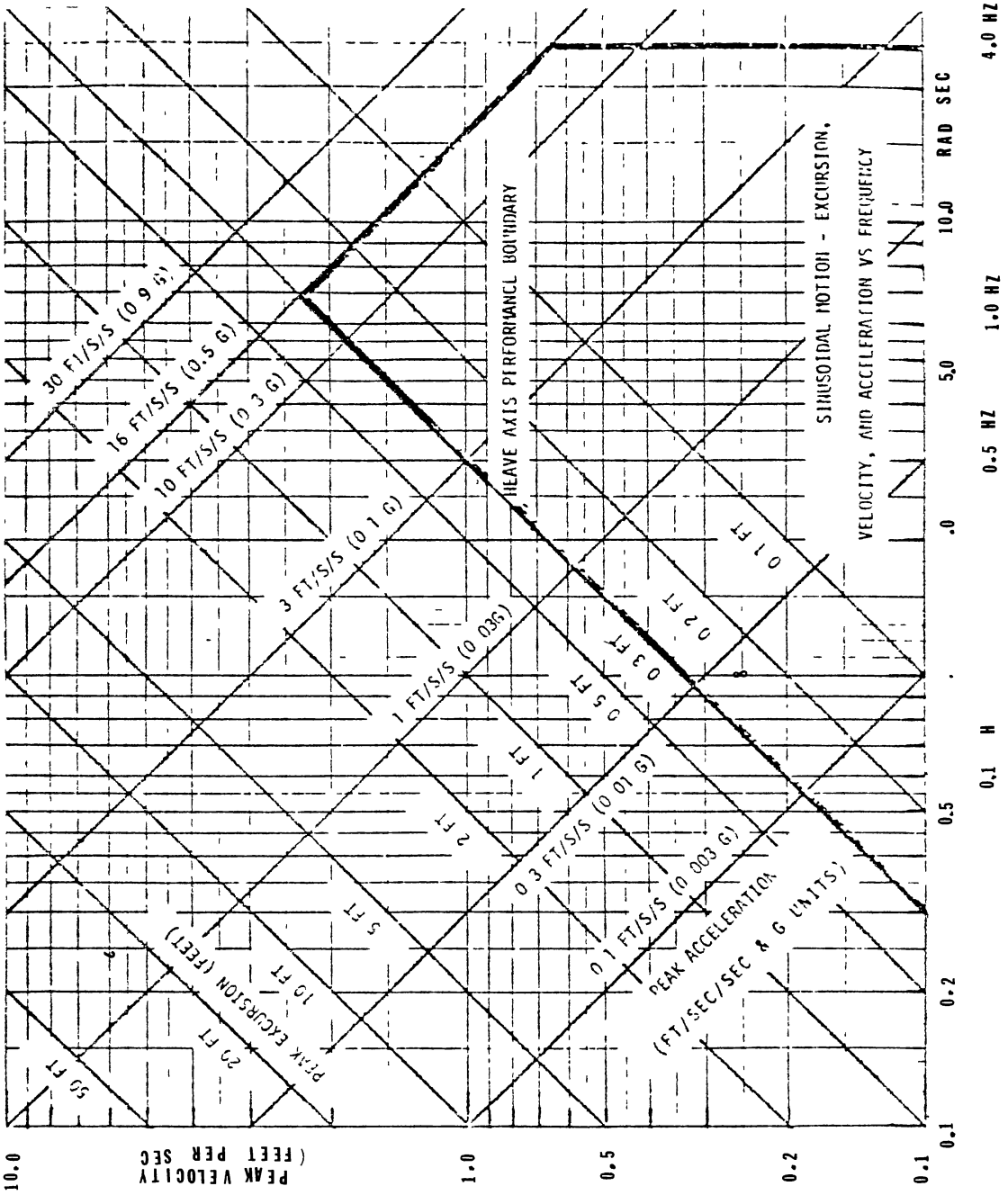

Figure 7. Vertical Axis Operational Envelope Recommended for Platform Motion Simulator. (From Ref. 1)

follower mode, the motion should follow the inputs from the aircraft dynamics as computed in real time by the computer. In case the platform measurements indicate that the physical limit of travel may be exceeded, the platform switches to the washout mode of operation. The latter mode is introduced to generate an input to the system independent of aircraft dynamics, in order to bring the platform to its neutral position.

A block diagram of the proposed system is depicted in Figure 8. Note that the aircraft dynamics model and the pilot model which appear on the block diagram are for descriptive and completeness purposes only. No attempt is made in this thesis to incorporate these models into the system under investigation. Furthermore, only one control variable is used, and it is assumed that complete state measurements are available.

3.1 Principles of Flow Control for Valve-operated Systems

Typically in hydraulic servo systems used in simulators, flow is controlled by throttling the fluid with the variable orifices of a control valve. If the flow rate demand is within its capacity, the pumping system normally acts as a source of substantially constant pressure.

The controlled flow rate is dependent on the pressure drops across the valve orifices, the density of the fluid,

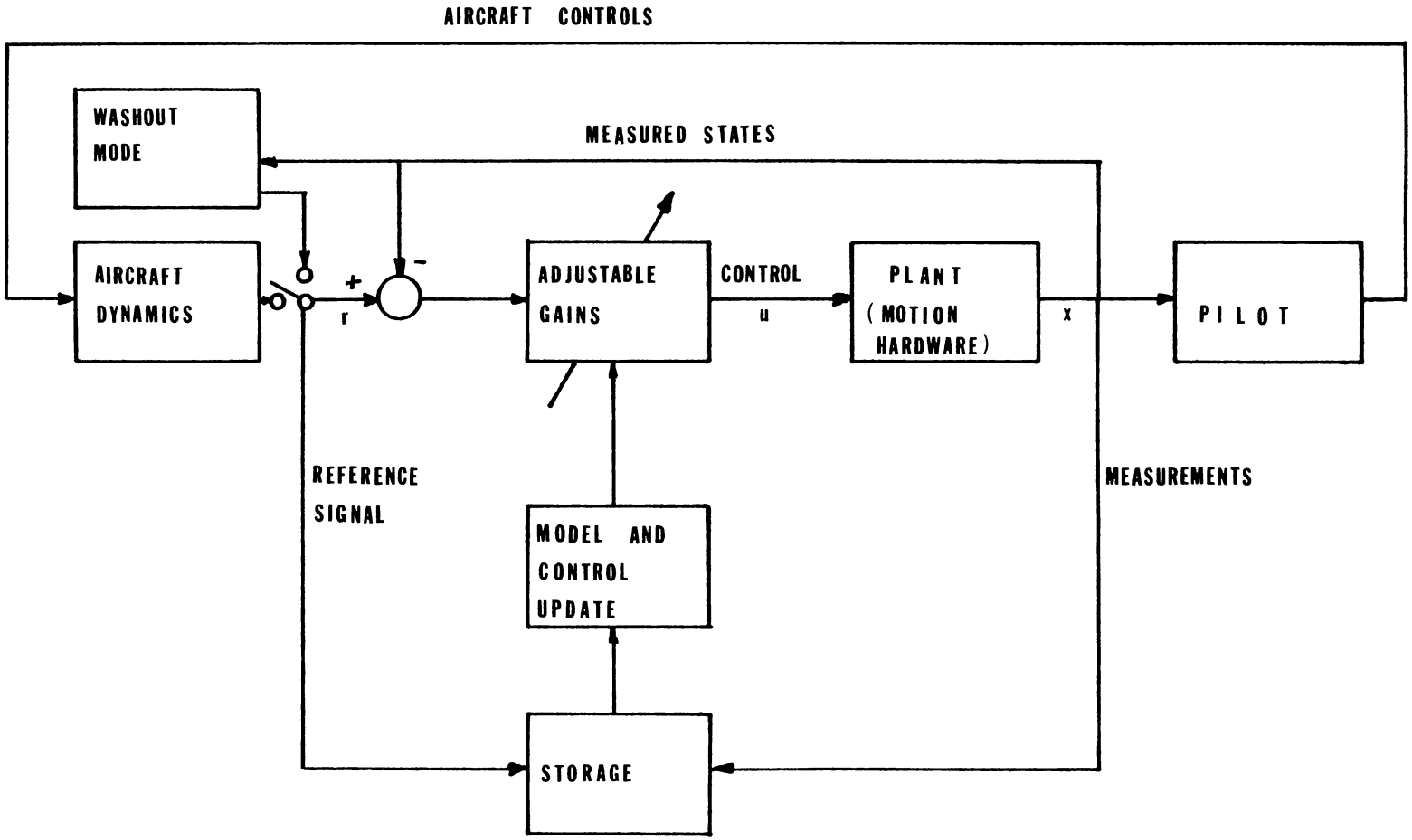

Figure 8. Proposed Control System for Motion Platform

the coefficients of discharge, and the orifice areas, and is relatively independent of the fluid temperature.

The general relationship between controlled flow rate, valve opening and pressure drop in a typical valve is:

$$q = C_d a \sqrt{\frac{2}{\rho}} \sqrt{\Delta P}, \qquad (3.1\text{-}1)$$

where q is the flow through a given orifice, a is the orifice area, ΔP is the pressure drop across the orifice, ρ is the mass density of the fluid and C_d is the coefficient of discharge.

The combination of four single orifices into the bridge arrangement shown on Figure 9, permits the modulation of fluid power to an actuator and load. The four variable orifices are coupled mechanically, so that:

$a_1 = a_4$ and $a_2 = a_3$

for a symmetrical valve,

$a_{1max} = a_{2max} = a_{3max} = a_{4max}$.

for a valve with negligible overlap or underlap (i.e. "line to line"),

if $a_1 > 0$, $a_1 = 0$ or if $a_3 > 0$, $a_1 = 0$.

Therefore, the flow path must be either through a_1, the Load and a_4, or through a_3, the Load and a_2.

The volumes of fluid in the actuator chambers are V_1 and V_2. The volumetric flow rate into chamber 1 is Q_1 the flow rate out of chamber 2 is Q_2, and the ram velocity is $\dot{x}$.

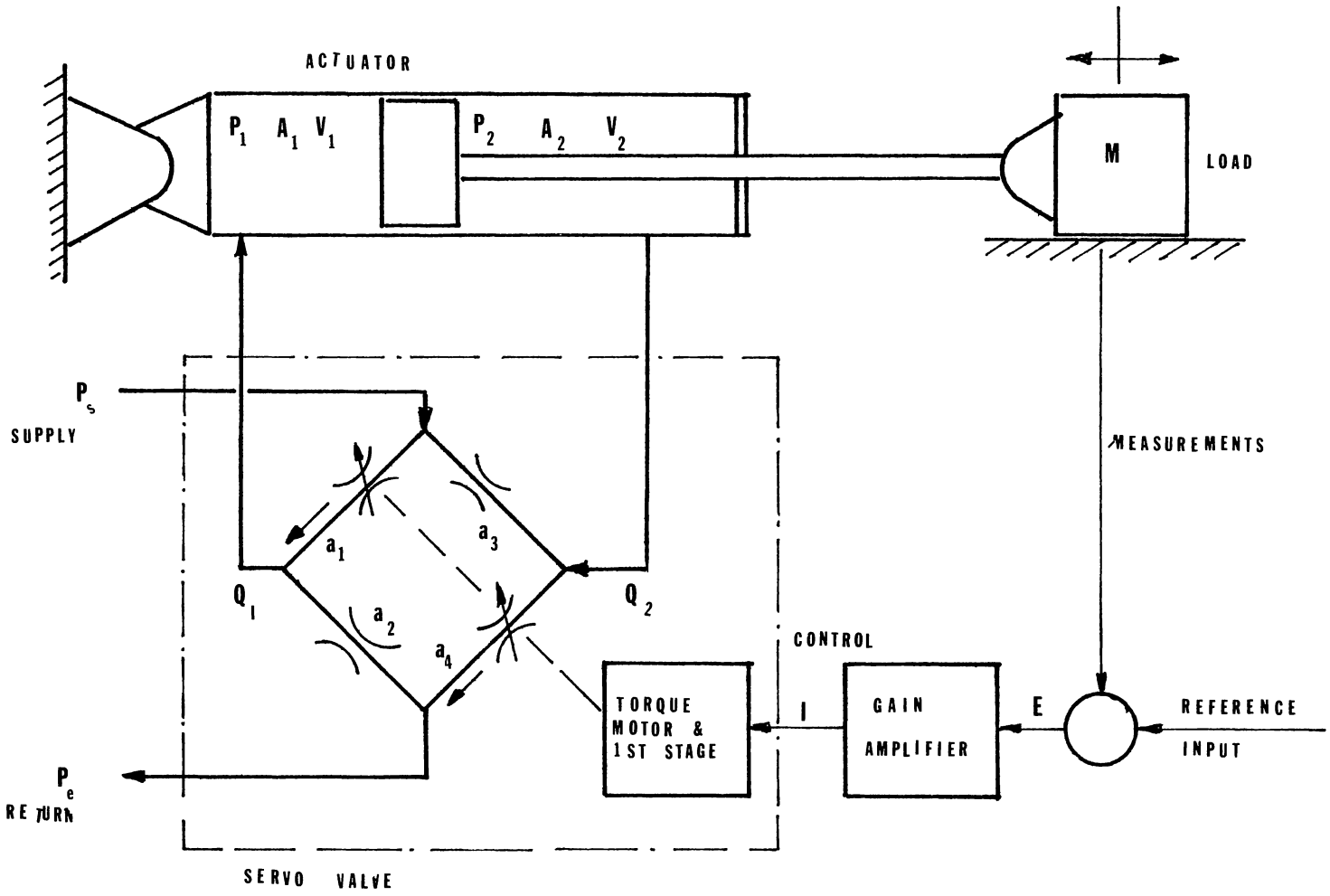

Figure 9. Typical Electrohydraulic Servosystem.

It is assumed that the pressure is uniform in each ram chamber and that no leakage occurs from chamber to chamber or past the piston rod.

The continuity equation applied to ram chamber 1 is:

$$\rho_1 Q_1 = \frac{d}{dt} \text{ (mass)} = \frac{d}{dt} (\rho_1 V_1)$$

$$\rho_1 Q_1 = \dot{\rho}_1 V_1 + \rho_1 \dot{V}_1 \quad . \tag{3.1-2}$$

Since the volume changes because of the ram motion, equation (3.1-2) may be written as:

$$Q_1 = \frac{\dot{\rho}_1 V_1}{\rho_1} + A_1 \dot{x} \tag{3.1-3}$$

From the definition of bulk modulus,

$$\frac{\dot{\rho}_1}{\rho_1} = \frac{\dot{P}_1}{\beta} \tag{3.1-4}$$

When equation (3.1-3) and equation (3.1-4) are combined

$$Q_1 = \dot{P}_1 \frac{V_1}{\beta} + A_1 \dot{X} \tag{3.1-5a}$$

Similarly, for ram chamber 2

$$Q_2 = -\dot{P}_2 \frac{V_2}{\beta} + A_2 \dot{X} \tag{3.1-5b}$$

3.2 Specific Hardware Configuration Chosen

A specific and typical hardware configuration was chosen to test the method. The actuator piston areas were chosen to be the same on both sides of the piston in order to simplify the model.

$$A = A_1 = A_2 = 8 \text{ in}^2.$$

Other parameters chosen were:

Actuator Displacement:	X_{max} =	±30.0 in.
Load Mass:	M =	8.635 Lbf sec/in^2
Supply Pressure:	P_s =	2000.0 lbf/in^2
Fluid Bulk Modulus:	=	150000.0 lbf/in^2
Servovalve:	Moog #78-14X; rated for 40 GPM at 1000 psi pressure drop, no load, 40 ma.	

3.2.1 Nonlinear Equations of Motion (Open Loop)

From the chosen valve specifications:

$$Q = 0.1217\ I\ \sqrt{P_v} \quad [\text{in}^3/\text{sec}] \qquad (3.2.1\text{-}1)$$

$$|I| \leq 40\text{ma} \quad \text{for saturation.}$$

where P_v is the pressure drop across the valve,

$$P_v = P_s - P_m$$

and P_m is the load pressure

$$P_m = P_1 - P_2$$

P_s is the supply pressure and I is the current input to the torque motor.

For a symmetrical valve and equal areas actuator:

$$P_s - P_1 = P_2 \qquad ; \text{ assumes } P_e = 0$$

$$P_v = 2(P_s - P_1) = 2P_2$$

$$Q_1 = 0.172\ I \sqrt{P_s - P_1}$$

$$Q_2 = 0.172\ I\ P_2 \qquad (3.2.1\text{-}2)$$

Taking into account the effects of negative current and back-pressure, equations (3.2.1-2) may be rewritten as:

$$Q_1 = 0.172I\left[\left|\frac{P_s}{2}+\left(\frac{P_s}{2} - P_1\right)\frac{I}{|I|}\right|\right]^{\frac{1}{2}}.\text{sign}\left[\frac{P_s}{2}+\left(\frac{P_s}{2} - P_1\right)\frac{I}{|I|}\right]$$

$$Q_2 = 0.172I\left[\left|\frac{P_s}{2}-\left(\frac{P_s}{2} - P_2\right)\frac{I}{|I|}\right|\right]^{\frac{1}{2}}.\text{sign}\left[\frac{P_s}{2}-\left(\frac{P_s}{2} - P_2\right)\frac{I}{|I|}\right] \qquad (3.2.1\text{-}3)$$

From equations (3.1-5), the flow equations for the actuator:

$$Q_1 = \dot{P}_1 \frac{V_1}{\beta} + A\dot{X}$$

$$Q_2 = -\dot{P}_2 \frac{V_2}{\beta} + A\dot{X} \qquad (3.2.1\text{-}4)$$

Neglecting the load friction (the friction force is less than 8 Lbf for moog Laminar actuators), the load equation is:

$$P_1 A - P_2 A = M\ddot{X} \qquad (3.2.1\text{-}5)$$

The nonlinear equations of motion are obtained by combining equations (3.2.1-4) and (3.2.1-5).

Two system variables can be defined as follows:

$$X_1 = X, \quad X_2 = \dot{X}_1$$

Then the equations of motion are:

$$\dot{X}_1 = X_2$$

$$\dot{X}_2 = \frac{1}{M}[P_1 A - P_2 A]$$

$$\dot{P}_1 = \frac{\beta}{V_1}[Q_1 - AX_2] \qquad (3.2.1\text{-}6)$$

$$\dot{P}_2 = \frac{\beta}{V_2}[-Q_2 + AX_2]$$

where Q_1 and Q_2 are defined in equation (3.2.1-3) and

$$V_1 = A(X_{max} + X)$$

$$V_2 = A(X_{max} - X)$$

3.2.2 Linearization of the equations of motion

The equations of motion of the model need to be linearized in order to find the required initial controller gains that will result in a stable system. The valve equations (3.2.1-2) are:

$$Q_1 = 0.172\ I\ \sqrt{P_s - P_1}$$

$$Q_2 = 0.172\ I\ \sqrt{P_2}$$

Then $$\Delta Q_1 = \frac{\partial Q_1}{\partial I}\Big|_i \Delta I + \frac{\partial Q_1}{\partial P_1}\Big|_i \Delta P_1$$

and $$\Delta Q_2 = \frac{\partial Q_2}{\partial I}\Big|_i \Delta I + \frac{\partial Q_2}{\partial P_2}\Big|_i \Delta P_2 \qquad (3.2.2\text{-}1)$$

or $$\Delta Q_1 = 0.172\ [\ \sqrt{(P_s - P_1)_i}\ \Delta I\ - \frac{I_i}{2\sqrt{(P_s - P_1)_i}}\ \Delta P_1]$$

and $$\Delta Q_2 = 0.172\ [\sqrt{(P_2)_i}\ \Delta I\ + \frac{I_i}{2\sqrt{(P_2)_i}}\ \Delta P_2]$$

Taking the specific cases where $I_i = 0.1 \times 40.0 = 4ma$ and $P_{2_i} = (P_s - P_1)_i = 0.05 \times 2000.0 = 100$ psi, the linearized valve equations are:

$$\begin{aligned} \Delta Q_1 &= 0.172\ (10\Delta I - 0.2\Delta P_1) \\ \Delta Q_2 &= 0.172\ (10\Delta I + 0.2\Delta P_2) \end{aligned} \qquad (3.2.2\text{-}2)$$

Using the given parameters and assuming $V_1 = V_2 = 240\ in^3$, the linearized equations of motion of the open-loop system are

$$\begin{aligned} \Delta\ddot{X} &= 0.926(\Delta P_1 - \Delta P_2) \\ \Delta\dot{P}_1 &= 625(\Delta Q_1 - 8\Delta\dot{X}) = 625(1.72\Delta I - 0.0345\Delta P_1 - 8\Delta\dot{X}) \\ \Delta\dot{P}_2 &= 625(-\Delta Q_2 + 8\Delta\dot{X}) = 625(-1.72\Delta I - 0.0345\Delta P_2 + 8\Delta\dot{X}) \end{aligned}$$

Substituting $X_1 = \Delta X$

$X_2 = \Delta\dot{X} = \dot{X}_1$

$X_3 = \Delta P_1$

$X_4 = \Delta P_2$ and $U = \Delta I$,

the resulting equations are

$$\begin{aligned} \dot{X}_1 &= X_2 \\ \dot{X}_2 &= 0.926X_3 - 0.926X_4 \\ \dot{X}_3 &= -5000X_2 - 21.5625X_3 + 1075U \\ \dot{X}_4 &= 5000X_2 - 21.5625X_4 - 1075U \end{aligned} \qquad (3.2.2\text{-}3)$$

In the standard state vector representation, the linearized equations of motion are of the form

$$\dot{X} = AX + BU \qquad (3.2.2\text{-}4)$$

In this example

$$A = \begin{bmatrix} 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.926 & -0.926 \\ 0.0 & -5000.0 & -21.5625 & 0.0 \\ 0.0 & 5000.0 & 0.0 & -21.5625 \end{bmatrix}$$

and

$$B = [0.0 \quad 0.0 \quad 1075.0 \quad -1075.0]$$

3.2.3 Calculation of Undamped Natural Frequency of Actuator Mass System

The selection of integration step sizes for on-line computation and system simulation is dependent to a large extent on the undamped natural frequency of the actuator-mass system. For the numerical values given above, the spring constant is

$$k_s = 2\beta A^2/V_0 = 2.150000.8^2/240 = 80000 \text{ Lb/in}$$

The undamped natural frequency of the system is

$$\omega_n = \sqrt{\frac{k_s}{m}} = 96.2 \text{ rad/sec}$$

or $f_n = 15.3$ Hz, and the period is

$$T = \frac{1}{f_n} = \frac{1}{15.3} = 0.065 \text{ sec.}$$

This result was used to select the integration step size for off-line simulation (step size chosen was 0.001 sec.).

3.2.4 Calculations of Position Control Gains for the Linearized Deterministic System

The excess pole specification method (14) and the computer programs that were provided by C. W. Merriam III, were used to calculate the control gains for the linearized deterministic system for this example. In order for the system to operate as a low pass filter up to 100 rad/sec (the undamped natural frequency of the open-loop system for the case considered is 96.2 rad/sec.), an appropriate closed-loop transfer function is

$$T_d(s) = \frac{1}{(1+0.01s)^3}$$

then

$$P(s) = T_d(s)^{-1} = (1+0.01s)^3 = 1.0 + 0.03s + 3.0x10^{-4}s^2 + 1.0x10^{-6}s^3,$$

so the P matrix is:

$$P_{(i,1)} = [1.0 \quad 0.03 \quad 3.0x10^{-4} \quad 1.0x10^{-6}].$$

Using the A and B matrices from equation (3.2.2-4) and selecting the H matrix (output: Y = HX) as

$$H = \lfloor 1.0 \quad 0.0 \quad 0.0 \quad 0.0 \rfloor,$$

which means position control, the following gain coefficients were obtained:

$$K = \lfloor 502.228 \quad 10.4174 \quad .1295058 \quad -.1295058 \rfloor$$

for the control law:

$$I = U = \lfloor K \rfloor \{r-X\} .$$

3.3 Simulation Results: Deterministic System

In this section, the simulated responses of linear and nonlinear deterministic systems are evaluated.

The purpose of these simulations was to determine whether a controller developed using the linearized model,

will produce a stable response of the nonlinear model under investigation. If the response is stable, the gains calculated for the linearized deterministic system are suitable to be initialized in the adaptive controller of the nonlinear system.

Both systems were given the same one inch step input and same position control law that was derived in the last section.

$$I = [K]\{ r(t) - X(t)\} , \qquad (3.3\text{-}1)$$

where $\{r\}$ - the input vector

$\{X\}$ - the state vector

and $[K] = [502.228 \quad 10.4174 \quad .1295058 \quad -.1295058]$

is the gain matrix.

Figure 10 shows the simulated response of the linearized model and Figure 11 shows the simulated response of the nonlinear model. The initial conditions were identical for the two simulations. The actuator position converged in both simulations to one inch (the same as the step input) and the control variable converged to zero. The other states recorded, P_1 and P_2, are far from being identical due to the differences between the linearized model and the nonlinear model, but a stable response is shown for both models.

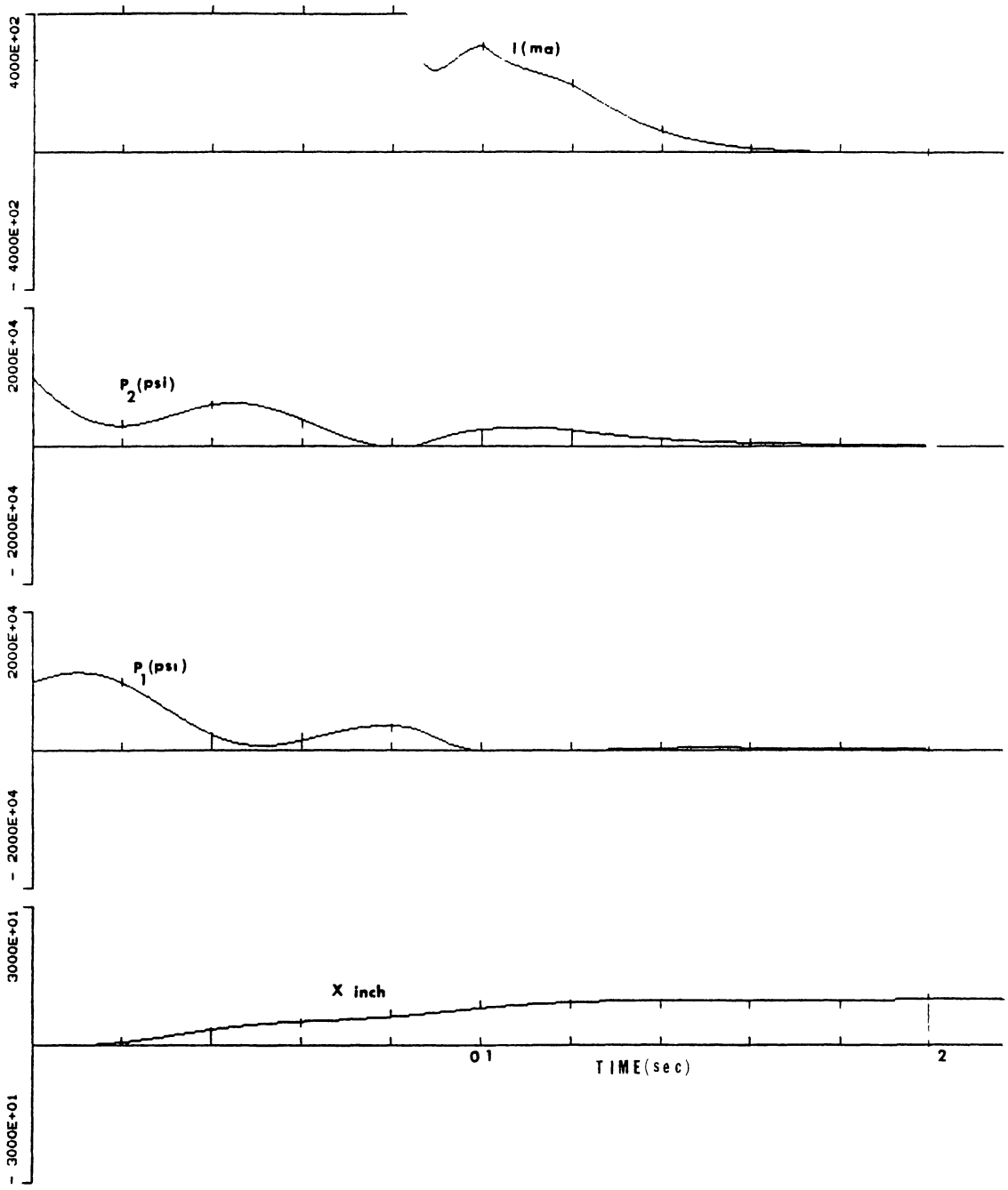

Figure 10. Simulated Response of Linearized model to 1" step input. (Deterministic System, constant gains)

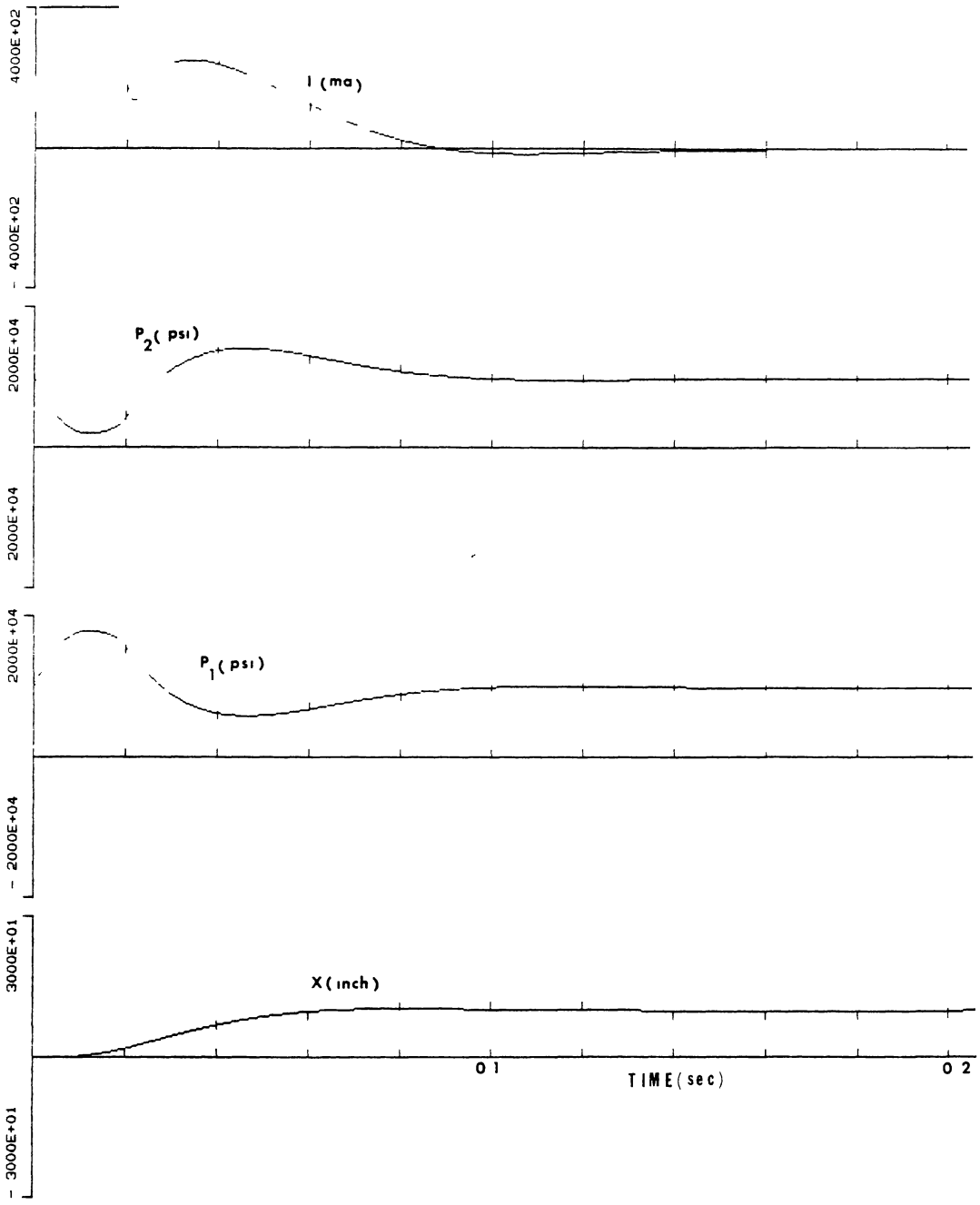

Figure 11. Simulate Response of Nonlinear model to 1" step input. (Deterministic System, constant gains)

3.4 Adaptive Control Law Simulation and Discussion

3.4.1 Simulation Program

A computer program was developed to simulate the identification, control synthesis and washout procedures discussed in Chapter II. The application model and proposed control system were described in section 3.0 of this chapter.

3.4.1.1 Program Description. The program includes a main calling program and eight subroutines. This program was written in the structured programming language Fortran 77. A flow chart description of the program is given in Figure 12.

3.4.1.2 Program NLCONT. This is the main program used to initialize parameters, to control timing and to store interim results. The program calls the integration routine (RK41) to simulate the "plant" response and calls the optimization routine (QNDAV1) twice: (1) for parameters identification and (2) for gain coefficients calculations.

3.4.1.3 Subroutine RK41. The purpose of the subroutine is to generate the input signal to the system, calculate the control variable and integrate the nonlinear system equations of motion. The input is generated for two modes: the follower mode and the washout mode. Integration is done

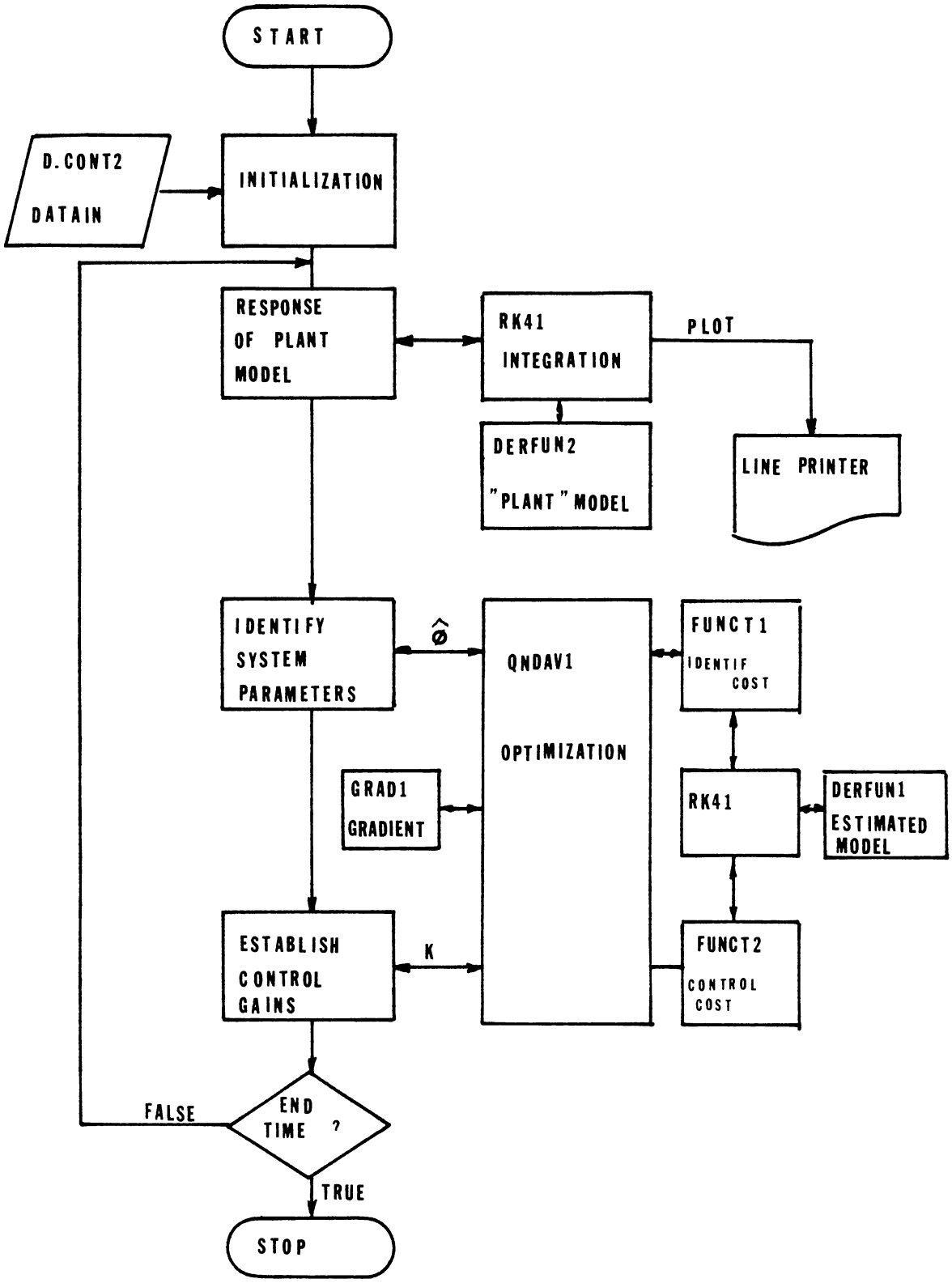

Figure 12. Simulation Program Flowchart.

using a fourth-order Runge-Kutta method. This subroutine is called by the main program (NLCONT) and by the cost function subroutines (FUNCT1, FUNCT2). The subroutines called by RK41 are the system equations of motion: DERFUN1 - the estimated model; DERFUN2 - the reference plant model.

3.4.1.4 Subroutine QNDAV1. The purpose of this subroutine is to calculate the minimum of a nonlinear function of N variables and to establish the values of the variables for the minimum point.

This program was written to implement the algorithm suggested by W.C.Davidon (25), which is basically a quasi-Newton method without line search. The subroutines called by QNDAV1 are: FUNCT1 (identification cost function), FUNCT2 (control cost function) and GRAD1 (gradient subroutine).

3.4.1.5 Subroutine GRAD1. The purpose of this subroutine is to calculate the gradient of a function of N variables. The subroutines called by GRAD1 are: FUNCT1 (identification cost function) and FUNCT2 (control cost function).

3.4.1.6 Subroutine FUNCT1. The purpose of this subroutine is to calculate the cost function in the parameter identification process. The subroutine called by FUNCT1 is the integration subroutine RK41.

3.4.1.7 Subroutine FUNCT2. The purpose of this subroutine is to calculate the cost function for the control synthesis process. The subroutine called by FUNCT2 is the integration subroutine RK41.

3.4.1.8 Subroutine DERFUN1. The purpose of this subroutine is to calculate the equations of motion of the estimated model.

3.4.1.9 Subroutine DERFUN2. The purpose of this subroutine is to calculate the equations of motion of the reference plant model.

3.4.1.10 Subroutine LXYPLT. The purpose of this subroutine is to generate plots on the Printronix line printer (at Burtek, Inc., Tulsa, OK.).

3.4.2 Simulation Results for the Adaptive Control System

In this section, the simulated responses of the adaptive control system are discussed and compared with the dynamic characteristics of an existing motion platform.

An acceleration input was chosen for the simulation, since it will transfer realistic cues to the pilot trainee. If a position input were selected, the trainee would get a cue which lags the acceleration of the aircraft by 180°.

Two parameters were selected for identification: Servo valve gain and bulk modulus of the hydraulic fluid. The "plant" parameters were fixed as follows:

Valve gain = 0.172 [$in^4/sec\text{-}ma\text{-}Lbf^{\frac{1}{2}}$]

Bulk modulus = 0.15 [$x10^6 Lbf/in^2$].

The values for the estimated model parameters were initialized for all runs as follows:

Valve gain = 0.180

Bulk modulus = 0.140

The nonlinear system that was described in section 3.2 was tested by the adaptive control simulation program of section 3.4.1 for three different time-dependent signal inputs:

r_1 = 22.0 sin3t [in/sec^2]
r_2 = 120.0 sin10t [in/sec^2]
r_3 = 80.0t [in/sec^2].

The system responses for the above inputs are given respectively in Figure 13, Figure 14 and Figure 15.

The low frequency input (Figure 13) and high frequency input (Figure 14) were selected from a typical simulator operating envelope chart (Figure 7).

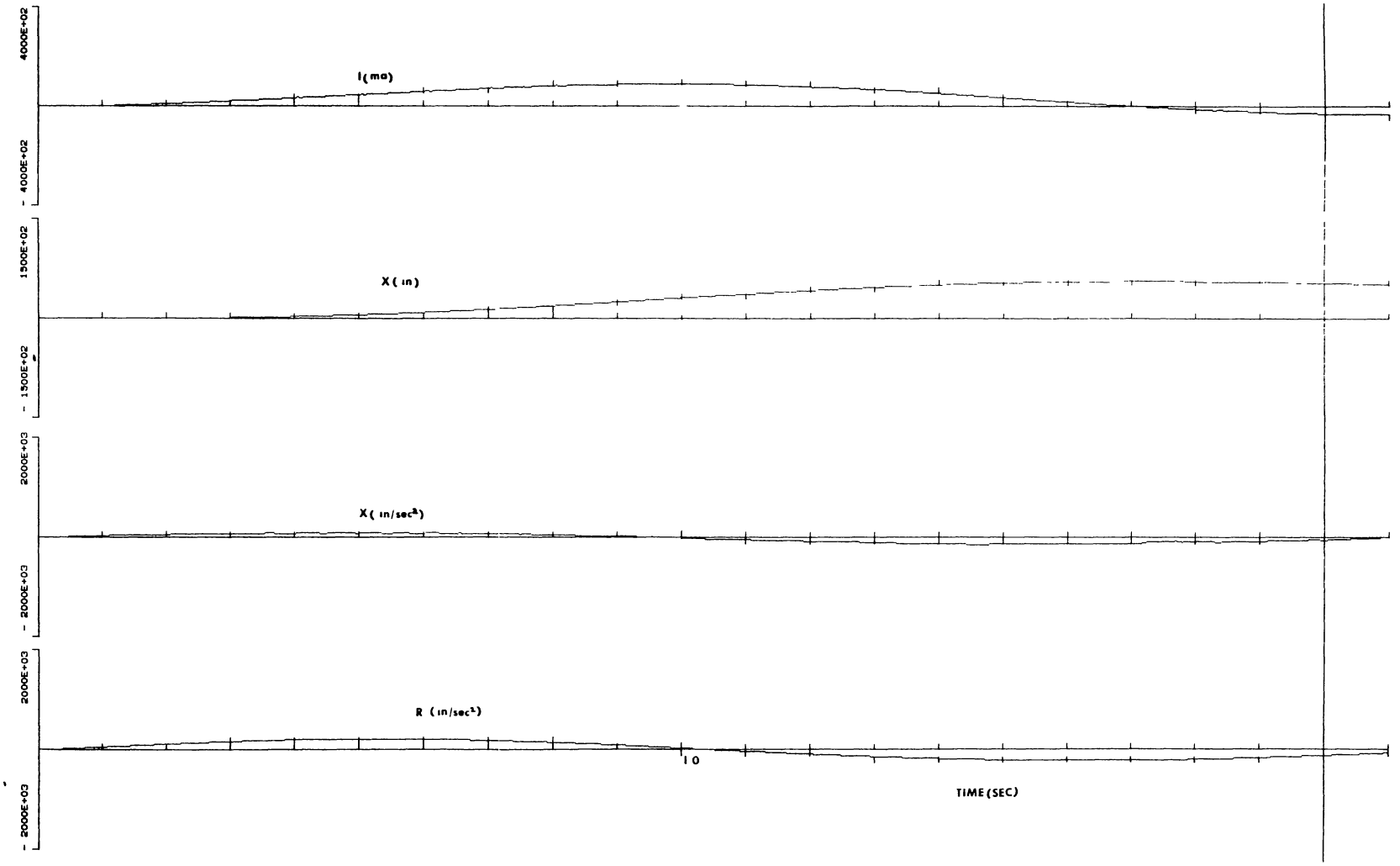

Figure 13. Low Frequency Input (Adaptive Control System).

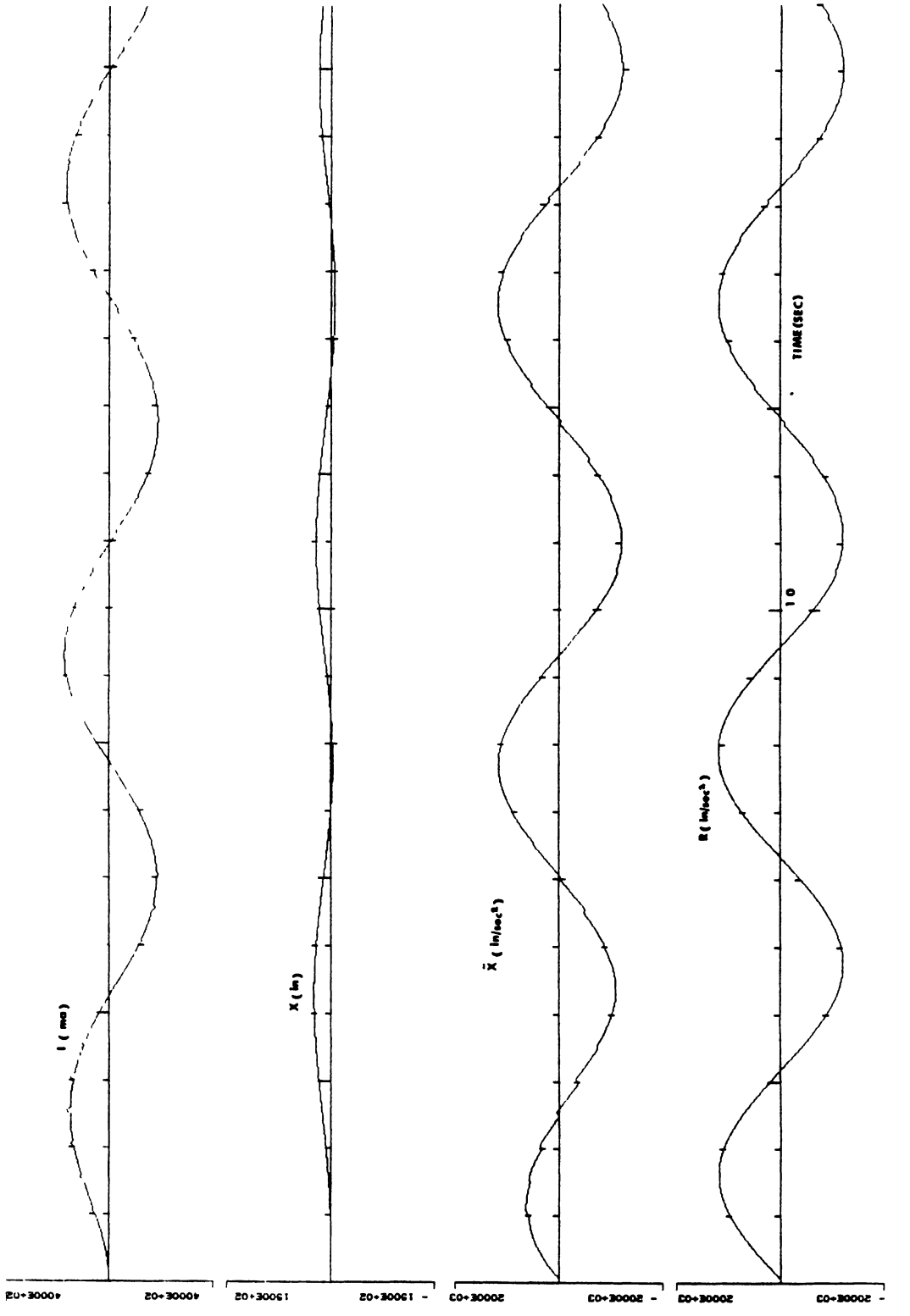

Figure 14. High Frequency Input (Adaptive Control System).

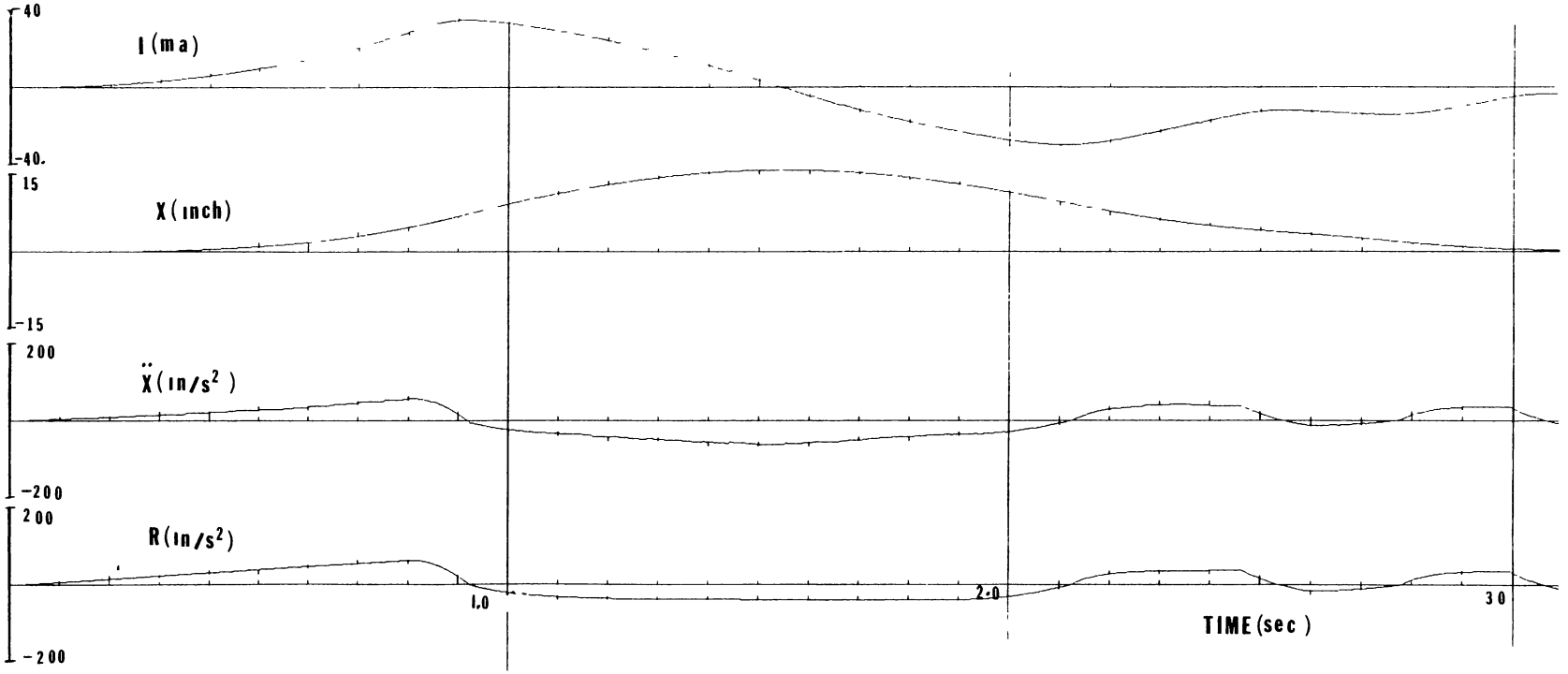

Figure 15. Ramp Input and Washout Process

The ramp input (Figure 15) was selected to test the washout algorithm. Figure 13 shows a good follow-up response of the system to a low-frequency input. But, because of the small amplitude of the input and the outputs, it is hard to make any decisive evaluation of the response.

The results in Figure 14 show that for a high frequency input, the phase lag and attenuation recorded became negligible after a short time interval. This short time interval (about one second) was needed to adapt the parameters of the model to the system parameters and to adjust the gains. The updates in the gains are noticed as minor discontinuities in the control I and in the response X during the initial time interval discussed above.

Figure 15 shows the response of the system to a ramp input. After about 0.8 sec it was calculated that the actuator limit may be exceeded, and the mode of operation was switched to the washout mode. This mode is intended to bring the motion platform to a neutral position (X=0), with an acceleration ($\ddot{X}$) limit of 0.1g. During the movement of the platform towards the neutral position, the input to the system is adjusted continuously, so that the platform will not overshoot the neutral position. Figure 16 shows the estimated values of the identified parameters as a function of time as the simulation progresses. The system input was 120 sin10t [in/sec^2] as shown in Figure 14. The parameters estimated were: valve gain (nominal value = 0.172) and bulk

modulus of the fluid (nominal value = 0.150×10^6).

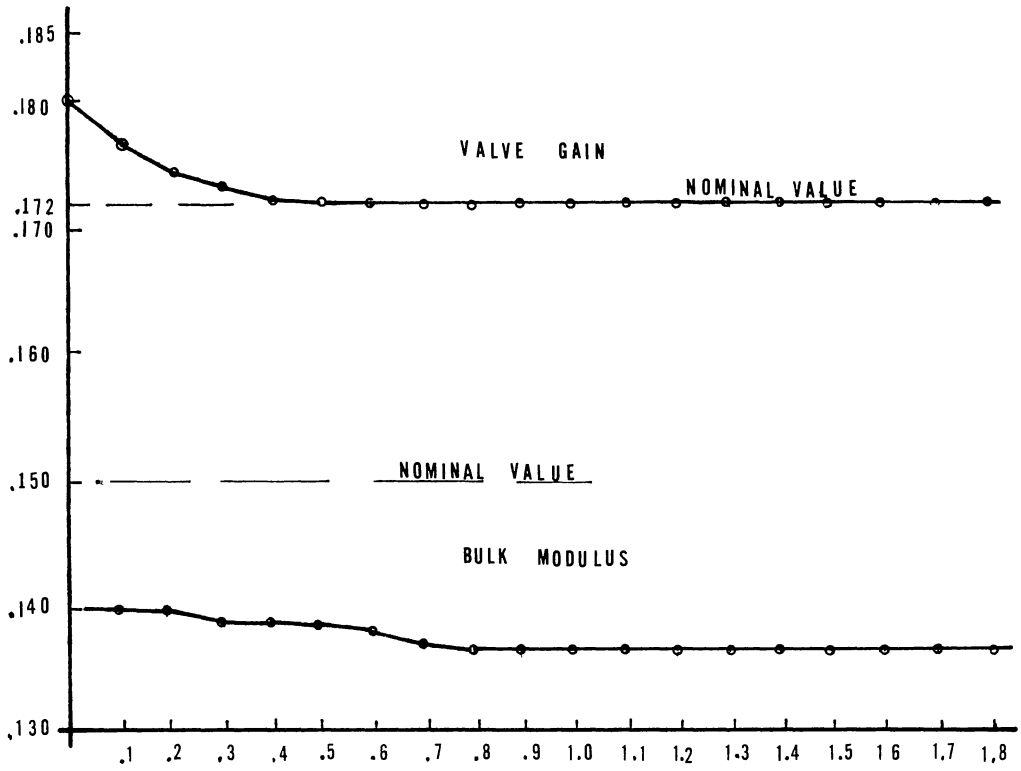

Figure 16. Parameter Estimation During Run Time
(Input r=120 sin10t [in/sec^2])
$Q = [1.0,\ 1.0\times10^{-2},\ 1.0\times10^{-4},\ 1.0\times10^{-4}]$

The valve gain converged to the nominal value at about 0.5 sec. The estimated bulk modulus of the fluid did not converge to the nominal value, but it reached a constant value after about 0.7 seconds. This result indicates that the system performance is not very sensitive to the value of

the bulk modulus except near resonance. (The estimated bulk modulus of the fluid converged to the nominal value in one of the simulation runs, when instability was introduced into the system).

Figure 17 shows the adjustment of the gains as they occured during the computation for the high frequency input (120 sin10t).

These adjustments occur automatically during the initial transient and settle to a nearly constant value after 1.5 seconds. The mode selected was acceleration (pressure) control and the initial gains selected were arbitrary. If the initial gains were selected by calculating the nominal gains from the linearized model, the convergence of the computed gains would be accelerated.

The gains displayed in Figure 17 were calculated for a particular input. Thus, different inputs lead to different gains.

Since identification and control synthesis depend upon the ability to minimize a defined cost function, a quantitative evaluation of the method can be made by investigating the values of the cost function during the iteration process. Table I gives the identification cost functions calculated for each iteration. The identification cost function is defined in section 2.2. The control cost function as defined in section 2.1 is given for each iteration in Table II.

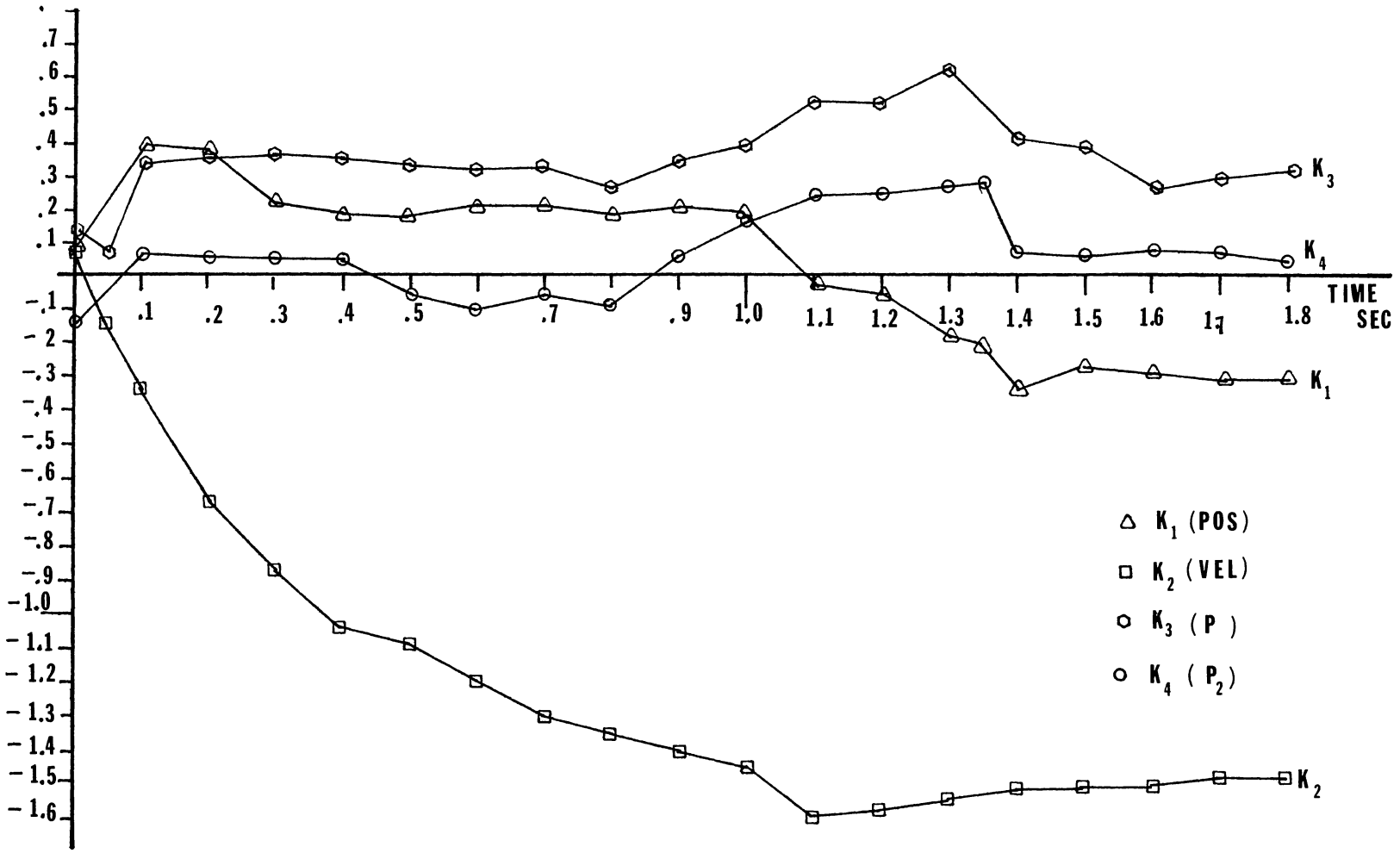

Figure 17. Gain Adjustment During Run Time
(Input. 120 sin10t)

TABLE I

RESULTANT ERROR FUNCTION OF IDENTIFICATION ALGORITHM.
(at 0.05 sec. update)

Iteration No.	Function Value
1	$.89 \times 10^{-4}$
2	$.11 \times 10^{-3}$
3	$.979 \times 10^{-7}$
4	$.349 \times 10^{-7}$
5	$.376 \times 10^{-7}$
6	$.357 \times 10^{-7}$

Both tables are given for simulation results shown in Figure 14. Both accepted points and rejected iteration points were recorded in these tables. Table III is a recording of the number of function evaluations and gradient calculations performed during the system simulation. These values are used to predict computation time needed for real time execution (section 4.2).

The results of the above adaptive control law simulation can be used to qualitatively evaluate the method against a traditional design of similar systems.

TABLE II

RESULTANT ERROR FUNCTION OF CONTROL ALGORITHM
(at 0.05 sec. update)

Iteration No.	Function Value
1	$.131 \times 10^{4}$
2	$.127 \times 10^{5}$
3	$.305 \times 10^{3}$
4	$.309 \times 10^{7}$
5	$.265 \times 10^{4}$
6	$.668 \times 10^{2}$
7	$.523 \times 10^{3}$
8	$.849 \times 10^{1}$
9	$.278 \times 10^{1}$
10	$.469 \times 10^{0}$
11	$.217 \times 10^{-1}$
12	$.129 \times 10^{-1}$
13	$.109 \times 10^{-1}$

Figure 18 and 19 show experimental results for small amplitude sinusoidal position inputs to an existing motion platform actuator, employing constant feedback gains.

For an input frequency of 10 rad/sec, the phase lag of the system with position, velocity and differential pressure feedback (Figure 18) is above 55°, and the attenuation is about -1.5 dB. For the system with only position and dif-

TABLE III

NUMBER OF FUNCTION AND GRADIENT EVALUATIONS DURING SIMULATION (6 function calls allowed for identification and 13 for control optimization).

Update Time (sec)	Identification		Control	
	Function	Gradient	Function	Gradient
0.05	6	3	13	9
0.10	6	3	13	5
0.15	6	2	13	4
0.20	6	3	13	4
0.25	6	3	13	5
0.30	6	3	13	5
0.35	6	3	13	5
0.40	6	3	13	3
0.45	6	3	13	3
0.50	6	3	13	4
0.55	6	4	13	5
0.60	6	4	13	3
0.65	6	5	13	4
0.70	6	5	13	3
0.75	6	2	13	5
0.80	6	4	13	3
0.85	6	3	13	4
0.90	6	4	13	3
0.95	6	1	13	2
1.00	6	1	13	4

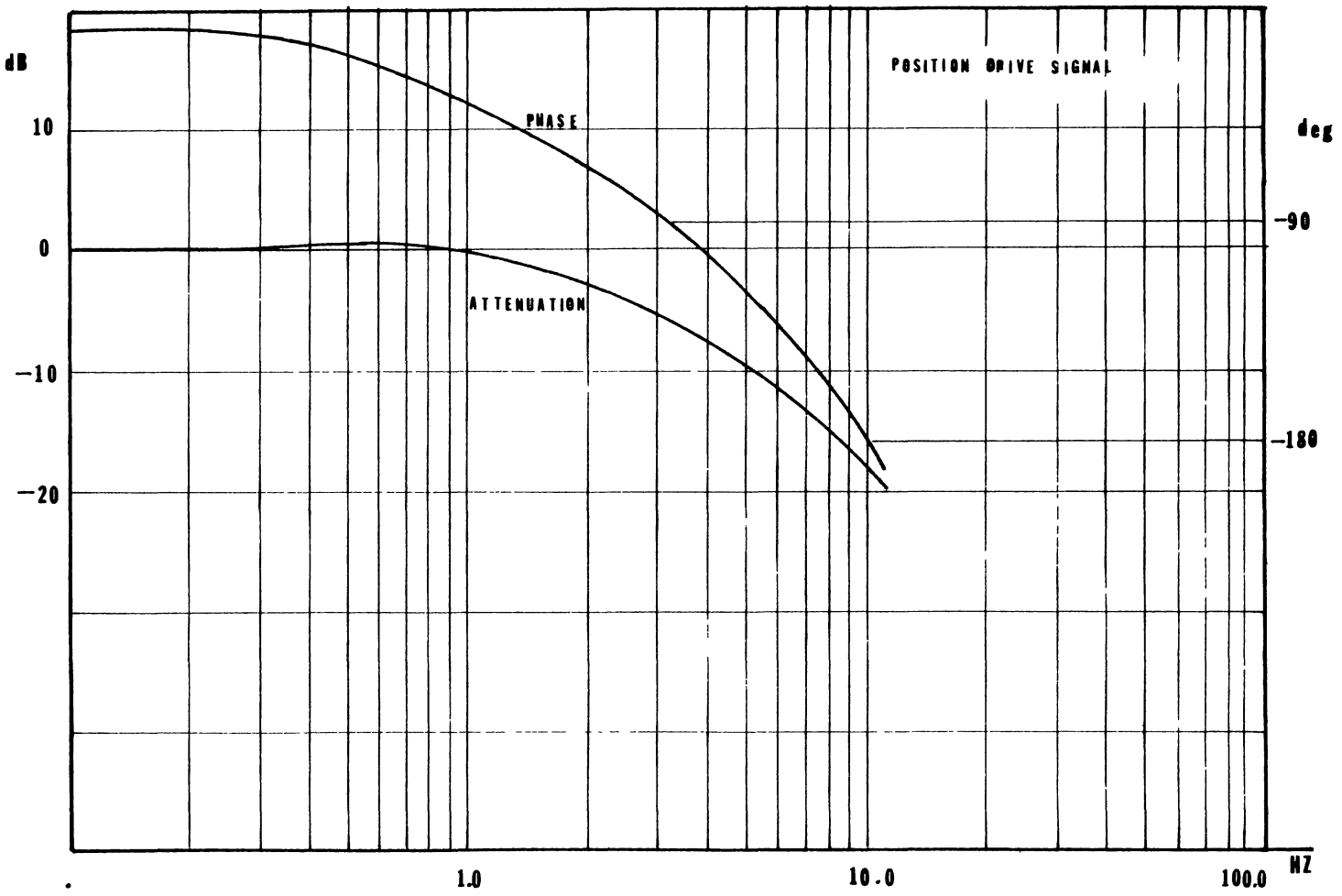

Figure 18. Frequency Response in Heave for Existing System (With Pressure and Velocity Feedback)

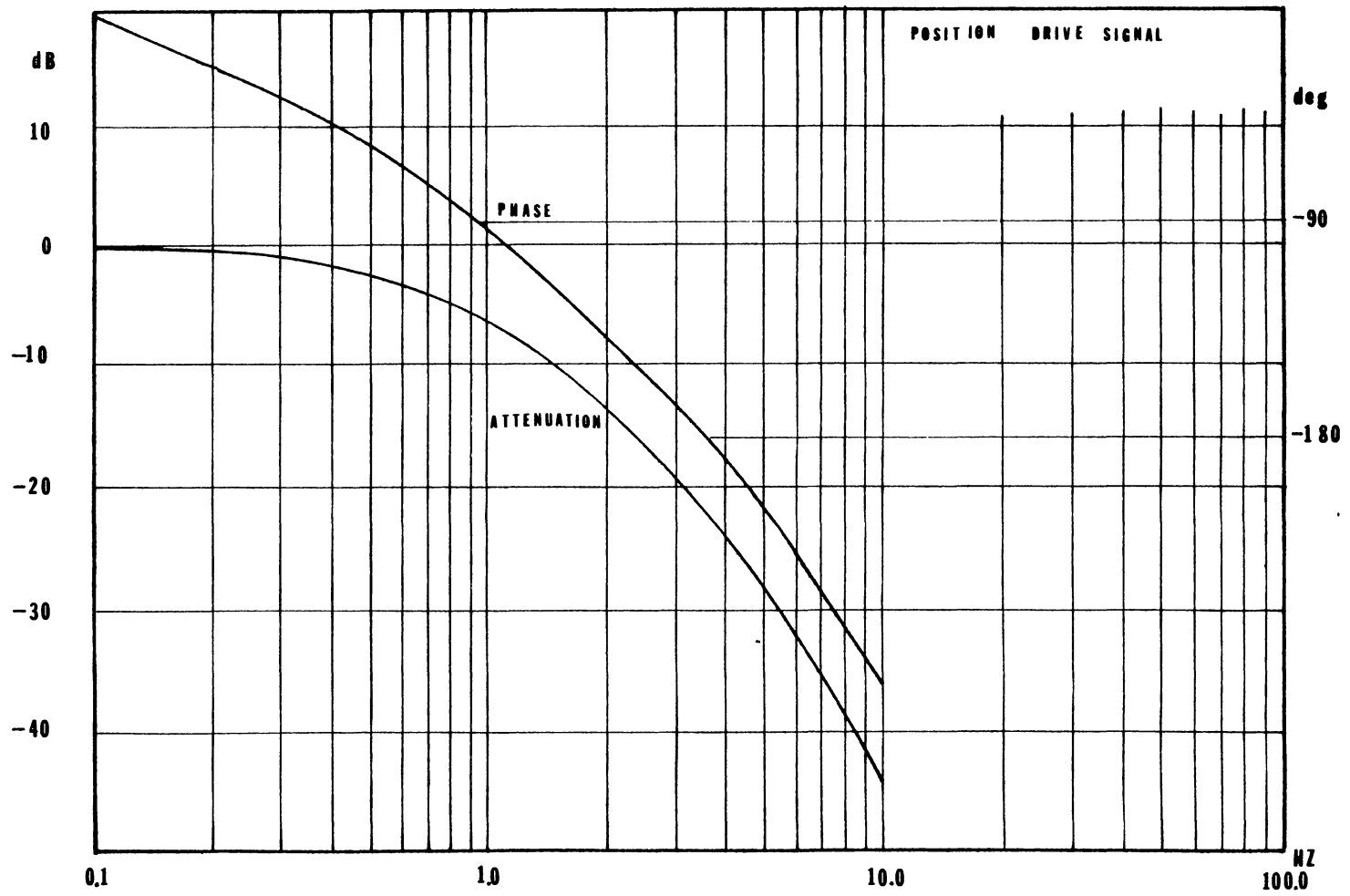

Figure 19. Frequency Response in Heave for Existing System (Pressure Feedback, no Velocity Feedback).

ferential pressure feedback (Figure 19), the phase lag (the same input) is 128° and the attenuation is -5.0 dB. The experimental results will give other characteristics for larger amplitude inputs since the system is highly nonlinear.

The experimental frequency response curves of Figure 18 and 19 show degradation in performance for high frequency inputs. By comparison, the results of the adaptive control law simulation of Figure 14 (time domain response) shows no degradation of the system for 10 rad/sec frequency input; both attenuation and phase lag are negligible.

In Figure 20, the conventional method of adjusting feedback gain values is depicted. The system is provided with a position step input and the actuator differential pressure output is recorded as a function of time. The pressure feedback gain (which is dependent on a resistance in the gain amplifier) is adjusted until the differential pressure recorded shows fast decay with an acceptable level of overshoot and no oscillations. This is a lengthy way to get the correct gain values. Whenever more than one feedback gain needs to be adjusted, the number of tests needed in order to adjust the system gains increase.
These gains will be satisfactory only for a limited envelope and need to be rechecked and adjusted periodically because of drift. On the other hand, using the method devised in this thesis, the system gains are adjusted continuously by

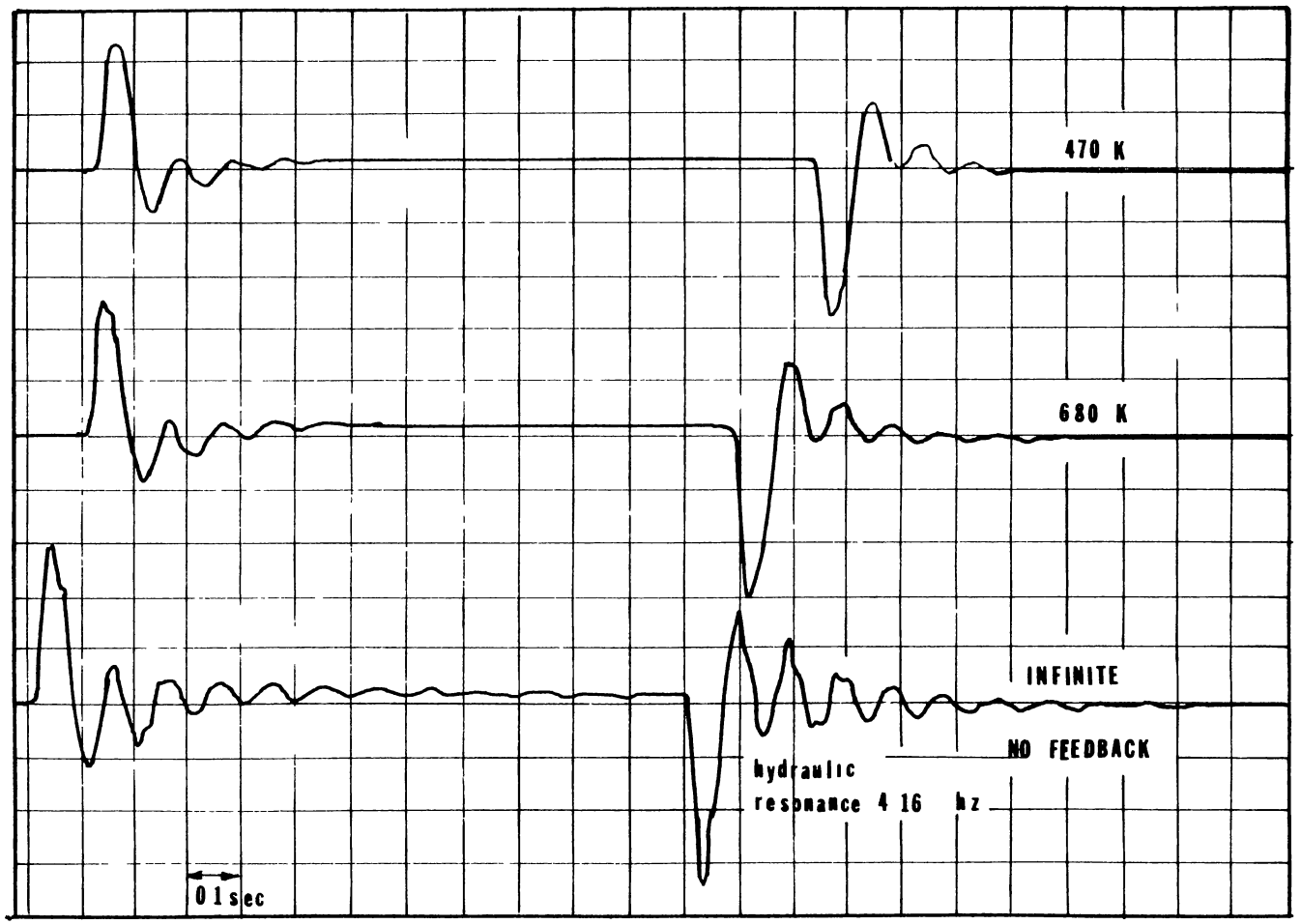

Figure 20. Measured Differential Pressure vs.Time for Different Pressure Feedback Gains.

the algorithm, thereby taking care of the nonlinearities and the drifts in system parameters.

CHAPTER IV

IMPLEMENTATION CONSIDERATIONS

4.0 GENERAL

Chapter III was concerned with the application of the proposed method to on-line control of a one degree-of-freedom motion platform. The control law was formulated, and the model was derived and coded in Fortran Language. Some tests were conducted and the system response was simulated using off-line aids like a nominal system model and plots. By using the trajectories obtained from the simulation runs and conparisons with existing control methods, controller performance was evaluated. The following sections are devoted to real-time implementation feasibility studies, especially the memory requirements, run-time prediction and expected noise.

4.1 Memory Requirements

Table IV shows the memory requirement for off-line program execution. These object code requirements include both program and data storage. The total memory requirement for off-line simulation of programs and data (FORT77 Compiler) on the SEL computer is 48K bytes.

Davidon (25) reported that the nonlinear optimization routine as programmed on a Wang 720 mini-computer, used less than 672 bytes of memory for program storage. Since the object code created by the SEL FORTRAN-77 Conpiler was 9072 bytes, then even if the data memory (20 parameter capability, 5880 bytes) requirement is subtracted, there is still a need for 3192 bytes of program memory. It is evident that for real time execution, the code has to be changed, optimized and written in assembler language.

TABLE IV

MEMORY REQUIREMENTS FOR OFF LINE PROGRAM EXECUTION

Program	Function	Object Code (Bytes)
NLCONT	Main	1592
LXYPLT	Plot graphs	3576
QNDAV1	Nonlinear function minimization	9072*
RK41	Integration	1048
FUNCT1	Identification cost function	176
FUNCT2	Control cost function	176
GRAD1	Numerical gradient	272
DERFUN1	'Plant' model	536
DERFUN2	'Update' model	536

* QNDAV1 can handle up to 20 parameters

Using Davidon's data for the storage requirement for the minimization routine, it can be expected that for real-time execution, the storage requirement for the application discussed in this thesis is about 5K bytes of memory. This storage requirement is very modest with respect to the capacity of most modern mini or micro computers.

4.2 On-Line Execution Time

The on-line execution time for the algorithm can be determined by counting the total number of additions, multiplications, function evaluations and gradient evaluations for a given iteration of the minimization routine and multiplying the operation count by a representative execution time for these operations. The communication time between the processor and the memory is ignored. This estimate, however, is problem dependent.

The dominant time consumer in the minimization routine is the function evaluation time which is called by the minimization subroutine and by the gradient subroutine.

A function evaluation consists of integrating a set of differential equations over a defined time interval and computing a suitably defined error function. Thus, the function evaluation time depends primarily on the numerical integration method employed and the complexity of the differential equations being employed.

4.2.1 Number of Operations Per Integration Step

Because of its extensive use in a wide variety of areas, as well as its high degree of accuracy, the fourth-order Runge-Kutta (RK4) integration method was used exclusively to obtain the results presented in the previous sections. For the run time estimation, three other integration methods were chosen for comparison: second-order Runge-Kutta method (RK2), Adams-Bashforth second-order predictor (AB2) and Adams-Moulton (Predictor-corrector) fourth-order (AM4). A short review of these methods and the selection of the integration step size are given in Appendix B.

The following letters may be assigned to represent the number of arithmetic operations:

n = the dimensionality of the system

A = additions/subtractions

M = multiplications/divisions

F = function evaluations.

For RK2 implementation, the total arithmetic operations which must be carried out per step:

$$2F + (4A + 2M)n \qquad (4.2.1\text{-}1)$$

For RK4 implementation the number is

$$4F + (9A + 11M)n \qquad (4.2.1\text{-}2)$$

For the AB2 predictor the number is

$$1F + (2A + 3M)n \qquad (4.2.1\text{-}3)$$

and for the AM4 predictor the number is

$$2F + (8A + 11M)n \qquad (4.2.1\text{-}4)$$

Number of arithmetic operations for evaluating $\dot{x} = f(x)$ for the example considered:

20M + 8A +2 square root evaluations

Each iteration of a square root can be regarded as 4M + 2A evaluations. Considering only one square root iteration per call, the total number of operations will be

$$F = 20M + 8A + (4M + 2A) = 24M + 10A \qquad (4.2.1\text{-}5)$$

The number of operations required for a single integration step is given in Table V for n = 4.

TABLE V

NUMBER OF ARITHMETIC OPERATIONS NEEDED
PER SINGLE INTEGRATION STEP

Method	Additions	Multiplications	Weighted operations assuming $t_m = 2t_A$
RK4	76	140	356
AM4	52	92	236
RK2	36	56	148
AB2	18	36	90

As reported earlier, the off-line tests of the proposed algorithm employed the RK4 method because of its simplicity in coding and its self start characteristics and not because of its efficiency (see Table V). The Adams-Moulton formulas have a significantly smaller truncation error than the Adams-Bashforth formulas, for comparable order methods. For example, the fourth-order Adams-Moulton formula has a truncation error 0.076 times that of the fourth-order Adams-Bashforth formula. This is the principal reason for using the implicit formulas, although there are other considerations.

The fourth-order Adams-Moulton formula has over twice the truncation error of the Simpson method

$$(X_{i+1} = X_{i-1} + \frac{h}{3}[f(X_{i-1},t_{i-1}) + 4f(X_i,t_i) + f(X_{i+1},t_{i+1})]).$$

The reason for using the Adams-Moulton formula is that it has much better stability properties than the Simpson rule.

Based on the above results and considerations, it is concluded that for real time applications the integration method used should be the AM4 or the AB2, and not the RK4 as implemented for off-line studies.

4.2.2 Number of Arithmetic Operations per Function Evaluation

Let L = number of steps of the integrations
I = number of operations per integration
P = number of parameters to be identified

Then the number (R) of operations per function evaluation will be:

$$R = PL(3A + 2M) + LI \qquad (4.2.2\text{-}1)$$

Substituting some representative numbers and using the AM4 integration method:

P = 4 for control calculations

I = 52A + 92M

L = 10

Then

$$R = 4 \times 10(3A + 2M) + 10(52A + 92M) = 640A + 1000M$$
$$= 2640 \text{ equivalent operations}$$

For the AB2 integration method:

P = 4

I = 18A + 36M

L = 5

$$R = 4 \times 5(3A + 2M) + 5(18A + 36M) = 150A + 220M$$
$$= 590 \text{ equivalent operations}$$

4.2.3 Number of Arithmetic Operations per Gradient Evaluation

Let G = number of operations needed for gradient evaluation

$$G = P.R \qquad (4.2.3\text{-}1)$$

Tables VI and VII summarize the number of operations needed for function and gradient evaluation, assuming $t_m = 2t_A$.

TABLE VI

NUMBER OF WEIGHTED OPERATIONS NEEDED FOR
FUNCTION AND GRADIENT EVALUATION
(5 steps of integration)

	Integration Method	
phase	AM4	AB2
Identification	Function: 1250	Function: 520
P = 2	Gradient: 2500	Gradient: 1040
Control	Function: 1320	Function: 590
P = 4	Gradient: 5280	Gradient: 2360

TABLE VII

NUMBER OF WEIGHTED OPERATIONS NEEDED FOR
FUNCTION AND GRADIENT EVALUATION
(10 steps of integration)

	Integration Method	
phase	AM4	AB2
Identification	Function: 2500	Function: 1040
P = 2	Gradient: 5000	Gradient: 2080
Control	Function: 2640	Function: 1180
P = 4	Gradient:10560	Gradient: 4270

4.2.4 Minimization Program - Number of Arithmetic Operations

The number of arithmetic operations for one iteration of the minimization routine (excluding function and gradient evaluation) are arranged in the order of the steps given in Appendix A:

Step 1: $P\,(\,1A + 4M) + 1A$

Step 2: $P^2(\,1A + 1M) + PM + 1M$

Step 3: $P^2(\,1A + 1M) + P(2M + 3A) + 1M$

Step 4: $P\,(\,8A + 11M) + 7M + 1A$

Step 5: $P\,(\,2A + 4M) + A + 2M$

Step 6: $56M + 26A$

Step 7: $P^2(\,2M + 1A) + P(20M + 10A) + 3A + 3M$

The total number of operations (less function and gradient evaluation) needed per minimization iteration is given by

$$O = P^2(3A+4M) + P(24A+42M) + 32A + 70M \qquad (4.2.4\text{-}1)$$

using equation (4.2.4-1), the total operations for identification (p=2) is

$$O_1 = 92A + 170M = 432 \text{ weighted operations}$$

and the total operations for control evaluation (P=4) is

$$O_2 = 176A + 302M = 780 \text{ weighted operations}$$

Assuming 6 calls for function and 3 calls for gradient evaluation during the identification process (from Table III), the number of operations using AM4 (10 integration steps) is given by

$$N_1 = 6 \times 2500 + 3 \times 5000 + 432 = 30,432$$

and the number of operations using AB2 integration method is

$$N_1 = 6 \times 1040 + 3 \times 2080 + 432 = 12,912$$

Assuming 13 calls for function and 5 calls for gradient evaluation during the control calculation process (from Table III), the number of operations using AM4 (10 integration steps) is

$$N_2 = 13 \times 2640 + 5 \times 10560 + 780 = 87,120$$

and using the AB2 integration method

$$N_2 = 13 \times 1180 + 5 \times 4720 + 780 = 39,720$$

is obtained. Using the above values for N_1 and N_2, the total number of operations using the AM4 method is

$$N_1 + N_2 = 30432 + 87120 = 117,552$$

and the total number of operations using the AB2 method is

$$N_1 + N_2 = 12912 + 39720 = 52632$$

Taking the instruction time as 0.30 μsec (Z-8002 microprocessor), one cycle of the combined identification and control process will require an estimated execution time of

$$T = 117552 \times .3 \times 10^{-6} = 0.0353 \text{ sec. (AM4), or}$$
$$T = 52632 \times .3 \times 10^{-6} = 0.0158 \text{ sec. (AB2).}$$

Since one iteration of the adaptive control optimization (both identification and control) consumes about 0.035 sec. (AM4) or 0.016 sec. (AB2), it appears that for the application in this thesis, serial processing will be adequate. The step size for the integration routine does not play much of a role in the computational time, but it should be truncation error dependent, as explained in Appendix B. The truncation error limit should be no less than the order of the measurement resolution in order to conserve evaluation time.

4.2.5 Prediction of Timing

The motion platform hardware exhibits a natural frequency of 4 HZ (Fig. 20). The correct values for the integration step size should be determined by off-line simulation employing a variable step method that depends upon the truncation error. Assuming here that integration step size is 1/25 of the natural period of the system to be controlled:

$$h < \frac{1}{4} \times \frac{1}{25} = 0.010 \text{ sec.}$$

The maximum frequency of input to the system (from aircraft dynamics) is normally 2 HZ. Assuming an update of the control law at a rate of 2 x 6 = 12 HZ, then the update time is 1/12 = 0.080 sec.

This update time gives enough processor time to execute the proposed algorithm with more than 50% spare time when the AM4 integration method is used, or more than 80% when the AB2 integration method is used. Since the integration is executed during a fraction of the update time, the step size may be reduced, and a less accurate but faster integration method like the AB2 could be used.

4.3 Noise Expectation

A real nonlinear dynamic system and measurement model may be represented by

$$\dot{x}(t) = f[x(t),t] + B(t)w(t) \qquad (4.3\text{-}1)$$

$$z(t) = h[x(t),t] + v(t) \qquad (4.3\text{-}2)$$

where w(t) is an m-vector of process noise and v(t) is an r-vector of measurement noise that corrupts the observation z(t). Few approaches are given in the literature for real-time estimation where the noise processes w(t) and v(t) are mutually independent zero-mean white noise processes. Kaufman and Teavassos (20) suggested solving the above problem using an optimal estimation method.

A feasibility study for real-time utilization of a continuous Kalman filter was given by Gaston and Rowland (17). But, this method relies on the linearization of the model around operating point and not directly on the nonlinear equations of motion.

Sinaha and Kuszta (27) presented the regression method used to select the parameters of the model $Y_k = f(k) + W_k$, such that W_k is a zero-mean white noise sequence of least-possible variance. This latter method involves determining the parameters that will minimize the mean-square error

$$J = \frac{1}{N}\sum_{k=1}^{N} (Y_k - f(k))^2 = \frac{1}{N}\sum_{k=1}^{N} W_k^2$$

This method is comparable to the direct estimation method used in this thesis. Kauffman and Teavassos recommended the

use of optimal estimation method in the presence of appreciable noise but did not recommend limits for the validity of the direct estimation method. It appears that the reason for not recommending limits is that these limits are problem dependent and the approach taken should be left at the discretion of the system designer.

In order to predict whether noise might constitute a problem for the application under study, an experiment was conducted using existing simulator hardware. A position step input command was given to one of the three actuators controlling the C141 flight simulator motion platform and the actuator pressure transient was recorded using a Gould 2400S thermal writing recorder. This test was repeated several times and the response plots for the various tests are compared in order to determine the presence of independent noise input to the system.

Figure 21 is a diagram of the pressure measurement circuits. Pressure transducers (sensitivity of 200mv/3000 psi) mounting in actuator chambers 1 and 2, transmit the measurements via amplifiers (gain value of about 50). The summer amplifier then gives the differential pressure (Test

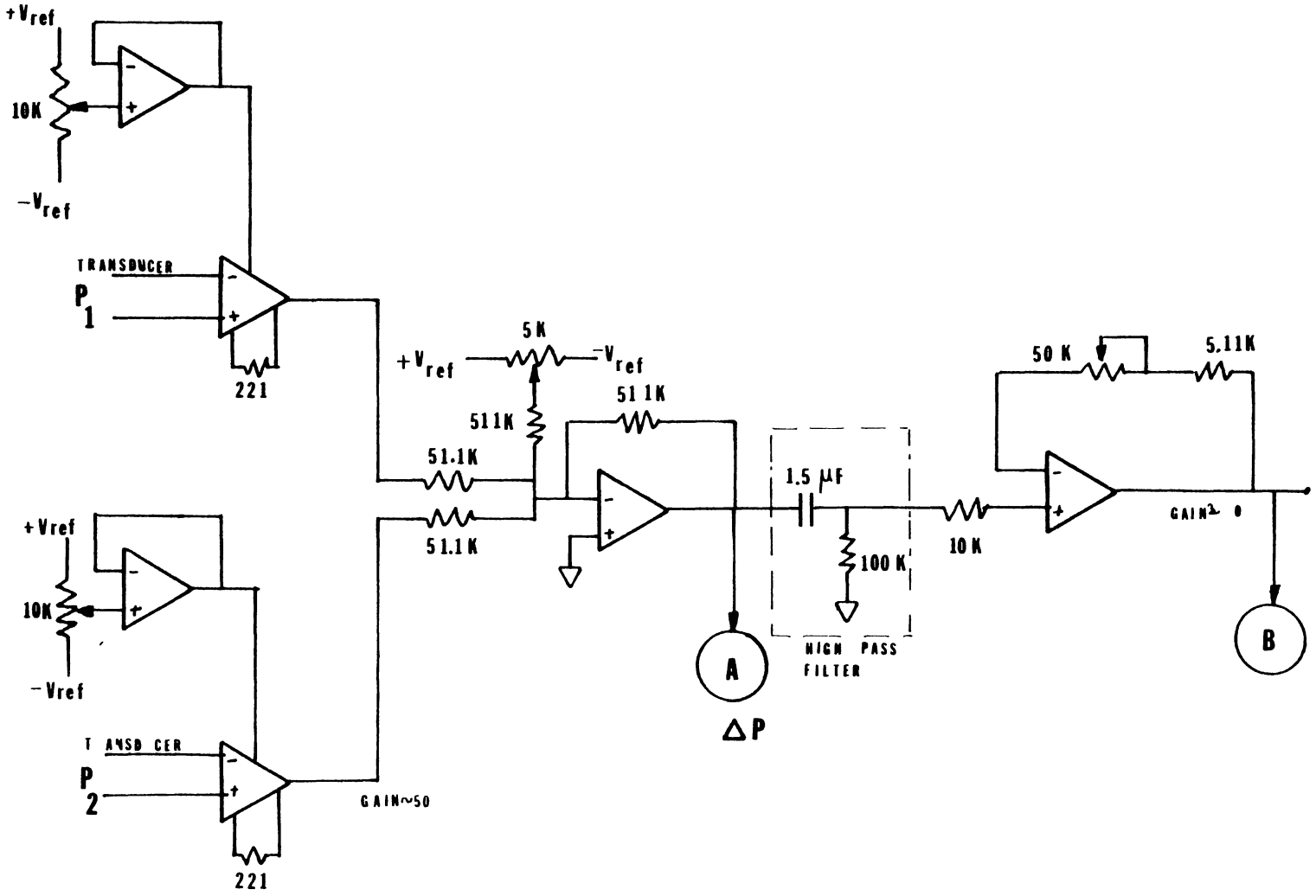

Figure 21. Pressure Measurement Schematics.

point A), and eliminates 60HZ and 400HZ noise corruption. This signal is transferred via a high pass filter (1.1 HZ break frequency) and amplified by a gain of about 3.0 (Test point B).

The measurements from test point A and test point B are given in Figure 22 for three identical step inputs to the system. (More identical runs were done with the same results). The recorder chart speed was 100 mm/sec and the full scale of the output was 5 volts.

Figure 23 is an enlargement (x4) of test point A recordings for the three runs, and Figure 24 is an enlargement (x3) of test point B recordings. By comparing the three runs, it can be observed that no appreciable random error or nonrepeatability exists in these responses. All three graphs are identical except for a small shift in run no. 2. This shift can be explained by the deviation of the starting point. The three axis motion platform by itself is a nonlinear system. Thus a small change of the starting point results in a different geometry and force balance. There is also a small nonlinearity in the chart recorder.

Measurement error prediction

The following data were taken from the instrument specifications:

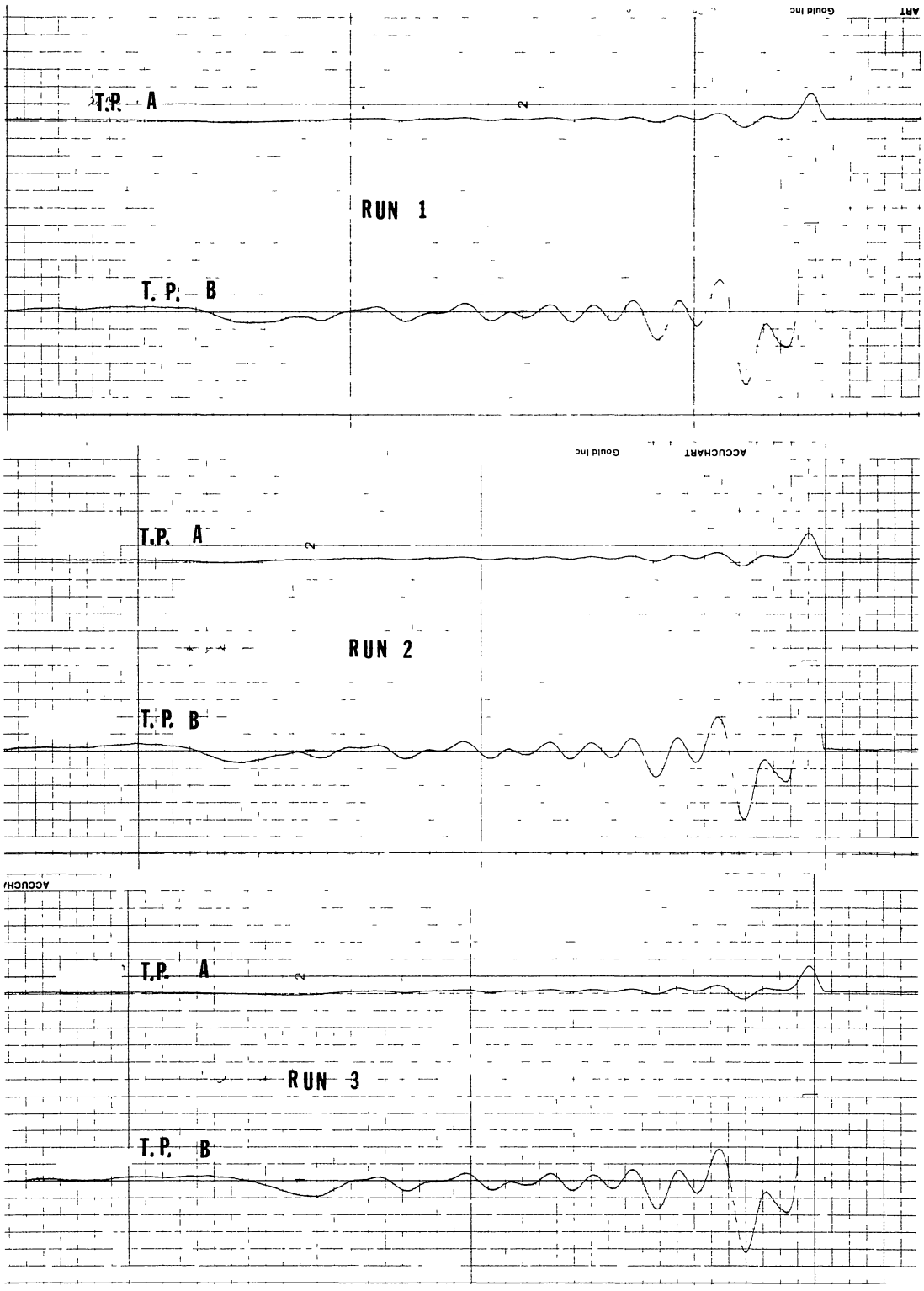

Figure 22. Pressure Measurement Test Results.

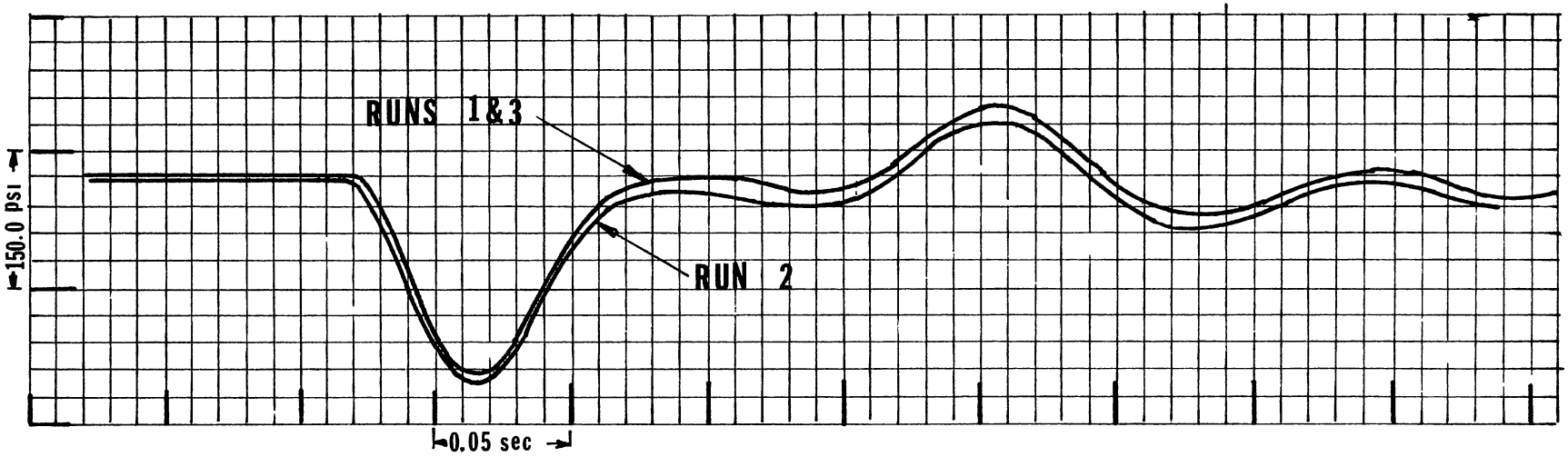

Figure 23. Test point A Recording for three runs (Enlargement X4)

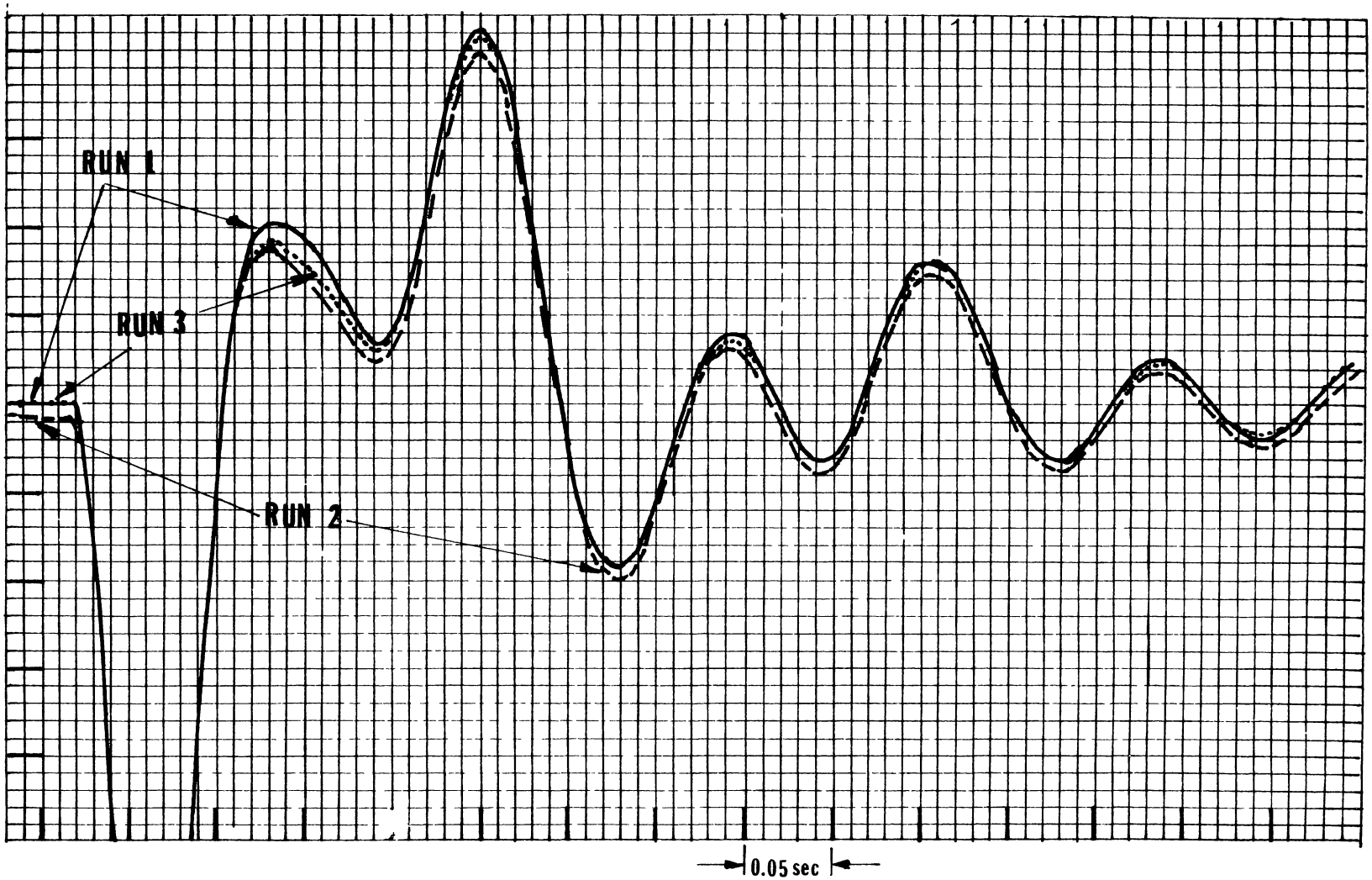

Figure 24. Test point B Recording for three runs (Enlargement X3)

Pressure Transducer: (MOOG Model 131-117):

Natural Frequency:	40 KHZ
Rise Time (90% F.S.):	0.4 msec
Hysteresis:	± .25% full scale
Repeatability:	< 0.1%
Static error:	< 0.25%

Displacement/Velocity Transducer

(Tempsonics Series DCTM)

Non-linearity:	< 0.05%
Repeatability:	< ±0.02%

Tempsonics claims that 60 HZ or 400 HZ noise will be rejected by the narrow band-width detector utilized in the transducer.

Gould Chart Recorder (Model 2400S)

Rise time to 40mm:	< 4 msec with 1% overshoot
Noise:	0.1 mm peak to peak
Attenuation:	0.99 (30 HZ)
	0.98 (50 HZ)
	0.9 (125 HZ)
	0.707 (140HZ)
Chart speed:	100 ± 0.25 mm/sec.
50 division per channel	(1 mm/division)
Non-linearity:	±0.35% full scale
Chart wander:	±0.25 mm

Actuator friction: 7.6 Lbf

(the equivalent value in differential pressure is $\frac{7.6}{8.295} = 0.916$ psi)

Accumulating the Data above

(in terms of pressure variables):

Frequency response for chart recorder ~1.0 psid

Chart wander = ±0.25 mm = ±7.5 psid

Recorder nonlinearity(50mm ~1500psi) =

1500 x 0.35% = 5.25 psid

Pressure transducer repeatability = 0.1% x 3000.0=3psid

Friction in actuator ~1 psid

The overall error expected for the chart recording:

$$e = \sqrt{5.25^2 + 7.5^2 + 3^2 + 1^2 + 1^2} = 9.73 \text{ psid}$$

The overall error expected for the pressure measurement alone:

$$e = \sqrt{3^2 + 1^2} = 3.1 \text{ psid}$$

The overall error expected in measuring the differential pressure is 3.1 psid, plus the static error which may be as large as 0.25% (7.5 psi). But since the identification process needs only incremental values from the previous measurement, such an error will not constitute a problem.

Furthermore, during large pressure transients, the pressure may be changed by up to 60 psi per 0.02 sec. integration interval (see Figure 23).

CHAPTER V

SUMMARY AND RECOMMENDATIONS

The purpose of this research was to develop a new synthesis method for adaptive control of nonlinear dynamic systems with application to a one-degree-of-freedom motion simulator.

Digital computer capabilities along with existing control theories and reported optimization algorithms were integrated to develop a new approach to the solution of the control problem for a broad class of systems where actuator nonlinearities and dynamics dominate the system response.

It was demonstrated that for the class of systems studied, the controller design is substantially different if actuator dynamics are included rather than ignored. The adaptive control scheme leads to substantially improved performance compared to the conventional "fixed gain" design.

The method developed is especially useful for control problems where the frequencies of the inputs to the system are close to the natural frequencies of the system and a multi-input multi-output configuration is required.

5.1 Summary

In section 1.1, the requirements of a motion platform

were given. The important problem of producing the washout motion needed to limit cues because of physical constraints was defined. Another problem identified was that the servo-actuator states were not employed in previous work (Ref. 2-11) in the design of control systems for motion platforms.

The potential for the utilization of digital computers in closed-loop electrohydraulic control systems was discussed in section 1.2. A conclusion was reached that a good application of computers in such systems may be in adaptive control. Methods of nonlinear control system synthesis were discussed in brief in section 1.3, and the recommendation of Kaufman and Teavassos of using direct gain optimization and direct estimation algorithm was given.

Section 1.5 evaluated the need for the development of an integrated, adaptive control scheme that includes the dominant nonlinearities and actuator subsystem dynamics in the plant model.

Methods and procedures used in the research to develop the adaptive control scheme were discussed in chapter II. The nonlinear identification and control algorithm developed in this chapter utilized the nonlinear process equations directly. The direct gain optimization and direct parameter optimization procedure used the Q-N (no line search) minimization algorithm reported by Davidon (25). The washout algorithm receives information (position,velocity), performs

decisions (when to switch mode of operation) and the input to the control system, based on the system response prediction is calculated.

Chapter III was devoted to the application model study. A general hardware configuration for a typical one degree-of-freedom motion platform was established and the nonlinear equations of motion were derived.

These equations were linearized about a steady-state operating point, and linearized control law was developed based upon the excess pole specification method (14). By comparing response characteristics of the linearized versus nonlinear model for the same control law, it was decided that controller gains derived from the linearized system of the case chosen are suitable to be initialized in the non-linear adaptive controller.

The methods discussed in chapter II were employed to develop an adaptive control law for the application example. An off-line simulation program was written and executed on SEL 32/27 minicomputer (at Burtek Inc., Tulsa, OK). Three runs were made to test the response characteristics of the control system to three different time-dependent inputs. The first was low frequency acceleration input; the second was high frequency acceleration input, and the third was ramp input used to test the washout algorithm. The results of the simulation were evaluated against a traditional closed-loop electrohydraulic servo design. It was shown

that the proposed method improves system performance in terms of attenuation and phase lag for a wide spectrum of inputs, subject to hardware constraints. It eliminates the need for the trial-and-error process often used in tuning control system gains during hardware integration and eliminates the need for periodic adjustment of these gains due to drifts. The inclusion of the actuator system dynamics in the plant model and the development of an integrated control law simplify the control synthesis: The acceleration (force) control law was calculated directly from pressure measurements.

The control law devised in this thesis is adaptive in the sense that the control mode can be changed from a follower mode to a washout mode and vice versa automatically. The use of a separate washout mode eliminates the need for washout filters.

Implementation feasibility is discussed in chapter IV. Computer memory requirements for the application chosen look modest, about 5K-bytes of memory for on-line execution. Sequential processing for a one degree-of-freedom motion platform is possible. Execution estimates for the application model under study using an add/subtract time of 0.3 microsec., 10 integration steps; was 0.035 sec (AM4 integration method) or 0.016 sec. (AB2 integration method).

In order to predict whether noise might constitute a problem for the application under study, a test was conduc-

ted using existing motion platform hardware (of a C141B simulator), and no noticeable independent random noise was recorded.

Simulations of the proposed algorithm were executed using a data base word size of 32 bits. Since commercial A/D and D/A converters have up to 16 bits resolution (0.0015% accuracy), the data base word size for an on-line execution should be chosen accordingly. It is also recommended that the step size for on-line integration should be based upon the truncation error that will be determined by off-line simulation, employing a variable step size/variable order integration routine. The truncation error permitted by this selection should not be smaller than the accumulated dynamic measurements error (problem dependent).

Another recommendation, based on the identification results in Section 3.4.3 is that a "dither" signal input be used to excite the system, not only to minimize valve "stiction" effects, but to improve the identification process for parameters that affect the spring constant of the actuator (e.g., bulk modulus of the fluid).

5.2 Areas for Further Research

Areas recommended for future research are as follows:

1. Work is needed to define the acceptable limit of noise (and related number of integration steps) for which the method of this thesis can be used.

2. A quantitative exit criterion is needed for the optimization routine that will both suppress instability and conserve execution time.

3. The application example studied in this thesis was a one degree-of-freedom motion platform (one control, four states). Motion platforms with multiple degree-of-freedom have a multi input/multi output configuration. While the modelling of such a system with actuator dynamics incorporated is no major problem, research is needed into the parallel processing of the identification and control phases for such a case, so that all computations can be performed during run time.

It may be that the solution to such problems is the use of a high-speed array processor. For example, the specifications of the NUMERIX MARS 432 array processor include an add or multiply time of 0.1 microsecond, a real/real vector multiply time of 0.2 microsecond and a square root calculation time of 0.8 microseconds.

4. This research was conducted using the lumped parameter approach: This approach is justified when the servovalve is mounted on the actuator, and the connecting line dynamics are negligible. An interesting area for research is the control of a distributed parameter system. K.N.Reid (28) discussed dynamic models of fluid transmission lines, but work is needed in their application to control system design.

5. A feasibility study of implementation, cost and reliability of the proposed design should be made prior to hardware design.

It is hoped that this research will motivate future research and design in the area covered by this thesis, and will enhance the solution of control problems involving systems which employ hydraulic actuation.

BIBLIOGRAPHY

1. Martin, E. A. 1980. Motion and Force Simulation System. Course notes from University of Dayton Flight Simulation Short Course.

2. Parrish, R. V., Dieudonne, J. E. and Martin, D. J. 1973 Motion Software for a Synergistic Six Degree-of-Freedom Motion Base. NASA TN-D-7350.

3. Dieudonne, J. E., Parrish R. V. and Bardusch R. E 1972. An Actuator Extension Transformation for a Motion Simulator and an Inverse Transformation Applying Newton-Raphson's Method. NASA TN-D-7067.

4. Parrish, R. V., Dieudonne, J. E., Bowles R. L. and Martin, D.J. 1973. Coordinated Adaptive Washout for Motion Simulators. AIAA paper 73-930.

5. Parrish, R. V. and Martin, D. J. 1975. Empirical Comparison of a Linear and a Nonlinear Washout for Motion Simulators. AIAA paper 75-106.

6. Kosut, R. L. 1978. Optimal Control Theory Applied to the Design of Cue Shaping Filters for Motion-Base Simulators. AIAA paper 78-1575.

7. Harris, W. T., Puig, J. A., Ricard G. L. and Weinman D. G 1978. Motion: Methods and Requirements. AIAA paper 78-1576.

8. Jaslow, H. 1980. New Engineering Approach to Motion Cueing Technology for Flight Simulators. AIAA paper 80-0047R.

9. Jaslow, H. 1981. Critique of the Graviey Vector Alignment Method for Motion Simulation. AIAA paper 81-0985-R.

10. Sturgeon, W. R. 1980. Controllers for Aircraft Motion Simulators. AIAA paper 80-050.

11. Kurosaki, M. 1978. Optimal Washout for Control of a Moving Base Simulator. Proceedings of the Seventh Triennial World Congress of the International Federation of Automatic Control, Helsinki, Finland.

12. MIL-STD-1588. Six Degree of Freedom Motion System Requirements for Aircrewmember Training Simulators.

13. Maskrey, R. H. The Role of Microprocessors in Closed Loop Electrohydraulic Control Systems. Proceedings of the 34th National Conference on Fluid Power.

14. Merriam, C. W. 1974. "Automated Design of Control Systems". Gordon and Breach Science Publishers. New York, N.Y.

15. Kirk, D. E. 1970. "Optimal Control Theory". Prentice Hall, Inc., Englewood Cliffs, New Jersey.

16. "Nonlinear System Analysis and Synthesis". 1978. Vol I, ASME Publication.

17. Gaston, J. A. and Rowland, J. R. 1975. Realtime Digital Integration for Continuous Kalman Filtering in Nonlinear Systems. Comput. & Elect. Eng. Vol. 2, pp. 131-140.

18. Garrard, W. L. 1972. Suboptimal Control for Nonlinear Systems. Automatica, Vol. 8, pp. 219-221.

19. Singh, R. N. P. and Johnson, T. L. 1981. A Functional Expansion Approach to the Solution of Nonlinear Feedback Problems. IEEE Transactions on Automatic Control, Vol. AC-26, No. 2.

20. Kaufman, H. and Teavassos, R. 1981. Parallel Computation for Developing Nonlinear Control Procedures. AFWAL-TR-81-3016.

22. Luenberger, D. G. 1973. "Introduction to Linear and Nonlinear Programming". Addison-Wesley Pub. Co. Reading, Massachusetts.

23. Fox, R. L. 1971. "Optimization Methods for Engineering Design". Addison-Wesley Pub. Co. Reading, Mass.

24. Graupe, D., Jain, V. K. and Salahi, J. 1980. A Comparative Analysis of Various Least-Squares Identification Algorithms. Automatica, Vol. 16, pp. 663-681.

25. Davidon, W. C. 1975. Optimally Conditioned Optimization Algorithms Without Line Searches. Mathematical Programming 9(1975) 1-30.

26. Morf, M., Dobbins, J. R., Friedlander, B. and Kailath, T. 1979. Square-Root Algorithms for Parallel Processing in Optimal Estimation. Automatica, Vol. 15, pp. 299-306.

27. Sinha, N. K. and Kuszta, B. 1983. "Modelling and Identification of Dynamic Systems". Van Nostrand Reinhold Co. Inc.

28. Reid, K. N. 1968. Dynamic Models of Fluid Transmission Lines. Proceedings of the Symposium on Fluidics and Internal Flows, Pennsylvania State University, October 24-25,1968.

APPENDIXES

APPENDIX A

ALGORITHM FOR UNCONSTRAINED MINIMIZATION

From the material discussed in Chapter II it is evident that an efficient algorithm for minimization of a nonlinear function, called the performance cost function must be employed. In the literature, the subject is discussed as nonlinear optimization or nonlinear programming.

The optimization problem will be of the form

minimize $f(x)$, subject to $x \in \Omega$ where f,

is a real-valued function and Ω, the feasible set, is a subset of E^n. For the completely unconstrained case, $\Omega = E^n$.

Usually optimization problems have constraints, like the differential constraints ($\dot{x} = f(x,t)$) of the dynamic system; however, some of the most powerful and convenient methods of solving constrained problems involve the conversion of the problem to one of unconstrained minimization.

A.1 Some Properties of a Minimum

The usual notion of a minimum is a point where a function has its least value, i.e. x_m such that $F(x_m) \leq F(x)$

for all x.

For a function of an n variable $x \equiv (x_1, x_2, \ldots, x_n)$ with continuous derivatives, the minimum will be a point where

$$\frac{\partial F}{\partial X_i} = 0 \qquad i = 1,2,\ldots,n \qquad \text{(A.1-1)}$$

A point satisfying the last equation is guaranteed to be a relative minimum if the Hessian matrix of F is positive definite (all eigenvalues are positive). The Hessian matrix is defined:

$$J = \begin{bmatrix} \frac{\partial^2 F}{\partial X_1^2} & \frac{\partial^2 F}{\partial X_1 \partial X_2} & \cdots\cdots\cdots & \frac{\partial^2 F}{\partial X_1 \partial X_n} \\ \vdots & & & \vdots \\ \frac{\partial^2 F}{\partial X_n \partial X_1} & \frac{\partial^2 F}{\partial X_n \partial X_2} & \cdots\cdots\cdots & \frac{\partial^2 F}{\partial X_n^2} \end{bmatrix} \qquad \text{(A.1-2)}$$

Geometrically, this property indicates that the quadratic function that approximates the original function at the point has its minimum there. Consider the Taylor series expansion of F about X_m up to quadratic terms:

$$F(x) - F(X_m) + \sum_{i=1}^{N} \left(\frac{\partial F}{\partial X_i}\right)_{X_m} (X_i - X_{mi}) + \tfrac{1}{2} \sum_{j=1}^{N} \sum_{i=1}^{N} \left(\frac{\partial^2 F}{\partial X_i \partial X_j}\right) (X_i - X_{mi})(X_j - X_{mj}) \qquad \text{(A.1-3)}$$

In matrix form this becomes:

$$F(x) \simeq F(X_m) + (X - X_m)^T \nabla F(X_m) + \tfrac{1}{2}(X - X_m)^T J_m (X - X_m) \qquad (A.1\text{-}4)$$

The problem of relative minima is one of the most vexing in optimum methodology. This is because most of the viable methods can seek only a relative minimum and cannot converge in the general case to the least minimum.

A.2 Gradient Methods for Minimization

A.2.1 The Gradient

Central to these methods is the concept of the gradient of the function being minimized. This vector, denoted F, lies in the direction of greatest rate of change of the function and has that rate of change as its magnitude. The gradient of the function F(X) is defined as

$$\nabla F \equiv G \equiv \left(\frac{\partial F}{\partial X_1}, \frac{\partial F}{\partial X_2}, \ldots, \frac{\partial F}{\partial X_n} \right) \qquad (A.2.1\text{-}1)$$

A.2.2 Line Search

Consider any vector S_q and the move prescription

$$X = X_q + \alpha S_q \qquad (A.2.2\text{-}1)$$

Then if α is considered a variable, the locus of X for a range of values of α is a straight line. Substitute this formally in F(X).

$$F(X) + F(X_q + \alpha S_q) = F(\alpha) \qquad \text{(A.2.2-2)}$$

Since F can be considered a function of α alone (X_q and S_q are fixed). The value of α which minimizes $F(\alpha)$ is denoted α^* and can be obtained by differentiating

$$\frac{dF}{d\alpha} = 0 \qquad \text{(A.2.2-3)}$$

subject to

$$\frac{d^2F(\alpha)}{d\alpha^2}\Big|_{\alpha=\alpha^*} > 0 \qquad \text{(A.2.2-4)}$$

This process is called a line search and usually is done by quadratic or cubic interpolation.

A.2.3 Gradient Algorithms

The procedure for function minimization by gradient methods is described in the flowchart of Figure 25. A few of the methods are given below.

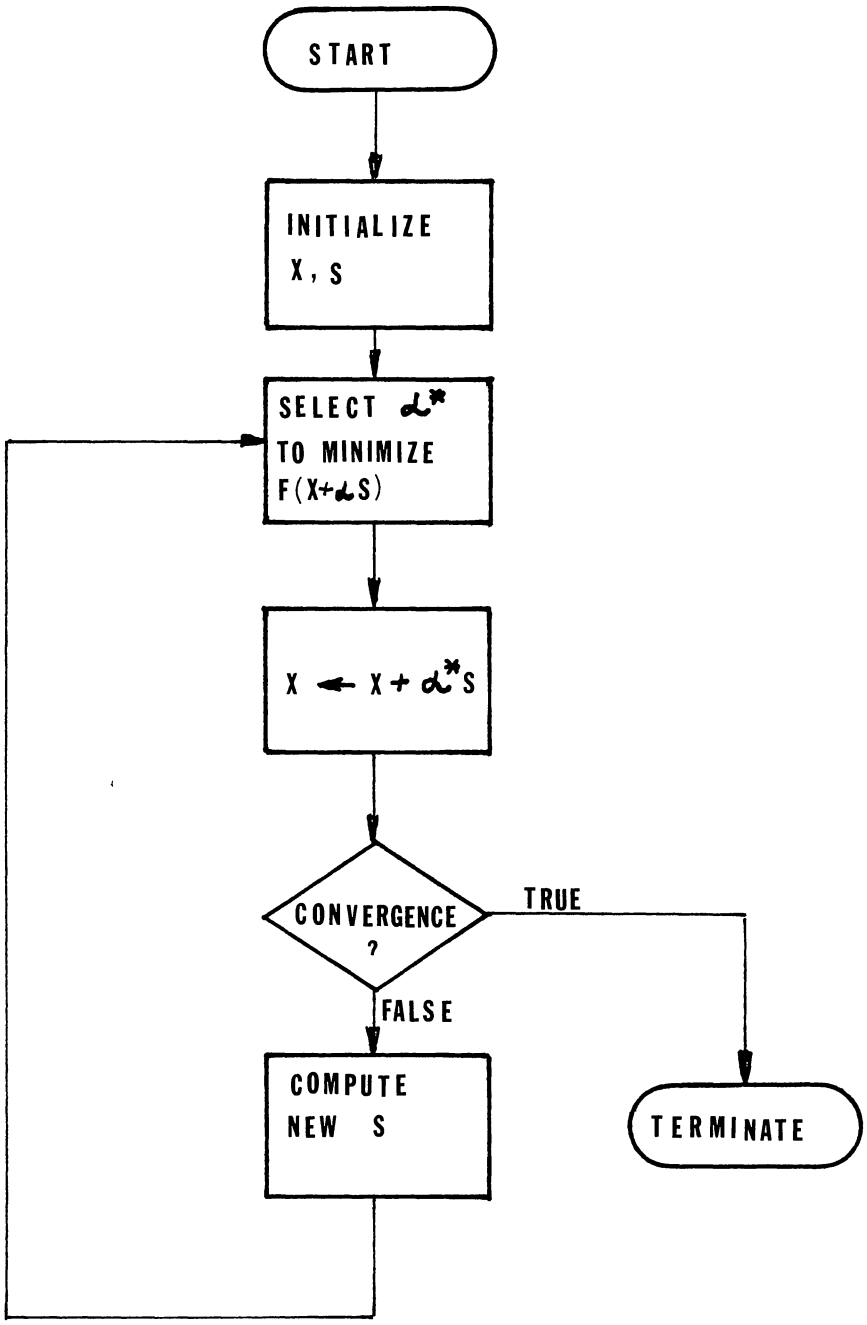

Figure 25. Gradient Algorithm Flow Chart.

A.2.3.1 Newton's Procedures. Expanding the value of F by Taylor series expansion up to the quadratic term

$$F(x) = F(X_q) + \nabla F_q^T (X-X_q) + \tfrac{1}{2}(X-X_q)^T J_q (X-X_q) \quad \text{(A.2.3.1-1)}$$

Where J_q is the matrix of second partial derivatives of F evaluated at X_q. If $\overline{F}_q(X)$ is an approximation to the minimum of F(X), then X should satisfy the vector equation

$$\nabla \overline{F}_q = \nabla F_q + J_q(X-X_q) = 0$$

or

$$J_q X = J_q X_q - \nabla F_q \quad \text{(A.2.3.1-2)}$$

multiplying through by the inverse of J_q,

$$X_{q+1} = X_q - J_q^{-1} \nabla F_q \quad \text{(A.2.3.1-3)}$$

Since F usually is not quadratic in X, the process can be improved by selecting

$$S = J_q^{-1} \nabla F_q \quad \text{(A.2.3.1-4)}$$

and producing a line search to get

$$X_{q+1} = X_q - \alpha^*_q s \qquad \text{(A.2.3.1-5)}$$

A.2.3.2 Davidon-Fletcher-Powell Method. (a quasi-Newton method). In this method the local Hessian J_q^{-1} is replaced by an approximate metric H_q. This substitution eliminates the need for evaluating second derivatives and performing matrix inversions.

Algorithm:

1. Start with X_0 and $H_0 = I$
   set $S_0 = -H_0 \nabla F_0$
2. Compute $X_{q+1} = X_q + \alpha^*_q S_q$
   where α^*_q minimizes $F(X_q + \alpha S_q)$.
3. Compute $H_{q+1} = H_q + M_q + N_q$
   where
   $$Y_q = G_{q+1} - G_q = \nabla F(X_{q+1}) - \nabla F(X_q)$$
   $$M_q = \alpha^*_q \frac{S_q S_q^T}{S_q^T Y_q}$$
   $$N_q = - \frac{(H_q Y_q)\,(H_q Y_q)^T}{Y_q^T H_q Y_q}$$
4. Compute $S_{q+1} = -H_{q+1} G_{q+1}$
   and repeat from step 2.

A.2.3.3 Davidon's optimization algorithm (without line

search). A representation of the algorithm was given by Davidon (Ref 25). M. J. D. Powell[1] has written: "Numerical experiments with Davidon's algorithm indicate that it may be the best numerical method for calculating the least value of a differentiable function of several variables".

Algorithm

Start evaluate $f(x)$, $g(x)$

set: $X_0 = X$ $f_0 = f$ $k_0 = J^T g$ $w = k_0$

1. Define $s = -k_0$, $f_0 = k_0{}^T s$, $\lambda = 2$

   If $4f_0 \geq -f'_0$ go to step 2.

   If not, set $s = -S4\ f_0/f'_0$

   and $f'_0 = -4f_0$

2. Set $X = X_0 + JS$

   If $-f'_0 < \epsilon$ stop. If not evaluate $f(X)$.

   If $f < f_0$ go to step 3

   If not set $S = S/2$, $f'_0 = f'_0/2$, $\lambda = 0.5$

   and repeat this step.

3. Evaluate $g(x)$. Set $k = J^T g$, $f' = k^T s$

   $b_0 = f' - f'_0$, $m = s+k_0-k, X_0=X, f_0=f, k_0=k, f'_0=f'$

   If $b_0 \geq \epsilon$ go to step 4

   If not, set $S = S\lambda$, $f_0' = f_0'\lambda$ and return to step 2.

---

[1]Powell, M. J. D. 1977. Quadratic Termination Properties of Davidon's New Variable Metric Algorithm. Math Programming 12, pp. 141-147.

4. Define $m^2 = m^T m$

If $m^2 < \epsilon$ return to step 1

Else, define: $\nu = m^T s, \mu = \nu - m^2, u = w - m m^T w/m^2$

If $10^6 (m^T u)^2 < m^2 u^T u$ go to step 4a, else,

set n = m x 1 null vector, $n^2 = 0$ and go to step 5

4a. set $n = u u^T s/u^T u$, $n^2 = (u^T s)^2/u^T u$

5. Set $b = n^2 + \mu\nu/m^2$

if $b \geq \epsilon$ go to step 6

Else $n = s - m\nu/m^2$, $n^2 = b_0 - \mu\nu/m^2$, $b = b_0$

6. If $\mu\nu < m^2 n^2$ go to step 6a

Else set: $\gamma = 0$, $\Delta = (\nu/\mu)^{\frac{1}{2}}$ and go to step 7

6a. Set $a = b - \mu$, $c = b + \nu$, $\gamma = \sqrt{\dfrac{1 - \mu\nu/m^2 n^2}{ab}}$

and $\Delta = \sqrt{c/a}$

If $c \geq a$ go to step 7, else, set $\gamma = -\gamma$

7. Set: $\alpha = \nu + \mu\Delta + m^2 n^2 \gamma$

$P = m(\Delta - n^2\gamma) + n\gamma\nu$

$q = m(1 + n^2\gamma)/\alpha - n\gamma\mu/\alpha$

$w = mn^2(1 + \gamma\mu\nu/\alpha) - n(1 + \Delta)\mu\nu/\alpha$

$k_0 = k_0 + pq^T k_0$, $J = J + J_q p^T$

If $n^2 > 0$ return to step 1. If not, set $w = k_0$

and return to step 1

APPENDIX B

NUMERICAL INTEGRATION METHODS USED IN SECTION 5.2

B.1 Runge-Kutta Methods

Integration methods may be classified as either single-step or multi-step methods. One important group of single-step algorithms are the Runge-Kutta (RK) methods. For differential equations of the form $dx/dt = f(x,t)$, RK methods require the evaluation of $f(x,t)$ at two, three and four values of t on the interval $t_i \leq t \leq t_{i+1}$ for second, third and fourth order approximations, respectively.

B.1.1 Second-order Runge-Kutta method

$$X_{i+1} = X_i + \frac{h}{2}(m_0 + m_1) \qquad \text{(B.1.1-1)}$$

where h is the step size and

$$m_0 = f(x_i, t_i)$$
$$m_1 = f(x_i + m_0 h,\ t_{i+1}) \qquad \text{(B.1.1-2)}$$

The expression for m_1 in equation (B.1.1-2) uses a predicted

value of x at the end point of the interval. The corrector equation (B.1.1-1) utilizes the average between the initial slope (m_0) and a predicted slope (m_1) for proceeding over the entire interval.

B.1.2 Fourth-order Runge-Kutta method (RK4)

$$X_{i+1} = X_i + \frac{h}{6}(m_0 + 2m_1 + 2m_2 + m_3) \qquad \text{(B.1.2-1)}$$

where,

$$\begin{aligned} m_0 &= f(x_i, t_i) \\ m_1 &= f\left(x_i + \frac{m_0 h}{2}, t_i + \frac{h}{2}\right) \\ m_2 &= f\left(x_i + \frac{m_1 h}{2}, t_i + \frac{h}{2}\right) \\ m_3 &= f(x_i + m_2 h, t_i + h) \end{aligned} \qquad \text{(B.1.2-2)}$$

B.2 Multi-step Methods

Integration formulas which require information not only at t_i but also outside the integration interval under consideration (t_i, t_{i+1}), are referred to as multi-step methods. A disadvantage of these methods is the requirement for additional information to start the procedure. However these methods usually require considerably less computational time than single-step methods.

B.2.1 Adams-Bashforth second order Predictor (AB2)

This multi-step method utilizes a single past slope:

$$X_{i+1} = X_i + \frac{h}{2}(3\dot{x}_i - \dot{x}_{i-1}) + \frac{5}{12} h^3 \ddot{x}(\xi_i) \qquad \text{(B.2.1-1)}$$

B.2.2 Adams-Moulton (Predictor-corrector) fourth order (AM4)

$$\text{Predictor: } X_{i1} = X_i + \frac{h}{24}(55\dot{x}_i - 59\dot{x}_{i-1} + 37\dot{x}_{i-2} - 9\dot{x}_{i-3})$$

$$\text{Corrector: } X_{i+1} = X_i + \frac{h}{24}(9\dot{x}_{i1} + 19\dot{x}_i - 5\dot{x}_{i-1} + \dot{x}_{i-2}) - \frac{19}{720}h^5 x^5(\xi_i) \qquad \text{(B.2.2-1)}$$

B.3 Selecting the Step Size for Integration

The step size for integration may be selected using a predicted truncation error. This error is usually defined as the difference between the predictor and corrector formulas. This selection is method- and problem-dependent, and needs to be evaluated for the specific application. Kaufman and Teavassos (20) proposed the following procedure for the predictor/corrector equations

$$Y^P_{i+1} = Y^C_{i-1} + \frac{h_i}{3}(8f_i^p - 5f^C_{i-1} + 4f^C_{i-2} - f^C_{i-3}) \qquad \text{(B.3.1)}$$

$$Y^C_i = Y^C_{i-1} + \frac{h_i}{24}(9f_i + 19f^C_{i-1} - 5f^C_{i-2} + f^C_{i-3}) \tag{B.3.2}$$

The local truncation error associated with the Adams-Moulton corrector is

$$d_{i,c} \triangleq Y(t_i) - Y^C_i = -\frac{19}{720}h^5 y^{(5)}(\xi_i),\ \xi_i \in [t_o, t_i] \tag{B.3-3}$$

By definition the local truncation error associated with the predictor is given by

$$d_{i+1,p} \triangleq Y(t_{i+1}) - Y^P_{i+1} \tag{B.3-4}$$

Assuming

$$Y(t) = t^5,\ t \geq 0.0 \quad \text{and} \quad Y' = f(y,t) = 5t^4$$

and substituting into the predictor equation (B.3-2)

$$d_{i+1,p} = \frac{232}{720} h^5\ y^{(5)}\ (\xi_2) \tag{B.3-5}$$

Then, assuming $\xi_1 = \xi_2 = \xi$

$$d_{i+1,p} - d_{1,c} = Y(t_{i+1}) - Y^{P}_{i+1} + (Y^{C}_{i} - Y(t_i))$$

$$= \frac{251}{720} h^5 Y^{(5)}(\xi) \qquad \text{(B.3-6)}$$

Using these results

$$d_{i,c} = -\frac{19}{251}[Y^{C}_{i} - Y^{P}_{i+1} + \delta_Y] \qquad \text{(B.3-7)}$$

where

$$\delta_Y = Y(t_{i+1}) - Y(t_i)$$

Using the Schwartz and triangle inequality

$$|d_{i,c}| \leq \frac{19}{251}\left\{|Y^{C}_{i} - Y^{P}_{i+1}| + |\delta_Y|\right\}$$

By approximating $|\delta_Y| = h_i |f[y(t_i), t]| + |O(h_i^2)|$,

then $d_{i,c} \leq \frac{19}{251}\left\{|Y^{C}_{i} - Y^{P}_{i+1}| + h_i |f(Y^{C}_{i}, t_i|\right\}$ (B.3-8)

The procedure that was advised by Kaufman and Teavassos was to evaluate the predictor corrector equations, then to estimate the local truncation error.

$$d_{ic} = \frac{19}{251}\left\{\left|Y^{C}_{i} - Y^{P}_{i+1}\right| + h_i \left|f^{C}_{i}\right|\right\} \qquad \text{(B.3-9)}$$

if $d_i > \epsilon$ max; replace $h_i \leftarrow h_i/2$

if $d_i < \epsilon$ min; replace $h_i \leftarrow h_i x2$

else leave h_i .

At present, the most popular predictor-corrector algorithms control the truncation error by varying both the step size and the order of the method; and all of these algorithms use the Adams family of formulas.

APPENDIX C

LISTING OF SIMULATION TEST PROGRAM

```
$JOB B NLCNT2 AMOS SLOF=L NLCNT2
$OPTION 2 3 4 5
#FORT77
C
C
C
            PROGRAM NLCONT
C
C
C     THIS PROGRAM DEMONSTRATES NONLINEAR CONTROL SYSTEM SYNTHESIS
C     WITH APPLICATION TO ELECTRO HYDRAULIC SERVO SYSTEM
C
C
C          SUBROUTINES CALLED   RK41,GNDAV1 LXYPLT
C
C
C       DEFINE LOCAL VARIABLES
C
       REAL*4         REPSILON    ' MINIMIZATION LIMIT
       REAL*4         RRKI(4)     ' TEMPORARY VALUE FOR GAINS
       REAL*4         RKFM(4)     ' TEMPORARY VALUE FOR PARAMETERS
       REAL*4         ENDTIME     ' END TIME OF SIMULATION
       INTEGER*2      NS          ' NUMBER OF VARIABLES FOR MINIMIZATION
       INTEGER*2      ITERI       ' NO  OF ITERATION FOR IDENTIFIC
       INTEGER*2      ITERC       ' NO  OF ITERATIONS FOR CONTROL
       INTEGER*2      NITER       ' MAX ITERATION FOR MINIMIZATION
       LOGICAL*1      LPRINT      ' PRINT FLAG(MINIMIZATION)
       LOGICAL*1      LSYS        ' FOR SYSTEM IDENTIF/CONTROL
       LOGICAL*1      LBUG        ' FOR DEBUG ONLY
       LOGICAL*1      COM(65)
C
C
C     DEFINE COMMONS
C
       COMMON/GN/REPSILON NITER LPRINT LSYS
C
C
        INCLUDE C CONTR2
        INCLUDE C DERF2
C
C
C
C************************************************************
C
C    INITIALIZATION
C
1000   FORMAT(65A1)
1001   FORMAT(4X,65A1)
1002   FORMAT(4F18 10)
1003   FORMAT(4I10)
1004   FORMAT(4X 4F18 10)
1005   FORMAT( 1 )
C
C
C
       WRITE(1 1005)
       DO 1017 I=1 3
       READ(5 1000)COM
1017   WRITE(1 1001)COM
       READ(5 1003)N,ITERI ITERC
       READ(5,1002)((FM(I J),I=1,N) J=1 N)
       READ(5,1002)(GM(I) I=1 N)
       READ(5,1002)BEGX,ENDX DELX ENDTIME
       READ(5 1002)(RREF(I) I=1 N)
       READ(5,1002)(RK(I) I=1 N)
       READ(5 1002)(Y(I),I=1 N)
       READ(5 1002)(RQF1(I),I=1,N)
       READ(5,1002)(RQF2(I) I=1 N)
       READ(5,1002)(RFM(I) I=1 2)
       LPRINT= FALSE
       LBUG= FALSE   '***********FOR DBUG ONLY ********
C
      WRITE(1 200)
200   FORMAT( 0  10X, PLOT OF INPUT ACC OUT ACC POSITION CONTROL  /)
      WRITE(1 210)BEGX ENDX DELX ENDTIME
210   FORMAT(5X  START TIME=  ,F7 3,5X  TIME INTERVAL=   F7 3 5X,
     X  INTEGRATION STEP=   F7 3 5X 'ENDTIME=  F7 3//)
C
       CALL LXYPLT(4 20,200 ,-200  200   200  15 0 -15 0 40  -40
     X RREF(5),Y(5) Y(1) RCONT 0 05)
C
       CALL GNDAV2     ' INITIALIZE MACHINE EPSILON
C
       Y(5)=0 0        ' INITIALIZE REFERENCE INPUT
C
C
C   ************************************************************
C
C
10     CONTINUE        '  START  LOOP  HERE
C
C   STORE STATES AT START OF INTEGRATION INTERVAL
C
       DO 20 I=1,N+1
        STRTY(I)=Y(I)
20     CONTINUE
C
C
C   INTEGRATE THE NON-LINEAR EQUATIONS (PLANT) FROM BEGX TO ENDX
C     AND PLOT RESULTS
C
C     LINEAR=TRUE FOR UPDATE MODEL,=FALSE FOR NONLINEAR   PLANT
C
       LINEAR= FALSE        '  REAL  PLANT
       LPLOT= TRUE          ' PLOT RESULTS
C
       CALL RK41            ' INTEGRATION ROUTINE
C
       LPLOT= FALSE      ' SUPPRESS PLOT FOR  CONTROL GENARATION
C
C  ************************************************************
C
```

04/11/84 16 57 22 TASK # 16000084 AMOS GOULD S E L MPX-3.

```
C
C
C        STORE LAST VALUES OF STATES AND REFERENCES
C
C
      RBEGX=BEGX
      DELTA=ENDX-RBEGX
C
      DO 30 I=1,N+1
         ENDY(I)=Y(I)
         RREFX(I)=RREF(I)
30    CONTINUE
C
C     STORE STATES,REFERENCES LAST 0 2*DELTA FOR IDENTIFICATION
C
C
      JL=JA
      DO 50 I=1,N +1
         DO 50 JK=1,JL/5
          RYXACT(I,JK)=YX(I,JL-JL/5+JK-1)
          RREFC(I,JK)=RREFI(I,JL-JL/5+JK)
50    CONTINUE
C
C
C     EVALUATE FIRST,SECOND ,THIRD ORDER DERIVATIVE
C     AT CONTROL POINT FOR REFERENCE INPUT EXTRAPOLATION
C
      RTANG(1)=(RREFI(5,JL-2)-4 0*RREFI(5,JL-1)+3 0*RREFI(5,JL))/2 0
      RTANG(2)=RREFI(5,JL)-2 0*RREFI(5,JL-1)+RREFI(5,JL-2)
      RTANG(3)=RREFI(5,JL)-3 0*RREFI(5,JL-1)+3 0*RREFI(5,JL-2)
     X                                       -RREFI(5,JL-3)
C
C
C ******************************************************
C
C                                    UPDATE SYSTEM PARAMETERS
C
C     RFM(  )-VARIABLES THAT MINIMIZE ERROR FUNCTION
C      RFM(1)-BULK MOD-FLUID,RFM(2)-VALVE GAIN
C
C
      REPSILON=0 1E-10   ' ACCURACY LIMIT
      NS=2  '              NO  OF VARIABLES FOR MINIMIZATION ROUTINE
      NITER=ITERI     '    MAX NO  OF ITERATIONS FOR IDENTIFICATION
      LPRINT= FALSE
      LSYS= TRUE         ' IDENTIFICATION  FLAG
C
      LINEAR= TRUE      ' ADAPTIVE MODEL
C
       DO 41 J=1,NS
        RKFM(J)=RFM(J)
41    CONTINUE
C
C
       CALL GNDAV1(RFM,NS)       ' OPTIMIZATION SUBROUTINE
C
C
C
C     RETURNED VALUE WILL BE UPDATED MODEL
C
C*********************************************************
C
C
C     CONTROL CALCULATION
C
C
C
       NITER=ITERC     ' MAX NO  OF ITERATIONS FOR CONTROL
C
C
C      OPTIMIZE CONTROL BASED ON UPDATED MODEL
C                  (CALCULATE NEW GAINS)
C
       LSYS= FALSE       ' CONTROL FLAG
C
C
       DO 31 J=1,4
        RRKI(J)=RK(J)
31    CONTINUE
C
C
C
       CALL GNDAV1(RK,N)     ' OPTIMIZATION SUBROUTINE
C
C
       DO 32 J=1,N
        RK(J)=RRKI(J)+ 15*(RK(J) -RRKI(J))
32    CONTINUE
C
C     RETURNED VALUES WILL BE OF UPDATED GAINS
C
C
C
C
C *********************************************
C
C
C            PREPARE FOR NEW CYCLE
C
       BEGX=RBEGX+DELTA
       ENDX=BEGX+DELTA
       DO 40 I=1,N+1
        Y(I)=ENDY(I)
        RREF(I)=RREFX(I)
40    CONTINUE
```

```
C                                                                       0226 000
      IF(X LT ENDTIME)GO TO 10          ' ELSE STOP SIMULATION          0227 000
C                                                                       0228 000
C           PRINT VALUES OF GAINS,PARAMETERS AND REFERENCES             0229 000
C           AT TERMINATION POINT                                        0230 000
C                                                                       0231 000
C                                                                       0232 000
      WRITE(1,901)(RK(I),I=1,4)                                         0233 000
901   FORMAT('0',5X,'K= ',4(E12 5,5X))                                  0234 000
      WRITE(1,902)(RFM(I),I=1,2)                                        0235 000
902   FORMAT(5X,'RFM= ',2(E12 5,5X))                                    0236 000
       WRITE(1,903)(RREFX(I),I=1,5)                                     0237 000
903    FORMAT(5X,'RREFX=  ,5(E12 5,5X))                                 0238 000
C                                                                       0239 000
       STOP                                                             0240 000
       END                                                              0241 000
$A1 LIB=HYDROLIB,,U                                                     0242 000
$A1 DIR=HYDRODIR,,U                                                     0243 000
$CATALOG                                                                0244 000
A1 5=D CONT2                                                            0245 000
A2 1=SLO,1000                                                           0246 000
A2 LP=SLO,20000                                                         0247 000
BUILD R NLCN12                                                          0248 000
$EOJ                                                                    0249 000
$$                                                                      0250 000
```

```
$JOB B RK42 AMOS SLOF=L RK42                                          0001 000
$OPTION 2 3 4 5                                                       0002 000
$FORT77                                                               0003 000
C                                                                     0004 000
C                                                                     0005 000
C                                                                     0006 000
C                                                                     0007 000
      SUBROUTINE RK41                                                 0008 000
C                                                                     0009 000
C                                                                     0010 000
C                                                                     0011 000
C                                                                     0012 000
C     PURPOSE OF THE SUBROUTINE IS TO GENERATE INPUT SIGNAL TO THE    0013 000
C     SYSTEM ,CALCULATE CONTROL AND INTEGRATE THE NON-LINEAR          0014 000
C     EQUATIONS OF MOTION                                             0015 000
C        INPUT IS GENERATED FOR TWO MODES  FOLLOWER MODE AND WASHOUT  0016 000
C     MODE   INTEGRATION  IS DONE BY FORTH ORDER RUNGE-KUTTA METHOD   0017 000
C       THIS SUBROUTINE IS CALLED BY MAIN PROGRAM(NLCONT) AND BY      0018 000
C     THE INDEX FUNCTION SUBROUTINES(FUNCT1,FUNCT2)                   0019 000
C       SUBROUTINES CALLED BY RK41 ARE THE SYSTEM EQNS  OF MOTION     0020 000
C     SUBROUTINES-DERFUN1 FOR UPDATE MODEL, DERFUN2 FOR PLANT MODEL   0021 000
C                                                                     0022 000
C                                                                     0023 000
C                                                                     0024 000
C                                                                     0025 000
C     DECLARE COMMONS                                                 0026 000
C                                                                     0027 000
      INCLUDE C CONTR2                                                0028 000
C                                                                     0029 000
C      DECLARE LOCALS                                                 0030 000
C                                                                     0031 000
       REAL*4  RJA       ' REAL(JA-1)                                0032 000
       REAL*4 RXLIM      ' LIMIT FOR TRAVEL                          0033 000
       REAL*4 RY1LIM     ' ABS (RXLIM-Y(1))                          0034 000
       REAL*4 RY2LIM     ' VEL LIMIT FOR BRAKE                       0035 000
       REAL*4    WK(16 ) ' WORK VECTOR                               0036 000
       INTEGER*1  IFUN   ' CONTROL FOR PLOT                          0037 000
       INTEGER*2    IC   ' CONTROL FOR PLOT SPACING                  0038 000
       LOGICAL*1 LBRAKF  ' BRAKE IN EFFECT FLAG                      0039 000
       LOGICAL*1  LREV   ' MOTION EXCEEDS LIMIT                      0040 000
       LOGICAL*1 LSYS    ' IDENTIFICATION FLAG                       0041 000
C                                                                     0042 000
C                                                                     0043 000
       COMMON/GN/LSYS                                                 0044 000
C                                                                     0045 000
C                                                                     0046 000
C                                                                     0047 000
C          INITIALIZATION                                             0048 000
C                                                                     0049 000
C                                                                     0050 000
C                                                                     0051 000
      JA=1                                                            0052 000
      IC=0                                                            0053 000
      RXLIM=10 0                                                      0054 000
      IFUN =0                                                         0055 000
      X=BEGX                                                          0056 000
```



```
      I1=2*N                                                          0057 000
      I2=3*N                                                          0058 000
C                                                                     0059 000
C                                                                     0060 000
C*****************************************************              0061 000
C                                                                     0062 000
C        START LOOP HERE                                              0063 000
C                                                                     0064 000
C                                                                     0065 000
10    CONTINUE                                                        0066 000
C                                                                     0067 000
      IF( NOT LINEAR)THEN           ' OPERATION IS ON PLANT MODEL     0068 000
C                                                                     0069 000
C                                                                     0070 000
C         CHECK PLANT  MEASUREMENTS  AND DETERMINF IF NFED TO         0071 000
C         SWITCH BETWEEN TWO MODES OF OPERATION                       0072 000
C                                                                     0073 000
C                                                                     0074 000
C                                                                     0075 000
       IF(ABS(Y(1)) LT 0 2)THEN                                       0076 000
         LBRAKE= FALSE                                                0077 000
         LREV= FALSE                                                  0078 000
       ELSE IF((Y(2) GT 0 0) AND (Y(1) GT 0 0))THEN                   0079 000
         RY1LIM=ABS(RXLIM-Y(1))                                       0080 000
         RY2LIM=SQRT(2 0*38 0*RY1LIM)                                 0081 000
         IF(Y(2) GT RY2LIM)LBRAKE= TRUE    'VELOCITY EXCEEDS LIMIT    0082 000
       ELSE IF((Y(2) LT 0 0) AND (Y(1) LT 0 0))THEN                   0083 000
         RY1LIM=ABS(RXLIM+Y(1))                                       0084 000
         RY2LIM=SQRT(2 0*38 0*RY1LIM)                                 0085 000
         IF(ABS(Y(2)) GT RY2LIM)LBRAKE= TRUE  'VELOC EXCEEDS LIMIT    0086 000
C                                                                     0087 000
C                                                                     0088 000
       END IF                                                         0089 000
C                                                                     0090 000
C                                                                     0091 000
        RXR(JA)=X                    ' STORE GREED TIME               0092 000
C                                                                     0093 000
C                                                                     0094 000
C      ****INPUT SIGNAL GENERATION***********                         0095 000
C                                                                     0096 000
C        INPUT SIGNAL HERE WILL BE ACCELARATION                       0097 000
C                                                                     0098 000
C                                                                     0099 000
C                                                                     0100 000
C       GENERATION OF INPUT SIGNAL IN BRAKE/WASHOUT MODE              0101 000
C                                                                     0102 000
C                                                                     0103 000
C                                                                     0104 000
       IF(LBRAKE)THEN                                                 0105 000
         RRR=-SIGN(38 0,Y(1))                                         0106 000
         LREV=LREV OR (ABS(Y(1)) GE RXLIM)                            0107 000
C                                                                     0108 000
          IF(LREV)THEN                                                0109 000
            RY1LIM=ABS(Y(1))                                          0110 000
            RY2LIM=SQRT(2 0*38 0*RY1LIM)                              0111 000
C                                                                     0112 000
```

```
            IF((Y(1)*Y(2) LT 0 0) AND (ABS(Y(2)) GT 0 5*RY2LIM))THEN  0113 000
               IF(RREF(5)*RRR GT 0 0)THEN                             0114 000
              RREF(5)=RREF(5)+ 05*SIGN(ABS(Y(2))-0 5*RY2LIM, Y(1))    0115 000
               ELSE                                                   0116 000
                RREF(5)=RREF(5)- 02*(RRR+RREF(5))                     0117 000
               END IF                                                 0118 000
C                                                                     0119 000
C                                                                     0120 000
C                                                                     0121 000
            ELSE                                                      0122 000
C                                                                     0123 000
              RREF(5)=RREF(5)+ 01*(RRR-RREF(5))                       0124 000
            END IF                                                    0125 000
C                                                                     0126 000
         ELSE                                                         0127 000
           RY1LIM=RXLIM-ABS(Y(1))                                     0128 000
           RY2LIM=SQRT(2 0*38 0*RY1LIM)                               0129 000
C                                                                     0130 000
         IF(RRR*RREF(5) LT 0 0)THEN                                   0131 000
           RREF(5)=RREF(5)- 10*SIGN(ABS(Y(2))-RY2LIM, Y(1))           0132 000
         ELSE                                                         0133 000
          RREF(5)=RREF(5)+ 01*(RRR-RREF(5))                           0134 000
         END IF                                                       0135 000
C                                                                     0136 000
         END IF                                                       0137 000
       ELSE                                                           0138 000
C                                                                     0139 000
C                                                                     0140 000
C                                                                     0141 000
C                                                                     0142 000
C            GENERATE INPUT SIGNAL FOR THE FOLLOWER MODE TO           0143 000
C            SIMULATE INPUT FROM  AIRCRAFT EQUATIONS OF MOTION        0144 000
C                                                                     0145 000
C                                                                     0146 000
C                                                                     0147 000
C        RREF(5)=120 0*SIN(10 0*X)   ' HIGH FREQUENCY INPUT           0148 000
C       RREF(5)=22 0*SIN(3 0*X)      ' LOW FREQUENCY INPUT            0149 000
          RREF(5)=80 0*X            ' RAMP TO CHECK WASHOUT           0150 000
C                                                                     0151 000
       END IF                                                         0152 000
C                                                                     0153 000
C                                                                     0154 000
     ELSE IF(LSYS)THEN     ' *********IDENTIFICATION********         0155 000
C                                                                     0156 000
C                                                                     0157 000
        RREF(5)=RREFC(5, JA)      ' FROM STORAGE                      0158 000
C                                                                     0159 000
C                                                                     0160 000
     ELSE        '*******CONTROL REFERENCE*********                   0161 000
C                                                                     0162 000
C                                                                     0163 000
C       EXTRAPOLATE SIGNAL INPUT FOR CONTROL CALCULATIONS             0164 000
C                                                                     0165 000
      RJA=JA-1                                                        0166 000
      RREF(5)=RREFX(5)+RTANG(1)*RJA+ 5*(RJA**2)*RTANG(2)              0167 000
     X                + 166667*(RJA**3)*RTANG(3)                     0168 000
```



```
C                                                                     0169 000
      END IF       ' SIGNAL INPUT ESTABLISHED                         0170 000
C                                                                     0171 000
C                                                                     0172 000
C      NOW CONVERT ACCELERATION INPUT                                 0173 000
C        TO PRESSURE VALUES                                           0174 000
C                                                                     0175 000
C                                                                     0176 000
C                                                                     0177 000
        RREF(3)=1000 0+ 5*RREF(5)/ 926                                0178 000
        RREF(4)=1000 0- 5*RREF(5)/ 926                                0179 000
C                                                                     0180 000
C                                                                     0181 000
          DO 11 I=1, N +1       ' RCORD REFERENCE ON GREED            0182 000
           RREFI(I, JA)=RREF(I)                                       0183 000
11      CONTINUE                                                      0184 000
C                                                                     0185 000
C                                                                     0186 000
C ******************************                                    0187 000
C                                                                     0188 000
C                                                                     0189 000
       RCONT=0 0                ' CONTROL CALCULATIONS                0190 000
      DO 15 I=1, N                                                    0191 000
      RCONT=RCONT+RK(I)*(RREF(I)-Y(I))                                0192 000
15    CONTINUE                                                        0193 000
C                                                                     0194 000
       IF(ABS(RCONT) GE 40 0)RCONT=SIGN(40 0 RCONT)                   0195 000
C                                                                     0196 000
C                                                                     0197 000
C************************************************                   0198 000
C                                                                     0199 000
C     INTEGRATION STARTS HERE                                         0200 000
C                                                                     0201 000
C                                                                     0202 000
C                                                                     0203 000
C                              ******** EVALUATE DERIVATIVE(V1)       0204 000
       IF(LINEAR)THEN                                                 0205 000
          CALL DERFUN1(RCONT, N, Y, DY, RFM)    ' UPDATED MODEL       0206 000
        ELSE                                                          0207 000
          CALL DERFUN2(RCONT N, Y, DY RFM)    '  PLANT  MODEL        0208 000
        END IF                                                        0209 000
C                                                                     0210 000
      DO 20 I=1, N                                                    0211 000
      WK(I)=Y(I)+DELX*DY(I)/2 0E00                                    0212 000
      WK(I+N)=DY(I)                    ' STORE V1                     0213 000
20    CONTINUE                                                        0214 000
C                                                                     0215 000
C                                                                     0216 000
C                                                                     0217 000
      X=X+DELX/2 0E00                                                 0218 000
C     ********************************* EVALUATE V2                   0219 000
       IF(LINEAR)THEN                                                 0220 000
          CALL DERFUN1(RCONT, N, WK, DY, RFM)                         0221 000
        ELSE                                                          0222 000
          CALL DERFUN2(RCONT, N WK, DY RFM)                           0223 000
        END IF                                                        0224 000
```

```
C                                                                       0225 000
C                                                                       0226 000
      DO 30 I=1,N                                                       0227 000
      WK(I)=Y(I)+DELX*DY(I)/2 0E00                                      0228 000
      WK(I+I1)=DY(I)                    ' STORE V2                      0229 000
30    CONTINUE                                                          0230 000
C                                                                       0231 000
C                                                                       0232 000
C     ********************************** COMPUTE V3                     0233 000
       IF(LINEAR)THEN                                                   0234 000
          CALL DERFUN1(RCONT,N,WK,DY,RFM)                               0235 000
        ELSE                                                            0236 000
          CALL DERFUN2(RCONT,N,WK,DY,RFM)                               0237 000
        END IF                                                          0238 000
C                                                                       0239 000
      DO 40 I=1,N                                                       0240 000
      WK(I)=Y(I)+DELX*DY(I)                                             0241 000
      WK(I+I2)=DY(I)            ' STORE V3                              0242 000
40    CONTINUE                                                          0243 000
C                                                                       0244 000
C                                                                       0245 000
      X=X+DELX/2 0E00                                                   0246 000
C                                                                       0247 000
C     ********************************** COMPUTE V4                     0248 000
       IF(LINEAR)THEN                                                   0249 000
          CALL DERFUN1(RCONT,N,WK,DY,RFM)                               0250 000
        ELSE                                                            0251 000
          CALL DERFUN2(RCONT,N,WK,DY,RFM)                               0252 000
        END IF                                                          0253 000
C                                                                       0254 000
C                                                                       0255 000
C      COMPUTE Y(K+1)   ******************                             0256 000
C                                                                       0257 000
      DO 50 I=1,N                                                       0258 000
      Y(I)=Y(I)+(WK(I+N)+2 0E00*WK(I+I1)+2 0E00*WK(I+I2)                0259 000
     X +DY(I))*DELX/6 0E00                                              0260 000
50    CONTINUE                        -                               0261 000
C                                                                       0262 000
C                                                                       0263 000
C                                                                       0264 000
C                                                                       0265 000
C     NEW VALUES FOR STATES WERE ESTABLISHED                            0266 000
C                                                                       0267 000
C                                                                       0268 000
C                                                                       0269 000
      IF(Y(3) LE 0 0)Y(3)=0 0                                           0270 000
      IF(Y(4) LE 0 0)Y(4)=0 0   ' LIMIT PRESSURES TO 0 PSI              0271 000
        Y(5)=DY(2)     '  ACCELARATION OUTPUT OF THE MASS               0272 000
C                                                                       0273 000
C                                                                       0274 000
C   ****************************************                            0275 000
C                                                                       0276 000
C                                                                       0277 000
C          PLOT RESULTS OF SIMULATION                                   0278 000
C                                                                       0279 000
C                                                                       0280 000
```



```
        IF(LPLOT)THEN                                                   0281 000
              IF(IC EQ 0)THEN                                           0282 000
          CALL LXYPLT(IFUN,20,200 ,-200 ,200 ,-200 ,15 0,-15 0          0283 000
       X 40 0,-40 0,RREF(5),Y(5),Y(1),RCONT,0 05)                       0284 000
           IC=4                                                         0285 000
          ELSE                                                          0286 000
            IC=IC-1                                                     0287 000
          END IF                                                        0288 000
C                                                                       0289 000
        END IF                                                          0290 000
C                                                                       0291 000
C  ************************************************                     0292 000
C                                                                       0293 000
         DO 60 I=1,N+1          ' RECORD STATES VALUES                  0294 000
           YX(I,JA)=Y(I)                                                0295 000
60       CONTINUE                                                       0296 000
C                                                                       0297 000
        IF(X LT ENDX)THEN       ' INTEGRATION INTERVAL ?                0298 000
         JA=JA+1                                                        0299 000
          GO TO 10                                                      0300 000
        END IF                                                          0301 000
C                                                                       0302 000
C                                                                       0303 000
        RETURN                                                          0304 000
        END                                                             0305 000
$A1 LIB=HYDROLIB,,U                                                     0306 000
$A1 DIR=HYDRODIR,,U                                                     0307 000
$LIBED                                                                  0308 000
$EOJ                                                                    0309 000
$$                                                                      0310 000
```

```
$JOB B GNDAV1 AMOS SLOF=L GNDAV1
$OPTION 2 3 4 5
$FORT77
C
C   SUBROUTINE TO FIND  MINIMUM OF NONLINEAR FUNCTION
C   OF N VARIABLES EMPLOYING QUASI-NEWTON METHOD OF
C   W C DAVIDON(MATH PROGRAMMING,9(1975)1-30)
C     WITHOUT LINE SEARCH
       SUBROUTINE GNDAV1 (RVX,N) ' CONTROL VERSION
C
      REAL*4    RMJ(20,20)
      REAL*4    RVX(20)
      REAL*4    RVXO(20)
      REAL*4    RF              ' FUNCTION VALUE
      REAL*4    RFO             ' F(0)
      REAL*4    RVG(20)         ' GRADIENT VECTOR
      REAL*4    RVKO(20)
      REAL*4    RVW(20)
      REAL*4    RVS(20)         ' DIRECTION VECTOR
      REAL*4    RFO1
      REAL*4    RLAMDA
      REAL*4    REPSILON
      REAL*4    RVK(20)
      REAL*4    RF1
      REAL*4    RBO
      REAL*4    RVM(20)
      REAL*4    RFM
      REAL*4    RFMU
      REAL*4    RFNU
      REAL*4    RVU(20)
      REAL*4    RVN(20)
      REAL*4    RFN
      REAL*4    RFB
      REAL*4    RFGAMA
      REAL*4    RFDELT
      REAL*4    RFA
      REAL*4    RFC
      REAL*4    RMJT(20,20)
      REAL*4    RTMP
      REAL*4    RFALFA
      REAL*4    RVP(20)
      REAL*4    RVQ(20)
      REAL*4    X
      REAL*4    B
      REAL*4    A
      REAL*4    EPS             ' MACHINE EPSILON
      REAL*4    ESP             ' SQRT(MACH EPS)
      REAL*4    RMTU            ' 10-6(MU)2
      REAL*4    RMUTU
      REAL*4    UTS
      REAL*4    UTU
      REAL*4    GTKO
      REAL*4    SMGK(20,20)
      REAL*4    D
      INTEGER*2   IX            ' CURRENT ITERATION #
      INTEGER*2   NITER         ' MAX # OF ITERATIONS
      INTEGER*2   N             ' #OF VARIABLES
      LOGICAL*1   LPRINT        ' TO PRINT EVERY ITERATION
      LOGICAL*1   LSYS
       COMMON/GN/     REPSILON,   NITER,LPRINT,LSYS
C
C*******************************************************
C
C*************************************************************
C
C         INITIALIZATION
C
C                                       FUNCTION VAL FOR START
      IF(LSYS)THEN
        CALL FUNCT1(RF)       ' IDENTIFICATION
      ELSE
        CALL FUNCT2(RF)       ' FFEDBACK
      END IF
C
C       J=[I]
C
      DO 50 I=1,N
      DO 50 J=1,N
C
      IF(I NE J)THEN
        RMJ(I,J)=0 0
      ELSE
        RMJ(I,J)=1 0
      END IF
50    CONTINUE
C
C
C       GRADIENT
C
      CALL GRAD1(RVX,N,RVG,ESP,LSYS)
C
C
      RFO=RF
C
      DO 60 I=1,N
       RVKO(I)=0 0
      DO 60 J=1,N
       RVKO(I)=RVKO(I)+RMJ(I,J)*RVG(J)
 60   CONTINUE
C
C
      DO 70 I=1,N
        RVXO(I)=RVX(I)
        RVW(I)=RVKO(I)
70    CONTINUE
C
C       PRINT STARTING VALUES
C
C     WRITE(1,1010)
C010   FORMAT(10X,'INITIAL X = )
C     WRITE(1,1012)(RVX(I) I=1 N)
```

```
CO12  FORMAT(4X,4D18 10)                                                0126 000
C     WRITE(1,1015)RF                                                   0127 000
CO15  FORMAT(6X, INITIAL F VALUE = ',D18 10)                            0128 000
C     WRITE(1,1020)                                                     0129 000
CO20  FORMAT(5X,'J-MATRIX,INITIALIZED =  )                              0130 000
C     WRITE(1,1012)((RMJ(I J),J=1,N),I=1,N)                             0131 000
C                                                                       0132 000
C     INITIALIZE # OF ITERATIONS                                        0133 000
C                                                                       0134 000
      IX=0                                                              0135 000
C                                                                       0136 000
C*********************************************************            0137 000
C                                                                       0138 000
C   STEP NO 1                                                           0139 000
C                                                                       0140 000
1      CONTINUE                                                         0141 000
C                                                                       0142 000
C                                                                       0143 000
       RFO1=0 0                                                         0144 000
       DO 80 I=1,N                                                      0145 000
        RVS(I)=-RVKO(I)                                                 0146 000
        RFO1=RFO1+RVKO(I)*RVS(I)                                        0147 000
80     CONTINUE                                                         0148 000
C                                                                       0149 000
      RLAMDA=2 0                                                        0150 000
C                                                                       0151 000
      IF(4 0*RFO LT -RFO1)THEN                                          0152 000
C                                                                       0153 000
         DO 90 I=1,N                                                    0154 000
          RVS(I)=-RVS(I)*4 0*RFO/RFO1                                   0155 000
90       CONTINUE                                                       0156 000
C                                                                       0157 000
         RFO1=-4 0*RFO                                                  0158 000
      END IF                                                            0159 000
C                                                                       0160 000
C*************************************************************        0161 000
C                                                                       0162 000
C       STEP NO  2                                                      0163 000
C                                                                       0164 000
2      CONTINUE                                                         0165 000
C                                                                       0166 000
C    UPDATE X-VALUES                                                    0167 000
C                                                                       0168 000
      DO 100 I=1,N                                                      0169 000
       RVX(I)=RVXO(I)                                                   0170 000
C                                                                       0171 000
      DO 100 J=1,N                                                      0172 000
C                                                                       0173 000
       RVX(I)=RVX(I)+RMJ(I,J)*RVS(J)                                    0174 000
100    CONTINUE                                                         0175 000
C                                                                       0176 000
C                                                                       0177 000
      IF(-RFO1 LE REPSILON)GO TO 999       ' EXIT **************       0178 000
C                                                                       0179 000
      IF(IX GE NITER)GO TO 999                                          0180 000
      IX=IX+1                                                           0181 000
```

04/11/84 16 59 16 TASK # 16000084 AMOS GOULD S E L MPX-3

```
C     CALL FUNCT(RF RVX,N)                                              0182 000
      IF(LSYS)THEN                                                      0183 000
        CALL FUNCT1(RF)      ' IDENTIFICATION                           0184 000
      ELSE                                                              0185 000
        CALL FUNCT2(RF)      ' FFEDBACK                                 0186 000
      END IF                                                            0187 000
C                                                                       0188 000
C        PRINT NEW VALUES FOR ITERATION POINT                           0189 000
C                                                                       0190 000
C     IF(LPRINT)THEN                                                    0191 000
C      WRITE(1,1050)IX,RF                                               0192 000
CO50   FORMAT(//,5X, ITERATION =  ,I3 3X, F= ' D18 10)                  0193 000
C      WRITE(1 1060)                                                    0194 000
CO60   FORMAT(/ 5X,'X-VALUES=  )                                        0195 000
C      WRITE(1 1012)(RVX(I),I-1,N)                                      0196 000
C                                                                       0197 000
C      WRITE(1,1070)                                                    0198 000
CO70   FORMAT(/,5X,  J-MTRIX =  )                                       0199 000
C      WRITE(1,1012)((RMJ(I J) J-1 N) I-1 N)                            0200 000
C                                                                       0201 000
C     END IF                                                            0202 000
C                                                                       0203 000
      IF(RF GE RFO)THEN                                                 0204 000
C                                                                       0205 000
        DO 120 I=1,N                                                    0206 000
          RVS(I)=0 5*RVS(I)                                             0207 000
120     CONTINUE                                                        0208 000
C                                                                       0209 000
        RFO1=0 5*RFO1                                                   0210 000
        RLAMDA=0 5                                                      0211 000
        GO TO 2                                                         0212 000
      END IF                                                            0213 000
C                                                                       0214 000
C*********************************************************            0215 000
C                                                                       0216 000
C      STEP NO  3                                                       0217 000
C                                                                       0218 000
3      CONTINUE                                                         0219 000
C                                                                       0220 000
      CALL GRAD1(RVX,N,RVG,ESP LSYS)                                    0221 000
C                                                                       0222 000
C    J-TRANSPOSE                                                        0223 000
C                                                                       0224 000
      DO 121 I=1,N                                                      0225 000
      DO 121 J=1,N                                                      0226 000
121   RMJT(I,J)=RMJ(J,I)                                                0227 000
C                                                                       0228 000
C          K-VECTOR                                                     0229 000
C                                                                       0230 000
      DO 125 I=1 N                                                      0231 000
       RVK(I)=0 0                                                       0232 000
      DO 125 J=1,N                                                      0233 000
       RVK(I)=RVK(I)+RMJT(I J)*RVG(J)                                   0234 000
125   CONTINUE                                                          0235 000
C                                                                       0236 000
C           F                                                           0237 000
```

```
C
      RF1=0 0
      DO 130 I=1,N
       RF1=RF1+RVK(I)*RVS(I)
130   CONTINUE
C
C
C        B(0)
C
      RB0=RF1-RF01
C
        '  CALC  M,X,K0
C
       DO 140 I=1,N
        RVM(I)=RVS(I)+RVK0(I)-RVK(I)
        RVX0(I)=RVX(I)
        RVK0(I)=RVK(I)
140    CONTINUE
C
C
       RF0=RF
       RF01=RF1
C
       IF(RB0 LT REPSILON)THEN
C
         DO 150 I=1,N
          RVS(I)=RVS(I)*RLAMDA
150      CONTINUE
         RF01=RF01*RLAMDA
         GO TO 2
       END IF
C
C*************************************************************
C
C              STEP NO  4
C
4      CONTINUE
C
      RFM=0 0
      DO 155 I=1,N
       RFM=RFM+RVM(I)*RVM(I)
155   CONTINUE
C
      IF(RFM LE REPSILON)GO TO 1
C
      RFMU=0 0
      DO 160 I=1,N
       RFMU=RFMU+RVM(I)*RVS(I)
160    CONTINUE
C
      RFNU=RFMU-RFM
C
C
C       CALC U
C
       RTMP=0 0
       DO 170 I=1,N
        RTMP=RTMP+RVM(I)*RVW(I)/RFM
170    CONTINUE
C
C
       DO 180 I=1 N
        RVU(I)=RVW(I)-RVM(I)*RTMP
180    CONTINUE
C
C
      RMTU=0 0
      RMUTU=0 0
C
       DO 190 I=1 N
        RMTU=RMTU+RVM(I)*RVU(I)
        RMUTU=RMUTU+RVU(I)*RVU(I)
190   CONTINUE
C
      RMTU=1 0E06*RMTU**2
      RMUTU=RFM*RMUTU
C
C
       IF(RMTU LT RMUTU)THEN      ' CAL  VECTOR N STEP 4A
C
        UTS=0 0
        UTU=0 0
C
        DO 200 I=1,N
         UTS=UTS+RVU(I)*RVS(I)
         UTU=UTU+RVU(I)*RVU(I)
200    CONTINUE
C
        DO 210 I=1,N
         RVN(I)=RVU(I)*UTS/UTU
210    CONTINUE
C
        RFN=UTS**2/UTU
C
      ELSE
C
        DO 220 I=1,N
         RVN(I)=0 0
220    CONTINUE
       RFN=0 0
C
      END IF
C
C*************************************************************
C
C           STEP NO  5
C
C
5      CONTINUE
C
      RFB=RFN+RFNU*RFMU/RFM
C
```

04/11/84 16 59 16 TASK # 16000084 AMOS GOULD S E L MPX-3

```
      IF(RFB LT REPSILON)THEN                                          0350 000
C                                                                      0351 000
        DO 230 I=1,N                                                   0352 000
C                                                                      0353 000
230     RVN(I)=RVS(I)-RVM(I)*RFMU/RFM                                  0354 000
        RFN=RBO-RFNU*RFMU/RFM                                          0355 000
        RFB=RBO                                                        0356 000
      END IF                                                           0357 000
C                                                                      0358 000
C*****************************************************************   0359 000
C                                                                      0360 000
C        STEP NO  6                                                    0361 000
C                                                                      0362 000
6       CONTINUE                                                       0363 000
C                                                                      0364 000
      IF(RFNU*RFMU LT RFM*RFN)THEN                                     0365 000
C                                                                      0366 000
        RFA=RFB-RFNU                                                   0367 000
        RFC=RFB+RFMU                                                   0368 000
C                                                                      0369 000
       RFGAMA= SQRT( (1 0-RFMU*RFNU/(RFN*RFM))/(RFA*RFB))              0370 000
C                                                                      0371 000
       RFDELT= SQRT(RFC/RFA)                                           0372 000
C                                                                      0373 000
        IF(RFC LT RFA)RFGAMA=-RFGAMA                                   0374 000
      ELSE                                                             0375 000
       RFGAMA=0 0                                                      0376 000
       RFDELT= SQRT(RFMU/RFNU)                                         0377 000
      END IF                                                           0378 000
C                                                                      0379 000
C                                                                      0380 000
C*****************************************************************   0381 000
C                                                                      0382 000
C           STEP NO  7                                                 0383 000
C                                                                      0384 000
       RFALFA=RFMU+RFNU*RFDELT+RFM*RFN*RFGAMA                          0385 000
C                                                                      0386 000
C         P - VECTOR                                                   0387 000
C                                                                      0388 000
      DO 240 I=1,N                                                     0389 000
       RVP(I)=RVM(I)*(RFDELT-RFN*RFGAMA)+RVN(I)*RFGAMA*RFMU            0390 000
C                                                                      0391 000
C        G-VECTOR                                                      0392 000
C                                                                      0393 000
      RVG(I)=RVM(I)*(1 0+RFN*RFGAMA)/RFALFA                            0394 000
     X       -RVN(I)*RFGAMA*RFNU/RFALFA                                0395 000
C                                                                      0396 000
C       W-VECTOR                                                       0397 000
C                                                                      0398 000
       RVW(I)=RVM(I)*RFN*(1 0+RFGAMA*RFNU*RFMU/RFALFA)                 0399 000
     X        -RVN(I)*(1 0+RFDELT)*RFNU*RFMU/RFALFA                   0400 000
C                                                                      0401 000
240   CONTINUE                                                         0402 000
C                                                                      0403 000
C                                                                      0404 000
C      UPDATE VECTOR K(0)                                              0405 000
```



```
C                                                                      0406 000
      GTKO=0 0                                                         0407 000
      DO 250 I=1,N                                                     0408 000
250    GTKO=GTKO+RVG(I)*RVKO(I)                                        0409 000
C                                                                      0410 000
      DO 260 I=1 N                                                     0411 000
        RVKO(I)=RVKO(I)+RVP(I)*GTKO                                    0412 000
260   CONTINUE                                                         0413 000
C                                                                      0414 000
C                                                                      0415 000
C      UPDATE J-MATRIX                                                 0416 000
C                                                                      0417 000
      DO 270 I=1 N                                                     0418 000
      DO 270 J=1 N                                                     0419 000
       SMGK(I J)=RVG(I)*RVP(J)                                         0420 000
270   CONTINUE                                                         0421 000
C                                                                      0422 000
C                                                                      0423 000
      DO 300 I=1,N                                                     0424 000
      DO 300 J=1,N                                                     0425 000
       D=RMJ(I,J)                                                      0426 000
      DO 290 K=1 N                                                     0427 000
290    D=D+RMJ(I,K)*SMGK(K J)                                          0428 000
       RMJ(I,J)=D                                                      0429 000
300   CONTINUE                                                         0430 000
C                                                                      0431 000
C                                                                      0432 000
      IF(RFN LE 0 ODO)THEN                                             0433 000
C                                                                      0434 000
        DO 310 I=1,N                                                   0435 000
         RVW(I)=RVKO(I)                                                0436 000
310     CONTINUE                                                       0437 000
      END IF                                                           0438 000
C                                                                      0439 000
      GO TO 1       '  LOOP BACK----------                             0440 000
C                                                                      0441 000
999   CONTINUE     ' *********** EXIT ************                      0442 000
C                                                                      0443 000
C     WRITE(1,1100)IX,RF                                               0444 000
C                                                                      0445 000
C100  FORMAT(/ 5X, ROUTINE TERMINATED ITER #=   I3 3X  F =  D18 10)    0446 000
C     WRITE(1,1060)                                                    0447 000
C     WRITE(1,1012)(RVX(I) I=1 N)                                      0448 000
C                                                                      0449 000
C                                                                      0450 000
      RETURN                                                           0451 000
       ENTRY GNDAV2                                                    0451 010
C                                                                      0451 020
C                                                                      0451 030
C       FIND MACHINE EPSILON                                           0451 040
C                                                                      0451 050
C                                                                      0451 060
      X=1 0                                                            0451 070
10    X=X*0 5                                                          0451 080
      A=1 0+X                                                          0451 090
      B=A-1 0                                                          0451 100
```

```
      IF(B NE 0 0)GO TO 10                                    0451 110
      EPS=X**2 0                                              0451 120
      ESP= SQRT(EPS)                                          0451 130
C                                                             0451 140
       RETURN                                                 0451 150
      END                                                     0452 000
$A1 LIB=HYDROLIB,,U                                           0453 000
$A1 DIR=HYDRODIR,,U                                           0454 000
$LIBED                                                        0455 000
$EOJ                                                          0456 000
$$                                                            0457 000
```

```
$JOB B FUNCT2 AMOS SLOF=L FUNCT2                                        0001 000
$OPTION 2 3 4 5                                                         0002 000
$FORT77                                                                 0003 000
C                                                                       0004 000
C                                                                       0005 000
C    FUNCT1-MINIMIZES ERROR TO ESTABLISH SYSTEM PARAMETERS              0006 000
C    FUNCT2-MINIMIZES ERROR TO ESTABLISH GAIN VALUES                    0007 000
C                                                                       0008 000
C                                                                       0009 000
C  ***********************************                                  0010 000
C                                                                       0011 000
      SUBROUTINE FUNCT1(RF)                                             0012 000
C                                                                       0013 000
C                                                                       0014 000
      INCLUDE C CONTR2                                                  0015 000
C                                                                       0016 000
      REAL*4     RF       ' RETURNED VALUE OF FUNCTION                 0017 000
C                                                                       0019 000
C                                                                       0019 100
C                                                                       0019 200
C                                                                       0019 300
       BEGX=RXR(JL-JL/5+1)                                              0020 000
      DO 5 I=1,N+1                                                      0021 000
       Y(I)=RYXACT(I,1)                                                 0022 000
5     CONTINUE                                                          0023 000
      LPLOT= FALSE                                                      0024 000
      CALL RK41        ' ******** INTEGRATION ROUTINE*******            0025 000
      RF=0 0                                                            0026 000
      DO 10 I=1,N                                                       0027 000
       DO 10 K=1,(JA-1)                                                 0028 000
       RF=RF+RQF1(I)*(YX(I,K)-RYXACT(I,K+1))**2                         0029 000
10    CONTINUE                                                          0030 000
C                                                                       0030 100
C                                                                       0030 200
C                                                                       0030 300
      RETURN                                                            0031 000
      END                                                               0032 000
C*********************************************************            0033 000
C                                                                       0034 000
C                                                                       0035 000
       SUBROUTINE FUNCT2(RF)                                            0063 000
C                                                                       0064 000
      INCLUDE C CONTR2                                                  0065 000
C                                                                       0066 000
C    LOCALS                                                             0067 000
      REAL*4  RF ' RETURNED VALUE OF FUNCTION                           0068 000
CC                                                                      0070 000
C                                                                       0071 000
       BEGX=RBEGX+DELTA                                                 0072 000
       ENDX=BEGX+0 2*DELTA                                              0073 000
C                                                                       0074 000
      DO 5 I=1,N+1                                                      0075 000
       Y(I)=ENDY(I)                                                     0076 000
5     CONTINUE                                                          0077 000
C                                                                       0078 000
      LPLOT= FALSE                                                      0080 000
```



```
      CALL RK41                                                         0081 000
C                                                                       0082 000
       RF=0 0                                                           0083 000
          DO 10 I=1,N                                                   0084 000
          DO 10 K=1,JA                                                  0085 000
C                                                                       0086 000
           RF=RF+RQF2(I)*(YX(I K)-RREFI(I,K))**2                        0087 000
10          CONTINUE                                                    0088 000
C                                                                       0088 100
C                                                                       0088 200
C                                                                       0088 300
       RETURN                                                           0089 000
       END                                                              0090 000
$A1 LIB=HYDROLIB,,U                                                     0091 000
$A1 DIR=HYDRODIR,,U                                                     0092 000
$LIBED                                                                  0093 000
$EOJ                                                                    0094 000
$$                                                                      0095 000
```

```
$JOB B DERFN2 AMOS SLOF=L DERFN2
$OPTION 2 3 4 5
$FORT77
C
C
C      TWO SUBROUTINES USED TO CALCULATE THE NONLINEAR STATE
C      EQUATIONS OF MOTION FOR ONE-DEGREE-OF-FREEDOM MOTION
C      SIMULATOR SUBROUTINE DERFUN2-'PLANT'MODEL,
C      SUBROUTINE DERFUN1-UPDATE MODEL
C
C
C
C   ************************************
C
C
          SUBROUTINE DERFUN2(RCONT,N,V DV,RFM)   ' 'PLANT' MODEL
C
C
C
C
C    DECLARE LOCALS
C
       INTEGER*2    N       ' NUMBER OF STATES
       REAL*4    RFM(2)     ' PARAMETERS
       REAL*4     RCONT     ' CONTROL VARIABLE MA
       REAL*4     V(4)      ' STATE VARIABLES
       REAL*4    DV(4)      ' STATE EQUATION
       REAL*4   BETA        ' BULK MODULUS OF FLUID
       REAL*4   XMAX        ' MAX PISTON TRAVEL
       REAL*4   A1          ' AREA #1, SQ INCH
       REAL*4   A2          ' AREA #2  SQ INCH
       REAL*4   RMASS       ' MASS,LB-SEC 2/INCH
       REAL*4   VOL1        ' #1 VOLUME,CU  INCH
       REAL*4   VOL2        ' #2 VOLUME CU INCH
       REAL*4   RQ1         ' #1 PORT FLOW,CU IN/SEC
       REAL*4   RQ2         ' #2 PORT FLOW,CU IN/SEC
       REAL*4   RPS         ' SUPPLY PRESSURE,PSI
C
C
C
       BETA=1 5E05
       XMAX=30 0
       A1=8 0
       A2=8 0
       RMASS=8 634
       RPS=2000 0
C
C********
C
C
C
       VOL1=A1*(XMAX+V(1))       ' VOLUME #1 OF ACTUATOR
       VOL2=A2*(XMAX-V(1))       ' VOLUME #2 OF ACTUATOR
C
C       FLOW CALCULATIONS
C
       RSIGN1=SIGN(1 0,RCONT)   ' SIGN OF CONTROL  VARIABLE
       RSQ1=RPS*0 5+(0 5*RPS-V(3))*RSIGN1
       RSQA1=ABS(RSQ1)
       RSQ2=0 5*RPS-(0 5*RPS-V(4))*RSIGN1
       RSQA2=ABS(RSQ2)
C
C
C        FLOW TO PORT #1 OF ACTUATOR
C
C
       IF(RSQA1 NE 0 0)THEN
       RQ1=0 172*RCONT*SQRT(RSQA1)*SIGN(1 0 RSQ1)
       ELSE
        RQ1=0 0
       END IF
C
C
C        FLOW TO PORT #2 OF ACTUATOR
C
C
       IF(RSQA2 NE 0 0)THEN
       RQ2=0 172*RCONT*SQRT(RSQA2)*SIGN(1 0 RSQ2)
       ELSE
        RQ2=0 0
       END IF
C
C
C
C
C           STATE EQUATIONS  ****************
C
C
       DV(1)=V(2)
       DV(2)=(1 0/RMASS)*(V(3)*A1-V(4)*A2)    '  LOAD EQN
       DV(3)=(BETA/VOL1)*(RQ1-A1*V(2))        '  P(1)
       DV(4)=(BETA/VOL2)*(-RQ2+A2*V(2))       '  P(2)
C
C
C
       RETURN
       END
C
C
C  ************************************************
C
C
C
C
C
          SUBROUTINE DERFUN1(RCONT N,V DV RFM)   ' UPDATED MODEL
C
C
C
C    DECLARE LOCALS
C
       INTEGER*2    N       ' NUMBER OF STATES
```

```
      REAL*4   RFM(2)     ' STATE EQ  PARAMETERS                    0113 000
      REAL*4    RCONT     ' CONTROL VARIABLE, MA                    0114 000
      REAL*4    V(4)      ' STATE VARIABLES                         0115 000
      REAL*4   DV(4)      ' STATE EQUATION                          0116 000
      REAL*4  XMAX        ' MAX PISTON TRAVEL                       0117 000
      REAL*4  A1          ' AREA #1, SQ INCH                        0118 000
      REAL*4  A2          ' AREA #2, SQ INCH                        0119 000
      REAL*4  RMASS       ' MASS, LB-SEC 2/INCH                     0120 000
      REAL*4  VOL1        ' #1 VOLUME, CU  INCH                     0121 000
      REAL*4  VOL2        ' #2 VOLUME, CU INCH                      0122 000
      REAL*4  RQ1         ' #1 PORT FLOW, CU IN/SEC                 0123 000
      REAL*4  RQ2         ' #2 PORT FLOW, CU IN/SEC                 0124 000
      REAL*4  RPS         ' SUPPLY PRESSURE, PSI                    0125 000
C                                                                   0126 000
C                                                                   0127 000
C                                                                   0128 000
      XMAX=30 0                                                     0129 000
      A1=8 0                                                        0130 000
      A2=8 0                                                        0131 000
      RMASS=8 634                                                   0132 000
      RPS=2000 0                                                    0133 000
C                                                                   0134 000
C********                                                           0135 000
C                                                                   0136 000
C                                                                   0137 000
C                                                                   0138 000
      VOL1=A1*(XMAX+V(1))                                           0139 000
      VOL2=A2*(XMAX-V(1))                                           0140 000
C                                                                   0141 000
C       FLOW CALCULATIONS                                           0142 000
C                                                                   0143 000
      RSIGN1=SIGN(1 0,RCONT)  ' SIGN OF CONTROL VARIABLE            0144 000
      RSQ1=RPS*0 5+(0 5*RPS-V(3))*RSIGN1                            0145 000
      RSQA1=ABS(RSQ1)                                               0146 000
      RSQ2=0 5*RPS-(0 5*RPS-V(4))*RSIGN1                            0147 000
      RSQA2=ABS(RSQ2)                                               0148 000
C                                                                   0149 000
C                                                                   0150 000
C           FLOW TO PORT #1 OF ACTUATOR                             0151 000
C                                                                   0152 000
C                                                                   0153 000
C                                                                   0154 000
      IF(RSQA1 NE 0 0)THEN                                          0155 000
      RQ1=RFM(2)*RCONT*SQRT(RSQA1)*SIGN(1 0 RSQ1)                   0156 000
      ELSE                                                          0157 000
       RQ1=0 0                                                      0158 000
      END IF                                                        0159 000
C                                                                   0160 000
C                                                                   0161 000
C          FLOW TO PORT #2 OF ACTUATOR                              0162 000
C                                                                   0163 000
C                                                                   0164 000
      IF(RSQA2 NE 0 0)THEN                                          0165 000
      RQ2=RFM(2)*RCONT*SQRT(RSQA2)*SIGN(1 0 RSQ2)                   0166 000
      ELSE                                                          0167 000
       RQ2=0 0                                                      0168 000
```



```
      END IF                                                        0169 000
C                                                                   0170 000
C                                                                   0171 000
C                                                                   0172 000
C                                                                   0173 000
C           STATE EQUATIONS  ***************                        0174 000
C                                                                   0175 000
C                                                                   0176 000
      DV(1)=V(2)                                                    0177 000
      DV(2)=(1 0/RMASS)*(V(3)*A1-V(4)*A2)    '  LOAD EQN            0178 000
      DV(3)=((1 0E06*RFM(1))/VOL1)*(RQ1-A1*V(2))        '  P(1)     0179 000
      DV(4)=((1 0E06*RFM(1))/VOL2)*(-RQ2+A2*V(2))       '  P(2)     0180 000
C                                                                   0181 000
C                                                                   0182 000
C                                                                   0183 000
      RETURN                                                        0184 000
      END                                                           0185 000
$A1 LIB=HYDROLIB,,U                                                 0186 000
$A1 DIR=HYDRODIR  U                                                 0187 000
$LIBED                                                              0188 000
$EOJ                                                                0189 000
$$                                                                  0190 000
```

```
$JOB B GRAD1 AMOS SLOF=L GRAD1                                          0000 100
$OPTION 2 3 4 5                                                         0000 200
$FORT77                                                                 0000 300
      SUBROUTINE GRAD1(X,N,GRAD,ESP,LSYS)                               0001 000
      REAL*4   X(20)                                                    0002 000
      REAL*4  GRAD(20)                                                  0003 000
      REAL*4  DELX                                                      0004 000
      REAL*4  FK        ' FUNCTION VALUE AT X(K)                        0005 000
      REAL*4  FK1       ' FUNCTION VAL AT X(K+1)                        0006 000
      REAL*4  ESP       ' FOR DELTA X                                   0007 000
      INTEGER*2  N      ' NO OF INDEPENDENT VARIABLES                   0008 000
      LOGICAL*1 LSYS                                                    0008 100
C                                                                       0009 000
C                                                                       0010 000
C     CALL FUNCT   (FK,X,N) ' FUNCTION VAL AT X(K)                      0011 000
      IF(LSYS)THEN                                                      0011 100
        CALL FUNCT1(FK)     ' IDENTIFICATION                            0011 200
      ELSE                                                              0011 300
        CALL FUNCT2(FK)     ' FEEDBACK                                  0011 400
      END IF                                                            0011 500
      DO 20 IA=1,N                                                      0012 000
      DELX=ESP                                                          0013 000
      IF(X(IA) NE 0 0)DELX=DELX* ABS(X(IA))                             0014 000
      X(IA)=X(IA)+DELX                                                  0015 000
C     CALL FUNCT   (FK1,X,N)  ' FUNCT AT X(K+1)                         0016 000
      IF(LSYS)THEN                                                      0016 100
        CALL FUNCT1(FK1)    ' IDENTIFICATION                            0016 200
      ELSE                                                              0016 300
        CALL FUNCT2(FK1)    ' FEEDBACK                                  0016 400
      END IF                                                            0016 500
      GRAD(IA)=(FK1-FK)/DELX                                            0017 000
      X(IA)=X(IA)-DELX          ' RESTORE X(K)                          0018 000
20    CONTINUE                                                          0019 000
      RETURN                                                            0020 000
      END                                                               0021 000
$A1 LIB=HYDROLIB, U                                                     0022 000
$A1 DIR=HYDRODIR  U                                                     0023 000
$LIBFD                                                                  0024 000
$EOJ                                                                    0025 000
$$                                                                      0026 000
```



```
C                                                                       0001 000
C                                                                       0002 000
C    COMMON FOR MAIN                                                    0003 000
C                                                                       0004 000
      REAL*4    Y(5)       ' STATES                                   0005 000
      REAL*4   DY(4)       ' STATE EQUATIONS                          0006 000
      REAL*4 RCONT         ' CONTROL VARIABLE                         0007 000
      REAL*4  BEGX                                                    0008 000
      REAL*4   RBEGX       ' STORAGE FOR TIME                         0009 000
      REAL*4  ENDX                                                    0010 000
      REAL*4  DELX                                                    0011 000
      REAL*4   DELTA       ' SAMPLING INTERVAL                        0012 000
      REAL*4  STRTY(5)     ' START SEGMENT                            0013 000
      REAL*4  ENDY(5)      ' END SEGMENT                              0014 000
      REAL*4   RK(4)       ' STATE FEEDBACK COEFFICIENTS              0015 000
      REAL*4   RREF(5)     ' REFERENCE SIGNAL INPUT                   0016 000
      REAL*4   RREFX(5)    ' STATIC REFERENCE FOR CONTROL             0017 000
      REAL*4   RFM(2)      ' STATE PARAMETERS(VARIABLES)              0018 000
      REAL*4   RREFC(5,15) ' REFERENCE FOR CONTROL                    0019 000
      REAL*4   RTANG (5)   ' SLOPE FOR REFERENCE,CONTR                0020 000
      REAL*4  X            '  TIME                                    0021 000
      REAL*4    RQF1(4)    ' WEIGHTING FACTORS,IDENTIF                0022 000
      REAL*4    RQF2(4)    ' WEIGHTING FACTORS,CONTROL                0023 000
      REAL*4   RREFI(5 60) ' GREED REFERENCE ACCUMULATED              0024 000
      REAL*4   RYXACT(5 15)' GREED POINT STATE VALUES                 0025 000
      REAL*4   RXR(60)     ' GREED POINTS TIME                        0026 000
      REAL*4   YX(5 60)    ' GREED POINTS STATE VALUES                0027 000
      INTEGER*2  JL        ' MAX GREED POINTS                         0028 000
      INTEGER*2  JA        ' GREED POINTS COUNTER                     0029 000
      INTEGER*2  N         ' NUMBER OF STATES                         0030 000
      LOGICAL*1  LPLOT     ' PLOT FLAG                                0031 000
      LOGICAL*1  LINEAR    ' LINEAR VERSION FLAG                      0032 000
C                                                                       0033 000
C                                                                       0034 000
      COMMON/CONTR/Y,DY,RCONT,BEGX,RBEGX,ENDX,DELX,DELTA                0035 000
     X  STRTY,ENDY,RK,RREF RREFX RFM RREFC RTANG X RQF1 RQF2            0036 000
     X  RREFI,RYXACT,RXR,YX                                             0037 000
     X JL,JA,N LPLOT LINEAR                                             0038 000
C                                                                       0039 000
C                                                                       0040 000
C                                                                       0041 000
```

```
$JOB B LXYPLT AMOS SLOF=L LXYPLT                                          0001 000
$OPTION 2 5                                                               0002 000
$FORT77                                                                   0003 000
      SUBROUTINE LXYPLT ( IFUN,MINT,MAX1,MIN1 MAX2,MIN2,MAX3,MIN3         0004 000
     1    ,MAX4,MIN4,VALN1,VALN2,VALN3,VALN4,PLTINC)                     0005 000
C                                                                         0006 000
C-------- GENERAL PURPOSE ON-LINE PLOT ROUTINE FOR LXY11 PRINTER -----    0007 000
C                                                                         0008 000
C                                                                         0009 000
C    AUTHOR -  MICHAEL I PILDITCH                                         0010 000
C    DATE -    16-AUG-82                                                  0011 000
C    VERSION - 04                                                         0012 000
C                                                                         0013 000
C    DESCRIPTION -                                                        0014 000
C        THIS SUBROUTINE IS DESIGNED AS A SIMPLE GENERAL PURPOSE          0015 000
C    GRAPHICS PROGRAM PROVIDING 'ON-LINE' TIME HISTORIES FOR UP TO        0016 000
C    FOUR VARIABLES  OUTPUT IS FORMATIED FOR THE PRINTRONICS (LXY11)      0017 000
C    TYPE PRINTER/PLOTTER ONLY                                            0018 000
C        THERE ARE TWO MODES OF OPERATION - FIRST INITIALISATION          0019 000
C    THIS DRAWS AND ANNOTATES THE Y OR HORIZONTAL AXES MAKING FULL        0020 000
C    USE OF THE PAPER WIDTH, AND SECOND DYNAMIC WHICH SCALES AND          0021 000
C    PLOTS THE Y VALUES AS THEY ARE INPUT                                 0022 000
C        MARKER FREQUENCY AND PAPER SCROLLING BETWEEN CONSEQUITIVE        0023 000
C    INPUTS MAY BE SPECIFIED BY THE USER  INTERPOLATION BETWEEN POINTS    0024 000
C    IS LINEAR AND EVERY TENTH MARKER SPANS THE FULL PAGE WIDTH TO        0025 000
C    ASSIST IN READING OF THE PRINTED OUTPUT  Y VALUES WHICH EXCEED THE   0026 000
C    PAPER MARGINS ARE FORCED TO THE NEAREST PERMISSABLE VALUE AT THE     0027 000
C    PAGE EDGE AND THEIR MARKERS REMOVED  THE X OR TIME AXIS IS           0028 000
C    AUTOMATICALLY DRAWN AT Y=0 0 IF THIS IS BETWEEN THE SPECIFIED MAX    0029 000
C    AND MIN VALUES OR ELSE IS ALLIGNED WITH THE LIMIT CLOSEST TO Y=0 0   0030 000
C        IF LESS THAN FOUR GRAPHS ARE SELECTED ALL VALUES PLACED          0031 000
C    IN THE UNUSED PARAMETERS ARE IGNORED                                 0032 000
C        THIS ROUTINE WILL ACCEPT ANY SCROLL INTERVAL GREATER OR          0033 000
C    EQUAL TO 0 05 INCHES, HOWEVER BECAUSE THERE ARE PHYSICALLY 72        0034 000
C    DOT LINES PER INCH IN THE X DIRECTION THOSE INTERVALS WHOSE          0035 000
C    RECIPROCALS ARE FACTORS OF 72 WILL BE ACCURATE WHILE OTHERS          0036 000
C    MAY BE IN ERROR BY UP TO 4/ DUE TO ROUNDING TO THE NEAREST DOT       0037 000
C    LINE                                                                 0038 000
C        IN THE EVENT THAT THE FIRST CALL TO THIS ROUTINE IS NOT          0039 000
C    AN INITIALISATION MODE (1,2,3,4) SUBSEQUENT CALLS WILL               0040 000
C    RESULT IN NO OUTPUT AND CONTROL WILL IMMEDIATLY RETURN TO THE        0041 000
C    CALLING PROGRAM                                                      0042 000
C                  ---------------------------                            0043 000
C                                                                         0044 000
C    FORMAT OF THE CALL TO THIS SUBROUTINE -                              0045 000
C                                                                         0046 000
C       CALL LXYPLT (IFUN  MINT, MAX1, MIN1, MAX2, MIN2, MAX3, MIN3,     0047 000
C     1  MAX4, MIN4, VALN1, VALN2, VALN3, VALN4, PLTINC)                  0048 000
C                                                                         0049 000
C                         INPUT PARAMETERS                                0050 000
C                                                                         0051 000
C  NAME   TYPE           DESCRIPTION                    TYPICAL VALUE     0052 000
C                                                                         0053 000
C  IFUN   INTEGER     CONTROLS MODE OF OPERATION                         0054 000
C                           0= PLOT Y VALUES                              0055 000
C                           1= SET AXIS FOR ONE GRAPH                     0056 000
C                           2= SET AXES FOR TWO GRAPHS                    0057 000
C                           3= SET AXES FOR THREE GRAPHS                  0058 000
C                           4= SET AXES FOR FOUR GRAPHS                   0059 000
C                                                                         0060 000
C  MINT   INTEGER     NUMBER OF CALLS BEFORE OUTPUT OF A MARKER   10     0061 000
C                                                                         0062 000
C  MAX1   REAL*4      MOST POSITIVE AXIS VALUE GRAPH 1          120 0    0063 000
C                                                                         0064 000
C  MIN1   REAL*4      MOST NEGATIVE AXIS VALUE GRAPH 1   -11 0, 2 1      0065 000
C                                                                         0066 000
C  MAX2   REAL*4      MOST POSITIVE AXIS VALUE GRAPH 2                   0067 000
C                     (IGNORED IF ONLY ONE GRAPH SELECTED)               0068 000
C                                                                         0069 000
C  MIN2   REAL*4      MOST NEGATIVE AXIS VALUE GRAPH 2                   0070 000
C                     (IGNORED IF ONLY ONE GRAPH SELECTED)               0071 000
C                                                                         0072 000
C  MAX3/MIN3 REAL*4   AS GRAPH 2 BUT IGNORED IF LESS THAN                0073 000
C                           THREE GRAPHS SELECTED                         0074 000
C                                                                         0075 000
C  MAX4/MIN4 REAL*4   AS GRAPH 3 BUT IGNORED IF LESS THAN                0076 000
C                           FOUR GRAPHS SELECTED                          0077 000
C                                                                         0078 000
C  VALN1    VALN4                                                         0079 000
C            REAL*4   INPUT Y VALUES (RELEVANT VALUE IGNORED            0080 000
C                           IF GRAPH NOT SELECTED)                        0081 000
C                                                                         0082 000
C  PLTINC  REAL*4     SCROLL DISTANCE BETWEEN CALLS (INCHES)     0 1      0083 000
C                                                                         0084 000
C                        --------------                                   0085 000
C                                                                         0086 000
C    NOTE - THIS SUBROUTINE MUST BE CATALOGUED WITH THE FOLLOWING         0087 000
C           ASSIGNMENT -                                                  0088 000
C                           A2 LP=SLO,20000                               0089 000
C                                                                         0090 000
C---------------------------------------------------------- ----------    0091 000
C                                                                         0092 000
C                                                                         0093 000
      LOGICAL*1  PLTBUF(132),  INC(7)                                     0094 000
      INTEGER*1  IBUF(132),IINC(7)                                        0095 000
      EQUIVALENCE (IBUF(1),PLTBUF(1))                                     0096 000
      EQUIVALENCE (IINC(1),INC(1))                                        0097 000
      REAL       FINC(6), MAX(4), MIN(4), VALO(4), VALN(4)  AXIS(4)       0098 000
      REAL       MAX1, MAX2, MAX3  MAX4, MIN1, MIN2, MIN3  MIN4         0099 000
      INTEGER  FLAG, INDX(4), IFUN, TI  LP                                0100 000
      DATA IINC/ 1, 2, 4,  8, 16, 32, 63/                                 0101 000
      DATA FINC/0 0166666,0 0333333,0 05,0 0666666,0 0833333 0 1001/      0102 000
C                                                                         0103 000
C SET LOGICAL UNIT NUMBERS                                                0104 000
C                                                                         0105 000
      DATA TI/ 1/                                                         0106 000
      DATA LP/ 3/                                                         0107 000
C                                                                         0108 000
      VALN(1)=VALN1                                                       0109 000
      VALN(2)=VALN2                                                       0110 000
      VALN(3)=VALN3                                                       0111 000
      VALN(4)=VALN4                                                       0112 000
```

```
C
      IF(IFUN  EQ  0) GOTO 1000
      IF(IFUN  EQ  1) GOTO 100
      IF(IFUN  EQ  2) GOTO 100
      IF(IFUN  EQ  3) GOTO 100
      IF(IFUN  EQ  4) GOTO 100
C
      WRITE('UT',50)
 50   FORMAT(' LXYPLT- - ILLEGAL MODE')
      RETURN
C
C
C                     DYNAMIC MODE
C
C
 1000 CONTINUE
C
      IF(FLAG  NE  1) RETURN
C
      MCOUNT=MCOUNT+1
C
      IF(IRND  NE  0) GOTO 1060
      NLPI=NLPI+1
 1060 CONTINUE
C
      DO 1050  L=1,NLPI
C
      DO 1010  I=1,130
      IBUF(I)= 64
 1010 CONTINUE
      IBUF(131)= 5
C
      DO 1055 IGN=1,NGR
      DO 1130 JJ=1,2
C
      VAL=VALO(IGN)+(VALN(IGN)-VALO(IGN))/NLPI*L
C
      IF(JJ  EQ  2) VAL=AXIS(IGN)
C
      FPOS=(VAL-MIN(IGN))/(MAX(IGN)-MIN(IGN))*GW
C
      IF(FPOS  GT  3200 0) FPOS = 3200 0
      IF(FPOS  LT -3200 0) FPOS =-3200 0
C
      IPOS=FPOS*10 0
      IF(FPOS  LT  0 0) IPOS=IPOS-1
      DELPOS=(FPOS*10 0-IPOS)/10 0
C
      J=1
 1020 IF(FINC(J)  GT  DELPOS) GOTO 1030
      J=J+1
      GOTO 1020
 1030 CONTINUE
C
      IF(MCOUNT/MINT*MINT  EQ  MCOUNT  AND  L  EQ  NLPI) J=7
C
```



```
      IF((IPOS+INDX(IGN))  LT  131) GOTO 1090
      PLTBUF(130)=PLTBUF(130)  OR  INC(6)
      GOTO 1080
 1090 CONTINUE
      IF((IPOS+INDX(IGN))  GT  1) GOTO 1100
      PLTBUF(2)=PLTBUF(2)  OR  INC(1)
      GOTO 1080
 1100 CONTINUE
C
      PLTBUF(IPOS+INDX(IGN))=PLTBUF(IPOS+INDX(IGN))  OR  INC(J)
      IF (J EQ 7)   IBUF(IPOS+INDX(IGN)) = IINC(J)
 1080 CONTINUE
 1130 CONTINUE
C
 1055 CONTINUE
C
      IF(J  EQ  7) MCNT=MCNT+1
C
      IF(MCNT/10*10  NE  MCNT  OR  J  NE  7) GOTO 1110
      DO 1120 II=1,130
      IBUF(II)= 63
 1120 CONTINUE
 1110 CONTINUE
C
      WRITE('LP',40) (PLTBUF(K),K=1,131)
 40   FORMAT(1X,131A1)
C
 1050 CONTINUE
C
      VALO(1)=VALN(1)
      VALO(2)=VALN(2)
      VALO(3)=VALN(3)
      VALO(4)=VALN(4)
C
      IRND=IRND-1
C
      IF(IRND  GE  0) GOTO 1070
      NLPI=NLPI-1
      IRND=IROUND
 1070 CONTINUE
C
      RETURN
C
C
C                     INITIALISATION MODE
C
C
 100  CONTINUE
C
      DO 140  I=1,130
      IBUF(I)= 64
 140  CONTINUE
      IBUF(131)= 5
C
      NGR=IFUN
      MCOUNT=0
```

```
      MCNT=0                                                                    0225 000
C                                                                               0226 000
      VALO(1)=VALN(1)                                                           0227 000
      VALO(2)=VALN(2)                                                           0228 000
      VALO(3)=VALN(3)                                                           0229 000
      VALO(4)=VALN(4)                                                           0230 000
C                                                                               0231 000
      IF(PLTINC  GT  0 0499) GOTO 300                                           0232 000
      WRITE('UT ,310)                                                           0233 000
 310  FORMAT(' LXYPLT-- SCROLL INTERVAL TOO SMALL (MIN 0 05 INCHES) )           0234 000
      RETURN                                                                    0235 000
 300  CONTINUE                                                                  0236 000
      NLPI=72 0*PLTINC                                                          0237 000
      IROUND=1 0/(72 0*PLTINC-NLPI)                                             0238 000
      IRND=IROUND                                                               0239 000
C                                                                               0240 000
      MAX(1)=MAX1                                                               0241 000
      MAX(2)=MAX2                                                               0242 000
      MAX(3)=MAX3                                                               0243 000
      MAX(4)=MAX4                                                               0244 000
      MIN(1)=MIN1                                                               0245 000
      MIN(2)=MIN2                                                               0246 000
      MIN(3)=MIN3                                                               0247 000
      MIN(4)=MIN4                                                               0248 000
C                                                                               0249 000
      DO 320  I=1,NGR                                                           0250 000
      IF(MIN(I)  LT  MAX(I)) GOTO 320                                           0251 000
      WRITE('UT',330) I                                                         0252 000
 330  FORMAT(' LXYPLT-- BAD LIMITS FOR GRAPH  ,I2)                              0253 000
      RETURN                                                                    0254 000
 320  CONTINUE                                                                  0255 000
C                                                                               0256 000
      DO 270 I=1,NGR                                                            0257 000
      IF(MAX(I)  GT  0 0) GOTO 280                                              0258 000
      AXIS(I)=MAX(I)                                                            0259 000
      GOTO 270                                                                  0260 000
 280  CONTINUE                                                                  0261 000
      IF(MIN(I)  LT  0 0) GOTO 290                                              0262 000
      AXIS(I)=MIN(I)                                                            0263 000
      GOTO 270                                                                  0264 000
 290  CONTINUE                                                                  0265 000
      AXIS(I)=0 0                                                               0266 000
 270  CONTINUE                                                                  0267 000
C                                                                               0268 000
      IF(NGR  NE  4) GOTO 180                                                   0269 000
      GW=3 0                                                                    0270 000
      INDX(1)=3                                                                 0271 000
      INDX(2)=35                                                                0272 000
      INDX(3)=68                                                                0273 000
      INDX(4)=100                                                               0274 000
      WRITE('LP ,113) MIN(1),MAX(1),MIN(2),MAX(2),MIN(3),MAX(3),                0275 000
     1   MIN(4),MAX(4)                                                          0276 000
 113  FORMAT(1X,E12 4,8X,E12 4,1X,E12 4 8X,E12 4,1X,E12 4,6X                    0277 000
     1   ,E12 4,1X,E12 4,8X,E12 4)                                              0278 000
      GOTO 150                                                                  0279 000
 180  CONTINUE                                                                  0280 000
```



```
      IF(NGR  NE  3) GOTO 160                                                   0281 000
      GW=4 0                                                                    0282 000
      INDX(1)=3                                                                 0283 000
      INDX(2)=46                                                                0284 000
      INDX(3)=89                                                                0285 000
      DIV1=MIN(1)+(MAX(1)-MIN(1))/2 0                                           0286 000
      DIV2=MIN(2)+(MAX(2)-MIN(2))/2 0                                           0287 000
      DIV3=MIN(3)+(MAX(3)-MIN(3))/2 0                                           0288 000
      WRITE( LP',110)MIN(1) DIV1,MAX(1) MIN(2),DIV2,MAX(2) MIN(3)               0289 000
     1   DIV3,MAX(3)                                                            0290 000
 110  FORMAT(1X,E12 4,4X,E12 4 4X E12 4 E12 4 3X,E12 4 4X F12 4                 0291 000
     1   E12 4,4X,E12 4 3X,E12 4)                                               0292 000
      GOTO 150                                                                  0293 000
 160  IF(NGR  NE  2) GOTO 170                                                   0294 000
      GW=6 0                                                                    0295 000
      INDX(1)=5                                                                 0296 000
      INDX(2)=70                                                                0297 000
      DIV=(MAX(1)-MIN(1))/3 0                                                   0298 000
      DIV1=MIN(1)+DIV                                                           0299 000
      DIV2=MIN(1)+DIV*2 0                                                       0300 000
      DIV=(MAX(2)-MIN(2))/3 0                                                   0301 000
      DIV3=MIN(2)+DIV                                                           0302 000
      DIV4=MIN(2)+DIV*2 0                                                       0303 000
      WRITE('LP',111)MIN(1),DIV1,DIV2 MAX(1),MIN(2),DIV3,DIV4 MAX(2)           0304 000
 111  FORMAT(1X,E12 4,5X,E12 4,6X,E12 4,6X E12 4 E12 4 6X F12 4 6X              0305 000
     1   E12 4,6X,E12 4)                                                        0306 000
      GOTO 150                                                                  0307 000
 170  CONTINUE                                                                  0308 000
      GW=12 0                                                                   0309 000
      INDX(1)=6                                                                 0310 000
      DIV=(MAX(1)-MIN(1))/6 0                                                   0311 000
      DIV1=MIN(1)+DIV                                                           0312 000
      DIV2=MIN(1)+DIV*2 0                                                       0313 000
      DIV3=MIN(1)+DIV*3 0                                                       0314 000
      DIV4=MIN(1)+DIV*4 0                                                       0315 000
      DIV5=MIN(1)+DIV*5 0                                                       0316 000
      WRITE('LP',112)MIN(1),DIV1,DIV2,DIV3,DIV4,DIV5,MAX(1)                    0317 000
 112  FORMAT(1X,6(E12 4,7X),E12 4)                                              0318 000
 150  CONTINUE                                                                  0319 000
C                                                                               0320 000
      DO 210 II=1,NGR                                                           0321 000
      PLTBUF(INDX(II))=PLTBUF(INDX(II))  OR  INC(1)                             0322 000
      IF(NGR  NE  1) GOTO 220                                                   0323 000
      DO 230 III=1,6                                                            0324 000
      PLTBUF(INDX(II)+III*20)=PLTBUF(INDX(II)+III*20)  OR  INC(1)               0325 000
 230  CONTINUE                                                                  0326 000
 220  CONTINUE                                                                  0327 000
      IF(NGR  NE  2) GOTO 240                                                   0328 000
      DO 250 III=1,3                                                            0329 000
      PLTBUF(INDX(II)+III*20)=PLTBUF(INDX(II)+III*20)  OR  INC(1)               0330 000
 250  CONTINUE                                                                  0331 000
 240  CONTINUE                                                                  0332 000
      IF(NGR  NE  3) GOTO 211                                                   0333 000
      DO 260 III=1,2                                                            0334 000
      PLTBUF(INDX(II)+III*20)=PLTBUF(INDX(II)+III*20)  OR  INC(1)               0335 000
 260  CONTINUE                                                                  0336 000
```

```
 04/11/84    17 01 27    TASK # 16000084          AMOS      GOULD S E L  MPX-3:

 211    CONTINUE                                                          0337 000
        IF(NGR  NE  4) GOTO 210                                           0338 000
        PLTBUF(INDX(II)+GW*10 0)=PLTBUF(INDX(II)+GW*10 0)  OR  INC(1)     0339 000
 210    CONTINUE                                                          0340 000
C                                                                         0341 000
        DO 120 I=1,7                                                      0342 000
        WRITE('LP',40)(PLTBUF(K),K=1,131)                                 0343 000
 120    CONTINUE                                                          0344 000
C                                                                         0345 000
        DO 200 II=1,NGR                                                   0346 000
        DO 130 I=INDX(II),INDX(II)+GW*10 0-1                              0347 000
        IBUF(I)= 63                                                       0348 000
 130    CONTINUE                                                          0349 000
 200    CONTINUE                                                          0350 000
C                                                                         0351 000
        WRITE ('LP ,40) (PLTBUF(K),K=1,131)                               0352 000
C                                                                         0353 000
        FLAG=1                                                            0354 000
C                                                                         0355 000
        RETURN                                                            0356 000
C                                                                         0357 000
        END                                                               0358 000
$A1 LIB=HYDROLIB,,U                                                       0358 100
$A1 DIR=HYDRODIR, U                                                       0358 200
$LIBED                                                                    0359 000
$EOJ                                                                      0360 000
$$                                                                        0361 000
```

```
 04/11/84    17 03 08    TASK # 16000084          AMOS      GOULD S E L  MPX-3

C                                                                         0001 000
C                                                                         0002 000
C       COMMON AREA TO BE USED BY STATE EQN(B DERFUN)                      0003 000
C                                                                         0004 000
      REAL*4     FM(4,4)        ' F-MATRIX                                0005 000
      REAL*4     GM(4)          ' G-MATRIX,HERE ONE CONTROL               0006 000
       COMMON/DERF/FM,GM                                                  0007 000
C                                                                         0008 000
C       STORED IN C DERF2                                                 0009 000
C                                                                         0010 000
C                                                                         0011 000
```

```
 04/11/84    17 02 16    TASK # 16000084          AMOS      GOULD S E L  MPX-3

  EXAMPLE FOR IMPLEMENTATION OF ADAPTIVE LINEAR CONTROL FEEDBACK          0000 100
  TO CONTROL HIGHLY NON-LINEAR ELECTRO-HYDRAULIC SYSTEM                   0000 200
  AMOS BURSHTEIN  BURTEK INC,DECEMBER  1983                               0000 300
   4                                                                      0000 400
0 0                  0 0              0 0            0 0                 0001 000
1 0                        0        -5000 0          5000 00             0002 000
0 0              926                  -21 5625        0 0                 0003 000
0 0            - 926                    0 0          -21 5625            0004 000
0 0             0 0                   1075 0         1075 0              0005 000
0 0             0 050                  0 001          0 2                 0006 000
0 0              0 0                    1000 0          1000 0           0007 000
0 10            0 1                     1295          - 1295              0008 000
0 0             0 0                    1000 0         1000 0              0009 000
 1 0             1 0E-2                  1 0E-4         1 0E-4             0010 000
0 0E-02          0 0E-02                 1 0E00         1 0E00             0011 000
 140              180                                                     0012 000
```

```
 04/11/84    17 03 25    TASK # 16000084          AMOS      GOULD S E L  MPX 3

C                                                                         0001 000
C                                                                         0002 000
C       C DERF1                                                           0003 000
C                                                                         0004 000
C       REAL*4 RFM(1) ' BULK MODULUS OF FLUID(1 5E5)                      0005 000
        REAL*4 RFM(2) ' VALVE GAIN ( 172)                                 0006 000
        COMMON/DERF1/RFM                                                  0007 000
C                                                                         0008 000
C                                                                         0009 000
C*****************                                                        0010 000
```

VITA

Amos Burshtein

Candidate for the Degree of

Doctor of Philosophy

Thesis: A SYNTHESIS METHOD FOR ADAPTIVE CONTROL OF NONLINEAR SYSTEMS WITH APPLICATION TO A ONE DEGREE-OF-FREEDOM MOTION SIMULATOR

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Tel-Aviv, Israel, June 21, 1940, the son of Jacob and Sara Burshtein. Married to Nili Arielly on October 21, 1973; have two daughters Iris and Tali, and one son, Eran.

Education: Received the Bachelor of Science degree in Mechanical Engineering from Israel Institute of Technology, Haifa, Israel, in June, 1962; received the Master of Science degree in Civil Engineering from the University of Minnesota in June,1971; completed requirements for the Doctor of Philosophy Degree at Oklahoma State University in December, 1984.

Professional Experience: Supervisor, assembly line of tanks, Israeli Defense Forces, 1963-1964; Project Engineer, material handling and recovery tank, Israeli Defense Forces, 1965-1968; Laboratory Engineer, American Hoist & Derrick Co., St. Paul, Minnesota, 1969-1970; Manager, Special Ground Support Equipment Engineering Department, Israel Aircraft Industries, 1972-1976; Manager, Logistic and Warehouse Planning Unit, Israel Aircraft Industries, 1977-1980; System Engineer, Burtek, Inc., Tulsa, Oklahoma, 1981 to present.