1976-06

Microcomputer based pitch and depth controller for a submarine using optimal control theory

Dockhorn, Heinz-Dieter

Monterey, California. Naval Postgraduate School

MICROCOMPUTER BASED PITCH AND DEPTH CONTROLLER FOR A SUBMARINE USING OPTIMAL CONTROL THEORY

Heinz-Dieter Dockhorn

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MICROCOMPUTER BASED PITCH AND DEPTH CONTROLLER FOR A SUBMARINE USING OPTIMAL CONTROL THEORY

by

Heinz-Dieter Dockhorn

June 1976

Thesis Advisor: U.R. Kodres

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
Microcomputer Based Pitch and Depth Controller for a Submarine Using Optimal Control Theory

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis; June 1976

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Heinz-Dieter Dockhorn

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

12. REPORT DATE
June 1976

13. NUMBER OF PAGES
139

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Microcomputer
Pitch and Depth Controller
Submarine microcomputer

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Microcomputer systems will influence future technical changes in a submarine. As a demonstration of the computational power of the microcomputer, an optimal controller for the pitch and depth of a submarine was developed using the INTEL 8080 microcomputer. A model simulating a submarine and the control algorithm were first written in a simulation language DSL/360. The control

(20. ABSTRACT Continued)

algorithm was then coded in the microcomputer language PLM and implemented on a microcomputer. Also included is an investigation of the feasibility to execute the PLM program in a given time frame, an analysis of the computational errors and the inherent errors in measurements from peripheral devices.

Microcomputer Based Pitch and Depth Controller for a Submarine Using Optimal Control Theory

by

Heinz-Dieter Dockhorn
Lieutenant Commander, Federal German Navy

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1976

ABSTRACT

Microcomputer systems will influence future technical changes in a submarine. As a demonstration of the computational power of the microcomputer, an optimal controller for the pitch and depth of a submarine was developed using the INTEL 8080 microcomputer. A model simulating a submarine and the control algorithm were first written in a simulation language DSL/360. The control algorithm was then coded in the microcomputer language PLM and implemented on a microcomputer. Also included is an investigation of the feasibility to execute the PLM program in a given time frame, an analysis of the computational errors and the inherent errors in measurements from peripheral devices.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The application of microprocessors, which began early in 1972, is rapidly becoming a major computational activity. The control of traffic lights, smart terminals and cash registers, printer controls, etc., were some of the first applications. Future applications are almost unlimited, starting with present implementations of complex plant process controls and numerically controlled machines.

Because of their small size, light weight, low power requirement and, most important of all, low cost, microcomputer systems are ideally suited for use on board submarines. Data processing with the aid of microcomputers is possible in almost all technical and organizational fields in a submarine, such as:

1. sensors and signal processors,
2. fire control and weapons control,
3. communications and navigation,
4. central command, information display and administration,
5. ship handling and automatic control.

This thesis developed an algorithm for controlling depth and pitch of a submarine using a microcomputer system as a demonstration of the above-mentioned possibilities. The hardware configuration used was an INTELLEC 8 microcomputer system.

The architecture of the microprocessor, the INTEL 8080, around which the microcomputer system INTELLEC 8 is built, consists of an ALU (arithmetic logic unit) with an 8 bit parallel operation and with the option of either binary coded decimal or binary arithmetic. The 8080 has a 16 bit address capability, which allows it to communicate to 64K bytes of memory. To achieve all the address and parallel data capability, the 8080 is packaged in a 40 pin package.

In developing the controller, a mathematical model for the simulation and the controller is developed first. These models are implemented in the digital simulation language DSL/360. The control part then is coded in the microcomputer language PLM to investigate the feasibility of implementing the controller.

II. MATHEMATICAL MODEL

There are six equations of motion of a submarine; one for each degree of freedom. Three equations are for linear motion along each of the three axes, and three equations are for rotary motion about each of the three axes [Figure]]. The submarine is controlled in all six degrees of freedom by a set of movable plane control surfaces, movement of ballast water and the propulsion system. The set of movable plane control surfaces consist of the fairwater planes, the stern planes and a rudder.

The controller for depth and pitch of a submarine described in the following sections minimizes the sum of the errors in depth and in pitch together with the control effort. The controller counteracts deviations from ordered depth and ordered pitch caused by external forces by the movements of fairwater and stern planes.

The simulated submarine is a hypothetical one. The dimensions and hydrodynamic coefficients of this submarine are taken from Ref. 1. The implemented controller is unique to this submarine, but the underlying idea is usable in general. The basic model for the simulation and the controller are taken from Ref. 6.

A. EQUATIONS OF MOTION FOR A SUBMARINE

The equations of motion are found in Ref. 2. With the assumption that the principal axes of inertia of the

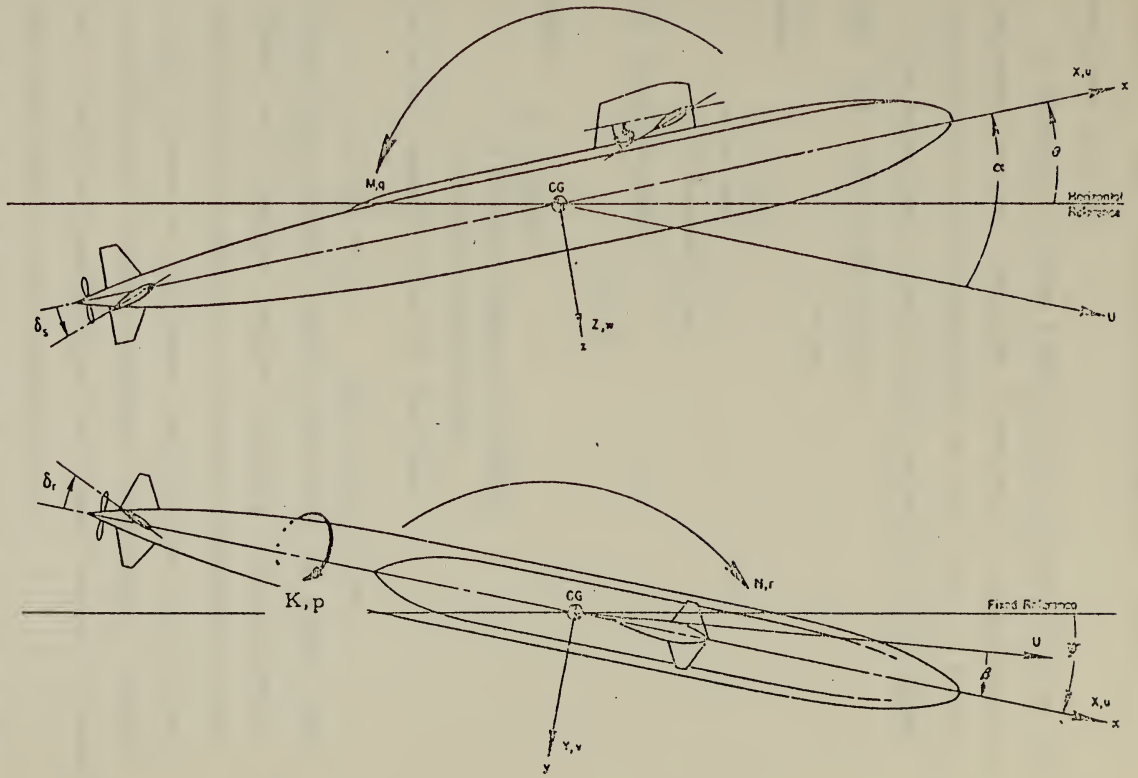


Figure 1 - Sketch Showing Positive Directions of Axes, Angles, Velocities, Forces, and Moments

submarine coincide with the origin placed at the center of gravity, a modified form of the equations of motion is obtained [Ref. 1].

In Appendix A these equations are linked together with the nomenclature. Positive directions of translations and rotations are in the directions of the arrows in Figure 1.

The equations of motion are basically of two forms [Ref. 4]. For linear motion:

$$f = ma$$

where

f = force
m = mass
a = acceleration.

For rotary motions:

$$M = I\ddot{\theta}$$

where

M = torque
I = angular moment of inertia
$\ddot{\theta}$ = angular acceleration.

The actual equations are of the form:

$$\sum (ma) = \sum f \quad \text{and} \quad \sum (I\ddot{\theta}) = \sum M .$$

The equations contain forcing terms for the planes, the rudder, the propeller and the trimming of ballastwater. The input for the propeller is a function obtained by curve fitting. There are different coefficients for different propeller modes (forward, backing, acceleration, slowing down). Only the coefficients for forward motion at constant speed were used in this investigation.

The manipulation of these equations to get the expressions for $\dot{u}$, $\dot{v}$, $\dot{w}$, $\dot{p}$, $\dot{q}$, $\dot{r}$ are shown in Appendix A.

B. DEPTH AND PITCH CONTROL USING OPTIMAL CONTROL THEORY

To use the equations of motion in the controller, the nonlinear equations were linearized. The linearization and the assumptions made are shown in Appendix A. With the resulting equations involving w, the vertical velocity component, and q, the angular velocity about the body y-axis, the problem is reduced to two degrees of freedom. The controls δ_s and δ_b correspond to deflections of the stern plane and bow plane, respectively.

$$(1)\quad \begin{bmatrix} m'-Z'_{\dot{w}} & -\ell Z'_{\dot{q}} \\ -M'_{\dot{w}}/\ell & I'_y-M'_{\dot{q}} \end{bmatrix} \begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} u_a Z'_w/\ell & (m'+Z_q)u_a \\ u_a M'_w/\ell^2 & u_a M'_q/\ell \end{bmatrix} \begin{bmatrix} w \\ q \end{bmatrix}$$

$$+ \begin{bmatrix} u_a^2 Z'\delta_s/\ell & u_a^2 Z'\delta_b/\ell \\ u_a^2 M'\delta_s/\ell^2 & u_a^2 M'\delta_b/\ell^2 \end{bmatrix} \begin{bmatrix} \delta_s \\ \delta_b \end{bmatrix} .$$

More generally, the linearized equations written in state equation form are

$$(2)\quad \dot{x} = Gx + Bu \quad ,$$

where

$$\dot{x} = \begin{bmatrix} \dot{z} \\ w \\ \dot{\theta} \\ \ddot{q} \end{bmatrix} \qquad G = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & g_{22} & 0 & g_{24} \\ 0 & 0 & 0 & 1 \\ 0 & g_{42} & 0 & g_{44} \end{bmatrix}$$

$$x = \begin{bmatrix} z \\ \dot{w} \\ \theta \\ \dot{q} \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ b_{21} & b_{22} \\ 0 & 0 \\ b_{41} & b_{42} \end{bmatrix} \qquad u = \begin{bmatrix} \delta_s \\ \delta_b \end{bmatrix}$$

with

$$g_{22} = [(I_y' - M_{\dot{q}}')Z_w + M_w'Z_{\dot{q}}']u_a/(\ell \cdot DET)$$

$$g_{24} = [(I_y' - M_{\dot{q}}')(m' + Z_q') + M_{\dot{q}}'Z_{\dot{q}}']u_a/DET$$

$$g_{42} = [M_{\dot{w}}'Z_w' + M_w'(m' - Z_{\dot{w}}')]u_a/(\ell^2 \cdot DET)$$

$$g_{44} = [M_{\dot{w}}'(m' + Z_q') + M_q'(m' - Z_{\dot{w}}')]u_a/(\ell \cdot DET)$$

$$b_{21} = [(I_y' - M_{\dot{q}}')Z_{\delta_s}' + M_{\delta_s}'Z_{\dot{q}}']u_a^2/(\ell \cdot DET)$$

$$b_{22} = [(I_y' - M_{\dot{q}}')Z_{\delta_b}' + M_{\delta_b}'Z_{\dot{q}}']u_a^2/(\ell \cdot DET)$$

$$b_{41} = [M_{\dot{w}}'Z_{\delta_s}' + (m' - Z_{\dot{w}}')M_{\delta_s}']u_a^2/(\ell \cdot DET)$$

$$b_{42} = [M_{\dot{w}}'Z_{\delta_b}' + (m' - Z_{\dot{w}}')M_{\delta_b}']u_a^2/\ell^2 \cdot DET)$$

$$DET = (m' - Z_{\dot{w}}')(I_y' - M_{\dot{q}}') - Z_{\dot{q}}'M_w'$$

Referring to Ref. 3, the linear tracking model was used to solve the problem because the desired value of the state vector is not the origin. The reference vector in this case is

$$r = \begin{bmatrix} r_1 \\ 0 \\ r_3 \\ 0 \end{bmatrix}$$

with r_1 = ordered depth and

r_3 = ordered pitch,

both are considered to be constant.

Let

$$y \triangleq x - r$$

$$\dot{y} = \dot{x} - \dot{r} \tag{3}$$

and substituting (2) into (3) yields

$$\dot{y} = Gx + Bu - \dot{r} \quad . \tag{4}$$

Realizing that

$$\dot{r} = 0$$

and

$$Gr = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

equation (4) can be written

$$\dot{y} = G[x - r] + Bu$$

$$\dot{y} = Gy + Bu \quad . \tag{5}$$

The performance measure to be minimized may be expressed as

$$J = \frac{1}{2} \int_{t_o}^{t_f} [y^T(t)Qy(t) + u^T(t)Su(t)]\, dt \tag{6}$$

where the final time t_f is fixed and $y(t_f)$ is free. Q is a real symmetric positive semi definite weighting matrix and S is a real symmetric positive definite weighting matrix. Then (5) and (6) define a linear regulator problem.

The Hamiltonian is given by

$$H(y(t),u(t);p(t),t) = \frac{1}{2} [y^T(t)Qy(t) + u(t)^T Su(t)] + p(t)^T Gy(t) + p(t)^T Bu(t) \quad .$$

The costate equation is

$$\dot{p}^*(t) = -\frac{\partial H}{\partial y} = -Qy(t) - G^T p^*(t) . \tag{7}$$ [1]

---

[1] Optimality is indicated by asterisks.

The algebraic relation which has to be satisfied is

$$0 = \frac{\partial H}{\partial u} = Su^*(t) + B^T p^*(t)$$

which gives the control vector

$$u^*(t) = -S^{-1}B^T p^*(t) \tag{8}$$

whose elements are the control plane deflections $_s$ and $_b$. Substituting (8) in the state equation (5) and forming the augmented matrix yields

$$\begin{bmatrix} \dot{y}^*(t) \\ \hline \dot{p}^*(t) \end{bmatrix} = \left[\begin{array}{c|c} G & -BS^{-1}B^T \\ \hline -Q & -G^T \end{array}\right] \begin{bmatrix} y^*(t) \\ \hline p^*(t) \end{bmatrix} . \tag{9}$$

Reference 3 gives the boundary condition as $p^*(t_f) = 0$.

Defining

$$p^*(t) = K(t)y^*(t) \qquad \text{[Ref. 3]} \tag{10}$$

$$\dot{p}^*(t) = \dot{K}(t)y^*(t) + K(t)\dot{y}^*(t) \tag{11}$$

and combining (7) and (11) gives

$$-Qy^*(t) - G^T p^*(t) = \dot{K}(t)y^*(t) + K(t)\dot{y}^*(t) .$$

Solving for $\dot{K}(t)$, substituting $\dot{y}^*(t)$ from (9) and replacing $p^*(t)$ by (10) yields

$$\dot{K}(t)y^*(t) = -K(t)Gy^*(t) + K(t)BS^{-1}B^TK(t)y^*(t) - Qy^*(t) - G^TK(t)y^*(t)$$

which must be satisfied for arbitrary $y^*(t)$. Hence

$$\dot{K}(t) = -K(t)G + K(t)BS^{-1}B^TK(t) - Q - G^TK(t) \quad . \tag{12}$$

The solution to equation (12), the Riccati equation, is the matrix of the optimal parameters for the defined mathematical model based on the quadratic performance measure (6). Equations (8) and (10) give the final expression for the control vector in terms of the matrix K

$$u^*(t) = -S^{-1}B^TK(t)y^*(t)$$

or

$$u^*(t) = S^{-1}F^Ty^*(t) \tag{13}$$

where

$$F(t) \triangleq -K(t)B \quad .$$

K(t) is a symmetric 4x4 matrix, so that ten differential equations for K(t) must be solved to determine the optimal gains for $u^*(t)$. The solution process is described in

section III.A.1. From Ref. 3 $K(t_f) = 0.0$. The matrices Q and S are weighting matrices and have the form

$$Q = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & w_2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$S = \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix}$$

where w_1 is the weight for the error in depth, w_2 is the weight for the error in pitch and c_1 and c_2 are the weights for the control input.

Since only the value of $K(t_f)$ is known, equation (12) has to be solved backwards. By multiplying the right side of (12) by -1.0 and reversing the order of integration, $K(t_f)$ becomes the initial condition.

III. COMPUTER IMPLEMENTATIONS OF THE MATHEMATICAL MODEL

The programming language for the simulation and the controller of the submarine was DSL/360, a digital simulation language. The programs were run on the IBM 360/67 computer at the Naval Postgraduate School.

For the implementation of the controller on an INTELLEC 8 microcomputer, the programming language PLM was used. The PLM programs were compiled with a cross-compiler on the IBM 360/67.

A. DSL/360 PROGRAMS

The DSL programming part was divided into two programs. The first program (GAIN) performs the computations which have to be done only once. The results were transferred to a datacell for later use by the second DSL/360 program (SIMUL), the actual simulation of the submarine. Figure 2 shows a general flow diagram for both DSL/360 programs. Figure 3 gives a graphical representation of the different timing intervals. A communications interval in DSL/360 is an interval after which information is returned to the user. In the case of these two programs the sample region is called in communication time intervals to collect data points for the graphical output. The length of the communication interval is determined by the program control variable DELS.

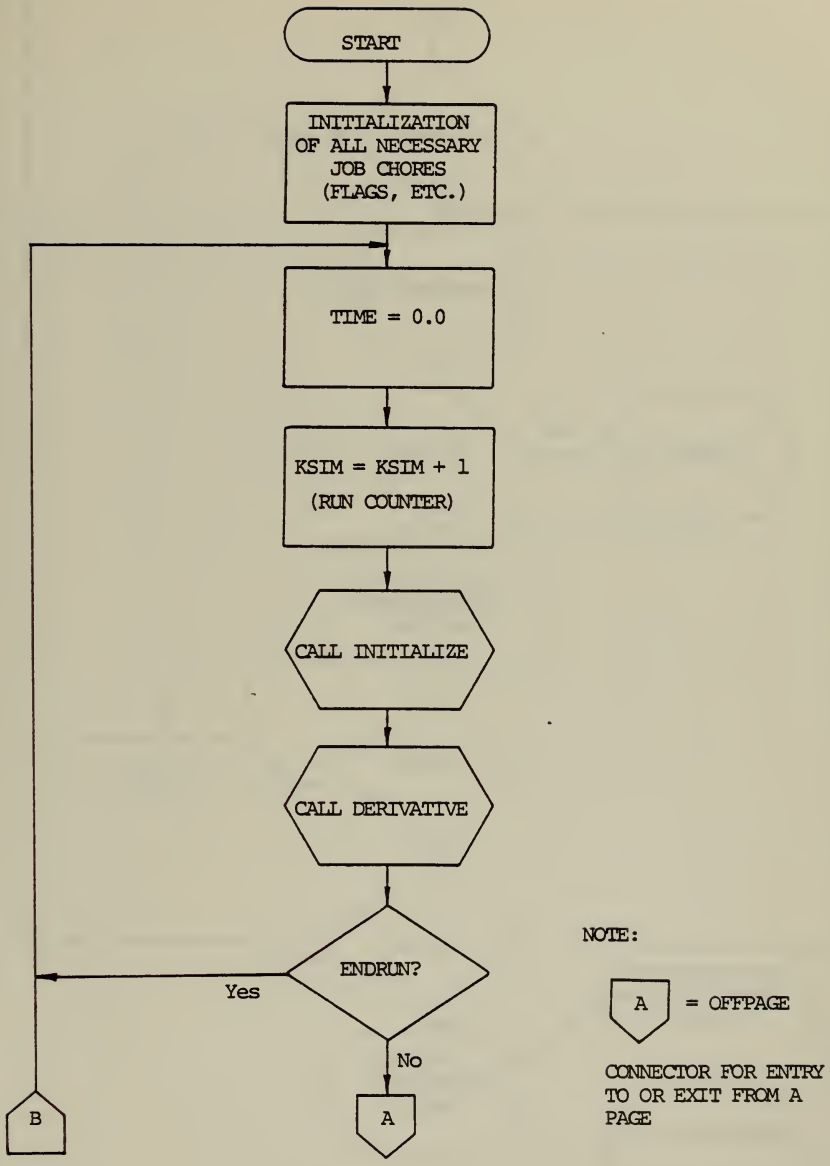

START
INITIALIZATION OF ALL NECESSARY JOB CHORES (FLAGS, ETC.)
TIME = 0.0
KSIM = KSIM + 1
(RUN COUNTER)
CALL INITIALIZE
CALL DERIVATIVE
ENDRUN?
Yes
No
B
A
NOTE:
A = OFFPAGE
CONNECTOR FOR ENTRY TO OR EXIT FROM A PAGE

Figure 2

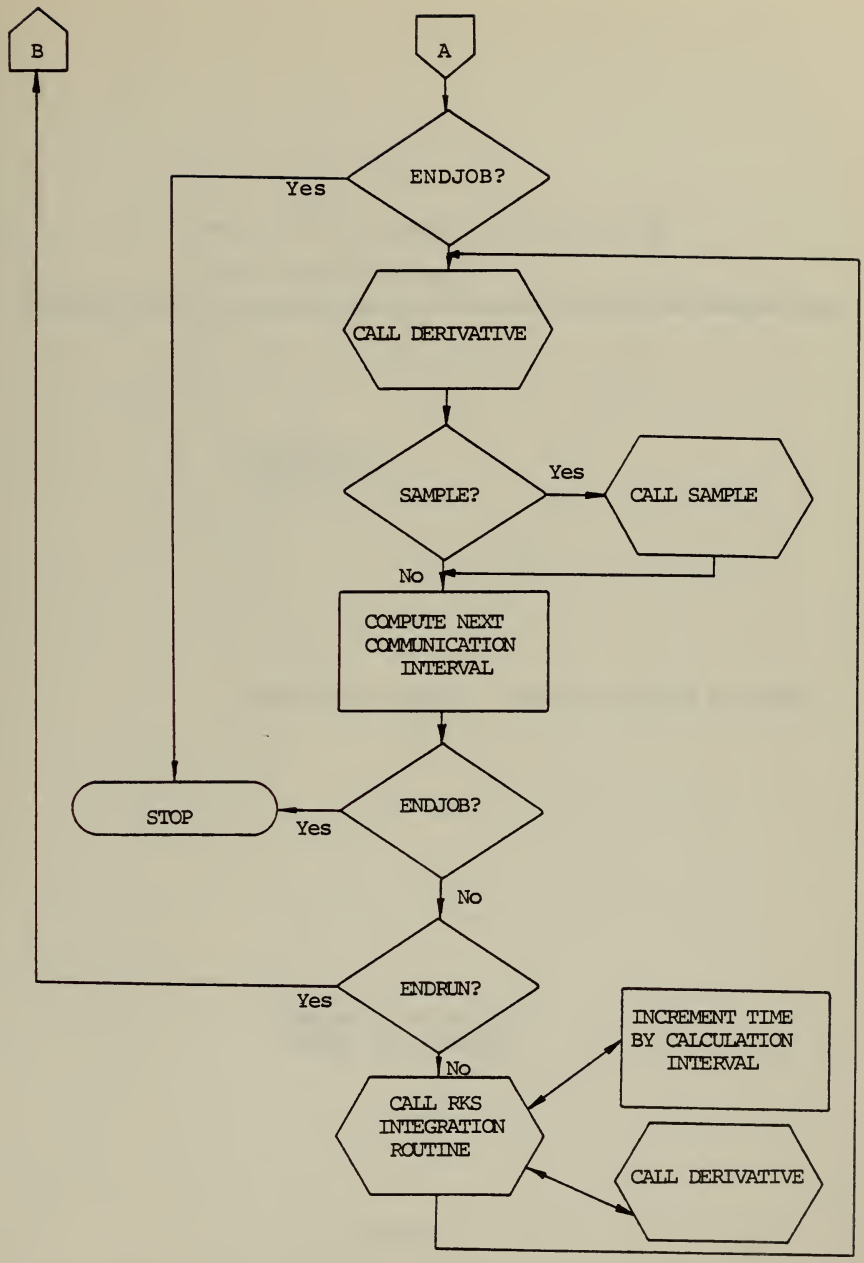


Figure 2 (Continued)

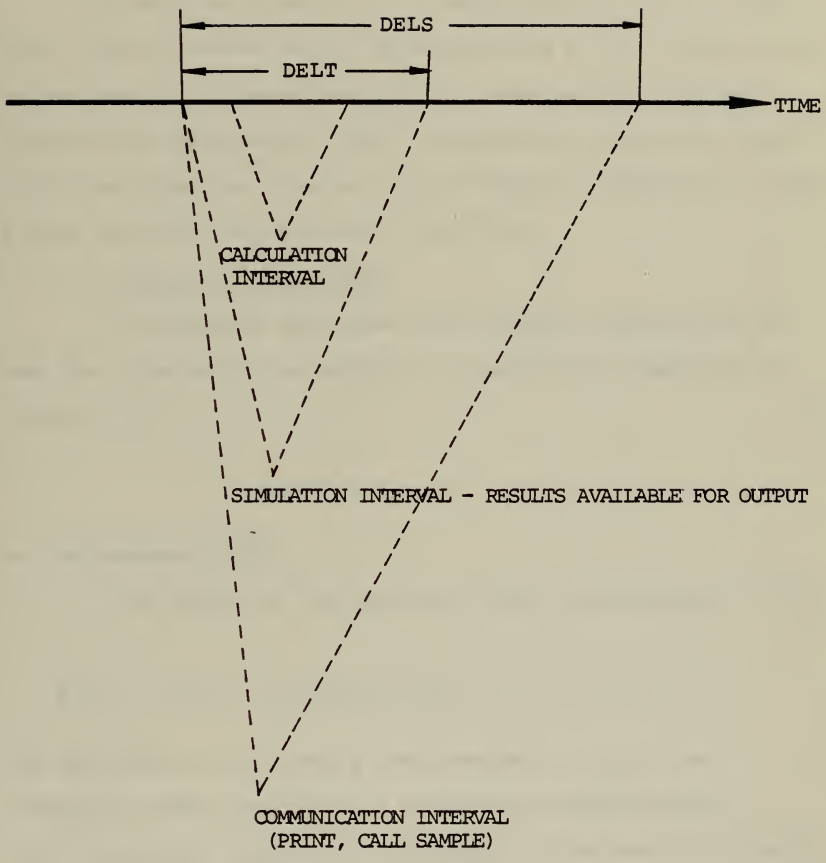

DELS
DELT
TIME
CALCULATION
INTERVAL
SIMULATION INTERVAL - RESULTS AVAILABLE FOR OUTPUT
COMMUNICATION INTERVAL
(PRINT, CALL SAMPLE)

Figure 3

A simulation interval is the unit of time for a particular integration routine to complete an integration step. The length of the simulation interval is determined by the program control variable DELT.

A calculation interval is a subinterval of the simulation interval whose length is determined by the integration method used. In these two DSL/360 programs the variable length step Runge Kutta (RKS) integration routine was used with a calculation interval of 0.25*DELT. Reference 5 gives a more detailed description of DSL/360.

1. DSL/360 Program GAIN

The program GAIN uses the library routine MINV to get the inverse of the matrix A to solve the equations of notation (1),

$$A \cdot DOT = Z$$

by the program SIMUL.

The kernel of the program GAIN is the equation (12)

$$\dot{K}(t) = -K(t) \cdot G + K(t)BS^{-1}B^{T}K(t) - Q - G^{T}K(t) \ .$$

The matrices G, B, R and Q are constant. Under the assumption that the plant is completely controllable, $K(t) \to K$ as $t_f \to \infty$ [Ref. 3]. But even if the control interval is finite and long with respect to the decay time of the optimal gains, it is possible to use the fixed control law (see Figs. 4.1 to 5.8).

To compute the steady state gains K, either the Riccati equation (12) is integrated or, setting $\dot{K}(t) = 0$, the algebraic equation

$$0 = -KG - G^TK - Q + KBS^{-1}B^TK$$

is solved. The method of integrating equation (12) was chosen because of the graphical output (Figs. 4.1 to 5.8) which shows the behavior of K(t) for $t \to \infty$. The gain is also a function of the submarine's speed u_a. Therefore different gains were computed for a representative set of speeds u_a. This approach may result in suboptimal gains when speed u_a deviates, e.g., in a turn or during a change of depth (see Figs. 9.5 and 10.5). But even in the maneuver of a turn (Fig. 10.5) the temporary drop in speed u_a occurs over a time period which is long compared to the decay time of the optimal gains.

Together with the computations for different speeds u_a, different weights w_1 and w_2 were chosen to have the possibility of adjusting to environmental effects (seastate, depth of water, on periscope depth, speed, etc.). The gains for these different combinations were computed in repeated DSL/360 runs.

The values chosen for this hypothetical submarine were:

i) Speed u_a from 1.689 ft./sec. to 30.402 ft./sec. in increments of 1.689 ft./sec. = 1.0kn

ii) Weight w_1: 0.8, 0.9, 1.0, 1.1, 1.2

iii) Weight w_2: 2400.0, 2700.0, 3000,0, 3300.0, 3600.0

These values depend on the specific type of submarine and have to be determined experimentally or by a simulation for that specific submarine.

The steady state gains were assumed after 70 sec. for small u_a's and sooner for larger u_a's (see Figs. 4.1 to 5.8). The length of each run was set to 200 sec. The sets of gain values were collected in a five-dimensional array FF which was printed and transferred to a datacell after the last run.

2. DSL/360 - Program SIMUL

Program SIMUL consists of the main program and the subroutines CNTRL, PLGEN, and PRIME. The program is centered around the equations

(1) $$DOT = A^{-1} \cdot Z$$

and

(13) $$u(t) = S^{-1} F^T y^*(t) \quad .$$

To use these equations the inverse of A and the matrix FF containing all F(I,J) values have to be read from the datacell at the beginning of the program.

With the integrated elements of DOT the values of the variables PITCH and DEPTH are computed. These variables are controlled by the automatic controller. To manipulate these variables, the movements of fairwater and stern planes have to be simulated. For turns about the vertical axes the rudder is simulated, too.

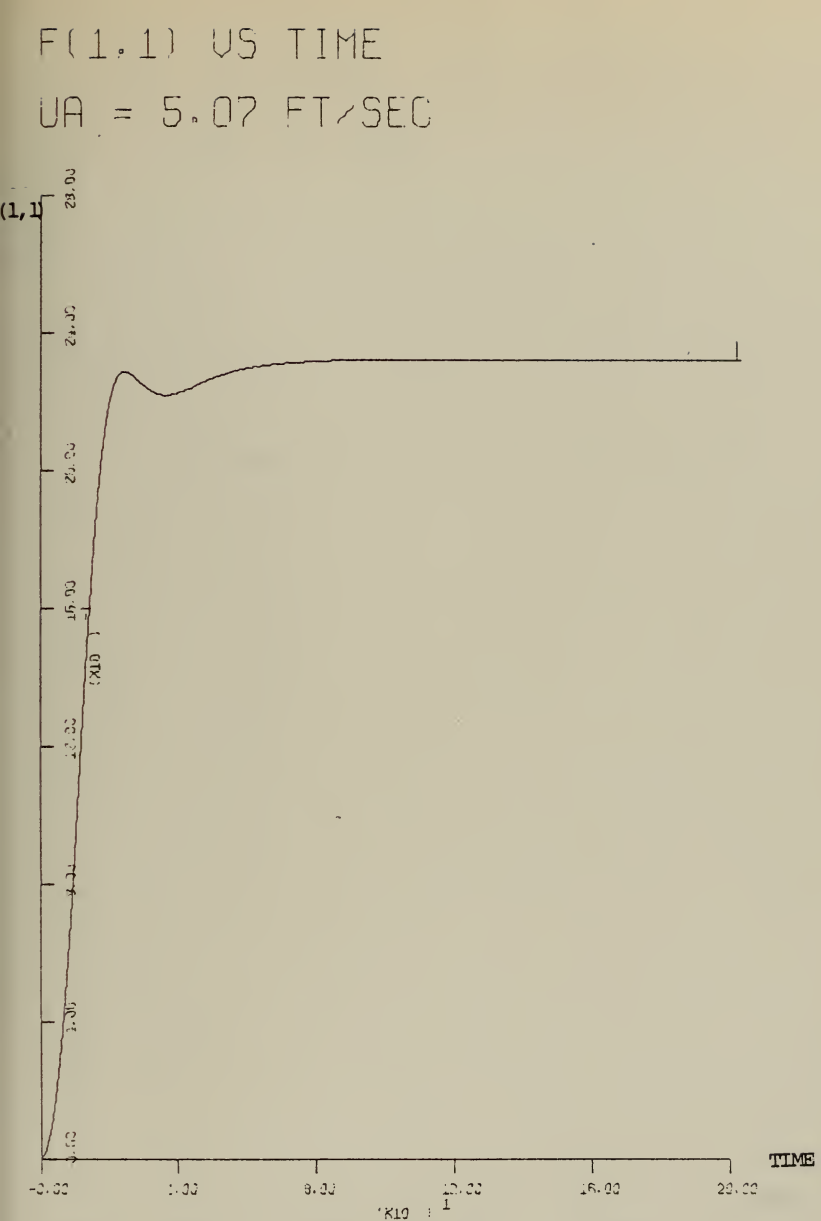

F(1,1) VS TIME
UA = 5.07 FT/SEC
(1,1)
TIME
XSCALE=40.00 UNITS/INCH RUN NO. 1
YSCALE=0.40 UNITS/INCH PLOT NO. 1

FIGURE 4.1

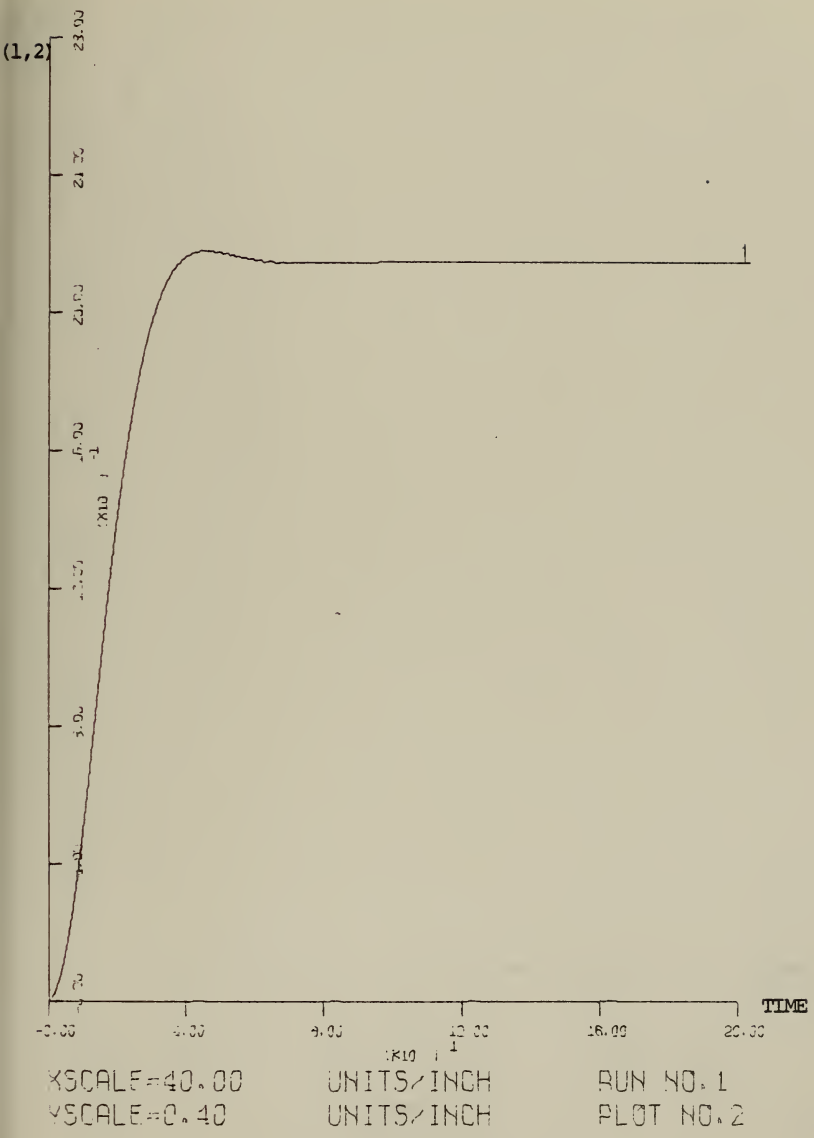

F(1,2) VS TIME
UA = 5.07 FT/SEC
(1,2)
TIME
XSCALE=40.00 UNITS/INCH RUN NO. 1
YSCALE=0.40 UNITS/INCH PLOT NO. 2

FIGURE 4.2

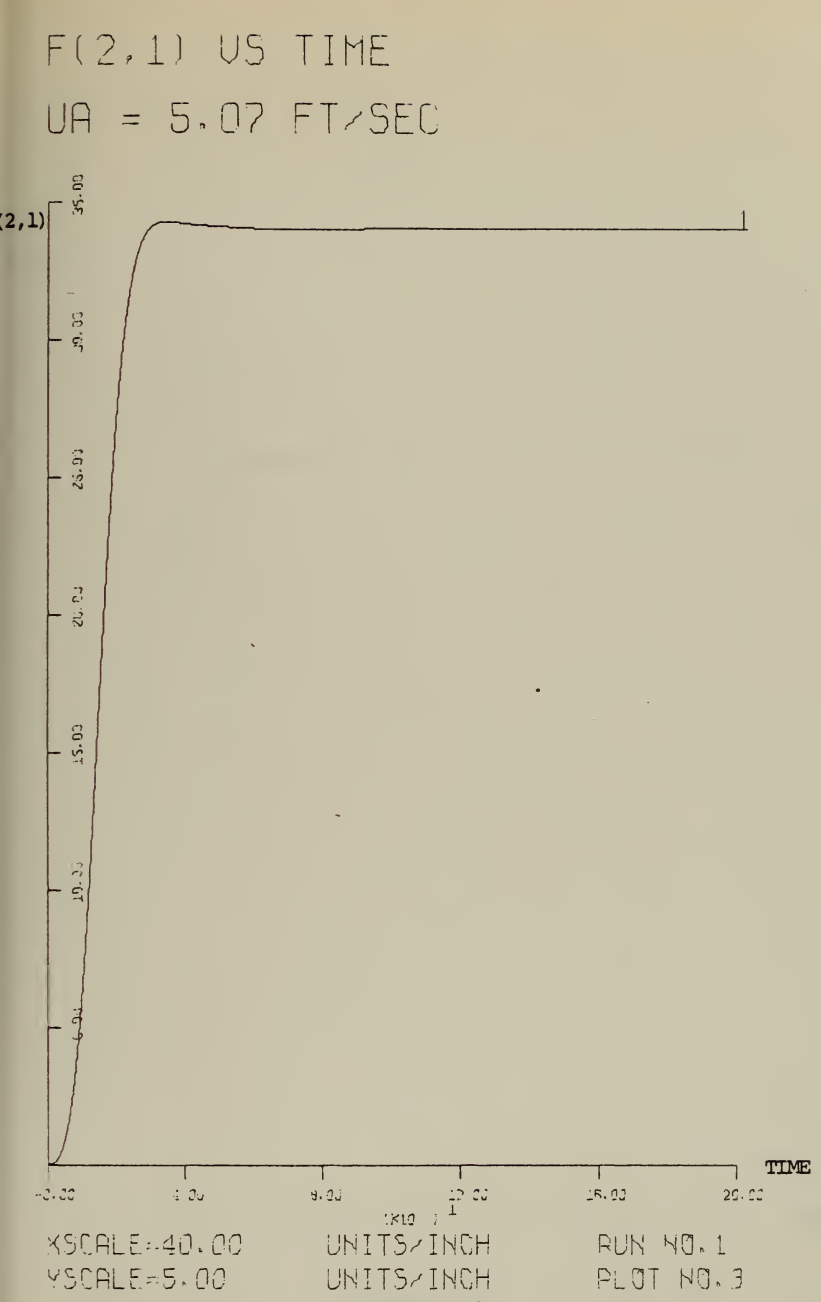

F(2,1) VS TIME
UA = 5.07 FT/SEC
F(2,1)
TIME
XSCALE=40.00 UNITS/INCH
YSCALE=5.00 UNITS/INCH
RUN NO. 1
PLOT NO. 3

FIGURE 4.3

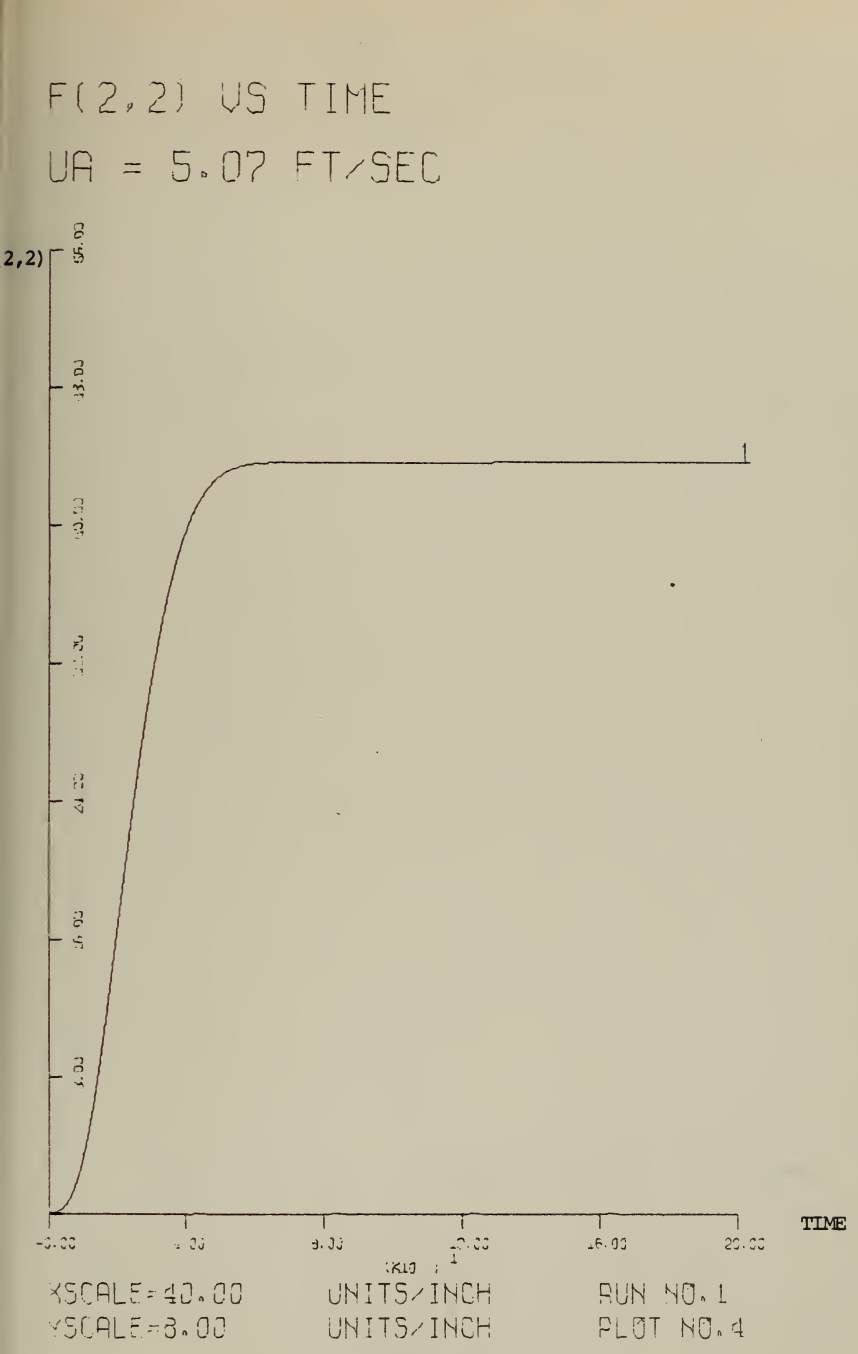

F(2,2) VS TIME
UA = 5.07 FT/SEC
2,2)
TIME
XSCALE=40.00 UNITS/INCH RUN NO. 1
YSCALE=8.00 UNITS/INCH PLOT NO. 4

Figure 4.4

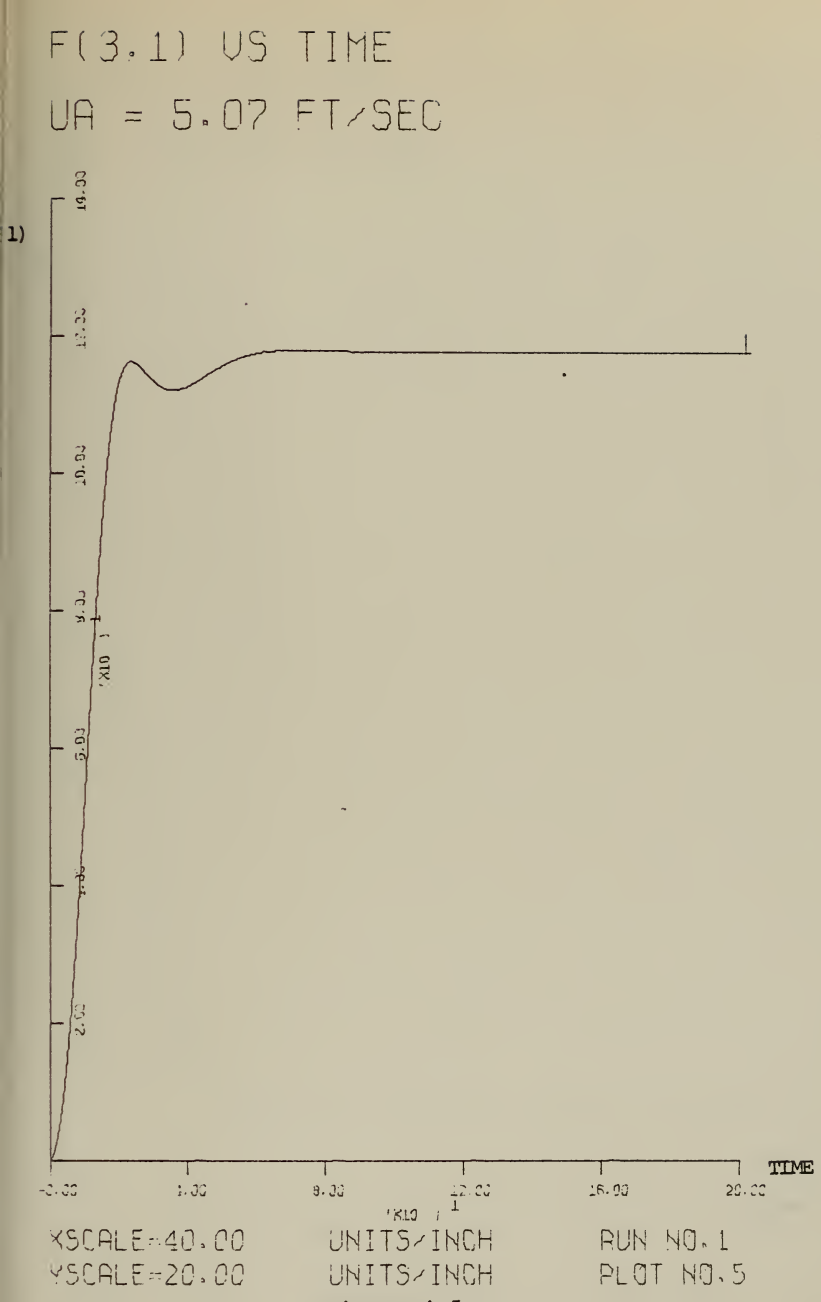

F(3,1) VS TIME
UA = 5.07 FT/SEC
1)
TIME
XSCALE=40.00 UNITS/INCH RUN NO. 1
YSCALE=20.00 UNITS/INCH PLOT NO. 5

Figure 4.5

F(3,2) VS TIME

UA = 5.07 FT/SEC

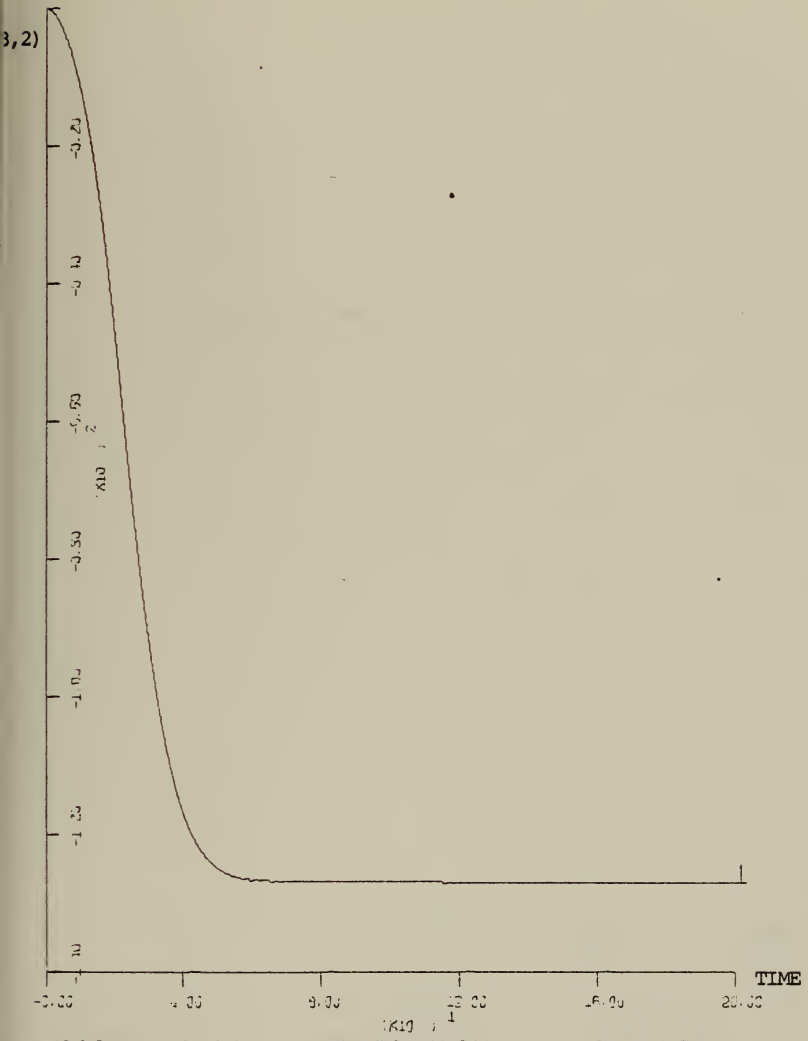


Figure 4.6

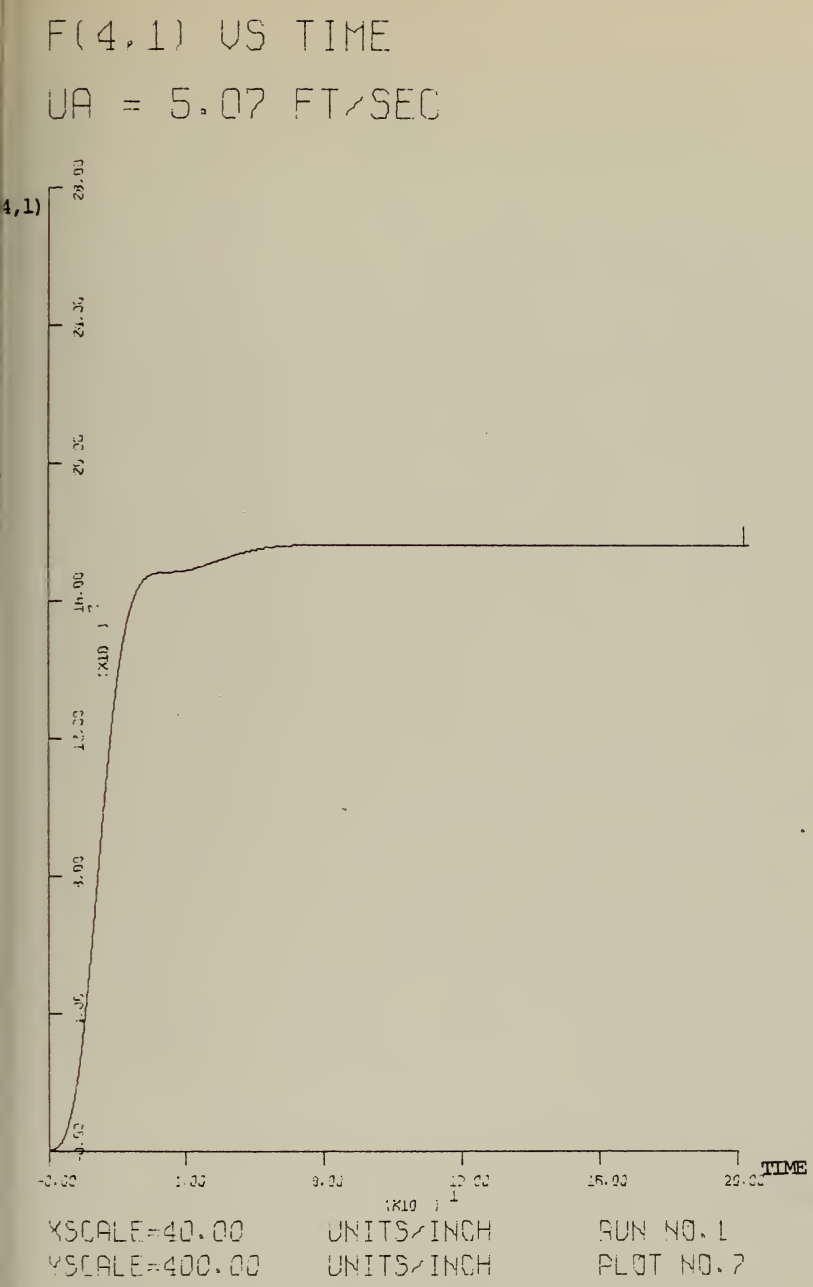

F(4,1) VS TIME
UA = 5.07 FT/SEC
4,1)
TIME
XSCALE=40.00 UNITS/INCH
YSCALE=400.00 UNITS/INCH
RUN NO. 1
PLOT NO. 7

FIGURE 4.7

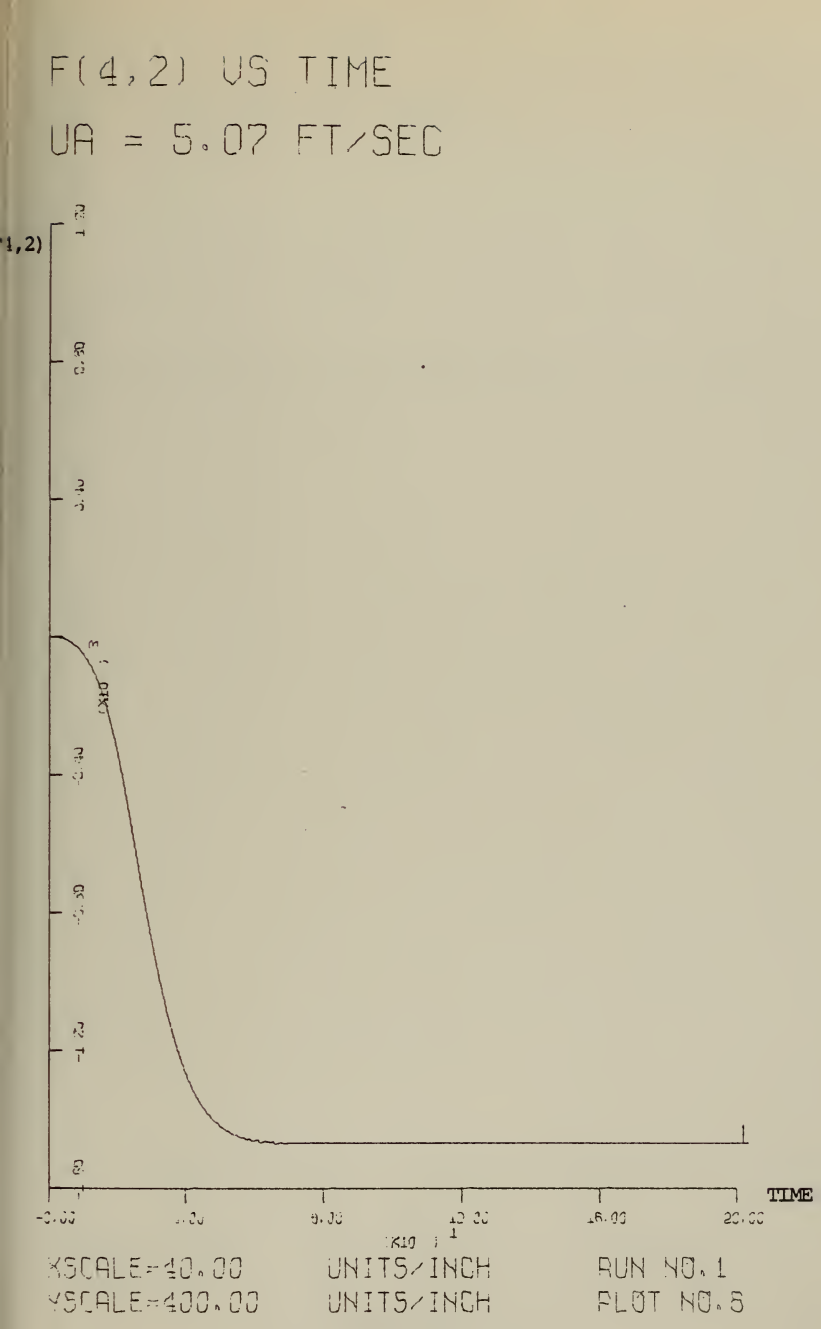

F(4,2) VS TIME
UA = 5.07 FT/SEC
TIME
XSCALE=40.00 UNITS/INCH RUN NO. 1
YSCALE=400.00 UNITS/INCH PLOT NO. 6

FIGURE 4.8

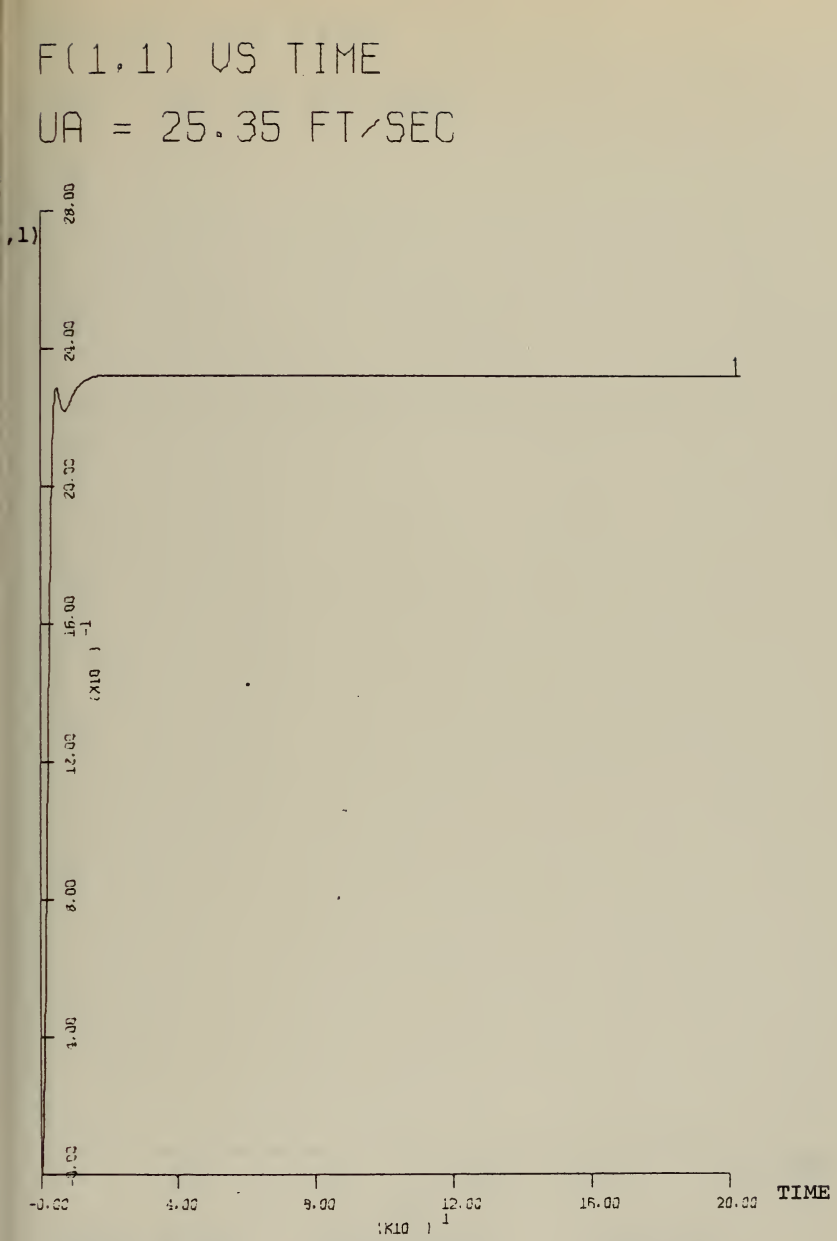

F(1,1) VS TIME
UA = 25.35 FT/SEC
,1)
28.00
24.00
20.00
16.00
12.00
8.00
4.00
0.00
1
(X10)
-1
-0.00
4.00
8.00
12.00
16.00
20.00
TIME
(X10)
1
XSCALE=40.00
UNITS/INCH
RUN NO.1
YSCALE=0.40
UNITS/INCH
PLOT NO.1

FIGURE 5.1

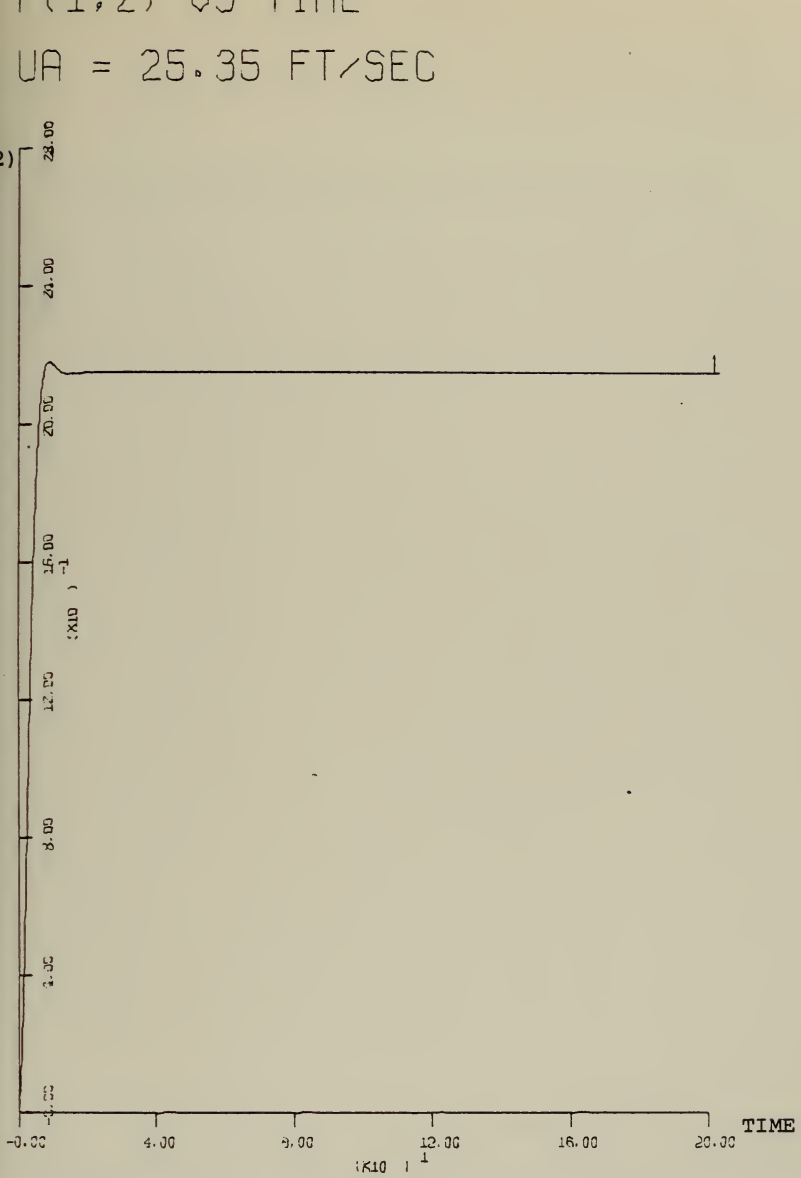

F(1,2) VS TIME
UA = 25.35 FT/SEC
TIME
0.00
4.00
8.00
12.00
16.00
20.00
XSCALE=40.00 UNITS/INCH RUN NO.1
YSCALE=0.40 UNITS/INCH PLOT NO.2

FIGURE 5.2

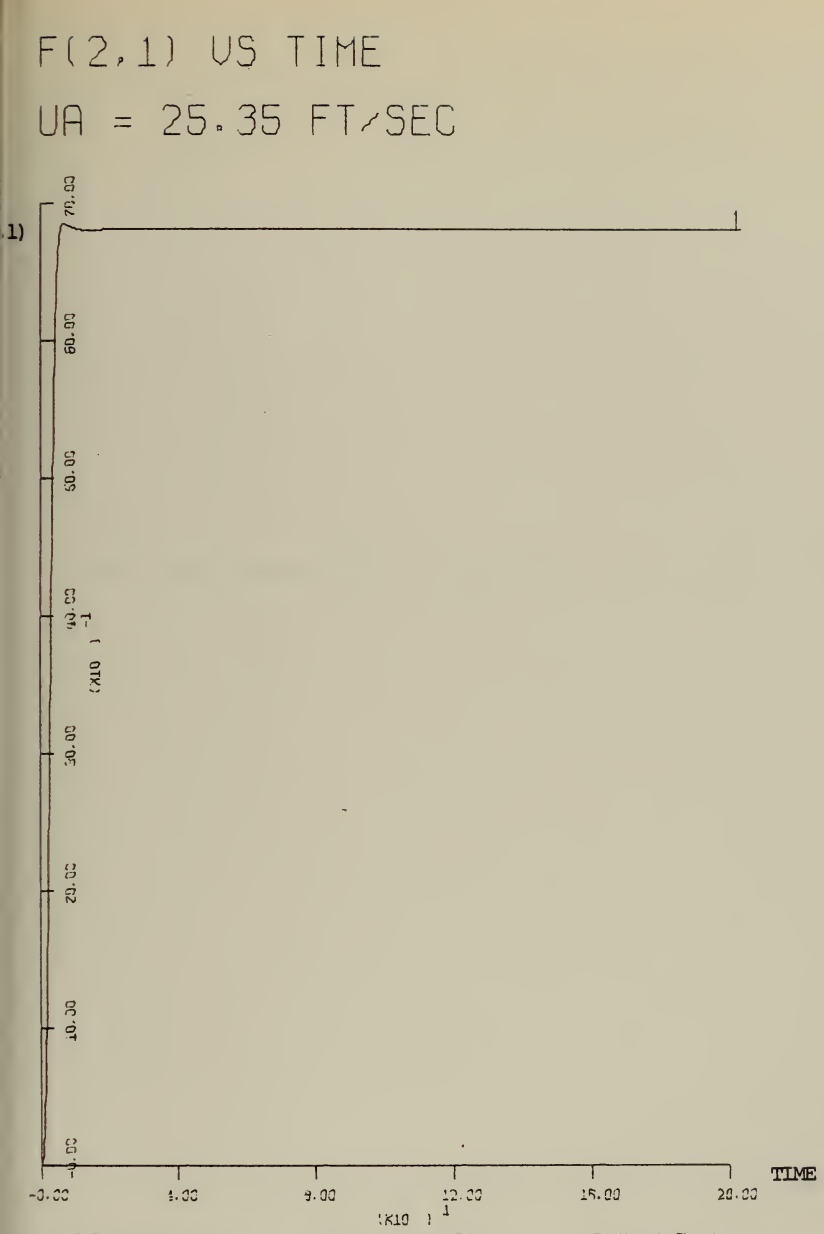

F(2,1) VS TIME
UA = 25.35 FT/SEC
1)
70.00
60.00
50.00
30.00
20.00
10.00
TIME
XSCALE=40.00 UNITS/INCH RUN NO. 1
YSCALE=1.00 UNITS/INCH PLOT NO. 3

FIGURE 5.3

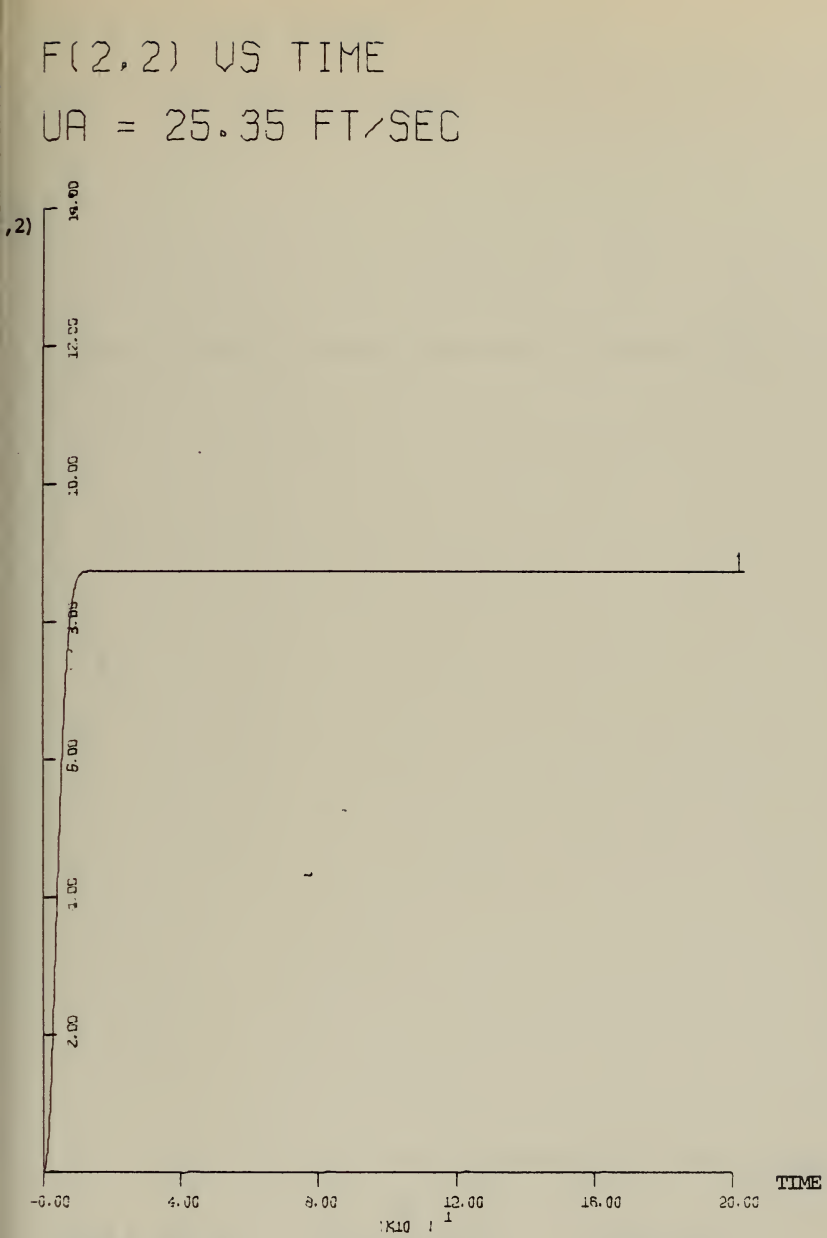
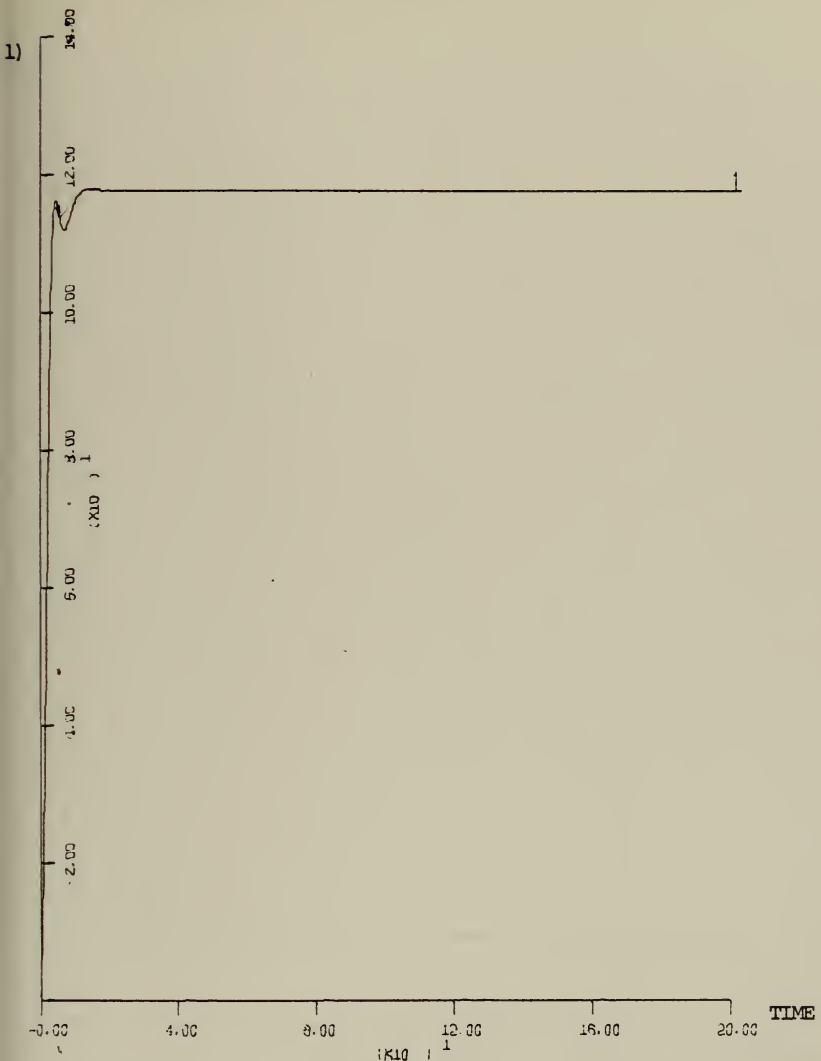

F(2,2) VS TIME
UA = 25.35 FT/SEC
,2)
14.00
12.00
10.00
8.00
6.00
4.00
2.00
1
-0.00
4.00
8.00
12.00
16.00
20.00
TIME
*K10 1
1
XSCALE=40.00 UNITS/INCH RUN NO. 1
YSCALE=2.00 UNITS/INCH PLOT NO. 4

FIGURE 5.4

F(3,1) VS TIME

UA = 25.35 FT/SEC



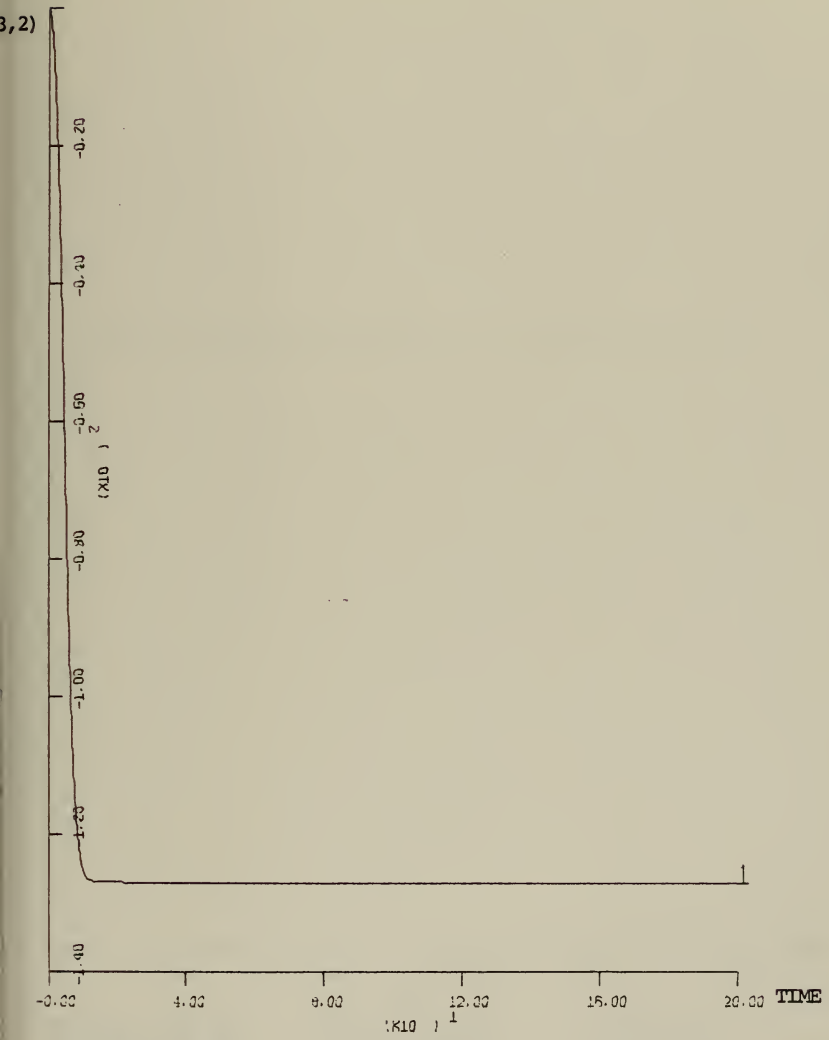
XSCALE=40.00 UNITS/INCH RUN NO.1

YSCALE=20.00 UNITS/INCH PLOT NO.5

FIGURE 5.5

F(3,2) VS TIME

UA = 25.35 FT/SEC



XSCALE=40.00 UNITS/INCH RUN NO. 1

YSCALE=20.00 UNITS/INCH PLOT NO. 6

FIGURE 5.6

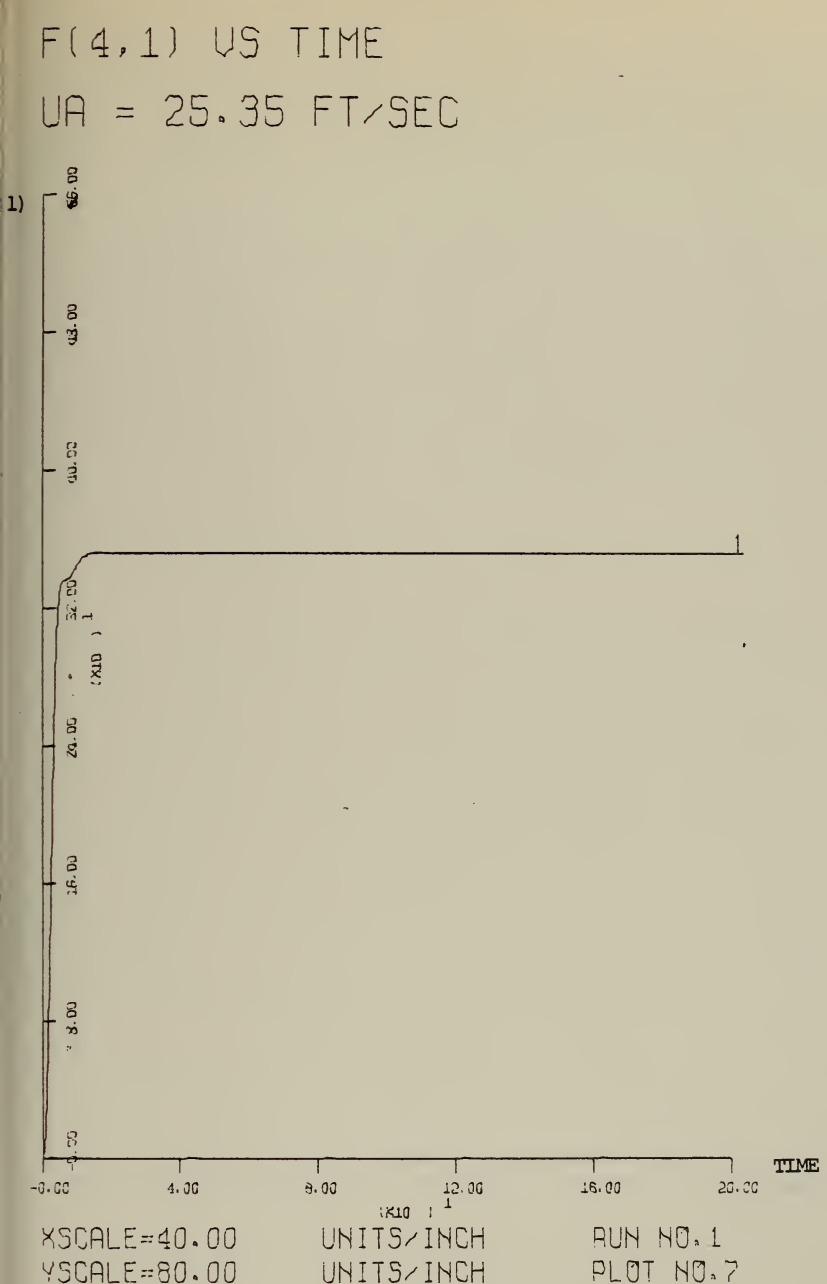

F(4,1) VS TIME
UA = 25.35 FT/SEC
1)
1
TIME
-0.00
4.00
8.00
12.00
16.00
20.00
XSCALE=40.00 UNITS/INCH RUN NO. 1
YSCALE=80.00 UNITS/INCH PLOT NO. 7

FIGURE 5.7

F(4,2) VS TIME

UA = 25.35 FT/SEC

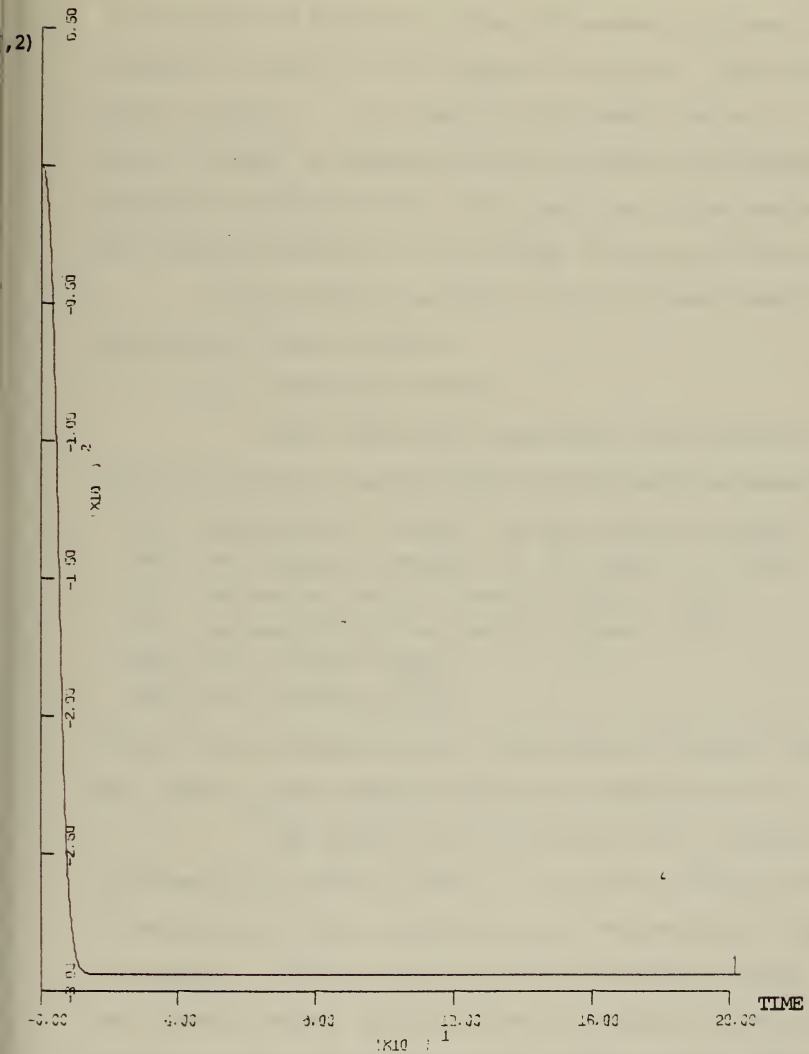


FIGURE 5.8

To change the various command inputs, data cards with the given values are read. The first variable read (TT) is the time of the next change of command followed by the numerical values of the command variables. The format is shown in Fig. 6. At least one data card has to be used. If no changes in command variable values are intended, the specified time TT on the first card has to be one not in the range between 0.0 and FINTIME, the end of the simulation.

At the end of section III.A.2.c some graphs of simulation runs are shown.

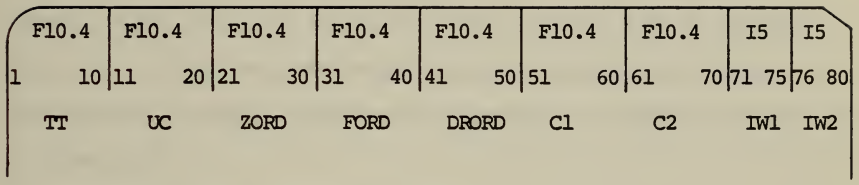
a. Subroutine CNTRL

This subroutine represents the automatic controller and is called with the following arguments:

IW1, the choice of weight for the error in depth;
IW2, the choice of weight for the error in pitch;
C1 , the weight for the control input D(1);
C2 , the weight for the control input D(2);
ZORD, the ordered depth;
PORD, the ordered pitch.

Beside these command inputs, the measured inputs DEPTH, PITCH and speed u_a are passed through the argument list.

The speed u_a is truncated to an integer IU (in increments of 1.689 ft./sec.). With IU, IW1 and IW2 as subscripts for the array FF the gain values F(I,J) are extracted. Together with the errors between actual depth and ordered depth, actual pitch and ordered pitch, rate of change indepth and pitch returned from routine PRIME, the



TT : Time of next change of command variables

UC : Command speed

ZORD : Ordered depth

PORD : Ordered pitch

DRORD : Ordered deflection of rudder
(for the simulation of turns)

C1 : Weight of control input for sterm plane
(range 8.0 to 12.0)

C2 : Weight of control input for fairwater plane
(range 8.0 to 12.0)

IW1 : Choice of weight for error in depth
(range 1, 2, ..., 5)

IW2 : Choice of weight for error in pitch
(range 1, 2, ..., 5)

FIGURE 6

control outputs D(1) and D(2) are computed with equation (13). These outputs are passed back to the simulation over the COMMON-area R1.

Graphical outputs for a change in speed (Figs. 8.1 to 8.5), a turn (Figs. 9.1 to 9.5) and a change in depth (Figs. 10.1 to 10.5) are shown at the end of Section III.A.2. All runs used error channel limiter and plane deflection limits. Each set of graphical outputs consists of a graph for

1) stern plane angle vs. time
2) fairwater plane angle vs. time
3) depth vs. time
4) pitch vs. time
5) speed u_a vs. time.

For a 60 ft. change in depth without constraints, the results are shown in Figs. 11.1 to 11.5. The large deflections of the planes are not desirable and in practice are also not achievable. The reason for this behavior is a non-optimal distribution of the weights for the errors and for the controls. The magnitude of the control outputs has to be smaller. Two possibilities exist to achieve this result: either redistribution of the weights in the weighting matrices or limitation of the control inputs. In this case the control inputs "error in depth" and "error in pitch" have been limited to 4.0 ft. and 10.0 degrees.

The latter approach is a compromise between the decrease of the control outputs and a good response for small errors in depth and pitch.

b. Subroutine PRIME

Instead of taking the rate of range in depth and in pitch from the simulation (ZODOT and PIDOT), this routine is used to keep the controller as independent as possible from the simulation.

To get the rate of change, a linear combination of discrete orthogonal Legendre polynomials $O_{mn}(x)$ has been used to fit a polynomial curve in the least square sense to a set of points with evenly spaced abscissas [Ref. 7]. The discrete Legendre polynomials for a second degree polynomial are

$$O_{0n} = 1$$

$$O_{1n} = 1 - 2\left(\frac{x}{n}\right)$$

$$O_{2n} = 1 - 6\left(\frac{x}{n}\right) + 6\left(\frac{x}{n}\right)\left(\frac{x-1}{n-1}\right) .$$

Using the properties of orthogonal polynomials

$$\sum_{x=0}^{n} O_{in}(x)\, O_{jn}(x) = 0 \quad , \quad i \neq j$$

the normal equations for the least squares fit are

$$(21) \quad \begin{bmatrix} \sum_{x=0}^{n} O_{0n}^{2}(x) & 0 & 0 \\ 0 & \sum_{x=0}^{n} O_{1n}^{2}(x) & 0 \\ 0 & 0 & \sum_{x=0}^{n} O_{2n}^{2}(x) \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{x=0}^{n} O_{0n}(x) y_x \\ \sum_{x=0}^{n} O_{1n}(x) y_x \\ \sum_{x=0}^{n} O_{2n}(x) y_x \end{bmatrix}$$

where the y_x represent the sampled data points.[2] The linear combination of Legendre polynomials which gives the least squares fit is

$$(22) \quad F(x) = a_0 O_{0n}(x) + a_1 O_{1n}(x) + a_2 O_{2n}(x)$$

$$= a_0[1] + a_1[1 - 2\frac{x}{n}] + a_2[1 - 6\frac{x}{n} + 6\frac{x(x-1)}{n(n-1)}] .$$

Transforming the set of evenly spaced points x' into integer x

$$(23) \quad x = \frac{x' - Z}{h} \quad \text{with} \quad h = x'_{i+1} - x'_i = \text{constant}$$

h = DELS in program implementation.

Taking five points (n=4) and evaluating the polynomial at the third point (Z=0.4) the values of the orthogonal polynomials are

x	O_{04}	O_{14}	O_{24}
0	1.0	1.0	1.0
1	1.0	0.5	-0.5
2	1.0	0.0	-1.0
3	1.0	-0.5	-0.5
4	1.0	-1.0	1.0

and

$$\sum_{x=0}^{4} O_{04}{}^2(x) = 5 \qquad \sum_{x=0}^{4} O_{14}{}^2(x) = 2.5 \qquad \sum_{x=0}^{4} O_{24}{}^2(x) = 3.5$$

---

[2] x denotes an integer 0, 1, 2.

From (21) then, using the above numerical values

$$a_0 = \frac{1}{5} (y_0 + y_1 + y_2 + y_3 + y_4)$$

$$a_1 = \frac{1}{5} (2y_0 + y_1 - y_3 - 2y_4)$$

$$a_2 = \frac{1}{7} (2y_0 - y_1 - 2y_2 - y_3 + 2y_4)$$

and (22) becomes

$$F(x) = a_0 + a_1 + a_2 - (\frac{a_1}{2} + 2a_2)x + \frac{a_2}{2} x^2 \quad .$$

The derivative of F(x) is

$$F'(x) = - \frac{1}{2} a_1 - 2a_2 + a_2 x \quad .$$

Replacing a_0, a_1, a_2 and using (23) gives

$$F'(x') = - \frac{1}{70} (54y_0 - 13y_1 - 40y_2 - 27y_3 + 26y_4)$$

$$+ \frac{1}{7h} (2y_0 - y_1 - 2y_2 - y_3 + 2y_4)(x' + 2.0)$$

If x' is the time of sampling, h the sampling interval and y_x the sampled data points, then F'(x') is the rate of change in F at x'.

c. Subroutine PLGEN

This routine is called three times in the derivative region for the simulation of the movements of fairwater and stern planes and the rudder. The time between

the repeated calls is determined by the length of the calculation interval. The routine is called with the following arguments:

(1) deflection of the plane or rudder;

(2) difference between deflection and ordered deflection from the controller;

(3) increment or decrement of deflection computed in the previous call;

(4) rate of change in the plane movement;

(5) length of time interval, in this case DELT * 0.25, the length of the calculation interval.

The increment in the plane or rudder deflection computed in the previous call is added to the current plane or rudder deflection unless the increment is smaller than the dead zone of $\pm$ 0.5 degrees. A limiter limits the maximum plane or rudder deflection to 36.0 degrees. A flow diagram is shown in Figure 7.

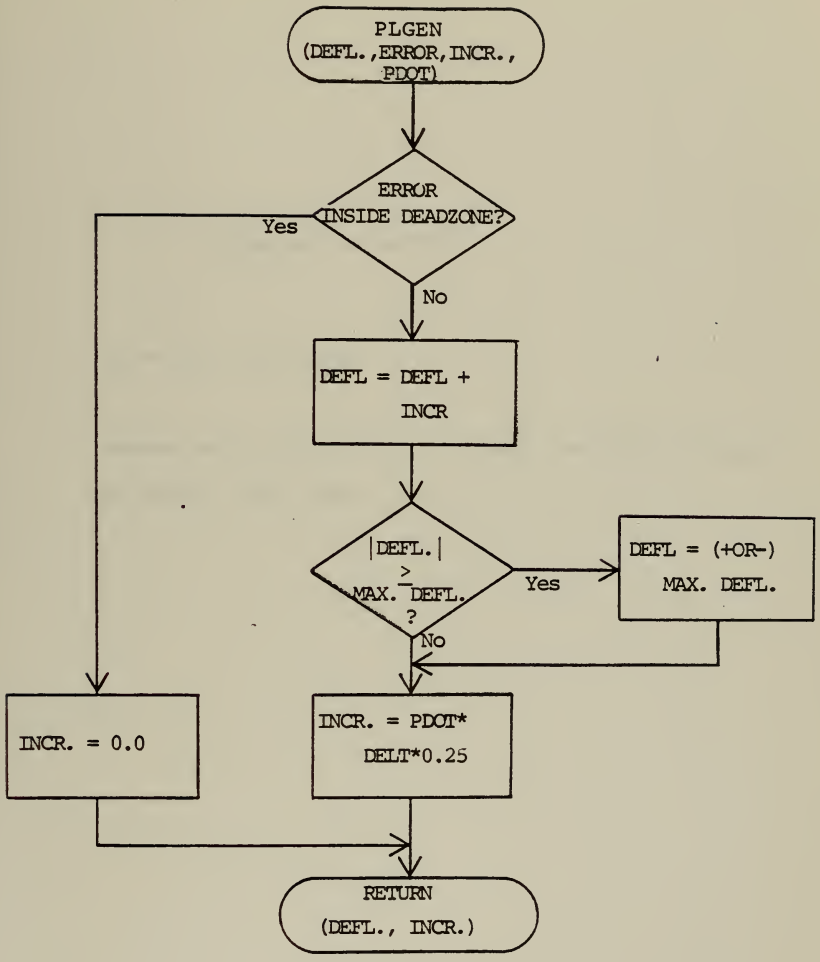

PLGEN
(DEFL.,ERROR,INCR.,
PDOT)
ERROR
INSIDE DEADZONE?
Yes
No
DEFL = DEFL +
INCR
|DEFL.|
>
MAX. DEFL.
?
Yes
DEFL = (+OR-)
MAX. DEFL.
No
INCR. = 0.0
INCR. = PDOT*
DELT*0.25
RETURN
(DEFL., INCR.)

Figure 7

Figures 8.1 to 8.5

u_c = 25.34 ft./sec.

change of u_c from 25.34 ft./sec. to 15.21 ft./sec.
at TIME = 50.0 sec.

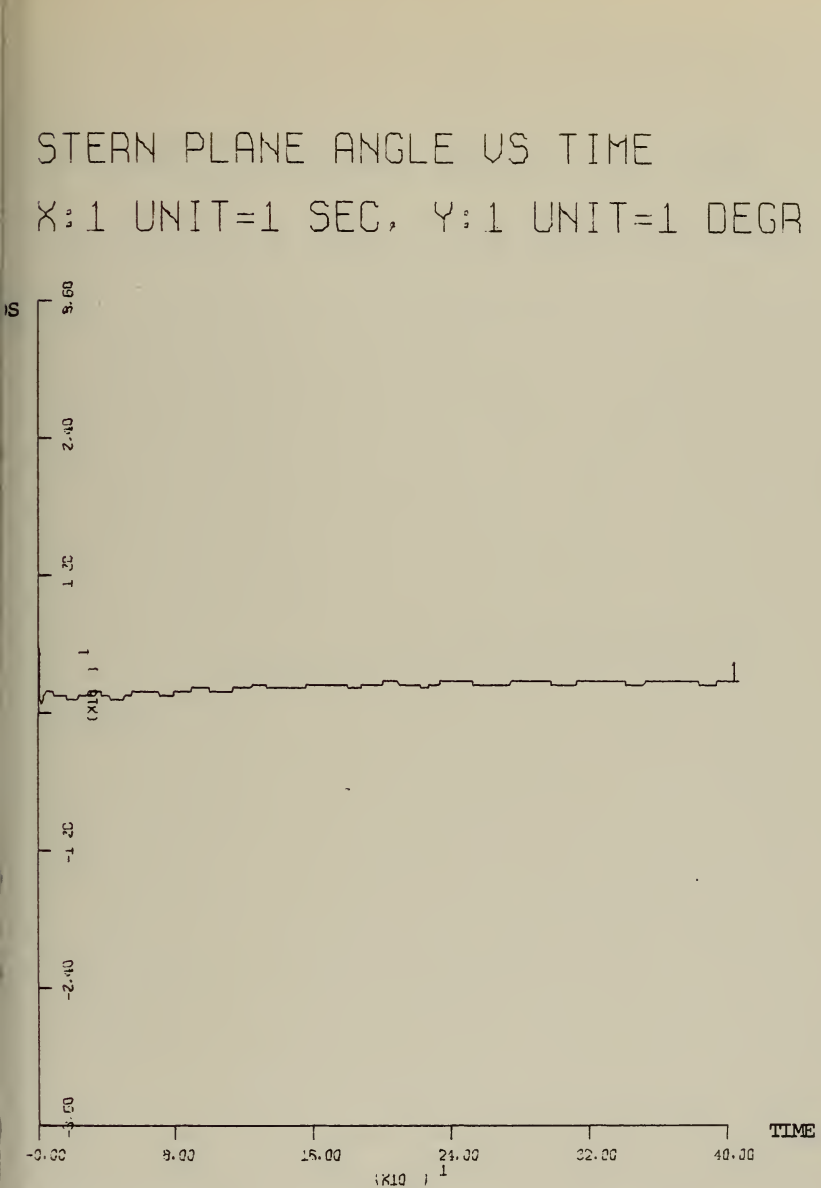

STERN PLANE ANGLE VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 DEGR
TIME
XSCALE=80.00 UNITS/INCH RUN NO. 1
YSCALE=12.00 UNITS/INCH PLOT NO. 1

FIGURE 8.1

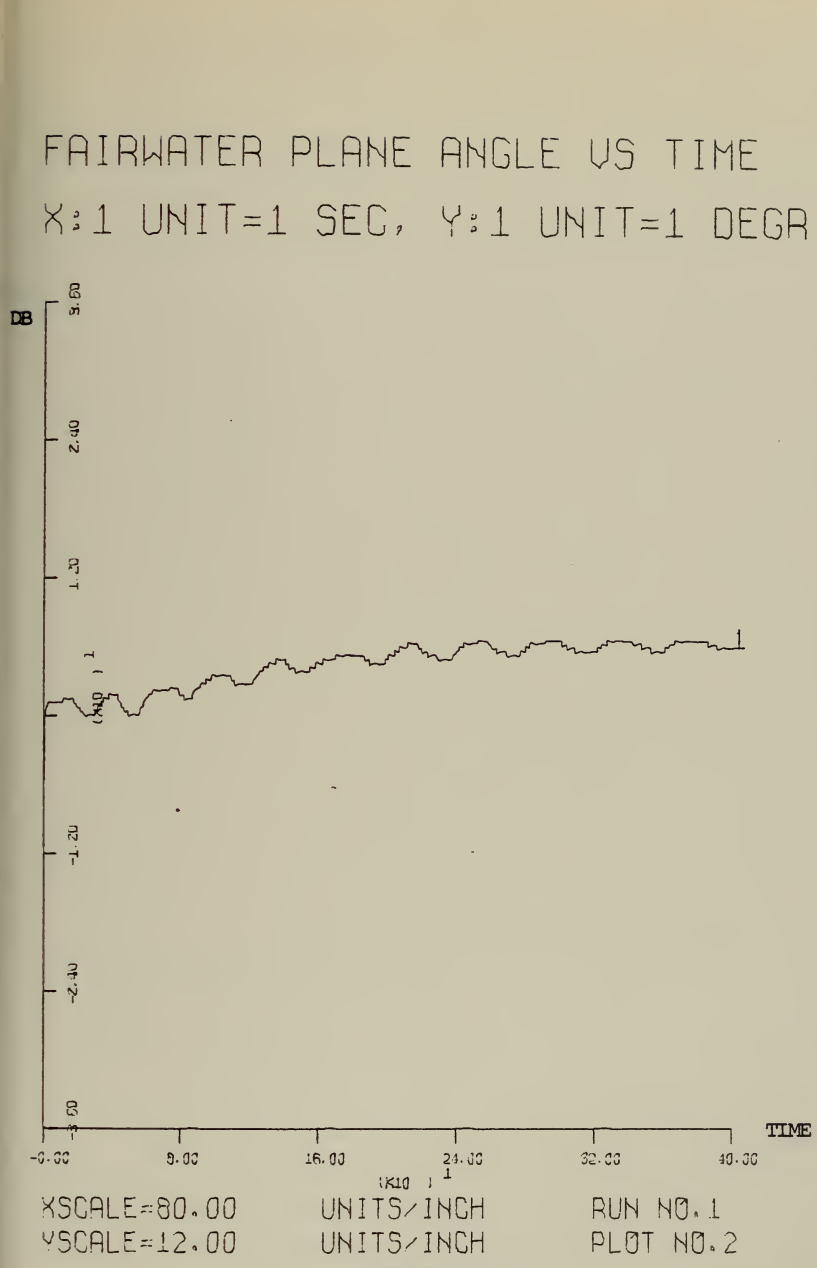

FAIRWATER PLANE ANGLE VS TIME
X:1 UNIT=1 SEC, Y:1 UNIT=1 DEGR
DB
3.60
2.40
1.20
-1.20
-2.40
TIME
-0.00
8.00
16.00
24.00
32.00
40.00
XSCALE=80.00 UNITS/INCH
YSCALE=12.00 UNITS/INCH
RUN NO. 1
PLOT NO. 2

FIGURE 8.2

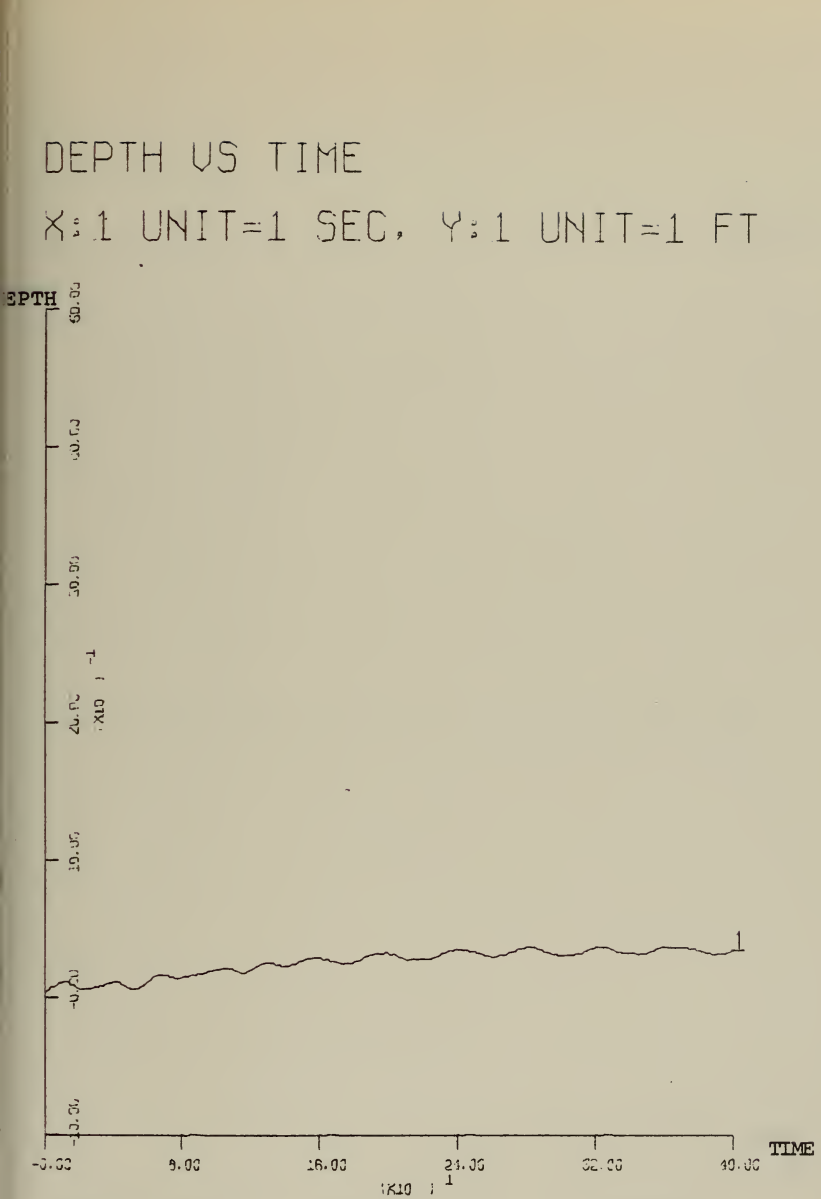
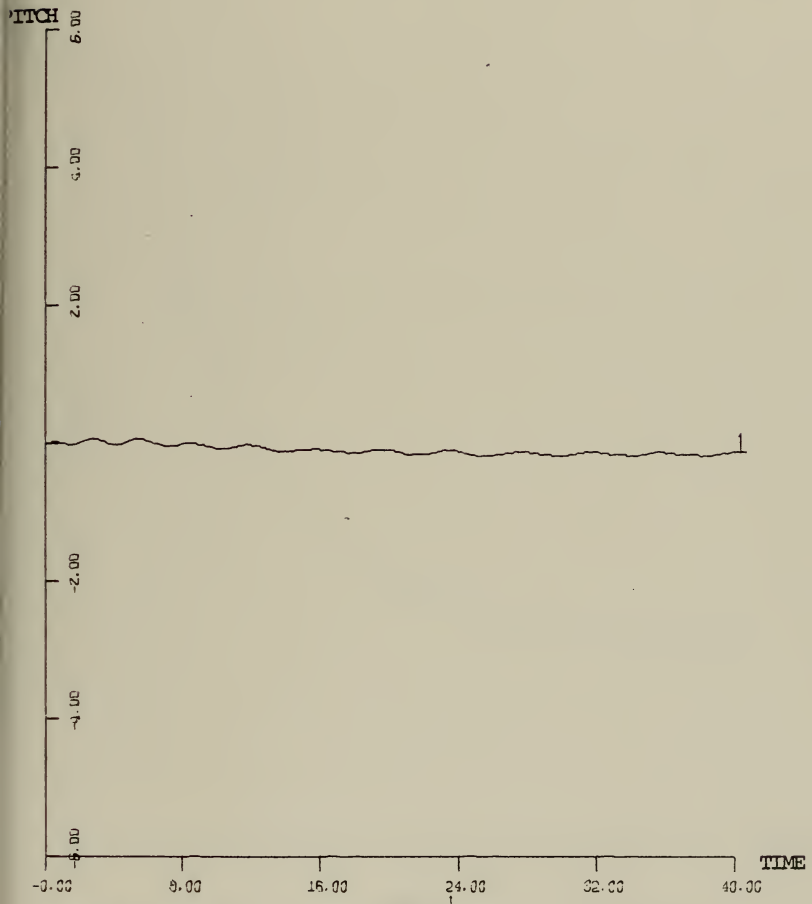

DEPTH VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 FT
DEPTH
TIME
1
XSCALE=80.00 UNITS/INCH RUN NO. 1
YSCALE=1.00 UNITS/INCH PLOT NO. 3

FIGURE 8.3

PITCH VS TIME

X: 1 UNIT=1 SEC, Y: 1 UNIT=1 DEGR



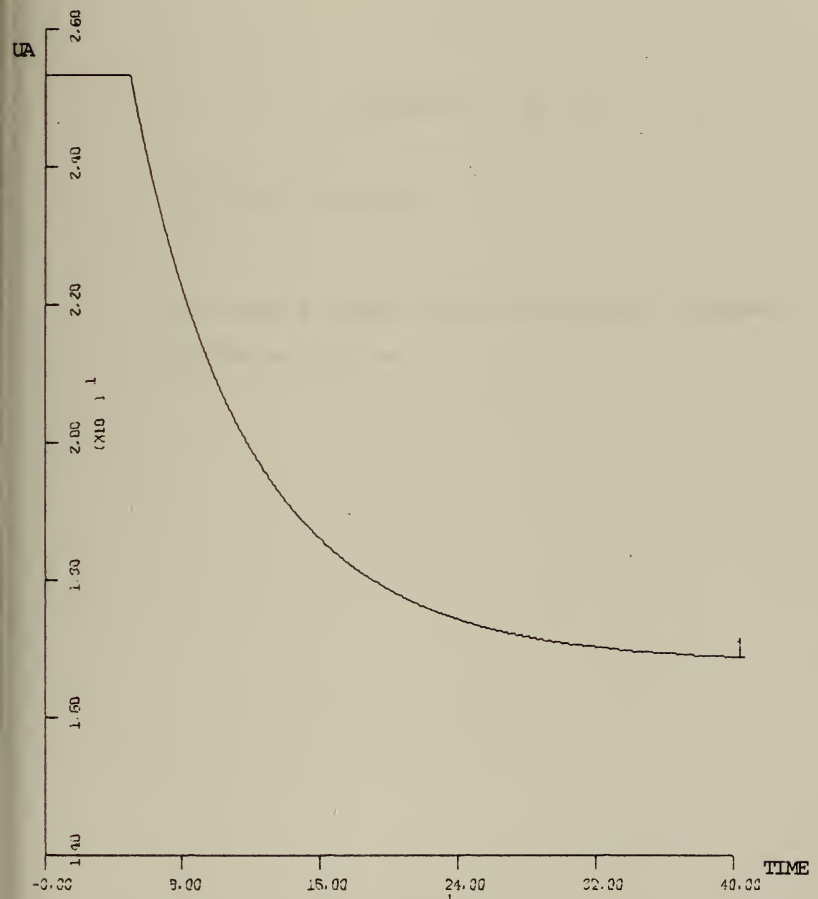
XSCALE=80.00 UNITS/INCH RUN NO. 1

YSCALE=2.00 UNITS/INCH PLOT NO. 4

FIGURE 8.4

SPEED UA VS TIME

X: 1 UNIT=1 SEC, Y: 1 UNIT=1 FT/SEC



XSCALE=80.00 UNITS/INCH RUN NO. 1

YSCALE=2.00 UNITS/INCH PLOT NO. 5

FIGURE 8.5

Figures 9.1 to 9.5

u_c = 25.34 ft./sec.

turn with a rudder deflection of 20.0 degrees

at TIME = 50.0 sec.

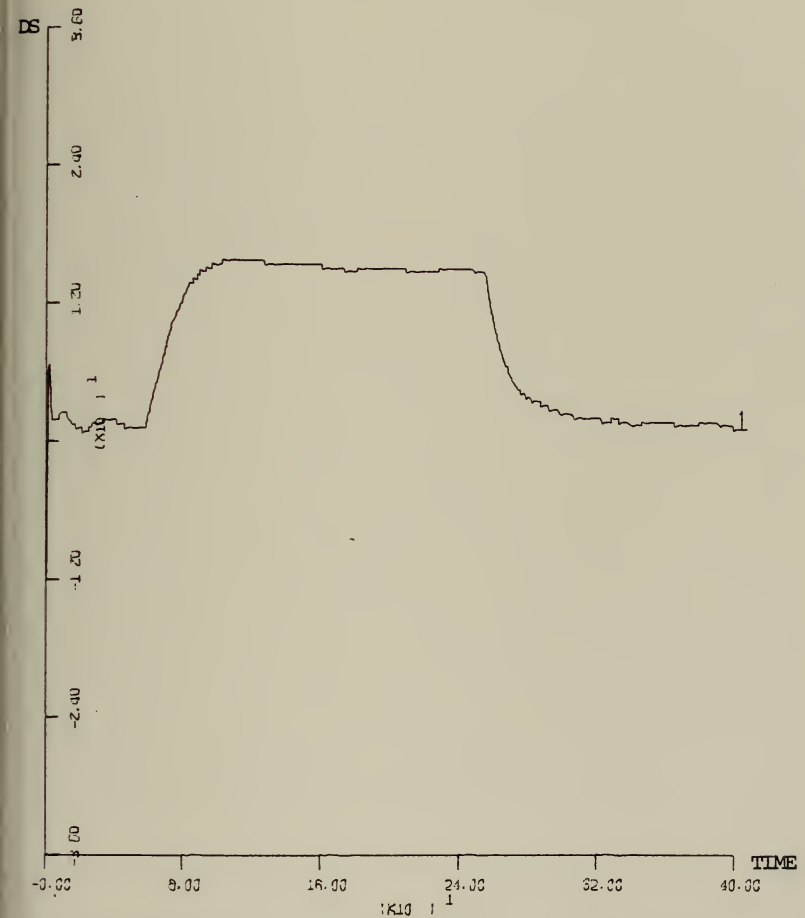

STERN PLANE ANGLE VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 DEGR
DS
TIME
-0.00
8.00
16.00
24.00
32.00
40.00
XSCALE=80.00 UNITS/INCH RUN NO.1
YSCALE=12.00 UNITS/INCH PLOT NO.1

FIGURE 9.1

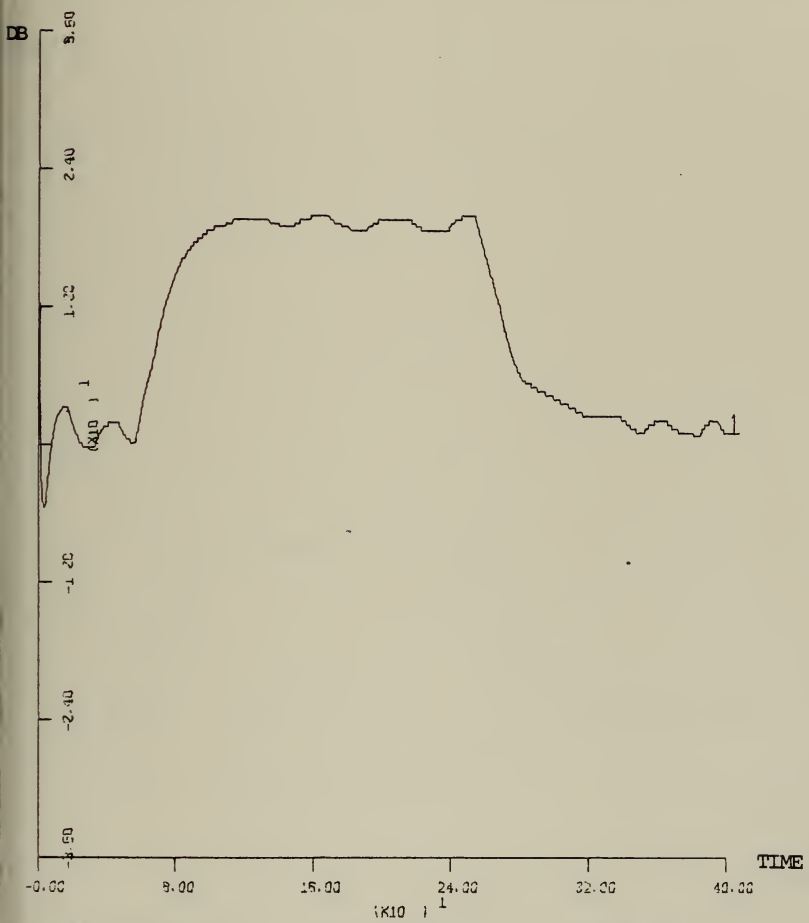

FAIRWATER PLANE ANGLE VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 DEGR
DB
3.60
2.40
1.20
(X10) 1
-1.20
-2.40
-3.60
TIME
-0.00
8.00
16.00
24.00
32.00
40.00
(X10) 1
XSCALE=80.00 UNITS/INCH RUN NO. 1
YSCALE=12.00 UNITS/INCH PLOT NO. 2

FIGURE 9.2

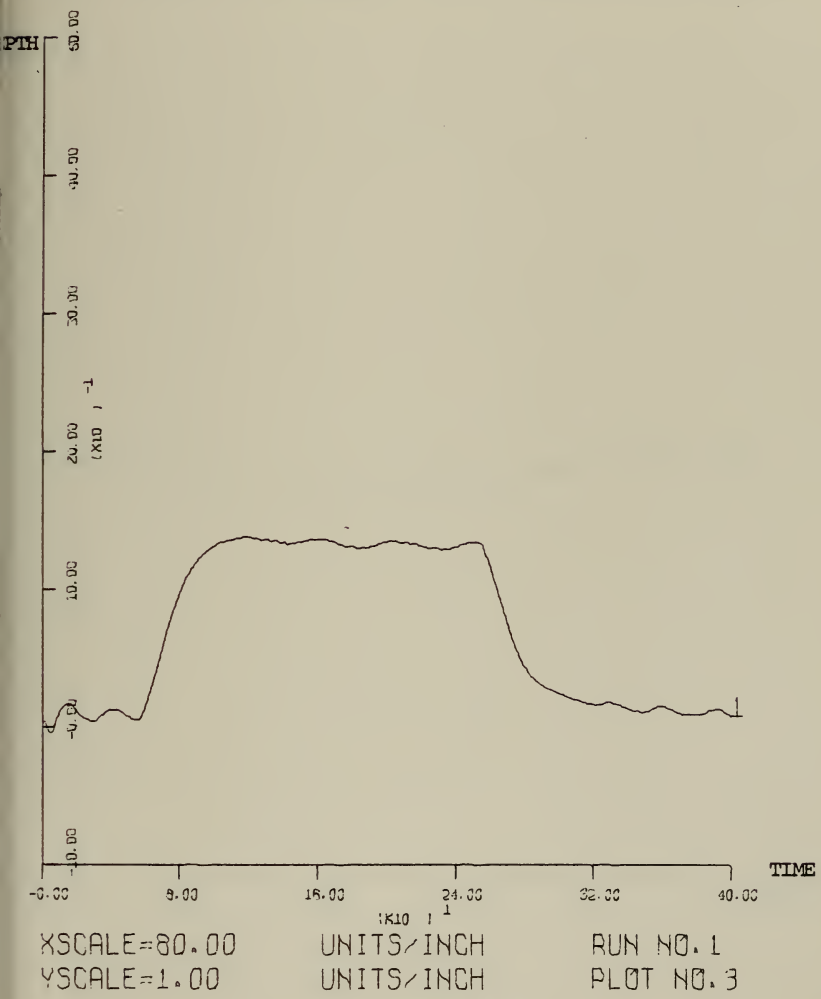

DEPTH VS TIME
X:1 UNIT=1 SEC, Y:1 UNIT=1 FT
PTH
50.00
30.00
20.00
10.00
-0.00
-10.00
TIME
-0.00
8.00
16.00
24.00
32.00
40.00
XSCALE=80.00 UNITS/INCH RUN NO.1
YSCALE=1.00 UNITS/INCH PLOT NO.3

FIGURE 9.3

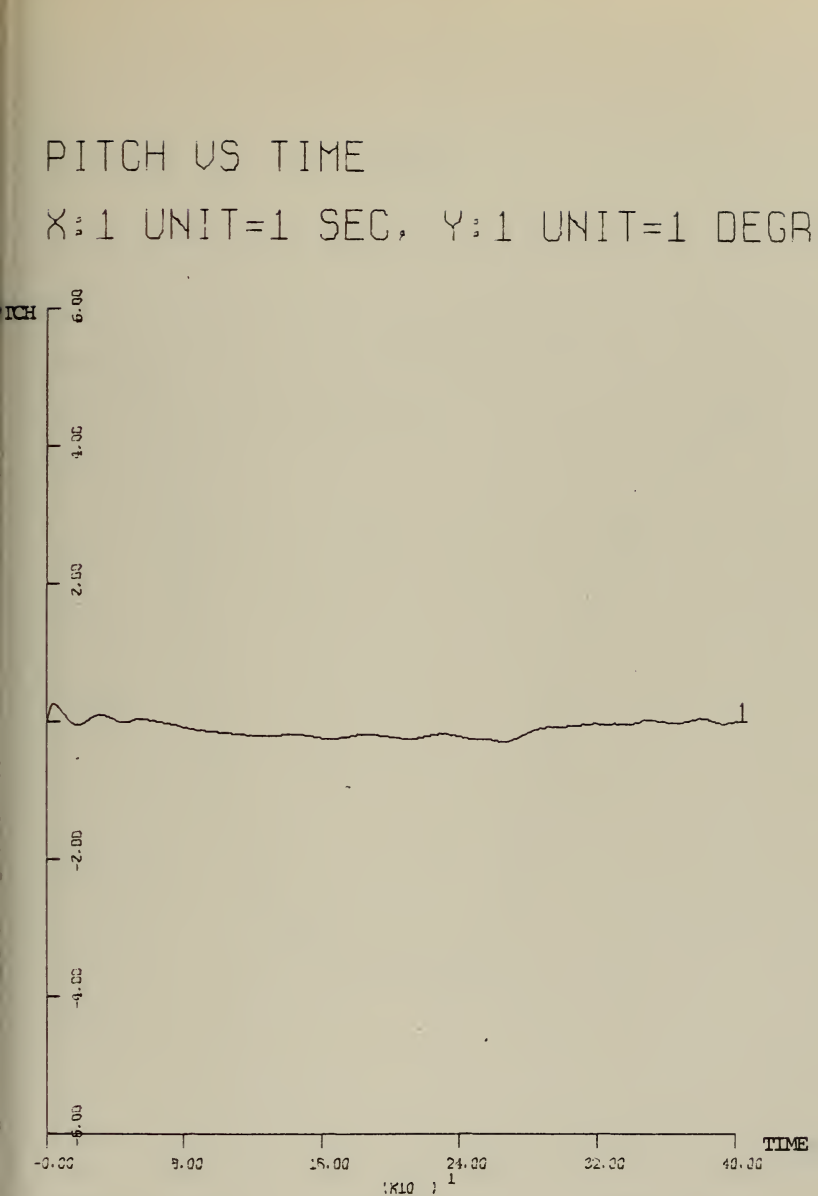
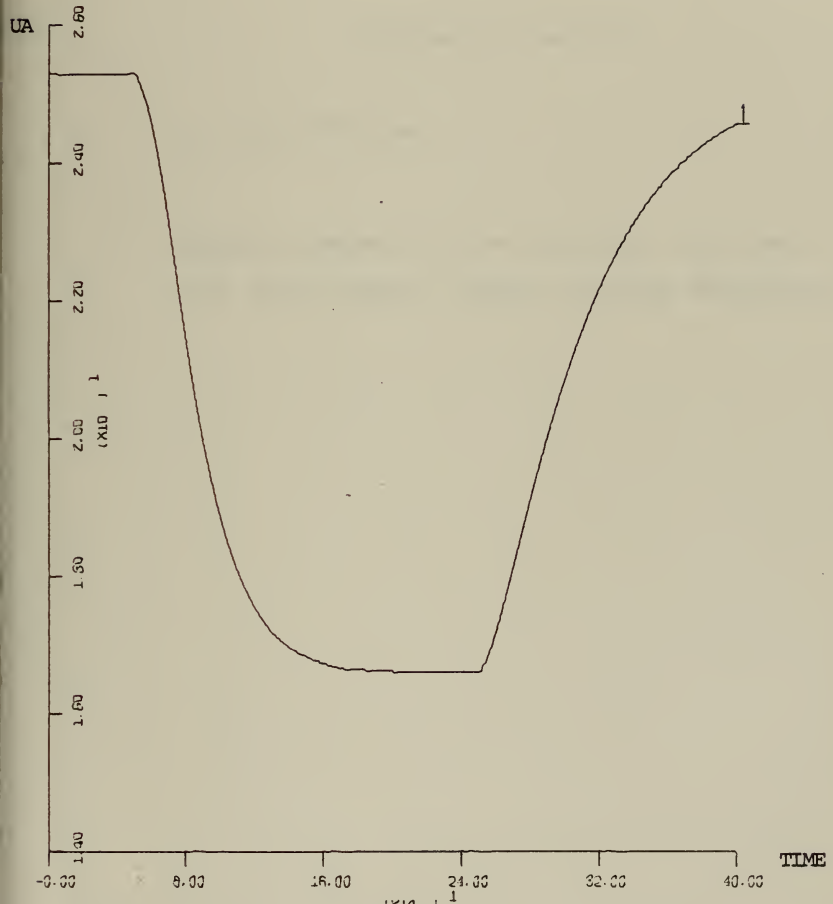

PITCH VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 DEGR
ITCH
6.00
4.00
2.00
-2.00
-4.00
-6.00
1
TIME
-0.00
8.00
16.00
24.00
32.00
40.00
(X10)
1
XSCALE=80.00 UNITS/INCH RUN NO.1
YSCALE=2.00 UNITS/INCH PLOT NO.4

FIGURE 9.4

SPEED UA VS TIME

X: 1 UNIT=1 SEC, Y: 1 UNIT=1 FT/SEC



XSCALE=80.00 UNITS/INCH RUN NO. 1

YSCALE=2.00 UNITS/INCH PLOT NO. 5

FIGURE 9.5

Figures 10.1 to 10.5

u_c = 25.34 ft./sec.

change of depth by 60 ft. at TIME = 100.0 sec.

with error channel limiter and plane deflection limits

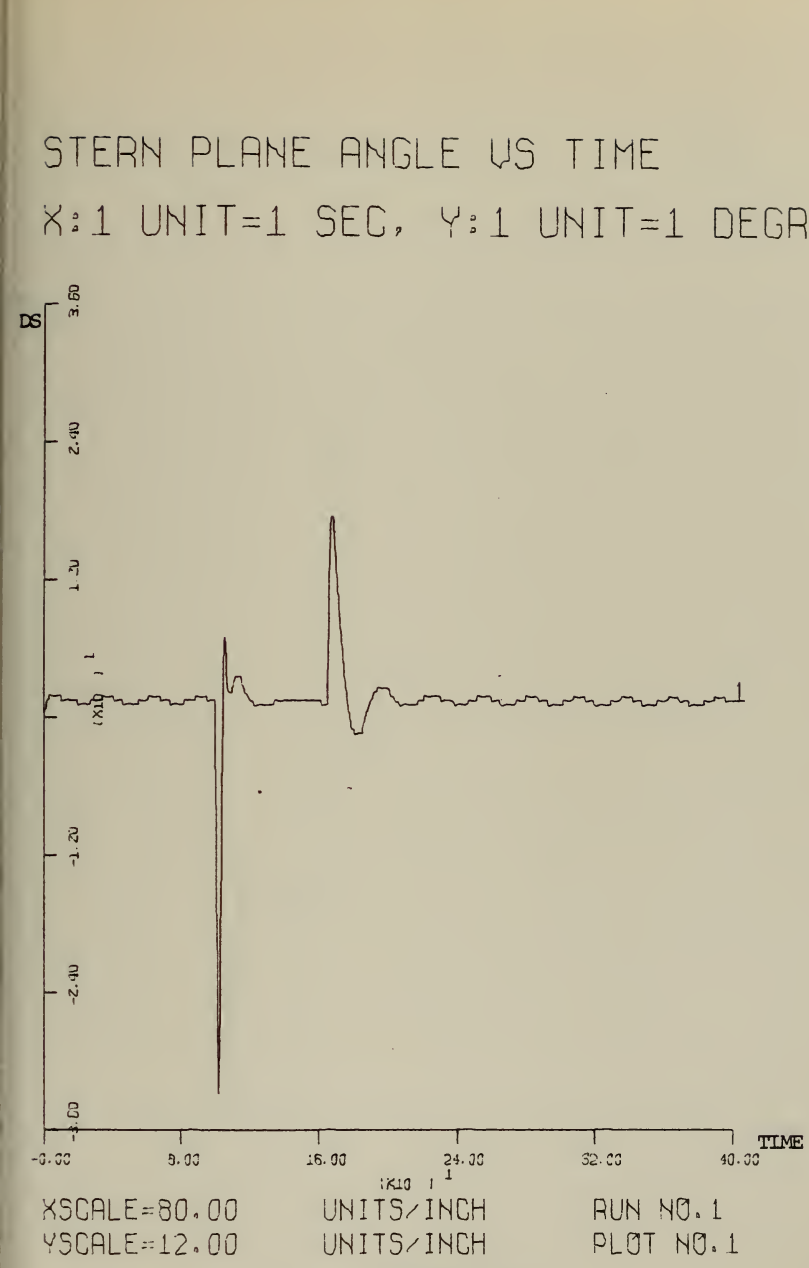

STERN PLANE ANGLE VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 DEGR
DS
3.60
2.40
1.20
-1.20
-2.40
-3.60
TIME
-0.00
8.00
16.00
24.00
32.00
40.00
XSCALE=80.00 UNITS/INCH RUN NO.1
YSCALE=12.00 UNITS/INCH PLOT NO.1

FIGURE 10.1

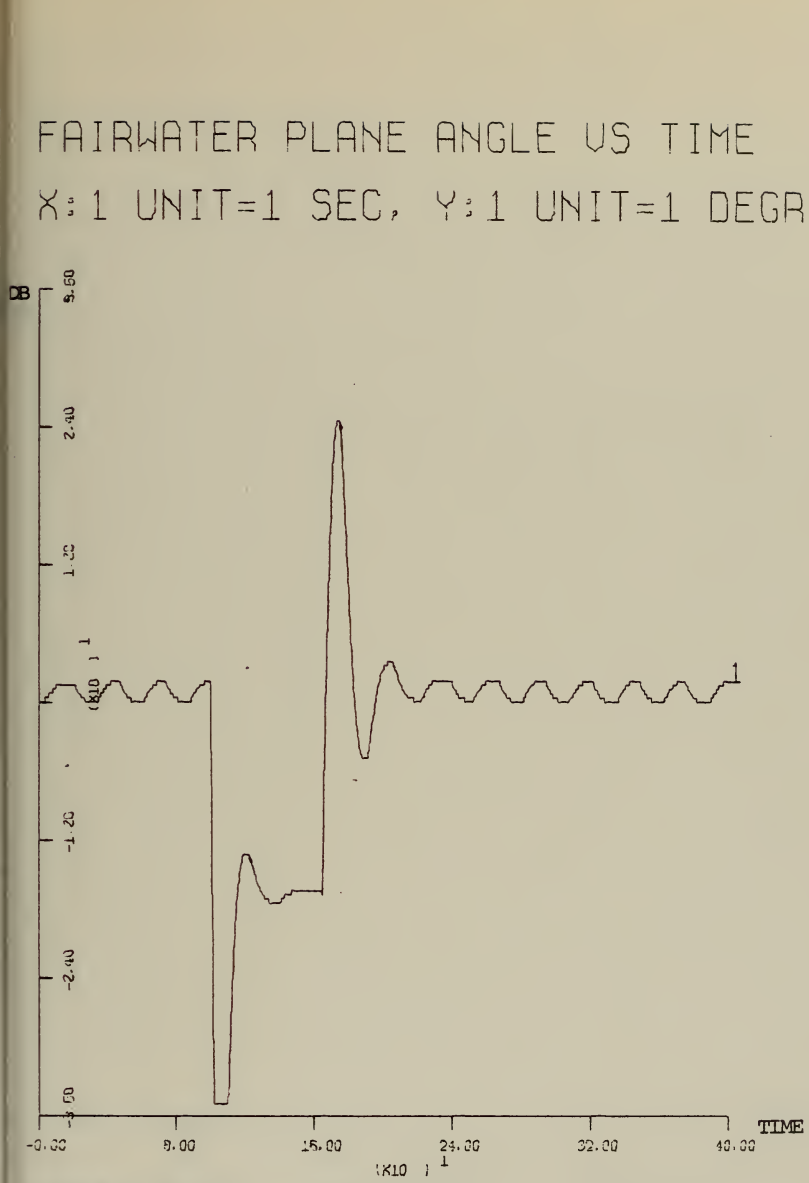

FAIRWATER PLANE ANGLE VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 DEGR
DB
3.60
2.40
1.20
-1.20
-2.40
-3.60
TIME
-0.00
8.00
16.00
24.00
32.00
40.00
XSCALE=80.00 UNITS/INCH RUN NO. 1
YSCALE=12.00 UNITS/INCH PLOT NO. 2

FIGURE 10.2

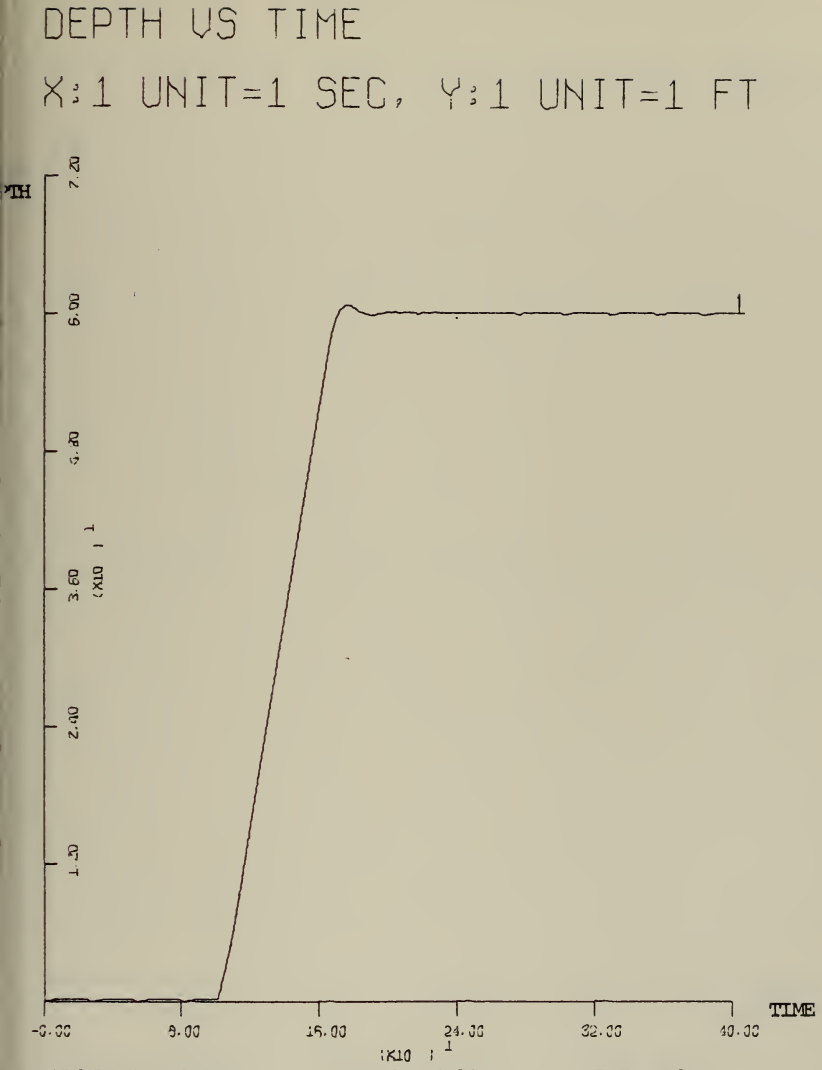

DEPTH VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 FT
DEPTH
7.20
6.00
4.80
3.60
2.40
1.20
(X10) 1
1
TIME
-0.00
8.00
16.00
24.00
32.00
40.00
(X10) 1
XSCALE=80.00 UNITS/INCH RUN NO. 1
YSCALE=12.00 UNITS/INCH PLOT NO. 3

FIGURE 10.3

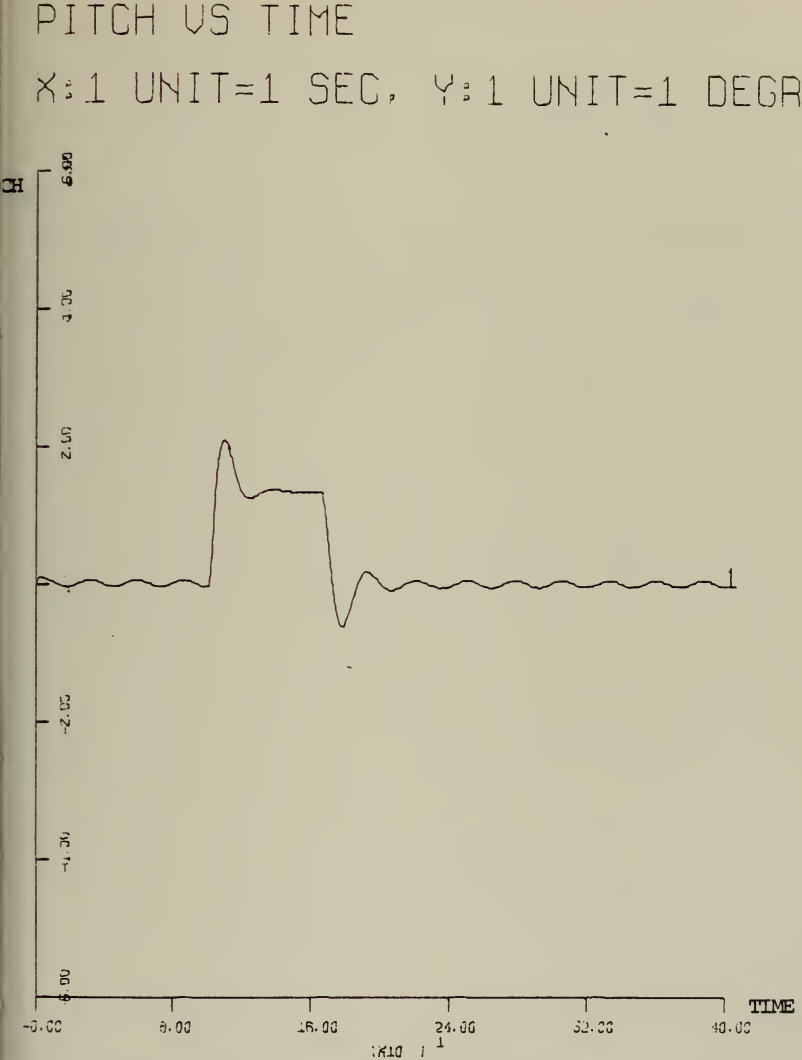
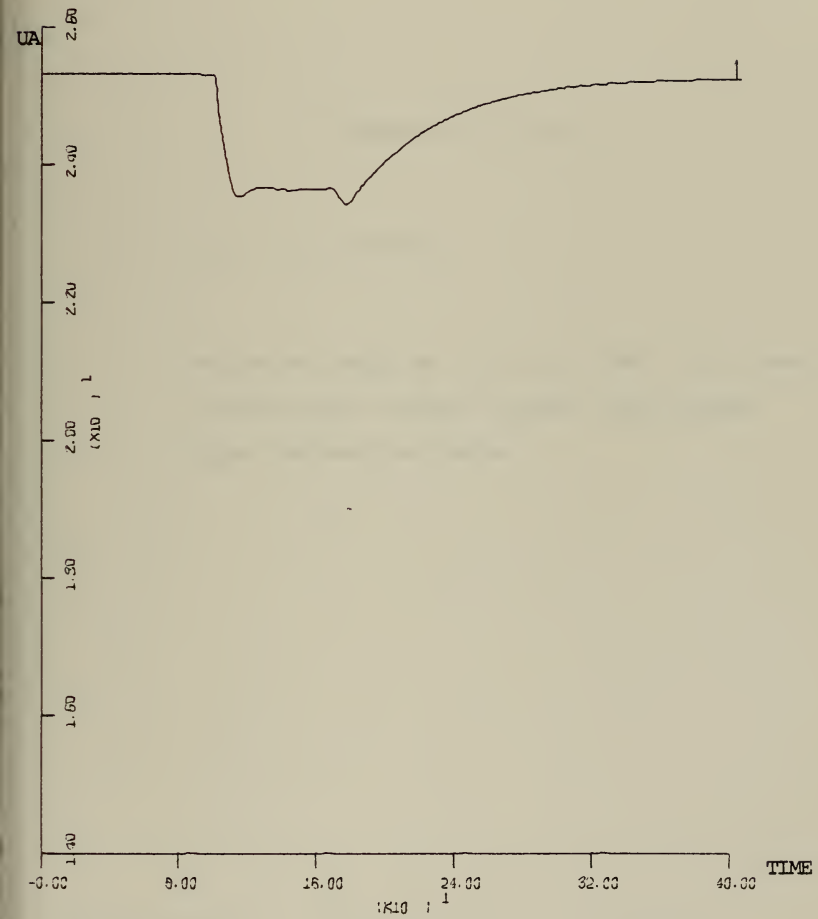

PITCH VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 DEGR
CH
2.00
-2.00
TIME
-0.00
8.00
16.00
24.00
32.00
40.00
XSCALE=80.00 UNITS/INCH RUN NO. 1
YSCALE=2.00 UNITS/INCH PLOT NO. 4

FIGURE 10.4

SPEED UA VS TIME

X: 1 UNIT=1 SEC, Y: 1 UNIT=1 FT/SEC



XSCALE=80.00 UNITS/INCH RUN NO. 1

YSCALE=2.00 UNITS/INCH PLOT NO. 5

FIGURE 10.5

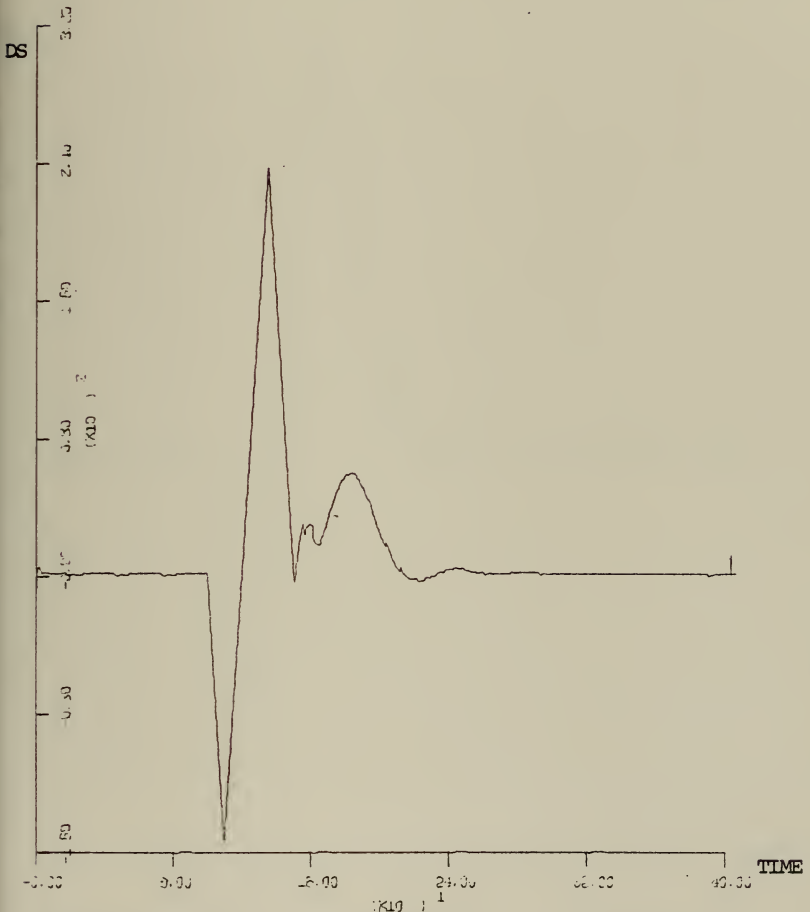
Figures 11.1 to 11.5

u_c = 25.34 ft./sec.

change of depth by 60 ft. at TIME = 100.0 sec.
without error channel limiter and without
plane deflection limits

STERN PLANE ANGLE VS TIME

X=1 UNIT=1 SEC, Y=1 UNIT=1 DEGR



XSCALE=80.00 UNITS/INCH RUN NO.1

YSCALE=80.00 UNITS/INCH PLOT NO.1

FIGURE 11.1

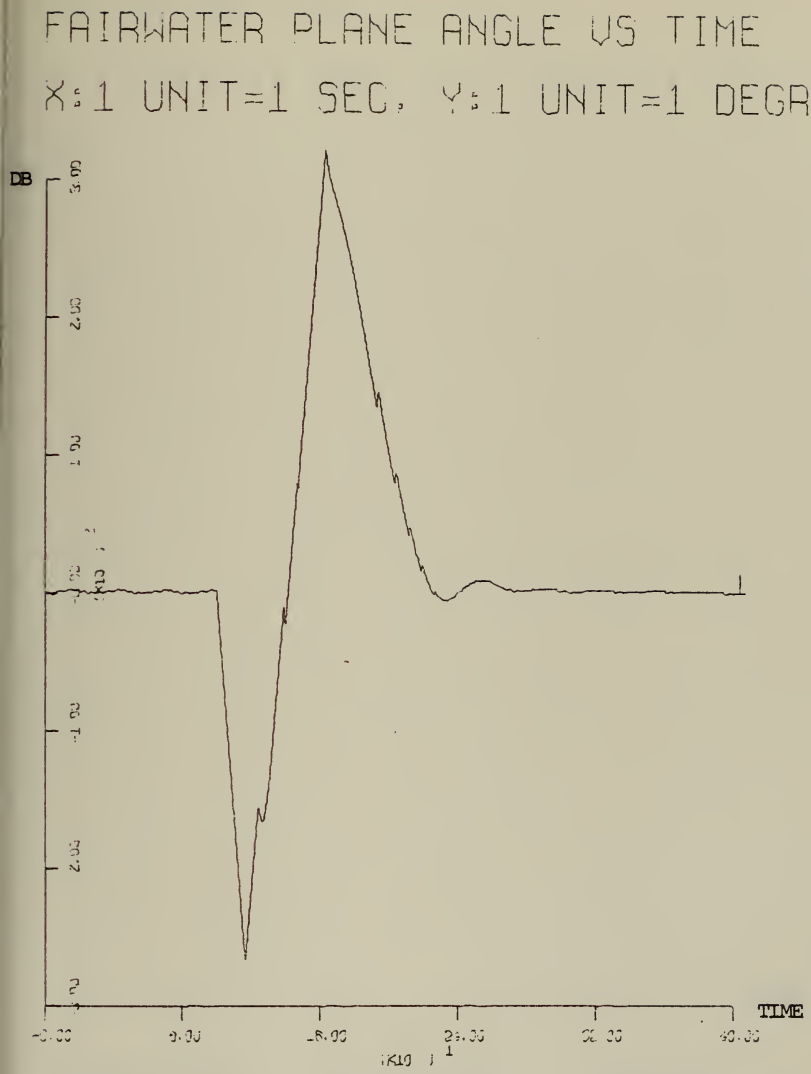

FAIRWATER PLANE ANGLE VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 DEGR
DB
TIME
XSCALE=80.00 UNITS/INCH RUN NO. 1
YSCALE=100.00 UNITS/INCH PLOT NO. 2

FIGURE 11.2

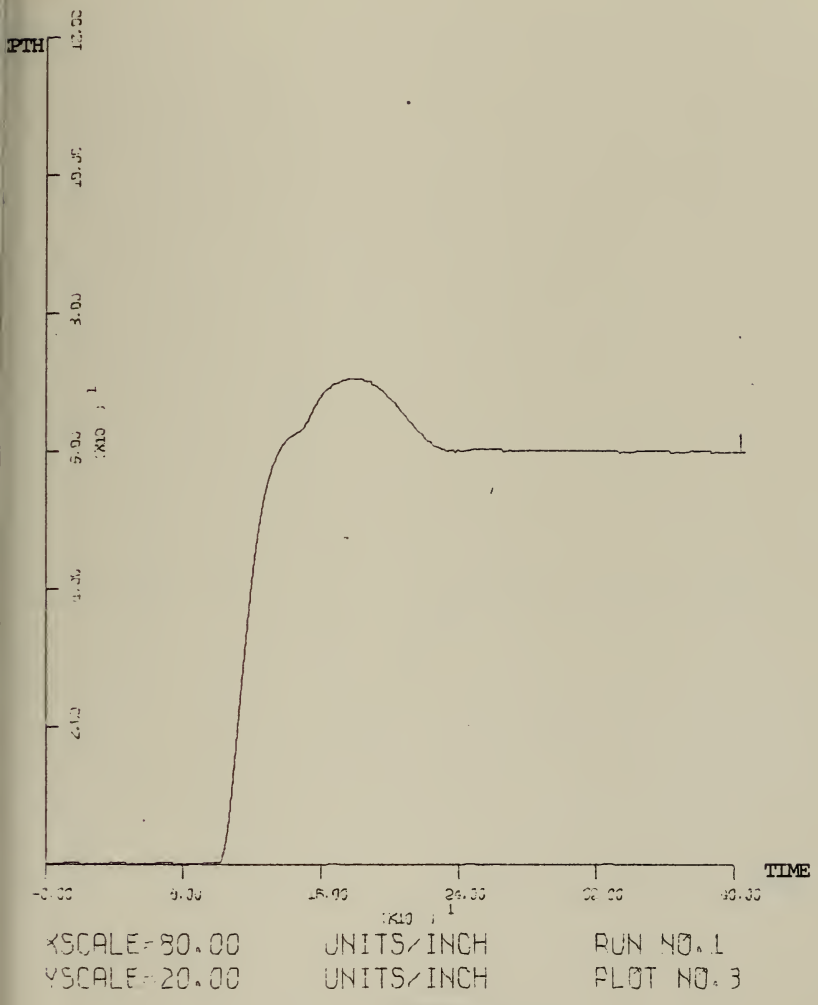

DEPTH VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 FT
TIME
XSCALE=80.00 UNITS/INCH RUN NO. 1
YSCALE=20.00 UNITS/INCH PLOT NO. 3

FIGURE 11.3

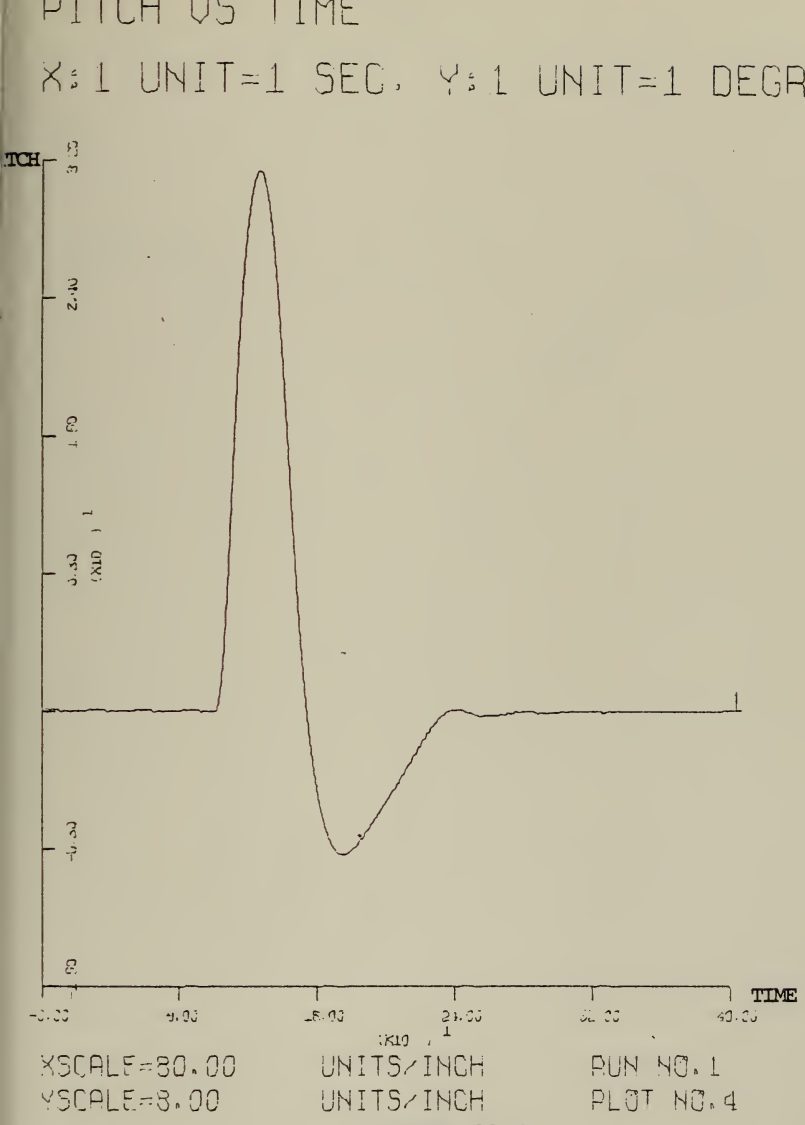

PITCH VS TIME
X: 1 UNIT=1 SEC. Y: 1 UNIT=1 DEGR
TCH
TIME
XSCALE=80.00 UNITS/INCH RUN NO. 1
YSCALE=8.00 UNITS/INCH PLOT NO. 4

FIGURE 11.4

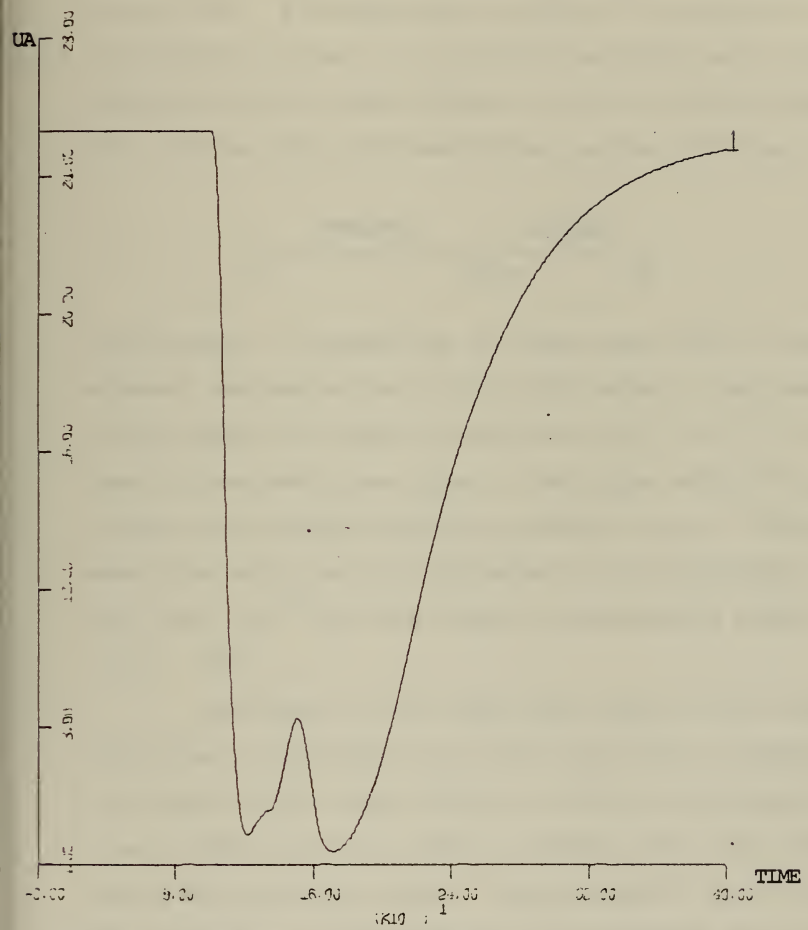
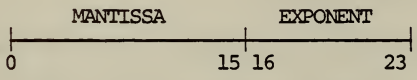

SPEED UA VS TIME
X: 1 UNIT=1 SEC, Y: 1 UNIT=1 FT/SEC
UA
TIME
1
XSCALE=30.00 UNITS/INCH RUN NO. 1
YSCALE=4.00 UNITS/INCH PLOT NO. 5

FIGURE 11.5

B. PLM - PROGRAM

1. Data Formats and Conversions

The hardware of the INTEL 8080 microcomputer is restricted to integer arithmetic. The basic wordlength is one byte. Floating point arithmetic is possible by means of software routines. The control algorithm uses routines for multiplication and addition in floating point format. The floating point representation has the following form.



The mantissa is normalized, the high order bit in the exponent represents the sign of the number. The lower half of the remaining range of the seven bits (0 to $2^6 - 1$) is used for exponents less than 0, the upper half (2^6 to $2^7 - 1$) is used for exponents equal or greater than 0. The smallest magnitude which can be represented in floating point format is $1.8446 \cdot 10^{-19}$ and the largest representable number is $9.2233 \cdot 10^{18}$.

The format of the input and output of the control algorithm is contained in a 16 bit word which represents an integer in the range from 0 to 65535. The range between 0 and 32768 represents negative values, the range over 32768 represents positive values. The contents of this 16 bit word is assumed to come from an A/D-converter and to go to a D/A-converter. The range of the 16-bit integer representation corresponds to:

depth: 0 to 32767 inches $\triangleq$ 32768 to 65535

(0 to 32767 not used)

pitch: 16.0 degrees to +16.0 degrees $\triangleq$ 0 to 65535

1.0 degrees $\triangleq$ 2048

$4.883 \cdot 10^{-4}$ degrees $\triangleq$ 1

plane deflection: -36.0 degrees to +36.0 degrees

$\triangleq$ 0 to 65535

1.0 degrees $\triangleq$ 910 approx.

1.0987 degrees $\triangleq$ 1

There are two routines for the necessary conversions:

- routine FLOAT converts from integer to floating point format.
- routine INTEG converts from floating point to integer format.

The use of the time consuming arithmetic software routines has been limited to the minimum necessary. All computations which could be done outside the PLM-program are done beforehand.

The results then are used in the PLM-program. Where possible, comparisons and arithmetic manipulations have been done with the faster integer format. This results in an execution time of ca. 0.12 sec. for one pass through the control algorithm. It is important that the time for one pass does not exceed 0.2 sec. Otherwise the sampling interval for depth and pitch becomes too large and the controller does not work properly. This observation results from the

DSL/360 simulation. The PLM-version of the controller consists only of one pass through the algorithm and only of two sets of gain values in the F array. This was considered to be sufficient for demonstrating the feasibility. In an actual implementation the F-array had to have the same number of elements as in the DSL-programs. Furthermore, the controller would have to work in an infinite loop with fixed intervals for reading the input values from the input ports. This sampling would replace parts of the input routines, conversion routines and input statements.

2. Error Analysis:

A floating point mantissa consists of 16 binary digits, or approximately five decimal digits. Starting with the assumption that the magnitude of the error connected with each number consists only of roundoff errors, then the maximum error per number can be $\frac{1}{2} \cdot 2^{1-16} = 2^{-16}$.

As an example, the absolute error bounds are computed for D(1) and Y(2) (see Appendix B). The results are:

$$
\begin{aligned}
|De(1)| \leq\ & |C6 \cdot Y(3) \cdot F(4,1)(\Delta 6 + \delta 1 + \alpha 1 + \alpha 2 + \alpha 3 + m1 + m2) \\
& + F(3,1) \cdot Y(4)(f5 + Ye(2) + m3 + \alpha 1 + \alpha 2 + \alpha 3) \\
& + F(2,1) \cdot Y(2)(f3 + Ye(4) + m4 + \alpha 2 + \alpha 3) \\
& + F(1,1) \cdot Y(1)(f1 + \delta 2 + m5 + \alpha 3)|
\end{aligned}
$$

and

$$|Ye(2)| \leq C1 \cdot d1(\Delta 1 + \delta 1 + m1 + \alpha 1 + \alpha 2 + \alpha 3 + \sigma)$$
$$+ C2 \cdot d2(\Delta 2 + \delta 2 + m2 + \alpha 1 + \alpha 2 + \alpha 3 + \sigma)$$
$$+ C4 \cdot d4(\Delta 4 + \delta 4 + m4 + \alpha 2 + \alpha 3 + \sigma)$$
$$+ C5 \cdot d5(\Delta 5 + \delta 5 + m5 + \alpha 3 + \sigma)$$
$$+ C3 \cdot d3(\Delta 3 + \delta 3 + m3 + \sigma)$$

where

Δi = roundoff error in coefficient for Legendre Polynomial Ci, i = 1,...,5;

δi = inherent error in measurement of depth or pitch, i = 1,...,5;

αi = roundoff error after addition i;

mi = roundoff error after multiplication i;

σ = roundoff error after subtraction;

fi = roundoff error in gain F(I,J);

C6 = normalizing factor to convert the pitch value from a 16-bit integer word to a compatible value with respect to depth.

With

$|\text{mantissa}|$ of Y(1), Y(2), F(I,J), C1 to C6 < 1.0

and

$$|\alpha i|, |\Delta i|, |mi|, |fi|, |\sigma| \leq 2^{-16}$$

the two bounds for the absolute errors become

$$\max|De(1)| \leq 20 \cdot 2^{-16} + Ye(1) + Ye(2)$$
$$\leq 1.25 \cdot 2^{-12} + Ye(1) + Ye(2)$$

and $\max|Ye(2)| \leq 29 \cdot 2^{-16}$.

Assuming for Ye(1) the same expression and combining these inequalities yields

$$\max|De(1)| \leq 1.25 \cdot 2^{-12} + 2 \cdot 29 \cdot 2^{-16}$$

$$\leq 0.001190 \equiv 0.068 \text{ degrees.}$$

The maximum computational error due to the 16 bit floating point arithmetic of 0.068 degrees is below the accuracy of the measurement of the plane deflection (assumed as ± 0.5 degrees). The error in D(I) is dominated by the errors in the measurements of plane deflections and therefore the 16 bit floating point arithmetic is sufficiently accurate to carry out the control functions.

3. Utility Routines

For debugging and display purposes, some additional routines exist.

a. Routine INTREAD reads four hexadecimal digits from the CRT in ASCII-code and converts it to binary code.

b. Routine INTRNT accomplishes the opposite of routine INTREAD.

c. Routine FXPNTREAD reads a fixed point decimal number in the range of the floating point format from the CRT in ASCII code and converts it to floating point format.

d. Routine FXPNTPRINT does the opposite of routine FXPNTREAD.

e. Routine FLTPRNT displays a number in floating point format.

IV. CONCLUSION

This investigation has shown the general feasibility of using a microcomputer as a controller for depth and pitch of a submarine. The computational speed and accuracy of the microcomputer are sufficient to permit the control of a submarine.

In addition it seems possible to incorporate the control of the rudder and the trim in the controller. Also, some different approaches should be investigated, i.e., the use of a second microcomputer to solve the Riccati equation in real time as well as to estimate the rates of change by a Kalman filter.

A further interesting project would be the implementation of the submarine simulation on a microcomputer. The microcomputer based simulation could be developed for use in a training facility to exercise manual control of depth and pitch of a submarine.

APPENDIX A

EQUATIONS OF MOTION, LINEARIZATION OF EQUATIONS OF MOTION AND CROSS REFERENCE OF NOTATION

A1. EQUATIONS OF MOTION

The following set of equations are referred to a body fixed system of axes which are coincident with the principal axes of inertia of the body. The origin of this axis-system is located at the assumed center of mass of the body

ation of Motion Along the Body Axis System x-Axis

$$\text{- vr + wq)} = \frac{\rho}{2}\ell^4\left[X_{qq}\,'\,q^2 + X_{rr}\,'\,r^2 + X_{rp}\,'\,rp\right]$$

$$+ \frac{\rho}{2}\ell^3\left[X_{\dot{u}}\,'\,\dot{u} + X_{vr}\,'\,vr + X_{wq}\,'\,wq\right]$$

$$+ \frac{\rho}{2}\ell^2\left[X_{uu}\,'\,u^2 + X_{vv}\,'\,v^2 + X_{ww}\,'\,w^2\right]$$

$$+ \frac{\rho}{2}\ell^2u^2\left[X_{\delta r\delta r}\,'\,\delta_r{}^2 + X_{\delta s\delta s}\,'\,\delta_s{}^2 + X_{\delta b\delta b}\,'\,\delta_b{}^2\right]$$

$$+ \frac{\rho}{2}\ell^2 X_{vvn'}\,'\,(n' - 1)\,v^2$$

$$+ \frac{\rho}{2}\ell^2 X_{wwn'}\,'\,(n' - 1)\,w^2$$

$$+ \frac{\rho}{2}\ell^2u^2 X_{\delta s\delta sn'}\,'\,(n' - 1)\,\delta_s{}^2$$

$$+ \frac{\rho}{2}\ell^2u^2 X_{\delta r\delta rn'}\,'\,(n' - 1)\,\delta_r{}^2$$

$$- \Sigma W_i \sin\theta$$

$$+ (F_x)_P$$

uation of Motion Along the Body Axis System y-Axis

$$\dot{v} - wp + ur) = \frac{\rho}{2} \ell^4 \left[Y_{\dot{r}} \, ' \dot{r} + Y_{\dot{p}} \, ' \dot{p} \right]$$

$$+ \frac{\rho}{2} \ell^4 \left[Y_{pq} \, ' pq + Y_{p|p|} \, ' p|p| \right]$$

$$+ \frac{\rho}{2} \ell^3 \left[Y_{\dot{v}} \, ' \dot{v} + Y_{wp} \, ' wp + Y_{v|r|} \, ' \frac{v}{|v|} \, |(v^2 + w^2)^{\frac{1}{2}}| \, |r| \right]$$

$$+ \frac{\rho}{2} \ell^3 \left[Y_{r} \, ' ur + Y_{|r|\delta r} \, ' u|r|\delta r + Y_{p} \, ' up \right]$$

$$+ \frac{\rho}{2} \ell^3 \, Y_{rn'} \, ' (n' - 1) ur$$

$$+ \frac{\rho}{2} \ell^2 \left[Y_{*} \, ' u^2 + Y_{v} \, ' uv + Y_{v|v|} \, ' v|(v^2 + w^2)^{\frac{1}{2}}| \right]$$

$$+ \frac{\rho}{2} \ell^2 u^2 \, Y_{\delta r} \, ' \delta_r$$

$$+ \frac{\rho}{2} \ell^2 u^2 \, Y_{\delta rn'} \, ' (n' - 1) \delta_r$$

$$+ \frac{\rho}{2} \ell^2 \, Y_{vn'} \, ' (n' - 1) uv$$

$$+ \frac{\rho}{2} \ell^2 \, Y_{v|v|n'} \, ' (n' - 1) v|(v^2 + w^2)^{\frac{1}{2}}|$$

$$+ \frac{\rho}{2} \ell^2 \, Y_{wv} \, ' wv\#$$

$$+ \frac{\rho}{2} \ell^2 \, (F_y)_{vs} \, \frac{v^2 + w^2}{U} \, (-w) \, \text{Sin } \omega t$$

$$+ \Sigma W_i \sin \phi \cos \theta$$

\# Multiplied by $\frac{u}{U}$ for large angles of attack near -90°

$$uq + vp) = \frac{\rho}{2} \ell^4 \, Z_{\dot{q}} \, ' \dot{q}$$

$$+ \frac{\rho}{2} \ell^4 \left[Z_{rr} \, ' r^2 \# + Z_{rp} \, ' rp\# \right]$$

Note 1

$$+ \frac{\rho}{2} \ell^3 \left[Z_{\dot{w}} \, ' \dot{w} + Z_{vr} \, ' vr\# + Z_{vp} \, ' vp + \Delta Z_{vp} \, ' vp\# \right]$$

$$+ \frac{\rho}{2} \ell^3 \left[Z_{q} \, ' uq + Z_{|q|\delta s} \, ' u|q|\delta s + Z_{w|q|} \, ' \frac{w}{|w|} \left| (v^2 + w^2)^{\frac{1}{2}} \right| |q| \right]$$

$$+ \frac{\rho}{2} \ell^3 \, Z_{qn'} \, ' (n' - 1) \, uq$$

$$+ \frac{\rho}{2} \ell^2 \left[Z_{*} \, ' u^2 + Z_{w} \, ' uw + Z_{w|w|} \, ' w \left| (v^2 + w^2)^{\frac{1}{2}} \right| \right]$$

$$+ \frac{\rho}{2} \ell^2 \left[Z_{|w|} \, ' u|w| + Z_{ww} \, ' \left| w \, (v^2 + w^2)^{\frac{1}{2}} \right| + Z_{vv} \, ' v^2 \# \right]$$

$$+ \frac{\rho}{2} \ell^2 u^2 \left[Z_{\delta s} \, ' \delta s + Z_{\delta b} \, ' \delta b \right]$$

$$+ \frac{\rho}{2} \ell^2 \left[Z_{wn'} \, ' (n' - 1) \, uw + Z_{w|w|n'} \, ' (n' - 1) \, w \left| (v^2 + w^2)^{\frac{1}{2}} \right| \right]$$

$$+ \frac{\rho}{2} \ell^2 u^2 \, Z_{\delta sn'} \, ' (n' - 1) \, \delta_s$$

$$+ \frac{\rho}{2} \ell^2 \, (F_z)_{vs} \, \frac{v^2 + w^2}{U} \quad v \, \text{Sin} \, \omega t$$

$$+ \Sigma W_i \cos \theta \cos \phi$$

\# Multiplied by $\frac{u}{U}$ for large angles of attack near -90°

Note 1
when not multiplied by $\frac{u}{U}$ add to Z_{vp} '

ation of Motion About the Body Axis System x-Axis

$$+ (I_z - I_y)\, qr = \frac{\rho}{2} \ell^5 \left[K_{\dot{p}}\, ' \dot{p} + K_{qr}\, ' qr + K_{\dot{r}}\, ' \dot{r} + K_{p|p|}\, ' p|p| \right]$$

$$+ \frac{\rho}{2} \ell^4 \left[K_{p}\, ' up + K_{r}\, ' ur + K_{\dot{v}}\, ' \dot{v} + K_{wp}\, ' wp \right]$$

$$+ \frac{\rho}{2} \ell^3 \left[K_{*}\, ' u^2 + K_{v}\, ' uv + K_{v|v|}\, ' v|(v^2 + w^2)^{\frac{1}{2}}| \right]$$

$$+ \frac{\rho}{2} \ell^3\, K_{vw}\, ' vw$$

$$+ \frac{\rho}{2} \ell^3\, u^2 K_{\delta r}\, ' \delta_r$$

$$+ Bz_B \sin\phi \cos\theta$$

Note 1

$$(I_x - I_z)\, rp = \frac{\rho}{2}\ell^5 \left[M_{\dot{q}}\,' \dot{q} + M_{rr}\,' r^2\# + M_{rp}\,' rp + \Delta M_{rp}\,' rp\#\right]$$

$$+ \frac{\rho}{2}\ell^4 \left[M_q\,' uq + M_{|q|\delta s}\,' u|q|\delta s + M_{|w|q}\,' |(v^2 + w^2)^{\frac{1}{2}}|q\right]$$

$$+ \frac{\rho}{2}\ell^4 \left[M_{\dot{w}}\,' \dot{w} + M_{vr}\,' vr\# + M_{vp}\,' vp\#\right]$$

$$+ \frac{\rho}{2}\ell^4 M_{qn'}\,' (n' - 1)\, uq$$

$$+ \frac{\rho}{2}\ell^3 \left[M_{\dot{=}}\,' u^2 + M_w\,' uw + M_{w|w|}\,' w|(v^2 + w^2)^{\frac{1}{2}}|\right]$$

$$+ \frac{\rho}{2}\ell^3 \left[M_{|w|}\,' u|w| + M_{ww}\,' |w(v^2 + w^2)^{\frac{1}{2}}| + M_{vv}\,' v^2\#\right]$$

$$+ \frac{\rho}{2}\ell^3\, u^2 \left[M_{\delta s}\,' \delta_s + M_{\delta b}\,' \delta_b\right]$$

$$+ \frac{\rho}{2}\ell^3 M_{wn'}\,' (n' - 1)\, uw$$

$$+ \frac{\rho}{2}\ell^3 M_{w|w|n'}\,' (n' - 1)\, w|(v^2 + w^2)^{\frac{1}{2}}|$$

$$+ \frac{\rho}{2}\ell^3\, u^2 M_{\delta sn'}\,' (n' - 1)\, \delta_s$$

$$+ Bz_B \sin\theta$$

$$- \Sigma W_i x_{ti} \cos\theta \cos\phi$$

\# Multiply by $\frac{u}{U}$ for large angles of attack near -90°

Note 1
when not multiplied by $\frac{u}{U}$ add to $M_{rp}\,'$

$$\dot{r} + (I_y - I_x)\,pq = \frac{\rho}{2}\ell^5 \left[N_{\dot{r}}{}' \dot{r} + N_{pq}{}' pq + N_{\dot{p}}{}' \dot{p}\right]$$

$$+ \frac{\rho}{2}\ell^4 \left[N_r{}' ur + N_{|r|\delta r}{}' u|r|\delta r + N_{|v|r}{}' |(v^2 + w^2)^{\frac{1}{2}}|r\right]$$

$$+ \frac{\rho}{2}\ell^4 \left[N_p{}' up + N_{\dot{v}}{}' \dot{v} + N_{wp}{}' wp\right]$$

$$+ \frac{\rho}{2}\ell^4 \; N_{rn'}{}' (n' - 1)\, ur$$

$$+ \frac{\rho}{2}\ell^3 \left[N_{*}{}' u^2 + N_v{}' uv + N_{v|v|}{}' v|(v^2 + w^2)^{\frac{1}{2}}|\right]$$

$$+ \frac{\rho}{2}\ell^3 \; u^2 N_{\delta r}{}' \delta_r$$

$$+ \frac{\rho}{2}\ell^3 \; u^2 N_{\delta r n'}{}' (n' - 1)\, \delta_r$$

$$+ \frac{\rho}{2}\ell^3 \; N_{vn'}{}' (n' - 1)\, uv$$

$$+ \frac{\rho}{2}\ell^3 \; N_{v|v|n'}{}' (n' - 1)\, v|(v^2 + w^2)^{\frac{1}{2}}|$$

$$+ \frac{\rho}{2}\ell^3 \; N_{wv}{}' wv\#$$

$$+ \Sigma W_i \, x_{ti} \cos\theta \sin\phi$$

\# Multiply by $\frac{u}{U}$ for large angles of attack near -90°

AUXILIARY EQUATIONS

$$= p + \dot{\psi}\sin\theta$$

$$= (q - \dot{\psi}\cos\theta\sin\phi) / \cos\phi$$

$$= (r + \dot{\theta}\sin\phi) / \cos\theta\cos\phi$$

$$= u\cos\theta\cos\psi + v(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + w(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi)$$

$$= u\cos\theta\sin\psi + v(\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi) + w(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)$$

$$= -u\sin\theta + v\cos\theta\sin\phi + w\cos\theta\cos\phi$$

$$= (u^2 + v^2 + w^2)^{\frac{1}{2}}$$

$$(F_x)_P = \frac{\rho}{2}\ell^2 u^2\left[a_1' + a_2' n' + a_3' n'^2\right] \quad \text{when } k_1 < n'$$

$$= \frac{\rho}{2}\ell^2 u^2\left[b_1' + b_2' n' + b_3' n'^2\right] \quad \text{when } k_2 < n' < k_1$$

$$= \frac{\rho}{2}\ell^2 u^2\left[c_1' + c_2' n' + c_3' n'^2\right] \quad \text{when } k_3 < n' < k_2$$

$$= \frac{\rho}{2}\ell^2 u^2\left[d_1' + d_2' n' + d_3' n'^2\right] \quad \text{when } n' < k_3$$

a_1', a_2', a_3'
b_1', b_2', b_3'
c_1', c_2', c_3'
d_1', d_2', d_3'

Sets of non-dimensional coefficients used in the propulsion equation above. The set which will be in effect at any time during a simulated maneuver will depend on the value of n' and the numbers k_1, k_2, k_3.

NOMENCLATURE

All symbols used in the equations of motion and in the auxiliary equations and relationships which appear in this report are defined below. Any dimensions involved will be consistent with the foot-pound-second system of units. All angles are in degrees.

SYMBOL	DEFINITION
$\dot{}$	A dot over any symbol signifies differentiation with respect to time.
B	Buoyancy force which is positive upwards.
m	Mass of the submarine including the water in the free flooding spaces.
$\mathscr{L}$	Overall length of the submarine
U	Linear velocity of origin of body axes relative to an earth-fixed axis system.
u	Component of U along the body x-axis.
v	Component of U along the body y-axis.
w	Component of U along the body z-axis.

u_c	Command speed: A steady value of u for a given propeller rpm when α, β and control surface angles are zero. Sign changes with propeller reversal.
x	Longitudinal axis of the body fixed coordinate axis system.
y	Transverse axis of the body fixed coordinate axis system.
z	Vertical axis of the body fixed coordinate axis system.
x_o	Distance along the x_o axis of an earth-fixed axis system.
y_o	Distance along the y_o axis of an earth-fixed axis system.
z_o	Distance along the z_o axis of an earth-fixed axis system.
p	Component of angular velocity about the body fixed x-axis.
q	Component of angular velocity about the body fixed y-axis.
r	Component of angular velocity about the body fixed z-axis.
z_B	The z coordinate of the center of buoyance (CB) of the submarine.

α	Angle of attack.
β	Angle of drift.
δb	Deflection of bowplane (or sailplane)
δr	Deflection of rudder.
δs	Deflection of sternplane.
n'	The ratio u_c/u.
θ	Pitch angle.
ψ	Yaw angle.
ϕ	Roll angle.
ρ	Mass density of sea water.
W_i	Weight of water blown from a particular ballast tank identified by the integer assigned to the index i.
ω	Angular velocity.
t	Time.
x_{ti}	Location along the body x-axis of the center of mass of the i^{th} ballast tank when this tank is filled with sea water.

$(F_x)_p$	Propulsion force (see auxiliary equations and relationships).
I_x	Moment of inertia of a submarine about the x-axis.
I_y	Moment of inertia of a submarine about the y-axis.
I_z	Moment of inertia of a submarine about the z-axis.
$K_{\dot{p}}'$, K_p', $K_{p\|p\|}'$, K_{qr}', K_r', $K_{\dot{r}}'$, $K_{\dot{v}}'$, K_{wp}', K_*', K_v', $K_{v\|v\|}'$, K_{vw}', $K_{\delta r}'$	Non-dimensional constants each of which is assigned to a particular force term in the equation of motion about the body x-axis.
M_{rr}', M_{rp}', $\Delta M_{rp}'$, M_q', $M_{\|q\|\delta s}'$, $M_{\dot{w}}'$, M_{vr}', M_{vp}', M_{qn}', M_*', $M_{w\|w\|}'$, $M_{\|w\|}'$, M_{ww}', M_{vv}', $M_{\delta s}'$, M_{wn}', $M_{w\|w\|n}'$, $M_{\delta sn}'$	Non-dimensional constants each of which is assigned to a particular force term in the equation of motion about the body y-axis.

N_{pq}', $N_{\dot{p}}'$, N_{r}', $N_{|r|\delta r}'$, $N_{|v|r}'$,

$N_{\dot{v}}'$, N_{wp}', N_{rn}', N_{*}', N_{v}',

$N_{v|v|}'$, $N_{\delta r}'$, $N_{\delta rn}'$, N_{vn}', $N_{v|v|n}'$,

N_{rv}'

Non-dimensional constants each of which is assigned to a particular force term in the equation of motion about the body z-axis.

X_{qq}', X_{rr}', X_{rp}', $X_{\dot{u}}'$, X_{vr}', X_{wq}',

X_{uu}', X_{vv}', X_{ww}', $X_{\delta r\delta r}'$, $X_{\delta s\delta s}'$,

$X_{\delta b\delta b}'$, X_{vvn}', X_{wwn}', $X_{\delta s\delta sn}'$,

$X_{\delta r\delta rn}'$

Non-dimensional constants each of which is assigned to a particular force term in the equation of motion along the body x-axis.

$Y_{\dot{r}}'$, $Y_{\dot{p}}'$, Y_{pq}', $Y_{p|p|}'$, $Y_{\dot{v}}'$, Y_{wp}',

$Y_{v|r|}'$, Y_{r}', $Y_{|r|\delta r}'$, Y_{p}', Y_{rn}',

Y_{*}', Y_{v}', $Y_{v|v|}'$, $Y_{\delta r}'$, $Y_{\delta rn}'$,

Y_{vn}', $Y_{v|v|n}'$, Y_{wv}', $(F_y)_{vs}$

Non-dimensional constants each of which is assigned to a particular force term in the equation of motion along the body y-axis.

$Z_{\dot{q}}'$, Z_{rr}', Z_{rp}', $Z_{\dot{w}}'$, Z_{vr}, Z_{vp}',

$\Delta Z_{vp}'$, Z_{q}', $Z_{|q|\delta s}$, $Z_{w|q|}'$,

Z_{qn}', Z_{*}', Z_{w}', $Z_{w|w|}'$, $Z_{|w|}'$,

Z_{ww}', Z_{vv}', $Z_{\delta s}'$, $Z_{\delta b}'$, Z_{wn}',

$Z_{w|w|n}'$, $Z_{\delta sn}'$, $(F_z)_{vs}$

Non-dimensional constants each of which is assigned to a particular force term in the equation of motion along the body z-axis.

A2. EQUATIONS OF MOTION SOLVED FOR $\dot{u}, \dot{v}, \dot{w}, \dot{p}, \dot{q}, \dot{r}$

The equations were separated, the terms involving $\dot{u}$, $\dot{v}$, $\dot{w}$, $\dot{p}$, $\dot{q}$ and $\dot{r}$ on the left side; the remaining terms on the right side. The equations containing I_x, I_y and I_z were divided by ℓ^5 and the equations containing m were divided by ℓ^3, where ℓ is the length of the submarine. ρ, the density of the fluid, was set to 2.0. This yields a system of equations of the form

$$A \cdot DOT = Z \qquad (1)$$

where

$$A = \begin{bmatrix} \frac{m}{\ell^3} - X'_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{m}{\ell^3} - Y'_{\dot{v}} & -\ell Y'_{\dot{p}} & 0 & 0 & -\ell Y'_{\dot{r}} \\ 0 & 0 & \frac{m}{\ell^3} - Z'_{\dot{w}} & 0 & -\ell Z'_{\dot{q}} & 0 \\ 0 & -\frac{K'_{\dot{v}}}{\ell} & 0 & \frac{I_x}{\ell^5} - K'_{\dot{p}} & 0 & -K'_{\dot{r}} \\ 0 & 0 & -\frac{M'_{\dot{w}}}{\ell} & 0 & \frac{I_y}{\ell^5} - M'_{\dot{q}} & 0 \\ 0 & -\frac{N'_{\dot{v}}}{\ell} & 0 & -N'_{\dot{p}} & 0 & \frac{I_z}{\ell^5} - N'_{\dot{r}} \end{bmatrix}$$

$$DOT = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}$$

and Z consists of the remaining linear and nonlinear terms on the right side. Premultiplying both sides of (1) by A^{-1} solves the system of equations for $\dot{u}$, $\dot{v}$, $\dot{w}$, $\dot{p}$, $\dot{q}$, $\dot{r}$. The terms $\frac{m}{\ell^3}$ and $\frac{I}{\ell^5}$ will be replaced by m' and I' respectively. In contrast to the notation in part A1 of Appendix A, the notation for the speed u will be replaced by u_a to distinguish it from the notation for the controls u.

A3. LINEARIZATION OF EQUATION OF MOTION

To use the equations of motion in the controller the nonlinear equations were linearized under the following assumptions:

i) $u_c = u_a$ yields $u' = 1$, all terms involving $(n' - 1)$ vanish.

ii) The linearized model is in trim, therefore,

$$\sum W_i x_{ti} \cos \Theta \cos \phi = 0$$

$$\sum W_i \cos \Theta \cos \phi = 0 .$$

iii) u_a is a measured input.

iv) Depth and pitch are the only parameters to be controlled.

The remaining nonlinear terms are dropped. Left with the equations involving w and q the problem is reduced to two degrees of freedom, the resulting equations are shown on the following page.

$$(2)\quad \begin{pmatrix} m'-Z_{\dot{w}}' & -\ell Z_{\dot{q}}' \\ -M_{\dot{w}}'/\ell & I_y'-M_{\dot{q}}' \end{pmatrix} \begin{pmatrix} \dot{w} \\ \dot{q} \end{pmatrix} = \begin{pmatrix} u\cdot Z_w'/\ell & (m'+Z_q)u \\ u\cdot M_w'/\ell^2 & uM_q'/\ell \end{pmatrix} \begin{pmatrix} w \\ q \end{pmatrix} + \begin{pmatrix} u^2 Z_{Ss}'/\ell & u^2 Z_{Sb}/\ell \\ u^2 M_{Ss}/\ell^2 & u^2 M_{Sb}/\ell^2 \end{pmatrix} \begin{pmatrix} \delta_s \\ \delta_b \end{pmatrix} + \begin{pmatrix} Z_* u^2/\ell \\ M_* u^2/\ell \end{pmatrix}$$

The constant additive last term in the set of linearized equations will be omitted in future use of these equations because of its insignificant contribution. Let

$$DET = (m' - Z'_{\dot{w}})(I'_y - M'_{\dot{q}}) - Z'_{\dot{q}}M'_{\dot{w}}$$

and

$$INV = \begin{bmatrix} m' - Z'_{\dot{w}} & -\ell Z'_{\dot{q}} \\ -M'_{\dot{w}}/\ell & I'_y - M'_{\dot{q}} \end{bmatrix} = \frac{1}{DET} \begin{bmatrix} I'_y - M'_{\dot{q}} & \ell Z'_{\dot{q}} \\ M'_{\dot{w}}/\ell & m' - Z'_{\dot{w}} \end{bmatrix}$$

then (2) becomes

$$\begin{bmatrix} \dot{w} \\ \dot{q} \end{bmatrix} = INV \begin{bmatrix} u_a Z'_w/\ell & (m' + Z'_q)u_a \\ u_a M'_w/\ell & u_a M'_q/\ell \end{bmatrix} \begin{bmatrix} w \\ q \end{bmatrix}$$

$$+ INV \begin{bmatrix} u_a^2 Z_{\delta_s}/\ell & u_a^2 Z_{\delta_b}/\ell \\ u_a^2 M_{\delta_s}/\ell^2 & u_a^2 M_{\delta_b}/\ell^2 \end{bmatrix} \begin{bmatrix} \delta_s \\ \delta_b \end{bmatrix}$$

A4. CROSS REFERENCE OF NOTATION IN COMPUTER PROGRAMS AND IN MATHEMATICAL MODEL

The following correspondence between the notations in the mathematical model and the computer program has been used:

1. Subscripted variables like I_x or $Z_{w|q|}$ are written IX and ZW1Q1 where the 1's denote absolute value signs.
2. Notations which are obvious from their use in the programs are not explained in the following list. e.g. IXY = IX - IY, or LC2 = LC**2.
3. Some of the notations in subroutines can be traced back through the argument list.

The following list is a correspondence between some notations and explains the notation where necessary.

Computer Program	Symbol in Mathematical Model	Description, if necessary
A(I,J)	A	Matrix of linear coefficients for $\dot{u}$, $\dot{v}$, $\dot{w}$, $\dot{p}$, $\dot{q}$, $\dot{r}$ of equations of motion, I,J = 1 to 6
AINV(I,J)	A^{-1}	Inverse of A, I,J = 1 to 6
B(I,J)	B	Matrix B from state equatin, I,J = 1 to 4
C1	C1	Element of weighting matrix S, weight for stern plane

C2	C2	Element of weighting matrix S, weight for fairwater plane
C1I	1/C1	Inverse of C1 combined with normalizing factor for D(1) to get 16-bit integer representation in PLM
C2I	1/C2	Inverse of C2 combined with normalizing factor for D(2) to get 16-bit integer representation in PLM-program
D(1)	δ_s	Control input for stern plane in radians (used in controller)
D(2)	δ_b	Control input for fairwater plane in radians (used in controller)
CON0		Normalizing factor for Θ in PLM-program
CON1		Limit for error in pitch in PLM-program
CON2		Limit for error in depth in PLM-program
CON3		Normalizing factor for Θ (1/CON0) in PLM-program
DB	δ_b	Defelction of fairwater plane (used in simulation of submarine) in radians
DBER		Difference between actual deflection of fairwater plane and control input for fairwater plane
DBINC		Increment or decrement of deflection of fairwater plane for next call to routine PLGEN
DBORD		Same as D(2), but in 16-bit integer format of PLM-program

DBDOT		Rate of change in movement of fairwater plane
DBGRA		Same as DB, but in degrees
DEPTH	Z_o	Depth of submarine (units in DSL/360 program: feet - units in PLM-program: inches)
DET	det A	Determinant of A
DOT(1)	$\dot{u}$	
DOT(2)	$\dot{v}$	
DOT(3)	$\dot{w}$	
DOT(4)	$\dot{p}$	
DOT(5)	$\dot{q}$	
DOT(6)	$\dot{r}$	
DPTH(I)		Last five values of DEPTH, I = 1 to 5
DR	δ_r	Deflection of rudder in radians
DRINC		Same as DBINC, but for rudder
DRER		Same as DBER, but for rudder
DRORD		Ordered deflection of rudder
DS	δ_s	Deflection of stern plane (used in simulation of submarine) in radians
DSER		Same as DBER, but for stern plane
DSDOT		Rate of movement of the stern plane
DSGRA		Same as DS, but in degrees
DSINC		Same as DBINC, but for stern plane
DSORD		Same as D(1), but in 16-bit integer format of PLM-program
F(I,J)	F	Product of matrices -K and B, I = 1 to 4, J = 1,2

FF(IU,IW1,IW2,I,J)		Matrix containing all values of F for different speeds and different weights w_1 and w_2
G(I,J)	G	Matrix G from state equation, I,J = 1 to 4
IU		Speed UA rounded to an integer
IW1		Subscript to select different weights w_1 in array FF
IW2		Subscript to select different weights w_2 in array FF
K(I,J)	K	Gain matrix, I = 1 to 4
KIJ		Same as K(I,J), only different notation
KDIG	$\dot{K}$	Derivative of K(I,J) as a function of time, I,J = 1 to 4
PERR		Error in pitch in 16-bit integer format of PLM-program
PIDOT	$\dot{\Theta}$	
PITCH	Θ	
PIGRA		PITCH in degrees
PORD	r_3	Ordered pitch
PTCH(I)		Last five values of PITCH, I = 1 to 5
Q(I,J)	Q	Weighting matrix for error in pitch and error in depth
RODOT	$\dot{\phi}$	
ROLL	ϕ	
S	S	Weighting matrix for control inputs
SI	S^{-1}	Inverse of S

UA	u, u_a	Speed along body x-axis (notation u is used in simulation of submarine, notation u_a is used in controller to distinguish between the control input u and speed
UC	u_c	Command speed
W1	w_1	Weight for error in depth, element of matrix Q
W2	w_2	Weight for error in pitch, element of matrix Q
Y(1)	$z_o - r_1$	Output variable of state equation, error in depth
Y(2)	$\dot{w}$	Output variable of state equation
Y(3)	$\Theta - r_2$	Output variable of state equation, error in pitch
Y(4)	$\dot{q}$	Output variable of state equation
YADOT	$\dot{\psi}$	
YAW	ψ	
Z(I)		Vector consisting of all nonlinear terms involving $\dot{u}$, $\dot{v}$, $\dot{w}$, $\dot{p}$, $\dot{q}$, $\dot{r}$ and remaining terms
ZERR		Error in depth in 16-bit integer format of PLM-program
ZORD	r_1	Ordered depth

APPENDIX B

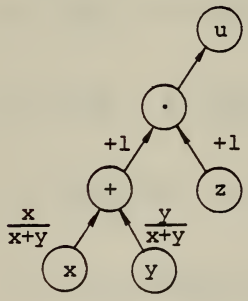
COMPUTATIONAL PROCEDURE FOR THE ERROR PROPAGATION
(shown for D(1) and Y(2))

The relative errors for addition and multiplication are defined as [Ref. 8].

1. Addition: $$\frac{e_{x+y}}{\bar{x}+\bar{y}} = \frac{\bar{x}}{\bar{x}+\bar{y}} \frac{e_x}{\bar{x}} + \frac{\bar{y}}{\bar{x}+\bar{y}} \frac{e_y}{\bar{y}}$$

2. Multiplication: $$\frac{e_{x\cdot y}}{\bar{x}\cdot\bar{y}} = \frac{e_x}{\bar{x}} + \frac{e_y}{\bar{y}}$$

A process graph is a graphical representation in which the arithmetic operations are carried out. In the case of addition and multiplication the scheme of labeling is the following:



To obtain the total error leaving a node, the error values entering a node are multiplied with the branch labels and added together. To this sum is then added the roundoff error of the particular operation.

The relative error Ye(2) in computing the rate of changes in di (see also process graphs at the end of Appendix B).

$$\left|\frac{Ye(2)}{Y(2)}\right| \leq \Big|(\Delta 1+\delta 1)\ \frac{C1\cdot d1}{C1\cdot d1+C2\cdot d2}\cdot\frac{C1\cdot d1+C2d2}{C1\cdot d1+C2d2+C4d4}$$

$$\cdot\frac{C1d1+C2d2+C4d4}{C1d1+C2d2+C4d4+C5d5}\cdot\frac{C1\cdot d1+C2\cdot d2+C4\cdot d4+C5d5}{C1d1+C2d2+C4d4-C3d3}$$

$$Y(I)$$

$$+\ (\Delta 2+\delta 2)\ \frac{C2\cdot d2}{Y(I)}+\ (\Delta 4+\delta 4)\frac{C4\cdot d4}{Y(I)}$$

$$+\ (\Delta 5+\delta 5)\ \frac{C5\cdot d5}{Y(I)}+\ (\Delta 3+\delta 3)\ \frac{C3\cdot d3}{Y(I)}$$

$$+\ m1\ \frac{C1\cdot d1}{Y(I)}+\ m2\ \frac{C2\cdot d2}{Y(I)}+\ m4\ \frac{C4\cdot d4}{Y(I)}$$

$$+\ m5\ \frac{C5\cdot d5}{Y(I)}+\ m3\ \frac{C3\cdot d3}{Y(I)}$$

$$+\ \alpha 1\ \frac{C1d1+C2d2}{Y(I)}+\ \alpha 2\ \frac{C1\cdot d1+C2\cdot d2+C4\cdot d4}{Y(I)}$$

$$+\ \alpha 3\ \frac{C1\cdot d1+C2\cdot d2+C4\cdot d4+C5\cdot d5}{Y(I)}+\ \sigma\Big|$$

where

Y(I) = rate of change in depth (Y(2)) and in pitch (Y(4));

Δi = roundoff error in coefficient for Legendre polynomial ci, i = 1,2,...,5;

δi = inherent error in measurements i, i = 1,2,...,5;

αi = roundoff error after addition i;

mi = roundoff error after multiplication i;

σ = roundoff error after subtraction;

di = sampled data (pitch,depth), i = 1,2,...,5.

Multiplying both sides with Y(2) and rearranging the terms yields the following expression for the absolute error

$$|Ye(2)| \leq |C1d1(\Delta 1+\delta 1+m1+\alpha 1+\alpha 2+\alpha 3+\sigma)$$
$$+ C2d2(\Delta 2+\delta 2+m2+\alpha 1+\alpha 2+\alpha 3+\sigma)$$
$$+ C4d4(\Delta 4+\delta 4+m4+\alpha 2+\alpha 3+\sigma)$$
$$+ C5d5(\Delta 5+\sigma 5+m5+\alpha 3+\sigma) + C3d3(\Delta 3+\delta 3+m3-\sigma)$$

The relative error De(I) in computing the ordered plane deflection is

$$\left|\frac{De(I)}{D(I)}\right| \leq |(\Delta 1+\delta 1)\ \frac{C6\cdot Y(3)F(4,1)}{D(I)} + f5+Ye(4))\ \frac{F(3,1)Y(4)}{D(I)}$$
$$+ (f3+Ye(2))\ \frac{F(2,1)Y(2)}{D(I)} + (f1+\delta 2)\ \frac{F(1,1)Y(1)}{D(I)}$$
$$+ (m1+m2)\ \frac{C6\cdot Y(3)F(4,1)}{D(I)} + m3\ \frac{F(3,1)Y(4)}{D(I)} +$$

$$+ m4 \frac{F(2,1)Y(2)}{D(I)} + m5 \frac{F(1,1)Y(1)}{D(I)}$$

$$+ \alpha 1 \frac{C6 \cdot Y(3)F(4,1)+Y(4)F(3,1)}{D(I)}$$

$$+ \alpha 3 \frac{C6 \cdot Y(3)F(4,1)+Y(4)F(3,1)+Y(2)F(2,1)}{D(I)} + \alpha 3|$$

where

fi = roundoff error in gains F(I,J);

D(I) = control output for plane deflection;

C6 = normalizing factor for variables representing pitch or error in pitch.

Multiplying both sides with D(I) and rearranging the terms yields the following expression for the absolute error

$$|De(I)| \leq |C6 \cdot Y(3)F(4,1)(\Delta 6+\delta 1+\alpha 1+\alpha 2+\alpha 3+m1+m2)$$

$$+ F(3,1) \cdot Y(4)(f5+Ye(2)+m3+\alpha 1+\alpha 2+\alpha 3)$$

$$+ F(2,1) \cdot Y(2)(f3+Ye(4)+m4+\alpha 2+\alpha 3)$$

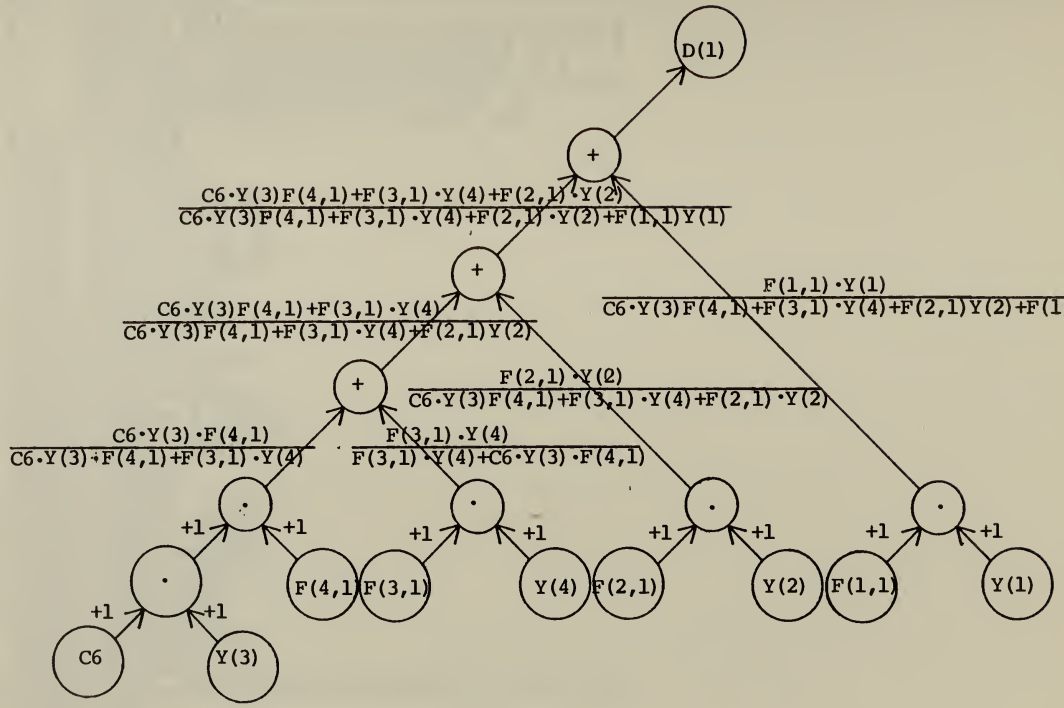
$$+ F(1,1) \cdot Y(1)(f1+\delta 2+m5+\alpha 3)|$$

Y(2) PROCESS GRAPH

Y(I) = (C1xd1) + (C2xd2) + (C4xd4) + (C5xd5) - (C3xd3)

di = SDATA(i), i = 1,...,5

$$D(1) = (C6 \cdot Y(3)) \cdot F(4,1) + F(3,1) \cdot Y(4) + F(2,1) \cdot Y(2) + F(1,1) \cdot Y(1)$$



D(1) PROCESS GRAPH

```
//             EXEC DSL
//DSL.INPUT DD *
*     PROGRAM GAIN.                                  *
*     PROGRAM GAIN COMPUTES THE STEADY STATE *
*     GAINS BY INTEGRATING THE RICCATI       *
*     EQUATION. THE GAINS ARE COMPUTED FOR   *
*     ALL COMBINATIONS OF 5 DIFFERENT WEIGHTS*
*     W1, 5 DIFFERENT WEIGHTS W2 AND 18      *
*     DIFFERENT VALUES OF THE SPEED UA.      *
PARAM C1=10.0
PARAM C2=10.0
PARAM LC=415.0
PARAM IX=7.3114E-06
PARAM IY=5.6667E-04
PARAM I2=5.6667E-04
PARAM KPDOT=-3.0E-06
PARAM KRDOT=-7.0E-06
PARAM KVDOT=-.00025
PARAM MQE=.0005
PARAM MQS=-.0025
PARAM ML=.0087445
PARAM MW=.003
PARAM MWDOT=-.0002
PARAM MQ=-.0025
PARAM MQDOT=-.0004
PARAM NPDOT=-7.0E-06
PARAM NRDOT=-5.0E-04
PARAM NVDOT=.0003
PARAM UC=0.0
PARAM W1=0.0
PARAM W2=0.0
PARAM XUDOT=-.00015
PARAM YPDOT=-.0003
PARAM YRDOT=.00009
PARAM YVDOT=-.011
PARAM ZQE=-.0025
PARAM ZQS=-.005
PARAM ZQ=-.0045
PARAM ZQDOT=-.0002
PARAM ZW=-.011
PARAM ZWDOT=-.0075
INTGER I,II,J,JJ,L,M,N,IW1,IW2,IU,LL,MM,NPLOT
PARAM IU=0
PARAM I=0
PARAM J=0
PARAM L=0
PARAM IW1=0
PARAM IW2=0
```

```
      CCNST NFLCT=1
      CCNTRL FINTIM=200.0,DELT=.01,DELS=200.0
      INITIAL
      D       DIMENSION FF(19,5,5,4,2),A(6,6),AINV(6,6),K(4,4),G(4,4)
      D       DIMENSION Q(4,4),KG(4,4),GTK(4,4),FSIFT(4,4),F(4,2),E(4,2),KC(4,4)
      D       DIMENSION LL(6),MM(6,6),SI(2,2)
              IF (IU.NE.0) GO TO 77
              LC2=LC**2
      * INITIALIZE A TO 0.0
              DO 71 I=1,6
              DO 71 J=1,6
           71 A(I,J)=0.0
      * INITIALIZE E,G,Q TO 0.0
              DO 72 I=1,4
              DO 72 J=1,4
              E(I,J)=0.0
              G(I,J)=0.0
           72 Q(I,J)=0.0
      * INITIALIZE S INV.
              SI(1,1)=1./C1
              SI(1,2)=0.0
              SI(2,1)=0.0
              SI(2,2)=1./C2
      * INITIALIZE A
              A(1,1)=ML-XUDOT
              A(2,2)=ML-YVDOT
              A(4,2)=-KVDOT/LC
              A(6,2)=-NVDOT/LC
              A(3,3)=ML-ZWDOT
              A(5,3)=-MWDOT/LC
              A(2,4)=-YPDOT*LC
              A(4,4)=IX-KPDOT
              A(6,4)=-NPDOT
              A(3,5)=-ZQDOT*LC
              A(5,5)=IY-MQDOT
              A(2,6)=-YRDOT*LC
              A(4,6)=-KRDOT
              A(6,6)=IZ-NRDOT
      * CALL LIBRARY ROUTINE MINV TO COMPUTE THE INVERSE OF A
              DO 73 I=1,6
              DO 73 J=1,6
           73 AINV(I,J)=A(I,J)
              N=6
              CALL MINV(AINV,N,D,LL,MM)
      * PRINT AINV AND TRANSFER AINV TO DATACELL
           74 FORMAT(6(' INV(',I1,',',I1,')=',E12.6))
           75 FORMAT(6E12.6)
              DO 76 I=1,6
```

```
      WRITE(6,74) ((I,J,AINV(I,J),J=1,6)
   76 WRITE(16,75) (AINV(I,J),J=1,6)
* STEP THROUGH THE DIFFERENT COMBINATIONS OF UC,W1,W2
   77 IW2=MOD(IW2,5)
      IF (IW2.EQ.0) IW1=MOD(IW1,5)
      IF (IW1.EQ.0) IU=IU+1
      IF (IW2.EQ.0) IW1=IW1+1
      IW2=IW2+1
      W1=0.7+IW1*0.1
      W2=2100.0+IW2*300.0
      UC=IU*1.69
      UC2=UC**2
* INITIALIZE C
      C(1,1)=W1
      C(3,3)=W2
      DELS=200.0
      IF ((IU.EQ.15).AND.(IW1.EQ.3).AND.(IW2.EQ.3)) DELS=0.1
      DET=(ML-ZWDOT)*(IY-MQDOT)-ZQDOT*MWDOT
* INITIALIZE G
      G(1,2)=1.0
      G(3,4)=1.0
      G(2,2)=((IY-MQDOT)*ZW+MW*ZQDOT)*UC/(UC*DET)
      G(2,4)=((IY-MQDOT)*(ML+ZQ)+MQ*ZQDOT)*UC/DET
      G(4,2)=(MWDOT*ZW+MW*(ML-ZWDOT))*UC/(UC2*DET)
      G(4,4)=(MWDOT*(ML+ZQ)+MQ*(ML-ZWDOT))*UC/(UC*DET)
* INITIALIZE E
      E(2,1)=((IY-MQDOT)*ZDS+MDS*ZQDOT)*UC**2/(UC*DET)
      E(2,2)=((IY-MQDOT)*ZDB+MDB*ZQDOT)*UC**2/(UC*DET)
      E(4,1)=(MWDOT*ZDS+(ML-ZWDOT)*MDS)*UC**2/(UC2*DET)
      E(4,2)=(MWDOT*ZDB+(ML-ZWDOT)*MDB)*UC**2/(UC2*DET)
DERIVATIVE
NOSORT
* INTEGRATE K DOT
      K11=INTGRL(0.0,KD11)
      K12=INTGRL(0.0,KD12)
      K13=INTGRL(0.0,KD13)
      K14=INTGRL(0.0,KD14)
      K22=INTGRL(0.0,KD22)
      K23=INTGRL(0.0,KD23)
      K24=INTGRL(0.0,KD24)
      K33=INTGRL(0.0,KD33)
      K34=INTGRL(0.0,KD34)
      K44=INTGRL(0.0,KD44)
      K(1,1)=K11
      K(1,2)=K12
      K(1,3)=K13
      K(1,4)=K14
      K(2,2)=K22
```

```
      K(2,3)=K23
      K(2,4)=K24
      K(3,3)=K33
      K(3,4)=K34
      K(4,4)=K44
      DO 81 I=1,4
      DO 81 J=I,4
   81 K(J,I)=K(I,J)
*  INITIALIZE  F  TO 0.0
      DO 82 I=1,4
      DO 82 J=1,2
   82 F(I,J)=0.0
*  COMPUTE  F=K*B
      DO 83 I=1,4
      DO 83 L=1,2
      DO 83 J=1,4
   83 F(I,L)=F(I,L)-K(I,J)*B(J,L)
*  INITIALIZE  INTERMEDIATE RESULTS KG,GTK,FSIFT TO 0.0
      DO 84 I=1,4
      DO 84 J=1,4
      KG(I,J)=0.0
      GTK(I,J)=0.0
   84 FSIFT(I,J)=0.0
*  COMPUTE  KG=K*G, GTK=G TRANSP.*K
      DO 85 I=1,4
      DO 85 J=1,4
      DO 85 L=1,4
      GTK(I,J)=GTK(I,J)+G(L,I)*K(L,J)
   85 KG(I,J)=KG(I,J)+K(I,L)*G(L,J)
*  COMPUTE  FSIFT=K*B*S  INV.*B TRANSP.*K=F*S INV.*F TRANSP.
      DO 86 I=1,4
      DO 86 J=1,4
      DO 86 L=1,2
   86 FSIFT(I,J)=FSIFT(I,J)+F(I,L)*F(J,L)*SI(L,L)
*  COMPUTE  K DOT=K*G-K*B*S  INV.*B  TRANSP.*K+G TRANSP.*K+C
*                =KG-FSIFT+GTK+C
      KD11=KG(1,1)+GTK(1,1)-FSIFT(1,1)+C(1,1)
      KD12=KG(1,2)+GTK(1,2)-FSIFT(1,2)+C(1,2)
      KD13=KG(1,3)+GTK(1,3)-FSIFT(1,3)+C(1,3)
      KD14=KG(1,4)+GTK(1,4)-FSIFT(1,4)+C(1,4)
      KD22=KG(2,2)+GTK(2,2)-FSIFT(2,2)+C(2,2)
      KD23=KG(2,3)+GTK(2,3)-FSIFT(2,3)+C(2,3)
      KD24=KG(2,4)+GTK(2,4)-FSIFT(2,4)+C(2,4)
      KD33=KG(3,3)+GTK(3,3)-FSIFT(3,3)+C(3,3)
      KD34=KG(3,4)+GTK(3,4)-FSIFT(3,4)+C(3,4)
      KD44=KG(4,4)+GTK(4,4)-FSIFT(4,4)+C(4,4)
SAMPLE
      IF ((IU.NE.15).OR.(IW1.NE.3).OR.(IW2.NE.3)) RETURN
```

```
      L=1
      DO 88 I=1,4
      DO 88 J=1,2
      CALL DRWG(L,1,TIME,F(I,J))
   88 L=L+1
TERMINAL
* ASSIGN FINAL VALUE OF F COMPUTED IN THIS RUN TO FF
      DO 91 I=1,4
      DO 91 J=1,2
   91 FF(IU,IW1,IW2,I,J)=F(I,J)
      IF (IU.NE.19) CALL RERUN
      IF (IU.NE.19) RETURN
* PRINT FF AND TRANSFER FF TO DATACELL
   92 FORMAT(///)
      WRITE(6,92)
   93 FORMAT(3X,8(' F(',I2,',',I1,',',I1,',',I1,',',I1,')'))
   94 FORMAT(2X,8(2X,F12.5))
   95 FORMAT(8F10.4)
      DO 96 IU=1,18
      DO 96 IW1=1,5
      DO 96 IW2=1,5
      WRITE(6,93) (((IU,IW1,IW2,I,J),J=1,2),I=1,4)
      WRITE(6,94) ((FF(IU,IW1,IW2,I,J),J=1,2),I=1,4)
   96 WRITE(16,95) ((FF(IU,IW1,IW2,I,J),J=1,2),I=1,4)
      CALL ENDRW(NPLOT)
END
STOP
//G.FT16F001 DD DSN=SO147.UBOOT7,UNIT=2321,VOL=SER=CEL008,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=2000),
// DISP=(NEW,KEEP),LABEL=RETPD=120,
// SPACE=(TRK,20,RLSE)
//PLOT.SYSIN DD *
F(1,1) \S TIME
UA = 25.35 FT/SEC                                                    4
                                        5.0         7.0
F(1,2) \S TIME
UA = 25.35 FT/SEC                                                    4
                                        5.0         7.0
F(2,1) \S TIME
UA = 25.35 FT/SEC                                                    4
                                        5.0         7.0
F(2,2) \S TIME
UA = 25.35 FT/SEC                                                    4
                                        5.0         7.0
F(3,1) \S TIME
UA = 25.35 FT/SEC                                                    4
                                        5.0         7.0
F(3,2) \S TIME
```

UA = 25.35 FT/SEC	5.0	7.0	4
F(4,1) VS TIME UA = 25.35 FT/SEC	5.0	7.0	4
F(4,2) VS TIME UA = 25.35 FT/SEC	5.0	7.0	4

```
//              EXEC DSL
//DSL.INPUT DD *
*    PROGRAM SIMUL.                                              *
*    THIS PROGRAM SIMULATES THE MOTION OF A SUBMARINE            *
*    AND CALLS A CONTROL ALGORITHM TO KEEP THE                   *
*    SUBMARINE ON AN ORDERED DEPTH AND AN ORDERED PITCH.         *
*    THE ORDERED PARAMETERS                                      *
*      ORDERED DEPTH                                             *
*      ORDERED PITCH                                             *
*      SPEED OF                                                  *
*      SUBSCRIPTS IW1 AND IW2 FOR GAIN MATRIX FF                 *
*    CAN BE CHANGED DURING THE SIMULATION                        *
PARAM A=0.0
PARAM A1=-1.0E-03
PARAM A2=-9.5E-04
PARAM A3=1.95E-03
PARAM B=0.0
PARAM B2E=1.011413E-03
PARAM C1=10.0
PARAM C2=10.0
PARAM DEDCT=0.1222
PARAM DSDCT=0.1222
PARAM DFDCT=0.1047
PARAM DFORD=0.0
PARAM IX=7.3114E-06
PARAM IY=5.6867E-04
PARAM IZ=5.6867E-04
PARAM KDF=7.0E-05
PARAM KDR=-1.0E-04
PARAM KS=-1.04E-05
PARAM K1F1P=-8.0E-07
PARAM KWP=2.5E-04
PARAM K1V1V=-5.0E-04
PARAM KVW=3.5E-03
PARAM KF=-3.5E-05
PARAM KR=-4.0E-05
PARAM KV=-7.0E-04
PARAM LC=415.0
PARAM MDE=5.0E-04
PARAM MDS=-2.5E-03
PARAM ML=8.7445E-03
PARAM MFF=1.5E-04
PARAM MRR=-5.5E-04
PARAM MVR=-4.0E-03
PARAM MVF=9.0E-04
PARAM M1W1Q=-2.0E-03
PARAM MVV=1.5E-02
PARAM M1W1W=-5.0E-03
```

```
PARAM ZC=-4.5E-03
PARAM ZW=-1.1E-02
INCCN YACCT=0.0
INCCN PCCCT=0.0
INCCN PICCT=0.0
INCCN FT=-2.0E-05
INCCN AT=2.0E-05
INCCN AL=4.0E-04
INCCN DS=0.0
INCCN DE=0.0
INCCN DR=0.0
INTGER I,J,K,L,M,IU,IW1,IW2,FLAG,NPLOT
CCNST NPLCT=1
CONTRL FINTIM=400.0,DELT=.01,DELS=.2
INITIAL
C       COMMCN/B1/ FF(18,5,5,4,2),PTCH(5),CPTH(5),C(3)
C       DIMENSICN Z(6),DOT(6),AINV(6,6)
   11 FCRMAT(6E12.6)
       DC 12 I=1,6
   12 READ(16,11) (AINV(I,J),J=1,6)
   13 FCRMAT(6F10.4)
       READ (16,13)((((FF(I,J,K,L,M),M=1,2),L=1,4),K=1,5),J=1,5),I=1,18)
   14 FCRMAT(2F10.4)
       IW1=3
       IW2=3
       FLAG=1
       ZCCT=0.0
       PICT=0.0
       UC2=UC**2
       UC3=UC2*UC
       IZX=IZ-IX
       IYX=IY-IX
       IZY=IZ-IY
       SINC=0.0
       BINC=0.0
       RINC=0.0
       READ(5,21) TT,UC,ZORD,PCRD,CRCRC,C1,C2,IW1,IW2
DERIVATIVE
NOSORT
       DEPTH=INTGRL(0.0,ZCDOT)
* INTEGRATE ALL 6 COMPONENTS OF Y DOT
       U=INTGRL(UC,DOT(1))
       V=INTGRL(0.,DOT(2))
       W=INTGRL(0.,DOT(3))
       P=INTGRL(0.,DOT(4))
       Q=INTGRL(0.,DOT(5))
       R=INTGRL(0.,DOT(6))
```

```
      ROLL=INTGRL(0.0,RDOOT)
      PITCH=INTGRL(0.0,PIDOT)
      YAW=INTGRL(0.0,YADOT)
* CALL PLANE ANGLE GENERATOR PLGEN FOR STERN PLANE, FAIRWATER PLANE
* AND RUDDER
      DSER=D(1)-DS
      DBER=D(2)-DB
      DRER=DRCRD-DR
      CALL PLGEN(DB,DBDOT,DBER,DBINC)
      CALL PLGEN(DS,DSDOT,DSER,DSINC)
      CALL PLGEN(DR,DRDOT,DRER,DRINC)
* EVALUATE RIGHT SIDE Z OF EQUATIONS OF MOTION
      ZDOOT=-L*SIN(PITCH)+V*COS(PITCH)*SIN(ROLL)...
      +W*COS(PITCH)*COS(ROLL)
      PIDOT=(Q-YADOT*COS(PITCH)*SIN(ROLL))/COS(ROLL)
      YADOT=(R+PIDOT*SIN(ROLL))/(COS(PITCH)*COS(ROLL))
      RDOOT=P+YADOT*SIN(PITCH)
      DSGRA=DS*57.296
      DBGRA=DB*57.296
      PITGRA=PITCH*57.296
      AEV=ABS(V)
      AEW=ABS(W)
      AEP=ABS(P)
      AEQ=ABS(Q)
      AER=ABS(R)
      VVWW=V*V+W*W
      AVW=SQRT(VVWW)
      AEWP=FCNSW(W,-1.0,0.0,1.0)
      AEVP=FCNSW(V,-1.0,0.0,1.0)
      SA1=+LC*(XQQ*Q**2+XRR*R**2+XRP*R*P)
      SA2=+(ML*V*R+XVR*V*R+XWQ*W*Q-ML*W*Q)
      SA3=+(XVV*V**2+XWW*W**2)/LC-SIN(PITCH)*(AT+FT+AU)
      SA4=+(A1*U**2+A2*U*UC+A3*UC**2)/LC
      SB1=+LC*YPQ*P*Q
      SB2=+(YWP*W*P+YV1R1*ABR*AVW*ABVP+ML*W*P-ML*L*R)
      SB3=+(YWV*W*V+Y1V1V*AVW*V)/LC+SIN(ROLL)*COS(PITCH)...
      *(AT+FT+AU)
      SB4=(YR*R+YP*P+YV*V/LC)*U
      SC1=LC*P*(ZRR*R+ZRP*P)
      SC2=+(ZVP*V*P+ZVR*V*R+ZW1Q1*ABQ*AVW*ABWP+ML*U*Q-ML*P*V)
      SC3=+(ZWW*W**2+ZVV*V**2+ZW1W1*W*AVW+U*Z1W1*ABW+U*U*ZS)/LC
      SC4=ZQ*L*Q+ZW*U*W/LC+COS(PITCH)*COS(ROLL)*(AT+FT+AL)
      SD1=+(KQR*Q*R+K1P1P*ABP*P)-IZY*Q*R
      SD2=(KWP*W*P-BZB*SIN(ROLL)*COS(PITCH))/LC
      SD3=+(K1V1V*V*AVW+KVW*V*W+KS*U**2)/LC2
      SD4=((KP*P+KR*R)/LC+KV*V/LC2)*U
      SE1=(MRP*P+MRR*R+IZX*P)*R
      SE2=((MVR*R+MVP*P)*V+M1W1Q*AVW*Q-EZB*SIN(PITCH))/LC
```

```
      SE3=(MVV*V**2+MWW*W**2+M1W1W*AVW*W+M1W1*U*AVW+U**2*MS)/LC2
      SE4=MG*L*G/LC+(MW*U*W-(175.5*FT-219.5*AT)*CCS(PITCH)...
      *CCS(ROLL))/LC2
      SF1=(NPQ-IYX)*P*Q
      SF2=+(NWP*W*P+N1V1R*AVW*R)/LC
      SF3=(NWV*W+N1V1V*AVW)*V/LC2
      SF4=(NP*P+NR*R)*U/LC+(NV*U*V+(175.5*FT-219.5*AT)*...
      CCS(PITCH)*SIN(ROLL))/LC2
      Z(1)=SA1+SA2+SA3+SA4+(XDRDR*DR**2+XDSDS*DS**2+XDBDB*DB**2)*L**2/LC
      Z(2)=SB1+SB2+SB3+SB4+YDR*DR*U**2/LC
      Z(3)=SC1+SC2+SC3+SC4+(ZDS*DS+ZDB*DB)*U**2/LC
      Z(4)=SD1+SD2+SD3+SD4+KDR*DR*U**2/LC2
      Z(5)=SE1+SE2+SE3+SE4+(MDS*DS+MDB*DB)*U**2/LC2
      Z(6)=SF1+SF2+SF3+SF4+NDR*DR*U**2/LC2
* MULTIPLY RIGHT SIDE Z OF ECUATIONS OF MOTION WITH  A INV.
      DO 15 I=1,6
   15 DOT(I)=0.0
      DO 16 I=1,6
      DO 16 J=1,6
   16 DOT(I)=DOT(I)+AINV(I,J)*Z(J)
SAMPLE
   21 FORMAT(7F10.4,2I5)
*    READ NEW INPUT PARAMETERS
      IF ((TIME.GE.TT).AND.(TIME.LT.TT+DELS)) READ(5,21) TT,UC,ZCRD,...
      PCRD,RCRD,C1,C2,IW1,IW2
* CALL CONTROLLER CNTRL
      CALL CNTRL(C1,C2,IW1,IW2,U,DEPTH,PITCH,ZCRD,PCRD)
PRINT 1.0,DEPTH,PITCH
* PLOTTING
      IF (FLAG.LT.0) GO TO 24
      CALL DRWG(1,1,TIME,DSGRA)
      CALL DRWG(2,1,TIME,DBGRA)
      CALL DRWG(3,1,TIME,DEPTH)
      CALL DRWG(4,1,TIME,PITGRA)
      CALL DRWG(5,1,TIME,U)
   24 FLAG=-FLAG
TERMINAL
      CALL ENDRW(NPLOT)
END
STOP
FORTRAN
      SUBROUTINE CNTRL(C1,C2,IW1,IW2,UA,DEPTH,PITCH,ZCRD,PCRD)
      COMMON/F1/ FF(18,5,5,4,2),PTCH(5),DPTH(5),D(3)
      DIMENSION Y(4),SI(2,2)
C INITIALIZE S INV.
      SI(1,1)=1./C1
      SI(1,2)=0.0
      SI(2,1)=0.0
```

```
      SI(2,2)=1./C2
C UPDATE ARRAYS 'PTCH' AND 'DEPTH'
      DO 22 I=1,4
      DPTH(I)=DPTH(I+1)
   22 PTCH(I)=PTCH(I+1)
      PTCH(5)=PITCH
      DPTH(5)=DEPTH
C TRUNCATE SPEED 'UA' TO AN INTEGER
      IU=UA/1.689+0.5
C SET ELEMENTS OF ARRAY 'Y' TO ERROR IN DEPTH, ERROR IN PITCH,
C RATE OF CHANGE IN DEPTH AND RATE OF CHANGE IN PITCH
      Y(1)=DEPTH-ZORD
      Y(3)=PITCH-PORD
      Y(2)=PRIME(DPTH)
      Y(4)=PRIME(PTCH)
C ERROR CHANNEL LIMITER
      IF (ABS(Y(1)).GT.4.0) Y(1)=SIGN(4.0,Y(1))
      IF (ABS(Y(3)).GT.0.174) Y(3)=SIGN(0.174,Y(3))
C SUM OF PRODUCTS OF GAINS 'F' AND THE ELEMENTS OF 'Y' TO GET
C ORDERED PLANE DEFLECTIONS 'D(1)' AND 'D(2)'
      D(1)=0.0
      D(2)=0.0
      DO 23 I=1,2
      DO 23 J=1,4
   23 D(I)=D(I)+FF(IU,IW1,IW2,J,I)*Y(J)*SI(I,I)
      RETURN
      END
      FUNCTION PRIME(DATA)
COMMON
      DIMENSION DATA(5)
      DD=4.0
      PRIME=(-94.0*DATA(1)+33.0*DATA(2)+60.0*DATA(3)+47.0*DATA(4)-66.0*
     1DATA(5)+DD*(20.0*DATA(1)-10.0*DATA(2)-20.0*DATA(3)-10.0*DATA(4)+
     220.0*DATA(5)))/14.0
      RETURN
      END
      SUBROUTINE PLGEN(PLANE,PDOT,ERROR,PINC)
COMMON
C IF THE ERROR IS SMALLER THAN THE DEADZONE, SET 'PINC' TO 0
C AND RETURN
      IF (ABS(ERROR).LE.0.008727) GO TO 777
      PLANE=PLANE+PINC
C PLANE DEFLECTION LIMITER
      IF (ABS(PLANE).GT.0.6108) PLANE=SIGN(0.6108,PLANE)
C GET INCREMENT OF DEFLECTION FOR NEXT CALL TO 'PLGEN'
      PINC=PDOT*DELT*0.25
      IF (ERROR.LT.0.0) PINC=-PINC
      RETURN
```

```
  777 PINC=0.0
      RETURN
      END
//G.FT16F001 DD DSN=S0147.UBCCT3,UNIT=2321,VOL=SER=CEL008,
// DISP=SHR,LABEL=(,,,IN)
//G.FLXDATA DD *
      50.0       25.33          0.0          0.0          0.0          10.0          10.0     3     3
       0.0       17.00          0.0          0.0          0.0          10.0          10.0     3     3
//PLOT.SYSIN DD *
STERN PLANE ANGLE VS TIME
X:1 UNIT=1 SEC, Y:1 UNIT=1 DEGR                                                                      7
     -20.0        12.0                      5.0          6.0
FAIRWATER PLANE ANGLE VS TIME
X:1 UNIT=1 SEC, Y:1 UNIT=1 DEGR                                                                      7
     -20.0        12.0                      5.0          6.0
DEPTH VS TIME
X:1 UNIT=1 SEC, Y:1 UNIT=1 FT                                                                        7
      -1.0         1.0                      5.0          6.0
PITCH VS TIME
X:1 UNIT=1 SEC, Y:1 UNIT=1 DEGR                                                                      7
      -6.0         2.0                      5.0          6.0
SPEED LA VS TIME
X:1 UNIT=1 SEC, Y:1 UNIT=1 FT/SEC                                                                    7
      14.0         2.0                      5.0          6.0
```

```
// EXEC PLM80I,REGION=150K
//PASS1.INPUT5 DD DSN=F0963.DISKIO,DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//     UNIT=2314,VOL=SER=LINDA,DISP=SHR
//PASS1.INPUT7 DD DSN=F0167.SUB2,DCB=(RECFM=FB,LRECL=80,BLKSIZE=2240),
//     UNIT=2314,VOL=SER=DUFFY,DISP=SHR
//PASS1.CARD DD *
$T=0
$W=120
$R=80
   /*  CONTROL ALGORITHM IN PLM.                          */
   /*  INPUTS AND OUTPUTS ARE 16 BIT INTEGERS             */
   /*  WITH  8000 HEXADECIMAL REPRESENTING                */
   /*  0, VALUES BETWEEN 0 AND 8000 HEX                   */
   /*  REPRESENTING NEGATIVE VALUES, VALUES               */
   /*  BETWEEN 8000 HEX AND FFFF HEX REPRE-               */
   /*  SENTING POSITIVE VALUES.                           */
   /*                                                     */
   /*  INPUT TO THE CONTROLLER ARE:                       */
   /*  - DEPTH                                            */
   /*  - PITCH                                            */
   /*  - ORDERED DEPTH                                    */
   /*  - ORDERED PITCH                                    */
   /*  - SPEED UA                                         */
   /*  - SUBSCRIPTS IW1 AND IW2 AS THE CHOICE             */
   /*    FOR THE WEIGHTS OF THE CONTROLS.                 */
   /*  OUTPUTS ARE THE ORDERED DEFLECTIONS OF             */
   /*  THE CONTROL PLANES D(1) AND D(2).                  */
   /*                                                     */
1000H:
      DECLARE BDCS ADDRESS INITIAL (3FFDH);
      DECLARE (DEPTH,PITCH,PORD,ZORD,PERR,ZERR,DBORD,DSORD,MM,UA)
              ADDRESS;
      DECLARE (DPTH,PTCH) (15) BYTE INITIAL (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
   /* DECLARE AND INITIALIZE '4-DIMENSIONAL' GAIN ARRAY 'FF' */
      DECLARE F(48) BYTE INITIAL (08FH,0F4H,03FH,0FAH,07EH,03EH,
                                  0A8H,05AH,040H,0C0H,0D7H,041H,
                                  0CBH,068H,047H,0E5H,0C0H,0C7H,
                                  0D6H,02CH,045H,0D6H,0CEH,0CAH,
      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
      DECLARE CON0 (3) BYTE INITIAL (0E5H,02EH,051H);
                /* 117339.97 = 1CA5C. HEX */
      DECLARE CON1 ADDRESS INITIAL (5000H);
                /* 5000H = 10.0 DEGR */
      DECLARE CON2 ADDRESS INITIAL (0300H);
                /* 4.0 FT = 30.0 HEX */
      DECLARE CON3 (3) BYTE INITIAL (08EH,0FBH,03CH);
                /* 0.000008522 = 0.00008EFAC HEX */
      DECLARE C11(3)  BYTE INITIAL (0A2H,0E3H,04DH);
```

```
      DECLARE C2I(3)  BYTE INITIAL (0A2H,0E3H,04DH);
                /* 0.1*52151.1 = 145F.19 HEX */
   /* DECLARE AND INITIALIZE POINTER ARRAYS FOR FIRST THREE */
   /* DIMENSIONS OF THE GAIN ARRAY                          */
      DECLARE M1(18) ADDRESS INITIAL (0,0,0,0,0,0,0,0,60H,60H,60H,60H,
                                      60H,60H,60H,60H,60H,60H);
      DECLARE M2(5)  ADDRESS INITIAL (0,0,0,0,0,0);
      DECLARE M3(3)  BYTE  INITIAL (0,0,0);
      DECLARE FLAG BYTE INITIAL (0);
      DECLARE IW1  BYTE INITIAL (0);
      DECLARE IW2  BYTE INITIAL (0);
      DECLARE IU   BYTE INITIAL (0);
      DECLARE ERR BYTE INITIAL (0);
      DECLARE ZE BYTE, ZZ ADDRESS;
      DECLARE YE BYTE, XE BYTE;
      DECLARE MODE BYTE;
      DECLARE I BYTE;
      DECLARE J BYTE;
      DECLARE D(6) BYTE;
      DECLARE (TEMP,CON5)(3) BYTE;
      DECLARE Y(12) BYTE;
      DECLARE BUFF (130) BYTE;
      DECLARE CR LITERALLY '0CH';
      DECLARE LF LITERALLY '0AH';
MON1: PROCEDURE(F,A);
      DECLARE F BYTE,
      A ADDRESS;
      GO TO BDOS;
END MON1;
PRINTCHAR: PROCEDURE (CHAR);
      DECLARE CHAR BYTE;
      CALL MON1(2,CHAR);
END PRINTCHAR;
CRLF: PROCEDURE;
      CALL PRINTCHAR(CR);
      CALL PRINTCHAR(LF);
END CRLF;
PRINT:PROCEDURE(A);
      DECLARE A ADDRESS;
      /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
      NEXT DOLLAR SIGN IS ENCOUNTERT */
      CALL CRLF;
      CALL MON1(9,A);
END PRINT;
READ: PROCEDURE;
      BUFF=128;
      CALL MON1(10,.BUFF);
END READ;
```

```
   /*                                                          */
   /* PROCEDURE TO READ AN INTEGER GIVEN IN ASCII CODE         */
   /*                                                          */
INTREAD:PROCEDURE (XA);
      DECLARE XA ADDRESS,X BASED XA ADDRESS;
      DECLARE (I,K) BYTE;
      DECLARE L ADDRESS;
      DO I=2 TO 129;
        BUFF(I)=' ';
      END;
      CALL CRLF;
      CALL READ;
      I=2;
      DO WHILE ((BUFF(I)=' ') AND (I<129));
      I=I+1;
      END;
      K=I;
   /* LOOK FOR HEXADECIMAL DIGIT */
      DO WHILE (((BUFF(K)>=30H) AND (BUFF(K)<=39H)) OR
           ((BUFF(K)>=41H) AND (BUFF(K)<=46H)));
         IF (((BUFF(K)<30H) OR (BUFF(K)>46H)) OR ((BUFF(K)>39H) AND
         (BUFF(K)<41H))) THEN DO;
                                 CALL PRINT(.'ERROR, INVALID INPUT CHARACTER$');
                                 ERR=1;
                                 RETURN;
                               END;
         K=K+1;
      END;
      IF (K-I)>4 THEN DO;
                        CALL PRINT (.'ERROR, INPUT OVERFLOW$');
                        X=0FFFFH;
                        ERR=1;
                        RETURN;
                      END;
      K=K-1;
      L=1;
      X=0;
   /* CONVERT ASCII CODE TO BINARY */
      DO WHILE K>=I;
         IF (BUFF(K)<=39H) THEN X=X+(BUFF(K)-30H)*L;
        ELSE X=X+(BUFF(K)-37H)*L;
        K=K-1;
        L=L*16;
      END;
      ERR=0;
      RETURN;
END INTREAD;
   /*                                                          */
```

```
   /* RCLTINE TO PRINT NUMBER IN HEXADECIMAL INTEGER REPRESENTATION */
   /*                                                               */
INTFRNT:PROCEDURE(XA);
      DECLARE PBUFF(5) BYTE INITIAL (0,0,0,0,'$');
      DECLARE XA ADDRESS,X BASED XA ADDRESS;
      DECLARE (Y,Z,I) BYTE;
      Y=HIGH(X);
      Z=LOW(X);
      PBUFF(0)=SHR(Y,4) AND 0FH;
      PBUFF(1)=Y AND 0FH;
      PBUFF(2)=SHR(Z,4) AND 0FH;
      PBUFF(3)=Z AND 0FH;
      DO I=0 TO 3;
         IF PBUFF(I)<10 THEN PBUFF(I)=PBUFF(I)+30H;
                        ELSE PBUFF(I)=PBUFF(I)+37H;
      END;
      CALL PRINT(.PBUFF);
      RETURN;
END INTFRNT;
   /*                                                               */
   /*  PROCEDURE TO PRINT FLOATING POINT NUMBER IN HEX FORMAT       */
   /*                                                               */
FLTFRNT: PROCEDURE(XA);
         DECLARE BUFF(9) BYTE INITIAL(00,00,00,00,' E',00,00,'$');
         DECLARE XA ADDRESS, X BASED XA BYTE;
          DECLARE I BYTE;

         BUFF(0) = (SHR((X AND 0F0H),4));
         BUFF(1) = (X AND 0FH);
         BUFF(2) = (SHR((X(1) AND 0F0H),4));
         BUFF(3) = (X(1) AND 0FH);
         BUFF(6) = (SHR((X(2) AND 0F0H),4));
         BUFF(7) = (X(2) AND 0FH);
   /* CONVERT CHARACTERS FROM BINARY TO ASCII */
         DO I=0 TO 3;
              IF BUFF(I) < 10 THEN BUFF(I) = BUFF(I) + 30H;
                              ELSE BUFF(I) = BUFF(I) + 37H;
              END;

         DO I=6 TO 7;
              IF BUFF(I) < 10 THEN BUFF(I) = BUFF(I) + 30H;
                              ELSE BUFF(I) = BUFF(I) + 37H;
              END;

         CALL PRINT (.BUFF);

         RETURN;
         END FLTFRNT;
```

```
  /*                                    */
  /* FLOATING POINT ADD ROUTINE */
  /*                                    */
ADD: PROCEDURE (XA,YA,ZA);
        DECLARE (XA,YA,ZA,XX,YY) ADDRESS,
             (I,XE,YE,RANGE,SIGNSEQUAL) BYTE,
             X BASED XA BYTE,
             Y BASED YA BYTE,
             Z BASED ZA BYTE;

  /*                                                    */
  /* PROCEDURE TO LEFT JUSTIFY MANTISSA IN BINARY */
  /*                                                    */
    ADJUST: PROCEDURE;
         DO I=0 TO 15;
             IF HIGH(ZZ)>07H THEN RETURN;
             IF ((ZE AND 7FH) = 0H) THEN DO;
                  ZE=0H;   ZZ=0H;   RETURN;
                  END;
                  ZE = ZE-1;
                  ZZ = SHL(ZZ,1);
         END;
    END ADJUST;

  /* DETERMINE DIFFERENCE IN EXPONENTS */
    IF (XE:=X(2) AND 7FH)>(YE:=Y(2) AND 7FH) THEN RANGE=XE-YE;
    ELSE RANGE=YE-XE;

  /* CHECK TO SEE IF NUMBERS ARE WITHIN SIGNIFICANCE RANGE */
    IF RANGE>15 THEN DO;

  /* VARIABLES NOT WITHIN SIGNIFICANCE RANGE */
         IF XE>YE THEN DO;
              Z=X; Z(1)=X(1); Z(2)=X(2); RETURN;
              END;
         Z=Y; Z(1)=Y(1); Z(2)=Y(2); RETURN;
         END;

  /* VARIABLES ARE WITHIN RANGE OF SIGNIFICANCE */

  /* FORM MANTISSAS */
    XX=SHL(DOUBLE(X),8) OR X(1);
    YY=SHL(DOUBLE(Y),8) OR Y(1);
    IF (X(2) AND 80H)=(Y(2) AND 80H) THEN SIGNSEQUAL=1;
```

```
      ELSE SIGNSEQUAL=0;

/* EXPONENTS EQUAL */
  IF YE=XE THEN DO;

/* Y>X */
      IF YY>XX THEN DO;
           ZE= Y(2);
           IF SIGNSEQUAL THEN GO TO EXIT1;
           GO TO EXIT2;
      END;

/* X>Y */
      IF YY<XX THEN DO;
           ZE =X(2);
           IF SIGNSEQUAL THEN GO TO EXIT1;
           GO TO EXIT3;
      END;

/* X=Y */
      IF SIGNSEQUAL THEN DO;
           Z=X; Z(1)=X(1); Z(2)=X(2)+1;
           RETURN;
           END;
      Z=0; Z(1)=0; Z(2)=0;
      RETURN;
      END;


/* EXPONENT OF X > EXPONENT OF Y */
  IF XE>YE THEN DO;
      ZE=X(2);
      YY=SHR( YY,RANGE );
      IF SIGNSEQUAL THEN GO TO EXIT1;
      GO TO EXIT3;
      END;


/* EXPONENT OF Y > EXPONENT OF X */
      ZE=  Y(2);
      XX=SHR(XX,RANGE);
      IF SIGNSEQUAL THEN GO TO EXIT1;
      GO TO EXIT2;


/* WHEN SIGNS OF THE MANTISSAS ARE EQUAL THE */
/* NUMBERS ARE ADDED                         */
```

```
EXIT1:
    ZZ=XX+YY;
    IF CARRY THEN DO;
        ZZ=SCR(ZZ,1);
        ZE=ZE  +1;
        END;
        Z=HIGH(ZZ); Z(1)=LOW(ZZ); Z(2)=ZE;
        RETURN;

  /* WHEN SIGNS ARE DIFFERENT AND Y>X */
EXIT2:
        ZZ=YY-XX;
        CALL ADJUST;
        Z=HIGH(ZZ); Z(1)=LOW(ZZ); Z(2)=ZE;
        RETURN;

  /* WHEN SIGNS ARE DIFFERENT AND X>Y */
EXIT3:
        ZZ=XX-YY;
        CALL ADJUST;
        Z=HIGH(ZZ); Z(1)=LOW(ZZ); Z(2)=ZE;
        RETURN;
END ADD;
  /*                                          */
  /* FLOATING POINT SUBTRACTION ROUTINE */
  /*                                          */
SUB: PROCEDURE (XA,YA,ZA);
    DECLARE (XA,YA,ZA) ADDRESS,
        YY BASED YA BYTE,
        YYMINUS (3) BYTE;

    YYMINUS=YY;
    YYMINUS(1)=YY(1);
    YYMINUS(2)=YY(2) XOR 80H;
    CALL ADD(XA,.YYMINUS,ZA);
END SUB;
MULT: PROCEDURE(XA,YA,ZA);
    DECLARE (XA,YA,ZA,XX,YY) ADDRESS;
    DECLARE (SAVE,I) BYTE,
        X BASED XA BYTE,
        Y BASED YA BYTE,
        Z BASED ZA BYTE;

  /* IF EITHER NUMBER IS ZERO */
```

```
    IF (X=0) OR (Y=0) THEN DO;
              Z=0; Z(1)=0; Z(2)=0; RETURN; END;

  /* NUMBERS NON-ZERO */
    XX=SHL(DOUBLE(X),8) OR X(1);
    YY=SHL(DOUBLE(Y),8) OR Y(1);
    ZZ=7FFFH;

    DO I=0 TO 14 ;
         YY= SCR(YY,1);
         IF (CARRY) THEN ZZ=ZZ+XX;
         ZZ=SCR(ZZ,1);
    END;

    ZZ=ZZ+XX;

    IF (CARRY) THEN DO;
         ZZ=SCR(ZZ,1);
         ZE=0;
         END;
         ELSE ZE=-1;


  /* ADD EXPONENTS  */
    SAVE=(X(2) AND 7FH)+(Y(2) AND 7FH)-40H+ZE;
    IF SAVE>7FH THEN DO;
    Z=0; Z(1)=0; Z(2)=0;
    RETURN;  END;
    Z(2)=SAVE OR ((X(2) XOR Y(2)) AND 80H);
    Z=HIGH(ZZ);
    Z(1)=LOW(ZZ);
    RETURN;
END MULT;


  /*                                                    */
  /* PRINT A FLOATING POINT NUMBER WITH SIGN            */
  /* AND DECIMAL POINT                                  */
  /*      XA=ADDRESS OF VARIABLE                        */
  /*      PT=NR OF DIGITS TO THE RT OF THE              */
  /*         DECIMAL POINT                              */
  /* NOTE: INPUT VARIABLE MUST BE LESS THAN             */
  /*              65536.                                */
  /* FIELD WILL BE 7 SPACES WIDE                        */
  /*           0 <= PT <= 5                             */
  /*                                                    */
```

```
EXPNTPRNT: PROCEDURE (XA,PT);
    DECLARE (XA,DEC) ADDRESS,
          X BASED XA BYTE,
          Y(3) BYTE,
          HALF DATA (80H,00H,40H),
          BUFFER (8) BYTE,
          (RADIX,I,N,ZSUP,SYMBOL,PT)BYTE,

          RATIO DATA (0A0H,00H,44H,0C8H,00H,47H,
          0FAH,00H,4AH,9CH,40H,4EH,0C3H,50H,51H);

    IF ((X(2) AND 7FH)>50H) THEN GO TO STAR;

    IF PT>0 THEN DO;
          N=3*(PT-1);
          CALL MULT (XA,.RATIO(N),.Y);
          END;
    ELSE DO;
          Y=X; Y(1)=X(1);  Y(2)=X(2);
          END;


    IF Y(2)<80H THEN CALL ADD (.Y,.HALF,.Y);
    ELSE CALL SUB (.Y,.HALF,.Y);


    YE=Y(2) AND 7FH;
    IF YE > 50H THEN GO TO STAR;

    RADIX=6-PT;
    BUFFER=' ';
    BUFFER(RADIX)='.';
    SYMBOL=0;
    ZZ=SHL(DOUBLE(Y),8) OR Y(1);


    IF YE<50H THEN DO;
          YE=50H - YE;
          ZZ=SHR(ZZ,YE);
          END;


    IF RADIX=1 THEN DO;
          ZSUP=0;
          N=2;
          END;
    ELSE DO;
```

```
            ZSUP=1;
            N=1;
            END;

        IF ZZ=0 THEN DO;
            DO WHILE (N+1)<RADIX;
                BUFFER(N)=' ';
                N=N+1;
                END;
            GO TO ZEROS;
            END;

        DEC=2710H;
        DO I=1 TO 5;
            IF N=RADIX THEN DO;
                N=N+1;
                ZSUP=0;
                END;
            BUFFER(N)=ZZ/DEC + '0';
            ZZ=ZZ MOD DEC;
            DEC=DEC/10;

            IF ZSUP THEN DO;
                IF BUFFER(N)='0' THEN DO;
                    SYMBOL=N;
                    BUFFER(N)=' ';
                    END;
                ELSE ZSUP=0;
            END;

            N=N + 1;
        END;

        IF (X(2)>=80H) THEN BUFFER(SYMBOL)='-';

ZEROS:  DO WHILE N<7;
            IF N=RADIX THEN N = N + 1;
            BUFFER(N)='0';
            N=N + 1;
        END;
```

```
EXIT: BUFFER(7)='$';
    CALL PRINT (.BUFFER);
    RETURN;

STAR: DO N=0 TO 6;
            BUFFER(N)='*';
            END;
        GO TO EXIT;
END EXPNTPRNT;
   /*                                                          */
   /* ROUTINE TO NORMALIZE AN UNNORMALIZED BINARY FRACTION     */
   /*                                                          */
NORMALIZE:PROCEDURE(WWA,ZZA);
      DECLARE (ZZA,WWA,Y) ADDRESS;
      DECLARE (WW BASED WWA) ADDRESS;
      DECLARE (ZZ BASED ZZA)(3) BYTE;
      DECLARE COUNT BYTE;
      Y=WW;
      COUNT=0;
      DO WHILE CARRY=0;
                  Y=SHL(Y,1);
                  COUNT=COUNT+1;
               END;
      IF (COUNT-1)<>0 THEN Y=SHL(WW,COUNT-1);
                      ELSE Y=WW;
      ZZ(0)=HIGH(Y);
      ZZ(1)=LOW(Y);
      ZZ(2)=51H-COUNT;
      RETURN;
END NORMALIZE;
   /*                                                          */
   /* ROUTINE TO READ A MAXIMUM OF  5 SIGNIFICANT DIGITS  OF A */
   /* FIXED POINT DECIMAL NUMBER AND CONVERT IT INTO FLOATING  */
   /* POINT FORMAT                                             */
   /*                                                          */
EXPNTREAD:PROCEDURE(ZA);
      DECLARE (I,J,K)BYTE;
      DECLARE (ZA,M) ADDRESS;
      DECLARE X ADDRESS;
      DECLARE U(3) BYTE;
      DECLARE (Z BASED ZA)(3) BYTE;
      DECLARE (S,EXP) BYTE;
   /* DECLARE AND INITIALIZE ARRAY 'DEC'  WITH 10 EXP I, I=-10 TO 5 */
   /* IN FLOATING POINT FORMAT                                      */
      DECLARE DEC(60) BYTE INITIAL
      (00CH,0C0H,00H,
       (89H,070H,23H,0ABH,0CCH,26H,0D6H,0C0H,29H,086H,038H,20H,0A7H,0C6H,30H,
```

```
    0D1H,0B7H,33H,083H,012H,37H,0A3H,0D7H,3AH,0CCH,0CCH,3DH,080H,000H,41H,
    0A0H,000H,44H,0C8H,000H,47H,0FAH,000H,4AH,09CH,040H,4EH,0C3H,050H,51H,
    0F4H,024H,54H,098H,096H,58H,0BEH,0BCH,50H,CEEH,06BH,5EH);
/* DECLARE AND INITIALIZE ARRAY 'A' WITH 1/2*J EXP I FOR I=1 TO  */
/* 5 AND J=0 TO 9                                                */
   DECLARE A(50) ADDRESS INITIAL
   (0,0199AH,03333H,04CCDH,06666H,08000H,0999AH,0B333H,0CCCDH,0E666H,
    0,0028FH,0051FH,007AEH,00A3DH,00CCDH,00F5CH,011ECH,0147BH,0170AH,
    0,00041H,00083H,000C5H,00106H,00148H,0018BH,001CBH,0020CH,0024EH,
    0,00007H,0000DH,00014H,0001AH,00021H,00027H,0002EH,00034H,0003BH,
    0,00001H,00001H,00002H,00003H,00004H,00005H,00005H,00006H);
   DO I=2 TO 129;
     BUFF(I)=' ';
   END;
   S=0;
   X=0;
   K=0;
   EXP=30;
   CALL CRLF;
   CALL READ;
   I=2;
/* LOOK FOR '-'-SIGN AND LEADING ZEROS */
   DO WHILE (((BUFF(I)=' ') OR (BUFF(I)='0')) AND (I<129));
     I=I+1;
   END;
   IF (BUFF(I)='-') THEN DO;
                           S=1;
                           I=I+1;
                         END;
   DO WHILE (((BUFF(I)=' ') OR (BUFF(I)='0')) AND (I<129));
     I=I+1;
   END;
   J=0;
/* CHECK FOR DEC. POINT */
   IF BUFF(I)='.' THEN DO;
                         I=I+1;
/* KEEP ENTRY 'EXP' FOR LOOKUP IN DECIMAL MULTIPLIER ARRAY 'DEC' */
                         DO WHILE ((BUFF(I)='0') AND (EXP>=0) AND (I<129));
                           EXP=EXP-3;
                           I=I+1;
                         END;
/* CONVERT ASCII-CODE INTO UNNORMALIZED BINARY FRACTION   */
/* BY TABLE LOOKUP IN ARRAY 'A'                           */
                         DO WHILE ((BUFF(I)<=39H) AND (BUFF(I)>=30H) AND
                                   (J<50) AND (I<129));
                           X=X+A(BUFF(I)-30H+J);
                           J=J+10;
                           I=I+1;
```

```
                                END;
                                GO TO FINAL;
                              END;
   /* CONVERT ASCII-CODE INTO UNNORMALIZED BINARY FRACTION   */
   /* BY TABLE LOOKUP IN ARRAY 'A' AND KEEP POSITION OF DEC. */
   /* POINT FOR ENTRY IN DEC. TABLE LOOKUP ARRAY 'DEC'       */
      DO WHILE ((((BUFF(I)<=39H) AND (BUFF(I)>=30H)) OR (BUFF(I)='.')) AND
               (I<129) AND (K<=30));
        IF (BUFF(I)<>'.') THEN DO;
                                  IF (J<50) THEN DO;
                                               X=X+A(BUFF(I)-30H+J);
                                               J=J+10;
                                                 END;
                                               K=K+1;
                                             END;
                                        ELSE EXP=EXP+K;
        I=I+1;
      END;
      FINAL:
      IF (X<>0) THEN DO;
   /* NORMALIZE BINARY FRACTION AND MULTIPLY IT WITH DEC. MULTIPLIER */
                        CALL NORMALIZE (.X,.U);
                        U(2)=U(2)-10H;
                        CALL MULT(.U,.DEC+EXP,.Z);
                      END;
      IF S=1 THEN Z(2)=Z(2) OR 80H;
      RETURN;
ENC FXFNTREAL;
   /*                                                    */
   /* ROUTINE TO CONVERT INTEGER IN BINARY INTO          */
   /* FLOATING POINT FORMAT                              */
   /*                                                    */
FLOAT:PROCEDURE (XA,ZA);
      DECLARE (XA,ZA,Y,W) ADDRESS;
      DECLARE S BYTE INITIAL (0);
      DECLARE (X BASED XA) ADDRESS;
      DECLARE (Z BASED ZA)(3) BYTE;
      IF X=8000H THEN DO;
                        Z(0)=0;
                        Z(1)=0;
                        Z(2)=0;
                        RETURN;
                      END;
      S=0;
      IF X<8000H THEN DO;
                        S=1;
                        W=8000H-X;
                      END;
```

```
      ELSE W=X-8000H;
      CALL NORMALIZE(.W,.Z);
      IF S=1 THEN Z(2)=Z(2) OR 80H;
      RETURN;
END FLOAT;
   /*                                                    */
   /* ROUTINE TO CONVERT NUMBER IN FLOATING POINT        */
   /* FORMAT INTO NUMBER IN INTEGER FORMAT               */
   /*                                                    */
INTEG:PROCEDURE (ZA,XA);
      DECLARE (XA,ZA) ADDRESS;
      DECLARE (S,Y) BYTE;
      DECLARE (X BASED XA) ADDRESS;
      DECLARE (Z BASED ZA) (3) BYTE;
      S=0;
      IF (Z(2) AND 7FH)<40H THEN DO;
                                 X=8000H;
                                 RETURN;
                               END;
      IF (Z(2) AND 80H)=80H THEN S=1;
      Y=Z(2) AND 7FH;
      Y=Y-40H;
      IF Y>10H THEN DO;
                      IF S=0 THEN X=0FFFFH;
                      IF S=1 THEN X=0;
                      RETURN;
                    END;
      X=SHL(DOUBLE(Z(0)),8) OR Z(1);
      IF (16-Y)<>0 THEN X=SHR(X,16-Y);
      IF S=0 THEN X=X+8000H;
      IF S=1 THEN X=8000H-X;
      RETURN;
END INTEG;
   /*                                                    */
   /* ROUTINE TO COMPUTE THE RATE OF CHANGE  IN          */
   /* DEPTH OR PITCH BY A SECOND DEGREE LEGENDRE         */
   /* POLYNOMIAL WITH 5 POINTS                           */
   /*                                                    */
PRIME:PROCEDURE (PDATA,PDOT);
      DECLARE (PDATA,PDOT) ADDRESS;
      DECLARE (SDATA BASED PDATA) (15) BYTE;
      DECLARE (DOT BASED PDOT) (3) BYTE;
      DECLARE TEMP(3) BYTE;
      DECLARE C(15) BYTE INITIAL(0CEH,0B4H,0C2H,
                                 0ACH,0E8H,0C2H,
                                 08FH,0A7H,044H,
                                 0DEH,0E0H,0C1H,
                                 0A8H,03EH,0C1H);
```

```
      DO I=0 TO 2;
         DOT(I)=0;
      END;
      DO I=0 TO 12 BY 3;
         CALL MULT(.SDATA+I,.C(I),.TEMP);
         CALL ADD(.TEMP,.DOT,.DOT);
      END;
      RETURN;
END PRIME;
      CALL PRINT(.'MODE=$');
      CALL INTREAD(.MODE);
      IF (MODE=1) THEN GO TO DECINP;
   /* INPUT IN HEXADECIMAL FORMAT */
   R1:CALL PRINT(.'UA=$');
      CALL INTREAD(.UA);
      IF ERR=1 THEN GO TO R1;
   R2:CALL PRINT(.'PITCH=$');
      CALL INTREAD(.PITCH);
      IF ERR=1 THEN GO TO R2;
   R3:CALL PRINT(.'PORD=$');
      CALL INTREAD(.PORD);
      IF ERR=1 THEN GO TO R3;
   R4:CALL PRINT(.'DEPTH=$');
      CALL INTREAD(.DEPTH);
      IF ERR=1 THEN GO TO R4;
   R5:CALL PRINT(.'ZORD=$');
      CALL INTREAD(.ZORD);
      IF ERR=1 THEN GO TO R5;
      GO TO ENDDECINP;
   /* INPUT IN FIXED POINT DECIMAL FORMAT */
DECINP:
      CALL PRINT(.'UA=$');
      CALL FXPNTREAD(.UA);
      CALL FXPNTPRNT(.UA,2);
      CALL PRINT(.'PITCH=$');
      CALL FXPNTREAD(.TEMP);
      CALL FXPNTPRNT(.TEMP,3);
      CALL MULT(.TEMP,.CONO,.TEMP);
      CALL INTEG(.TEMP,.PITCH);
      CALL INTPRNT(.PITCH);
      CALL PRINT(.'PORD=$');
      CALL FXPNTREAD(.TEMP);
      CALL FXPNTPRNT(.TEMP,3);
      CALL MULT(.TEMP,.CONO,.TEMP);
      CALL INTEG(.TEMP,.PORD);
      CALL INTPRNT(.PORD);
      CALL PRINT(.'DEPTH=$');
      CALL FXPNTREAD(.TEMP);
```

```
      CALL FXPNTPRNT(.TEMP,3);
      CALL INTEG(.TEMP,.DEPTH);
      CALL INTPRNT(.DEPTH);
      CALL PRINT(.'ZORD=$');
      CALL FXPNTREAD(.TEMP);
      CALL FXPNTPRNT(.TEMP,3);
      CALL INTEG(.TEMP,.ZORD);
      CALL INTPRNT(.ZORD);
ENDDECINF:
   /* TRUNCATE UA TO INTEGER BETWEEN 0 AND 18 */
      UA= UA-8000H;
      IF UA<0 THEN UA=0;
      UA= SHL(UA,4);                              /*MULTIPLY UA BY 16*/
      UA=UA/18H;       /*DIVIDE UA BY 1.689*16=27.02=18 HEX*/
   /* GET ERROR IN PITCH AND DEPTH */
      IF PITCH<PORD THEN DO;
                           PERR=PORD-PITCH;
                           FLAG=1;
                         END;
                    ELSE PERR=PITCH-PORD;
      IF PERR>CON1 THEN PERR=CON1;
      IF FLAG=0 THEN PERR=PERR+8000H;
                ELSE PERR=8000H-PERR;
      FLAG=0;
      IF DEPTH<ZORD THEN DO;
                           ZERR=ZORD-DEPTH;
                           FLAG=1;
                         END;
                    ELSE ZERR=DEPTH-ZORD;
      IF ZERR>CON2 THEN ZERR=CON2;
      IF FLAG=0 THEN ZERR=ZERR+8000H;
                ELSE ZERR=8000H-ZERR;
      FLAG=0;
   /* UPDATE ARRAYS 'PTCH' AND 'DPTH' */
      DO  I=0  TO 11;
          DPTH(I)=DPTH(I+3);
          PTCH(I)=PTCH(I+3);
      END;
   /* INITIALIZE ARRAY 'Y' */
      CALL FLOAT (.PERR,.Y+6);
      CALL MULT (.Y+6,.CON3,.Y+6);
      CALL FLOAT (.ZERR,.Y);
      CALL FLOAT (.DEPTH,.DPTH+12);
      CALL FLOAT (.PITCH,.PTCH+12);
      CALL MULT (.PTCH+12,.CON3,.PTCH+12);
      CALL PRIME (.DPTH,.Y+3);
      CALL PRIME (.PTCH,.Y+9);
      CALL PRINT(.'PICT=$');
```

```
      CALL FLTPRNT(.Y+9);
      CALL PRINT(.'ZDOT=$');
      CALL FLTPRNT(.Y+3);
      MM=M1(IL)+M2(IW1)+M3(IW2);
   /* SUM UP PRODUCTS OF GAINS 'F' AND VALUES OF 'Y' TO GET */
   /* PLANE DEFLECTIONS D(1) AND D(2)                       */
      DO I=0 TO 5;
         D(I)=0;
      END;
      DO I=0 TO 3 BY 3;
         DO J=0 TO 18 BY 6;
            CALL MULT(.F(MM)+J+I,.Y+J/2,.TEMP);
            CALL ADD(.TEMP,.D+I,.D+I);
         END;
      END;
      CALL PRINT(.'D(1)=$');
      CALL FLTPRNT(.D);
      CALL FXFNTPRNT(.D,3);
      CALL PRINT(.'D(2)=$');
      CALL FLTPRNT(.D+3);
      CALL FXFNTPRNT(.D+3,3);
   /*  MULTIPLY'D' REPRESENTING DEFLECTION IN RADIANS */
   /*  WITH 'CI', THE FACTOR WHICH CONSISTS OF CONTROL*/
   /*  WEIGHT AND FACTOR TO CONVERT RADIANS TO 16 BIT */
   /*  INTEGER REPRESENTATION                         */
      CALL MULT (.D,.CII,.D);
      CALL MULT (.D+3,.C2I,.D+3);
      CALL INTEG (.D+3,.DBORD);
      CALL INTEG (.D,.DSCRD);
      CALL PRINT(.'DBORD=$');
      CALL INTPRNT(.DBORD);
      CALL PRINT(.'DSORD=$');
      CALL INTPRNT(.DSORD);
EOF
//PASS2.CARD DD *
$B=7 $C=28 $W=120 $T=0 $F=1 $G=1 $S=0 $M=1
```

REFERENCES

1. Naval Ship Research and Development Center, Bethesda, Maryland 20034, Report No. P-433-H-01, User's Guide NSRDC Digital Programs for Simulating Submarine Motion, ZZMN-Revision 1.0, by Ronald W. Richard, June 1971.
2. Naval Ship Research and Development Center, Bethesda, Maryland 20034, Report No. 2510, Standard Equations of Motion for Submarine Simulation, by Morton Gentler and Grant R. Hagen, June 1967.
3. Kirk, D.E. Optimal Control Theory, Prentice-Hall, 1970
4. NAVSHIPS 0911-003-6010, Fundamentals of Submarine Hydrodynamics, Motion and Control.
5. DSL/360 Digital Simulation Language User Manual, IBM System 1360, by W. M. Syn, N. N. Turner, D. G. Wyman, November 1968 (informal).
6. Drurey, H. L. Automatic Control of Submarine Depth, Pitch and Trim, Masters Thesis, Naval Postgraduate School, 1975.
7. McCalla, T. R. Introduction to Numerical Methods and FORTRAN Programming, Wiley, 1967.
8. Dorn, W. S., and McCracken, D. D. Numerical Methods with FORTRAN IV Case Studies, Wiley, 1972.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 72 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4.	Professor U.R. Kodres, Code 72KR Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	Marineamt -A1- 294 Wilhelmshaven Federal Republic of Germany	1
6.	Dokumentationszentrale der Bundeswehr (See) 53 Bonn Friederich-Ebert-Allee 34 Federal Republic of Germany	1
7.	Professor D.E. Kirk, Code 52KI Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
8.	Professor G. Thaler, Code 52TR Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
9.	LCDR Heinz-Dieter Dockhorn, FGN Wiker Strasse 6 234 Kappeln/Ellenberg Federal Republic of Germany	2