

DTIC FILE COPY

2

NPS69-88-007

# NAVAL POSTGRADUATE SCHOOL Monterey, California

AD-A200 824



## THESIS

LOCAL PATH PLANNING USING OPTIMAL  
CONTROL TECHNIQUES

by

Winston Smith

June 1988

Thesis Advisor:

David L. Smith

DTIC  
ELECTE  
DEC 01 1988  
S D  
CE

Approved for public release; distribution unlimited

Prepared for: Naval Postgraduate School  
Monterey, California 93943-5000

88 12 1 017

NAVAL POSTGRADUATE SCHOOL  
Monterey, CA 93943

Rear Admiral R. C. Austin  
Superintendent

Kneale T. Marshall  
Acting Provost

This thesis was prepared in conjunction with research sponsored by the Arctic Submarine Laboratory, Naval Ocean Systems Center, San Diego, California, and funded by the Naval Postgraduate School. Reproduction of all or part of this report is authorized.

Released by:



---

GORDON E. SCHACHER  
Dean of Science and Engineering

## REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION <b>UNCLASSIFIED</b>		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release: distribution unlimited	
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S)  NPS69-88-007		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	
8a. NAME OF FUNDING SPONSORING ORGANIZATION Naval Postgraduate School	8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER  O&MN Direct Funding	
8c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000		10. SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO	PROJECT NO
		TASK NO	WORK UNIT ACCESSION NO
11. TITLE (Include Security Classification) Local Path Planning using Optimal Control Techniques UNCLASSIFIED			
12. PERSONAL AUTHOR(S) Winston Smith			
13a. TYPE OF REPORT Masters thesis	13b. TIME COVERED FROM TO	14. DATE OF REPORT (Year, Month, Day) June 1988	15. PAGE COUNT 109
16. SUPPLEMENTARY NOTATION "The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U. S. Government."			
17. COSAT <sup>T</sup> CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP	
		Path planning, obstacle avoidance, autonomous vehicle maneuvering	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) The ability of an autonomous vehicle control system to plan a safe, collision-free local path from one vehicle position to another is one of the most important functions. In this thesis, it is shown how a safe obstacle-free local path can be planned using optimal control theory and optimization techniques. The problem is posed as a two point boundary value problem with various problem constraints which control the vehicle behavior in transversing from one point to another. The objective function being minimized is a control performance index which includes vehicle energy saving parameters. Numerous fixed and moving obstacles in the dive plane are introduced and successfully avoided using this technique. Three-dimensional path planning is also successfully demonstrated on a 12 state linear model of an underwater vehicle. This technique is shown to be a feasible method for a local path planning applications. (KR)			
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Smith		22b. TELEPHONE (Include Area Code) 408-646-3383	22c. OFFICE SYMBOL 695m

Approved for public release; distribution unlimited

**Local Path Planning Using Optimal  
Control Techniques**

by

Winston Smith  
Lieutenant, United States Navy  
B.S., University of Mississippi, 1980


Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN MECHANICAL ENGINEERING**

from the

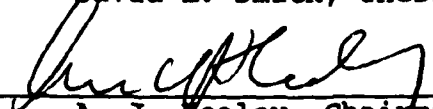
NAVAL POSTGRADUATE SCHOOL  
June 1988


Author:

  
Winston Smith

Approved by:

  
David L. Smith, Thesis Advisor

  
A. J. Kealey, Chairman  
Department of Mechanical Engineering

  
Gordon E. Schacher,  
Dean of Science and Engineering

ABSTRACT

The ability of an autonomous vehicle control system to plan a safe, collision-free local path from one vehicle position to another is one of the most important functions. In this thesis, it is shown how a safe obstacle-free local path can be planned using optimal control theory and optimization techniques. The problem is posed as a two point boundary value problem with various problem constraints which control the vehicle behavior in transversing from one point to another. The objective function being minimized is a control performance index which includes vehicle energy saving parameters. Numerous fixed and moving obstacles in the dive plane are introduced and successfully avoided using this technique. Three dimensional path planning is also successfully demonstrated on a 12 state linear model of an underwater vehicle. This technique is shown to be a feasible method for local path planning applications.



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input checked="" type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

## THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they can not be considered validated. Any application of these programs without additional verification is at the risk of the user.

## TABLE OF CONTENTS

I.	INTRODUCTION -----	1
	A. GENERAL -----	1
	B. PREVIOUS WORK -----	2
	C. AIM OF THE PRESENT STUDY -----	6
II.	METHOD OF APPROACH -----	7
III.	OBSTACLE AVOIDANCE -----	8
IV.	OPTIMIZATION OPTIONS -----	12
	A. CLUSTERED FIXED OBSTACLE TEST -----	12
	B. IMPOSSIBLE FIELD TEST -----	19
	C. SELECTION RESULT -----	21
V.	EVALUATION OF MANEUVERING TIME (FINTIM)-----	23
VI.	PROGRAMMING FOR THREE DIMENSIONS -----	26
	A. SIDE CONSTRAINTS -----	26
	B. EQUALITY CONSTRAINTS -----	27
	C. CONSTRAINT SCALING -----	27
	D. LINEAR/NONLINEAR DYNAMICS -----	28
	E. THREE-DIMENSIONAL COMPUTATIONAL COSTS -----	33
VII.	VALIDATION RUNS -----	34

VIII. CONCLUSIONS AND RECOMMENDATIONS -----	47
A. CONCLUSIONS -----	47
B. RECOMMENDATIONS -----	48
APPENDIX (PROGRAMS) -----	49
REFERENCES -----	93
INITIAL DISTRIBUTION LIST -----	96



LIST OF TABLES

Table 1.	ADS Level Options -----	14
Table 2.	17 Obstacle Test Results -----	17
Table 3.	FINTIM Computational Cost -----	24
Table 4.	Constraint Scaling Factors -----	27
Table 5.	Linear vs. Nonlinear Computational Costs ----	33
Table 6.	Program 311 Computational Cost-2D -----	46

## LIST OF FIGURES

Figure 1.1	Global Planning System -----	3
Figure 1.2	Discrete vs. Continuous Controls -----	5
Figure 3.1	Fixed Obstacle Computational Results -----	11
Figure 4.1	Allowable ADS Algorithm Combinations -----	13
Figure 4.2	Impossible Test Algorithm Comparison -----	20
Figure 4.3	Solution of Impossible Test with ----- Increased FINTIM	22
Figure 5.1	FINTIM Effects -----	25
Figure 6.1	Nonlinear Model Maneuver with Linear ----- Control Inputs	29
Figure 6.2	Bow and Stern Plane Control Inputs -----	30
Figure 6.3	Rudder Control Input -----	31
Figure 6.4	Linear Model Maneuver -----	32
Figure 7.1	One Obstacle Solution -----	35
Figure 7.2	Two Obstacles Solution -----	36
Figure 7.3	Three Obstacles Solution -----	37
Figure 7.4	Four Obstacles Solution -----	38
Figure 7.5	Five Obstacles Solution -----	39
Figure 7.6	Six Obstacles Solution -----	40
Figure 7.7	Nine Obstacles Solution -----	41
Figure 7.8	Seventeen Obstacles Solution -----	42
Figure 7.9	One Moving Obstacle Solution -----	43
Figure 7.10	Two Moving Obstacle Solution (Obstacle 1)-	44
Figure 7.11	Two Moving Obstacle Solution (Obstacle 2)-	45

---

#### ACKNOWLEDGMENT

The author would like to extend his deepest appreciation and thanks to Associate Professor David L. Smith for his continuing encouragement and enthusiasm expressed during the conduct of this research.

A special and well deserved note of appreciation is extended to my entire family for their continuous support throughout the many months of thesis research.

## I. INTRODUCTION

### A. GENERAL

The presently forecast missions of an Autonomous Underwater Vehicle (AUV) vary in scope from mine detection and avoidance to surveying the bottom of oceans. Further, it is expected that many of these missions will be conducted within the context of military objectives.

Admiral William H. Rowden, Commander Naval Sea Systems Command stated that, "With the NAVSEA (Naval Sea System Command) Integrated Robotics Program about to enter its fifth year of existence, it seems appropriate to look back and ahead to establish a baseline for the promulgation of policy guidelines to facilitate the continuing evolution of this important program." [Ref. 1] He goes on to say that the time has come to incorporate the value of robotics and automation into the Navy's expanding mission. Recent articles of Military Robotics [Refs. 2-6], have pointed out the increased availability of robotic vehicles. These include Remotely Piloted Aircraft, Unmanned Submarines, Teleoperated Combat Vehicles, Cruise Missiles and Teleoperated and Autonomous Weapons.

An extremely important part of the total AUV vehicle control logic is its need to plan and execute a safe passage in the undersea environment. Local path planning

is the function provided by an intelligent system, which determines safe, collision-free trajectory of travel between two points, a start point and a target point, for a specific time lapse. One possible total system block diagram that shows how the local path planner could be interfaced, is shown in Figure 1.1. Here, the Global Planning System would provide the Local Path Planner with a series of data sets. Included in the data sets would be destination position, destination time, start position, start time, obstacles and boundaries. In return, the path planner would provide an optimal path based upon the limitations of the vehicle dynamics, power plant efficiency, obstacle field, and required maneuver time.

Numerous techniques have been used to achieve collision free local paths for various vehicle types and manipulators. These include graphical search methods [Refs 7-11], potential field methods [Refs. 12-16] and optimal control theory [Refs. 17, 18]. This thesis is concerned with developing a method of autonomous planning using optimal control theory.

## B. PREVIOUS WORK

A basic investigation of local path planning was previously conducted using optimal control theory [Ref. 19]. In that study, major emphasis was placed on the solution of a SISO (Single Input Single Output) problem, a

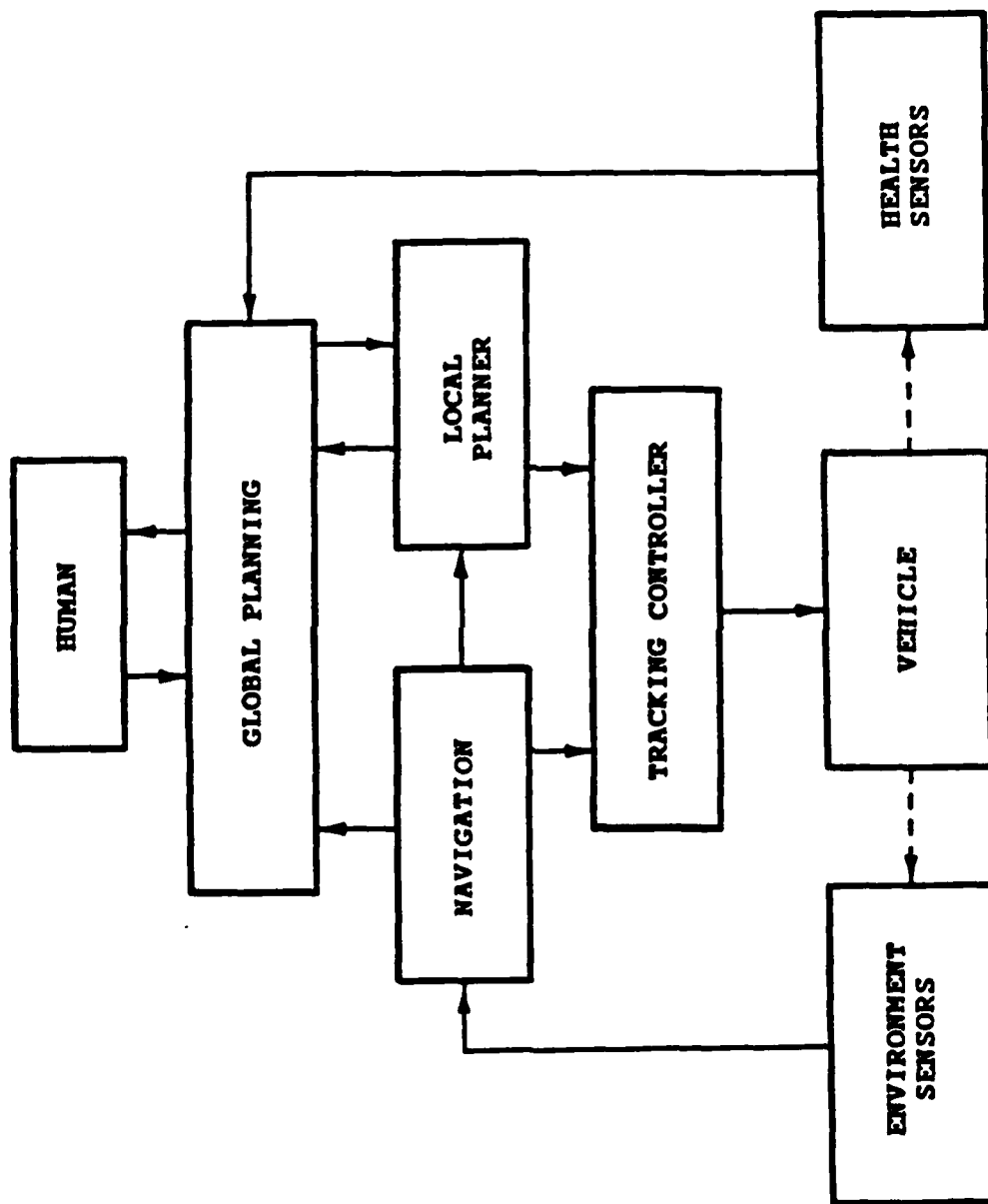


Figure 1.1 Global Planning System

MIMO (Multiple Input Multiple Output) problem and its generalization to a submersible. That work included objective function determination, integration method studies, linear versus nonlinear solution results, computational expense and an obstacle avoidance solution with one fixed obstacle. The objective function used for optimization was a quadratic performance index of the form:

$$J = \int_0^{FINTIM} (\underset{\sim}{X}^T \underset{\sim}{Q} \underset{\sim}{X} + \underset{\sim}{U}^T \underset{\sim}{R} \underset{\sim}{U}) dt$$

where,

$\underset{\sim}{U}$  = the control vector; and

$\underset{\sim}{X}$  = desired states-actual states (i.e. state error)

The nonlinear hydrodynamic equations of motion for the Autonomous Underwater Vehicle being studied were of the following form:

$$\underset{\sim}{M} \ddot{\underset{\sim}{X}} + \underset{\sim}{f}(\underset{\sim}{X}, \dot{\underset{\sim}{X}}) = \underset{\sim}{g}(\underset{\sim}{U})$$

The "best" solution was obtained by minimizing the objective function (J) in order to find the best  $\underset{\sim}{U}(t)$  and  $\underset{\sim}{X}(t)$  values.

The Automatic Design Synthesis (ADS) Fortran Program [Ref. 20] was utilized for problem optimization and the Dynamic Simulation Language (DSL) Program [Ref. 21] was utilized for objective function calculations and integrations of the vehicle dynamic equations. These

software programs were made to be interactive and now perform as one software package [Ref. 22]. The combined package is called ADSL and has been incorporated on the IBM 3033 Mainframe Computer System at the Naval Postgraduate School. The basic optimization approach was as follows:

1. Discretize the control vector into a time-wise uniform distribution of control signals (Figure 1.2)

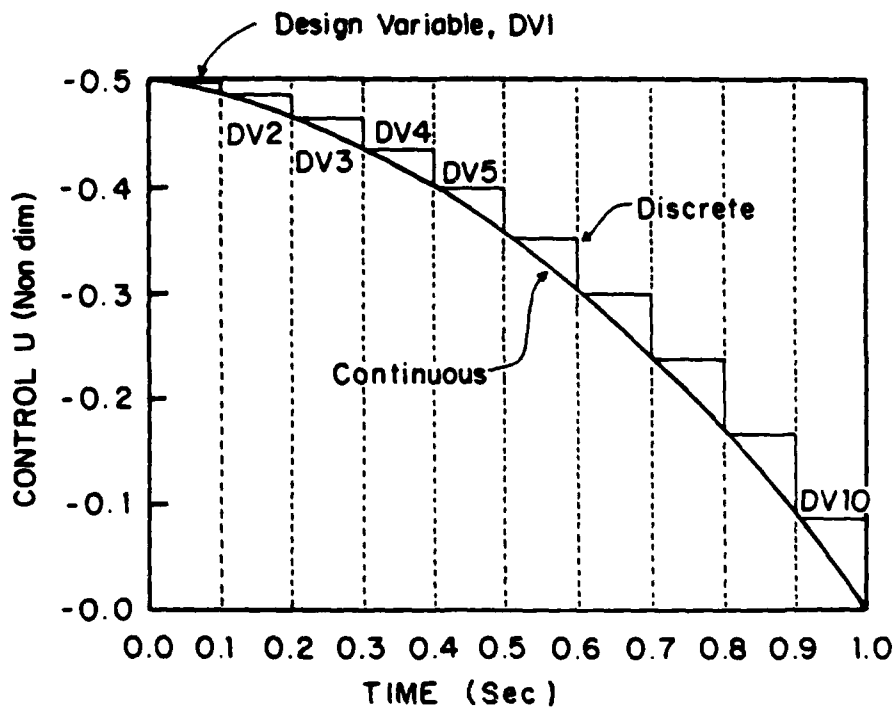


Figure 1.2 Discrete vs. Continuous Controls

2. Determine the best control sequence via an optimization routine based upon the objective function and problem constraints.

In the two-dimensional problem (dive plane only), the vehicle was ordered to achieve an ordered depth of 17.425



feet using minimum bow and stern plane deflections. Additionally, the vehicle was required to have a minimum pitch angle at its final end condition. The control vector  $\tilde{U}$  was the bow and stern plane angles, while the  $\tilde{X}$  vector was the  $x$  and  $z$  positions of the vehicle and velocities in the  $x$  and  $z$  directions.

### C. AIM OF THE PRESENT STUDY

This thesis is concerned with furthering the understanding of local path planning using optimal control theory. The purpose of this work is to:

1. Further develop the planning level control logic to consider three-dimensional maneuvers, and
2. Evaluate the performance of this logic.

## II. METHOD OF APPROACH

The basic approach was as follows:

1. Improve the treatment of obstacles, both fixed and moving in the two-dimensional problem.
2. Determine the best set of optimization program options based on computational cost, robustness, flexibility and solution accuracy in the two-dimensional problem.
3. Select guidance for maneuvering time (FINTIM) and determine how it effects problem solution in the two-dimensional problem.
4. Evaluate two dimensional versus three dimensional computational costs and accuracy.

The basic assumption in this study was that the work previously done [Ref. 19] remained valid. Specifically, that the integration method selected, the step size, objective function, number of design variables and optimization program options remained relevant.

### III. OBSTACLE AVOIDANCE

The approach previously presented [Ref. 19] was to compute the distance to the obstacle at ten equally divided time intervals from start time to the time of closest obstacle approach. These updated distances were then incorporated into the optimization algorithm for constraint value determination. ADSL placed constraint equations into the algorithm in the form:

$$G_j(k) = 0 \quad j = 1, m$$

which in the actual program is:

$$G_k(k) = (\text{avoidance zone}) - (\text{updated vehicle distance})$$

The time interval for distance calculations was determined based on the FINTIM and clock time as follows:

```
If (time.ge.0.0.and.le.xobs/u) then
time1 = xobs/u
qn = time/(time1/10. - delt/10000.)
d = int(qn + 1)
dist(d) = sqrt((xpos-xobs) x (zpos-zobs))
g(d) = 1. - dist(d)
```

where:

```
time = DSL clock time
xobs = x position of the obstacle
delt = integration time step interval
xpos = x position of the vehicle
```

$u$  = vehicle velocity in the  $x$  direction

$\text{dist}(d)$  = computed distance from the vehicle to the obstacle

$z_{\text{pos}}$  =  $z$  position of the vehicle

$z_{\text{obs}}$  =  $z$  position of the obstacle

$g(d)$  = constraint value placed in optimization routine

The problem with this approach is that the further an obstacle is from the start position, the longer the time intervals become for obstacle distance calculations. This is satisfactory for a single fixed obstacle but for multiple obstacles, this method results in inadequate distance computations. This is because the closer obstacles do not have sufficient constraint inputs compared to the distant obstacles. As a result, the distant obstacles tend to dominate the solution. Using multiple "if" statements in this logic is also computationally expensive and failed when used with three or more obstacles.

Another inherent problem was that distances were not computed after the time of closest approach. This sometimes resulted in maneuvers with distances to the obstacle that violated the avoidance zone regions after the first time of closest approach.

A better method is to continuously compute distances to the obstacle, independently of FINTIM, but dependent on FINTIM step intervals. This approach worked very well and

was adopted for all further analysis. An additional advantage to this approach is that only one constraint assignment was needed for each obstacle vice ten.

The computational time for obstacle avoidance varies as the number of obstacles increases. The motivation for this study was to determine if this approach was computationally too expensive to remain as a viable approach. Figure 3.1 shows the computational cost for one to seventeen fixed obstacles. The computational time is based on the virtual machine time for the IBM 3033 system at the Naval Postgraduate School. In all cases, the final depth was the desired depth of 17.425 feet.

Various optimization techniques have various computational costs; however, the times in Figure 3.1 are based upon the final optimization option selected for this thesis. The selection criteria with results will be presented in the following chapter.

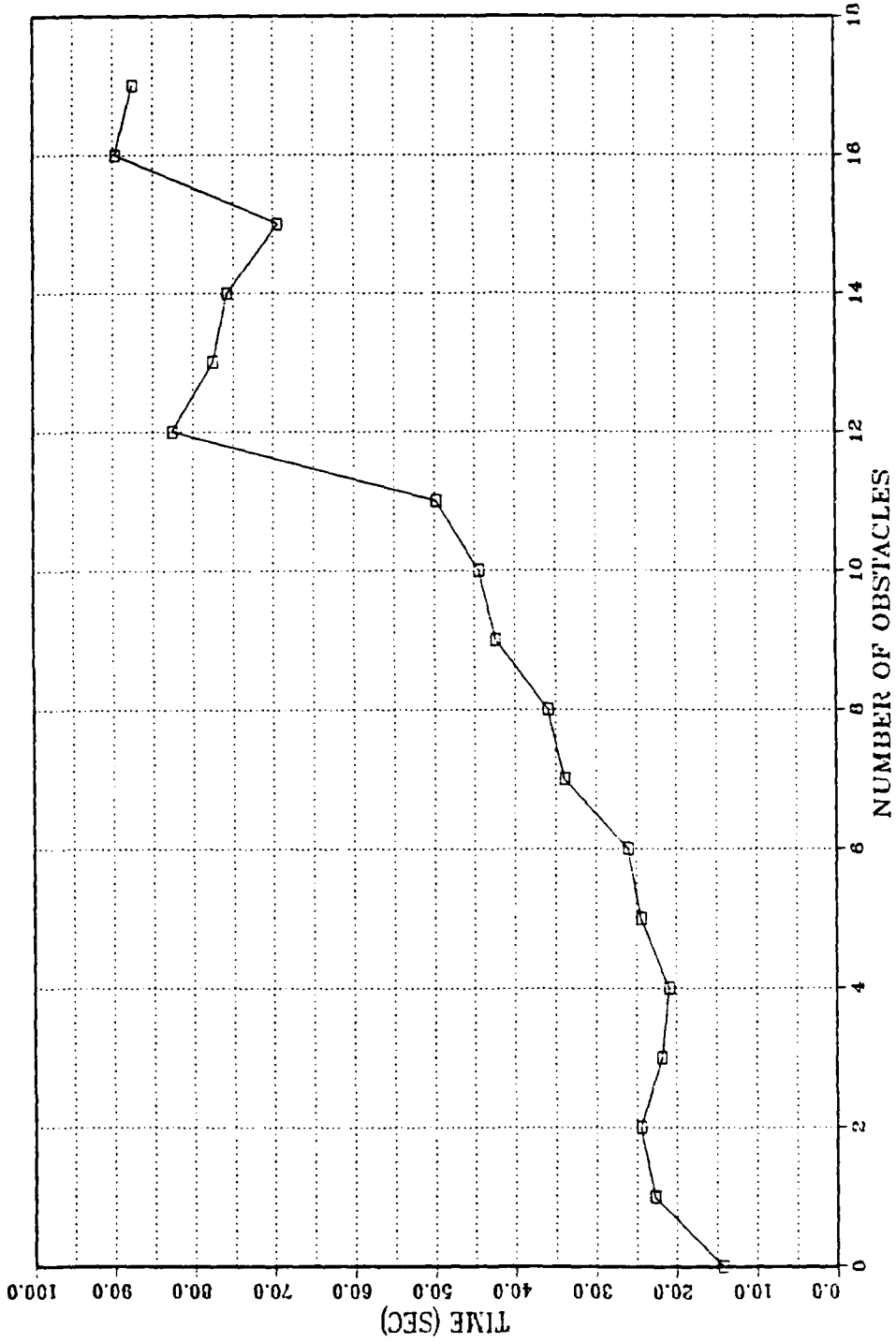


Figure 3.1 Fixed Obstacle Computational Results

#### IV. OPTIMIZATION OPTIONS

The ADS (Advanced Design Synthesis) Program allows for the selection of numerous optimization techniques for problem solution. There are three levels by which to select a particular technique. The three levels are the strategy level, optimizer level and the one-dimensional search level. Table 1 lists the various levels and the various methods contained in each. Figure 4.1 identifies the large number of possible algorithm combinations allowed. Vanderplaats provides a detailed discussion of the various methods and algorithms in Reference 23.

##### A. CLUSTERED FIXED OBSTACLE TEST

In order to be effective as a path planning algorithm, it is necessary for the program option to be robust and flexible enough to solve problems involving numerous fixed obstacles as well as moving obstacles. Therefore, an initial test was conducted where the number of obstacles in the vehicle's path were varied from one to seventeen. Program option 057 was selected first based upon the recommendation of the previous study, which involved one fixed obstacle in the vehicle's path. Option 057 appeared acceptable until a four obstacle field was encountered. At this point, the option failed to reach an adequate

---

IOPT	OPTIMIZER							
1	Fletcher-Reeves							
2	Davidon-Fletcher-Powell (DFP)							
3	Broydon-Fletcher-Goldfarb-Shanno (BFGS)							
4	Method of Feasible Directions							
5	Modified Method of Feasible Directions							

<u>STRATEGY</u>	<u>ISTRAT</u>	<u>IOPT</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>
None	0		X	X	X	X	X
SUMT, Exterior	1		X	X	X	0	0
SUMT, Linear Extended Interior	2		X	X	X	0	0
SUMT, Quadratic Extended Interior	3		X	X	X	0	0
SUMT, Cubic Extended Interior	4		X	X	X	0	0
Augmented Lagrange Multiplier Meth.	5		X	X	X	0	0
Sequential Linear Programming	6		0	0	0	X	X
Method of Centers	7		0	0	0	X	X
Sequential Quadratic Programming	8		0	0	0	X	X
Sequential Convex Programming	9		0	0	0	X	X

<u>ONE-DIMENSIONAL SEARCH</u>	<u>IONED</u>						
Golden Section Method	1		X	X	X	0	0
Golden Section + Polynomial	2		X	X	X	0	0
Polynomial Interpolation (bounded)	3		X	X	X	0	0
Polynomial Extrapolation	4		X	X	X	0	0
Golden Section Method	5		0	0	0	X	X
Golden Section + Polynomial	6		0	0	0	X	X
Polynomial Interpolation (bounded)	7		0	0	0	X	X
Polynomial Extrapolation	8		0	0	0	X	X

NOTE: An X denotes an allowed combination of algorithms.

---

Figure 4.1 Allowable ADS Algorithm Combinations



TABLE 1. ADS LEVEL OPTIONS

---

STRATEGY (ISRRAT)

- 0 - None
- 1 - SUMT, Exterior Penalty Function
- 2 - SUMT, Linear Extended Interior
- 3 - SUMT, Quadratic Extended Interior
- 4 - Cubic Extended Interior
- 5 - Augmented Lagrange Multiplier Method
- 6 - Sequential Linear Programming
- 7 - Method of Centers
- 8 - Sequential Quadratic Programming
- 9 - Sequential Convex Programming

OPTIMIZER (IOPT)

- 1 - Fletcher-Reeves
- 2 - Davidon-Fletcher-Powell (DFP)
- 3 - Broydon-Fletcher-Golfarb-Shanno (BFGS)
- 4 - Method of Feasible Directions
- 5 - Modified Method of Feasible Directions

ONE-DIMENSIONAL SEARCH (IONED)

- 1 - Golden Section Method
- 2 - Golden Section and Polynomial
- 3 - Polynomial Interpolation, bounded
- 4 - Polynomial Extrapolation

- 5 - Golden Section Method
- 6 - Golden Section and Polynomial
- 7 - Polynomial Interpolation, bounded
- 8 - Polynomial Extrapolation

solution. Option 133 was then selected based upon Olson's work [Ref. 22]. Option 133 is more robust than option 057 and it appeared to be very well suited for this problem until the ten obstacle field was encountered. This method achieved the correct ordered depth; however, it violated many of the obstacle avoidance zones. It was apparent, at this point, that all program options would have to be tested in order to determine the most appropriate algorithm.

A test was conducted to determine if any of the one hundred and twelve program options could solve a seventeen obstacle problem. The complete test problem required the program option to solve a seventeen obstacle field problem with FINTIM set at seven seconds and an ordered depth of 17.425 feet. Each obstacle had a one foot radius avoidance zone. The results of that study are presented in Table 2.

TABLE 2. 17 OBSTACLE TEST RESULT

PROGRAM OPTION	COMPUTATIONAL COST (sec)	DEPTH (feet)
311	87.92	17.425
312	92.56	17.425
313	58.78	17.425
314	73.51	17.425
321	146.23	17.425
322	142.32	17.425
323	115.11	17.425
324	165.12	17.425
331	167.25	17.425
332	136.84	17.425
333	151.90	17.425
334	120.78	17.425
411	76.19	17.425
412	96.00	17.421
413	42.99	17.425
414	64.92	17.424
421	116.61	17.425
422	140.87	17.425
423	82.18	17.425
424	94.39	17.425
431	115.67	17.425
432	143.50	17.425
433	81.64	17.425
434	97.29	17.425
512	70.17	17.425
534	48.87	17.449

Of the one hundred and twelve program options, only twenty six achieved a correct solution. The computational time varied significantly among the various successful program options. In some cases, the exact depth was not achieved; however, all results were considered excellent.

After determining which algorithms could solve the seventeen obstacle problem, it was then necessary to verify

that cases involving various obstacle combinations from one to sixteen were solvable by these methods. It may seem intuitively obvious that if an algorithm can achieve a solution involving seventeen obstacles that it can solve all other cases from one obstacle to sixteen obstacles. Contrary to intuition, this is not the case. For example, program option 313, which had a relatively small computational cost, successfully solved the seventeen obstacle problem but failed to achieve the correct depth when an eleven obstacle field was encountered.

In conducting the varying fixed obstacle investigation, obstacles were purposefully placed in various positions in the field in order to ensure that obstacle position had no negative effect upon the problem solution. This was significant because program option 057 (method chosen in the previous study), failed when it encountered an obstacle field with three fixed obstacles. Two obstacles were placed in the vehicle's path and one was placed far from the vehicle's path. The obstacle far away from the vehicle's path was determined to be the cause of failure because the algorithm successfully solved a problem with three obstacles when all three were placed near the vehicle's path. Using the cases of one to seventeen obstacles, the cases were further reduced from 26 to 22. Program options 313, 413, 534, and 314 were eliminated.

## B. IMPOSSIBLE FIELD TEST

After an investigation of the varying obstacle test, the reduced list of programs were subjected to an impossible problem. Four obstacles were placed in the vehicle's path with nine, five, three, and six feet radii. They were placed in such a way that the algorithm could not achieve the correct solution in the allotted time. It is important to point out that a correct solution would have been obtainable if the simulation time was increased. The motivation for this study was to determine the failure modes of various algorithms. It was evident from this study that some algorithms, namely those which employed a strategy of Sequential Unconstrained Minimization using the Cubic Extended Interior Penalty Function Method, were more sensitive to achieving the desired depth constraints, when they were imposed as equality constraints. The algorithms which employed the strategy of Sequential Unconstrained Minimization using the Quadratic Extended Interior Penalty Function Method, were more sensitive in avoiding obstacle avoidance zones when they were imposed as inequality constraints. Figure 4.2 illustrates the performance of three different algorithms in solving this problem. Of the algorithms with relatively small computational costs, program option 311 did the best job of avoiding the avoidance zones.

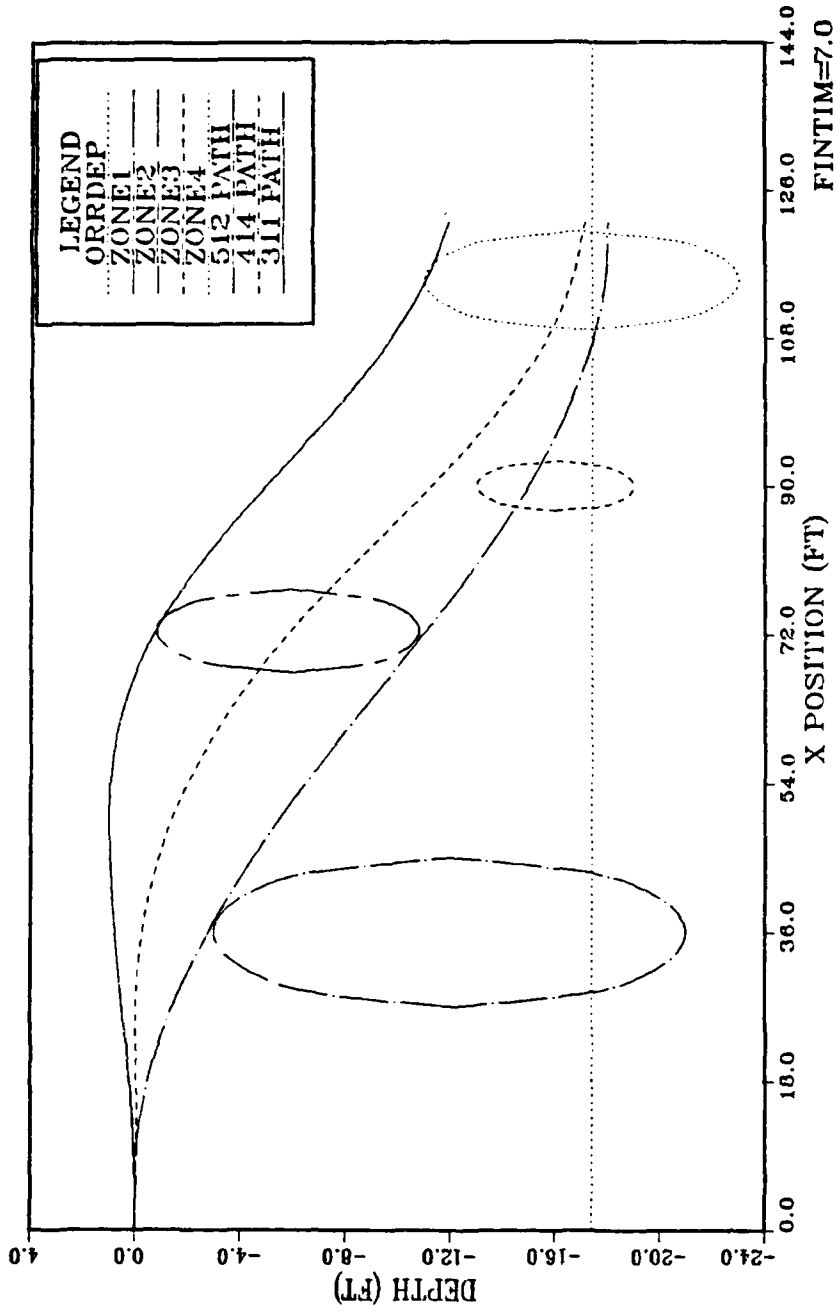


Figure 4.2 Impossible Test Algorithm Comparison

With the impossible field test, computational cost uniformly increased compared to the four obstacle problem with smaller avoidance zones. Program option 311 had a computational cost of 74.30 Virtual Machine second in the impossible problem; however, with four obstacles it took 21 seconds (Figure 3.1). Figure 4.3 is the solution result obtained if FINTIM is increased to 15.0. Although FINTIM was more than doubled, the computational cost did not significantly increase. With FINTIM set to 15.0, the virtual machine time was 76.51 seconds.

#### C. SELECTION RESULT

Program option 311 with a strategy of Sequential Unconstrained Minimization using the Quadratic Extended Interior Penalty Function Method; an Optimizer using the Fletcher-Reeves algorithm and a one-dimensional search method using the Golden Section Method was chosen as the best algorithm. It was selected because it had the least computational cost of any algorithm which could solve the seventeen obstacle test problem and was very sensitive to the obstacle avoidance zones. In other words, it proved to be very good at finding a safe, collision-free path between the start condition and the end condition.



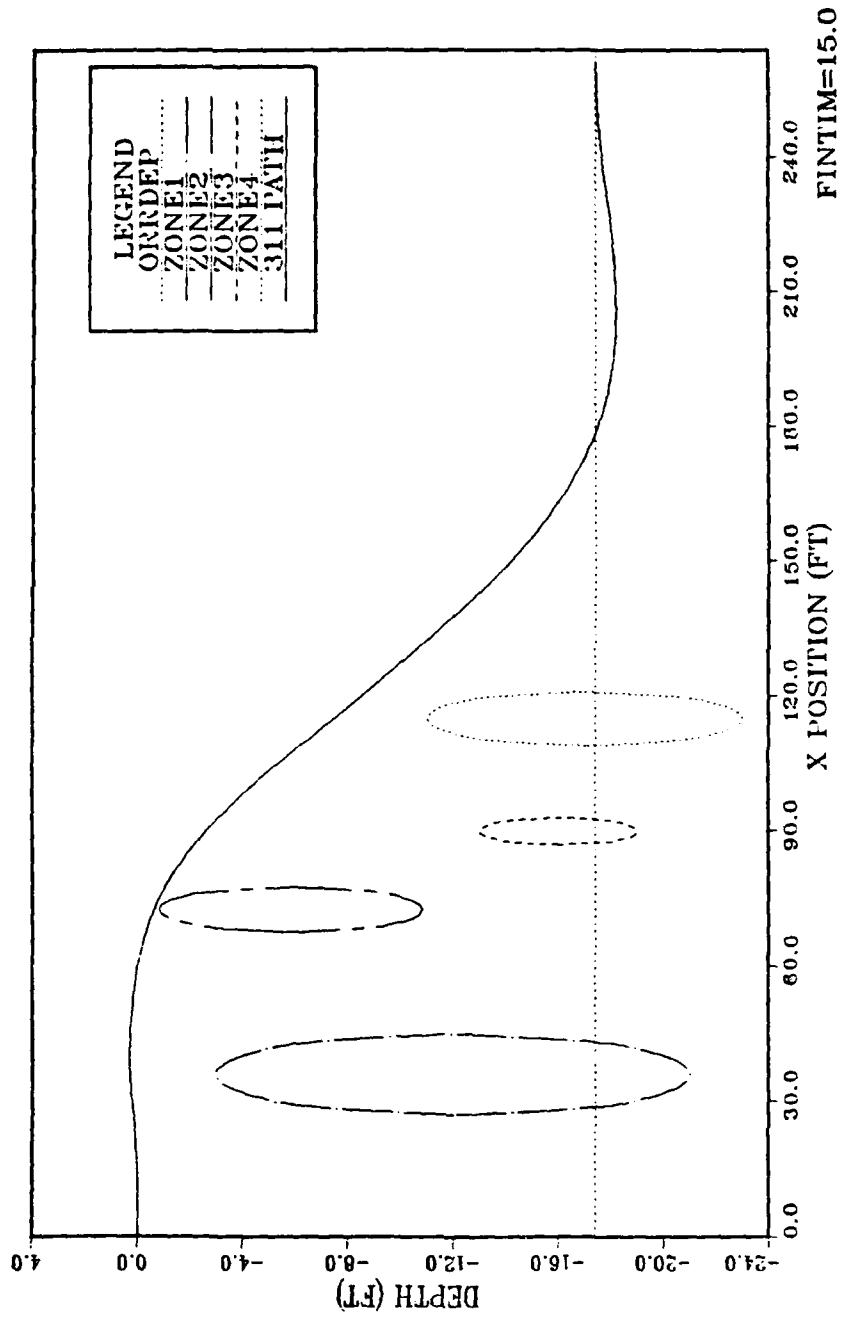


Figure 4.3 Solution of Impossible Test with Increased FINTIM

## V. EVALUATION OF MANEUVERING TIME (FINTIM)

Qualitatively, there are two possible FINTIM effects. Those which are associated with a small FINTIM and those which are associated with an excessively large FINTIM. The net effect of too small a FINTIM is an over constraining of the problem, which leads to a violation of problem constraints and excessive computational time.

Two things happen when the FINTIM is too large. The solution adheres more to problem constraints and the computational cost decreases. The mission objectives of the vehicle (i.e., loitering at start position), are significant considerations, which FINTIM selection must take into account. Therefore, FINTIM is a critical parameter which effects problem solution and also, when chosen correctly, significantly reduces computer computational costs. The selection of FINTIM poses an important problem which requires solving in view of high-level vehicle objectives.

One guideline for selecting FINTIM is to select it based on the time required to achieve a solution while transversing an obstacle-free field, then arbitrarily increase FINTIM to allow for obstacle avoidance. The following results using program options 533 points out the importance of choosing a correct FINTIM. As can be seen in

Figure 5.1, FINTIM can adversely affect the problem solution if the time allotted is not large enough to achieve the desired result. When FINTIM is chosen to be 6.0 non-dimensional time units (NTU), the avoidance zone constraint for zone 3 is violated and the desired depth of 17.425 feet is exceeded. When FINTIM is increased to 7.0 NTU, avoidance zone constraints are violated for zone 1 and zone 3 and the desired depth is not achieved. However, the severity of the violations are not as blatant. When FINTIM is increased to 8.0 NTU, the desired problem solution is obtained. Table 3 presents the computational costs associated with each FINTIM selection. Note that the optimization problem is easier with more maneuvering time, therefore the computational cost is less.

TABLE 3. FINTIM COMPUTATIONAL COST

FINTIM	TIME (sec)
6.0	95.06
7.0	76.75
8.0	32.43

# FINTIM EFFECTS

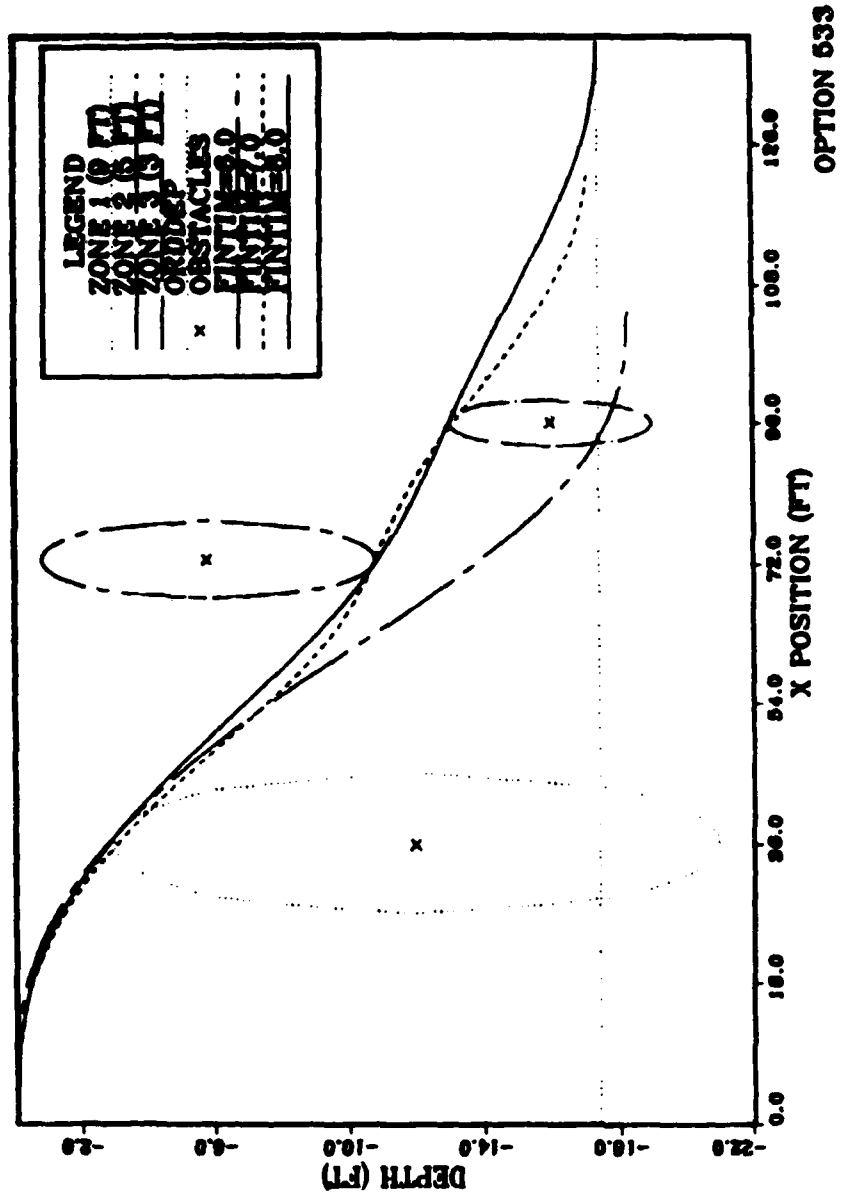


Figure 5.1 FINTIM Effects

## VI. PROGRAMMING FOR THREE DIMENSIONS

In programming for three dimensions, we not only optimize the problem to obtain bow plane and stern plane commands, but we also optimize to obtain the rudder commands. In order to achieve the desired result, it was necessary to increase the number of design variables for the rudder in the linear model. The problems discussed previously, have all been solved using ten discretizations (design variables) for the stern plane and bow plane inputs. In the three-dimensional work, the number of design variables were arbitrarily increased to twenty (less discretizations would not achieve a satisfactory result).

### A. SIDE CONSTRAINTS

Additional constraints were added to the problem in order to ensure reasonable vehicle control surface reactions. The maximum rudder angles were set at plus or minus thirty degrees. In order to ensure this, the side constraint approach was invoked. These values were assigned to the Design Variable Lower Bound (VLB) and the Design Variable Upper Bound (VUB) ADS parameters.

## B. EQUALITY CONSTRAINTS

Six additional equality constraints were needed to achieve the desired y position and the desired vehicle condition (yaw and roll) at the desired end condition.

## C. CONSTRAINT SCALING

Sanders [Ref. 19] points out that constraint weighting is important in achieving the desired results. This is even more crucial in a three-dimensional problem solution because of the increased number of constraints on yaw, roll, rudder control and y positioning. It appears that problem sensitivity to constraint weighting is also increased. In order to achieve the desired solution result, it was necessary to adjust constraint scaling factors until all constraint conditions were satisfactorily obtained. Table 4 shows the constraint scaling factors used in the full three-dimensional linear model.

TABLE 4. CONSTRAINT SCALING FACTORS

CONSTRAINT	SCALING FACTOR
depth	0.5
y position	2.5
pitch	1.0
yaw	1.0
roll	1.0
minimum depth	1.0

#### D. LINEAR/NONLINEAR DYNAMICS

Figure 6.1 illustrates the nonlinear model behavior when using control inputs for bow plane, stern plane, and rudder from the optimized linear model. It is evident that these commands are invalid for the full scale nonlinear model since the final objective state is not closely met. Therefore, the essential dynamics of the linear model are not valid in three dimensions as might be suggested from the results of the previous study. However, even though the control surface inputs are invalid, the vehicle state trajectory is valid because the path chosen achieved the desired result. For a desired position of  $y=40.0$  feet and depth  $=-20.0$  feet, the obtained result was  $y=40.269$  feet with a depth of  $-20.699$  feet. These values can be fine tuned by varying the scaling factors. Figures 6.2 and 6.3 illustrate the control inputs to the linear model and Figure 6.4 illustrates the linear model response to those inputs.

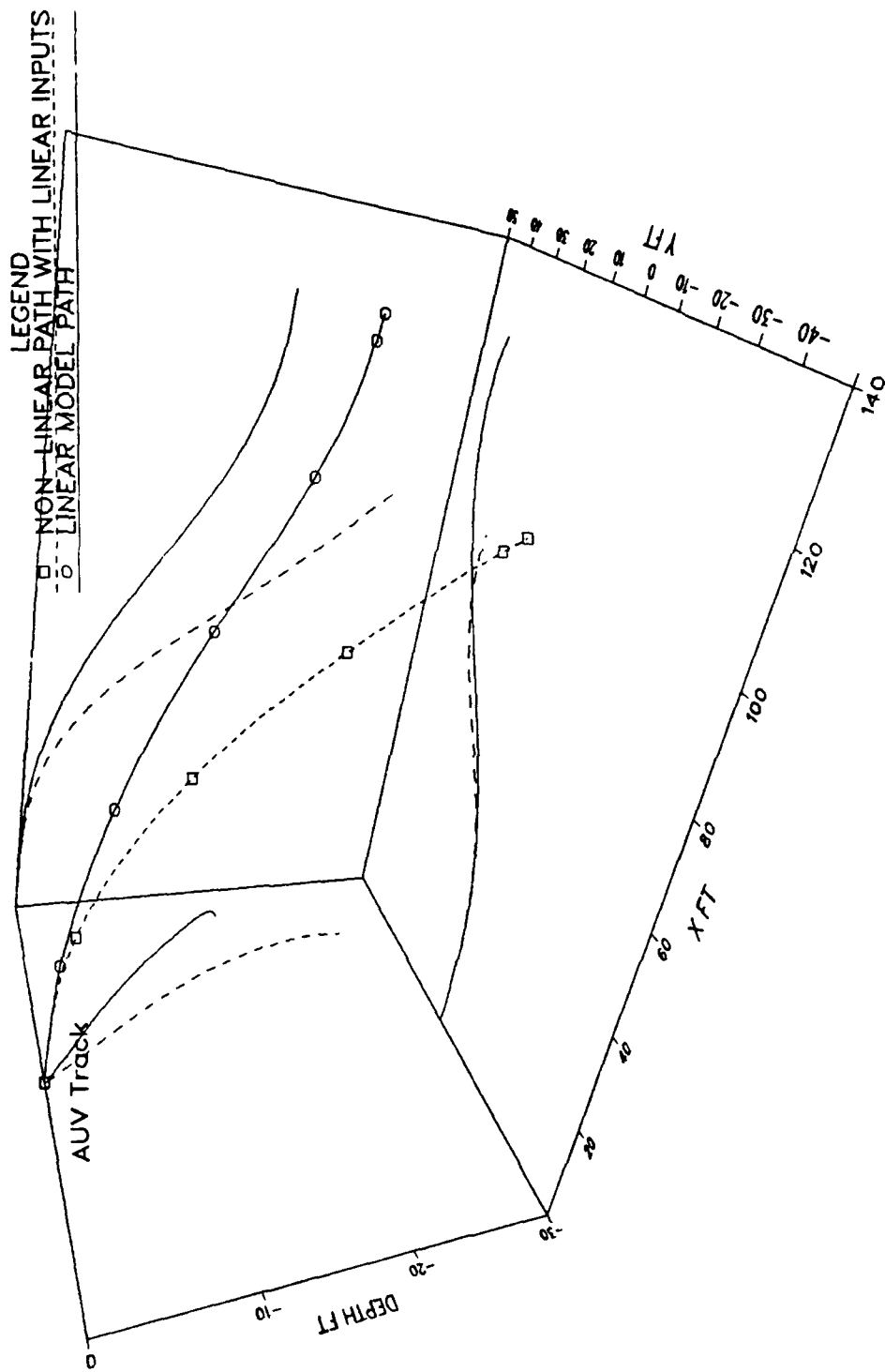


Figure 6.1 Nonlinear Model Maneuver with Linear Control Inputs



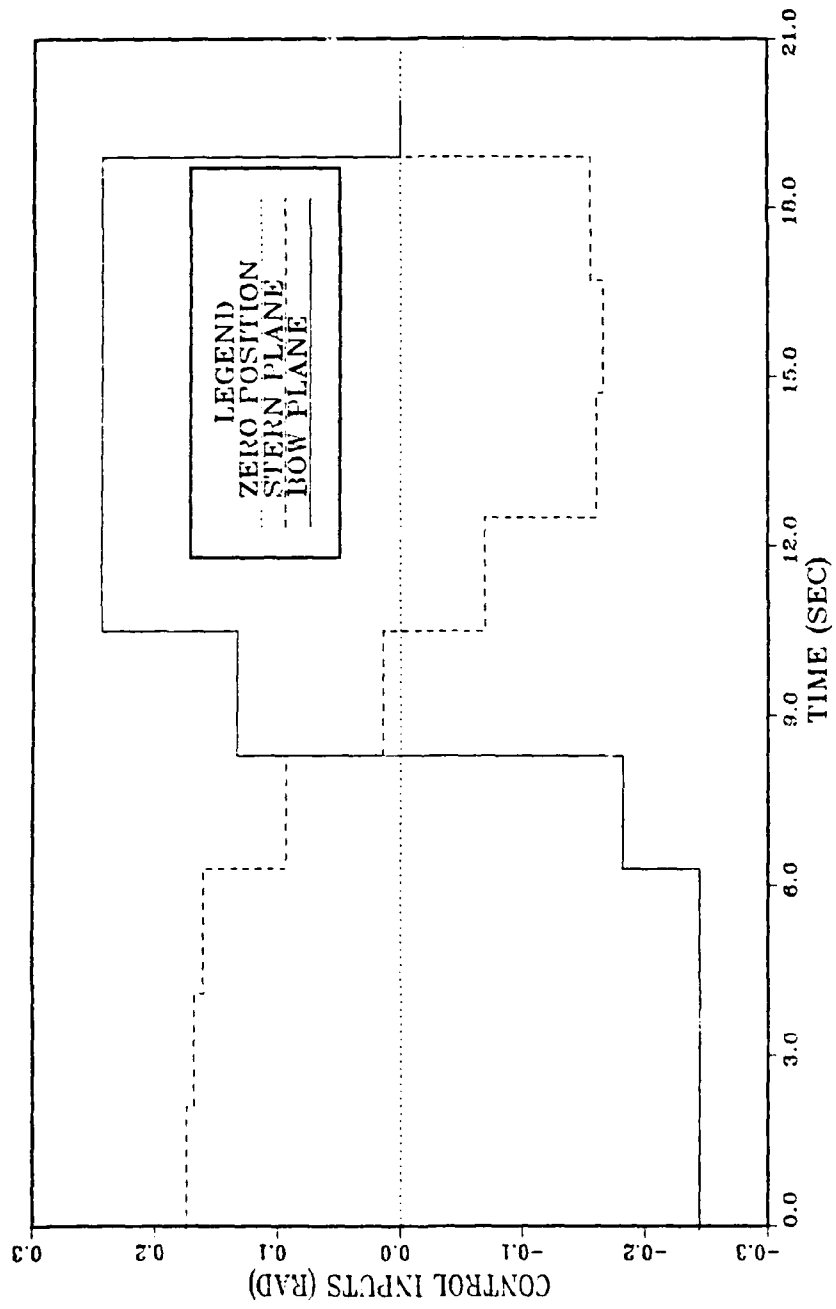


Figure 6.2 Bow and Stern Plane Control Inputs

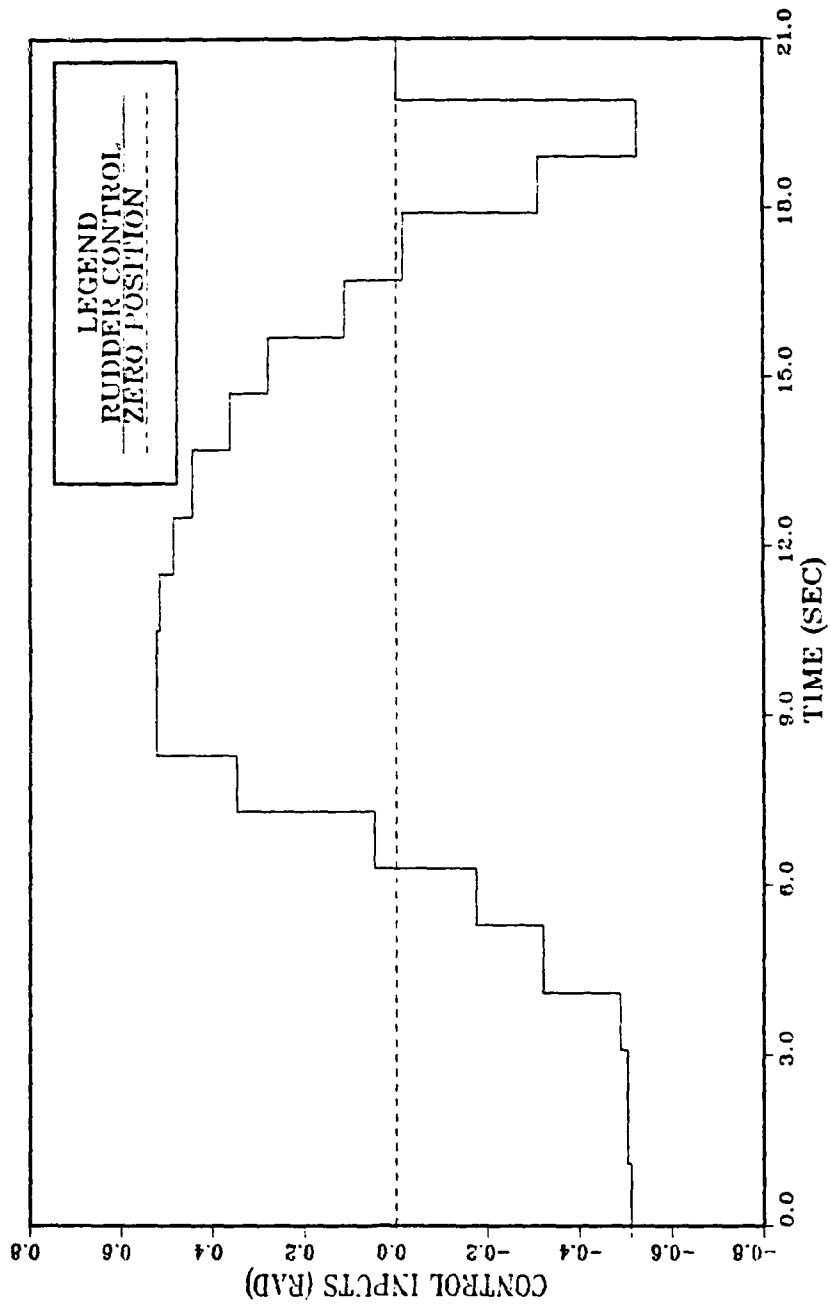


Figure 6.3 Rudder Control Input

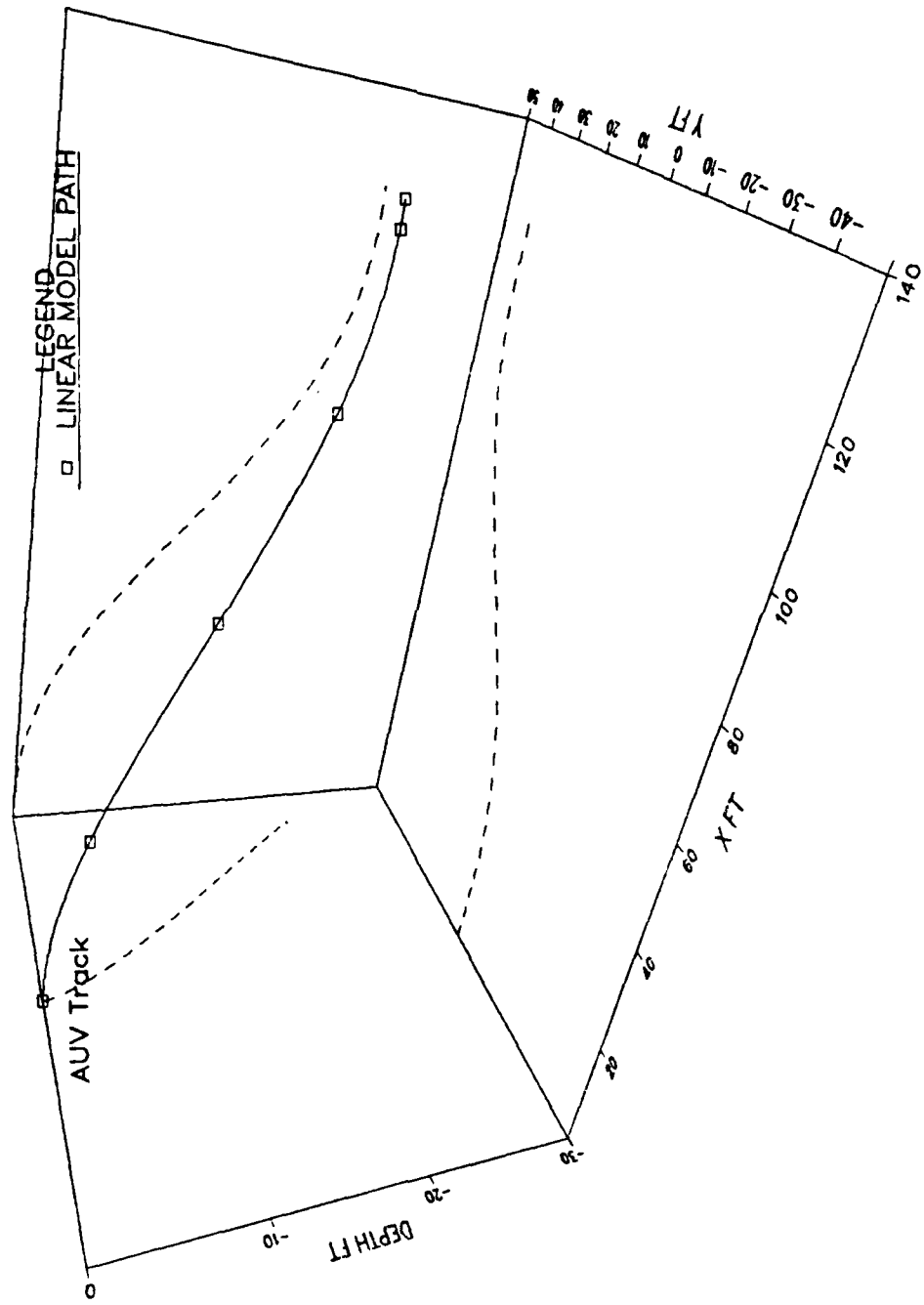


Figure 6.4 Linear Model Maneuver

E. THREE-DIMENSIONAL COMPUTATIONAL COSTS

Table 5 compares the virtual machine time of the full scale linear model with no obstacles as compared to the full scale nonlinear model with no obstacles.

TABLE 5. LINEAR VS. NONLINEAR COMPUTATIONAL COSTS

---

	LINEAR (sec)	NONLINEAR (sec)
DIVE PLANE ONLY	14.23	108.81
FULL 3D	130.34	370.61

---

## VII. VALIDATION RUNS

As previously mentioned, in order to validate the selected optimization configuration, fixed obstacles were placed at various positions in the AUVs field of view with 1.0 foot radius avoidance zones around the obstacles. Figure 7.1 to 7.8 illustrate the paths chosen by the 311 algorithm to avoid the obstacles and their avoidance zones for various obstacle positions.

The moving obstacles were simulated using rectilinear average velocity equations of the form:

$$s = Vt$$

where:

s = position of obstacle (X and/or Y)

V = constant velocity

t = time of travel

Figures 7.9 to 7.11 present the distance between the AUV and the moving obstacle(s) as a function of time. As seen, the algorithm chooses a path in both cases which avoids impact. Table 6 presents the computational cost comparison of various obstacle case(s). Some three-dimensional linear and non-linear state trajectory results were presented earlier.

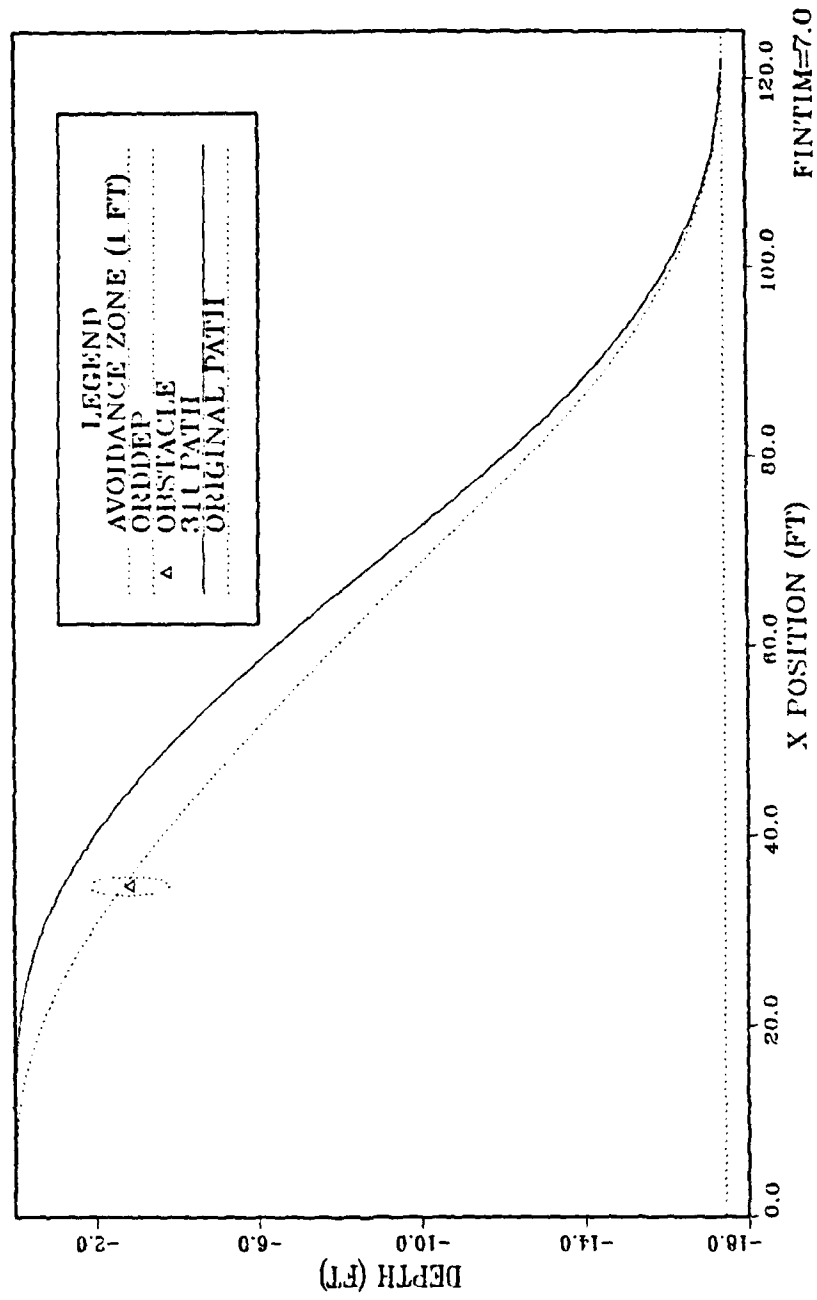


Figure 7.1 One Obstacle Solution

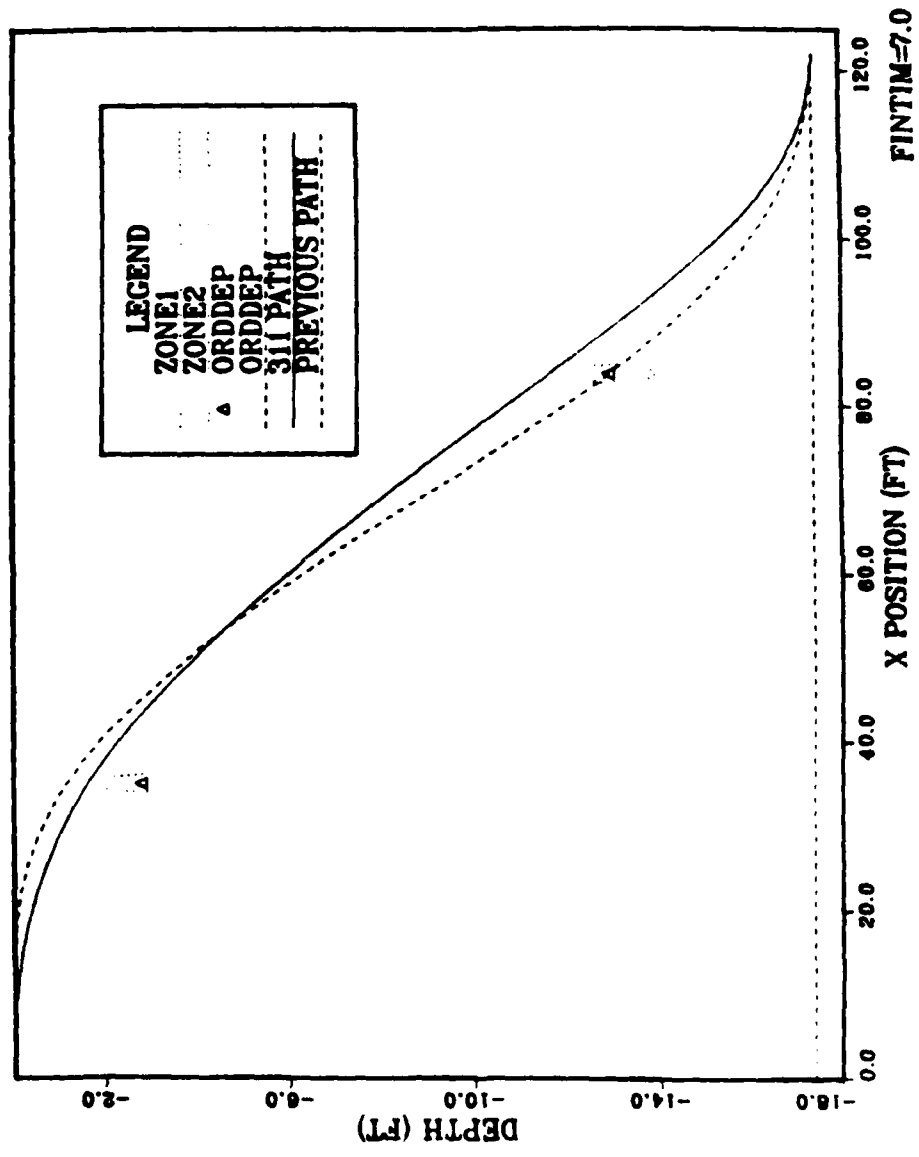


Figure 7.2 Two Obstacles Solution

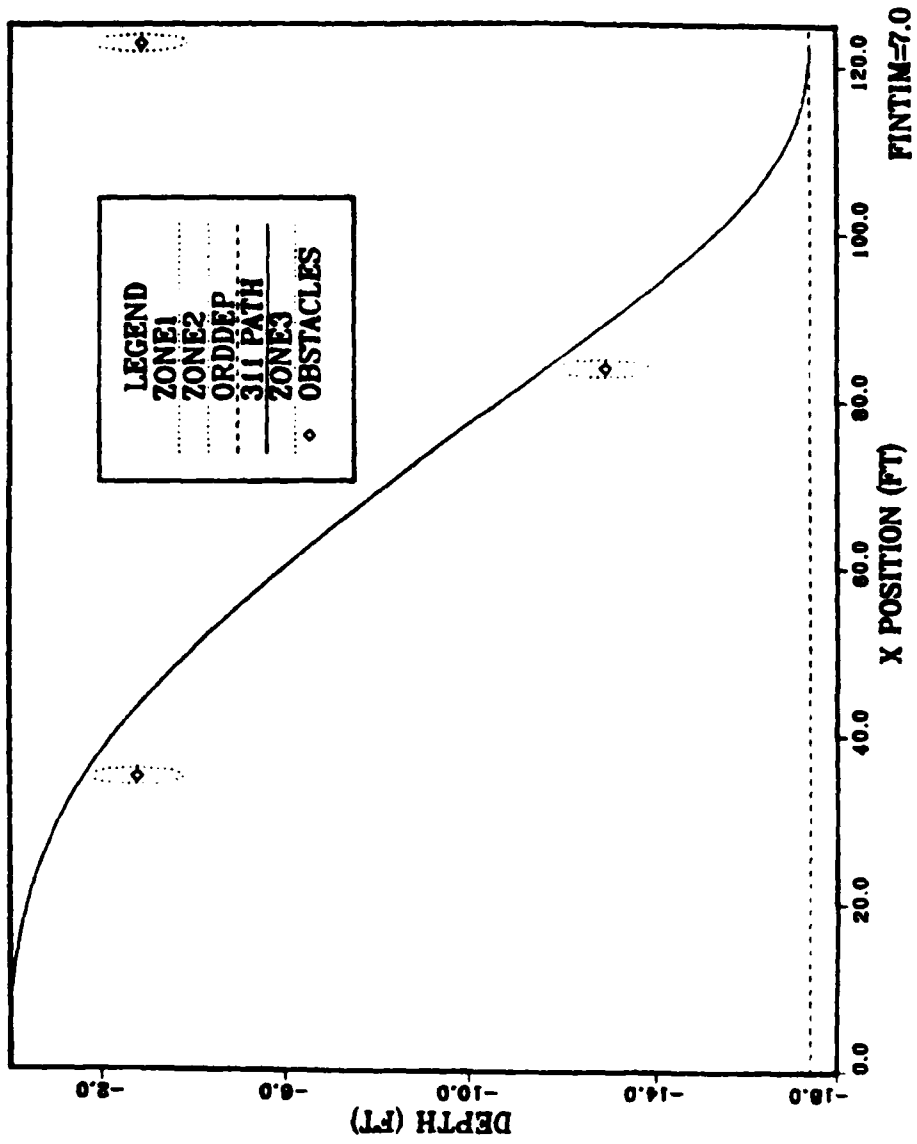


Figure 7.3 Three Obstacles Solution



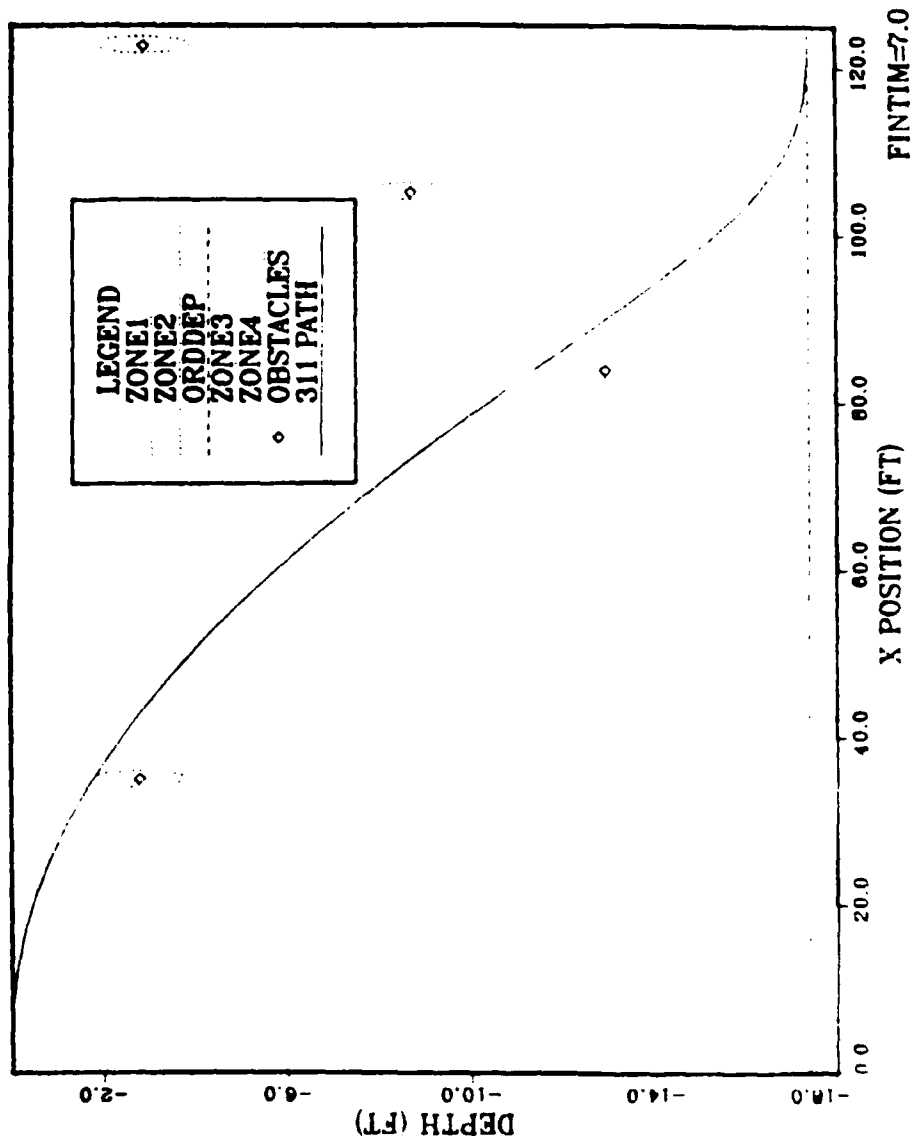


Figure 7.4 Four Obstacles Solution

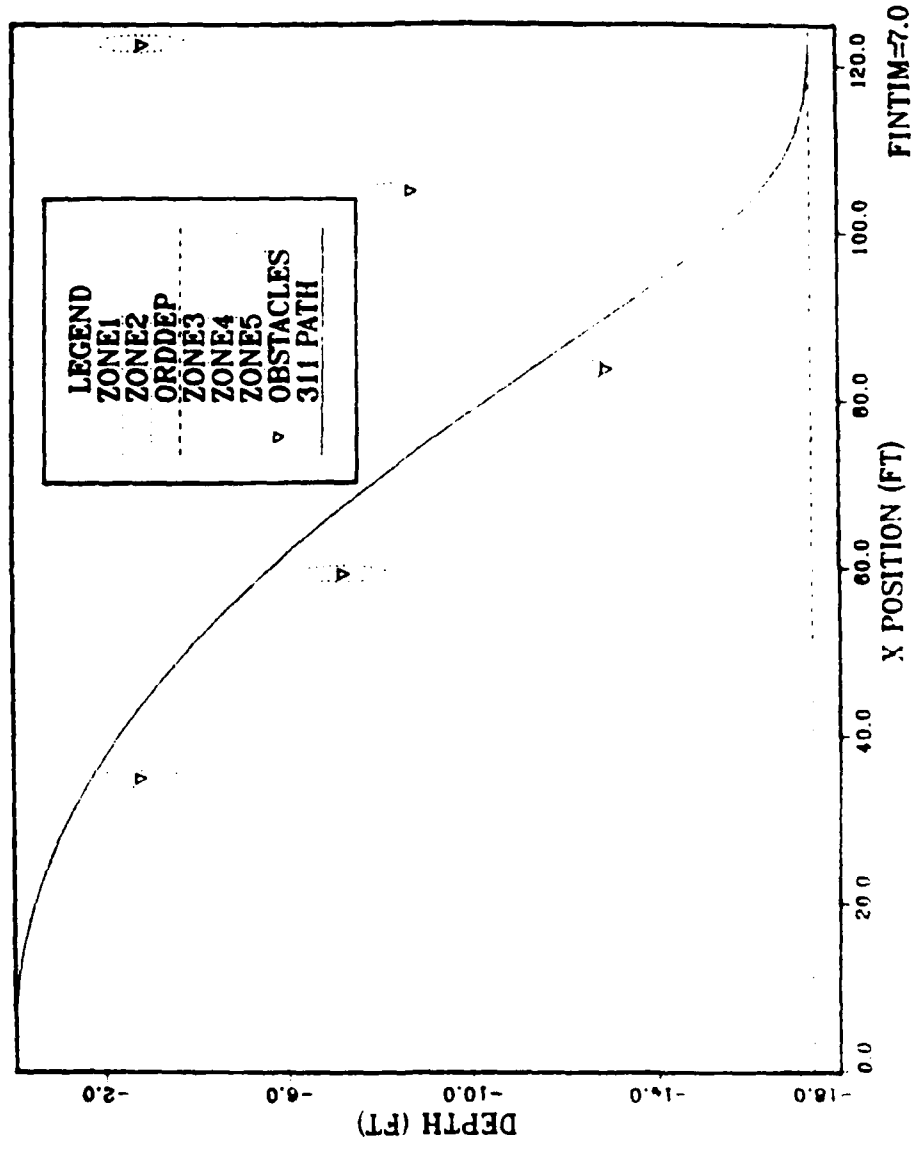


Figure 7.5 Five Obstacles Solution

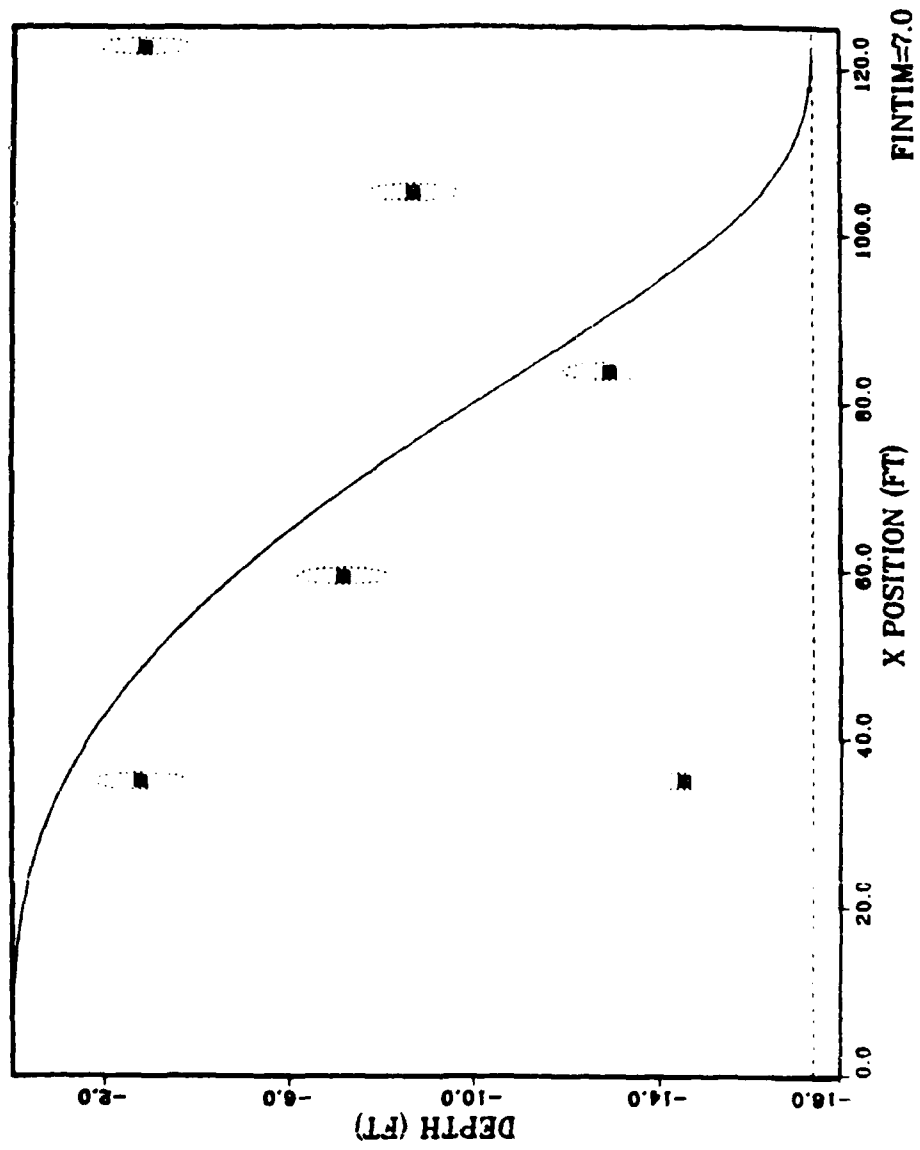


Figure 7.6 Six Obstacles Solution

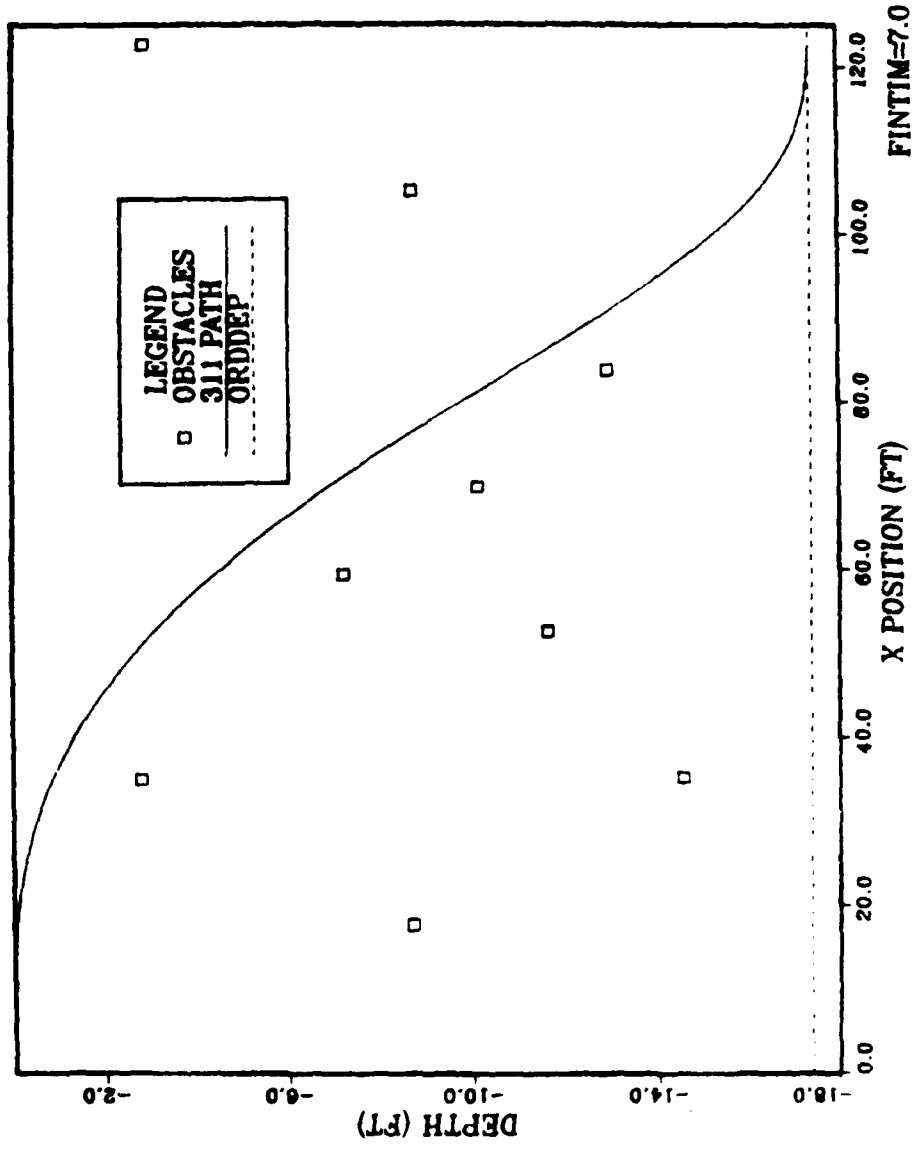


Figure 7.7 Nine Obstacles Solution

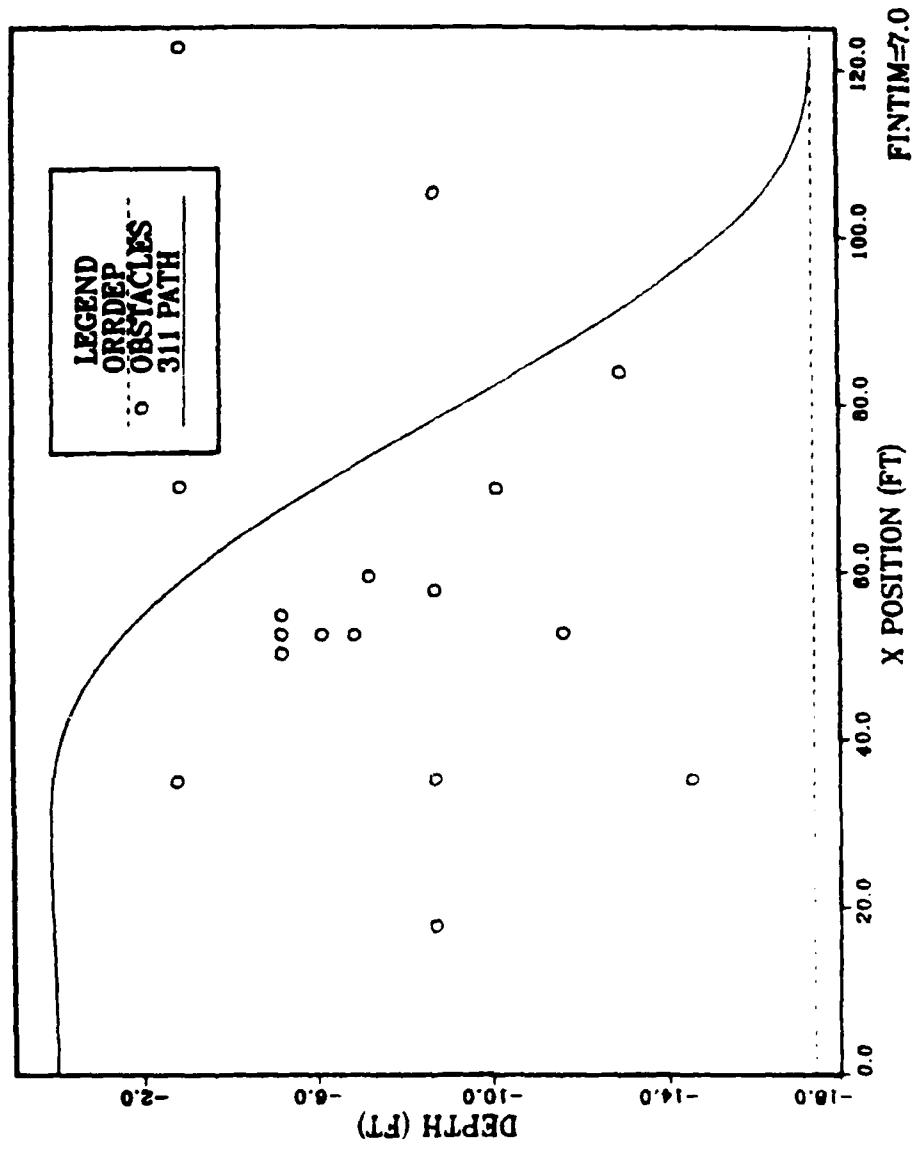


Figure 7.8 Seventeen Obstacles Solution

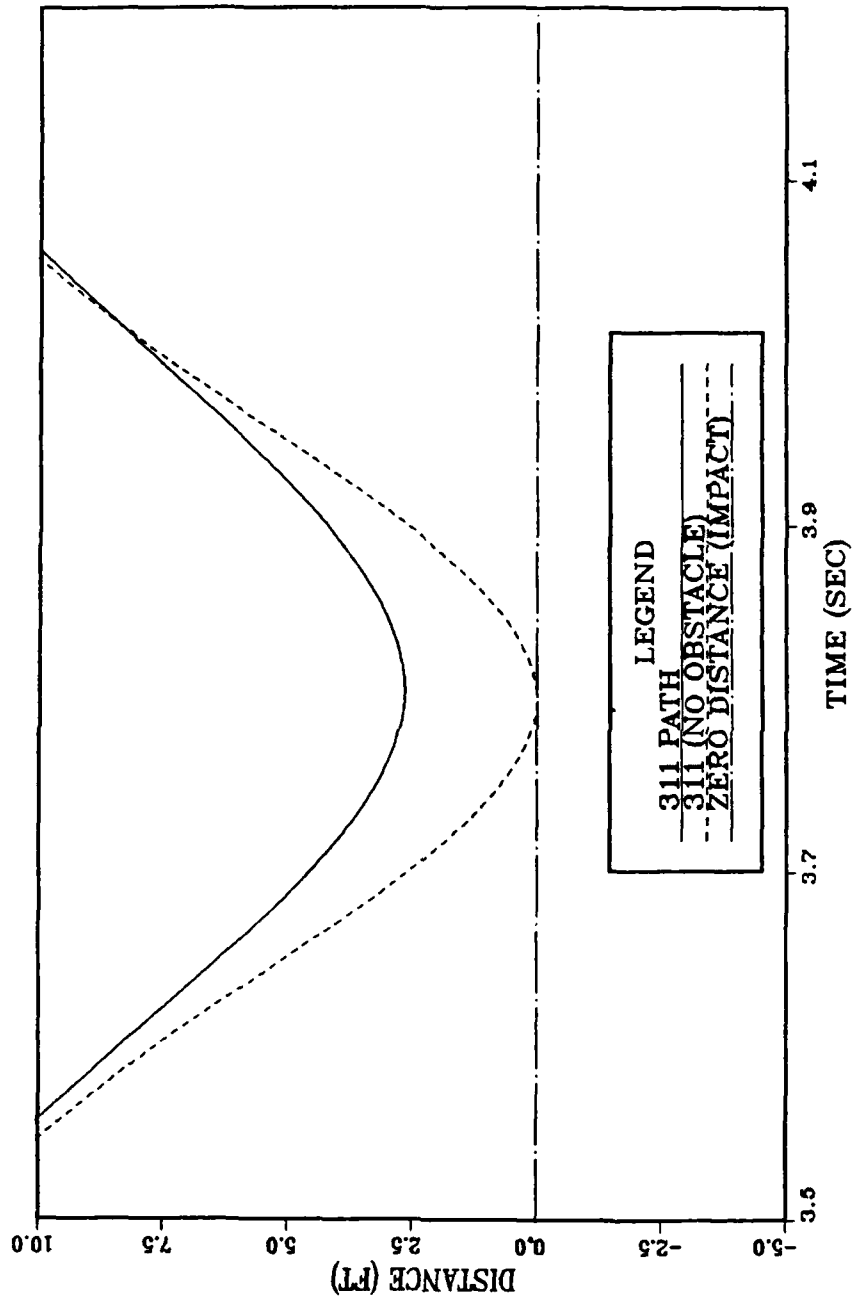


Figure 7.9 One Moving Obstacle Solution

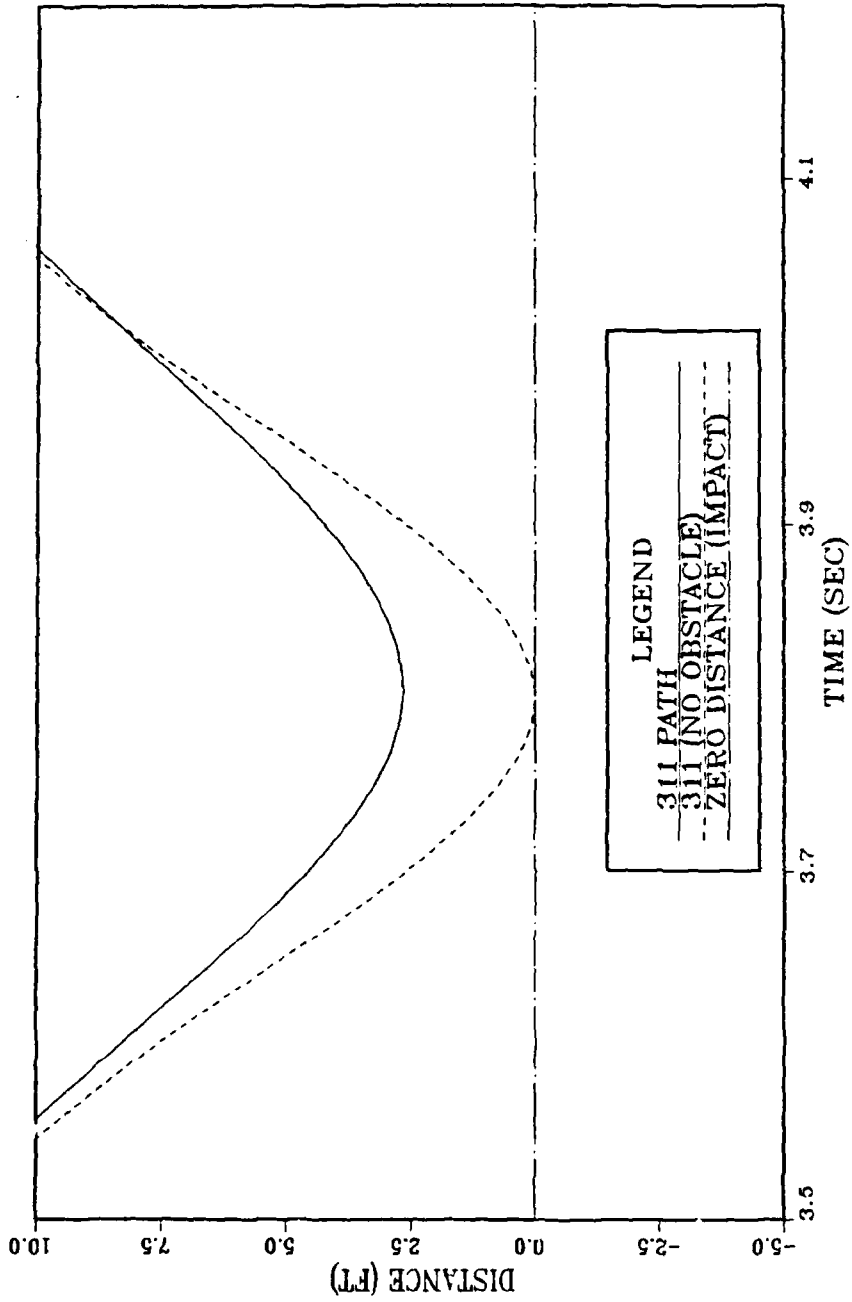


Figure 7.10 Two Moving Obstacles Solution (Obstacle 1)

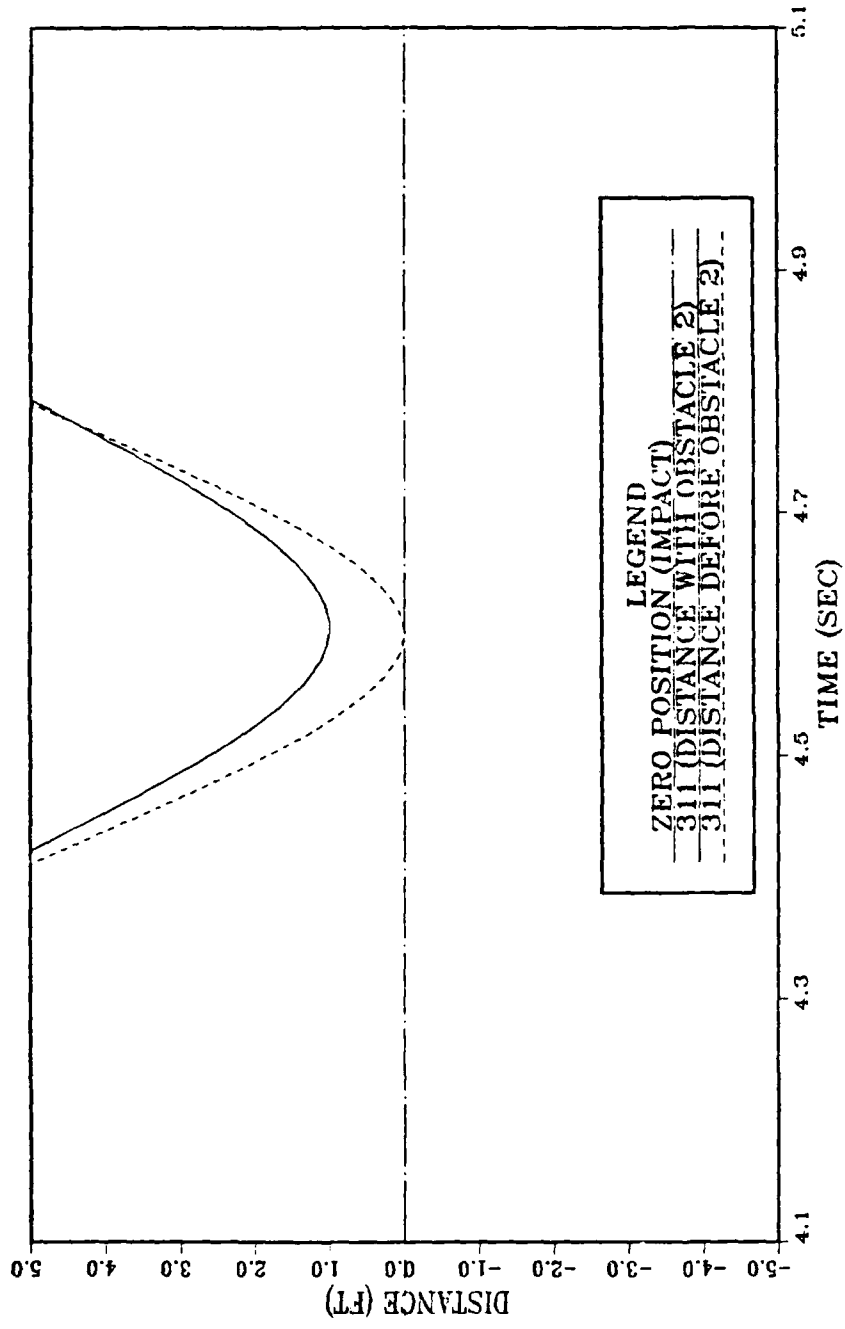


Figure 7.10 Two Moving Obstacles Solution (Obstacle 2)



TABLE 6. PROGRAM 311 COMPUTATIONAL COST-2D

---

NUMBER OF OBSTACLES	TIME (sec)
0	14.23
1 (moving)	23.88
2 (moving)	24.68
17 (fixed)	87.52

---

## VIII. CONCLUSIONS AND RECOMMENDATIONS

### A. CONCLUSIONS

The following conclusions can be drawn from this feasibility study:

1. Optimal control theory is a feasible method for determining obstacle avoidance paths in the presence of fixed and moving obstacles.
2. The introduction of obstacle constraints into the algorithm increases the computer computational costs.
3. The full linear model control inputs are not compatible with the full nonlinear model. When linear commands were placed in the nonlinear model, the vehicle's end condition was inconsistent with the desired end condition.
4. The best general algorithm for the two dimensional case was determined based on the ability of the algorithm to solve a variety of obstacle problems with a shortened FINTIM.
5. Scaling factors can be critical in achieving a desired problem solution.
6. In order to achieve a solution in three dimensions, it is necessary to increase the number of design variables for certain vehicle control inputs.

7. FINTIM is a critical variable whose value can change the solution to a specific problem.

#### B. RECOMMENDATIONS

1. Find a procedure to estimate the optimal scaling factors for end constraints.
2. Study the discretization factors in the three-dimensional case(s).
3. Study the selection of FINTIM in the two-dimensional case and in the three-dimensional case. Vehicle mission objectives should be considered in establishing guidelines.
4. Develop the optimization of the three-dimensional nonlinear model.
5. Develop the algorithm to include optimization of the propeller rpm control input.
6. Develop the program for efficient programming in a microprocessor.
7. Determine if the general algorithm recommended in this thesis is also applicable for three-dimensional obstacle avoidance cases.

## **APPENDIX**

### **PROGRAMS**

This appendix contains the four primary programs that were used for this feasibility study. They were:

1. OBST DSL - This is the state linear 2D model used for the 2D analysis.
2. TLO DSL - This is the ADSL program used to optimize the full scale linear model of bow plane, stern plane and rudder control vectors.
3. TNLO DSL - This is the ADSL program used to optimize the full scale nonlinear model for timing comparison studies.
4. TLNLO DSL - This is the ADSL program used to optimize the full scale linear model for bow plane, stern plane and rudder control inputs and with simulation of the full scale nonlinear model.

FILE: OBS DSL A1

TITLE LINEAR AUV DYNAMIC PATH PLANNER FOR VERTICAL PLANE MOTION  
\* SEPARATED BOW AND STERN PLANE CONTROL NON-DIMENSIONAL

\* 2D STATIONARY OBSTACLES

\*\*\*\*\* ADSL SET UP \*\*\*\*\*

\*  
FIXED ISTRAT, IOPT, IONED, IPRINT, INFO, IGRAD, NDV, NCON  
FIXED IDG, NGT, IC, NRA, NCOLA, NRWK, IWK, NRIWK, 0  
D DIMENSION AW(42,42)  
ARRAY WK(5000), IWK(500)  
ARRAY DX(21), VLB(21), VUB(21), GW(05), DF(21), IDG(05), IC(05)  
PARAM NRA=42, NCOLA=42, NRWK=5000, NRIWK=500  
PARAM IGRAD=0, INFO=0, NDV=20, NCON=05, NGT=05  
TABLE DX(1-2)=2\*.0, DX(3-21)=19\*0., IDG(1-4)=4\*-1  
TABLE VLB(1-9)=9\*-.17452, VLB(11-19)=9\*-.2443, VLB(10)=0., VLB(20-21)=0.  
TABLE VUB(1-9)=9\*.17452, VUB(11-19)=9\*.2443, VUB(10)=0., VUB(20-21)=0.  
TABLE IDG(5)=01\*1  
PARAM ISTRAT=3, IOPT=1, IONED=1, IPRINT=0000  
INCON U=0.0  
METHOD RECT  
CONTROL FINTIM=7.0, DELT=.1  
PRINT XPOS, ZPOS  
\*RINT DS, DB, DEPTH, PITCH, XPOS, ZPOS, DT

\*\*\*\*\* DSL MODEL SET UP \*\*\*\*\*

\* EQUILIBRIUM CONDITION IS CONSTANT SPEED (NON-DIMENSIONALIZED) BY

\* UO = 6 FT/SEC  
CONST UO=1.0, A=10, B=11, C=12, D=13, E=14, F=15, G=16, H=17  
\*ONST XOBS1=36.0, ZOBS1=-12.0, XOBS2=72.5, ZOBS2=-5.82  
\*ONST XOBS3=90.0, ZOBS3=-16., XOBS4=115., ZOBS4=-17.

\*\*\*\*\*  
CONST XOBS1=34.85, ZOBS1=-2.8297  
CONST XOBS2=83.64, ZOBS2=-12.960  
CONST XOBS3=122.5, ZOBS3=-2.90  
CONST XOBS4=105.0, ZOBS4=-8.74  
CONST XOBS5=59.245, ZOBS5=-7.2363  
CONST XOBS6=35.0, ZOBS6=-14.58  
CONST XOBS7=17.5, ZOBS7=-8.74  
CONST XOBS8=52.5, ZOBS8=-11.66  
CONST XOBS9=69.7, ZOBS9=-10.155  
CONST XOBS10=70.0, ZOBS10=-2.9  
CONST XOBS11=52.275, ZOBS11=-6.1408  
CONST XOBS12=52.275, ZOBS12=-5.21  
CONST XOBS13=52.275, ZOBS13=-6.8836  
CONST XOBS14=50.0, ZOBS14=-5.21  
CONST XOBS15=54.55, ZOBS15=-5.21  
CONST XOBS16=57.5, ZOBS16=-8.74  
CONST XOBS17=35.0, ZOBS17=-8.74

```

CONST UO=
*
CONST MA=      , THETA0=      , ZO=      , WO=      , IY=
*
CONST ZW=      , ZQ=      , ZQDOT=      , ZWDOT=
CONST ZDB=     , ZDS=      , ZO=
*
CONST MW=      , MQ=      , MQDOT=      , MWDOT=
CONST MDB=     , MDS=      , MTHETA=
*
INITIAL
DSAVE1= SQRT((XPOS-XOBS1)*(XPOS-XOBS1)+(ZPOS-ZOBS1)*(ZPOS-ZOBS1))
DSAVE2= SQRT((XPOS-XOBS2)*(XPOS-XOBS2)+(ZPOS-ZOBS2)*(ZPOS-ZOBS2))
DSAVE3= SQRT((XPOS-XOBS3)*(XPOS-XOBS3)+(ZPOS-ZOBS3)*(ZPOS-ZOBS3))
DSAVE4= SQRT((XPOS-XOBS4)*(XPOS-XOBS4)+(ZPOS-ZOBS4)*(ZPOS-ZOBS4))
DSAVE5= SQRT((XPOS-XOBS5)*(XPOS-XOBS5)+(ZPOS-ZOBS5)*(ZPOS-ZOBS5))

DSAVE6= SQRT((XPOS-XOBS6)*(XPOS-XOBS6)+(ZPOS-ZOBS6)*(ZPOS-ZOBS6))
DSAVE7= SQRT((XPOS-XOBS7)*(XPOS-XOBS7)+(ZPOS-ZOBS7)*(ZPOS-ZOBS7))
DSAVE8= SQRT((XPOS-XOBS8)*(XPOS-XOBS8)+(ZPOS-ZOBS8)*(ZPOS-ZOBS8))
DSAVE9= SQRT((XPOS-XOBS9)*(XPOS-XOBS9)+(ZPOS-ZOBS9)*(ZPOS-ZOBS9))
DSAVEA= SQRT((XPOS-XOBS10)*(XPOS-XOBS10)+...
(ZPOS-ZOBS10)*(ZPOS-ZOBS10))
DSAVEB= SQRT((XPOS-XOBS10)*(XPOS-XOBS10)+...
(ZPOS-ZOBS10)*(ZPOS-ZOBS10))
DSAVEC= SQRT((XPOS-XOBS10)*(XPOS-XOBS10)+...
(ZPOS-ZOBS10)*(ZPOS-ZOBS10))
DSAVED= SQRT((XPOS-XOBS10)*(XPOS-XOBS10)+...
(ZPOS-ZOBS10)*(ZPOS-ZOBS10))
DSAVEE= SQRT((XPOS-XOBS10)*(XPOS-XOBS10)+...
(ZPOS-ZOBS10)*(ZPOS-ZOBS10))
DSAVEF= SQRT((XPOS-XOBS10)*(XPOS-XOBS10)+...
(ZPOS-ZOBS10)*(ZPOS-ZOBS10))
DSAVEG= SQRT((XPOS-XOBS10)*(XPOS-XOBS10)+...
(ZPOS-ZOBS10)*(ZPOS-ZOBS10))
DSAVEH= SQRT((XPOS-XOBS10)*(XPOS-XOBS10)+...
(ZPOS-ZOBS10)*(ZPOS-ZOBS10))
QRDDEP = 1.0
A1=-ZW
A2=(MA-ZWDOT)
A3=-(ZQ+MA)
A4=-ZQDOT
B1=-MW
B2=-MWDOT
B3=-MQ
B4=(IY-MQDOT)
B5=-MTHETA
C1=ZDS
C2=ZDB
C3=MDS
C4=MDB
C5=C3-(B4*C1)/A4
C6=C4-(B4*C2)/A4
D1=B2-(B4*A2)/A4
D2=B1-(B4*A1)/A4
D3=B3-(B4*A3)/A4
CALL DADS(INFO,ISTRAT,IOPT,IONED,IPRINT,IRAD,NDV,NCON,DX,...
VLB,VUB,OBJ,GN,IDG,NGT,IC,DF,AW,NRA,NCOLA,WK,NRWK,...
IWK,NRIWK)
IF(INFO.EQ.0) THEN
SAVE THETDD,THETAD,THETA,W,Z,DEPTH,PITCH,DS,DB,BOWANG,STNANG
ELSE
ENDIF
IF(INFO.EQ.0) DELPRT = 0.2
IF(INFO.EQ.0) DELPLT = 0.2
DERIVATIVE
*
```

```

THETDD=(1/A4)*(C1*DS+C2*DB-A1*W-A2*WDOT-A3*THETAD)
WDOT= (1/D1)*(C5*DS + C6*DB - D2*W - D3*THETAD - B5*THETA)
THETAD= INTGRL(THETA0,THETDD)
THETA = INTGRL(THETA0,THETAD)
W = INTGRL(W0,WDOT)
Z = INTGRL(Z0,W-UO*THETA)
DEPTH=-Z
PITCH=THETA/.01745
BOWANG=(DB/.01745)
STNANG=(DS/.01745)
INTGRD = ((W*W+(Z-ORDDEP)*(Z-ORDDEP)+...
          THETAD*THETAD+THETA*THETA)) + (DS*DS+DB*DB)
OBJ1 = INTGRL(0.,(0.5)*INTGRD)
OBJ = OBJ1
*
* DYNAMIC
RN=TIME/(FINTIM/10.-DELT/10000.)
O=INT(RN)+1
IF(O.EQ.11) O=10
*
* ADDITIONALLY THE PLANES SHOULD BE AT EQUILIBRIUM SO THE
* VEHICLE WILL PROCEED AT THIS NEW DEPTH WITHIN SOME TOLERANCE
*
* DS=DX(0)
* DB=DX(10+0)
* IF(O.GE.10) DS=0.
* IF(O.GE.10) DB=0.
*
* CONSTRAINTS FOR A DIVE
*
* ORDERED DEPTH = ORDDEP
* GW(1) =(Z-ORDDEP)/2.
* GW(2) =(ORDDEP-Z)/2.
* AUV'S FINAL STATE MUST BE LEVEL FLIGHT AS FOLLOWS
* GW(3) = THETA *10.
* GW(4) = -THETA*10.
* GW(5)=-Z*100.
*
* X-Z POSITIONING FOR OBSTACLE AVOIDANCE
*
* XPOS=17.425*TIME
* ZPOS=-Z*17.425
* DT=TIME*20./FINTIM

```

\*  
\*

AVOIDING THE OBSTACLE

```
DIST1=SQRT((XPOS-XOBS1)*(XPOS-XOBS1)+(ZPOS-ZOBS1)*(ZPOS-ZOBS1))
DIST2=SQRT((XPOS-XOBS2)*(XPOS-XOBS2)+(ZPOS-ZOBS2)*(ZPOS-ZOBS2))
DIST3=SQRT((XPOS-XOBS3)*(XPOS-XOBS3)+(ZPOS-ZOBS3)*(ZPOS-ZOBS3))
DIST4=SQRT((XPOS-XOBS4)*(XPOS-XOBS4)+(ZPOS-ZOBS4)*(ZPOS-ZOBS4))
DIST5=SQRT((XPOS-XOBS5)*(XPOS-XOBS5)+(ZPOS-ZOBS5)*(ZPOS-ZOBS5))
DIST6=SQRT((XPOS-XOBS6)*(XPOS-XOBS6)+(ZPOS-ZOBS6)*(ZPOS-ZOBS6))
DIST7=SQRT((XPOS-XOBS7)*(XPOS-XOBS7)+(ZPOS-ZOBS7)*(ZPOS-ZOBS7))
DIST8=SQRT((XPOS-XOBS8)*(XPOS-XOBS8)+(ZPOS-ZOBS8)*(ZPOS-ZOBS8))
DIST9=SQRT((XPOS-XOBS9)*(XPOS-XOBS9)+(ZPOS-ZOBS9)*(ZPOS-ZOBS9))
DIST10=SQRT((XPOS-XOBS10)*(XPOS-XOBS10)+(ZPOS-ZOBS10)*...
(ZPOS-ZOBS10))
DIST11=SQRT((XPOS-XOBS11)*(XPOS-XOBS11)+(ZPOS-ZOBS11)*...
(ZPOS-ZOBS11))
DIST12=SQRT((XPOS-XOBS12)*(XPOS-XOBS12)+(ZPOS-ZOBS12)*...
(ZPOS-ZOBS12))
DIST13=SQRT((XPOS-XOBS13)*(XPOS-XOBS13)+(ZPOS-ZOBS13)*...
(ZPOS-ZOBS13))
DIST14=SQRT((XPOS-XOBS14)*(XPOS-XOBS14)+(ZPOS-ZOBS14)*...
(ZPOS-ZOBS14))
DIST15=SQRT((XPOS-XOBS15)*(XPOS-XOBS15)+(ZPOS-ZOBS15)*...
(ZPOS-ZOBS15))
DIST16=SQRT((XPOS-XOBS16)*(XPOS-XOBS16)+(ZPOS-ZOBS16)*...
(ZPOS-ZOBS16))
DIST17=SQRT((XPOS-XOBS17)*(XPOS-XOBS17)+(ZPOS-ZOBS17)*...
(ZPOS-ZOBS17))
IF(DIST1.LT.DSAVE1) DSAVE1=DIST1
IF(DIST2.LT.DSAVE2) DSAVE2=DIST2
IF(DIST3.LT.DSAVE3) DSAVE3=DIST3
IF(DIST4.LT.DSAVE4) DSAVE4=DIST4
IF(DIST5.LT.DSAVE5) DSAVE5=DIST5
IF(DIST6.LT.DSAVE6) DSAVE6=DIST6
IF(DIST7.LT.DSAVE7) DSAVE7=DIST7
IF(DIST8.LT.DSAVE8) DSAVE8=DIST8
IF(DIST9.LT.DSAVE9) DSAVE9=DIST9
IF(DIST10.LT.DSAVEA) DSAVEA=DIST10
IF(DIST11.LT.DSAVEB) DSAVEB=DIST11
IF(DIST12.LT.DSAVEC) DSAVEC=DIST12
IF(DIST13.LT.DSAVED) DSAVED=DIST13
IF(DIST14.LT.DSAVEE) DSAVEE=DIST14
IF(DIST15.LT.DSAVEF) DSAVEF=DIST15
IF(DIST16.LT.DSAVEG) DSAVEG=DIST16
IF(DIST17.LT.DSAVEH) DSAVEH=DIST17
GW(6)=(9.-DSAVE1)
GW(7)=(5.-DSAVE2)
GW(8)=(3.-DSAVE3)
GW(9)=(6.-DSAVE4)
GW(10)=(1.-DSAVE5)
GW(11)=(1.-DSAVE6)
GW(12)=(1.-DSAVE7)
```



FILE: OBS      DSL      A1

GW(13)=(1.-DSAVE8)  
GW(14)=(1.-DSAVE9)  
GW(15)=(1.-DSAVEA)  
GW(16)=(1.-DSAVEB)  
GW(17)=(1.-DSAVEC)  
GW(18)=(1.-DSAVED)  
GW(19)=(1.-DSAVEE)  
GW(20)=(1.-DSAVEF)  
GW(21)=(1.-DSAVEG)  
GW(22)=(1.-DSAVEH)

TERMINAL

IF(INFO.EQ.0) THEN  
PRINT\*,DSAVE1,DSAVE2  
PRINT\*,DSAVE3,DSAVE4  
PRINT\*,DSAVE5,DSAVE6  
PRINT\*,DSAVE7,DSAVE8  
PRINT\*,DSAVE9,DSAVEA  
PRINT\*,DSAVEB,DSAVEC  
PRINT\*,DSAVED,DSAVEE  
PRINT\*,DSAVEF,DSAVEG  
PRINT\*,DSAVEH  
ELSE  
ENDIF  
IF(INFO.EQ.0) CALL ENDJOB  
CALL RERUN

\*  
END  
STOP

FILE: TLO DSL A1

TITLE RUN:16-5 LINEAR AUV MODEL / STERN PLANE AND BOW PLANE SEPARATED

```
* (1) UPDATED:04/16/88
* (2) 100.00 FT DEPTH CHANGE IN 20 SEC
* (3) RIGHT OBJ EQUATION
* (4) ADS CONSTRAINTS ON DEPTH AND PITCH
* (5) OBSTACLE FURTHER DOWN THE TRAJECTORY AND ABOVE IT
* (6) CORRECT OBSTACLE AVOIDANCE ROUTINE ADDED
*****ADSL SET-UP*****
FIXED ISTRAT, IOPT, IONED, IPRINT, INFO, IGRAD, NDV, NCON
FIXED IDG, NGT, IC, NRA, NCOLA, NRWK, IWK, NRIWK, O, H,D,C,P
D DIMENSION AW(42,42)
ARRAY WK(4000), IWK(1000)
ARRAY DX(40), VLB(40), VUB(40), GW(11), DF(41), IDG(11), IC(11)
PARAM NRA=42, NCOLA=42, NRWK=4000, NRIWK=1000
PARAM IGRAD=0, INFO=0, NDV=40, NCON=11, NGT=11
TABLE DX(1-2)=2*.0,DX(3-21)=19*0., IDG(1-10)=10*-1
TABLE DX(22-40)=19*0.
TABLE IDG(7-0)=1*1
TABLE VLB(1-9)=9*-.17452, VLB(11-19)=09*-.2443,VLB(10)=0.,VLB(20)=0.
TABLE VUB(1-9)=9*.17452, VUB(11-19)=9*.2443,VUB(10)=0.,VUB(20)=0.
TABLE VLB(21-39)=19*-.523627,VUB(21-39)=19*.523627,VUB(40-41)=0.
TABLE VLB(40-41)=0.
PARAM ISTRAT=3, IOPT=1, IONED=1, IPRINT=0000
INCON H=0, OBS1=0.,YZONE=0.
METHOD RECT
CONTROL FINTIM=21., DELT=.10
*PRINT XPOSM,YPOSM,DEPTH,THETAM,PHIM,PSIM,DSM,DBM,DRM
PRINT XPOSM,YPOSM,DEPTH
*PRINT DSM,DBM,DRM,PITCHM,XPOSM,YPOSM,DEPTH,NDT
*PRINT THETAD,W,DEPTH,PITCH,XPOS,DEPTH,NDX,NDZ,NDT
*AVE THETA,W,Z,DEPTH,PITCH,DS,DB,BOWANG,STNANG
*RAPH(DE=TEK618) TIME,DS
*RAPH(DE=TEK618) TIME,DEPTH
*RAPH(DE=TEK618) TIME,WDOT
*RAPH(DE=TEK618) TIME,W
*RAPH(DE=TEK618) TIME,THETDD
*RAPH(DE=TEK618) TIME,THETAD
*RAPH(DE=TEK618) TIME,THETA
*RAPH(DE=TEK618) TIME,PITCH
*RAPH(DE=TEK618) TIME,BOWANG
*RAPH(DE=TEK618) TIME,STNANG
*****D S L MODEL FOR LINEAR SIMULATION *****
* LINEAR MODEL ONLY
D COMMON/BLOCK1/ MMINV(6,6), MM(6,6), AA(12,12), BB(6,6)
D COMMON/BLOCK2/ B(6,6),A(12,12), UMOD(6),GKK(6,21)
D COMMON/BLOCK3/ F(12), FP(6), UCF(4)
D COMMON/BLOCK4/ G4(4),GK4(4),BR(4),HH(4)
D COMMON/BLOCK5/ XDOT(12),XDOTX(12), XDOTU(6)
FIXED N,IA,IDGT,IER,LAST,J,K,M,JJ,KK,I
INTEGER
ARRAY WKAREA(54), X(12)
*
*
* CONST
*
* LONGITUDINAL HYDRODYNAMIC COEFFICIENTS
*
CONST XPP = 7.03E-3 ,XQQ = -1.47E-2 ,XRR = 4.01E-3 ,XPR = 7.64E-4,...
XUDOT=-7.58E-3 ,XWQ = -1.92E-1 ,XVP = -3.24E-3 ,XVR = 1.89E-2,...
XQDS= 2.61E-2 ,XQDB= -2.6E-3 ,XRDR= -8.18E-4 ,XVV = 5.29E-2,...
XWV = 1.71E-1 ,XVDR= 1.73E-3 ,XWDS= 4.6E-2 ,XWDB= 9.66E-3,...
XDSDS=-1.16E-2 ,XDBDB=-8.07E-3 ,XDRDR=-1.01E-2 ,XQDSN=1.96E-3,...
XWDSN=3.46E-3 ,XSDSN=-1.62E-3
```

```

*
* LATERAL HYDRODYNAMIC COEFFICIENTS
*
CONST YPDOT=1.27E-4 , YRDOT=1.24E-3 , YPQ = 4.125E-3 , YQR =-6.51E-3,...
      YVDOT=-5.55E-2 , YP = 3.055E-3 , YR = 2.97E-2 , YVQ = 2.36E-2,...
      YWP = 2.35E-1 , YWR = -1.88E-2 , YV = -9.31E-2 , YVW = 6.84E-2,...
      YDR = 2.73E-2 , CDY = 3.5E-1
*
* NORMAL HYDRODYNAMIC COEFFICIENTS
*
CONST ZQDOT=-6.81E-3 , ZPP = 1.27E-4 , ZPR = 6.67E-3 , ZRR =-7.35E-3,...
      ZWDOT=-2.43E-1 , ZQ = -1.35E-1 , ZVP = -4.81E-2 , ZVR = 4.55E-2,...
      ZW = -3.02E-1 , ZVV = -6.84E-2 , ZDS = -7.255E-2 , ZDB =-2.58E-2,...
      ZQN = -2.88E-3 , ZWN = -5.07E-3 , ZDSN = -1.015E-2 , CDZ = 1.0
*
* ROLL HYDRODYNAMIC COEFFICIENTS
*
CONST KPDOT=-1.01E-3 , KRDOT=-3.37E-5 , KPQ = -6.93E-5 , KQR = 1.68E-2,...
      KVDOT=1.27E-4 , KP = -1.1E-2 , KR = -8.41E-4 , KVQ=-5.115E-3,...
      KWP = -1.27E-4 , KWR = 1.39E-2 , KV = 3.055E-3 , KVV =-1.87E-1,...
      KPN = -5.73E-4 , KDB = 6.94E-3
*
* PITCH HYDRODYNAMIC COEFFICIENTS
*
CONST MQDOT= -1.68E-2 , MPP = 5.26E-5 , MPR = 5.04E-3 , MRR =-2.86E-2,...
      MWDOT=-6.81E-3 , MQ = -6.86E-2 , MVP = 1.18E-3 , MVR = 1.73E-2,...
      MW = 9.86E-2 , MVV = -2.51E-2 , MDS = -4.12E-2 , MDB = 6.94E-3,...
      MQN = -1.64E-3 , MWN = -2.88E-3 , MDSN = -5.76E-3
*
* YAW HYDRODYNAMIC COEFFICIENTS
*
CONST NPDOT=-3.37E-5 , NRDOT=-3.4E-3 , NPQ = -2.11E-2 , NQR = 2.75E-3,...
      NVDOT=1.24E-3 , NP = -8.405E-4 , NR = -1.64E-2 , NVQ =-9.99E-3,...
      NWP = -1.75E-2 , NWR = 7.35E-3 , NV = -7.42E-3 , NVW =-2.67E-2,...
      NDR = -1.29E-2
*
* MASS CHARACTERISTICS OF THE FLOODED MARK IX VEHICLE
*
CONST WEIGHT = 15900 , BOY = 15900 , VOL = 248.44 , XG =-0.1 , ...
      YG = 0.0 , ZG = 0.061 , XB =-0.1 , ZB =0.023 , ...
      IX = 1760 , IY = 9450 , IZ = 10700 , IXZ = -6.65 , ...
      IYZ = -7.0 , IXY = -7.0 , YB = 0.0 , ...
      L = 17.425 , RHO = 1.94 , G = 32.2 , NU = 1.5E-5 , ...
      AO = 1.57 , KPROP = 0.0 , NPROP = 0.0 , X1TEST=0.01 , ...
      DEGRUD= 10.0 , DEGSTN= 0.0
*
CONST XOBS1=36.0
CONST ZOBS1= -12.0

```

```

* INPUT INITIAL CONDITIONS HERE IF REQUIRED
*
* INITIAL
*
* DSAVE1=SQRT((XPOSM-XOBS1)*(XPOSM-XOBS1)+...
* (ZPOSM-ZOBS1)*(ZPOSM-ZOBS1))
* INITIALIZE ALL MATRICES AND ARRAYS TO ZERO
*
NOSORT
ORDDEP=20.
YORD= 40.
D=0
H=H+1
IF (H.EQ.1) THEN
N = 6
DO 2 J = 1,N
JJ= J+N
DO 1 K = 1,N
KK= K+N
KKK= KK + N
MMINV(J,K) = 0.0
X(J) = 0.0
X(JJ) = 0.0
XDOT(J) = 0.0
XDOT(JJ) = 0.0
XDOTX(J) = 0.0
XDOTX(JJ) = 0.0
XDOTU(J) = 0.0
UMOD(J) = 0.0
MM(J,K) = 0.0
BB(J,K) = 0.0
B(J,K) = 0.0

AA(J,K)= 0.0
AA(JJ,KK)= 0.0
AA(J,KK)= 0.0
AA(JJ,K)= 0.0
A(J,KK)= 0.0
A(JJ,K) = 0.0
A(J,K) = 0.0
A(JJ,KK)= 0.0
GKK(J,K)= 0.0
GKK(J,KK)=0.0
GKK(J,KKK)=0.0
CONTINUE
1
2
*
* INPUT THE LINEARIZATION POINT PARAMETERS
*
UO ≈6.0
VO = 0.0
WO = 0.0
PO = 0.0
QO = 0.0
RO = 0.0
PHIO = 0.0
THETAO = 0.0
PSIO = 0.0
SUM = 0.0
JFLAG = 0
IFLAG = 0
KFLAG = 0
*

```

```

* INPUT THE MODEL STATES INITIAL CONDITIONS
*
UM = 6.0
VM = 0.0
WM = 0.0
PM = 0.0
QM = 0.0
RM = 0.0
XPOSM = 0.0
YPOSM = 0.0
ZPOSM = 0.0
PHIM = 0.0
THETAM = 0.0
PSIM = 0.0

*
* INPUT THE VEHICLE INITIAL CONDITIONS
*
* INITIALIZE THE CONTROLS
*
DBOY= 0.0
DR=0.0
DS= 0.00000
DSM=0.
DBM=0.
DB=0.000000
DRM=0.0
DRPM=0.
RPM = 500.00
LATYAW = 0.0
NORPIT = 0.0

*
*
* MASS = WEIGHT/G
*
DIVAMP = DEGSTN*0.0174532925
RUDAMP = DEGRUD*0.0174532925

*
* THE LINEAR PROPULSION MODEL
*
*
ETA = 0.012*RPM/U0
ETA = 1.0
RE = U0*L/NU
CDO = .00385 + (1.296E-17)*(RE - 1.2E7)**2
CT = 0.008*L**2*ETA*ABS(ETA)/(A0)
CT1 = 0.008*L**2/(A0)
EPS = -1.0+(SQRT(CT+1.0)-1.0)/(SQRT(CT+1.0)-1.0)
XPROP = CDO*(ETA*ABS(ETA) - 1.0)

*
* N=6
* DO 15 J = 1,N
* DO 10 K =1,N
* MMINV(J,K)=0.0
* MM(J,K) = 0.0
*0 CONTINUE
*5 CONTINUE
*
* CALCULATE THE MASS MATRIX
*
MM(1,1) = MASS -((RHO/2)*(L**3)*XUDOT)
MM(1,5) = MASS*ZG
MM(1,6) = -MASS*YG

```

```

MM(2,2) = MASS - ((RHO/2)*(L**3)*YVDOT)
MM(2,4) = -MASS*ZG - ((RHO/2)*(L**4)*YPDOT)
MM(2,6) = MASS*XG - ((RHO/2)*(L**4)*YRDOT)

MM(3,3) = MASS - ((RHO/2)*(L**3)*ZWDOT)
MM(3,4) = MASS*YG
MM(3,5) = -MASS*XG - ((RHO/2)*(L**4)*ZQDOT)

MM(4,2) = -MASS*ZG - ((RHO/2)*(L**4)*KVDOT)
MM(4,3) = MASS*YG
MM(4,4) = IX - ((RHO/2)*(L**5)*KPDOT)
MM(4,5) = -IXY
MM(4,6) = -IXZ - ((RHO/2)*(L**5)*KRDOT)

MM(5,1) = MASS*ZG
MM(5,3) = -MASS*XG - ((RHO/2)*(L**4)*MQDOT)
MM(5,4) = -IXY
MM(5,5) = IY - ((RHO/2)*(L**5)*MQDOT)
MM(5,6) = -IYZ

MM(6,1) = -MASS*YG
MM(6,2) = MASS*XG - ((RHO/2)*(L**4)*NVDOT)
MM(6,4) = -IXZ - ((RHO/2)*(L**5)*NPDOT)
MM(6,5) = -IYZ
MM(6,6) = IZ - ((RHO/2)*(L**5)*NRDOT)

```

```
*
*
```

```

LAST = N*N+3*N
DO 20 M = 1, LAST
WKAREA(M) = 0.0
CONTINUE

```

```
20
*
```

```

IER = 0
IA = 6
IDGT = 4

```

```
*
```

```
CALL LINV2F(MM,N,IA,MMINV,IDGT,WKAREA,IER)
```

```
*
```

```
CALCULATE THE A MATRIX FOR THE LINEAR MODEL
```

```
*
```

```
*
```

```
*
```

```
*
```

```

A(1,1) = RHO/2*L**3*(XQDS*DS*Q0+XQDB/2*DB*Q0+XRDR*R0*DR)+...
RHO/2*L**2*(XVDR*V0*DR+XWDS*DS*W0+XWDB/2*DB*W0 + ...
2*U0*(XSDS*DS**2 + XDBDB/2*DB**2 + XDRDR*DR**2))+ ...
RHO/2*L**3*XQDSN*Q0*DS*EPS+RHO/2*L**2*(XWDSN*W0*DS+...
2*XSDSN*U0*DS**2)*EPS+RHO*L**2*U0*XPROP+RHO/2*L**3*...
XQDB/2*DB*Q0+RHO/2*L**2*XWDB/2*DB*W0+RHO*L**2*U0*...
XDBDB/2*DB**2

```

FILE: TLO DSL A1

A(1,2) = MASS\*R0+RHO/2\*L\*\*3\*(XVP\*P0+ XVR\*R0) + RHO/2\*L\*\*2\* ...  
(2\*XVV\*V0 + XVDR\*U0\*DR)  
A(1,3) = -MASS\*Q0 + RHO/2\*L\*\*3\*(XWQ\*Q0)+RHO/2\*L\*\*2\*(2\*XWW\*W0+...  
XWDS\*DS\*U0+(XWDB/2\*DB+XWDB/2\*DB)\*U0 +XWDSN\*U0\*DS\*EPS)  
A(1,4) = -MASS\*YG\*Q0-MASS\*ZG\*R0+ RHO/2\*L\*\*4\*(2\*XPP\*P0+XPR\*R0)...  
+ RHO/2\*L\*\*3\*(XVP\*V0)  
A(1,5) = -MASS\*W0+2\*MASS\*XG\*Q0 -MASS\*YG\*P0+RHO/2\*L\*\*4\*2\*XQQ\*Q0...  
+RHO/2\*L\*\*3\*(XWQ\*W0+XQDS\*DS\*U0+XQDB/2\*DB\*U0)+RHO/2\* ...  
L\*\*3\*XQDSN\*U0\*DS\*EPS+RHO/2\*L\*\*3\*XQDB/2\*DB\*U0  
A(1,6) = MASS\*V0+2\*MASS\*XG\*R0-MASS\*ZG\*P0+RHO/2\*L\*\*4\*(2\*XRR\*R0...  
+ XPR\*P0) + RHO/2\*L\*\*3\*(XVR\*V0 + XRDR\*U0\*DR)  
A(1,11)= -(WEIGHT - BOY)\*COS(THETA0)  
\*  
A(2,1) = -MASS\*R0+RHO/2\*L\*\*3\*(YP\*P0+YR\*R0)+RHO/2\*L\*\*2\*(YV\*V0+...  
2\*YDR\*U0\*DR)  
A(2,2) = RHO/2\*L\*\*3\*YVQ\*Q0+RHO/2\*L\*\*2\*(YV\*U0+YVW\*W0)  
A(2,3) = MASS\*P0+ RHO/2\*L\*\*3\*(YWP\*P0+YWR\*R0)+RHO/2\*L\*\*2\*YVW\*V0  
A(2,4) = MASS\*W0-MASS\*XG\*Q0+2\*MASS\*YG\*P0+RHO/2\*L\*\*4\*YP\*Q0+...  
RHO/2\*L\*\*3\*(YP\*U0+ YWP\*W0)  
A(2,5) = -MASS\*XG\*P0-MASS\*ZG\*R0+RHO/2\*L\*\*4\*(YP\*P0+YQR\*R0) +...  
RHO/2\*L\*\*3\*YVQ\*V0  
A(2,6) = -MASS\*U0+2\*MASS\*YG\*R0-MASS\*ZG\*Q0+RHO/2\*L\*\*4\*YQR\*Q0 +...  
RHO/2\*L\*\*3\*(YR\*U0 + YWR\*W0)  
A(2,10)= (WEIGHT - BOY)\*COS(THETA0)\*COS(PHIO)  
A(2,11)= -(WEIGHT - BOY)\*SIN(THETA0)\*SIN(PHIO)  
\*  
A(3,1) = MASS\*Q0+RHO/2\*L\*\*3\*ZQ\*Q0+RHO/2\*L\*\*2\*(ZW\*W0+2\*U0\*ZDS\*DS...  
+2\*U0\*ZDB/2\*DB+(ZWN\*W0+2\*ZDSN\*U0\*DS)\*EPS)+RHO/2\*L\*\*3\*...  
ZQN\*Q0\*EPS+ RHO/2\*L\*\*2\*2\*U0\*ZDB/2\*DB  
A(3,2) = -MASS\*P0+RHO/2\*L\*\*3\*(ZVP\*P0+ZVR\*R0)+RHO/2\*L\*\*2\*2\*ZVV\*V0  
A(3,3) = RHO/2\*L\*\*2\*(ZU\*U0 + ZWN\*U0\*EPS)  
A(3,4) = -MASS\*V0-MASS\*XG\*R0+2\*MASS\*ZG\*P0+ RHO/2\*L\*\*4\*(2\*ZPP\*...  
P0 + ZPR\*R0) + RHO/2\*L\*\*3\*ZVP\*V0  
A(3,5) = MASS\*U0 - MASS\*YG\*R0+2\*MASS\*ZG\*Q0+RHO/2\*L\*\*3\*ZQ\*U0 +...  
RHO/2\*L\*\*3\*ZQN\*U0\*EPS  
A(3,6) = -MASS\*XG\*P0-MASS\*YG\*Q0+RHO/2\*L\*\*4\*(ZPR\*P0+2\*ZRR\*R0)+...  
RHO/2\*L\*\*3\*ZVR\*V0  
A(3,10)= -(WEIGHT - BOY)\*COS(THETA0)\*SIN(PHIO)  
A(3,11)= -(WEIGHT - BOY)\*SIN(THETA0)\*COS(PHIO)  
\*  
A(4,1) = MASS\*YG\*Q0 + MASS\*ZG\*R0 + RHO/2\*L\*\*4\*(KP\*P0 + ...  
KR\*R0)+RHO/2\*L\*\*3\*(KV\*V0+2\*U0\*(KDB/2\*DB-KDB/2\*DB))+...  
RHO/2\*L\*\*3\*U0\*KPROP+ RHO/2\*L\*\*4\*KPN\*P0\*EPS  
A(4,2) = -MASS\*YG\*P0 + RHO/2\*L\*\*4\*KV\*Q0 + RHO/2\*L\*\*2\*(KV\*U0...  
+ KVW\*W0)  
A(4,3) = -MASS\*ZG\*P0 + RHO/2\*L\*\*4\*(KWP\*P0 + KWR\*R0) + ...  
RHO/2\*L\*\*3\*KVW\*V0  
A(4,4) = -IXY\*R0 + IXZ\*Q0 - MASS\*YG\*V0 - MASS\*ZG\*W0 + ...  
RHO/2\*L\*\*5\*KPQ\*Q0 + RHO/2\*L\*\*4\*(KP\*U0+KWP\*W0)  
A(4,5) = -IZ\*R0 + IY\*R0 + 2\*IYZ\*Q0 + IXZ\*P0 + MASS\*YG\*U0 +...  
RHO/2\*L\*\*5\*(KPQ\*P0 + KQR\*R0) + RHO/2\*L\*\*4\*KVQ\*V0  
A(4,6) = -IZ\*Q0 + IY\*Q0 - 2\*IYZ\*R0 + MASS\*ZG\*U0 + ...  
RHO/2\*L\*\*5\*KQR\*Q0 + RHO/2\*L\*\*4\*(KR\*U0+KWR\*W0)  
A(4,10)= -(YG\*WEIGHT-YB\*BOY)\*COS(THETA0)\*SIN(PHIO)...  
-(ZG\*WEIGHT-ZB\*BOY)\*COS(THETA0)\*COS(PHIO)  
A(4,11)= -(YG\*WEIGHT-YB\*BOY)\*SIN(THETA0)\*COS(PHIO)...  
+(ZG\*WEIGHT-ZB\*BOY)\*SIN(THETA0)\*SIN(PHIO)  
\*

$A(5,1) = -MASS*XC*QO + RHO/2*L**4*MQ*QO + RHO/2*L**3*MW*WO + \dots$   
 $RHO/2*L**3*UO*(MDS*DS+MDB/2*DB) + RHO/2*L**4*MQN*QO* \dots$   
 $EPS + RHO/2*L**3*(MWN*WO + 2*MDSN*UO*DS)*EPS + \dots$   
 $RHO/2*L**3*UO*MDB/2*DB$   
 $A(5,2) = MASS*XC*PO + MASS*ZG*RO + RHO/2*L**4*(MVP*PO + \dots$   
 $MVR*RO) + RHO*L**3*MVV*VO$   
 $A(5,3) = -MASS*ZG*QO + RHO/2*L**3*MW*UO + RHO/2*L**3*MWN*UO*EPS$   
 $A(5,4) = -IX*RO + IZ*RO - IYZ*QO - 2*IXZ*PO + MASS*XC*VO + \dots$   
 $RHO/2*L**5*(2*MPP*PO + MPR*RO) + RHO/2*L**4*MVP*VO$   
 $A(5,5) = IXY*RO - IYZ*PO - MASS*XC*UO - MASS*ZG*WO + RHO/2* \dots$   
 $L**4*MQ*UO + RHO/2*L**4*MQN*UO*EPS$   
 $A(5,6) = -IX*PO + IZ*PO + IXY*QO + 2*IXZ*RO + MASS*ZG*VO + \dots$   
 $RHO/2*L**5*(MPR*PO+2*MRR*RO)+RHO/2*L**4*MVR*VO$   
 $A(5,10) = (XC*WEIGHT-XB*BOY)*COS(THETAO)*SIN(PHIO)$

$A(5,11) = (XC*WEIGHT-XB*BOY)*SIN(THETAO)*COS(PHIO) - \dots$   
 $(ZG*WEIGHT-ZB*BOY)*COS(THETAO)$

\*

$A(6,1) = -MASS*XC*RO + RHO/2*L**4*(NP*PO + NR*RO) + RHO/2* \dots$   
 $L**3*(NV*VO+2*NDR*UO*DR)+RHO*L**3*UO*NPROP$   
 $A(6,2) = -MASS*YG*RO + RHO/2*L**4*NVQ*QO + RHO/2*L**3*(NV*UO+ \dots$   
 $NVW*WO)$   
 $A(6,3) = MASS*XC*PO + MASS*YG*QO + RHO/2*L**4*(NWP*PO+NWR*RO)+ \dots$   
 $RHO/2*L**3*NVW*VO$   
 $A(6,4) = -IY*QO + IX*QO + 2*IXY*PO + IYZ*RO + MASS*XC*WO+ \dots$   
 $RHO/2*L**5*NPQ*QO + RHO/2*L**4*(NP*UO+NWP*WO)$   
 $A(6,5) = -IY*PO + IX*PO - 2*IXY*QO - IXZ*RO + MASS*YG*WO+ \dots$   
 $RHO/2*L**5*(NPQ*PO+NQR*RO) + RHO/2*L**4*NVQ*VO$   
 $A(6,6) = IYZ*PO - IXZ*QO - MASS*XC*UO - MASS*YG*VO + \dots$   
 $RHO/2*L**5*NQR*QO + RHO/2*L**4*(NR*UO + NWR*WO)$   
 $A(6,10) = (XC*WEIGHT-XB*BOY)*COS(THETAO)*COS(PHIO)$   
 $A(6,11) = -(XC*WEIGHT-XB*BOY)*SIN(THETAO)*SIN(PHIO) + \dots$   
 $(YG*WEIGHT-YB*BOY)*COS(THETAO)$

\*

$A(7,1) = COS(PSIO)*COS(THETAO)$   
 $A(7,2) = COS(PSIO)*SIN(THETAO)*SIN(PHIO) - SIN(PSIO)*COS(PHIO)$   
 $A(7,3) = COS(PSIO)*SIN(THETAO)*COS(PHIO) + SIN(PSIO)*SIN(PHIO)$   
 $A(7,10) = VO*COS(PSIO)*SIN(THETAO)*COS(PHIO) + VO*SIN(PSIO)* \dots$   
 $SIN(PHIO) - WO*COS(PSIO)*SIN(THETAO)*SIN(PHIO) + \dots$   
 $WO*SIN(PSIO)*COS(PHIO)$   
 $A(7,11) = -UO*COS(PSIO)*SIN(THETAO) + VO*COS(PSIO)*COS(THETAO)* \dots$   
 $SIN(PHIO) + WO*COS(PSIO)*COS(THETAO)*COS(PHIO)$   
 $A(7,12) = -UO*SIN(PSIO)*COS(THETAO) - VO*SIN(PSIO)*SIN(THETAO)* \dots$   
 $SIN(PHIO) - VO*COS(PSIO)*COS(PHIO) - WO*SIN(PSIO)* \dots$   
 $SIN(THETAO)*SIN(PHIO) + WO*COS(PSIO)*SIN(PHIO)$

\*

$A(8,1) = SIN(PSIO)*COS(THETAO)$   
 $A(8,2) = SIN(PSIO)*SIN(THETAO)*SIN(PHIO) + COS(PSIO)*COS(PHIO)$   
 $A(8,3) = SIN(PSIO)*SIN(THETAO)*COS(PHIO) - COS(PSIO)*SIN(PHIO)$   
 $A(8,10) = VO*SIN(PSIO)*SIN(THETAO)*COS(PHIO) - VO*COS(PSIO)* \dots$   
 $SIN(PHIO) - WO*SIN(PSIO)*SIN(THETAO)*SIN(PHIO) - \dots$   
 $WO*COS(PSIO)*COS(PHIO)$   
 $A(8,11) = -UO*SIN(PSIO)*SIN(THETAO) + VO*SIN(PSIO)*COS(THETAO)* \dots$   
 $SIN(PHIO) + WO*SIN(PSIO)*COS(THETAO)*COS(PHIO)$   
 $A(8,12) = UO*COS(PSIO)*COS(THETAO) + VO*COS(PSIO)*SIN(THETAO)* \dots$   
 $SIN(PHIO) - VO*SIN(PSIO)*COS(PHIO) + WO*COS(PSIO)* \dots$   
 $SIN(THETAO)*COS(PHIO) + WO*SIN(PSIO)*SIN(PHIO)$

\*

$A(9,1) = -SIN(THETAO)$   
 $A(9,2) = COS(THETAO)*SIN(PHIO)$   
 $A(9,3) = COS(THETAO)*COS(PHIO)$   
 $A(9,10) = VO*COS(THETAO)*COS(PHIO) - WO*COS(THETAO)*SIN(PHIO)$   
 $A(9,11) = -UO*COS(THETAO) - VO*SIN(THETAO)*SIN(PHIO) - \dots$   
 $WO*SIN(THETAO)*COS(PHIO)$

\*



```

A(10,4) = 1.0
A(10,5) = SIN(PHIO)*TAN(THETA0)
A(10,6) = COS(PHIO)*TAN(THETA0)
A(10,10) = Q0*COS(PHIO)*TAN(THETA0) - R0*SIN(PHIO)*TAN(THETA0)
A(10,11) = Q0*SIN(PHIO)/COS(THETA0)*1.0/COS(THETA0) + ...
R0*COS(PHIO)/COS(THETA0)*1.0/COS(THETA0)
*
A(11,5) = COS(PHIO)
A(11,6) = -SIN(PHIO)
A(11,10) = -Q0*SIN(PHIO) - R0*COS(PHIO)
*
A(12,5) = SIN(PHIO)/COS(THETA0)
A(12,6) = COS(PHIO)/COS(THETA0)
A(12,10) = Q0*COS(PHIO)/COS(THETA0) - R0*SIN(PHIO)/COS(THETA0)
A(12,11) = Q0*SIN(PHIO)/COS(THETA0)*TAN(THETA0) + ...
R0*COS(PHIO)/COS(THETA0)*TAN(THETA0)
*
WRITE(10,200)((A(I,J),J=1,12),I=1,12)
*
*
*
*
*
CALCULATE THE B MATRIX
*
*
*
*
*
B(1,1) = RHO/2*L**3*XRDR*U0*R0+RHO/2*L**2*(XRDR*U0*V0+U0**2*...
2*XDRDR*DR)
B(1,2) = U0*Q0*XQDB/2 + U0*W0*XWDB/2 + U0**2*XDBDB*DB
B(1,3) = U0*Q0*XQDB/2 + U0*W0*XWDB/2 + U0**2*XDBDB*DB
B(1,4) = U0*Q0*XQDS + U0*W0*XWDS +U0**2*2*XDSDS*DS+RHO/2*L**3*...
XQDSN*U0*Q0*EPS + RHO/2*L**2*(XWDSN*U0*W0 + 2*XDSDSN*...
U0**2*DS)*EPS
B(1,5) = RHO/2*L**2*0.12*CD0*0.12*RPM
B(1,6) = SIN(THETA0)
*
B(2,1) = RHO/2*L**2*YDR*U0**2
B(2,6) = -COS(THETA0)*SIN(PHIO)
*
B(3,2) = U0**2*ZDB/2*RHO/2*L**2
B(3,3) = U0**2*ZDB/2*RHO/2*L**2
B(3,4) = U0**2*ZDS*RHO/2*L**2 + RHO/2*L**2*ZDSN*U0**2*EPS
B(3,6) = -COS(THETA0)*COS(PHIO)
*
B(4,2) = -RHO/2*L**3*U0**2*KDB/2
B(4,3) = RHO/2*L**3*U0**2*KDB/2
B(4,6) = -YB*COS(THETA0)*COS(PHIO) + ZB*COS(THETA0)*SIN(PHIO)
*
B(5,2) = RHO/2*L**3*U0**2*MDB/2
B(5,3) = RHO/2*L**3*U0**2*MDB/2
B(5,4) = RHO/2*L**3*(U0**2*MDS+MDSN*U0**2*EPS)
B(5,6) = XB*COS(THETA0)*COS(PHIO) + ZB*SIN(THETA0)
*
B(6,1) = RHO/2*L**3*NDR*U0**2
B(6,6) = -XB*COS(THETA0)*SIN(PHIO) - YB*SIN(THETA0)
*
*
*
FORMULATE THE A AND B MATRIX FOR STATE SPACE REPRESENTATION
*
*
*
MULTIPLY MMINV AND DF/DX
*
*
DO 80 I = 1,6
DO 70 J = 1,6
SUM = 0.0
DO 60 K = 1,6
SUM = SUM + MMINV(I,K)*A(K,J)
60 CONTINUE
AA(I,J) = SUM
70 CONTINUE
80 CONTINUE
*

```

```

*
* MULTIPLY MMINV AND DF/DZ
*
DO 50 I = 1,6
  DO 40 J = 7,12
    SUM = 0.0
    DO 30 K = 1,6
      SUM = SUM + MMINV(I,K)*A(K,J)
30    CONTINUE
    AA(I,J) = SUM
40  CONTINUE
50  CONTINUE
*
*
DO 5 I = 7,12
  DO 6 J = 1,12
    AA(I,J) = A(I,J)
6    CONTINUE
5    CONTINUE
*
*
WRITE(10,200)((AA(I,J),J=1,12),I=1,12)
200 FORMAT( 6E12.4)
*
*
* MULTIPLY MMINV AND DF/DU
*
DO 110 I = 1,6
  DO 100 J = 1,6
    SUM = 0.0
    DO 90 K = 1,6
      SUM = SUM + MMINV(I,K)*B(K,J)
90    CONTINUE
    BB(I,J) = SUM
100  CONTINUE
110  CONTINUE
*
*
WRITE( 9,300)((BB(I,J),J=1,6),I=1,6)
300 FORMAT(6E12.4)
*
*
DO 405 I = 1,6
  READ (2,401)(GKK(I,J), J=1,21)
405  WRITE(3,401)(GKK(I,J), J=1,21)
401  FORMAT(3E20.10)
*
*
ELSE
  END IF
*
* CALL ERRSET (209,256,-1,1,1)
*
* PRINT*,INFO,ISTRAT,IOPT,IONED,IPRINT,IGRAD,NDV,NCON,DX,...
* OBJ,GW,IDG,NGT,IC,DF,AW,NRA,NCOLA,WK,NRWK,...
* IWK,NRIWK
* CALL DADS(INFO,ISTRAT,IOPT,IONED,IPRINT,IGRAD,NDV,NCON,DX,...
*          VLB,VUB,OBJ,GW,IDG,NGT,IC,DF,AW,NRA,NCOLA,WK,NRWK,...
*          IWK,NRIWK)
* IF (INFO.EQ. 0) DELPRT=0.2
*
* DERIVATIVE
* NOSORT
*
*

```

```

*****LINEAR MODEL*****
*
*
*   CALCULATE BB*U PART OF XDOT = AA*X + BB*U
*
      DO 10 J = 1,6
        SUM = 0.0
        DO 15 K = 1,6
          SUM = SUM + BB(J,K)*UMOD(K)
15      CONTINUE
        XDOTU(J) = SUM
10      CONTINUE
*   CALCULATE AA*X
      DO 21 J = 1,12
        SUM = 0.0
        DO 25 K = 1,12
          SUM = SUM + AA(J,K)*X(K)
25      CONTINUE
        XDOTX(J) = SUM
21      CONTINUE
*   CALCULATE XDOT = AA*X + BB*U
      DO 31 J = 1,6
        XDOT(J) = XDOTX(J) + XDOTU(J)
31      CONTINUE
      DO 35 J = 7,12
        XDOT(J) = XDOTX(J)
35      CONTINUE
*
      UDOTM = XDOT(1)
      VDOTM = XDOT(2)
      WDOTM = XDOT(3)
      PDOTM = XDOT(4)

      QDOTM = XDOT(5)
      RDOTM = XDOT(6)
      XDOTM = XDOT(7)
      YDOTM = XDOT(8)
      ZDOTM = XDOT(9)
      PHMDOT = XDOT(10)
      THETMD = XDOT(11)
      PSMDOT = XDOT(12)
*   WRITE(8,600)
*   INTEGRATE XDOT TO GET THE STATE VECTOR X
*
      UM = INTGRL(6.0, UDOTM)
      VM = INTGRL(0.0, VDOTM)
      WM = INTGRL(0.0, WDOTM)
      PM = INTGRL(0.0, PDOTM)
      QM = INTGRL(0.0, QDOTM)
      RM = INTGRL(0.0, RDOTM)
      XPOSM = INTGRL(0.0, XDOTM)
      YPOSM = INTGRL(0.0, YDOTM)
      ZPOSM = INTGRL(0.0, ZDOTM)
      PHIM = INTGRL(0.0, PHMDOT)
      THETAM = INTGRL(0.0, THETMD)
      PSIM = INTGRL(0.0, PSMDOT)
*

```

```

*
X(1) = UM
X(2) = VM
X(3) = WM
X(4) = PM
X(5) = QM
X(6) = RM
X(7) = XPOSM
X(8) = YPOSM
X(9) = ZPOSM
X(10) = PHIM
X(11) = THETAM
X(12) = PSIM
*
*
ZDEPTH = ZORD - X(9)
THMANG = X(11)*57.3
UMOD(1)=DRM
UMOD(2)=DBM
*
DBPM= UMOD(3)
UMOD(3)=DBM
UMOD(4)=DSM
UMOD(5)=DRPM
UMOD(6)=DBOY
*
PHANG=PHIM/0.0174532925
THMANG=THETAM/0.0174532925
PSMANG= PSIM/ 0.0174532925
PITCHM=THMANG
DEPTH=-ZPOSM

*
*****CONTROL LAW*****
*
*
*
DBS = UMOD(2)
*
DBP = UMOD(3)
*
DS = UMOD(4)
*
DR = UMOD(1)
*
*
*
PUT IN STERN AND BOW PLANE STOPS
*
*
IF(ABS(DBS).GT.0.6) THEN
*
DBS = 0.6*ABS(DBS)/DBS
*
ENDIF
*
IF(ABS(DBP).GT.0.6) THEN
*
DBP = 0.6*ABS(DBP)/DBP
*

```

```

*      ENDIF
*      IF(ABS(DS).GT.0.6) THEN
*          DS = 0.6*ABS(DS)/DS
*      ENDIF
*
INTGRD = (UM*UM+VM*VM+WM*WM+PM*PM+QM*QM+RM*RM+...
          XPOSM*XPOSM+(YPOSM-YORD)*(YPOSM-YORD)...
          +(ZPOSM-ORDDEP)*(ZPOSM-ORDDEP)+PHIM*PHIM+...
          THETAM*THETAM+PSIM*PSIM) + (DSM*DSM+DBM*DBM+DRM*DRM)
OBJ1 = INTGRL(0.,(0.5)*INTGRD)
OBJ = OBJ1

DYNAMIC
RN=TIME/(FINTIM/10.-DELT/10000.)
PN=TIME/(FINTIM/20.-DELT/10000.)
O=INT(RN)+1
P=INT(PN)+1
IF(O.GE.11) O=10
IF(P.GE.20) P=20
*
* ADDITIONALLY THE PLANES SHOULD BE AT EQUILIBRIUM SO THE
* VEHICLE WILL PROCEED AT THIS NEW DEPTH WITHIN SOME TOLERANCE
*
          DSM=DX(0)
          DBM=DX(10+0)
          IF(O.GE.10) DSM=0.000
          IF(O.GE.10) DBM=0.000
          DRM=DX(20+P)
          RPM=DX(30+0)
*
*
*          CONSTRAINTS FOR A DIVE
*
* ORDERED DEPTH = ORDDEP
          GW(1) = (ZPOSM-ORDDEP)*.5
          GW(2) = (ORDDEP-ZPOSM)*.5
* AUV'S FINAL STATE MUST BE LEVEL FLIGHT AS FOLLOWS
          GW(3) = THETAM
          GW(4) = -THETAM
          GW(5) = PHIM
          GW(6) = -PHIM
          GW(7) = PSIM
          GW(8) = -PSIM
          GW(9)=(YPOSM-YORD)/.4
          GW(10)=(YORD-YPOSM)/.4
          GW(11) = -ZPOSM
*
* AVOIDING THE OBSTACLE
*
          DIST1=SQRT((XPOSM-XOBS1)*(XPOSM-XOBS1)+...
                   (ZPOSM-ZOBS1)*(ZPOSM-ZOBS1))
*          IF (DIST1.LT.DSAVE1) DSAVE1=DIST1
*          GW(6) = (1.-DSAVE1)
          NDX=XPOSM/17.425
          NDZ=ZPOSM/17.425
          NDT=TIME*6./17.425
*
* TERMINAL
* WRITE(11,66) XPOSM,YPOSM,DEPTH
*66  FORMAT (1X,F10.3,F10.3)
          IF(INFO.EQ.0) THEN
              PRINT*,O,P
          9000 CONTINUE
          ELSE
              ENDIF
          IF(INFO.EQ.0) CALL ENDJOB
          CALL RERUN
*
* END
* STOP

```

FILE: TNLO DSL A1

TITLE RUN:16-5 NONLINEAR AUV MODEL / STERN PLANE AND BOW PLANE SEPARATED

```
* (1) UPDATED:05/20/88
* (2) RIGHT OBJ EQUATION
* (3) ADS CONSTRAINTS ON DEPTH,PITCH,YAW,ROLL AND Y POSITION
* (4) CORRECTED OBSTACLE AVOIDANCE ROUTINE
*****ADSL SET-UP*****
FIXED ISTRAT, IOPT, IONED, IPRINT, INFO, IGRAD, NDV, NCON
FIXED IDG, NGT, IC, NRA, NCOLA, NRWK, IWK, NRIWK, O, H,D,C,PP
D DIMENSION AW(42,42)
ARRAY WK(5000), IWK(1000)
ARRAY DX(40), VLB(40), VUB(40), GW(11), DF(41), IDG(11), IC(11)
PARAM NRA=42, NCOLA=42, NRWK=5000, NRIWK=1000
PARAM IGRAD=0, INFO=0, NDV=40, NCON=11, NGT=11
TABLE DX(1-2)=2*.0,DX(3-40)=38*.0., IDG(1-10)=10*-1
TABLE IDG(11)=1*1
TABLE VLB(1-09)=09*-.17452, VLB(11-19)=09*-.2443,VLB(20)=0.,VLB(10)=0.
TABLE VUB(1-09)=09*.17452, VUB(11-19)=09*.2443,VUB(20)=0.,VUB(10)=0.
TABLE VLB(21-39)=19*-.62367,VUB(21-39)=19*.623627,VUB(40-41)=2*0.
TABLE VLB(40-41)=2*0.
PARAM ISTRAT=3, IOPT=1, IONED=1, IPRINT=0000
INCON H=0, OBS1=0.,YZONE=0.
METHOD RECT
CONTROL FINTIM=21.0,DELT=.10
PRINT XPOS,YPOS,ZPOS,PITCH,THEANG,DR,DS
*PRINT THETAD,W,DEPTH,PITCH,XPOS,DEPTH,NDX,NDZ,NDT
*PRINT DS,DB,DR,DEPTH,PITCH,XPOS,YPOS,ZPOS,NDT
*AVE THETA,W,Z,DEPTH,PITCH,DS,DB,BOWANG,STNANG
*RAPH(DE=TEK618) TIME,DS
*RAPH(DE=TEK618) TIME,DEPTH
*RAPH(DE=TEK618) TIME,WDOT
*RAPH(DE=TEK618) TIME,W
*RAPH(DE=TEK618) TIME,THETDD
*RAPH(DE=TEK618) TIME,THETAD
*RAPH(DE=TEK618) TIME,THETA
*RAPH(DE=TEK618) TIME,PITCH
*RAPH(DE=TEK618) TIME,BOWANG
*RAPH(DE=TEK618) TIME,STNANG
*****ADSL MODEL SET UP*****
*
D DIMENSION MM(6,6),G4(4),GK4(4),BR(4),HH(4)
D DIMENSION B(6,6),BB(6,6)
D DIMENSION A(12,12), AA(12,12)
D COMMON / BLOCK1 / F(12), FP(6), MMINV(6,6), UCF(4)
FIXED N,IA,IDGT,IER,LAST,J,K,M,JJ,KK,I
INTEGER
ARRAY WKAREA(54), X(12)
*
*
CONST
*
* LONGITUDINAL HYDRODYNAMIC COEFFICIENTS
*
CONST XPP = ,XQQ = ,XRR = ,XPR = ,...
XUDOT= ,XWQ = ,XVP = ,XVR = ,...
XQDS= ,XQDB= ,XRDR= ,XVV = ,...
XWW = ,XVDR= ,XWDS= ,XWDB= ,...
XSDS= ,XDBDB= ,XDRDR= ,XQDSN= ,...
XWDSN= ,XSDSN=
*

```

```

*      LATERAL HYDRODYNAMIC COEFFICIENTS
*
CONST YPDOT=          ,YRDOT=          ,YPQ =          ,YQR =          ,...
      YVDOT=          ,YP =           ,YR =           ,YVQ =          ,...
      YWP =           ,YWR =           ,YV =           ,YVW =          ,...
      YDR =           ,CDY =

*
*      NORMAL HYDRODYNAMIC COEFFICIENTS
*
CONST ZQDOT=          ,ZPP =           ,ZPR =           ,ZRR =          ,...
      ZWDOT=          ,ZQ =           ,ZVP =           ,ZVR =          ,...
      ZW =            ,ZVV =           ,ZDS =           ,ZDB =          ,...
      ZQN =           ,ZWN =           ,ZDSN=          ,CDZ =

*
*      ROLL HYDRODYNAMIC COEFFICIENTS
*
CONST KPDOT=          , KRDOT=          ,KPQ =          ,KQR =          ,...
      KVDOT=          , KP =           ,KR =           ,KVQ =          ,...
      KWP =           , KWR =           ,KV =           ,KVV =          ,...
      KPN =           , KDB =

*
*      PITCH HYDRODYNAMIC COEFFICIENTS
*
CONST MQDOT=          , MPP =           ,MPR =           ,MRR =          ,...
      MHDDOT=         , MQ =           ,MVP =           ,MVR =          ,...
      MW =            , MVV =           ,MDS =           ,MDB =          ,...
      MQN =           , MWN =           ,MDSN =

*
*      YAW HYDRODYNAMIC COEFFICIENTS
*
CONST NPDOT=          , NRDOT=          ,NPQ =          ,NQR =          ,...
      NVDOT=          , NP =           ,NR =           ,NVQ =          ,...
      NWP =           , NWR =           ,NV =           ,NVW =          ,...
      NDR =

*
*      MASS CHARACTERISTICS OF THE FLOODED MARK IX VEHICLE
*
CONST WEIGHT =        , BOY =           ,VOL =           ,XG =           ,...
      YG =            , ZG =           ,XB =           ,ZB =           ,...
      IX =            , IY =           ,IZ =           ,IXZ =          ,...
      IYZ =           , IXY =           ,YB =           ,...
      L =             , RHO =          ,G =            ,NU =           ,...
      A0 =            ,KPROP =         ,NPROP =         ,X1TEST=        ,...
      DEGRUD= 0.0    ,DEGSTN= 0.0

*
*ONST XOBS1=36.0
*ONST ZOBS1=-12.0
*
*      INPUT INITIAL CONDITIONS HERE IF REQUIRED
*

```

```

INITIAL
* DSAVE1=SQRT((XPOS-XOBS1)*(XPOS-XOBS1)+(ZPOS-ZOBS1)*(ZPOS-ZOBS1))
NOSORT
*   ORDDEP = 17.425
*   YORD=40.0
*   D=0
*   H=H+1
*   IF(H.EQ.1) THEN
*     U = 0.0
*     V = 0.0
*     W = 0.0
*     P = 0.0
*     Q = 0.0
*     R = 0.0
*     XPOS = 0.0
*     YPOS = 0.0
*     ZPOS = 0.0
*     PSI = 0.0
*     THETA = 0.0
*     PHI = 0.0
*
*     U0 = 6.0
*     V0 = 0.0
*     W0 = 0.0
*     P0 = 0.0
*     Q0 = 0.0
*     R0 = 0.0
*     PHI0 = 0.0
*     THETA0 = 0.0
*     PSIO = 0.0
*     DB = 0.0
*     DS = 0.0
*     DR = 0.0
*     RPM = 500
*
*     LATYAW = 0.0
*     NORPIT = 0.0
*     RE = U0*L/NU
*     CDO = .00385 + (1.296E-17)*(RE - 1.2E7)**2
*
*     DEFINE LENGTH FRACTIONS FOR GAUSS QUADUTURE TERMS
*
*     G4(1) = 0.069431844
*     G4(2) = 0.330009478
*     G4(3) = 0.669990521
*     G4(4) = 0.930568155
*
*     DEFINE WEIGHT FRACTIONS FOR GAUSS QUADUTURE TERMS
*
*     GK4(1) = 0.1739274225687
*     GK4(2) = 0.3260725774312
*     GK4(3) = 0.3260725774312
*     GK4(4) = 0.1739274225687
*
*     DEFINE THE BREADTH BB AND HEIGHT HH TERMS FOR THE INTEGRATION
*
*     BR(1) = 75.7/12
*     BR(2) = 75.7/12
*     BR(3) = 75.7/12
*     BR(4) = 55.08/12
*

```



```

HH(1) = 16.38/12
HH(2) = 31.85/12
HH(3) = 31.85/12
HH(4) = 23.76/12
*
MASS = WEIGHT/G
*
DIVAMP = DEGSTN*0.0174532925
RUDAMP = DEGRUD*0.0174532925
*
N = 6
DO 15 J = 1,N
  DO 10 K = 1,N
    MMINV(J,K) = 0.0
    MM(J,K) = 0.0
  CONTINUE
10 CONTINUE
15 CONTINUE
*
MM(1,1) = MASS - ((RHO/2)*(L**3)*XUDOT)
MM(1,5) = MASS*ZG
MM(1,6) = -MASS*YG
*
MM(2,2) = MASS - ((RHO/2)*(L**3)*YVDOT)
MM(2,4) = -MASS*ZG - ((RHO/2)*(L**4)*YPDOT)
MM(2,6) = MASS*XG - ((RHO/2)*(L**4)*YRDOT)
*
MM(3,3) = MASS - ((RHO/2)*(L**3)*ZWDOT)
MM(3,4) = MASS*YG
MM(3,5) = -MASS*XG - ((RHO/2)*(L**4)*ZQDOT)
*
MM(4,2) = -MASS*ZG - ((RHO/2)*(L**4)*KVDOT)
MM(4,3) = MASS*YG
MM(4,4) = IX - ((RHO/2)*(L**5)*KPDOT)
MM(4,5) = -IXY
MM(4,6) = -IXZ - ((RHO/2)*(L**5)*KRDOT)
*
MM(5,1) = MASS*ZG
MM(5,3) = -MASS*XG - ((RHO/2)*(L**4)*MWDOT)
MM(5,4) = -IXY
MM(5,5) = IY - ((RHO/2)*(L**5)*MQDOT)
*
MM(5,6) = -IYZ
*
MM(6,1) = -MASS*YG
MM(6,2) = MASS*XG - ((RHO/2)*(L**4)*NVDOT)
MM(6,4) = -IXZ - ((RHO/2)*(L**5)*NPDOT)
MM(6,5) = -IYZ
MM(6,6) = IZ - ((RHO/2)*(L**5)*NRDOT)
*

```

```

*
* LAST = N*N+3*N
* DO 20 M = 1, LAST
* WKAREA(M) = 0.0
20 * CONTINUE
*
* IER = 0
* IA = 6
* IDGT = 4
* WRITE( 8,400)((MM(I,J), J = 1,6), I = 1,6)
*
* CALL LINV2F(MM,N,IA,MMINV,IDGT,WKAREA,IER)
*
* WRITE( 8,400)((MMINV(I,J), J = 1,6), I = 1,6)
400 * FORMAT(6E12.4)
*
* ELSE
* ENDIF
*
*
* CALL DADS(INFO,ISTRAT,IOPT,IONED,IPRINT,IGRAD,NDV,NCON,DX,...
* VLB,VUB,OBJ,GW,IDG,NGT,IC,DF,AW,NRA,NCOLA,WK,NRWK,...
* IWK,NRIWK)
C IF(H.EQ.1) THEN
C WK(12) = .002
C CALL DADS(INFO,ISTRAT,IOPT,IONED,IPRINT,IGRAD,NDV,NCON,DX,...
C VLB,VUB,OBJ,GW,IDG,NGT,IC,DF,AW,NRA,NCOLA,WK,NRWK,...
C IWK,NRIWK)
* IF(INFO.EQ.0) DELPRT = 0.2
* IF(INFO.EQ.0) DELPLT = 0.2
*
*
*
* DERIVATIVE
* NOSORT
*
*
* PROPULSION MODEL
*
*
* SIGNU = 1.0
* IF (U.LT.0.0) SIGNU = -1.0
* IF (ABS(U).LT.XITEST) U = XITEST
* SIGNN = 1.0
* IF (RPM.LT.0.0) SIGNN = -1.0
* ETA = 0.012*RPM/U
* RE = U*L/NU
* CDO = .00385 + (1.296E-17)*(RE - 1.2E7)**2
* CT = 0.008*L**2*ETA*ABS(ETA)/(A0)
* CT1 = 0.008*L**2/(A0)
* EPS = -1.0+SIGNN/SIGNU*(SQRT(CT+1.0)-1.0)/(SQRT(CT1+1.0)-1.0)
* XPROP = CDO*(ETA*ABS(ETA) - 1.0)
*
*
*
* CALCULATE THE DRAG FORCE, INTEGRATE THE DRAG OVER THE VEHICLE
* INTEGRATE USING A 4 TERM GAUSS QUADATURE
*
*
* LATYAW = 0.0
* NORPIT = 0.0
* DO 500 K = 1,4
* UCF(K) = SQRT((V+G4(K)*R*L)**2 + (W-G4(K)*Q*L)**2)
* IF(UCF(K).GT.1E-10) THEN

```

FILE: TNLO DSL AI

TERM0 = (RHO/2)\*(CDY\*HH(K))\*(V+G4(K)\*R\*L)\*\*2 + ...  
CDZ\*BR(K)\*(W-G4(K)\*Q\*L)\*\*2)  
TERM1 = TERM0\*(V+G4(K)\*R\*L)/UCF(K)  
TERM2 = TERM0\*(W-G4(K)\*Q\*L)/UCF(K)  
LATYAW = LATYAW + TERM1\*GK4(K)\*L  
NORPIT = NORPIT + TERM2\*GK4(K)\*L  
END IF

500 CONTINUE

\*  
\*  
\*  
\*  
\*  
\*  
\*

FORCE EQUATIONS

LONGITUDINAL FORCE

FP(1) = MASS\*V\*R - MASS\*W\*Q + MASS\*XG\*Q\*\*2 + MASS\*XG\*R\*\*2 - ...  
MASS\*YG\*P\*Q - MASS\*ZG\*P\*R + (RHO/2)\*L\*\*4\*(XPP\*P\*\*2 + ...  
XQQ\*Q\*\*2 + XRR\*R\*\*2 + XPR\*P\*R) + (RHO/2)\*L\*\*3\*(XWQ\*W\*Q + ...  
XVP\*V\*P + XVR\*V\*R + U\*Q\*(XQDS\*DS + XQDB\*DB) + XRDR\*U\*R\*DR) + ...  
(RHO/2)\*L\*\*2\*(XVV\*V\*\*2 + XWWW\*\*2 + XVDR\*U\*V\*DR + U\*W\*...  
(XWDS\*DS + XWDB\*DB) + U\*\*2\*(XDS\*DS\*\*2 + XDB\*DB\*\*2 + ...  
XDR\*DR\*\*2)) - (WEIGHT - BOY)\*SIN(THETA) + (RHO/2)\*L\*\*3\* ...  
XQDSN\*U\*Q\*DS\*EPS + (RHO/2)\*L\*\*2\*(XWDSN\*U\*W\*DS + XDS\*DSN\*U\*\*2\*...  
DS\*\*2)\*EPS + (RHO/2)\*L\*\*2\*U\*\*2\*XPROP

\*  
\*  
\*

LATERAL FORCE

FP(2) = -MASS\*U\*R - MASS\*XG\*P\*Q + MASS\*YG\*R\*\*2 - MASS\*ZG\*Q\*R + ...  
(RHO/2)\*L\*\*4\*(YPQ\*P\*Q + YQR\*Q\*R) + (RHO/2)\*L\*\*3\*(YP\*U\*P + ...  
YR\*U\*R + YVQ\*V\*Q + YWP\*W\*P + YWR\*W\*R) + (RHO/2)\*L\*\*2\* ...  
(YV\*U\*V + YVW\*V\*W + YDR\*U\*\*2\*DR) - LATYAW + (WEIGHT - BOY)\*...  
COS(THETA)\*SIN(PHI)

\*  
\*  
\*

NORMAL FORCE

FP(3) = MASS\*U\*Q - MASS\*V\*P - MASS\*XG\*P\*R - MASS\*YG\*Q\*R + ...  
MASS\*ZG\*P\*\*2 + MASS\*ZG\*Q\*\*2 + (RHO/2)\*L\*\*4\*(ZPP\*P\*\*2 + ...  
ZPR\*P\*R + ZRR\*R\*\*2) + (RHO/2)\*L\*\*3\*(ZQ\*U\*Q + ZVP\*V\*P + ...  
ZVR\*V\*R) + (RHO/2)\*L\*\*2\*(ZW\*U\*W + ZVV\*V\*\*2 + U\*\*2\*(ZDS\*...  
DS + ZDB\*DB)) - NORPIT + (WEIGHT - BOY)\*COS(THETA)\*COS(PHI) + ...  
(RHO/2)\*L\*\*3\*ZQN\*U\*Q\*EPS + (RHO/2)\*L\*\*2\*(ZWN\*U\*W + ZDSN\*...  
U\*\*2\*DS)\*EPS

\*  
\*  
\*

ROLL FORCE

FP(4) = -IZ\*Q\*R + IY\*Q\*R - IX\*P\*R + IYZ\*Q\*\*2 - IYZ\*R\*\*2 + IXZ\*P\*Q + ...  
MASS\*YG\*U\*Q - MASS\*YG\*V\*P - MASS\*ZG\*W\*P + (RHO/2)\*L\*\*5\*(KPQ\*...  
P\*Q + KQR\*Q\*R) + (RHO/2)\*L\*\*4\*(KP\*U\*P + KR\*U\*R + KVQ\*V\*Q + ...  
KWP\*W\*P + KWR\*W\*R) + (RHO/2)\*L\*\*3\*(KV\*U\*V + KVW\*V\*W) + ...  
(YG\*WEIGHT - YB\*BOY)\*COS(THETA)\*COS(PHI) - (ZG\*WEIGHT - ...  
ZB\*BOY)\*COS(THETA)\*SIN(PHI) + (RHO/2)\*L\*\*4\*KPN\*U\*P\*EPS + ...  
(RHO/2)\*L\*\*3\*U\*\*2\*KPROP + MASS\*ZG\*U\*R

\*

```

* PITCH FORCE
*
FP(5) = -IX*P*R + IZ*P*R + IXY*Q*R - IYZ*P*Q - IXZ*P**2 + IXZ*R**2 - ...
        MASS*XG*U*Q + MASS*XG*V*P + MASS*ZG*V*R - MASS*ZG*W*Q + ...
        (RHO/2)*L**5*(MPP*P**2 + MPR*P*R + MRR*R**2) + (RHO/2)*L**4*...
        (MQ*U*Q + MVP*V*P + MVR*V*R) + (RHO/2)*L**3*(MW*U*W + ...
        MVV*V**2 + U**2*(MDS*DS + MDB*DB)) + NORPIT - (XG*WEIGHT - ...
        XB*BOY)*COS(THETA)*COS(PHI) + ...
        (RHO/2)*L**4*MQN*U*Q*EPS + ...
        (RHO/2)*L**3*(MWN*U*W + MDSN*U**2*DS)*EPS - ...
        (ZG*WEIGHT - ZB*BOY)*SIN(THETA)
*
* YAW FORCE
*
FP(6) = -IY*P*Q + IX*P*Q + IXY*P**2 - IXY*Q**2 + IYZ*P*R - IXZ*Q*R - ...
        MASS*XG*U*R + MASS*XG*W*P - MASS*YG*V*R + MASS*YG*W*Q + ...
        (RHO/2)*L**5*(NPQ*P*Q + NQR*Q*R) + (RHO/2)*L**4*(NP*U*P + ...
        NR*U*R + NVQ*V*Q + NWP*W*P + NWR*W*R) + (RHO/2)*L**3*(NV*...
        U*V + NVW*V*W + NDR*U**2*DR) - LATYAW + (XG*WEIGHT - ...
        XB*BOY)*COS(THETA)*SIN(PHI) + (YG*WEIGHT)*SIN(THETA)...
        + (RHO/2)*L**3*U**2*NPROP - YB*BOY*SIN(THETA)
*
* IF(Z.EQ.50.0)THEN
*   WRITE (8,500)(FP(I), I = 1,6)
*   Z = 0.0
* END IF
*
* NOW COMPUTE THE F(1-6) FUNCTIONS
*
DO 600 J = 1,6
  F(J) = 0.0
DO 600 K = 1,6
  F(J) = MMINV(J,K)*FP(K) + F(J)
600 CONTINUE
*
* THE LAST SIX EQUATIONS COME FROM THE KINEMATIC RELATIONS
*
* FIRST SET THE DRIFT CURRENT VALUES
*
UCO = 0.0
VCO = 0.0
WCO = 0.0
*
* INERTIAL POSITION RATES F(7-9)
*
F(7) = UCO + U*COS(PSI)*COS(THETA) + V*(COS(PSI)*SIN(THETA)*...
        SIN(PHI) - SIN(PSI)*COS(PHI)) + W*(COS(PSI)*SIN(THETA)*...
        COS(PHI) + SIN(PSI)*SIN(PHI))
*
F(8) = VCO + U*SIN(PSI)*COS(THETA) + V*(SIN(PSI)*SIN(THETA)*...
        SIN(PHI) + COS(PSI)*COS(PHI)) + W*(SIN(PSI)*SIN(THETA)*...
        COS(PHI) - COS(PSI)*SIN(PHI))
*
F(9) = WCO - U*SIN(THETA) + V*COS(THETA)*SIN(PHI) + W*COS(THETA)*...
        COS(PHI)
*
* EULER ANGLE RATES F(10-12)
*
F(10) = P + Q*SIN(PHI)*TAN(THETA) + R*COS(PHI)*TAN(THETA)
*
F(11) = Q*COS(PHI) - R*SIN(PHI)
*
F(12) = Q*SIN(PHI)/COS(THETA) + R*COS(PHI)/COS(THETA)

```

```

*
*
*00 IF (Z.EQ.1.0)WRITE (9,500)(F(I), I = 1,12)
*   FORMAT(6E12.4)
*   Z = Z + 1
*
*
*   UDOT = F(1)
*   VDOT = F(2)
*   WDOT = F(3)
*   PDOT = F(4)
*   QDOT = F(5)
*   RDOT = F(6)
*   XDOT = F(7)
*   YDOT = F(8)
*   ZDOT = F(9)
*   PHIDOT = F(10)
*   THETAD = F(11)
*   PSIDOT = F(12)
*
*   U = INTGRL (U0,UDOT)
*   X(1) = U
*   V = INTGRL(0.0,VDOT)
*   X(2) = V
*   W = INTGRL(0.0,WDOT)
*   X(3) = W
*   P = INTGRL(0.0,PDOT)
*
*   X(4) = P
*   Q = INTGRL(0.0,QDOT)
*   X(5) = Q
*   R = INTGRL(0.0,RDOT)
*   X(6) = R
*   XPOS = INTGRL(0.0,XDOT)
*   X(7) = XPOS
*   YPOS = INTGRL(0.0,YDOT)
*   X(8) = YPOS
*   Z = INTGRL(0.0,ZDOT)
*   X(9) = ZPOS
*   PHI = INTGRL(0.0,PHIDOT)
*   X(10) = PHI
*   THETA = INTGRL(0.0,THETAD)
*   X(11) = THETA
*   PSI = INTGRL(0.0,PSIDOT)
*   X(12) = PSI
*
*   PHIANG = PHI/0.0174532925
*   THEANG = THETA/0.0174532925
*   PSIANGL = PSI/0.0174532925
*   ZPOS=-Z
*
*   DEPTH=ZPOS
*   PITCH=THEANG
*   BOWANG=(DB/.01745)
*   STNANG=(DS/.01745)
*   INTGRD = (U*XU+V*V+W*W+P*P+Q*Q+R*R+XPOS*XPOS+(YPOS-YORD)*...
*             (YPOS-YORD)+(Z-ORDDEP)*(Z-ORDDEP)+PHI*PHI+...
*             THETA*THETA+PSI*PSI) + (DS*DS+DB*DB)+(DR*DR)
*   OBJ1 = INTGRL(0.,(0.5)*INTGRD)
*   OBJ = OBJ1

```

```

*
DYNAMIC
  RN=TIME/(FINTIM/10.-DELT/10000.)
  PN=TIME/(FINTIM/20.-DELT/10000.)
  O=INT(RN)+1
  PP=INT(PN)+1
  IF(O.GE.10) O=10
  IF(PP.GE.20) PP=20
*
* ADDITIONALLY THE PLANES SHOULD BE AT EQUILIBRIUM SO THE
* VEHICLE WILL PROCEED AT THIS NEW DEPTH WITHIN SOME TOLERANCE
*
  DS=DX(0)
  DB=DX(10+0)
  IF(O.GE.10) DS=0.
  IF(O.GE.10) DB=0.
  DR=DX(20+PP)
  RPM=DX(30+0)
*
*
*          CONSTRAINTS FOR A DIVE
*
* ORDERED DEPTH = ORDDEP
  GW(1) = (Z-ORDDEP)*.5
  GW(2) = (ORDDEP-Z)*.5
* AUV'S FINAL STATE MUST BE LEVEL FLIGHT AS FOLLOWS
  GW(3) = THETA
  GW(4) = -THETA
  GW(5) = (YPOS-YORD)/.4
  GW(6) = (YORD-YPOS)/.4
  GW(7) = PSI
  GW(8) = -PSI
  GW(9) = PHI
  GW(10) = -PHI
  GW(11) = ZPOS
*
* AVOIDING THE OBSTACLE
*
*
* DIST1=SQRT((XPOS-XOBS1)*(XPOS-XOBS1)+(ZPOS-ZOBS1)*(ZPOS-ZOBS1))
* IF (DIST1.LT.DSAVE1) DSAVE1=DIST1
* GW(6) = (1.-DSAVE1)
*
  NDX=XPOS/17.425
  NDZ=ZPOS/17.425
  NDT=TIME*6./17.425
*
* TERMINAL
  IF(INFO.EQ.0) THEN
* PRINT*,DSAVE1
*9999 FORMAT(1X,E15.4)
  9000 CONTINUE
  ELSE
  ENDIF
  IF(INFO.EQ.0) CALL ENDJOB
  CALL RERUN
*
*
* END
* STOP

```

FILE: TX DSL A1

```
TITLE LINEAR AUV MODEL / STERN PLANE AND BOW PLANE SEPARATED
TITLE WITH COMMANDS TO NONLINEAR MODEL
* (1) UPDATED:05/30/88
* (3) RIGHT OBJ EQUATION
* (4) ADS CONSTRAINTS ON DEPTH AND PITCH
* (5) OBSTACLE FURTHER DOWN THE TRAJECTORY AND ABOVE IT
* (6) CORRECT OBSTACLE AVOIDANCE ROUTINE ADDED
*****ADSL SET-UP*****
FIXED ISTRAT, IOPT, IONED, IPRINT, INFO, IGRAD, NDV, NCON
FIXED IDG, NGT, IC, NRA, NCOLA, NRWK, IWK, NRIWK, O, H,D,C,PP
D DIMENSION AW(42,42)
ARRAY WK(4000), IWK(1000)
ARRAY DX(40), VLB(40), VUB(40), GW(07), DF(41), IDG(07), IC(07)
PARAM NRA=42, NCOLA=42, NRWK=4000, NRIWK=1000
PARAM IGRAD=0, INFO=0, NDV=40, NCON=07, NGT=07
TABLE DX(1-2)=2*0,DX(3-21)=19*0., IDG(1-6)=6*-1
TABLE DX(22-40)=19*0.
TABLE IDG(7-0)=1*1
TABLE VLB(1-9)=9*-.17452, VLB(11-19)=9*-.2443,VLB(10)=0.,VLB(20)=0.
TABLE VUB(1-9)=9*.17452, VUB(11-19)=9*.2443,VUB(10)=0.,VUB(20)=0.
TABLE VLB(21-39)=19*-.523627,VUB(21-39)=19*.523627,VUB(40-41)=2*0.
TABLE VLB(40-41)=2*0.
PARAM ISTRAT=3, IOPT=1, IONED=1, IPRINT=0000
INCON H=0, OBS1=0.,YZONE=0.
METHOD RECT
CONTROL FINTIM=21., DELT=.1
PRINT XPOS,YPOS,ZPOS,XPOSM,YPOSM,ZPOSM
*PRINT DS,DSM,DBPM,DBP,PITCHM,PITCH,XPOSM,YPOSM,XPOS,YPOS,ZPOSM,ZPOS,NDT
*PRINT YPOSM,YPOS
*PRINT THETAD,W,DEPTH,PITCH,XPOS,DEPTH,NDX,NDZ,NDT
*AVE THETA,W,Z,DEPTH,PITCH,DS,DB,BOWANG,STNANG
*RAPH(DE=TEK618) TIME,DS
*RAPH(DE=TEK618) TIME,DEPTH
*RAPH(DE=TEK618) TIME,WDOT
*RAPH(DE=TEK618) TIME,W
*RAPH(DE=TEK618) TIME,THETDD
*RAPH(DE=TEK618) TIME,THETAD
*RAPH(DE=TEK618) TIME,THETA
*RAPH(DE=TEK618) TIME,PITCH
*RAPH(DE=TEK618) TIME,BOWANG
*RAPH(DE=TEK618) TIME,STNANG
*****D S L MODEL FOR LINEAR SIMULATION *****
*****AND NON-LINEAR SIMULATION VARYING CONTROL LAW*****
* LINEAR MODEL/NON-LINEAR MODEL
D COMMON/BLOCK1/ MMINV(6,6), MM(6,6), AA(12,12), BB(6,6)
D COMMON/BLOCK2/ B(6,6),A(12,12), UMOD(6),GKK(6,21)
D COMMON/BLOCK3/ F(12), FP(6), UCF(4)
D COMMON/BLOCK4/ G4(4),GK4(4),BR(4),HH(4)
D COMMON/BLOCK5/ XDOT(12),XDOTX(12), XDOTU(6)
FIXED H,IA,IDGT,IER,LAST,J,K,M,JJ,KK,I
INTEGER
ARRAY WKAREA(54), X(12)
*
*
*
```

```

CONST
*
* LONGITUDINAL HYDRODYNAMIC COEFFICIENTS
*
CONST XPP = , XQQ = , XRR = , XPR = ,...
XUDOT= , XWQ = , XVP = , XVR = ,...
XQDS= , XQDB= , XRDR= , XVV = ,...
XWN = , XVDR= , XWDS= , XWDB= ,...
XDSDS= , XDBDB= , XDRDR= , XQDSN= ,...
XWDSN= , XDSDSN=

*
* LATERAL HYDRODYNAMIC COEFFICIENTS
*
CONST YPDOT= , YRDOT= , YPQ = , YQR = ,...
YVDOT= , YP = , YR = , YVQ = ,...
YWP = , YWR = , YV = , YVW = ,...
YDR = , CDY =

*
* NORMAL HYDRODYNAMIC COEFFICIENTS
*
CONST ZQDOT= , ZPP = , ZPR = , ZRR = ,...
ZWDOT= , ZQ = , ZVP = , ZVR = ,...
ZN = , ZVV = , ZDS = , ZDB = ,...
ZQN = , ZWN = , ZDSN= , CDZ =

*
* ROLL HYDRODYNAMIC COEFFICIENTS
*
CONST KPDOT= , KRDOT= , KPQ = , KQR = ,...
KVDOT= , KP = , KR = , KVQ= ,...
KWP = , KWR = , KV = , KVV = ,...
KPN = , KDB =

*
* PITCH HYDRODYNAMIC COEFFICIENTS
*
CONST MQDOT= , MPP = , MPR = , MRR = ,...
MWDOT= , MQ = , MVP = , MVR = ,...
MW = , MVV = , MDS = , MDB = ,...
MQN = , MWN = , MDSN =

*
* YAW HYDRODYNAMIC COEFFICIENTS
*
CONST NPDOT= , NRDOT= , NPQ = , NQR = ,...
NVDOT= , NP = , NR = , NVQ = ,...
NWP = , NWR = , NV = , NVW = ,...
NDR =

*
* MASS CHARACTERISTICS OF THE FLOODED MARK IX VEHICLE
*
CONST WEIGHT = , BOY = , VOL = , XG = ,...
YG = , ZG = , XB = , ZB = ,...
IX = , IY = , IZ = , IXZ = ,...
IYZ = , IXY = , YB = ,...
L = , RHO = , G = , NU = ,...
AO = , KPROP = , NPROP = , X1TEST= ,...
DEGRUD= 0.0 , DEGSTN= 0.0

*
CONST XOBS1=36.0
CONST ZOBS1= -12.0

```



```

*      INPUT INITIAL CONDITIONS HERE IF REQUIRED
*
* INITIAL
*
*      DSAVE1=SQRT((XPOSM-XOBS1)*(XPOSM-XOBS1)+...
*              (ZPOSM-ZOBS1)*(ZPOSM-ZOBS1))
*      DSAVEV=DSAVE1
*      INITIALIZE ALL MATRICES AND ARRAYS TO ZERO
NOSORT
ORDDEP=20.0
YORD=40.0
D=0
M=M+1
IF (H.EQ.1) THEN
N = 6
DO 2 J = 1,N
  JJ= J+N
  DO 1 K = 1,N
    KK= K+N
    KKK= KK + N
    MMINV(J,K) = 0.0
    X(J) = 0.0
    X(JJ) = 0.0
    XDOT(J) = 0.0
    XDOT(JJ) = 0.0
    XDOTX(J) = 0.0
    XDOTX(JJ) = 0.0
    XDOTU(J) = 0.0
*
*      UMOD(J) = 0.0
*      MM(J,K) = 0.0
*      BB(J,K) = 0.0
*      B(J,K) = 0.0
*      AA(J,K)= 0.0
*      AA(JJ,KK)= 0.0
*      AA(J, KK)= 0.0
*      AA(JJ,K)= 0.0
*      A(J, KK)= 0.0
*      A(JJ,K) = 0.0
*      A(J,K) = 0.0
*      A(JJ, KK)= 0.0
*      GKK(J,K)= 0.0
*      GKK(J, KK)=0.0
*      GKK(J, KKK)=0.0
1
2
*      CONTINUE
*
*      INPUT THE LINEARIZATION POINT PARAMETERS
*
*      U0 =6.0
*      V0 = 0.0
*      W0 = 0.0
*      P0 = 0.0
*      Q0 = 0.0
*      R0 = 0.0
*      PHIO = 0.0
*      THETA0 = 0.0
*      PSIO = 0.0
*      SUM = 0.0
*      JFLAG = 0
*      IFLAG = 0
*      KFLAG = 0
*

```

\* INPUT THE MODEL STATES INITIAL CONDITIONS  
\*

UM = 6.0  
VM = 0.0  
WM = 0.0  
PM = 0.0  
QM = 0.0  
RM = 0.0  
XPOSM = 0.0  
YPOSM = 0.0  
ZPOSM = 0.0  
PHIM = 0.0  
THETAM = 0.0  
PSIM = 0.0  
U = 6.0  
V = 0.0  
W = 0.0  
P = 0.0  
Q = 0.0  
R = 0.0  
XPOS = 0.0  
YPOS = 0.0  
ZPOS = 0.0  
PHI = 0.0  
THETA = 0.0  
PSI = 0.0

\* INPUT THE VEHICLE INITIAL CONDITIONS  
\*

INITIALIZE THE CONTROLS

DELBOY= 0.0  
DBOY=0.  
DS= 0.0  
DSM=0.0  
DBM=0.0

DB=0.0  
DR= 0.0  
DRM=0.0  
DRPM=0.0  
RPM = 500.0  
LATYAW = 0.0  
NORPIT = 0.0

MASS = WEIGHT/G

DIVAMP = DEGSTN\*0.0174532925  
RUDAMP = DEGRUD\*0.0174532925

DEFINE LENGTH FRACTIONS FOR GAUSS QUADATURE TERMS

G4(1) = 0.069431844  
G4(2) = 0.330009478  
G4(3) = 0.669990521  
G4(4) = 0.930568155

DEFINE WEIGHT FRACTIONS FOR GAUSS QUADATURE TERMS

GK4(1) = 0.1739274225687  
GK4(2) = 0.3260725774312  
GK4(3) = 0.3260725774312  
GK4(4) = 0.1739274225687

```
* DEFINE THE BREADTH BB AND HEIGHT HH TERMS FOR THE INTEGRATION
```

```
*  
BR(1) = 75.7/12  
BR(2) = 75.7/12  
BR(3) = 75.7/12  
BR(4) = 55.08/12
```

```
*  
HH(1) = 16.38/12  
HH(2) = 31.85/12  
HH(3) = 31.85/12  
HH(4) = 23.76/12
```

```
* THE LINEAR PROPULSION MODEL
```

```
*  
ETA = 0.012*RPM/U0  
ETA = 1.0  
RE = U0*L/NU  
CDO = .00385 + (1.296E-17)*(RE - 1.2E7)**2  
CT = 0.008*L**2*ETA*ABS(ETA)/(A0)  
CT1 = 0.008*L**2/(A0)  
EPS = -1.0+(SQRT(CT+1.0)-1.0)/(SQRT(CT1+1.0)-1.0)  
XPROP = CDO*(ETA*ABS(ETA) - 1.0)
```

```
* CALCULATE THE MASS MATRIX
```

```
*  
MM(1,1) = MASS - ((RHO/2)*(L**3)*XUDOT)  
MM(1,5) = MASS*ZG  
MM(1,6) = -MASS*YG
```

```
MM(2,2) = MASS - ((RHO/2)*(L**3)*YVDOT)  
MM(2,4) = -MASS*ZG - ((RHO/2)*(L**4)*YPDOT)  
MM(2,6) = MASS*XG - ((RHO/2)*(L**4)*YRDOT)
```

```
MM(3,3) = MASS - ((RHO/2)*(L**3)*ZWDOT)  
MM(3,4) = MASS*YG  
MM(3,5) = -MASS*XG - ((RHO/2)*(L**4)*ZQDOT)
```

```
MM(4,2) = -MASS*ZG - ((RHO/2)*(L**4)*KVDOT)  
MM(4,3) = MASS*YG  
MM(4,4) = IX - ((RHO/2)*(L**5)*KPDOT)  
MM(4,5) = -IXY  
MM(4,6) = -IXZ - ((RHO/2)*(L**5)*KRDOT)
```

```
MM(5,1) = MASS*ZG  
MM(5,3) = -MASS*XG - ((RHO/2)*(L**4)*MWDOT)  
MM(5,4) = -IXY  
MM(5,5) = IY - ((RHO/2)*(L**5)*MQDOT)  
MM(5,6) = -IYZ
```

```
MM(6,1) = -MASS*YG  
MM(6,2) = MASS*XG - ((RHO/2)*(L**4)*NVDOT)  
MM(6,4) = -IXZ - ((RHO/2)*(L**5)*NPDOT)  
MM(6,5) = -IYZ  
MM(6,6) = IZ - ((RHO/2)*(L**5)*NRDOT)
```

```
*  
*  
LAST = N*N+3*N  
DO 20 M = 1, LAST  
WKAREA(M) = 0.0  
20 CONTINUE
```

```
*  
IER = 0  
IA = 6  
IDGT = 4  
*
```

CALL LINV2F(MM,N,IA,MMINV, IDGT,WKAREA, IER)

\*  
\*  
\*  
\*  
\*

CALCULATE THE A MATRIX FOR THE LINEAR MODEL

A(1,1) = RHO/2\*L\*\*3\*(XQDS\*DS\*Q0+XQDB/2\*DBP\*Q0+XRDR\*R0\*DR)+...  
RHO/2\*L\*\*2\*(XVDR\*V0\*DR+XWDS\*DS\*W0+XWDB/2\*DBP\*W0 + ...  
2\*U0\*(XDS\*DS\*\*2 + XDBDB/2\*DBP\*\*2 + XDRDR\*DR\*\*2))+...  
RHO/2\*L\*\*3\*XQDSN\*Q0\*DS\*EPS+RHO/2\*L\*\*2\*(XWDSN\*W0\*DS+...  
2\*XDS\*DSN\*U0\*DS\*\*2)\*EPS+RHO\*L\*\*2\*U0\*XPROP+RHO/2\*L\*\*3\*...  
XQDB/2\*DBS\*Q0+RHO/2\*L\*\*2\*XWDB/2\*DBS\*W0+RHO\*L\*\*2\*U0\*...  
XDBDB/2\*DBS\*\*2  
A(1,2) = MASS\*R0+RHO/2\*L\*\*3\*(XVP\*P0+ XVR\*R0) + RHO/2\*L\*\*2\*...  
(2\*XVV\*V0 + XVDR\*U0\*DR)  
A(1,3) = -MASS\*Q0 + RHO/2\*L\*\*3\*(XWQ\*Q0)+RHO/2\*L\*\*2\*(2\*XWW\*W0+...  
XWDS\*DS\*U0+(XWDB/2\*DBP+XWDB/2\*DBS)\*U0 +XWDSN\*U0\*DS\*EPS)  
A(1,4) = -MASS\*YG\*Q0-MASS\*ZG\*R0+ RHO/2\*L\*\*4\*(2\*XPP\*P0+XPR\*R0)...  
+ RHO/2\*L\*\*3\*(XVP\*V0)  
A(1,5) = -MASS\*W0+2\*MASS\*XG\*Q0 -MASS\*YG\*P0+RHO/2\*L\*\*4\*2\*XQQ\*Q0...  
+RHO/2\*L\*\*3\*(XWQ\*W0+XQDS\*DS\*U0+XQDB/2\*DBP\*U0)+RHO/2\*...  
L\*\*3\*XQDSN\*U0\*DS\*EPS+RHO/2\*L\*\*3\*XQDB/2\*DBS\*U0  
A(1,6) = MASS\*V0+2\*MASS\*XG\*R0-MASS\*ZG\*P0+RHO/2\*L\*\*4\*(2\*XRR\*R0...  
+ XPR\*P0) + RHO/2\*L\*\*3\*(XVR\*V0 + XRDR\*U0\*DR)  
A(1,11) = -(WEIGHT - BOY)\*COS(THETA0)  
\*  
A(2,1) = -MASS\*R0+RHO/2\*L\*\*3\*(YP\*P0+YR\*R0)+RHO/2\*L\*\*2\*(YV\*V0+...  
2\*YDR\*U0\*DR)  
A(2,2) = RHO/2\*L\*\*3\*YVQ\*Q0+RHO/2\*L\*\*2\*(YV\*U0+YVW\*W0)  
A(2,3) = MASS\*P0+ RHO/2\*L\*\*3\*(YWP\*P0+YWR\*R0)+RHO/2\*L\*\*2\*YVW\*V0  
A(2,4) = MASS\*W0-MASS\*XG\*Q0+2\*MASS\*YG\*P0+RHO/2\*L\*\*4\*YVQ\*Q0+...  
RHO/2\*L\*\*3\*(YV\*U0+ YVW\*W0)  
A(2,5) = -MASS\*XG\*P0-MASS\*ZG\*R0+RHO/2\*L\*\*4\*(YPQ\*P0+YQR\*R0) +...  
RHO/2\*L\*\*3\*YVQ\*V0  
A(2,6) = -MASS\*U0+2\*MASS\*YG\*R0-MASS\*ZG\*Q0+RHO/2\*L\*\*4\*YQR\*Q0 +...  
RHO/2\*L\*\*3\*(YR\*U0 + YWR\*W0)  
A(2,10) = (WEIGHT - BOY)\*COS(THETA0)\*COS(PHI0)  
A(2,11) = -(WEIGHT - BOY)\*SIN(THETA0)\*SIN(PHI0)  
\*  
A(3,1) = MASS\*Q0+RHO/2\*L\*\*3\*ZQ\*Q0+RHO/2\*L\*\*2\*(ZW\*W0+2\*U0\*ZDS\*DS...  
+2\*U0\*ZDB/2\*DBP+(ZWN\*W0+2\*ZDSN\*U0\*DS)\*EPS)+RHO/2\*L\*\*3\*...  
ZQN\*Q0\*EPS+ RHO/2\*L\*\*2\*2\*U0\*ZDB/2\*DBS  
A(3,2) = -MASS\*P0+RHO/2\*L\*\*3\*(ZVP\*P0+ZVR\*R0)+RHO/2\*L\*\*2\*2\*ZVW\*V0  
A(3,3) = RHO/2\*L\*\*2\*(ZW\*U0 + ZWN\*U0\*EPS)  
A(3,4) = -MASS\*V0-MASS\*XG\*R0+2\*MASS\*ZG\*P0+ RHO/2\*L\*\*4\*(2\*ZPP\*...  
P0 + ZPR\*R0) + RHO/2\*L\*\*3\*ZVP\*V0  
A(3,5) = MASS\*U0 - MASS\*YG\*R0+2\*MASS\*ZG\*Q0+RHO/2\*L\*\*3\*ZQ\*U0 +...  
RHO/2\*L\*\*3\*ZQN\*U0\*EPS  
A(3,6) = -MASS\*XG\*P0-MASS\*YG\*Q0+RHO/2\*L\*\*4\*(ZPR\*P0+2\*ZRR\*R0)+...

FILE: TX DSL A1

RHO/2\*L\*\*3\*ZVR\*V0  
A(3,10)= -(WEIGHT - BOY)\*COS(THETA0)\*SIN(PHI0)  
A(3,11)= -(WEIGHT - BOY)\*SIN(THETA0)\*COS(PHI0)  
\*  
A(4,1) = MASS\*YG\*Q0 + MASS\*ZG\*RO + RHO/2\*L\*\*4\*(KP\*PO + ...  
KR\*RO)+RHO/2\*L\*\*3\*(KV\*V0+2\*U0\*(KDB/2\*DBP-KDB/2\*DBS))+...  
RHO/2\*L\*\*3\*U0\*KPROP+ RHO/2\*L\*\*4\*KPN\*PO\*EPS  
A(4,2) = -MASS\*YG\*PO + RHO/2\*L\*\*4\*KVQ\*Q0 + RHO/2\*L\*\*2\*(KV\*U0...  
+ KVV\*W0)  
A(4,3) = -MASS\*ZG\*PO + RHO/2\*L\*\*4\*(KWP\*PO + KWR\*RO) + ...  
RHO/2\*L\*\*3\*KVV\*V0  
A(4,4) = -IXY\*RO + IXZ\*Q0 - MASS\*YG\*V0 - MASS\*ZG\*W0 + ...  
RHO/2\*L\*\*5\*KPQ\*Q0 + RHO/2\*L\*\*4\*(KP\*U0+KWP\*W0)  
A(4,5) = -IZ\*RO + IY\*RO + 2\*IYZ\*Q0 + IXZ\*PO + MASS\*YG\*U0 + ...  
RHO/2\*L\*\*5\*(KPQ\*PO + KQR\*RO) + RHO/2\*L\*\*4\*KVQ\*V0  
A(4,6) = -IZ\*Q0 + IY\*Q0 - 2\*IYZ\*RO + MASS\*ZG\*U0 + ...  
RHO/2\*L\*\*5\*KQR\*Q0 + RHO/2\*L\*\*4\*(KR\*U0+KWR\*W0)  
A(4,10)= -(YG\*WEIGHT-YB\*BOY)\*COS(THETA0)\*SIN(PHI0)...  
-(ZG\*WEIGHT-ZB\*BOY)\*COS(THETA0)\*COS(PHI0)  
A(4,11)= -(YG\*WEIGHT-YB\*BOY)\*SIN(THETA0)\*COS(PHI0)...  
+(ZG\*WEIGHT-ZB\*BOY)\*SIN(THETA0)\*SIN(PHI0)  
\*  
A(5,1) = -MASS\*XG\*Q0 + RHO/2\*L\*\*4\*MQ\*Q0 + RHO/2\*L\*\*3\*MW\*W0 + ...  
RHO/2\*L\*\*3\*U0\*(MDS\*DS+MDB/2\*DBP) + RHO/2\*L\*\*4\*MQN\*Q0\*...  
EPS + RHO/2\*L\*\*3\*(MWN\*W0 + 2\*MDSN\*U0\*DS)\*EPS+...  
RHO/2\*L\*\*3\*U0\*MDB/2\*DBS  
A(5,2) = MASS\*XG\*PO + MASS\*ZG\*RO + RHO/2\*L\*\*4\*(MVP\*PO + ...  
MVR\*RO) + RHO\*L\*\*3\*MVV\*V0  
A(5,3) = -MASS\*ZG\*Q0 + RHO/2\*L\*\*3\*MW\*U0 + RHO/2\*L\*\*3\*MWN\*U0\*EPS  
A(5,4) = -IX\*RO + IZ\*RO - IYZ\*Q0 - 2\*IXZ\*PO + MASS\*XG\*V0 + ...  
RHO/2\*L\*\*5\*(2\*MPP\*PO + MPR\*RO) + RHO/2\*L\*\*4\*MVP\*V0  
A(5,5) = IXY\*RO - IYZ\*PO - MASS\*XG\*U0 -MASS\*ZG\*W0 + RHO/2\*...  
L\*\*4\*MQ\*U0 + RHO/2\*L\*\*4\*MQN\*U0\*EPS  
A(5,6) = -IX\*PO + IZ\*PO + IXY\*Q0 + 2\*IXZ\*RO + MASS\*ZG\*V0 + ...  
RHO/2\*L\*\*5\*(MPR\*PO+2\*MRR\*RO)+RHO/2\*L\*\*4\*MVR\*V0  
A(5,10)= (XG\*WEIGHT-XB\*BOY)\*COS(THETA0)\*SIN(PHI0)  
A(5,11)= (XG\*WEIGHT-XB\*BOY)\*SIN(THETA0)\*COS(PHI0) - ...  
(ZG\*WEIGHT-ZB\*BOY)\*COS(THETA0)  
\*  
A(6,1) = -MASS\*XG\*RO + RHO/2\*L\*\*4\*(NP\*PO +NR\*RO) + RHO/2\*...  
L\*\*3\*(NV\*V0+2\*NDR\*U0\*DR)+RHO\*L\*\*3\*U0\*NPROP  
A(6,2) = -MASS\*YG\*RO + RHO/2\*L\*\*4\*NVQ\*Q0 + RHO/2\*L\*\*3\*(NV\*U0+...  
NVW\*W0)  
A(6,3) = MASS\*XG\*PO + MASS\*YG\*Q0 + RHO/2\*L\*\*4\*(NWP\*PO+NWR\*RO)+...  
RHO/2\*L\*\*3\*NVW\*V0  
A(6,4) = -IY\*Q0 + IX\*Q0 + 2\*IXY\*PO + IYZ\*RO + MASS\*XG\*W0+...  
RHO/2\*L\*\*5\*NPQ\*Q0 + RHO/2\*L\*\*4\*(NP\*U0+NWP\*W0)  
A(6,5) = -IY\*PO + IX\*PO - 2\*IXY\*Q0 - IXZ\*RO + MASS\*YG\*W0+...  
RHO/2\*L\*\*5\*(NPQ\*PO+NQR\*RO) + RHO/2\*L\*\*4\*NVQ\*V0  
A(6,6) = IYZ\*PO - IXZ\*Q0 - MASS\*XG\*U0 -MASS\*YG\*V0 + ...  
RHO/2\*L\*\*5\*NQR\*Q0 + RHO/2\*L\*\*4\*(NR\*U0 +NWR\*W0)  
A(6,10)= (XG\*WEIGHT-XB\*BOY)\*COS(THETA0)\*COS(PHI0)  
A(6,11)= -(XG\*WEIGHT-XB\*BOY)\*SIN(THETA0)\*SIN(PHI0) +...  
(YG\*WEIGHT-YB\*BOY)\*COS(THETA0)

```

*
A(7,1) = COS(PSIO)*COS(THETA0)
A(7,2) = COS(PSIO)*SIN(THETA0)*SIN(PHIO) - SIN(PSIO)*COS(PHIO)
A(7,3) = COS(PSIO)*SIN(THETA0)*COS(PHIO) + SIN(PSIO)*SIN(PHIO)
A(7,10) = V0*COS(PSIO)*SIN(THETA0)*COS(PHIO) + V0*SIN(PSIO)*...
SIN(PHIO) - W0*COS(PSIO)*SIN(THETA0)*SIN(PHIO) + ...
W0*SIN(PSIO)*COS(PHIO)
A(7,11) = -U0*COS(PSIO)*SIN(THETA0) + V0*COS(PSIO)*COS(THETA0)*...
SIN(PHIO) + W0*COS(PSIO)*COS(THETA0)*COS(PHIO)
A(7,12) = -U0*SIN(PSIO)*COS(THETA0) - V0*SIN(PSIO)*SIN(THETA0)*...
SIN(PHIO) - V0*COS(PSIO)*COS(PHIO) - W0*SIN(PSIO)*...
SIN(THETA0)*SIN(PHIO) + W0*COS(PSIO)*SIN(PHIO)
-
*
A(8,1) = SIN(PSIO)*COS(THETA0)
A(8,2) = SIN(PSIO)*SIN(THETA0)*SIN(PHIO) + COS(PSIO)*COS(PHIO)
A(8,3) = SIN(PSIO)*SIN(THETA0)*COS(PHIO) - COS(PSIO)*SIN(PHIO)
A(8,10) = V0*SIN(PSIO)*SIN(THETA0)*COS(PHIO) - V0*COS(PSIO)*...
SIN(PHIO) - W0*SIN(PSIO)*SIN(THETA0)*SIN(PHIO) - ...
W0*COS(PSIO)*COS(PHIO)
A(8,11) = -U0*SIN(PSIO)*SIN(THETA0) + V0*SIN(PSIO)*COS(THETA0)*...
SIN(PHIO) + W0*SIN(PSIO)*COS(THETA0)*COS(PHIO)
A(8,12) = U0*COS(PSIO)*COS(THETA0) + V0*COS(PSIO)*SIN(THETA0)*...
SIN(PHIO) - V0*SIN(PSIO)*COS(PHIO) + W0*COS(PSIO)*...
SIN(THETA0)*COS(PHIO) + W0*SIN(PSIO)*SIN(PHIO)
*
A(9,1) = -SIN(THETA0)
A(9,2) = COS(THETA0)*SIN(PHIO)
A(9,3) = COS(THETA0)*COS(PHIO)
A(9,10) = V0*COS(THETA0)*COS(PHIO) - W0*COS(THETA0)*SIN(PHIO)
A(9,11) = -U0*COS(THETA0) - V0*SIN(THETA0)*SIN(PHIO) - ...
W0*SIN(THETA0)*COS(PHIO)
*
A(10,4) = 1.0
A(10,5) = SIN(PHIO)*TAN(THETA0)
A(10,6) = COS(PHIO)*TAN(THETA0)
A(10,10) = Q0*COS(PHIO)*TAN(THETA0) - R0*SIN(PHIO)*TAN(THETA0)
A(10,11) = Q0*SIN(PHIO)/COS(THETA0)*1.0/COS(THETA0) + ...
R0*COS(PHIO)/COS(THETA0)*1.0/COS(THETA0)
*
A(11,5) = COS(PHIO)
A(11,6) = -SIN(PHIO)
A(11,10) = -Q0*SIN(PHIO) - R0*COS(PHIO)
*
A(12,5) = SIN(PHIO)/COS(THETA0)
A(12,6) = COS(PHIO)/COS(THETA0)
A(12,10) = Q0*COS(PHIO)/COS(THETA0) - R0*SIN(PHIO)/COS(THETA0)
A(12,11) = Q0*SIN(PHIO)/COS(THETA0)*TAN(THETA0) + ...
R0*COS(PHIO)/COS(THETA0)*TAN(THETA0)
*
*
WRITE(10,200)((A(I,J),J=1,12),I=1,12)
*

```

```

*
* CALCULATE THE B MATRIX
*
*
* B(1,1) = RHO/2*L**3*XRDR*U0*R0+RHO/2*L**2*(XRDR*U0*V0+U0**2*...
*           2*XDRDR*DR)
* B(1,2) = U0*Q0*XQDB/2 + U0*W0*XWDB/2 + U0**2*XDBDB*DBS
* B(1,3) = U0*Q0*XQDB/2 + U0*W0*XWDB/2 + U0**2*XDBDB*DBP
* B(1,4) = U0*Q0*XQDS + U0*W0*XWDS +U0**2*2*XDSDS*DS+RHO/2*L**3*...
*           XQDSN*U0*Q0*EPS + RHO/2*L**2*(XWDSN*U0*W0 + 2*XDSDSN*...
*           U0**2*DS)*EPS
* B(1,5) = RHO/2*L**2*0.12*CD0*0.12*RPM
* B(1,6) = SIN(THETA0)
*
*
* B(2,1) = RHO/2*L**2*YDR*U0**2
* B(2,6) = -COS(THETA0)*SIN(PHI0)
*
*
* B(3,2) = U0**2*ZDB/2*RHO/2*L**2
* B(3,3) = U0**2*ZDB/2*RHO/2*L**2
* B(3,4) = U0**2*ZDS*RHO/2*L**2 + RHO/2*L**2*ZDSN*U0**2*EPS
* B(3,6) = -COS(THETA0)*COS(PHI0)
*
*
* B(4,2) = -RHO/2*L**3*U0**2*KDB/2
* B(4,3) = RHO/2*L**3*U0**2*KDB/2
* B(4,6) = -YB*COS(THETA0)*COS(PHI0) + ZB*COS(THETA0)*SIN(PHI0)
*
*
* B(5,2) = RHO/2*L**3*U0**2*MDB/2
* B(5,3) = RHO/2*L**3*U0**2*MDB/2
* B(5,4) = RHO/2*L**3*(U0**2*MDS+MDSN*U0**2*EPS)
* B(5,6) = XB*COS(THETA0)*COS(PHI0) + ZB*SIN(THETA0)
*
*
* B(6,1) = RHO/2*L**3*NDR*U0**2
* B(6,6) = -XB*COS(THETA0)*SIN(PHI0) - YB*SIN(THETA0)
*
*
* FORMULATE THE A AND B MATRIX FOR STATE SPACE REPRESENTATION
*
* MULTIPLY MMINV AND DF/DX
*
* DO 80 I = 1,6
* DO 70 J = 1,6

```

FILE: TX           DSL           A1

```

        SUM = 0.0
        DO 60 K = 1,6
        SUM = SUM + MMINV(I,K)*A(K,J)
60      CONTINUE
        AA(I,J) = SUM
70      CONTINUE
80      CONTINUE
*
*      MULTIPLY MMINV AND DF/DZ
*
        DO 50 I = 1,6
        DO 40 J = 7,12
        SUM = 0.0
        DO 30 K = 1,6
        SUM = SUM + MMINV(I,K)*A(K,J)
30      CONTINUE
        AA(I,J) = SUM
40      CONTINUE
50      CONTINUE
*
*
        DO 5 I = 7,12
        DO 6 J = 1,12
        AA(I,J) = A(I,J)
6      CONTINUE
5      CONTINUE
*
*
        WRITE(10,200)((AA(I,J),J=1,12),I=1,12)
200     FORMAT( 6E12.4)
*
*      MULTIPLY MMINV AND DF/DU
*
        DO 110 I = 1,6
        DO 100 J = 1,6
        SUM = 0.0
        DO 90 K = 1,6
        SUM = SUM + MMINV(I,K)*B(K,J)
90      CONTINUE
        BB(I,J) = SUM
100     CONTINUE
110     CONTINUE
*
*
        WRITE( 9,300)((BB(I,J),J=1,6),I=1,6)
300     FORMAT(6E12.4)
*
*
        DO 405 I = 1,6
        READ (2,401)(GKK(I,J), J=1,21)
405     WRITE(3,401)(GKK(I,J), J=1,21)
401     FORMAT(3E20.10)
*
```



```

*
*   ELSE
*   END IF
*
*   CALL DADS(INFO,ISTRAT,IOPT,IONED,IPRINT,IGRAD,NDV,NCON,DX,...
*           VLB,VUB,OBJ,GW,IDG,NGT,IC,DF,AW,NRA,NCOLA,WK,NRWK,...
*           IWK,NRIWK)
*   IF (INFO.EQ. 0) DELPRT=0.2
*
* DERIVATIVE
* NOSORT
*
*
*   LATYAW = 0.0
*   NORPIT = 0.0
*
*
* *****LINEAR MODEL*****
*
*
*   CALCULATE BB*U PART OF XDOT = AA*X + BB*U
*
*   DO 10 J = 1,6
*     SUM = 0.0
*     DO 15 K = 1,6
*       SUM = SUM + BB(J,K)*UMOD(K)
15    CONTINUE
*   XDOTU(J) = SUM
10  CONTINUE
*   CALCULATE AA*X
*   DO 21 J = 1,12
*     SUM = 0.0
*     DO 25 K = 1,12
*       SUM = SUM + AA(J,K)*X(K)
25  CONTINUE
*   XDOTX(J) = SUM
21  CONTINUE
*   CALCULATE XDOT = AA*X + BB*U
*   DO 31 J = 1,6
*     XDOT(J) = XDOTX(J) + XDOTU(J)
31  CONTINUE
*   DO 35 J = 7,12
*     XDOT(J) = XDOTX(J)
35  CONTINUE
*
*   UDOTM = XDOT(1)
*   VDOTM = XDOT(2)
*   WDOTM = XDOT(3)
*   PDOTM = XDOT(4)
*   QDOTM = XDOT(5)
*   RDOTM = XDOT(6)
*   XDOTM = XDOT(7)
*   YDOTM = XDOT(8)
*   ZDOTM = XDOT(9)
*   PHMDOT = XDOT(10)
*   THETMD = XDOT(11)
*   PSMDOT = XDOT(12)
*   WRITE(8,600)
*   INTEGRATE XDOT TO GET THE STATE VECTOR X

```

```

*
UM =INTGRL(6.0, UDOTM)
VM= INTGRL(0.0, VDOTM)
WM= INTGRL(0.0, WDOTM)
PM= INTGRL(0.0, PDOTM)
QM= INTGRL(0.0, QDOTM)
RM= INTGRL(0.0, RDOTM)
XPOSM = INTGRL(0.0, XDOTM)
YPOSM = INTGRL(0.0, YDOTM)
ZPOSM = INTGRL(0.0, ZDOTM)
PHIM = INTGRL(0.0, PHMDOT)
THETAM = INTGRL(0.0, THETMD)
INTGRD = (UM*UM+VM*VM+WM*WM+PM*PM+QM*QM+RM*RM+...
          XPOSM*XPOSM+(YPOSM-YORD)*(YPOSM-YORD)+...
          (ZPOSM-ORDDEP)*(ZPOSM-ORDDEP)+ PHIM*PHIM+...
          THETAM*THETAM+PSIM*PSIM) + (DSM*DSM+DBSM*DBSM)+...
          (DBPM*DBPM)+(DRM*DRM)
OBJ1 = INTGRL(0.,(0.5)*INTGRD)
OBJ = OBJ1
PSIM = INTGRL(0.0, PSMDOT)
*
*
X(1) = UM
X(2) = VM
X(3) = WM
X(4) = PM
X(5) = QM
X(6) = RM
X(7) = XPOSM
X(8) = YPOSM
X(9) = ZPOSM
X(10) = PHIM
X(11) = THETAM
X(12) = PSIM
*
*
ZDEPTH = ZORD - X(9)
THMANG = X(11)*57.3
UMOD(1)=DRM
UMOD(2) = DBSM
UMOD(3) = DBPM
UMOD(4) = DSM
UMOD(5) = DRPM
UMOD(6) = DBOY
*
PHANG=PHIM/0.0174532925
THMANG=THETAM/0.0174532925
PSMANG= PSIM/ 0.0174532925
PITCHM=THMANG

```

\*\*\*\*\*CONTROL LAW\*\*\*\*\*

```
DBS = UMOD(2)
DBP = UMOD(3)
DS = UMOD(4)
DR = UMOD(1)
DBS = -(GKK(2,1)*U + GKK(2,2)*V + GKK(2,3)*W + GKK(2,4)*P + ...
GKK(2,5)*Q + GKK(2,6)*R + GKK(2,7)*XPOS + GKK(2,8)*YPOS + ...
GKK(2,9)*ZPOS + GKK(2,10)*PHI + GKK(2,11)*THETA + ...
GKK(2,12)*PSI + GKK(2,13)*WM + GKK(2,14)*QM + GKK(2,15)*...
ZPQSM + GKK(2,16)*THETAM + GKK(2,17)*UMOD(2) + GKK(2,18)*...
UMOD(3) + GKK(2,19)*UMOD(4))
DBP = -(GKK(3,1)*U + GKK(3,2)*V + GKK(3,3)*W + GKK(3,4)*P + ...
GKK(3,5)*Q + GKK(3,6)*R + GKK(3,7)*XPOS + GKK(3,8)*YPOS + ...
GKK(3,9)*ZPOS + GKK(3,10)*PHI + GKK(3,11)*THETA + ...
GKK(3,12)*PSI + GKK(3,13)*WM + GKK(3,14)*QM + GKK(3,15)*...
ZPQSM + GKK(3,16)*THETAM + GKK(3,17)*UMOD(2) + GKK(3,18)*...
UMOD(3) + GKK(3,19)*UMOD(4))
DS = -(GKK(4,1)*U + GKK(4,2)*V + GKK(4,3)*W + GKK(4,4)*P + ...
GKK(4,5)*Q + GKK(4,6)*R + GKK(4,7)*XPOS + GKK(4,8)*YPOS + ...
GKK(4,9)*ZPOS + GKK(4,10)*PHI + GKK(4,11)*THETA + ...
GKK(4,12)*PSI + GKK(4,13)*WM + GKK(4,14)*QM + GKK(4,15)*...
ZPQSM + GKK(4,16)*THETAM + GKK(4,17)*UMOD(2) + GKK(4,18)*...
UMOD(3) + GKK(4,19)*UMOD(4))
```

PUT IN STERN AND BOW PLANE STOPS

```
IF(ABS(DBS).GT.0.6) THEN
  DBS = 0.6*ABS(DBS)/DBS
ENDIF
IF(ABS(DBP).GT.0.6) THEN
  DBP = 0.6*ABS(DBP)/DBP
ENDIF
IF(ABS(DS).GT.0.6) THEN
  DS = 0.6*ABS(DS)/DS
ENDIF
```

\*\*\*\*\*NON-LINEAR MODEL\*\*\*\*\*

PROPULSION MODEL

```
SIGNU = 1.0
IF (U.LT.0.0) SIGNU = -1.0
IF (ABS(U).LT.X1TEST) U = X1TEST
SIGNN = 1.0
IF (RPM.LT.0.0) SIGNN = -1.0
ETA = 0.012*RPM/U
```

FILE: TX DSL A1

```
RE = U*L/NU
CDO = .00385 + (1.296E-17)*(RE - 1.2E7)**2
CT = 0.008*L**2*ETA*ABS(ETA)/(A0)
CT1 = 0.008*L**2/(A0)
EPS = -1.0+SIGNN/SIGNU*(SQRT(CT+1.0)-1.0)/(SQRT(CT1+1.0)-1.0)
XPROP = CDO*(ETA*ABS(ETA) - 1.0)
```

\*  
\*  
\*  
\*  
\*  
\*

CALCULATE THE DRAG FORCE, INTEGRATE THE DRAG OVER THE VEHICLE  
INTEGRATE USING A 4 TERM GAUSS QUADATURE

```
LATYAW = 0.0
NORPIT = 0.0
DO 500 K = 1,4
UCF(K) = SQRT((V+G4(K)*R*L)**2 + (W-G4(K)*Q*L)**2)
IF(UCF(K).GT.1E-10) THEN
  TERM0 = (RHO/2)*(CDY*HH(K)*(V+G4(K)*R*L)**2 + ...
           CDZ*BR(K)*(W-G4(K)*Q*L)**2)
  TERM1 = TERM0*(V+G4(K)*R*L)/UCF(K)
  TERM2 = TERM0*(W-G4(K)*Q*L)/UCF(K)
  LATYAW = LATYAW + TERM1*GK4(K)*L
  NORPIT = NORPIT + TERM2*GK4(K)*L
END IF
```

500

CONTINUE

\*  
\*  
\*  
\*  
\*  
\*

FORCE EQUATIONS

LONGITUDINAL FORCE

```
FP(1) = MASS*V*R - MASS*W*Q + MASS*XG*Q**2 + MASS*XG*R**2-...
        MASS*YG*P*Q - MASS*ZG*P*R + (RHO/2)*L**4*(XPP*P**2 + ...
        XQQ*Q**2 + XRR*R**2 + XPR*P*R) + (RHO/2)*L**3*(XWQ*W*Q + ...
        XVP*V*P + XVR*V*R + U*Q*(XQDS*DS + XQDB/2*DBP) + XRDR*U*R*DR) + ...
        (RHO/2)*L**2*(XVV*V**2 + XWH*W**2 + XVDR*U*V*DR + U*W*...
        (XWDS*DS + XWDB/2*DBP) + U**2*(XDS*DS**2 + XDBDB/2*DBP**2 + ...
        XDRDR*DR**2)) - (WEIGHT - BOY)*SIN(THETA) + (RHO/2)*L**3*...
        XQDSN*U*Q*DS*EPS + (RHO/2)*L**2*(XWDSN*U*W*DS + XDS*U**2*...
        DS**2)*EPS + (RHO/2)*L**2*U**2*XPROP + RHO/2*L**3*U*Q*...
        XQDB/2*DBS + RHO/2*L**2*U**2*XDBDB/2*DBS**2 + ...
        RHO/2*L**2*XWDB/2*DBS*U*W
```

\*  
\*  
\*

LATERAL FORCE

```
FP(2) = -MASS*U*R + MASS*XG*P*Q + MASS*YG*R**2 - MASS*ZG*Q*R + ...
        (RHO/2)*L**4*(YPQ*P*Q + YQR*Q*R) + (RHO/2)*L**3*(YP*U*P + ...
        YR*U*R + YVQ*V*Q + YWP*W*P + YWR*W*R) + (RHO/2)*L**2*...
        (YV*U*V + YVW*V*W + YDR*U**2*DR) - LATYAW + (WEIGHT-BOY)*...
        COS(THETA)*SIN(PHI)
```

```

*
*
* NORMAL FORCE
*
FP(3) = MASS*U*Q - MASS*V*P - MASS*XG*P*R - MASS*YG*Q*R + ...
      MASS*ZG*P**2 + MASS*ZG*Q**2 + (RHO/2)*L**4*(ZPP*P**2 + ...
      ZPR*P*R + ZRR*R**2) + (RHO/2)*L**3*(ZQ*U*Q + ZVP*V*P + ...
      ZVR*V*R) + (RHO/2)*L**2*(ZW*U*W + ZVV*V**2 + U**2*(ZDS*...
      DS+ZDB/2*DBP))-NORPIT+(WEIGHT-BOY)*COS(THETA)*COS(PHI)+...
      (RHO/2)*L**3*ZQN*U*Q*EPS + (RHO/2)*L**2*(ZWN*U*W + ZDSN*...
      U**2*DS)*EPS+ RHO/2*L**2*U**2*ZDB/2*DBS
*
*
* ROLL FORCE
*
FP(4) = -IZ*Q*R + IY*Q*R - IXY*P*R + IYZ*Q**2 - IYZ*R**2 + IXZ*P*Q + ...
      MASS*YG*U*Q - MASS*YG*V*P - MASS*ZG*W*P + (RHO/2)*L**5*(KPQ*...
      P*Q + KQR*Q*R) + (RHO/2)*L**4*(KP*U*P + KR*U*R + KVQ*V*Q + ...
      KWP*W*P + KWR*W*R) + (RHO/2)*L**3*(KV*U*V + KVW*V*W) + ...
      (YG*WEIGHT - YB*BOY)*COS(THETA)*COS(PHI) - (ZG*WEIGHT - ...
      ZB*BOY)*COS(THETA)*SIN(PHI) + (RHO/2)*L**4*KPN*U*P*EPS + ...
      (RHO/2)*L**3*U**2*KPROP + MASS*ZG*U*R+ ...
      RHO/2*L**3*U**2*(KDB/2*DBP-KDB/2*DBS)
*
*
* PITCH FORCE
*
FP(5) = -IX*P*R + IZ*P*R + IXY*Q*R - IYZ*P*Q - IXZ*P**2 + IXZ*R**2 - ...
      MASS*XG*U*Q + MASS*XG*V*P + MASS*ZG*V*R - MASS*ZG*W*Q + ...
      (RHO/2)*L**5*(MPP*P**2 + MPR*P*R + MRR*R**2)+(RHO/2)*L**4*...
      (MQ*U*Q + MVP*V*P + MVR*V*R) + (RHO/2)*L**3*(MW*U*W + ...
      MVV*V**2+U**2*(MDS*DS+MDB/2*DBP))+ NORPIT -(XG*WEIGHT- ...
      XB*BOY)*COS(THETA)*COS(PHI)
      + ...
      (RHO/2)*L**3*(MWN*U*W+MDSN*U**2*DS)*EPS+ RHO/2*L**3*...
      U**2*MDB/2*DBS-(ZG*WEIGHT-ZB*BOY)*SIN(THETA)
*
*
* YAW FORCE
*
FP(6) = -IY*P*Q + IX*P*Q + IXY*P**2 - IXY*Q**2 + IYZ*P*R - IXZ*Q*R - ...
      MASS*XG*U*R + MASS*XG*W*P - MASS*YG*V*R + MASS*YG*W*Q + ...
      (RHO/2)*L**5*(NPQ*P*Q + NQR*Q*R) + (RHO/2)*L**4*(NP*U*P+...
      NR*U*R + NVQ*V*Q + NWP*W*P + NWR*W*R) + (RHO/2)*L**3*(NV*...
      U*V + NVW*V*W + NDR*U**2*DR) - LATYAW + (XG*WEIGHT - ...
      XB*BOY)*COS(THETA)*SIN(PHI)+(YG*WEIGHT)*SIN(THETA)...
      +(RHO/2)*L**3*U**2*NPROP-YB*BOY*SIN(THETA)
*
*
* IF(Z.EQ.50.0)THEN
*   WRITE (8,500)(FP(I), I = 1,6)
*   Z = 0.0
*   END IF
*
* NOW COMPUTE THE F(1-6) FUNCTIONS
*
DO 600 J = 1,6
  F(J) = 0.0
DO 600 K = 1,6
  F(J) = MMINV(J,K)*FP(K) + F(J)
600 CONTINUE
*
* THE LAST SIX EQUATIONS COME FROM THE KINEMATIC RELATIONS
*
* FIRST SET THE DRIFT CURRENT VALUES
*
UCO = 0.0
VCO = 0.0
WCO = 0.0
*

```

```

*      INERTIAL POSITION RATES F(7-9)
*
F(7) = UCO + U*COS(PSI)*COS(THETA) + V*(COS(PSI)*SIN(THETA)*...
      SIN(PHI) - SIN(PSI)*COS(PHI)) + W*(COS(PSI)*SIN(THETA)*...
      COS(PHI) + SIN(PSI)*SIN(PHI))
*
F(8) = VCO + U*SIN(PSI)*COS(THETA) + V*(SIN(PSI)*SIN(THETA)*...
      SIN(PHI) + COS(PSI)*COS(PHI)) + W*(SIN(PSI)*SIN(THETA)*...
      COS(PHI) - COS(PSI)*SIN(PHI))
*
F(9) = WCO - U*SIN(THETA) + V*COS(THETA)*SIN(PHI) + W*COS(THETA)*...
      COS(PHI)
*
*      EULER ANGLE RATES F(10-12)
*
F(10) = P + Q*SIN(PHI)*TAN(THETA) + R*COS(PHI)*TAN(THETA)
*
F(11) = Q*COS(PHI) - R*SIN(PHI)
*
F(12) = Q*SIN(PHI)/COS(THETA) + R*COS(PHI)/COS(THETA)
*
*
*      IF (Z.EQ.1.0)WRITE (9,500)(F(I), I = 1,12)
*00  FORMAT(6E12.4)
*      Z = Z + 1
*
*
UDOT = F(1)
VDOT = F(2)
WDOT = F(3)
PDOT = F(4)
QDOT = F(5)
RDOT = F(6)
XDOTA = F(7)
YDOT = F(8)
ZDOT = F(9)
PHIDOT = F(10)
THETAD = F(11)
PSIDOT = F(12)
*
U = INTGRL(6.0,UDOT)
V = INTGRL(0.0,VDOT)
W = INTGRL(0.0,WDOT)
P = INTGRL(0.0,PDOT)
Q = INTGRL(0.0,QDOT)
R = INTGRL(0.0,RDOT)
XPOS = INTGRL(0.0,XDOTA)
YPOS = INTGRL(0.0,YDOT)
ZPOS = INTGRL(0.0,ZDOT)
PHI = INTGRL(0.0,PHIDOT)
THETA = INTGRL(0.0,THETAD)
PSI = INTGRL(0.0,PSIDOT)
*
ZNEW = -ZPOS
PHIANG = PHI/0.0174532925
THEANG = THETA/0.0174532925
PSIANG = PSI/0.0174532925
*
*
*

```

```

DYNAMIC
RN=TIME/(FINTIM/10.-DELT/10000.)
PN=TIME/(FINTIM/20.-DELT/10000.)
O=INT(RN)+1
PP=INT(PN)+1
IF(PP.GE.20) PP=20
IF(O.EQ.11) O=10
*
* ADDITIONALLY THE PLANES SHOULD BE AT EQUILIBRIUM SO THE
* VEHICLE WILL PROCEED AT THIS NEW DEPTH WITHIN SOME TOLERANCE
*
DSM=DX(O)
DBSM=DX(10+O)
DBPM=DX(10+O)
IF(O.GE.10) DSM=0.
IF(O.GE.10) DBPM =0.000
IF(O.GE.10) DBSM =0.000
DRM=DX(20+PP)
*
RPM=DX(30+O)
*
* CONSTRAINTS FOR A DIVE
*
* ORDERED DEPTH = ORDDEP
GW(1) = (ZPOSM-ORDDEP)*.5
GW(2) = (ORDDEP-ZPOSM)*.5
* AUV'S FINAL STATE MUST BE LEVEL FLIGHT AS FOLLOWS
GW(3) = THETAM*10.
GW(4) = -THETAM*10.
GW(5)=(YPOSM-YORD)/.4
GW(6)=(YORD-YPOSM)/.4
GW(7) = -ZPOSM
*
* AVOIDING THE OBSTACLE
*
* IF (DIST1.LT.DSAVE1) DSAVE1=DIST1
*
* IF (DISTV.LT.DSAVE1) DSAVEV=DISTV
*
* DIST1=SQRT((XPOSM-XOBS1)*(XPOSM-XOBS1)+...
* (ZPOSM-ZOBS1)*(ZPOSM-ZOBS1))
* DISTV=SQRT((XPOS-XOBS1)*(XPOS-XOBS1)+...
* (ZPOS-ZOBS1)*(ZPOS-ZOBS1))
*
* GW(8) = (1.-DSAVE1)
NDX=XPOSM/17.425
NDZ=ZPOSM/17.425
NDT=TIME*6./17.425
TERMINAL
IF(INFO.EQ.0) THEN
PRINT*,DSAVE1,DSAVEV
9000 CONTINUE
ELSE
ENDIF
IF(INFO.EQ.0) CALL ENDJOB
CALL RERUN
*
END
STOP

```

## REFERENCES

1. Rowden, W. H., "Policy Guidance for the NAVSEA Integrated Robotics Program," COMNAVESEA Ltr 2269, March 1988.
2. Levece, J. A., "Positive Future for Battlefield Robotics," Military Robotics, v.2, no. 7, pp. 3-4, April 6, 1988.
3. Levece, J. A., "Towtaxi ROV Delivered," Military Robotics, v.2, no. 8, p. 7, April 20, 1988.
4. Levece, J. A., "Dolphin UUV for Minsweepers," Military Robotics, v.2, no. 6, p. 4, March 23, 1988.
5. Levece, J. A., "RECON UUV Reaches New Depths," Military Robotics, v.2, no. 9, p. 5, May 4, 1988.
6. Levece, J. A., "Unmanned Spaceship to Test Aero-braking," Military Robotics, v.2, no. 10, p. 5, May 18, 1988.
7. Kambhampati, s. and Davis, L. S., "Multiresolution Path Planning for Mobile Robots," Computer Vision Laboratory Report, University of Maryland, College Park, May 1985.
8. Wong, E. K. and Fuk, K. S., "Hierarchical Orthogonal Space Approach to Three-Dimensional Path Planning," IEEE Journal of Robotics and Automation, V. RA-2, No. 1, March 1986.
9. Herman, M., "Fast, Three-Dimensional, Collision-Free Motion Planning," Proceedings 1986 IEEE International Conference on Robotics and Automation, V. 2, pp. 1056-1063, April 1986.
10. Singh, S. and Wagh, M. D., "Robot Path Planning Using Intersecting Convex Shapes," Proceedings 1986 IEEE International Conference on Robotics and Automation, V. 3, pp. 1743-1748, April 1986.



11. Kuan, D. T., Zamiska, J. C. and Brooks, R. A., "Natural Decomposition of Free Space for Path Planning," Proceedings IEEE International Conference on Robotics and Automation, pp. 168-173, 1985.
12. Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," International Journal of Robotics Research, V. 5, No. 1, Spring 1986.
13. Tournassoud, P., "A Strategy for Obstacle Avoidance and Its Application to Multi-Robot Systems," Proceedings IEEE 1986 Conference on Robotics and Automation, V. 2, pp. 1224-1229, April 1986.
14. Krogh, B. H., "Guaranteed Steering Control," Proceedings of the 1985 American Control Conference, V. 2, pp. 950-955, June 1985.
15. Thorpe, C. E., "Path Relaxation: Path Planning for a Mobile Robot," CMU Robotics Institute Report CMU-RJ-TR-84-5, April 1984.
16. Krogh, B. H. and Thorpe, C. E., "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles," Proceedings 1986 International Conference on Robotics and Automation, V. 3, pp. 1664-1669, April 1986.
17. Gilbert, E. G. and Johnson, D. W., "Distance Functions and Their Application to Robot Path Planning in the Presence of Obstacles," IEEE Journal of Robotics and Automation, V. RA-1, No. 1, March 1985.
18. Johnson, D. W. and Gilbert, E. G., "Minimum Time Robot Path Planning in the Presence of Obstacles," Proceedings of the 24th Conference on Decision and Control, V. 3, pp. 1748-1753, December 1985.
19. Sanders, D. W., A Feasibility Study in Path Planning Application Using Optimization Techniques, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1987.

20. Vanderplaats, G. N. and Sugimoto, H., ADS Design Optimization Program, FORTRAN Source Code, Naval Postgraduate School, Monterey, California, June 1982.
21. International Business Machines Program Number, 5798-PXJ, Dynamic Simulation Language, June 1984.
22. Olson, T. F., Application of Numerical Optimization in Modern Control, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1986.

### INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2. Library, code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3. Chairman, Code 69Hy Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5004	3
4. Professor D. L. Smith, Code 69Sm Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5004	10
5. Professor R. McGhee, Code 52Mz Computer Science Department Naval Postgraduate School Monterey, California 93943-5004	1
6. Professor R. Christi, Code 62Cx Electrical and Computer Engineering Department Naval Postgraduate School Monterey, California 93943-5004	1
7. Dr. G. N. Vanderplaats Engineering Design Optimization, Inc. 1275 Camino Rio Verde Santa Barbara, California 93111	1
8. LCDR David W. Sanders, USN 204 First Street Butler, Pennsylvania 16001	1
9. LCDR T. Olson, USN 6905 Ashbury Dr. Springfield, Virginia 22152	1

10. Russ Weneth, Code u25 1  
Naval Surface Weapons Center  
White Oak, Maryland 20910
11. Paul Heckman, 1  
Head, Underseas AI & Robotics Branch  
Naval Ocean System Center  
San Diego, California 92152
12. RADM G. Curtis, USN PMS-350 1  
Naval Sea Systems Command  
Washington, D. C. 20362-5101
13. LT Relle L. Lyman, Jr., USN Code 90G 1  
Naval Sea Systems Command  
Washington, D. C. 20362-5101
14. LT Richard Boncal, USN 1  
Raymes Neck Road, RFD 2  
York, Maine 03909
15. LT Winston Smith 3  
7979 Cornet Place  
Pensacola, Florida 32514