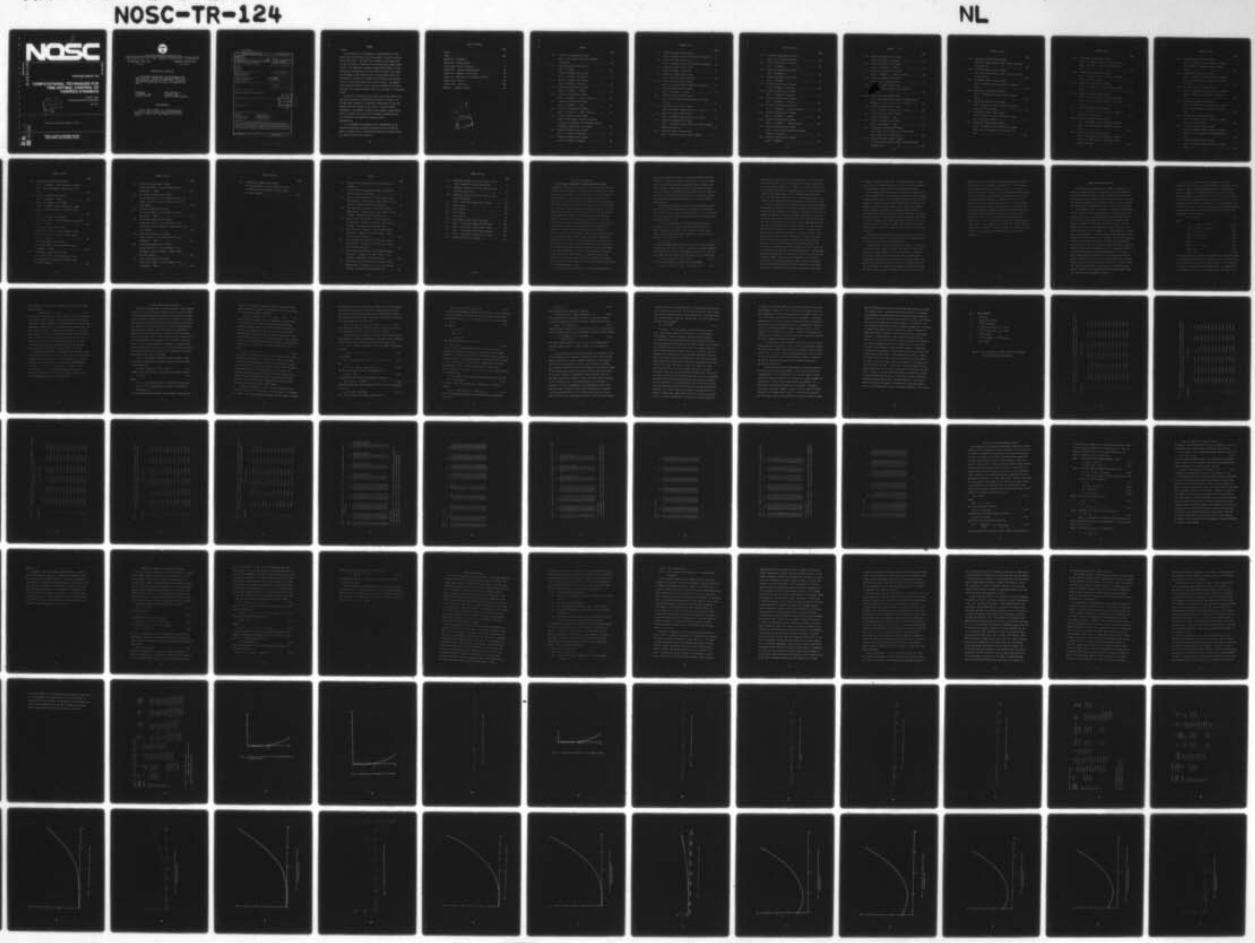
AD-A042 206 NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CALIF F/G 19/8
COMPUTATIONAL TECHNIQUES FOR TIME-OPTIMAL CONTROL OF TORPEDO DY--ETC(U)
MAY 77 W S LAPP

UNCLASSIFIED NOSC-TR-124 NL

1 OF 3
ADA
042206



NOSC

NOSC TR 124

AD A042206

NOSC TR 124

Technical Report 124

# COMPUTATIONAL TECHNIQUES FOR TIME-OPTIMAL CONTROL OF TORPEDO DYNAMICS

by W.S. Lapp
Undersea Systems Department

May 1977



Approved for public release; distribution unlimited



NAVAL OCEAN SYSTEMS CENTER
SAN DIEGO, CALIFORNIA 92152

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA. 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

RR GAVAZZI CAPT, USN
Commander

HOWARD L BLOOD, PhD
Technical Director

## ADMINISTRATIVE INFORMATION

The research reported herein was performed from May, 1972 to May 1977. The work was supported by NOSC in the form of computer time given on IR/IED basis. The work is the author's dissertation for a Ph.D. in Engineering from UCLA.

Released by
J R Campbell, Head
Acoustic Division

Under Authority of
C G Beatty, Head
Undersea Systems Department

## ACKNOWLEDGMENTS

I would like to thank Dr. A. V. Balakrishnan and Dr. Louis A. Lopes for their support and encouragement. Thanks is also due to Mr. James Campbell and Dr. A. G. DiLoreto.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|
| 1. REPORT NUMBER: NOSC-TR-124 | 2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle): COMPUTATIONAL TECHNIQUES FOR TIME-OPTICAL CONTROL OF TORPEDO DYNAMICS | 5. TYPE OF REPORT & PERIOD COVERED: Research and Development May 1972 – May 1977 / 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s): W.S. Lapp | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Ocean Systems Center, San Diego, CA 92132 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Ocean Systems Center, San Diego, CA 92132 | 12. REPORT DATE: May 1977 / 13. NUMBER OF PAGES |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | 15. SECURITY CLASS. (of this report): Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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



18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

| | |
|---|---|
| Torpedoes | Epsilon Technique |
| Trajectory calculation | Conjugate gradient method |
| Time-optimal control | Linearization method |

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

This report develops computational techniques for calculating time-optimal pursuit trajectories for an underwater torpedo. A comparative study of three classes of techniques is presented, the Epsilon Technique, the conjugate gradient method and the method of linearization. It is shown that a combination of the first and third techniques yields the best results.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## SUMMARY

### Problem

The problem which is investigated is the determination of the optimal open loop controls so that the torpedo's trajectory minimizes the cost functional. The basic cost functional is the elapsed time so that the problem is referred to as the time-optimal torpedo control problem. For comparison of different computational methods, a fixed-time problem of minimizing the miss distance was solved. The results are restricted to the torpedo's motion in the horizontal plane, and the equations of motion of a torpedo are simplified so that the equations are linear with respect to the control variable. The computational techniques that were used to solve the problem are the Epsilon Technique, the conjugate gradient method, and the method of linearization.

### Results

The results show that the best controls for the time-optimal torpedo control problem are obtained when the Epsilon Technique is initialized from the output of the method of linearization to the corresponding fixed-time torpedo control problem. The basic problem encountered is the computer limitation in the number of sample points and basis functions able to be used on present-day computers due to size limitations and roundoff and truncation errors.

### Recommendations

It is recommended that different ways of implementing the computational techniques be explored to determine how to increase the number of sample points or of removing the need for more sample points by a better choice of basis functions.

TABLE OF CONTENTS



FIGURES

FIGURES (Cont'd)

FIGURES (Cont'd)

FIGURES (Cont'd)

FIGURES (Cont'd)

FIGURES (Cont'd)

FIGURES (Cont'd)

FIGURES (Cont'd)

FIGURES (Cont'd)

FIGURES (Cont'd)

FIGURES (Cont'd)

FIGURES (Cont'd)

TABLES

TABLES (Cont'd)

## Chapter One: Introduction

In this report, the results of applying several different computational techniques to the torpedo control problem are presented. The torpedo control problem is defined to be the determination of the optimal open loop controls so that the torpedo's trajectory minimizes the cost functional. In the fixed time torpedo control the cost functional is the distance between the torpedo and the target and the final time is fixed. For the time-optimal torpedo control problem the cost functional is the final time and the target's position is a final endpoint condition that the state equation must satisfy. The motivation behind choosing this problem is an attempt to answer the following question. Given that the torpedo has a finite amount of fuel that is expended at a constant rate, will the effective range of the torpedo be significantly increased if the time-optimal pursuit trajectory is followed? In order to answer the above question the time-optimal pursuit trajectories of the torpedo must be calculated. Hence, we have the time-optimal torpedo control problem. Since most computational techniques are restricted to fixed time optimal control problems, the fixed time torpedo control problem was defined to obtain a comparison of different computational methods. Although this problem would appear to have been investigated in the past, the author has not been able to find any record in the literature, by contacting people at the Naval Ocean Systems Center who have worked on torpedoes and torpedo control systems for over twenty years or by contacting several professors at the Naval Postgraduate School.

The computational techniques used to solve the torpedo control

problem are the Epsilon Technique, the conjugate gradient method and the method of linearization about a known trajectory. The only method which is directly applicable to time-optimal control problems is the Epsilon Technique. A new method of applying the Epsilon Technique is given. In solving the time-optimal torpedo control problem the Epsilon Technique is initialized by using the results of the method of linearization about a known trajectory applied to the fixed time torpedo control problem.

Before proceeding further let us define some notational abbreviations that will be used throughout this dissertation. The dependence of a vector on time will usually be implied, that is $\underline{x} \equiv \underline{x}(t)$, especially when being used as an argument. Partial derivatives will be denoted by a subscript, that is $H_u \equiv \frac{\partial H}{\partial u}$.

The Epsilon Technique was first presented by Balakrishnan in references [1] - [4]. Basically, the Epsilon Technique converts a dynamic optimal control problem into a nondynamic optimal control problem by incorporating the dynamic equation as a penalty function added to the cost functional. Let

$$\dot{\underline{x}}(t) = \underline{f}(t,\underline{x},\underline{u}) \tag{1.1}$$

be the dynamic equation where the state vector $\underline{x}(t)$ is continuous and differentiable, the control vector is contained in a compact subspace of $C^1$ and appropiate initial and final endpoint conditions are given. We want to determine the controls $\underline{u}(t)$ that minimize the cost functional

$$J(t,\underline{x},\underline{u}) = \phi(t_f,\underline{x}(t_f)) + \int_0^{t_f} L(t,\underline{x},\underline{u})\, dt \tag{1.2}$$

The Epsilon Technique forms the epsilon cost functional

$$J_\varepsilon(t,\underline{x},\underline{u}) = J(t,\underline{x},\underline{u}) + \frac{1}{\varepsilon}\int_0^{t_f} || \dot{\underline{x}} - \underline{f}(t,\underline{x},\underline{u})||^2\, dt \tag{1.3}$$

and minimizes it over all admissible $\underline{x}$ and $\underline{u}$ subject to the initial and final conditions being met. In this way one is no longer constrained to satisfy the dynamic equation (1.1). However, the minimum of the epsilon cost functional must satisfy the dynamic equation due to the huge penalty incurred of the dynamic equation is not satisfied. Balakrishnan has shown that the minimum of the epsilon cost functional converges to the minimum of the original optimal control problem as epsilon goes to zero.

The Epsilon Technique has been applied to several different problems. Balakrishnan in reference [4] and Taylor in reference [5] determined the minimum time and trajectory for a supersonic interceptor to reach a given altitude and velocity. The results compared favorably with the previous solutions to this problem although the final times were slightly different. Taylor and Constantinides in reference [6]-[7] applied the Epsilon Technique to the Earth-Mars orbit transfer problem. The final time that they obtained did not match the previous results. In reference [8], Mikami solved the minimum time problem of a missile flying to a fixed point by the Epsilon Technique. In reference [9], Mikami solved the minimum time problem of a particle subjected to constant magnitude thrust which is controlled by two angles. He compared the Epsilon Technique to the conjugate gradient method using sine basis functions and polynomial basis functions in the Epsilon Technique. Good agreement between the three methods was obtained for the final time and the state trajectory. However, the controls were different. The most recent application of the Epsilon Technique was done by Hewett in reference [10]. He developed a new penalty function to handle state constraints and control constraints and states that the convergence of the method

is improved if the second order terms in the Newton-Raphson iteration are included. This method was applied to a missile intercept problem, a climb performance problem and an air combat maneuvering problem.

The conjugate gradient method was first presented by Hestenes and Stiefel in reference [11]. The method was expanded by Fletcher and Reeves in reference [12]. Lasdon, et al, expanded the method to optimal control problems in reference [13] and Pagurek and Woodside, in reference [14], expanded the method to handle optimal control problems with bounded control variables. Essentially, the conjugate gradient method is a modification of the method of steepest descents that takes into account second order effects to improve the convergence properties. Since there is no way of presenting the method without going into specifics, discussion of the method is deferred to chapter four. The number of optimal control problems to which this method has been applied are too numerous to mention.

The method of linearization about a known trajectory is a very old method of solving nonlinear optimal control problems. Reference [15] is one of many good books that discuss the method.

The arrangement of this dissertation is as follows. In chapter two the torpedo control problem is defined. The equations of motion of a torpedo are presented along with the cost functionals for the fixed time torpedo control problem and the time-optimal torpedo control problem. Then, the questions of existence of an optimal trajectory and of controllability of the system are discussed. In chapter three, the Epsilon Technique is presented for the general optimal control problem. The Rayleigh-Ritz technique of expanding the state vector in terms of

known basis functions and the Newton-Raphson technique for minimizing the the epsilon cost functional are presented. Then the technique is applied to the torpedo control problem. Finally, a discussion is given of the basis functions which are used in the Rayleigh-Ritz expansion and a comparison is made between the different basis functions. The conjugate gradient method is presented in chapter four. The method of linearization about a known trajectory is given in chapter five along with a discussion on how the method is used to initialize the Epsilon Technique. In chapter six both the maximum principle and the epsilon maximum principle are applied to the torpedo control problem. An explicit form is obtained for the optimal controls. Chapter seven contains the results of applying the different computational methods to the torpedo control problem. First, the fixed time torpedo control problem is discussed and then the time-optimal torpedo control problem. Finally, in chapter eight, the conclusions are presented.

## Chapter Two: System Equations

In this chapter the torpedo control problem is presented as a system of equations that are to be optimized. First, the equations of motion of a torpedo are given in a state space formulation. Then, the cost functionals are presented for the two problems that are considered, the fixed time problem and the time-optimal problem. Finally, a discussion of existence of optimal controls and of controllability is given.

The equations of motion of a torpedo are derived in reference [16] and are presented in state space formulation in reference [17]. Since our main concern is with determining optimal pursuit trajectories for the torpedo only the equations that were used are presented and the reader is referred to the references for their derivation. For our purposes the control variables were chosen to be the position of the rudders and elevators of the torpedo. It is assumed that they can change positions instantaneously. With this assumption the state space for the torpedo motion equations has twelve independent variables. The equations of motion are nonlinear in both the state and control variables. By assuming that the roll of the torpedo is always controlled to zero, the equations of motion can be decoupled into a six dimensional system for the horizontal plane and a six dimensional system for the vertical plane. These systems are referred to as the yaw system and the pitch system, respectively. Due to the lack of gravitational forces, the yaw system is simpler than the pitch system. Therefore , in order to solve the simplest system first our investigation is confined to the yaw system. By making a minor approximation the yaw system becomes linear with respect to the scalar control variable.

Let $(x_1,x_2)$ be the position of the torpedo in an inertial coordinate system. Let $(x_3,x_4)$ be the components of the velocity of the torpedo in a body fixed coordinate system which is centered at the center of buoyancy of the torpedo with the positive X-axis pointing through the head of the torpedo parallel to its length and the positive Y-axis being 90° counterclockwise from the positive X-axis. Let $x_5$ be the angular rate of change of the torpedo and $x_6$ be the heading of the torpedo in the inertial coordinate system with a positive angle measured couterclockwise from the X-axis. Then the equations of motion of the torpedo for the yaw system are:

$$\dot{\underline{x}} = \underline{f}(\underline{x}) + \underline{g}(\underline{x})u \tag{2.1}$$

where,

$$f_1(\underline{x}) = x_3 \cos(x_6) - x_4 \sin(x_6) \tag{2.2}$$

$$f_2(\underline{x}) = x_3 \sin(x_6) + x_4 \cos(x_6) \tag{2.3}$$

$$f_3(\underline{x}) = c_1 x_3^2 + c_2 x_4 x_5 + c_3 \tag{2.4}$$

$$f_4(\underline{x}) = c_4 x_3 x_4 + c_5 x_3 x_5 \tag{2.5}$$

$$f_5(\underline{x}) = c_7 x_3 x_4 + c_8 x_3 x_5 \tag{2.6}$$

$$f_6(\underline{x}) = x_5 \tag{2.7}$$

$$g_i(\underline{x}) = 0 \ , \ i = 1,2,3,6 \tag{2.8}$$

$$g_4(\underline{x}) = c_6 x_3^2 \tag{2.9}$$

$$g_5(\underline{x}) = c_9 x_3^2 \tag{2.10}$$

The initial and final position of the torpedo are given and the control u is constrained to be within the closed set [-U,U]. The values of the constants used in (2.2)-(2.10) are given in Table 2.1. If one wishes to consider a moving target then the state vector can be modified in the following way. Let $(x_T,y_T)$ be the position of the target in the

Table 2.1: Values of the Constants Used in the Torpedo Motion Equations

| Constant | Value |
|---|---|
| $c_1$ | -0.005365516 |
| $c_2$ | 1.961174 |
| $c_3$ | 24.49011 |
| $c_4$ | -0.06383694 |
| $c_5$ | -0.08583888 |
| $c_6$ | 0.02323317 |
| $c_7$ | -0.0009094651 |
| $c_8$ | -0.2250926 |
| $c_9$ | -0.01158485 |

inertial coordinate system. Letting $x_1'=x_1-x_T$, $x_2'=x_2-y_T$ we obtain the equations:

$$\dot{x}_1' = f_1(\underline{x}) - \dot{x}_T \tag{2.11}$$

$$\dot{x}_2' = f_2(\underline{x}) - \dot{y}_T \tag{2.12}$$

By redefining the state vector to be $\underline{x}^T=(x_1', x_2', x_3, x_4, x_5, x_6)$ we have that the initial position is now the difference between the initial torpedo position and the initial target position and the final position is the origin. The two systems are identical for stationary targets. The advantage of the above transformation is that for a moving target the final point is the origin rather than a curve in the inertial coordinate system. However, since most of this thesis only deals with stationary targets we will confine our discussion to the unmodified yaw system.

Since only the Epsilon Technique is directly applicable to time-optimal control problems a fixed time problem was considered as well as the time-optimal control problem. The cost functional for the time optimal control problem is:

$$J(t,\underline{x},u) = t_f \tag{2.13}$$

The cost functional for the fixed time problem is

$$J(t,\underline{x},u) = 100.(x_1(t_f)-x_{1f})^2+100.(x_2(t_f)-x_{2f})^2 \tag{2.14}$$

For some cases (2.14) was expanded to include all of the components of the state vector. The problem is to determine the control u which minimizes the cost functional $J(t,\underline{x},u)$.

Before attempting to solve an optimal control problem one usually shows that an optimal control exists and that the system is controllable. Existence of an optimal control is usually shown by showing that the

system equations satisfy Filipov's condition. For our case one would have to show that

$$[\underline{x},\underline{f}(\underline{x})+\underline{g}(\underline{x})u] \leq c(1 + ||\underline{x}||^2) \tag{2.15}$$

is satisfied for all admissible $\underline{x}$ and u. However, the above condition is not true in general for our case since the left hand side contains cubic terms. Several conditions are present that seem to indicate that an optimal trajectory exists. If we assume that the torpedo varies its thrust in order to keep $x_3$ constant then the dimension of the state vector reduces to five, the state equations become linear and (2.15) is easily satisfied. Also, for any constant admissible u there exists a stable critical point $\underline{x}_u$. Although the above conditions are not sufficient to prove that an optimal control exists they do agree with the intuitive feeling that an optimal control does exist.

A sufficient condition for controllability of a nonlinear system is that the following matrix, $[B,AB,\ldots,A^nB]$ have rank n+1 where the dimension of the state vector is n+1, where $B=\underline{g}$ and $A=\underline{f}_x + \underline{g}_x u$ are evaluated at a critical point of the system. This condition holds if we choose a critical point such that $u \neq 0$. However, if the critical point has $u = 0$ then the above condition is not met. Since the above condition is met for all critical points but one it is my feeling that the system is controllable.

## Chapter Three: Epsilon Technique

The theory of Balakrishnan's Epsilon Technique has been presented several times in the literature, references [1]-[3]. Since this thesis is mainly concerned with applying the Epsilon Technique to the torpedo guidance problem the presentation here will be confined to describing the technique with regard to the epsilon cost functional, the Rayleigh-Ritz expansion of the state variables and the modified Newton-Raphson algorithm used to determine the coefficients of the Rayleigh-Ritz expansion that minimize the epsilon cost function. The presentation will be for a general optimal control problem. Then, the application of the Epsilon Technique to the torpedo control problem is presented. Finally, a least squares technique is presented as a method for comparing different basis functions for use in the Rayleigh-Ritz expansion of the state variables and the results of the comparison for several different basis functions are presented

First, a description of the general control problem is given. The problem is to determine the control variables $\underline{u}(t)$ that minimize the cost functional

$$J(t,\underline{x},\underline{u}) = \phi(t_f,\underline{x}(t_f)) + \int_0^{t_f} L(t,\underline{x},\underline{u})\, dt \tag{3.1}$$

subject to the state variables, $\underline{x}$, satisfying the dynamic equation

$$\dot{\underline{x}} = \underline{f}(\underline{x},\underline{u}) \tag{3.2}$$

where,

$\underline{x}(0) = \underline{x}_o$, $\underline{x}(t_f) \varepsilon G(\underline{x}(t_f)) = \{$the set of admissible final points$\}$

$\underline{f}(\underline{x},\underline{u})$ is $C^1$ in $\underline{x}$ and continuous in $\underline{u}$ and $\underline{u}$ is contained in the compact set U.

The Epsilon Technique reformulates the above dynamic optimal control

problem as a nondynamic optimization problem by incorporating the dynamic equation as a penalty function added to the cost functional. The epsilon cost functional is given by,

$$J_\varepsilon(t,\underline{x},\underline{u}) = J(t,\underline{x},\underline{u}) + \int_0^t f \; || \; \dot{\underline{x}} - \underline{f}(\underline{x},\underline{u}) \; ||^2 \; dt/2\varepsilon \qquad (3.3)$$

The epsilon cost functional is then minimized over all admissible $\underline{x}$ and $\underline{u}$ where the set of admissible control variables has been expanded to include relaxed, or chattering controls and the set of admissible state variables is the class of absolutely continuous functions that satisfy the initial and final end conditions. In the Epsilon Technique one gives up satisfying the dynamic equation exactly in return for satisfying the end conditions exactly. The minimization of the epsilon cost functional is performed in two steps. Let us define,

$$I(\underline{x},\underline{u}) = || \; \dot{\underline{x}} - \underline{f}(\underline{x},\underline{u}) \; ||^2 \qquad (3.4)$$

First, $I(\underline{x},\underline{u})$ is minimized with respect to $\underline{u}$ at each time step. Then the epsilon cost functional is minimized with respect to the state variables using the controls obtained in the first step. Let $\underline{u}_i$ and $\underline{x}_i$ be the controls and state vector obtained from the above two steps. Balakrishnan has shown in reference [1] that as i increases the sequences converge to an optimal state vector and an optimal control vector that is dependent upon epsilon. As epsilon goes to zero the optimal controls and state vector converge to the optimal controls and state vector of the original control problem.

A Rayleigh-Ritz procedure is used to reduce the dimensionality of the the system to finite dimensions. The state vector is represented as a vector in a finite dimensional subspace. Then the problem is reduced

to determining the controls and the coefficients of the basis functions that minimize the epsilon cost functional. Although several different basis functions were tried and compared the following basis functions were found to be as good or better than those tried and will be referred to as the standard basis functions. Let

$$x_j(t) = A_j^{\circ} + A_j^{\circ\circ} t/t_f + \sum_i A_{j,i} \sin(i\pi t/t_f) \ , \ j=1,\ldots,n \qquad (3.5)$$

Then the end conditions are satisfied exactly by the choice of $A^{\circ}$ and $A^{\circ\circ}$ since the basis functions equal zero at the end points.

Let $\underline{A}$ be the vector containing all of the coefficients which are to be optimized. A modified Newton-Raphson technique is used to determine the coefficients that minimize the epsilon cost functional. Let $t_i$, $i = 1,\ldots,N$ be an equally spaced partition of the interval $[0,t_f]$ where we define

$$\Delta \equiv t_{i+1} - t_i = t_f / ( N - 1 ) \qquad (3.6)$$

Let us define

$$W_{i,j} = (\dot{x}_j(t_i) - f_j(t_i,\underline{x},\underline{u}))(\Delta/\varepsilon)^{1/2} \quad i=1,..,N, \ j=1,..,n \qquad (3.7)$$

and

$$\underline{W}^T = ( W_{1,1},..,W_{N,1},\ldots,W_{N,n}, J(t,\underline{x},\underline{u})^{1/2} ) \qquad (3.8)$$

Then the epsilon cost functional is approximated by;

$$J_{\varepsilon}(t,\underline{x},\underline{u}) = W^T W \qquad (3.9)$$

The gradient of the epsilon cost functional is given by;

$$\nabla_{\underline{A}} J_{\varepsilon}(t,\underline{x},\underline{u}) = ( \nabla_{\underline{A}} W)^T W \qquad (3.10)$$

Neglecting second order terms the Hessian of the epsilon cost functional is given by;

$$\Delta_{\underline{A}} J_{\varepsilon}(t,\underline{x},\underline{u}) = (\nabla_{\underline{A}} W)^T (\nabla_{\underline{A}} W) \qquad (3.11)$$

Hence, the coefficients are updated by the iteration

$$\underline{A}_{i+1}=\underline{A}_i-(\Delta_{\underline{A}_i} J_\varepsilon(t,\underline{x},\underline{u}))^{-1}\nabla_{\underline{A}_i} J_\varepsilon(t,\underline{x},\underline{u}) \tag{3.12}$$

In using the Newton-Raphson algorithm described above it is not necessary to compute the inverse of the Hessian of the epsilon cost functional as shown in equation (3.12). It is sufficient to solve the linear equation

$$C\underline{y} = \underline{d} \tag{3.13}$$

for $\underline{y}$ where,

$$C = \Delta_{\underline{A}} J_\varepsilon(t,\underline{x},\underline{u})$$

$$\underline{y} = \underline{A}_{i+1} - \underline{A}_i$$

$$\underline{d} = -\nabla_{\underline{A}} J_\varepsilon(t,\underline{x},\underline{u})$$

Then, the new coefficients are given by;

$$\underline{A}_{i+1} = \underline{A}_i + \underline{y} \tag{3.14}$$

Since C is very ill-conditioned we follow the suggestion given by Luenberger in reference [18] and add a small constant to the diagonal elements of C. Even with the addition of 0.01 to the diagonal elements of the Hessian matrix C the condition number of the Hessian matrix was seven or eight. The vector $\underline{y}$ is computed using a double precision routine that decomposes C into a lower triangular matrix $C_L$ and an upper triangular matrix $C_U$ such that $C = C_L C_U$. Then $\underline{y}$ is found by two successive back substitutions.

For the torpedo control problem the dynamic equation has the form

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}) + \underline{g}(\underline{x})u \tag{3.15}$$

The cost functional for the fixed time torpedo control problem is given by;

$$J(t,\underline{x},u) = 100(x_1(t_f)-x_{1f})^2 + 100(x_2(t_f)-x_{2f})^2 \tag{3.16}$$

and for the time-optimal torpedo guidance problem the cost functional is given by;

$$J(t,\underline{x},u) = t_f \tag{3.17}$$

Hence the epsilon cost functional is given by

$$J_\varepsilon(t,\underline{x},u)=J(t,\underline{x},u) + \frac{1}{\varepsilon}\int_0^{t_f} ||\dot{\underline{x}}-\underline{f}(\underline{x})-\underline{g}(\underline{x})u||^2\, dt \tag{3.18}$$

Since the epsilon cost functional is linear in the control variable u an analytic expression exists for the control variable that minimizes

$$I(\underline{x},u) = ||\dot{\underline{x}} -\underline{f}(\underline{x})-g(\underline{x})u||^2 \tag{3.19}$$

Taking the partial derivative of $I(\underline{x},u)$ with respect to u and equating to zero the following expression is obtained for the optimal control variable

$$u_o = -\frac{(\dot{x}_4-f_4(\underline{x}))g_4(\underline{x}) + (\dot{x}_5 -f_5(\underline{x}))g_5(\underline{x})}{g_4(\underline{x})^2 + g_5(\underline{x})^2} \tag{3.20}$$

Since the second derivative of $I(\underline{x},u)$ is equal to the denominator of the above expression which is always positive $u_o$ is the minimum of $I(\underline{x},u)$.

In any application of the Epsilon Technique it is important to use basis functions that can adequately represent the state vector over the entire time interval.In order to determine whether the basis functions given in (3.5) are adequate to represent the trajectory of a torpedo and to obtain a comparison of several different basis functions the method of least squares approximation was used to obtain the best coefficients for the basis functions to approximate a known trajectory. Then the trajectory obtained by using these coefficients can be compared with the actual trajectory to see if the approximation of the trajectory is adequate. In this way a comparison can be made between different basis functions. Now, the method of least squares approximation is presented. Using the same partition of the time interval given before let D be a matrix with the i,j th element being the value of the j th state variable

at the i th time, B be a matrix with the i,k th element being the value of the k th basis function at the i th time step and A be the matrix containing the coefficients of the basis functions with the i,j th element being the j th coefficient for the i th state variable. The coefficients A are obtained from the equation

$$A^T = (B^T B)^{-1} B^T D \tag{3.21}$$

In Table 3.1 a list of the basis functions that were tried is given. These functions were first tried with the epsilon technique for the general minimum time problem. The basis functions numbered 3 through 6 were rejected because the basis functions approach zero as time increases and the trajectory of the torpedo converges to the straight line connecting the initial and final points. The polynomial functions shown as number seven didn't perform well at all. Therefore, only the standard basis functions and the basis functions which were proposed by Taylor in reference [19] and are numbered two in the table are compared. Taylor modifies the standard basis functions by approximating the initial and final conditions with a cubic instead of a straight line and by multiplying each basis function by the term $\sin(\pi t/t_f)$. The latter modification causes both the state variables and their first derivatives to be equal to zero at the initial and final times.In order to understand the effect of each of the above modifications the case of the standard basis functions with a cubic approximation to the initial and final end point conditions was considered. The three cases that are considered for comparison are; a maximum turn for 0.6 seconds, a maximum turn for 0.3 seconds followed by zero rudder deflection for 0.3 seconds and a maximum turn for 0.3 seconds followed by 9.7 seconds of zero rudder deflection.

The values of the optimal coefficients for the different basis functions are presented in Tables 3.2 - 3.8. The values of the comparisons of the basis functions with the actual trajectories are presented in Tables 3.9 - 3.11. The results show that the position of the torpedo is represented adequately by all the basis functions, however the velocity components of the state vector are approximated better by the standard basis functions. There is no significant difference between the case where the initial trajectory is a straight line or a cubic. Therefore, the standard basis functions with a straight line initial approximation between the endpoints was used in applying the Epsilon Technique.

In applying the Epsilon Technique the value of epsilon is usually decreased until no improvement is shown. For the torpedo control problem the results were virtually identical for epsilon having a value of either 0.01 or 0.001. The results started to deteriorate when a value of 0.0001 was used for epsilon as a result of computational problems. Therefore, the value of epsilon that was used in 0.001 although occassionally epsilon has the value 0.01.

There are several computational problems in applying the Epsilon Technique to the torpedo control problem. The epsilon cost functional is approximated in (3.9) by evaluating it at a finite number of sample points and treating the epsilon cost functional as a constant between the sample points. At each of these sample points the gradient of the epsilon cost functional with respect to $\underline{A}$ is evaluated. Therefore, a large amount of computer storage is needed in order to apply the Epsilon Technique to the torpedo control problem. For twelve basis functions and fifty sample points over 100,000 words of computer storage are needed.

Special techniques had to be used to obtain that much storage on an Univac 1110 computer system. In applying the Epsilon Technique to the torpedo control problem it was determined that 50 sample points were needed for a trajectory of 0.6 seconds duration and even then the solution was not exact. On this basis 500 sample points would be needed for a trajectory of 6.0 seconds duration which would require more computer storage than is presently available in one computer system. Present day computers are unable to handle that amount of computer storage without elaborate, time-consuming techniques. A second problem is that the number of basis functions is limited. For 12 basis functions with a free final endpoint the dimension the system of equations that has to be solved, (3.13), is equal to 77. The number of steps needed to solve the matrix equation is proportional to the cube of its dimension. The more steps that are used increases the truncation and roundoff errors in the computational algorithm. Several cases were tried using 14 basis functions but there was no improvement in the results. Another problem is the slow convergence for the cases where the initial point is fixed. This problem is typical of penalty function techniques and the, see [18], Epsilon Technique is of the penalty function genre. In order to try and increase the speed of convergence the method of steepest descents was used for the first nine iterations and then the Newton-Raphson algorithm was used. Another method was to apply Aiken's method of increasing the speed of convergence of a convergent sequence. Both of these methods didn't give any noticeable improvement in the results.

| No. | Basis Functions |
|---|---|
| 1 | $\sin(i\pi t/t_f)$ |
| 2 | $\sin(i\pi t/t_f)\ \sin(\pi t/t_f)$ |
| 3 | $e^{-\alpha t/t_f}\ \sin(i\pi t/t_f)$ |
| 4 | $e^{-\alpha_1 t/t_f}\ \sin(i\pi t/t_f)$ if i is odd |
| | $e^{-\alpha_2 t/t_f}\ \cos(i\pi t/t_f)$ if i is even |
| 5 | $e^{-\alpha t/t_f}\ \sin(Ut)$ |
| 6 | $e^{-\alpha_1 t/t_f}\ \sin(Ut) + e^{-\alpha_2 t/t_f}\ \cos(Ut)$ |
| 7 | $(t/t_f)^i(t_f-t)$ |

Table 3.1: List of the basis functions that were considered for use with the Epsilon Technique.

TABLE 3.2: Basis functions are type No. 1 with straight initial approximation. Trajectory was 0.3 sec. curve, 0.3 sec. straight. Twelve basis functions were used.

| Coefficients | State Variables | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 3.85E-1 | -1.67 | -4.87E-1 | -3.82 | 6.99E-1 | 6.40E-2 |
| 2 | 1.69E-3 | -6.22E-2 | 3.12E-1 | -6.20E-1 | 3.93E-1 | -2.63E-2 |
| 3 | -1.20E-2 | 2.06E-2 | 1.29E-1 | 3.23E-1 | -1.26E-3 | -1.19E-2 |
| 4 | -9.78E-4 | 3.07E-3 | -2.42E-2 | -7.62E-3 | 2.08E-3 | -1.94E-2 |
| 5 | 1.97E-2 | -3.76E-2 | -9.56E-2 | -1.41E-1 | 7.80E-1 | -8.57E-4 |
| 6 | -8.36E-4 | 4.48E-3 | 1.87E-2 | -1.70E-2 | 3.68E-2 | -2.41E-3 |
| 7 | -2.54E-5 | 2.26E-3 | 6.54E-3 | 6.64E-2 | -1.64E-2 | -1.12E-2 |
| 8 | -3.44E-5 | 4.54E-4 | -6.50E-3 | -6.47E-6 | 2.99E-5 | -3.48E-5 |
| 9 | 3.57E-4 | -6.69E-5 | -1.49E-3 | -4.19E-1 | 2.45E-1 | -2.77E-3 |
| 10 | -2.88E-4 | 1.27E-3 | 4.25E-3 | -3.75E-3 | 9.32E-3 | -5.93E-4 |
| 11 | 5.93E-5 | 5.50E-4 | 1.09E-3 | 2.75E-3 | -9.42E-3 | -2.84E-4 |
| 12 | -5.92E-5 | 1.38E-4 | -2.24E-3 | 4.16E-3 | -1.86E-4 | -1.11E-4 |

TABLE 3.3: Basis functions are type No. 2 with cubic initial approximation. Trajectory was 0.3 sec. curve, 0.3 sec. straight. Twelve basis functions were used.

| Coefficients | State Variables | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1.13E-1 | -4.52E-1 | -4.54E-1 | -4.04 | 8.82E-1 | 7.75E-2 |
| 2 | 4.55E-2 | -8.06E-2 | 1.82E-1 | -1.68 | 6.83E-1 | 1.58E-2 |
| 3 | -5.57E-3 | -1.76E-2 | 8.25E-2 | -7.48E-1 | 3.23E-1 | 2.91E-3 |
| 4 | 4.99E-3 | -1.90E-2 | -1.28E-2 | -7.50E-1 | 2.89E-1 | 5.33E-3 |
| 5 | 1.42E-3 | -1.32E-2 | -1.51E-3 | -6.89E-1 | 2.98E-1 | 2.63E-3 |
| 6 | -4.65E-4 | -3.58E-3 | 2.22E-2 | -4.64E-1 | 2.26E-1 | -1.49E-4 |
| 7 | -3.92E-4 | -2.22E-3 | 8.96E-3 | -2.68E-1 | 1.27E-1 | 3.84E-4 |
| 8 | 1.85E-4 | -3.06E-3 | -3.93E-3 | -2.82E-1 | 1.26E-1 | 9.03E-4 |
| 9 | 1.45E-4 | -1.82E-3 | 6.16E-4 | -2.22E-1 | 1.04E-1 | 3.45E-4 |
| 10 | -103E-4 | -5.20E-4 | 4.44E-2 | -1.66E-1 | 8.24E-2 | -5.97E-5 |
| 11 | -9.85E-5 | -1.69E-4 | 1.81E-3 | -4.35E-2 | 2.13E-2 | 2.03E-5 |
| 12 | 1.03E-4 | -7.93E-4 | -2.93E-3 | -7.38E-2 | 3.23E-2 | 3.03E-4 |

TABLE 3.4: Basis functions are type No. 1 with cubic initial approximation. Trajectory is 0.3 sec. curve, 0.3 sec. straight. Twelve basis functions were used.

| Coefficients | State Variables | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 9.65E-2 | -3.80E-1 | -4.00E-1 | -3.28 | 6.86E-1 | 6.53E-2 |
| 2 | 3.00E-3 | -5.09E-2 | 1.25E-1 | -9.76E-1 | 3.99E-1 | 3.18E-5 |
| 3 | -2.30E-2 | 6.80E-2 | 1.31E-1 | 3.44E-1 | -2.67E-3 | -1.18E-2 |
| 4 | -4.72E-4 | 4.23E-3 | -4.80E-2 | -5.50E-1 | 3.70E-3 | 3.13E-3 |
| 5 | -6.14E-4 | 6.54E-3 | -8.53E-3 | -1.36E-1 | 7.81E-2 | -8.69E-4 |
| 6 | -5.86E-4 | 4.89E-3 | 1.18E-2 | -2.88E-2 | 3.64E-2 | -1.44E-3 |
| 7 | -1.28E-3 | 5.96E-3 | 6.57E-3 | 6.80E-2 | -1.67E-2 | -1.10E-3 |
| 8 | -1.57E-3 | 5.95E-4 | -9.48E-3 | -6.74E-3 | 6.19E-4 | 3.82E-3 |
| 9 | -1.93E-4 | 1.71E-3 | -1.24E-3 | -4.11E-2 | 2.46E-2 | -2.84E-4 |
| 10 | -1.46E-4 | 1.35E-3 | 2.77E-3 | -5.76E-3 | 8.96E-2 | -3.82E-4 |
| 11 | -2.77E-4 | 1.50E-3 | 1.06E-3 | 2.79E-2 | -9.53E-3 | -2.75E-4 |
| 12 | -5.99E-5 | 1.80E-4 | -3.13E-3 | -1.99E-3 | 1.92E-4 | 1.12E-4 |

TABLE 3.5: Basis functions of type No. 1 with straight initial conditions. Trajectory was 0.6 sec. curve. Twelve basis functions were used.

| Coefficients | State Variables | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 6.50E-1 | -2.05 | 6.83E-1 | -2.06 | 5.35E-1 | -3.24E-2 |
| 2 | -9.30E-2 | 9.40E-2 | 1.80E-1 | -2.49E-1 | 1.98E-1 | -1.37E-2 |
| 3 | 2.20E-2 | -4.11E-2 | 4.26E-2 | -8.12E-2 | 9.38E-2 | -6.18E-3 |
| 4 | -1.09E-2 | 2.01E-2 | 1.81E-1 | -2.19E-1 | 5.02E-1 | -3.38E-3 |
| 5 | 4.97E-3 | -6.15E-3 | 5.90E-3 | -1.38E-2 | 2.94E-2 | -1.93E-3 |
| 6 | -3.12E-3 | 6.95E-2 | 3.55E-3 | -4.86E-2 | 1.84E-2 | -1.24E-3 |
| 7 | 1.85E-3 | -1.81E-3 | 1.24E-3 | -4.34E-3 | 1.22E-2 | -8.04E-3 |
| 8 | -1.29E-3 | 3.12E-3 | 1.02E-3 | -1.73E-3 | 8.48E-2 | -5.68E-4 |
| 9 | 8.81E-4 | -7.54E-4 | 3.33E-4 | -1.88E-3 | 6.11E-3 | -4.02E-3 |
| 10 | -6.57E-4 | 1.65E-3 | 3.84E-3 | -7.96E-4 | 4.53E-3 | -3.03E-4 |
| 11 | 4.85E-4 | -3.83E-4 | 1.01E-4 | -9.82E-4 | 3.45E-3 | -2.27E-4 |
| 12 | -3.78E-4 | 9.73E-4 | 1.73E-4 | -4.31E-4 | 2.69E-3 | -1.80E-4 |

TABLE 3.6: Basis functions are of type No. 2 with cubic initial approximation. Trajectory is 0.6 sec. curve. Twelve basis functions were used.

| Coefficients | State Variables | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 8.84E-3 | -2.79E-1 | -2.31E-3 | -2.82 | 7.11E-1 | 4.42E-2 |
| 2 | -6.11E-3 | -5.60E-2 | 5.99E-2 | -1.50 | 5.28E-1 | 1.20E-2 |
| 3 | -5.76E-4 | -3.53E-2 | 2.92E-2 | -1.06 | 4.14E-1 | 6.53E-3 |
| 4 | -1.69E-3 | -1.22E-2 | 2.10E-2 | -7.59E-1 | 3.27E-1 | 2.76E-3 |
| 5 | -1.09E-4 | -9.74E-3 | 1.13E-2 | -5.77E-1 | 2.56E-1 | 1.79E-3 |
| 6 | -6.67E-4 | -3.91E-3 | 8.22E-3 | -4.56E-1 | 2.10E-1 | 9.13E-4 |
| 7 | 2.06E-5 | -3.67E-3 | 4.72E-3 | -3.44E-1 | 1.59E-1 | 6.47E-4 |
| 8 | -3.20E-4 | -1.51E-3 | 3.62E-3 | -2.83E-1 | 1.34E-1 | 3.60E-4 |
| 9 | 3.69E-5 | -1.53E-3 | 2.03E-3 | -1.96E-1 | 9.30E-2 | 2.55E-4 |
| 10 | -1.59E-4 | -6.04E-4 | 1.63E-3 | -1.65E-1 | 7.93E-1 | 1.48E-4 |
| 11 | 2.26E-5 | -5.47E-4 | 7.33E-4 | -8.74E-2 | 4.20E-2 | 8.57E-5 |
| 12 | -6.48E-5 | -2.02E-4 | 6.04E-4 | -7.45E-2 | 3.61E-2 | 5.07E-4 |

TABLE 3.7: Basis functions are type No. 1 with straight initial conditions. Trajectory is 0.3 sec. curve, 2.7 sec. straight. Twelve basis functions were used.

| Coefficients | State Variables | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 3.16E-1 | -4.01 | -9.66E-1 | -5.98E-1 | 4.33E-2 | 1.73E-1 |
| 2 | 3.61E-1 | -1.66 | -6.16E-1 | -9.71E-1 | 8.16E-2 | 7.75E-2 |
| 3 | 2.06E-1 | -8.88E-1 | -3.52E-1 | -1.08 | 1.12E-1 | 4.28E-2 |
| 4 | 1.28E-1 | -5.14E-1 | -2.02E-1 | -1.01 | 1.32E-1 | 2.42E-2 |
| 5 | 7.47E-2 | -3.10E-1 | -9.93E-2 | -8.46E-1 | 1.42E-1 | 1.26E-2 |
| 6 | 4.53E-2 | -1.88E-1 | -3.20E-2 | -6.62E-1 | 1.42E-1 | 5.00E-2 |
| 7 | 2.51E-2 | -1.14E-2 | 1.41E-1 | -4.86E-1 | 1.33E-1 | 9.16E-5 |
| 8 | 1.34E-2 | -6.56E-2 | 4.19E-2 | -3.32E-1 | 1.18E-1 | -2.93E-2 |
| 9 | 5.34E-3 | -3.53E-2 | 5.75E-2 | -2.06E-1 | 9.90E-2 | -4.57E-2 |
| 10 | 9.80E-4 | -1.61E-2 | 6.28E-1 | -1.08E-1 | 7.79E-2 | -5.23E-3 |
| 11 | -1.81E-2 | -4.69E-2 | 6.14E-2 | -3.64E-2 | 5.69E-1 | -5.21E-3 |
| 12 | -2.90E-3 | 1.77E-3 | 5.50E-2 | 1.28E-2 | 3.76E-2 | -4.74E-3 |

TABLE 3.8: Basis functions are of type No. 2 with cubic initial approximation. Trajectory is 0.3 sec. turn, 9.75 sec. straight. Twelve basis functions were used.

| Coefficients | State Variables | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | -7.53 | 2.38E+1 | -5.11E-1 | -1.85E-1 | 9.21E-3 | 2.59E-1 |
| 2 | -2.80E-1 | 1.09E+1 | -8.33E-1 | -3.95E-1 | 2.06E-2 | 2.35E-1 |
| 3 | -3.58 | 5.92 | -9.43E-1 | -5.14E-1 | 2.61E-2 | 2.14E-1 |
| 4 | 8.99E-1 | 4.30 | -1.01 | -7.12E-1 | 3.83E-2 | 1.96E-1 |
| 5 | -2.22 | 2.04 | -9.32E-1 | -7.31E-1 | 3.87E-2 | 1.69E-1 |
| 6 | 1.09 | 1.97 | -9.15E-1 | -8.92E-1 | 5.02E-2 | 1.55E-1 |
| 7 | -1.45 | 6.70E-1 | -7.60E-1 | -7.94E-1 | 4.42E-2 | 1.25E-1 |
| 8 | 9.59E-1 | 9.39E-1 | -7.25E-1 | -9.05E-1 | 5.40E-2 | 1.14E-1 |
| 9 | -8.89E-1 | 1.48E-1 | -5.27E-1 | -6.86E-1 | 4.07E-1 | 8.18E-2 |
| 10 | 6.93E-1 | 4.24E-1 | -4.94E-1 | -7.50E-1 | 4.78E-2 | 7.47E-2 |
| 11 | -4.22E-1 | -1.39E-1 | -2.67E-1 | -4.14E-1 | 2.64E-2 | 4.02E-2 |
| 12 | 3.63E-1 | 1.51E-1 | -2.49E-1 | -4.40E-1 | 3.01E-2 | 3.66E-2 |

TABLE 3.9: Comparison of values for 0.3 sec. curve, 0.3 sec. straight generated by optimal coefficients

| Time | $x_{1_\alpha}$ | $x_{1_\beta}$ | $x_{1_\gamma}$ | $x_{1_\delta}$ | $x_{2_\alpha}$ | $x_{2_\beta}$ | $x_{2_\gamma}$ | $x_{2_\delta}$ |
|---|---|---|---|---|---|---|---|---|
| 0.03 | -997.97 | -997.97 | -997.97 | -997.97 | -8.45E-3 | -9.00E-3 | -9.56E-3 | -7.96E-3 |
| 0.06 | -995.95 | -995.95 | -995.95 | -995.95 | -2.12E-2 | -2.05E-2 | -1.95E-2 | -2.14E-2 |
| 0.09 | -993.92 | -993.92 | -993.92 | -993.92 | -2.32E-2 | -2.36E-2 | -2.40E-2 | -2.32E-2 |
| 0.12 | -991.90 | -991.90 | -991.90 | -991.90 | -3.88E-3 | -4.01E-3 | -4.33E-3 | -3.74E-3 |
| 0.15 | -989.88 | -989.88 | -989.88 | -989.88 | 4.39E-2 | 4.44E-2 | 4.49E-2 | 4.39E-2 |
| 0.18 | -987.86 | -987.86 | -987.86 | -987.86 | 1.25E-1 | 1.25E-1 | 1.25E-1 | 1.25E-1 |
| 0.21 | -985.85 | -985.85 | -985.85 | -985.85 | 2.44E-1 | 2.44E-1 | 2.44E-1 | 2.44E-1 |
| 0.24 | -983.85 | -983.85 | -983.85 | -983.85 | 4.03E-1 | 4.03E-1 | 4.03E-1 | 4.03E-1 |
| 0.27 | -981.86 | -981.86 | -981.86 | -981.86 | 6.03E-1 | 6.03E-1 | 6.03E-1 | 6.03E-1 |
| 0.30 | -979.88 | -979.88 | -979.88 | -979.88 | 8.47E-1 | 8.47E-1 | 8.47E-1 | 8.47E-1 |
| 0.33 | -977.92 | -977.92 | -977.92 | -977.92 | 1.14 | 1.14 | 1.14 | 1.14 |
| 0.36 | -975.97 | -975.97 | -975.97 | -975.97 | 1.49 | 1.49 | 1.49 | 1.49 |
| 0.39 | -974.03 | -974.03 | -974.03 | -974.03 | 1.87 | 1.87 | 1.87 | 1.87 |
| 0.42 | -972.09 | -972.09 | -972.09 | -972.09 | 2.28 | 2.28 | 2.28 | 2.28 |
| 0.45 | -970.17 | -970.17 | -970.17 | -970.17 | 2.71 | 2.71 | 2.71 | 2.71 |
| 0.48 | -968.24 | -968.24 | -968.24 | -968.24 | 3.16 | 3.16 | 3.16 | 3.16 |
| 0.51 | -966.32 | -966.32 | -966.32 | -966.32 | 3.62 | 3.62 | 3.62 | 3.62 |
| 0.54 | -964.40 | -964.40 | -964.40 | -964.40 | 4.10 | 4.10 | 4.10 | 4.10 |
| 0.57 | -962.48 | -962.48 | -962.48 | -962.48 | 4.58 | 4.58 | 4.58 | 4.58 |
| 0.60 | -960.57 | -960.57 | -960.57 | -960.57 | 5.07 | 5.07 | 5.07 | 5.07 |

$\alpha$ - actual value

$\beta$ - value generated by type No. 1 basis function with straight line initial approximation

$\gamma$ - value generated by type No. 1 basis function with cubic initial approximation

$\delta$ - value generated by type No. 2 basis function with cubic initial approximation

TABLE 3.9: continued

| Time | $x_{4_\alpha}$ | $x_{4_\beta}$ | $x_{4_\gamma}$ | $x_{4_\delta}$ | $x_{5_\alpha}$ | $x_{5_\beta}$ | $x_{5_\gamma}$ | $x_{5_\delta}$ |
|---|---|---|---|---|---|---|---|---|
| 0.03 | -8.12E-1 | -8.00E-1 | -8.00E-1 | -6.16E-1 | 3.36E-1 | 3.26E-1 | 3.26E-1 | 2.40E-1 |
| 0.06 | -1.57 | -1.58 | -1.58 | -1.65 | 5.49E-1 | 5.60E-1 | 5.60E-1 | 5.90E-1 |
| 0.09 | -2.26 | -2.27 | -2.27 | -2.28 | 6.85E-1 | 6.85E-1 | 6.85E-1 | 6.93E-1 |
| 0.12 | -2.89 | -2.87 | -2.87 | -2.83 | 7.71E-1 | 7.62E-1 | 7.62E-1 | 7.42E-1 |
| 0.15 | -3.44 | -3.45 | -3.45 | -3.48 | 8.25E-1 | 8.31E-1 | 8.30E-1 | 8.43E-1 |
| 0.18 | -3.94 | -3.95 | -3.95 | -3.95 | 8.60E-1 | 8.66E-1 | 8.66E-1 | 8.67E-1 |
| 0.21 | -4.37 | -4.35 | -4.35 | -4.33 | 8.81E-1 | 8.70E-1 | 8.70E-1 | 8.62E-1 |
| 0.24 | -4.75 | -4.75 | -4.75 | -4.77 | 8.95E-1 | 8.95E-1 | 8.95E-1 | 9.03E-1 |
| 0.27 | -5.09 | -5.14 | -5.14 | -5.13 | 9.02E-1 | 9.25E-1 | 9.25E-1 | 9.22E-1 |
| 0.30 | -5.38 | -5.25 | -5.25 | -5.24 | 9.06E-1 | 8.41E-1 | 8.41E-1 | 8.38E-1 |
| 0.33 | -4.86 | -4.91 | -4.91 | -4.93 | 5.86E-1 | 6.12E-1 | 6.12E-1 | 6.20E-1 |
| 0.36 | -4.36 | -4.35 | -4.25 | -4.34 | 3.81E-1 | 3.75E-1 | 3.75E-1 | 3.69E-1 |
| 0.39 | -3.89 | -3.87 | -3.87 | -3.87 | 2.49E-1 | 2.41E-1 | 2.41E-1 | 2.39E-1 |
| 0.42 | -3.46 | -3.48 | -3.48 | -3.50 | 1.65E-1 | 1.73E-1 | 1.73E-1 | 1.82E-1 |
| 0.45 | -3.07 | -3.07 | -3.07 | -3.06 | 1.10E-1 | 1.10E-1 | 1.10E-1 | 1.03E-1 |
| 0.48 | -2.72 | -2.71 | -2.71 | -2.70 | 7.45E-2 | 6.80E-3 | 6.79E-2 | 6.60E-2 |
| 0.51 | -2.41 | -2.42 | -2.42 | -2.44 | 5.14E-2 | 5.47E-2 | 5.50E-2 | 6.51E-2 |
| 0.54 | -2.13 | -2.14 | -2.13 | -2.12 | 3.61E-2 | 3.97E-2 | 3.96E-2 | 3.22E-2 |
| 0.57 | -1.88 | -1.87 | -1.87 | -1.86 | 2.60E-2 | 2.09E-2 | 2.06E-2 | 1.41E-2 |
| 0.60 | -1.66 | -1.66 | -1.66 | -1.66 | 1.92E-2 | 1.88E-2 | 1.92E-2 | 1.92E-2 |

TABLE 3.10: Comparison of values for 0.6 sec. curve generated by optimal coefficients

| Time | $x_{1_\alpha}$ | $x_{1_\beta}$ | $x_{1_\delta}$ | $x_{2_\alpha}$ | $x_{2_\beta}$ | $x_{2_\delta}$ |
|---|---|---|---|---|---|---|
| 0.03 | -997.97 | -997.97 | -997.97 | -8.45E-3 | -8.65E-3 | -7.95E-3 |
| 0.06 | -995.95 | -995.95 | -995.95 | -2.12E-2 | -2.07E-2 | -2.14E-2 |
| 0.09 | -993.92 | -993.92 | -993.92 | -2.32E-2 | -2.37E-2 | -2.33E-2 |
| 0.12 | -991.90 | -991.90 | -991.90 | -3.88E-3 | -3.73E-2 | -3.67E-3 |
| 0.15 | -989.88 | -989.88 | -989.88 | 4.39E-2 | 4.42E-2 | 4.38E-2 |
| 0.18 | -987.86 | -987.86 | -987.86 | 1.25E-1 | 1.25E-1 | 1.25E-1 |
| 0.21 | -985.85 | -985.85 | -985.85 | 2.44E-1 | 2.44E-1 | 2.44E-1 |
| 0.24 | -983.85 | -983.85 | -983.85 | 4.03E-1 | 4.03E-1 | 4.03E-1 |
| 0.27 | -981.86 | -981.86 | -981.86 | 6.03E-1 | 6.03E-1 | 6.03E-1 |
| 0.30 | -979.88 | -979.88 | -979.88 | 8.47E-1 | 8.47E-1 | 8.47E-1 |
| 0.33 | -977.92 | -977.92 | -977.92 | 1.134 | 1.135 | 1.134 |
| 0.36 | -975.97 | -975.97 | -975.97 | 1.466 | 1.466 | 1.466 |
| 0.39 | -974.03 | -974.03 | -974.03 | 1.843 | 1.844 | 1.844 |
| 0.42 | -972.11 | -972.11 | -972.11 | 2.265 | 2.266 | 2.265 |
| 0.45 | -970.21 | -970.21 | -970.21 | 2.732 | 2.732 | 2.732 |
| 0.48 | -968.33 | -968.33 | -968.33 | 3.244 | 3.244 | 3.244 |
| 0.51 | -966.47 | -966.47 | -966.47 | 3.800 | 3.801 | 3.800 |
| 0.54 | -964.64 | -964.64 | -964.64 | 4.400 | 4.399 | 4.400 |
| 0.57 | -962.82 | -962.82 | -962.82 | 5.043 | 5.044 | 5.043 |
| 0.60 | -961.03 | -961.03 | -961.03 | 5.730 | 5.730 | 5.730 |

$\alpha$ - actual value

$\beta$ - value generated by type No. 1 basis functions with linear initial approximation

$\delta$ - value generated by type No. 2 basis functions with cubic initial approximation

TABLE 3.10: continued

| Time | $x_{4_\alpha}$ | $x_{4_\beta}$ | $x_{4_\delta}$ | $x_{5_\alpha}$ | $x_{5_\beta}$ | $x_{5_\delta}$ |
|---|---|---|---|---|---|---|
| 0.03 | -0.812 | -0.811 | -0.625 | 0.336 | 0.333 | 0.245 |
| 0.06 | -1.569 | -1.570 | -1.656 | 0.549 | 0.555 | 0.592 |
| 0.09 | -2.261 | -2.261 | -2.266 | 0.685 | 0.682 | 0.687 |
| 0.12 | -2.886 | -2.885 | -2.834 | 0.771 | 0.770 | 0.746 |
| 0.15 | -3.443 | -3.443 | -3.485 | 0.825 | 0.828 | 0.846 |
| 0.18 | -3.937 | -3.937 | -3.934 | 0.860 | 0.858 | 0.858 |
| 0.21 | -4.372 | -4.372 | -4.344 | 0.881 | 0.880 | 0.867 |
| 0.24 | -4.755 | -4.755 | -4.784 | 0.895 | 0.896 | 0.909 |
| 0.27 | -5.089 | -5.089 | -5.084 | 0.902 | 0.902 | 0.900 |
| 0.30 | -5.380 | -5.380 | -5.360 | 0.906 | 0.905 | 0.897 |
| 0.33 | -5.633 | -5.634 | -5.657 | 0.908 | 0.910 | 0.920 |
| 0.36 | -5.853 | -5.853 | -5.847 | 0.908 | 0.909 | 0.906 |
| 0.39 | -6.042 | -6.042 | -6.027 | 0.908 | 0.906 | 0.900 |
| 0.42 | -6.206 | -6.206 | -6.226 | 0.906 | 0.907 | 0.916 |
| 0.45 | -6.346 | -6.346 | -6.340 | 0.904 | 0.904 | 0.901 |
| 0.48 | -6.465 | -6.465 | -6.452 | 0.901 | 0.900 | 0.895 |
| 0.51 | -6.567 | -6.567 | -6.586 | 0.898 | 0.899 | 0.908 |
| 0.54 | -6.653 | -6.654 | -6.649 | 0.895 | 0.896 | 0.893 |
| 0.57 | -6.726 | -6.725 | -6.708 | 0.892 | 0.891 | 0.884 |
| 0.60 | -6.786 | -6.786 | -6.786 | 0.889 | 0.889 | 0.889 |

TABLE 3.11: Comparison of values for 0.3 sec. curve, 9.7 sec. straight trajectory

| Time | $x_{1_\alpha}$ | $x_{1_\beta}$ | $x_{1_\delta}$ | $x_{2_\alpha}$ | $x_{2_\beta}$ | $x_{2_\delta}$ |
|---|---|---|---|---|---|---|
| 0.50l | -966.90 | -966.91 | -966.86 | 3.48 | 3.77 | 3.66 |
| 1.00l | -934.99 | -934.92 | -934.94 | 12.11 | 11.64 | 12.02 |
| 1.50l | -903.02 | -903.11 | -903.13 | 21.30 | 21.57 | 21.33 |
| 2.00l | -870.91 | -870.85 | -870.83 | 30.59 | 30.66 | 30.59 |
| 2.50l | -838.70 | -838.68 | -838.67 | 39.92 | 39.70 | 39.92 |
| 3.00l | -806.42 | -806.50 | -806.53 | 49.27 | 49.38 | 49.26 |
| 3.50l | -774.10 | -774.04 | -774.01 | 58.63 | 58.73 | 58.64 |
| 4.00l | -741.73 | -741.70 | -741.71 | 68.01 | 67.86 | 68.02 |
| 4.50l | -709.34 | -709.43 | -709.46 | 77.39 | 77.42 | 77.36 |
| 5.00l | -676.94 | -676.88 | -676.83 | 86.77 | 86.88 | 86.80 |
| 5.50l | -644.52 | -644.47 | -644.51 | 96.16 | 96.06 | 96.17 |
| 6.00l | -612.10 | -612.21 | -612.24 | 105.56 | 105.53 | 105.52 |
| 6.50l | -579.66 | -579.62 | -579.50 | 114.95 | 115.06 | 114.99 |
| 7.00l | -547.23 | -547.13 | -547.23 | 124.35 | 124.29 | 124.35 |
| 7.50l | -514.79 | -514.96 | -515.01 | 133.74 | 133.67 | 133.68 |
| 8.00l | -482.35 | -482.32 | -482.06 | 143.14 | 143.24 | 143.21 |
| 8.50l | -449.91 | -449.69 | -449.96 | 152.53 | 152.55 | 152.53 |
| 9.00l | -417.47 | -417.84 | -417.97 | 161.93 | 161.77 | 161.78 |
| 9.50l | -385.02 | -384.91 | -383.74 | 171.33 | 171.45 | 171.69 |
| 9.95l | -353.73 | -353.23 | -353.22 | 180.54 | 180.54 | 180.53 |

$\alpha$ - actual value

$\beta$ - value generated by type No. 1 basis functions with linear initial approximation

$\delta$ - value generated by type No. 2 basis functions with cubic initial approximation

TABLE 3.11: continued

| Time | $x_{4_\alpha}$ | $x_{4_\beta}$ | $x_{4_\delta}$ | $x_{5_\alpha}$ | $x_{5_\beta}$ | $x_{5_\delta}$ |
|---|---|---|---|---|---|---|
| 0.50l | -2.50 | -2.48 | -9.26E-1 | 5.73E-2 | 2.39E-1 | 5.42E-2 |
| 1.00l | -3.09E-1 | -1.15 | -1.21 | 1.76E-3 | 8.29E-2 | 6.51E-2 |
| 1.50l | -3.74E-2 | 5.85E-1 | 1.38E-1 | 2.10E-4 | -8.28E-2 | -1.83E-2 |
| 2.00l | -4.48E-3 | 2.39E-1 | 2.73E-1 | 2.52E-5 | 6.69E-3 | -1.67E-2 |
| 2.50l | -5.33E-4 | -3.79E-1 | -3.18E-1 | 3.00E-1 | 4.53E-2 | 2.27E-2 |
| 3.00l | -6.31E-5 | 2.51E-1 | 8.38E-2 | 3.55E-7 | 3.35E-2 | 7.24E-3 |
| 3.50l | -7.45E-6 | 1.02E-1 | 1.54E-1 | 4.19E-8 | -1.06E-3 | -1.01E-2 |
| 4.00l | -8.77E-7 | -2.66E-1 | -2.05E-1 | 4.93E-9 | 3.30E-2 | 1.46E-2 |
| 4.50l | -1.03E-7 | 1.08E-1 | 6.75E-2 | 5.80E-10 | -1.46E-2 | -5.39E-3 |
| 5.00l | -1.21E-8 | 1.42E-1 | 1.05E-1 | 6.81E-11 | -1.70E-2 | -7.10E-3 |
| 5.50l | -1.42E-9 | -1.96E-1 | -1.59E-1 | 7.98E-12 | 2.47E-2 | 1.13E-2 |
| 6.00l | -1.66E-10 | 2.09E-2 | 6.31E-2 | 9.36E-13 | -3.23E-3 | -4.72E-3 |
| 6.50l | -1.95E-11 | 1.59E-1 | 7.98E-2 | 1.10E-13 | -1.97E-2 | -5.47E-3 |
| 7.00l | -2.29E-12 | -1.39E-1 | -1.36E-1 | 1.29E-14 | 1.76E-2 | 9.67E-3 |
| 7.50l | -2.68E-13 | -4.18E-2 | 6.16E-2 | 1.51E-15 | 4.92E-3 | -4.49E-3 |
| 8.00l | -3.14E-14 | 1.62E-1 | 6.77E-2 | 1.76E-16 | -2.02E-2 | -4.71E-3 |
| 8.50l | -3.68E-15 | -8.49E-2 | -1.26E-1 | 2.07E-17 | 1.08E-2 | 8.96E-3 |
| 9.00l | -4.30E-16 | -9.06E-2 | 5.29E-2 | 2.42E-18 | -1.12E-2 | -3.79E-3 |
| 9.50l | -5.04E-17 | 1.53E-1 | 1.01E-1 | 2.83E-19 | -1.92E-2 | -7.14E-3 |
| 9.95l | -7.32E-18 | 1.15E-7 | 2.52E-16 | 4.11E-20 | -1.43E-9 | -1.78E-17 |

## Chapter Four: Conjugate Gradient Method

The conjugate gradient method was developed by Hestenes and Stiefel, reference [11], and expanded by Fletcher and Reeves, reference [12], to be a rapidly convergent finite dimensional unconstrained minimization technique. In reference [13], Lasdon, Mitter and Warren expanded the conjugate gradient method to optimal control problems. Pagurek and Woodside, reference [14], further extended the method to handle optimal control problems with bounded control variables. In their paper two separate algorithms were presented. The first is a direct extension of Lasdon's algorithm. The second algorithm uses second order terms and gave improved convergence in several sample cases. Therefore, the second method was used for the torpedo control problem. The discussion here will be confined to presenting the steps of the algorithm for the general optimal control system that has the form of the torpedo control problem. Let the dynamic equation be given by

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x},u) \tag{4.1}$$

where,

$$\underline{x}(0) = \underline{x}_o \ , \quad u \in [-U,U]$$

The cost functional has the form

$$J(t,\underline{x},u) = \phi(t_f,\underline{x}(t_f)) \tag{4.2}$$

The Hamiltonian of the above system is given by

$$H(t,\underline{x},\underline{p},u) = \underline{p}^T\underline{f}(\underline{x},u) \tag{4.3}$$

where $\underline{p}$ is the costate variable defined by

$$\dot{\underline{p}}(t) = -\frac{\partial H(t,\underline{x},\underline{p},u)}{\partial \underline{x}(t)} \ , \ \underline{p}(t_f) = \frac{\partial J(t,\underline{x},u)}{\partial \underline{x}}\bigg|_{t_f} \tag{4.4}$$

Let $\underline{z}(t)$ and $\underline{c}(t)$ be n-dimensional vectors and $s(t)$, $w(t)$ and $e(t)$ be

scalars and let us denote partial derivatives by subscripts, that is $\underline{f}_{\underline{x}}$ means the partial derivative of $\underline{f}$ with respect to $\underline{x}$. Then, the conjugate gradient method is given in the following six steps.

Step 1: Determine an initial control u(t).

Step 2: Solve the following equations forward in time;

$$\dot{\underline{x}} = \underline{f}(\underline{x},u) \ , \ \underline{x}(0) = \underline{x}_o \tag{4.1}$$

$$\dot{\underline{z}} = \underline{f}_{\underline{x}}\underline{z} + \underline{f}_u ws \ , \ \underline{z}(0) = 0 \tag{4.5}$$

Step 3: Solve the following equations backward in time;

$$\dot{\underline{p}} = -H_{\underline{x}} \ , \ \underline{p}(t_f) = J_{\underline{x}}(t_f,\underline{x},u) \tag{4.4}$$

$$\dot{\underline{c}} = -\underline{f}_{\underline{x}}^T\underline{c} - H_{\underline{xx}}\underline{z} - H_{\underline{x}u}ws_i \ , \ \underline{c}(t_f)=J_{\underline{xx}}(t_f,\underline{x},u)\underline{z}(t_f) \tag{4.6}$$

and at the same time evaluate;

$$e_i = H_u \tag{4.7}$$

$$d_i = H_{u\underline{x}}\underline{z} + H_{uu}s_i + \underline{f}_u^T\underline{c} \tag{4.8}$$

$$I_{vs} = \int_0^t f\, we_i d_i \, dt \tag{4.9}$$

$$I_{ss} = \int_0^t f\, ws_i d_i \, dt \tag{4.10}$$

Step 4: Determine a new value of s by

$$s_{i+1} = e_i + \beta_i s_i \tag{4.11}$$

where,

$$\beta_i = -I_{vs}/I_{ss} \ , \ s_o = e_o \ , \beta_o = 0 \tag{4.12}$$

Step 5: Determine a new control u from the equation

$$u = u - \alpha s_i \tag{4.13}$$

where $\alpha$ is determined by a one dimensional search to be the value that minimizes $J(t,\underline{x},u)$. During the search u is truncated to be within the limits of $\pm U$.

Step 6: After defining a new u, w is defined by

$$w = \begin{cases} 0 & \text{if } |u| = U \\ 1 & \text{if } |u| < U \end{cases}$$

Then set i equal to i+1 and go to Step 2.

The method is terminated when Step 5 yields no improvement in the cost functional or when a preset number of iterations have occurred.

For the torpedo control problem the dynamic equation has the form

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}) + \underline{g}(\underline{x})u \tag{4.14}$$

Since the conjugate gradient method is not designed for time-optimal control problems it is only applied to the fixed time torpedo control problem. Therefore, the cost functional is given by

$$J(t,\underline{x},u) = 100(x_1(t_f)-x_{1f})^2 + 100(x_2(t_f) - x_{2f})^2 \tag{4.15}$$

Since the dynamic equation is linear in u, $H_{uu}$ is equal to zero.

The integrations in Step 2 and in Step 3 were originally performed using a four point predictor-corrector method which is known as the Improved Adams or Moulton's method. Several of the cases were rerun using the simple one point Euler method. Due to the small step size there was no noticeable difference in the results. Therefore, in order to save computer time the simple Euler method was used for integrating the dynamic equations. The one dimensional search in Step 5 was done by evaluating the cost functional at 20 equally spaced points and then fitting a quadratic about the minimum of the 20 points. If the minimum occurred on the first point and the value of the cost functional was not reduced, a new search was performed in the interval between the old point and the first point of the old search. If after reducing the region of search several times no improvement is made the algorithm is assumed to have converged.

## Chapter Five: Method of Linearization

The method of linearization about a known trajectory has always been one of the most frequently used methods to solve nonlinear optimal control problems. One advantage of this method is that a feedback control is obtained. This method is only applied to the fixed time torpedo guidance problem.

The trajectory $\underline{x}'(t)$ about which the trajectory is linearized is a straight line between the initial and final end points. Let us define $\underline{z}(t) = \underline{x}(t) - \underline{x}'(t)$. Then, for the torpedo motion equations which were presented previously, the dynamic equation for the linearized state vector is;

$$\dot{\underline{z}}(t) = C\underline{z}(t) + Du(t) \tag{5.1}$$

where,

$$\underline{z}(0) = \underline{z}_o \ , \ \underline{z}(t_f) = 0 \ , \ C = \left.\frac{\partial \underline{f}(\underline{x})}{\partial \underline{x}}\right|_{x'(t)} + \left.\frac{\partial g(\underline{x})u}{\partial \underline{x}}\right|_{x'(t)}$$

$$D = g(\underline{x})\big|_{\underline{x}'(t)}$$

Although in general C and D are time dependent, for the torpedo control problem that we are considering with the initial trajectory being a straight line C and D are independent of time. C and D are given by;

$$C = \begin{bmatrix} 0 & 0 & \cos(x_6') & -\sin(x_6') & 0 & -x_3'\sin(x_6') \\ 0 & 0 & \sin(x_6') & \cos(x_6') & 0 & x_3'\cos(x_6') \\ 0 & 0 & 2c_1x_3' & 0 & 0 & 0 \\ 0 & 0 & 0 & c_4x_3' & c_5x_3' & 0 \\ 0 & 0 & 0 & c_7x_3' & c_8x_3' & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.2}$$

$$D^T = ( 0\ 0\ 0\ c_6 x_3'^2\ c_9 x_3'^2\ 0 ) \tag{5.3}$$

In order to insure that the constraints on the control u are satisfied a penalty function of the integral of u is added to the cost functional and the amount of the penalty is increased until the constraints are satisfied. The cost functional is given by;

$$J(t,\underline{z},u) = \underline{z}^T E \underline{z} + \lambda \int_0^{t_f} u(t)^2\, dt \tag{5.4}$$

where E has non zero values only on the diagonal. $E_{ii}$ is equal to 100 if $x_i(t_f)$ is fixed. Otherwise, $E_{ii}$ is equal to zero. The Hamiltonian of the above system is;

$$H(t,\underline{z},\underline{p},u) = p_o \lambda u^2 + \underline{p}^T C \underline{z} + \underline{p}^T D u \tag{5.5}$$

Letting $p_o = -1/2$ and setting the derivative of the Hamiltonian with respect to u equal to zero we obtain the following relationship for u;

$$u^*(t) = D^T \underline{p} / \lambda \tag{5.6}$$

Hence, the optimal state and costate vectors satisfy the equations;

$$\dot{\underline{z}}^*(t) = C\underline{z}(t) + \frac{1}{\lambda} D D^T \underline{p}(t) \tag{5.7}$$

$$\dot{\underline{p}}^*(t) = -C\underline{p}(t) \tag{5.8}$$

Let us assume that $\underline{p}(t)=R(t)\underline{z}(t)$. Then we have

$$\dot{\underline{p}}(t) = \dot{R}(t)\underline{z}(t) + R(t)\dot{\underline{z}}(t) \tag{5.9}$$

Using (5.7) and (5.8) in (5.9) we obtain

$$-C^T R(t)\underline{z}(t) = \dot{R}(t)\underline{z}(t) + R(t)C\underline{z}(t) + \frac{1}{\lambda} R(t) D D^T R(t)\underline{z}(t) \tag{5.10}$$

Since $\underline{z}(t)$ is arbitrary we must have;

$$\dot{R}(t) = -C^T R(t) - R(t)C - \frac{1}{\lambda} R(t) D D^T R(t) \tag{5.11}$$

$$u^*(t) = \frac{1}{\lambda} D^T R(t) \underline{z}^*(t) \tag{5.12}$$

$$\dot{\underline{z}}^*(t) = (C + \frac{1}{\lambda} D D^T R(t))\underline{z}(t) \tag{5.13}$$

Therefore, the optimal trajectories and controls are obtained by solving (5.11) - (5.13) subject to the endpoint conditions on $\underline{z}(t)$

and $R(t_f) = -E$.

The method of linearization about a known trajectory is used as a way of initializing the trajectory for the Epsilon Technique. First, the fixed time torpedo control problem is solved using the method of linearization for an approximate final time that is less than the actual final time. Then, the method of least squares approximation is used to obtain the optimal coefficients of the Rayleigh-Ritz expansion of the state vector using the standard basis functions. The method of least squares approximation is given in chapter 3. Finally, the minimum time torpedo control problem is solved by the Epsilon Technique using the coefficients obtained above as a start off point.

## Chapter Six: Application of the Maximum Principle

In this chapter the formalism of the maximum principle is applied to the torpedo control problem to obtain the expected form of the optimal controls. Then the Epsilon Technique formulation of the torpedo control problem is given and the Epsilon maximum principle is applied. Since the only differences between the time optimal torpedo control problem and the fixed time torpedo control problem is the addition of the constant $p_o$ in the Hamiltonian of the time optimal problem and the way of initializing the costate vector $\underline{p}(t)$, we will confine our discussion to the time optimal torpedo control problem.

The dynamic equation for the torpedo control problem is

$$\dot{\underline{x}}(t) = \underline{f}(\underline{x}) + \underline{g}(\underline{x})u \tag{6.1}$$

The cost functional is

$$J(t,\underline{x},u) = t_f \tag{6.2}$$

The Hamiltonian of the system is given by

$$H(t,\underline{x},\underline{p},u) = \underline{p}(t)^T\underline{f}(\underline{x}) + \underline{p}(t)^T\underline{g}(\underline{x})u + p_o \tag{6.3}$$

The costate vector $\underline{p}(t)$ satisfies the equation

$$\dot{\underline{p}}(t) = -\frac{\partial H(t,\underline{x},\underline{p},u)}{\partial \underline{x}} \tag{6.4}$$

Now, it is a necessary condition of the maximum principle that the Hamiltonian is maximized for the optimal state vector $\underline{x}^*(t)$ and the optimal control $u^*(t)$. From (6.3) we see that the optimal control must have the form

$$u^*(t) = \text{sgn}(\underline{p}^*(t)^T\underline{g}(t)) \tag{6.5}$$

provided that $\underline{p}^*(t)\underline{g}(t)$ is not equal to zero for any interval of time. If $H_u(t,\underline{x}^*,\underline{p}^*,u^*) = \underline{p}^*(t)^T\underline{g}(t) = 0$ over a non zero interval of time then we have singular controls. For this to be the case we must

have $\dot{H}_u(t,\underline{x}^*,\underline{p}^*,u^*) = 0$. Since (6.1) is autonomous we must have $H(t,\underline{x}^*,\underline{p}^*,u^*) = 0$ for the time optimal case by the maximum principle. Using the above equations we can obtain expressions for $p_4, p_5$ and $p_6$ in terms of $p_1, p_2, p_3$ and $\underline{x}^*$. If we take the above expression for $p_4$ and differentiate it with respect to time and try to equate it with the expression for $\dot{p}_4$ we obtain four relationships that must hold.These relationships are consistent only if $u = x_4 = x_5 = 0$. Therefore, the only possible singular control is $u = 0$ with the torpedo moving in a straight line. Therefore, the expected optimal controls are $\pm U$ and 0.

Now, let us consider the epsilon maximum principle. Define,

$$\underline{p}^{\varepsilon}(t) = \frac{1}{\varepsilon}(\dot{\underline{x}}^{\varepsilon}(t) - \underline{f}(\underline{x}^{\varepsilon}) - \underline{g}(\underline{x}^{\varepsilon})u^{\varepsilon}) \tag{6.6}$$

where $\underline{x}^{\varepsilon}$ and $u^{\varepsilon}$ are the state vector and controls that minimize the epsilon cost functional

$$J_{\varepsilon}(t,\underline{x},u) = J(t,\underline{x},u) + \frac{1}{\varepsilon}\int_0^t f\ ||\underline{x} - \underline{f}(\underline{x}) - \underline{g}(\underline{x})u||^2\ dt \tag{6.7}$$

Then, by the epsilon maximum principle,

$$\lim_{\varepsilon \to 0} \underline{p}^{\varepsilon}(t) = \underline{p}(t) \tag{6.8}$$

Define,

$$H^{\varepsilon}(t,\underline{x}^{\varepsilon},\underline{p}^{\varepsilon},u^{\varepsilon}) = \underline{p}^{\varepsilon}(t)^T(\ \underline{f}(\underline{x}^{\varepsilon}) + \underline{g}(\underline{x}^{\varepsilon})u^{\varepsilon}) \tag{6.9}$$

The epsilon maximum principle states that the epsilon Hamiltonian should be a maximum at the optimal control and state vector. From (6.9) it is clear that we should have

$$u^{\varepsilon} = U\mathrm{sgn}(\underline{p}^{\varepsilon}(t)^T\underline{g}(\underline{x}^{\varepsilon})) \tag{6.10}$$

Taking the derivative of the epsilon Hamiltonian with respect to the control we have;

$$H_u^{\varepsilon}(t,\underline{x}^{\varepsilon},\underline{p}^{\varepsilon},u) = p_4 g_4(\underline{x}^{\varepsilon}) + p_5 g_5(\underline{x}^{\varepsilon}) = 0 \tag{6.11}$$

Using (6.6) in (6.11) and gathering terms we obtain

$$u^{\varepsilon} = \frac{(\dot{x}_4^{\varepsilon} - f_4)g_4 + (\dot{x}_5^{\varepsilon} - f_5)g_5}{g_4^2 + g_5^2} \tag{6.12}$$

The above expression for u is exactly what was obtained in chapter 3 by minimizing $I(\underline{x},u)$.

Since the maximum principle gives the form of the optimal controls to be either +U or 0, the Epsilon Technique and the conjugate gradient method were tried with the controls restricted to be those given by the maximum principle. The results of limiting the control in this fashion will be given in the next chapter.

## Chapter Seven: Results

The results of applying the Epsilon Technique, the conjugate gradient method and the method of linearization about a known trajectory to the torpedo control problem are presented in this chapter. All of the above methods were applied to the fixed time torpedo control problem. Only the Epsilon Technique is applied to the time-optimal torpedo control problem. There are two ways of using the Epsilon Technique; one is the standard way of assuming a straight line between the endpoints as the initial trajectory of the state variables and the second way is to first solve a fixed time torpedo control problem to obtain an initial estimate for the trajectory of the state variables. The arrangement of this chapter is as follows. First, definitions will be given of several terms that will be used throughout the text. Then the fixed time torpedo control problem will be discussed. The last part of the chapter is devoted to the time-optimal torpedo control problem.

In the torpedo control problem the state vector is composed of six variables; the position of the torpedo in the inertial coordinate system, the velocity of the torpedo in the body fixed coordinate system and the angular velocity and heading of the torpedo. Only the first two components of the state vector, those referring to the position of the torpedo, are always held fixed at the initial and final endpoints. By the term fixed endpoint we mean that the last four components of the state vector are held constant and by the term free endpoint we mean that the last four components of the state vector are allowed to vary in computing the optimal trajectories. The term x-trajectory refers to the trajectory of the state vector generated by the optimization method. The term

u-trajectory refers to the trajectory of the state variables generated from the controls given by the optimization method. A curve is defined to be the trajectory generated by having the rudders of the torpedo at their maximum value and a straight trajectory is the trajectory generated by having the rudders with no deflection at all. The optimization methods that are used are referred to by the following abbreviations:

SE - the standard Epsilon Technique

ME - the Epsilon Technique with the control restricted to be either ±U or 0.

CG - the conjugate gradient method

MCG - the CG method with the contols the same as the ME method.

LIN - the method of linearization about a known trajectory between the endpoints.

LSE - the method of using the output of the LIN method to initialize the trajectory of the state vector for the SE method.

From the results of applying the maximum principle to the torpedo control problem one has that the optimal controls should be either ±U or 0. Therefore, several cases were chosen for comparison where the final endpoint was obtained by a curve or by a curve followed by a straight trajectory. In this way one can obtain an estimate of how well the various optimization algorithms work in solving the torpedo control problem. The cases that were chosen are:

case A - an 0.3 second curve.

case B - an 0.6 second curve.

case C - an 0.3 second curve followed by an 0.3 second straight trajectory.

case D - an 3.0 second curve.

case E - an 0.5 second curve followed by an 2.5 second straight trajectory.

Now the results of the fixed time torpedo control problem are presented. Each case will be discussed separately. Then the results will be summarized for general conclusions. Table 7.1 and Figures 7.1-7.8 contain the results for case A. As could be expected, the ME method gives an x-trajectory and a u-trajectory which are exactly the same as the original trajectory from which the endpoints were obtained. The SE method gives a very good approximation but the controls take too long to reach the maximum and hence the u-trajectory does not reach the desired final endpoint. The CG method gives a good trajectory however the control changes to +U from -U at the end of the run and this seems to only affect the heading at the end of the run and not the position. Several cases were run with the LIN method. For all of the cases the controls decreased to zero and the resulting trajectories didn't develop the proper curvature. Hence, the LIN method gives the worst results for this case.

In addition to comparing the different optimization algorithms, case B is used to compare different ways of implementing the algorithms. The results are presented in Table 7.2 and Figures 7.9 - 7.22. There are three separate cases that are considered; the torpedo initially heading straight, the torpedo initially in a curve and the torpedo initially heading straight with the aimpoint being the head of the torpedo instead of the center of buoyancy of the torpedo. First, we discuss the case of the torpedo initially heading straight.In the SE

method the buildup of the angular velocity is gradual and since the controls depend upon the angular velocity the optimal controls are not reached immediately. Therefore, even though the x-trajectory is almost exact the u-trajectory is slightly flatter and doesn't reach the final endpoint or the desired heading. If the final endpoint is fixed the controls reach the maximum value sooner and both the x-trajectory and the u-trajectory are closer to the optimal trajectory. The CG method has good results for the position of the torpedo but the controls change sign abruptly at the end of the run and the heading is slightly off. If the final endpoint is fixed the control stays the same for the entire run and the result is the exact trajectory. The CG method contains several differential equations that are solved numerically. Case B was used to compare solving these equations by a four point predictor-corrector routine known as Moulton's method with solving these equations by the simple Euler's method. As can be seen in Table 7.2 and Figures 7.15 and 7.16 the results are nearly identical. Therefore, the simple Euler's method was used in all further runs with the CG method to save computer time. The ME method gives the exact trajectory. The LIN method gives the poorest results due to the control decreasing in absolute value to zero. For the cases where the torpedo was initially in a curve all of the methods tried gave very good results. This case was used to see if a noticeable improvement could be seen in the way we applied the CG method over the simpler way of applying the CG method. No improvement was noticed but since Pagurek and Woodside showed some improvement by using the more complicated method we continued to use the second order CG method. When we changed the aimpoint of the torpedo to be its head

instead of its center of buoyancy we found that the SE method gave almost exactly the same results as the previous case where the aimpoint is the torpedo's center of buoyancy. However, the CG method gives the exact trajectory and controls. This seems to indicate that the trajectory of the state variables are unique only when the aimpoint of the torpedo is its head and not its center of buoyancy.

For case C the results are given in Table 7.3 and Figures 7.23 - 7.32. The CG method gives a good estimate of the final position but the final heading is slightly larger than desired. The controls are always greater in absolute value than 0.15, that is the trajectory is more of a continuous curve than a curve followed by a straight trajectory. The MCG method gives the same results as the CG method gave in case B. The SE method gives a good approximation of the x-trajectory but the controls are too small and the u-trajectory is not close to the x-trajectory. The ME method gives an x-trajectory that is close to the desired final position of the torpedo but the trajectory is even more curved than the trajectory from the CG method. The u-trajectory is the same as the exact trajectory of case B. The LIN method gives an u-trajectory that is almost identical to the x-trajectory. The final heading is about 0.05 radians less than the desired heading and the final point is about one foot away from the desired final position. The best x-trajectory is obtained by the SE method and the best u-trajectory is obtained by the CG and LIN methods.

Table 7.4 and Figures 7.33 - 7.37 contain the results for case D. The CG and the ME methods give very good results for both the x-trajectory and the u-trajectory. The SE method gives a good approximation of

the x-trajectory but due to the slowness of building up a large angular velocity the controls take too long to reach the stops and u-trajectory is not close to the x-trajectory. The LIN method gives very poor results when R is held constant and not much better results when R is exact. The x-trajectory in Figure 7.36 exhibits too sharp of a turn in the middle of the trajectory. Therefore, the LIN method does not give good results for cases of long periods of turning.

In case E the time interval is the same as in case D but the amount of turning is much less. The results are given in Table 7.5 and Figures 7.38 - 7.43. The head of the torpedo is the aimpoint except for the LIN method. The SE method gives an x-trajectory that is more like a continuous curve than like the exact trajectory and its final position is close to the desired final position. The u-trajectory is almost a straight line along the X-axis. The CG method didn't converge at all. When the initial trajectory was given to be a 0.4 second curve followed by 2.6 secs. of straight trajectory the CG method converged in one step to almost the same controls. Instead of no rudder deflection for the last 2.6 seconds a small deflection was given, just enough to almost reach the desired final point with a slightly larger heading. The ME method yields an x-trajectory that turns slowly and doesn't reach the final point or heading and an u-trajectory that turns too much and misses the desired final position with too large of a heading. The x-trajectory and the u-trajectory of the LIN method are very close to one another. The final position is close to the desired position only the final heading is larger due to the slower turnrate and hence longer time before the torpedo starts travelling in a straight line.

The LIN method gives the best results for case E.

From the above results it seems that one can formulate the rule that for the fixed time torpedo control problem the best results are obtained by the ME method or the MCG method if the torpedo is always in a maximum curve but if the trajectory includes periods of straight trajectories then the LIN method is the best one to use. We used the LIN method for longer periods of time but since these cases were used with the LSE method for the time-optimal torpedo control problem the results will be presented later in this chapter.

Now the results for the time-optimal torpedo control problem are presented. The only method which is directly applicable to time-optimal problems is the Epsilon Technique. Therefore, the only two methods used in the time-optimal torpedo control problem are the SE method and the LSE method. Originally, the geometry of the problem was as follows; the torpedo is located at (-1000,0) in the inertial coordinate system heading straight down the X-axis and the target is located at the origin heading straight up the Y-axis.The aimpoint was the point of intersection. For these runs the transformation of the state equations changing the desired endpoint from a point on a known curve to the origin as given in chapter two was used. A few of the cases are presented in Table 7.6 and in Figures 7.44 - 7.47. If both the initial and final endpoints are free then converges in very few steps to the torpedo heading straight to the intersection point. If the initial endpoint is fixed then the SE method gives good results only if the target is stationary and the torpedo is facing the target. When the target was moving the SE method generated a good x-trajectory as in Figure 7.45 but the u-trajectory

barely leaves the X-axis. Therefore, in an attempt to understand why the SE method was failing to generate the proper controls to coincide with the x-trajectory cases A - D were tried.

The results for case A are given in Table 7.7 and in Figures 7.48 - 7.53. Initially, the final endpoint was fixed. We found that it is necessary to increase the number of sample points in order to obtain a good x-trajectory with reasonable controls. When the final endpoint is allowed to be free the x-trajectory is almost the same as when the final endpoint is fixed but it takes much longer for the angular velocity to increase and the controls to reach the proper value. The ME method gives a very close approximation of the exact trajectory with its x-trajectory and the controls and u-trajectory are exact. The LSE method gives a good approximation and is slightly better than the SE method in that the final heading is closer to optimal. The SE method slightly underestimates the optimal time and the LSE method slightly overestimates the optimal time. Comparing these results with those from the fixed time torpedo guidance problem we see that the SE method and the ME method give the same results.

In Table 7.8 and Figures 7.54 - 7.63 the results for case B are presented. The results for case B are similar to the results of case A. For the SE method the fixed final endpoint case gives better results than the free final endpoint case. If the torpedo is initially in a curve then the SE method and the ME method give exact controls and almost exact x-trajectories. For the case of free final endpoint the method proposed by Hewett, in reference [10], of including the second order terms in the Newton-Raphson step was tried. There was no improvement in the results.

Another modification that we tried was letting $x_3$, the velocity of the torpedo parallel to its length, be constant. For this case the system is known to be controllable and an optimal control is known to exist. The results are very close to the cases run with no constraint on $x_3$ and there is no improvement in the value of the cost functional or the number of steps needed to converge. A case was run where the coefficients of Rayleigh-Ritz expansion of the state vector were initiallized to their optimal values but the final time was not correct. The x-trajectory is identical with the previous case but the controls and the u-trajectory are much closer to the optimal controls and trajectory. The LSE method gives very close agreement between the u-trajectory and the x-trajectory and both of these trajectories are very close to the x-trajectory given by the SE method.

The results for case C are given in Table 7.9 and Figures 7.64 - 7.72. For the SE method all of the x-trajectories are close to optimal with the free final endpoint case having a slightly lower heading than desired and the fixed final endpoint case having a larger heading than desired at the end of the run. The only case where good controls were obtained was when the coefficients were initiallized to the optimal values. In one case the method of steepest descents was used for the first nine iterations but no improvement was obtained. The LSE method has good agreement between the x-trajectory and the u-trajectory. The final heading is slightly larger than the desired heading and the controls are the best obtained. The ME method gives the same controls as for case B.

The results for case D are presented in Table 7.10 and Figures 7.73

- 7.78. The SE method gives a good approximation of the exact trajectory with the x-trajectory but the u-trajectory is not very close to the x-trajectory as can be seen in Figure 7.74. Increasing the number of sample points doesn't give any improvement in the results. This is probably because the number of sample points would have to be increased by several hundred points before any improvement would be noticed. The ME method gives perfect controls but the final time was too short and the x-trajectory is not good. The LSE method gives a good approximation of the controls, the x-trajectory and the u-trajectory.

From the above discussion we see that the LSE method gives the best results for the general time-optimal torpedo control problem. Therefore, the LSE method was applied to several cases with longer ranges. The cases are; case E with the initial estimate of time being slightly less than exact and with the initial estimate of time being exact, the target being the origin with the torpedo located at (-500,0) with initial headings of 0.1, 0.2, 0.3, 0.4 radians and with the torpedo located at (-750,0) and (-1000,0) with an initial heading of 0.1 radians. The results are given in Table 7.11 and Figures 7.79 - 7.94. The results of the LIN method applied to the fixed time torpedo control problem are presented along with the results of the LSE method applied to the time-optimal torpedo control problem. As can be seen in Figures 7.79 - 7.94 the SE method applied after the LIN method did little more than increase the final time to be sufficient for the state vector to reach the final endpoint. The LIN method gives a trajectory that reaches the final endpoint but it is not the optimal trajectory of a maximum turn until the torpedo is facing the aimpoint and then a straight trajectory

to the final endpoint. It is the feeling of the author that the reason for no improvement is the inability to use a sufficient number of sample points in the SE method. The LSE method is an improvement over applying the SE method alone in that the SE method didn't generate controls that guided the torpedo to the desired final endpoint.

TABLE 7.1: Case A: Fixed-Time Torpedo Control Problem

| Method<br>Comments | Exact | CG | SE | ME | LIN<br>$\lambda$=1000 | LIN<br>$\lambda$=5000<br>R Const. | LIN<br>$\lambda$=3000 | LIN<br>$\lambda$=3000<br>R Const. |
|---|---|---|---|---|---|---|---|---|
| Figures | 7.1 | 7.2 | 7.3 | 7.4,7.1 | 7.5 | 7.6 | 7.7 | 7.8 |
| J | | 0.30006 | 3.3026 | 8.8241 | | | | |
| De/$\varepsilon$ | | | 2.8413 | 8.8241 | | | | |
| $x_1^*$ | -979.89 | -979.89 | -979.84 | -979.89 | -980.07 | -980.02 | -980.03 | -980.02 |
| $x_2^*$ | 0.85414 | 0.8539 | 0.80916 | 0.85574 | 0.359 | 0.4496 | 0.60294 | 0.52204 |
| $x_6^*$ | 0.21533 | 0.20160 | 0.19419 | 0.21498 | 0.055524 | 0.07178 | 0.09210 | 0.07155 |
| $x_1^u$ | | | -979.84 | -979.89 | -979.91 | -979.92 | -979.94 | -979.93 |
| $x_2^u$ | | | 0.59493 | 0.8468 | 0.362 | 0.44995 | 0.59821 | 0.52058 |
| $x_6^u$ | | | 0.17767 | 0.21531 | 0.055 | 0.071103 | 0.09125 | 0.07092 |
| $u^1$ | -U | -U,i$\leq$246 | -0.145,i=1 | -U | -0.165,i=10 | -0.204,10 | -0.273,10 | -0.270,10 |
| | | -0.22,i=251 | -0.2,i-15 | | -0.100,150 | -0.103,190 | -0.203,100 | -0.206,70 |
| | | 0.04,i=271 | -0.26,i=31 | | -0.012,390 | -0.0102,450 | -0.104,240 | -0.102,190 |
| | | 0.11,i=281 | | | 0.001,450 | 0.0007,500 | -0.012,410 | -0.0028,380 |
| | | U,i=301 | | | 0.0085,530 | 0.0167,i=600 | -0.0016,440 | 0.012,430 |
| | | | | | 0.0025,590 | | 0.014,530 | 0.037,600 |
| | | | | | | | 0.0004,600 | 0.0327,600 |

1) (a,b) a -value of control variable

b - sample number to which a corresponds

Values vary linearly between points

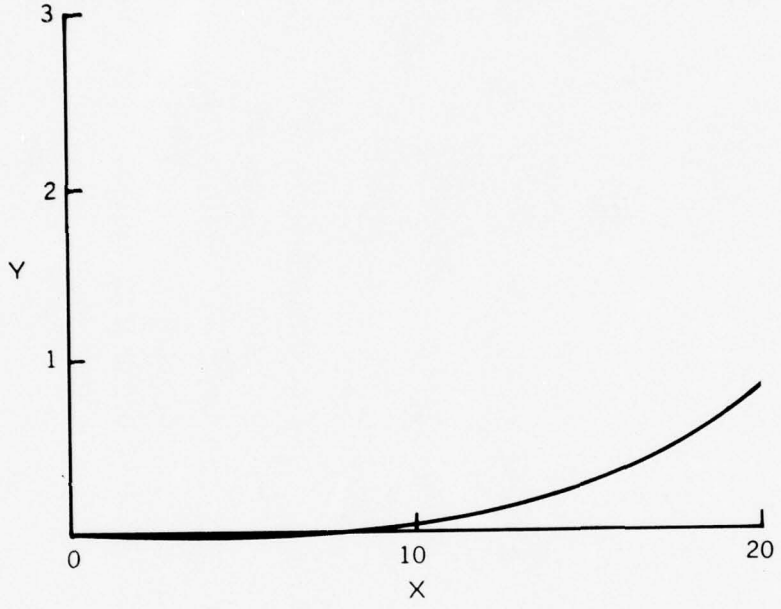



Figure 7.1. Fixed-Time Torpedo Control Problem. Case A: Exact Torpedo Trajectory; ME Method, u-Trajectory.

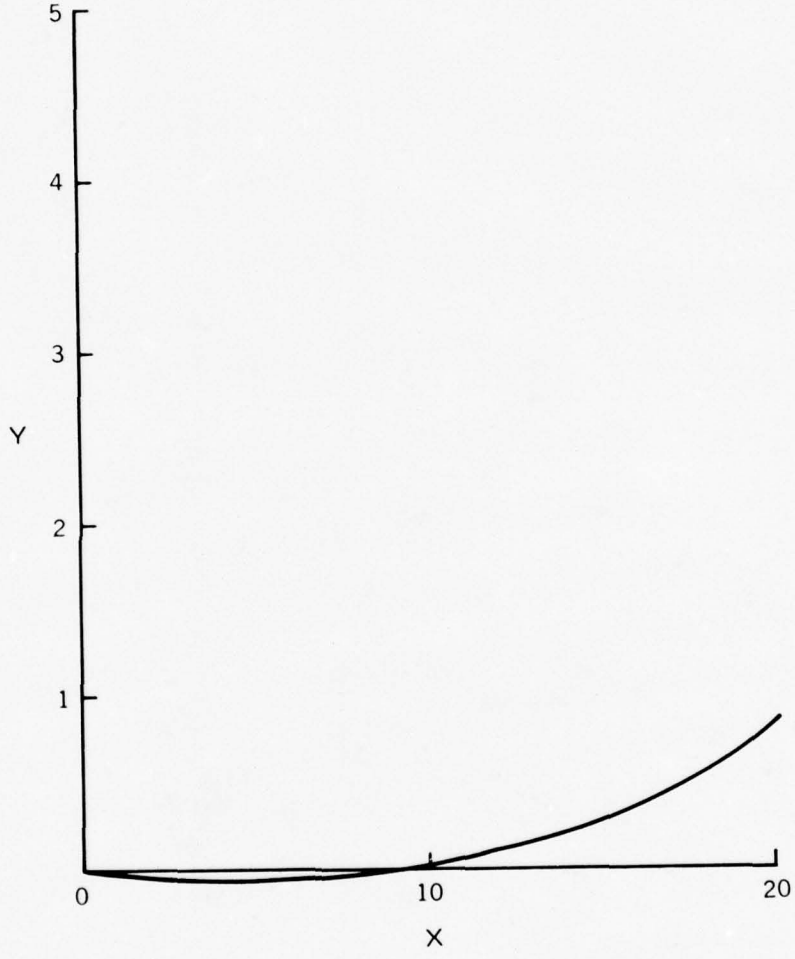



Figure 7.2. Fixed-Time Torpedo Control Problem. Case A: CG Method.

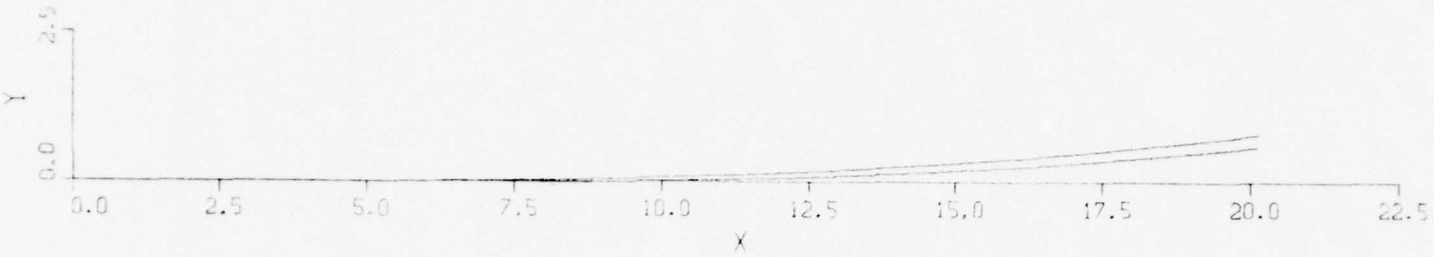



Figure 7.3. Fixed-Time Torpedo Control Problem. Case A: SE Method, x-Trajectory.

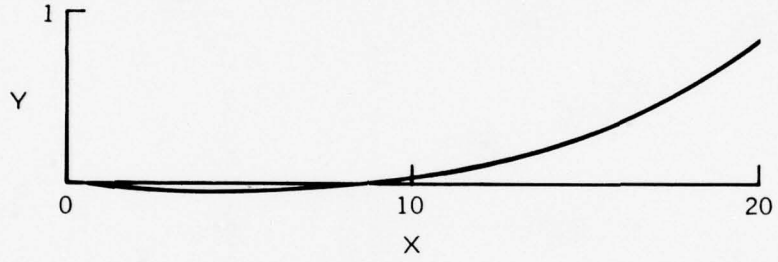



Figure 7.4 Fixed-Time Torpedo Control Problem. Case A: ME Method, x-Trajectory.

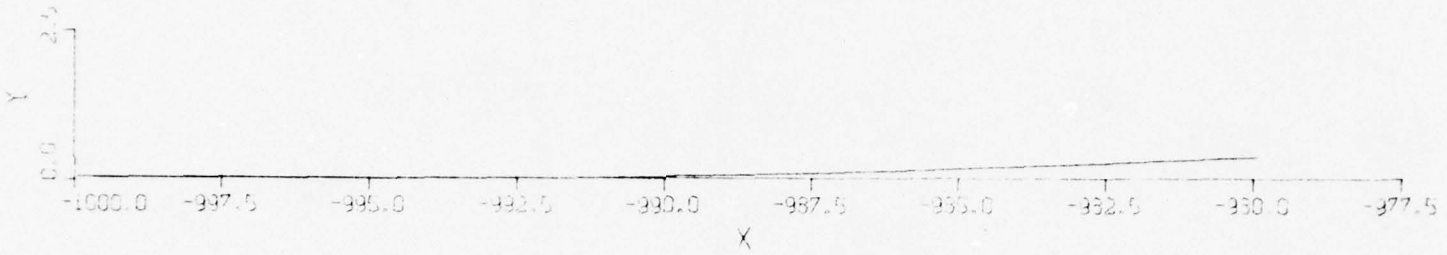



Figure 7.5. Fixed-Time Torpedo Control Problem. Case A: LIN Method, $\lambda = 10{,}000$.

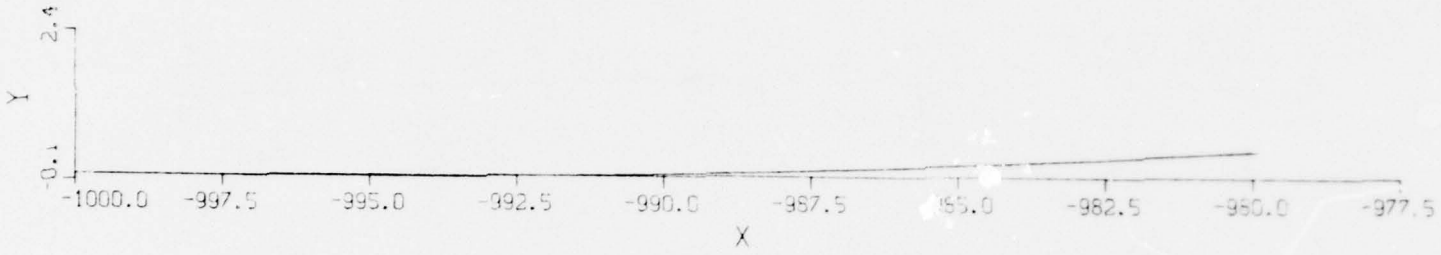



Figure 7.6. Fixed-Time Torpedo Control Problem. Case A: LIN Method, $\lambda = 5000$, R Constant.

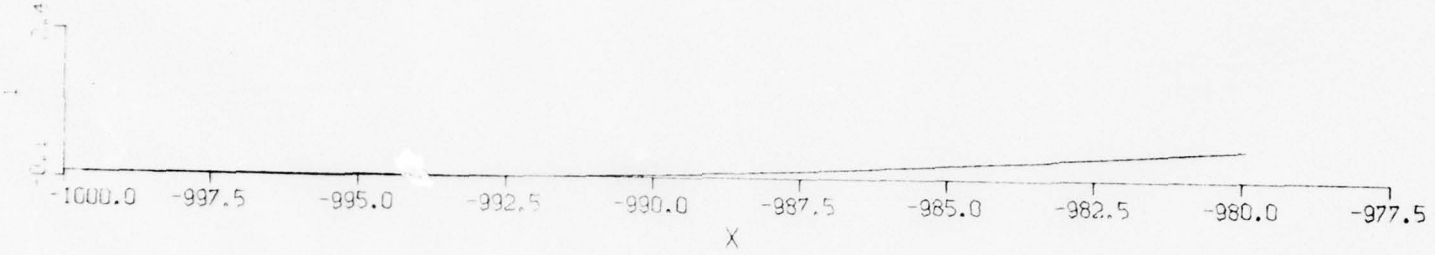



Figure 7.7. Fixed-Time Torpedo Control Problem. Case A: LIN Method, $\lambda = 3000$.

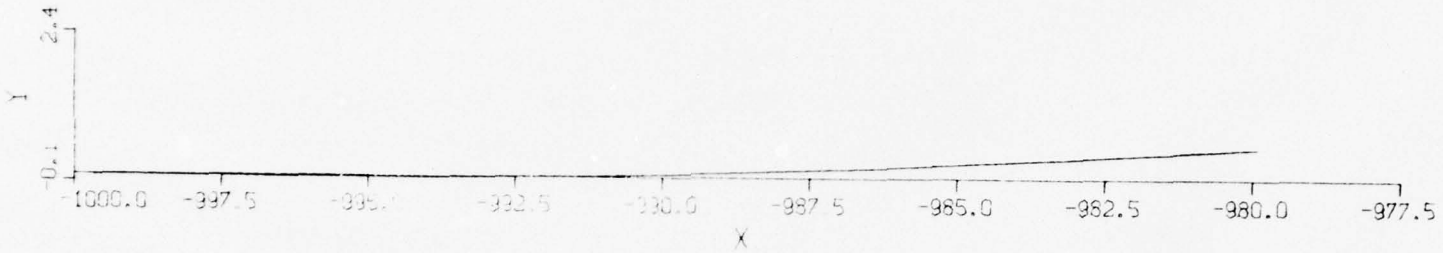



Figure 7.8. Fixed-Time Torpedo Control Problem. Case A: LIN Method, $\lambda = 3000$, R Constant.

TABLE 7.2: Case B Results

| Method<br>Comments<br>Figures | Exact<br><br>7.9 | SE<br><br>7.10,7.11 | SE<br>ffe<br>7.12,7.13 | ME<br><br>7.14 | CG<br>Euler int.<br>7.15 | CG<br>Moulton's Int.<br>7.16 | LIN<br>$\lambda$=6000<br>7.17 | CG<br>ffe<br>7.9 |
|---|---|---|---|---|---|---|---|---|
| J | | 43.816 | 155.56 | 66.996 | 0.61344 | 0.61598 | | 0.6000 |
| De/$\varepsilon$ | | 25.764 | 155.56 | 66.686 | | | | |
| $x_1^*$ | -961.03 | -960.55 | -960.98 | -961.03 | -961.01 | -961.03 | -961.005 | -961.03 |
| $x_2^*$ | 5.71329 | 5.7593 | 5.7534 | 5.7431 | 5.7156 | 5.7488 | 4.077 | 5.71329 |
| $x_6^*$ | 0.4859 | 0.45723 | 0.4868 | 0.4815 | 0.4305 | 0.42876 | 0.23176 | 0.4859 |
| $x_1^u$ | | -960.71 | -961.36 | | | | -960.42 | |
| $x_2^u$ | | 3.3546 | 4.9824 | | | | 4.107 | |
| $x_6^u$ | | 0.35704 | 0.45515 | | | | 0.229 | |
| u | -U | -0.09,1 | -0.15,1 | -U | -U,261 | -U,261 | -0.274,10 | -U, |
| | | -0.2,23 | -0.21,8 | | U,266 | U,266 | -0.103,290 | |
| | | -U,35 | -0.26,11 | | | | -0.012,480 | |
| | | | | | | | -0.0027,590<br>-0.0003,600 | |

ffe - fixed final endpoint

IIC - torpedo initially in a curve

AHOT - aimpoint is head of torpedo

TABLE 7.2: continued

| Method<br>Comments | Exact<br>initially<br>in curve | ME<br>initially<br>in curve | CG<br>IIC | CG<br>IIC<br>standard<br>method | SE<br>AHOT | CG<br>AHOT |
|---|---|---|---|---|---|---|
| Figures | 7.18 | 7.19 | 7.20 | 7.21 | 7.22 | 7.9 |
| J | | 3.7651 | 0.66214 | 0.64 | 52.769 | 0.600 |
| De/ε | | 3.7644 | | | 31.308 | |
| $x_1^*$ | -967.66 | -967.66 | -967.63 | -967.64 | 960.6 | -961.03 |
| $x_2^*$ | 3.6621 | 3.6606 | 3.6584 | 3.67425 | 5.8309 | 5.71329 |
| $x_6^*$ | 0.45513 | 0.4531 | 0.40087 | 0.4016 | 0.47925 | 0.4859 |
| $x_1^u$ | | -967.66 | | | -960.199 | |
| $x_2^u$ | | 3.6743 | | | 3.468 | |
| $x_6^u$ | | 0.45512 | | | 0.36212 | |
| u | | -U | -U,251 | -U,251 | -0.09,1 | -U |
| | | | U,256 | U,256 | -0.2,23 | |
| | | | | | -U,35 | |

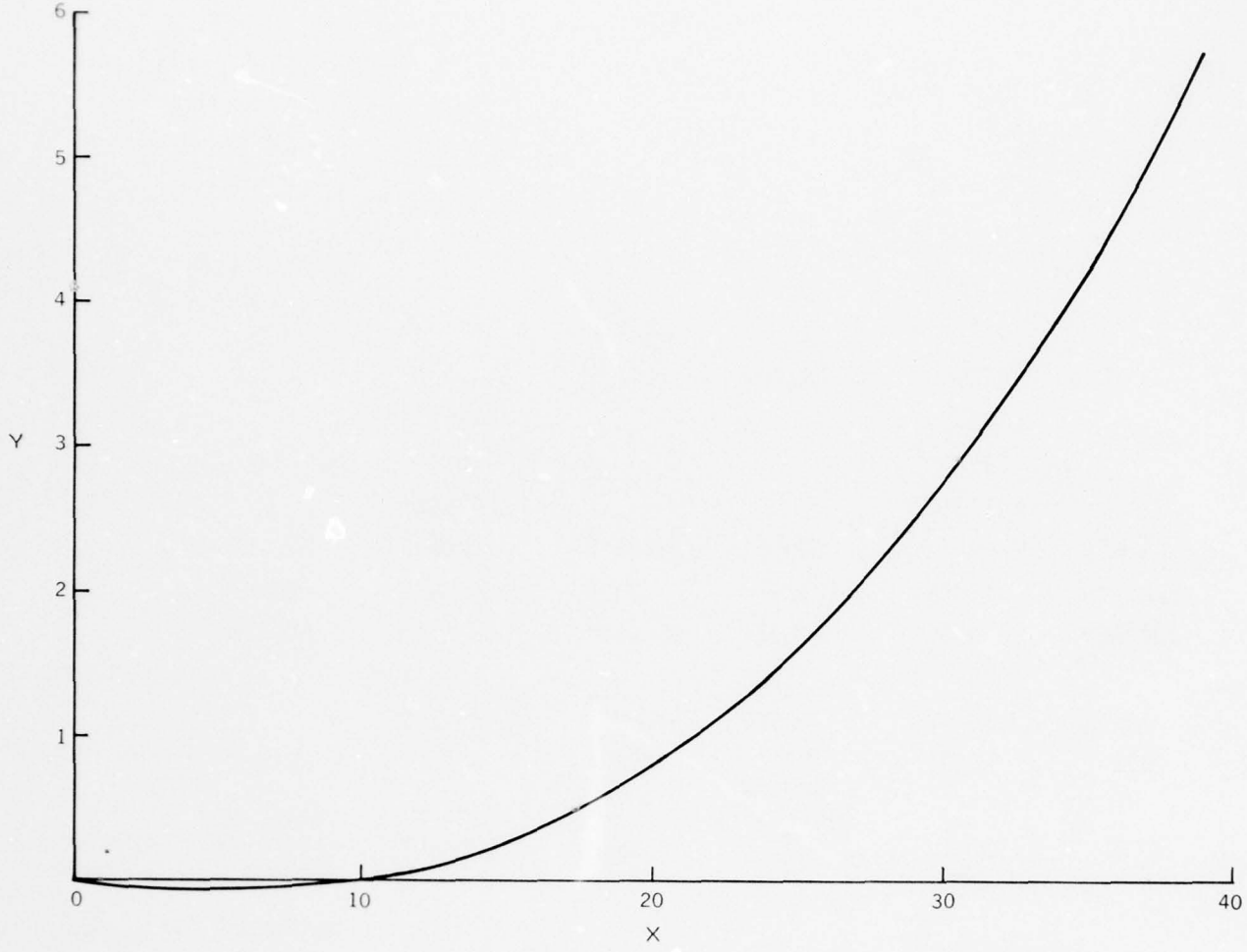



Figure 7.9. Fixed-Time Torpedo Control Problem. Case B: Exact Torpedo Trajectory: ME Method, u-Trajectory: CG Method, Final Endpoint Fixed and CG Method, Aimpoint Head of the Torpedo.

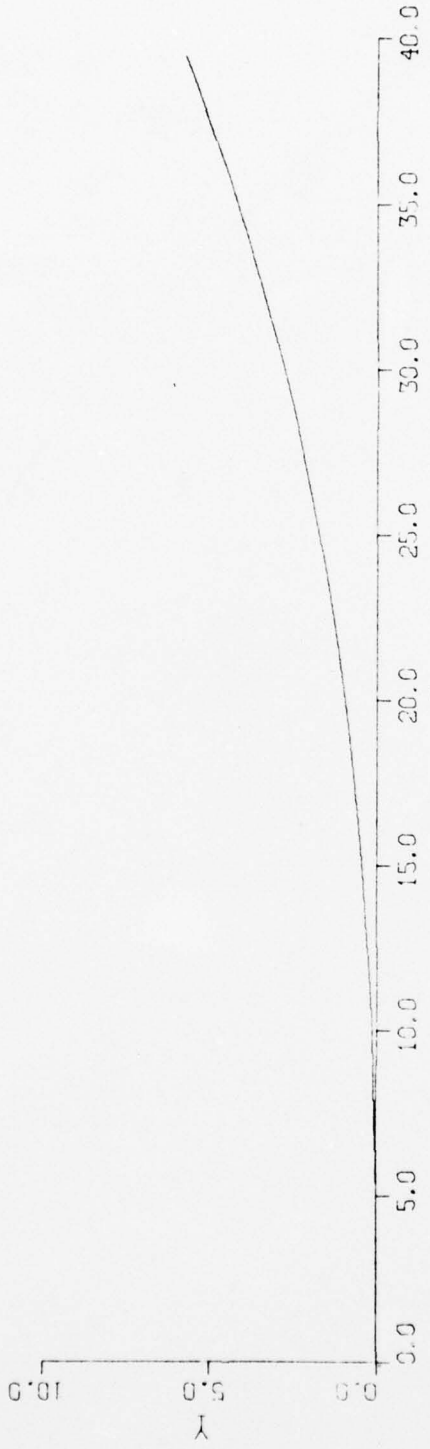



Figure 7.10. Fixed-Time Torpedo Control Problem. Case B: SE Method, x-Trajectory.

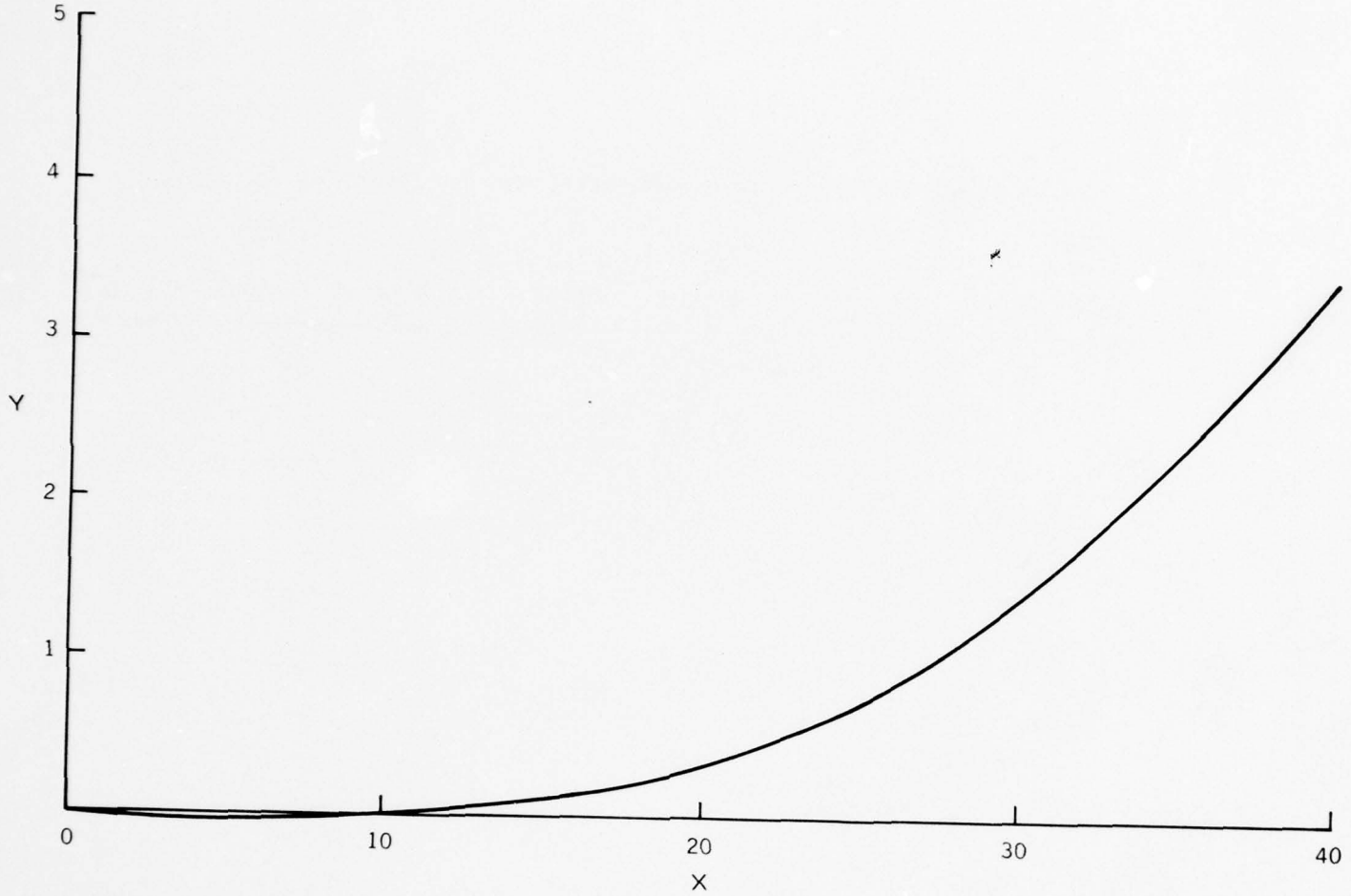



Figure 7.11. Fixed-Time Torpedo Control Problem. Case B: SE Method, u-Trajectory.

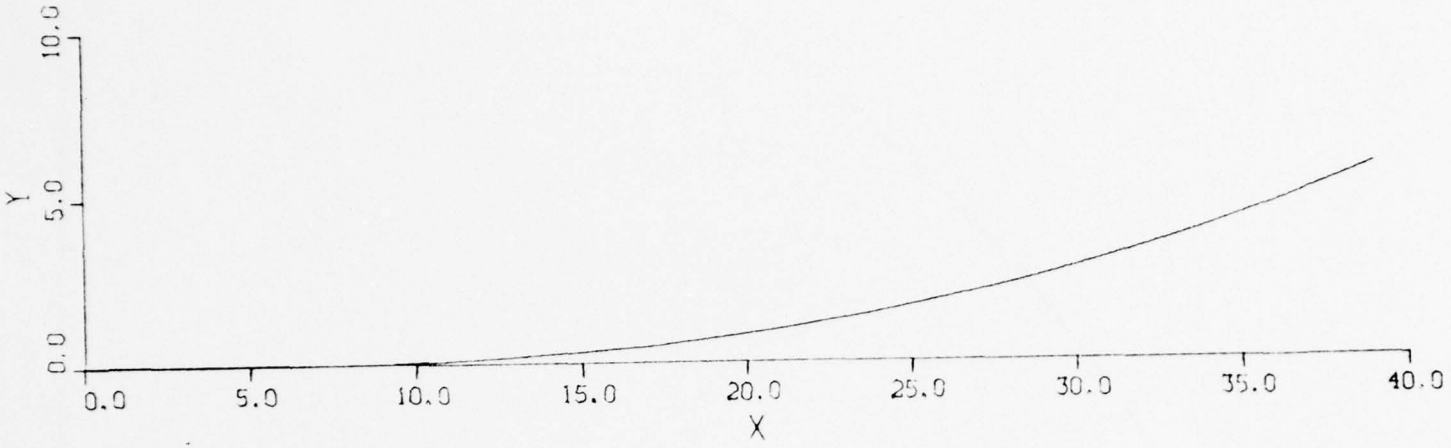



Figure 7.12. Fixed-Time Torpedo Control Problem. Case B: SE Method, Fixed Final Endpoint, x-Trajectory.

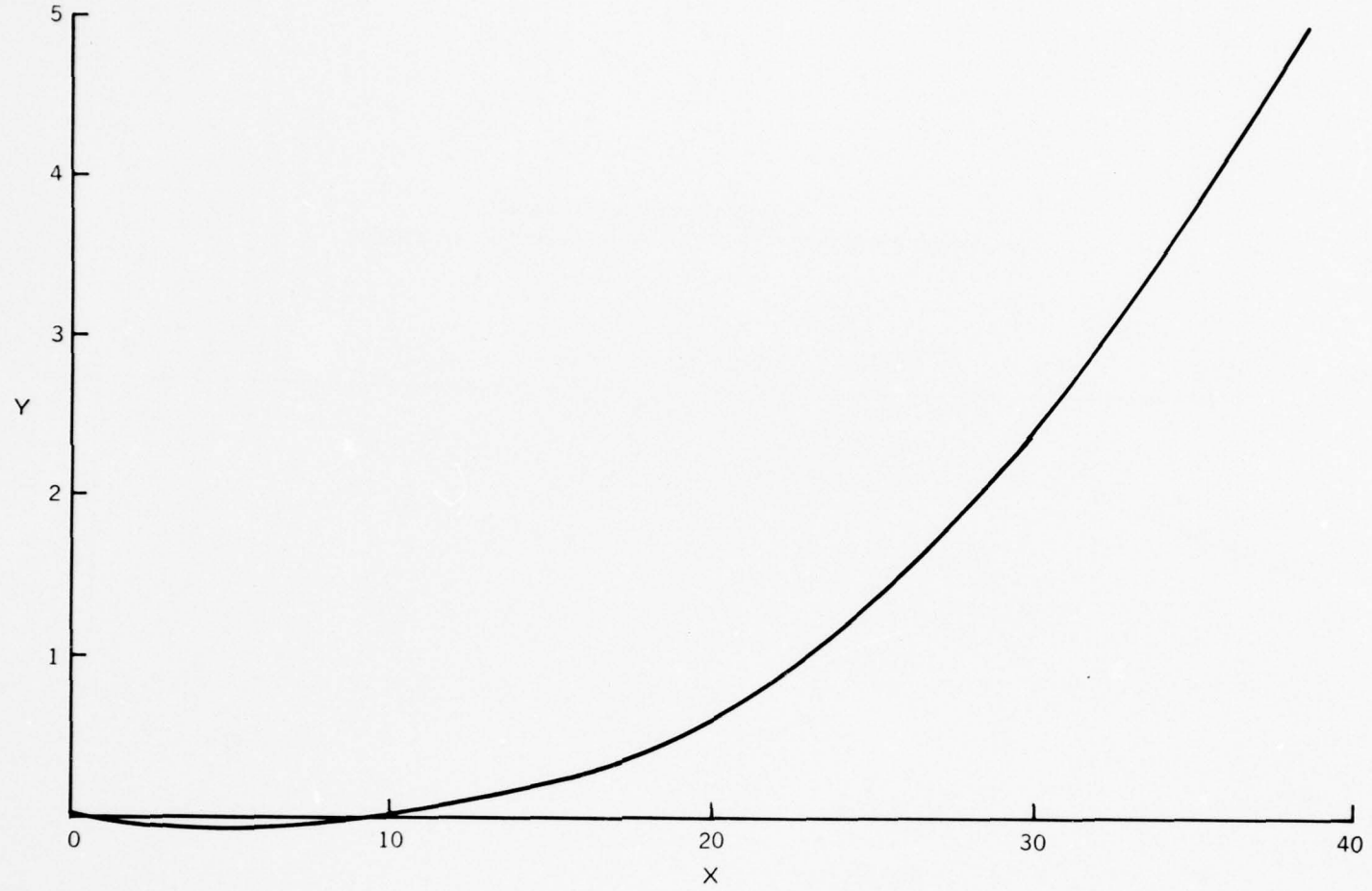



Figure 7.13. Fixed-Time Torpedo Control Problem. Case B: SE Method, Fixed Final Endpoint, u-Trajectory.

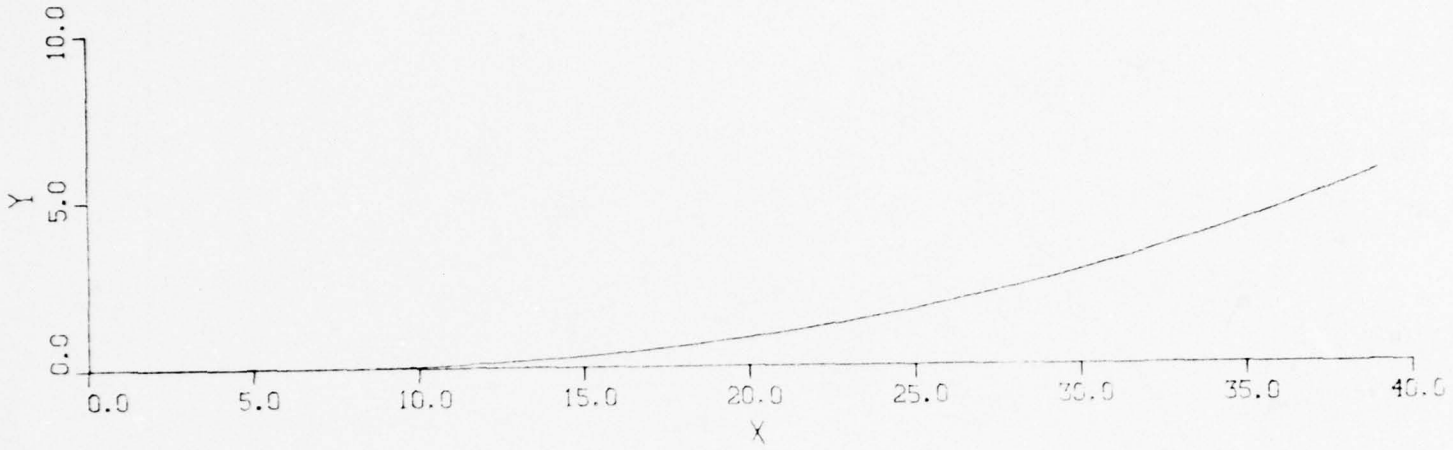



Figure 7.14. Fixed-Time Torpedo Control Problem. Case B: ME Method, x-Trajectory.

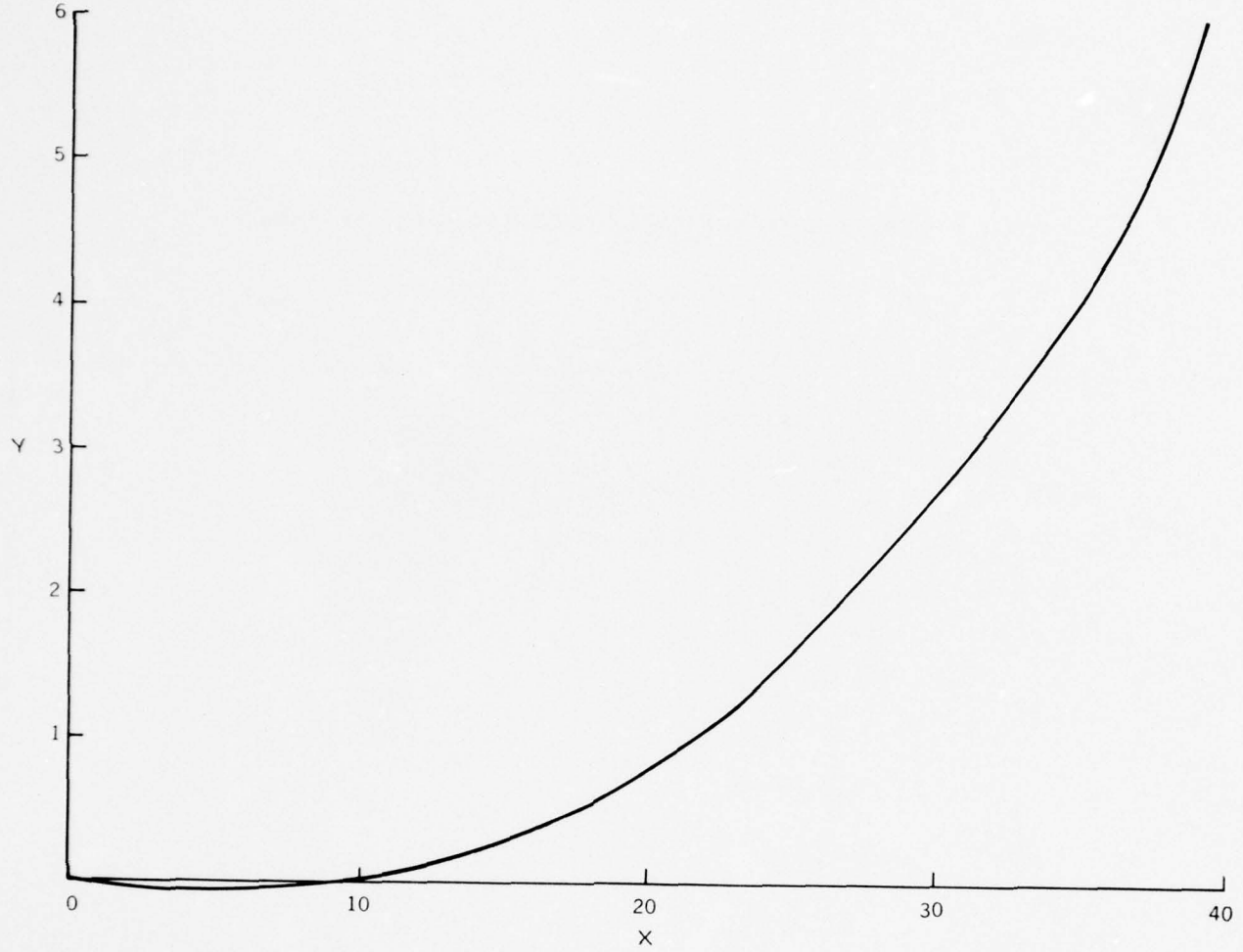



Figure 7.15. Fixed-Time Torpedo Control Problem. Case B: CG Method, Euler Integration.

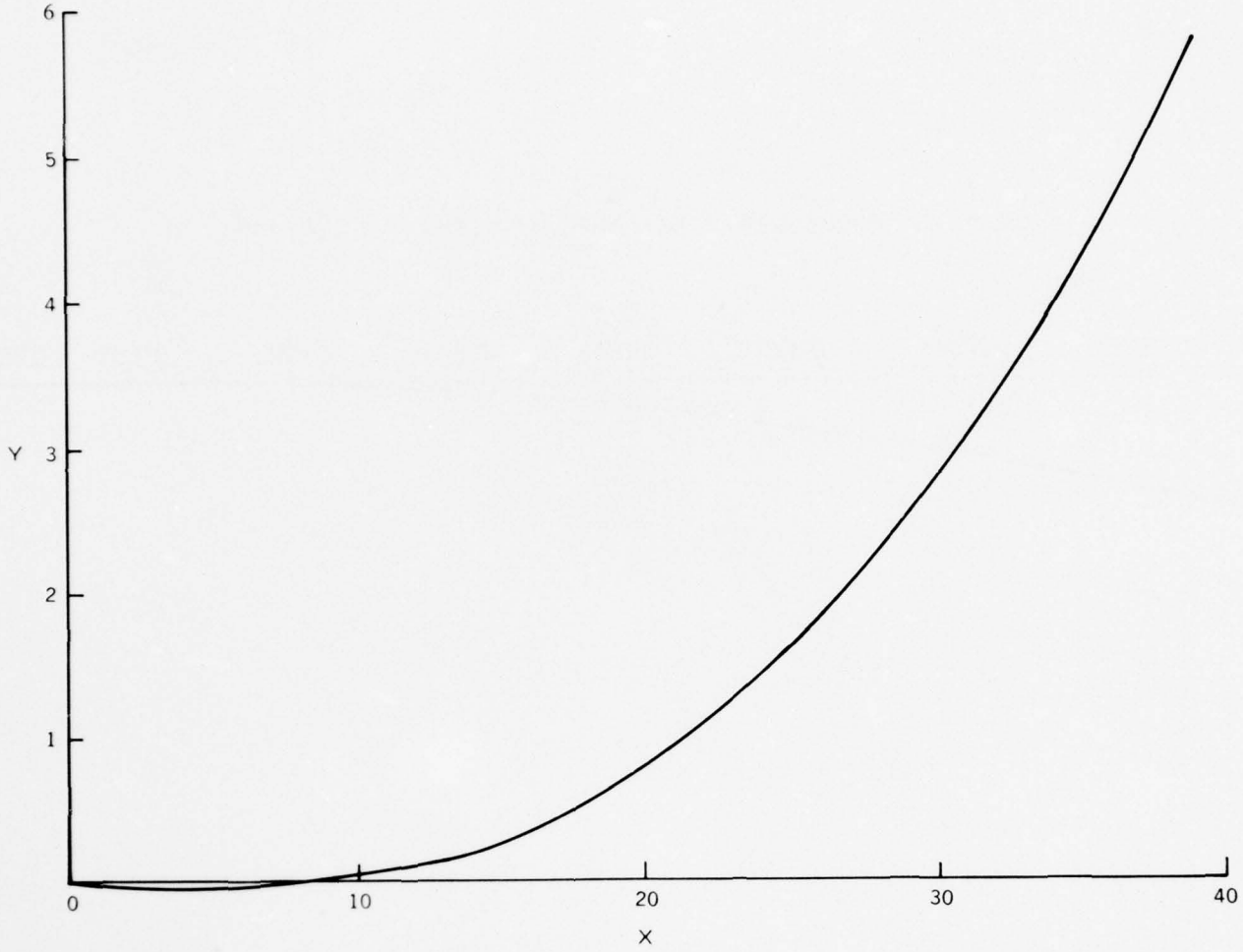



Figure 7.16. Fixed-Time Torpedo Control Problem. Case B: CG Method, Moulton's Method of Integration.

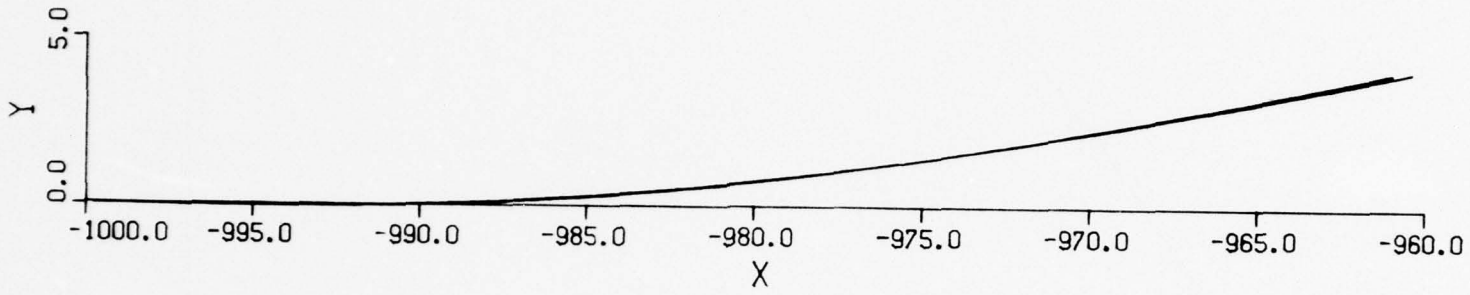



Figure 7.17. Fixed-Time Torpedo Control Problem. Case B: LIN Method, $\lambda = 60{,}000$.

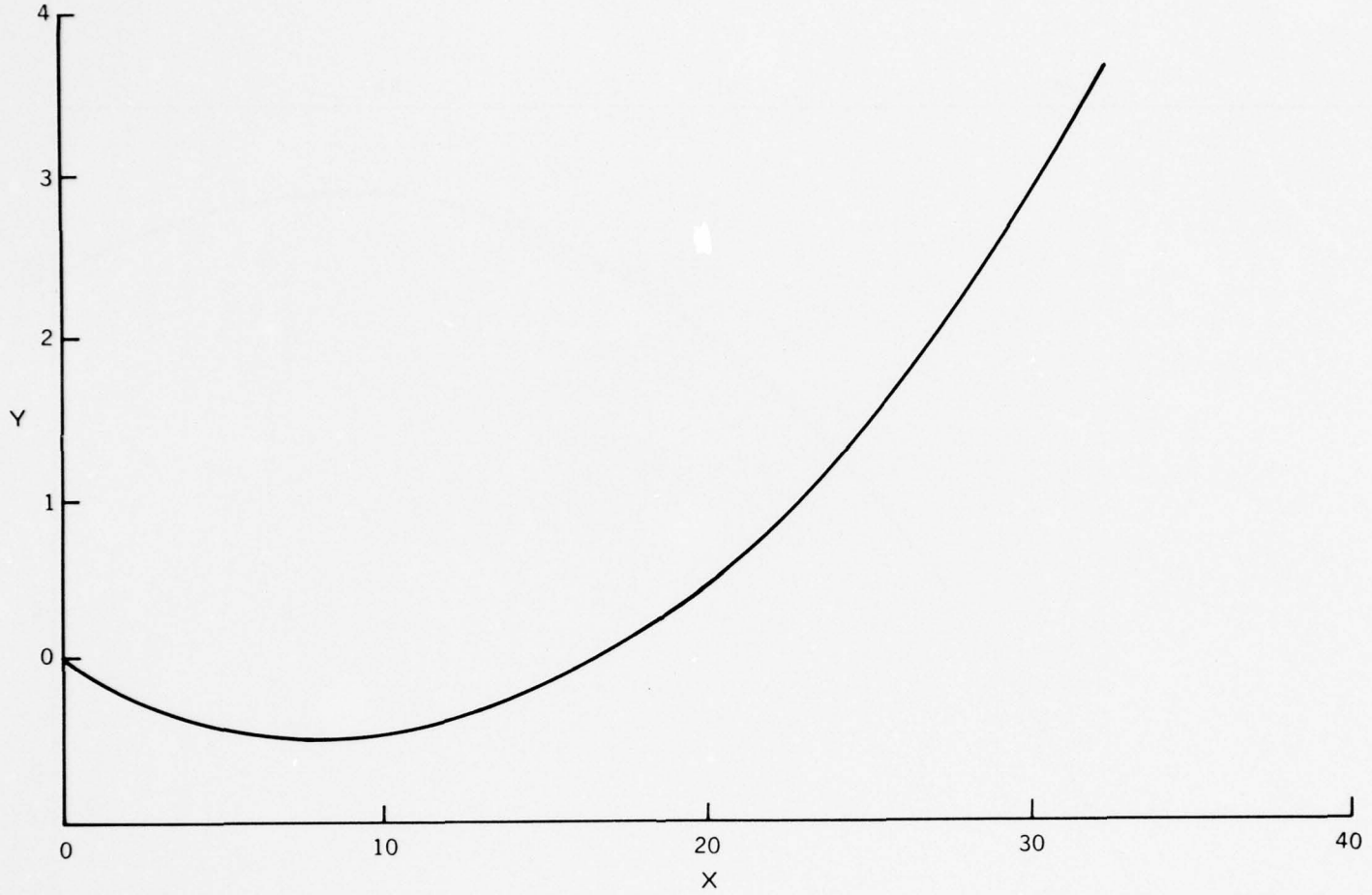



Figure 7.18. Fixed-Time Torpedo Control Problem. Case B: Exact Torpedo Trajectory, Torpedo Initially in a Curve.

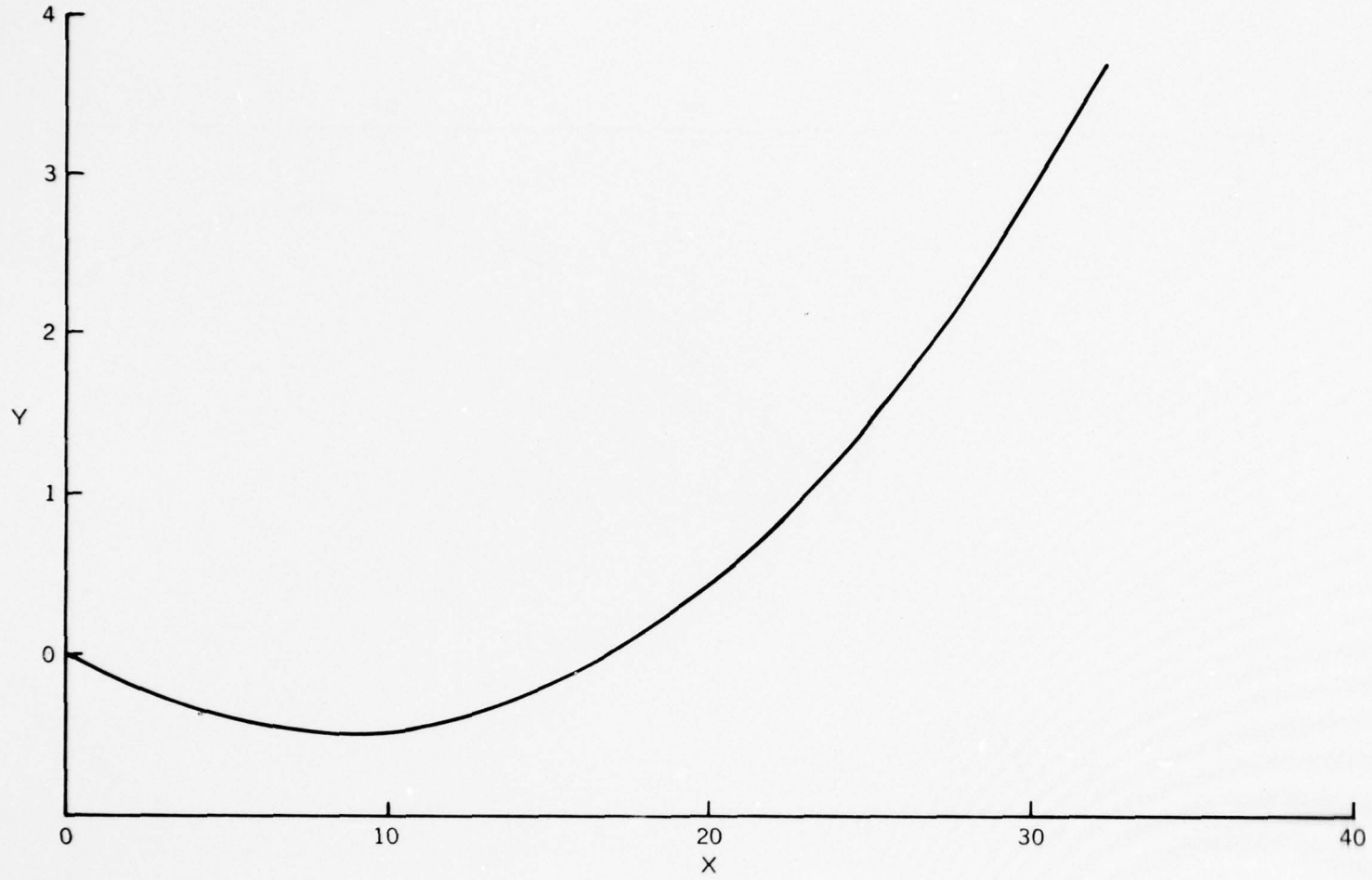



Figure 7.19. Fixed-Time Torpedo Control Problem. Case B: ME Method, Torpedo Initially in a Curve.

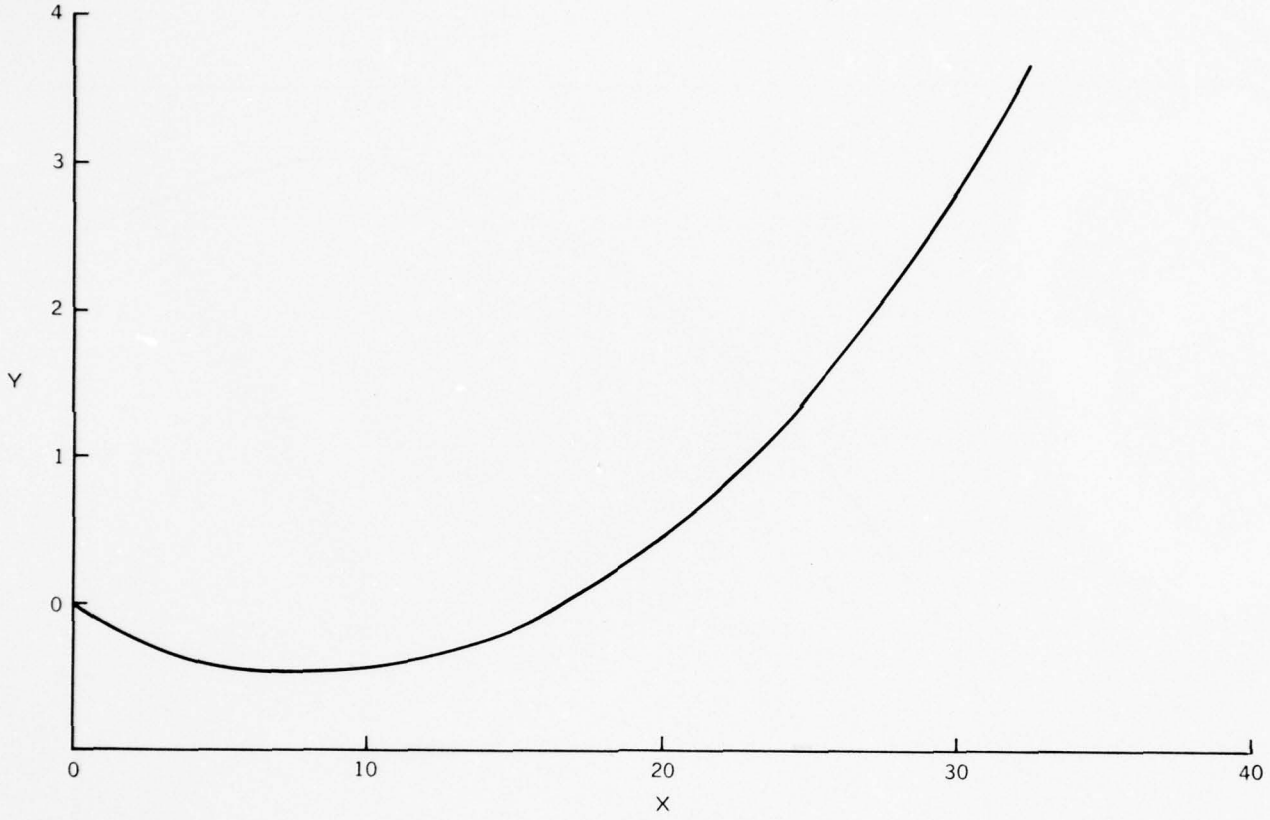



Figure 7.20. Fixed-Time Torpedo Control Problem. Case B: CG Method, Torpedo Initially in a Curve.

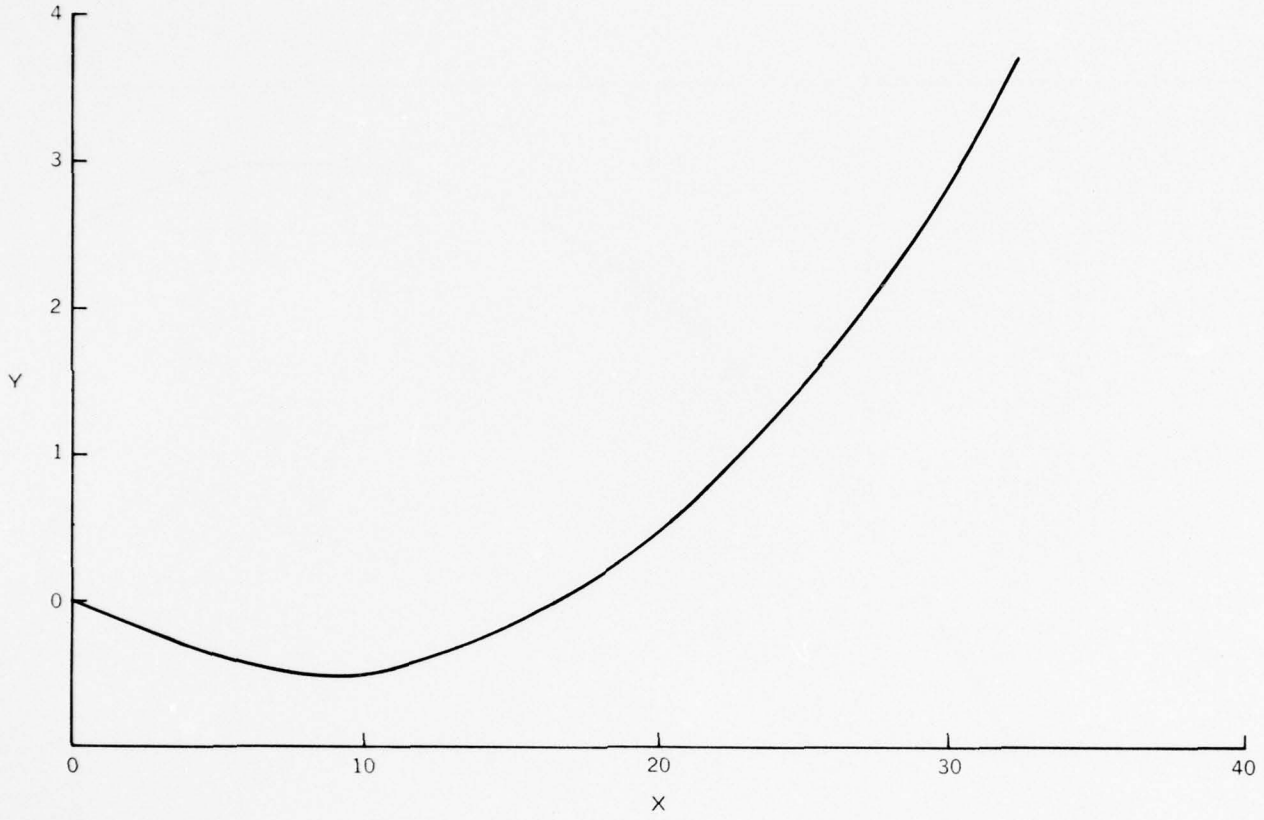



Figure 7.21. Fixed-Time Torpedo Control Problem. Case B: CG Method, Torpedo Initially in a Curve, Original Method of Application.

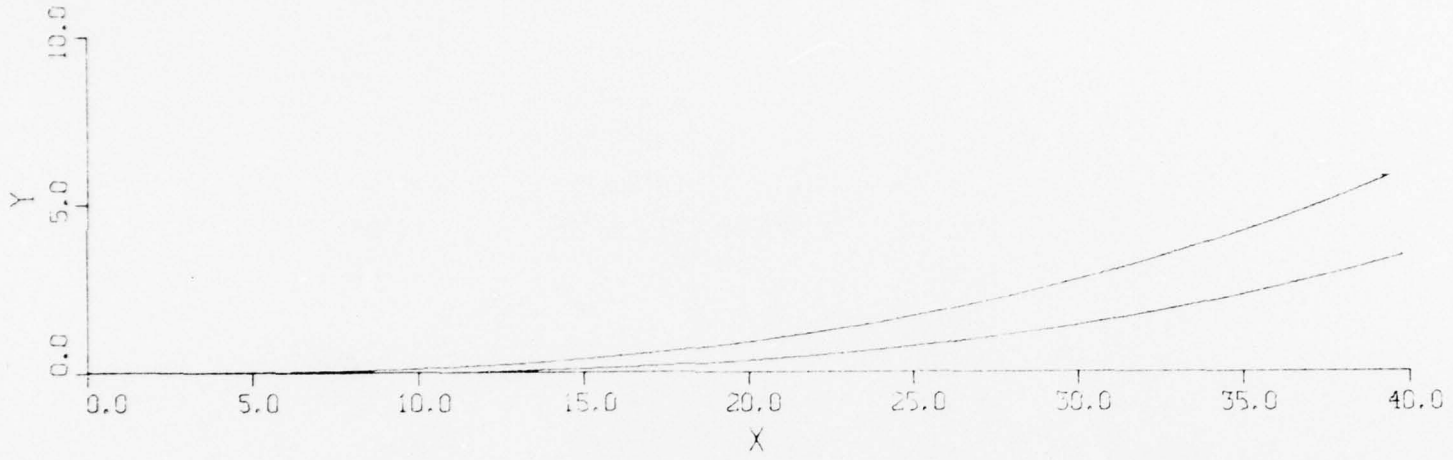



Figure 7.22. Fixed-Time Torpedo Control Problem. Case B: SE Method, Aimpoint Head of Torpedo.

AD-A042 206 NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CALIF F/G 19/8
COMPUTATIONAL TECHNIQUES FOR TIME-OPTIMAL CONTROL OF TORPEDO DY--ETC(U)
MAY 77 W S LAPP
UNCLASSIFIED NOSC-TR-124 NL

2 OF 3
ADA
042206

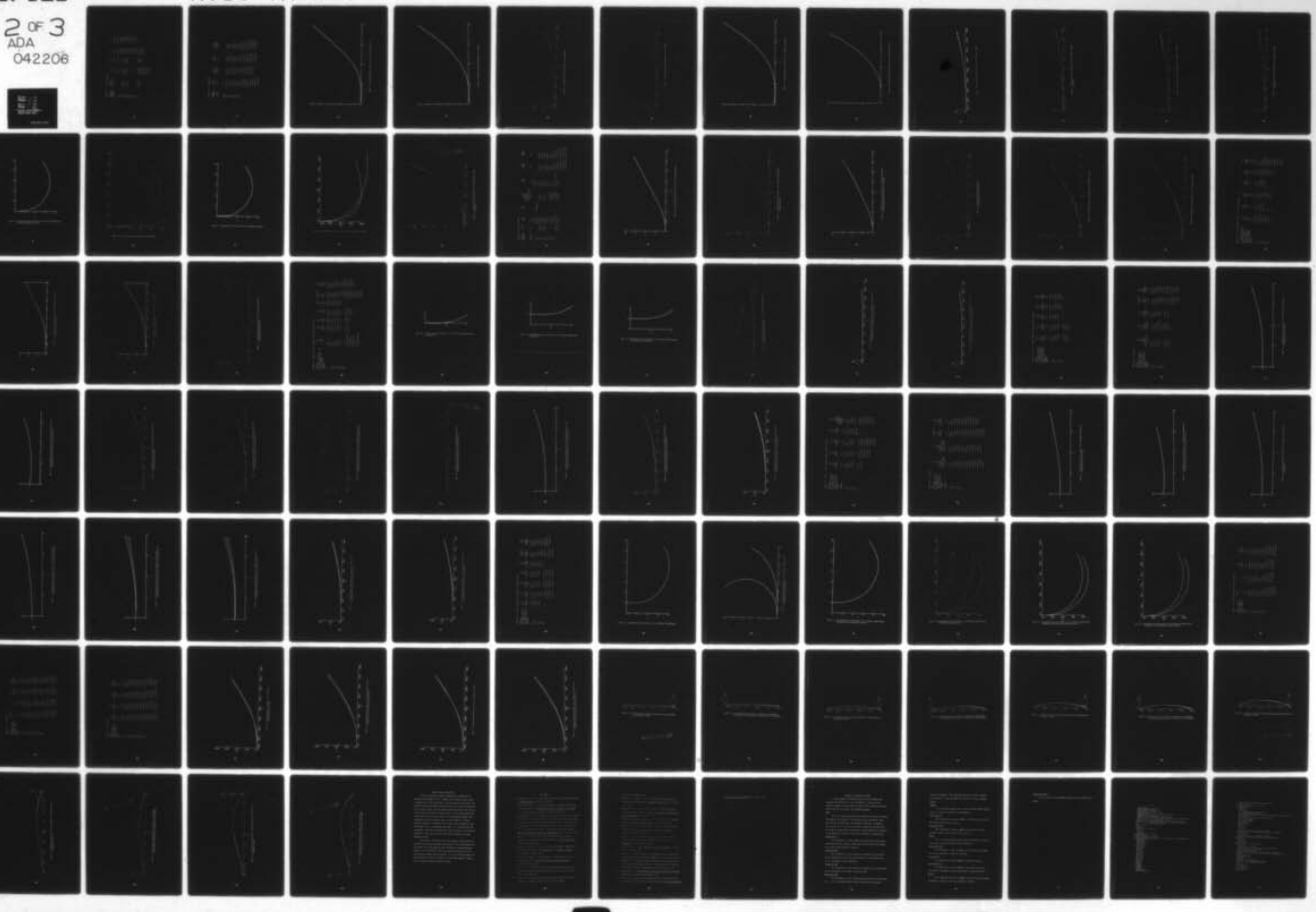


TABLE 7.3: Case C Results

| Method<br>Comments<br>Figures | Exact<br>Trajectory<br>7.23 | CG<br>7.24 | MCG<br>7.25 | SE<br>7.26 | ME<br>7.27,7.28 |
|---|---|---|---|---|---|
| J | | 0.6136 | 70.85 | 98.776 | 79.126 |
| De/ε | | | | 59.328 | 69.597 |
| $x_1^*$ | -960.56 | -960.57 | -961.04 | -959.94 | -960.87 |
| $x_2^*$ | 5.0743 | 5.0689 | 5.7635 | 5.0389 | 5.0469 |
| $x_6^*$ | 0.27809 | 0.3343 | 0.43274 | 0.25138 | 0.44902 |
| $x_1^u$ | | | | -959.57 | -961.03 |
| $x_2^u$ | | | | 0.8222 | 5.7298 |
| $x_6^u$ | | | | 0.08601 | 0.4859 |
| u | -U,t≤.3 | -0.26,≤71 | -U,≤261 | -0.021,1 | -U |
| | 0,t<.6 | -0.204,96 | U,≥266 | -0.48,25 | |
| | | -0.15,161 | | -0.77,51 | |
| | | -0.24,261 | | | |
| | | -0.26≥266 | | | |

TABLE 7.3: continued

| Method | LIN | LIN | LIN | LIN |
|---|---|---|---|---|
| Comments | $\lambda$=60000 | $\lambda$=60000 R Const. | $\lambda$=40000 | $\lambda$=40000 R Const. |
| Figures | 7.29 | 7.30 | 7.31 | 7.32 |
| J | | | | |
| De/$\varepsilon$ | | | | |
| $x_1^*$ | -960.585 | -960.51 | -960.63 | -960.56 |
| $x_2^*$ | 3.618 | 3.0706 | 4.0001 | 3.418 |
| $x_6^*$ | 0.20287 | 0.17558 | 0.22383 | 0.18068 |
| $x_1^u$ | -960.76 | -960.119 | -960.37 | -960.20 |
| $x_2^u$ | 3.6154 | 3.0984 | 3.9716 | 3.4309 |
| $x_6^u$ | 0.2007 | 0.17383 | 0.22140 | 0.17893 |
| u | -0.239,10 | -0.216,10 | -0.264,10 | -0.261,10 |
| | -0.196,90 | -0.203,30 | -0.203,110 | -0.207,70 |
| | -0.1,270 | -0.101,220 | -0.105,280 | -0.102,220 |
| | -0.01,480 | -0.01,530 | -0.0006,520 | -0.0006,500 |
| | 0.013,530 | 0.0002,600 | 0.0047,570 | 0.0106,570 |
| | 0.0042,570 | | 0.0003,600 | 0.1419,600 |
| | 0.0023,590 | | | |

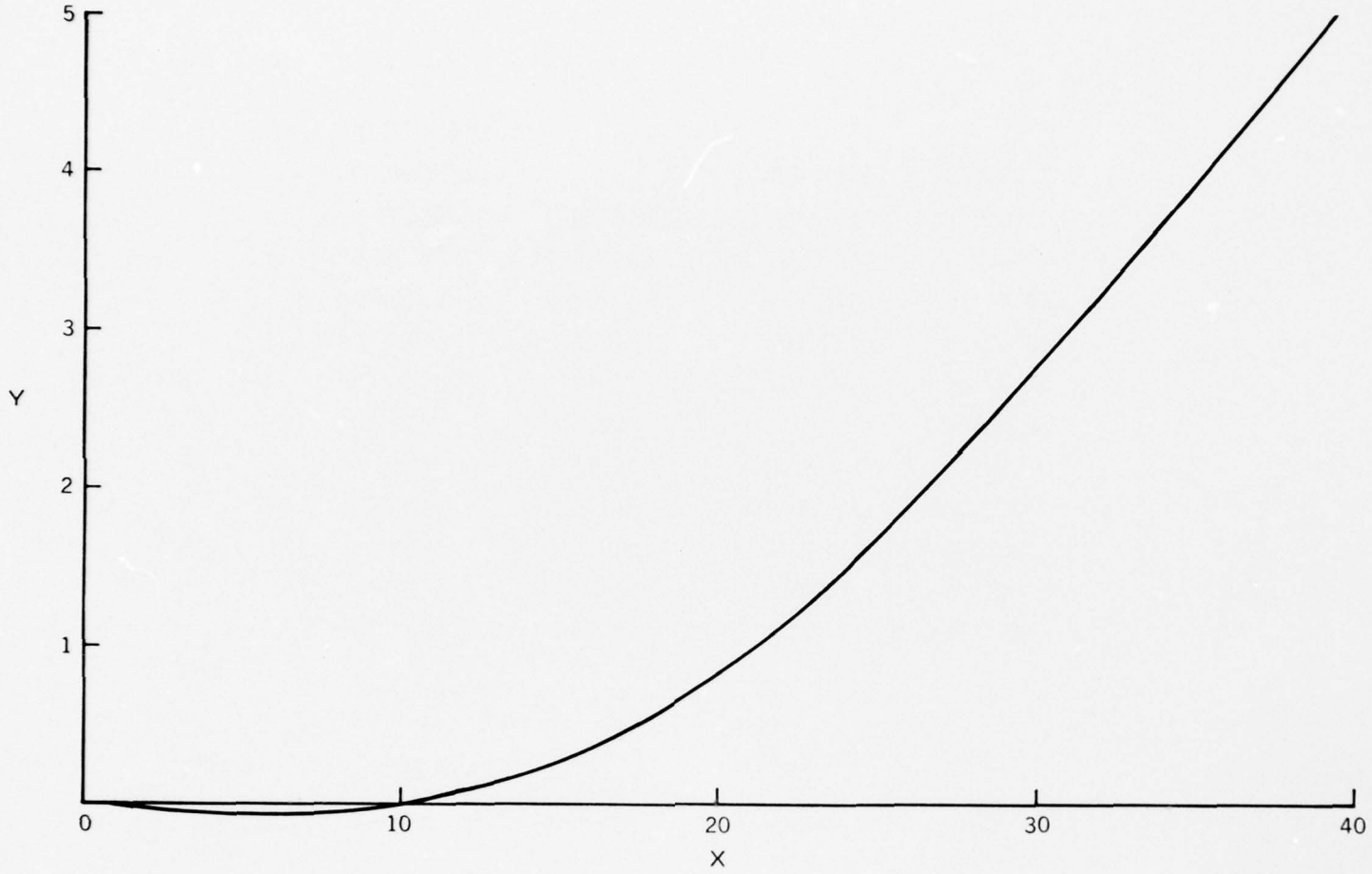



Figure 7.23. Fixed-Time Torpedo Control Problem. Case C: Exact Torpedo Trajectory.

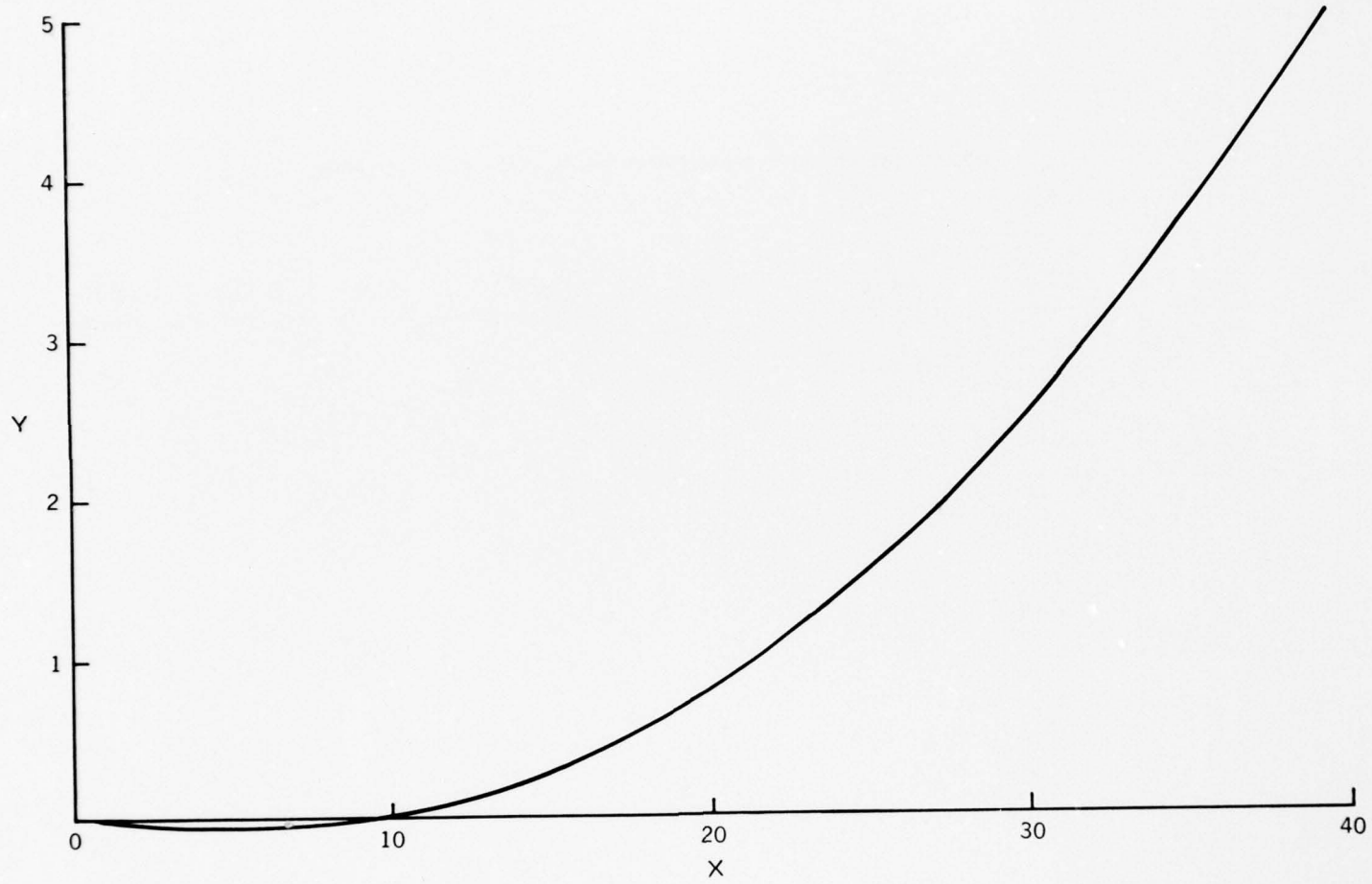



Figure 7.24. Fixed-Time Torpedo Control Problem. Case C: CG Method.

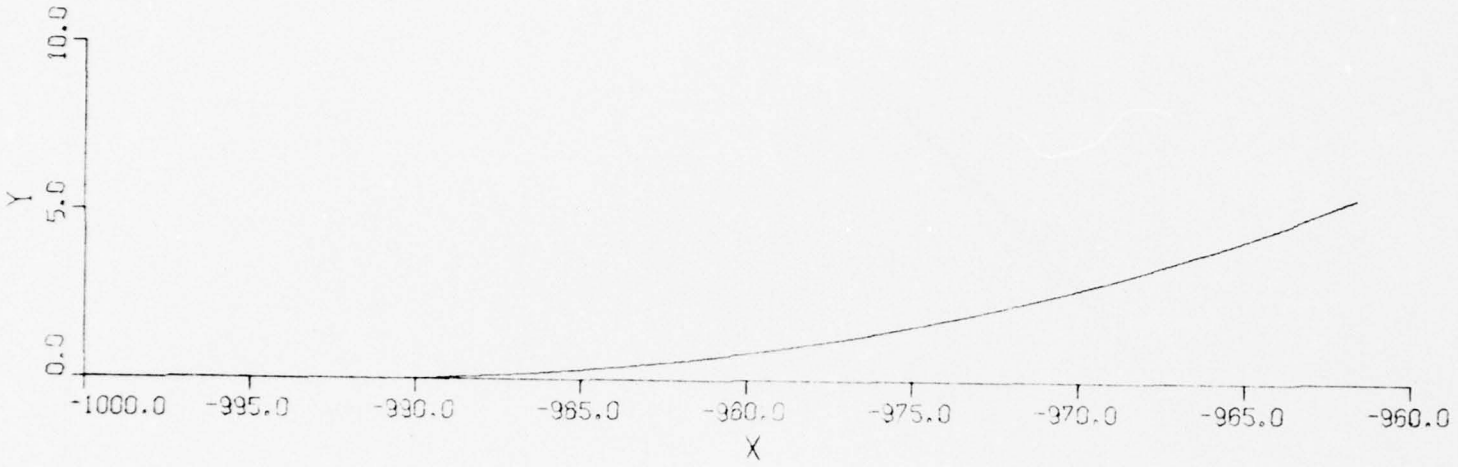



Figure 7.25. Fixed-Time Torpedo Control Problem. Case C: MCG Method.

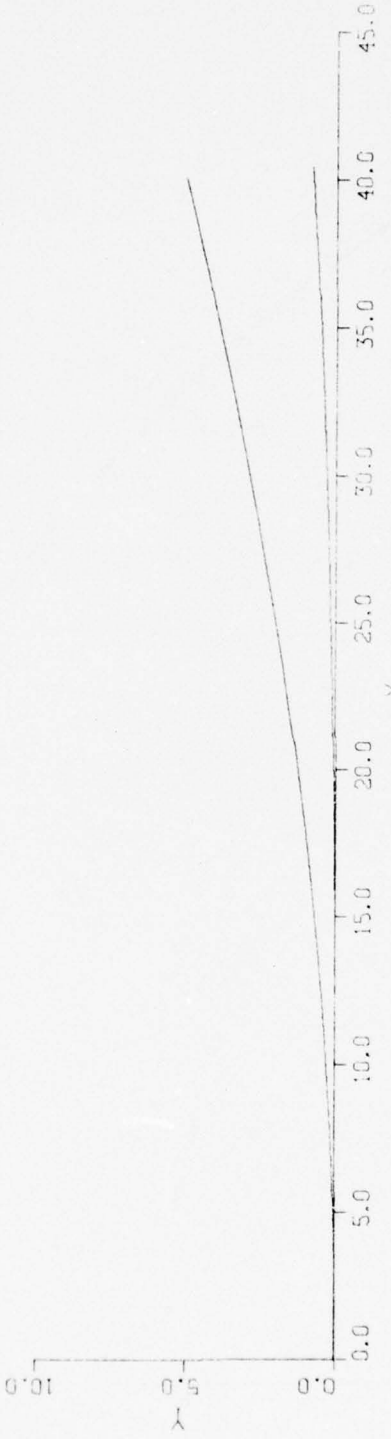



Figure 7.26. Fixed-Time Torpedo Control Problem. Case C: SE Method.

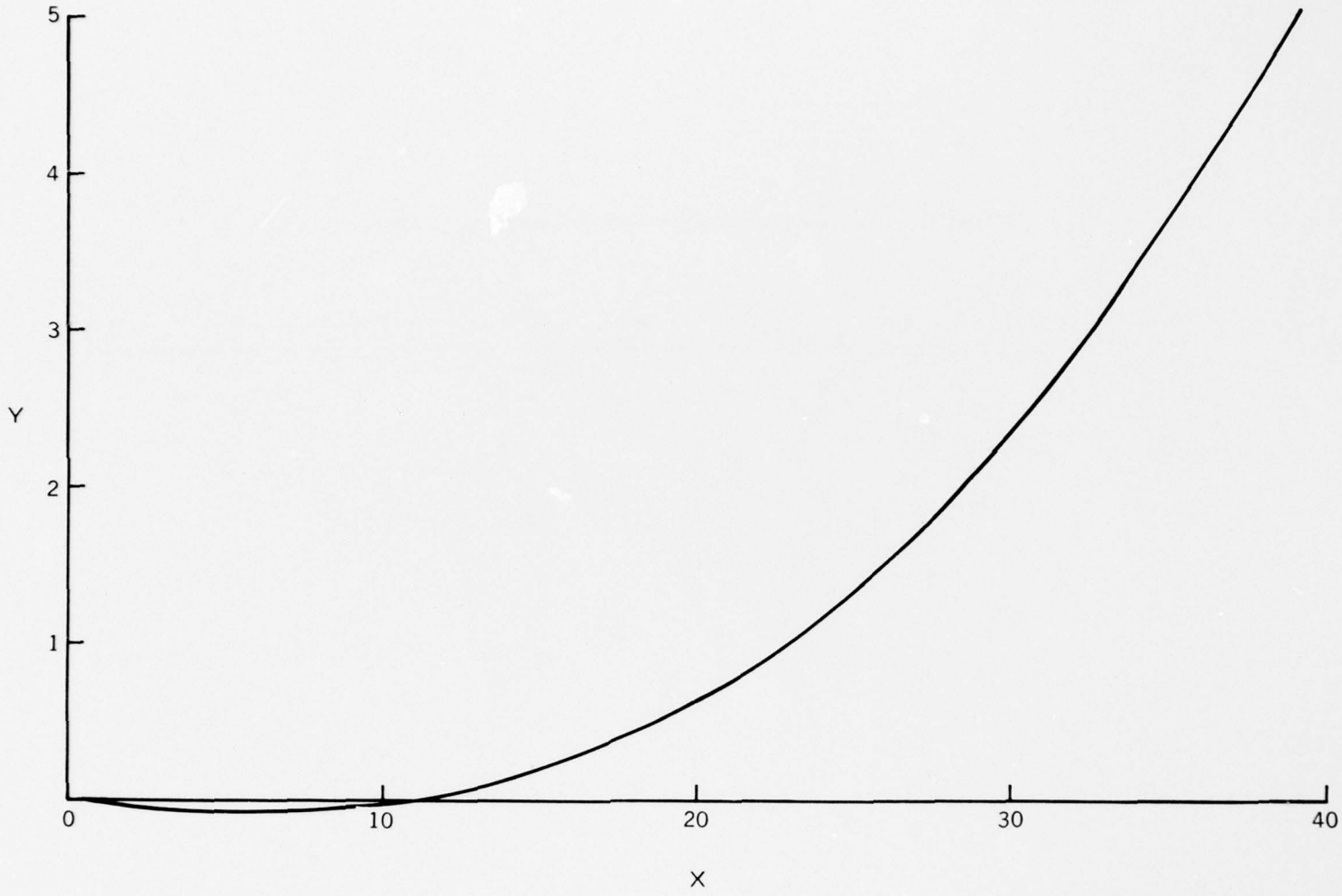



Figure 7.27. Fixed-Time Torpedo Control Problem. Case C: ME Method, x-Trajectory.

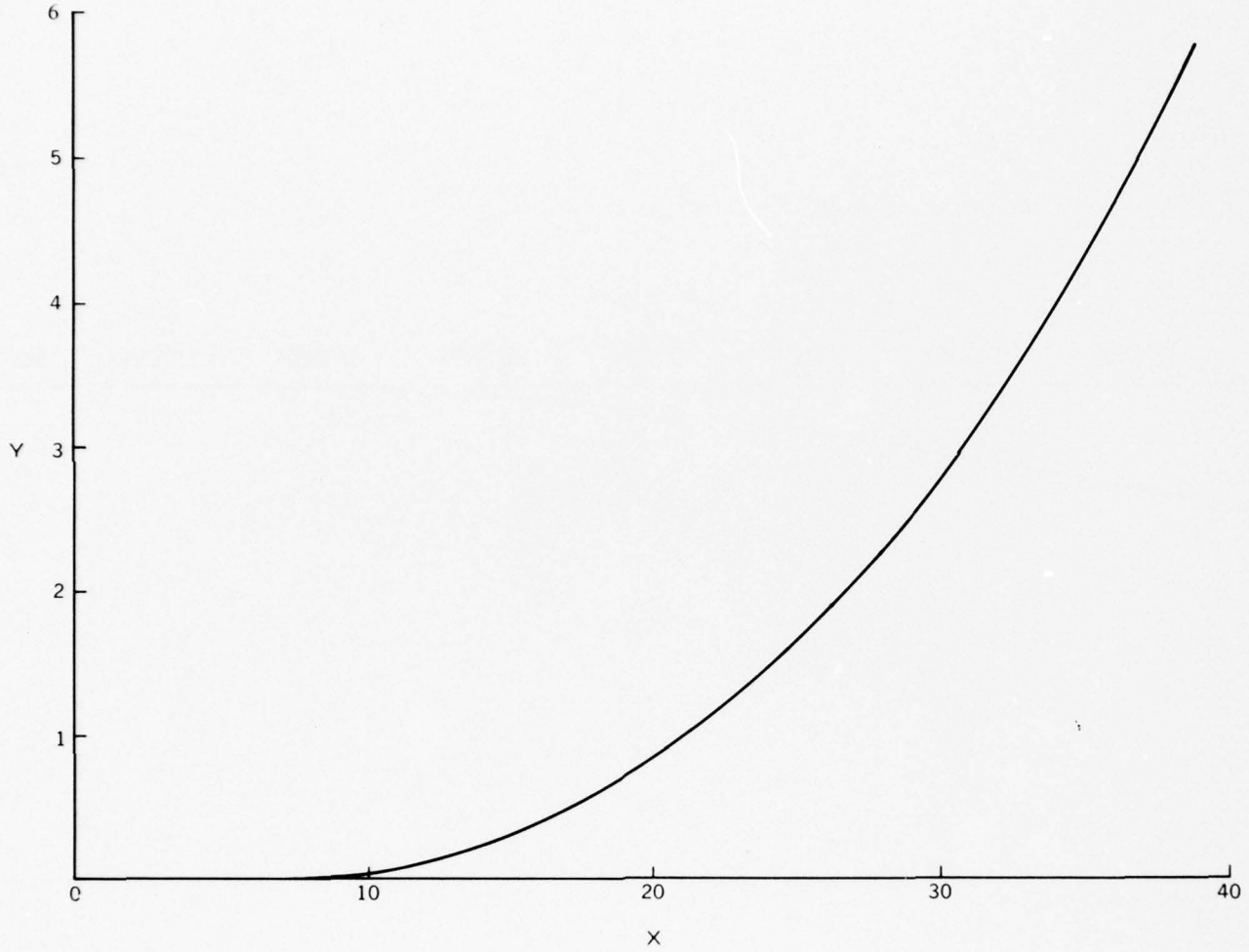



Figure 7.28. Fixed-Time Torpedo Control Problem. Case C: ME Method, u-Trajectory.

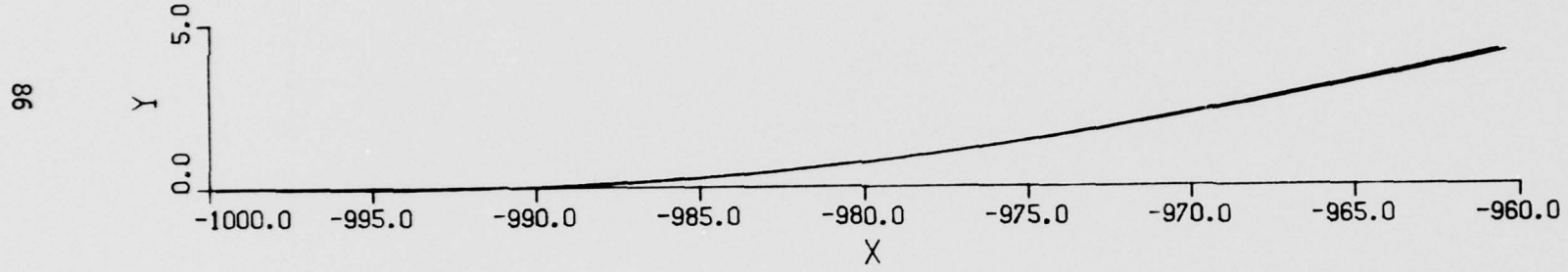



Figure 7.29. Fixed-Time Torpedo Control Problem. Case C: LIN Method, $\lambda = 60{,}000$.

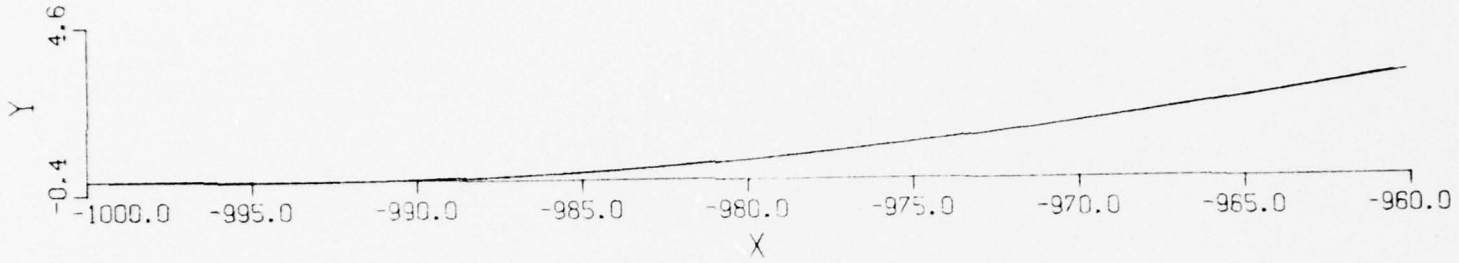



Figure 7.30. Fixed-Time Torpedo Control Problem. Case C: LIN Method, $\lambda = 60{,}000$, R Constant.

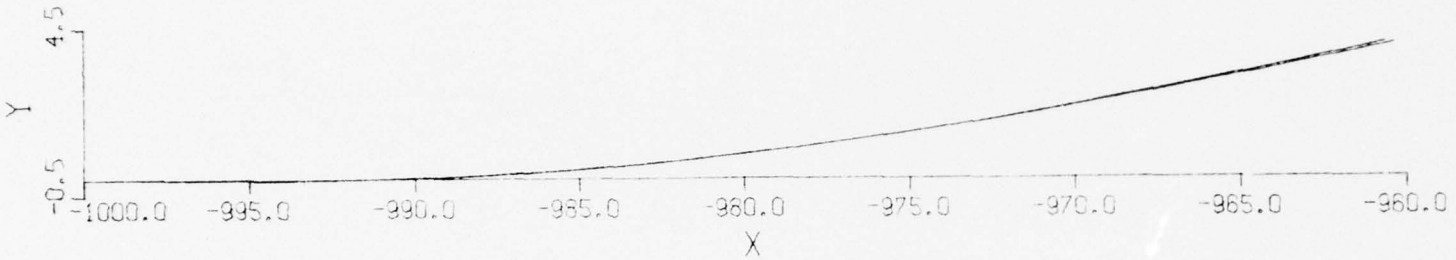



Figure 7.31. Fixed-Time Torpedo Control Problem. Case C: LIN Method, $\lambda = 40{,}000$.

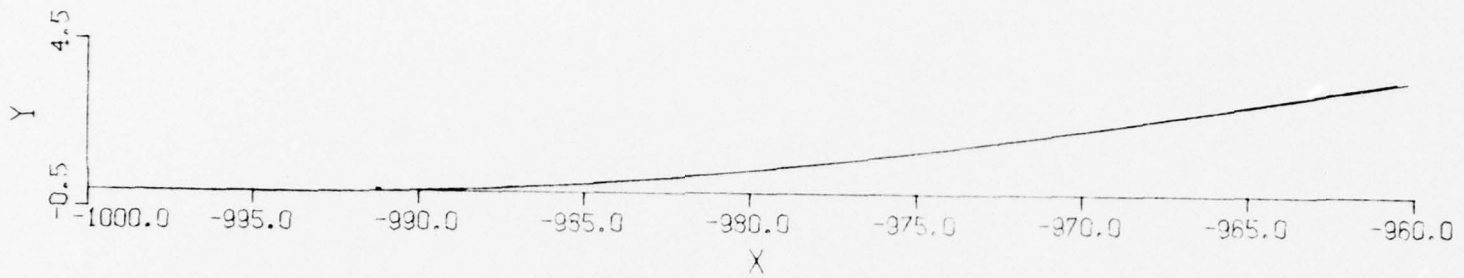



Figure 7.32. Fixed-Time Torpedo Control Problem. Case C: LIN Method, $\lambda = 40{,}000$, R Constant.

TABLE 7.4: Case D Results

| Method<br>Comments | Exact | CG | SE | ME | LIN<br>$\lambda$=60000 | LIN<br>$\lambda$=60000<br>R Const. |
|---|---|---|---|---|---|---|
| Figures | 7.33 | 7.33 | 7.34 | 7.35,7.33 | 7.36 | 7.37 |
| J | | 3.00004 | 1336.7 | 5844.7 | | |
| De/$\varepsilon$ | | | 733.37 | 5844.1 | | |
| $x_1^*$ | -933.22 | -933.2 | -933.63 | -933.18 | -933.14 | -885.63 |
| $x_2^*$ | 120.41 | 120.411 | 122.03 | 120.35 | 119.58 | 93.234 |
| $x_6^*$ | 2.4159 | 2.37465 | 2.5967 | 2.3552 | 1.84777 | 1.24188 |
| $x_1^u$ | | | -830.87 | -933.67 | -911.19 | -870.275 |
| $x_2^u$ | | | 71.22 | 120.69 | 131.40 | 122.74 |
| $x_6^u$ | | | 1.3714 | 2.41598 | 1.69583 | 1.180 |
| u | | -U,$\leq$291 | -0.016,1 | -U | -0.303,10 | -0.298,10 |
| | | U,$\geq$296 | -0.149,17 | | -0.201,240 | -0.207,110 |
| | | | -0.2,37 | | -0.102,470 | -0.102,280 |
| | | | -U,49 | | -0.017,570 | -0.0137,600 |
| | | | | | -0.0067,580 | |
| | | | | | 0.0007,600 | |

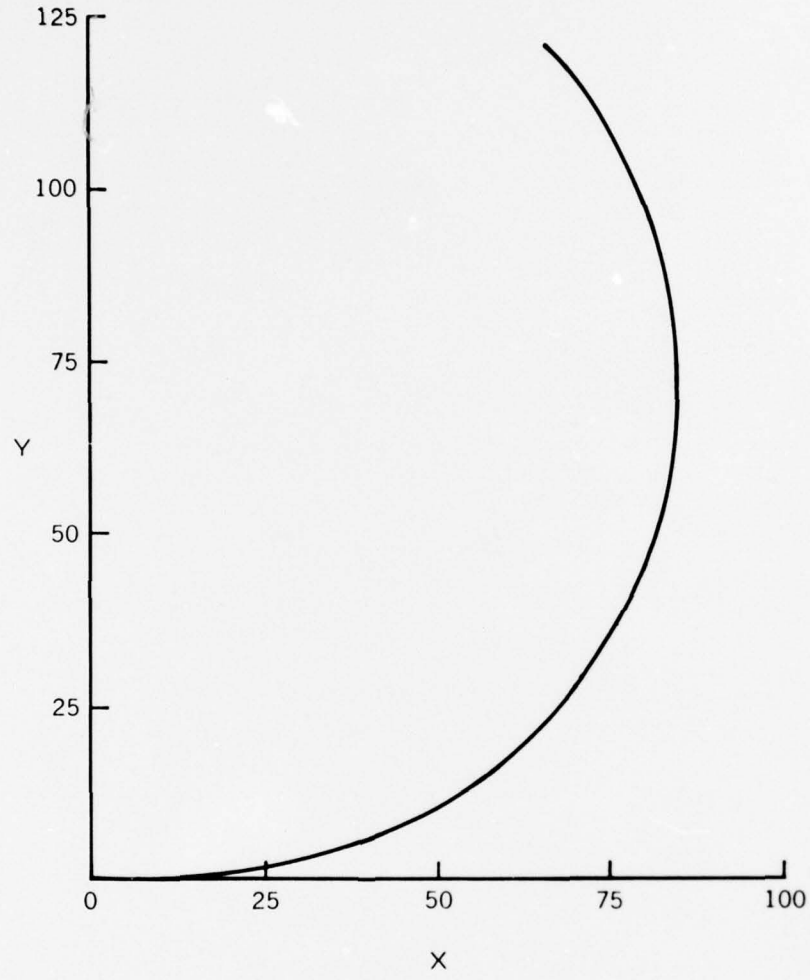



Figure 7.33. Fixed-Time Torpedo Control Problem. Case D: Exact Torpedo Trajectory; CG Method; ME Method, u-Trajectory.

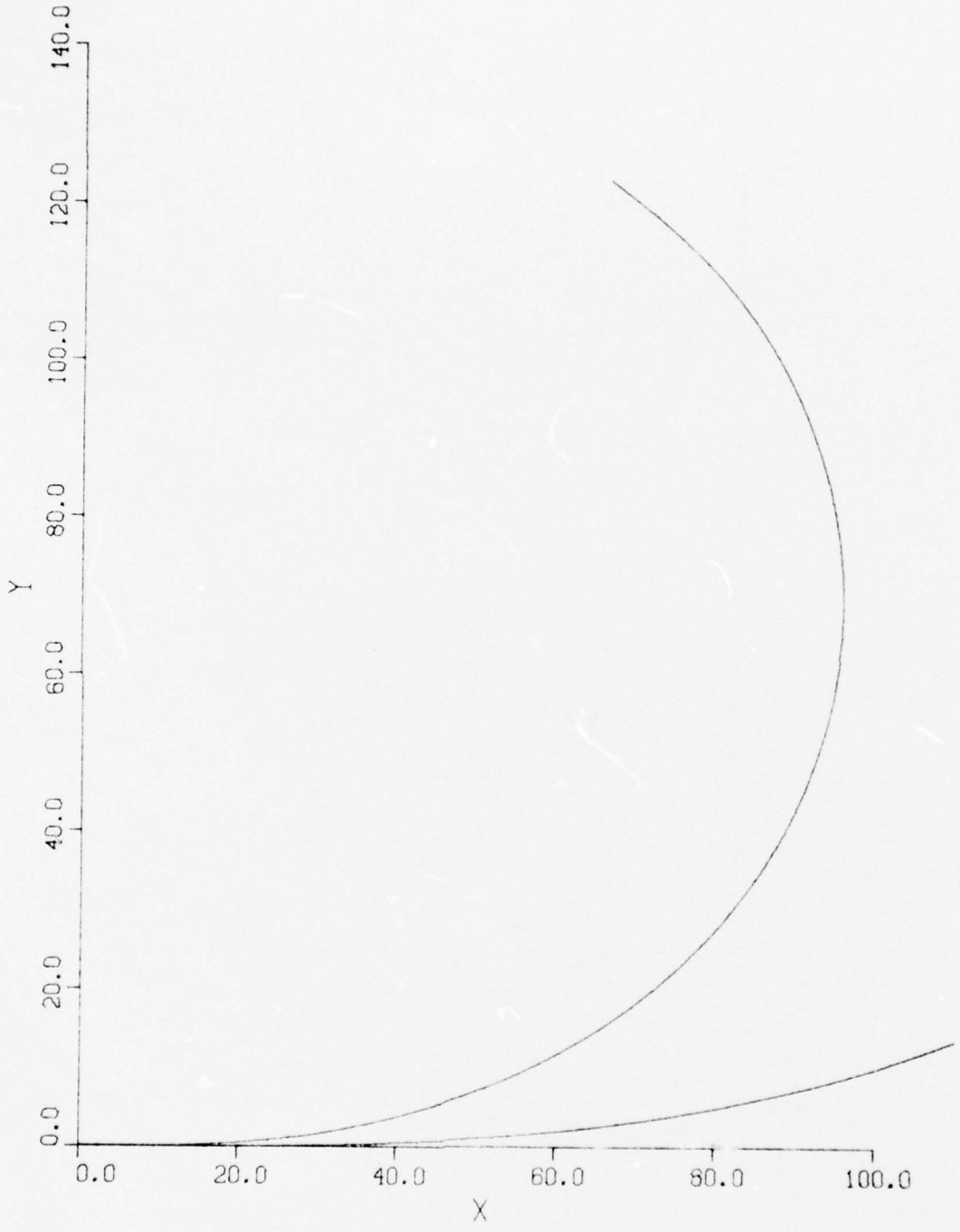



Figure 7.34. Fixed-Time Torpedo Control Problem. Case D: SE Method.

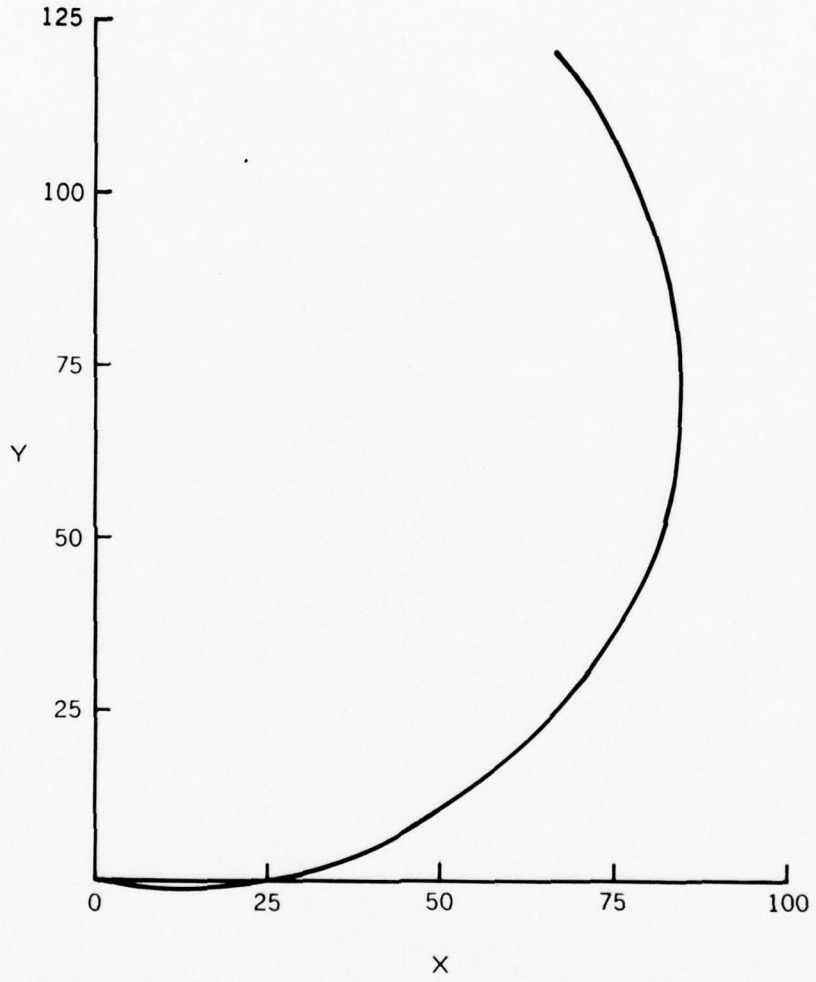



Figure 7.35. Fixed-Time Torpedo Control Problem. Case D: ME Method, x-Trajectory.

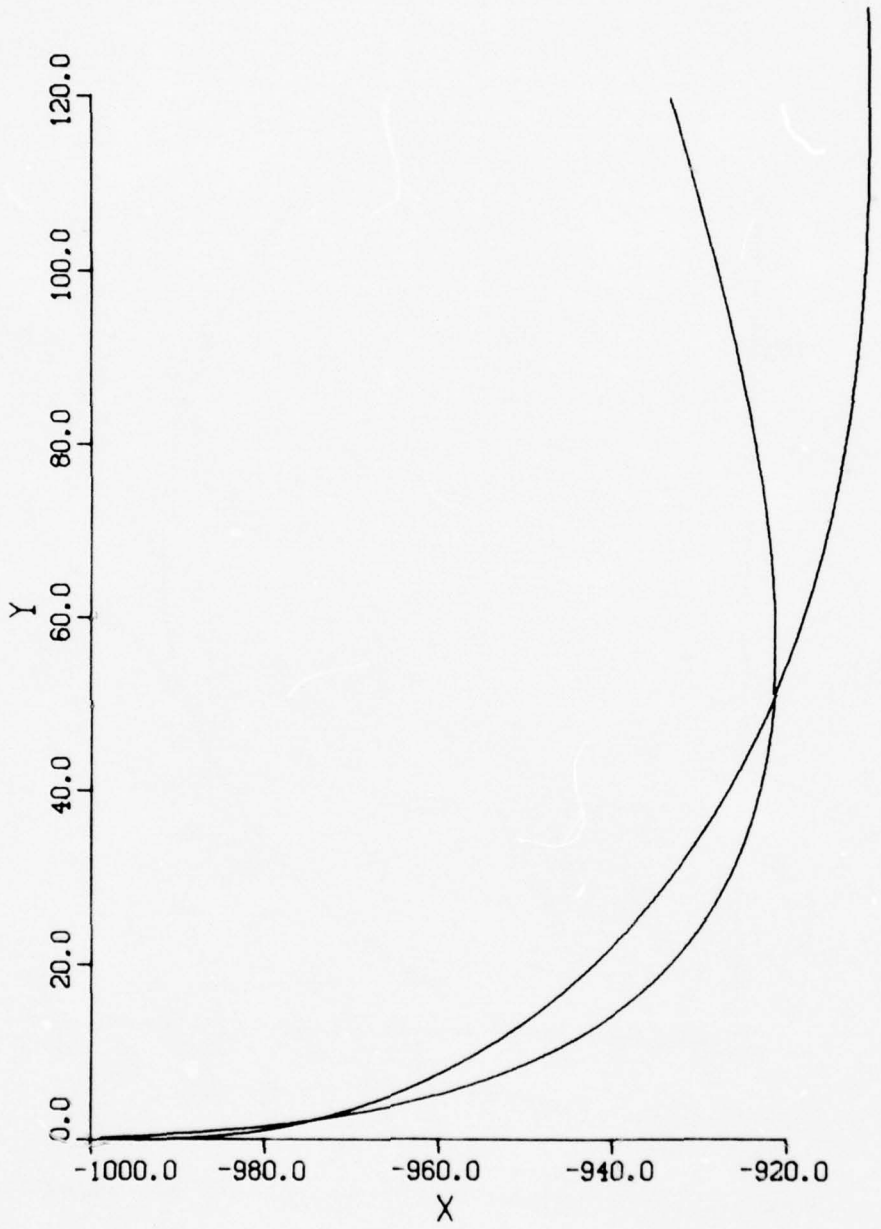



Figure 7.36. Fixed-Time Torpedo Control Problem. Case D: LIN Method, $\lambda = 60{,}000$.

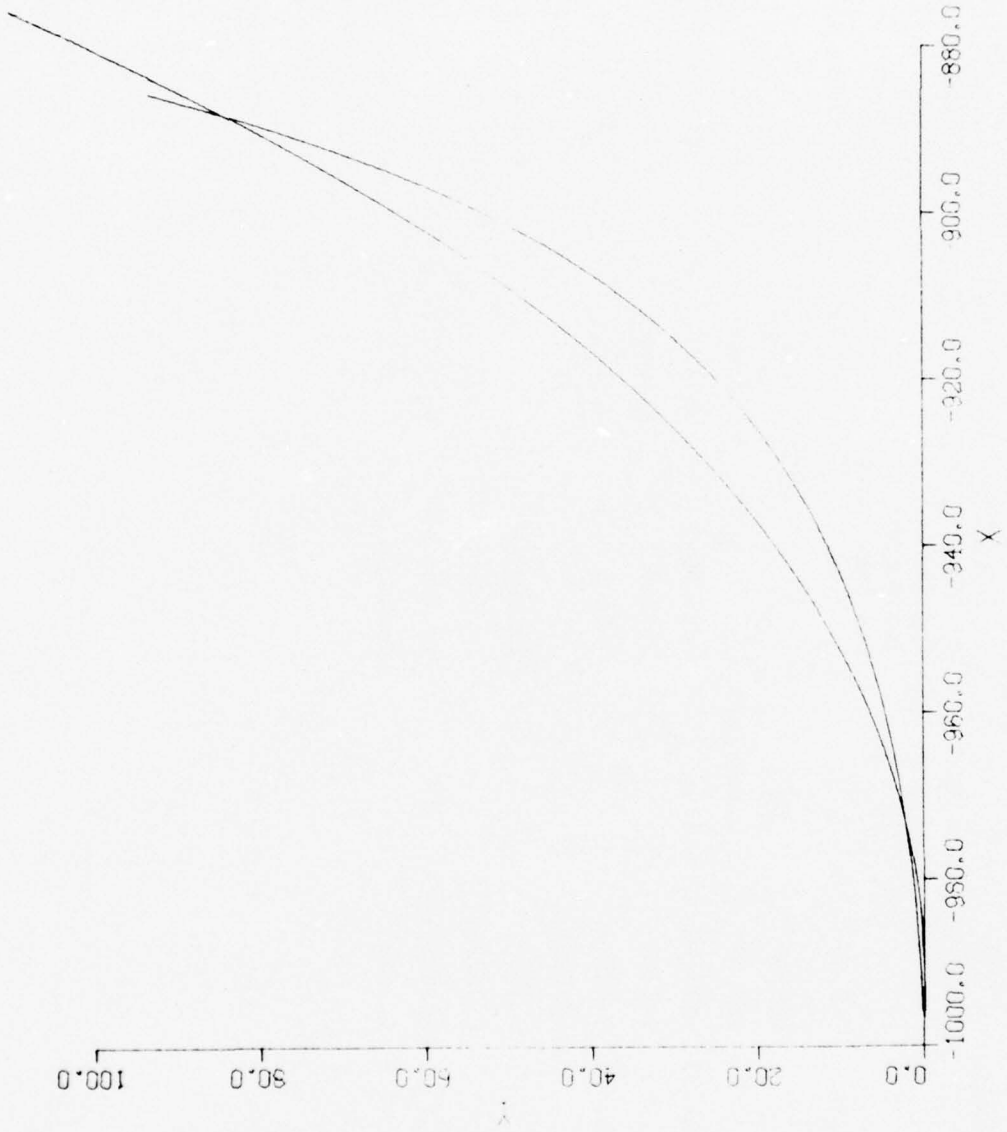



Figure 3.37. Fixed-Time Torpedo Control Problem. Case D: LIN Method, $\lambda = 60{,}000$, R Constant.

TABLE 7.5: Case E Results

| Method<br>Comments | Exact | SE<br>AHOT | CG<br>AHOT | CG<br>AHOT<br>Initial Trajectory<br>0.4 sec. curve | ME<br>AHOT | LIN<br>λ=60000 | LIN<br>λ=30000 |
|---|---|---|---|---|---|---|---|
| Figures | 7.38 | 7.39 | | 7.40 | 7.41 | 7.42 | 7.43 |
| J | | 299.98 | No | | 1.169E6 | | |
| De/ε | | 189.36 | Convergence | | 1.095E6 | | |
| $x_1^*$ | -819.39 | -817.49 | | -818.56 | -804.39 | -819.97 | -819.99 |
| $x_2^*$ | 75.36 | 74.399 | | 75.557 | 53.059 | 74.97 | 75.025 |
| $x_6^*$ | 0.46584 | 0.70344 | | 0.5156 | 0.41545 | 0.65006 | 0.66205 |
| $x_1^u$ | | -797.33 | | | -871.18 | -819.43 | -819.54 |
| $x_2^u$ | | 0.2476 | | literation | 122.52 | 73.89 | 73.90 |
| $x_6^u$ | | 0.00228 | | no change | 1.02 | 0.63938 | 0.65120 |
| u | -U,t≤.5 | -0.0003,i=7 | 0 | to initial | -U,i=2,10,13-18, | -0.125,10 | -0.119,10 |
| | 0,t>.5 | is max. | | controls | 0(-1)0.w. | -0.1002,330 | -0.1006,300 |
| | | | | | | -0.0103,1620 | -0.0110,1650 |
| | | | | | | 0.00075,1770 | 0.00024,1770 |
| | | | | | | 0.00005,1800 | 0.00006,1800 |

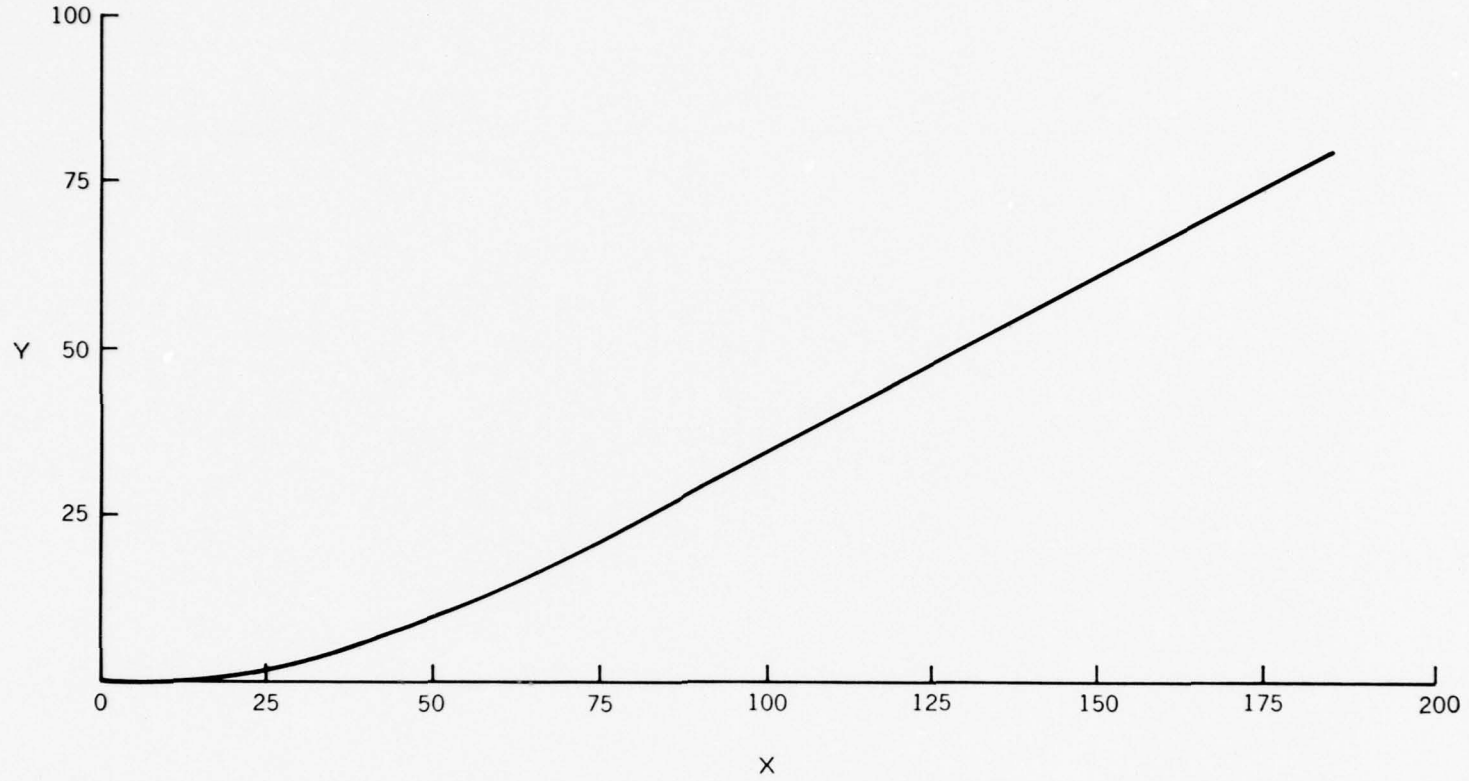



Figure 7.38. Fixed-Time Torpedo Control Problem. Case E: Exact Torpedo Trajectory.

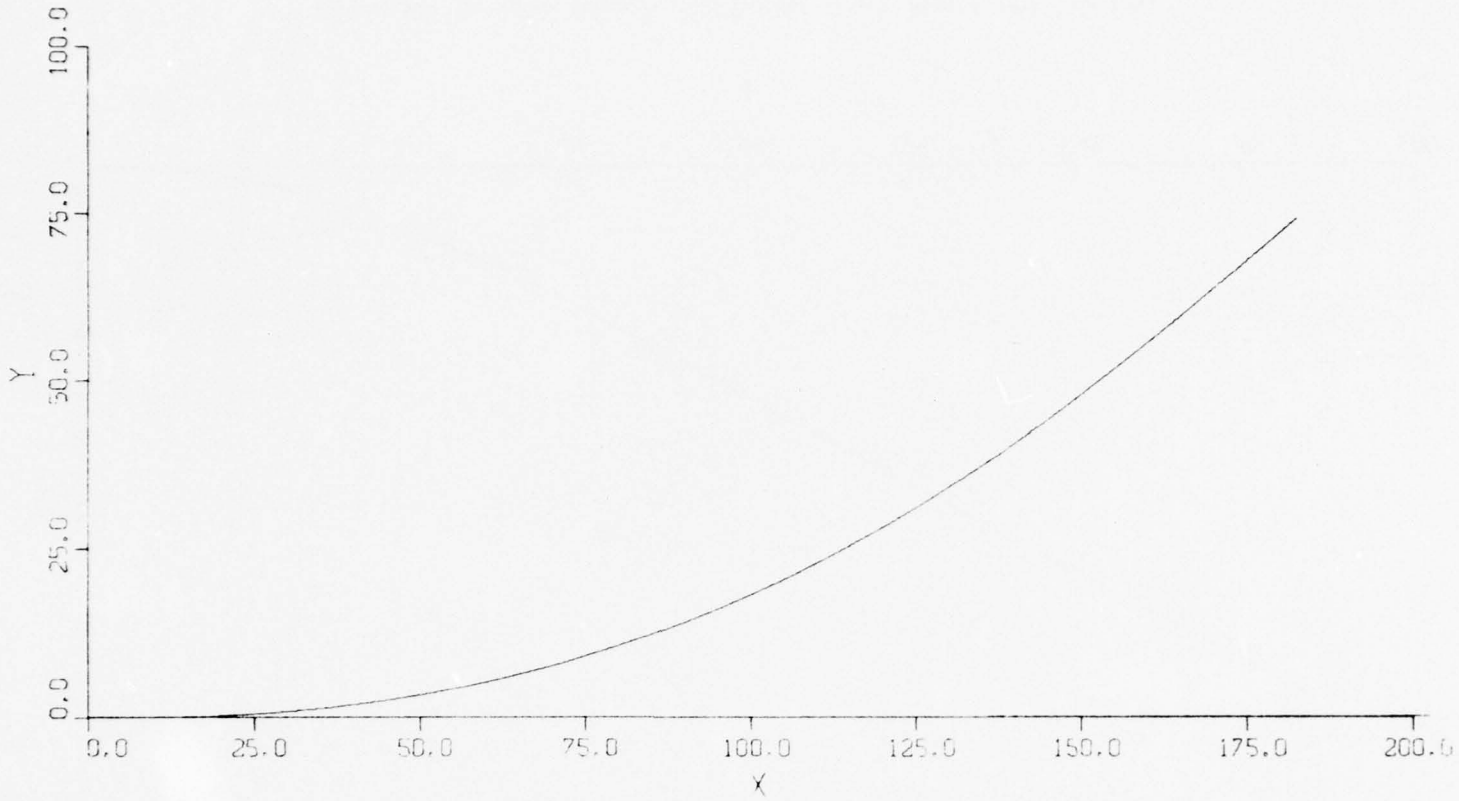



Figure 7.39. Fixed-Time Torpedo Control Problem. Case E: SE Method, Aimpoint Head of Torpedo, x-Trajectory.

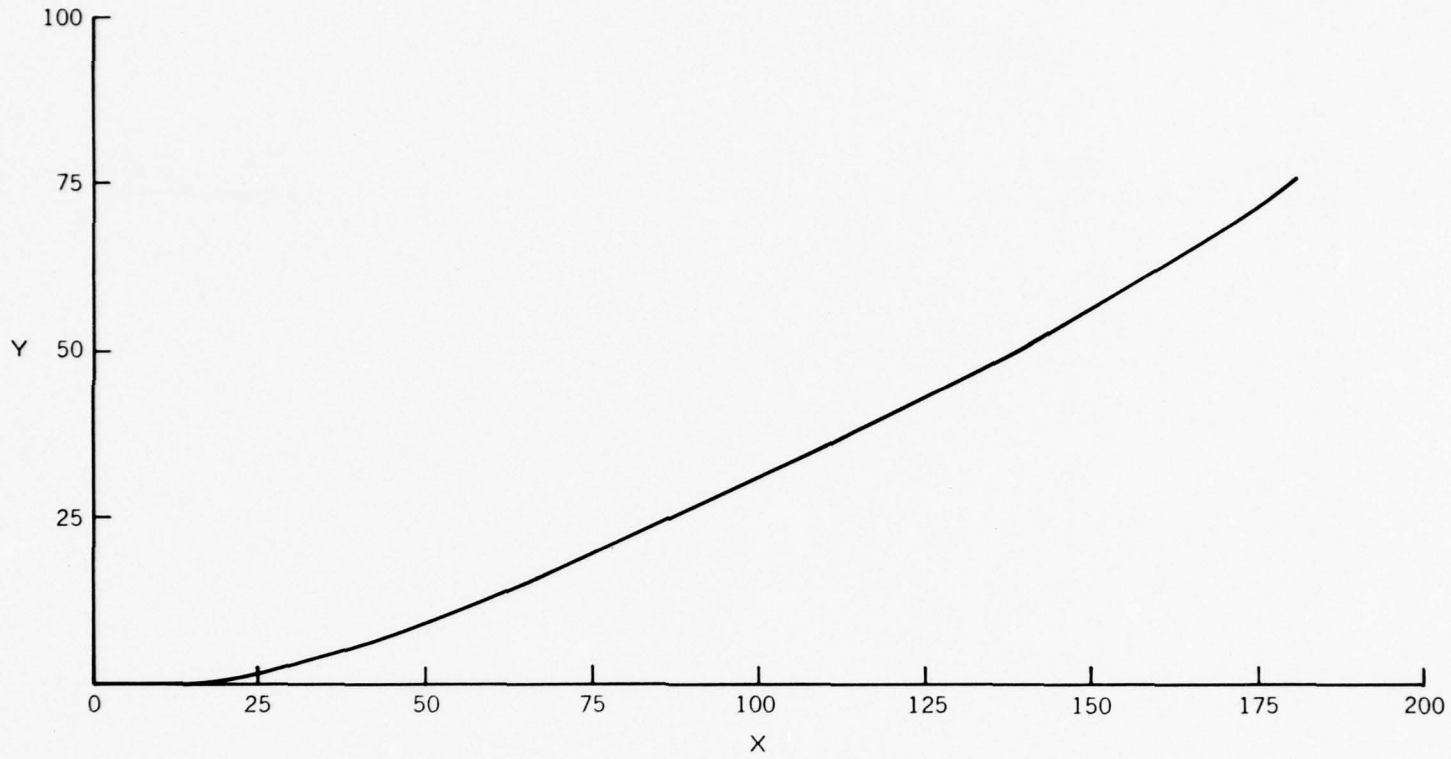



Figure 7.40. Fixed-Time Torpedo Control Problem. Case E: CG Method, Aimpoint Head of Torpedo, Initial Trajectory, 0.4 sec Curve, 2.6 sec Straight.

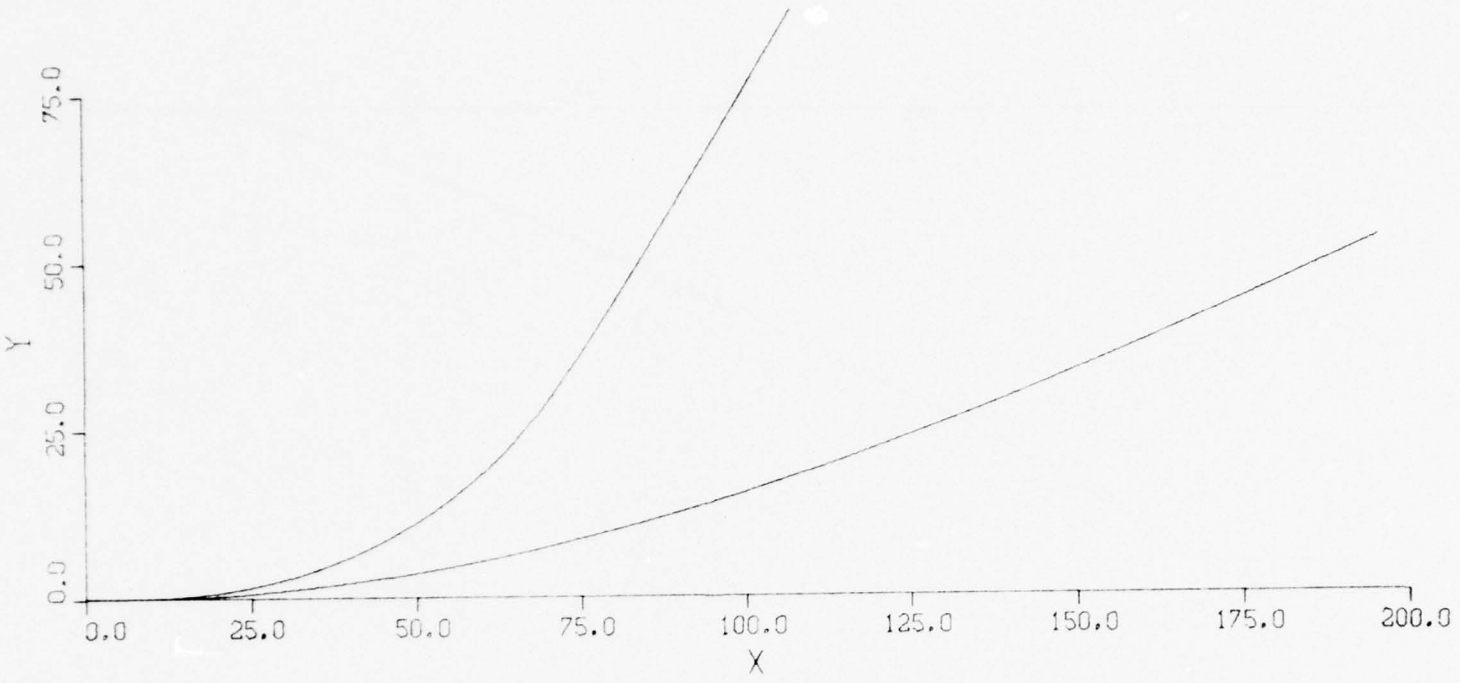



Figure 7.41. Fixed-Time Torpedo Control Problem. Case E: ME Method, Aimpoint Head of Torpedo.

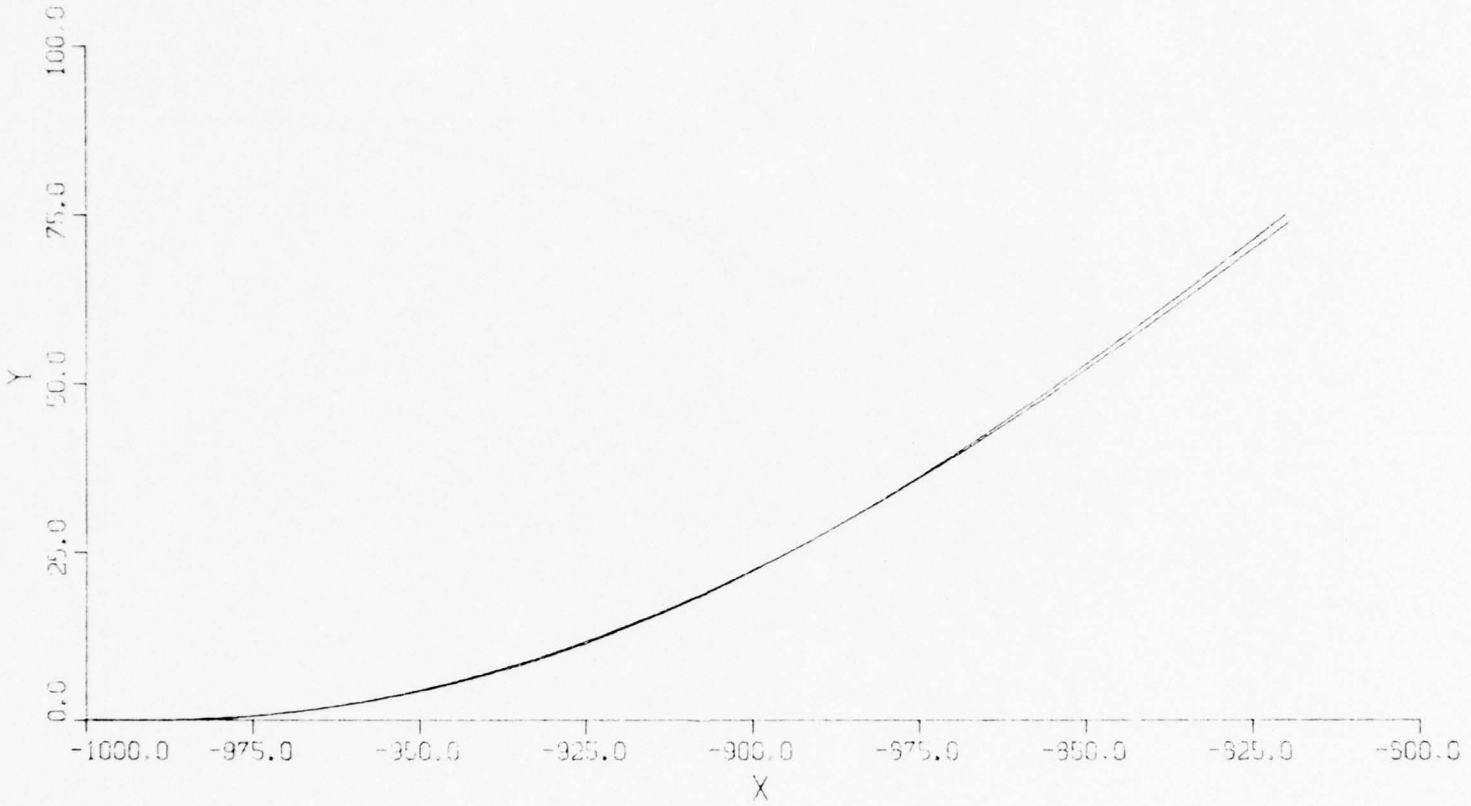



Figure 7.42. Fixed-Time Torpedo Control Problem. Case E: LIN Method, $\lambda = 60{,}000$.

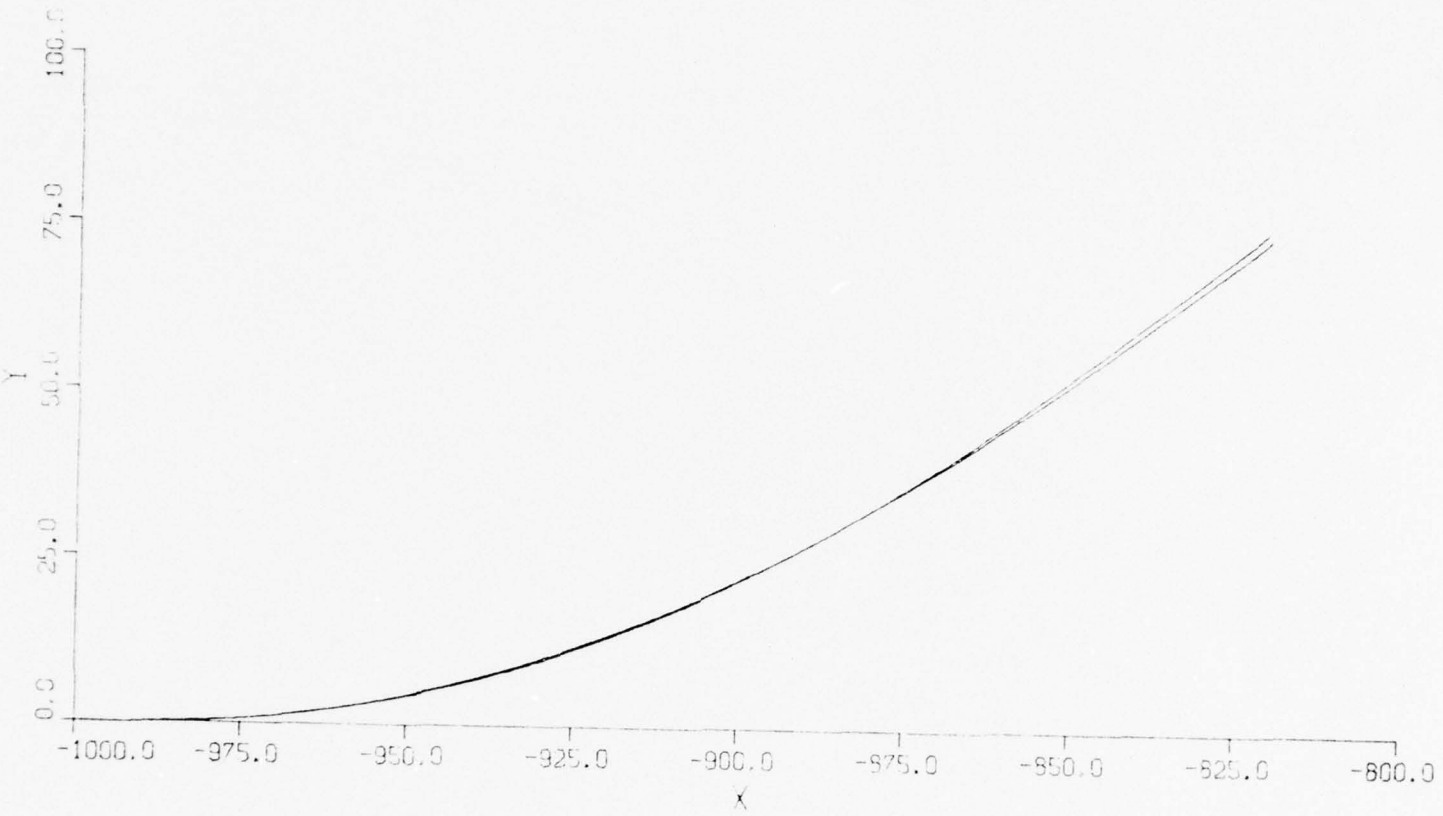



Figure 4.43. Fixed-Time Torpedo Control Problem. Case E: LIN Method, $\lambda = 30{,}000$.

TABLE 7.6: Time Optimal Torpedo Control, Long Range

| | | | | | | |
|---|---|---|---|---|---|---|
| No. Basis Function | 6 | 6 | 6 | 3 | 3 | 11 |
| No. of Sample Points | 10 | 10 | 10 | 10 | 10 | 10 |
| Initial Endpoint | Free | Fixed | Fixed | Fixed | Fixed | Fixed |
| Final Endpoint | Free | Free | Free | Free | Free | Free |
| Initial Heading | 0. | 0. | 0. | 0. | 0. | 0. |
| Speed of Target | 20. | 0. | 20. | 0. | 20. | 20. |
| Figures | 7.44 | | 7.45 | | 7.46 | 7.47 |
| $t_f$ | 15.496168 | 14.802 | 15.793 | 14.802 | 15.886 | 14.287 |
| J | 15.496 | 14.802 | 18.405 | 14.802 | 36.779 | 14.6 |
| De/ε | 0.33379E-4 | 0.199E-4 | 2.6089 | 0.19193E-4 | 20.893 | 0.25259 |
| $x_1^\varepsilon$ | 0.33E-10 | 0.117E-10 | 0.708E-6 | 0.282E-11 | 0.93E-6 | 0.7007E-6 |
| $x_2^\varepsilon$ | 0.91E-11 | 0. | -0.2286E-5 | 0. | 0.25E-5 | 0.28862E-5 |
| $x_6^\varepsilon$ | 0.30054 | 0. | 0.52268 | 0. | 0.57721 | 0.18467 |
| u | ≈0. | 0. | 0(-4) | 0. | 0(-4) | -0.16E-4 |
| | | | | | | -0.13E-1 |
| | | | | | | -0.56E-1 |
| | | | | | | 0.009 |
| | | | | | | -0.126E-1 |

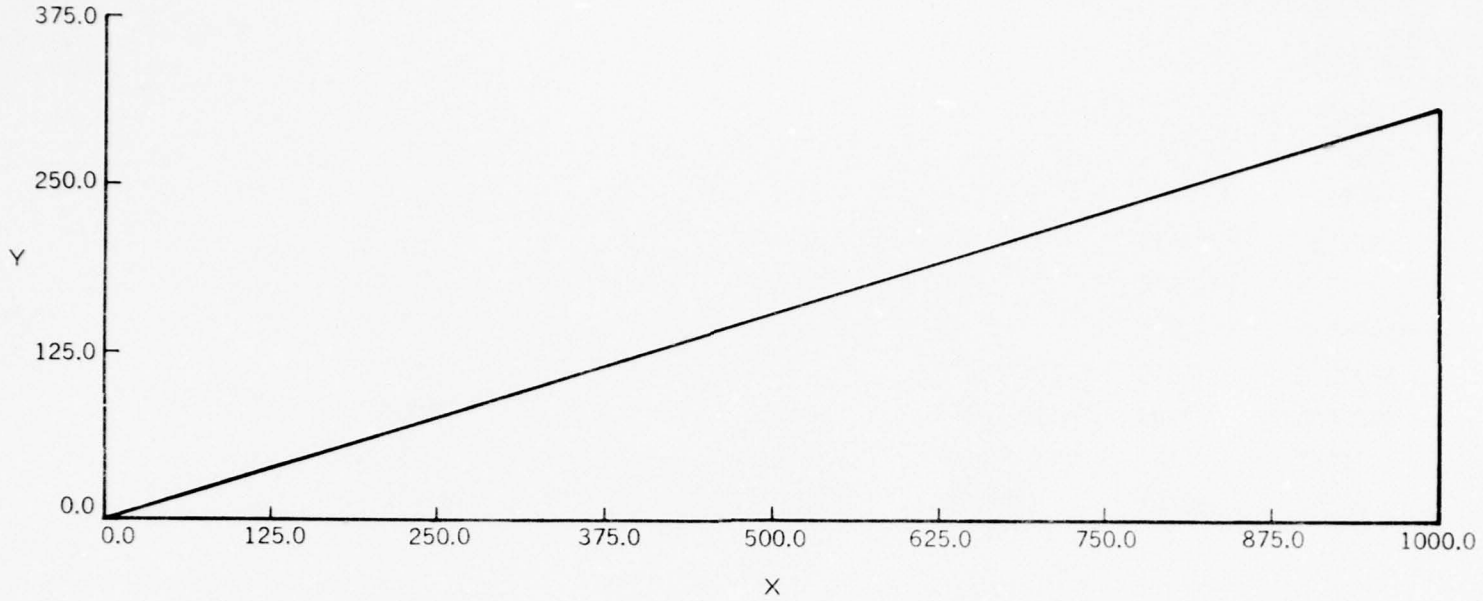



Figure 7.44 Time-Optimal Torpedo Control Problem. Free Initial and Final Endpoints, Moving Target, 6 Basis Functions, 10 Sample Points.

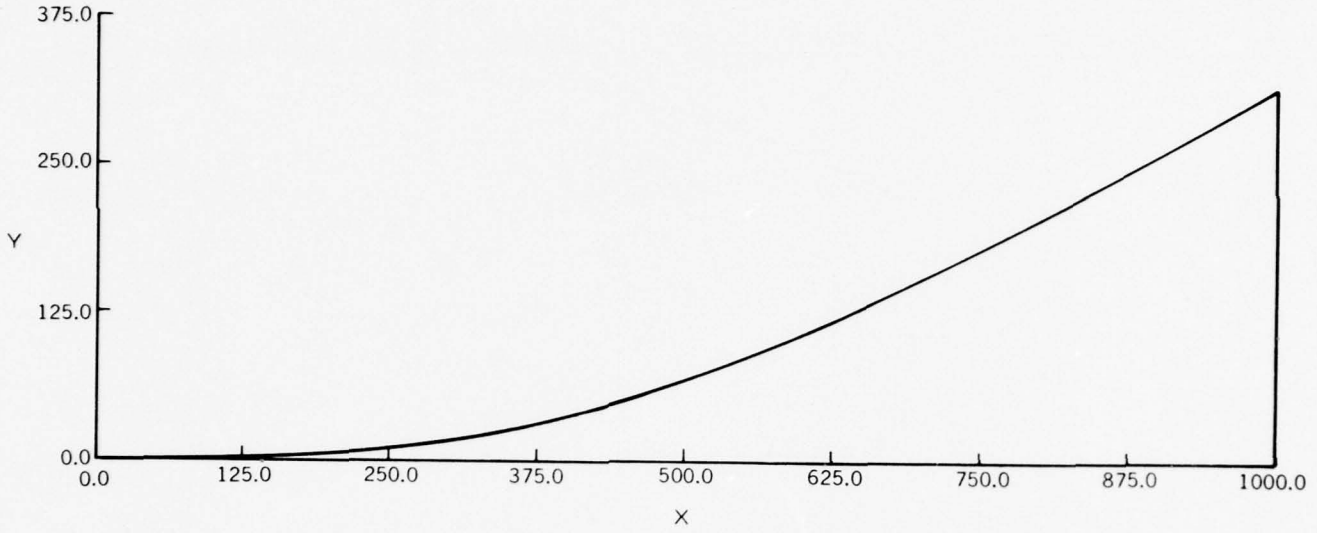



Figure 7.45. Time-Optimal Torpedo Control Problem. Free Final Endpoint, Moving Target, 6 Basis Functions, 10 Sample Points.

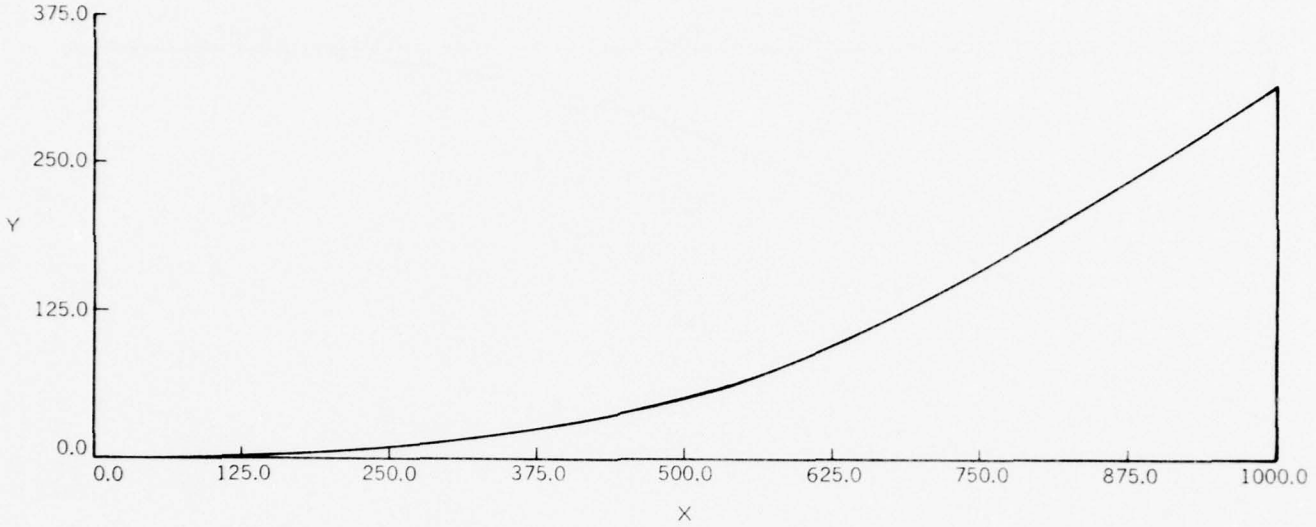



Figure 7.46. Time-Optimal Torpedo Control Problem. Free Final Endpoint, Moving Target, 3 Basis Functions, 10 Sample Points.

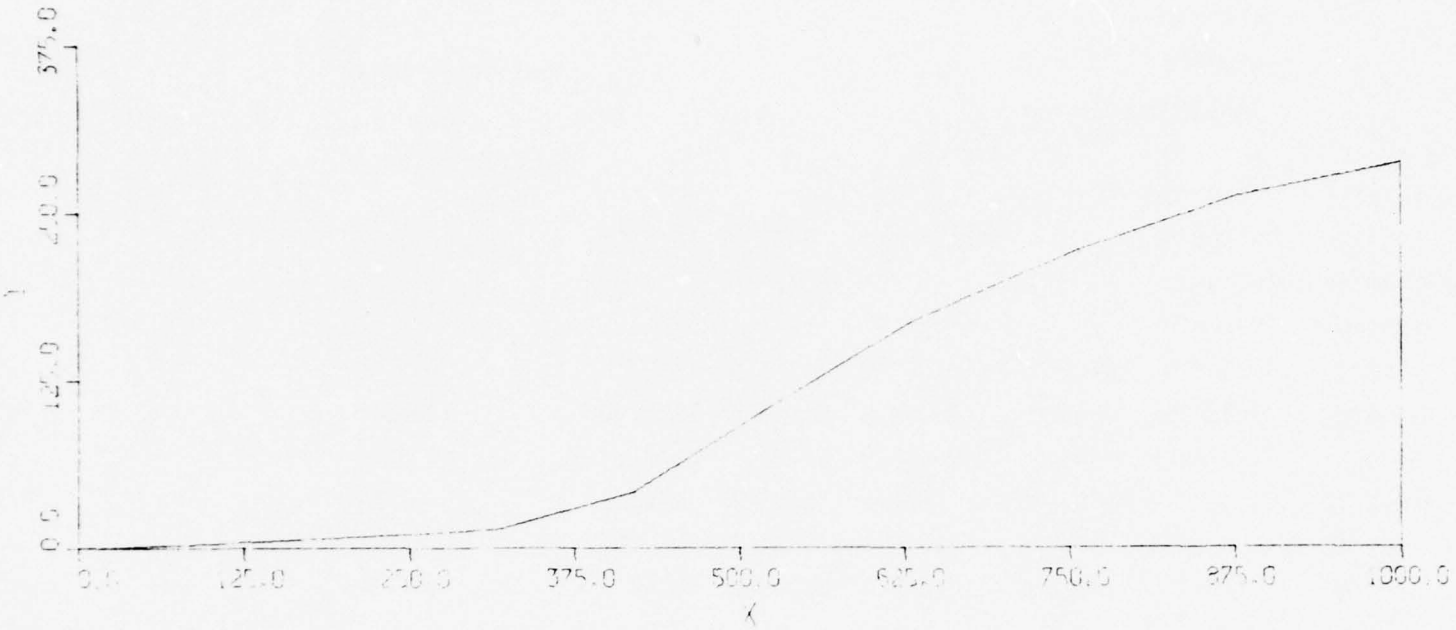



Figure 7.47. Time-Optimal Torpedo Control Problem. Free Final Endpoint, Moving Target, 11 Basis Functions, 10 Sample Points.

TABLE 7.7: Case A: Time Optimal Torpedo Control Problem

| Method | | SE | SE | SE | SE | ME | LSE,Short | LSE |
|---|---|---|---|---|---|---|---|---|
| No. Basis Function | m | 10 | 10 | 10 | 10 | 12 | 12 | 12 |
| No. of Sample Points | $N_p$ | 10 | 30 | 50 | 50 | 50 | 50 | 50 |
| Final Endpoint | | Fixed | Fixed | Fixed | Free | Free | Free | Free |
| Figures | | | 7.48 | 7.49 | 7.50 | 7.51 | 7.52 | 7.53 |
| $t_f$ | | 42.419 | 0.30024 | 0.30022 | 0.29927 | 0.29999 | 0.30213 | 0.302 |
| J | | 21878 | 28.126 | 23.125 | 4.5473 | 9.0368 | 31.439 | 31.595 |
| De/ε | | 21836 | 27.826 | 22.825 | 4.2480 | 8.7368 | 31.137 | 31.243 |
| $x_1^\varepsilon$ | | -979.89 | -979.89 | -979.89 | -979.89 | -979.89 | -979.89 | -979.89 |
| $x_2^\varepsilon$ | | 0.84681 | 0.84681 | 0.84681 | 0.84681 | 0.85414 | 0.84681 | 0.84681 |
| $x_6^\varepsilon$ | | 0.21530 | 0.21531 | 0.21531 | 0.19684 | 0.21484 | 0.2008 | 0.201 |
| $x_6^u$ | | | | | | | 0.20305 | 0.20297 |
| u | | ≃ 0(1) | -0.21,1 | -0.213,1 | -0.145,1 | -U | -0.25,1-30 | -0.25,1-30 |
| | | No Convergence | -U,i≥4 | -U,i≥8 | -0.203,15 | | -0.232,33 | -0.2,39 |
| | | Trajectory | | | -U,i≥32 | | -0.26,36 | -0.014,48 |
| | | Unreasonable | | | | | -0.18,41 | 0.258.51 |
| | | No plot | | | | | -0.26,45 | |
| | | Wild Trajectory | | | | | 0.115,49 | |
| | | | | | | | 0.259,51 | |

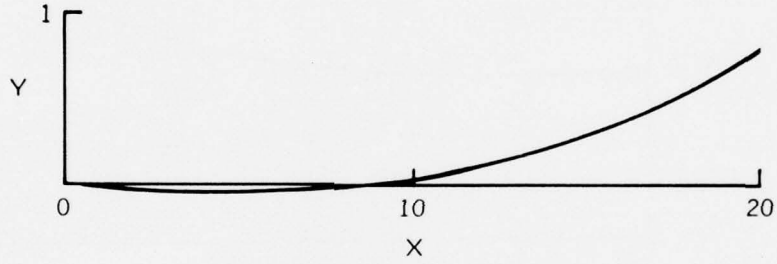



Figure 7.48. Time-Optimal Torpedo Control Problem. Case A: SE Method, 10 Basis Functions, 30 Sample Points.

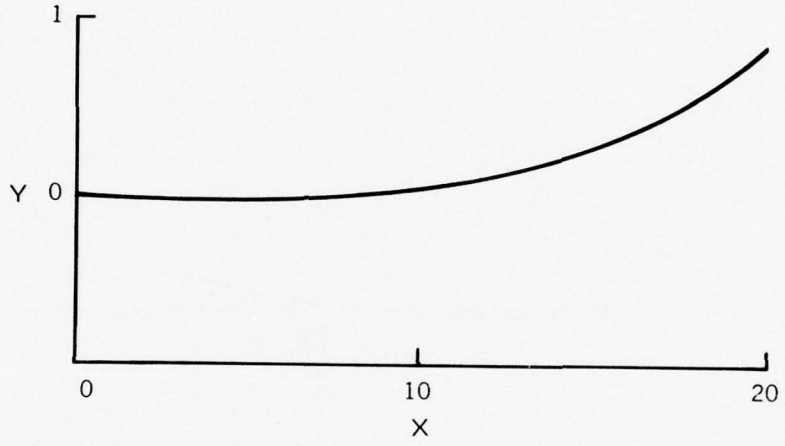



Figure 7.49. Time-Optimal Torpedo Control Problem. Case A: SE Method, 10 Basis Functions, 50 Sample Points.

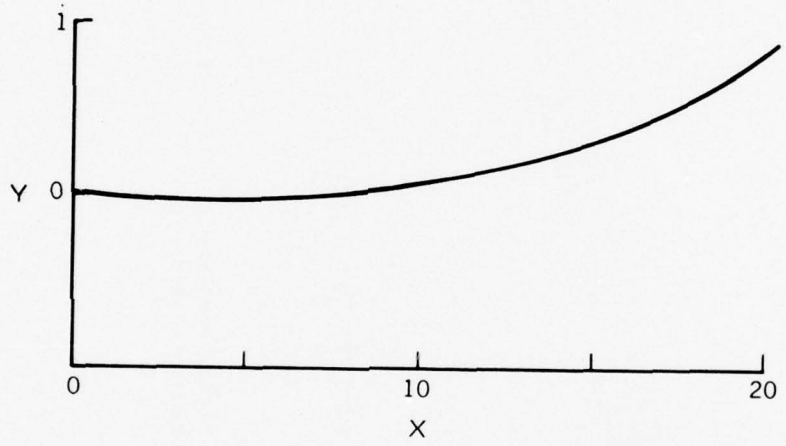



Figure 7.50. Time-Optimal Torpedo Control Problem. Case A: SE Method, 10 Basis Functions, 50 Sample Points, Free Final Endpoint.

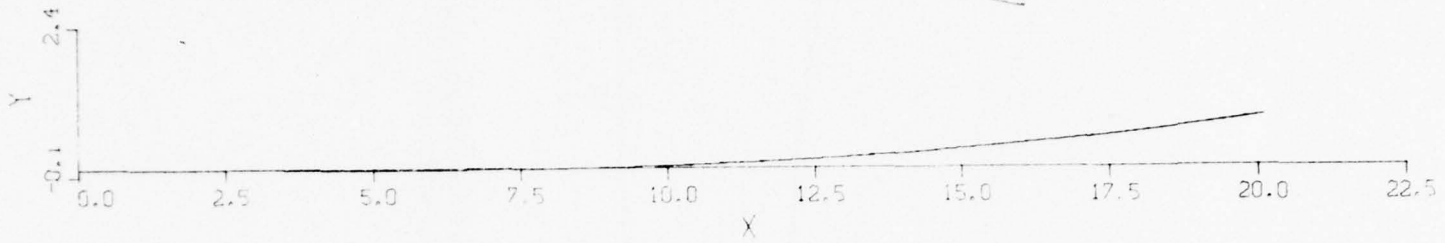



Figure 7.51. Time-Optimal Torpedo Control Problem. Case A: ME Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint.

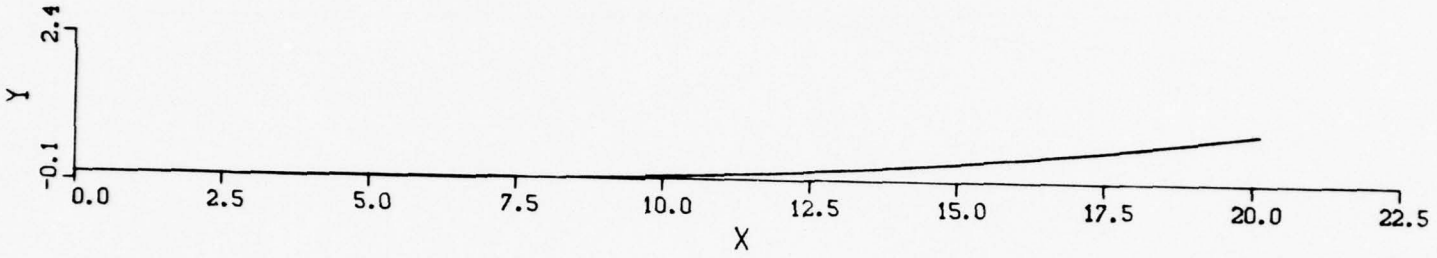



Figure 7.52. Time-Optimal Torpedo Control Problem. Case A: LSE Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint, Short Initial Final Time.

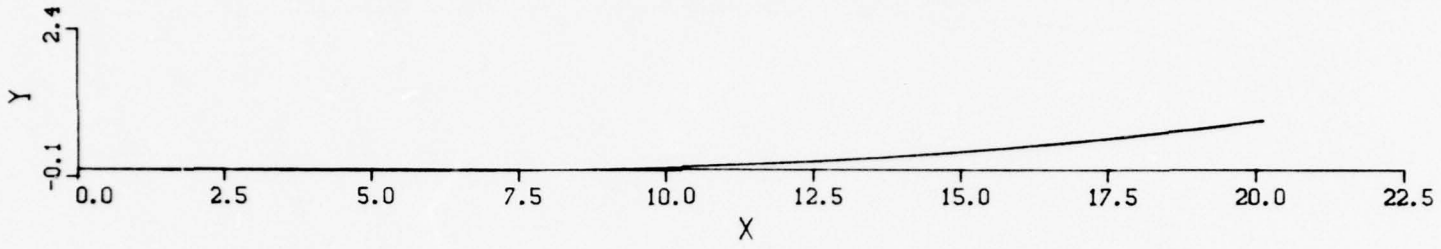



Figure 7.53. Time-Optimal Torpedo Control Problem. Case A: LSE Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoints, Exact Intial Final Time.

TABLE 7.8: Case B: Time Optimal Torpedo Control Problem

| Method | SE | SE | SE | ME | ME |
|---|---|---|---|---|---|
| No. Basis Function | 12 | 12 | 12 | 12 | 12 |
| No. Sample Points | 50 | 50 | 60 | 50 | 50 |
| Final Conditions | Free | Fixed | Free | Free | Free |
| Initial Conditions | Straight | Straight | Curved | Straight | Curved |
| Comments | | | | | |
| Figures | 7.54 | 7.55 | 7.56 | 7.57 | 7.58 |
| J | 21.797 | 12.983 | 4.2792 | 66.589 | 4.3325 |
| De/ε | 21.204 | 12.382 | 3.6792 | 65.989 | 3.7324 |
| $t_f$ | 0.59332 | 6.0137 | 0.59998 | 0.59997 | 0.60004 |
| $x_6^{\varepsilon}$ | 0.45889 | 0.4868 | 0.44466 | 0.48088 | 0.45315 |
| $x_6^{u}$ | | | | 0.48503 | 0.45512 |
| u | -0.125,1 | -0.158,1 | -U | -U | -U |
| | -0.202,19 | -0.205,5 | | | |
| | -U,29 | -U,11 | | | |

TABLE 7.8: continued

| Method | SE | SE | SE | LSE,Short | LSE |
|---|---|---|---|---|---|
| No. Basis Function | 12 | 12 | 12 | 12 | 12 |
| No. Sample Points | 50 | 50 | 50 | 51 | 51 |
| Final Conditions | Free | Free | Fixed | Free | Free |
| Initial Conditions | Straight | Straight | Straight | Straight | Straight |
| Comments | Hewett's Method | Fixed Speed | A Opt. Val. | | |
| Figures | 7.59 | 7.60 | 7.61 | 7.62 | 7.63 |
| J | 23.584 | 25.308 | 43.996 | 137.85 | 121.63 |
| De/ε | 22.991 | 24.708 | 43.394 | 137.25 | 121.03 |
| $t_f$ | 0.59349 | 0.59978 | 0.60116 | 0.60402 | 0.50411 |
| $x_6^{\varepsilon}$ | 0.45391 | 0.463(.496) | 0.4868 | 0.45369 | 0.45694 |
| $x_6^{u}$ | | 0.376 | | 0.45701 | 0.45994 |
| u | -0.095,1 | -0.098,1 | -0.2475,1 | -0.24,1 | -0.239,1 |
| | -0.202,21 | -0.202,22 | -U,i≥3 | -0.25,37 | -0.25,38 |
| | -0.25,30 | -U,35 | | -0.213,41 | -0.22,41 |
| | | | | -0.259,44 | -0.26,44 |
| | | | | 0.073,49 | 0.04,49 |
| | | | | 0.211,51 | 0.16,51 |

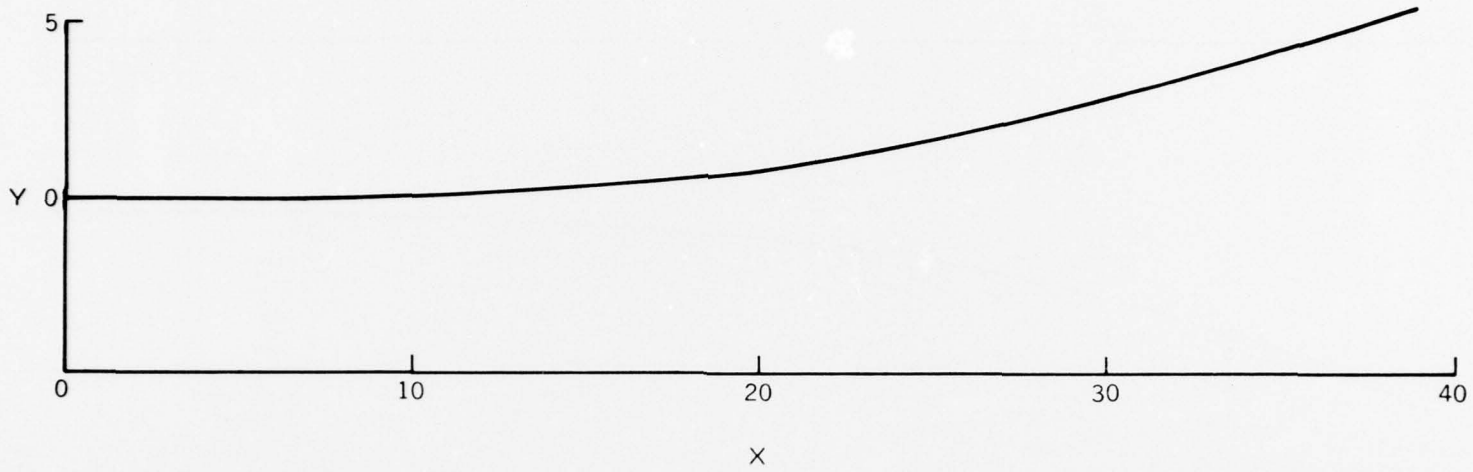



Figure 7.54. Time-Optimal Torpedo Control Problem. Case B: SE Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint.

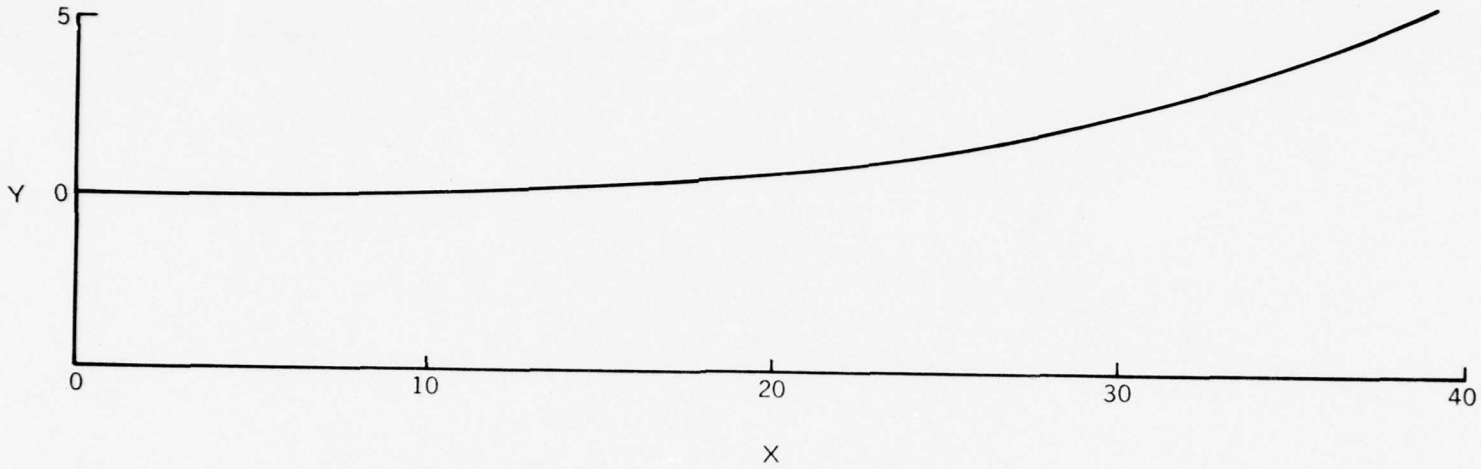



Figure 7.55. Time-Optimal Torpedo Control Problem. Case B: SE Method, 12 Basis Functions, 50 Sample Points.

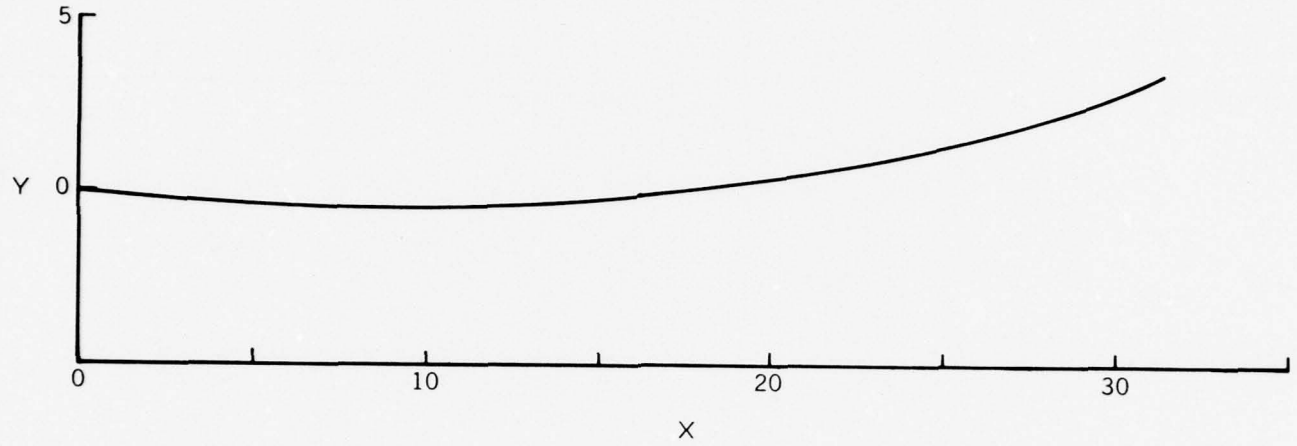



Figure 7.56. Time-Optimal Torpedo Control Problem. Case B: SE Method, 12 Basis Functions, 60 Sample Points, Free Final Endpoint, Torpedo Initially in a Curve.

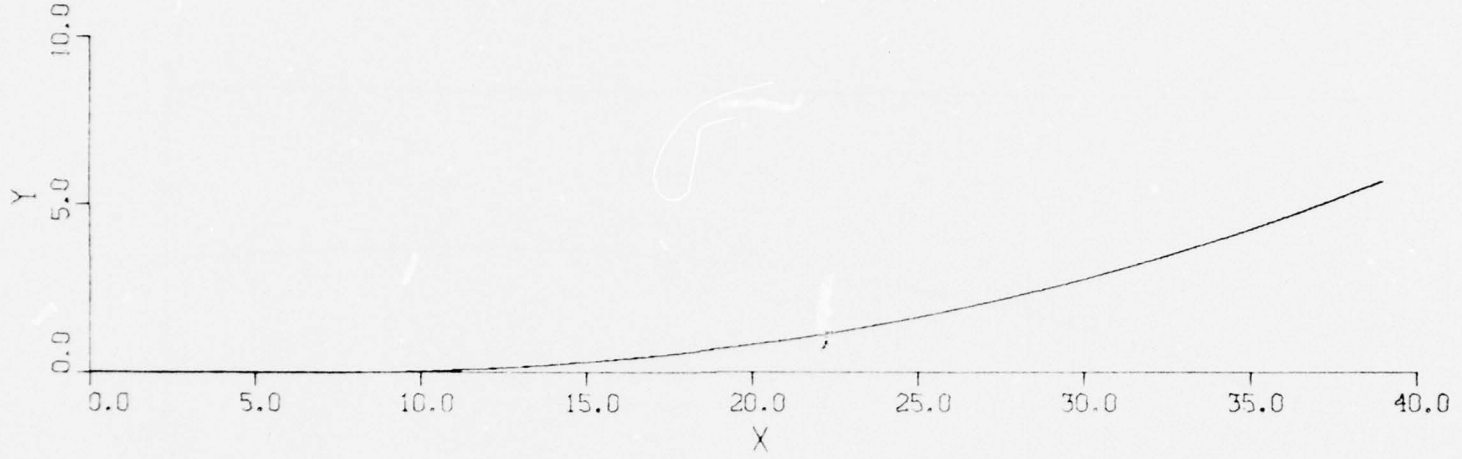



Figure 7.57. Time-Optimal Torpedo Control Problem. Case B: ME Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint.

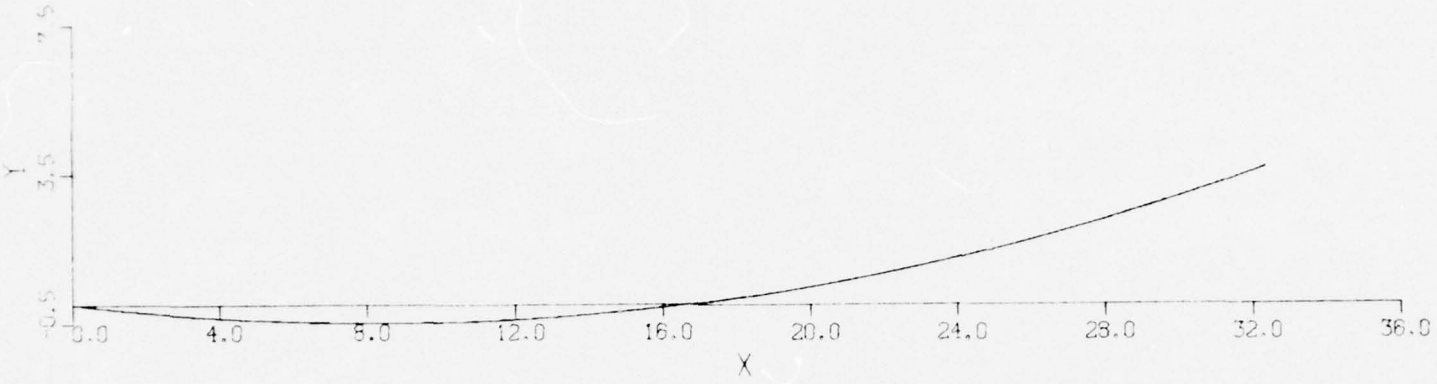



Figure 7.58. Time-Optimal Torpedo Control Problem. Case B: ME Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint, Torpedo Initially in a Curve.

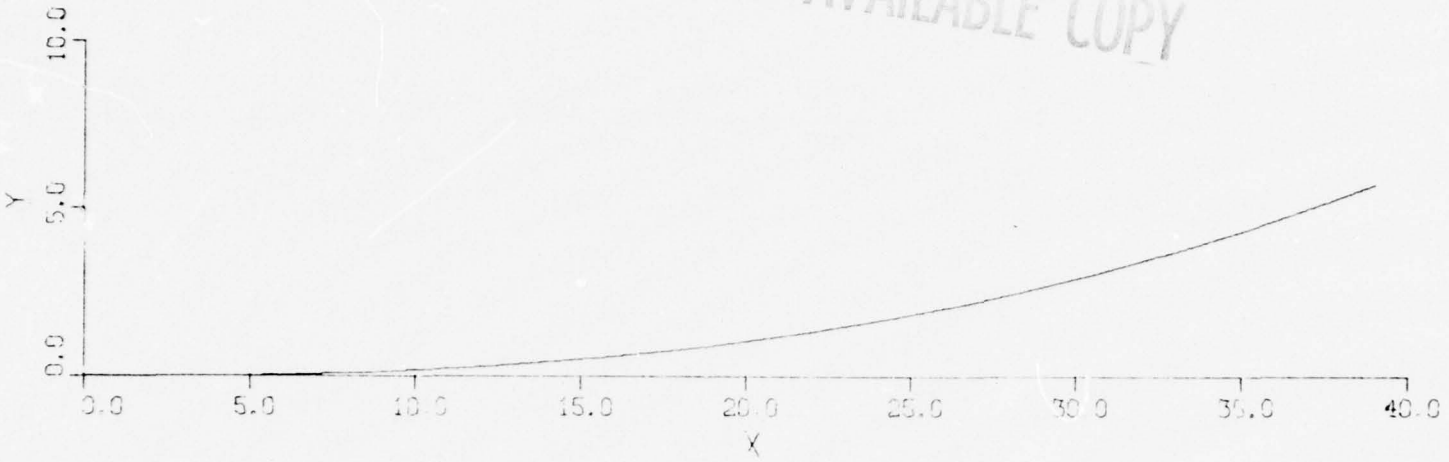



Figure 7.59. Time-Optimal Torpedo Control Problem. Case B: SE Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint, Second-Order Newton-Raphson Technique.

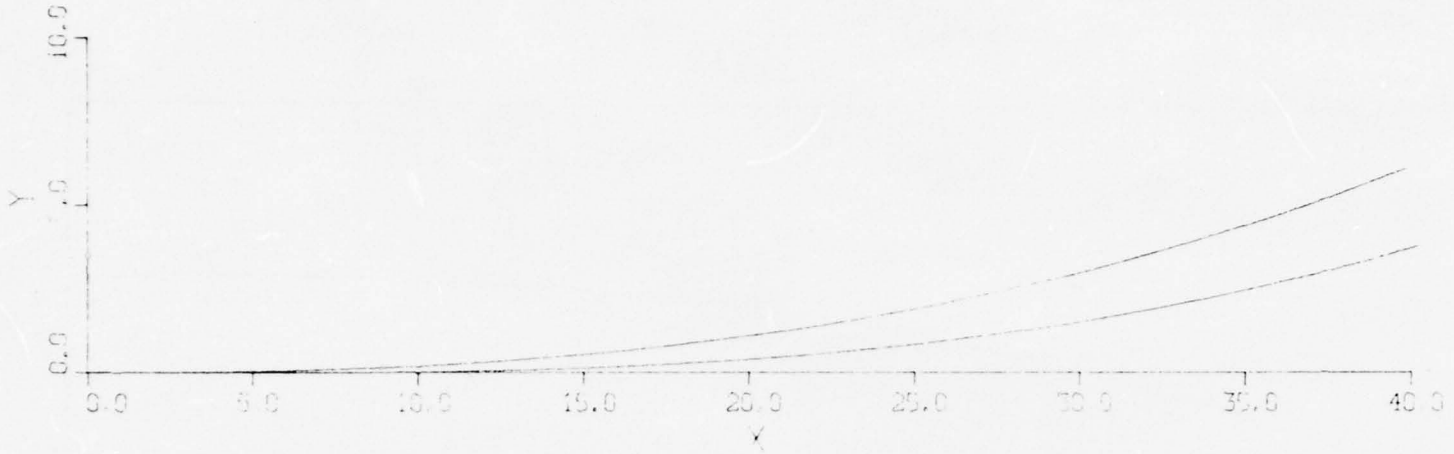



Figure 7.60. Time-Optimal Torpedo Control Problem. Case B: SE Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint, Constant Forward Velocity.

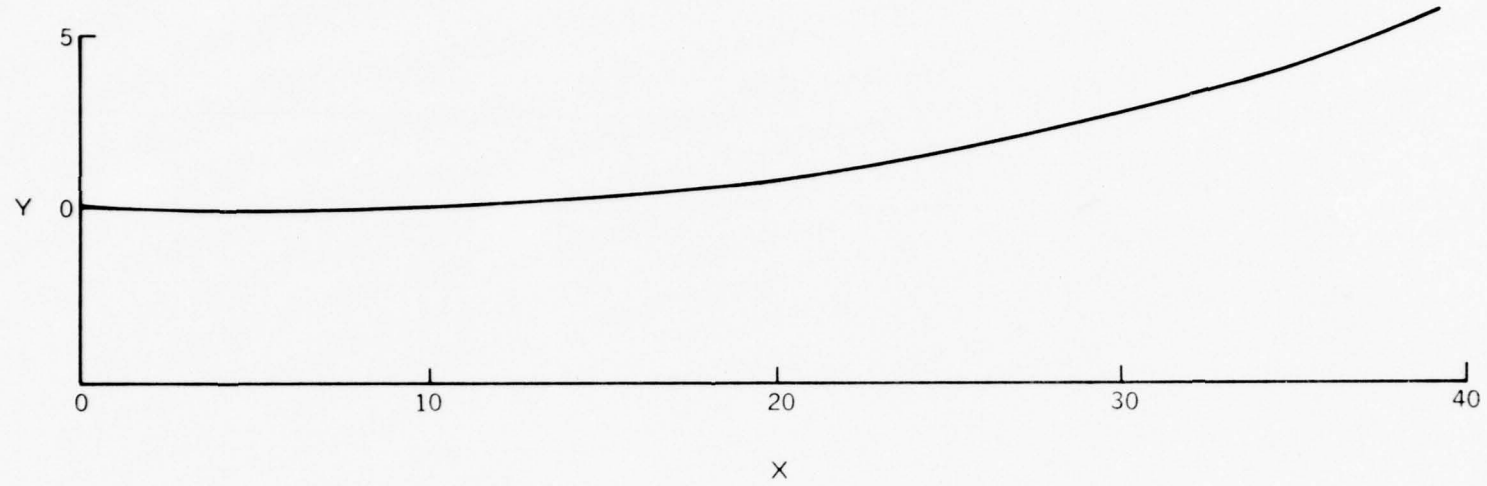



Figure 7.61. Time-Optimal Torpedo Control Problem. Case B: SE Method, 12 Basis Functions, 50 Sample Points, Optimal Initial Trajectory, Short Initial Final Time.

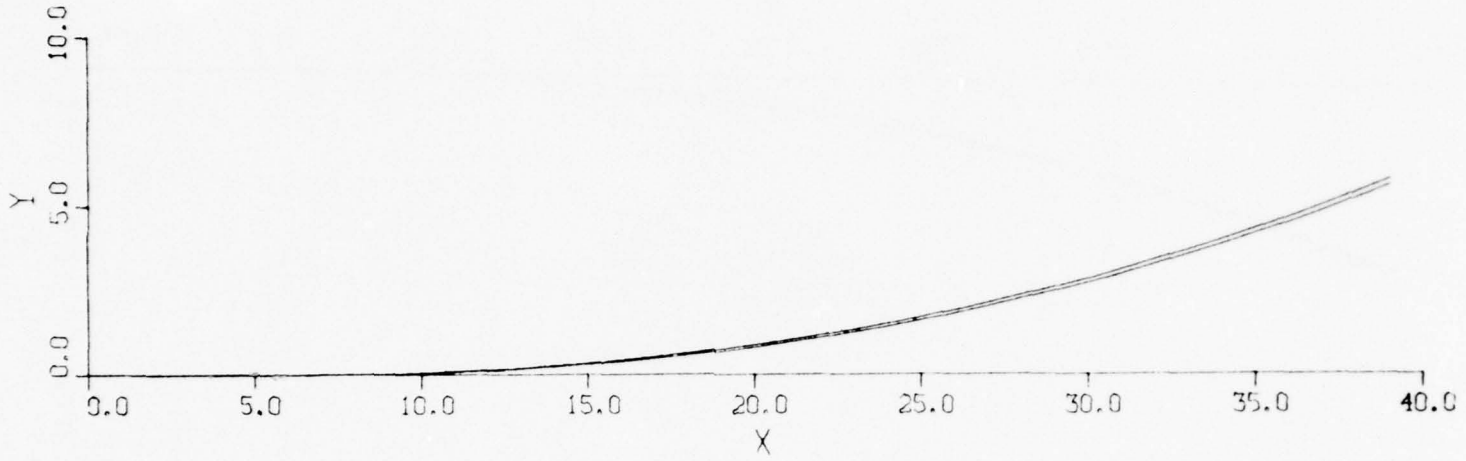



Figure 7.62. Time-Optimal Torpedo Control Problem. Case B: LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint, Short Initial Final Time.

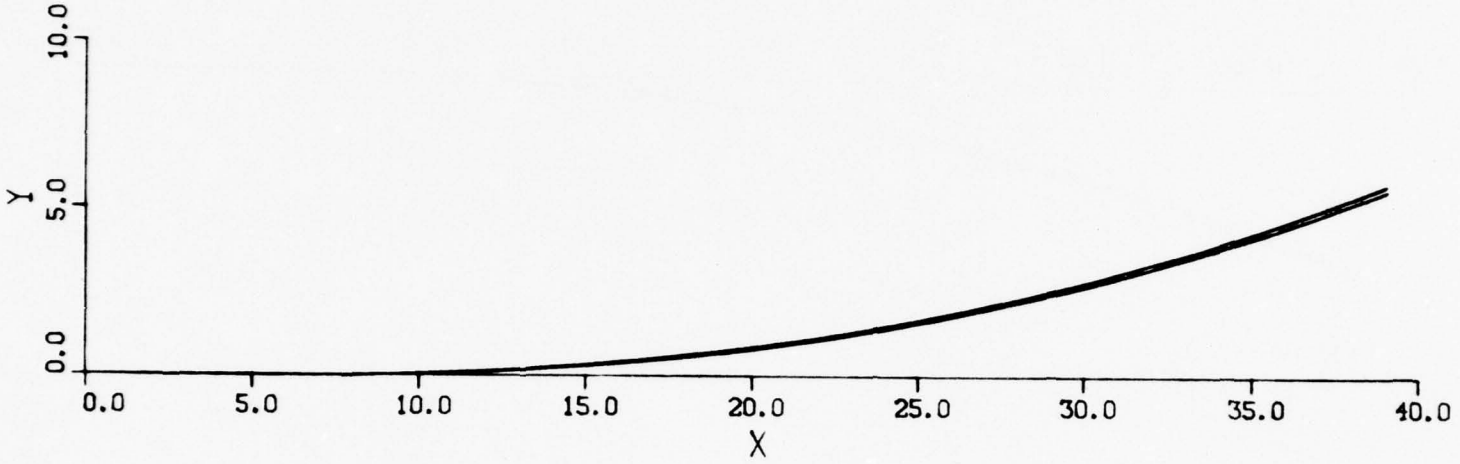



Figure 7.63. Time-Optimal Torpedo Control Problem. Case B: LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint, Exact Initial Final Time.

TABLE 7.9: Case C: Time Optimal Torpedo Control Problem

| Method | SE | SE | SE | ME | SE |
|---|---|---|---|---|---|
| No. Basis Function | 12 | 12 | 12 | 12 | 12 |
| No. Sample Points | 60 | 60 | 50 | 50 | 50 |
| Final Condition | Free | Free | Fixed | Free | Fixed |
| Initial Condition | Straight | Curved | Straight | Straight | Straight |
| Comments | | | | | Steepest Descent |
| Figures | 7.64 | 7.65 | 7.66 | 7.67 | 7.68 |
| J | 47.515 | 17.754 | 949.3 | 70.758 | 245.90 |
| De/ε | 46.923 | 17.163 | 948.7 | 70.153 | 245.30 |
| $t_f$ | 0.59180 | 0.59110 | 0.59877 | 0.60537 | 0.59826 |
| $x_6^{\varepsilon}$ | 0.27638 | 0.26248 | 0.27906 | 0.45560 | 0.27906 |
| $x_6^{u}$ | | | | 0.49036 | |
| u | -0.04,1 | -0.17,1 | -0.089,1 | -U | -0.051,1 |
| | -0.107,60 | -0.103,22 | -0.183,17 | | -0.103,15 |
| | | -0.013,53 | -0.184,23 | | -0.128,31 |
| | | -0.0014,57 | -0.099,35 | | -0.102,38 |
| | | 0.005,60 | 0.0005,42 | | 0.003,45 |
| | | | 0.063,46 | | 0.089,50 |
| | | | 0.057,49 | | |

TABLE 7.9: continued

| Method | SE | SE | LSE, Short | LSE |
|---|---|---|---|---|
| No. Basis Function | 12 | 12 | 12 | 12 |
| No. Sample Points | 50 | 50 | 51 | 51 |
| Final Condition | Free | Fixed | Free | Free |
| Initial Condition | Straight | Straight | Straight | Straight |
| Comments | A Initialized Option Values | A Initialized Option Values | | |
| Figures | 7.69 | 7.70 | 7.71 | 7.72 |
| J | 50.477 | 180.70 | 82.884 | 80.165 |
| De/ε | 49.880 | 180.10 | 82.280 | 79.561 |
| $t_f$ | 0.59648 | 0.59963 | 0.6042 | 0.60361 |
| $x_6^{\varepsilon}$ | 0.23504 | 0.27906 | 0.31828 | 0.31392 |
| $x_6^{u}$ | 0.18751 | 0.25338 | 0.32109 | 0.31102 |
| u | -0.153,1 | -0.08,1 | -0.247,1 | -0.246,1 |
| | -0.209,3 | -0.10,3 | -0.2601,4 | -0.206,15 |
| | -0.259,5 | -0.204,11 | -0.205,20 | -0.1165,31 |
| | -0.159,9 | -0.259,17 | -0.088,33 | -0.036,41 |
| | -0.259,13 | -0.19,27 | -0.122,37 | -0.131,45 |
| | -0.137,17 | 0.005,37 | -0.033,41 | 0.022,48 |
| | -0.260,21 | 0.05,43 | -0.136,45 | 0.259,51 |
| | 0.0056,26 | -0.006,50 | 0.017,48 | |
| | ± 0(-1) | | 0.259,51 | |

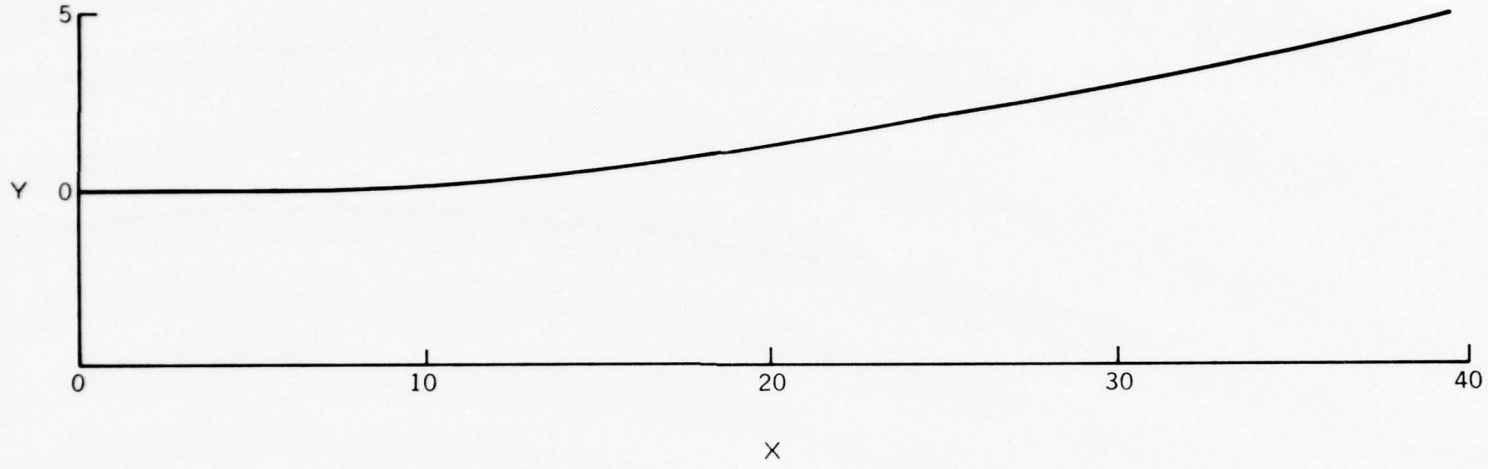



Figure 7.64. Time-Optimal Torpedo Control Problem. Case C: SE Method, 12 Basis Functions, 60 Sample Points, Free Final Endpoint.

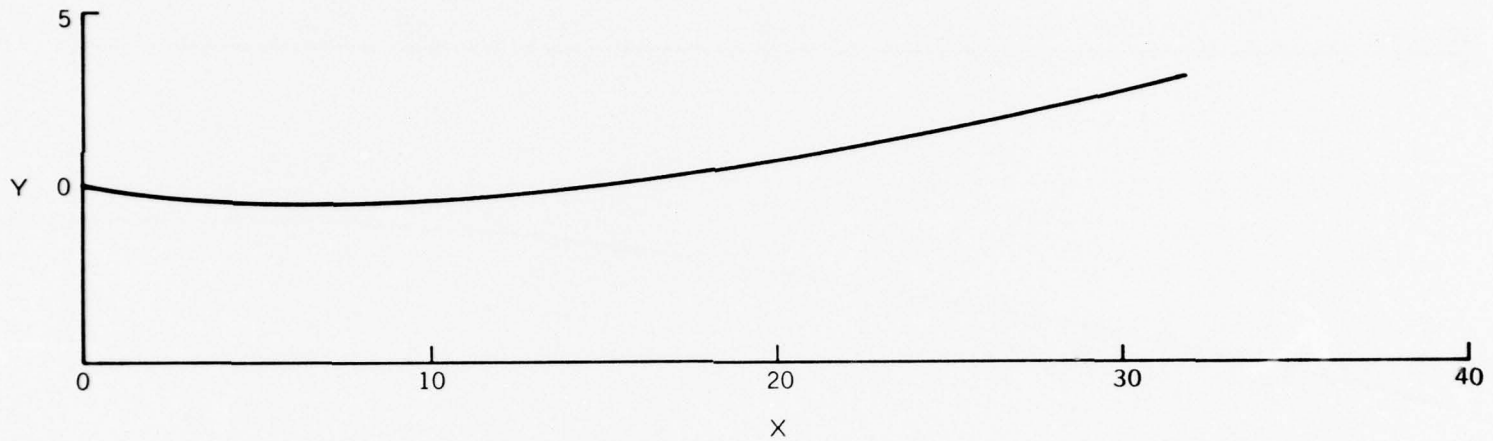



Figure 7.65. Time-Optimal Torpedo Control Problem. Case C: SE Method, 12 Basis Functions, 60 Sample Points, Free Final Endpoint, Torpedo Initially in a Curve.

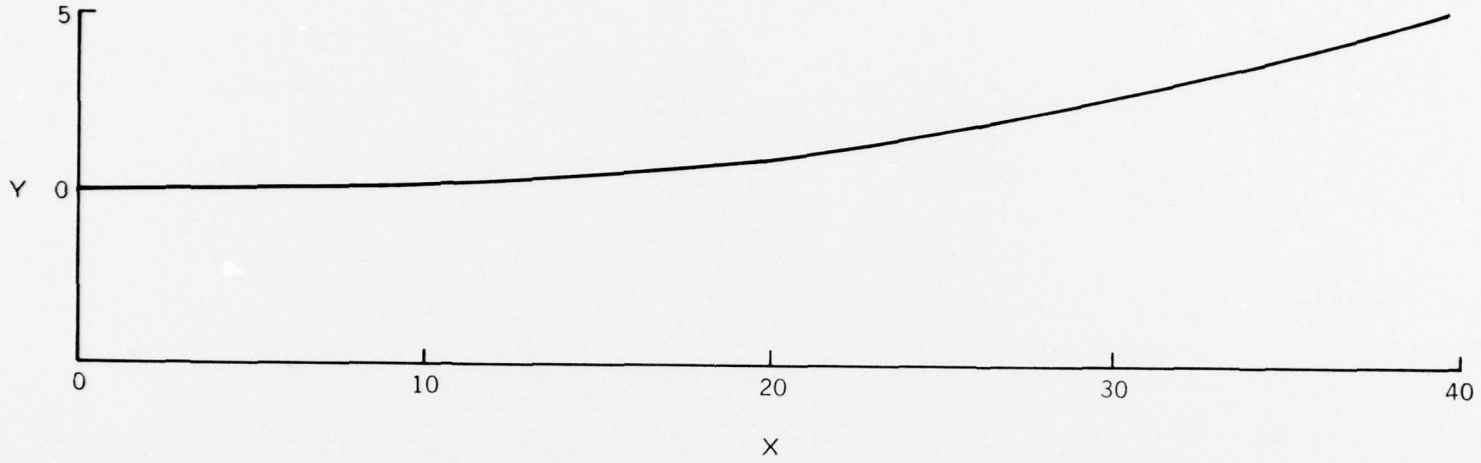



Figure 7.66. Time-Optimal Torpedo Control Problem. Case C: SE Method, 12 Basis Functions, 50 Sample Points.

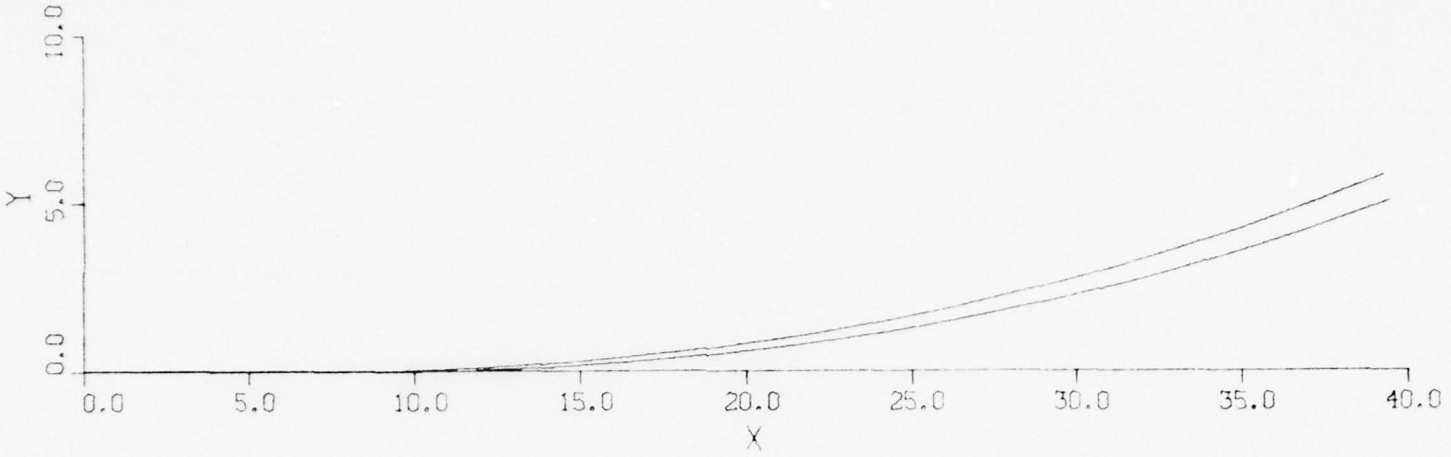



Figure 7.67. Time-Optimal Torpedo Control Problem. Case C: ME Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint.

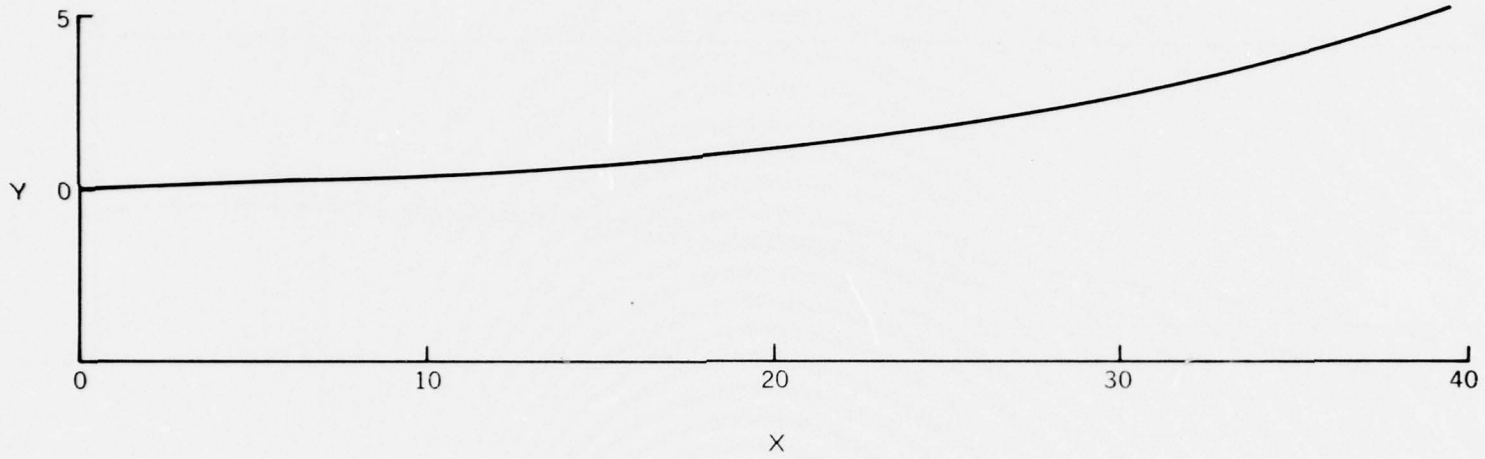



Figure 7.68. Time-Optimal Torpedo Control Problem. Case C: SE Method, 12 Basis Functions, 50 Sample Points, Method of Steepest Descent Used for Nine Iterations, x-Trajectory.

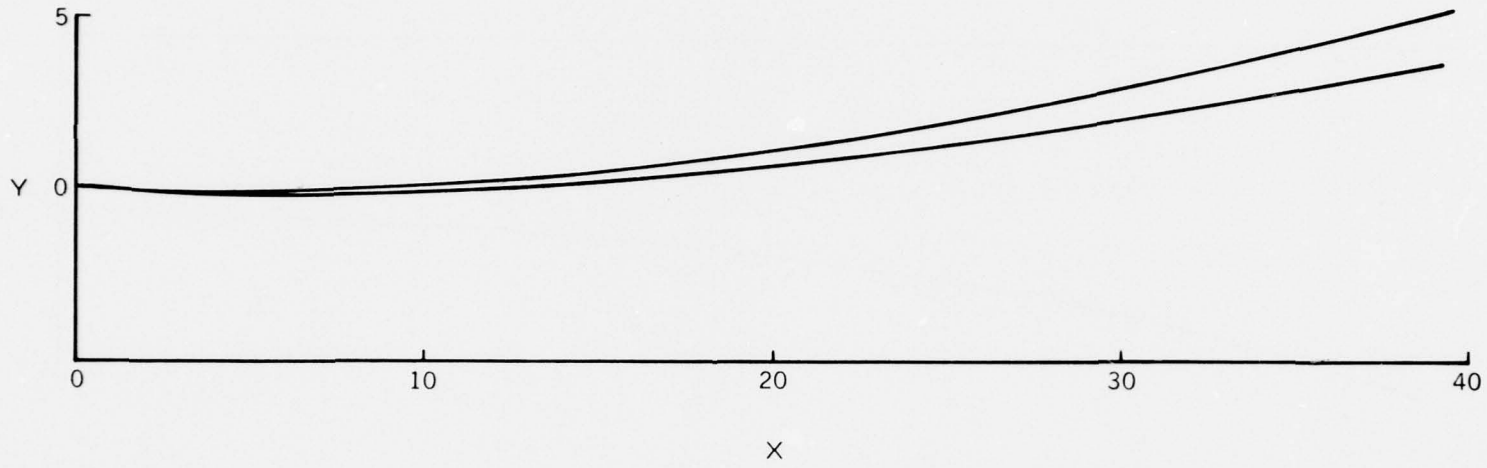



Figure 7.69. Time-Optimal Torpedo Control Problem. Case C: SE Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint, Optimal Initial Trajectory, Short Initial Final Time.

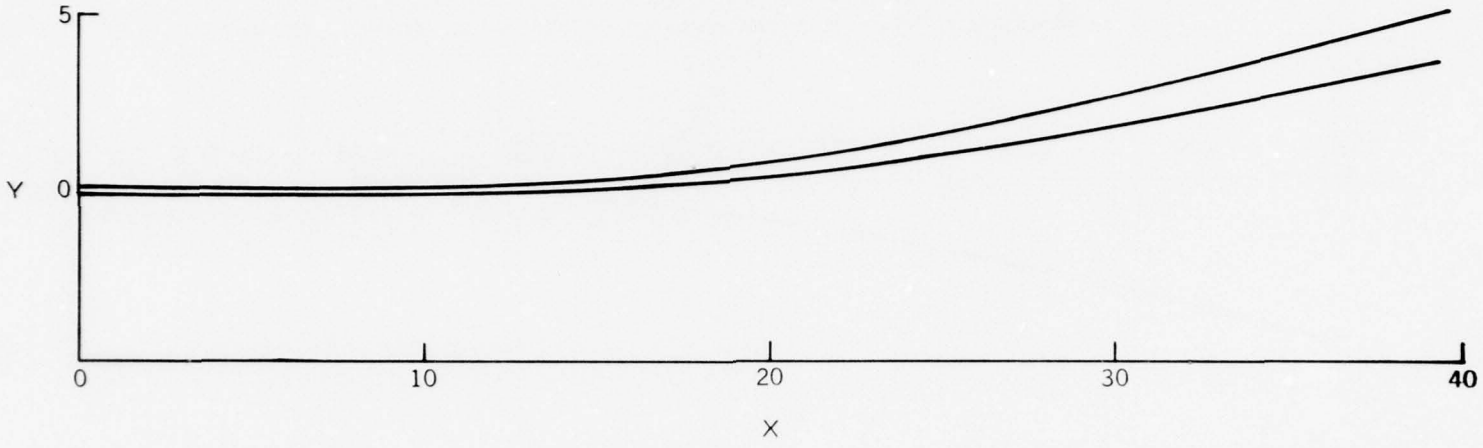



Figure 7.70. Time-Optimal Torpedo Control Problem. Case C: SE Method, 12 Basis Functions, 50 Sample Points, Optimal Initial Trajectory, Short Initial Final Time.

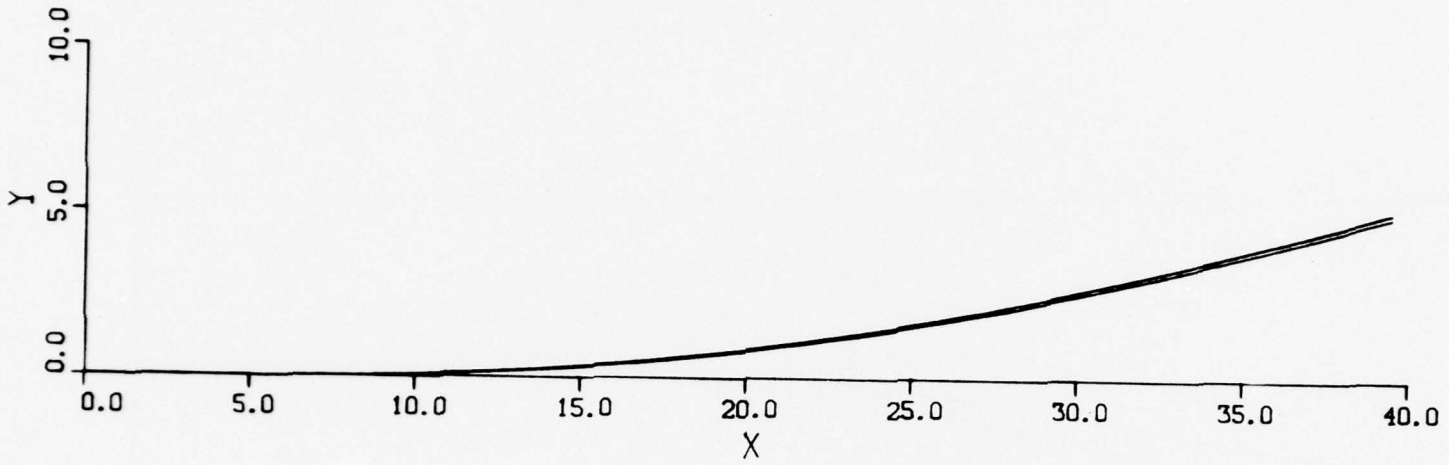



Figure 7.71. Time-Optimal Torpedo Control Problem Case C: LSE Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint, Short Initial Final Time.

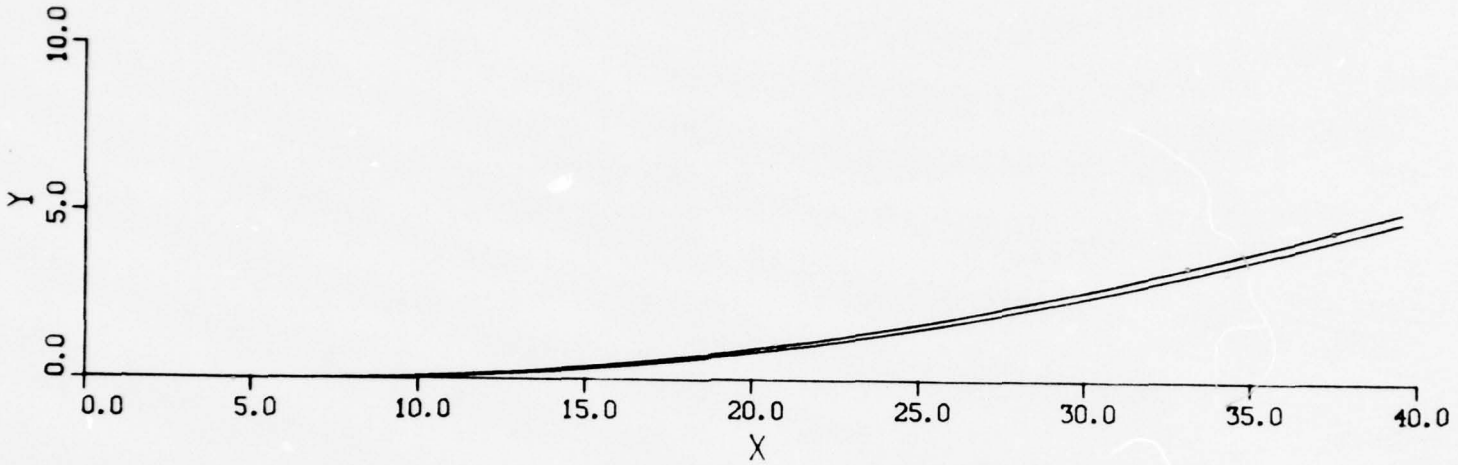



Figure 7.72. Time-Optimal Torpedo Control Problem. Case C: LSE Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint, Exact Initial Final Time.

TABLE 7.10: Case D: Time Optimal Torpedo Control Problem

| Method | SE | SE | SE | SE | ME | LSE,Short | LSE |
|---|---|---|---|---|---|---|---|
| No. Basis Function | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| No. Sample Points | 60 | 50 | 60 | 80 | 50 | 51 | 51 |
| Initial Condition | Curved | Straight | Straight | Straight | Straight | Straight | Straight |
| Figures | 7.73 | 7.74 | 7.75 | | 7.76 | 7.77 | 7.78 |
| J | 123.21 | 662.40 | 658.17 | 652.35 | 4452.2 | 993.56 | 983.61 |
| De/ε | 120.18 | 659.65 | 655.42 | 649.61 | 4449.6 | 990.74 | 980.78 |
| $t_f$ | 3.0375 | 2.7470 | 2.7452 | 2.7425 | 2.5867 | 2.825 | 2.826 |
| $x_6^{\varepsilon}$ | 2.1973 | 2.3931 | 2.3930 | 2.3928 | 1.8617 | 2.1705 | 2.1742 |
| $x_6^{u}$ | | 1.2958 | | | 2.0992 | 2.20006 | 2.2035 |
| u | -U | -0.014,1 | -0.015,1 | -0.015,1 | -U | -0.159,1 | -0.159,1 |
| | | -0.103,18 | -0.101,21 | -0.102,28 | | -U,67 | -0.25,2-17 |
| | | -0.201,36 | -0.200,43 | -0.202,58 | | -0.25,8-47 | -0.207,51 |
| | | -0.262,50 | -0.261,60 | -0.262,80 | | -0.196,51 | |

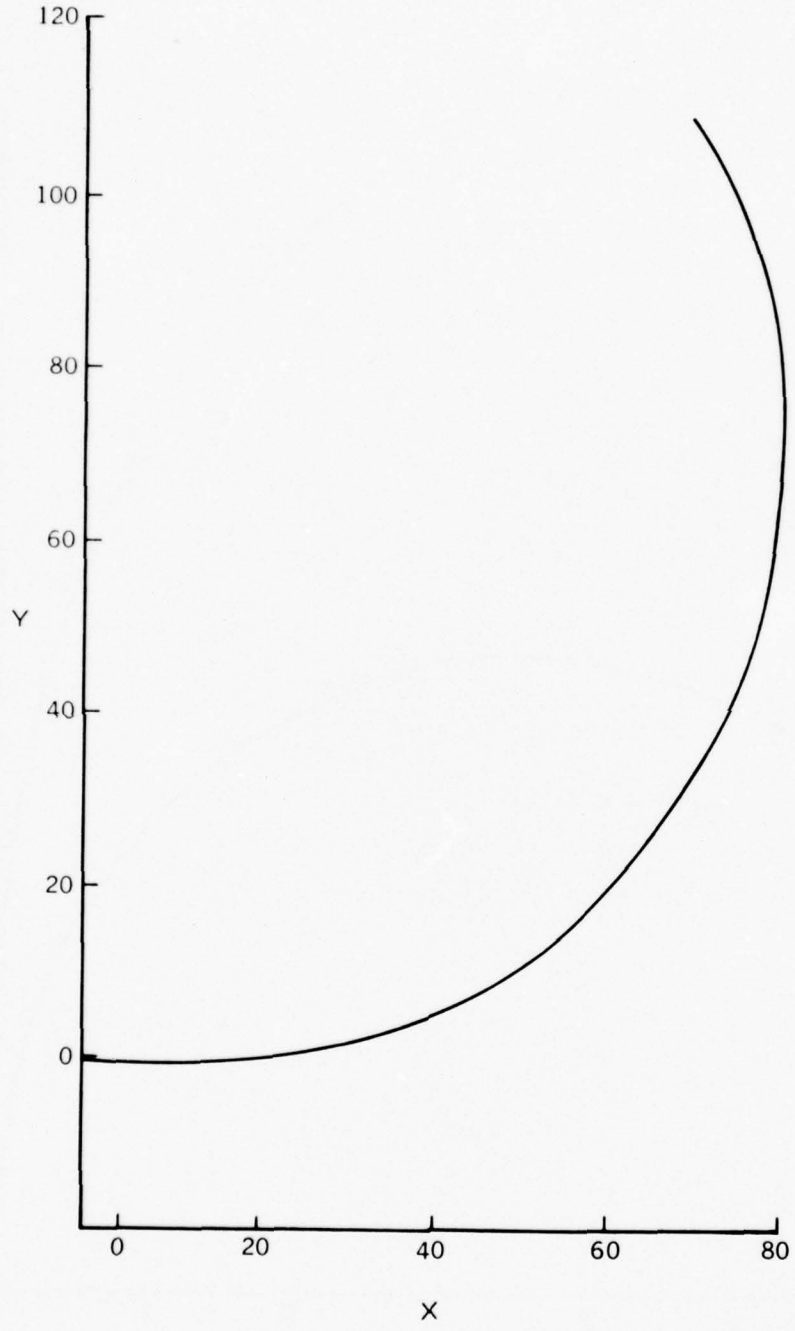



Figure 7.73. Time-Optimal Torpedo Control Problem. Case D: SE Method, 12 Basis Functions,

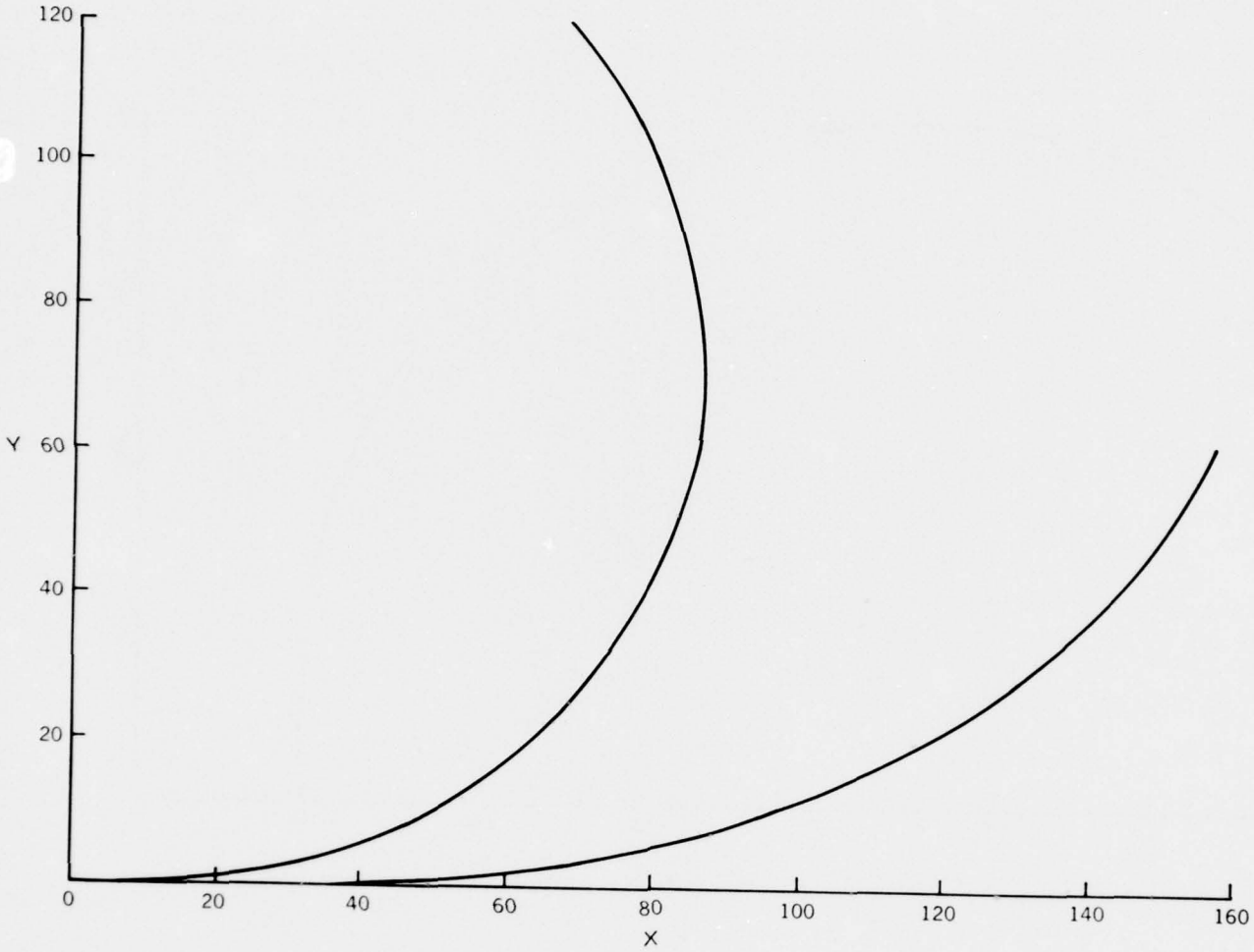



Figure 7.74. Time-Optimal Torpedo Control Problem. Case D: SE Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint.

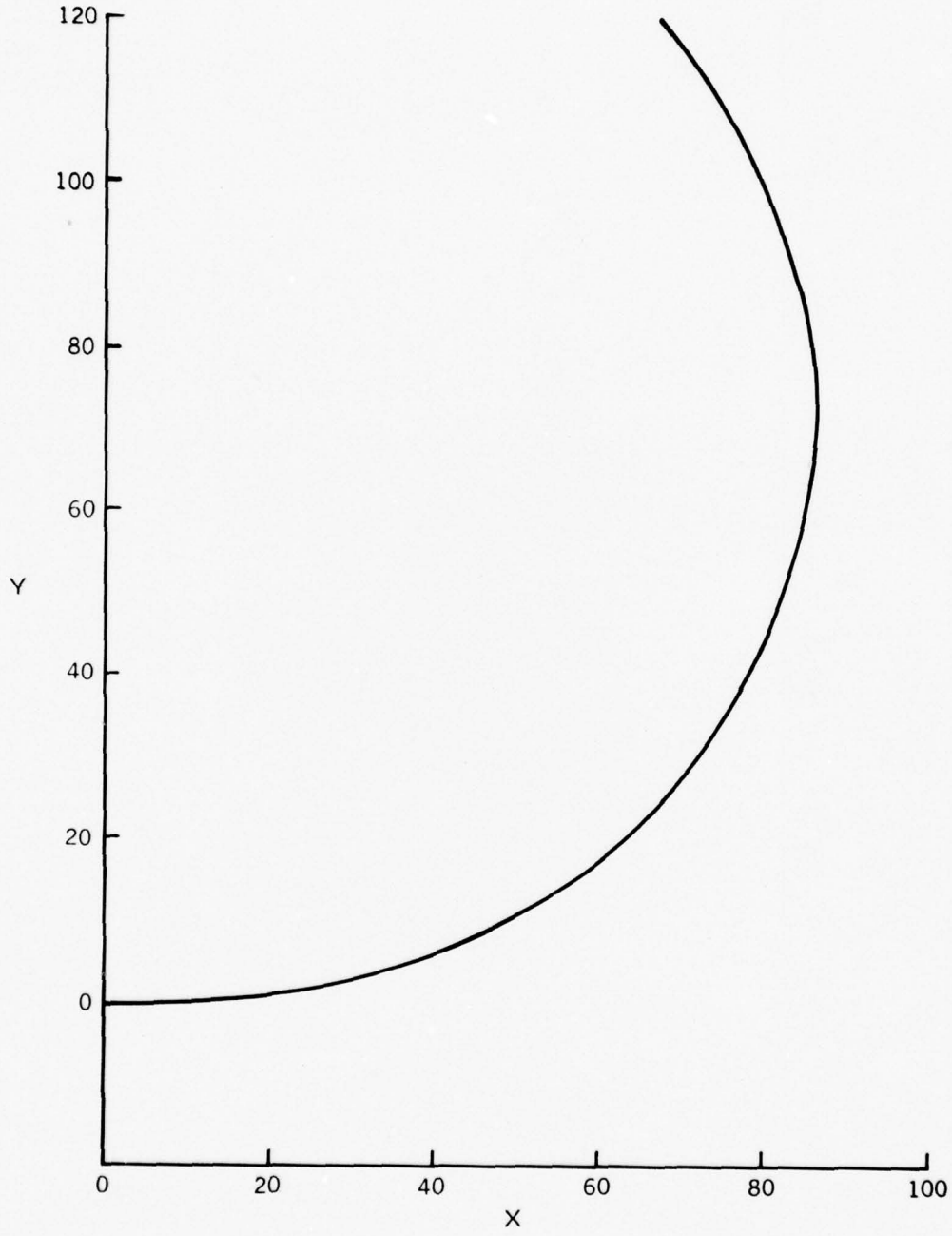



Figure 7.75. Time-Optimal Torpedo Control Problem. Case D: SE Method, 12 Basis Functions, 60 Sample Points, Free Final Endpoint, x-Trajectory.

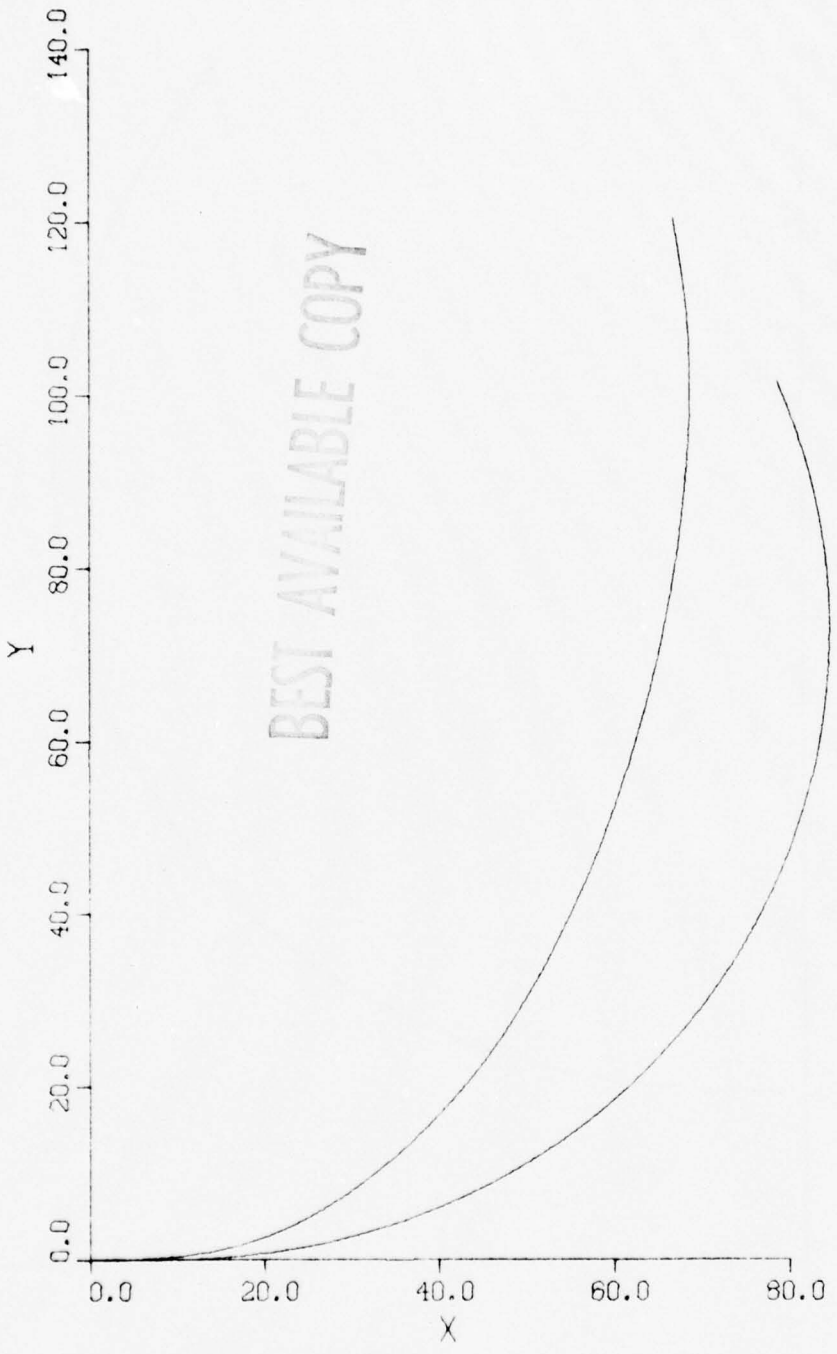





Figure 7.76. Time-Optimal Torpedo Control Problem. Case D: ME Method, 12 Basis Functions, 50 Sample Points, Free Final Endpoint.

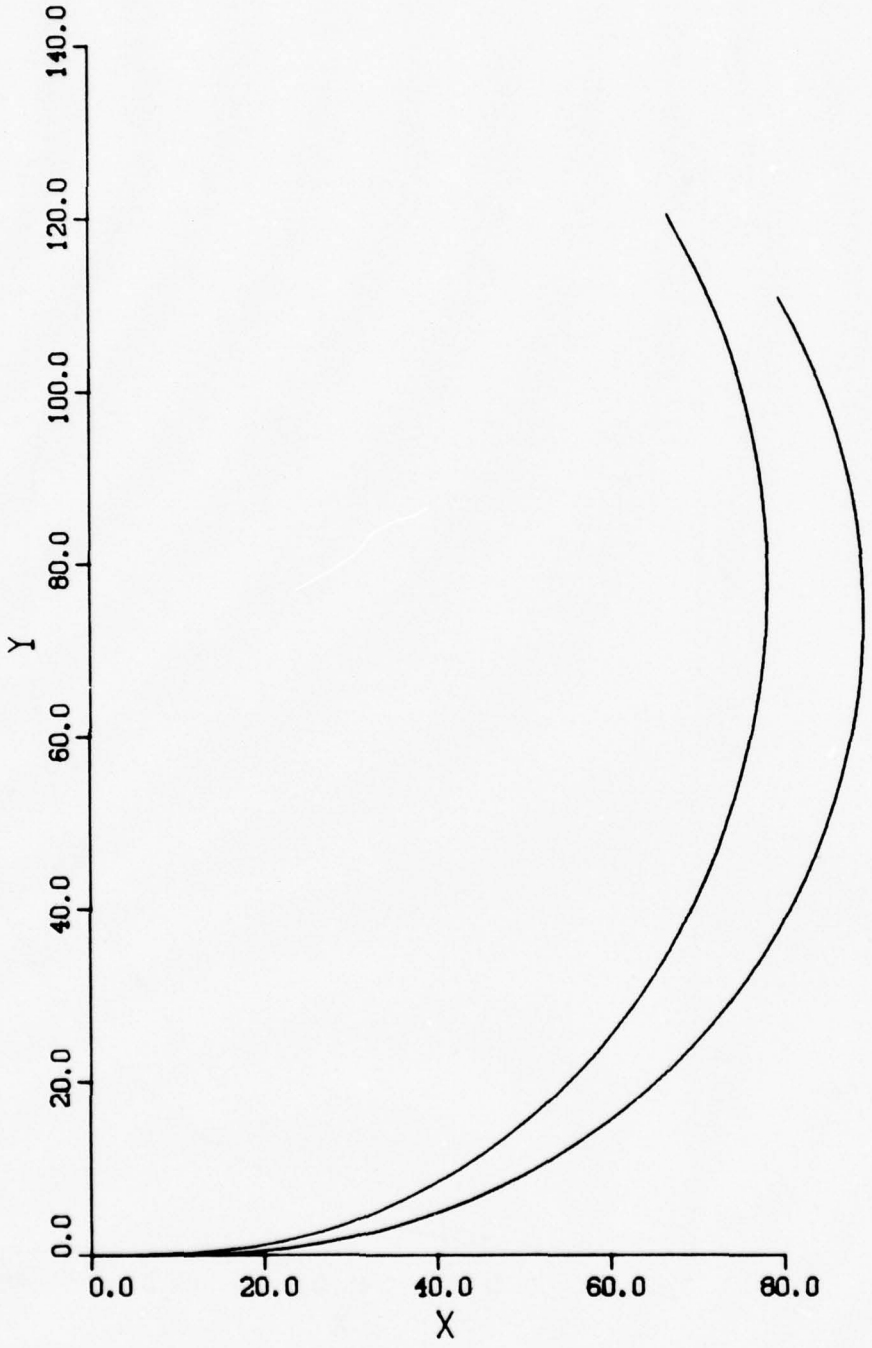



Figure 7.77. Time-Optimal Torpedo Control Problem. Case D: LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint, Short Initial Final Time.

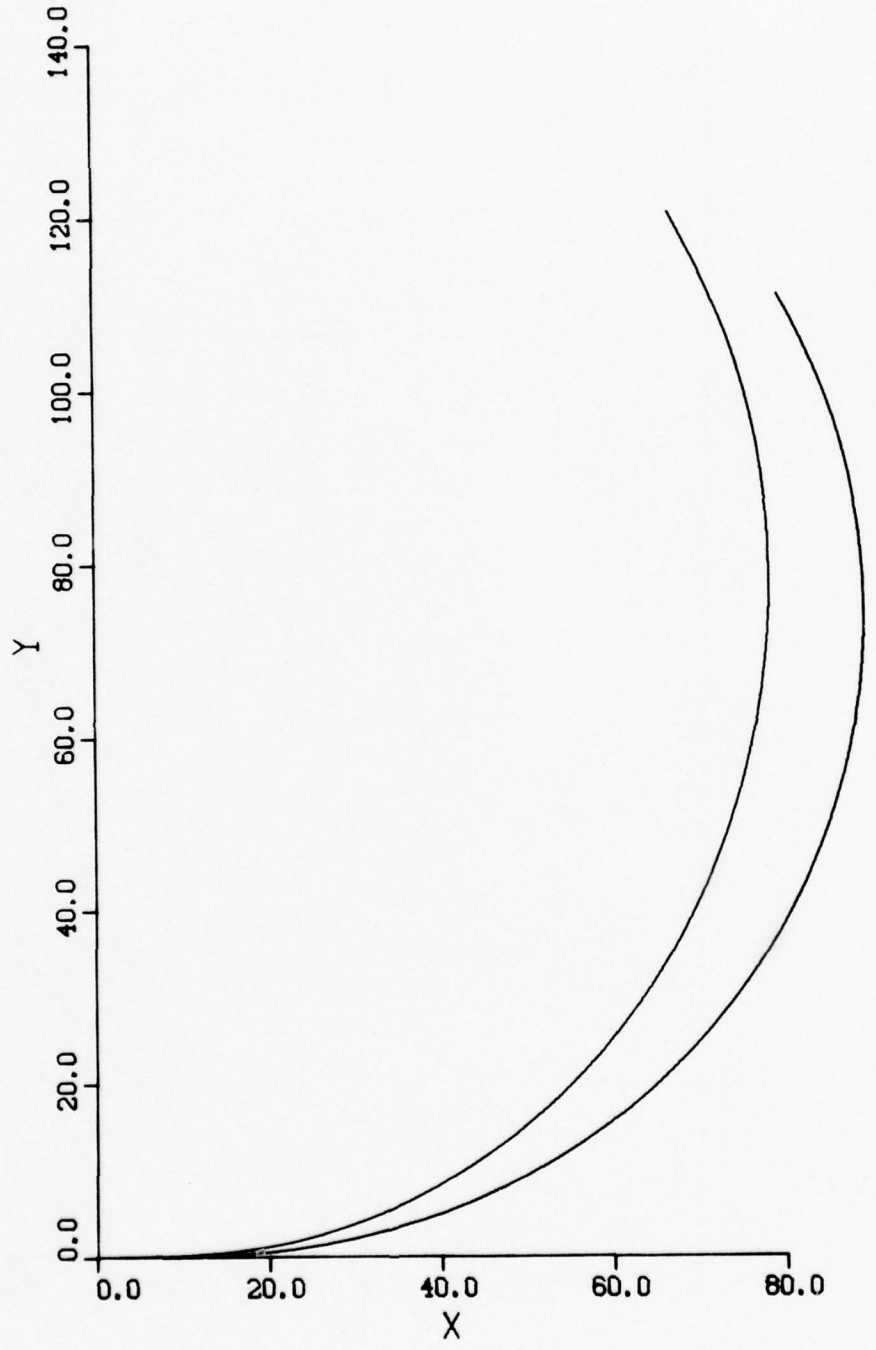



Figure 7.78. Time-Optimal Torpedo Control Problem. Case D: LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint, Exact Initial Final Time.

TABLE 7.11: LSE Method Applied to Long Ranges

| Case | E,Short | E | 51 | 52 |
|---|---|---|---|---|
| Initial Range | | | 500 | 500 |
| Initial Heading | | | 0.1 rad. | 0.2 rad. |
| Initial Time | 2.8 | 3.0 | 7.401 | 7.401 |
| LIN Plot | 7.79 | 7.81 | 7.83 | 7.85 |
| $x_1$ | -819.94 | -819.97 | -0.76884 | -0.77064 |
| $x_2$ | 74.966 | 74.976 | 0.02551 | 0.05102 |
| $x_6$ | 0.65123 | 0.65006 | -0.08795 | -0.17589 |
| $x_1^u$ | -831.74 | -819.43 | -1.63777 | -4.2546 |
| $x_2^u$ | 68.706 | 73.89 | 0.13395 | 0.30778 |
| $x_6^u$ | 0.63973 | 0.63938 | -0.0875 | -0.17488 |
| u | -0.132,10 | -0.122,10 | 0.009,10 | 0.018,10 |
| | -0.100,430 | -0.100,320 | 0.0015,1760 | 0.0102,1700 |
| | -0.0108,1610 | 0.0002,1770 | 0.0003,1780 | 0.0006,1780 |
| | 0.0001,1760 | 0.0001,1800 | -0.0004,1800 | -0.0001,1790 |
| | 0.0005,1780 | | | -0.00004,1800 |

TABLE 7.11: continued

| Case | 53 | 54 | 71 | 11 |
|---|---|---|---|---|
| Initial Range | 500 | 500 | 750 | 1000 |
| Initial Heading | 0.3 rad. | 0.4 rad. | 0.1 rad. | 0.1 rad. |
| Initial Time | 7.401 | 7.401 | 11.6005 | 14.802 |
| LIN Plot | 7.87 | 7.89 | 7.91 | 7.93 |
| $x_1$ | -0.77064 | -0.77064 | -0.77725 | -0.77601 |
| $x_2$ | 0.07654 | 0.10205 | 0.15744 | 0.39963 |
| $x_6$ | -0.26384 | -0.35178 | -0.40981 | -0.77509 |
| $x_1^u$ | -8.5835 | -14.576 | -11.243 | -44.791 |
| $x_2^u$ | 0.55954 | 0.92364 | 0.69889 | 5.6183 |
| $x_6^u$ | -0.2618 | -0.34811 | -0.40902 | -0.77121 |
| u | 0.0276,10 | 0.0368,10 | -0.0102,10 | -0.0213,10 |
| | 0.0172,1160 | 0.0117,1740 | -0.0002,390 | 0.0002,510 |
| | 0.0009,1780 | 0.0013,1780 | 0.0102,780 | 0.0573,1720 |
| | -0.0002,1796 | -0.0001,1800 | 0.037,1710 | -0.0006.1800 |
| | -0.0001,1800 | | -0.0004,1800 | |

TABLE 7.11: continued

| Case | E,Short | E | 51 | 52 |
|---|---|---|---|---|
| Initial Range | | | 500 | 500 |
| Initial Heading | | | 0.1 rad. | 0.2 rad. |
| Initial Time | 2.8 | 3.0 | 7.401 | 7.401 |
| EPS | | | | |
| Plot | 7.80 | 7.82 | 7.84 | 7.86 |
| $t_f$ | 3.016 | 3.0103 | 7.4262 | 7.4683 |
| De/ε | 297.72 | 251.57 | 12.168 | 24.541 |
| $x_1^{\varepsilon}$ | 294.70 | 248.56 | 4.7417 | 17.073 |
| $x_2^{\varepsilon}$ | -819.39 | -819.39 | -0.38-05 | -0.38-05 |
| $x_6^{\varepsilon}$ | 75.36 | 75.36 | 0.47-05 | 0.96-06 |
| $x_1^{u}$ | 0.63419 | 0.63735 | -0.08483 | -0.16965 |
| $x_2^{u}$ | -822.00 | -818.97 | 0.02216 | 0.13157 |
| $x_6^{u}$ | 73.167 | 73.927 | 1.24153 | 1.9682 |
| u | 0.69210 | 0.64519 | -0.08586 | -0.17355 |
| | -0.087,1 | -0.08,1 | 0.004,1 | 0.0083,1 |
| | -0.1342,5 | -0.1236,50 | 0.0098,5 | 0.0196,5 |
| | -0.103,14 | -0.1056,8 | 0.006,47 | 0.012,47 |
| | -0.011,47 | -0.011,47 | 0.0017,49 | 0.0035,49 |
| | 0.0302,51 | 0.026,51 | -0.004,51 | -0.0081,51 |

TABLE 7.11: continued

| Case | 53 | 54 | 71 | 11 |
|---|---|---|---|---|
| Initial Range | 500 | 500 | 750 | 1000 |
| Initial Heading | 0.3 rad. | 0.4 rad. | 0.1 rad. | 0.1 rad. |
| Initial Time | 7.401 | 7.401 | 11.6005 | 14.802 |
| EPS | | | | |
| Plot | 7.88 | 7.90 | 7.92 | 7.94 |
| $t_f$ | 7.5391 | 7.6394 | 11.260 | 15.501 |
| J | 45.237 | 74.408 | 41.494 | 124.24 |
| De/ε | 37.698 | 66.769 | 30.235 | 108.74 |
| $x_1^{\varepsilon}$ | -0.38-05 | -0.75-05 | -0.72-05 | -0.54-05 |
| $x_2^{\varepsilon}$ | 0.14-05 | 0.19-05 | 0.27-05 | 0.65-05 |
| $x_6^{\varepsilon}$ | -0.25439 | -0.33879 | -0.33187 | -0.62781 |
| $x_1^{u}$ | 0.22866 | 0.06973 | -0.72098 | -2.0182 |
| $x_2^{u}$ | 1.6457 | -0.24700 | -2.4249 | -7.8593 |
| $x_6^{u}$ | -0.26495 | -0.36198 | -0.40451 | -0.78966 |
| u | 0.0124,1 | 0.0164,1 | -0.0033,1 | -0.006,1 |
| | 0.0294,5 | 0.0392,5 | 0.0004,12 | 0.0009,15 |
| | 0.013,48 | 0.0174,48 | 0.0106,22 | 0.0599,46 |
| | 0.005,49 | -0.007,49 | 0.0405,46 | 0.031,49 |
| | -0.012,51 | -0.016,51 | -0.0169,51 | -0.195,51 |

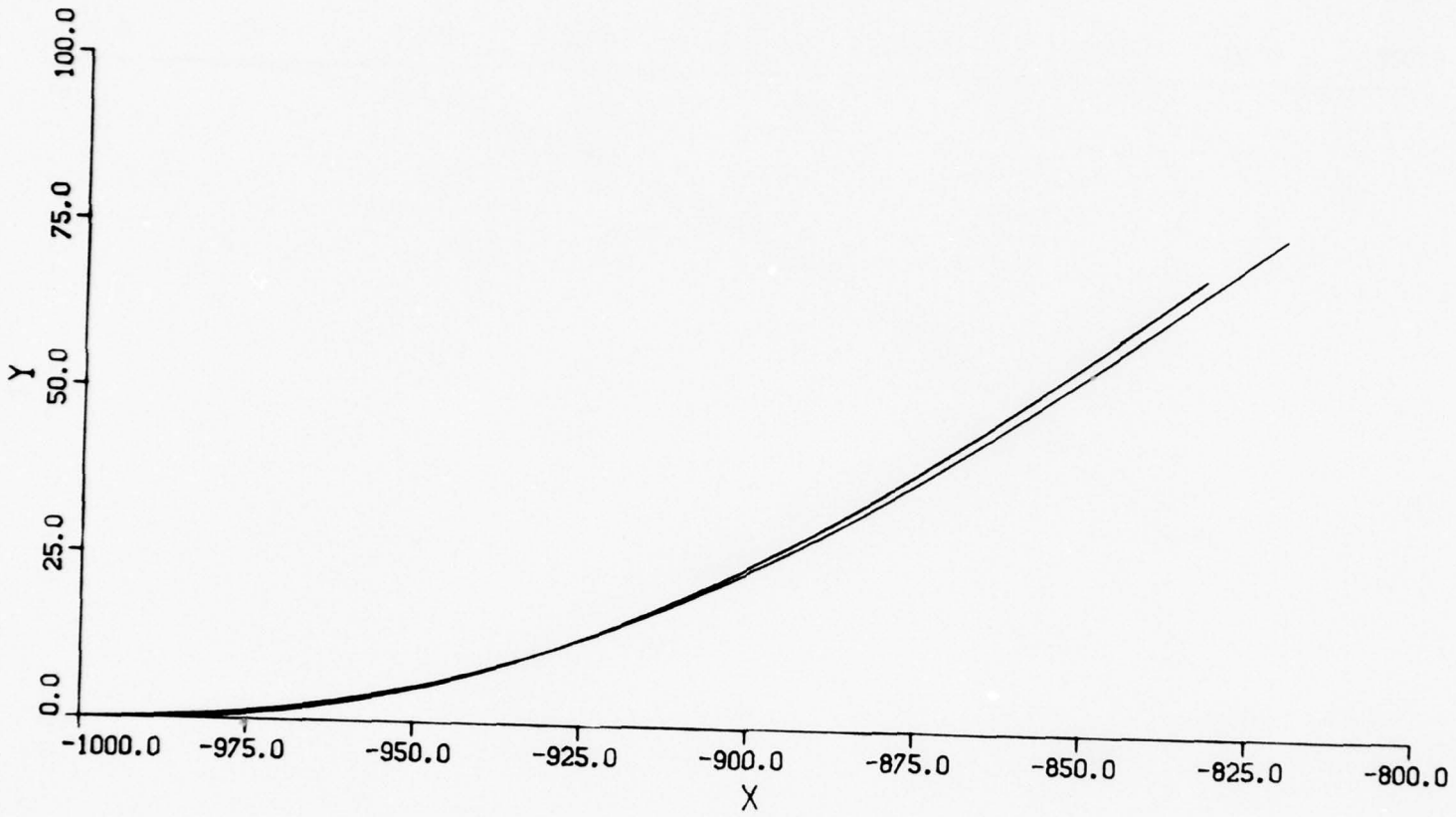



Figure 7.79. Fixed-Time Torpedo Control Problem. Case E: LIN Method, Short Initial Final Time, $\lambda = 60{,}000$.

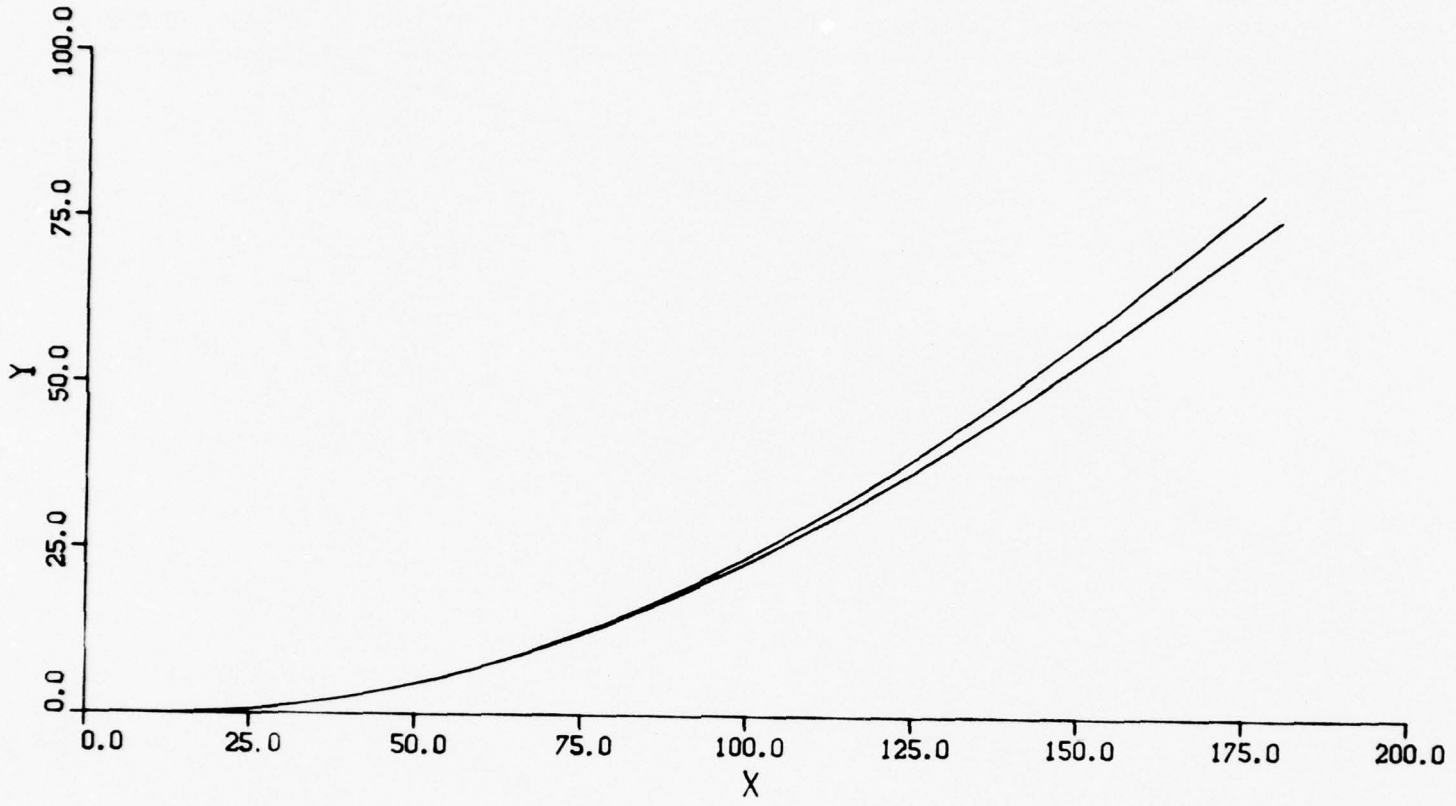



Figure 7.80. Time-Optimal Torpedo Control Problem. Case E: LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint, Short Initial Final Time.

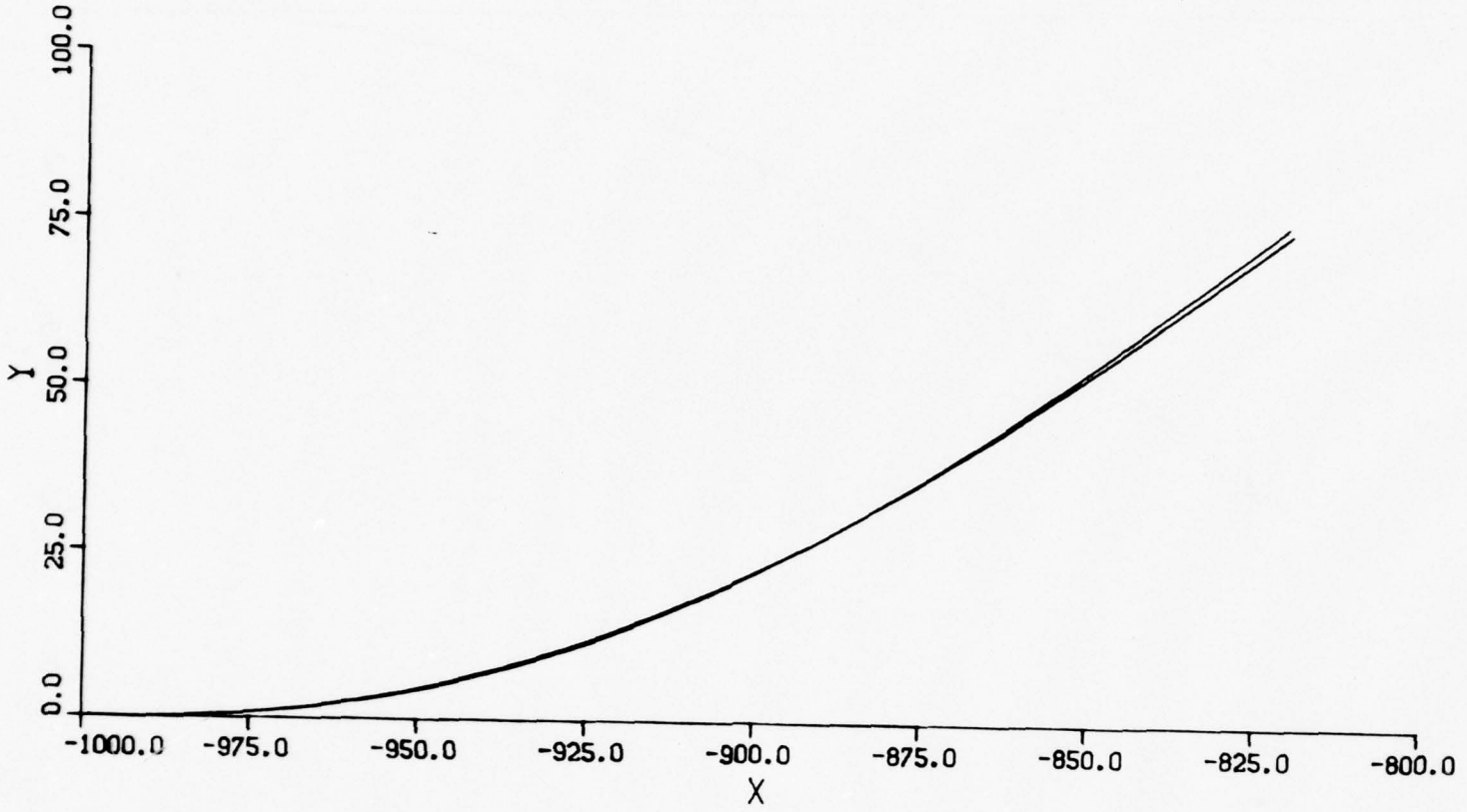



Figure 7.81. Fixed-Time Torpedo Control Problem. Case E: LIN Method, $\lambda = 60{,}000$.

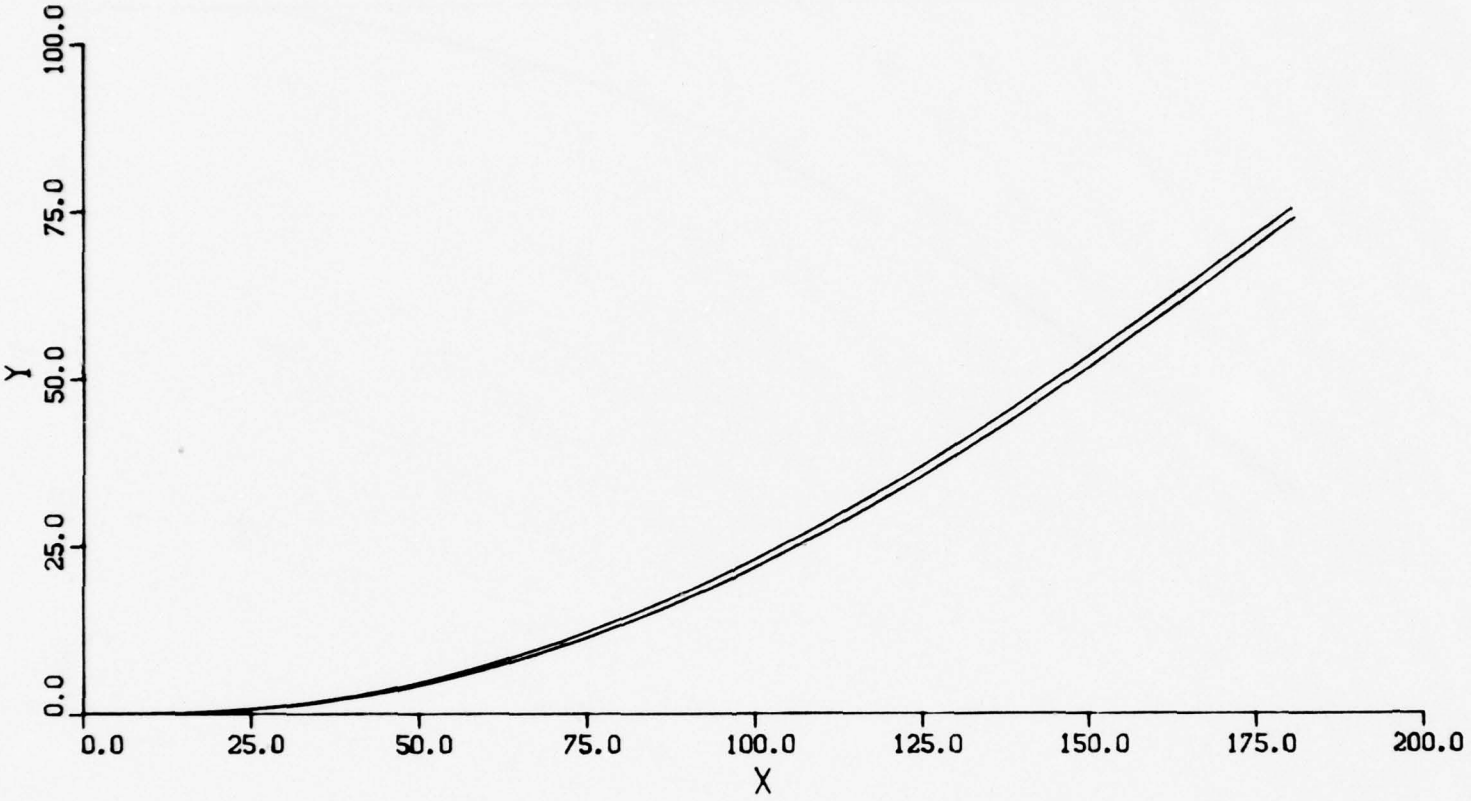



Figure 7.82. Time-Optimal Torpedo Control Problem. Case E: LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint, Exact Initial Final Time.

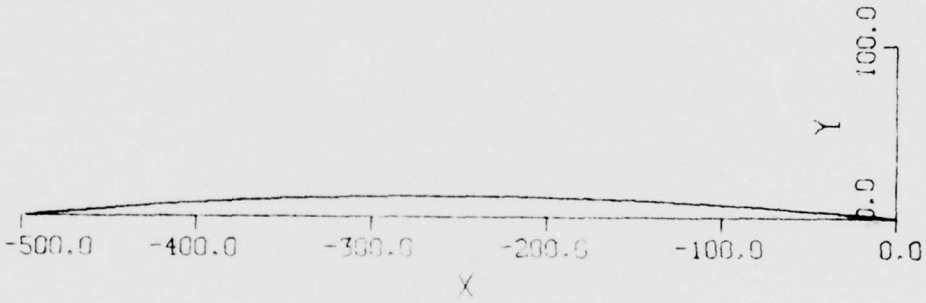



Figure 7.83. Fixed-Time Torpedo Control Problem. Initial Range, 500 ft, Initial Heading, 0.1 rad, LIN Method, $\lambda = 60{,}000$.

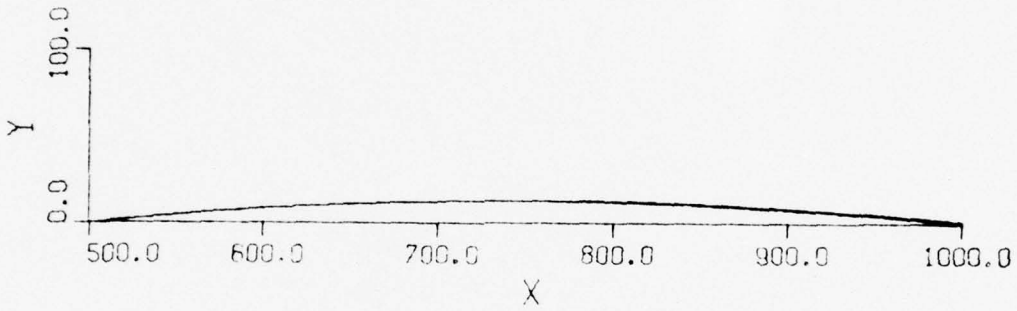



Figure 7.84. Time-Optimal Torpedo Control Problem. Initial Range, 50 ft, Initial Heading, 0.1 rad, LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint.

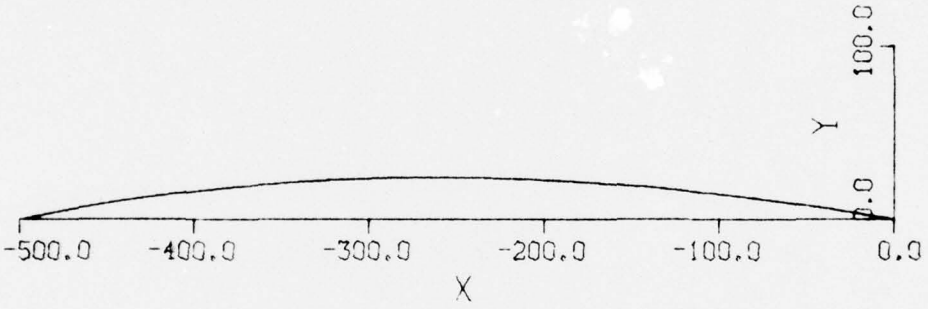



Figure 7.85. Fixed-Time Torpedo Control Problem. Initial Range, 50 ft, Initial Heading, 0.2 rad, LIN Method, $\lambda = 60{,}000$.

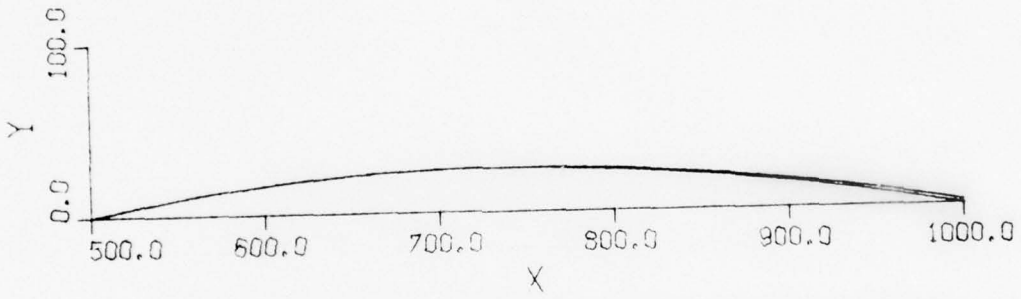



Figure 7.86. Time-Optimal Torpedo Control Problem. Initial Range, 500 ft, Initial Heading, 0.2 rad, LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint.

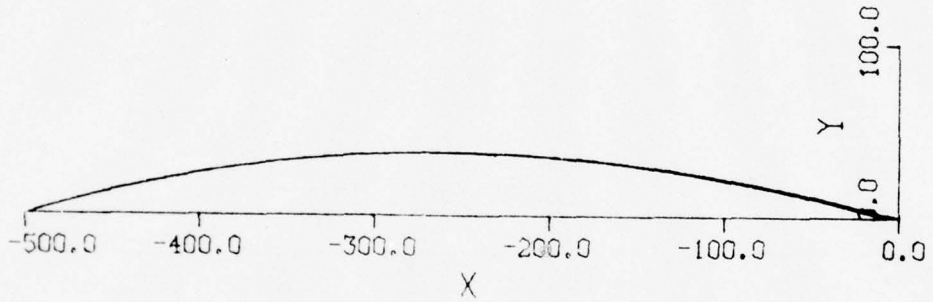



Figure 7.87. Fixed-Time Torpedo Control Problem. Initial Range, 500 ft, Initial Heading, 0.3 rad, LIN Method, $\lambda = 60{,}000$.

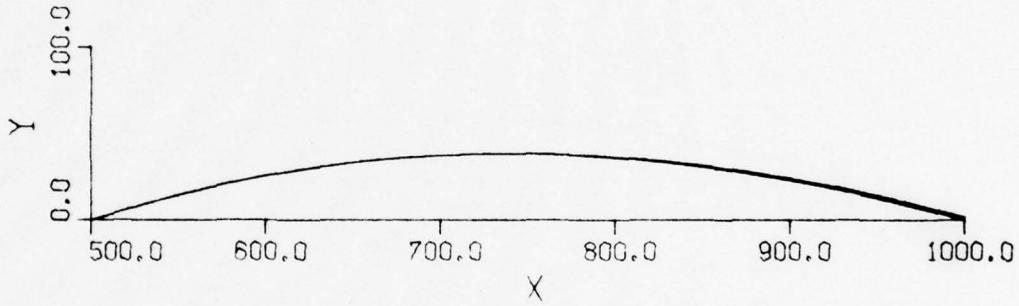



Figure 7.88. Time-Optimal Torpedo Control Problem. Initial Range, 500 ft, Initial Heading, 0.3 rad, LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint.

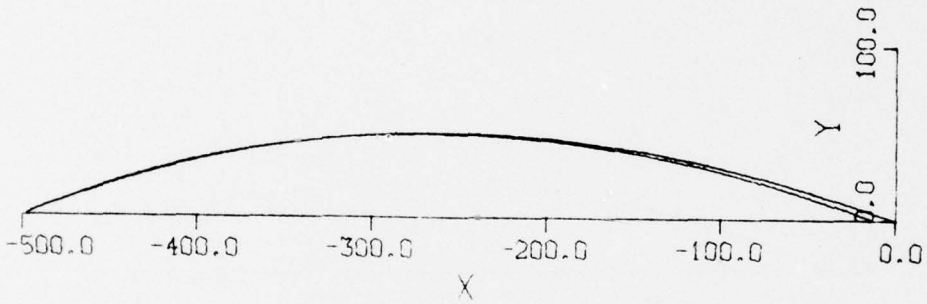



Figure 7.89. Fixed-Time Torpedo Control Problem. Initial Range, 500 ft, Initial Heading, 0.4 rad, LIN Method, $\lambda = 60{,}000$.

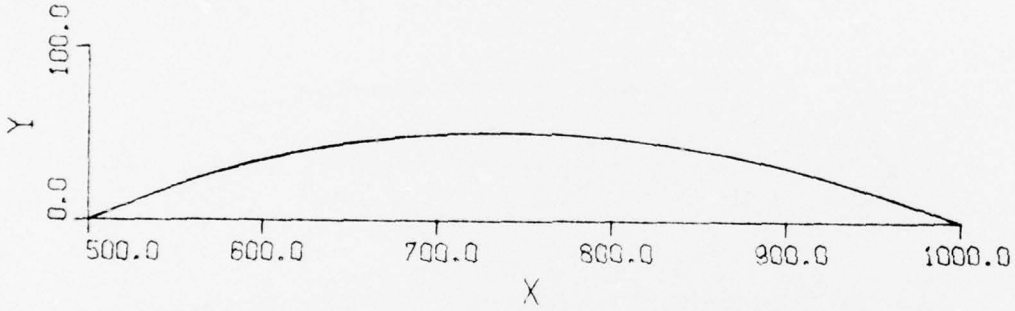



Figure 7.90. Time-Optimal Torpedo Control Problem. Initial Range, 500 ft, Initial Heading, 0.4 rad, LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint.

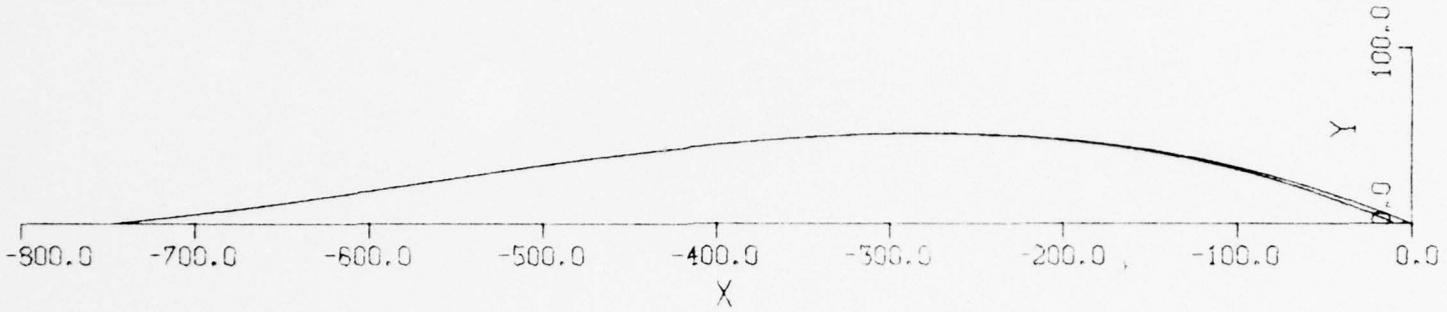



Figure 7.91. Fixed-Time Torpedo Control Problem. Initial Range, 750 ft, Initial Heading, 0.1 rad, LIN Method, $\lambda = 60{,}000$

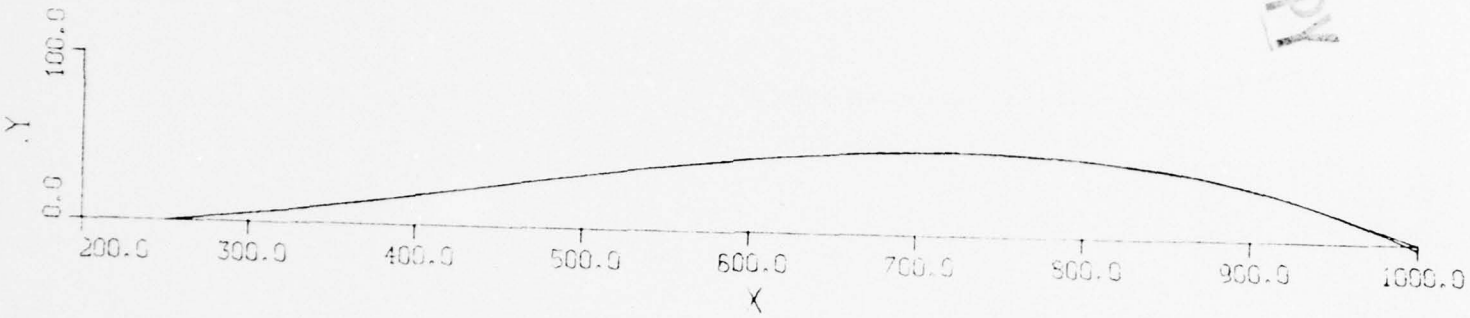



Figure 7.92. Time-Optimal Torpedo Control Problem. Initial Range, 750 ft, Initial Heading, 0.1 rad, LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint.

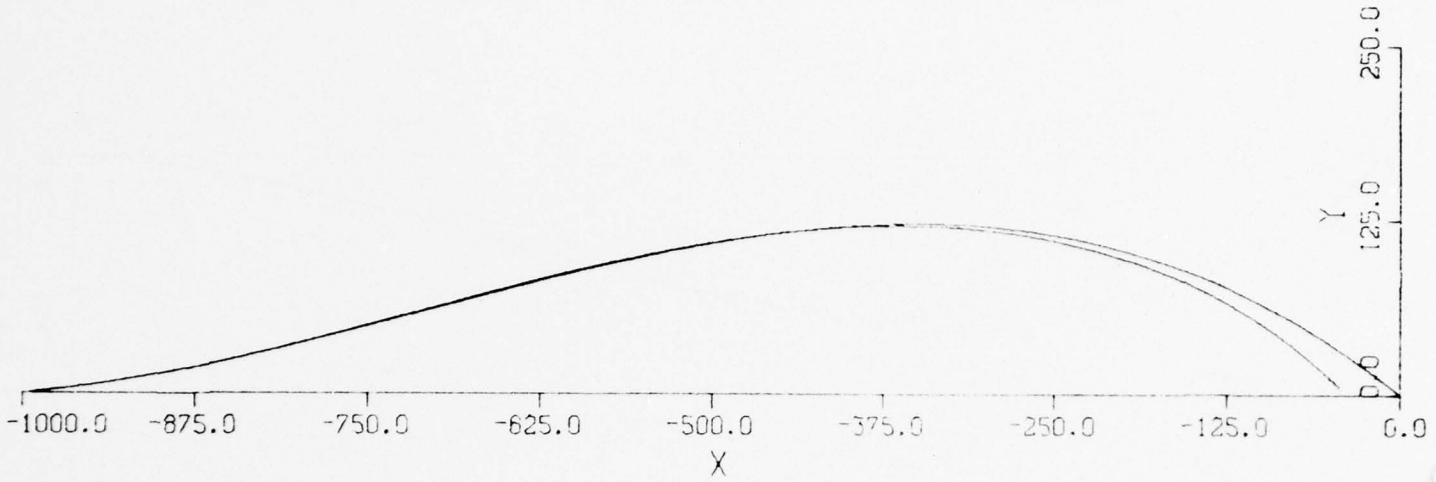



Figure 7.93. Fixed-Time Torpedo Control Problem. Initial Range, 1000 ft, Initial Heading, 0.1 rad, LIN Method, $\lambda = 60{,}000$.

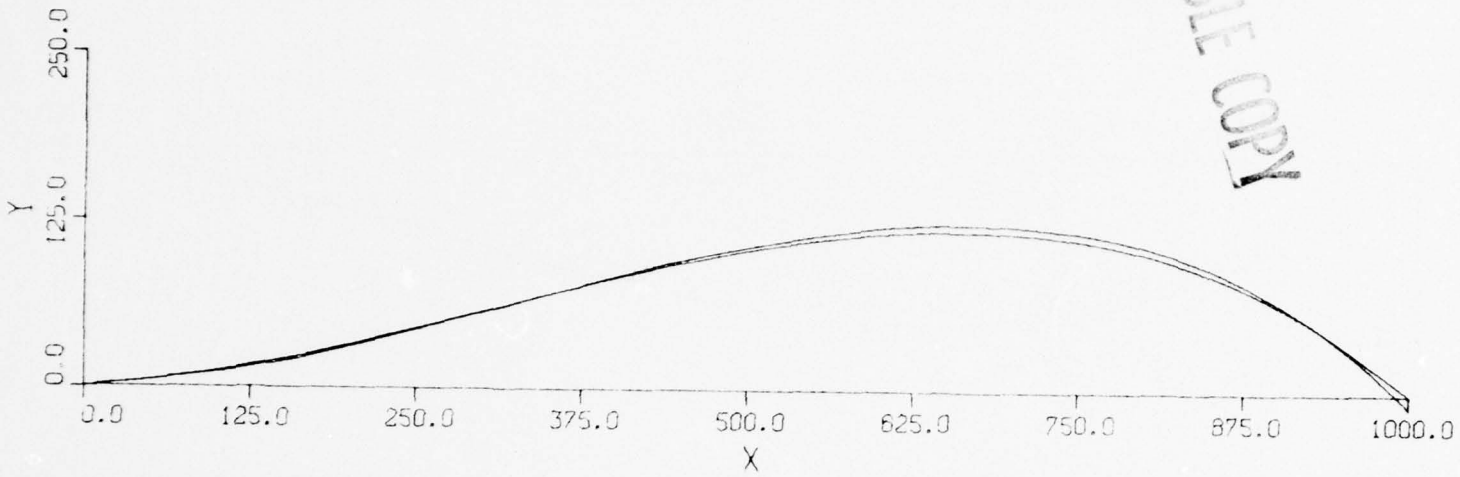



Figure 7.94. Time-Optimal Torpedo Control Problem. Initial Range, 1000 ft, Initial Heading, 0.1 rad, LSE Method, 12 Basis Functions, 51 Sample Points, Free Final Endpoint.

## Chapter Eight: Conclusions

In this dissertation, several computational techniques are applied to the torpedo control problem. The SE method gives a good x-trajectory but the u-trajectory is unsatisfactory in many cases. The ME method and the CG method yield good results for cases A,B and D but not for cases C and E. The LIN method yields poor results for cases A, B and D and fair results for cases C and E. The LSE method gives good results for all of the cases. There is close agreement between the x-trajectory and the u-trajectory and the final endpoint is always reached. Therefore, the feasibility of using the LSE method to solve the torpedo control problem has been shown. It is possible that the LIN method is not the only method that could be used to initialize the trajectory of the state vector in the Epsilon Technique to obtain adequate results.

The computer limitations of limited computer storage that was discussed at the end of chapter three severely limit the optimality of the results of the SE method and the LSE method to the time-optimal torpedo control problem for the cases where the trajectory is of a time duration greater than two seconds. It is our conclusion that the results would be closer to optimal if a sufficient number of sample points and basis functions could be chosen.

REFERENCES


1. Balakrishnan, A. V. "On a New Computing Technique in Optimal Control", J. SIAM on Control, 6:149-173, May 1968.
2. Balakrishnan, A. V. "The Epsilon Technique - A Constructive Approach to Optimal Control", Control Theory and the Calculus of Variations, A. V. Balakrishnan (ed), Academic Press, New York 1969.
3. Balakrishnan, A. V. "A Computational Approach to the Maximum Principle", System Science Department, School of Engineering and Applied Science, University of California, Los Angeles, UCLA - ENG - 7077, August 1970.
4. Balakrishnan, A. V. "On a New Computing Technique in Optimal Control and its Application to Minimal - Time Flight Profile Optimization", Journal of Optimization Theory and Applications, 4(1): 1-21, July 1969.
5. Taylor, L. W. et. al. "Experience Using Balakrishnan's Epsilon Technique to Compute Optimum Flight Profiles", Journal of Aircraft 7(2): 182-187, March 1970.
6. Taylor, J. M. and C. T. Constantinides "Optimization: Application of the Epsilon Method", IEEE Transactions on Automatic Control AC 17:128-131, February 1972.
7. Taylor, J. M. and C. T. Constantinides "A Computational Aspect of the Epsilon Method", IEEE Transactions on Automatic Control AC 17:738, October 1972.
8. Mikami, E. Y. "A Fortran Program for Trajectory Optimiation Using the Epsilon Method", Naval Weapons Center, China Lake, California NWC TP 5369, July 1972.
9. Mikami, E. Y. "A Fortran Optimization Program Using the Epsilon Method, Naval Weapons Center, China Lake, California,

NWC TP 5573, December 1973.

10. Hewett, M. C. and D. E. Kirk "Trajectory Optimization Using a Second-Order Epsilon Method", Journal of Aircraft 13(3): 174-179, March 1976.

11. Hestenes, M. R. and E. Stiefel "Methods of Conjugate Gradients for Solving Linear Systems", Journal of Research of the National Bureau of Standards 49:409-436, 1952.

12. Fletcher, R. and C. M. Reeves "Function Minimization by Conjugate Gradients", British Computer Journal 7:149-154, July 1964.

13. Lasdon, L. S. et. al. "The Conjugate Gradient Method for Optimal Control Problems", IEEE Transactions on Automatic Control AC 12(2): 132-138, April 1967.

14. Pagurek, B. and C. M. Woodside "The Conjugate Gradient Method for Optimal Control Problems With Bounded Control Variables", Automatica 4: 337-349 November 1968.

15. Lee, E. B. and L. Markus Foundations of Optimal Control, John Wiley and Sons, New York 1967.

16. Lopes, L. A. "Motion Equations for Torpedoes", Naval Ordnance Test Station, China Lake, California, NAVORD Report 2090, February 1954.

17. Brumback, R. P. "Nonlinear Dynamic Equations for a Torpedo in the Cavitating and Fully Wetted Regimes", Naval Ordnance Test Station, China Lake, California, Memorandum No. P8040-35, January 1965.

18. Luenberger, D. G. Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, Massachusetts 1972.

19. Taylor, L. W. and N. W. Rees "Application of the Epsilon Technique to a Realistic Optimal Pursuit Evasion Problem", Hawaii International

Conference on System Sciences Oahu, Hawaii 1971.

## Appendix A: Computer Programs

In this appendix a brief description of the different main programs and subroutines is given followed by a listing of the computer programs. In most cases the names of the variables that were used in the text are used in the computer programs.

MAIN

This is the main program used for applying the Epsilon Technique. The order of the program is to define the initial conditions. Then, the epsilon cost functional, its gradient and Hessian is computed. Then, system (3.13) is set up and solved to determine the new vector $\underline{A}$. Finally, the epsilon cost functional is again evaluated to determine if the method has converged. If not, the iteration is started again.

Subroutine FCT

This subroutine is used by MAIN to calculate the state vector, its derivative with respect to time, and the derivatives with respect to $\underline{A}$ using the Rayleigh-Ritz expansion.

Subroutine DFCT

This subroutine is used by MAIN to calculate $\underline{f}(t,\underline{x},u)$ and the partial derivatives of $\underline{f}(t,\underline{x},u)$ with respect to $\underline{x}$. The optimal value of u is calculated in this subroutine.

Subroutine COST

This subroutine is used by MAIN to calculate the cost functional and its derivative with respect to the time step.

Subroutine RDLU

This subroutine is used to calculate the solution of the equation $C\underline{x} = \underline{y}$ by a LU decomposition and back substitution using double

precision arithmetic. This subroutine can also be used to compute the inverse of C. This subroutine is used by all of the different methods.

CONGRA

This is the main program used in the conjugate gradient method. Steps 1 - 6 given in chapter 4 are implemented.

Subroutine FCT

This subroutine is used by CONGRA to calculate $\underline{f}(t,\underline{x},u)$ and its derivatives with respect to $\underline{x}$ and u.

Subroutine COFCT

This subroutine is used by CONGRA to calculate all of the variables given in Step 3 except for $I_{vs}$ and $I_{ss}$.

MNLNRL

This is the main program used to apply the method of linearization about a known trajectory as given in chapter 5.

Subroutine COEFC

This subroutine is used by MNLNRL to calculate the constants used in the equations of motion of a torpedo.

Subroutine FC

This subroutine is used by MNLNRL to calculate $\underline{f}(t,\underline{x},u)$.

Subroutine FCEV

This subroutine is used by MNLNRL to calculate the Riccati equation for the method of linearization about a known trajectory.

MNLNEP

This is MAIN modified to call MNLNRL first and to use the output of MNLNRL to initialize the state trajectory in MAIN.

Subroutine CALLIN

This is the main program MNLNRL modified to be a subroutine of MNLNEP.

```
                    MAIN

      COMPILER(XM=1)
      DOUBLE PRECISION HGRWD,HGRWI
      DOUBLE PRECISION D,E,E2
      DOUBLE PRECISION HGRW,HDMC,HDMD
      PARAMETER NOP=13
      PARAMETER N=6,M=12,NP=60,ID1=N*NP+1,ID2=N*M+5
      DIMENSION E(ID2),E2(ID2)
      DIMENSION XPLT(NP),YPLT(NP),XPTG(NP),YPTG(NP)
      COMMON X(N),XD(N),AO(N),AOO(N),DXDA(N,M),DXDDA(N,M),F(N),DXDAO(N),
     $DWDDT(NP,N),A(N,M),DEL(NP),DWIDA1(ID1,ID2),
     $DXDDAO(N),DXDT(N),DXODT(N),DFDX(N,N),W(NP,N),W1(ID1),A1(ID2),
     $DWDA(NP,N,N,M),DWDAO(NP,N,8),HGRW(ID2,ID2),HDMC(ID2),HDMD(ID2),
     $                HGRWI(ID2,ID2),D(ID2)
     $,DFDT(N),XFP(N)
     $,AOF(N)
     $,A10(ID2)
     $,A1S(ID2,30),A1P(ID2,28)
      INA=1
      XFP(1)=0.
      XFP(2)=0.
   21 READ 22,COEF,XTD,YTD,ALPHA
   22 FORMAT(8F10.4)
      PRINT 23,COEF,XTD,YTD,ALPHA
   23 FORMAT(8H COEF = ,F10.4,6HXTD = ,F10.4,2X,6HYTD = ,F10.4,2X,F10.7)
      ZETEST=SQRT(3.)/2.
      COEFO=COEF
      DT=3./(NP-1.)
      DT=12./(NP-1.)
      AO(1)=-1000.
      AO(2)=0.
      AO(3)=67.56
      AO(4)=0.
      AO(5)=0.
      AO(6)=0.
      AOO(1)=1000.
      AOO(2)=0.
      AOO(3)=0.
      AOO(4)=0.
      AOO(5)=0.
      AOO(6)=0.
      DIAG=.1
      EPS=.01
      EPS=.001
      NPS=NP-9
      ILOP=0
   40 ILOP=ILOP+1
      ILOOP=0
      DIAG=DIAG/10.
      PRINT 41,EPS,DIAG
```

```
 41 FORMAT(7H EPS = ,E20.15,5X,6HDIA = ,E20.15)
170 FORMAT(4F20.10)
    PRINT 103,((A(I,J),J=1,M),I=1,N)
 20 SQDE=SQRT(DT/EPS)
    XTARG=1000.
    YTARG=0.
    DO 1 L=1,NP
    IF(L.LE.NPS)SQDE=SQRT(DT/(NPS*EPS))
    IF(L.GT.NPS)SQDE=SQRT(DT/EPS)
    CALL FCT(X,XD,A,AO,AOO,DXDA,DXDDA,DXDAO,DXDDAO,DXDT,DXDDT,L,N,M,NP
   $,DT,ALPHA,AOF)
    CALL DFCT(F,X,XD,DFDX,L,N,DT,DEL,NP,XTD,YTD)
    IF(ILOOP.NE.29)GO TO 90
    XPLT(L)=X(1)+XTARG
    YPLT(L)=X(2)+YTARG
    XPTG(L)=XTARG
    YPTG(L)=YTARG
 90 CONTINUE
    DO 2 J=1,N
    DO 2 K=1,N
    DO 2 I=1,M
    DWDA(L,J,K,I)=-DFDX(J,K)*DXDA(K,I)*SQDE
  2 IF(J.EQ.K)DWDA(L,J,K,I)=DWDA(L,J,K,I)+DXDDA(J,I)*SQDE
    DO 65 J=1,N
    DO 66 K=1,4
    DWDAO(L,J,K)=-DFDX(J,K+2)*DXDAO(K+2)*SQDE
 66 IF(J.EQ.K+2)DWDAO(L,J,K)=DWDAO(L,J,K)+SQDE*DXDDAO(J)
 65 CONTINUE
    DO 87 J=1,N
    DFDT(J)=0.
    DO 87 I=1,N
 87 DFDT(J)=DFDT(J)+DFDX(J,I)*DXDT(I)
    DO 3 J=1,N
    W(L,J)=(XD(J)-F(J))*SQDE
  3 DWDDT(L,J)=W(L,J)/(2.*DT)+(DXDDT(J)-DFDT(J))*SQDE
    IF(L.LE.NPS)TIME=(L-1.)*DT/NPS
    IF(L.GT.NPS)TIME=(L-NPS)*DT
    PRINT 105,TIME,(X(I),I=1,6),(XD(I),I=1,6),(F(I),I=1,6)
105 FORMAT(5H T = ,F10.6,4HX = ,6E15.5,/,6H XD = ,6E20.5,/,5H F = ,
   $ 6E20.5)
    XTARG=XTARG+XTD*DT
    YTARG=YTARG+YTD*DT
  1 CONTINUE
    PRINT 50,(I,DEL(I),I=1,NP)
 50 FORMAT(5H I = ,I3,2X,6HDEL = ,E20.15)
    CALL        COST(DT,NP,G,DGDT,X,XFP,N)
    DO 4 J=1,N
    DO 4 I=1,M
  4 A1(J,M+I)=A(J,I)
    DO 67 I=1,4
    A1(N,M+I)=AOO(I+2)
 67 CONTINUE
```

AD-A042 206 NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CALIF F/G 19/8
COMPUTATIONAL TECHNIQUES FOR TIME-OPTIMAL CONTROL OF TORPEDO DY--ETC(U)
MAY 77 W S LAPP

UNCLASSIFIED NOSC-TR-124 NL

3 OF 3
ADA
042206

END

DATE
FILMED
8-77

```
      A1(ID2)=DT
      COEF=COEF0
      DO 149 I=1,ID2
  149 A10(I)=A1(I)
      DO 5 J=1,N
      DO 5 L=1,NP
    5 W1(J*NP-NP+L)=W(L,J)
      W1(ID1)=G
      DO 6 J=1,N
      DO 6 L=1,NP
      DO 7 K=1,N
      DO 7 I=1,M
    7 DW1DA1(J*NP-NP+L,K*M-M+I)=DWDA(L,J,K,I)
      DO 68 K=1,4
   68 DW1DA1(J*NP-NP+L,N*M+K)=DWDA0(L,J,K)
    6 DW1DA1(J*NP-NP+L,ID2  )=DWDDT(L,J)
      DW1DA1(ID1,ID2)=DGDT
      DO 9 I=1,ID2
      DO 8 J=1,ID2
      HGRW(I,J)=0.
      DO 8 K=1,ID1
    8 HGRW(I,J)=HGRW(I,J)+DW1DA1(K,I)*DW1DA1(K,J)
      HDMC(I)=0.
      DO 9 K=1,ID1
    9 HDMC(I)=HDMC(I)+DW1DA1(K,I)*W1(K)
      CV=0.
      DO 12 I=1,ID1
   12 CV=CV+W1(I)**2
      CV0=CV
      PRINT 102,ILOOP,CV0
  102 FORMAT(5H I = ,I5,2X,E20.15)
      DO 34 I=1,ID2
   34 HGRW(I,I)=HGRW(I,I)+DIAG
      DO 402 I=1,ID2
      DO 402 J=1,ID2
  402 HGRW1(I,J)=HGRW(I,J)
      CALL TRED1(ID2,ID2,HGRW1,D,E,E2)
      CALL IMTQL1(ID2,D,E,IERR)
      PRINT 400,IERR,(D(I),I=1,ID2)
  400 FORMAT(7HIER = ,I5,4HD = ,100(T18,E20.15,/))
      CALL RDLU(HGRW,HGRW1,HDMC,HDMD,ID2 ,0,0)
      GO TO 76
   75 DO 74 I=1,ID2
   74 HDMD(I)=HDMC(I)
   76 CONTINUE
      DO 10 I=1,ID2
      A1(I)=A1(I)+COEF*HDMD(I)
   10 CONTINUE
      DO 11 J=1,N
      DO 11 I=1,M
   11 A(J,I)=A1(J*M-M+I)
      PRINT 103,((A(I,J),J=1,M),I=1,N)
```

```
103 FORMAT(6(1H ,6E20.15,/,T3,6E20.15,/),//)
    DO 69 I=1,4
    AOO(I+2)=A1(N+M+I)
 69 CONTINUE
    DT=A1(ID2)
    IF(DT.LT.0.)DT=.1
    IF(DT.GT..5)DT=.5
    CV=0.
    SQDE=SQRT(DT/EPS)
    DO 151 L=1,NP
    CALL FCT(X,XD,A,AO,AOO,DXDA,DXDDA,DXDAO,DXDDAO,DXDT,DXDDT,L,N,M,NP
   $,DT,ALPHA,AOF)
    CALL DFCT(F,X,XD,DFDX,L,N,DT,DEL,NP,XTD,YTD)
    DO 151 J=1,N
    W(L,J)=(XD(J)-F(J))*SQDE
    CV=CV+W(L,J)**2
151 CONTINUE
    CALL        COST(DT,NP,G,DGDT,X,XFP,N)
    CV=CV+G*G
    IF(CV.LT.CV0)GO TO 153
    COEF=COEF/4.
    DO 158 I=1,ID2
158 A1(I)=A10(I)
    GO TO 76
153 CONTINUE
    PRINT 104,(AO(I),I=1,N)
    PRINT 104,(AOO(I),I=1,N)
104 FORMAT(1H ,3E20.15)
    DE=CV-G*G
    TF=(NP-1)*DT
    ILOOP=ILOOP+1
    PRINT 100,ILOOP,TF,CV,DE
100 FORMAT(5H I = ,I4,2X,5HTF = ,E15.5,2X,5HCV = ,E15.5,2X,5HDE = ,
   $E15.5)
    DO 160 I=1,ID2
160 A1S(I,ILOOP)=A1(I)
    IF(ILOOP.LT.30 )GO TO 20
    DO 162 I=1,28
    DO 161 J=1,ID2
161 A1P(J,I)=A1S(J,I)-((A1S(J,I+1)-A1S(J,I))**2)/(A1S(J,I+2)-
   $          2.*A1S(J,I+1)+A1S(J,I))
    DO 163 K=1,N
    DO 163 L=1,M
163 A(K,L)=A1P(K*M-M+L,I)
162 PRINT 103,((A(K,L),L=1,M),K=1,N)
    PRINT 165,(A1P(ID2,I),I=1,28)
165 FORMAT(7H A1P = ,(T8,E15.10))
    TPP=.01
    T=0.
    DT=TF/1377.
    DO 120 I=1,N
120 X(I)=AO(I)
```

```
      L=NP+1
      DO 121 I=1,1377
      IF(I.GT.ITEST)L=L+1
      IF(L.LE.NP+NPS)ITESTD=3
      IF(L.GT.NP+NPS)ITESTD=15
      IF(I.GT.ITEST)ITEST=ITEST+ITESTD
      CALL DFCT(F,X,XD,DFDX,L,N,DT,DEL,NP,XTD,YTD)
      DO 122 J=1,N
  122 X(J)=X(J)+DT*F(J)
      T=T+DT
      IF(T.LT.TPP)GO TO 121
      TPP=TPP+.01
      PRINT 123,T,(X(J),J=1,N)
  123 FORMAT(5H T = ,F10.3,6E15.10)
  121 CONTINUE
      CALL OPNPLT
      CALL XYEPLT
      CALL LINPLT(XPLT,YPLT,NP)
      CALL CURVE(XPTG,YPTG,NP,0)
      CALL ENDPL(1)
      GO TO 21
  230 CONTINUE
      STOP
      END
```

FCT

```
      SUBROUTINE FCT(X,XD,A,AD,ADD,DXDA,DXDDA,DXDAD,DXDDAD,DXDT,DXDDT,L,
     $N,M,NP,DT,ALPHA,AOF)
      DIMENSION X(N),XD(N),DXDA(N,M),DXDDA(N,M),DXDAD(N),DXDDAD(N),
     $DXDT(N),DXDDT(N),A(N,M),AD(N),ADD(N)
     $,AOF(N)
      DEFINE D(I)=SIN(I*PI*T/TF)
      DEFINE DD(I)=I*PI/TF*COS(I*PI*T/TF)
      PI=3.1415926536
      RL=L
      T=DT*(L-1.)
      TF=(NP-1.)*DT
      DO 1 J=1,N
      X(J)=0.
      XD(J)=0.
      DO 2 I=1,M
      DXDA(J,I)=D(I)
      DXDDA(J,I)=DD(I)
      X(J)=X(J)+A(J,I)*DXDA(J,I)
    2 XD(J)=XD(J)+A(J,I)*DXDDA(J,I)
      X(J)=AD(J)+ADD(J)*T/TF+X(J)
      XD(J)=ADD(J)/TF+XD(J)
      DXDAD(J)=T/TF
      DXDDAD(J)=1./TF
      DXDT(J)=0.
      DXDDT(J)=-XD(J)/DT
    1 CONTINUE
      RETURN
      END
```

DFCT

```
      SUBROUTINE DFCT(F,X,XD,DFDX,L,N,DT,DEL,NP,XTD,YTD)
      DIMENSION F(N),X(N),XD(N),DFDX(N,N),DEL(NP)
      COMMON/A/DFDDX(6,6,6),DUTEST
      RM=25.
      AA=.9
      RHO=2.
      RL=15.
      DMAX=15./57.29578
      XTA=-7.5
      RK=1.07/2.1
      DP=.15
      RF=RL/(1.128*SQRT(AA))
      RK1=.65*RF**(-1.49)
      RK2=.73*RF**.1139
      RML=800.*(1+RK1)/32.2
      RMT=800.*(1+RK2)/32.2
      RKP=.42*RF**.331
      RJZ=400.+340.*RKP
      RLTAP=2.5
      THRUST=(RHO/2.)*AA*DP*67.56*67.56
      H8=(RHO/2.)*AA*RLTAP
      B1=COS(X(6))
      B2=-SIN(X(6))
      B3=-B2
      B4=B1
      RHOH=RHO/2.
      H1=RHOH*AA*DP
      H2=RHOH*AA
      H3=RHOH*AA*RL**.3
      H6=-1.3*(RHO/2.)*AA*RL
      H7=RHOH*AA*RL*RL**(-.2)
      IOC=300
      IOC=600
      K=L
      IF(L.GT.NP)K=L-NP
      IF(L.GT.NP)GO TO 5
      IF(L.EQ.0)GO TO 2
      A=XD(4)-X(3)*(X(5)*(H3-RML)-H2*X(4)-    H8*(X(4)+XTA*X(5)))/RMT
      B=-    H8*RK*X(3)*X(3)/RMT
      C=XD(5)-(H6*X(3)*X(4)+H7*X(3)*X(5)-    H8*XTA*X(3)*(X(4)+XTA*X(5)))
     $  /RJZ
      D=-    H8*RK*XTA*X(3)*X(3)/RJZ
      DEL(K)=-(A+B*C+D)/(B*B+D*D)
      IF(ABS(DEL(K)).GT.DUTEST)DEL(K)=SIGN(DMAX,DEL(K))
    5 CONTINUE
      VT=X(4)+XTA*X(5)-RK*X(3)*DEL(K)
      GO TO 3
    2 IF(NP.LE.IOC)VT=X(4)+XTA*X(5)+RK*X(3)*DMAX
      IF(NP.GT.IOC)VT=X(4)+XTA*X(5)
```

```
3 CONTINUE
  F(1)=B1*X(3)+B2*X(4)-XTD
  F(2)=B3*X(3)+B4*X(4)-YTD
  F(3)=(RMT*X(4)*X(5)+THRUST-H1*X(3)*X(3))/RML
  F(4)=X(3)*(X(5)*(H3-RML)-H2*X(4)-   H8*VT)/RMT
  F(5)=(H6*X(3)*X(4)+H7*X(3)*X(5)-   XTA*H8*X(3)*VT)/RJZ
  F(6)=X(5)
  IF(L.GT.NP)RETURN
  IF(L.EQ.0 )RETURN
  DFDX(1,3)=B1
  DFDX(1,4)=B2
  DFDX(1,6)=B2*X(3)-B1*X(4)
  DFDX(2,3)=B3
  DFDX(2,4)=B4
  DFDX(2,6)=B1*X(3)+B2*X(4)
  DFDX(3,3)=-2.*H1*X(3)/RML
  DFDX(3,4)=X(5)*RMT/RML
  DFDX(3,5)=X(4)*RMT/RML
  DFDX(4,3)=F(4)/X(3)+   H8*RK*DEL(L)*X(3)/RMT
  DFDX(4,4)=(-H2*X(3)-   H8*X(3))/RMT
  DFDX(4,5)=(X(3)*(H3-RML)-   H8*X(3)*XTA)/RMT
  DFDX(5,3)=(H6*X(4)+H7*X(5)-   XTA*H8*(VT-X(3)*RK*DEL(L)))/RJZ
  DFDX(5,4)=(H6*X(3)-   XTA*H8*X(3))/RJZ
  DFDX(5,5)=(H7*X(3)-   XTA*H8*X(3)*XTA)/RJZ
  DFDX(6,5)=1.
  DFDDX(1,3,6)=B2
  DFDDX(1,4,6)=-B1
  DFDDX(1,6,3)=B2
  DFDDX(1,6,4)=-B1
  DFDDX(2,3,6)=B1
  DFDDX(2,4,6)=B2
  DFDDX(2,6,3)=B1
  DFDDX(2,6,4)=B2
  DFDDX(3,3,3)=-2.*H1/RML
  DFDDX(3,4,5)=RMT/RML
  DFDDX(3,5,4)=RMT/RML
  DFDDX(4,3,3)=H8*RK*DEL(K)/RMT
  DFDDX(4,3,4)=(-H2-H8)/RMT
  DFDDX(4,3,5)=(H3-RML-H8*XTA)/RMT
  DFDDX(4,4,3)=(-H2-H8)/RMT
  DFDDX(4,5,3)=(H3-RML-H8*XTA)/RMT
  DFDDX(5,3,3)=2.*XTA*H8*RK*DEL(K)/RJZ
  DFDDX(5,3,4)=(H6-XTA*H8)/RJZ
  DFDDX(5,3,5)=(H7-XTA*XTA*H8)/RJZ
  DFDDX(5,4,3)=(H6-XTA*H8)/RJZ
  DFDDX(5,5,3)=(H7-XTA*XTA*H8)/RJZ
  RETURN
  END
```

COST

```
      SUBROUTINE COST(DT,NP,G,DGDT,X,XFP,N)
      DIMENSION X(N),XFP(N)
      G=SQRT((NP-1.)*DT)
      DGDT=(NP-1.)/(2.*G)
      RETURN
      END
```

```
      SUBROUTINE RDLU(A,AINV,CVECTR,X,NCAP,MDFA,INVFG)
      DIMENSION A(NCAP,NCAP),AINV(NCAP,NCAP),CVECTR(NCAP),X(NCAP)
      DOUBLE PRECISION A,AINV,CVECTR,X
      DOUBLE PRECISION DETR,DCOND,DETI
      DOUBLE PRECISION MAXPIV,FLTTS,TSC,P,MINUSI,NTSIGN,DCHKMX,DCHKMN
C * * * * * PROGRAMMED BY DON BARACH  NUC-SD  CODE 601 * * * * *
C     * * * * MAXPIV,MINUSI,NTSIGN MUST BE TYPED REAL FOR SINGLE PRECISION
      DIMENSION ROW(200)
      INTEGER ROW
      COMMON/VALDET/DETR,DETI
      COMMON/DECOND/DCOND
      DETI=0.0
C * * * * * SOLUTION OF THE SYSTEM AX=B FOR X (INVFG=0)
C           OR COMPUTE A INVERSE (INVFG=1)
      MINUSI=-1.0
      NTSIGN=1.0
      MDF=MDFA
      LIMD=1
      IF(INVFG.EQ.1) LIMD=NCAP
      DO 525 II=1, LIMD
      IF(INVFG.EQ.0) GO TO 225
      DO 415 KK=1, NCAP
  415 CVECTR(KK)=0.0
      CVECTR(II)=1.0
      MDF=1
      IF(II.EQ.1) MDF=0
  225 IF(MDF.EQ.1) GO TO 65
C * * * * * DECOMPOSE THE MATRIX A INTO LOWER AND UPPER TRIANGULAR MATRICES
C           WHERE A=LU (WITH COLUMN PIVOTING)
C * * * * * THE ORIGINAL A MATRIX IS DESTROYED
      DO 1  I=1, NCAP
    1 ROW(I)=I
      MAXPIV=-1.0
      NPIVOT=1
      DO 4 I=1, NCAP
      FLTTS=DABS(A(I,1))
      IF(MAXPIV.GT.FLTTS) GO TO 4
      MAXPIV=FLTTS
      NPIVOT=I
    4 CONTINUE
      PRINT 779
  779 FORMAT(1H1)
      L = 1
  781 FORMAT(I3,2X,8HMAXPIV =,D15.6)
      IF ( MAXPIV .EQ. 0. ) GO TO 625
      DCHKMX=MAXPIV
      DCHKMN=MAXPIV
      IF(NPIVOT.EQ.1) GO TO 7
      NTSIGN=NTSIGN*MINUSI
```

```
      NTEMP=ROW(I)
      ROW(I)=ROW(NPIVOT)
      ROW(NPIVOT)=NTEMP
      J=NPIVOT
      DO 6 I=1, NCAP
      TSC=A(1,I)
      A(1,I)=A(J,I)
    6 A(J,I)=TSC
    7 P=1.0D0/A(1,1)
      DO 8 I=2, NCAP
    8 A(1,I)=P*A(1,I)
      DO 55 L=2, NCAP
      MAXPIV=-1.0
      K=L-1
      DO 25 I=L, NCAP
      DO 15 J=1, K
    5 A(I,L)=A(I,L)-A(J,L)*A(I,J)
      FLTTS=DABS(A(I,L))
      IF(MAXPIV.GT.FLTTS) GO TO 25
      MAXPIV=FLTTS
      NPIVOT=I
   25 CONTINUE
      IF ( MAXPIV .EQ. 0. ) GO TO 625
      IF(MAXPIV.GT.DCHKMX) DCHKMX=MAXPIV
      IF(MAXPIV.LT.DCHKMN) DCHKMN=MAXPIV
      IF(NPIVOT.EQ.L) GO TO 28
      NTSIGN=NTSIGN*MINUS1
      NTEMP=ROW(L)
      ROW(L)=ROW(NPIVOT)
      ROW(NPIVOT)=NTEMP
      J=NPIVOT
      DO 27 I=1, NCAP
      TSC=A(L,I)
      A(L,I)=A(J,I)
   27 A(J,I)=TSC
   28 P=1.0  /A(L,L)
      K=L+1
      M=L-1
      IF(L.EQ.NCAP) GO TO 55
      DO 45 I=K, NCAP
      DO 35 J=1, M
   35 A(L,I)=A(L,I)-A(L,J)*A(J,I)
   45 A(L,I)=P*A(L,I)
   55 CONTINUE
      DCOND=DLOG10(DCHKMX)-DLOG10(DCHKMN)
      DETR=A(1,1)
      DO 205 I=2, NCAP
  205 DETR=DETR*A(I,I)
      DETR=DETR*NTSIGN
C * * * * * USES A RIGHT HAND SIDE B AND SOLVES FOR THE UNKNOWN X BY TWO
C           SUCCESSIVE BACK SUBSTITUTIONS LY=B AND UX=Y
   65 DO 66 I=1, NCAP
```

```
   66 X(I)=0.0
      J=ROW(I)
      X(I)=CVECTR(J)/A(I,1)
      DO 85 I=2, NCAP
      J=ROW(I)
      K=I-1
      DO 75 L=1, K
   75 X(I)=X(I)+A(I,L)*X(L)
   85 X(I)=(CVECTR(J)-X(I))/A(I,I)
      K=NCAP-1
      DO 105 I=1, K
      J=NCAP-I
      M=NCAP
      DO 95 L=1, I
      X(J)=X(J)-X(M)*A(J,M)
   95 M=M-1
  105 CONTINUE
      IF(INVFG.EQ.0) GO TO 525
C * * * * * LOADS THE INVERSE OF THE MATRIX
      DO 420 LL=1, NCAP
  420 AINV(LL,II)=X(LL)
  525 CONTINUE
      RETURN
  625 PRINT 626, L
  626 FORMAT(' SINGULAR MATRIX - - COLUMN', I4, ' ALL ZEROS' )
      STOP
      END
```

```
                    CONGRA

      PARAMETER N=6,NP=301
      DIMENSION       DFDX(N,N,NP),X(N,NP),DFDU(N,NP),DEL(NP),P(N,NP),
     $PF(N,NP),G(NP),DHXX(N,N),DHXU(N,NP),W(NP),Z(N,NP),C(N,NP),D(NP),
     $CF(N,NP),S(NP),DEL1(NP),  ALP(3),DEL2(NP),Q(3,3),COEF(3)    ,VEL(3)
      DIMENSION VVAL(10),Q1(3,3)
      DIMENSION VAL(20),DAL(20)
      DIMENSION F(N,NP),ZF(N,NP),VF(NP),V(NP),VSF(NP),SSF(NP),VSI(NP),
     $SSI(NP)
      NP1=NP-1
      TF=.6
      TF=3.
      DT=TF/(NP-1.)
      E1=100.
      E2=100.
      DMAX=15./57.29578
      X(1,1)=-1000.
      X(2,1)=0.
      X(3,1)=67.56
      X(4,1)=0.
      X(5,1)=0.
      X(6,1)=0.
      DO 32 I=1,50
   32 DEL(I)=-DMAX
      DO 31 I=1,NP1
      L=I+1
      CALL FCT(F,DFDX,X,N,I,DFDU,NP,DEL)
      DO 29 J=1,N
      X(J,L)=X(J,1)+DT*F(J,I)
   29 CONTINUE
      GO TO 31
   50 DO 51 J=1,N
   51 X(J,L)=X(J,I)+DT*(55.*F(J,I)-59.*F(J,I-1)+37.*F(J,I-2)
     $  -9.*F(J,I-3))/24.
      CALL FCT(F,DFDX,X,N,L,DFDU,NP,DEL)
      DO 30 J=1,N
      X(J,L)=X(J,I)+DT*(9.*F(J,L)+19.*F(J,I)-5.*F(J,I-1)+F(J,I-2))/24.
   30 CONTINUE
   31 CONTINUE
      DN=8.
      X6F=X(6,NP)
      X1F=X(1,NP)
      X2F=X(2,NP)
      X1F=X(1,NP)+DN*COS(X6F)
      X2F=X(2,NP)+DN*SIN(X6F)
      X3F=X(3,NP)
      X4F=X(4,NP)
      X5F=X(5,NP)
      PRINT 100,(DEL(I),X(1,I),X(2,I),X(3,I),X(4,I),X(5,I),X(6,I),I,
     $I=1,NP,5)
```

```
      TFI=.6
      TFI=3.
      TFD=-.01
      DO 52 M=1,2
      TF=TFI+(M-1.)*TFD
      DO 33 I=40,50
   33 DEL(I)=0.
      TO=0.
      DO 18 I=1,NP
   18 W(I)=1.
      ILOOP=0
      DT=TF/(NP-1.)
      PRINT 106,M,TF,DT
  106 FORMAT(1H1,5H M = ,I5,5X,5HTF = ,E15.10,5X,5HDT = ,E15.10)
      DO 54 I=1,NP
   54 S(I)=0.
      BET=0.
   15 DO 16 I=1,N
   16 Z(I,1)=0.
      VALO=0.
      DO 1 I=1,NP1
      II=I+1
      CALL FCT(F,DFDX,X,N,I,DFDU,NP,DEL)
      DO 34 J=1,N
      ZF(J,I)=DFDU(J,I)*W(I)*S(I)
      DO 34 K=1,N
   34 ZF(J,I)=ZF(J,I)+DFDX(J,K,I)*Z(K,I)
      DO 36 J=1,N
      X(J,II)=X(J,I)+DT*F(J,I)
   36 Z(J,II)=Z(J,I)+DT*ZF(J,I)
      GO TO 1
   35 DO 37 J=1,N
      X(J,II)=X(J,I)+DT*(55.*F(J,I)-59.*F(J,I-1)+37.*F(J,I-2)-9.*F(J,I-3
     $))/24.
   37 Z(J,II)=Z(J,I)+DT*(55.*ZF(J,I)-59.*ZF(J,I-1)+37.*ZF(J,I-2)-
     $9.*ZF(J,I-3))/24.
      CALL FCT(F,DFDX,X,N,II,DFDU,NP,DEL)
      DO 38 J=1,N
      ZF(J,II)=DFDU(J,II)*W(II)*S(II)
      DO 38 K=1,N
   38 ZF(J,II)=ZF(J,II)+DFDX(J,K,II)*Z(K,II)
      DO 39 J=1,N
      X(J,II)=X(J,I)+DT*(9.*F(J,II)+19.*F(J,I)-5.*F(J,I-1)+F(J,I-2))/24.
   39 Z(J,II)=Z(J,I)+DT*(9.*ZF(J,II)+19.*ZF(J,I)-5.*ZF(J,I-1)+ZF(J,I-2)
     $)/24.
    1 CONTINUE
      PRINT 100,(DEL(I),X(1,I),X(2,I),X(3,I),X(4,I),X(5,I),X(6,I),I,
     $I=1,NP,5)
      VALO=E1*(X(1,NP)-X1F)**2+E2*(X(2,NP)-X2F)**2
      VALO=E1*(X(1,NP)+DN*COS(X(6,NP))-X1F)**2+E2*(X(2,NP)+DN*SIN(X(6,NP
     $))-X2F)**2
      VALO=VALO+TF
```

```
      VVAL(M)=VALO
      C(1,NP)=2.*E1*Z(1,NP)
      C(2,NP)=2.*E2*Z(2,NP)
      C(3,NP)=0.
      C(4,NP)=0.
      C(5,NP)=0.
      C(6,NP)=0.
      P(1,NP)=2.*E1*(X(1,NP)-X1F)
      P(1,NP)=2.*E1*(X(1,NP)-X1F+DN*COS(X(6,NP)))
      P(2,NP)=2.*E2*(X(2,NP)-X2F)
      P(2,NP)=2.*E2*(X(2,NP)-X2F+DN*SIN(X(6,NP)))
      P(3,NP)=0.
      P(4,NP)=0.
      P(5,NP)=0.
      P(6,NP)=0.
      P(6,NP)=DN*(P(2,NP)*COS(X(6,NP))-P(1,NP)*SIN(X(6,NP)))
      DEL(NP)=DEL(NP1)
      CALL FCT(F,DFDX,X,N,NP,DFDU,NP,DEL)
      ITE=0
      VSI(NP)=0.
      SSI(NP)=0.
      DO 4 I=NP1,1,-1
      L=I+1
      CALL COFCT(X,P,PF,G,DHX,DHXU,Z,C,D,CF,N,L,NP,DFDX,DFDU,W,S,DEL)
      VSF(L)=W(I)*G(L)*G(L)
      SSF(L)=W(I)*S(L)*D(L)
      DO 3 J=1,N
      P(J,I)=P(J,L)-DT*PF(J,L)
    3 C(J,I)=C(J,L)-DT*CF(J,L)
      VSI(I)=VSI(L)-DT*VSF(L)
      SSI(I)=SSI(L)-DT*SSF(L)
      GO TO 4
   40 DO 41 J=1,N
      P(J,I)=P(J,L)-DT*(55.*PF(J,L)-59.*PF(J,L+1)+37.*PF(J,L+2)-
     $9.*PF(J,L+3))/24.
   41 C(J,I)=C(J,L)-DT*(55.*CF(J,L)-59.*CF(J,L+1)+37.*CF(J,L+2)-
     $9.*CF(J,L+3))/24.
      VSI(I)=VSI(L)-DT*(55.*VSF(L)-59.*VSF(L+1)+37.*VSF(L+2)+9.*VSF(L+3)
     $)/24.
      SSI(I)=SSI(L)-DT*(55.*SSF(L)-59.*SSF(L+1)+37.*SSF(L+2)+9.*SSF(L+3)
     $)/24.
      CALL COFCT(X,P,PF,G,DHX,DHXU,Z,C,D,CF,N,I,NP,DFDX,DFDU,W,S,DEL)
      SSF(I)=W(I-1)*S(I)*D(I)
      VSF(I)=W(I-1)*G(I)*D(I)
      DO 42 J=1,N
      P(J,I)=P(J,L)-DT*(9.*PF(J,I)+19.*PF(J,L)-5.*PF(J,L+1)+PF(J,L+2))
     $/24.
   42 C(J,I)=C(J,L)-DT*(9.*CF(J,I)+19.*CF(J,L)-5.*CF(J,L+1)+CF(J,L+2))
     $/24.
      SSI(I)=SSI(L)-DT*(9.*SSF(I)+19.*SSF(L)-5.*SSF(L+1)+SSF(L+2))/24.
      VSI(I)=VSI(L)-DT*(9.*VSF(I)+19.*VSF(L)-5.*VSF(L+1)+VSF(L+2))/24.
    4 CONTINUE
```

```
      CALL COFCT(X,P,PF,G,DMX,DMXU,Z,C,D,CF,N,I,NP,DFDX,DFDU,W,S,DEL)
      PRINT 101,((P(I,J),I=1,6),J,J=1,NP,10)
      PRINT 102,(G(I),D(I),W(I),S(I),I,I=1,NP,10)
  102 FORMAT(5H G = ,(T7,4(E15.5,10,2X),I5))
      IF(ILOOP.EQ.0)GO TO 55
      BET=VSI(1)/VSOLD
   55 CONTINUE
      VSOLD=VSI(1)
      DO 5 I=1,NP1
    5 S(I)=G(I)+BET*S(I)
      DO 21 I=1,20
   21 DAL(I)=I*.1
   12 DO 8 K=1,20
      VAL(K)=0.
      DO 6 I=1,NP
      DEL1(I)  =DEL(I)-DAL(K)*S(I)
    6 IF(ABS(DEL1(I)).GT.DMAX)DEL1(I)=SIGN(DMAX,DEL1(I))
      DO 7 I=1,NP1
      II=I+1
      CALL FCT(F,DFDX,X,N,I,DFDU,NP,DEL1)
      DO 44 J=1,N
   44 X(J,II)=X(J,I)+DT*F(J,I)
      GO TO 7
   43 DO 45 J=1,N
   45 X(J,II)=X(J,I)+DT*(55.*F(J,I)-59.*F(J,I-1)+37.*F(J,I-2)-
     $9.*F(J,I-3))/24.
      CALL FCT(F,DFDX,X,N,II,DFDU,NP,DEL1)
      DO 46 J=1,N
   46 X(J,II)=X(J,I)+DT*(9.*F(J,II)+19.*F(J,I)-5.*F(J,I-1)+F(J,I-2))/24.
    7 CONTINUE
    8 VAL(K)=E1*(X(1,NP)+DN*COS(X(6,NP))-X1F)**2+TF*
     $ E2*(X(2,NP)+DN*SIN(X(6,NP))-X2F)**2
      L=1
      VEL(2)=VAL(1)
      DO 22 I=2,20
      IF(VAL(I).GE.VEL(2))GO TO 22
      VEL(2)=VAL(I)
      L=I
   22 CONTINUE
      IF(VEL(2).GT.VALO)GO TO 23
      IF(L.EQ.1)GO TO 27
      VEL(1)=VAL(L-1)
      VEL(3)=VAL(L+1)
      ALP(1)=DAL(L-1)
      ALP(2)=DAL(L)
      ALP(3)=DAL(L+1)
      GO TO 24
   27 VEL(1)=VALO
      VEL(3)=VAL(2)
      ALP(1)=0.
      ALP(2)=DAL(1)
      ALP(3)=DAL(2)
```

```
      GO TO 24
   23 ITE=ITE+1
      DO 25 I=1,20
   25 DAL(I)=DAL(I)/20.
      IF(ITE.GT.5)GO TO 26
      GO TO 12
   26 PRINT 110,VALO
  110 FORMAT('    FUNCTION CONVERGED TO ',E20.15)
      GO TO 52
   24 CONTINUE
      PRINT 103,VALO,(VEL(I),I=1,3)
  103 FORMAT(7H VAL = ,E15.10,3(5X,E15.10))
      IF(ABS(VALO-VEL(2)).LT.,001)GO TO 52
      Q(1,1)=1.
      Q(2,1)=1.
      Q(3,1)=1.
      Q(1,2)=ALP(1)
      Q(2,2)=ALP(2)
      Q(3,2)=ALP(3)
      Q(1,3)=ALP(1)**2
      Q(2,3)=ALP(2)**2
      Q(3,3)=ALP(3)**2
      D1=VEL(1)-VEL(2)
      D2=VEL(2)-VEL(3)
      IF(ABS(D1).LT..0000001)GO TO 19
      IF(ABS(D2).LT..0000001)GO TO 19
      GO TO 20
   19 ALPOP=ALP(2)
      GO TO 13
   20 CALL RDLU(Q,QI,VEL,COEF,3,0,0)
      PRINT 105,(ALP(I),I=1,3),(COEF(I),I=1,3)
  105 FORMAT(7H ALP = ,3(E15.10,2X),3(E12.7,1X))
      ALPOP=-COEF(2)/(2.*COEF(3))
      PRINT 104,ALPOP,(COEF(I),I=1,3)
  104 FORMAT(7H ALP = ,5(E15.10,5X))
   13 DO 14 I=1,NP
      DEL(I)=DEL(I)-ALPOP*S(I)
      W(I)=1.
      IF(ABS(DEL(I)).GT.DMAX)W(I)=0.
   14 IF(ABS(DEL(I)).GT.DMAX)DEL(I)=SIGN(DMAX,DEL(I))
      ILOOP=ILOOP+1
  100 FORMAT((7H DEL = ,E15.10,6(2X,E13.8),2X,I5))
  101 FORMAT(5H P = ,(T7,6(E15.10,2X),I5))
      IF(ILOOP.LE.20)GO TO 15
   52 CONTINUE
      PRINT 53,(M,VVAL(M),M=1,10)
   53 FORMAT(10(5H M = ,I5,5X,E15.10,/))
  200 FORMAT((5H I = ,I5,5X,6HDEL = ,E15.10))
      STOP
      END
```

FCT

```
      SUBROUTINE FCT(F,DFDX,X,N,M,DFDU,NP,DEL)
      DIMENSION       DFDX(N,N,NP),X(N,NP),DFDU(N,NP),DEL(NP),F(N,NP)
      COMMON C1,C2,C3,C4,C5,C6,C7,C8,C9
      B1=COS(X(6,M))
      B2=-SIN(X(6,M))
      X1=X(1,M)
      X2=X(2,M)
      X3=X(3,M)
      X4=X(4,M)
      X5=X(5,M)
      I=M+1
      F(1,M)=B1*X3+B2*X4
      F(2,M)=-B2*X3+B1*X4
      F(3,M)=C1*X3*X3+C2*X5*X4+C3
      F(4,M)=C4*X3*X4+C5*X3*X5+C6*X3*X3*DEL(M)
      F(5,M)=C7*X3*X4+C8*X3*X5+C9*X3*X3*DEL(M)
      F(6,M)=X5
      DFDX(1,3,M)=B1
      DFDX(1,4,M)=B2
      DFDX(1,6,M)=B2*X3-B1*X4
      DFDX(2,3,M)=-B2
      DFDX(2,4,M)=B1
      DFDX(2,6,M)=B1*X3+B2*X4
      DFDX(3,3,M)=2.*C1*X3
      DFDX(3,4,M)=C2*X5
      DFDX(3,5,M)=C2*X4
      DFDX(4,3,M)=C4*X4+C5*X5*2.*C6*X3*DEL(M)
      DFDX(4,4,M)=C4*X3
      DFDX(4,5,M)=C5*X3
      DFDX(5,3,M)=C7*X4+C8*X5*2.*C9*X3*DEL(M)
      DFDX(5,4,M)=C7*X3
      DFDX(5,5,M)=C8*X3
      DFDX(6,5,M)=1.
      DFDU(4,M)=C6*X3*X3
      DFDU(5,M)=C9*X3*X3
      RETURN
      END
```

COFCT

```
      SUBROUTINE COFCT(X,P,PF,G,DHXX,DHXU,Z,C,D,CF,N,M,NP,DFDX,DFDU,W,S,
     $DEL)
      DIMENSION X(N,NP),P(N,NP),PF(N,NP),G(NP),DHXX(N,N),DHXU(N,NP)
     $,Z(N,NP),C(N,NP),D(NP),CF(N,NP),DFDX(N,N,NP),DFDU(N,NP),S(NP)
     $,W(NP),DEL(NP)
      COMMON C1,C2,C3,C4,C5,C6,C7,C8,C9
      DATA IND/0/
      IF(IND.EQ.0)PRINT 10,C1,C2,C3,C4,C5,C6,C7,C8,C9
   10 FORMAT(4H C = ,/,1X,9(E12.7,1X))
      P1=P(1,M)
      P2=P(2,M)
      P3=P(3,M)
      P4=P(4,M)
      P5=P(5,M)
      P6=P(6,M)
      X1=X(1,M)
      X2=X(2,M)
      X3=X(3,M)
      X4=X(4,M)
      X5=X(5,M)
      X6=X(6,M)
      B1=COS(X6)
      B2=-SIN(X6)
      L=M-1
      PF(3,M)=-P1*B1+P2*B2-2.*P3*C1*X3-P4*(C4*X4+C5*X5+2.*C6*X3*DEL(L))
     $  -P5*(C7*X4+C8*X5+2.*C9*X3*DEL(L))
      PF(4,M)=-P1*B2-P2*B1-P3*C2*X5-P4*C4*X3-P5*C7*X3
      PF(5,M)=-P3*C2*X4-P4*C5*X3-P5*C8*X3-P6
      PF(6,M)=-P1*X3*B2+P1*X4*B1-P2*X3*B1-P2*X4*B2
      DHXU(3,M)=2.*X3*(P4*C6+P5*C9)
      G(M)=(P4*C6+P5*C9)*X3*X3
      IF(IND.EQ.0)S(M)=G(M)
      IF(M.EQ.2)IND=2
      DHXX(3,3)=2.*(P3*C1+(P4*C6+P5*C9)*DEL(L))
      DHXX(3,4)=P4*C4+P5*C7
      DHXX(3,5)=P4*C5+P5*C8
      DHXX(3,6)=P1*B2+P2*B1
      DHXX(4,3)=DHXX(3,4)
      DHXX(4,5)=P3*C2
      DHXX(4,6)=-P1*B1+P2*B2
      DHXX(5,3)=DHXX(3,5)
      DHXX(5,4)=DHXX(4,5)
      DHXX(6,3)=DHXX(3,6)
      DHXX(6,4)=DHXX(4,6)
      DHXX(6,6)=-P1*(X3*B1+X4*B2)+P2*(X3*B2-X4*B1)
      DO 1 I=3,6
      CF(I,M)=0.
      DO 2 J=1,N
    2 CF(I,M)=CF(I,M)-DFDX(J,I,M)*C(J,M)-DHXX(I,J)*Z(J,M)
```

```
1 CF(I,M)=CF(I,M)+DHXU(I,M)*W(L)*S(M)
  D(M)=DHXU(3,M)*Z(3,M)+DFDU(4,M)*C(4,M)+DFDU(5,M)*C(5,M)
  RETURN
  END
```

```
                    MNLNRL

   COMPILER(XM=1)
   PARAMETER NP=1501,NP1=NP-1,NL=150
   DIMENSION O(NL),V(NL)
   COMMON P(6),A(6,6),B(6),X(6),G(6,6),XF(6),R(6,6),RD(6,6),XD(6),
  $U(NP),RS(6,6,NP),XPL(6,NL),XI(6),XDIF(6,NP),XE(6),XDFS(NP)
   CALL COEFC(C1,C2,C3,C4,C5,C6,C7,C8,C9)
   X3=67.56
50 READ 1,(X(I),I=1,6),UP,UIND
 1 FORMAT(8F10.5)
   READ 1,(XF(I),I=1,6),TF
   UP=UP*1000.
   PRINT 33,UIND
33 FORMAT('  UIND = ',F10.5,//)
   PRINT 20,(X(I),I=1,6),(XF(I),I=1,6),UP
20 FORMAT(5H X = ,6(E15.10,5X),/,6H XF = ,6(E15.10,5X),/,6H UP = ,
  $F10.3,//)
   DO 13 I=1,6
13 XI(I)=X(I)
   X6=ATAN2(XF(2)-X(2),XF(1)-X(1))
   A(1,3)=COS(X6)
   A(2,3)=SIN(X6)
   A(3,3)=2.*C1*X3
   A(1,4)=-A(2,3)
   A(2,4)=A(1,3)
   A(4,4)=C4*X3
   A(5,4)=C7*X3
   A(4,5)=C5*X3
   A(5,5)=C8*X3
   A(6,5)=1.
   A(1,6)=-X3*A(2,3)
   A(2,6)=X3*A(1,3)
   B(4)=C6*X3*X3
   B(5)=C9*X3*X3
   X(1)=0.
   X(2)=0.
   X(3)=X(3)-X3
   X(6)=X(6)-X6
   T=0.
   XDFI=0.
   DO 22 I=1,2
   G(I,I)=100.
   RS(I,I,NP)=-100.
22 CONTINUE
   IPL=0
   PRINT 26,C1,C2,C3,C4,C5,C6,C7,C8,C9
26 FORMAT(5H C = ,9(E13.8,1X)  )
   PRINT 25,((A(I,J),J=1,6),I=1,6),(B(I),I=1,6)
25 FORMAT(5H A = ,6(T6,6(E15.10,5X),/),///,5H B = ,6(E15.10,5X),//)
   H=TF/(NP-1.)
```

```
      IP=0
      DO 2 II=1,NP1
      K=NP-II
      M=K+1
      DO 4 I=1,6
      DO 4 J=1,6
    4 R(I,J)=RS(I,J,M)
      CALL FCEV(A,B,UP,RD,R)
      DO 3 I=1,6
      DO 3 J=1,6
    3 RS(I,J,K)=RS(I,J,M)-H*RD(I,J)
      IP=IP+1
      IF(IP.LT.50)GO TO 2
   34 CONTINUE
      PRINT 21,II,K,((RS(I,J,K),J=1,6),I=1,6),((RD(I,J),J=1,6),I=1,6)
   21 FORMAT(6H II = ,I5,5H K = ,I5,/,6(6(5X,E15.10),/),//,6(6(5X,E15.10
     $),/))
      IP=0
    2 CONTINUE
      IF(UND.GT.1.)GO TO 32
      DO 31 K=2,NP
      DO 31 I=1,6
      DO 31 J=1,6
   31 RS(I,J,K)=RS(I,J,1)
   32 CONTINUE
      IP=0
      DO 5 I=1,NP1
      II=I+1
      U(I)=0.
      DO 6 J=1,6
    6 U(I)=U(I)+X(J)*(RS(4,J,I)*B(4)+RS(5,J,I)*B(5))/UP
      DO 7 J=1,6
      XD(J)=B(J)*U(I)
      DO 7 K=1,6
    7 XD(J)=XD(J)+A(J,K)*X(K)
      UC=U(I)
       DO 19 J=1,6
   19 P(J)=X(J)+XI(J)
      CALL FC(P,XE,C1,C2,C3,C4,C5,C6,C7,C8,C9,UC)
      DO 28 J=1,6
   28 XDIF(J,I)=XE(J)-XD(J)
      XDFS(I)=0.
      DO 29 J=1,6
   29 XDFS(I)=XDFS(I)+XDIF(J,I)**2
      XDF1=XDF1+H*XDFS(I)
      DO 8 J=1,6
    8 X(J)=X(J)+H*XD(J)
      IP=IP+1
      T=T+H
      IF(IP.EQ.10)GO TO 9
      GO TO 5
    9 IPL=IPL+1
```

```
      PRINT 10,U(I),(X(J),J=1,6),I
      DO 11 J=1,6
   11 XPL(J,IPL)=X(J)+XI(J)+(XF(J)-XI(J))*T/TF
      PRINT 18,T,(XPL(J,IPL),J=1,6)
   18 FORMAT(8H TIME = ,E10.5,6(2X,E15.10))
      IP=0
   10 FORMAT(6H U = ,E15.10,3X,6(E15.10,1X),I5)
    5 CONTINUE
      UINT=0.
      DO 27 I=1,NP1
   27 UINT=UINT+U(I)*H
      PRINT 30,UINT,XDFI,(I,XDFS(I),(XDIF(J,I),J=1,6),I=1,NP1,10)
   30 FORMAT(8H UINT = ,E15.10,8H XDFI = ,E15.10,/,8H XDIF = ,/,(1X,I8,
     $1X,7(E15.10,2X)))
      IP=0
      IPL=0
      DO 12 I=1,NL
      O(I)=XPL(1,I)
   12 V(I)=XPL(2,I)
      CALL OPNPLT
      CALL XYEPLT
      CALL LINPLT(O,V,NL)
      DO 14 I=1,6
   14 X(I)=XI(I)
      DO 15 I=1,NP1
      UC=U(I)
      CALL FC(X,XD,C1,C2,C3,C4,C5,C6,C7,C8,C9,UC)
      DO 16 J=1,6
   16 X(J)=X(J)+H*XD(J)
      IP=IP+1
      IF(IP.EQ.10)GO TO 17
      GO TO 15
   17 IPL=IPL+1
      PRINT 10,U(I),(X(J),J=1,6),I
      O(IPL)=X(1)
      V(IPL)=X(2)
      IP=0
   15 CONTINUE
      CALL CURVE(O,V,NL,0)
      CALL ENDPL(1)
      STOP
      END
```

COEFC

```
      SUBROUTINE COEFC(C1,C2,C3,C4,C5,C6,C7,C8,C9)
      RM=25.
      AA=.9
      RHO=2
      RL=15.
      XTA=-7.5
      RK=1.07/2.1
      DP=.15
      RF=RL/(1.128*SQRT(AA))
      RK1=.65*RF**(-1.49)
      RK2=.73*RF**(.1139)
      RML=800.*(1.+RK1)/32.2
      RMT=800.*(1.+RK2)/32.2
      RKP=.42*RF**.331
      RJZ=400.+340.*RKP
      RLTAP=2.5
      THRUST=(RHO/2.)*AA*DP*67.56*67.56
      H8=(RHO/2.)*AA*RLTAP
      H2=(RHO/2.)*AA
      H1=H2*DP
      H3=H2*RL*.3
      H6=-1.3*H2*RL
      H7=-.2*H2*RL*RL
      C1=-H1/RML
      C2=RMT/RML
      C3=THRUST/RML
      C4=-(H2+H8)/RMT
      C5=(H3-RML-H8*XTA)/RMT
      C6=H8*RK/RMT
      C7=(H6-XTA*H8)/RJZ
      C8=(H7-XTA*XTA*H8)/RJZ
      C9=XTA*H8*RK/RJZ
      RETURN
      END
```

FC

```
      SUBROUTINE FC(X,XD,C1,C2,C3,C4,C5,C6,C7,C8,C9,U)
      DIMENSION X(6),XD(6)
      XD(1)=X(3)*COS(X(6))-X(4)*SIN(X(6))
      XD(2)=X(3)*SIN(X(6))+X(4)*COS(X(6))
      XD(3)=C1*X(3)**2+C2*X(4)*X(5)+C3
      XD(4)=C4*X(3)*X(4)+C5*X(3)*X(5)+C6*X(3)*X(3)*U
      XD(5)=C7*X(3)*X(4)+C8*X(3)*X(5)+C9*X(3)*X(3)*U
      XD(6)=X(5)
      RETURN
      END
```

FCEV

```
      SUBROUTINE FCEV(A,B,UP,RD,R)
      DIMENSION A(6,6),B(6),RD(6,6),R(6,6)
      DO 1 I=1,6
      DO 1 J=1,6
      RD(I,J)=-(R(I,4)*B(4)+R(I,5)*B(5))*(R(4,J)*B(4)+R(5,J)*B(5))/UP
      DO 1 K=1,6
    1 RD(I,J)=RD(I,J)-A(K,I)*R(K,J)-R(I,K)*A(K,J)
      RETURN
      END
```

```
                    MNLNEP

      COMPILER(XM=1)
      DOUBLE PRECISION HGRWD,HGRWI
      DOUBLE PRECISION D,E,E2
      DOUBLE PRECISION COD,COH,COHI,CODD
      DOUBLE PRECISION HGRW,HDMC,HDMD
      PARAMETER NOP=13
      PARAMETER N=6,M=12,NP=51,ID1=N*NP+1,ID2=N*M+5,N2=N-2
      DIMENSION E(ID2),E2(ID2)
      DIMENSION XPLT(NP),YPLT(NP),XPTG(NP),YPTG(NP)
      COMMON/A/DFDDX(6,6,6),DUTEST
      COMMON/CALL/TF
      COMMON X(N),XIN(N),XPL(N,180),XD(N),AD(N),ADD(N),DXDA(N,M),F(N),
     $DXDDA(N,M),DXDAD(N),DWDDT(NP,N),A(N,M),DEL(NP),DW1DA(ID1,ID2),
     $DXDDAD(N),DXDT(N),DXDDT(N),DFDX(N,N),W(NP,N),W1(ID1),A1(ID2),
     $DWDA(NP,N,N,M),DWDAD(NP,N,8),HGRW(ID2,ID2),HDMC(ID2),HDMD(ID2),
     $HGRW1(ID2,ID2),D(ID2),DFDT(N),XFP(N),AOF(N),A10(ID2),A15(ID2,30),
     $A1P(ID2,20),DGDA(ID2),COC(N,180),COB(M,180),COD(M),COH(M,M),
     $COHI(M,M),CODD(M),COG(N,M)
      INA=1
      XFP(1)=0.
      XFP(2)=0.
      PI=3.1415926536
      COEF=1.
      DN=8
   22 FORMAT(8F10.4)
      PRINT 23,COEF,XTD,YTD,ALPHA
   23 FORMAT(8H COEF = ,F10.4,6HXTD = ,F10.4,2X,6HYTD = ,F10.4,2X,F10.7)
      ZETEST=SQRT(3.)/2.
      COEFO=COEF
      DUTEST=.0002
      DMAX=15./57.29578
      DUTEST=DMAX
   21 CALL CALLIN
      DO 13 I=1,N
      F(I)=0.
      DXDAD(I)=0.
      DXDDAD(I)=0.
      DXDT(I)=0.
      DXDDT(I)=0.
      DO 14 J=1,M
      DXDA(I,J)=0.
      DXDDA(I,J)=0.
   14 COG(I,J)=0.
      XD(I)=0.
      AOF(I)=0.
      XFP(I)=0.
      DO 25 K=1,N
   25 DFDX(I,K)=0.
   13 DFDT(I)=0.
```

```
      DO 15 I=1,M
      COD(I)=0.
      CODD(I)=0.
      DO 15 J=1,M
   15 COHI(I,J)=0.
      DO 16 I=1,NP
      DEL(I)=0.
      DO 16 J=1,N
      DWDDT(I,J)=0.
      W(I,J)=0.
      DO 17 K=1,8
   17 DWDAD(I,J,K)=0.
      DO 16 K=1,N
      DO 16 L=1,M
   16 DWDA(I,J,K,L)=0.
      DO 18 I=1,ID2
      DGDA(I)=0.
      A1(ID2)=0.
      HDMC(ID2)=0.
      HDMD(ID2)=0.
      D(ID2)=0.
      DO 24 J=1,28
      AIS(I,J)=0.
   24 AIP(I,J)=0.
      AIS(I,29)=0.
      AIS(I,30)=0.
      A10(I)=0.
      DO 18 J=1,ID2
      HGRW(I,J)=0.
   18 HGRWI(I,J)=0.
      DO 19 I=1,ID1
      W1(ID1)=0.
      DO 19 J=1,ID2
   19 DW1DA1(I,J)=0.
      DT=TF/179.
      DO 48 I=1,6
      AD(I)=XIN(I)
   48 ADD(I)=X(I)-XIN(I)
      DO 42 K=1,180
      T=(K-1.)*DT
      DO 43 J=1,N
   43 COC(J,K)=XPL(J,K)-AD(J)-ADD(J)*T/TF
      DO 44 J=1,M
   44 COB(J,K)=SIN(J*P1*T/TF)
   42 CONTINUE
      DO 45 I=1,M
      DO 45 J=1,M
      COH(I,J)=0.
      DO 45 K=1,180
   45 COH(I,J)=COH(I,J)+COB(I,K)*COB(J,K)
      CALL RDLU(COH,COHI,COD,CODD,M,0,1)
      DO 46 I=1,N
```

```
      DO 46 J=1,M
      COG(I,J)=0.
      DO 46 K=1,180
   46 COG(I,J)=COG(I,J)+COC(I,K)*COB(J,K)
      DO 47 I=1,N
      DO 47 J=1,M
      A(I,J)=0.
      DO 47 K=1,M
   47 A(I,J)=A(I,J)+COG(I,K)*COHI(K,J)
      PRINT 105,TF,(AO(I),I=1,5),(AOO(I),I=1,5)
      DT=TF/(NP-1.)
      DIAG=.1
      EPS=.01
      EPS=.001
      ILOP=0
   40 ILOP=ILOP+1
      ILOOP=0
      DIAG=DIAG/10.
      PRINT 41,EPS,DIAG
   41 FORMAT(7H EPS = ,E20.15,5X,6HDIA = ,E20.15)
  170 FORMAT(4F20.10)
      PRINT 103,((A(I,J),J=1,M),I=1,N)
   20 SQDE=SQRT(DT/EPS)
      DO 500 I=1,ID2
      DO 500 J=1,ID2
  500 HGRW(I,J)=0.
      XTARG=1000.
      YTARG=0.
      DO 1 L=1,NP
      CALL FCT(X,XD,A,AO,AOO,DXDA,DXDDA,DXDAO,DXDDAO,DXDT,DXDDT,L,N,M,NP
     $,DT,ALPHA,AOF)
      CALL DFCT(F,X,XD,DFDX,L,N,DT,DEL,NP,XTD,YTD)
      XPLT(L)=X(1)+XTARG
      YPLT(L)=X(2)+YTARG
      XPTG(L)=XTARG
      YPTG(L)=YTARG
   90 CONTINUE
      DO 2 J=1,N
      DO 2 K=1,N
      DO 2 I=1,M
      DWDA(L,J,K,I)=-DFDX(J,K)*DXDA(K,I)*SQDE
    2 IF(J.EQ.K)DWDA(L,J,K,I)=DWDA(L,J,K,I)+DXDDA(J,I)*SQDE
      DO 65 J=1,N
      DO 66 K=1,N2
      DWDAO(L,J,K)=-DFDX(J,K+2)*DXDAO(K+2)*SQDE
   66 IF(J.EQ.K+2)DWDAO(L,J,K)=DWDAO(L,J,K)+SQDE*DXDDAO(J)
   65 CONTINUE
      DO 87 J=1,N
      DFDT(J)=0.
      DO 87 I=1,N
   87 DFDT(J)=DFDT(J)+DFDX(J,I)*DXDT(I)
      DO 3 J=1,N
```

```
      W(L,J)=(XD(J)-F(J))*SQDE
    3 DWDDT(L,J)=W(L,J)/(2.*DT)+(DXDDT(J)-DFDT(J))*SQDE
      TIME=(L-1.)*DT
      GO TO 509
      IF(ILOOP.LT.5)GO TO 509
      DO 501 I2=1,N
      DO 501 I4=1,M
      J1=I2*M-M+I4
      DO 502 I3=1,N
      DO 502 I5=1,M
      J2=I3*M-M+I5
      DO 502 I1=1,N
  502 HGRW(J1,J2)=HGRW(J1,J2)-DFDDX(I1,I2,I3)*DXDA(I2,I4)*DXDA(I3,I5)*
     $SQDE*W(L,I1)
      DO 503 I3=1,4
      I5=N*M+I3
      DO 505 I1=1,N
  505 HGRW(J1,J3)=HGRW(J1,J3)-DFDDX(I1,I2,I3)*DXDA(I2,I4)*DXDAO(I3)*SQDE
     $*W(L,I1)
  503 HGRW(J3,J1)=HGRW(J1,J3)
      DO 504 I1=1,N
  504 HGRW(J1,ID2)=HGRW(J1,ID2)+(DWDA(L,I1,I2,I4)/(2.*DT)-DXDDA(I2,I4)*
     $SQDE/DT)*W(L,I1)
  501 HGRW(ID2,J1)=HGRW(J1,ID2)
      DO 506 J2=1,4
      J3=N*M+J2
      DO 507 I1=1,N
  507 HGRW(J3,ID2)=HGRW(J3,ID2)+(DWDAO(L,I1,I2)/(2.*DT)-DXDDAO(J2*2)*
     $SQDE/DT)*W(L,I1)
  506 HGRW(ID2,J3)=HGRW(J3,ID2)
      DO 508 I1=1,N
  508 HGRW(ID2,ID2)=HGRW(ID2,ID2)+(SQDE*XD(I1)-W(L,I1)/4.)*W(L,I1)/DT/DT
  509 CONTINUE
      PRINT 105,TIME,(X(I),I=1,6),(XD(I),I=1,6),(F(I),I=1,6)
  105 FORMAT(5H T = ,F10.6,4HX = ,6E15.5,/,6H XD = ,6E20.5,/,5H F = ,
     $ 6E20.5)
      XTARG=XTARG+XTD*DT
      YTARG=YTARG+YTD*DT
    1 CONTINUE
      PRINT 50,(I,DEL(I),I=1,NP)
   50 FORMAT(5H I = ,I3,2X,6HDEL = ,E20.15)
      G=SQRT((NP-1.)*DT)
      DGDT=G/(2.*DT)
      DO 4 J=1,N
      DO 4 I=1,M
    4 A1(J*M-M+I)=A(J,I)
      A1(ID2)=DT
      DO 67 I=1,N2
      A1(N*M+I)=ADO(I+2)
   67 CONTINUE
      COEF=COEF0
      DO 149 I=1,ID2
```

```
149 A1O(I)=A1(I)
    DO 5 J=1,N
    DO 5 L=1,NP
  5 W1(J*NP-NP+L)=W(L,J)
    W1(ID1)=G
    DO 6 J=1,N
    DO 6 L=1,NP
    DO 7 K=1,N
    DO 7 I=1,M
  7 DW1DA1(J*NP-NP+L,K*M-M+I)=DWDA(L,J,K,I)
    DO 68 K=1,N2
 68 DW1DA1(J*NP-NP+L,N*M+K)=DWDAO(L,J,K)
  6 DW1DA1(J*NP-NP+L,ID2  )=DWDDT(L,J)
    DW1DA1(ID1,ID2)=DGDT
    DO 9 I=1,ID2
    DO 8 J=1,ID2
    DO 8 K=1,ID1
  8 HGRW(I,J)=HGRW(I,J)+DW1DA1(K,I)*DW1DA1(K,J)
    HDMC(I)=0.
    DO 9 K=1,ID1
  9 HDMC(I)=HDMC(I)-DW1DA1(K,I)*W1(K)
    CV=0.
    DO 12 I=1,ID1
 12 CV=CV+W1(I)**2
    IT=0
    CVO=CV
    PRINT 102,ILOOP,CVO
102 FORMAT(5H I = ,I5,2X,E20.15)
    DO 34 I=1,ID2
 34 HGRW(I,I)=HGRW(I,I)+DIAG
    GO TO 403
    DO 402 I=1,ID2
    DO 402 J=1,ID2
402 HGRW1(I,J)=HGRW(I,J)
    CALL TRED1(ID2,ID2,HGRW1,D,E,E2)
    CALL IMTQL1(ID2,D,E,IERR)
    PRINT 400,IERR,(D(I),I=1,ID2)
400 FORMAT(7HIER = ,I5,4HD = ,100(T18,E20.15,/))
403 CONTINUE
    CALL RDLU(HGRW,HGRW1,HDMC,HDMD,ID2 ,0,0)
    GO TO 76
 75 DO 74 I=1,ID2
 74 HDMD(I)=HDMC(I)
 76 CONTINUE
    DO 10 I=1,ID2
    A1(I)=A1(I)+COEF*HDMD(I)
 10 CONTINUE
    DO 11 J=1,N
    DO 11 I=1,M
 11 A(J,I)=A1(J*M-M+I)
    DO 69 I=1,N2
    ADO(I+2)=A1(N*M+I)
```

```
 69 CONTINUE
    DT=A1(ID2)
    IF(DT.LT.0.)DT=.01
    PRINT 103,((A(I,J),J=1,M),I=1,N)
103 FORMAT(6(1H ,6E20.15,/,T3,6E20.15,/),//)
    CV=0.
    DO 151 L=1,NP
    SQDE=SQRT(DT/EPS)
    CALL FCT(X,XD,A,A0,A00,DXDA,DXDDA,DXDA0,DXDDA0,DXDT,DXDDT,L,N,M,NP
   $,DT,ALPHA,AOF)
    CALL DFCT(F,X,XD,DFDX,L,N,DT,DEL,NP,XTD,YTD)
    DO 151 J=1,N
    W(L,J)=(XD(J)-F(J))*SQDE
    CV=CV+W(L,J)**2
151 CONTINUE
    G=SQRT((NP-1.)*DT)
    CV=CV+G*G
    GO TO 153
    IF(CV.LT.CV0)GO TO 153
    COEF=COEF/4.
    DO 158 I=1,ID2
158 A1(I)=A10(I)
    IT=IT+1
    IF(IT.GT.5)GO TO 153
    GO TO 76
153 CONTINUE
    PRINT 104,(A0(I),I=1,N)
    PRINT 104,(A00(I),I=1,N)
104 FORMAT(1H ,6E20.15)
    DE=CV-G*G
    TF=(NP-1.)*DT
    ILOOP=ILOOP+1
    PRINT 100,ILOOP,TF,CV,DE
100 FORMAT(5H I = ,I4,2X,5HTF = ,E15.5,2X,5HCV = ,E15.5,2X,5HDE = ,
   $E15.5)
    DO 160 I=1,ID2
160 A1S(I,ILOOP)=A1(I)
    IF(ABS(CV0-CV).LT..0001)GO TO 420
    IF(ILOOP.LT.15)GO TO 20
420 ILOP2=ILOOP-2
165 FORMAT(7H A1P = ,(T8,E15.10))
    TPP=.01
    T=0.
    DT=.001
    DO 120 I=1,N
120 X(I)=A0(I)
    L=NP+1
    LIM=TF/.001
    XPTG(1)=X(1)+XTARG
    YPTG(1)=X(2)+YTARG
    TTEST=TF/(NP-1.)
    TTESTD=TTEST
```

```
      PRINT 123,T,(X(J),J=1,N)
      DO 121 I=1,LIM
      IF(T.GT.TTEST)L=L+1
      IF(T.GT.TTEST)XPTG(L-NP)=X(1)+XTARG
      IF(T.GT.TTEST)YPTG(L-NP)=X(2)+YTARG
      IF(T.GT.TTEST)TTEST=TTEST+TTESTD
      CALL DFCT(F,X,XD,DFDX,L,N,DT,DEL,NP,XTD,YTD)
      DO 122 J=1,N
  122 X(J)=X(J)+DT*F(J)
      T=T+DT
      IF(T.LT.TPP)GO TO 121
      TPP=TPP+.01
      PRINT 123,T,(X(J),J=1,N)
  123 FORMAT(5H T = ,F10.3,6E15.10)
  121 CONTINUE
      XPTG(NP)=X(1)+XTARG
      YPTG(NP)=X(2)+YTARG
      PRINT 123,T,(X(J),J=1,6)
      CALL OPNPLT
      CALL XYEPLT
      CALL LINPLT(XPLT,YPLT,NP)
      CALL CURVE(XPTG,YPTG,NP,0)
      CALL ENDPL(1)
   91 CONTINUE
      GO TO 21
  230 CONTINUE
      STOP
      END
```

CALLIN

```
      COMPILER (XM=1)
      SUBROUTINE CALLIN
      PARAMETER NP=1801,NP1=NP-1,NL=180
      COMMON/CALL/TF
      DIMENSION O(NL),V(NL)
      COMMON XF(6),XI(6),XPL(6,NL),A(6,6),B(6),X(6),G(6,6),P(6),R(6,6),
     $RD(6,6),XD(6),U(NP),RS(6,6,NP),XDIF(6,NP),XE(6),XDFS(NP)
      CALL COEFC(C1,C2,C3,C4,C5,C6,C7,C8,C9)
      X3=67.56
   50 READ 1,(X(I),I=1,6),UP,UIND
    1 FORMAT(8F10.5)
      READ 1,(XF(I),I=1,6),TF
      UP=UP*1000.
      PRINT 33,UIND
   33 FORMAT('  UIND = ',F10.5,//)
      PRINT 20,(X(I),I=1,6),(XF(I),I=1,6),UP
   20 FORMAT(5H X = ,6(E15.10,5X),/,6H XF = ,6(E15.10,5X),/,6H UP = ,
     $F10.3,//)
      DO 13 I=1,6
   13 XI(I)=X(I)
      X6=ATAN2(XF(2)-X(2),XF(1)-X(1))
      A(1,3)=COS(X6)
      A(2,3)=SIN(X6)
      A(3,3)=2.*C1*X3
      A(1,4)=-A(2,3)
      A(2,4)=A(1,3)
      A(4,4)=C4*X3
      A(5,4)=C7*X3
      A(4,5)=C5*X3
      A(5,5)=C8*X3
      A(6,5)=1.
      A(1,6)=-X3*A(2,3)
      A(2,6)=X3*A(1,3)
      B(4)=C6*X3*X3
      B(5)=C9*X3*X3
      X(1)=0.
      X(2)=0.
      X(3)=X(3)-X3
      X(6)=X(6)-X6
      T=0.
      XDFI=0.
      DO 22 I=1,2
      G(I,I)=100.
      RS(I,I,NP)=-100.
   22 CONTINUE
      IPL=0
      PRINT 26,C1,C2,C3,C4,C5,C6,C7,C8,C9
   26 FORMAT(5H C = ,9(E13.8,1X)  )
      PRINT 25,((A(I,J),J=1,6),I=1,6),(B(I),I=1,6)
```

```
25 FORMAT(5H A = ,6(T6,6(E15.10,5X),/),//,5H B = ,6(E15.10,5X),//)
   H=TF/(NP-1.)
   IP=0
   DO 2 II=1,NP1
   K=NP-II
   M=K+1
   DO 4 I=1,6
   DO 4 J=1,6
 4 R(I,J)=RS(I,J,M)
   CALL FCEV(A,B,UP,RD,R)
   DO 3 I=1,6
   DO 3 J=1,6
 3 RS(I,J,K)=RS(I,J,M)-H*RD(I,J)
   IP=IP+1
   IF(IP.LT.50)GO TO 2
34 CONTINUE
   PRINT 21,II,K,((RS(I,J,K),J=1,6),I=1,6),((RD(I,J),J=1,6),I=1,6)
21 FORMAT(6H II = ,I5,5H K = ,I5,/,6(6(5X,E15.10),/),//,6(6(5X,E15.10
  5),/))
   IP=0
 2 CONTINUE
   IF(UIND.GT.1.)GO TO 32
   DO 31 K=2,NP
   DO 31 I=1,6
   DO 31 J=1,6
31 RS(I,J,K)=RS(I,J,1)
32 CONTINUE
   IP=0
   DO 5 I=1,NP1
   II=I+1
   U(I)=0.
   DO 6 J=1,6
 6 U(I)=U(I)+X(J)*(RS(4,J,I)*B(4)+RS(5,J,I)*B(5))/UP
   DO 7 J=1,6
   XD(J)=B(J)*U(I)
   DO 7 K=1,6
 7 XD(J)=XD(J)+A(J,K)*X(K)
   UC=U(I)
    DO 19 J=1,6
19 P(J)=X(J)+XI(J)
   CALL FC(P,XE,C1,C2,C3,C4,C5,C6,C7,C8,C9,UC)
   DO 28 J=1,6
28 XDIF(J,I)=XE(J)-XD(J)
   XDFS(I)=0.
   DO 29 J=1,6
29 XDFS(I)=XDFS(I)+XDIF(J,I)**2
   XDFI=XDFI+H*XDFS(I)
   DO 8 J=1,6
 8 X(J)=X(J)+H*XD(J)
   IP=IP+1
   T=T+H
   IF(IP.EQ.10)GO TO 9
```

```
      GO TO 5
    9 IPL=IPL+1
      PRINT 10,U(I),(X(J),J=1,6),I
      DO 11 J=1,6
   11 XPL(J,IPL)=X(J)+XI(J)+(XF(J)-XI(J))*T/TF
      PRINT 18,T,(XPL(J,IPL),J=1,6)
   18 FORMAT(8H TIME = ,E10.5,6(2X,E15.10))
      IP=0
   10 FORMAT(6H U = ,E15.10,3X,6(E15.10,1X),I5)
    5 CONTINUE
      UINT=0.
      DO 27 I=1,NP1
   27 UINT=UINT+U(I)*H
      PRINT 30,UINT,XDF1,(I,XDFS(I),(XDIF(J,I),J=1,6),I=1,NP1,10)
   30 FORMAT(8H UINT = ,E15.10,8H XDF1 = ,E15.10,/,8H XDIF = ,/,(1X,I8,
     $1X,7(E15.10,2X)))
      IP=0
      IPL=0
      DO 12 I=1,NL
      O(I)=XPL(1,I)
   12 V(I)=XPL(2,I)
      CALL OPNPLT
      CALL XYEPLT
      CALL LINPLT(O,V,NL)
      DO 14 I=1,6
   14 X(I)=XI(I)
      DO 15 I=1,NP1
      UC=U(I)
      CALL FC(X,XD,C1,C2,C3,C4,C5,C6,C7,C8,C9,UC)
      DO 16 J=1,6
   16 X(J)=X(J)+H*XD(J)
      IP=IP+1
      IF(IP.EQ.10)GO TO 17
      GO TO 15
   17 IPL=IPL+1
      PRINT 70,U(I),(X(J),J=1,6),I
      O(IPL)=X(1)
      V(IPL)=X(2)
      IP=0
   15 CONTINUE
      CALL CURVE(O,V,NL,0)
      CALL ENDPL(1)
      RETURN
      END
```