January, 1968 Report ESL-R-333 Copy_____

OPTIMAL INTERCEPT GUIDANCE FOR MULTIPLE TARGET SETS

by

Robert J. Norbutas

The preparation and publication of this report, including the research on which it is based, was sponsored by the Naval Ordnance Systems Command, Department of the Navy, under Contract No. NOW-66-0178-d, M.I.T. DSR Project No. 76094. This report is published for technical information only and does not represent recommendations or conclusions by the sponsoring agency. Reproduction in whole or in part is permitted for any purpose of the United States Government.

Approved by: George C. Newton, Jr.
George C. Newton, Jr.
Associate Director

Electronic Systems Laboratory
Department of Electrical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

The problem of optimally guiding a vehicle to intercept more than one target is investigated. The major contributions are the following: (a) the extension of the variational calculus and two numerical algorithms (steepest-descent and Newton-Raphson) to multiple target set problems; and (b) the design of a suboptimal feedback controller for a specific problem of a vehicle intercepting two targets.

The problems considered are in the form of N-point ($N > 2$) optimal control problems. Application of the calculus of variations results in a set of (N-1) two-point boundary value problems coupled through their boundary conditions. The additional boundary conditions are sets of intermediate transversality conditions in terms of discontinuities in the costate and Hamiltonian that are of the same form as the terminal transversality conditions.

The steepest-descent and Newton-Raphson algorithms are extended to handle N-point optimal control problems. The modification of the steepest-descent algorithm involves the computation of an additional influence function for each intermediate state constraint, thereby increasing the computation time required per iteration proportionately. The Newton-Raphson algorithm is found to be inferior to the steepest-descent algorithm for computing optimal two target intercept trajectories because of the difficulty with which it handles free-time problems.

Optimal intercept trajectories are computed for a particular two target missile guidance problem. A minimum control effort suboptimal controller is developed for this problem by approximating the system by a double integrator model. The turning-rate control for the missile is computed as a function of the optimal control for the double integrator system. The optimal control for the model is obtained analytically in the form of a feedback control law based on an assumed future target motion. Against straight-running targets for which the approximate model is valid, the performance of the suboptimal control law is within five percent of the optimal. Against turning targets its performance is very nearly optimal with respect to the assumed form of the future target motion. In situations where the approximate model is not valid the control law is augmented, and the deviation from the optimal for a typical trajectory is 15 percent.

ACKNOWLEDGEMENT

The material presented in this report is based on a thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Electrical Engineering Department.

This work was supported by the Naval Ordnance Systems Command, Department of the Navy, under Contract No. NOW-66-0178-d, M.I.T. DSR Project No. 76094.

The author wishes to acknowledge his appreciation to Professor George C. Newton, Jr., for supervising this thesis and to Professors Michael Athans and Leonard Gould for serving on the thesis committee. Special thanks are due to Dr. Richard W. Bush of the Electronic Systems Laboratory, M.I.T., for his many helpful suggestions during the course of this research effort.

Thanks are owed to the personnel at the M.I.T. Computation Center for their assistance in obtaining the computational results of this research.

Finally the efforts of the Drafting and Publications Staffs of the Electronic Systems Laboratory are gratefully acknowledged.

CONTENTS

CONTENTS (Contd.)

CONTENTS (Contd.)

LIST OF FIGURES

LIST OF FIGURES (Contd.)

LIST OF TABLES

CHAPTER I

INTRODUCTION AND SUMMARY

1.1 INTRODUCTION AND BACKGROUND

The results of an investigation of the problem of guiding a vehicle to intercept more than one target are summarized in this chapter; the details are presented in the remainder of the report. The engineering problem is formulated as an optimal control problem, i.e., one in which a mathematical expression is used for the performance criterion, and the optimal solution is obtained. Because of the complexity involved in implementing an optimal controller, a suboptimal one based on an approximate mathematical model is developed. The major contributions of the work are: (a) the extension of the variational calculus and two numerical algorithms (steepest-descent and Newton-Raphson) to multiple target set problems; and (b) the design of a suboptimal feedback controller for a specific problem of a vehicle intercepting two targets.

Although this research concentrates on a simple deterministic two-target intercept problem, most of the theoretical work is applicable to more complicated multiple target set problems. An optimal control theory approach was selected because, with a minimum control effort choice for the cost functional, it yields the well-known proportional control law for the single-target intercept problem.

Some examples in which the problem of guiding a vehicle to intercept more than one target occurs are a missile attacking a target employing effective decoys, an airplane bombing several targets, and a booster rocket injecting several objects into orbit or inspecting several objects in orbit. Similar problems that have been investigated by others include the following: systems with inequality state constraints by Berkovitz,[11]* Dreyfus,[12,13] Chang,[14] and Bryson and others;[15] booster staging problems by Mason, Dickerson, and Smith;[16] and "hybrid-state" systems by Witsenhausen.[17] All these problems involve sets of intermediate equality state constraints, and their optimal solutions are characterized by

---

* Superscripts refer to numbered items in the Bibliography.

discontinuities in the costate vector and the Hamiltonian at the intermediate times.

In this chapter, the variational theory associated with multiple target set optimal control problems is presented first. Then two numerical methods for solving the resulting N-point boundary value problems are presented. Finally, optimal and suboptimal controllers for the specific problem of interest are discussed.

1.2 THE NECESSARY CONDITIONS VIA VARIATIONAL CALCULUS

The necessary conditions on the optimal control for an N-point optimization problem are determined by extending the conventional calculus of variations techniques. The Euler-Lagrange equations[3] to be satisfied throughout the intervals between the N-points at which the states are constrained are identical to those for the two-point optimization problems. The equations specifying the boundary conditions at the N-points are of the same general form at all N-points as is shown below.

Consider a system which can be mathematically modeled by a set of n first-order ordinary differential equations of the form

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

It is desired to determine the control vector $\bar{u}(t)$ (assuming, of course, that such an optimal control exists) over the interval $t_0 \leq t \leq t_{N_f}$ such that a cost functional of the form

$$J(u(t)) = \sum_{i=0}^{N_f} \phi^i(x(t_i),t_i) + \int_{t_0}^{t_{N_f}} L(x(t),u(t),t)\,dt \tag{1.2}$$

is minimized subject to Eq. 1.1 and to constraints on the state of the form

$$\psi^i(x(t_i),t_i) = 0\,, \qquad i = 0,1,\ldots,N_f \tag{1.3}$$

It is assumed that no bounds are placed on either x or u, and that $\bar{x}(t)$ and $\bar{u}(t)$ are restricted to continuous and piecewise continuous functions of time, respectively. In addition, the usual restrictions[3] are imposed on the continuity and differentiability of $f(x,u,t)$, $L(x,u,t)$,

$\phi^i(x,t)$ and $\psi^i(x,t)$ in order to insure the existence of a solution of Eq. 1.1 and to insure that Eqs. 1.1 - 1.3 can be expanded in a Taylor series about the optimal solution (this is fundamental to the calculus of variations)[1,2]

Following the approach used in the development of the Weierstrass-Erdmann corner conditions,[3] the integral in Eq. 1.2 is written as the sum of integrals over the subintervals of (t_0, t_{N_f}) defined by the t_i. The necessary conditions to be satisfied by the optimal control and corresponding state and costate vectors in each subinterval are found to be the system differential equations and the Euler-Lagrange equations given by*

$$\frac{dx^i}{dt} = f(x^i(t), u^i(t), t) \tag{1.4}$$

$$\frac{\partial H}{\partial u^i}(x^i, u^i, \lambda^i, t) = 0 \tag{1.5}$$

$$\frac{d\lambda^i}{dt} = -\frac{\partial H}{\partial x}(x^i, u^i, \lambda^i, t) \tag{1.6}$$

where $\lambda(t)$ is the Lagrange multiplier,[4] or "costate" vector, and H is the "Hamiltonian" function defined by

$$H[x,u,\lambda,t] = L[x,u,t] + \lambda^T \cdot f(x,u,t) \tag{1.7}$$

The optimal $\bar{x}(t)$ and $\bar{\lambda}(t)$ vector functions of time are subject to boundary conditions at $(N_f + 1)$ points in time given by**

$$\psi^i(\bar{x}(t_i), t_i) = 0 \tag{1.8}$$

$$\Delta\bar{\lambda}(t_i) + \frac{\partial \Phi^i}{\partial x}^T(\bar{x}(t_i), t_i) = 0 \tag{1.9}$$

---

* The superscripts "i" denote the optimal solution (x,u,λ) over the subinterval $t_{i-1} < t < t_i$.

** For consistency in the notation we arbitrarily define

$$\bar{\lambda}(t_0^-) = \bar{\lambda}(t_{N_f}^+) = \bar{H}(t_0^-) = \bar{H}(t_{N_f}^+) \triangleq 0$$

and the boundary conditions on the Hamiltonian at points of free time are

$$\Delta\bar{H}(t_i) - \frac{\partial \Phi^i}{\partial t}(\bar{x}(t_i), t_i) = 0 \tag{1.10}$$

for $i = 0, 1, \ldots, N_f$, where

$$\Phi^i(x(t), t) = \phi^i(x(t), t) + \upsilon^{i^T} \cdot \psi^i(x(t), t) \tag{1.11}$$

The υ^i are unspecified vectors which are Lagrange multipliers for the state equality constraints at t_i.

The N-point optimal control problem has been subdivided into (N-1) two-point boundary value problems (BVP) which are coupled through their boundary conditions. Since each subproblem requires n boundary conditions at each terminus on x and/or λ, a total of 2n conditions are required at each intermediate time t_i. n of these are supplied by Eqs. 1.8 and 1.9 and the remaining n by the continuity of state. An additional N conditions are required to specify the t_i -- these are supplied by Eq. 1.8 and/or Eq. 1.10.

1.3 A SIMPLE TWO-TARGET EXAMPLE

A double integrator two-target, fixed-time optimal intercept problem demonstrates the application of the necessary conditions. Its solution is in the form of a linear time-varying feedback control law which is used in Section 1.6 in the design of a suboptimal feedback controller for a more complicated problem. The system equations are

$$\begin{aligned} \dot{x}_1 &= x_2 \qquad \text{a)} \\ \dot{x}_2 &= u \qquad \text{b)} \end{aligned} \tag{1.12}$$

and the cost criterion is a quadratic measure of the control effort,

$$J = \frac{1}{2}\int_{t_0}^{t_2} (u(t))^2 \, dt \tag{1.13}$$

The objective is to take the system from any given initial state at time t_0 to the specified positions of x_{11} and zero at times t_1 and t_2, respectively, while minimizing the control effort required.

The control $u(t)$ can be eliminated from the Euler-Lagrange equations giving a set of four first-order differential equations, which can be solved explicitly by using the following boundary conditions:

$$\begin{aligned} &\Delta\lambda_1(t_1) = \upsilon_1 && x_1(t_1) = x_{11} && x_1(t_2) = 0 \\ &\Delta\lambda_2(t_1) = 0 && \Delta x_1(t_1) = 0 && \lambda_1(t_2) = \upsilon_2 \\ & && \Delta x_2(t_1) = 0 && \lambda_2(t_2) = 0 \end{aligned} \qquad (1.14)$$

where υ_1 and υ_2 are unspecified constants. Because the initial states and time t_0 are arbitrary, the solution is in the form of a feedback control law.

For $t_0 < t < t_1$ the equation for the optimal feedback control law is given by

$$u(t) = -\frac{6\cdot(3\tau_1+2\tau_2)}{\tau_1^2\cdot(3\tau_1+4\tau_2)}\cdot x_1(t) - \frac{12\cdot(\tau_1+\tau_2)}{\tau_1\cdot(3\tau_1+4\tau_2)}\cdot x_2(t) + \frac{6\,(\tau_1+\tau_2)(\tau_1+2\tau_2)}{\tau_2\cdot\tau_1^2\cdot(3\tau_1+4\tau_2)}\cdot x_{11} \qquad (1.15)$$

where $\tau_1 = t_1 - t$ and $\tau_2 = t_2 - t_1$.

For $t > t_1$, the control law is given by

$$u(t) = -\frac{3}{(t_2-t)^2}\cdot x_1(t) - \frac{3}{(t_2-t)}\cdot x_2(t) \qquad (1.16)$$

which is the optimal single target intercept control law. It is interesting to note that as τ_1 becomes smaller than τ_2, the form of the control law given by Eq. 1.15 approaches the optimal single-target intercept control law, and the effect of the state constraint at t_2 is diminished.

Figure 4.1 illustrates the optimal solution for initial, intermediate, and terminal conditions given by

$$x_1(t_0) = 1 \qquad x_1(t_1) = 0 \qquad x_1(t_2) = 0$$
$$x_2(t_0) = 0 \qquad t_1 = 0.8 \qquad t_2 = 1.0 \qquad (1.17)$$
$$t_0 = 0$$

Also illustrated, for the purpose of comparison, is the "sequential optimal" solution, i.e., the solution obtained by considering the targets one at a time. For this particular example, the cost for the sequential solution is 308 percent higher than the minimum.

1.4 NUMERICAL SOLUTIONS OF N-POINT BOUNDARY VALUE PROBLEMS

Numerous computational algorithms are available for the solution of the two-point boundary value problems that occur in optimal control problems. Two of the more popular of these, the steepest-descent and Newton-Raphson algorithms, are used in the present work to solve a particular three-point optimal control problem. The results obtained using the modified algorithms on the problem of interest are described in the next section.

The steepest-descent algorithm is well documented in the literature.[23] Its highlights are presented here along with the extension required to handle N-point optimal control problems. At each iteration Eq. 1.1 is integrated forward in time from the initial to the terminal time using a nominal control history to obtain a nominal state trajectory. If any of the t_i are not specified explicitly, one component of the corresponding constraint vector is used as a transition condition (or a stopping condition in the case of the terminal constraint), $\theta^i(x,t)$, in order to define a nominal t_i, i.e.,

$$\theta^i(x(t_i),t_i) = \psi^i_j(x(t_i),t_i) = 0 \qquad (1.18)$$

for some j. This reduces by one the number of constraints to be satisfied at t_i.* Next, Eqs. 1.1 - 1.3 are linearized about the nominal trajectory. The perturbations in control can then be related to perturbations

* For the remainder of this discussion the ψ^i referred to are the reduced constraint vectors.

in the cost and the constraint violations by

$$dJ = \int_{t_0}^{t_{N_f}} H_u(t)\ \delta u(t) \cdot dt \tag{1.19}$$

$$d\psi^i = \int_{t_0}^{t_i} \Lambda^i(t) \cdot f_u(t) \cdot \delta u(t) \cdot dt \tag{1.20}$$

where the subscripts denote partial derivatives evaluated along the nominal trajectory. $\lambda(t)$ in Eq. 1.7 is determined by integrating Eq. 1.6, evaluated along the nominal trajectory, backward in time subject to boundary conditions at each of the t_i given by

$$-\Delta\lambda(t_i) = \left[\phi_x^i - \left(\frac{\phi_x^i \cdot f + \phi_t + L}{\theta_x^i \cdot f + \theta_t^i}\right) \cdot \theta_x^i\right]^T_{t\,=\,t_i} \tag{1.21}$$

The $\Lambda^i(t)$ in Eq. 1.20 are called influence functions since they specify the effect of a control perturbation on the constraint violation ψ^i. Each $\Lambda^i(t)$ is determined by integrating the homogeneous part of Eq. 1.6, evaluated along the nominal trajectory, backward in time subject to a boundary condition at t_i given by

$$\Lambda^i(t_i) = \left[\psi_x - \left(\frac{\psi_x \cdot f + \psi_t}{\theta_x \cdot f + \theta_t}\right) \cdot \theta_x\right]^T_{t\,=\,t_i} \tag{1.22}$$

Furthermore, $\Lambda^i(t) \triangleq 0$ for $t > t_i$ since control perturbations after t_i do not affect the constraint violation at t_i.

To insure that the linearized analysis is not invalidated, the size of the control perturbation is constrained by

$$S^2 = \frac{1}{2} \int_{t_0}^{t_{N_f}} \delta u^T(t) \cdot W^{-1}(t) \cdot \delta u(t) \cdot dt \tag{1.23}$$

where $W^{-1}(t)$ is an arbitrary positive definite weighting matrix, and S^2 is the step size. A new optimization problem can now be formulated by requiring that the control perturbation maximize the change in cost, Eq. 1.19, subject to the subsidisary integral constraints given by Eqs. 1.20 and 1.23. This optimum perturbation is given by

$$\delta u(t) = K_J \cdot W(t) \cdot (H_u^T(t) - G^T(t) \cdot \Lambda(t) \cdot I_{\psi\psi}^{-1} \cdot I_{\psi J}) + W(t) \cdot G^T(t) \cdot \Lambda(t) \cdot I_{\psi\psi}^{-1} \cdot d\psi \tag{1.24}$$

where

K_J is a gain chosen to satisfy the step-size constraint,

$G(t)$ is the matrix $f_u(t)$ evaluated along the nominal solution

$\Lambda(t)$ is a matrix composed of the $\Lambda^i(t)$,

$d\psi$ is the constant vector of the specified reductions in the constraint violations at all of the t_i, and

and $I_{\psi\psi}$, $I_{\psi J}$ are integrals of the influence functions and the Hamiltonian

defined by

$$I_{\psi\psi} = \int_{t_0}^{t_{N_f}} \Lambda^T(t) \cdot G(t) \cdot W(t) \cdot G^T(t) \cdot \Lambda(t) \cdot dt \tag{1.25}$$

$$I_{\psi J} = \int_{t_0}^{t_{N_f}} \Lambda^T(t) \cdot G(t) \cdot W(t) \cdot H_u^T(t) \cdot dt \tag{1.26}$$

The second term in Eq. 1.24 is orthogonal to the state constraint surface and results only (to first-order terms) in a decrease in the constraint violation. The first term (to first-order terms) has no effect on the constraint violations. The gain K_J can be determined as a function of the step-size, S^2, by substituting Eq. 1.24 into Eq. 1.23. As the optimal solution is approached, one can speed up the convergence by choosing K_J to achieve the greatest possible decrease in cost in the current gradient direction. To accomplish this one must apply a small

control perturbation, with an arbitrary K_J, in order to gain second-order information on the dependence of dJ on K_J.

The application of the method of steepest-descent to N-point optimal control problems requires the expansion of the constraint violation vector, ψ, and the influence function matrix, $\Lambda(t)$, to include the added state constraints. The number of variables to be integrated, I, and stored, S, at each iteration in these problems are

$$I = \frac{1}{2}(k+2)(2n+k+1) \qquad a)$$

$$S = m(k+2) + n \qquad b) \qquad (1.27)$$

$$k \triangleq \sum_{i=1}^{N_f} k_i \qquad c)$$

where k_i is the (reduced) number of constraints at each t_i, m is the dimension of the control vector, and n is the dimension of the state vector.

The Newton-Raphson algorithm was also modified to handle N-point optimal control problems. This algorithm is a second-order indirect method by virtue of a linearization about the first-order necessary conditions. The linearization results in a set of 2n linear differential equations which can be solved at each iteration by means of superposition to satisfy the boundary conditions. Its primary advantage is that it can converge quadratically; however, the initial guess of the nominal solution must be "good" or else the algorithm will diverge. Another disadvantage is that it handles free-time problems by solving a series of fixed-time problems.

For three-point boundary value problems, the algorithm requires the integration and storage of the following numbers of variables:

$$I = 2n(n+1) \qquad a)$$

$$S = n(n+3) \qquad b) \qquad (1.28)$$

These requirements are the same as for two-point BVP's; however, the number of unknown boundary conditions to be determined has increased. For problems with more than one intermediate boundary point the number of integrations required also increases significantly.

In free-time problems a Newton method can be used to determine the unknown intermediate and terminal times. First order information on the dependence of cost upon the free times is available from the left-hand side of the transversality conditions, Eq. 1.10, which, in general, is not satisfied for each fixed-time problem. Second-order information can be obtained by perturbing each of the free times. Therefore, each iteration on the free times requires the solution of N additional free-time problems.

1.5 COMPUTATION OF OPTIMAL TWO-TARGET INTERCEPT TRAJECTORIES

Results obtained by applying the numerical algorithms discussed in the preceding section to a particular deterministic missile guidance problem are presented in this section. The problem consists of a missile moving with a constant velocity (normalized to unity) on a two-dimensional playing field. The equations describing this motion are:

$$\dot{x}_1 = \cos x_3 \qquad \text{a)}$$

$$\dot{x}_2 = \sin x_3 \qquad \text{b)} \qquad (1.29)$$

$$\dot{x}_3 = u \qquad \text{c)}$$

where x_1 and x_2 are position coordinates, x_3 is the yaw angle, and u is the turning-rate control. The objective is to guide the missile to intercept two targets and minimize the integral square control effort, Eq. 1.13. It is assumed that the trajectories of the two targets, which are denoted y and z, are known a priori.

The two algorithms discussed in the preceding section were programmed and applied to this problem for several different target motions. The programs used second-order Runga-Kutta integration with an integration step size of approximately 0.05. The steepest-descent (S-D) algorithm required the integration of twelve variables and the Newton-Raphson (N-R) algorithm required twenty. On trajectories 200 steps long, the computation times required per iteration on an IBM 360/65 were 1.5 and 2.5 seconds, respectively.

Figure 4.2 shows the optimal control and missile trajectory for a representative example in which

$$y(t) = \begin{bmatrix} 2 + \frac{1}{2}t \\ 4 \\ 0 \end{bmatrix} \quad , \quad z(t) = \begin{bmatrix} 4 + \frac{1}{2}t \\ 4 \\ 0 \end{bmatrix} \tag{1.30}$$

i.e., the targets move in colinear paths at a velocity of 1/2. The solutions by the two algorithms agreed to an accuracy of four significant figures for this example. The S-D algorithm converged to the solution within ten iterations after starting with a nominal control equal to zero. The N-R algorithm would not converge for this and numerous other starting conditions. Convergence was finally obtained by using an approximation based on the results of the S-D algorithm. The N-R algorithm then converged rapidly (two to four iterations) for each fixed-time problem but required three to four iterations on the free intercept times. Thus, typically, twenty to thirty iterations through the algorithm were required to obtain the optimal solution.

Figure 4.3 illustrates the optimal control and optimal trajectory for another example. The target motion is given by

$$y(t) = \begin{bmatrix} 2.5 \\ 0.84\,t \\ \pi/2 \end{bmatrix} \quad , \quad z(t) = \begin{bmatrix} 2.5 + 0.42\,t \\ 0.42\,t \\ \pi/4 \end{bmatrix} \tag{1.31}$$

The missile is originally on an intercept course with the y target which is moving at a velocity of 0.84. At $t = 0$ the z target emanates from the first target at a velocity of 0.60. The optimal control requires that the missile initially turn away from target y, but by doing so results in a cost J that is 42 percent lower than the cost of optimally attacking the two targets sequentially.

The two major conclusions derived from working with the two numerical algorithms are the following:

1. The S-D algorithm is better suited to this two-target problem than the N-R algorithm because of the relative ease with which it accommodates free-time problems, and also because of its ability to converge from poorer initial guesses.

2. When attacking two targets and attempting to minimize an integral square control effort criterion, significant improvement can be realized by using a two-target control law rather than attacking the targets in sequence.

1.6 SUBOPTIMAL TWO-TARGET INTERCEPT GUIDANCE

In many practical problems of the type being considered, a feedback controller is required because the future motion of the targets is not known with certainty or is subject to change arbitrarily. Closed form solutions for optimal feedback controllers can be obtained for only a small class of problems. To implement an optimal feedback controller in other problems, one must continually update the optimal control law or else use a perturbation type control about the optimal while updating the optimal at a slower rate. Such approaches require that a rather large computational facility be available to perform the update calculations. The computational requirements can often be reduced by making appropriate approximations to simplify the problem, resulting in a mathematically suboptimal controller. The design and evaluation of one such suboptimal controller are described in this section.

The approach taken here is the familiar one of linearization and decoupling. The system equations are replaced by a set of approximate linear equations for which an optimal feedback control law can be obtained. The target and missile state variable estimators and predictors are assumed to be available and decoupled from the feedback control problem. The predicted target trajectories and a nominal two-segment straight-line intercept trajectory, as illustrated by $\hat{\mathbf{x}}$ in Fig. 4.5, are used to determine a set of nominal intercept points. A new coordinate system, $x_1'-x_2'$, is defined. The x_2' axis passes through the second intercept point and intersects the two segments of $\hat{\mathbf{x}}$ at equal angles. With this choice, the x_2' component of velocity of a missile traversing the nominal trajectory is a constant.

An approximate model for the system is now obtained by considering the actual missile velocity in the x_2' direction to be a constant, and defining the two new state variables to be the distance from the missile to the x_2' axis and its derivative, respectively. The control, v, for

the resulting double integrator system is defined to be the acceleration of the missile in the x_1' direction. The v which minimizes the control effort is given in Section 1.3 in the form of a feedback control law, Eqs. 1.15 and 1.16. The quantities required to compute v (the two nominal intercept times and the distance from the first nominal intercept point to the x_2' axis, x_{11}) are easily determined from the nominal intercept trajectory. The turning rate control in the actual problem is related to v by

$$u = v/\cos(x_3 - \psi_b) \tag{1.32}$$

where $(x_3 - \psi_b)$ is the angle between the missile velocity vector and the x_2' axis.

Equations 1.15, 1.16, and 1.32 constitute a suboptimal feedback control law for the two-target intercept problem of interest. It obviously encounters difficulty when the magnitude of the angle $(x_3 - \psi_b)$ equals $\pi/2$ radians, in which case it must be augmented. The proposed augmentation is as follows: (1) if $x_3 - \psi_b$ exceeds $\pi/2$, the missile is to turn at a specified constant turning rate in order to decrease the angle; (2) if this angle exceeds $(\pi/2 - \epsilon)$ radians (in the examples discussed below a value of 1.5 radians is used for $(\pi/2 - \epsilon)$) and Eq. 1.32 specifies a control that would increase this angle, then the control required to maintain it at $(\pi/2 - \epsilon)$ is applied.

The suboptimal control law was simulated on an IBM 360/65 and tested against straight running targets. For all of the trajectories computed, the targets are predicted to move at constant velocity and heading from their current positions. The predictions of target motion, the nominal intercept points, and turning rate control are updated at a rate of twenty times per time unit.

For trajectories in which the augmentation is not required, the performance of the controller is very nearly optimal, as demonstrated in Figs. 4.2 and 4.3 in which the suboptimal performance is within 5 percent of the optimum. The performance deteriorates as the augmentation condition is approached and met. This is apparent from the example in Fig. 4.7 in which the z velocity is 0.4 and the control effort required by the suboptimal controller is 15 percent higher than the minimum. It is interesting to note, however, that in many situations where augmentation

is required, the practicality of attacking two targets in this manner is questionable. In this particular example a 38 percent lower cost is achieved by attacking the targets in the reverse order as shown in Fig. 4.8.

The adequacy with which the proposed controller handles maneuvering targets is shown by the example in Fig. 4.4. In this case the targets move on arcs of circles, but the motions assumed by the controller are tangential estimates of the target trajectories. The trajectory estimate update and control law update times are identical and the same as in the previous examples. The cost of the suboptimal solution is considerably higher than the optimal solution, since the latter is based on a priori knowledge of the motion of the targets. In this case the optimal solution is not a meaningful reference for evaluating the suboptimal controller.

It is concluded that a suboptimal feedback control law of the type proposed adequately solves the problem of interest. In a practical design, however, the dependence of the performance upon update times must be investigated; this has not been done.

1.7 COMMENTS AND CONCLUSIONS

The major goal of this research was to investigate the engineering problem of attacking two targets with a single missile. Extensions were made to existing optimal control theory and numerical methods for solving optimal control problems in the course of the research. The final result was a suboptimal feedback control law which appears to satisfactorily solve the problem. When far away from the first target, it takes both targets into account but when near the first target or after intercepting it, the control law is identical to that for a single target. The major practical objection to the control law is that it results in a large control effort when near the first target. This is the result of choosing an integral square control effort performance criterion.

Several problems of academic and/or practical interest have arisen as a result of this research. The extension of the first-order necessary conditions to bounded control problems should be made. Further research is required on numerical techniques for solving N-point optimal control problems -- both on new techniques and on

modifications of the Newton-Raphson algorithm to circumvent the difficulties encountered in making initial guesses and in handling free-time problems. Several aspects of the development of the suboptimal controller should be pursued further. These include the choice of cost functional, the mechanisms for estimating the missile and target states, and the effect of the frequency of updating this information on the missile performance.

CHAPTER II

OPTIMAL CONTROL WITH MULTIPLE TARGET SETS — THE NECESSARY CONDITIONS

2.1 INTRODUCTION

In this chapter the necessary conditions to be satisfied by the solution of an N-point optimal control problem are determined by an extension of the calculus of variations. The optimal control problem is extended from a two-point boundary valve problem (BVP) to a general N-point BVP by the addition of equality constraints and penalty functions on the state of the system at a discrete set of intermediate points in time. Such constraints can arise, for example, in the formulation of a missile guidance problem in the presence of several targets (see Chapter IV).

In Section 2.2 of this chapter the general N-point optimal control problem is rigorously formulated, and the first-order necessary conditions on the optimal control are derived in Section 2.3. The application of these necessary conditions is demonstrated in Section 2.4 by means of a single integrator example in which the value of the state is specified at three points in time, with the intermediate time left free. In Section 2.5 the closed form solution is presented for the general class of problems of linear systems with quadratic cost functionals which include penalty functions on the state at two or more points in time. The chapter is concluded in Section 2.6 by a general discussion of the nature of the necessary conditions.

Berkovitz,[11] Dreyfus,[12,13] Chang,[14] and Bryson and others[15] have investigated the problem of optimal control with inequality state constraints, i.e., problems in which the state of the system is constrained to remain in a given region of the state space throughout the time interval of definition of the problem. Such problems can be subdivided into sections in which the state lies wholly in the interior of the admissible region, and sections in which the state moves along the boundary of the admissible region. At the point where the state enters onto the boundary of the admissible region, called the transition point, certain conditions must be satisfied by the state of the system in order that it can be maintained on the boundary with a finite control. These conditions are in the form of

equality constraints on the state which are obtained by requiring that the higher order time derivatives of the constraint function be zero. Mason, Dickerson and Smith[16] have investigated booster staging problems for which the staging times constitute intermediate boundary points at which certain state constraints must be satisfied. In addition, the dynamics of the system being controlled change as the system passes through the various staging points. Witsenhausen[17] has classified as "hybrid-state systems" those systems containing both discrete-level and continuous-level states, e.g., systems with relays in which a discrete-level state is required to identify the state of each relay. The conditions under which the relays open or close define constraints on the state of the system at the switching (intermediate) times. Note that many systems without relays can be formulated as hybrid-state, e.g., booster staging problems where the state of a set of hypothetical relays can be used to define each stage of the booster. Bellman, Kagiwada, and Kalaba[42,44] have investigated the problems of state identification by making a least squares fit between the output (with measurement noise) of the system at discrete points in time and the corresponding output (without measurement noise) of a system model. Such problems involve penalty functions on the state of the system at intermediate points in time. Another area in which intermediate boundary conditions can arise is in the optimization of certain chemical processes[18] in which changes in inlet gas concentrations lead to changes in the process parameters. While a change in inlet concentration represents a step input, if one specifies the state conditions under which one inlet gas is to be favored over another, the inlet concentration can be made strictly a function of the present state of the system. Such problems can be formulated as hybrid-state systems in which switching from one inlet concentration to another is analogous to the opening or closing of a relay and is determined by appropriate conditions on the state of the system.

2.2 PROBLEM FORMULATION

Consider a system which can be described by a set of n first-order ordinary differential equations of the form

$$\dot{x}_i = f_i[x, u, t] \qquad , \quad i = 1, 2, \ldots, n \tag{2.1}$$

where $x = (x_1, x_2, \ldots, x_n)$ is the n-dimensional state vector and $u = (u_1, \ldots, u_m)$ is the m-vector of control variables ($1 \leq m \leq n$). It is desired to determine the control $\bar{u}(t)$ which yields a minimum of a functional of the form

$$J[u(t), x(t)] = \sum_{i=0}^{N_f} \phi^i [x(t_i), t_i] + \int_{t_0}^{t_{N_f}} L[x(t), u(t), t]\, dt \tag{2.2}$$

in taking the state from an initial to a terminal state, while satisfying a set of initial, terminal, and intermediate state constraint equations given by

$$\psi_j^i [x(t_i), t_i] = 0, \qquad \begin{matrix} j = 1, \ldots, k_i ; \\ i = 0, 1, \ldots, N_f \end{matrix} \tag{2.3}$$

where

$$t_0 < t_1 < \ldots < t_{N_f} \tag{2.4}$$

To insure that a solution of Eq. 2.1 exists, and that the techniques of the classical calculus of variations are applicable, it is necessary to impose the restriction that the functions $f_i(x,u,t)$, $L(x,u,t)$, $\phi^j(x,t)$, and $\psi_\ell^j(x,t)$ ($i=1,\ldots,n, j=0,\ldots,N_f$, and $\ell=1,\ldots,k_i$) be continuous, have continuous first derivatives with respect to each of their arguments, and have second derivatives with respect to each of their arguments which exist for all $x \in R^n$, $u \in \Omega$, and $t \in (t_0, t_{N_f})$. Ω is a subset of R^m which represents the set of admissible controls. For the calculus of variations to be applicable, all admissible values of the control must be interior points of Ω. In practice this is tantamont to assuming that Ω is also unbounded, since if bounds are placed on u, one must usually include the boundary of Ω as admissible. For such problems, one must make use of the Minimum principle.* While the calculus of variations requires that $x, \dot{x}, u$, and $\dot{u}$ be continuous functions of time, these restrictions can be relaxed by application of the Weierstrass-Erdmann corner condition

---

* See Section A.8. The extension to bounded controls is discussed further in Section 2.6.

(see Section A.4), so that only continuity of x is required.

2.3 DERIVATION OF THE NECESSARY CONDITIONS

The calculus of variations solution of the two-point optimal control problem is reviewed in Appendix A. The primary results are the Euler-Lagrange equations, Eqs. A.57 through A.59, and the initial and terminal transversality conditions, Eqs. A.68 and A.69, to be satisfied by the optimal solution (as well as the initial and terminal state constraints, Eqs. A.51 and A.52). In the present section the derivation is extended to the general N-point optimal control problem. In this case one finds that the same Euler-Lagrange equations must be satisfied over each of the subintervals of (t_0, t_f), and that in addition to the initial and terminal transversality conditions, a set of intermediate transversality conditions in terms of discontinuities in the costate and the Hamiltonian, Eq. 2.24, must be satisfied at each of the intermediate boundary times. To simplify the following discussion, let us initially consider a three-point BVP, so that the cost functional is of the form

$$J = \phi^0[x(t_0), t_0] + \phi^1[x(t_1), t_1] + \phi^2[x(t_2), t_2] + \int_{t_0}^{t_2} L[x(t), u(t), t]\, dt \tag{2.5}$$

and the state constraints are of the form

$$\psi^i[x(t_i), t_i] = 0 \quad , \quad i = 0, 1, 2 \tag{2.6}$$

where ψ^i is a k_i-vector.

Since, in general, one does not expect $\dot{x}$ and $\dot{u}$ to be continuous at $t = t_1$, Eq. 2.5 cannot be expanded in a Taylor series at that point; but it can be written as

$$J = \sum_{i=0}^{2} \phi^i[x(t_i), t_i] + \int_{t_0}^{t_1^-} L[x(t), u(t), t]\, dt + \int_{t_1^+}^{t_2} L[x(t), u(t), t]\, dt \tag{2.7}$$

By expressing the cost functional in this form, the expansion can now be made over the two subintervals of (t_0, t_2). After adjoining the differential

constraints, Eq. 2.1, by a set of Lagrange multipliers, $\lambda(t) = (\lambda_1(t), \ldots, \lambda_n(t))$, performing the Taylor series expansion, and integrating by parts to eliminate $\delta \dot{\lambda}(t)$, the first variation of J is found to be

$$\delta J = \sum_{i=0}^{2} \left[\left(\frac{\partial \phi^i}{\partial x} \right)^T \cdot dx \right]_{t=t_i} + \sum_{i=0}^{2} \left[\frac{\partial \phi^i}{\partial t} \cdot dt \right]_{t=t_i} + \left[-\bar{\lambda}^T \cdot dx + \bar{H} \cdot dt \right]_{t_0}^{\bar{t}_1}$$

$$+ \int_{t_0}^{\bar{t}_1} \left[\bar{H}_u^T \cdot \delta u + (\bar{H}_x + \dot{\bar{\lambda}})^T \cdot \delta x + (\bar{f} - \dot{\bar{x}})^T \cdot \delta \lambda \right] dt + \left[-\bar{\lambda}^T \cdot dx + \bar{H} \cdot dt \right]_{t_1^+}^{t_2}$$

$$+ \int_{t_1^+}^{t_2} \left[\bar{H}_u^T \cdot \delta u + (\bar{H}_x + \dot{\bar{\lambda}})^T \cdot \delta x + (\bar{f} - \dot{\bar{x}})^T \cdot \delta \lambda \right] dt \qquad (2.8)$$

where H, the Hamiltonian function, is defined by Eq. 1.7. In order that $\bar{u}(t), \bar{x}(t)$, and $\bar{\lambda}(t)$ constitute an optimal solution, it is necessary that the first variation of J be zero for all admissible variations in x,u, and λ.

For convenience, let us adopt the notation of Chapter I, where the subscript i denotes the optimal solution for $t_{i-1} < t < t_i$. Considering the variations in x, u, and λ over each of the subintervals $(t_0, \bar{t}_1)$ and (t_1^+, t_2) independently, with no perturbations of the boundary points, one obtains the result that the Euler-Lagrange equations must be satisfied over each subinterval, i.e.

$$\frac{dx^i(t)}{dt} = f(x^i(t), u^i(t), t) \qquad (2.9)$$

$$\frac{d\lambda^i(t)}{dt} = -\frac{\partial H}{\partial x}\left[x^i(t), u^i(t), \lambda^i(t), t \right] \qquad (2.10)$$

and

$$\frac{\partial H}{\partial u}\left[x^i(t), u^i(t), \lambda^i(t), t \right] = 0 \qquad (2.11)$$

for $i = 1, 2$. Equation 2.8 now reduces to

$$\delta J = \left[\left(\frac{\partial \phi^0}{\partial x} + \lambda^1(t)\right)^T \cdot dx + \left(\frac{\partial \phi^0}{\partial t} - H^1(t)\right)\cdot dt\right]_{t_0} + \left[\left(\frac{\partial \phi^2}{\partial x} - \lambda^2(t)\right)\cdot dx + \left(\frac{\partial \phi^2}{\partial t} + H^2(t)\right)\cdot dt\right]_{t_2}$$

$$+ \left[\left(\frac{\partial \phi^1}{\partial x} - \lambda^2(t) + \lambda^1(t)\right)\cdot dx + \left(\frac{\partial \phi^1}{\partial t} + H^1(t) - H^2(t)\right)\cdot dt\right]_{t_1} \qquad (2.12)$$

For small perturbations of the boundary conditions, the change in the state constraints is given to first order terms by

$$d\psi^i[\bar{x}(t_i) + \delta x(t_i), t_i + \delta t_i] = \frac{\partial \psi^i}{\partial x}[\bar{x}(t_i), t_i]\cdot dx(t_i) + \frac{\partial \psi^i}{\partial t}[\bar{x}(t_i), t_i]\cdot dt_i = 0, \quad i=0,1,2 \qquad (2.13)$$

Since only those $\delta x(t)$ that satisfy all the state constraints are admissible, $x(t) = \bar{x}(t) + \delta x(t)$ must intersect, at time $\bar{t}_i + \delta t_i$, the tangent plane to ψ^i at $(\bar{x}(\bar{t}_i), \bar{t}_i)$, and therefore the right hand side of Eq. 2.13 must be zero. Considering successively those perturbations $\delta x(t)$ which are nonzero at only one boundary, one observes that each of the terms of Eq. 2.12 must be zero, when evaluated on $\bar{x}(t)$ at $\bar{t}_i$, in order that the first variation of J be zero. Combining the first two terms of Eq. 2.12 with Eq. 2.13 for $i=0,2$, one obtains the initial and terminal transversality conditions given by

$$\lambda^1(\bar{t}_0) + \frac{\partial \phi^0}{\partial x}[x^1(\bar{t}_0), \bar{t}_0] + \sum_{i=1}^{k_0} \mu_i \cdot \frac{\partial \psi_i^0}{\partial x}[x^1(\bar{t}_0), \bar{t}_0] = 0 \qquad (2.14)$$

$$H[x^1(\bar{t}_0), u^1(\bar{t}_0), \lambda^1(\bar{t}_0), \bar{t}_0] - \frac{\partial \phi^0}{\partial t}[x^1(\bar{t}_0), \bar{t}_0] + \sum_{i=1}^{k_0} \mu_i \cdot \frac{\partial \psi_i^0}{\partial t}[x^1(\bar{t}_0), \bar{t}_0] = 0 \qquad (2.15)$$

$$\lambda^2(\bar{t}_2) - \frac{\partial \phi^2}{\partial x}[x^2(\bar{t}_2), \bar{t}_2] + \sum_{i=1}^{k_2} \nu_i \cdot \frac{\partial \psi_i^2}{\partial x}[x^2(\bar{t}_2), \bar{t}_2] = 0 \qquad (2.16)$$

and

$$H[x^2(\bar{t}_2),u^2(\bar{t}_2),\lambda^2(\bar{t}_2),\bar{t}_2]+\frac{\partial\phi^2}{\partial t}[x^2(\bar{t}_2),\bar{t}_2] + \sum_{i=1}^{k_2} \nu_i \cdot \frac{\partial\psi_i^2}{\partial t}[x^2(\bar{t}_2),\bar{t}_2] = 0 \tag{2.17}$$

which must be satisfied by the optimal trajectory. The last term of Eq. 2.12, when combined with Eq. 2.13 for i = 1, gives a set of intermediate transversality conditions in terms of discontinuities in $\overline{\lambda}(t)$ and $\overline{H}(t)$ which are given by

$$\lambda^2(\bar{t}_1^+) - \lambda^1(\bar{t}_1^-)+\frac{\partial\phi^1}{\partial x}[\bar{x}(\bar{t}_1),\bar{t}_1] + \sum_{i=1}^{k_1} \gamma_i \cdot \frac{\partial\psi_i^1}{\partial x}[\bar{x}(\bar{t}_1),\bar{t}_1] = 0 \tag{2.18}$$

$$H[\bar{x}(\bar{t}_1),u^2(\bar{t}_1^+),\lambda^2(\bar{t}_1^+),\bar{t}_1] - H[\bar{x}(\bar{t}_1),u^1(\bar{t}_1^-),\lambda'(\bar{t}_1^-),t_1] - \frac{\partial\phi^1}{\partial t}[\bar{x}(\bar{t}_1),\bar{t}_1]$$

$$+ \sum_{i=1}^{k_1} \gamma_i \cdot \frac{\partial\psi_i^1}{\partial t}[\bar{x}(\bar{t}_1),\bar{t}_1] = 0 \tag{2.19}$$

Adopting the notation of Eqs. A.66 and A.67, Eqs. 2.14 through 2.19 can be more briefly written as

$$\left[\nabla\phi^0 + \underline{\overline{\lambda}}(t) + \sum_{i=1}^{k_0} \mu_i \cdot \nabla\psi_i^0\right]_{\bar{t}_0,\bar{x}(\bar{t}_0),\bar{u}(\bar{t}_0),\overline{\lambda}(\bar{t}_0)} = 0 \tag{2.20}$$

$$\left[\nabla\phi^2 + \underline{\overline{\lambda}}(t) + \sum_{i=1}^{k_2} \nu_i \nabla\psi_i^2\right]_{\bar{t}_2,\bar{x}(\bar{t}_2),\bar{u}(\bar{t}_2),\overline{\lambda}(\bar{t}_2)} = 0 \tag{2.21}$$

and

$$\left[\nabla\phi^1 + \Delta\,\underline{\overline{\lambda}}(t) + \sum_{i=1}^{k_1} \gamma_i \cdot \nabla\psi_i^1\right]_{\bar{t}_1,\bar{x}(\bar{t}_1),\bar{u}(\bar{t}_1^{\pm}),\overline{\lambda}(\bar{t}_1^{\pm})} = 0 \tag{2.22}$$

Thus for $\bar{u}(t)$ to be the optimal control, and $\bar{x}(t)$ the resulting optimal trajectory, it is necessary that there exist a $\bar{\lambda}(t)$ such that Eqs. 2.9 through 2.11 and 2.20 through 2.22 are satisfied.

By carrying out the Taylor series expansion to second order terms and requiring that the second variation of J be nonnegative, one finds that the Legendre condition* must be satisfied over each subinterval, i.e., the second partial derivative of H with respect to u must be nonnegative,

$$\frac{\partial^2 H}{\partial u^2} \geq 0 \tag{2.23}$$

for $t_0 < t < t_1$ and $t_1 < t < t_2$. While this condition is necessary for the optimal solution, it does not constitute a sufficiency condition since it may also be satisfied by local maxima.

The above derivation has been limited to a three-point optimal control problem for simplicity of presentation only, and the extension to intermediate equality constraints at several points in time is straightforward. The cost functional, Eq. 2.2, must be written as the sum of the integral over the subintervals of (t_0, t_f) as defined by the t_i. Treating the perturbations in the same way as before, one obtains the result that one must satisfy the Euler-Lagrange equations, Eqs. 2.9 through 2.11, over each subinterval, i.e., (t_{i-1}, t_i) for $i = 1, 2, \ldots, N_f$, the initial and terminal transversality conditions given by Eqs. 2.20 and 2.21, and a set of intermediate transversality conditions at each intermediate t_i, $(i=1, \ldots, N_f-1)$, given by

$$\left[\nabla \phi^i + \nabla \underline{\bar{\lambda}}(t) + \sum_{j=1}^{k_i} \gamma_j^i \cdot \nabla \psi_j^i \right]_{\bar{t}_i, \bar{x}(\bar{t}_i), \bar{u}(\bar{t}_i^{\pm}), \bar{\lambda}(\bar{t}_i^{\pm})} = 0 \tag{2.24}$$

---

* See Section A.7.

2.4 EXAMPLE: SINGLE INTEGRATOR— FREE INTERMEDIATE TIME

In this section a simple example is presented to demonstrate the application of the necessary conditions to the solution of a three-point optimal control problem. The system is assumed to be a single integrator,

$$\dot{x} = u \tag{2.25}$$

with a cost functional of the form

$$J = \int_{t_0}^{t_f} L[x,u,t]\,dt = \frac{1}{2}\int_{t_0}^{t_2} K(t)[u(t)]^2 dt \tag{2.26}$$

where

$$K(t) = \begin{cases} K_1 = 1 & t_0 \leq t < t_1 \\ K_2 = K & t_1 < t \leq t_2 \end{cases} \tag{2.27}$$

and $K_2 \neq 0$. The state constraints are given by

$$\psi_1^0[x(t),t] = x(t) - x_0 = 0 \qquad \text{a)}$$

$$\psi_2^0[x(t),t] = t - t_0 = 0 \qquad \text{b)}$$

$$\psi_1^1[x(t),t] = x(t) - x_1 = 0 \qquad \text{c)} \tag{2.28}$$

$$\psi_1^2[x(t),t] = x(t) - x_2 = 0 \qquad \text{d)}$$

$$\psi_2^2[x(t),t] = t - t_2 = 0 \qquad \text{e)}$$

with

$$t_0 < t_1 < t_2 \qquad \text{f)}$$

Thus, it is desired to take the state x from the value x_0 at $t = t_0$ to the value x_2 at $t = t_2$, while requiring that the state attain the value x_1 at an intermediate time t_1 and minimize the cost

functional, Eq. 2.26. The Hamiltonian for this problem is given by

$$H[x,u,\lambda,t] = \frac{1}{2} K(t)[u]^2 + \lambda \cdot u \tag{2.29}$$

and the Euler-Lagrange equations, Eqs. 2.9 through 2.11, become*

$$\frac{dx^i}{dt} - \frac{\partial H^i}{\partial \lambda} = \frac{dx^i}{dt} - u^i = 0 \tag{2.30}$$

$$\frac{d\lambda^i}{dt} + \frac{\partial H^i}{\partial x} = \frac{d\lambda^i}{dt} = 0 \tag{2.31}$$

and

$$\frac{\partial H^i}{\partial u} = K_i u^i + \lambda^i = 0 \tag{2.32}$$

respectively. From Eq. 2.31 the costate is piecewise constant. Solving Eq. 2.32 for u^i as a function of λ^i, the control can be eliminated from Eq. 2.30, i.e.,

$$\frac{dx^i}{dt} = -\frac{\lambda^i}{K_i} \tag{2.33}$$

Let us assume that the problem has been previously scaled so that $x_0 = t_0 = 0$ and $x_2 = t_2 = 1$. Equation 2.33 can be integrated forward in time from t_0 to determine $x^1(t)$ and backward in time from t_2 to determine $x^2(t)$, using the boundary conditions of Eq. 2.28, resulting in

$$x^1(t) = -\lambda^1 \cdot t \qquad \text{for } 0 \le t \le t_1 \tag{2.34}$$

and

$$x^2(t) = 1 - \frac{\lambda^2}{K}(t-1) \qquad \text{for } t_1 \le t \le t_2 \tag{2.35}$$

At this point there are three unknowns to be determined, λ^1, λ^2, and t_1, and three intermediate boundary conditions, two of them given by Eq. 2.28c and the continuity of state, and the third by the intermediate

---

* Note that the superscripts denote the time interval of definition as before. Exponents are denoted by $[\ ]^n$.

transversality condition, Eq. 2.19. Substituting Eqs. 2.34 and 2.35 into Eq. 2.28c, one solves for λ^1 and λ^2 in terms of quantity the x_1 and the unknown t_1 and obtains

$$\lambda^1 = -\frac{x_1}{t_1} \tag{2.36}$$

and

$$\lambda^2 = -\frac{K(x_1 - 1)}{(t_1 - 1)} \tag{2.37}$$

The transversality and the intermediate transversality conditions given by Eqs. 2.14 through 2.18 yield no information. The intermediate transversality condition given by Eq. 2.19 becomes

$$\Delta H(t_1) = H[x(t_1),u^2(t_1^+),\lambda^2(t_1^+),t_1] - H[x(t_1),u^1(t_1^-),\lambda^1(t_1^-),t_1] = 0 \tag{2.38}$$

This relation provides the additional condition required to determine t_1. From Eq. 2.38

$$\lambda^2 = \pm\sqrt{K}\cdot\lambda^1 \tag{2.39}$$

so that one can solve for t_1, as a function of x_1 and K, given by

$$t_1 = \frac{x_1}{x_1 \pm \sqrt{K}\cdot(1 - x_1)} \tag{2.40}$$

The two solutions of t_1, as functions of x_1, are drawn in Fig. 2.1 for K =1, from which it can be seen that for each x_1 there is one and only one of the solutions which satisfies the ordering of the boundary times specified by Eq. 2.28f. For example, if $K = 1$ and $x_1 = 2$, then $t_{1(+)} = 2$ and $t_{1(-)} = \frac{2}{3}$. Thus the optimal value of t_1 is 2/3 and the optimal control, given by

$$u(t) = \begin{cases} +3 & 0 \le t < \frac{2}{3} \\ -3 & \frac{2}{3} < t \le 1 \end{cases} \tag{2.41}$$

is plotted in Fig. 2.2 along with the optimal state trajectory. Note that the control u, and thus the costate λ, is discontinuous at t_1. As a final observation, note that as a result of Eq. 2.40 and Fig. 2.1, the

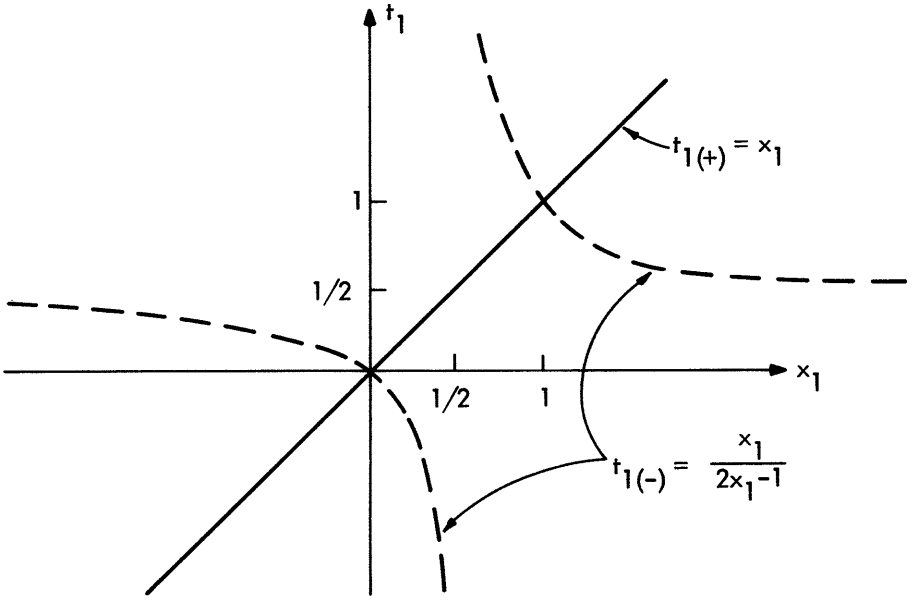
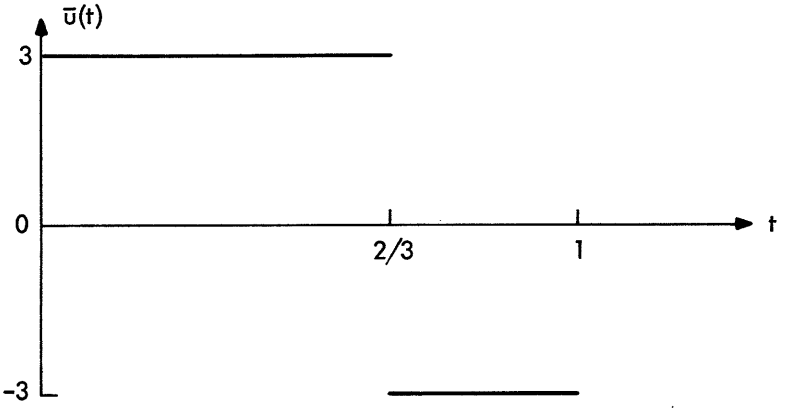
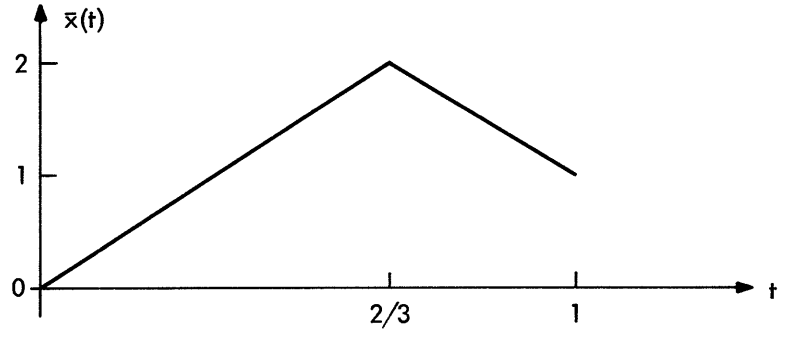


Fig. 2.1 Example 2.4 - Optimal t_1 vs. x_1, Two Solutions



(a) $\bar{u}(t)$ vs. t



(b) $\bar{x}(t)$ vs. t

Fig. 2.2 Example 2.4 - Optimal Control and State vs. Time (normalized)

solution to the more general problem of the optimal ordering of the constraint sets has been obtained. While it was not the case for this example, one may wish to determine the optimal order in which the constraint sets should be satisfied. The two solutions of t_1 in Eq. 2.40 correspond to two local minima for this more general problem, and the optimal ordering of ψ^0, ψ^1, and ψ^2 can be determined by directly comparing the costs for each of the local minima.

2.5 LINEAR SYSTEM-QUADRATIC COST OPTIMAL CONTROL PROBLEM

While a closed form solution is obtained for the three-point optimal control example of Section 2.4, one cannot, in general, obtain such a solution for problems with more complex system dynamics, cost functional, and state constraints. One must then resort to an iterative algorithm to obtain a numerical solution. There is, however, a general class of problems which permits an analytical solution which generally requires the use of a digital computer to obtain a time-varying feedback gain matrix (in this case, however, the computer is not employed in an iterative fashion). These problems are characterized by linear system dynamics and quadratic cost functionals. With some increase in complexity, the class of problems can be extended to include linear state constraints.[55] In this section, the discussion is limited to problems with no state constraints, but with quadratic penalty functions on the state at several points in time.

Let us consider a system described by a set of n first order linear differential equations of the form

$$\frac{dx}{dt}(t) = A(t)x(t) + B(t)u(t) \tag{2.42}$$

with a given set of initial conditions

$$x(t_0) = x_0 \tag{2.43}$$

that are arbitrary, where $x(t)$ is the n dimensional state vector, $u(t)$ is the m dimensional control vector, and $A(t)$ and $B(t)$ are $(n \times n)$ and $(n \times m)$ matrices, respectively. One seeks the control $\bar{u}(t)$ which minimizes a cost functional of the form

$$J = \frac{1}{2}\sum_{i=1}^{N_f} x^T(t_i)S_i x(t_i) + \frac{1}{2}\int_{t_0}^{t_{N_f}} [x^T(t)P(t)x(t) + u^T(t)Q(t)u(t)]\,dt \quad (2.44)$$

where the S_i and $P(t)$ are positive semi-definite $(n \times n)$ matrices for all $i = 1,\ldots,N_f$ and $t_0 < t < t_{N_f}$ (at least one of these matrices must be nonzero in order that the solution not be trivial), $Q(t)$ is a positive definite $(m \times m)$ matrix for all $t_0 < t < t_{N_f}$, and the t_i are explicitly specified and are ordered as in Eq. 2.4.

Let us first consider the case where $N_f = 2$. If $S_1 = 0$, then one has a penalty imposed on the state at only one point in time, and the solution is well known.[5] The Hamiltonian is given by

$$H[x,u,\lambda,t] = \frac{1}{2}x^T P(t)x + \frac{1}{2}u^T A(t)u + \lambda^T A(t)x + \lambda^T B(t)u \quad (2.45)$$

and the Euler-Lagrange equations are given by Eq. 2.42,

$$\frac{\partial H}{\partial u} = Q(t)u(t) + B(t)\lambda(t) = 0 \quad (2.46)$$

and

$$\frac{d\lambda}{dt} = -\frac{\partial H}{\partial x} = -P(t)x(t) - A^T(t)\lambda(t) \quad (2.47)$$

Since $Q(t)$ is positive definite, and therefore its inverse exists,[4] one can solve Eq. 2.46 for $u(t)$,

$$u(t) = -Q^{-1}(t)B^T(t)\lambda(t) \quad (2.48)$$

which upon substitution into Eqs. 2.42 gives

$$\frac{dx}{dt}(t) = A(t)x(t) - B(t)Q^{-1}(t)B^T(t)\lambda(t) \quad (2.49)$$

Equations 2.47 and 2.49 form a set of $2n$ linear homogeneous differential equations in $x(t)$ and $\lambda(t)$, from which one can argue that $x(t)$ and $\lambda(t)$ must be linearly related by[5]

$$\lambda(t) = S(t)x(t) \quad (2.50)$$

The transversality condition at t_2 is given, from Eq. 2.16, by

$$\lambda(t_2) = \frac{\partial \phi^2}{\partial x}(t_2) = S_2 x(t_2) \tag{2.51}$$

and thus the boundary condition on $S(t)$ at t_2 is

$$S(t_2) = S_2 \tag{2.52}$$

Furthermore, by differentiating Eq. 2.50, and substituting Eqs. 2.47 and 2.49 into the result, one determines that $S(t)$ must satisfy the differential equation

$$\frac{dS}{dt}(t) = -A^T(t)S(t) - S(t)A(t) + S(t)B(t)Q^{-1}(t)B^T(t)S(t) - P(t) \tag{2.53}$$

which is of the matrix Riccati type. Thus, by integrating (numerically) Eq. 2.53 backward in time from t_2 to t_0, using the boundary condition given by Eq. 2.52, one obtains a linear feedback control law

$$\bar{u}(t) = -Q^{-1}(t) \cdot B^T(t) \cdot S(t) \cdot x(t) \tag{2.54}$$

which gives the optimal control as a function of the state.

Now, if S_1 is not zero, one must satisfy the intermediate transversality condition at t_1 given by

$$-\Delta\lambda(t_1) = \frac{\partial \phi^1}{\partial x}(t_1) = S_1 x(t_1) \tag{2.55}$$

However, since the Euler-Lagrange equations to be satisfied over each subinterval, (t_0, t_1^-) and (t_1^+, t_2), are the same, it follows that the solution over each subinterval must be of the same form as before, i.e.,

$$\lambda^1(t) = S^1(t) \cdot x^1(t) \tag{2.56}$$

$$\lambda^2(t) = S^2(t) \cdot x^2(t) \tag{2.57}$$

where both $S^1(t)$ and $S^2(t)$ must satisfy the matrix Riccati differential equation given by Eq. 2.53. The transversality condition at t_2 is still given by Eq. 2.51, and therefore, the boundary condition on $S^2(t)$ is given by

$$S^2(t_2) = S_2 \tag{2.58}$$

Substituting Eqs. 2.56 and 2.57 into Eq. 2.55 one obtains

$$S^1(t_1^-)x^1(t_1) - S^2(t_1^+)x^2(t_1) = S_1 x(t_1) \tag{2.59}$$

from which the boundary condition on $S^1(t)$ is found to be

$$S^1(t_1^-) = S_1 + S^2(t_1^+) \tag{2.60}$$

The procedure, then, is to integrate the Riccati equation, Eq. 2.53, backward in time from t_2 to t_1, starting with the boundary condition given by Eq. 2.58. At t_1 one applies the discontinuity given by Eq. 2.60 and then continues to integrate from t_1 back to t_0. Thus, Eq. 2.54 gives a linear feedback control law which has a discontinuous gain at t_1. In the general case where N_f is greater than 2, one follows the same procedure, obtaining discontinuities in $S(t)$ at each of the t_i.

To demonstrate these results, consider a double integrator system

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \end{aligned} \tag{2.61}$$

and a cost functional given by

$$J = \frac{1}{2} s_1 \cdot [x_1(.8)]^2 + \frac{1}{2} s_2 \cdot [x_1(1)]^2 + \frac{1}{2}\int_0^1 [u(t)]^2 dt \tag{2.62}$$

For this example $t_1 = .8$, $t_2 = 1$, $P = 0$, $Q = 1$, and

$$b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \quad A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}; \quad S_1 = \begin{bmatrix} s_1 & 0 \\ 0 & 0 \end{bmatrix}; \quad S_2 = \begin{bmatrix} s_2 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.63}$$

The matrix Riccati differential equation becomes

$$\begin{aligned} \dot{s}_{11} &= [s_{12}]^2 \\ \dot{s}_{12} &= -s_{11} + s_{12} \cdot s_{22} = \dot{s}_{21} \\ \dot{s}_{22} &= -2 \cdot s_{12} + [s_{22}]^2 \end{aligned} \tag{2.64}$$

and the feedback control law is given by

$$u(t) = - s_{11}(t) x_1(t) - s_{22}(t) x_2(t) \tag{2.65}$$

The boundary conditions on $S(t)$ are given by

$$\begin{aligned} s_{11}(1) &= s_2 \\ -\Delta s_{11}(.8) &= s_{11}(.8^-) - s_{11}(.8^+) = s_1 \\ s_{12}(1) &= \Delta s_{12}(.8) = s_{22}(1) = \Delta s_{22}(.8) = 0 \end{aligned} \quad (2.66)$$

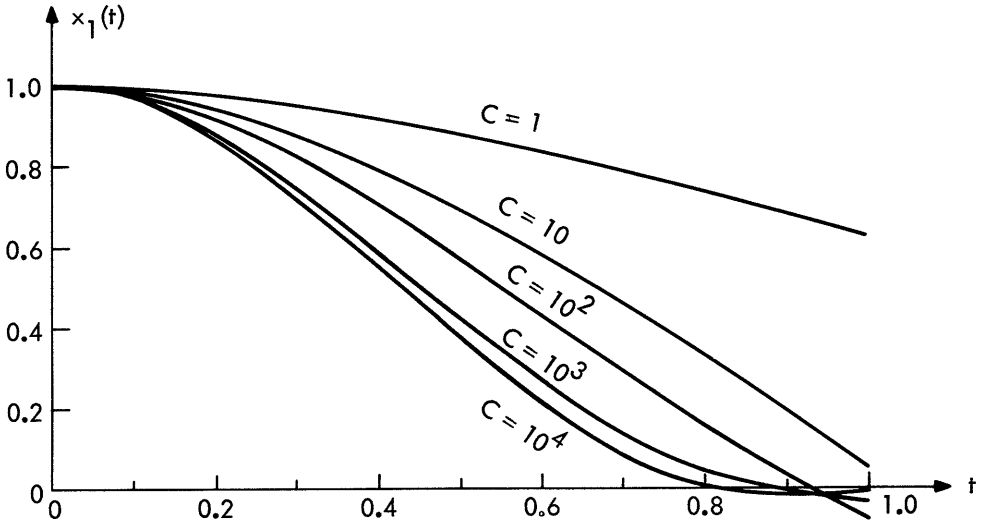
Figure 2.3 gives the optimal $x_1(t)$ and $u(t)$ for several values of s_1 and s_2, and Fig. 2.4 demonstrates the discontinuity in $s_{11}(t)$ at $t = t_1 = .8$ for $s_1 = s_2 = 10^4$. Since the feedback gain is not directly a function of s_{11}, the optimal control is not discontinuous at t_1. Its derivative, however, is discontinuous at that point, since the feedback gain is a function of s_{12} and s_{22} which have discontinuous derivatives at t_1.

2.6 DISCUSSION OF THE NECESSARY CONDITIONS

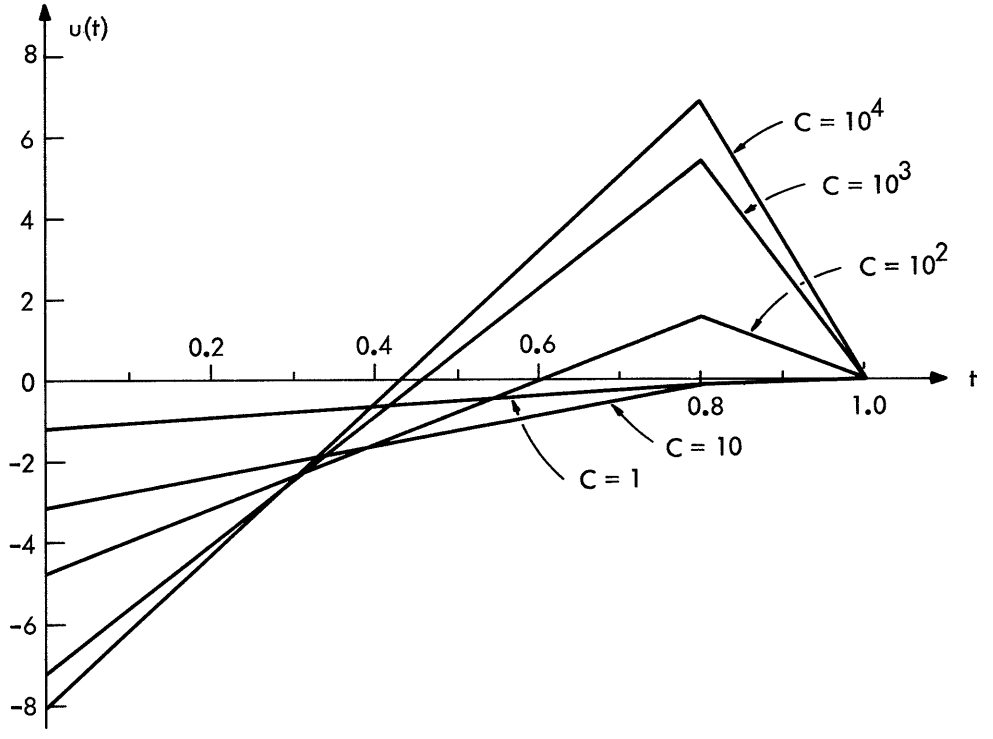
The necessary conditions on the solution of a general N-point optimal control problem are derived in Section 2.3 and their application to the solution of a simple example and to the solution of a general class of problems is demonstrated in Sections 2.4 and 2.5, respectively. The usual two-point optimization problems may have as few as none or as many as n constraints (n being the dimension of the state vector) placed on the state variables at both the initial and the terminal times. Also, the initial and terminal times may be either specified explicitly or left free. Application of the calculus of variations to the optimal control problem provides additional conditions to be satisfied at the end points (direct extensions of the "natural boundary conditions" of the calculus of variations* to the case where the initial and terminal state and time variations are not free), so that a total of $(n+1)$ conditions are placed on the state, costate, and time at each of the end points. These conditions provide an adequate number of boundary values to completely specify a solution to the $2n$ differential equations for $\bar{x}(t)$ and $\bar{\lambda}(t)$ which are given by the Euler-Lagrange equations, resulting in a two-point boundary value problem.

---

* See Sections A.2 and A.3.



(a) $\bar{x}_1(t)$ vs. t; $C = S_1 = S_2$



(b) $\bar{u}(t)$ vs. t, $C = S_1 = S_2$

Fig. 2.3 Example 2.6 - Double Integrator, Optimal Solution

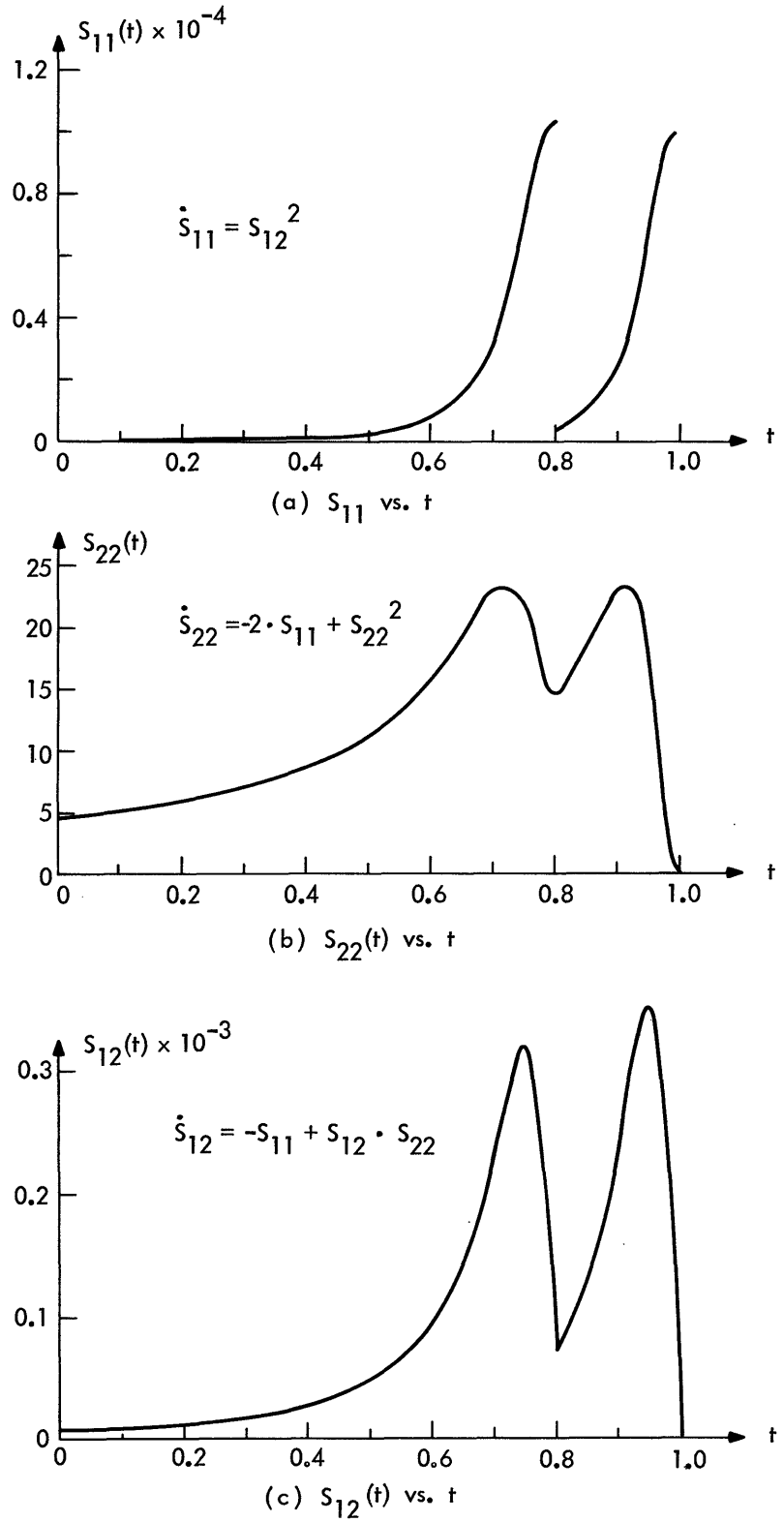


Fig. 2.4 Example 2.6 - Solution of Matrix Riccati Differential Equation

In the case of the N-point optimal control problem, k_i constraints ($0 \leq k_i \leq n$) may be placed on the state at each of the intermediate times, t_i, which themselves may or may not be specified. The extension of the calculus of variations to this problem furnishes additional intermediate boundary conditions so that at each t_i there are $(2n+1)$ conditions to be satisfied by the state, costate, and time. This problem can be called an N-point BVP or a set of (N-1) coupled 2-point BVP's.

In the 2-point BVP one seeks that solution (or those solutions, if there is not a unique extremal), from the family of solutions of the Euler-Lagrange equations, which satisfies the boundary conditions at each end point. In a 3-point BVP, one must determine two dintinct solutions to the Euler-Lagrange equations, one defined over (t_0, t_1) which satisfies n boundary conditions at each of t_0 and t_1, and the other defined over the interval (t_1, t_2) which satisfies n boundary conditions at each of t_1 and t_2. The two solutions, however, are not, in general, independent, but are coupled through the boundary conditions at t_1. A trivial case occurs if at the intermediate time $x(t_1)$ and t_1 are all specified explicitly. In this case, the two solutions, while sharing a common boundary, are totally independent and can be solved separately. Some freedom must be allowed in either the state or the time at the intermediate point in order to obtain a coupled problem, in which one must determine the optimum values of the free states (and/or time) such that the sum of the cost required to achieve that intermediate state and the cost of taking the system from that intermediate state to the desired terminal state is minimized. Extension of these ideas to the general N-point BVP is straightforward.

The development of the necessary conditions includes the class of problems for which either $f(x,u,t)$ or $L(x,u,t)$ may change functional form from one subinterval to the next. In this case, the Euler-Lagrange equations to be satisfied over each subinterval are different, and thus the family of solutions from which a particular solution is to be selected is different for each subinterval. The above development does not include, however, discontinuous-state problems. If the state is allowed

to have discontinuities as may occur, for example, in a booster-staging problem, Eq. 2.24 is not directly applicable, and one must revert to the general expression for the first variation of J and treat the state variations at the intermediate points appropriately.

CHAPTER III

NUMERICAL ALGORITHMS FOR SOLVING THE N-POINT BVP

3.1 INTRODUCTION

Since optimal control problems, by their very nature, require the solution of high-order nonlinear differential equations with boundary conditions imposed on the solution at more than one point in time, analytical closed-form solutions are often not attainable. One must then resort to numerical techniques to obtain a solution. The numerical methods available for the solution of two-point BVP's invariably involve making some initial guess at the solution and then attempting to arrive at a better guess. Thus, in an iterative fashion the optimal solution is converged upon. The various numerical techniques can be classified as being either direct or indirect methods, depending on their fundamental approach to the iterative solution. The direct methods search for the optimal solution by a direct comparison of the value of the cost functional, J, at each iteration, and select that solution which yields the smallest value of J. The most crude (and impractical) direct method is to make an exhaustive search of all possible control functions and select the one that yields the smallest value of J. A somewhat more sophisticated, and more efficient, direct method is that of steepest-descent, which evaluates not only J, but also its gradient in function space at each iteration. It then utilizes this information by making the next guess in (basically) the negative gradient direction to insure that a smaller value of J is obtained. By their very nature, the direct methods seek out (at least) a local minimum of J. The indirect methods, however, attempt to converge on a solution which satisfies the first order necessary conditions which are imposed on the optimal solution. As a result, these methods may converge to local maximum as well as local minimum.

Since the iterative methods involve approximating the general nonlinear optimal control problem in the vicinity of the guessed solution by another simpler problem that can be solved, these methods can be

further classified as to the degree of accuracy of the approximation. First-order methods,[21,22] such as steepest-descent,[23] involve linearization about the nominal guess. They generally exhibit relatively slow convergence as the optimal solution is approached, since then the higher order terms begin to dominate. The second-order methods,[30-32] such as Newton-Raphson,[33] approximate the problem to second-order terms, and, therefore, they exhibit much more rapid convergence once a guess close enough to the solution is attained. However, these second-order methods may not converge at all if the initial guess is poor.

Of the many computational methods available for solving two-point BVP's, two of the more popular methods, the steepest-descent and the Newton-Raphson, are modified in this chapter to extend their range of application to include general N-point optimal control problems. Two different approaches to the steepest-descent method, with respect to the treatment of the boundary conditions, are considered. Also, a method for incorporating some second order information into the choice of step size in order to speed up the convergence of the method, as developed in Appendix B, is utilized. The application of the steepest-descent algorithm to a three-point optimal control problem is demonstrated by means of a two-target missile guidance problem. The example involves state constraints at three points in time, with the intercept times unspecified. It also allows a demonstration of the extension to bounded control problems by imposing turning rate constraints on the missile.

The Newton-Raphson method is extended first to the three-point optimal control problem, and then to the general N-point BVP, for fixed terminal and intermediate times (that is, the t_i are all explicitly specified in the statement of the problem). The method is then further extended to the solution of free-time problems, by solving a sequence of fixed time problems in order to converge upon the optimal boundary times. Finally, a transformation of variables technique is presented which allows a certain class of free-time two-point BVP's to be transformed into fixed-time problems in order to take advantage of the fact that the Newton-Raphson method is inherently better suited to the solution of such problems.

The application of the Newton-Raphson algorithm to a three-point BVP is demonstrated by means of the same missile guidance example used for the steepest-descent method. In addition, the utility of the change of variables technique is demonstrated by a single target intercept problem.

Before proceeding with the development of the steepest-descent and Newton-Raphson methods, an indirect method is discussed in Section 3.2, which enables one to determine the optimal solution by solving a sequence of initial value problems. While this method can be impractical to implement due to computational difficulties, its discussion is considered worthwhile because of the insight it gives into the nature of the transversality and intermediate transversality conditions.

3.2 AN INDIRECT METHOD

The two-point optimal control problem cannot be solved as an initial value problem because it involves the integration of 2n differential equations in which only n boundary conditions are specified at either end. The technique[22] which first comes to mind is to guess the n unknown initial conditions and integrate the Euler-Lagrange differential equations up to the terminal time, at which point the n required terminal conditions will not, in general, be satisfied. The objective is to iteratively converge on the correct guess of the unknown initial conditions, so that the terminal conditions are satisfied. One can consider the miss distances on the terminal conditions as functions of the n guessed initial conditions and apply various minimization techniques, such as the steepest-descent or Newton-Raphson, to bring these miss distances to zero.

In three-point optimal control problems the number of unknown conditions increases. For example, if there are n constraints put on the state at each of three specified points in time t_0, t_1 and t_2 (the special case of two independent two-point optimal control problems sharing a common boundary point), the number of unknown parameters to be determined is equal to 2n. One guesses the n initial costates and integrates the Euler-Lagrange differential equations from t_0 to t_1, at which point the 2n intermediate conditions must be specified. The n necessary conditions given by the continuity of state can be trivially

satisfied. From the intermediate transversality conditions, Eq. 2.18, the optimal costate undergoes an unspecified discontinuity at t_1. Therefore, one must guess the discontinuities in the n costate variables before continuing the integration on to t_2. One must now iterate on a total of 2n guessed parameters, the n costates at t_0 and n discontinuities in the costates at t_1, in order that 2n conditions be satisfied, n state constraints at each of t_1 and t_2.* In general, if k_1 equality constraints are placed on the state of the system at t_1, then one must iterate on a total of $(n+k_1)$ unspecified parameters, in order to satisfy the same number of boundary conditions at t_1 and t_2. One could start by guessing n unknown terminal conditions** and integrate the Euler-Lagrange equations backward in time as suggested by Breakwell.[22] Or, in fact, one could guess n initial and n terminal conditions, integrating them forward and backward, respectively, to t_1, and then iterate on the correct values which allow the satisfaction of the intermediate state constraints, transversality conditions, and the continuity of state (a total of 2n intermediate boundary conditions). With this approach, a total of 2n parameters must be guessed regardless of the number of intermediate state constraints.

The extension to the general N-point optimal control problem is straightforward. If k is the total number of constraints placed on the state of the system at $t_1,\ldots,t_{N-1}$, then k parameters must be guessed (the initial costates at t_0 and discontinuities in the costates at the $t_i, i=1,2,\ldots,N-2$) and iterated on. While this method is quite straightforward, it has many computational disadvantages. For example, intermediate and terminal constraint violations may be extremely sensitive to changes in the initial conditions, and one is required to make an

---

* On the other hand, if a penalty function, rather than a set of equality constraints, is placed on the state at t_1, then one can compute discontinuities in the costates at t_1 directly from Eq. 2.18, as functions of the state at t_1. In this case, no additional guesses need to be made.

** This approach may be preferred for systems in which the terminal conditions are extremely sensitive to changes in the initial conditions.

educated guess at the unknown parameters in order to obtain convergence. It is also possible that small changes in the guessed parameters may produce no changes in the terminal conditions.[45]

3.3 THE METHOD OF STEEPEST-DESCENT

The steepest-descent algorithm for solving two-point optimal control problems is reviewed in Appendix B. The method basically involves linearizing the system dynamics, terminal state constraints, and cost function about a nominal solution. One then formulates a new optimization problem, that of determining the control perturbation $\delta u(t)$ which minimizes the change in the cost function (i.e., makes the largest negative change possible) subject to constraints on the control perturbation step size and on the change in the terminal constraint violation. Two approaches to handling the state equality constraints are considered -- through penalty functions or through influence functions (as discussed in Appendix B).

Let us assume that the system dynamics are given by

$$\dot{x}(t) = f[x(t), u(t), t] \tag{3.1}$$

the state equality constraints are given by

$$\psi^i[x(t_i), t_i] = 0 \quad , \quad i = 0, 1, \ldots, N_f \tag{3.2}$$

and the cost functional is given by

$$J = \sum_{i=0}^{N_f} \phi^i[x(t_i), t_i] + \int_{t_0}^{t_{N_f}} L[x(t), u(t), t] \cdot dt \tag{3.3}$$

where $x(t)$ is the n-vector of state variables, $u(t)$ is the m-vector of control variables, and ψ^i is the k_i-vector of state (and time) constraints at t_i.

3.3a The Penalty Function Approach

With the penalty function approach one does not treat the state equality constraints (Eq. 3.2) explicitly. Rather, one formulates a new optimization problem with no state equality constraints but with the cost functional modified such that the new penalty functions are given by

$$\hat{\phi}^i[x(t_i),t_i] = \phi^i[x(t_i),t_i] + \frac{1}{2}\cdot\sum_{j=1}^{k_i} C_j^i\left[\psi_j^i[x(t_i),t_i]\right]^2 \qquad (3.4)$$

The C_j^i are arbitrary weighting factors such that, in the limit as they become arbitrarily large, the violations of the state equality constraints become arbitrarily small, and the solution to the modified problem approaches the solution of the original problem.[28,29,39,40] The new cost functional is now given by

$$\hat{J} = \sum_{i=0}^{N_f} \hat{\phi}^i[x(t_i)] + \int_{t_0}^{t_f} L(x,u,t)\,dt \qquad (3.5)$$

To facilitate the following discussion, it is assumed that the t_i have been explicitly specified. The first variation of J about a nominal solution, $(u_j(t),\ x_j(t))$,* is given by

$$\delta\hat{J} = \sum_{i=0}^{N_f} \frac{\partial\hat{\phi}^i}{\partial x}[x_j(t_i)]\cdot\delta x(t_i) + \int_{t_0}^{t_f}\left[L_x(u_j,x_j,t)\cdot\delta x + L_u(u_j,x_j,t)\cdot\delta u\right]dt \qquad (3.6)$$

and the linearized system dynamics are given by

$$\delta\dot{x}(t) = \frac{\partial f}{\partial x}[x_j(t),u_j(t),t]\cdot\delta x(t) + \frac{\partial f}{\partial u}[x_j(t),u_j(t),t]\cdot\delta u(t)$$

$$= F_j(t)\cdot\delta x(t) + G_j(t)\cdot\delta u(t) \qquad (3.7)$$

We now define an n-vector function $\lambda(t)$ which satisfies the differential equation

$$\dot{\lambda}_{j+1}(t) = -\frac{\partial f}{\partial x}[x_j(t),u_j(t),t]\cdot\lambda_{j+1}(t) - L_x[x_j(t),u_j(t),t] \qquad (3.8)$$

* The subscript j is used here to denote the index of the iteration.

and note that

$$\frac{d}{dt}(\lambda^T_{j+1}\cdot\delta x) = \lambda^T_{j+1}\cdot\delta\dot{x} + \dot{\lambda}^T_{j+1}\cdot\delta x = \lambda^T_{j+1}(t)G_j(t)\cdot\delta u(t) - L_{x,j}(t)\cdot\delta x(t) \tag{3.9}$$

Integrating Eq. 3.9, one can write

$$\lambda^T_{j+1}(t_1^-)\dot{x}(t_1^-) - \lambda^T_{j+1}(t_0)\dot{x}(t_0) = \int_{t_0}^{t_1^-} [\lambda^T_{j+1}(t)G_j(t)\cdot\delta u(t) - L_{x,j}(t)\cdot\delta x(t)]\,dt$$

$$\vdots \tag{3.10}$$

$$\lambda^T_{j+1}(t_{N_f})\cdot\delta x(t_{N_f}) - \lambda^T_{j+1}(t_{N_f-1}^+)\cdot\delta x(t_{N_f-1}^+) = \int_{t_{N_f-1}}^{t_{N_f}} [\lambda^T_{j+1}(t)G_j(t)\delta u(t) - L_{x,j}(t)\cdot\delta x(t)]dt \tag{3.11}$$

Adding Eqs. 3.10 through 3.11, one obtains

$$\lambda^T_{j+1}(t_{N_f})\cdot\delta x(t_{N_f}) - \sum_{i=1}^{N_f-1}\Delta\lambda^T_{j+1}(t_i)\cdot\delta x(t_i) - \lambda^T_{j+1}(t_0)\cdot\delta x(t_0) = \int_{t_0}^{t_{N_f}} [\lambda^T_{j+1}(t)G_j(t)\delta u(t) - L_{x,j}(t)\cdot\delta x(t)]\cdot dt \tag{3.12}$$

Assuming that $x(t_0)$ is specified, so that $\delta x(t_0)$ is zero, one defines

$$\lambda^T_{j+1}(t_{N_f}) = \frac{\partial \hat{\phi}^{N_f}}{\partial x}[x_j(t_{N_f})] \tag{3.13}$$

and

$$-\Delta\lambda^T_{j+1}(t_i) = \lambda^T_{j+1}(t_i^-) - \lambda^T_{j+1}(t_i^+) = \frac{\partial \hat{\phi}^i}{\partial x}[x_j(t_i)] \qquad \text{for } i=1,\ldots,N_f-1 \tag{3.14}$$

so that Eq. 3.6 becomes

$$\delta\hat{J} = \int_{t_0}^{t_{N_f}} H_u[u_j(t), x_j(t), \lambda_{j+1}(t), t]\cdot\delta u(t)\,dt \tag{3.15}$$

where

$$H_u[u_j(t), x_j(t), \lambda_{j+1}(t), t] = L_u[x_j(t), u_j(t), t] + \lambda^T_{j+1}(t) \cdot f_u[x_j(t), u_j(t), t] \tag{3.16}$$

One now wishes to determine the $\delta\bar{u}(t)$ which minimizes $\delta\hat{J}$ subject to the constraint that

$$(dE)^2 = \int_{t_0}^{t_{N_f}} \delta u^T(t) \cdot W^{-1}(t) \cdot \delta u(t)\, dt \tag{3.17}$$

where $W(t)$ is an arbitrary positive-definite (m x m) matrix chosen to improve convergence. By adjoining Eq. 3.17 to Eq. 3.15, one finds from the Euler-Lagrange equations that

$$\delta\bar{u}_j(t) = - K_J \cdot W(t) \cdot H_{u,j}(t) \tag{3.18}$$

where

$$K_J = (dE)^2 / I_{JJ} \tag{3.19}$$

and

$$I_{JJ} = \int_{t_0}^{t_{N_f}} H^T_{u,j}(t) \cdot W(t) \cdot H_{u,j}(t)\, dt \tag{3.20}$$

The procedure then is as follows:

1. With a nominal $u_j(t)$ ($u_o(t)$ selected arbitrarily for the first iteration), integrate Eq. 3.1 forward in time from t_0 to t_{N_f} to get $x_j(t)$.

2. Integrate $\lambda_{j+1}(t)$ and I_{JJ} backward from t_{N_f} to t_0 applying the boundary condition on $\lambda_{j+1}(t_{N_f})$ given by Eq. 3.13, and the discontinuities in $\lambda_{j+1}(t)$ at the t_i given by Eq. 3.14.

3. For a specified value of $(dE)^2$, compute a new nominal control from Eq. 3.18 and $u_{j+1}(t) = u_j(t) + \delta\bar{u}_j(t)$.

4. Repeat from step 1 until the state constraints violations are reduced to within a specified tolerance and no additional improvement can be achieved in $\hat{J}$.

While this method is relatively simple, requiring only (2n+1) integrations per iteration (x(t), λ(t), and I_{JJ}), it has been found to be a relatively inefficient means of solving optimal control problems. The extension to additional state constraint sets has not increased the number of integrations; however, it is extremely difficult to achieve rapid convergence using this method, since the choice of the C_j^i is extremely critical. When the weighting factors are made very large in order to bring the state constraint violations within a specified tolerance, the augmented cost $\hat{J}$ becomes very sensitive to the control perturbation. A point is reached where a small positive δu tends to decrease some of the constraint violations while increasing others. Thus the step size cannot be too large, or else $\hat{J}$ actually increases. The effect is much like moving down a narrow winding steep-sided ravine, where, if one moves in the negative gradient direction, one can only move a short distance before going up the steep slope on the opposite side of the ravine. The process of moving to the bottom of the valley, then, can be very slow since only a short distance can be traveled at each iteration.

The inefficiency of the penalty function algorithm is demonstrated by means of the double integrator example previously considered in Section 2.6. If the state constraints are given by

$$\psi^1[x(t),t] = \begin{bmatrix} x_1(t) \\ \\ t - 0.8 \end{bmatrix} = 0 \qquad (3.21)$$

and

$$\psi^2[x(t),t] = \begin{bmatrix} x_1(t) \\ \\ t - 1 \end{bmatrix} = 0 \qquad (3.22)$$

then the cost functional is augmented to become

$$J = \frac{1}{2} C_1^1 [x_1(0.8)]^2 + \frac{1}{2} C_1^2 [x_1(1)]^2 + \frac{1}{2} \int_0^1 [u(t)]^2 dt \qquad (3.23)$$

in order that the state constraints may be removed. The variable $\lambda_{j+1}(t)$ satisfies the differential equations, from Eq. 3.8, given by

$$\dot{\lambda}_{1,j+1}(t) = 0 \qquad (3.24)$$

$$\dot{\lambda}_{2,j+1}(t) = \lambda_{1,j+1}(t) \qquad (3.25)$$

and the boundary conditions, from Eqs. 3.13 and 3.14, given by

$$\lambda_{1,j+1}(1) = C_1^2 \, x_{1,j}(1) \qquad (3.26)$$

and

$$\lambda_{1,j+1}(0.8^-) = \lambda_{1,j+1}(0.8^+) + C_1^1 x_{1,j}(0.8) = C_1^2 x_{1,j}(1) + C_1^1 x_{1,j}(0.8) \qquad (3.27)$$

The optimum control perturbation at each iteration is given by

$$\delta \bar{u}_j(t) = K_J W(t) H_{u,j}(t) = - K_J \cdot [u_j(t) + \lambda_{2,j+1}(t)] \qquad (3.28)$$

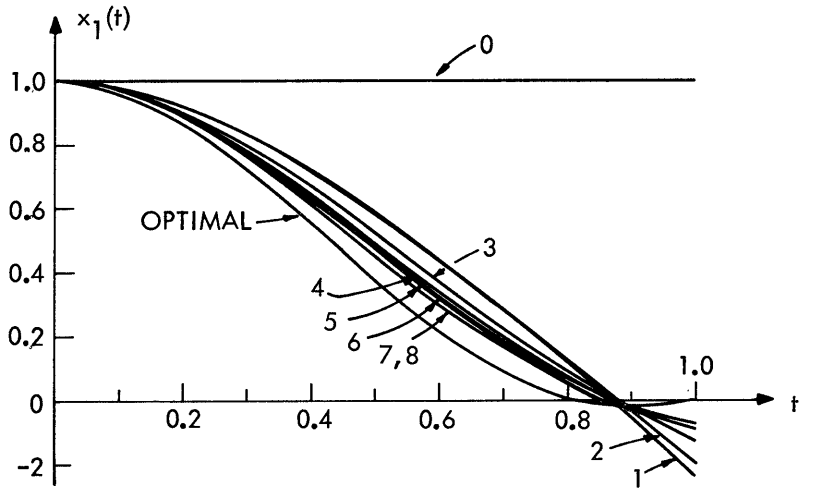
where the arbitrary weighting factor $W(t)$ has been set equal to unity. An attempt was made to introduce some second order information into the choice of the step size $(dE)^2$. To achieve this, two small steps were taken so that $\hat{J}$ could be approximated to second order terms in $(dE)^2$. With this information, the next step was taken with the $(dE)^2$ which minimized the value of $\hat{J}$ attainable at the iteration, i.e., a step is taken in the negative gradient direction to the minimum of $\hat{J}$ in that direction. The performance of the algorithm for this example is shown in Fig. 3.1 where the control $u(t)$ and the state variable $x_1(t)$ obtained for the first several iterations are shown. Each iteration consists of the three steps, the two small steps to determine the second order information followed by the larger step incorporating this information. A value of $C_1 = C_2 = 10^4$ was used, since the results of Section 2.6 (see Fig. 2.3) indicate that for this magnitude of weighting factor the constraint violations will be small. The futility of using smaller weighting factors for early iterations can be implied from Fig. 2.3 in which

the sensitivity of the optimal solutions to the weighting factors is clearly shown.

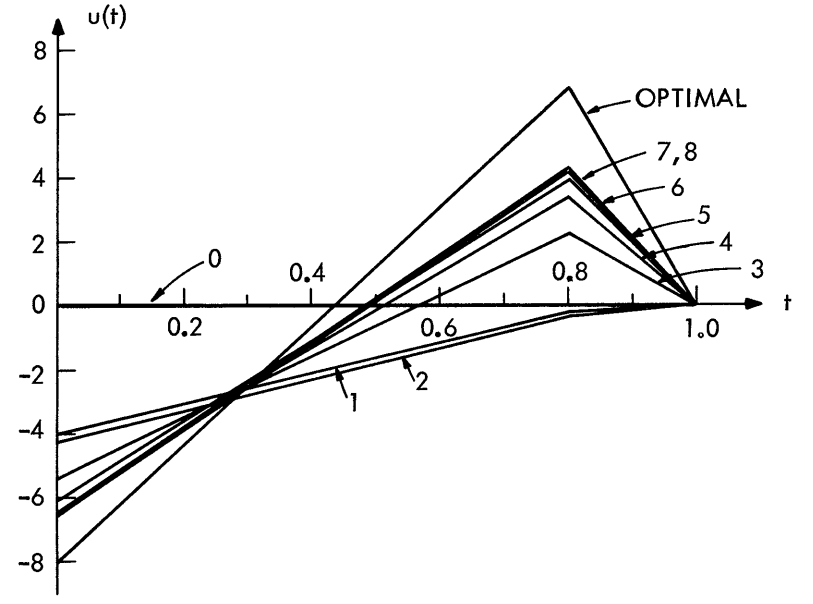
As can be seen from Fig. 3.1, the first iteration provides a large improvement in the state constraint violations at t_1 and t_2. This is because a negative δu decreases both of the errors. After the first iteration, a negative $\delta u(t)$ for $0 < t < 0.8$ further decreases the error at $t_1 = 0.8$, but it increases the error at $t_2 = 1$, and conversely, a positive $\delta u(t)$ has the opposite effect. The result is that only a small step can now be taken at each iteration without increasing the cost $\hat{J}$. By the sixth iteration, the step size is extremely small, while the "distance" to the optimal solution is still quite large. Various refinements have been tried in an attempt to speed up the convergence. These included making the weighting factors of different magnitudes, giving $W(t)$ different values over each of the two subintervals, and changing the weighting factors at each iteration. However, none of these modifications yielded a significant improvement in the rate of convergence. Therefore it is concluded that, because of the sensitivity of $\hat{J}$ to $\delta u(t)$, the penalty function approach in combination with the steepest-descent algorithm does not provide an efficient algorithm for the numerical solution of three-point BVP's.

3.3b The Influence Function Approach

While the penalty function approach treats the intermediate and terminal boundary constraints indirectly by penalizing their violations, the influence function approach treats them more directly by specifying at each step the amount by which each of the constraint violations is to be decreased. The steepest-descent algorithm for two-point optimal control problems using influence functions is reviewed in Appendix B. In the present section it is extended to general N-point optimal control problems, and its application to a two-target missile guidance problem is demonstrated. The influence function approach involves linearizing the system differential equations, the cost functional, and the terminal and intermediate state equations about a nominal solution. Through the theory of adjoint equations, the first order changes in the state constraint violations are related to a small control perturbation by means of a set of influence functions. With these relationships one computes



(a) $x_1(t)$ vs. t



(b) u(t) vs. t

Fig. 3.1 Steepest - Descent Algorithm, Penalty Function Approach - $-x_1(t)$ and u(t) vs. t

the control perturbation that yields the greatest decrease in the cost while achieving a specified decrease in the constraint violations.

Consider the state-equality constraint vector at t_i,

$$\psi^i[x(t_i), t_i] = 0 \tag{3.29}$$

At the j^{th} iteration Eq. 3.1 is integrated forward in time with the control $u_j(t)$, resulting in values of $t_{i,j}$ and $x_j(t_{i,j})$. The time $t_{i,j}$ is determined by defining one of the state constraints at t_i, say the q^{th} component of ψ^i, as a "transition" condition* θ, i.e., $t_{i,j}$ is defined as the time at which the q^{th} component of ψ^i is satisfied,

$$\psi^i_q[x_j(t_{i,j}), t_{i,j}] = \theta[x_j(t_{i,j}), t_{i,j}] = 0 \tag{3.30}$$

If a "small" perturbation is applied to the control $u_j(t)$ at the next iteration, the values of $x_{j+1}(t_{i,j+1})$ and $t_{i,j+1}$ will be perturbed, [illegible]. The changes in Eqs. 3.29 and 3.30 are given to first-order terms by

$$d\psi^i_{j+1} = \frac{\partial \psi^i}{\partial x}[x_j(t_{i,j}), t_{i,j}] \cdot dx_{j+1}(t_{i,j+1}) + \frac{\partial \psi^i}{\partial t}[x_j(t_{i,j}), t_{i,j}] \cdot dt_{i,j+1}$$

$$= [\psi^i_x \cdot \delta x_{j+1} + (\psi^i_x \cdot f + \psi^i_t) \cdot dt_{i,j+1}]_{t_{i,j}, x_j(t_{i,j})} \tag{3.31}$$

and

$$d\theta_{j+1} = \frac{\partial \theta}{\partial x}[x_j(t_{i,j}), t_{i,j}] \cdot dx_{j+1}(t_{i,j+1}) + \frac{\partial \theta}{\partial t}[x_j(t_{i,j}), t_{i,j}] \cdot dt_{i,j+1}$$

$$= [\theta_x \cdot \delta x_{j+1} + (\theta_x \cdot f + \theta_t) \cdot dt_{i,j+1}]_{t_{1,j}, x_j(t_{i,j})} \tag{3.32}$$

---

* This is analogous to the stopping condition discussed in Appendix B. In this case it defines the point of transition from one subinterval, (t_{i-1}, t_i), to the next.

Since at the next iteration $t_{i,j+1}$ is defined as the time at which θ_{j+1} is zero, then $d\theta_{j+1}$ is also zero, and the expected change in t_i can be determined from Eq. 3.32 in terms of $\delta x(t_{i,j+1})$ as*

$$dt_{i,j+1} = - \left[(\theta_x \cdot f + \theta_t)^{-1} \cdot \theta_x \cdot \delta x_{j+1} \right]_{t_{i,j}} \tag{3.33}$$

Substituting Eq. 3.33 into Eq. 3.31 one obtains

$$d\psi^i_{j+1} = \left[\psi^i_x - (\theta_x \cdot f + \theta_t)^{-1} (\psi^i_x \cdot f + \psi^i_t) \cdot \theta_x \right]_{t_{i,j}} \cdot \delta x_{j+1}(t_{i,j}) \tag{3.34}$$

Now one defines an $(n \times k_i)$ matrix $\Lambda^i(t)$ which satisfies the differential equation

$$\frac{d\Lambda^i_{j+1}}{dt} = - \frac{\partial f}{\partial x} [x_j(t), u_j(t), t] \cdot \Lambda^i_{j+1}(t) \tag{3.35}$$

and the boundary conditions

$$\Lambda^i_{j+1}(t_{i,j}) = \left[\psi^i_x - (\theta_x \cdot f + \theta_t)^{-1} (\psi^i_x \cdot f + \psi^i_t) \cdot \theta_x \right]^T_{t_{i,j}, x_j(t_{i,j})} \tag{3.36}$$

Then, since

$$\frac{d}{dt} (\Lambda^{i\,T}_{j+1} \cdot \delta x_{j+1}) = \dot{\Lambda}^{i\,T}_{j+1} \cdot \delta x_{j+1} + \Lambda^{i\,T}_{j+1} \cdot \delta \dot{x}_{j+1} = \Lambda^{i\,T} \cdot G_j(t) \cdot \delta u_j(t) \tag{3.37}$$

where

$$G_j(t) = \frac{\partial f}{\partial u} [x_j(t), u_j(t), t] \tag{3.38}$$

by combining Eqs. 3.34 through 3.37 one can write, with $\delta x(t_0)$ equal to zero, the change in ψ^i as

$$d\psi^i_{j+1} = \int_{t_0}^{t_{i,j}} \Lambda^{i\,T}_{j+1}(t) \cdot G_j(t) \cdot \delta u_{j+1}(t)\, dt \tag{3.39}$$

---

* Of course, the transition condition must be well defined for the required inverse to exist.

Thus, $\Lambda^i_{j+1}(t)$ is called the influence function for ψ^i since it specifies, at each point in time, the effect of a small perturbation in the control on that constraint violation. In view of this interpretation, it is clear that

$$\Lambda^i_{j+1}(t) = 0 \qquad \text{for} \quad t_{i,j} < t \leq t_{N_f} \tag{3.40}$$

since perturbations in the control after t_i cannot affect the state at t_i. Now, defining an $n \times (k_1 + \ldots + k_{N_f})$ matrix Λ and a k-vector ψ by

$$\Lambda = [\Lambda^1 \vdots \cdots \vdots \Lambda^{N_f}] \qquad \text{a)}$$

$$\psi = \text{col}\,(\psi^{i^T}, \ldots, \psi^{N_f^T}) \qquad \text{b)} \tag{3.41}$$

the perturbation in the control to be applied at the $(j+1)^{th}$ iteration is given, as in Eq. B.37, by

$$\delta u_{j+1}(t) = K_J W_{j+1}(t) \cdot [H_{u,j}(t) - G_j^T(t) \cdot \Lambda_{j+1}(t) \cdot I^{-1}_{\psi\psi} \cdot I_{\psi J}]$$
$$+ W_{j+1}(t) G_j^T(t) \cdot \Lambda_{j+1}(t) I^{-1}_{\psi\psi} \cdot d\psi_{j+1} \tag{3.42}$$

where $I_{\psi\psi}$ and $I_{\psi J}$ are given by Eqs. B.38 and B.39, respectively, and $H_{u,j}(t)$ is defined by Eq. B.30. The steepest-descent algorithm for a general N-point optimal control problem can be summarized as in Appendix B, where the matrix Λ is as defined by Eqs. 3.40 and 3.41a, with the boundary conditions given by Eq. 3.36. The total number of variables to be integrated, I, and stored, S, remain the same as given in Appendix B:

$$I = \frac{1}{2}(2n + k + 1)(k + 2) \tag{3.43}$$

and

$$S = m(k+2) + n \tag{3.44}$$

In this case k is the total number of state constraints imposed at the t_i, $i = 1, 2, \ldots, N_f$, i.e.,

$$k = \sum_{i=1}^{N_f} k_i \tag{3.45}$$

where k_i is the number of state constraints imposed at t_i (minus one, of course, if t_i is not explicitly specified, in which case one constraint is used as the transition condition). Thus, the number of integrations to be performed per iteration increases significantly as the number of state constraints is increased. However, not all of the variables need to be integrated over the entire interval (t_0, t_{N_f}), as noted by Eq. 3.40.

To demonstrate the improvement in convergence over the penalty function approach, the same double integrator example is considered here as in Section 3.3a. The influence functions must satisfy the differential equations

$$\dot{\Lambda}_1^i(t) = 0$$

$$\text{for} \quad i = 1,2 \qquad (3.46)$$

$$\dot{\Lambda}_2^i(t) = -\Lambda_1^i(t)$$

and the boundary conditions

$$\Lambda_1^i(t_i) = \frac{\partial \psi^i}{\partial x_1}[x(t_i)] = 1$$

$$\text{for} \quad i = 1,2 \qquad (3.47)$$

$$\Lambda_2^i(t_i) = \frac{\partial \psi^i}{\partial x_2}[x(t_i)] = 0$$

Thus, the influence functions can be integrated analytically to give

$$\Lambda_1^i(t) = 1$$

$$\text{for} \quad 0 \leq t \leq t_i \qquad (3.48)$$

$$\Lambda_2^i(t) = (t_i - t)$$

The technique for incorporating second-order information in the choice of K_J discussed in Appendix B is used here. However, for this particular example, this step is greatly simplified by virtue of the simple form of the cost functional. By expanding the cost functional to second order terms, and substituting the control perturbation given by Eq. 3.42 with $d\psi_{j+1}$ equal to zero, one finds that the minimum of the function $J(K_J)$ is attained for

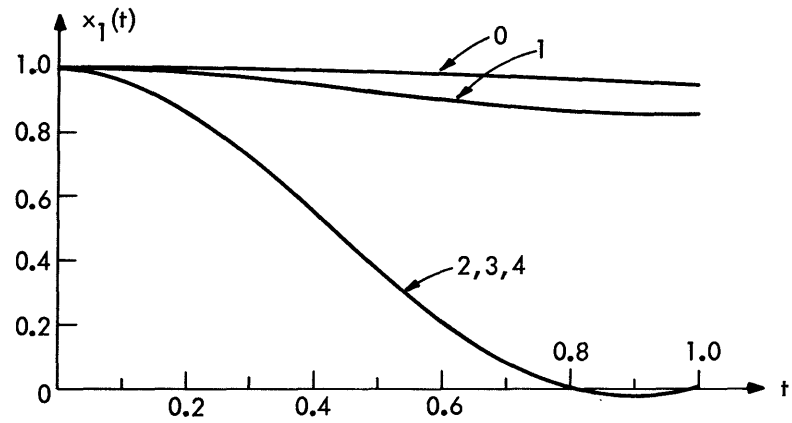
$$K_J = -\frac{\partial^2 H}{\partial u^2} = -1 \tag{3.49}$$

In addition, a relatively simple algorithm has been set up to specify the $d\psi^i$ at each iteration. At the first iteration somewhat arbitrary $d\psi^i$ are chosen (for this example one-tenth of the constraint violations due to the initial nominal control). On successive iterations, if the actual decrease in a given constraint violation on the previous iteration equals the specified $d\psi^i_j$, indicating operation in a relatively linear region, then $d\psi^i_{j+1}$ is set equal to the remaining constraint violation. On the other hand, if the actual change in the violation differs significantly from the specified change, indicating operation in a relatively nonlinear region, then $d\psi^i_{j+1}$ is set equal to only a certain percentage of the remaining violation. This percentage is a piecewise linear function of the ratio of the decrease in the constraint violation on the previous iteration to the specified $d\psi_j$, as illustrated in Fig. 3.3. The value K^i_ψ is determined for $i = 1, 2$, and $d\psi_{j+1}$ is determined from

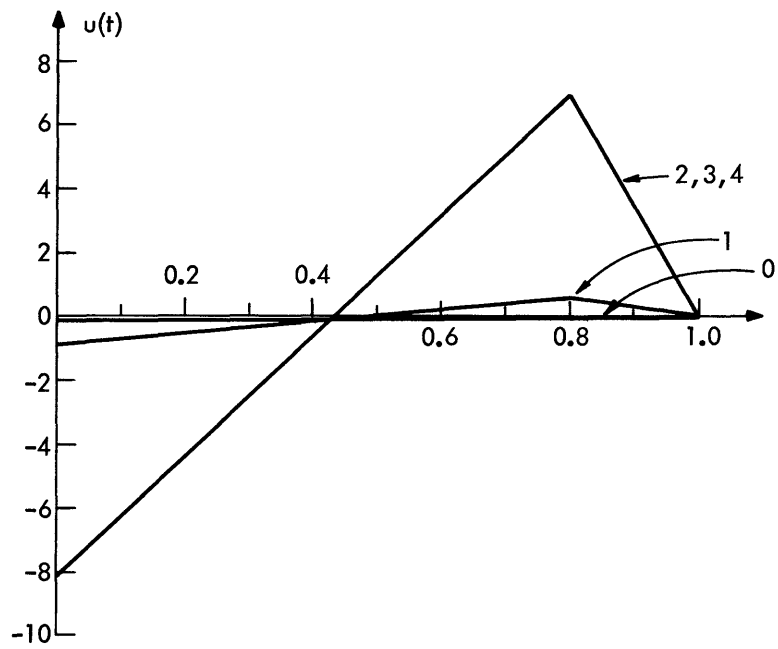
$$d\psi_{j+1} = -K_\psi \cdot d\psi_j \tag{3.50}$$

where K_ψ is the minimum of the K^i_ψ. Figure 3.2 indicates the performance of this algorithm, starting with a relatively small $u_0(t)$, and the initial conditions $x_1(0) = 1$ and $x_2(0) = 0$. Because of the linearity in the system equations and state constraints, the $d\psi^i_1$ specified for the first iteration are easily achieved, and thus the second iteration calls for the reduction of the constraint violations to zero. The quadratic nature of J enabled the minimum to be reached, for all practical purposes, at the second iteration. Thus, the direct control exerted over the state constraints enables one to obtain, for this simple example, a significant improvement over the penalty function approach.

A more rigorous test is given to the steepest-descent algorithm in applying it to the two-target missile guidance problem (which is discussed in more detail in Section 4.3) in which the system dynamics are nonlinear, the state equality constraints are functions of time, and the intermediate and terminal intercept times are unspecified.



(a) $x_1(t)$ vs. t



(b) u(t) vs. t

Fig. 3.2 Steepest - Descent Algorithm, Influence Function Approach - $-x_1(t)$ and u(t) vs. t

The system equations are

$$\begin{aligned} \dot{x}_1 &= V_m \cdot \cos x_3 \\ \dot{x}_2 &= V_m \cdot \sin x_3 \\ \dot{x}_3 &= u \end{aligned} \tag{3.51}$$

the cost functional is

$$J = \frac{1}{2}\int_{t_0}^{t_2} [u(t)]^2 \, dt \tag{3.52}$$

and the state constraints are

$$\begin{aligned} \psi_1^1[\, x(t), t] &= x_1(t) - y_1(t) = 0 \\ \psi_2^1[x(t), t] &= x_2(t) - y_2(t) = 0 \\ \psi_1^2[x(t), t] &= x_1(t) - z_1(t) = 0 \\ \psi_2^2[x(t), t] &= x_2(t) - z_2(t) = 0 \end{aligned} \tag{3.53}$$

where $y(t)$ and $z(t)$ are vector functions of time specifying the motion of the targets to be intercepted, and the intercept times t_1 and t_2 are unspecified. The constraints ψ_2^1 and ψ_2^2 are used to specify at each iteration the nominal t_1 and t_2.

As a result of the nonlinearities in Eq. 3.51, it was found that a more conservative choice of K_J and K_ψ had to be made. Unless the nominal trajectory is "close enough" to the optimal, by specifying $K_J = -1$ and $d\psi_{j+1}^i = -\psi_j^i$ one obtains a control perturbation that is too large, invalidating the linearized analysis. In addition, choosing the lowest value of K_ψ^i for K_ψ was found to slow down the convergence in certain situations. For example, if one constraint violation becomes much smaller than the other, on succeeding iterations the larger constraint violation dominates the control perturbation. The result is that the specified change in the small constraint violation is not achieved, giving a low value for K_ψ even though the specified decrease in the larger violation is achieved. In view of these considerations, the algorithm has been modified so that K_J and $d\psi$ are specified by

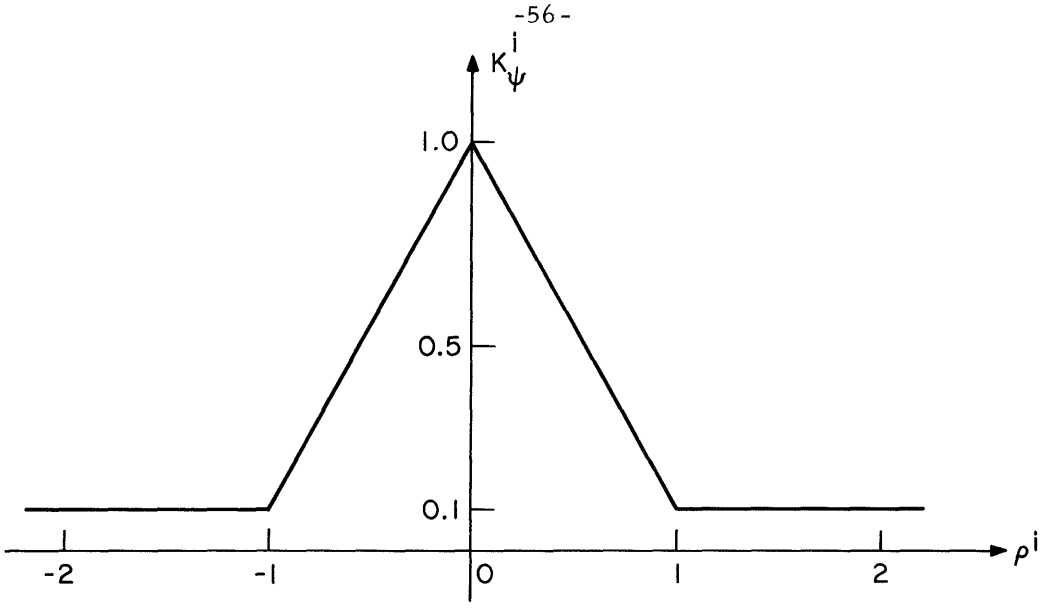


Fig. 3.3 K_ψ^i vs. ρ^i, $\rho^i = \dfrac{|\psi_j^i| - |\psi_{j-1}^i|}{d\psi_j^i}$

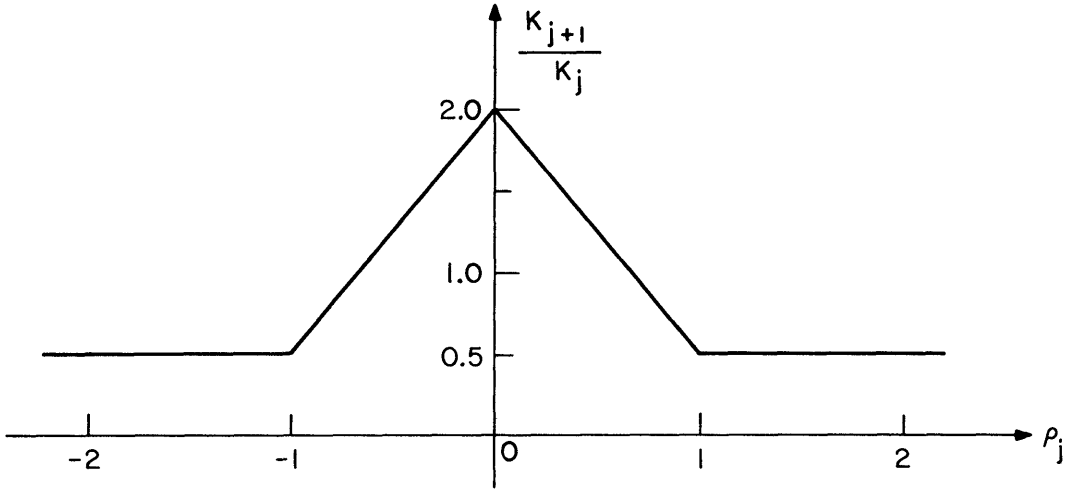


Fig. 3.4 Algorithm for Step-Size Control, $\rho_j = \dfrac{|\psi_j| - |\psi_{j-1}|}{d\psi_j}$

$$K_{J,j+1} = -K_{j+1} \qquad \text{a)}$$

and

$$d\psi_{j+1} = -K_{j+1}\,\psi_j \qquad \text{b)} \tag{3.54}$$

where K_{j+1}, $0 < K_{j+1} \leq 1$, is determined from Fig. 3.4. In computing ρ_{j+1} only the state constraint with greatest violation on the previous iteration is considered.

Figure 3.5 illustrates the convergence properties of this algorithm when the motion of the targets is given by

$$y(t) = \begin{bmatrix} 2. + 0.5\ t \\ 4. \\ 0. \end{bmatrix} \qquad \text{a)}$$

and

$$z(t) = \begin{bmatrix} 4. + 0.5\ t \\ 4. \\ 0. \end{bmatrix} \qquad \text{b)} \tag{3.55}$$

with the missile velocity normalized to unity. The control functions and the resulting missile trajectories computed for successive iterations, starting with a nominal control

$$u_0(t) = 0 \tag{3.56}$$

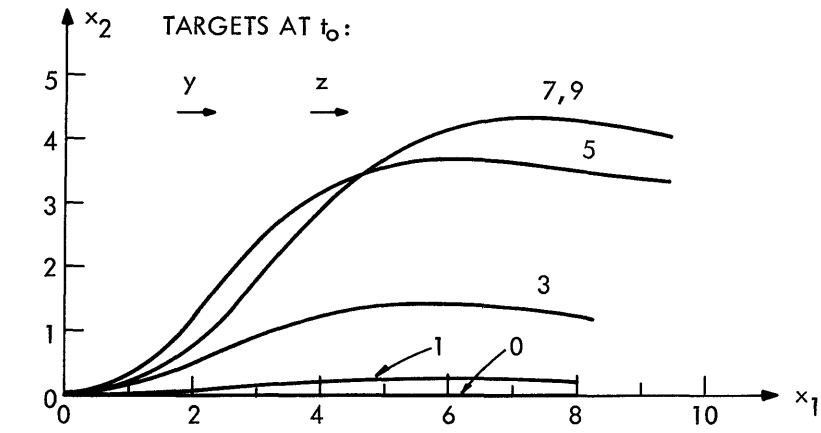
and a gain $K_0 = 0.05$, are given in Figs. 3.5a and 3.5b. The value of the cost functional, a measure of the constraint violations given by

$$D_j = 20 \log_{10}\left[\,|\psi^1_{1,j}(t_{1,j})| + |\psi^2_{1,j}(t_{2,j})|\,\right] \tag{3.57}$$

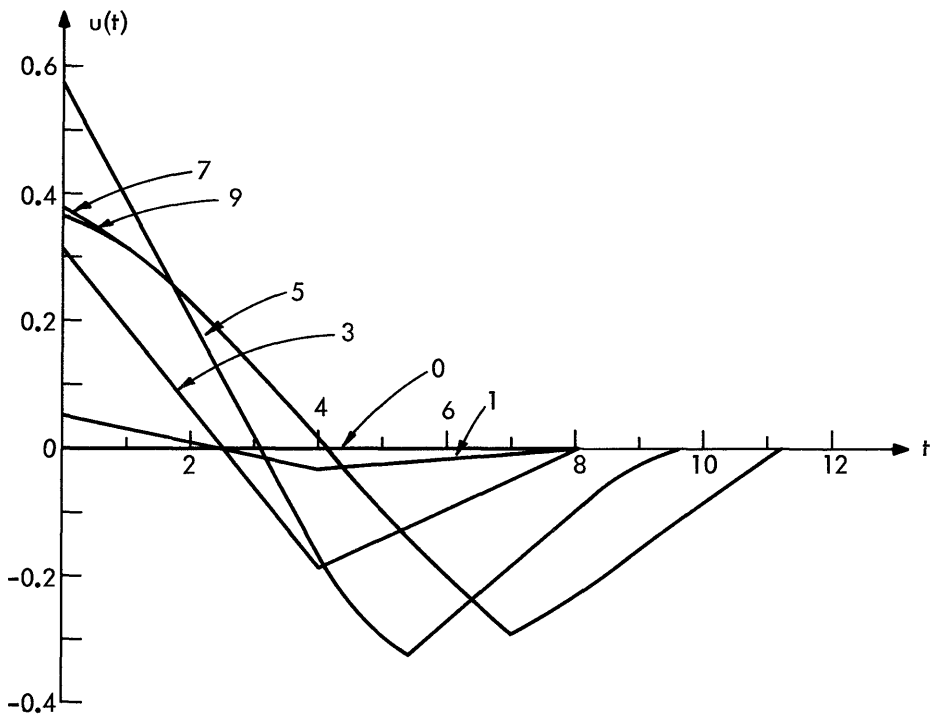
and the value of the gain K_j are given for each iteration in Figs. 3.5c through 3.5e, .Figures 3.5c through 3.5e also illustrate the convergence of the optimal solution starting with a nominal control given by

$$u_0(t) = \begin{cases} +0.25 & 0 \leq t < 6.0 \\ -0.25 & 6.0 < t < 12.0 \\ 0 & 12.0 < t \end{cases} \tag{3.58}$$

As can be seen from Fig. 3.5, the convergence for this example is obtained in 9 - 11 iterations. The algorithm is considered to have converged if on two successive iterations the constraint errors are less

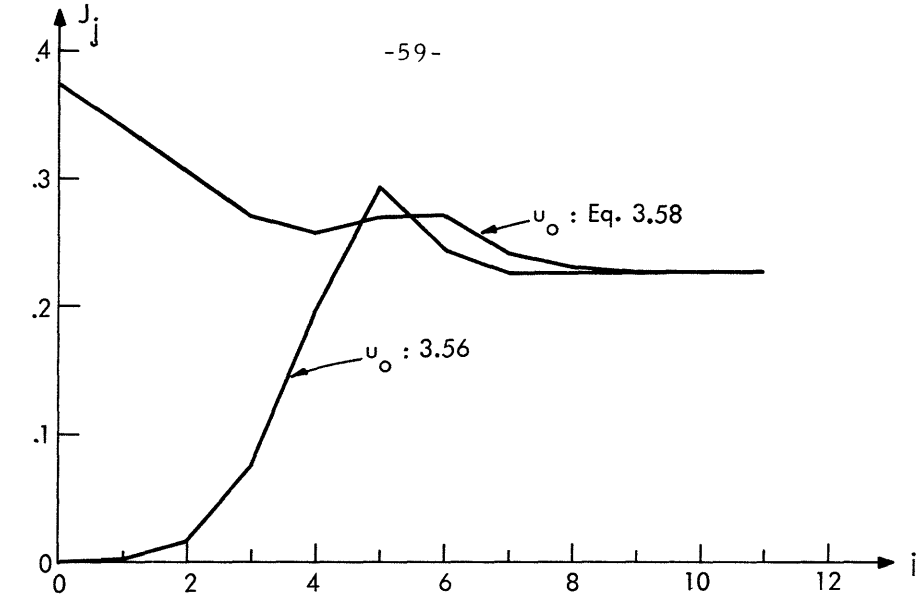


(a) $x_2(t)$ vs. $x_1(t)$

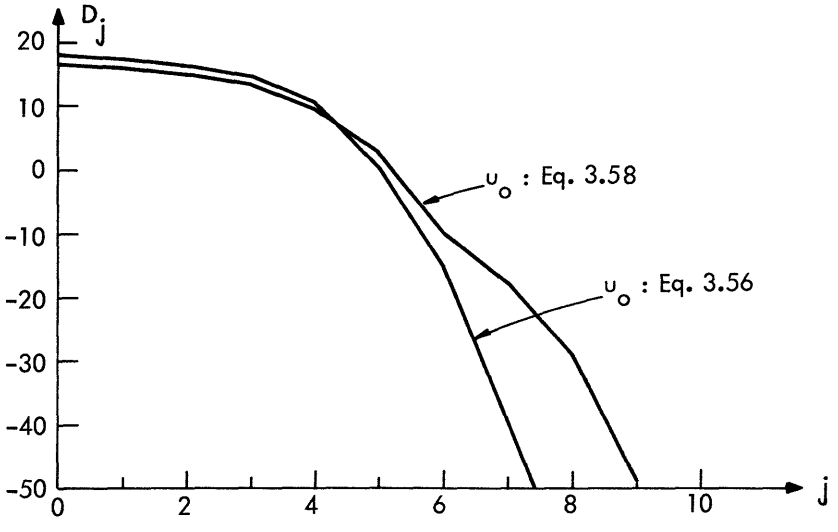


(b) u(t) vs. t

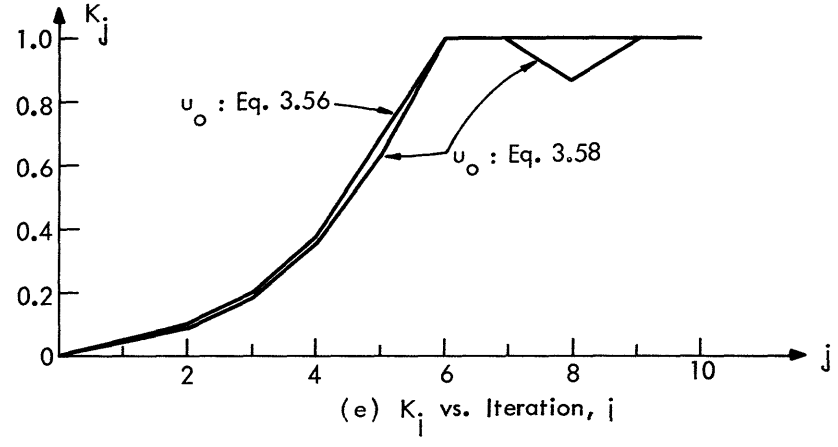
Fig. 3.5 Steepest-Descent, Two Target Intercept



(c) J vs. Iteration, j

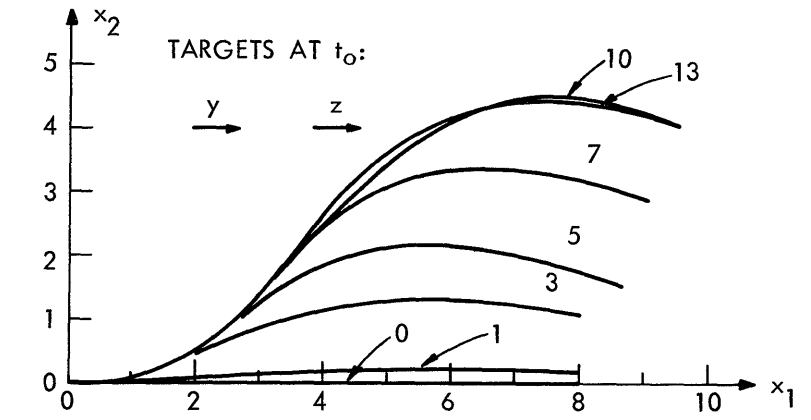


(d) D_j vs. Iteration, j

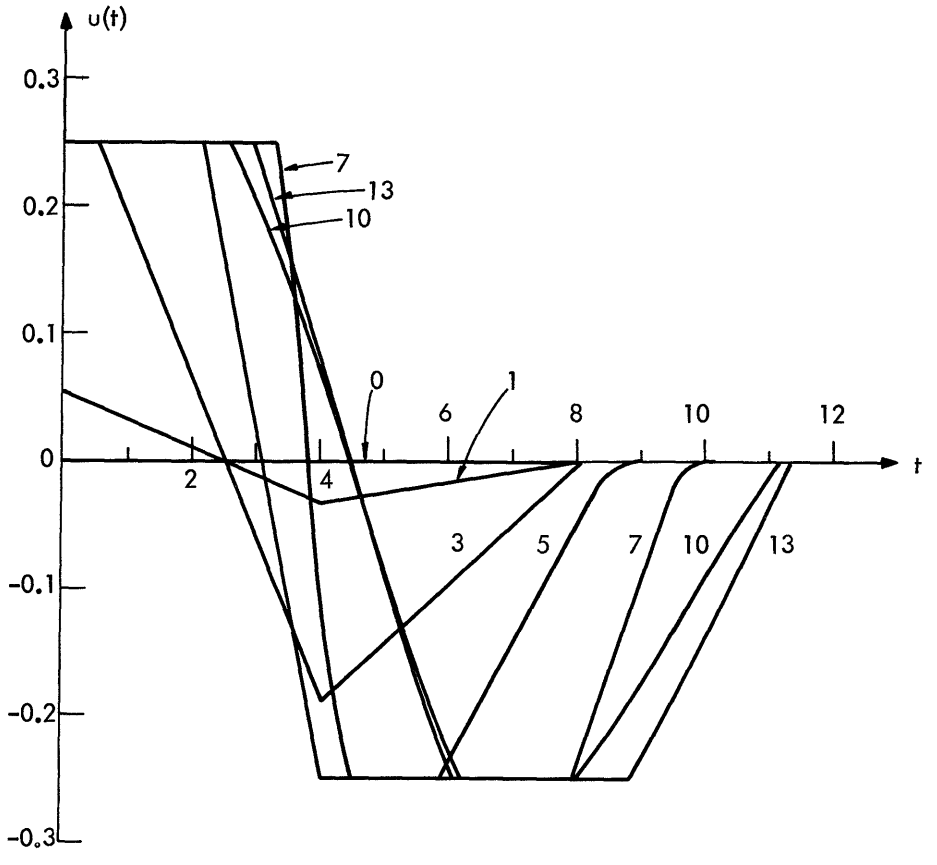


(e) K_j vs. Iteration, j

Fig. 3.5 (cont.) Steepest-Descent, Two Target Intercept

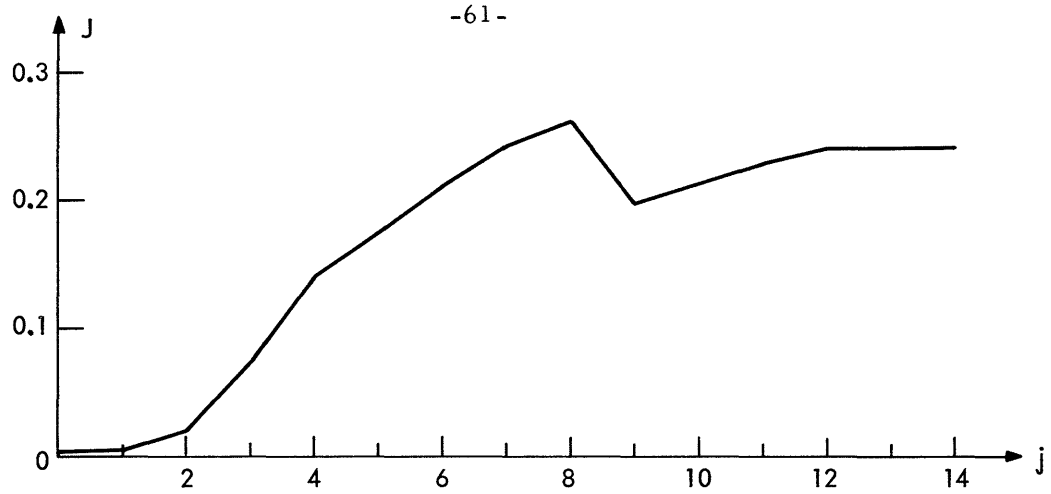


(a) $x_2(t)$ vs. $x_1(t)$

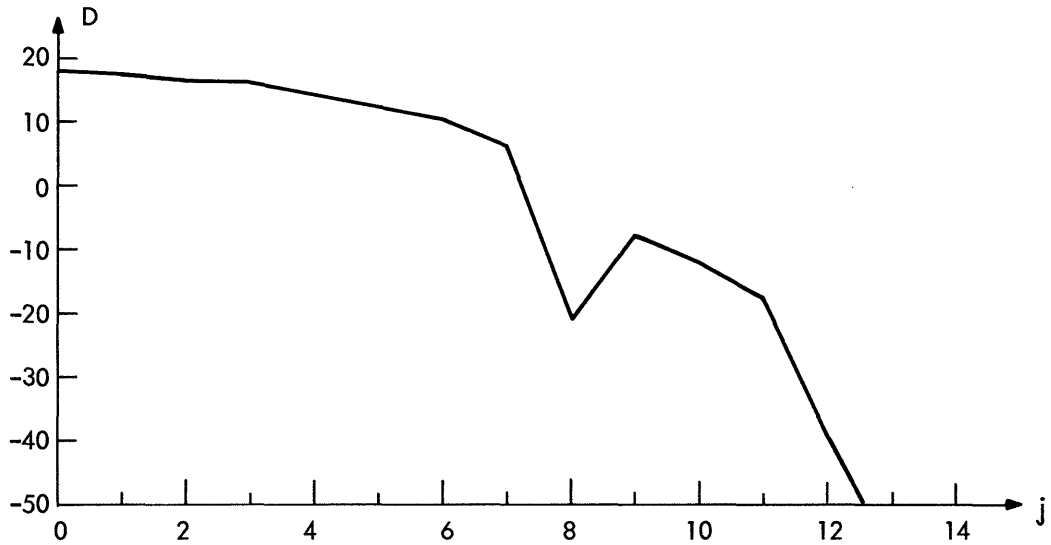


(b) u(t) vs. t

Fig. 3.6 Steepest-Descent Solution, Bounded Control Variable



(c) Cost, J, vs. Iteration j



(d) Miss Distance, $D = 20 \log_{10}\left[\left|\psi_i^1(t_1)\right| + \left|\psi_i^2(t_2)\right|\right]$ vs. Iteration, j

(e) Step-Size Gain, K_j vs. Iteration, j

Fig. 3.6 (cont'd) Steepest-Descent Solution, Bounded Control Variable

than 0.001 and the value of the cost functional has changed by less than 0.1 percent.

The steepest-descent algorithm is quite easily modified to treat bounded control problems (see Appendix B). While this research effort has been restricted to unbounded control problems, an example of a bounded control problem is given here in order to better demonstrate the application of the automatic step-size control given by Eq. 3.54 and Fig. 3.4. The modification required in order to treat inequality control constraints, as discussed in Appendix B, is to suppress the integration of $I_{\psi\psi}$ and $I_{\psi J}$ over the intervals of time when $u_j(t)$ is on the boundary of the admissible region and, of course, to limit the control at each iteration to the constrained region after adding the computed perturbation. As can be seen from Fig. 3.6 for

$$|u(t)| \leq 0.25 \qquad t_0 < t < t_2 \tag{3.59}$$

the algorithm takes several more iterations than required in the unconstrained case, since on several iterations the control perturbation must be truncated. The result is that less than the specified decrease in the constraint violations is achieved, and the automatic step-size control decreases the gain K_j at the fifth iteration.

The computer program used in obtaining the results presented in Figs. 3.5 and 3.6 is listed in Appendix D, and a flow chart of the program is given in Fig. 3.7. The integration method used for this algorithm is a second-order Runge-Kutta scheme. With a total of about 200 integration increments per iteration, the steepest-descent algorithm requires about 1.5 seconds* per iteration on the M.I.T. IBM 360/65 (under the Attached Support Processor (ASP) system). With this integration step-size, the solution given in Fig. 3.5 has been found, by comparison with the results of the Newton-Raphson algorithm (Section 3.4), to be accurate to four significant figures. The actual number of variables to be integrated is quite small. Since the system dynamics are independent

* This has been determined by subtracting from the total time charged the time required for the computer system to load the program, and then dividing by the number of iterations run.

of x_1 and x_2 the first two components of the vectors $\lambda(t), \Lambda^1(t)$, and $\Lambda^2(t)$ are merely constants at each iteration. Thus, a total of only twelve variables have to be integrated: three state variables; three components of the (2×2) matrix $I_{\psi\psi}$; $\lambda_3(t)$, $\Lambda_3^1(t)$, and $\Lambda_3^2(t)$; the two components of $I_{\psi J}$; and the cost functional, J. Furthermore, the total number of variables to be stored along the trajectory is six: the three state variables, the control, and the two components of $\Lambda^T G$.

This algorithm is used further in the next section to determine a starting guess for the Newton-Raphson algorithm, and also in Section 4.3 to compute optimal intercept trajectories for various target maneuvers and initial conditions.

3.4 THE NEWTON-RAPHSON METHOD

3.4a Introduction

The Newton-Raphson method for computing solutions to two-point control problems with fixed initial and terminal times is reviewed in Appendix C. By eliminating the control variable from the Euler-Lagrange equations (assuming, of course, that one can indeed solve explicitly for the control as a function of the state and costate variables), one can write the remaining Euler-Lagrange equations as a single 2n-dimensional vector differential equation

$$\dot{X}(t) = F(X,t) \tag{3.60}$$

where $X(t)$ and $F(X,t)$ are defined in Eqs. C.33 and C.34. The basic algorithm is to iteratively solve the linear differential equation

$$\dot{X}_{i+1}(t) = \frac{\partial F}{\partial X}(X_i,t)\cdot X_{i+1}(t) + \left[F(X_i,t) - \frac{\partial F}{\partial X}(X_i,t)\cdot X_i(t)\right] \tag{3.61}$$

applying the principle of superposition in order to guarantee that the boundary conditions at the initial and terminal times are satisfied at each iteration by $X_{i+1}(t)$. In this section this basic algorithm is modified in order to extend its application to three-point, and in general, N-point BVP's as formulated in Section 2.2. Next, control problems with free boundary times are considered. Since the basic Newton-Raphson algorithm is geared to fixed-time problems, a modification is developed which allows one to solve free-time problems by means of

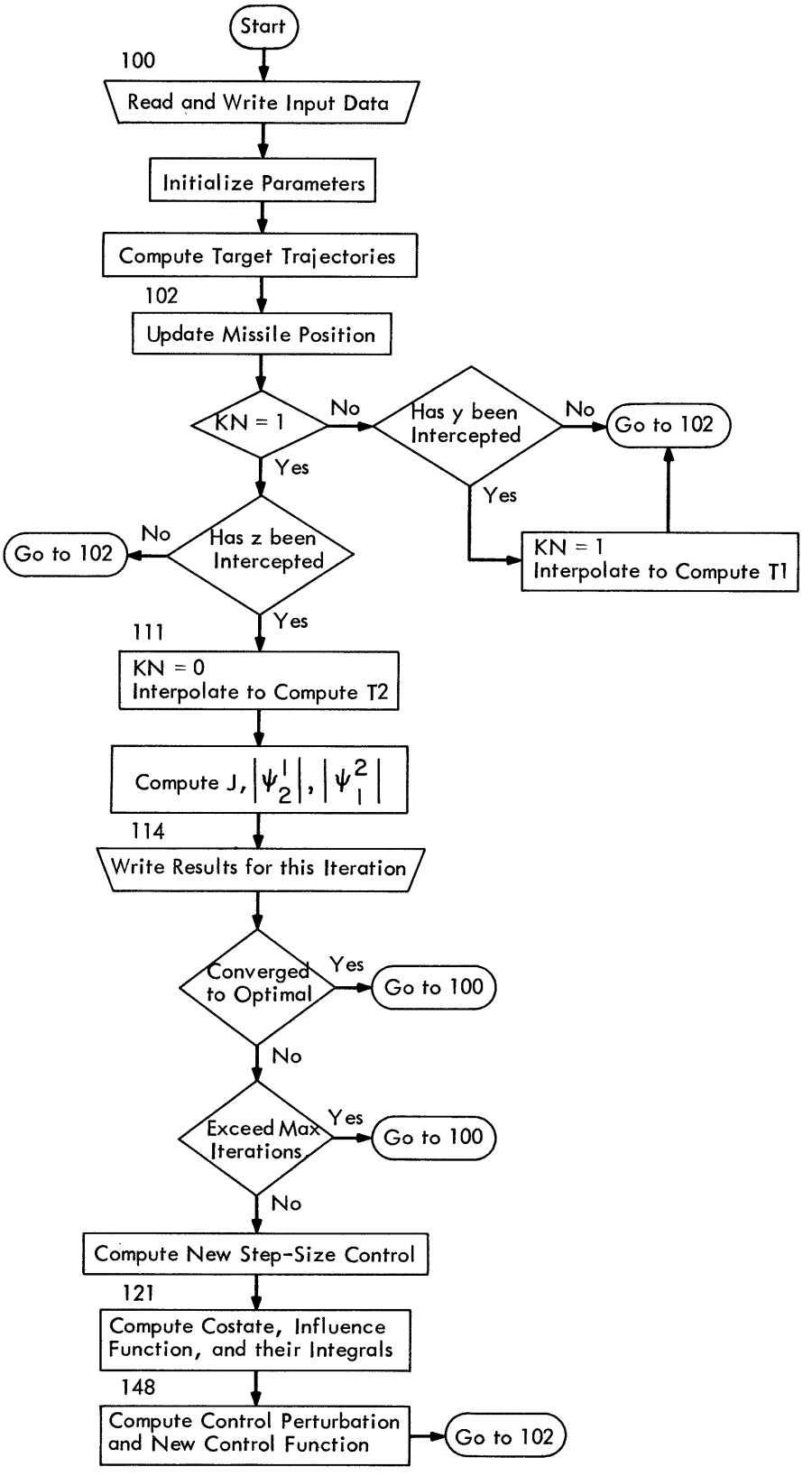


Fig. 3.7 Flow Chart for Steepest-Descent Computer Program

the solution of a sequence of fixed-time problems. In addition, a transformation of variables technique is presented which transforms a certain class of control problems from free- to fixed-time BVP's. These extensions are demonstrated by means of the two-target missile guidance problem considered in Section 3.3b.

3.4b Three-Point, Fixed-Time BVP's

To facilitate the following discussion, let us assume that the initial and terminal states and times, and the intermediate time, t_1, are explicitly specified, i.e.,

$$\psi^0[x(t),t] = \begin{bmatrix} x(t) - \zeta_0 \\ t - t_0 \end{bmatrix} = 0 \qquad a)$$

$$\psi^1[x(t),t] = \begin{bmatrix} \psi^1[x(t),t] \\ t - t_1 \end{bmatrix} = 0 \qquad b) \qquad (3.62)$$

$$\psi^2[x(t),t] = \begin{bmatrix} x(t) - \zeta_2 \\ t - t_2 \end{bmatrix} = 0 \qquad c)$$

Integrating forward in time from t_0 as in Appendix C and applying the principal of superposition, one can write*

$$X_{i+1}(t) = X^1_{i+1}(t) = \Phi^1_p(t) + \sum_{j=n+1}^{2n} C^1_j \cdot \Phi^1_j(t), \qquad t_0 \leq t < t_1 \qquad (3.63)$$

---

* The superscripts, remember, indicate the time interval of definition of the solution.

The subscripts i and i+1 indicate the index of the iteration, while p and j indicate the particular and homogeneous solutions, as discussed in Appendix B. These latter have additional implied subscripts i+1, since they must be recomputed each iteration, which are omitted for brevity.

where $\Phi_p^1(t)$ is the solution of Eq. 3.61 starting with the initial conditions

$$X(t_0) = \begin{bmatrix} \zeta_0 \\ \\ \eta_0 \end{bmatrix} \tag{3.64}$$

The $\Phi_j^1(t)$ are solutions to the homogeneous part of Eq. 3.61 with all the initial conditions equal to zero, except the j^{th} component of $X_{i+1}(t_0)$, which is set equal to one $(j = n+1, \ldots, 2n)$. Equation 3.63 is valid only over the interval $t_0 \leq t < t_1$, since at t_1 the costates undergo unspecified discontinuities. To determine a solution for $t_1 < t \leq t_2$, one can follow the same procedure used in obtaining Eq. 3.63, with the exception that the integrations are to be performed backwards in time from t_2 to t_1, giving the solution

$$X_{i+1}(t) = X_{i+1}^2(t) = \Phi_p^2(t) + \sum_{j=n+1}^{2n} C_j^2 \cdot \Phi_j^2(t), \qquad t_1 < t \leq t_2 \tag{3.65}$$

where the C_j^2 represent undetermined terminal values of the costates. Equations 3.63 and 3.65 can now be required to satisfy the $2n$ boundary conditions at t_1 given by the state constraints, Eq. 3.62, the intermediate transversality conditions, Eq. 2.18, and the continuity of state, thereby yielding $2n$ equations in $2n$ unknowns (the C_j^1 and C_j^2) to be solved. If the state constraints (Eq. 3.62) are linear, and if the penalty function, $(\phi^1[x(t_1)])$ is quadratic, then the $2n$ intermediate boundary conditions will be linear in $x(t_1)$ and $\lambda(t_1^{\pm})$. The resulting set of $2n$ linear equations can then be readily solved for the $2n$ unknowns, and the required initial and terminal values of $X_{i+1}(t)$ are given by

$$X_{i+1}(t_0) = \begin{bmatrix} \zeta_0 \\ \\ \eta_0 + C^1 \end{bmatrix} \quad ; \quad X_{i+1}(t_2) = \begin{bmatrix} \zeta_2 \\ \\ \eta_2 + C^2 \end{bmatrix} \tag{3.66}$$

where η_0 and η_2 are the arbitrary vectors used in determining $\Phi_p^1(t)$ and $\Phi_p^2(t)$, respectively. In addition, one can solve for the discontinuity in $X_{i+1}(t)$ at $t = t_1$,

$$\Delta X_{i+1}(t_1) = X^2_{i+1}(t_1^+) - X^1_{i+1}(t_1^-) = \begin{bmatrix} 0 \\ \\ \lambda^2(t_1^+) - \lambda^1(t_1^-) \end{bmatrix} \qquad (3.67)$$

by evaluating Eqs. 3.63 and 3.65 at t_1 using the appropriate C^1_j and C^2_j.

Since a great amount of computer storage space is required in order to compute $X_{i+1}(t)$ directly from Eqs. 3.63 and 3.65 once the C^1_j and C^2_j are known, one may wish to store the values of the Φ^i_p and Φ^i_j $(i=1,2;\ j=n+1,\ldots,2n)$ only at t_1, for the purpose of computing the C^1_j and C^2_j. $X_{i+1}(t)$ can then be determined by integrating Eq. 3.61, starting with the initial conditions at t_0 given by Eq. 3.66 and applying the appropriate discontinuity at t_1 as given by Eq. 3.67. Thus, only the 2n variables $x_{k,i}(t)$ and $\lambda_{k,i}(t)$ $(k=1,\ldots,n)$ need to be stored over the entire interval. However, in order to obviate the necessity of recomputing the $(2n \times 2n)$ matrix $\frac{\partial F}{\partial X}(X_i(t),t)$ and the n-vector $[F(X_i(t),t) - \frac{\partial F}{\partial X}(X_i(t),t)\cdot X_i(t)]$ each time Eq. 3.61 is to be integrated (n+1 times each iteration), one should also store their total of n(n+1) elements over the entire interval. Thus, a total of n(n+3) variables are to be stored.

The introduction of the intermediate state constraints does not increase the number of integrations to be performed (for this special case) -- one must still integrate a 2n vector differential equation (n+1) times over the entire interval, for a total of 2n(n+1) integrations. The difference is that for the three-point BVP one integrates forward in time over one subinterval and backwards in time over the remaining subinterval. Added computations are encountered, however, in determining the unknown boundary conditions, which have doubled in number through the addition of the intermediate state constraints. Thus, one must now invert a $(2n \times 2n)$ matrix, rather than an $(n \times n)$ matrix. For large n, the accuracy with which the matrix can be inverted may limit the overall performance of the algorithm. Two methods are presented here for minimizing this limitation.

The first method is to iteratively solve the matrix inversion problem. For example, if one has a set of n linear equations in n unknowns,

$$Ax = b \tag{3.68}$$

the solution is given by

$$x = A^{-1}b \tag{3.69}$$

if A^{-1} exists. If the determinant of A is zero, there is not a unique solution of Eq. 3.68. If the inversion of A cannot be performed accurately (i.e., the number of significant figures in A^{-1} is appreciably less than that in A), one obtains an inaccurate solution, x_1, from Eq. 3.69 which is related to the true solution, x, by

$$x_1 = x - \delta x \tag{3.70}$$

where

$$x_1 = \hat{A}^{-1}b \tag{3.71}$$

and $\hat{A}^{-1}$ is the computed inverse of A.
Substituting Eq. 3.70 into Eq. 3.68, one can solve for δx as

$$\delta x_1 = \hat{A}^{-1}b_1 \tag{3.72}$$

where

$$b_1 = b - Ax_1 \tag{3.73}$$

From Eqs. 3.70 and 3.72, one can then compute a more precise extimate, x_2, of x, given by

$$x_2 = x_1 + \delta x_1 \tag{3.74}$$

Since the inaccurate inverse matrix $\hat{A}^{-1}$, is used in Eq. 3.72, the perturbation δx_1 is also in error, and thus x_2 does not yet give the correct solution. However, by repeating this procedure iteratively, one can obtain a sequence of estimates $\{x_i\}$ which converges exponentially to x (if it converges at all). To demonstrate this technique, let us consider a scalar example with A = 1, b = 1, and two significant figures of accuracy in the inversion of A. While there is no difficulty in determining the inverse of a scalar to as many significant figures as A itself has, this assumption is made to keep the example as simple as possible. If the

inverse is computed to be 0.99, i.e., $\hat{A}^{-1} = 0.99$ (one percent error), then one obtains

$$x_1 = \hat{A}^{-1} b = (0.99) \cdot (1.0) = 0.99 \tag{3.75}$$

From Eqs. 3.72 and 3.73 the first correction is found to be

$$\delta x_1 = (0.99)(1.0 - 0.99) = 0.0099 \tag{3.76}$$

giving

$$x_2 = x_1 + \delta x_1 = 0.99 + 0.0099 = 0.9999 \tag{3.77}$$

Succeeding corrections are found to be

$$\delta x_i = (1 - \frac{P}{100}) (\frac{P}{100})^i \tag{3.78}$$

where P is the percentage of error in determining A^{-1}. The error at each iteration is given by

$$x - x_i = (\frac{P}{100})^i \tag{3.79}$$

Thus, if the error in determining A^{-1} is less than 100 percent, the solution of Eq. 3.68 can be determined to any desired accuracy, since the algorithm converges exponentially.

A second method to bypass the limitations of the matrix inversion is to make a judicious choice of the arbitrary initial and terminal costates, η_0 and η_2, used in determining the particular solutions, $\Phi_p^1(t)$ and $\Phi_p^2(t)$. If these variables are set to zero, one computes the initial and terminal costates directly via C_j^1 and C_j^2, $j = n+1, \ldots, 2n$. Now, if these quantities can be computed to only s_0 significant figures, then the Newton-Raphson algorithm is limited to that accuracy (the actual accuracy for all t may be even less when one integrates Eq. 3.61 to obtain $X_{i+1}(t)$, if the solution is very sensitive to slight perturbations in initial conditions). However, if one chooses η_0 and η_2 to be the initial and terminal costates $\lambda_i(t_0)$ and $\lambda_i(t_2)$, respectively, that were computed on the previous iteration, then the C_j^1 and C_j^2 constitute perturbations of these quantities at each iteration. Therefore, if $\lambda_i(t_0)$ and $\lambda_i(t_2)$ have been determined to s_0 correct significant

figures, and if the $C^1_{j,i+1}$ and $C^2_{j,i+1}$ can also be determined to s_0 significant figures, then one obtains $\lambda_{i+1}(t_0)$ and $\lambda_{i+1}(t_2)$ to $2 \cdot s_0$ significant figures. This second method is the one used in the computer simulations discussed in Section 3.4e.

If the initial and terminal state constraints are not of the "simple" form assumed in Eqs. 3.62a and 3.62c, the computational requirements are increased, as discussed in Section C.3, with respect to both the number of integrations per iteration and the size of the matrix to be inverted. For each initial (or terminal) condition that is not explicitly specified, one must compute the transition vector, $\Phi_j(t)$, as is done in the above discussion for the unknown initial and terminal costates. As a worst case, if none of the initial or terminal states are specified explicitly (or left entirely free), and if the state constraints are nonlinear functions, one may have to compute the entire transition matrix $\Phi(t,t_0)$, and solve a set of 4n equations (the intermediate state constraints and transversality conditions) for 4n unknowns (2n initial and 2n terminal states and costates). The initial and terminal transversality conditions can be used to reduce the number of unknowns in some problems, thus reducing the computational requirements. For example, if $n=3$, $k_0=1$, and

$$\psi^0[x(t_0),t_0] = x_1(t_0) + a_1 \cdot x_2(t_0) + a_2 \cdot x_3(t_0) \tag{3.80}$$

then the initial transversality conditions give

$$\lambda_1(t_0) = a_1 \cdot \lambda_2(t_0) = a_2 \cdot \lambda_3(t_0) \tag{3.81}$$

Since none of the initial states are specified, one must determine a transition vector, $\Phi^1_j(t)$, $j = 1,2,3$, for each of them. However, by virtue of Eq. 3.80, only one transition vector, $\Phi^1_4(t)$, must be determined for the initial costates, using the initial condition

$$\Phi^1_4(t_0) = \text{col}\,(0,0,0,1,a_1,a_2) \tag{3.82}$$

For this example, while none of the initial states are known, the number of transition vectors to be determined, and thus the size of the matrix to be inverted, has increased by only one over the number required if all the states are explicitly specified.

3.4c An Alternate Approach

One may, in general, reduce the number of unknowns to be computed, and thus the size of the matrix to be inverted, at the expense of added integrations. While the desirability of increasing the number of integrations may be questionable for three-point BVP's, when additional intermediate state constraint sets are introduced, this alternate approach may become more efficient. If the number of state constraints at t_1 is k_1, there are k_1 independent discontinuities on the costates at t_1, e.g., if the state constraints at $t = t_1$ are of the form given by Eq. 3.79,

$$\psi^1[x(t_1),t_1] = x_1(t_1) + a_2 \cdot x_2(t_1) + a_3 \cdot x_3(t_1) = 0 \qquad (3.83)$$

then the intermediate transversality conditions are given by

$$\Delta\lambda_1(t_1) = a_2 \cdot \Delta\lambda_2(t_1) = a_3 \cdot \Delta\lambda_3(t_1) \qquad (3.84)$$

and therefore, there is only one independent discontinuity at t_1. One proceeds, as before (assuming simple initial and terminal constraints), to determine $\Phi_p^1(t)$ and $\Phi_j^1(t)$, $j = n+1,\ldots,2n$, but now over the entire interval (t_0,t_2), requiring that the costates be continuous at t_1. In addition, one determines the effect, $\Psi_k^2(t)$, of the discontinuities in the costates at t_1 by integrating the homogeneous part of Eq. 3.61, over the interval (t_1,t_2), k_1 times (once for each of the independent discontinuities). The general solution for $X_{i+1}(t)$ is then given by

$$X_{i+1}(t) = \Phi_p^1(t) + \sum_{j=n+1}^{2n} C_j^1 \cdot \Phi_j^1(t) + \sum_{k=1}^{k_1} D_k^2 \cdot \Psi_k^2(t) \qquad (3.85)$$

where

$$\Psi_k^2(t) = 0 \qquad (3.86)$$

for $t_0 < t < t_1$ and $k = 1,\ldots,k_1$.

One now has only $(n+k_1)$ unknowns to be determined (C_j^1 for $j = n+1,\ldots,2n$, and D_k^2 for $k = 1,\ldots,k_1$) in order to satisfy the $(n+k_1)$ state constraints imposed at t_1 and t_2. However, the computation of $\Phi_p^1(t)$ and

and $\Phi_j^1(t)$ involves integrating the 2n vector, $X(t)$, (n+1) times over the entire interval (t_0, t_2) (as much computation as previously required in obtaining $\Phi_p^1(t)$, $\Phi_p^2(t)$, and the $\Phi_j^1(t)$ and $\Phi_j^2(t)$ over their respective subintervals), and the computation of the $\Psi_k^2(t)$ involves an additional k_1 integrations of $X(t)$ over the interval (t_1, t_2). For the example given by Eqs. 3.83 and 3.84, the computation of $\Psi_1^2(t)$ involves integrating the homogeneous part of Eq. 3.61 over the interval (t_1, t_2) starting with the "initial" condition

$$X(t_1) = \text{col}\,(0, 0, 0, 1, a_2, a_3) \tag{3.87}$$

3.4d N-Point BVP's

The extension of the above discussion to control problems with additional intermediate state constraint sets is straightforward. Consider a general (N+1)-point optimal control problem. For the initial and final subintervals, (t_0, t_1) and (t_{N-1}, t_N), one can determine solutions for $X(t)$ in the form of Eq. 3.63, requiring (n+1) integrations of the 2n vector, $X(t)$, and involving 2n unknowns, C_j^1 and C_j^N, for $j = n+1, \ldots, 2n$. For each of the interior intervals, however, one must perform additional integrations to determine the effect of the unknown values of the state variables at the beginning of each of these intervals. Therefore over each interior subinterval one must obtain a particular solution, $\Phi_p^k(t)$, and up to 2n homogeneous solutions, Φ_j^k, for $j = 1, \ldots, 2n$ and $k = 2, \ldots, N-2$, writing the solution as

$$X_{i+1}^k(t) = \Phi_p^k(t) + \sum_{j=1}^{2n} C_j^k \cdot \Phi_j^k(t) \qquad k = 2, \ldots, N-2 \tag{3.88}$$

Of course, if any of the intermediate constraints are of "simple" form, i.e., of the form $x_m(t_k) = \zeta_m$ where x_m is the m^{th} component of x, then the number of unknown initial values for that subinterval can be reduced by including these initial conditions in the particular solution, $\Phi_p^k(t)$. But, if the constraints are of a more general form, as in Eq. 3.83, then none of the state variables may be known at t_k, and one may

have to determine the entire $(2n \times 2n)$ transition matrix over each interior subinterval. Thus, while one intermediate boundary point adds n unknowns to be determined, each additional intermediate boundary point may increase the number of unknowns to be determined by as many as 2n. In addition, the number of integrations required has also been increased since as many as 2n homogeneous solutions may have to be determined over the interior subintervals (as opposed to only n homogeneous solutions over the first and last intervals).

As more intermediate state constraint sets are imposed, the alternate method discussed in Section 3.4c may become more efficient. For that method, the number of unknowns to be determined always equals the total number of state constraints to be satisfied at the intermediate and terminal times, and always is less than (or perhaps equal to) the number of unknowns to be determined by the original method presented in Section 3.4b. Which method requires less overall computation for a given problem depends on the number and nature of the intermediate state constraints and the spacing of the intermediate times t_i.

3.4e Free-Time Problems

The discussions in Sections 3.4b - 3.4d have been limited to fixed-time optimal control problems, i.e., problems in which the times, t_i, at which the state equality constraints are imposed are explicitly specified. McGill and Kenneth[33] applied the Newton-Raphson algorithm to a minimum time orbit transfer problem by solving instead a sequence of fixed time, maximum orbital radius problems, which converged iteratively on the optimal terminal time required in order that the maximum orbital radius equal the specified terminal radius. In this section, the Newton-Raphson algorithm is extended to free-time, minimum energy problems by a similar technique of solving a sequence of fixed-time problems. For a certain class of problems, by making an appropriate change of variables, one need solve only a single fixed-time problem.

To facilitate the following discussion, only two- and three-point BVP's are considered and it is assumed that the initial time is specified explicitly. One can consider a two-point free terminal time

problem as that of determining the value $\bar{t}_2$ such that the optimal cost functional, considered as a function of a fixed terminal time, is minimized, i.e.,

$$\bar{J}(\bar{t}_2) = \min_{t_2 \in (t_0, \infty)} \bar{J}(t_2) \tag{3.89}$$

This statement is to be understood as follows: for each fixed terminal time, t_2, J has a minimum value $\bar{J}(t_2)$ -- find that value $\bar{t}_2$, which minimizes the function $\bar{J}(t_2)$.

For three-point BVP's one must minimize a function of two variables

$$\bar{J}(\bar{t}_1, \bar{t}_2) = \min \bar{J}(t_1, t_2) \quad t_1 \in (t_0, \infty) \quad t_2 \in (t_0, \infty) \quad t_2 > t_1 \tag{3.90}$$

Thus, if one assumes that $\bar{J}(t_1, t_2)$ has continuous first derivatives with respect to both arguments, one must find the times $\bar{t}_1$ and $\bar{t}_2$ such that

$$\frac{\partial \bar{J}}{\partial t_1}(\bar{t}_1, \bar{t}_2) = 0 \tag{3.91}$$

$$\frac{\partial \bar{J}}{\partial t_2}(\bar{t}_1, \bar{t}_2) = 0 \tag{3.92}$$

In evaluating Eqs. 3.91 and 3.92, one must take into account the fact that the state constraint surfaces may move as the terminal and intermediate times are changed. From the expressions for the general first variations of J and the ψ^i, Eqs. 2.12 and 2.13, one finds that

$$\frac{\partial \bar{J}}{\partial t_1}(t_1, t_2) = -\Delta \bar{H}(t_1) + \frac{\partial \bar{\phi}^1}{\partial t}(t_1) - \sum_{i=1}^{k_1} \gamma_i \frac{\partial \bar{\psi}^1_i}{\partial t}(t_1) = 0 \tag{3.93}$$

and

$$\frac{\partial \bar{J}}{\partial t_2}(t_1,t_2) = \bar{H}(t_2) + \frac{\partial \bar{\phi}^2}{\partial t} + \sum_{i=1}^{k_2} \nu_i \frac{\partial \bar{\psi}_i^2}{\partial t}(t_2) = 0 \qquad (3.94)$$

Note that Eqs. 3.93 and 3.94 are identical to Eqs. 2.19 and 2.17, respectively, which represent the necessary conditions imposed on the optimal trajectory resulting from the freedom in the variations δt_1 and δt_2. Only Eq. 3.94 is of concern in a two-point BVP, where t_2 is the terminal time. In order to solve Eq. 3.90, or alternately Eqs. 3.93 and 3.94, one must solve a sequence of fixed-time problems. This approach is not valid, of course, in minimum time problems, since it is meaningless to pose a minimum time problem with a fixed terminal time. In such cases, one must transform the problem, either by changing the cost functional and terminal constraints, as was done by McGill and Kenneth, or by a transformation of variables (if applicable) as discussed in Section 3.4f. Equation 3.90 represents a parameter optimization problem, for which one can use any of the methods, as for example the steepest-descent or Newton methods, available for such minimization problems. If one uses a steepest-descent technique, the gradient of the cost with respect to t_1 and t_2 can be obtained directly from Eqs. 3.93 and 3.94. A Newton method, on the other hand, solves for the roots of Eqs. 3.93 and 3.94, and requires information on the second derivative of $\bar{J}$ with respect to t_1 and t_2.

Since the information on the second derivatives of $\bar{J}$ with respect to t_1 and t_2 is not directly available, one must approximate them by

$$\bar{J}_{ij}(t_1,t_2) = \frac{\partial^2 \bar{J}}{\partial t_i \partial t_j}(t_1,t_2) = \frac{\frac{\partial \bar{J}}{\partial t_i}(t_i,t_j+\delta t_j) - \frac{\partial \bar{J}}{\partial t_i}(t_i,t_j)}{\delta t_j} \qquad (3.95)$$

In order to determine all the required second derivatives one must solve a total of three fixed-point problems -- for (t_1,t_2), $(t_1+\delta t_1,t_2)$, and $(t_1+\delta t_1,t_2+\delta t_2)$. With these second derivatives one can apply the

Newton method to compute a new t_1' and t_2' given by*

$$\begin{bmatrix} t_1' \\ t_2' \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} - \begin{bmatrix} \bar{J}_{11} & \bar{J}_{12} \\ \bar{J}_{12} & \bar{J}_{22} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \frac{\partial \bar{J}}{\partial t_1}(t_1, t_2) \\ \frac{\partial \bar{J}}{\partial t_2}(t_1, t_2) \end{bmatrix} \tag{3.96}$$

One then repeats this procedure iteratively until the optimal $\bar{t}_1$ and $\bar{t}_2$ are converged upon (i.e., if the method converges and, in fact, converges to the minimum of J rather than a maximum). Since this method of computing the second derivatives is not exact, this algorithm for converging on the optimal $\bar{t}_1$ and $\bar{t}_2$ should, perhaps, more appropriately be called a method of secants.

The above technique for solving two- and three-point optimal control problems is demonstrated by means of the two-target missile guidance problem discussed in Sections 3.3b and 4.3. The system dynamics, cost functional, and state constraints, are given by Eqs. 3.51 - 3.53. After eliminating the control variable from the Euler-Lagrange equations, Eq. 3.60 becomes

$$\dot{X}(t) = \begin{bmatrix} V_m \cos x_3 \\ V_m \sin x_3 \\ -x_6 \\ 0 \\ 0 \\ V_m \cdot (x_4 \cdot \sin x_3 - x_5 \cdot \cos x_3) \end{bmatrix} \tag{3.97}$$

where $x_4 = \lambda_1, x_5 = \lambda_2$, and $x_6 = \lambda_3$. The intermediate and terminal transversality conditions become

$$\Delta\bar{x}_4(t_1) = \upsilon_1 \qquad \text{a)}$$

$$\Delta\bar{x}_5(t_1) = \upsilon_2 \qquad \text{b)} \tag{3.98}$$

$$\Delta\bar{x}_6(t_1) = 0 \qquad \text{c)}$$

(Eq. 3.98 continued on next page)

* See Appendix C.

$$\frac{\partial \bar{J}}{\partial t_1} = -\Delta\bar{H}(t_1) - \upsilon_1[-\dot{y}_i(t_1)] - \upsilon_2\cdot[-\dot{y}_2(t_1)] = 0 \qquad \text{d) (Eq. 3.98 contd.)}$$

and

$$\begin{aligned} \bar{x}_4(t_2) &= \gamma_1 && \text{a)} \\ \bar{x}_5(t_2) &= \gamma_2 && \text{b)} \\ \bar{x}_6(t_2) &= 0 && \text{c)} \end{aligned} \qquad (3.99)$$

$$\frac{\partial \bar{J}}{\partial t_2} = \bar{H}(t_2) + \gamma_1\cdot[-\dot{z}_1(t_2)] + \gamma_2\cdot[-\dot{z}_2(t_2)] = 0 \qquad \text{d)}$$

where υ_1, υ_2, γ_1, γ_2 are unspecified constants.

The linear differential equation to be solved at each iteration, Eq. 3.61, becomes (setting $V_m = 1$ for simplicity)

$$\dot{x}_{1,i+1} = [-\sin x_{3,i}]\cdot x_{3,i+1} + [\cos x_{3,i} + x_{3,i}\cdot \sin x_{3,i}] \qquad \text{a)}$$

$$\dot{x}_{2,i+1} = [\cos x_{3,i}]\cdot x_{3,i+1} + [\sin x_{3,i} - x_{3,i}\cdot \cos x_{3,i}] \qquad \text{b)}$$

$$\dot{x}_{3,i+1} = -x_{6,i+1} \qquad \text{c)} \qquad (3.100)$$

$$\dot{x}_{4,i+1} = 0 \qquad \text{d)}$$

$$\dot{x}_{5,i+1} = 0 \qquad \text{e)}$$

$$\dot{x}_{6,i+1} = [x_{4,i}\cdot\cos x_{3,i} + x_{5,i}\cdot\sin x_{3,i}]\cdot x_{3,i+1} + [\sin x_{3,i}]\cdot x_{4,i+1}$$

$$-[\cos x_{3,i}]\cdot x_{5,i+1} - [(x_{4,i}\cos x_{3,i} + x_{5,i}\cdot\sin x_{3,i})\cdot x_{3,i}] \qquad \text{f)}$$

Let us consider first a single target guidance problem, where

$$x(t_o) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \qquad z(t) = \begin{bmatrix} 1+.5t \\ 5 \\ 0 \end{bmatrix} \qquad (3.101)$$

An initial guess is made of

$$x_{3,0}(t) = 1-(0.3)\cdot\left(\frac{t}{t_2}\right) \quad \text{a)}$$

$$x_{6,0}(t) = (0.1)\left(1.0-\frac{t}{t_2}\right) \quad \text{b)} \qquad (3.102)$$

$$x_{4,0}(t) = x_{5,0}(t) = 0 \quad \text{c)}$$

$$t_2 = 6.6 \quad \text{d)}$$

Thus the initial guess on $x_3(t)$ and $x_6(t)$ are linear functions of time, while since one has little physical intuition on the behavior of the co-states, they are initially set to zero. The state variables $x_1(t)$ and $x_2(t)$ need not be guessed, since they do not appear in the right side of Eq. 3.100. The algorithm is determined to have converged when a measure of the distance between the solutions at two successive iterations, given by

$$\rho(X_{i+1}, X_i) = \int_{t_0}^{t_2} \sum_{j=1}^{2n} \left| x_{j,i+1}(t) - x_{j,i}(t) \right| dt \qquad (3.103)$$

is less than a specified tolerance, set at 0.001 for this example. With this starting guess and error measure, the Newton-Raphson algorithm converges for this fixed-time problem in four iterations. The solution obtained is then used as a starting solution for a new problem with a terminal time of $t_2' = t_2 + \delta t_2 = 6.601$. Figures 3.8 and 3.9 illustrate the results of the solution of a sequence of such fixed time problems where the method of secants is used to determine each new terminal time (after the first two). The cost, $\bar{J}$, and the partial derivative of $\bar{J}$ with respect to t_2, $\frac{\partial \bar{J}}{\partial t_2}$, are plotted as functions of the fixed terminal time in Fig. 3.8. The straight-line approximation to the curve $\frac{\partial \bar{J}}{\partial t_2}$ represents the secant method used in computing $\frac{\partial^2 \bar{J}}{\partial t_2^2}$. (The smooth curves representing $\bar{J}(t_2)$ and $\frac{\partial \bar{J}}{\partial t_2}(t_2)$ are only approximations to the actual undetermined functions.) As can be seen from Fig. 3.8, the secant method actually converges faster than the Newton method

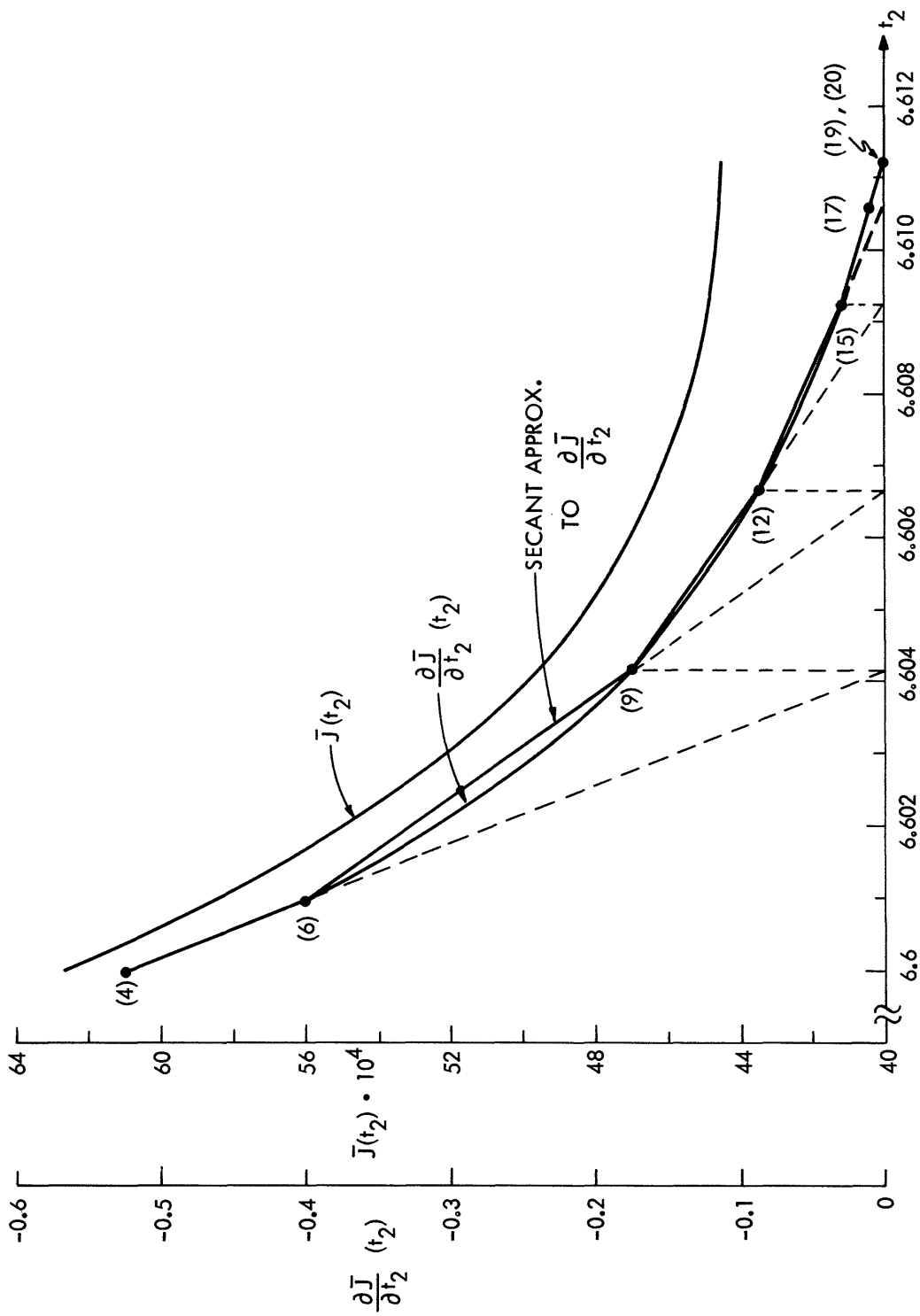
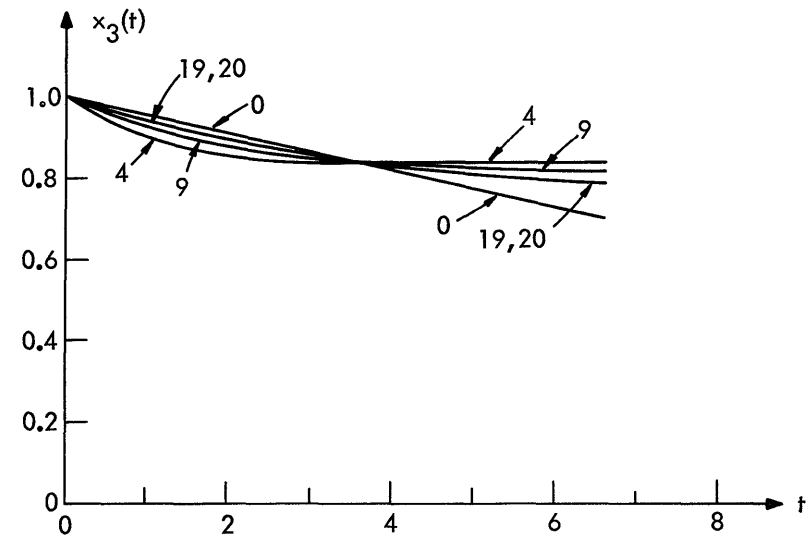
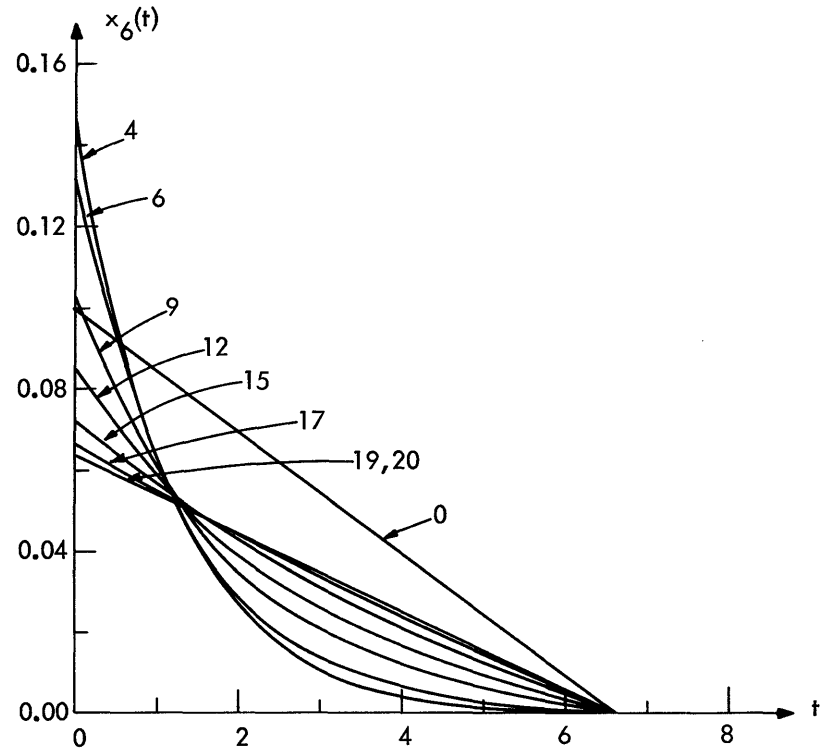


Fig. 3.8 Newton-Raphson Algorithm—$\bar{J}$ and $\frac{\partial \bar{J}}{\partial t_2}$ vs. t_2



(a) $\bar{x}_3(t)$ vs. t



(b) $\bar{x}_6(t)$ vs. t

Fig. 3.9 Single-Target Intercept Problem—Solution of Sequence of Fixed-Time Problems

(method of tangents) would if $\frac{\partial^2 \bar{J}}{\partial t_2^2}$ could be computed exactly. The numbers in parentheses at the different $t_{2,j}$ are the number of the iteration at which the algorithm converged to the solution for that fixed-time problem. Thus, while each fixed-time problem requires only a few iterations to converge, since a good starting solution is available from the solution to the previous problem, a total of 20 iterations are required in order to solve the free-time problem, and an optimal terminal time $\bar{t}_1 = 6.6112$ is obtained. Table 3.1 summarizes the results of the algorithm at each iteration, giving the error measure ρ, the terminal time t_2, the cost $\bar{J}$, and its partial derivative $\frac{\partial \bar{J}}{\partial t_2}$. The optimal solutions for $\bar{x}_3(t)$ and $\bar{x}_6(t)$ are given for the sequence of fixed-time problem in Fig. 3.9. The number of iterations required for convergence to the optimal solution is dependent on the degree of convergence required for each fixed-time problem, as specified by ρ. By increasing ρ an order of magnitude, assuming this will not appreciably affect the results of subsequent fixed-time problems, one can reduce the number of iterations required to 16.

Several significant observations can be made with respect to this example. First of all, while the results of Table 3.1 and Figs. 3.8 and 3.9 indicate a straightforward application to the Newton-Raphson algorithm, some difficulty was encountered in selecting an initial starting solution such that the algorithm could converge for the initial fixed-time problem. The algorithm seemed to be particularly sensitive to the choice of $x_6(t)$. Secondly, a guess of the terminal time of $t_2 = 7.0$ was originally made, for which $\frac{\partial \bar{J}}{\partial t_2}$ is positive and $\frac{\partial^2 \bar{J}}{\partial t_2^2}$ is negative. The Newton (or secant) method for iterating on the terminal time then specifies a positive δt_2, away from the optimal solution $\bar{t}_2 = 6.6112$. The algorithm, from this starting point, converges to a relative maximum of $\bar{J}(t_2)$. Thus, one must always check the second derivative to insure that the stationary point being converged upon is indeed a (relative) minimum of $\bar{J}$. Finally, as noted above, while the method converged rapidly (when convergence was obtained) for each fixed-time problem, the fundamental advantage of the Newton-Raphson method,

Table 3.1

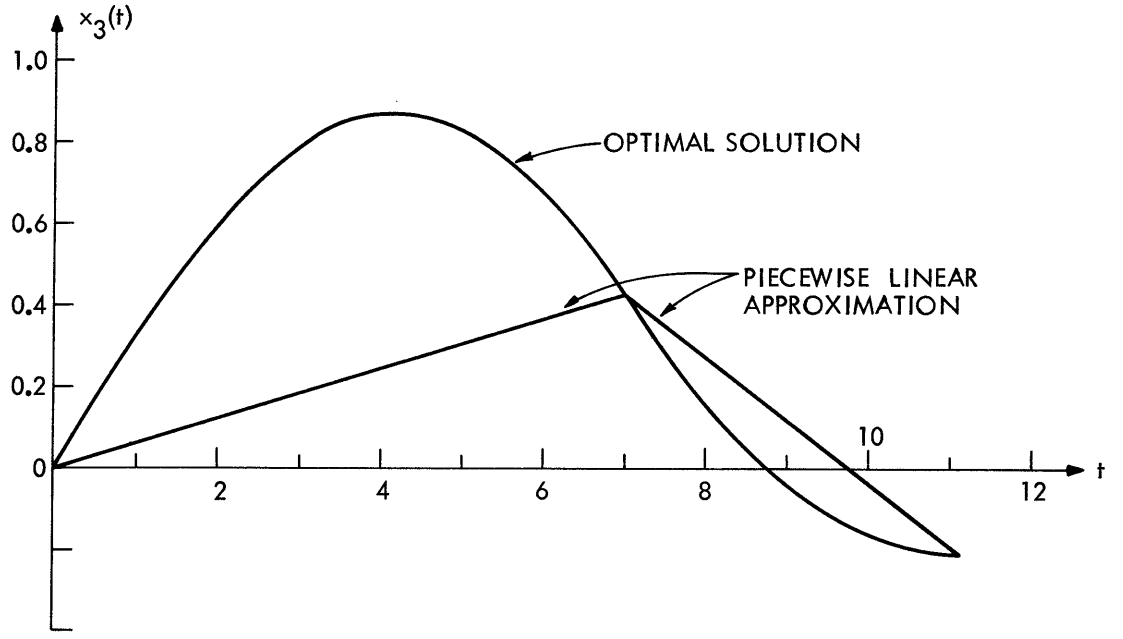
Newton-Raphson Algorithm—Single-Target Intercept Problem

I (Iteration)	ρ (Error Measure)	t_2 (Fixed Terminal Time)	J (Cost)	$\frac{\partial J}{\partial t_2}$ (Cost Gradient)
1	0.17380	6.60000	-	-
2	0.06037	6.60000	-	-
3	0.01359	6.60000	-	-
4	0.00089	6.60000	0.0062812	-0.522064
5	0.01177	6.60100	-	-
6	0.00038	6.60100	0.0058250	-0.396793
7	0.03538	6.60417	-	-
8	0.00337	6.60417	-	-
9	0.00035	6.60417	0.0049607	-0.176907
10	0.02337	6.60671	-	-
11	0.00170	6.60671	-	-
12	0.00006	6.60671	0.0046314	-0.087890
13	0.02009	6.60923	-	-
14	0.00109	6.60923	-	-
15	0.00006	6.60923	0.0044844	-0.032077
16	0.01036	6.61068	-	-
17	0.00030	6.61068	0.0044553	-0.008162
18	0.00337	6.61117	-	-
19	0.00006	6.61117	0.0044530	-0.000982
20	0.00042	6.61124	0.0044529	-0.000098

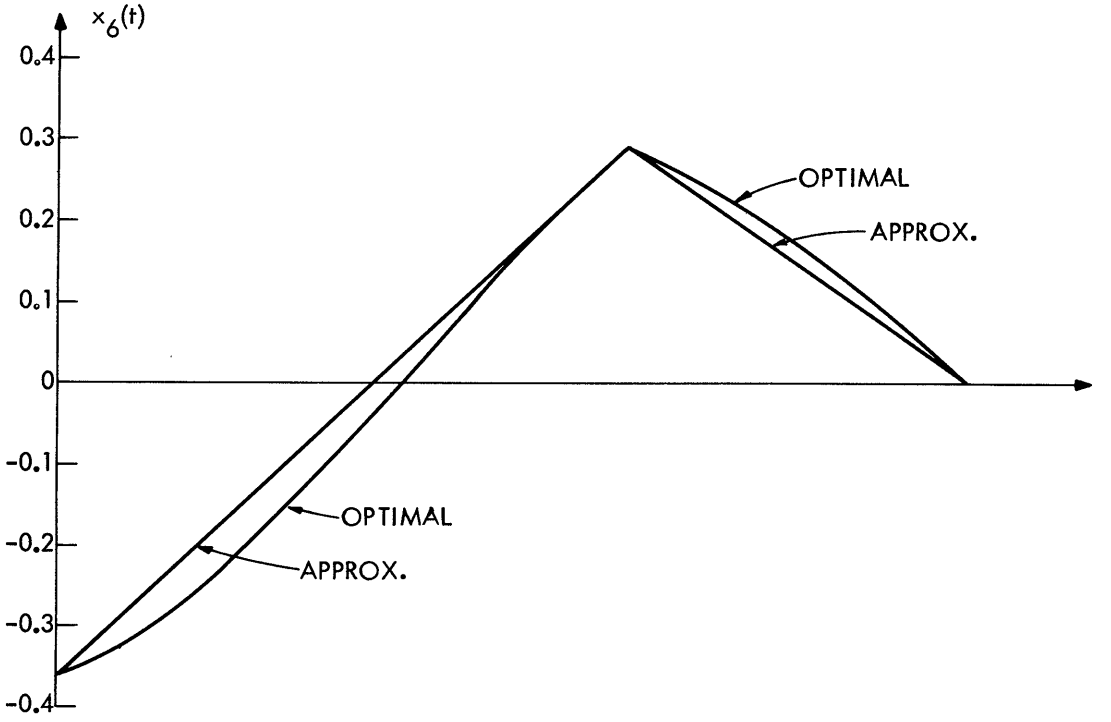
rapid convergence, is offset by the need to solve a sequence of optimum problems. The transformation of variables technique discussed in Section 3.4f allows this limitation to be circumvented for this particular problem.

The Newton-Raphson algorithm has also been applied to the two-target missile guidance problem for which the optimal solution was determined in Section 3.3 by the method of steepest descent, for the target motions given by Eq. 3.55. As in the one-target example, some difficulty was encountered in guessing a starting solution such that the algorithm would converge for the initial fixed-time problem. The optimal solution, as obtained by the steepest descent method was used to facilitate this choice. Figure 3.10 gives the optimal $\bar{x}_3(t)$ and $\bar{x}_6(t)$ trajectories (where $x_6(t) = \lambda_3(t) = -u(t)$) and approximating curves consisting of piecewise linear (two segments each) time functions, connecting the optimal trajectory at $t_0 = 0$ and the optimal intercept times, $\bar{t}_1 = 6.982$ and $\bar{t}_2 = 11.136$. With these linear approximating curves, and with the costates, $x_4(t)$ and $x_5(t)$ and the intercept times, t_1 and t_2, set at their optimal values (the state variables $x_1(t)$ and $x_2(t)$ need not be guessed, as noted previously in the single-target example, since they do not appear in the right-hand side of Eq. 3.100), the Newton-Raphson algorithm diverges. A converging solution is obtained, however, when a closer approximation to $x_3(t)$, consisting of about ten piecewise linear segments, is used.

To demonstrate the convergence of a sequence of fixed-time optimal control problems to the solution of a three-point free-time (both t_1 and t_2 free) optimal control problem, the specified intermediate and terminal times are increased by one percent over their optimal values, giving an initial fixed-time problem with $t_1 = 7.05$ and $t_2 = 11.24$. Table 3.2 summarizes the results of the convergent Newton-Raphson solutions. The solution to the initial fixed-time problem is obtained in three iterations, when the error measure as given by Eq. 3.103 is set at 0.001. The intermediate and terminal times are then perturbed (sequentially) by $t = 0.001$ in order to determine the required second-order information needed to apply the Newton method to minimize $\bar{J}(t_1, t_2)$. With this information one computes new times, $t_1 = 6.951$ and $t_2 = 11.082$, after a total of seven iterations of the Newton-Raphson



(a) $x_3(t)$ vs. t



(b) $x_6(t) = -u(t)$ vs. t

Fig. 3.10 Optimal Solution for $x_3(t)$ and $x_6(t)$ and Piecewise Linear Approximation

Table 3.2

Newton-Raphson Algorithm—Two Target Intercept Problem

I	ρ	t_1	t_2	J	$\frac{\partial J}{\partial t_1}$	$\frac{\partial J}{\partial t_2}$
(Iteration)	(Error Measurement)	(Fixed Intermediate Time)	(Fixed Terminal Time)	(Cost)	(Cost Gradient)	
1	0.420512	7.05000	11.24000	-	-	-
2	0.006115	7.05000	11.24000	-	-	-
3	0.000314	7.05000	11.24000	0.23060	0.05305	0.03135
4	0.001228	7.05100	11.24000	-	-	-
5	0.000016	7.05100	11.24000	0.23066	0.05471	0.03063
6	0.001111	7.05100	11.24100	-	-	-
7	0.000019	7.05100	11.24100	0.23069	0.05400	0.03128
8	0.180450	6.95080	11.08190	-	-	-
9	0.015028	6.95080	11.08190	-	-	-
10	0.001322	6.95080	11.08190	-	-	-
11	0.000013	6.95080	11.08190	0.22817	-0.02828	-0.03668
12	0.001374	6.95180	11.08190	-	-	-
13	0.000009	6.95180	11.08190	0.22814	-0.02487	-0.03806
14	0.001293	6.95180	11.08290	-	-	-
15	0.000014	6.95180	11.08290	0.22810	-0.02626	-0.03640
16	0.051120	6.97700	11.12577	-	-	-
17	0.001581	6.97700	11.12577	-	-	-
18	0.000040	6.97700	11.12577	0.22684	-0.00205	-0.00662

(Continued on next page)

Table 3.2 (Contd.)

Newton-Raphson Algorithm—Two Target Intercept Problem

I (Iteration)	ρ (Error Measurement)	t_1 (Fixed Intermediate Time)	t_2 (Fixed Terminal Time)	J (Cost)	$\frac{\partial J}{\partial t_1}$	$\frac{\partial J}{\partial t_2}$ (Cost Gradient)
19	0.001300	6.97800	11.12577	-	-	-
20	0.000014	6.97800	11.12577	0.22684	-0.00065	-0.00769
21	0.001221	6.97800	11.12677	-	-	-
22	0.000009	6.97800	11.12677	0.22683	-0.00042	-0.00650
23	0.010350	6.98162	11.13549	-	-	-
24	0.000066	6.98162	11.13549	0.22680	0.00005	-0.00019
25	0.001293	6.98262	11.13549	-	-	-
26	0.000016	6.98262	11.13549	0.22680	0.00261	-0.00120
27	0.001222	6.98262	11.13644	-	-	-
28	0.000004	6.98262	11.13644	0.22680	0.00159	-0.00008
29	0.001076	6.98170	11.13572	-	-	-
30	0.000026	6.98170	11.13572	0.22680	0.00002	-0.00001

algorithm. The above procedure is then repeated until, after thirty iterations, one has obtained the optimal intercept times, $\bar{t}_1 = 6.982$ and $\bar{t}_2 = 11.136$, which agree with the results of the steepest descent solution after ten iterations. From the value of the error measure, ρ, at successive iterations, one observes the quadratic convergence of the Newton-Raphson algorithm for each fixed-time problem, and from the values of $\frac{\partial \bar{J}}{\partial t_1}$ and $\frac{\partial \bar{J}}{\partial t_2}$ at iterations 3, 11, 18, 24, and 30, one notes the quadratic convergence of the Newton (secant) method for minimizing $\bar{J}(t_1, t_2)$. It should be kept in mind that the number of iterations required depends on the value of the error measure specified in order to define the convergence of each fixed-time problem. For example, if ρ can be increased by an order of magnitude without appreciably affecting the results of subsequent fixed-time problems, only about eighteen iterations will be required. Perhaps a variable ρ should be used -- one that is relatively large for the first few fixed-time problems, from which one only desires rough estimates of $\frac{\partial \bar{J}}{\partial t_1}$ and $\frac{\partial \bar{J}}{\partial t_2}$, and becomes smaller as the optimal t_1 and t_2 are approached and a more precise solution is desired. This extension was not tried and is suggested as a topic for future research.

As in the single-target example, one should note that difficulty can be encountered in guessing a solution close enough to the optimal to insure convergence, and that the speed of convergence is offset by the necessity of solving a sequence of problems to iterate on the optimal intercept times. The latter problem is accentuated in the three-point BVP by the additional iterations required to obtain the second-order information. This may perhaps be alleviated by not applying the perturbations, δt_1 and δt_2, as computed by Eq. 3.96 simultaneously, but rather by applying them sequentially. That is, each time t_1 is perturbed one can update $\frac{\partial^2 \bar{J}}{\partial t_1^2}$ and $\frac{\partial^2 \bar{J}}{\partial t_2 \partial t_1}$, and when t_2 is perturbed, one can update $\frac{\partial^2 \bar{J}}{\partial t_1 \partial t_2}$ and $\frac{\partial^2 \bar{J}}{\partial t_2^2}$. While this approach obviates the unproductive iterations required to obtain the second-order information after a simultaneous change in both t_1 and t_2, it is not clear that it

will converge quadratically. The author leaves this question open for further research.

The computer program used to obtain the results given in Figs. 3.8 and 3.9 and Tables 3.1 and 3.2 is listed in Appendix E, and a flow chart of the program is given in Fig. 3.11. A second-order Runge-Kutta integration scheme is used in carrying out the required integrations, just as in the method of steepest-descent. Since two of the costates, $x_4(t)$ and $x_5(t)$, are piecewise constant functions of time, only four variables have to be integrated $x_1(t)$, $x_2(t)$, $x_3(t)$, and $x_6(t)$. With the number of state variables being three, Eq. 3.100 has to be integrated a total of four times per iteration, to obtain the particular solution and three homogeneous solutions. Thus, a total of 16 integrations have to be performed each iteration. An additional four integrations are required if one chooses not to store the entire $\Phi^i_p(t)$ and $\Phi^i_j(t)$ functions, as is the case for this example. Using an integration time increment equal to 0.005 times t_2, i.e., $\Delta t_{INT} = \frac{t_2}{200}$, approximately 2.5 seconds of computation time on an IBM 360 are required for each iteration. The solution obtained by the Newton-Raphson agrees with the solution obtained by the steepest-descent algorithm (Section 3.3b) to four significant figures.

3.4f Change of Variables Technique

For a certain class of free-time problems, one can avoid the necessity of solving a sequence of fixed-time problems by making an appropriate change of variables. To accomplish this, one needs to identify a state variable, or function of several state variables, whose initial and terminal values are known, and whose time rate of change does not change sign (or become zero) along the optimal trajectory. By making this quantity the independent variable, the problem can be transformed from a free-time to a "fixed-time" problem in which time becomes a "state" variable, and one need not know its optimal terminal value in order to apply the Newton-Raphson method directly. The requirement that the function relating the new independent variable and time be strictly monotonic is imposed in order that the state differential equations and the cost functional can be integrated with respect to the new independent variable. An example of such a free-time problem is the

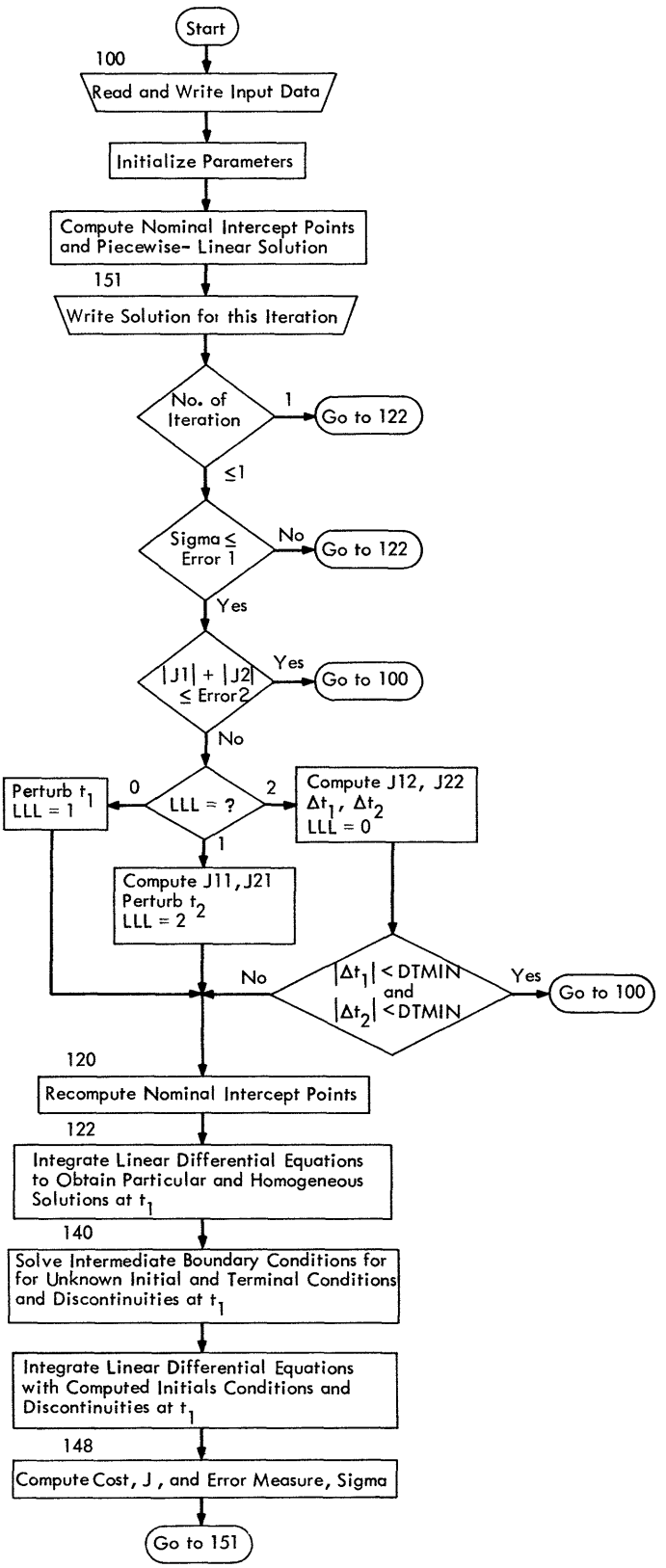


Fig. 3.11 Flow Chart for Newton-Raphson Computer Program

single-target intercept problem, in which the range is (usually) a monotonically decreasing function of time whose initial and terminal values are known.

Let $\tau[x(t)]$ be a mapping of $t \epsilon (t_0, t_1)$ into $\tau \epsilon (\tau_0, \tau_1)$ which is one to one and onto, with

$$\begin{aligned} \tau[x(t_0)] &= \tau_0 \\ \tau[x(t_1)] &= \tau_1 \end{aligned} \tag{3.104}$$

for which the inverse mapping is $t = g(\tau)$.
Then, one has

$$\frac{d\tau}{dt}(t) = \frac{\partial \tau[x(t)]}{\partial x}^T \cdot \frac{dx}{dt} = h(x,u,t) \tag{3.105}$$

and

$$\frac{dt}{d\tau}(\tau) = h^{-1}(x,u,g(\tau)) = \frac{1}{h[x,u,g(\tau)]} \tag{3.106}$$

The system differential equations, Eq. 2.1, can be written in terms of the new independent variable, τ,

$$\frac{dx_i}{d\tau} = \frac{dx_i}{dt} \cdot \frac{dt}{d\tau} = f[x,u,g(\tau)] \cdot h^{-1}(x,u,g(\tau)) \quad i=1,\ldots,n \tag{3.107}$$

and the cost functional Eq. 2.2 can be written as

$$J = \phi^0[x(\tau_0)] + \phi^1[x(\tau_1)] + \int_{\tau_0}^{\tau_1} L(x,u,g(\tau)) \cdot h^{-1}[x,u,g(\tau)]\, d\tau \tag{3.108}$$

The initial and terminal constraints, Eq. 2.3 become

$$\psi^{i'}[x(\tau),\tau] = \begin{bmatrix} \psi^i[x(\tau),g(\tau)] \\ \tau - \tau_i \end{bmatrix} = 0 \qquad i = 0,1 \tag{3.109}$$

Equations 3.104 - 3.109 define a new "fixed-time", optimal control problem.

To demonstrate this technique, consider the single-target intercept problem which is solved in the previous section. Let us define, for convenience,

$$\tau[x(t)] = x_2(t) \tag{3.110}$$

A more practical choice of τ for more general target motions would be the range to the target, but for the target motion given by Eq. 3.101, $x_2(t)$ is a (strictly) monotonically increasing function of time, and this choice simplifies subsequent computations. Making this variable the new independent variable, and defining

$$v(\tau) = \begin{bmatrix} v_1(\tau) \\ v_2(\tau) \\ v_3(\tau) \end{bmatrix} = \begin{bmatrix} x_1(\tau) - y_1(\tau) \\ t(\tau) \\ x_3(\tau) \end{bmatrix} \tag{3.111}$$

the new state differential equations become

$$\frac{dv_1}{d\tau} = \frac{dx_1}{dt} \cdot \frac{dt}{d\tau} = [\cos v_3(\tau) - \dot{y}_1(\tau)]/\sin v_3(\tau) \qquad \text{a)}$$

$$\frac{dv_2}{d\tau} = \frac{dt}{d\tau} = 1/\sin v_3(\tau) \qquad \text{b)} \tag{3.112}$$

$$\frac{dv_3}{d\tau} = \frac{dx_3}{dt} \cdot \frac{dt}{d\tau} = u(\tau)/\sin v_3(\tau) \qquad \text{c)}$$

The cost functional becomes

$$J = \int_{\tau_0}^{\tau_1} u^2(\tau) \cdot \frac{dt}{d\tau} \cdot d\tau = \int_0^5 [u^2(\tau)/\sin v_3(\tau)]\, d\tau \tag{3.113}$$

and the initial and terminal constraints become

$$\psi^0[v(\tau), \tau] = \begin{bmatrix} v_1(\tau) + 1 \\ v_2(\tau) \\ v_3(\tau) \\ \tau \end{bmatrix} = 0 \qquad \text{a)}$$

$$\psi^1[v(\tau), \tau] = \begin{bmatrix} v_1(\tau) \\ \tau - 5 \end{bmatrix} = 0 \qquad \text{b)} \tag{3.114}$$

Equations 3.112 - 3.114 represent a "fixed-time" problem for which the Newton-Raphson algorithm, as discussed above and in Appendix C, can be applied directly. Table 3.3 gives the error measurement between the solutions obtained for successive iterations and the value of $t(\tau_1)$, the optimal intercept time, at each iteration after making a starting guess

$$\begin{aligned} v_1(\tau) &= -1 + (\tfrac{\tau}{5}) & v_4(\tau) &= 0 \\ v_2(\tau) &= 6.6\ (\tfrac{\tau}{5}) & v_5(\tau) &= 0 \\ v_3(\tau) &= 1 - (0.3)\ (\tfrac{\tau}{5}) & v_6(\tau) &= (0.1)(1 - \tfrac{\tau}{5}) \end{aligned} \tag{3.115}$$

which is equivalent to the starting guess given by Eq. 3.102.

As can be seen from Table 3.3, the convergence is quadratic, and the optimal solution is obtained in only three iterations. While a good estimate for the computation time per iteration is not available, it is actually less than that required by the unmodified Newton-Raphson algorithm. From Eqs. 3.112 - 3.114, and the Euler-Lagrange equations and terminal transversality conditions, it can be shown that for this new problem

$$\frac{d\lambda_2}{d\tau} = \frac{dv_5}{d\tau} = 0 \tag{3.116}$$

and

$$\lambda_2(\tau_1) = 0 \tag{3.117}$$

and thus this costate is identically equal to zero. This reduces by one the number of homogeneous solutions to be obtained. In addition, since the right-hand side of Eq. 3.112 is independent of $v_2(\tau)$, and since its terminal value is free, one need not integrate $v_2(\tau)$ until one has finally converged on the optimal solution. Thus, only nine integrations need be performed per iteration (plus an additional three if one chooses not to store $\Phi_p(t)$ and the $\Phi_j(t)$ in Eq. C.24).

Unfortunately, in extending this change of variables technique to three-point BVP's, the class of problems for which it is applicable becomes even more limited. For example, the method can be applied to the two-target intercept problem considered above by defining

$$\tau[x(t)] = R(t) = [(x_1(t) - z_1(t))^2 + (x_2(t) - z_2(t))^2]^{1/2} \tag{3.118}$$

Table 3.3
Change of Variables Technique

I (Iteration)	ρ (Error Measure)	t_1 (Terminal Time)
1	1.16851	6.60000
2	.01559	6.60505
3	.00003	6.61101

where $\tau(t_1) = 2$ is known only because the targets are moving with the same velocity. If the targets are moving at different velocities or in different directions, then one does not, in general, know the range to target z at the time of intercept of target y (since the intercept time t_1 is not known and the distance between the targets is a function of time). However, one does succeed in reducing the problem from one of finding the minimum of $\overline{J}(t_1, t_2)$ to one of finding the minimum of $\overline{J}(\tau_1)$, obviously a simpler problem.

3.5 SUMMARY AND CONCLUSIONS OF NUMERICAL TECHNIQUES

Several numerical techniques for the solution of two-point BVP's have been extended in this chapter to the general N-point BVP. First an indirect method, that of finding a convergent sequence of guesses at the unknown initial conditions and the unknown discontinuities of the co-state at the intermediate boundary times, has been considered. Second, the method of steepest-descent has been modified to handle N-point BVP's; both the penalty function and influence function techniques for treating state equality constraints have been considered. Finally, the Newton-Raphson method has been modified to handle N-point BVP's, both for fixed- and free-time problems. The free-time problem is solved by means of a sequence of fixed-time problems, or if applicable, by means of a change of variables which converts it into a fixed-time problem. A second-order method is used to converge upon the optimal times in the former method. Numerical results are given to demonstrate the applicability of both the steepest-descent and Newton-Raphson algorithms to optimal target intercept problems.

While one would like to be able to determine which of the various numerical algorithms is "best",[19,20,46,47] such broad generalizations

are not valid. One should, however, be aware of the advantages and disadvantages of each method, so that the method best suited to a particular problem can be selected. The numerical results of the extensions of the steepest-descent and Newton-Raphson algorithms, as discussed above, demonstrate the various advantages and disadvantages of these two algorithms. The modifications do not, in general, alter the fundamental properties of the algorithms, but the increased complexity of the problems considered tends to accentuate the shortcomings of each method.

The primary advantages of the method of steepest-descent are that one is guaranteed convergence to a (local) minimum, even though no a priori information is available with respect to the optimal solution. In addition, it is easily modified to treat free-time problems and problems with bounded control and/or state variables,[26] with little or no degradation in performance. The fundamental disadvantage is that it exhibits relatively slow convergence in the vicinity of the optimal solution, where first order effects are small. This drawback can be overcome to some extent by incorporating second-order information in the choice of step size, as discussed in Sections B.2 and 3.3b, at a cost of increased computational requirements. The fact that one must specify, in some way, the step size at each iteration is often cited as a disadvantage of this algorithm. Actually, quite the contrary is true, since it gives the individual greater flexibility in applying the algorithm -- allowing him to take more conservative steps when "far" from the optimum to insure steady, controlled convergence, and to include second-order information as the optimum is approached in order to speed up the convergence in that region.

The penalty function approach to state equality constraints has been found to be ineffective for N-point problems because the added penalty functions conflict, which results in a "ravine" effect and subsequently extremely slow convergence. The influence function approach to state equality constraints has been found to be quite effective; however, the amount of computation required is increased significantly, since one must compute an n-vector influence function for each new state constraint. Both methods were tried on a two-target intercept problem to demonstrate these conclusions. Solutions obtained using

the influence function approach demonstrate the ability to converge from a poor initial guess as well as the incorporation of second-order information for faster convergence as the optimum is approached. An automatic step-size control is incorporated which essentially doubles the step size if the previous step is too small and halves the step-size if the previous step is too large. The algorithm is found to require the integration of twelve quantities per iteration and to converge in eight to ten iterations, requiring about 1.5 seconds per iteration on an IBM 360/65.

The primary advantage of the Newton-Raphson method is its quadratic convergence. Once a solution "near" to the optimum is obtained, the convergence to the optimum can be quite dramatic, leaving no doubt as to the fact that the algorithm has indeed converged. However, its primary disadvantage is that one must start with a solution sufficiently close to the optimum to guarantee that it will converge at all. If one makes the best guess at the optimum solution that he can with the available information and the algorithm diverges, he is in a rather bleak situation, since no information is forthcoming as to the reason for the divergence or to how a better guess can be made. Other disadvantages are that the algorithm is rather rigid, lacking the flexibility the steepest-descent method has with step-size control, and that its rapid convergence is offset when one considers free-time and bounded state and/or control problems.[34,36] These properties are clearly demonstrated by the target intercept examples considered. In Tables 3.1 and 3.2 one observes the quadratic convergence for each of the fixed-time problems. However, as noted in Section 3.4d, some difficulty was encountered in obtaining initial guesses for which the algorithm would converge for the first fixed-time problem. In addition, the speed of convergence is offset by the need to solve a sequence of fixed-time problems, since in the example considered, the optimal intercept times are unknown. The algorithm involves a total of twenty integrations per iteration, requiring twenty to thirty iterations to converge and about 2.5 seconds of computation time on an IBM 360 per iteration.

Finally, an example is given demonstrating the advantage to be gained by using the change of variables technique in a free-time problem. For the single-target intercept problem, the number of iterations required for convergence is reduced from twenty to three. The number of computations required at each iteration is also reduced for this particular example.

In conclusion, for the particular example considered, the steepest-descent method appears to give better results than the Newton-Raphson method, mainly because the problem involves unspecified intercept times. In addition, the author considers the potential divergence of the Newton-Raphson method to be a serious drawback. If one is willing to pay the cost of additional computer programming and storage requirements, a hybrid method might be utilized in order to incorporate the best features of each of the algorithms. Other methods such as Fletcher-Powell[56] might be employed to generate second-order information from gradient calculations. Extensions of the latter two methods to N-point boundary value problems are topics suggested for future research.

CHAPTER IV

OPTIMAL AND SUBOPTIMAL MULTIPLE TARGET INTERCEPT GUIDANCE

4.1 INTRODUCTION

In this chapter the results of Chapters II and III are applied to a specific optimal control problem -- missile guidance against multiple targets. The discussion is limited to a constant velocity missile having no turning rate constraints, moving on a two-dimensional playing field, and attacking two targets. In Section 4.2 an optimum feedback control law is obtained for a much simpler problem -- a double integrator system for which the intermediate and terminal boundary conditions require that the "position" state be brought to specified levels at two fixed points in time. The control law which minimizes a quadratic measure of the control effort is linear and time-varying. In Section 4.3 optimal trajectories are computed for the missile guidance problem of interest. The steepest-descent algorithm discussed in Section 3.3b is used to compute the trajectories. An analytic expression for an optimum feedback control law cannot be obtained for this problem because the differential equations for the missile kinematics are nonlinear. Various target motions, both maneuvering and non-maneuvering, are considered under the assumption that they are known in advance. Since the future motion of the targets is not generally known a priori, one requires a feedback control law in order to compensate for unpredictable target maneuvers. In Section 4.4 suitable approximations are made in order to obtain a feedback control law. It is assumed that one has perfect (noiseless) knowledge of the present state of motion (the position, heading, and velocity) of each of the targets. By making an appropriate change in the coordinate system, the model for the actual problem is reduced to the double integrator example solved in Section 4.2. A "suboptimal" control law is then obtained by computing the optimal control for the model and relating it to the missile turning-rate control. This feedback control law has been simulated on an IBM 360/65, and the suboptimal intercept trajectories resulting for various targets are presented. A discussion of the suboptimal control law and suggestions

for future work in this general area are contained in Sections 4.5 and 4.6.

4.2 DOUBLE INTEGRATOR, TWO-TARGET EXAMPLE

In this section the optimal feedback control law is determined for a double integrator system* such that the control effort is minimized in taking the system from any initial state to specified states at two fixed points in time. The system differential equations are given by

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) \\ \dot{x}_2(t) &= u(t) \end{aligned} \tag{4.1}$$

with the state constraints

$$\psi^0[x(t),t] = \begin{bmatrix} x_1(t) - x_{10} \\ x_2(t) - x_{20} \\ t - t_0 \end{bmatrix} = 0 \qquad \text{a)}$$

$$\psi^1[x(t),t] = \begin{bmatrix} x_1(t) - x_{11} \\ t - t_1 \end{bmatrix} = 0 \qquad \text{b)} \tag{4.2}$$

$$\psi^2[x(t),t] = \begin{bmatrix} x_1(t) \\ t - t_2 \end{bmatrix} = 0 \qquad \text{c)}$$

and

$$t_0 < t_1 < t_2 \qquad \text{d)}$$

The cost functional is given by

$$J = \frac{1}{2}\int_{t_0}^{t_2} [u(t)]^2 \, dt \tag{4.3}$$

---

* This system is also considered in Section 2.5 in connection with the solution of the general class of linear system-quadratic cost problems via the solution of a matrix differential equation of the Riccati type.

Thus, the objective is to take the state $x_1(t)$ from any given initial starting state to the value x_{11} at t_1 and to zero at t_2, while minimizing the "control energy" expended.

While the above problem can be solved directly, a similar problem, based on the penalty function approach is first solved. That is, rather than impose the intermediate and terminal boundary conditions given by Eqs. 4.2b and 4.2c directly, we minimize instead a modified cost functional given by

$$J = \frac{1}{2} \cdot C_1 \cdot [x_1(t_1) - x_{11}]^2 + \frac{1}{2} \cdot C_2 \cdot [x_1(t_2)]^2 + \frac{1}{2} \int_{t_0}^{t_2} [u(t)]^2 \, dt \quad (4.4)$$

with no state constraints. This problem formulation is in a sense more general, since once its solution is obtained, the solution to the original problem with state constraints can be obtained by letting C_1 and C_2 become arbitrarily large.

For this problem, the Hamiltonian function is given by

$$H[x, u, \lambda, t] = \frac{1}{2} [u(t)]^2 + \lambda_1(t) \cdot x_2(t) + \lambda_2(t) \cdot u(t) \quad (4.5)$$

and therefore the Euler-Lagrange equations become*

$$\begin{aligned} \dot{x}_1^i(t) &= x_2^i(t) && \text{a)} \\ \dot{x}_2^i(t) &= u^i(t) && \text{b)} \end{aligned} \quad (4.6)$$

$$u^i(t) + \lambda_2^i(t) = 0 \quad (4.7)$$

and

$$\begin{aligned} \dot{\lambda}_1^i(t) &= 0 && \text{a)} \\ \dot{\lambda}_2^i(t) &= -\lambda_1^i(t) && \text{b)} \end{aligned} \quad (4.8)$$

for $i = 1, 2$.

---

* In keeping with the notation in previous chapters, the superscripts denote the time interval of definition of the solutions to the Euler-Lagrange equations, while exponents are denoted by $[\ ]^n$.

Solving for $u^i(t)$ in Eq. 4.7 and substituting it into Eq. 4.6b, the control variable can be eliminated, giving

$$\dot{x}_2^i(t) = -\lambda_2^i(t) \tag{4.9}$$

Equations 4.6a, 4.8a, 4.8b, and 4.9 represent four linear differential equations in terms of the four variables $x_1^i(t)$, $x_2^i(t)$, $\lambda_1^i(t)$, and $\lambda_2^i(t)$, which can be integrated forward in time from t_0 and backward in time from t_2 to give

$$x_1^1(t) = \frac{1}{6}\lambda_{10}(t-t_0)^3 - \frac{1}{2}\lambda_{20}(t-t_0)^2 + x_{20}(t-t_0) + x_{10} \quad \text{a)}$$

$$x_2^1(t) = \frac{1}{2}\lambda_{10}(t-t_0)^2 - \lambda_{20}(t-t_0) + x_{20} \quad \text{b)} \qquad (4.10)$$

$$\lambda_1^1(t) = \lambda_{10} \quad \text{c)}$$

$$\lambda_2^1(t) = -\lambda_{10}(t-t_0) + \lambda_{20} \quad \text{d)}$$

and

$$x_1^2(t) = \frac{1}{6}\lambda_{12}(t-t_2)^3 - \frac{1}{2}\lambda_{22}(t-t_2)^2 + x_{22}(t-t_2) + x_{12} \quad \text{a)}$$

$$x_2^2(t) = \frac{1}{2}\lambda_{12}(t-t_2)^2 - \lambda_{22}(t-t_2) + x_{22} \quad \text{b)} \qquad (4.11)$$

$$\lambda_1^2(t) = \lambda_{12} \quad \text{c)}$$

$$\lambda_2^2(t) = -\lambda_{12}(t-t_2) + \lambda_{22} \quad \text{d)}$$

where the x_{ij} and λ_{ij} represent the boundary values of $x_i(t)$ and $\lambda_i(t)$ at the times t_j. Equations 4.10 and 4.11 involve eight boundary values, x_{ij} and λ_{ij}, of which only two, the initial values x_{10} and x_{20} are specified. The remaining six unknowns must be determined from the four intermediate and two terminal transversality conditions which, for this problem, are given by (see Chapter II)

$$-\Delta\bar{\lambda}_1(t_1) = -\lambda_1^2(t_1^+) + \lambda_1^1(t_1^-) = C_1 \cdot [\bar{x}_1(t_1) - x_{11}] \quad \text{a)}$$

$$-\Delta\lambda_2(t_1) = -\lambda_2^2(t_1^+) + \lambda_2^1(t_1^-) = 0 \quad \text{b)}$$

$$\Delta\bar{x}_1(t_1) = x_1^2(t_1^+) - x_1^1(t_1^-) = 0 \qquad c) \qquad (4.12)$$

$$\Delta\bar{x}_2(t_1) = x_2^2(t_1^+) - x_2^1(t_1^-) = 0 \qquad d)$$

$$\lambda_1^2(t_2) = C_2 \cdot \bar{x}_2(t_2) \qquad e)$$

and

$$\lambda_2^2(t_2) = 0 \qquad f)$$

From Eq. 4.7

$$u^1(t_0) = \lambda_2^1(t_0) = \lambda_{20} \qquad (4.13)$$

and substituting Eqs. 4.10 and 4.11 into Eq. 4.12, one can solve for λ_{20} and obtain

$$u^1(t_0) = -\frac{N_{11}}{D_1} \cdot x_1(t_0) - \frac{N_{21}}{D_1} x_2(t_0) + \frac{N_{31}}{D_1} \cdot x_{11} \qquad (4.14)$$

where

$$N_{11} = \frac{1}{2}\tau_2^2\tau_1^2 + (\frac{1}{3}\tau_2^3 + \frac{1}{C_1} + \frac{1}{C_2})\tau_1 + \frac{1}{C_1} \cdot \tau_2 \qquad a)$$

$$N_{21} = \frac{1}{3}\tau_2^2\tau_1^3 + (\frac{1}{3}\tau_2^3 + \frac{1}{C_1} + \frac{1}{C_2})\tau_1^2 + \frac{2}{C_1}\tau_2\tau_1 + \frac{1}{C_1}\tau_2^2 \qquad b)$$

$$N_{31} = \frac{1}{6}\tau_2\tau_1^3 + \frac{1}{2}\tau_2^2\tau_1^2 + (\frac{1}{3}\tau_2^3 + \frac{1}{C_2})\tau_1 \qquad c)$$

$$D_1 = \frac{1}{12}\tau_2^2\tau_1^4 + (\frac{1}{9}\tau_2^3 + \frac{1}{3C_1} + \frac{1}{3C_2})\tau_1^3 + \frac{1}{C_1}(\tau_2\tau_1^2 + \tau_2^2\tau_1) + (\frac{1}{3C_1}\tau_2^3 + \frac{1}{C_1C_2}) \qquad d)$$

$$\tau_1 = (t_1 - t_0) \qquad e)$$

and

$$\tau_2 = (t_2 - t_1) \qquad f)$$

(4.15)

Now, since t_0, $x_1(t_0)$, and $x_2(t_0)$ have been arbitrarily specified, Eqs. 4.14 and 4.15 hold for all t_0 and all initial conditions, and thus Eq. 4.14 can be written as

$$u^1(t) = -\frac{N_{11}}{D_1} \cdot x_1(t) - \frac{N_{21}}{D_1} \cdot x_2(t) + \frac{N_{31}}{D_1} \cdot x_{11} \tag{4.16}$$

for $t < t_1$, where now

$$\tau_1 = (t_1 - t) \tag{4.17}$$

in Eq. 4.15. At $t = t_1$, τ_1 is zero, and since $\bar{u}(t)$ is continuous at t_1 (by virtue of Eqs. 4.7 and 4.12f), Eq. 4.16 gives

$$u^2(t_1^+) = -\frac{N_{12}}{D_2} \cdot x_1(t_1^+) - \frac{N_{22}}{D_2} \cdot x_2(t_1^+) \tag{4.18}$$

where

$$N_{12} = \tau_2 \qquad \text{a)}$$

$$N_{22} = \tau_2^2 \qquad \text{b)} \tag{4.19}$$

and

$$D_2 = \frac{1}{3}\tau_2^3 + \frac{1}{C_2} \qquad \text{c)}$$

Since only one boundary condition remains to be satisfied after t_1, t_1^+ can be considered arbitrary and Eq. 4.18 can be written

$$u^2(t) = -\frac{N_{12}}{D_2} \cdot x_1(t) - \frac{N_{22}}{D_2} \cdot x_2(t) \tag{4.20}$$

where

$$\tau_2 = \tau_2 - t \tag{4.21}$$

for $t_1 < t < t_2$. Thus, Eqs. 4.16 and 4.20 represent a linear time-varying feedback control law which minimizes the modified cost functional, J, given by Eq. 4.4.

This penalty function approach enables one to bring $x_1(t)$ as close as desired to the intermediate and terminal targets by choosing C_1 and C_2 large enough. By solving for all six of the unknowns, one can compute from Eqs. 4.10a and 4.11a the required values of C_1 and C_2, as functions of $x_1(t_0)$, $x_2(t_0)$, and t_0, such that

$[x_1(t_1) - x_{11}]$ and $x_1(t_2)$ will have any specified values. The optimal $\bar{x}_1(t)$ for various values of $C_1 = C_2$ are plotted in Fig. 2.3 (for $x_{11} = 0$), as determined by means of the solution of a matrix differential equation of the Riccati type discussed in Section 2.5.

In order to find the optimal feedback control law which satisfies the intermediate and terminal state constants given by Eq. 4.2b and 4.2c, one can either re-solve the problem with the appropriate boundary conditions, or alternately let $C_1, C_2 \to \infty$ in Eqs. 4.15 and 4.19. Taking this latter approach, one finds that

$$u^1(t) = - \frac{6\cdot[3(t_1-t)+2\cdot(t_2-t_1)]}{(t_1-t)^2\cdot[3(t_1-t)+4(t_2+t_1)]}\cdot x_1(t) - \frac{12[(t_1-t)+(t_2-t_1)]}{(t_1-t)\cdot[3(t_1-t)+(t_2-t_1)]}\, x_2(t)$$

$$+ \frac{6\cdot[(t_1-t)+(t_2-t_1)]\cdot[(t_1-t)+2(t_2-t_1)]}{(t_2-t_1)\cdot(t_1-t)^2\cdot[3\cdot(t_1-t)+4\cdot(t_2-t_1)]}\cdot x_{11} \qquad (4.22)$$

for $t_0 < t < t_1$, and

$$u^2(t) = - \frac{3}{(t_2-t)^2}\cdot x_1(t) - \frac{3}{(t_2-t)}\cdot x_2(t) \qquad (4.23)$$

for $t_1 < t < t_2$. Thus, for time prior to the intermediate boundary point, the control law is a function of the times to go to each of the boundary points and the intermediate boundary value x_{11}, while for $t > t_1$, it is a function only of the time to go to the terminal boundary point, as one expects. Moreover, for t such that $(t_1-t) \ll (t_2-t_1)$, one finds that

$$\lim_{t \to t_1} u^1(t) = - \frac{3}{(t_1-t)^2}\cdot[x_1(t) - x_{11}] - \frac{3}{(t_1-t)}\cdot x_2(t) \qquad (4.24)$$

which is of the same form as Eq. 4.23, i.e., while the optimal feedback control law takes into account both state constraints prior to t_1 (by virtue of both t_1 and t_2 appearing in Eq. 4.22), the effect of the terminal constraint on the control law is diminished as the intermediate time is approached. However, this similarity of control laws does not imply that the controls, as functions of time, will be of the same form when near the targets. The choice of minimum control energy as the

performance criterion causes the control u to be linear with respect to time with a single discontinuity in slope at the intermediate target point and with a value of zero at the terminal time. At the intermediate target the magnitude of u is generally nonzero.

Figure 4.1 gives the optimal solution for $t_1 = 0.8$, $t_2 = 1.0$, $x_{10} = 1.0$, $x_{20} = 0$, and $x_{11} = 0$, as well as the "sequential optimal" solution obtained by solving the problem as two separate minimization problems, i.e., first finding the optimal control which satisfies only the initial and intermediate constraints, Eqs. 4.2a and 4.2b and then finding the optimal control which takes the system from the resulting conditions at t_1 to the terminal conditions specified by Eq. 4.2c. The minimum control effort is found to be $\overline{J} = 9.528$, while the control effort required by the sequential optimal solution is $J' = 29.297$, 308 percent higher than the minimum.

The problem solved in this section is actually a simplified target intercept problem, where one "intercepts" the points $x_1(t_1) = x_{11}$ and $x_1(t_2) = 0$. using a minimum amount of control effort. From Eqs. 4.23 and 4.24, we find that the feedback gains become infinite for $t = t_1$ and $t = t_2$. However, if one uses the penalty function approach, as is necessary in practical problems with noise, these gains are finite for finite values of C_1 and C_2.

4.3 OPTIMAL MULTIPLE TARGET INTERCEPT GUIDANCE

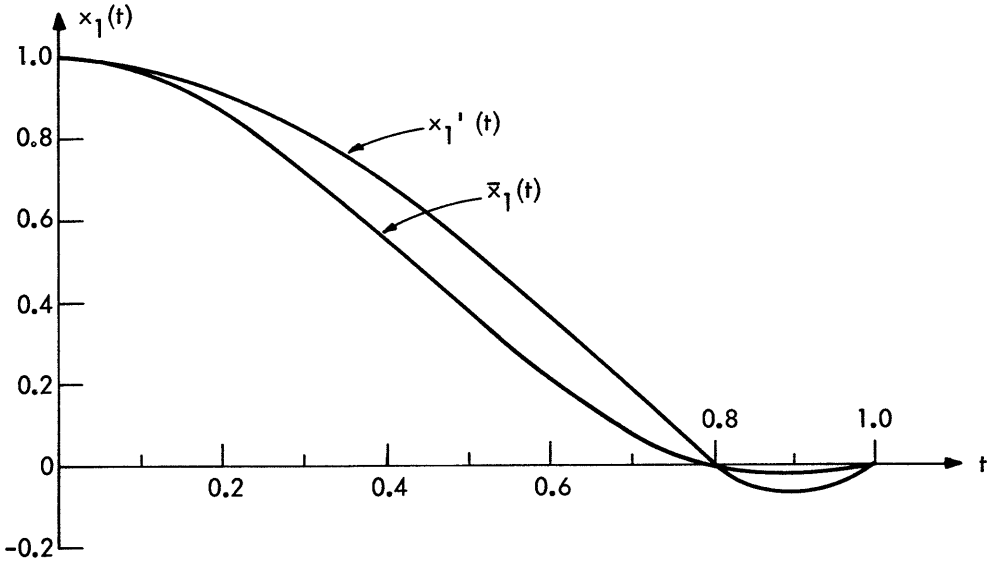
In this section we consider the problem of optimally controlling a constant velocity missile moving on a two-dimensional playing field, for which the system equations are

$$\dot{x}_1(t) = V_m \cdot \cos x_3(t) \qquad \text{a)}$$

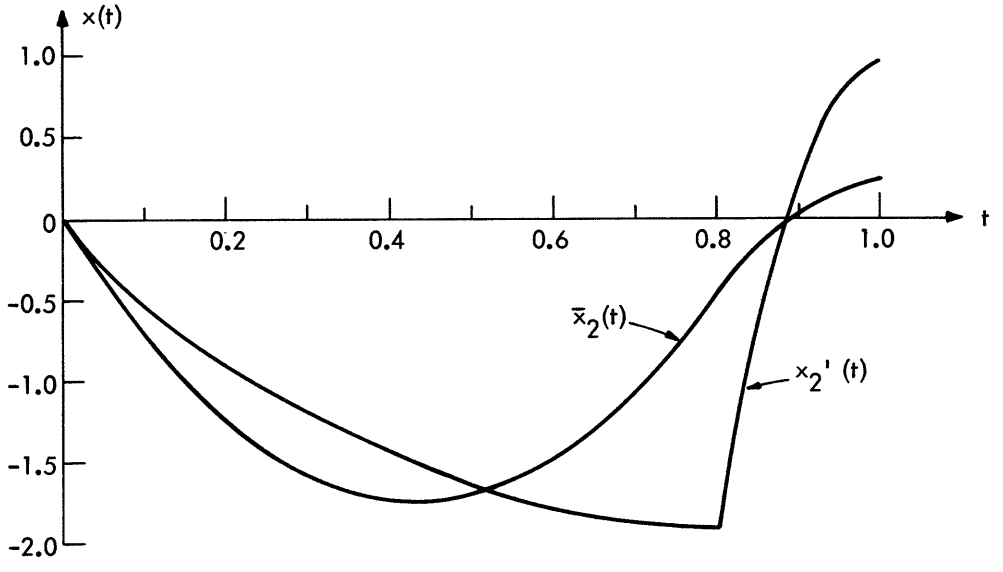
$$\dot{x}_2(t) = V_m \cdot \sin x_3(t) \qquad \text{b)} \qquad (4.25)$$

$$\dot{x}_3(t) = u(t) \qquad \text{c)}$$

where $x_1(t)$ and $x_2(t)$ are the coordinates of the missile, $x_3(t)$ is its heading angle, and u(t) is the turning-rate control. Various criteria, such as time-to-intercept, total fuel expended, and so forth, can be postulated as the performance functional to be minimized in target-intercept problems.[50] In this application, the performance criteria is

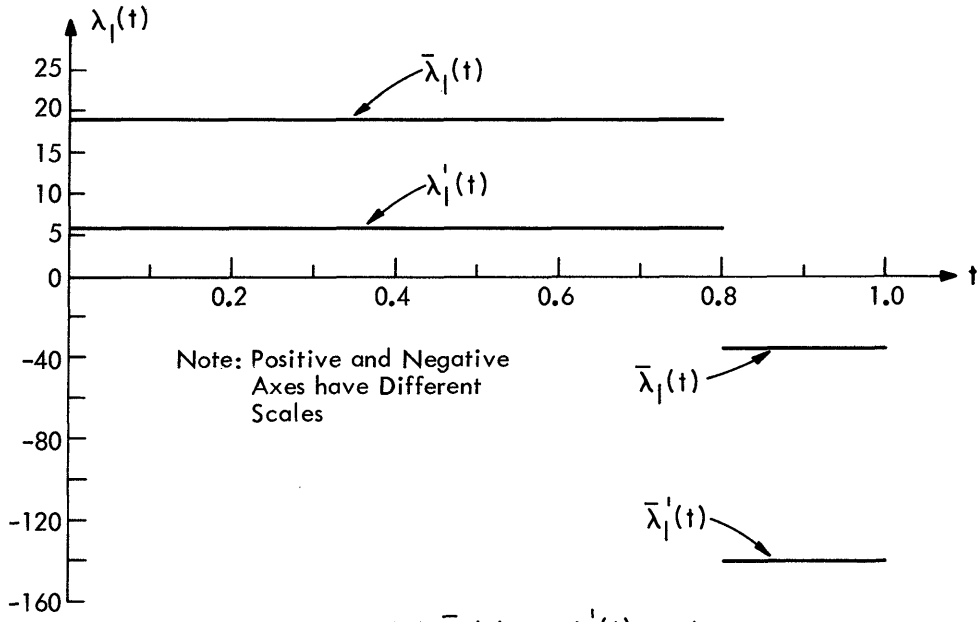


(a) $\bar{x}_1(t)$ and $x_1'(t)$ vs. t

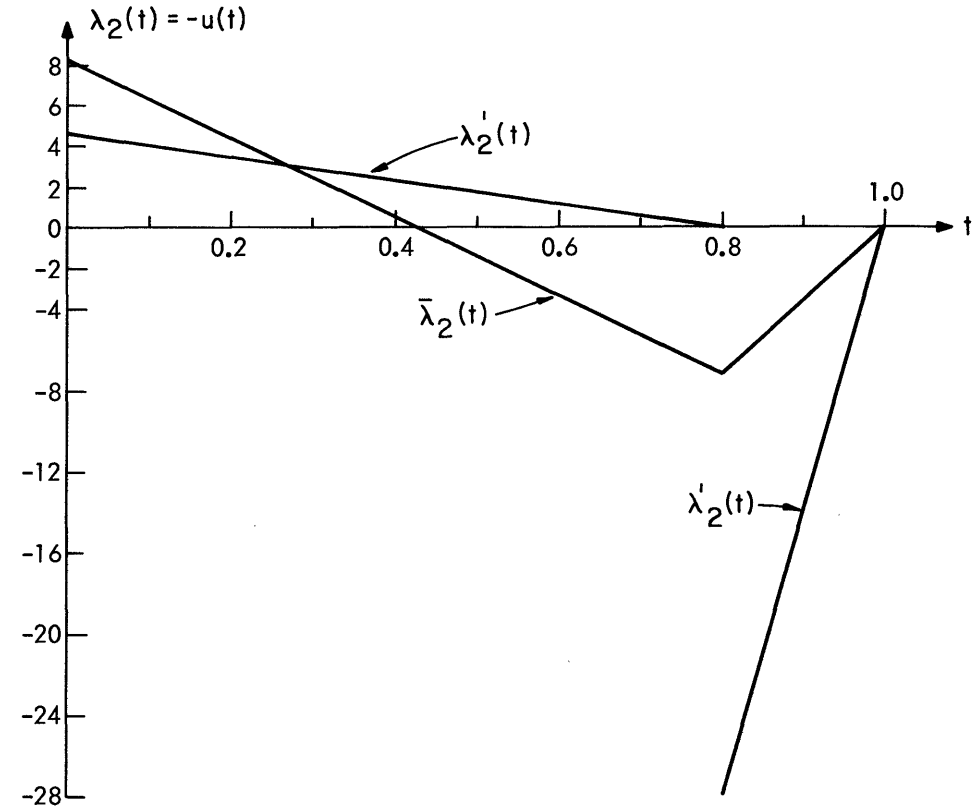


(b) $\bar{x}_2(t)$ and $x_2'(t)$ vs. t

Fig. 4.1 Optimal and Sequential Optimal Solutions for Double Integrator Example



(c) $\bar{\lambda}_1(t)$ and $\lambda_1'(t)$ vs. t



(d) $\bar{\lambda}_2(t)$ and $\lambda_2'(t)$ vs. t, $\lambda_2(t) = -u(t)$

Fig. 4.1 (cont.) Optimal and Sequential Solutions for Double Integrator Example

chosen to be a measure of the "control effort" given by

$$J = \frac{1}{2}\int_{t_i}^{t_f} [u(t)]^2 \, dt \tag{4.26}$$

This choice is motivated by previous work on missile intercept guidance[8,49,51] which has shown that the popular proportional guidance law can be considered optimal with respect to Eq. 4.26. What we are striving for is an optimal control law analogous to the "proportional control law" for the intercept of two targets whose motions as functions of time are given by $y(t)$ and $z(t)$, where

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix}, \qquad z(t) = \begin{bmatrix} z_1(t) \\ z_2(t) \\ z_3(t) \end{bmatrix} \tag{4.27}$$

i.e., we must take the system given by Eq. 4.25 from the initial conditions

$$\psi^0[x(t),t] = \begin{bmatrix} x_1(t) - x_{10} \\ x_2(t) - x_{20} \\ x_3(t) - x_{30} \\ t - t_0 \end{bmatrix} = 0 \tag{4.28}$$

to intermediate and terminal state conditions

$$\psi^1[x(t),t] = \begin{bmatrix} x_1(t) - y_1(t) \\ x_2(t) - y_2(t) \end{bmatrix} = 0 \tag{4.29}$$

and

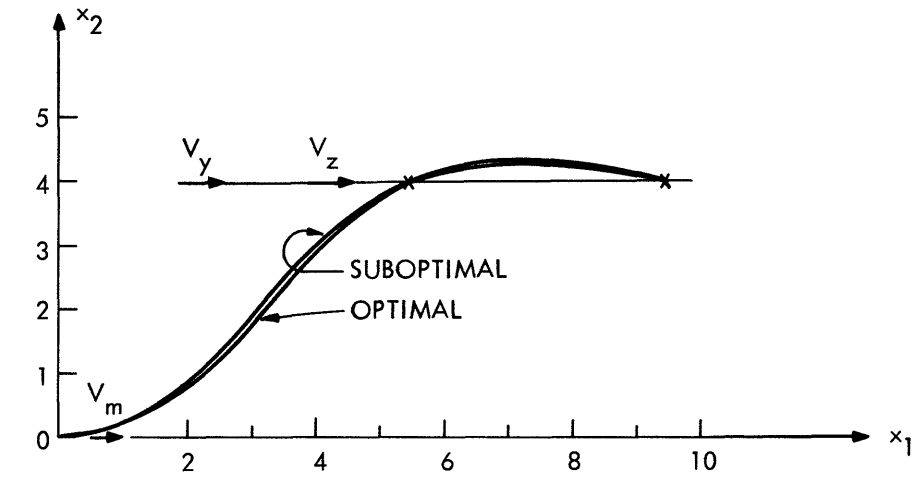
$$\psi^2[x(t),t] = \begin{bmatrix} x_1(t) - z_1(t) \\ x_2(t) - z_2(t) \end{bmatrix} = 0 \tag{4.30}$$

The intercept times, t_1 and t_2, as well as the heading of the missile at the intercept times are left unspecified.

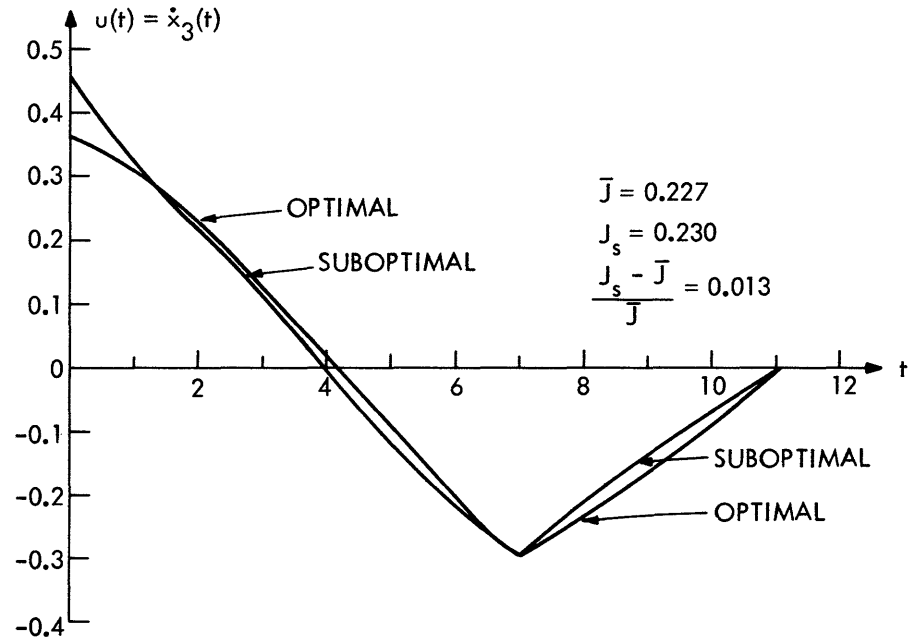
Because of the nonlinearities in the system equations, a closed form solution for the optimal control as a function of the state is not known, and one must resort to the numerical techniques discussed in Chapter III to obtain an open-loop control function for a given set of initial conditions and target motions. The steepest-descent algorithm which is discussed in Section 3.3b was utilized in computing optimal intercept trajectories for several sets of target motion. A flow chart of the computer program is given in Fig. 3.5, and the program is listed in Appendix D. An integration step-size of $\Delta t = 0.05$ seconds was used, and the algorithm is considered to have converged if on two successive iterations the sum of the magnitude of the distances by which the targets are missed is less than 0.001 and the value of the cost functional has not changed by more than 0.1 percent. As discussed in Chapter III, under these conditions the solution for the optimal intercept trajectory for the target motion given in Fig. 4.2 agrees with that obtained by the Newton-Raphson algorithm to an accuracy of four significant figures.

The optimal control function, and the resulting state trajectory, for various sets of initial conditions and target motions are illustrated in Figs. 4.2 through 4.4. Also shown are the "suboptimal" solutions which are discussed in Section 4.4. In each case the missile starts at the origin of the coordinate systems and has a normalized velocity of unity. The vector labeled V_m indicates the initial position and heading of the missile, and its length is scaled to the distance traveled by the missile in one time unit (hereafter called seconds for convenience). The vectors labeled V_y and V_z indicate the initial positions and headings of the two targets, and these are also scaled to indicate the velocities of the targets relative to the missile velocity.

In Fig. 4.2, the targets are moving on straight, colinear paths at velocities equal to one-half of the missile velocity. As can be seen from Fig. 4.2, the optimal control anticipates the second target by turning, for the first four seconds, more than is actually required for the intercept of the first target. This is done so that as the first target is approached the missile is turning not only towards the first target, but also towards the second target. Of course, by virtue of the minimiztion of Eq. 4.26, the total control effort required by this control



(a) $x_2(t)$ vs. $x_1(t)$



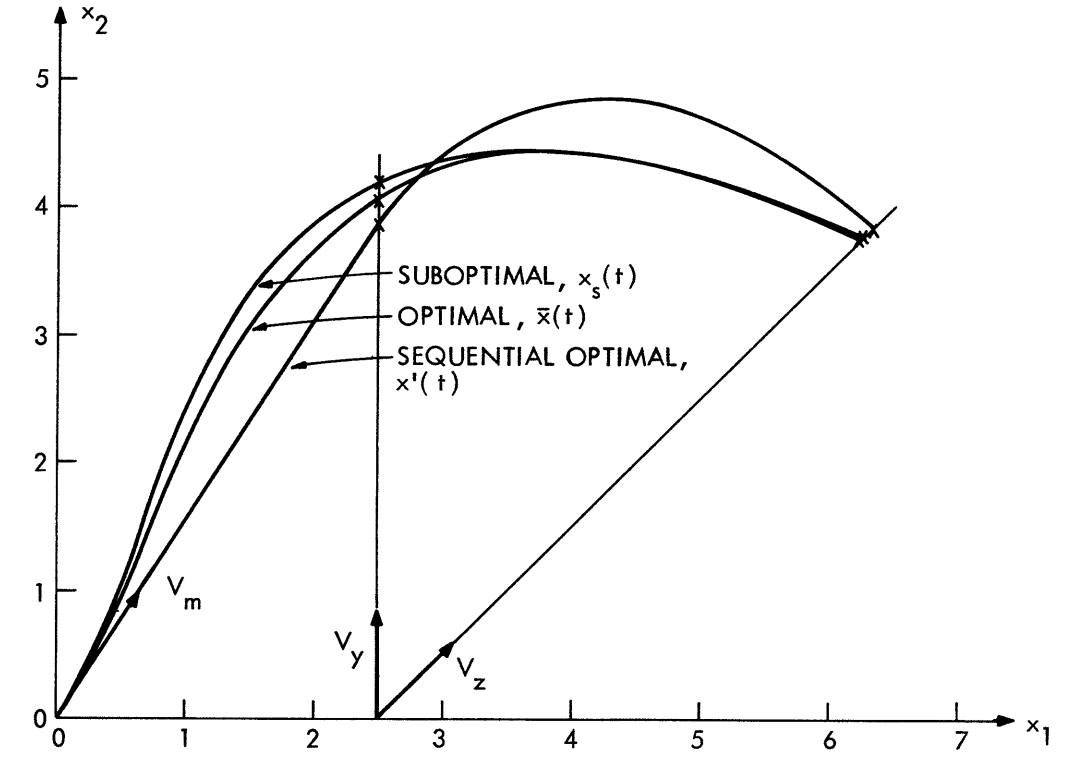
(b) u(t) vs. t

Fig. 4.2 Optimal and Sequential Solutions-
Nonmaneuvering Targets

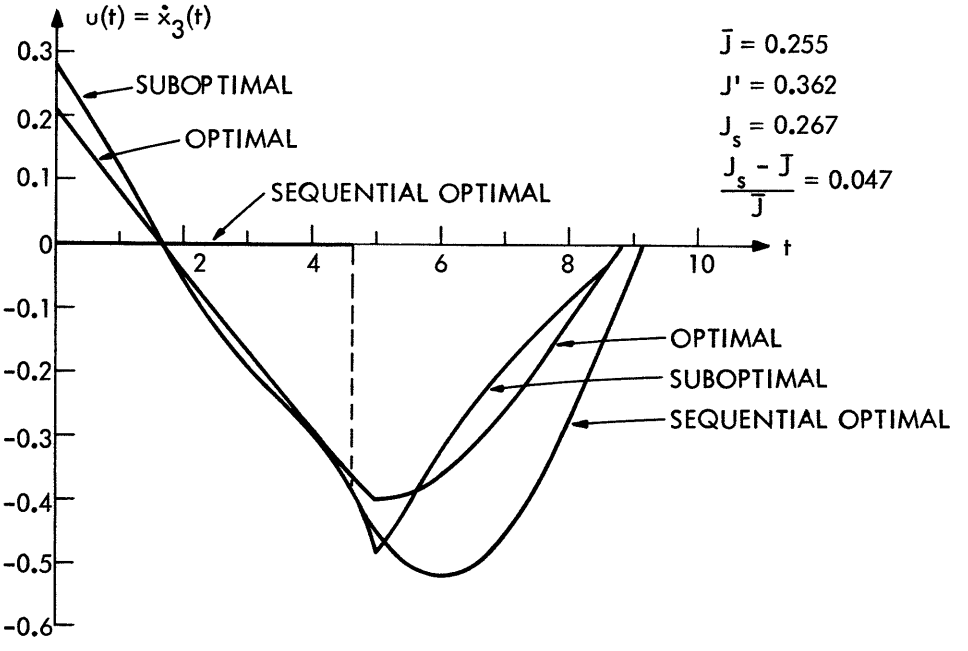
function over the entire interval, (0, 11.13), is less than that required by any other trajectory that intercepts both targets.

The "overturn" produced by the optimal control is perhaps more clearly demonstrated in Fig. 4.3, where the initial missile heading is one radian, and the velocity of targets y and z are 0.8414 and 0.6, respectively. The velocity of target y has been chosen so that at a heading of $\pi/2$ radians it is initially on an intercept course with the missile. At time $t = t_0 = 0$, the second target, z, emanates from the first target at a heading of $\pi/4$ radians. If the missile initially ignores target z, no commands need to be given in order to intercept target y, and the trajectory followed is indicated by $x'(t)$. If after intercepting y in this manner the missile proceeds to intercept z optimally, the resulting value of the control effort required is $J' = 0.362$. The optimal two-target intercept control law, however, yields the smallest possible value of control effort, $\overline{J} = 0.255$, by first turning off of the intercept course, so that as target y is approached, the missile is already turning towards target z. Thus, while no control is required in this example to intercept y by considering the targets sequentially, the total control effort required using this approach is increased by 42 percent over the minimum control effort obtainable by considering the targets simultaneously.

Figure 4.4 demonstrates the optimal intercept of two maneuvering targets. At $t = t_0 = 0$, the missile is directly behind the two targets, and if the targets do not maneuver, the missile can intercept both of them with zero control effort. From this position, the targets perform an evasive maneuver which consists of their turning in opposite directions. In order to determine the optimal intercept trajectory, it must be assumed that the evasive maneuver is known in advance by the missile. With this information, the optimal solution is readily obtained via the method of steepest descent, and it is found to be of the same general form as has been obtained for the straight-running targets in Figs. 4.2 and 4.3. Actually, this is to be expected, since if the target motion is known a priori, the optimal control is exactly the same for any target motions for which the optimal intercept points coincide, regardless of the maneuvers each of the targets has performed in arriving at these intercept points.

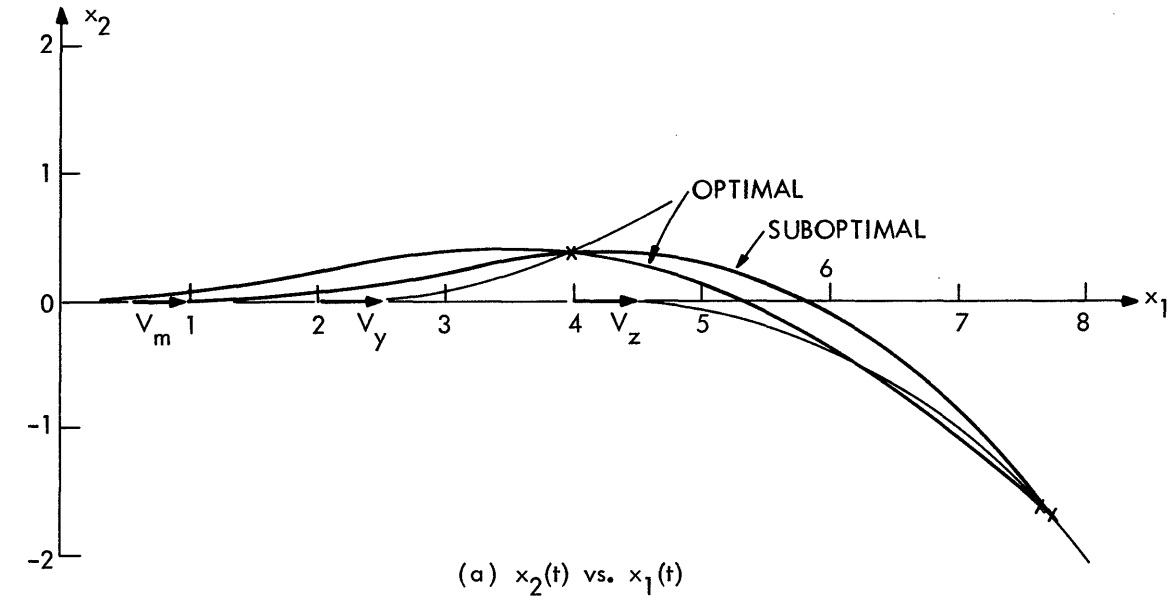


(a) $x_2(t)$ vs. $x_1(t)$

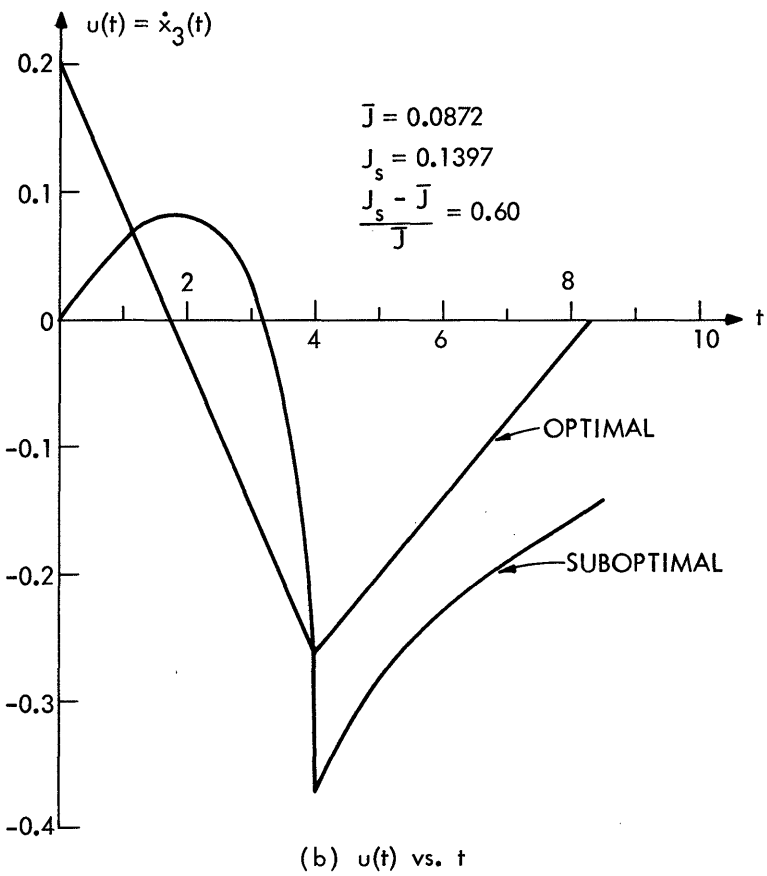


(b) u(t) vs. t

Fig. 4.3 Optimal and Suboptimal Solutions-
Nonmaneuvering Targets



(a) $x_2(t)$ vs. $x_1(t)$



(b) $u(t)$ vs. t

Fig. 4.4 Optimal and Suboptimal Solutions-
Maneuvering Targets

The above discussion has assumed that the future motion of the two targets is known exactly in order that the optimal solutions may be computed numerically. Actually, such information is not usually available, and one must resort to making a guess at the most likely evasive maneuvers. One could, perhaps, take the differential games approach[50,51] in which one attempts to find not only the missile control function that minimized J, but also the target evasive maneuvers that will maximize J. The mathematical formulation of this point of view involves the solution of a min-max problem, rather than simply a minimization problem, and while much research is being conducted on such problems, there are a number of theoretical questions which remain to be solved. The approach taken above is to separate the min and max operations by assuming the form of the optimal evasive maneuvers and using this information to determine the corresponding optimal intercept control law. However, even this approach is lacking, since one necessarily computes an open-loop control which is obviously inadequate if the targets perform evasive maneuvers other than those assumed. In this case one must recompute the optimal open-loop control each time a better estimate of the future motion of the targets is obtained. Such a method can be prohibitively expensive to implement, even if one could develop a numerical algorithm fast enough to follow rapid maneuvers of the targets. As noted in Section 3.3b, the solution of the optimal two-target intercept problem via the steepest-descent algorithm* requires approximately 1.5 seconds per iteration on an IBM 360/65. Thus, if the target maneuvers cause the optimal open-loop trajectory to change significantly so that more than one iteration is required to compute the new optimal, several seconds may elapse during which one has only the out-dated open-loop control function to guide the missile. In addition to the requirements on the speed of solution, the steepest-descent algorithm (or any other iterative method) requires a large computer memory since several variables (see Section 3.3b) must be stored as functions of time along the entire open-loop trajectory.

---

* Reasons for the selection of this algorithm over the Newton-Raphson one for this problem are contained in Section 3.5.

Various second-order numerical algorithms[30-32] yield, as a by-product of the optimal open-loop control and trajectory, a perturbation feedback control law valid for "small" deviations from the optimal solution. However, in "game" type problems, one cannot expect the opponent to "play along" by allowing only those small deviations from the optimal open-loop intercept trajectory for which the perturbation feedback control law is valid. If target maneuvers cause a significant change in the intercept trajectory, one is once again left temporarily without a valid control law. An evaluation of perturbation guidance was not carried out for the present problem.

When the future target motions are not known with certainty or are subject to change quite arbitrarily at the whim of an opponent, one requires a feedback control law. As discussed in Sections 2.6 and 4.2, a feedback control law can be computed for two-target problems if the system dynamics and state constraints are linear, and if the cost functional is quadratic. The approach presented in the next section is to make an appropriate change in the definition of the state variables, in terms of a nominal intercept path based on assumed future target motion, in order to obtain a double integrator model. The results of Section 4.2 are then applied as a "suboptimal" feedback control law.

4.4 SUBOPTIMAL MULTIPLE TARGET INTERCEPT GUIDANCE

Consider the two-target missile guidance problem illustrated in Fig. 4.5 where the vectors, V_m, V_y, and V_z indicate the position, heading, and velocity of the missile and the targets y and z, respectively. Since, in general, one does not know what the future motion of the target will be, an estimate of the most probable target maneuvers must be made. For the purpose of demonstrating the development of the suboptimal control law, it is assumed that the targets continue to move on straight-line paths. Under this assumption, the results of Section 4.3 indicate that the optimal trajectory will be of the form given by $\bar{x}(t)$ in Fig. 4.5. Since the computation of this optimal path each time the targets maneuver is undesirable, let us instead determine approximate intercept points by considering the piecewise linear intercept path indicated by $\hat{x}(t)$. In order to obtain $\hat{x}(t)$ it is assumed that the missile changes heading instantaneously and proceeds on a straight-line

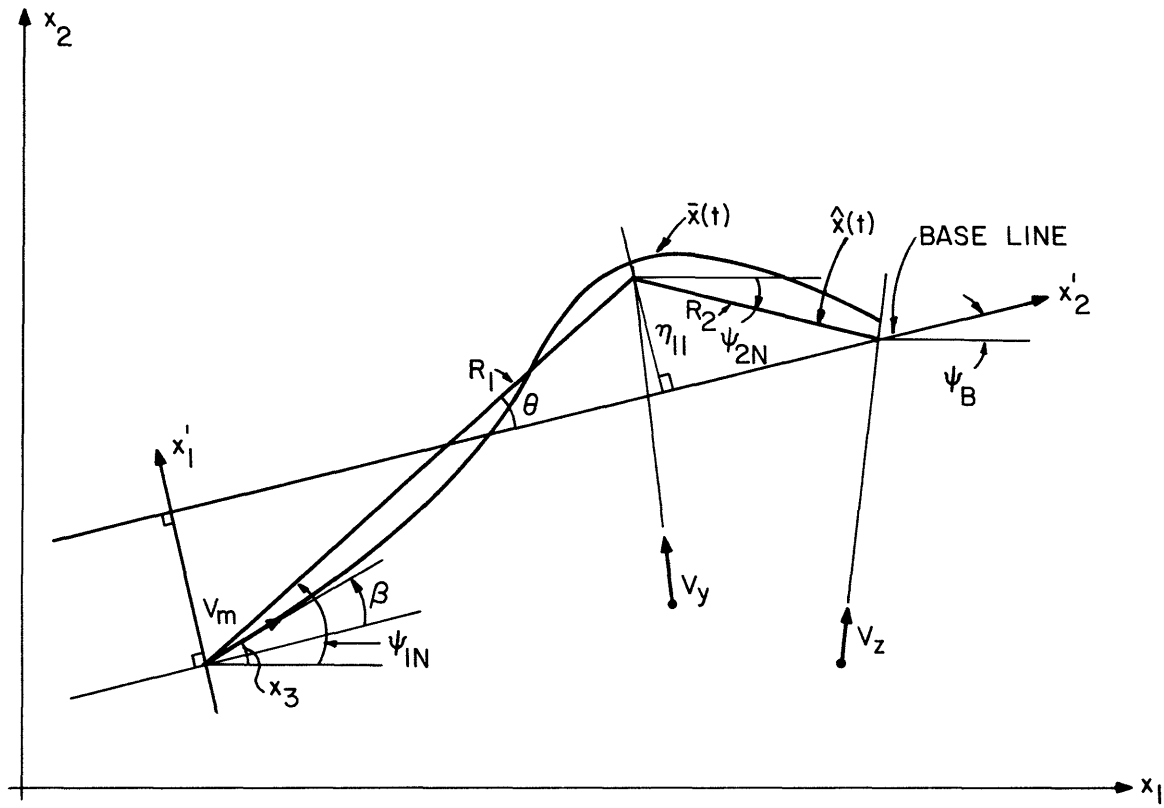


Fig. 4.5 Nominal Intercept Trajectory and Definition of Base line

intercept path to each of the targets, first intercepting y and then z. It is also assumed that the targets maintain constant headings and velocities. For the particular configuration presented in Fig. 4.5, it appears that a linearization of the system dynamics about $\hat{x}(t)$ should produce a fairly accurate model of the system. Rather than linearizing Eqs. 4.25 directly, let us consider the motion of the missile with respect to the coordinate system $x_1' - x_2'$ in Fig. 4.5. The x_2' axis (hereafter called the base line) is chosen to pass through the second nominal intercept point at an angle ψ_b * which is given by a

$$\psi_b = \frac{1}{2}(\psi_{1N} + \psi_{2N}) \tag{4.31}$$

where ψ_{1N} and ψ_{2N} are the angles of the nominal intercept lines to targets y and z, respectively. If $(\psi_{1N} - \psi_{2N})$ is small, the velocity of the missile in the x_2' direction along the trajectory $\bar{x}(t)$ is approximately constant. Let us define the state variable $\eta_1(t)$ as the x_1' coordinate of the missile. Then

$$\begin{aligned} \dot{\eta}_1(t) &= \eta_2(t) && \text{a)} \\ \dot{\eta}_2(t) &= v(t) && \text{b)} \end{aligned} \tag{4.32}$$

where η_2 is the velocity of the missile in the x_1' direction given by

$$\eta_2(t) = V_m \cdot \sin(x_3 - \psi_b) \tag{4.33}$$

In order to "intercept" both of the nominal intercept points, it is necessary to bring $\eta_1(t)$ to the value η_{11} at t_1 and to zero at t_2, where

$$\eta_{11} = R_2 \cdot \sin\left[\frac{1}{2}(\psi_{1N} - \psi_{2N})\right] \tag{4.34}$$

η_1 can be computed from

$$\eta_1 = \eta_{11} - R_1 \cdot \sin(x_3 - \psi_b) \tag{4.35}$$

---

* This angle was chosen for symmetry reasons from the allowable set of angles which have the property that the missile velocity, along the straight-line intercept trajectory, in the x_2' direction is always positive.

where R_1 and R_2 are the distance between the missile and the first nominal intercept point and the distance between the two nominal intercept points, respectively. That is, $\eta(t)$ must satisfy the boundary conditions

$$\psi^1[\eta(t),t] = \begin{bmatrix} \eta_1(t) - \eta_{11} \\ t - t_1 \end{bmatrix} = 0 \tag{4.36}$$

and

$$\psi^2[\eta(t),t] = \begin{bmatrix} \eta_1(t) \\ t - t_2 \end{bmatrix} = 0 \tag{4.37}$$

By requiring that $v(t)$ minimize a "control energy" cost functional

$$J = \frac{1}{2}\int_{t_0}^{t_2} [v(t)]^2 \, dt \tag{4.38}$$

the results of the double integrator example in Section 4.2 can be used to obtain the optimal control law for $v(t)$. In particular, from Eqs. 4.22 and 4.23,

$$\begin{aligned} \bar{v}^1(t) = &- \frac{6\cdot[3\cdot(t_1-t)+2\cdot(t_2-t_1)]}{(t_1-t)^2\cdot[3\cdot(t_1-t)+4\cdot(t_2-t_1)]}\cdot\eta_1(t) \\ &- \frac{12\cdot[(t_1-t)+(t_2-t_1)]}{(t_1-t)\cdot[3\cdot(t_1-t)+4\cdot(t_2-t_1)]}\cdot\eta_2(t) \\ &+ \frac{12\cdot[(t_1-t)+(t_2-t_1)]\cdot[(t_1-t)+2\cdot(t_2-t_1)]}{(t_2-t_1)\cdot(t_1-t)^2\cdot[3\cdot(t_1-t)+4\cdot(t_2-t_1)]}\cdot\eta_{11} \end{aligned} \tag{4.39}$$

for $t_0 < t < t_1$, and

$$\bar{v}^2(t) = -\frac{3}{(t_2-t)^2}\cdot\eta_1(t) - \frac{3}{(t_2-t)}\cdot\eta_2(t) \tag{4.40}$$

for $t_1 < t < t_2$. From Eq. 4.33, one can relate $v(t)$ to $u(t)$ by

$$u(t) = \dot{x}_3(t) = v(t)/V_m \cdot \cos [x_3(t) - \psi_b] \qquad (4.41)$$

Equations 4.33 through 4.41 constitute a "suboptimal" feedback control law for the missile turning rate control. This solution has the advantage, over any solution based on linearizing about the optimal trajectory, that the nominal intercept points that are "aimed at" are obtained much more easily than the actual optimal intercept points. This fact, coupled with the simplification obtained by reducing the model to a double integrator, yields a feedback control law which can be used in "real time" against maneuvering targets.

The above suboptimal two-target intercept control law has been simulated on an IBM 360/65. The computer program is listed in Appendix F, and a flow chart of the program is given in Fig. 4.6. As in the steepest-descent algorithm discussed in Chapter III, the program is written in Fortran IV using single precision arithmetic (seven digits of precision), and the integration of the missile dynamics and target motions is performed using a second-order Runge-Kutta algorithm. Figures 4.2 and 4.3 demonstrate the application of this control law to the intercept of two straight-running targets and provide a comparison with the optimal intercept solutions discussed in Section 4.3. As can be seen from these figures, the difference between the optimal and suboptimal trajectories gets larger as the range of the missile yaw angle, $x_3(t)$, along the trajectory increases, thereby increasing the errors due to the simplified model. An integration step-size of $\Delta t = 0.05$ has been used on all trajectories. With this step size the accuracy of the second-order Runge-Kutta integration algorithm is better than four significant figures, as shown in Section 3.3b. At every Δt a new nominal intercept path is computed, redefining the base line with respect to which $\eta_1(t)$, $\eta_2(t)$, and η_{11} are measured. The base line changes each Δt due to the target maneuvers as well as to the errors inherent in the simplified model.* Since the base line is updated in a step-like fashion, the

---

* In addition, the measurement noise which has been neglected in this investigation will cause the base line to change each Δt.

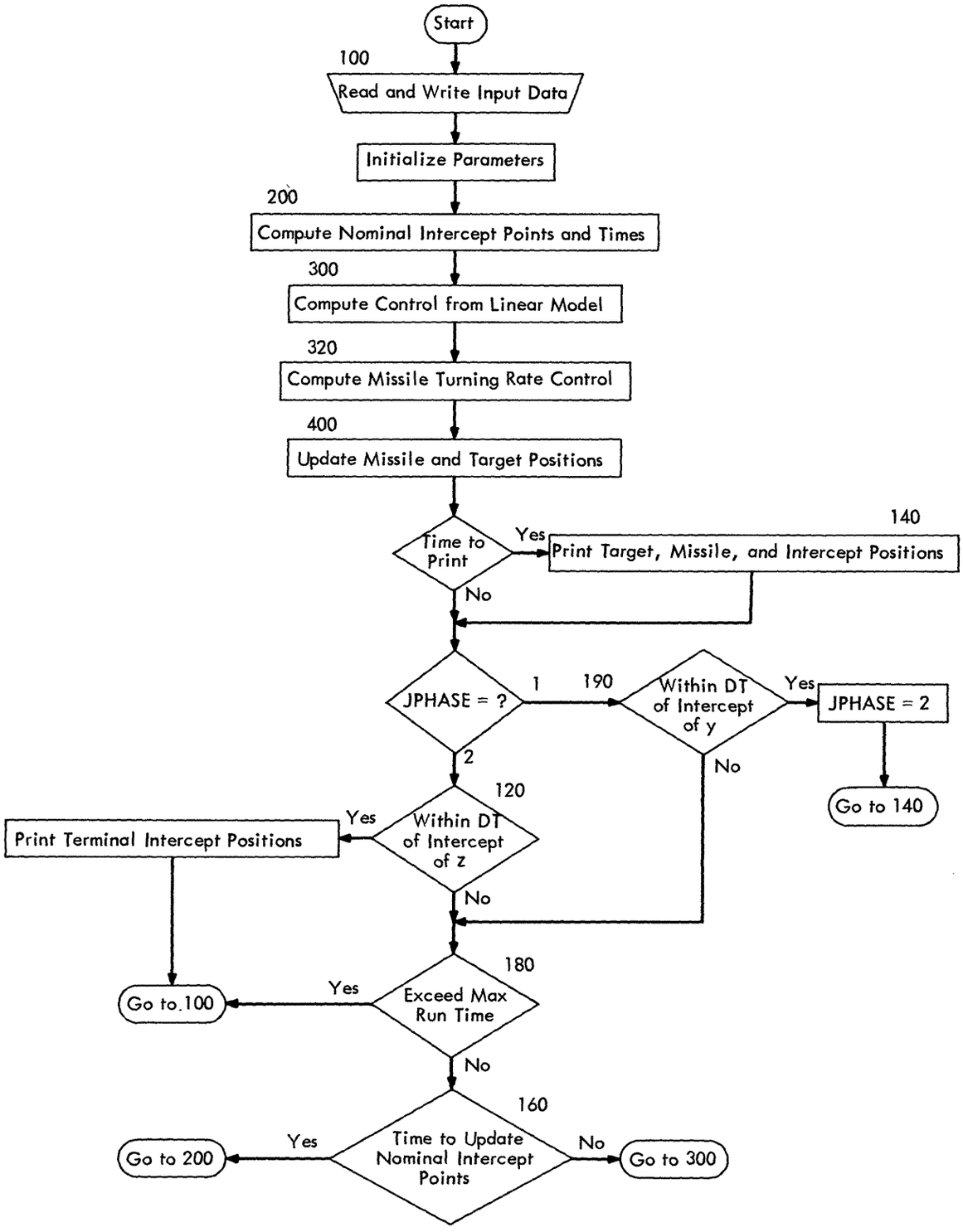


Fig. 4.6 Flow Chart for Suboptimal Computer Program

turning-rate applied to the vehicle has been made a piecewise continuous function of time, taking a new value each time the new base line is computed. With this integration step-size, the solution of the suboptimal trajectory can be achieved in considerably less than real time.* For the example in Fig. 4.2 the control effort required by the suboptimal control law is $J_s = 0.230$, in comparison with the minimal control effort of $\overline{J} = 0.227$, an increase of only 1.3 percent. For the example in Fig. 4.3 the increase in control effort is 4.5 percent, in comparison with an increase of 42 percent for the sequential optimal solution that is discussed in Section 4.3.

The application of the suboptimal control law to the intercept of two maneuvering targets is demonstrated in Fig. 4.4. While one expects the simplified model to be quite accurate for this example (since $x_3(t)$ varies only over the range $-0.95 < x_3 < 0.18$ radians), it can be seen that the difference between the optimal and the suboptimal solutions is significant. This difference is due to the fact that the optimal solution "knows" what the future target motion will be, while the suboptimal solution assumes only that at each point in time the targets will continue on straight-line paths. The performance of the suboptimal solution could perhaps be improved in this example by extrapolating the past motion of the targets to predict circular rather than straight-line paths. However, since the future motion of the targets is arbitrary, such a prediction could, in general, do more harm than good. For this particular example, the control effort required by the suboptimal control law is 60 percent higher than the minimal amount obtainable, and this increase is due mainly to the lack of knowledge of the future target motion.

While the above suboptimal control law is applicable to a wide range of target intercept situations, its performance tends to deteriorate for trajectories in which the angle between the missile heading and the base line, $(x_3 - \psi_b)$, approaches $\pi/2$ radians. As the difference between the angles ψ_{1N} and ψ_{2N} increases, the motion of the missile

---

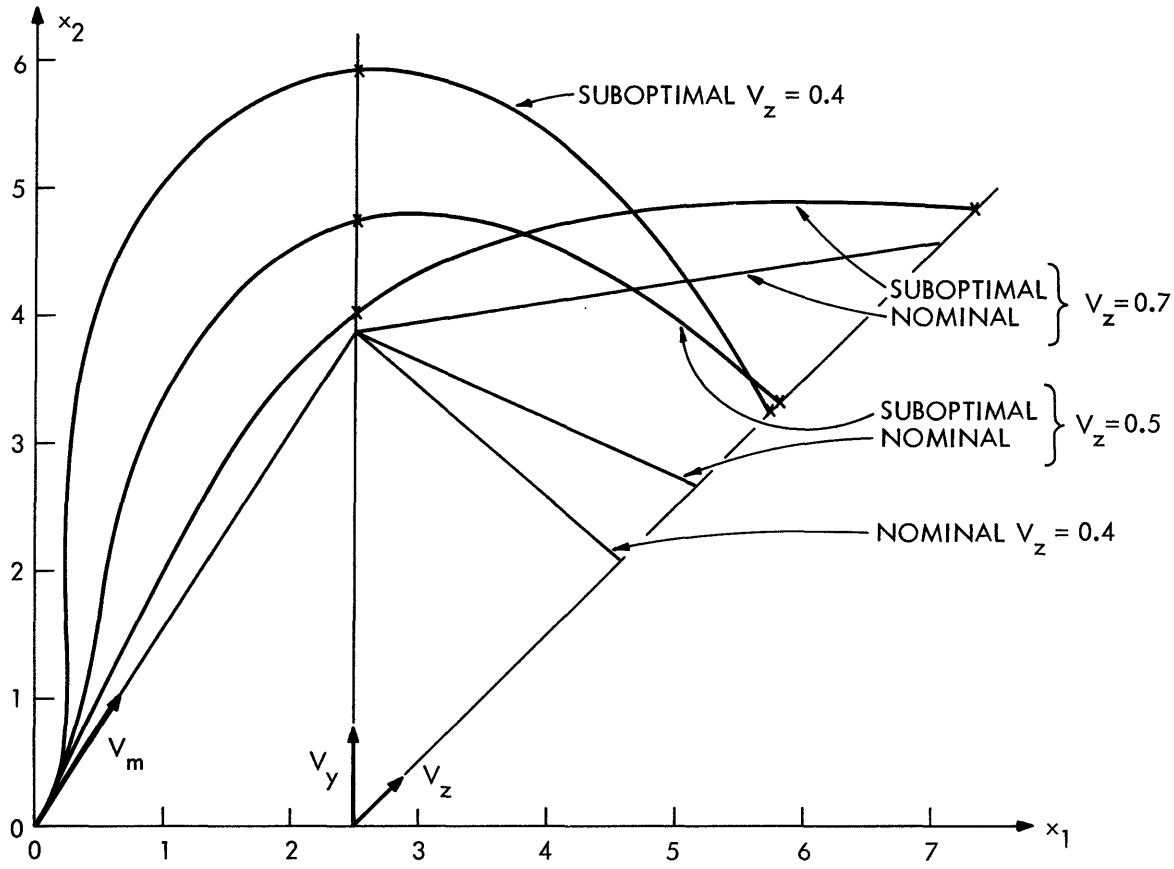
* The computation time required is 0.2 seconds for every second of real time. This includes printing out the missile and target trajectories and the updated nominal intercept trajectory once every 0.25 seconds of real time.

tends to be more transverse to the base line and the optimal solution to the double integrator model calls for a "velocity" $\eta_2(t)$ that is greater than can be achieved with the missile velocity, V_m. In this case, Eq. 4.41 is no longer a valid means for computing $u(t)$ since it specifies large turning rates when, in fact, no further increases in η_2 can be achieved. To handle this situation, a somewhat artificial modification has been made. If $(x_3 - \psi_b)$ becomes greater than $(\pi/2-\epsilon)$, (for the results presented here the value of $(\pi/2-\epsilon)$ has been arbitrarily set to 1.4 radians), then Eq. 4.41 is used only if a negative $u(t)$ is computed.* Otherwise, $u(t)$ is computed so as to maintain $(x_3(t) - \psi_b)$ at a value of $(\frac{\pi}{2} - \frac{\epsilon}{2})$. The suboptimal trajectories resulting from this modification have been determined for the same initial condition as in Fig. 4.3, but with velocities of 0.7, 0.5, and 0.4 for target z. As can be seen from Fig. 4.7, the angle between the initial nominal intercept paths, $\psi_{1N} - \psi_{2N}$, increases and the accuracy of the initial nominal intercept points decreases as V_z decreases. The increase in the control effort required by the suboptimal solutions over the minimal control changes from 4.55 percent to 5.9 percent when V_z is reduced from 0.6 to 0.5; but in further reducing the velocity to 0.4 (causing $(x_3 - \psi_b)$ to exceed 1.5 radians), the control effort required jumps to 15 percent higher than the minimal. However, as $(\psi_{1N} - \psi_{2N})$ increases, it becomes more likely that a lower level of control effort is required in attacking the targets in reverse order. Figure 4.8 illustrates the trajectories resulting from the suboptimal control law for $V_z = 0.4$ in attacking the targets in either order. While the total time-to-intercept increases from 11.5 sec to 28.7 sec in attacking target z first**(due to the higher velocity of y), the control effort required is reduced

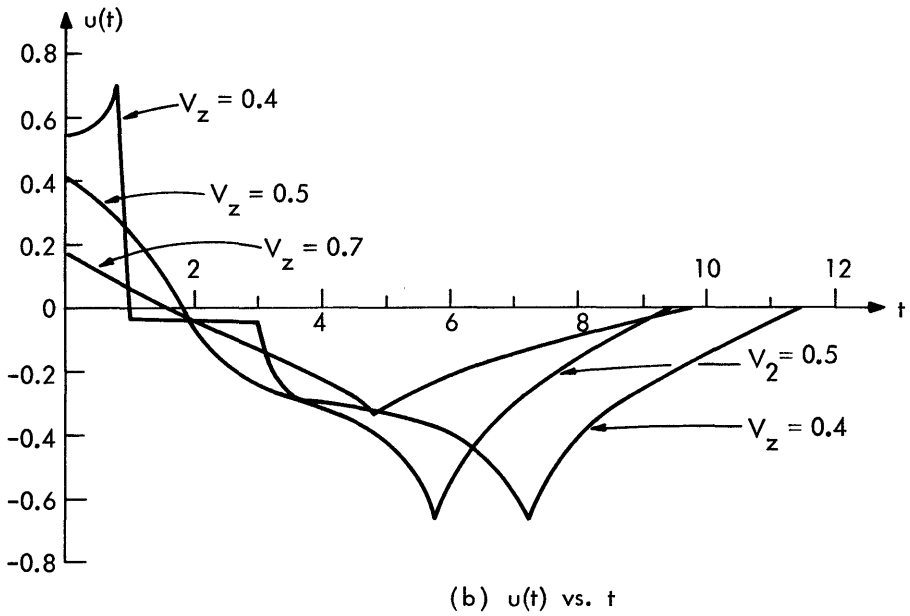
---

* The converse is true, of course, for $(x_3-\psi_b)$ less than $-(\frac{\pi}{2} - \epsilon)$.

** The optimal trajectory for the reverse target ordering has not been determined. In order to compute it the steepest-descent program listed in Appendix D must be modified so that the nominal intercept times $t_{1,j}$ and $t_{2,j}$ can be correctly defined at each iteration.

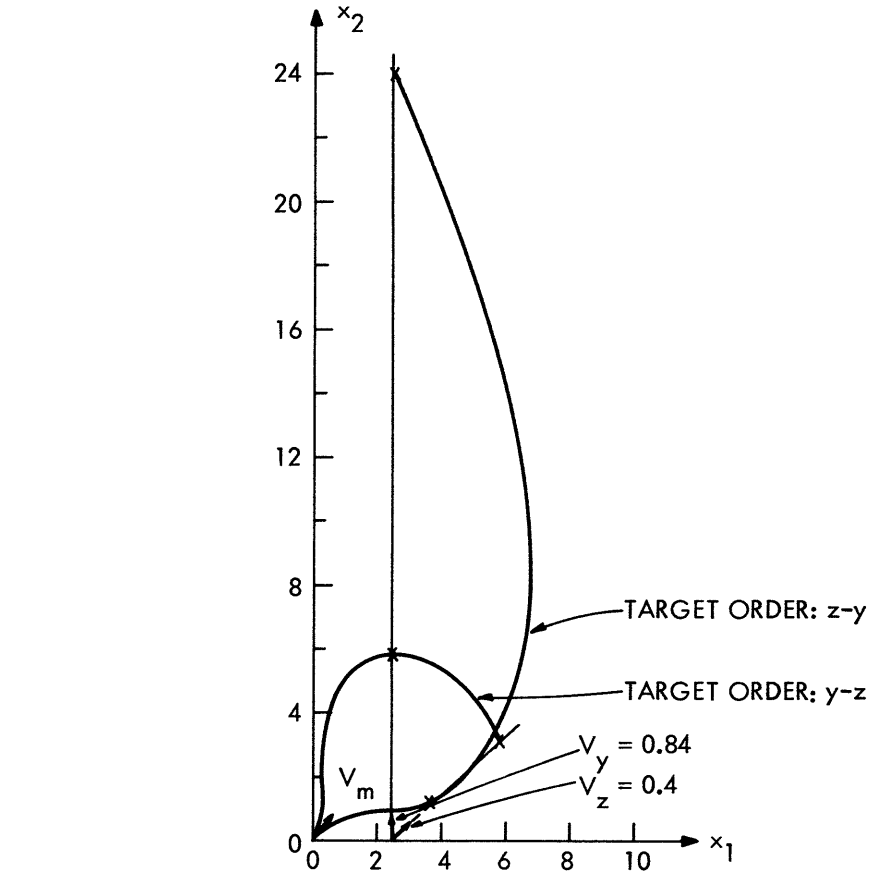


(a) $x_2(t)$ vs. $x_1(t)$

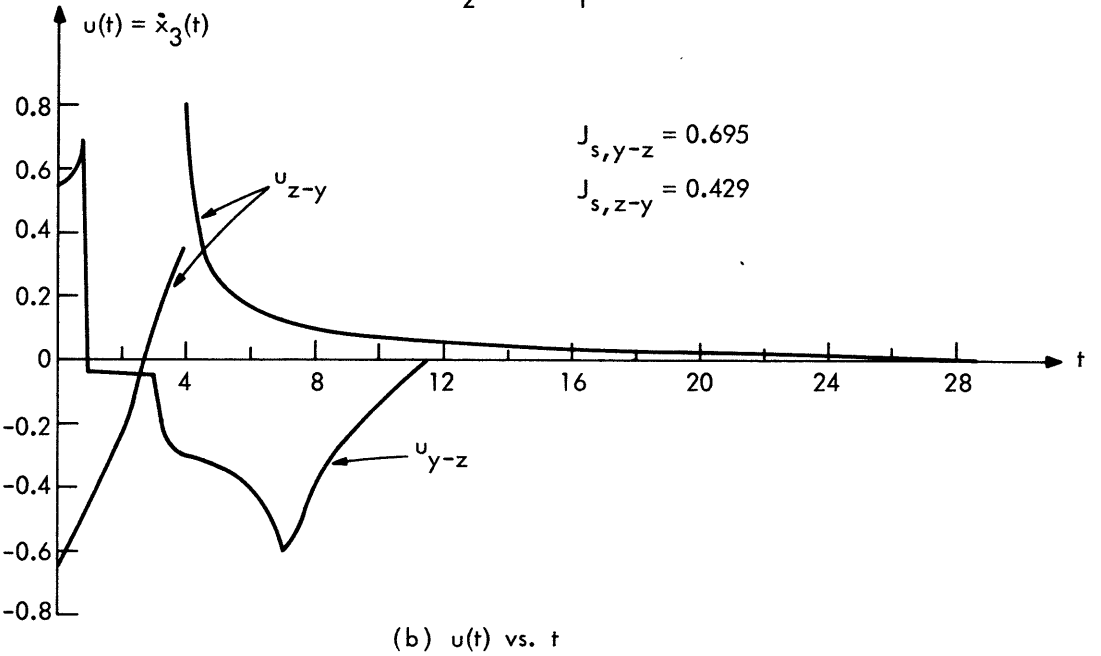


(b) u(t) vs. t

Fig. 4.7 Suboptimal Controls and Trajectories for Several Values of V_z



(a) $x_2(t)$ vs. $x_1(t)$



(b) u(t) vs. t

Fig. 4.8 Suboptimal Controls and Trajectories for Target Orders y-z and z-y

from $J_{s,y-z} = 0.695$ to $J_{s,z-y} = 0.429$.

While much testing is required in order to more fully evaluate the performance of the suboptimal control law, it appears that it requires a control effort within ten percent of the minimum as long as the angle $(x_3 - \psi_b)$ does not equal $\frac{\pi}{2}$ radians along the optimal trajectory. If this angle exceeds $\frac{\pi}{2}$ (or more appropriately $\frac{\pi}{2} - \epsilon$) the control effort required can be significantly higher than the minimum. However, in the light of practical considerations which are not taken up here, the strategy of optimally intercepting both targets becomes questionable in such cases.

4.5 AREAS FOR FURTHER RESEARCH

The discussion in this chapter has been concerned with the development of a feedback control law for the intercept of two maneuvering targets. Many interesting facets of the problem have been touched upon only briefly, or not at all, due to the introductory nature of this investigation, and these are left for future research. The system dynamics considered have neglected several important physical characteristics such as induced drag, turning-rate constraints, and field-of-vision constraints, which exist in any practical situation. In addition, the assumption of perfect knowledge of the state of motion of the targets may be unrealistic in a given application, and thus the effect of noisy or incomplete measurements of the target motions on the performance of the control law must be determined.

While these constitute a few of the physical properties to be considered, which depend on the particular application, other areas open for further research deal with the development of the control law itself. For example, the choice of the cost functional has been motivated in part by its association with the proportional control law for single target intercept problems (which relates the turning-rate control to the rate of change of the line-of-sight angle to the target). One may wish to develop a similar control law for two-target intercept which relates the turning-rate control to the physical quantities actually measured -- the ranges and line-of-sight angles to the targets. A desirable property of the proportional control law for a single target is that for non-maneuvering targets the control required decreases as one gets closer to the target,

reducing the possibility of the target "out maneuvering" the missile at the last second. Unfortunately, in the two-target problem the control effort required often increases as one approaches the first target. This may be intolerable when one includes a turning-rate constraint on the missile since the first target may be able to maneuver so that the missile can no longer turn fast enough to intercept it. One possible approach to alleviating this problem is to include in the cost functional a time-varying weighting factor which penalizes the control more as the first target is approached. An alternate approach is to use as the estimate of the future target motion the worst (from the missile's viewpoint) maneuvers possible. In the latter case, one is touching upon the differential games formulation of the intercept problem.

Finally, the suboptimal control law, as developed, must be more thoroughly evaluated with respect to the conditions under which its performance deteriorates. This question is coupled to the choice of the order in which the targets are to be intercepted, and, in fact, to the decision of whether or not to forego the two-target intercept (concentrating the attack on only one of the targets if it becomes too "costly" to attack both). The dependence of the performance on the frequency of the updating of the nominal intercept solution is, of course, of great interest. A significant decrease in the frequency can perhaps be achieved by relaxing the restriction of the control to piecewise constant functions of time by using Eqs. 4.22 and 4.23 throughout the interval, $t_i < t < t_i + \Delta t$, and the base line computed at t_i. However, the answer to this question is dependent on the nature of the measurement uncertainty and target maneuverability, and thus it will vary from one application to another.

4.6 SUMMARY

In this chapter the problem of controlling a missile to intercept two maneuvering targets has been considered. The numerical techniques discussed in Chapter III, in particular the steepest-descent algorithm, have been used to compute intercept trajectories that minimize a quadratic measure of the control effort expended, for both maneuvering and non-maneuvering targets. In computing the optimal solutions it is necessary to assume complete knowledge of the future motion of each of the targets. Since the iterative techniques for solving optimal control

problems result in open-loop control functions, it becomes impractical to incorporate them into guidance laws against maneuvering targets. Whenever the targets maneuver, thus changing one's estimate of the future target motion, a new optimal open-loop solution must be obtained, resulting in a period of time during which one does not have a valid control law.

Against maneuvering targets one requires a feedback control law that enables a rapid compensation for the targets' evasive maneuvers. To accomplish this, a "suboptimal" feedback control law is obtained based on the optimal solution to a double-integrator, minimum control energy problem with state constraints at two points in time. As a result of the linearity of the double integrator dynamics and the state constraints (and also the quadratic cost functional which results in a linear set of Euler-Lagrange equations), the optimal solution can be obtained as a time-varying feedback control law. This control law is such that when the times to go to each of the boundary points are of the same order of magnitude, the control is a function of each of the boundary conditions; but, as the time to the first boundary point becomes much smaller, the control law asymptotically approaches the optimal control law for the intermediate boundary point alone. After the first target has been intercepted, the control law becomes that which one would get if the terminal boundary condition alone were imposed (as one would expect).

The optimal solution to the double integrator problem is then incorporated into a suboptimal feedback control law for the two-target intercept problem. This is done by constructing a double integrator model based on a set of nominal intercept points. These points are determined by assuming that the targets continue to move on straight lines at constant velocities and solving for the straight line paths for the missile to follow in order to achieve intercept of both of the targets. The suboptimal control law has been simulated on an IBM 360/65 and tested against several sets of target motions, both maneuvering and non-maneuvering. In the case of non-maneuvering targets, it is found that the control law is very nearly optimal for trajectories for which the nominal intercept paths provide a good approximation to the optimal

trajectory, and that its performance deteriorates as the accuracy of this approximation decreases. There are, in fact, conditions under which the control law must be augmented since the simplified model is no longer valid. However, such situations can usually be overcome by simply reversing the order in which the targets are to be intercepted. Furthermore, in such situations the strategy of optimally attacking both targets probably would not be used because of practical reasons not considered. In the case of maneuvering targets, the difference between the optimal and suboptimal trajectories can be significant in spite of the validity of the simplified model since the optimal solution "knows" the target maneuvers in advance, while the suboptimal control law assumes that the targets will perform no further maneuvers. Thus, the use of optimal trajectories as references for evaluating the suboptimal trajectories is, in a sense, unrealistic since the optimal solution presupposes information which cannot possibly be known with certainty.

APPENDIX A

THE CALCULUS OF VARIATIONS AND OPTIMAL CONTROL

A.1 INTRODUCTION

Results of the classical variational theory which are used in Chapter I are presented in this appendix. The primary results of interest are the equation for the general first variation, Eq. A.20; the extension to the n-dimensional case, Section A.5; and the theorem for treating differential constraints, Section A.6. These results are applied in Section A.7 to the solution of the optimal control problem. The Weierstrass-Erdmann corner conditions are also developed, in Section A.4, because of the similarity between their development and the development of the intermediate transversality conditions in Chapter II. In fact, the intermediate transversality conditions can be regarded as direct extensions of the Weierstrass-Erdmann corner conditions.

What is known as the "simplest problem of the calculus of variations" consists of determining the function $x(t)$, or rather the conditions which it must necessarily satisfy, such that a functional of $x(t)$ and t of the form

$$J = \int_{t_0}^{t_f} F[x(t), \dot{x}(t), t]\, dt \tag{A.1}$$

is minimized subject to the boundary conditions on $x(t)$ given by

$$x(t_0) = x_0 \quad ; \quad x(t_f) = x_f \tag{A.2}$$

The development of these necessary conditions requires that certain continuity restrictions be placed on $F[x, \dot{x}, t]$ and $x(t)$. The term "calculus of variations" is used since the point of view taken is that one has some $\bar{x}(t)$ which satisfies the boundary conditions and yields a value of J less than that produced by any "neighboring" $x(t)$ which also satisfies the boundary and continuity conditions (such $x(t)$ are termed "admissible" functions). One then considers small "variations" of $\bar{x}(t)$, i.e., those admissible $x(t)$ which are "close" to $\bar{x}(t)$, and

requires that

$$J[\bar{x}(t)] \leq J[x(t)] \tag{A.3}$$

for all admissible $x(t)$. The approach is analogous to the ordinary calculus problem of determining the minimum of a function whose first derivative exists and is continuous by finding those points at which its first derivative is zero and its second derivative is greater than or equal to zero. In the variational problem one considers the first and second variations of J (terms involving the variation of x to first and second order, respectively) and requires that the first variation be zero and the second variation be non-negative. The first order necessary conditions are called the Euler-Lagrange equations, and conditions based on the second variation are the Legendre and the Jacobi conditions. Several good texts[1-4,7] are available for the details of the results summarized in this appendix.

The first order conditions given by the Euler-Lagrange equations are often adequate to solve a given physical problem, since various physical arguments may be given to assure that the solution obtained is a relative minimum rather than a relative maximum. Therefore, if a unique $x(t)$ is obtained, it can be reasoned that it must be the desired solution. It must be emphasized that the minimum is with respect to the class of admissible functions, and there may very well exist some $x(t)$ which, though not of the admissible class, yields an even smaller value of J. An example of such a case is given in Section A.4. In that section the Weierstrass-Erdmann corner conditions are determined which allow an expansion of the class of admissible functions. In addition, the Weierstrass E-function can be used to consider stronger variations of $x(t)$ (i.e., variations for which $\dot{x}(t)$ may not be close to $\dot{\bar{x}}(t)$ even though $x(t)$ is close to $\bar{x}(t)$). With this brief heuristic insight into the calculus of variations, the simplest problem is now formally presented and solved.

A.2 THE SIMPLEST PROBLEM—THE NECESSARY CONDITIONS

Let $F(x, \dot{x}, t)$ be a function with continuous first and second partial derivatives with respect to each of its arguments; let X be the set of all functions $x(t)$ satisfying the boundary conditions of Eq. A.2 which are continuous and have continuous first derivatives, $\frac{dx}{dt}$, for $t_0 < t < t_f$; and, let J be the functional given by Eq. A.1. The "simplest problem" of the calculus of variations is to determine the function $\bar{x}(t) \in X$ such that the functional J is minimized (assuming, of course, that such a function exists).

Let $\bar{x}(t)$ and $x(t)$ be elements of X, as shown in Fig. A.1, such that

$$x(t) = \bar{x}(t) + \delta x(t) \tag{A.4}$$

and

$$\max_{t \in (t_0, t_f)} |\delta x(t)| < \epsilon \tag{A.5}$$

$$\max_{t \in (t_0, t_f)} |\delta \dot{x}(t)| < \epsilon \tag{A.6}$$

where ϵ is an arbitrarily small number. That is, we are considering only "weak" variations in x for which the derivatives, as well as the functions themselves, must be close to that of $\bar{x}(t)$. From Eq. A.1 we have

$$\bar{J} = \int_{t_0}^{t_f} F[\bar{x}(t), \dot{\bar{x}}(t), t]\, dt \tag{A.7}$$

and

$$J = \int_{t_0}^{t_f} F[\bar{x}(t) + \delta x(t), \dot{\bar{x}}(t) + \delta \dot{x}(t), t]\, dt \tag{A.8}$$

Expanding the integrand of Eq. A.8 in a Taylor series about $\bar{x}(t)$ and considering only first order variational terms, we obtain the first variation of J, δJ,

$$\delta J = \int_{t_0}^{t_f} [\bar{F}_x(t) \cdot \delta x(t) + \bar{F}_{\dot{x}}(t)\, \delta \dot{x}(t)]\, dt \qquad (A.9)$$

where the subscripts denote partial differentiation with respect to the subscripted variable and the overbar denotes evaluation along $\bar{x}(t)$, e.g., $\bar{F}_x(t) = \frac{\partial F}{\partial x}[\bar{x}(t), \dot{\bar{x}}(t), t]$. Integrating the second term in the integrand of Eq. A.9 by parts yields

$$\delta J = \bar{F}_{\dot{x}}(t) \cdot \delta x(t)\Big|_{t_0}^{t_f} + \int_{t_0}^{t_f} [\bar{F}_x(t) - \frac{d}{dt}\bar{F}_{\dot{x}}(t)]\, \delta x(t)\, dt \qquad (A.10)$$

However, since $\bar{x}(t)$, $x(t) \in X$, by Eq. A.4 it is necessary that

$$\delta x(t_0) = \delta x(t_f) = 0 \qquad (A.11)$$

as indicated in Fig. A.1. It can be proven that a necessary condition in order that J be a minimum is that δJ be zero, and since the variation δx is arbitrary, it is necessary that

$$F_x[\bar{x}(t), \dot{\bar{x}}(t), t] - \frac{d}{dt} F_{\dot{x}}[\bar{x}(t), \dot{\bar{x}}(t), t] = 0 \qquad (A.12)$$

If Eq. A.12 were not true, then $\delta x(t)$ could be chosen to have the same sign as the left hand side of Eq. A.12, and thus δJ would necessarily be greater than zero. Equation A.12 is referred to as the Euler-Lagrange equation, and it must be stressed that it is only a condition which the minimal $\bar{x}(t)$ must satisfy (assuming such a function exists) -- it will also be satisfied by all local minima from which the absolute minimum must be determined, and by all local maxima and saddle points. In addition, there may exist an $x(t)$ not in X which satisfies the boundary conditions and yields a smaller value of J. These points must be kept in mind when extending these results to the optimal control problem.

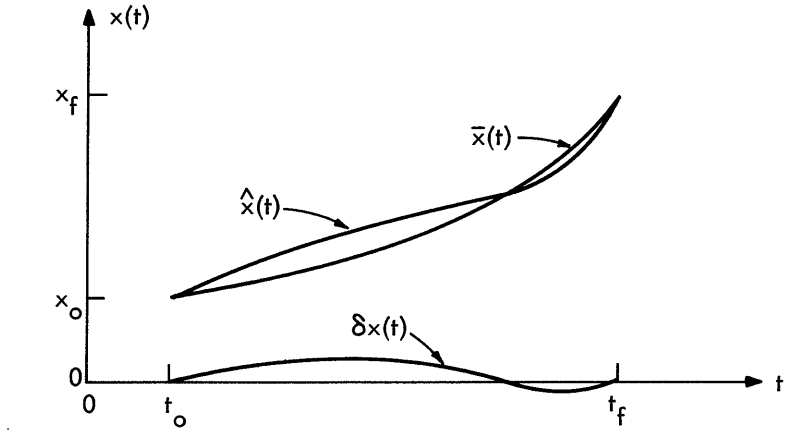


Fig. A.1 Admissible Variation About Optimal,
$\hat{x}(t) = \bar{x}(t) + \delta x(t)$

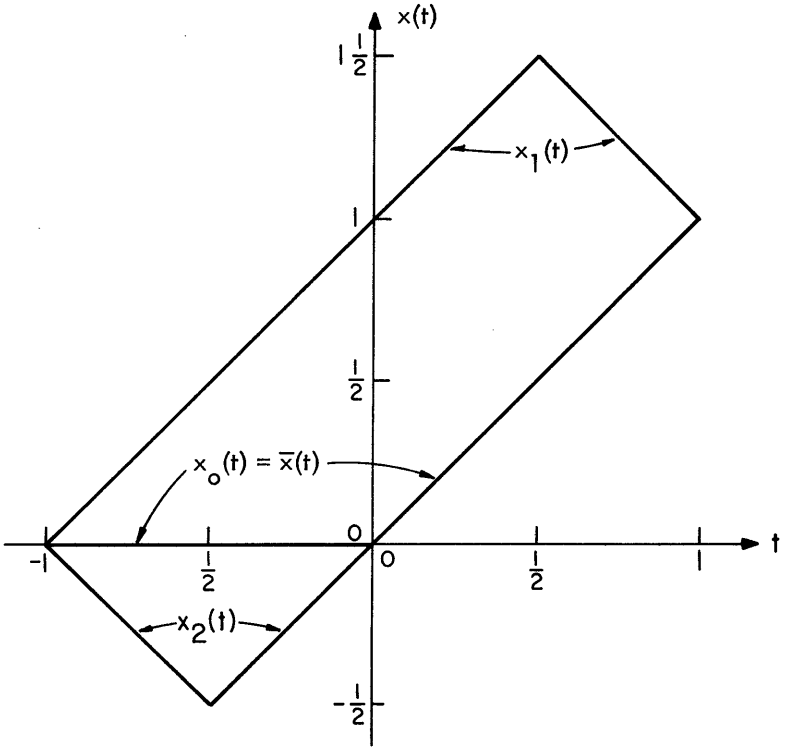


Fig. A.2 Example-Sect. A.4, Three Solutions to Euler-Lagrange Equations with Corners

If the boundary condition on $x(t)$ is unspecified at either (or both) of the end points, then $\delta x(t)$ need not be zero at that point. However, δJ must still be zero for those $\delta x(t)$ which are zero at the end points, and thus Eq. A.12 still follows. Then, from Eq. A.10, it is also necessary that

$$F_{\dot{x}}[\bar{x}(t_0), \dot{\bar{x}}(t_0), t_0] = 0 \qquad \text{if} \quad x(t_0) \text{ is free} \qquad (A.13)$$

$$F_{\dot{x}}[\bar{x}(t_f), \dot{\bar{x}}(t_f), t_f] = 0 \qquad \text{if} \quad x(t_f) \text{ is free} \qquad (A.14)$$

These are often called the natural boundary conditions. In Section A.7 the boundary conditions are generalized further for the optimal control problem and a set of "transversality conditions" is obtained, for which Eqs. A.13 and A.14 are a special case. In the following section, A.3, the initial and terminal times are allowed to be unspecified, and the general first variation is obtained which allows variations in any of the end conditions, including the initial and terminal times.

Carrying out the Taylor series expansion of Eq. A.8 to second order terms in $x(t)$ and $\dot{x}(t)$, one obtains the second variation of J,

$$\delta^2 J = \frac{1}{2}\int_{t_0}^{t_f} [\bar{F}_{xx}(t)\cdot \delta x^2(t) + 2\bar{F}_{x\dot{x}}(t)\cdot \delta x(t)\cdot \delta\dot{x}(t) + \bar{F}_{\dot{x}\dot{x}}(t)\delta\dot{x}^2(t)]\, dt \qquad (A.15)$$

A necessary condition for $\bar{x}(t)$ to yield a relative minimum of J is that the second variation, $\delta^2 J$, as given by Eq. A.15, be non-negative when evaluated along $\bar{x}(t)$ for any admissible $\delta x(t)$. Since Eq. A.15 involves all the admissible variations, $\delta x(t)$, this condition cannot be verified directly in practice, and therefore is of little value. One can show, however, that it is necessary that

$$F_{\dot{x}\dot{x}}[\bar{x}(t), \dot{\bar{x}}(t), t] \geq 0 \qquad \text{for} \quad t_0 < t < t_f \qquad (A.16)$$

a condition somewhat analogous to the necessary condition on the second derivative in ordinary calculus minimization problems. This condition is called the Legendre necessary condition. One can reason heuristically that if $\delta\dot{x}(t)$ is "small", then $\delta x(t)$ must be small, but that $\delta x(t)$ may be small for "large" $\delta\dot{x}(t)$. Thus the last term in the second variation is the

dominant one, resulting in Eq. A.16. While Eq. A.15 must be satisfied by all relative minima, it is possible that $F_{\dot{x}\dot{x}} = 0$ for some relative maxima also. One would hope that the strict inequality in Eq. A.16, $F_{\dot{x}\dot{x}} > 0$, might provide a sufficiency condition for a minimal solution (along with the necessary Euler-Lagrange equation), by extending the analogy with ordinary minimization problems. Since $F_{\dot{x}\dot{x}}$ constitutes only a part of the second variation, such a sufficiency condition is not directly forthcoming. However, if $\bar{x}(t)$ satisfies the Jacobi condition, i.e., if one can find a solution, $v(t)$, to the Jacobi equation

$$[\bar{F}_{\dot{x}\dot{x}}(t)] \cdot \frac{d^2v(t)}{dt^2} + [\frac{d}{dt}\bar{F}_{\dot{x}\dot{x}}(t)] \cdot \frac{dv(t)}{dt} + [\frac{d}{dt}\bar{F}_{x\dot{x}}(t) - \bar{F}_{xx}(t)]\, v(t) = 0 \tag{A.17}$$

such that $v(t_f) = 0$ and $v(t) \neq 0$ for any $t_0 < t < t_f$, then one can conclude that the satisfaction of the strengthened Legendre condition,

$$\bar{F}_{\dot{x}\dot{x}}(t) > 0 \quad \text{for all} \quad t_0 < t < t_f \tag{A.18}$$

does indeed constitute a sufficiency condition for $\bar{x}(t)$ to yield a relative minimum of J (for all "weak" variations of $\hat{x}(t)$ in the neighborhood of $\bar{x}(t)$). In addition, the Jacobi condition is a necessary condition for $\bar{x}(t)$ to be minimal.

The above discussion has dealt only with weak variations of $x(t)$. It is possible that one may find a solution, $\bar{x}(t)$, to the Euler-Lagrange equation, which also satisfies the Jacobi condition and the strengthened Legendre condition, but for which one can find a variation, $\delta\hat{x}(t)$, which satisfies Eq. A.5 but not Eq. A.6 and yields a smaller value of J. Such variations are called "strong" variations, since they form a wider class of variations, of which the weak variations are only a subset. To cope with such a possibility one can invoke the Weierstrass necessary condition (often called the Weierstrass E-function) that

$$E(\bar{x}, \dot{\bar{x}}, \dot{X}, t) = F(\bar{x}, \dot{X}, t) - F(\bar{x}, \dot{\bar{x}}, t) - (\dot{X} - \dot{\bar{x}})\, F_{\dot{x}}(\bar{x}, \dot{\bar{x}}, t) \geq 0 \tag{A.19}$$

a condition to be satisfied by all strong variations of $x(t)$ at every point along the solution $\bar{x}(t)$. The equality is required in order to insure the satisfaction of this condition by the minimal solution, but it may also be satisfied by a relative maximum. Thus Eq. A.19 is only

a necessary condition. Elimination of the equality will exclude all relative maxima, resulting in a sufficiency condition, but this may also exclude some relative minima. In Section A.7 it is shown that for the extension to optimal control problems, the Weierstrass E-function is identically equal to zero and thus, in its present form, does not provide a sufficiency condition.

A.3 THE GENERAL FIRST VARIATION

In some optimization problems one may wish to have either the initial or terminal time (or both) unspecified, to be determined optimally. By proceeding in a manner analogous to that in the last section, one can determine the first variation of J to be

$$\delta J = [\bar{F}_{\dot{x}} \cdot dx]\Big|_{\bar{t}_0}^{\bar{t}_f} + [(\bar{F} - \dot{\bar{x}}\bar{F}_{\dot{x}}) \cdot dt]\Big|_{\bar{t}_0}^{\bar{t}_f} + \int_{\bar{t}_0}^{\bar{t}_f} [\bar{F}_x(t) - \frac{d}{dt}\bar{F}_{\dot{x}}(t)] \cdot \delta x(t) \cdot dt \quad (A.20)$$

Since δJ must be zero even for variations δx for which $dx(\bar{t}_0) = dx(\bar{t}_f) = dt_0 = dt_f = 0$, one again obtains the Euler-Lagrange equation, Eq. A.12, which must be satisfied by the optimal $\bar{x}(t)$. Since each of the four end variations can be considered individually, one obtains, in addition to the natural boundary conditions of Eqs. A.13 and A.14, the added boundary conditions

$$F[\bar{x}(\bar{t}_0), \dot{\bar{x}}(\bar{t}_0), \bar{t}_0] - \dot{\bar{x}}(\bar{t}_0)F_{\dot{x}}[\bar{x}(\bar{t}_0), \dot{\bar{x}}(\bar{t}_0), \bar{t}_0] = 0 \quad \text{if } t_0 \text{ is free} \quad (A.21)$$

$$F[\bar{x}(\bar{t}_f), \dot{\bar{x}}(\bar{t}_f), \bar{t}_f] - \dot{\bar{x}}(\bar{t}_f)F_{\dot{x}}[\bar{x}(\bar{t}_f), \dot{\bar{x}}(\bar{t}_f), \bar{t}_f] = 0 \quad \text{if } t_f \text{ is free} \quad (A.22)$$

The general first variation of J, as given by Eq. A.20, is used in the next section to determine the Weierstrass-Erdmann corner conditions, and again in Section A.7 to determine the necessary conditions for the optimal control problem (these results are then extended to Chapter II to the intermediate target set problem).

A.4 WEIERSTRASS-ERDMANN CORNER CONDITIONS

The results of Sections A.2 and A.3 suffer from the restriction that $\bar{x}(t)$ and $\dot{\bar{x}}(t)$ must both be continuous functions of time. There is no assurance that an $x(t)$ not satisfying these requirements might not produce a lower value of J. It is also possible that there may be no admissible function producing a minimum, but that if the class of admissible functions is extended, then a minimum can indeed be achieved. This situation can be demonstrated by the following example. Let $F(x,\dot{x},t) = x^2(1-\dot{x})^2$, and $x(-1) = 0$, $x(+1) = 1$ be the boundary conditions, so that the functional J to be minimized is given by

$$J=\int_{-1}^{+1} x^2(t)\cdot[1-\dot{x}(t)]^2\,dt \tag{A.23}$$

Since both factors are square, it follows that $J \geq 0$ for any $x(t)$. The Euler-Lagrange equation for this problem is given by

$$x^2\cdot\ddot{x} - x(1-\dot{x}^2) = 0 \tag{A.24}$$

Whether or not a solution exists for this equation which satisfies the appropriate boundary conditions is difficult to ascertain. However, it can be seen by inspection of Eq. A.23 that $J = 0$ if and only if

$$x(t) = \begin{cases} 0 & \text{for } -1 \leq t \leq 0 \\ t & \text{for } 0 \leq t \leq 1 \end{cases} \tag{A.25}$$

One can verify that $J = 0$ for this function by substituting Eq. A.25 into A.23. In addition, from Eq. A.23 one sees that $J = 0$ only if at each point on the interval $-1 < t < +1$ either $x(t) = 0$ or $\dot{x}(t) = +1$. It is easy to see that Eq. A.25 is the only such function which satisfies the specified boundary conditions. But this $x(t)$ does not have a derivative at $t = 0$ and thus is not an admissible function. It satisfies Eq. A.24 on the interiors of each of the time intervals, but not at $t = 0$, since $\dot{x}$ does not exist at that point. Thus, any solution of the Euler-Lagrange equation satisfying the specified boundary conditions can at best yield a "local" minimum of J, but it can not yield the absolute minimum given

by Eq. A.25. With this example as motivation, the Weierstrass-Erdmann corner conditions are now reviewed.

Let X' be the set of continuous functions mapping (t_0, t_f) into R^1 which satisfy the boundary conditions given by Eq. A.2, and whose derivatives $\dot{x}$ are continuous except at one point, $t = t_1$ where $t_0 < t_1 < t_f$, t_1 is unspecified, and $\dot{x}$ fails to exist. Since $\dot{x}$ does not exist at t_1, Eq. A.1 cannot be expanded in a Taylor series expansion at that point. To proceed, the functional J must first be written as

$$J = \int_{t_0}^{t_f} F(x, \dot{x}, t)\, dt = \int_{t_0}^{t_1^-} F(x, \dot{x}, t)\, dt + \int_{t_1^+}^{t_f} F(x, \dot{x}, t)\, dt \qquad (A.26)$$

Now since both x and $\dot{x}$ are continuous over each subinterval, F can be expanded under each integral of Eq. A.26 in a Taylor series about $\bar{x}(t)$. Assuming for simplicity that $t_0, t_f, x(t_0)$ and $x(t_f)$ are specified, by applying the equation of the general first variation, Eq. A.15, to both terms of Eq. A.26, one obtains the first variation of J, given by

$$\delta J = \int_{t_0}^{t_1^-} [\bar{F}_x(t) - \frac{d}{dt} \bar{F}_{\dot{x}}(t)] \cdot \delta x(t)\, dt + \int_{t_1^+}^{t_f} [\bar{F}_x(t) - \frac{d}{dt} \bar{F}_{\dot{x}}(t)] \cdot \delta x(t)\, dt$$

$$- [\bar{F}_{\dot{x}}(t) \cdot dx(t)] \Big|_{t_1^-}^{t_1^+} - [\bar{F}(t) - \dot{\bar{x}}(t) \bar{F}_{\dot{x}}(t) \cdot dt_1] \Big|_{t_1^-}^{t_1^+} = 0 \qquad (A.27)$$

Considering first those variations for which dt_1 and $dx(t_1)$ are zero, one deduces that it is necessary that the Euler-Lagrange equation, Eq. A.12, must be satisfied over each of the subintervals (t_0, t_1^-) and (t_1^+, t_f). The first variation then becomes

$$\delta J = - [\bar{F}_x(t) \cdot dx(t)] \Big|_{t_1^-}^{t_1^+} - [(\bar{F}(t) - \dot{\bar{x}}(t) \cdot \bar{F}_{\dot{x}}(t)) \cdot dt_1] \Big|_{t_1^-}^{t_1^+} = 0 \qquad (A.28)$$

Since both t_1 and $x(t_1)$ are unspecified, considering their variations separately, one finds that it is necessary that

$$F_{\dot{x}}[\bar{x}(t_1), \dot{\bar{x}}(t_1^+), t_1] = F_{\dot{x}}[\bar{x}(t_1), \dot{\bar{x}}(t_1^-), t_1] \tag{A.29}$$

and

$$F[\bar{x}(t_1), \dot{\bar{x}}(t_1^+), t_1] - \dot{\bar{x}}(t_1^+) F_{\dot{x}}[\bar{x}(t_1), \dot{\bar{x}}(t_1^+), t_1] = F[\bar{x}(t_1), \dot{\bar{x}}(t_1^-), t_1]$$

$$- \dot{\bar{x}}(t_1^-) F_{\dot{x}}[\bar{x}(t_1), \dot{\bar{x}}(t_1^-), t_1] \tag{A.30}$$

These are the Weierstrass-Erdmann corner conditions which must hold at the point t_1 at which $\dot{\bar{x}}(t)$ does not exist. The results can, of course, be extended to cases where more than one corner exists, and Eqs. A.29 and A.30 must be satisfied at each such point. Making use of Eq. A.29, Eq. A.30 can be written

$$\bar{F}(t_1^+) - \bar{F}(t_1^-) = \bar{F}_{\dot{x}}(t_1)[\dot{\bar{x}}(t_1^+) - \dot{\bar{x}}(t_1^-)] \tag{A.31}$$

which says that the change in $\bar{F}$ across the corner is proportional to the change in $\dot{\bar{x}}$, with the constant of proportionality being $\bar{F}_{\dot{x}}$ at the corner (which cannot change across the corner), i.e.,

$$\bar{F}_{\dot{x}}(t_1) = \frac{\Delta \bar{F}(t_1)}{\Delta \dot{\bar{x}}(t_1)} \tag{A.32}$$

Returning to the example presented by Eq. A.23, three solutions which satisfy the Euler-Lagrange equation over two subintervals are shown in Fig. A.2 by the curves x_0, x_1, and x_2. The Weierstrass-Erdmann corner conditions for this problem are given by

$$\Delta \bar{F}_{\dot{x}}(t_1) = -2\bar{x}^2(t_1) \cdot [1 - \dot{\bar{x}}(t_1^+)] + 2\bar{x}^2(t_1)[1 - \dot{\bar{x}}(t_1^-)] = 0 \tag{A.33}$$

and

$$\Delta\{\bar{F} - \dot{\bar{x}}\bar{F}_{\dot{x}}\}(t_1) = \bar{x}^2(t_1) \cdot [1 - \dot{\bar{x}}^2(t_1^+)] - \bar{x}^2(t_1)[1 - \dot{\bar{x}}^2(t_1^-)] = 0 \tag{A.34}$$

Equation A.34 requires that at a corner either $\bar{x}(t_1) = 0$ or $\dot{\bar{x}}(t_1^+) = \pm \dot{\bar{x}}(t_1^-)$, which is satisfied by all three of the solutions in Fig. A.2. However, Eq. A.33 further restricts the minimal solution to have corners only where $x(t_1) = 0$ or $\dot{\bar{x}}(t_1^+) = \dot{\bar{x}}(t_1^-)$, the second condition being a trivial case since then there is no corner. Only the solution $x_0(t)$ in Fig. A.2,

which is the minimal solution given in Eq. A.25, satisfies both Weierstrass-Erdmann corner conditions, and therefore $x_1(t)$ and $x_2(t)$ cannot be local extremals.

The Weierstrass-Erdmann corner conditions are generalized in Chapter II to optimal control problems in which the variations in $x(t_1)$ and t_1 are not free, leading to a set of intermediate transversality conditions which are analogous to the terminal transversality conditions developed in Section A.7.

A.5 EXTENSION TO n-DIMENSIONS

The extension of the simplest problem to the minimization of a scalar functional of an n-dimensional vector function of time is quite straightforward. Let $x(t) = (x_1(t), x_2(t), \ldots, x_n(t))$, let $t_0, t_f, x(t_0), x(t_f)$ be given, and let

$$J = \int_{t_0}^{t_f} F[x(t), \dot{x}(t), t]\, dt = \int_{t_0}^{t_f} F[x_1(t), \dot{x}_1(t), \ldots, x_n(t), \dot{x}_n(t), t]\, dt \tag{A.35}$$

where F has continuous first and second partial derivatives with respect to its (2n+1) arguments. F can be expanded in a Taylor series about $\bar{x}(t)$ to give a first variation of

$$\delta J = \int_{t_0}^{t_f} \left\{ \sum_{i=1}^{n} \left[\bar{F}_{x_i}(t) - \frac{d}{dt} \bar{F}_{\dot{x}_i}(t) \right] \delta x_i(t) \right\} dt + \left[\sum_{i=1}^{n} \bar{F}_{\dot{x}_i}(t) \cdot \delta x_i(t) \right]_{t_0}^{t_f} = 0 \tag{A.36}$$

Considering the $\delta x_i(t)$ independently, one determines that $\bar{x}(t)$ must satisfy the n Euler-Lagrange equations

$$F_{x_i}[\bar{x}(t), \dot{\bar{x}}(t), t] - \frac{d}{dt} F_{\dot{x}_i}[\bar{x}(t), \dot{\bar{x}}(t), t] = 0, \quad i = 1, \ldots, n \tag{A.37}$$

and the 2n natural boundary conditions

$$F_{\dot{x}_i}[\bar{x}(t_0), \dot{\bar{x}}(t_0), t_0] = 0 \quad \text{if } x_i(t_0) \text{ is free} \quad i = 1, \ldots, n \tag{A.38}$$

$$F_{\dot{x}_i}[\bar{x}(t_f), \dot{\bar{x}}(t_f), t_f] = 0 \quad \text{if } x_i(t_f) \text{ is free} \quad i = 1, \ldots, n \tag{A.39}$$

A.6 DIFFERENTIAL CONSTRAINTS

Consider the minimization problem of Section A.2 with the added constraint that

$$g[\bar{x}(t), \dot{\bar{x}}(t), t] = 0 \tag{A.40}$$

where the first and second partial derivatives of g with respect to x, $\dot{x}$, and t are assumed to be continuous. Along a neighboring path the differential constraint must also be satisfied

$$g[x(t), \dot{x}(t), t] = 0 \tag{A.41}$$

since one now compares only functions which satisfy the constraint. But Eq. A.41 can be written as

$$g[\bar{x}(t) + \delta x(t), \dot{\bar{x}}(t) + \delta \dot{x}(t), t] = 0 \tag{A.42}$$

and thus one observes that the variations $\delta x(t)$ are not free but are governed by Eq. A.42. It can be proven[3] that, as long as $g_{\dot{x}}$ does not become zero along $\bar{x}(t)$, there exists a function $\lambda(t)$ such that the minimization of Eq. A.1, subject to the constraint Eq. A.35, is equivalent to the minimization of

$$J' = \int_{t_0}^{t_f} \Phi[x(t), \dot{x}(t), \lambda(t), t]\, dt \tag{A.43}$$

with respect to $x(t)$ and $\lambda(t)$, with no differential constraint, where Φ is given by

$$\Phi[x(t), \dot{x}(t), \lambda(t), x] = F[x(t), \dot{x}(t), t] + \lambda \cdot g[x(t), \dot{x}(t), t] \tag{A.44}$$

Since the minimization is now with respect to two functions, $x(t)$ and $\lambda(t)$, two Euler-Lagrange equations must be satisfied by the optimal $\bar{x}(t)$ and $\bar{\lambda}(t)$ (see Section A.5):

$$\Phi_x[\bar{x}, \dot{\bar{x}}, \bar{\lambda}, t] - \frac{d}{dt} \Phi_{\dot{x}}[\bar{x}, \dot{\bar{x}}, \bar{\lambda}, t] = 0 \tag{A.45}$$

and

$$\Phi_\lambda[\bar{x}, \dot{\bar{x}}, \bar{\lambda}, t] - \frac{d}{dt} \Phi_{\dot{\lambda}}[\bar{x}, \dot{\bar{x}}, \bar{\lambda}, t] = 0 \tag{A.46}$$

Substituting Eq. A.44 into Eqs. A.45 and A.46, the Euler-Lagrange equations become

$$\bar{F}_x(t) + \bar{\lambda}(t) \cdot \bar{g}_x(t) - \frac{d}{dt}[\bar{F}_{\dot{x}}(t) + \lambda(t) \cdot \bar{g}_{\dot{x}}(t)] = 0 \qquad (A.47)$$

$$g[\bar{x}, \dot{\bar{x}}, t] = 0 \qquad (A.48)$$

Since Φ_χ is zero, as noted by Eq. A.48, no additional natural boundary conditions are forthcoming.

If x is an n-vector and g is an m-vector function of x and $\dot{x}$, then λ will be an m-vector, and one then has (n+m) Euler-Lagrange equations and 2n boundary conditions. Of course, it is necessary that $m \leq n$ in order that the problem not be over-specified.

A.7 THE OPTIMAL CONTROL PROBLEM

In this section the results obtained thus far for the calculus of variations are applied to the general optimal control problem. Consider a system governed by a set of n first-order ordinary differential equations

$$\dot{x}_i = f_i(x, u, t) \quad , \qquad i = 1, \ldots, n \qquad (A.49)$$

where $x = (x_1, \ldots, x_n)$ is the n-vector which specifies the state of the system and $u = (u_1, \ldots, u_m)$ is the m-vector of control variables. One wishes to transfer the system from a specified set of initial conditions to a desired set of terminal conditions while minimizing a functional of the form

$$J = \phi^0[x(t_0), t_0] + \phi^f[x(t_f), t_f] + \int_{t_0}^{t_f} L[x(t), u(t), t]\, dt \qquad (A.50)$$

The initial and terminal conditions on the state are in the form of sets of equality constraints given by

$$\psi^0[\,x(t_0),t_0] = \begin{bmatrix} \psi_1^0[\,x(t_0),t_0] \\ \vdots \\ \psi_{k_0}^0[\,x(t_0),t_0] \end{bmatrix} = 0 \quad , \quad 0 \le k_0 \le n+1 \tag{A.51}$$

and

$$\psi^f[\,x(t_f),t_f] = \begin{bmatrix} \psi_1^f[\,x(t_f),t_f] \\ \vdots \\ \psi_{k_f}^f[\,x(t_f),t_f] \end{bmatrix} = 0 \quad , \quad 0 \le k_f \le n+1 \tag{A.52}$$

One again requires the continuity of the first and second partial derivatives of f, L, ϕ^0, ϕ^f, ψ^0, ψ^f, x, and u with respect to each of their arguments. In this case one has the differential constraints

$$g[\,x,\dot{x},u,t] = f(x,u,t) - \dot{x} = 0 \tag{A.53}$$

and since $g_{\dot{x}} = -1 \neq 0$ the results of Section A.6 are directly applicable. Thus one minimizes

$$J' = \phi^0[\,x(t_0),t_0] + \phi^f[\,x(t_f),t_f] + \int_{t_0}^{t_f} \Phi[\,x,\dot{x},u,\lambda,t]\,dt \tag{A.54}$$

with respect to x, u, and λ with no differential constraints, where

$$\Phi[\,x,\dot{x},u,\lambda,t] = L[\,x,u,t] + \lambda^T \cdot f(x,u,t) - \lambda^T \cdot \dot{x} \tag{A.55}$$

For convenience, let us define a function called the "Hamiltonian" by

$$H[\,x,u,\lambda,t] = L(x,u,t) + \lambda^T \cdot f(x,u,t) \tag{A.56}$$

The Euler-Lagrange equations, Eq. A.45 and A.44, can be written as

$$\Phi_x(t) - \frac{d}{dt}\Phi_{\dot{x}}(t) = H_x^T[\,\bar{x}(t),\bar{u}(t),\bar{\lambda}(t),t] + \frac{d\bar{\lambda}}{dt}(t) = 0 \tag{A.57}$$

$$\Phi_u(t) - \frac{d}{dt}\Phi_{\dot{u}}(t) = H_u[\,\bar{x}(t),\bar{u}(t),\bar{\lambda}(t),t] = 0 \tag{A.58}$$

$$\Phi_{\lambda}(t) - \frac{d}{dt}\Phi_{\chi}(t) = f[\bar{x}(t),\bar{u}(t),t] - \frac{d\bar{x}}{dt}(t) = 0 \tag{A.59}$$

Adding the first variations of ϕ^0 and ϕ^f into the first variation of J, one finds that the natural boundary conditions are determined from

$$\left\{\frac{\partial\phi^0}{\partial x}[\bar{x}(\bar{t}_0),\bar{t}_0] + \bar{\lambda}^T(\bar{t}_0)\right\}\cdot dx(\bar{t}_0) = 0 \tag{A.60}$$

$$\left\{\frac{\partial\phi^f}{\partial x}[\bar{x}(\bar{t}_f),\bar{t}_f] - \bar{\lambda}^T(\bar{t}_f)\right\}\cdot dx(\bar{t}_f) = 0 \tag{A.61}$$

$$\left\{\frac{\partial\phi^0}{\partial t}[\bar{x}(\bar{t}_0),\bar{t}_0] - H[\bar{x}(\bar{t}_0),\bar{u}(\bar{t}_0),\bar{\lambda}(\bar{t}_0),\bar{t}_0]\right\}\cdot dt_0 = 0 \tag{A.62}$$

$$\left\{\frac{\partial\phi^f}{\partial t}[\bar{x}(\bar{t}_f),\bar{t}_f] + H[\bar{x}(\bar{t}_f),\bar{u}(\bar{t}_f),\bar{\lambda}(\bar{t}_f),\bar{t}_f]\right\}\cdot dt_f = 0 \tag{A.63}$$

The initial and terminal state constraints may allow some freedom of choice of $dx(t_i)$ and dt_i, $i = 0,f$, but in general, these variations are not independent. The constraint equations given by Eqs. A.51 and A.52 define hypersurfaces in the (n+1)-dimensional state-time space. As a result of the continuity and differentiability restrictions made on ψ^0 and ψ^f these hypersurfaces have a unique tangent plane at each point on their surfaces, and the first order variations dx and dt must lie on these tangent planes, i.e., at each end point dx and dt must satisfy

$$d\psi_j^0 = \left[\sum_{i=1}^{n}\frac{\partial\psi_j^0}{\partial x_i}(\bar{x}(t),t)\cdot dx_i + \frac{\partial\psi_j^0}{\partial t}(\bar{x}(t),t)\cdot dt\right]_{t=\bar{t}_0} = 0,\ j=1,\ldots,k_0 \tag{A.64}$$

$$d\psi_j^f = \left[\sum_{i=1}^{n}\frac{\partial\psi_j^f}{\partial x_i}(\bar{x}(t),t)\cdot dx_i + \frac{\partial\psi_j^f}{\partial t}(\bar{x}(t),t)\cdot dt\right]_{t=\bar{t}_f} = 0,\ j=1,\ldots,k_f \tag{A.65}$$

Using the gradient notation in the (n+1)-dimensional state-time space,

$$\nabla\psi_j = \text{col}\left[\frac{\partial\psi_j}{\partial x_1}, \ldots, \frac{\partial\psi_j}{\partial x_n}, \frac{\partial\psi_j}{\partial t}\right] \tag{A.66}$$

and defining

$$\underline{\lambda}(t) = \text{col}\left(\lambda_1(t), \ldots, \lambda_n(t), -H(t)\right) \tag{A.67}$$

Equations A.60 through A.65 can be combined to obtain the initial and terminal boundary conditions

$$\nabla\phi^0\left[\bar{x}(\bar{t}_0), \bar{t}_0\right] + \underline{\bar{\lambda}}(\bar{t}_0) + \sum_{j=1}^{k_0} \nu_j \cdot \nabla\psi_j^0\left[\bar{x}(\bar{t}_0), \bar{t}_0\right] = 0 \tag{A.68}$$

and

$$\nabla\phi^f\left[\bar{x}(\bar{t}_f), \bar{t}_f\right] - \underline{\bar{\lambda}}(\bar{t}_f) + \sum_{j=1}^{k_f} \mu_j \cdot \nabla\psi_j^f\left[\bar{x}(\bar{t}_f), \bar{t}_f\right] = 0 \tag{A.69}$$

where the ν_j and μ_j are unspecified multiplier functions. Equations A.68 and A.69 are often called the transversality conditions, since in the absence of the penalty functions ϕ^0 and ϕ^f, the multiplier functions $\lambda(t)$ are normal (transversal) to the initial and terminal state constraint surfaces. Equations A.51, A.52, A.68, and A.69 constitute a set of (2n+2) boundary conditions, enough to specify a solution to the 2n differential equations given in Eqs. A.57 and A.59. However, since (n+1) of the boundary conditions are imposed at t_0 and (n+1) of them at t_f, the set of necessary conditions form a two-point boundary value problem for which, in general, straightforward methods of solution are not available. One must often resort to iterative numerical algorithms, such as those discussed in Appendices B and C, to obtain solutions.

Thus, for $\bar{u}(t)$ to be the minimal control function it is necessary that a set of multiplier functions $\bar{\lambda}(t)$ exist such that Eqs. A.57 through A.59, A.51, A.52, A.68, and A.69 be satisfied. It should be stressed that these are actually necessary conditions for stationary points, i.e., they may be satisfied by relative minima, relative maxima, and saddle points. In addition, there may exist a $u(t)$ not in the admissible class, which yields a smaller value of the cost functional while satisfying the differential constraints and boundary conditions. The class of

admissible controls can easily be extended to include discontinuities in $u(t)$ or $\dot{u}(t)$ by direct application of the Weierstrass-Erdmann corner conditions developed in Section A.4. Substituting Eq. A.55 into Eqs. A.29 and A.30, one finds that the Weierstrass-Erdmann corner conditions require that at a point of discontinuity in $\bar{u}(t)$ or $\dot{\bar{u}}(t)$ the Hamiltonian function, $H[\bar{x}(t), \bar{u}(t), \overline{\lambda}(t), t]$, and the n-vector of Lagrange multipliers, $\overline{\lambda}(t)$, must be continuous. In the next section the class of admissible control functions will be further extended to allow the specification of bounds on the controls through the use of the well-known Maximum Principle of Pontryagin. The remainder of this section considers conditions on second derivatives and on H.

Since the function $\Phi[x, \dot{x}, u, \lambda, t]$ in Eq. A.54, as defined by Eq. A.55 is linear in $\dot{x}$, then

$$\Phi_{\dot{x}\dot{x}} = 0 \tag{A.70}$$

for all $x(t)$, $u(t)$, and $\lambda(t)$, and thus the Legendre and Weierstrass necessary conditions, in the form discussed in Section A.2, are of no value in determining the solution to the optimal control problem. The equalities in Eq. A.16, the Legendre condition, and Eq. A.19, the Weierstrass condition, are always satisfied. However, by heuristically extending the arguments of Section A.2 to the optimal control problem, one obtains an analogous set of conditions that

$$\frac{\partial^2 H}{\partial u^2}[\bar{x}(t), \bar{u}(t), \overline{\lambda}(t), t] \geq 0 \tag{A.71}$$

and

$$H[\bar{x}(t), \bar{u}(t), \overline{\lambda}(t), t] = \min_{U} H[\bar{x}(t), U, \overline{\lambda}(t), t] \tag{A.72}$$

for all $t_0 < t < t_f$. As a result of Eq. A.70, one can write the second variation of J in terms of products of $\delta x(t)$, $\delta u(t)$, and $\delta\lambda(t)$ only. Since $\delta x(t)$ and $\delta\lambda(t)$ must satisfy the differential equations obtained by linearizing Eqs. A.57 and A.59, one can argue that if $\delta u(t)$ is "small", then $\delta x(t)$ and $\delta\lambda(t)$ will necessarily be small, but $\delta x(t)$ and $\delta\lambda(t)$ may also be small for a "large" $\delta u(t)$. Thus, the term in the second variation involving $\delta u^2(t)$ is dominant, and the inequality of

Eq. A.71, analogous to the Legendre condition, results. The Weierstrass condition is obtained by considering strong variations of $\dot{x}(t)$, i.e., by considering all possible $\dot{x}(t)$ at any point on the optimum curve, $\bar{x}(t)$. In the optimal control problem, from Eq. A.59, a discontinuity in $\dot{x}(t)$ is dependent on a discontinuity in $u(t)$, as a result of the continuity assumptions on $x(t)$ and $f(x,u,t)$. Writing the Weierstrass condition directly in terms of strong variations of $u(t)$ one obtains

$$\Phi[\bar{x},\dot{\bar{x}},U,\lambda,t] - \Phi[\bar{x},\dot{\bar{x}},\bar{u},\bar{\lambda},t] - (U-u)\cdot\Phi_u[\bar{x},\dot{\bar{x}},\bar{u},\bar{\lambda},t] \geq 0 \tag{A.73}$$

In view of Eq. A.58, which also must be satisfied by the optimal solution, Eq. A.73 reduces to Eq. A.72. Thus, at each point along the optimal solution $\bar{u}(t)$ produces not only a stationary point of the Hamiltonian, Eq. A.56, but also its absolute minimum.

A.8 THE MINIMUM PRINCIPLE

The development of the necessary conditions on the optimal control in the previous section required that the set of admissible controls be unbounded (actually only that the set be open) and that H_u exist and be continuous so that the appropriate Taylor series expansions could be made. However, in many engineering problems the control variables have definite bounds set on them, as for example, maximum rudder deflection, maximum engine thrust, etc. Such problems cannot be simply handled by the calculus of variations. Consider the ordinary function minimization problem shown in Fig. A.3. Analogous to requiring that the first variation of J be zero in the calculus of variations, one requires that $\frac{dg}{dy}$ be zero at the minimum point of $g(y)$. This condition is satisfied at points y_1, y_2, y_4, y_5, and $y_7 - -y_1$ and y_7 being relative minima, y_2 and y_4 being relative maxima, and y_5 being a saddle point. By direct comparison one finds the absolute minimum to occur at y_7. However, if the independent variable y were restricted to either $y_0 \leq y \leq y_4$ or $y_0 \leq y \leq y_6$, the absolute minimum could not be found by this method. For the first range the absolute minimum occurs at y_3, a point where $\frac{dg}{dy}$ does not exist, and for the second it occurs at y_6, on the boundary of the range of y. Similarly, the

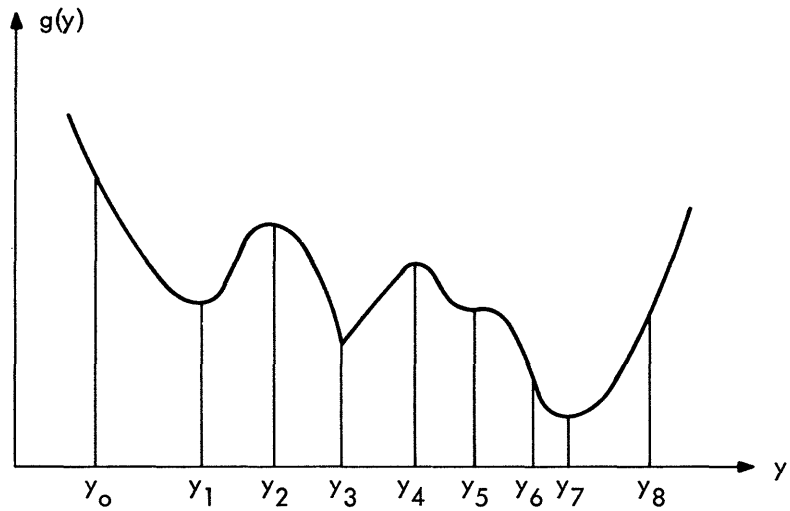


Fig. A.3 Minimization of Arbitrary g(y), Section A.8

classical variational approach is not valid when bounds are placed on $u(t)$ or when H_u may have points of nonexistence (as in the formulation of certain minimum fuel problems). To cope with these circumstances the more powerful Minimum Principle has evolved (often called Maximum Principle as a result of the work of Pontryagin[6] and others). Because of the complexity of its proof, only the statement of the theorem will be given here. Various references[5,6] are available which present either detailed or heuristic proofs of the theorem.

Let $u = (u_1, \ldots, u_m)$ be the m-vector of control variables for a system governed by Eq. A.49, let Ω be a closed subset of R^m and U be the set of all piecewise continuous functions taking values in Ω. Such functions, $u(t) \in U$ are called admissible controls. One wishes to determine the control function $\bar{u}(t) \in U$ which minimizes a cost functional of the form given by Eq. A.50, subject to the differential constraints given by Eq. A.49 and the initial and terminal boundary conditions given by Eqs. A.51 and A.52. The functions $f(x,u,t)$, $L(x,u,t)$, $\phi^i(x,t)$, and $\psi^i(x,t)$ are assumed to be continuous with respect to $x, u,$ and t and continuously differentiable with respect to x and t for all x, $u \in \Omega$, and $t_0 < t < t_f$. Let $\bar{u}(t)$ be an element of U and $\bar{x}(t)$ the corresponding state. In order that $\bar{u}(t)$ minimize J it is necessary that there exist an n-vector function $\bar{\lambda}(t)$ such that Eqs. A.57, A.59, A.51, A.52, A.68, and A.69 are satisfied and

$$H(\bar{x},\bar{u},\bar{\lambda},t) = \min_{u \in \Omega} H(\bar{x},u,\bar{\lambda},t) \qquad (A.74)$$

for all $t_0 < t < t_f$.

Thus, the Minimum Principle reduces the problem of finding an entire function $u(t)$ that absolutely minimizes J to that of finding the absolute minimum of the function H with respect to u at each point in time. As noted in the above discussion, the minimum may occur on the boundary Ω, at any points where H_u does not exist, or where Eq. A.58 is satisfied.

APPENDIX B

THE METHOD OF STEEPEST-DESCENT

B.1 INTRODUCTION--ORDINARY MINIMIZATION PROBLEMS

The steepest-descent algorithm for solving optimal control problems is reviewed in this appendix. The method is a direct extension of the gradient method for finding the extremals, i.e., the maxima or minima, of a function of several variables to functional problems. In this section ordinary minimization problems will be discussed in general terms to establish the basic concepts of the steepest-descent method, and in Section B.2 the algorithm for two-point functional minimization problems will be developed in detail.

The various methods for solving minimization problems are generally classed as being either direct or indirect. The direct methods are those which search directly for the smallest value of the function being minimized, while the indirect methods search instead for a solution which satisfies the first order necessary condition for a minimum as determined from the calculus. To demonstrate these ideas consider the function

$$J(x) = \frac{1}{2}x^T Ax + b^T x + c \tag{B.1}$$

where x and b are n-vectors and A is an $(n \times n)$ matrix and c is a scalar. For this function to have a minimum it is necessary that A be a "positive" matrix; and for the minimum to be unique, it is necessary that A be "positive-definite." A positive-definite matrix is one for which the function

$$G(x) = x^T Ax \tag{B.2}$$

satisfies the two conditions

$$G(x) \geq 0 \tag{B.3}$$

and

$$G(x) = 0 \text{ if and only if } x = 0 \tag{B.4}$$

If only condition B.3 is satisfied then the matrix A is merely positive.

From the calculus, for $\bar{x}$ to yield a minimum of $J(x)$ it is necessary that

$$\frac{dJ}{dx}(\bar{x}) = 0 \tag{B.5}$$

For the example considered this condition becomes

$$A\bar{x} + b = 0 \tag{B.6}$$

If A is positive definite its inverse exists, and one can solve for the unique optimum

$$\bar{x} = -A^{-1}b \tag{B.7}$$

In general, one will not be able to solve Eq. B.5 analytically, and an iterative numerical algorithm must be employed to determine the solution. One such indirect method, the Newton algorithm is discussed in detail in Appendix C.

The most straightforward, and in many cases the most inefficient, of the direct methods, the direct search, evaluates $J(x)$ at a discrete set of x_i^* and by direct comparison determines the x_j which yields the smallest value of J. While this method does not determine the optimal value of x, by making the spacing between the x_i small enough the desired precision can be attained. The steepest-descent method attempts to make this search more efficient by selecting x_{i+1} not in a random fashion, but by making use of the gradient of J at x_i. Since the gradient of J defines the derivative of J with respect to x in the direction in which this derivative is the greatest, one expects to achieve the greatest decrease in J by making the next guess in the negative gradient direction, i.e.,

$$x_{i+1} = x_i - K_i \frac{dJ}{dx}(x_i) \tag{B.8}$$

---

* The subscripts here denote the iteration index.

One must specify the gain K_i at each step, which determines how large a step is to be taken in that direction. If K_i is too small, the convergence to the solution is slow. On the other hand, if K_i is excessive, the higher order derivatives of J will become significant and the linearized analysis on which the above arguments is based is no longer valid.

An ordinary minimization problem may be complicated by the existence of equality constraints on the components of the vector x of the form

$$g(x) = 0 \tag{B.9}$$

where g is an m-vector of functions of x. For example, Fig. B.1 gives a set of contours of constant J as a function of two variables and a curve which corresponds to the solution of Eq. B.9. The problem is to find the value of x along the curve $g(x) = 0$ which yields a minimum of J, as indicated by $\bar{x}$. In general, the first guess ζ_0 will satisfy neither Eq. B.9 nor will it minimize J. If one makes the next guess along the negative gradient, as indicated by the point ζ_1', the value of J will be decreased, but the violation of the constraint equation may increase. Clearly some control must be exerted over the direction of δx to insure that one eventually converges to a solution which satisfies the constraint. This can be done by separating the negative gradient vector into two orthogonal directions, d_1 and d_2, as indicated in Fig. B.1. The direction d_1 is tangent to the curve

$$g(x) = \gamma \tag{B.10}$$

which passes through ζ_0, and d_2 is perpendicular to it. By moving in the direction d_1, one can reduce J while making no change in the violation of the constraints (within first order terms). Independent control can be exerted over the constraint violation by moving in the d_2 direction not to decrease J but to decrease the amount by which Eq. B.10 is violated, as noted in Fig. B.1 by the point ζ_1. Since one moves not in the negative gradient direction, but in a direction of greatest decrease of J subject to a specified decrease in the constraint violation, this method is more appropriately called "steepest-descent" rather than gradient.

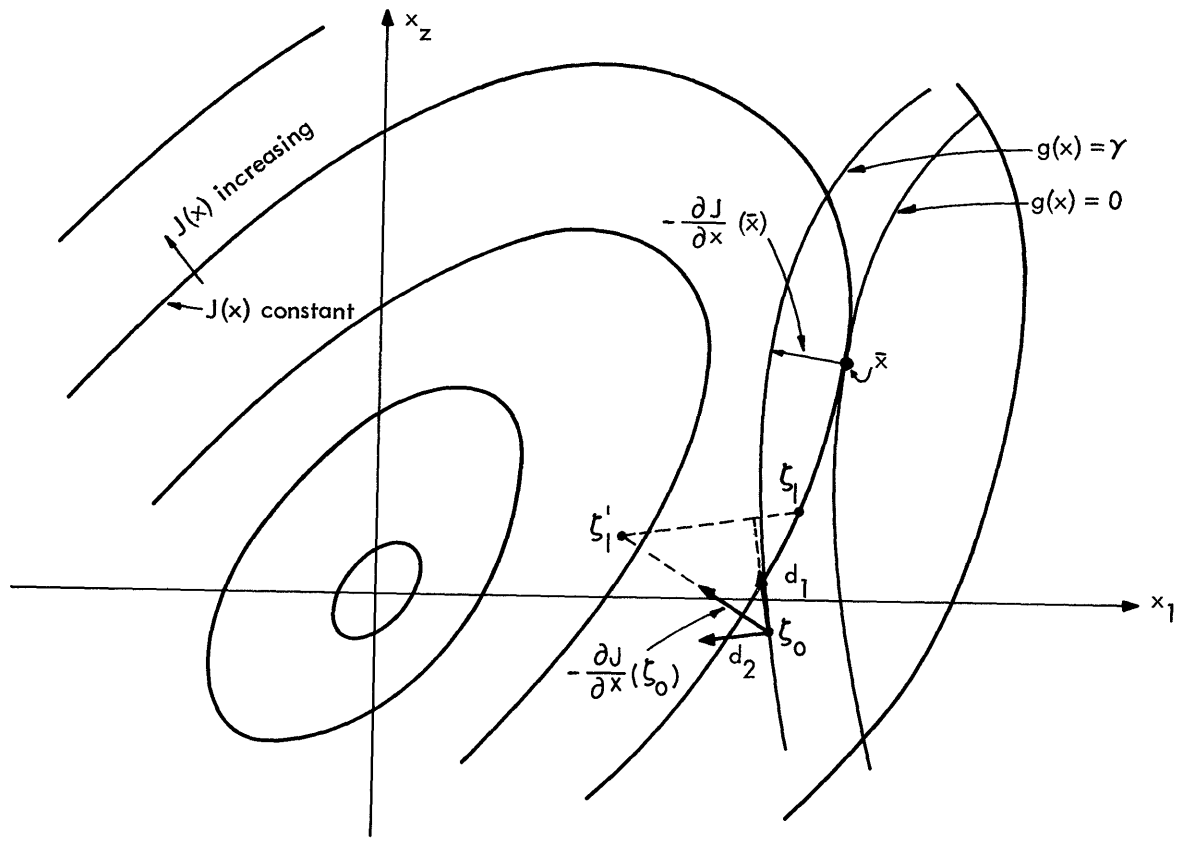


Fig. B.1 Minimization of J(x) with Constraint g(x) = 0 by Steepest Descent

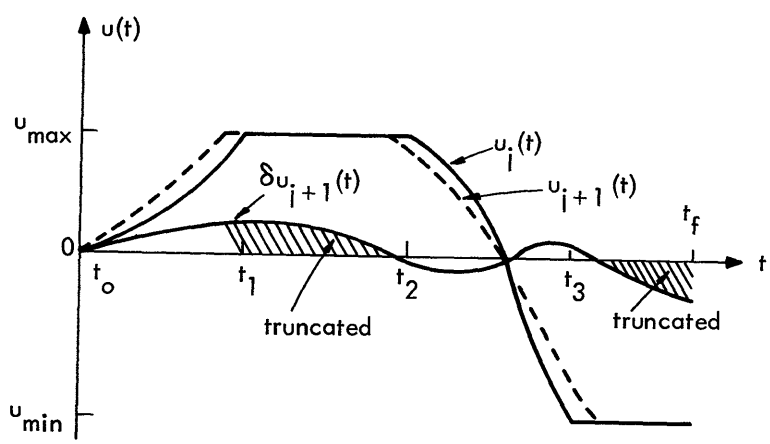


Fig. B.2 Control Perturbation, Bounded Control

The discussion up to this point has been quite heuristic and in terms of function minimization problems in order to give insight into the concepts involved in the steepest-descent method. Since many of these concepts are difficult to visualize in function space, it is convenient to refer back to the analogous geometrical representations in vector space.

B.2 STEEPEST-DESCENT IN FUNCTION SPACE

In this section the steepest-descent iterative technique for the solution of optimal control problems is reviewed.[21,23-27] One is interested in controlling a system governed by the differential equations

$$\dot{x} = f(x, u, t) \tag{B.11}$$

such that the system is taken from an initial state

$$x(t_0) = \zeta \tag{B.12}$$

to a terminal state which satisfies the equations

$$\psi[x(t), t] = 0 \tag{B.13}$$

and minimizes a functional of the form

$$J[x(t), u(t)] = \phi[x(t_1), t_1] + \int_{t_0}^{t_1} L(x, u, t)dt \tag{B.14}$$

The n-vector x is called the state vector, u is the m-vector control, and ψ is a (k+1)-vector function of x and t. At each iteration we have a nominal control function $u_j(t)$ (chosen arbitrarily for the first iteration) with which Eq. B.11 is integrated forward in time to a nominal terminal time $t_{1,j}$ using the initial conditions of Eq. B.12. If the terminal time t_1 is specified explicitly in Eq. B.13 by

$$t - t_1 = 0 \tag{B.15}$$

then the nominal terminal time $t_{1,j}$ is set equal to t_1.

However, if it is not specified then one of the constraint equations must be used as a stopping condition. Thus the integration will continue until*

$$\theta[x(t),t] = \psi_i[x(t),t] = 0 \qquad \text{(B.16)}$$

The nominal $t_{1,j}$ is the time at which this constraint is satisfied. The remaining constraints will not, in general, be satisfied at this time. The constraint vector is now considered as a k-vector function of x and t with the stopping condition deleted.

The objective is to determine a perturbation $\delta u(t)$ to apply to $u_j(t)$ such that the maximum possible decrease in J is obtained, while decreasing the terminal state constraint violations $\psi[x(t_{1,j}),t_{1,j}]$ by a specified amount. To determine the effect of a control perturbation on Eqs. B.11, B.13, B.14, and B.16, they are expanded in Taylor series' about the nominal solution, to first order terms, giving

$$\frac{d}{dt}\delta x(t) = F(t)\cdot\delta x(t) + G(t)\cdot\delta u(t) \qquad \text{(B.17)}$$

$$d\psi = \psi_x(t_{1,j})\cdot\delta x(t_{1,j}) + [\psi_x\cdot f + \psi_t]_{t=t_{1,j}}\cdot dt_1 \qquad \text{(B.18)}$$

$$dJ = \phi_x(t_{1,j})\cdot\delta x(t_{1,j}) + [\phi_x\cdot f + \phi_t + L]_{t=t_{1,j}}\cdot dt_1 + \int_{t_0}^{t_{1,j}} [L_x(t)\cdot\delta x(t) + L_u(t)\cdot\delta u(t)]\,dt \qquad \text{(B.19)}$$

and

$$d\theta = \theta_x(t_{1,j})\cdot\delta x(t_{1,j}) + [\theta_x\cdot f + \theta_t]_{t=t_{1,j}}\cdot dt_1 \qquad \text{(B.20)}$$

---

* In this case the subscript denotes the i-th component of Eq. B.13.

where

$$F(t) = f_x[x_j(t), u_j(t), t] \; ; \; G(t) = f_u[x_j(t), u_j(t), t] \tag{B.21}$$

and the subscripts x, u, and t indicate partial derivatives with respect to these variables which are evaluated on the nominal solution $u_j(t)$, $x_j(t)$, and $t_{1,j}$.

Since at each iteration Eq. B.16 is used to define a new nominal terminal time t_{1j}, $d\theta$ is zero and Eq. B.20 can be used to estimate the change in the nominal terminal time, i.e.,

$$dt_1 = -[(\theta_x \cdot f + \theta_t)^{-1} \cdot \theta_x]_{t=t_{1,j}} \delta x(t_{1,j}) \tag{B.22}$$

Substituting Eq. B.22 into Eqs. B.18 and B.19, one can eliminate dt_1, i.e.,

$$d\psi = [\psi_x - (\psi_x \cdot f + \psi_t)(\theta_x \cdot f + \theta_t)^{-1}\theta_x]_{t=t_{1,j}} \cdot \delta x(t_{1,j}) \tag{B.23}$$

and

$$dJ = [\phi_x - (\phi_x \cdot f + \phi_t + L)(\theta_x f + \theta_t)^{-1}\theta_x]_{t,j} \cdot \delta x(t_{1,j}) + \int_{t_0}^{t_{1,j}} [L_x(t)\delta x(t) + L_u(t) \cdot \delta u(t)]\, dt \tag{B.24}$$

The next step is to determine a set of "influence functions" on the cost functional, J, and the state constraint equations, ψ. First, we define an n-vector λ which satisfies the differential equation

$$\dot{\lambda}(t) = -L_x^T(t) - F^T(t) \cdot \lambda(t) \tag{B.25}$$

Noting that

$$\frac{d}{dt}(\lambda^T \cdot \delta x) = \dot{\lambda}^T \cdot \delta x + \lambda^T \cdot \delta \dot{x}$$

$$= -L_x \delta x - \lambda^T F \delta x + \lambda^T F \delta x + \lambda^T G \delta u$$

$$= -L_x \cdot \delta x + \lambda^T \cdot G \cdot \delta u \qquad \text{(B.26)}$$

we integrate Eq. B.26 to obtain

$$\lambda^T(t_{1,j}) \cdot \delta x(t_{1,j}) - \lambda^T(t_0) \cdot \delta x(t_0) = \int_{t_0}^{t_{1,j}} (-L_x \delta x + \lambda^T G \delta u) dt \qquad \text{(B.27)}$$

We now define the terminal value of $\lambda(t)$ to be

$$\lambda(t_{1,j}) = [\phi_x - (\phi_x f + \phi_t + L)(\theta_x f + \theta_t)^{-1} \theta_x]^T_{t=t_{1,j}} \qquad \text{(B.28)}$$

and since $\delta x(t_0)$ is zero, Eqs. B.27 and B.28 can be substituted into Eq. B.24 to reduce it to

$$dJ = \int_{t_0}^{t_{1,j}} (L_x \cdot \delta x + L_u \cdot \delta u - L_x \cdot \delta x + \lambda^T \cdot G \cdot \delta u) dt$$

$$= \int_{t_0}^{t_{1,j}} [H_u(t) \cdot \delta u(t)] \, dt \qquad \text{(B.29)}$$

where H is defined as

$$H = L(x, u, t) + \lambda^T \cdot f(x, u, t) \qquad \text{(B.30)}$$

Similarly Eq. B.23 can be reduced by defining an $(n \times k)$-matrix, $\Lambda(t)$, which satisfies the adjoint differential equation

$$\dot{\Lambda}(t) = -F^T(t) \cdot \Lambda(t) \qquad \text{(B.31)}$$

and the terminal boundary condition

$$\Lambda(t_{1,j}) = [\psi_x - (\psi_x f + \psi_t)(\theta_x f + \theta_t)^{-1}\theta_x]^T_{t=t_{1,j}} \tag{B.32}$$

As before, we note that

$$\begin{aligned} \frac{d}{dt}(\Lambda^T \cdot \delta x) &= \dot{\Lambda}^T \cdot \delta x + \Lambda^T \cdot \delta \dot{x} \\ &= -\Lambda^T \cdot F \cdot \delta x + \Lambda^T \cdot F \cdot \delta x + \Lambda^T \cdot G \cdot \delta u \qquad \text{(B.33)} \\ &= \Lambda^T \cdot G \cdot \delta u \end{aligned}$$

Integrating Eq. B.33 we have

$$\Lambda^T(t_{1,j}) \cdot \delta x(t_{1,j}) - \Lambda^T(t_0) \cdot \delta x(t_0) = \int_{t_0}^{t_{1,j}} [\Lambda^T G \cdot \delta u]\, dt \tag{B.34}$$

Since $\delta x(t_0)$ is zero, Eq. B.23 now becomes

$$d\psi = \int_{t_0}^{t_{1,j}} [\Lambda^T G\, \delta u]\, dt \tag{B.35}$$

Thus $H_u(t)$ and $\Lambda^T(t) \cdot G(t)$ serve as influence functions of the perturbation in the control variable on the cost and state constraints, respectively.

In order to insure the validity of this linearized analysis, the size of the control perturbation must be constrained. The measure of the step size is given by

$$(dE)^2 = \int_{t_0}^{t_{1j}} [\delta u^T(t)\, W^{-1}(t) \cdot \delta u(t)]\, dt \tag{B.36}$$

where $W(t)$ is an arbitrary weighting matrix chosen to improve convergence.

Equations B.17 and B.29 constitute the state differential equation and cost functional, respectively, for a new optimization problem, subject to constraints given by Eqs. B.35 and B.36. Applying the calculus of variations to this new optimization problem, the control perturbation $\delta u(t)$ which minimizes the change in the cost functional (i.e., maximize the decrease in J) is found to be

$$\delta u_{j+1}(t) = K_J W(H_u^T - G^T \Lambda \cdot I_{\psi\psi}^{-1} \cdot I_{\psi J}) + W G^T \Lambda \cdot I_{\psi\psi}^{-1} \; d\psi_{j+1} \qquad (B.37)$$

where $I_{\psi\psi}$ and $I_{\psi J}$ are a (k x k) matrix and a k-vector, respectively, defined by

$$I_{\psi\psi} = \int_{t_0}^{t_{1,j}} \Lambda^T \cdot GWG^T \Lambda \, dt \qquad (B.38)$$

$$I_{\psi J} = \int_{t_0}^{t_{1,j}} \Lambda^T \, GW \, H_u^T \, dt \qquad (B.39)$$

The first term in Eq. B.37 corresponds to the component of the cost gradient in function space which is parallel to the terminal state constraint surface, and the second term corresponds to the component of the cost gradient orthogonal to the constraint surface. Thus at each iteration the choices of K_J and $d\psi_{j+1}$ exert separate control over the change in the cost and the change in the constraint violations, respectively.

The value of $d\psi_{j+1}$ specified at each iteration can be chosen to be some fraction of the constraint violation on the previous iteration, i.e.,

$$d\psi_{j+1} = -K_{\psi, j+1} \cdot \psi[x_j(t_{1,j}), \, t_{1,j}] \qquad (B.40)$$

The larger one makes K_ψ $(0 \leq K_\psi \leq 1)$, the more rapid will be the convergence to the constraint surface. By choosing large K_ψ one can

first converge to a control which satisfies the terminal constraints, and then, on successive iteration one can maintain the satisfaction of the constraint equations while searching for the control which minimizes J. However, such a procedure may increase the chance of converging to local minimum rather than the absolute minimum, as pointed out by Hague.[27] In addition if the component of $\delta u(t)$ due to $d\psi$ is too large, the linearized analysis may be invalid and the effectiveness of the algorithm will be diminished. Thus one may try to choose K_ψ so that the second term in Eq. B.30 is of comparable magnitude to the first term.

The choice of K_J can be made in several ways. First, by substituting Eq. B.37 into Eq. B.36, one can solve for K_J in terms of a specified $(dE)^2$ for each iteration as being[23]

$$K_J = \pm\left[\frac{dE^2 - d\psi^T \cdot I_{\psi\psi}^{-1}\, d\psi}{I_{JJ} - I_{\psi J}^T \cdot I_{\psi\psi}^{-1} \cdot I_{\psi J}}\right]^{1/2} \tag{B.41}$$

where

$$I_{JJ} = \int_{t_0}^{t_{1,j}} [H_u\, W H_u^T]\, dt \tag{B.42}$$

The plus sign is used to maximize J and the minus sign is used to minimize J. A second alternative is to specify the desired change in cost,[24] ΔJ. Substituting Eq. B.37 into Eq. B.29 one obtains

$$K_J = \frac{\Delta J - I_{\psi J}^T \cdot I_{\psi\psi}^{-1} \cdot d\psi}{I_{JJ} - I_{\psi J}^T \cdot I_{\psi\psi}^{-1} \cdot I_{\psi J}} \tag{B.43}$$

These two methods use only the first order information which is available. A third approach attempts to incorporate second order information into the choice of K_J. By analogy with the discussion of ordinary minimization problem in Section B.1, the choice of K_J corresponds to specifying the length of d_1, the size of the step in the

direction parallel to the constraint surface. To speed up convergence, one may wish to move to a minimum of J in the d_1 direction. To do this we express J as a function of K_J to second order terms, and then determine the value $\bar{K}_J$ which minimizes this approximating function,[27] i.e.,

$$J(K_J) = J(0) + \left.\frac{dJ}{dK_J}\right|_{K_J=0} \cdot K_J + \frac{1}{2}\left.\frac{d^2J}{dK_J^2}\right|_{K_J=0} \cdot K_J^2 + \ldots \qquad (B.44)$$

The minimum of $J(K_J)$ occurs for

$$\bar{K}_J = -\left[\left.\frac{dJ}{dK_J}\right|_0\right] \cdot \left[\left.\frac{d^2J}{dK_J^2}\right|_0\right]^{-1} \qquad (B.45)$$

Substituting Eq. B.37 into Eq. B.29 with K_ψ equal to zero, we find that

$$\left.\frac{dJ}{dK_J}\right|_{K_J=0} = I_{JJ} - I_{\psi J}^T \cdot I_{\psi\psi}^{-1} \cdot I_{\psi J} \qquad (B.46)$$

In order to obtain an estimate of the second-derivative we must apply the $\delta u(t)$ given by Eq. B.37 with an arbitrary gain K_{J_1} (and with K_ψ zero) to obtain the new cost $J(K_{J1})$; then

$$J(K_{J1}) = J(0) + \left.\frac{dJ}{dK_{J1}}\right|_0 \cdot K_{J_1} + \frac{1}{2}\left.\frac{d^2J}{dK_J^2}\right|_0 \cdot K_{J1}^2 \qquad (B.47)$$

from which

$$\left.\frac{d^2J}{dK_J^2}\right|_{K_J=0} = 2\left[J(K_{J1}) - J(0) - \left.\frac{dJ}{dK_J}\right|_0 \cdot K_{J1}\right]/K_{J1}^2 \qquad (B.48)$$

Substituting Eqs. B.46 and B.48 into Eq. B.45, we compute the desired optimum value of K_J based on second order information. While this

method is highly desirable since it speeds up the convergence as one gets near to the solution, it must be used with caution when far from the solution, since the second order approximation may no longer be valid.

The steepest-descent algorithm is easily modified to treat problems with bounded controls.[26] To demonstrate this extension let us limit the discussion to scalar control with constraints of the form

$$|u| \leq U \tag{B.49}$$

After the j^{th} iteration the control function $u_j(t)$ function may, typically, lie in the boundary of its admissible region over several time segments, as shown in Fig. B.2 for $t_1 < t < t_2$ and $t_3 < t < t_f$. If one computes the control perturbation, $\delta u'_{j+1}(t)$, as before a new control $u'_{j+1}(t)$ will be obtained which may violate the constraints. Since $u_{j+1}(t)$ must be limited to the set of admissible controls given by Eq. B.49, the desired change in the terminal constraint violations, $d\psi_{j+1}$, will not be achieved. One can accommodate the fact that a positive perturbation cannot be allowed during intervals on the upper boundary and a negative perturbation cannot be allowed when on the lower boundary by increasing the weighting of control perturbations in Eq. B.36 over these intervals. Thus by increasing $W^{-1}(t)$ over the intervals when $u_j(t)$ is on the boundaries of its admissible region, $W(t)$ can be made arbitrarily small. In the limit, the effect is to suppress the integration of the integrals $I_{\psi\psi}$, $I_{\psi J}$, and I_{JJ} (see Eqs. B.38, B.39, and B.42) as well as the computation of $\delta u_{j+1}(t)$ in Eq. B.37 over these intervals. Since $I_{\psi\psi}$ will be decreased (and $I^{-1}_{\psi\psi}$ thereby increased) the control perturbation will be scaled up over the remaining intervals to enable the achievement of the desired $d\psi_{j+1}$. However, while perturbations which would result in a violation of the constraints have been eliminated, one has also excluded the possibility of computing perturbations which will take the control off of the boundaries into the interior of the admissible region. For this reason the suggested algorithm is to set W equal to zero over the appropriate intervals only in

the computation of the integrals $I_{\psi\psi}$, $I_{\psi J}$, and I_{JJ}. The new control perturbation, $\delta u_{j+1}(t)$, is to be computed for all t and then $u_{j+1}(t)$ limited to the admissible region, by truncating $\delta u_{j+1}(t)$ as shown in Fig. B.2.

If the control problem is such that control remains on the boundary for all t (i.e., a bang-bang control), then this method is not directly applicable (since then $I_{\psi\psi}=0$), and further modifications are required as suggested by Denham and Bryson.[26] If there is more than one control variable, then one simply makes the i^{th} row and i^{th} column of the matrix $W(t)$ equal to zero, in the computation of $I_{\psi\psi}$, $I_{\psi J}$, and I_{JJ}, over the intervals for which the i^{th} component of the control vector is on the boundaries of its admissible region.

The steepest-descent algorithm can be summarized as follows:

1. With the nominal control function $u_j(t)$ ($u_0(t)$ arbitrarily chosen for first iteration), and the initial conditions given in Eq. B.12, integrate Eq. B.11 from t_0 until the stopping condition Eq. B.16, is satisfied, thus defining the nominal terminal time $t_{1,j}$.

2. Compute boundary conditions on λ and Λ at $t=t_{1,j}$ from Eqs. B.28 and B.32.

3. Integrate $\lambda(t)$, $\Lambda(t)$, $I_{\psi\psi}$, $I_{\psi J}$, I_{JJ} backwards in time from $t_{1,j}$ to t_0 using Eqs. B.25, B.31, B.38, B.39, and B.42, saving the values of λ and Λ over the entire interval and the terminal values of $I_{\psi\psi}$, $I_{\psi J}$, and I_{JJ}.

4. Determine the gain K_J from either Eq. B.41, B.43, or B.45 and specify desired $d\psi_{j+1}$.

5. Compute $\delta u_{j+1}(t)$ from Eq. B.37 and the new nominal control

$$u_{j+1}(t) = u_j(t) + \delta u_{j+1}(t) \qquad \text{(B.50)}$$

6. Repeat the above procedure starting with step 1 until satisfactory convergence is obtained.

One can determine when the minimal solution has been obtained by comparing at successive iterations, the values of J, $\psi(t_{1,j})$, $t_{1,j}$, $H_u(t)$, and $\delta u(t)$. From the necessary conditions on the optimal control, Appendix A, H_u should be zero except when u is on the boundary of the set of admissible controls, and thus it serves as a measure of the violation of the necessary conditions. However, it is possible that H_u can be quite small while the change in u(t) required to reach the minimum can be quite large if the problem contains a "shallow" minimum. Thus a more direct approach is to compare directly the cost at each iteration and accept a solution when the decrease in the cost achieved falls below a preset percentage of the cost while maintaining the violations of the terminal state constraints within a set of predetermined tolerances. It is a good practice to make these comparisons over a span of several iterations, since one may likely make little improvement on any given iteration while achieving significant gains on succeeding iterations. One should be aware of the fact that he may only have attained a local minimum. Since it is in general difficult to prove uniqueness of extremals for a given problem, one is forced to resort to argue the reasonableness of the answer and to rely on engineering intuition in order to accept a given solution as being indeed the absolute minimum. One may also try repeating the algorithm with a wide variety of initial control functions, $u_0(t)$. Continual convergence to the same solution can bolster one's confidence in the solution, but, of course, there is no guarantee that one has not failed to assume a $u_0(t)$ close enough to the minimal $\bar{u}(t)$ in order to converge to it rather than a local minimum.

From the above summary of the steepest-descent algorithm it can be seen that the number of variables which must be integrated are as follows: the n-vector of state variables, x; the n-vector of adjoint variables, λ; the (n x k)-matrix of influence functions, Λ; the (k x k)-matrix $I_{\psi\psi}$; the k-vector $I_{\psi J}$; and the scalar I_{JJ}. Since $I_{\psi\psi}$ is symmetric only $k(k+1)/2$ of its k^2 elements need to be integrated giving a total of

$$I = \frac{1}{2}(2n + k + 1)(k + 2) \qquad (B.51)$$

variables to be integrated. In terms of computer storage requirements, one must store the m-vector of control variables, the n-vector of state variables, the (k x m) matrix Λ_G^T, and the m-vector H_u, for a total of

$$S = m(k + 2) + n \tag{B.52}$$

variables to be stored. If N is the total number of integration increments used, then S x N words of computer memory are required. In addition, one may wish to store the two components of δu given by Eq. B. 37, so that if it is necessary to repeat an iteration with a smaller step size (or if second-order information is desired in specifying K_J), it will not be necessary to recompute the control perturbation.

APPENDIX C

THE NEWTON-RAPHSON METHOD

C.1 INTRODUCTION

The Newton-Raphson method[33] for the solution of two-point boundary value problems is a direct extension of the classical Newton method for determining the roots of a function of a single variable.[38] It is applicable to minimization problems since one can find a (local) minimum of a function by searching for that point at which its first derivative is zero. Thus, the method is an indirect one, and, as will be seen, it makes use of second-order information. It was first proposed for the solution of two-point boundary value problems by Hestenes[52] who called it "differential variations." Bellman and Kalaba[41-44] developed and generalized the method further, incorporating the ideas of dynamic programming, and called it "quasilinearization." The generalization of Newton's method to function spaces, which constitutes a special case of quasilinearization, was originated by Kantorovich.[38,53] Convergence theorems are presented by Kantorovich,[53] Kalaba,[41] and McGill and Kenneth.[37] The basic algorithm was extended to problems with bounded control variables by Kenneth and Taylor,[36] and to problems with bounded state variables by McGill.[34] Further modifications were developed by others[35,45] to extend the basic Newton method to a wider class of control problems. In this appendix, the Newton-Raphson method for fixed end-point, two-point optimal control problems is reviewed. In Chapter III the algorithm is extended to the general N-point optimal control problem and to free terminal-time problems.

C.2 NEWTON'S METHOD

In this section the basic Newton method for finding the roots of a (vector) function of several variables is presented. Consider a set of simultaneous nonlinear equations

$$f(x) = 0 \tag{C.1}$$

where x is an n-vector and f is a continuously differential n-vector

function of n variables. It is desired to find that $\bar{x}$ which represents a solution of Eq. C.1. Expanding Eq. C.1 in a Taylor series about a nominal x_0, one obtains

$$f(x_1) = f(x_0) + J(x_0) \cdot (x_1 - x_0) + \ldots \tag{C.2}$$

where the second and higher-order terms are neglected and $J(x_0)$ is the Jacobian matrix of $f(x)$ given by

$$J(x_0) = \frac{df}{dx}(x_0) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}_{x = x_0} \tag{C.3}$$

If the Jacobian is nonsingular (i.e., its determinant is nonzero), one can solve for the new nominal x_1, by requiring that $f(x_1)$ be zero.

$$x_1 = x_0 - J(x_0)^{-1} \cdot f(x_0) \tag{C.4}$$

The algorithm can be repeated in an iterative manner by using the more general relation

$$x_{n+1} = x_n - J(x_n)^{-1} \cdot f(x_n) \tag{C.5}$$

Since the algorithm consists of approximating the function $f(x)$ at each iteration by its tangent at the nominal point, and then finding the root of this approximating function, the Newton method is often called the "method of tangents." Since Eq. C.5 is obtained from Eq. C.2 by considering only linear terms, the algorithm will converge exactly in one step if and only if Eq. C.1 is linear with respect to all n variables. If $f(x)$ is not linear but the initial guess, x_0, is "close enough" to $\bar{x}$, the convergence may be quadratic, i.e., the number of correct significant figures on successive iterations approximately doubles (it may be only asymptotically quadratic if the initial guess is poorer). Unfortunately, the method does not always converge quadratically, and,

in fact, it may not converge at all. For example, if $f(x) = \frac{1}{6}x^3$, then

$$x_{n+1} = x_n - (2x_n^2)^{-1} \cdot \frac{1}{6}x_n^3 = \frac{2}{3}x_n \tag{C.6}$$

Thus, the distance to the solution, $\bar{x} = 0$, is decreased by only a factor of $\frac{2}{3}$ at each iteration. On the other hand, if $f(x)$ is given by

$$f(x) = |x|^{1/2} \cdot \text{sgn}(x) \tag{C.7}$$

the algorithm gives

$$x_{n+1} = -x_n \tag{C.8}$$

which does not converge at all.

The reader is referred to Kantorovich and Akilov[38] for a precise statement of the fundamental convergence theorem and its detailed proof. In many applications, as in finding the zero of $\frac{\partial \bar{J}}{\partial t_1}(t_1)$ in Section 3.4, the convergence theorems are of little value since the functional form of $f(x)$ is unknown.

Application of the Newton method to function minimization problems simply involves letting the derivative of the function equal $f(x)$. The minimum (local minima) of a function of n variables, $P(x)$, occurs at a point where x is an n-vector as before and P is twice differentiable with respect to each of its variables.

$$\frac{dP}{dx} = \left(\frac{\partial P}{\partial x_1}, \ldots, \frac{\partial P}{\partial x_n}\right) = 0 \tag{C.9}$$

i.e., the n-vector of partial derivatives, the gradient vector, must be zero. One can now apply the Newton algorithm, Eq. C.5, directly to Eq. C.9, where

$$f(x) = \frac{dP}{dx}(x) \tag{C.10}$$

and

$$J(x) = \begin{bmatrix} \frac{\partial^2 P}{\partial x_1^2}(x) & \cdots & \frac{\partial^2 P}{\partial x_1 \partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial^2 P}{\partial x_1 \partial x_n}(x) & \cdots & \frac{\partial^2 P}{\partial x_n^2}(x) \end{bmatrix} \tag{C.11}$$

If $P(x)$ contains only linear and quadratic functions of its variables, $\frac{dP}{dx}(x)$ is linear, and, as noted above, the algorithm converges to the exact minimum in one step. For general nonlinear $P(x)$, the convergence theorem of Kantorovich can be applied directly to $\frac{dP}{dx}(x)$, in order to determine if the convergence is quadratic.

Unfortunately, in addition to perhaps diverging, the method may converge to a local maximum or a saddle point, as well as a local minimum, since Eq. C.9 is the necessary condition for any stationary point. For example, consider the function

$$P(x) = \frac{1}{6} \cdot x^3 - x \tag{C.12}$$

which is illustrated in Fig. C.1, and which, incidently, does not possess a minimum. The first derivative of P is given by

$$\frac{dP}{dx} = \frac{1}{2} x^2 - 1 \tag{C.13}$$

and the Newton algorithm becomes

$$x_{n+1} = x_n - (\frac{1}{2} x_n^2 - 1)^{-1}(\frac{1}{6} x_n^3 - x_n) = \frac{x_n^3}{3(\frac{1}{2} x_n^2 - 1)} \tag{C.14}$$

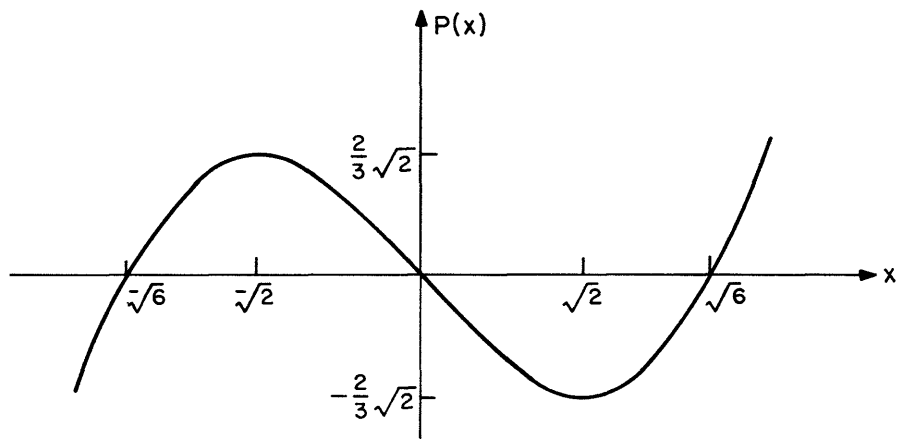
For any positive x_0 the algorithm will converge to the relative minimum at $x = +\sqrt{2}$ and for any negative x_0 it will converge to the relative maximum at $x = -\sqrt{2}$. Thus, one must check the second derivative at any solution to which the algorithm converges in order to determine if that solution is a relative minimum. In addition, one cannot tell if there is another relative minimum which is smaller, or if an absolute minimum even exists.

C.3 THE NEWTON-RAPHSON METHOD FOR TWO-POINT BVP's

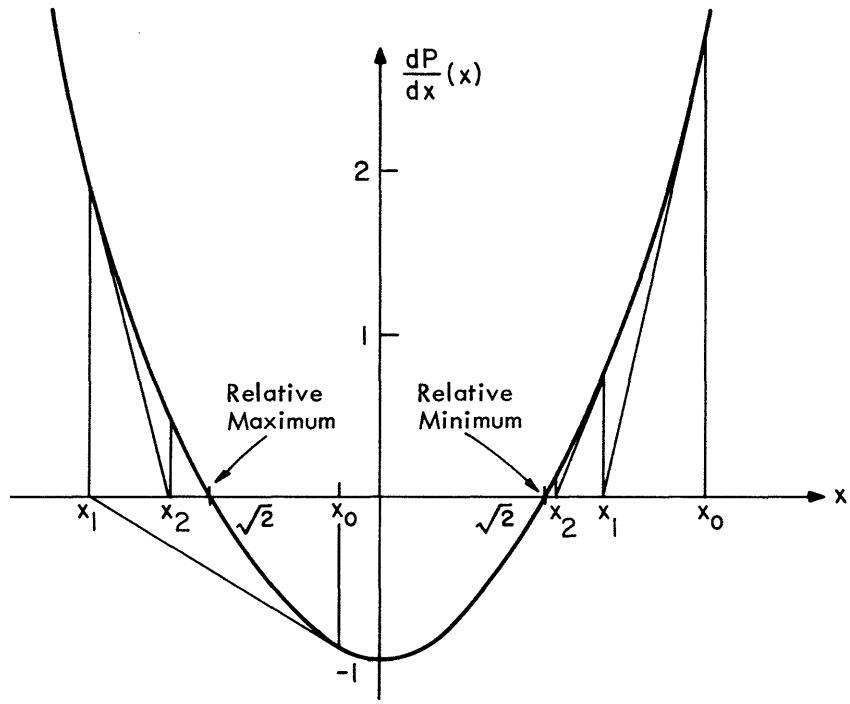
When one applies the N-R method to two-point BVP's, the objective is to determine the solution of a nonlinear differential equation

$$\dot{X}(t) = F(X(t), t) \tag{C.15}$$

where $X(t)$ is an N-vector, subject to $N/2$ boundary conditions at each



(a) $P(x) = (\frac{1}{6}\ x^3 - x)$ vs. x



(b) $\frac{dP}{dx} = \frac{1}{2}x^2 - 1$ vs. x

Fig. C.1 Example of Newton's Method

of two points in time,* such as

$$X^{(j)}(t_0) = a_j \tag{C.16}$$

$$X^{(j')}(t_1) = b_j' \tag{C.17}$$

where the superscripts denote the components of the N-vector X. If one lets $Y = \dot{X}$, then Eq. C.15 can be written

$$g(X,Y,t) = Y - F(X,t) = 0 \tag{C.18}$$

Expanding Eq. C.18 in a Taylor series up to first order terms about a nominal (X_i, Y_i) gives

$$g(X_{i+1},Y_{i+1},t) = 0 = g(X_i,Y_i,t) + \frac{\partial g}{\partial X}(X_i,Y_i,t)\cdot(X_{i+1}-X_i) + \frac{\partial g}{\partial Y}(X_i,Y_i,t)\cdot(Y_{i+1} - Y_i) + \ldots \tag{C.19}$$

$$0 = Y_i - F(X_i,t) - \frac{\partial F}{\partial X}(X_i,t)\cdot(X_{i+1} - X_i) + Y_{i+1} - Y_i \tag{C.20}$$

and thus, eliminating Y one obtains

$$\dot{X}_{i+1}(t) = \frac{\partial F}{\partial X}(X_i,t)\cdot X_{i+1}(t) + \left[F(X_i,t) - \frac{\partial F}{\partial X}(X_i,t)\cdot X_i(t)\right] \tag{C.21}$$

Equation C.21 represents a linear nonhomogenous differential equation to be satisfied by the N-vector $X_{i+1}(t)$. Because of the linearity of this differential equation, superposition can be employed to obtain a solution that satisfies the boundary conditions given by Eqs. C.16 and C.17.

The solution to Eq. C.21 can be written as

---

* This choice of boundary conditions is made to simplify the following discussion, and is subsequently generalized.

$$X_{i+1}(t) = \Phi(t,t_0)\cdot X_{i+1}(t_0) + \int_{t_0}^{t} \Phi(t_0,\tau)\cdot\left[F(X_i(\tau),\tau) - \frac{\partial F}{\partial X}(X_i(\tau),\tau)\cdot X_i(\tau)\right]d\tau \tag{C.22}$$

where $\Phi(t,t_0)$ is the fundamental, or "transition", matrix of Eq. C.21. By defining

$$\Phi_{p,i+1}(t) = \sum_{j=1}^{N/2} \Phi_j(t,t_0)\cdot a_j + \int_{t_0}^{t} \Phi(t_0,\tau)\cdot\left[F(X_i(\tau),\tau) - \frac{\partial F}{\partial X}(X_i(\tau),\tau)\cdot X_i(\tau)\right] d\tau \tag{C.23}$$

Equation C.22 can be written as

$$X_{i+1}(t) = \Phi_{p,i+1}(t) + \sum_{j=\frac{N}{2}+1}^{N} \Phi_j(t)\cdot C_j \tag{C.24}$$

where $\Phi_j(t)$ represents the jth column vector of the transition matrix. Thus, $\Phi_{p,i+1}(t)$ can be obtained by integrating Eq. C.21 forward with $N/2$ components set equal to the values specified by Eq. C.16, and the other $N/2$ components set equal to arbitrary constants at time t_0, i.e.,

$$\Phi_p(t_0) = \text{col}\,(a_1,\ldots,a_{\frac{N}{2}}, a_{\frac{N}{2}+1},\ldots,a_N) \tag{C.25}$$

if the state variables are ordered such that the first $N/2$ are specified at time t_0. $\Phi_j(t)$ can be obtained by integrating the homogeneous part of Eq. C.21 with the j^{th} component of $X(t_0)$ set equal to one and all other initial values set to zero (for $j = N/2 + 1,\ldots,N$). Evaluating Eq. C.24 at $t = t_1$, and setting the $N/2$ components of $X_{i+1}(t_1)$ equal to the terminal values specified by Eq. C.17, one obtains $N/2$ linear equations in terms of the $N/2$ unknown constants C_j, $j = N/2 + 1,\ldots,N$. Solving for the C_j, one then obtains the appropriate initial conditions for $X_{i+1}(t)$ given by

$$X_{i+1}(t_0) = \text{col}\,(a_1,\ldots,a_{\frac{N}{2}}, a_{\frac{N}{2}+1} + C_{\frac{N}{2}+1},\ldots, a_N + C_N) \tag{C.26}$$

The entire solution for $X_{i+1}(t)$ can be obtained directly from Eq. C.24, or, to obviate the necessity of storing the solutions for $\Phi_p(t)$ and the $\Phi_j(t)$ for all t, by integrating Eq. C.21 once more, with the initial conditions given by Eq. C.26.

The Newton-Raphson algorithm is not restricted to having exactly $N/2$ values of $X(t)$ specified at each end point--actually, the boundary conditions can be linear or, more generally, nonlinear functions of the state and can be split in any way between the two boundaries (as long as a total of N, and only N, boundary conditions are specified. Any initial values explicitly specified, as in Eq. C.16, can be included in the determination of $\Phi_p(t)$, as above. The effects of the remaining initial values must be determined by obtaining their transition vectors, $\Phi_j(t)$, as was done in the above development for the last $N/2$ components of $X(t)$. Of course, if the boundary conditions are all linear the problem of determining the unknown C_j's is greatly simplified, requiring the inversion of an $M \times M$ matrix, where M is the number of unknowns. The more initial values that are explicitly specified, the fewer unknowns there are to be determined. On the other hand, if a greater number of terminal values of $X(t)$ are explicitly specified, then one ought to apply the algorithm in reverse, integrating backwards from t_1 to t_0, in order to reduce the number of unknowns to be determined. If all of the N boundary values are explicitly specified, the case where exactly $N/2$ conditions are imposed at each end, then, represents a worse case as far as the number of unknowns to be determined is concerned. Unfortunately, this is the case in all optimal control problems. If the boundary conditions are not explicitly specified, one may have to compute the entire transition matrix and determine N unknowns.

The application to two-point optimal control problems with unbounded control and state vectors, and fixed initial and terminal times, is straightforward in certain problems. For such problems, the Euler-Lagrange equations are given by*

---

* See Appendix A, Section 7

$$\dot{x} = f(x,u,t) \tag{C.27}$$

$$\dot{\lambda} = -H_x(x,u,\lambda,t) = g(x,u,\lambda,t) \tag{C.28}$$

and

$$H_u(x,u,\lambda,t) = 0 \tag{C.29}$$

Let us assume that Eq. C.29 can be solved explicitly for u as a function of x and λ,

$$u = h(x,\lambda,t) \tag{C.30}$$

Equations C.27 and C.28 can then be rewritten as

$$\dot{x} = f[x,h(x,\lambda,t),t] \tag{C.31}$$

$$\dot{\lambda} = g[x,h(x,\lambda,t),\lambda,t] \tag{C.32}$$

which are a set of 2n differential equations in terms of the 2n variables x and λ. Let us define a new 2n vector $X(t)$ by

$$X(t) = \begin{bmatrix} x(t) \\ --- \\ \lambda(t) \end{bmatrix} \tag{C.33}$$

Equations C.31 and C.32 can be rewritten as a single vector differential equation,

$$\dot{X}(t) = F(X,t) = \begin{bmatrix} f[x,g(x,\lambda,t),t] \\ \\ g[x,g(x,\lambda,t),\lambda,t] \end{bmatrix} \tag{C.34}$$

which is of the form specified by Eq. C.15 where $N = 2n$. If the state is specified at both the initial and terminal times, one then has a set of $N/2$ boundary conditions at each end, as specified by Eqs. C.16 and C.17, and the algorithm represented by Eqs. C.21 - C.26 is directly applicable. If the state equality constraints are linear and the penalty functions on the state are quadratic, then the n boundary conditions on each end, the state constraints and the transversality conditions, are linear functions of the state and costate variables which can be easily solved for the unknown initial conditions. If these restrictions are not met, one must solve a set of nonlinear equations for the unknown initial conditions, a somewhat more difficult task.

Since Eqs. C.27 - C.29 represent first-order necessary conditions that are satisfied by any stationary solution, the Newton-Raphson algorithm may converge to a local maximum rather than a local minimum. In addition, if the initial guessed solution $(x_0(t), \lambda_0(t))$ is not "close enough" to the desired solution, the algorithm may not converge at all. Of course, if the system dynamics and state constraints are linear, and if the cost functional is a quadratic function of the state and the control, the Newton-Raphson algorithm converges exactly in only one step, regardless of the initial guess. The reader is referred to the references cited in Section C.1 for convergence theorems and their proofs.

The above development is limited to fixed-time problems, and to problems in which one can obtain u as an explicit function of x and λ (Eq. C.30). The algorithm is extended to free-time problems in Section 3.4e. If one cannot solve Eq. C.29 explicitly for u, Eqs. C.27 - C.29 must be linearized about a nominal $x(t)$, $\lambda_N(t)$, and $u_N(t)$. The linearization of Eq. C.29 can then be used (if H_{uu} is nonsingular for all t) to eliminate $u_{N+1}(t)$, resulting in a set of $2n$ differential equations in terms of the $2n$ variable $x_{N+1}(t)$ and $\lambda_{N+1}(t)$. One then proceeds, as above, to find the solution for $x_{N+1}(t)$ and $\lambda_{N+1}(t)$ that satisfies the initial and terminal constraints.

APPENDIX D

STEEPEST-DESCENT COMPUTER PROGRAM FOR TWO-TARGET INTERCEPT

This appendix contains the computer program used in obtaining the numerical results via the method of steepest-descent as discussed in Section 3.3b and 4.3. A flow chart for the program is given in Fig. 3.7. The input data is read in the NAMELIST format with the name INPUT1. The input variables include the initial positions and headings of the missile (X10, X20, X30) and the targets (Y10, Y20, Y30, Z10, Z20, Z30) and the target turning rates and velocities (Y3D, Z3D, VY, VZ). The targets are constrained to have constant turning rates. The nominal control, U, can be read in as a constant (U0) if INFORM is equal to one, or as a one-dimensional array (of dimension 1000) under the NAMELIST name INPUT2 if INFORM is set equal to two. The integration step size, the printout frequency, and the maximum number of iterations are specified by DT, NWRITE, and NMAX, respectively. If both targets are not intercepted after IMAX time increments (of length DT), then the run is automatically terminated. The maximum control and the gains K_J and K_0 which determine the control perturbation step size are specified by UMAX, GAINJ, and DP, respectively. The tolerances on the constraint violations and the change in the cost functional are given by ERP and ERJ. If the constraint violations are less than ERP on two successive iterations and the percentage change in the cost functional is less than ERJ, then the algorithm is considered to have converged and the run is terminated.

The forward integration of the state equations and the backward integration of the influence functions and the integrals $I_{\psi\psi}$ and $I_{\psi J}$ are performed using the second-order Runge-Kutta method. At each integration increment the transition and stopping conditions are tested in order to define the nominal intercept times. If in a given increment the appropriate condition changes sign, the intercept time (i.e., the time at which the condition is zero) is precisely determined by linear interpolation. Since the intercept of targets Y and Z are defined as the times when the value X2 first exceeds the values of Y2 and Z2, respectively, the initial conditions should be chosen so that X2 is less

than Y2 and Z2. Target Y should be defined as the one that is the first to be intercepted.

Since the intercept times t_1 and t_2, in general, occur at times other than an integral number of DT's, the influence functions and the integrals $I_{\psi\psi}$ and $I_{\psi J}$ are integrated only over the appropriate fractions of the DT increment at these times. In addition, for bounded control problems (in which $I_{\psi\psi}$ and $I_{\psi J}$ are not to be integrated over the intervals when the control lies on the boundary) the precise time at which the control enters or leaves the boundary is determined by linearly extrapolating the values of the control in the region just off the boundary. $I_{\psi\psi}$ and $I_{\psi J}$ are then integrated only over that portion of the DT for which the control is not on the boundary.

The algorithm for specifying the control perturbation step size is discussed in Section 3.3b. The control perturbation is then added to the nominal control to obtain a new nominal (which is truncated in bounded control problems so that it lies entirely in the admissible region), and the algorithm is repeated until the optimal solution is obtained.

```
C
C          STEEPEST DESCENT - OPTIMAL INTERCEPT OF TWO TARGETS
C
0001        DIMENSION X1(1000),X2(1000),X3(1000),P3(1000),P13(1000),
           1   P23(1000),U(1000),Y1(1000),Y2(1000),Z1(1000),Z2(1000),
           2   SX3(1000),CX3(1000),LIMIT(1000)
0002        REAL IPJ1,IPJ2,IPP11,IPP12,IPP22,I11,I12,I22
0003        NAMELIST/INPUT1/X10,X20,X30,Y10,Y20,Y30,Z10,Z20,Z30,Y3D,Z3D,
           1   VY,VZ,U0,DT,INFORM,NWRITE,IMAX,NMAX,DP,GAINJ,UMAX,ERP,ERJ
0004        NAMELIST/INPUT2/ U
0005        NAMELIST/VARS/ COST0,COST,D1,D2,D3,D4,D5,D6,D7,D10,D20,N1,
           1   N2,DP1,DP2,DP,KP,IPJ1,IPJ2,IPP11,IPP12,IPP22,I11,I12,
           2   I22,ITERAT,T1,T2,GAIN1,EPSLN1,EPSLN2,UT2
0006      1   FORMAT('1',30X,'INPUT DATA'///)
0007      2   FORMAT('1',30X,'OUTPUT DATA'///)
0008      3   FORMAT('1',1X,'I=',8X,'X1(I)=',12X,'X2(I)=',12X,'X3(I)=',12X,
           1  'Y1(I)=',12X,'Y2(I)='/12X,'P3(I)=',11X,'P13(I)=',11X,
           2   'P23(I)=',12X,'Z1(I)=',12X,'Z2(I)=',13X,'U(I)='//)
0009      4   FORMAT(I4,3X,E15.7,3X,E15.7,3X,E15.7,3X,E15.7,3X,E15.7/7X,
           1   E15.7,3X,E15.7,3X,E15.7,3X,E15.7,3X,E15.7,3X,E15.7)
0010      5   FORMAT('1',30X,'RESULTS FOR ITERATION NO.',I4///)
0011      6   FORMAT(10(I4,I3))
0012      7   FORMAT('1',25X,'TURNING RATE LIMIT DATA- I,LIMIT(I)'/30X,
           1   'SATURATION IF LIMIT(I)=1'//)
C     READ INPUT DATA
0013      100 WRITE(6,1)
0014          READ(5,INPUT1)
0015          WRITE(6,INPUT1)
0016          WRITE(6,2)
0017          IF(INFORM.EQ.1) GO TO 103
C     READ IN NOMINAL CONTROL HISTORY IF NOT A CONSTANT
0018          READ(5,INPUT2)
C     INITIALIZE PARAMETERS
0019      103 COST0=0.
0020          N2=IMAX
0021          KN=0
0022          KP=0
0023          X1(1)=X10
0024          X2(1)=X20
0025          X3(1)=X30
0026          Y1(1)=Y10
0027          Y2(1)=Y20
0028          Z1(1)=Z10
0029          Z2(1)=Z20
0030          D10=0.
0031          D20=0.
0032          ITERAT=0
0033          NERROR=0
0034          IJK=0
C     COMPUTE TARGET TRAJECTORIES - CONSTANT TURNING RATES
0035          Y3N=Y30
0036          Z3N=Z30
0037          VYN=VY*DT
0038          VZN=VZ*DT
0039          DO 101 I=2,IMAX
0040          K=I-1
0041          Y1(I)=Y1(K)+VYN*COS(Y3N)
0042          Y2(I)=Y2(K)+VYN*SIN(Y3N)
0043          Y3N=Y3N+DT*Y3D
0044          Z1(I)=Z1(K)+VZN*COS(Z3N)
0045          Z2(I)=Z2(K)+VZN*SIN(Z3N)
0046          Z3N=Z3N+DT*Z3D
0047          LIMIT(K)=0
0048          P3(K)=0.
0049          P13(K)=0.
```

```
0050              P23(K)=0.
0051              IF(INFORM.NE.1) GO TO 101
0052              U(K)=U0
0053            101 CONTINUE
0054              C=DT/2.
0055              SX3(1)=SIN(X3(1))
0056              CX3(1)=COS(X3(1))
                C   INTEGRATE MISSILE DYNAMICS
0057            102 DO 105 I=2,N2
0058              K=I-1
                C   IF CONTROL ENTERS OR LEAVES SATURATION, MUST FIND
                C             TRANSITION POINT ACCURATELY
0059              IF(LIMIT(K).EQ.1.AND.LIMIT(I).EQ.0) GO TO 300
0060              IF(LIMIT(K).EQ.0.AND.LIMIT(I).EQ.1) GO TO 301
0061              X3(I)=X3(K)+C*(U(K)+U(I))
0062            106 CX3(I)=COS(X3(I))
0063              SX3(I)=SIN(X3(I))
0064              X1(I)=X1(K)+C*(CX3(I)+CX3(K))
0065              X2(I)=X2(K)+C*(SX3(I)+SX3(K))
0066              IF(KN.EQ.1) GO TO 104
                C   CHECK SWITCHING CONDITION ON FIRST TARGET
0067              IF(X1(I)-Y1(I).LT.0.) GO TO 105
0068              KN=1
0069              N1=I
                C   FIND INTERMEDIATE INTERCEPT TIME EXACTLY
0070              THETA2=X1(I)-Y1(I)
0071              THETA1=X1(I-1)-Y1(I-1)
0072              EPSLN1=THETA2/(THETA2-THETA1)
0073              T1=((I-1)-EPSLN1)*DT
                C   CHECK STOPPING CONDITION ON SECOND TARGET
0074            104 IF(X1(I)-Z1(I).GT.0.) GO TO 111
0075            105 CONTINUE
0076              I=N2+1
0077            112 K=I-1
                C   IF PAST PREVIOUS NOMINAL TERMINAL INTERCEPT TIME, MAINTAIN
                C         CONSTANT X3 UNTIL NEW INTERCEPT DEFINED
0078              U(I)=0.
0079              X3(I)=X3(K)
0080              SX3(I)=SIN(X3(I))
0081              CX3(I)=COS(X3(I))
0082              X1(I)=X1(K)+DT*CX3(K)
0083              X2(I)=X2(K)+DT*SX3(K)
                C   CHECK IF EXCEED MAX RUNNING TIME
0084              IF(I.GT.IMAX) GO TO 118
0085            116 IF(X1(I)-Z1(I).GT.0.) GO TO 111
0086              I=I+1
0087              GO TO 112
0088            111 N2=I
0089              KN=0
                C   FIND TERMINAL INTERCEPT TIME EXACTLY
0090              THETA2=X1(I)-Z1(I)
0091              THETA1=X1(I-1)-Z1(I-1)
0092              EPSLN2=THETA2/(THETA2-THETA1)
0093              T2=(I-1-EPSLN2)*DT
                C   COMPUTE COST
0094              UT2=U(N2-1)+(U(N2)-U(N2-1))*(1.-EPSLN2)
0095              COST=(U(1)*U(1)+UT2*UT2*(1.-EPSLN2)-U(N2-1)*U(N2-1)*EPSLN2)/2.
0096              NF=N2-1
0097              DO 113 I=2,NF
0098            113 COST=COST+U(I)*U(I)
0099              COST=COST*C
                C   COMPUTE CONSTRAINT VIOLATIONS
0100              ER2=X2(N1)-Y2(N1)
0101              ER1=X2(N1-1)-Y2(N1-1)
0102              D1=ER1+(ER2-ER1)*(1.-EPSLN1)
```

```
0103              ER2=X2(N2)-Z2(N2)
Q104              ER1=X2(N2-1)-Z2(N2-1)
0105              D2=ER1+(ER2-ER1)*(1.-EPSLN2)
0106              D3=D1-D10
0107              D4=D2-D20
                C   CHECK IF CONSTRAINT VIOLATION WITHIN TOLERANCE ON TWO
                C             SUCCESSIVE ITERATIONS
0108              IF(ABS(D1)+ABS(D2).GT.ERP) GO TO 120
0109              IF(KP.EQ.0) GO TO 119
0110              IF(ABS(1.-COST0/COST).GT.ERJ) GO TO 114
0111            118 NWRITE=1
0112              NERROR=1
0113              GO TO 114
0114            119 KP=1
0115              GO TO 114
0116            120 KP=0
0117            114 WRITE(6,5) ITERAT
0118              WRITE(6,VARS)
0119              WRITE(6,3)
0120              WRITE(6,4) (I,X1(I),X2(I),X3(I),Y1(I),Y2(I),P3(I),
                 1   P13(I),P23(I),Z1(I),Z2(I),U(I),I=1,N2,NWRITE)
0121              WRITE(6,7)
0122              WRITE(6,6) (I,LIMIT(I),I=1,N2)
0123              IF(ITERAT.GE.NMAX) GO TO 100
0124              IF(NERROR.EQ.1) GO TO 100
0125              ITERAT=ITERAT+1
0126              COST0=COST
0127              IF(ITERAT.EQ.1) GO TO 117
0128              IF(ABS(D1).GT.ABS(D2)) GO TO 128
0129              D5=D4
0130              D6=DP2
0131              GO TO 132
0132            128 D5=D3
0133              D6=DP1
0134            132 D7=D5/D6
0135              IF(D7.LE.0..OR.D7.GE.2.) GO TO 133
0136              IF(D7.LE.1.)    DP=(.5+1.5*D7)*DP
0137              IF(D7.GT.1.)    DP=(.5+1.5*(2.-D7))*DP
0138              GO TO 117
0139            133 DP=.5*DP
0140            117 IF(DP.GT.1.)    DP=1.
0141              DP1=-DP*D1
0142              DP2=-DP*D2
0143              GAIN1=DP*GAINJ
0144              D10=D1
0145              D20=D2
0146              KTIME=2
                C   INITIALIZE INFLUENCE FUNCTIONS AND THEIR INTEGRALS
0147              P13(N1)=0.
0148              IPJ1=0.
0149              IPJ2=0.
0150              IPP11=0.
0151              IPP12=0.
0152              IPP22=0.
0153              Z1D=(Z1(N2)-Z1(N2-1))/DT
0154              Z2D=(Z2(N2)-Z2(N2-1))/DT
0155              X1D=CX3(N2-1)+(CX3(N2)-CX3(N2-1))*(1.-EPSLN2)
0156              X2D=SX3(N2-1)+(SX3(N2)-SX3(N2-1))*(1.-EPSLN2)
0157              P=-UT2*UT2/(2.*(X1D-Z1D))
0158              P3(N2)=P*X2D*DT*EPSLN2
0159              P3D=-P*C
0160              P3DN=P3D*SX3(N2)
0161              P=-(X2D-Z2D)/(X1D-Z1D)
0162              P23(N2)=(P*X2D-X1D)*DT*EPSLN2
```

```
0163              P23D=-P*C
0164              P23DN=P23D*SX3(N2)+C*CX3(N2)
0165              NI=1
0166              NF=N2-N1+1
0167            121 DO 125 I=NI,NF
0168              K=N2-I
0169              J=K+1
0170              P3DO=P3DN
0171              P3DN=P3D*SX3(K)
0172              P3(K)=P3(J)+P3DN+P3DO
0173              P23DO=P23DN
0174              P23DN=P23D*SX3(K)+C*CX3(K)
0175              P23(K)=P23(J)+P23DN+P23DO
0176              IF(KTIME.EQ.2) GO TO 122
0177              P13DO=P13DN
0178              P13DN=P13D*SX3(K)+C*CX3(K)
0179              P13(K)=P13(J)+P13DN+P13DO
0180              GO TO 123
0181            122  P13(K)=0.
                C    IF CONTROL ENTERING OR LEAVING BOUNDARY, INTEGRATE ONLY
                C          OVER PART OF DT
0182            123  IF(LIMIT(K).EQ.1) GO TO 140
0183              IF(LIMIT(J).EQ.1) GO TO 142
0184              TERM1=U(K)+P3(K)
0185              IPJ1=IPJ1+TERM1*P13(K)
0186              IPJ2=IPJ2+TERM1*P23(K)
0187              IPP11=IPP11+P13(K)*P13(K)
0188              IPP12=IPP12+P13(K)*P23(K)
0189              IPP22=IPP22+P23(K)*P23(K)
0190            125  CONTINUE
0191              IF(KTIME.EQ.1) GO TO 126
0192              KTIME=1
0193              NI=N2-N1+2
0194              NF=N2-1
0195              Y1D=(Y1(N1)-Y1(N1-1))/DT
0196              Y2D=(Y2(N1)-Y2(N1-1))/DT
0197              X1D=CX3(N1-1)+(CX3(N1)-CX3(N1-1))*(1.-EPSLN1)
0198              X2D=SX3(N1-1)+(SX3(N1)-SX3(N1-1))*(1.-EPSLN1)
0199              P=-(X2D-Y2D)/(X1D-Y1D)
0200              P13(N1-1)=-(P*X2D-X1D)*DT*(1.-EPSLN1)
0201              P13D=-P*C
0202              P13DN=P13D*SX3(N1-1)+C*CX3(N1-1)
0203              GO TO 121
0204            126 TERM2=U(N2)+P3(N2)
                C    ADD CORRECTIONS FOR INTEGRALS AT BOUNDARIES
0205              IF(LIMIT(1).EQ.1) GO TO 147
0206              IF(LIMIT(N1).EQ.1) GO TO 149
0207              P13P=P13(1)
0208              P23P=P23(1)
0209            151 P13PN=P13(N1-1)
0210            150 IPJ1=(IPJ1-(TERM1*P13P+(U(N1-1)+P3(N1-1))*P13PN*EPSLN1)/2.)*DT
0211              IPJ2=(IPJ2-(TERM1*P23P+TERM2*P23(N2-1)*EPSLN2)/2.)*DT
0212              IPP11=(IPP11-(P13P*P13P+P13PN*P13PN*EPSLN1)/2.)*DT
0213              IPP12=(IPP12-(P13P*P23P+P13PN*P23(N1-1)*EPSLN1)/2.)*DT
0214              IPP22=(IPP22-(P23P*P23P+P23(N2-1)*P23(N2-1)*EPSLN2)/2.)*DT
0215            148 DEN=IPP11*IPP22-IPP12*IPP12
0216              I11=IPP22/DEN
0217              I12=-IPP12/DEN
0218              I22=IPP11/DEN
                C    COMPUTE NEW NOMINAL CONTROL
0219              DO 127 I=1,N2
0220              COEF1=P13(I)*I11+P23(I)*I12
0221              COEF2=P13(I)*I12+P23(I)*I22
0222              UTERM1=U(I)-COEF1*IPJ1-COEF2*IPJ2+P3(I)
0223              UTERM2=COEF1*DP1+COEF2*DP2
```

```
0224              DU=GAIN1*UTERM1+UTERM2
0225              U(I)=U(I)+DU
0226              IF(U(I).LT.UMAX) GO TO 130
0227              U(I)=UMAX
0228              LIMIT(I)=1
0229              GO TO 127
0230            130 IF(U(I).GT.-UMAX) GO TO 131
0231              U(I)=-UMAX
0232              LIMIT(I)=1
0233              GO TO 127
0234            131 LIMIT(I)=0
0235            127 CONTINUE
0236              GO TO 102
                C   FIND ACCURATELY POINT AT WHICH CONTROL ENTERS BOUNDARY
0237            300 A=(U(I+1)-U(I))/DT
0238              B=2.*U(I)-U(I+1)
0239              EPS=ABS((U(K)-B)/A)
0240              IF(EPS.GT.1.)   EPS=1.
0241              IF(IJK.EQ.1) GO TO 141
0242              DX3=EPS*DT*U(K)+(1.-EPS)*C*(U(K)+U(I))
0243              GO TO 302
                C   FIND ACCURATELY POINT AT WHICH CONTROL LEAVES BOUNDARY
0244            301 A=(U(K)-U(K-1))/DT
0245              B=2.*U(K)-U(K-1)
0246              EPS=ABS((U(I)-B)/A)
0247              IF(EPS.GT.1.)   EPS=1.
0248              IF(IJK.EQ.1) GO TO 143
0249              DX3=EPS*DT*U(I)+(1.-EPS)*C*(U(K)+U(I))
0250            302 X3(I)=X3(K)+DX3
0251              GO TO 106
0252            140 IF(LIMIT(J).EQ.1) GO TO 125
0253              IJK=1
0254              IO=I
0255              I=J
0256              GO TO 300
0257            141 IJK=0
0258              I=IO
0259              P3P=P3(J)+(P3(K)-P3(J))*(1.-EPS)
0260              P13P=P13(J)+(P13(K)-P13(J))*(1.-EPS)
0261              P23P=P23(J)+(P23(K)-P23(J))*(1.-EPS)
0262              TERM2=U(K)+P3P
0263            144 EPS2=EPS/2.
0264              EPS22=(1.-EPS)/2.
0265              IPJ1=IPJ1-TERM1*P13(J)*EPS2+TERM2*P13P*EPS22
0266              IPJ2=IPJ2-TERM1*P23(J)*EPS2+TERM2*P23P*EPS22
0267              IPP11=IPP11-P13(J)*P13(J)*EPS2+P13P*P13P*EPS22
0268              IPP12=IPP12-P13(J)*P23(J)*EPS2+P13P*P23P*EPS22
0269              IPP22=IPP22-P23(J)*P23(J)*EPS2+P23P*P23P*EPS22
0270              GO TO 125
0271            142 IJK=1
0272              IO=I
0273              I=J
0274              GO TO 301
0275            143 TERM1=U(K)+P13(K)
0276              I=IO
0277              P3P=P3(J)+(P3(K)-P3(J))*EPS
0278              P13P=P13(J)+(P13(K)-P13(J))*EPS
0279              P23P=P23(J)+(P23(K)-P23(J))*EPS
0280              TERM2=U(J)+P3P
0281              IJK=0
0282              GO TO 144
0283            147 TERM1=0.
0284              P13P=0.
0285              P23P=0.
0286              IF(LIMIT(N1).NE.1) GO TO 151
0287            149 P13PN=0.
0288              GO TO 150
0289              END
```

APPENDIX E

NEWTON-RAPHSON COMPUTER PROGRAM FOR TWO-TARGET INTERCEPT

This appendix contains a listing of the program used in obtaining the numerical results for the Newton-Raphson algorithm discussed in Section 3.4. A flow chart of the program is given in Fig. 3.11. The program computes either one or two target optimal intercept trajectories for straight-running targets. The input data is in the NAMELIST format under the name INPUT1. The input variables are defined at the top of the program listing. If the program is used to determine the optimal intercept for only one target, N2 is set equal to N1, and the initial conditions for target Z need not be read in.

Straight-line nominal intercept paths are computed which connect the specified values of the state and costate variables at the initial (0), intermediate (T1), and terminal (T2) times. In addition by setting INFORM equal to 2, a point-by-point solution for any of these variables can be read in, under the name INPUT2, to override the straight-line nominal. The integration step size for the first segment of the trajectory is computed by dividing the nominal intercept time, T1, by the specified number of increments, N1. For the second segment the step size is determined by dividing the difference in intercept times, T2-T1, by the remaining number of increments, N2 - N1.

The successive integrations of the linear differential equations are performed via a second-order Runge-Kutta method. At each iteration one computes the perturbations in the unknown initial and terminal conditions that are required in order to satisfy the intermediate state constraints. At each iteration the error measure, SIGMA, is computed by a rectangular integration of the difference between the nominal solution for the state variables for the present and the previous iterations. If SIGMA is less than ERROR1, the algorithm is considered to have converged to the optimal fixed-time solution. If the sum of the magnitudes of the partial derivatives of J with respect to t_1 and t_2, J1 and J2, is less than ERROR2, or if the computed perturbation in

the intercept times is less than DTMIN, the algorithm is considered to have converged to the optimal free-time solution, and the run is terminated. Otherwise the intercept times are perturbed and the current fixed-time optimal solution is used as the initial nominal for a new fixed-time problem.

```
C
C       **NEWTON-RAPHSON  ONE OR TWO TARGET OPTIMAL INTERCEPT**
C
C       X10,X20,X30=MISSILE INITIAL STATES
C       Y10,Y20,Y30,VY=TARGET Y INITIAL STATES AND VELOCITY
C       Z10,Z20,Z30,VZ=TARGET Z INITIAL STATES AND VELOCITY
C       X31,X32,X41,X42,X51,X52=NOM. X AT T0,T1+,T2
C       T1,T2,N1,N2=INTRCPT TIMES & NO. OF INTEGRATION INCRMNTS
C       ERROR1=TOLERANCE ON FIXED-TIME TRAJECTORIES
C       ERROR2=TOLERANCE ON J1 AND J2
C       DDT1,DDT2,DTMIN=INITIAL AND MIN INCREMENTS IN T1 AND T2
C       INFORM=1 IF LINEAR NOMINAL, 2 IF NOMINAL READ IN
C       KSTOP=MAX ITERATIONS FOR EACH FIXED-TIME PROBLEM
C       ITRTMX=MAX TOTAL ITERATIONS
C       SIGK1,SIGK2,SIGK3=WEIGHTING FACTORS ON X FOR SIGMA
C
0001          DIMENSION X1(1000),X2(1000),X3(1000),X6(1000),SX3(1000),
             1   CX3(1000),X1N(1000),X2N(1000),X3N(1000),X6N(1000),
             2   TRM1(1000),TRM2(1000),W1(2),W2(2),W3(2),W6(2)
0002          NAMELIST/INPUT1/ X10,X20,X30,X60,Y10,Y20,Y30,VY,Z10,Z20,
             1   Z30,VZ,X31,X32,X41,X42,X51,X52,X61,X62,T1,T2,N1,N2,
             2   KSTOP,ITRTMX,NWRITE,ERROR1,ERROR2,DDT1,DDT2,
             3   DTMIN,SIGK1,SIGK2,SIGK3,INFORM
0003          NAMELIST/INPUT2/X1,X2,X3,X6
0004          REAL J1,J2,J11,J12,J21,J22,J10,J20
0005     1    FORMAT ('1',//30X,'INPUT DATA'//)
0006     2    FORMAT ('1',//30X,'OUTPUT DATA'//)
0007     3    FORMAT('1',//30X,'RESULTS FOR ITERATION NO.',I5////25X,
             1   'T1=',E16.8,5X,'T2=',E16.8)
0008     4    FORMAT( //4X,'I',11X,'T=',15X,'X1=',15X,'X2=',15X,'X3=',
             1   15X,'P1=',15X,'P2=',15X,'P3='//)
0009     5    FORMAT (2X,I4,7(2X,E16.8))
0010     6    FORMAT(//20X,'HAVE CONVERGED TO OPTIMAL'/20X,'FOR T1=',
             1   E15.8,3X,'T2=',E15.8//20X,'INCREMENT T1 AND T2'//)
0011      7   FORMAT(20X,'DISCONTINUITIES IN P1 AND P2,DP1=',E15.8,
             1   5X,'DP2=',E15.8/)
0012      8   FORMAT(/30X,'VALUE OF COST FUNCTIONAL FOR THIS ITERATION=',
             1   E15.8/30X,'ERROR MEASURE IS,  SIGMA=',E15.8/)
0013      9   FORMAT(//30X,'******'/30X,'******'/20X,'TERMINATE RUN-',
             1   'HAVE OPTIMAL TRAJ.'/20X,'MIN CONTROL EFFORT=',E15.8,
             2   /20X,'INTERCEPT TARGET Y AT T1=',E15.8/20X,'INTERCEPT '
             3   'TARGET Z AT T2=',E15.8/30X,'******'/30X,'******'//)
0014     10   FORMAT(//20X,'TERMINATE RUN-NOT CONVERGING'//)
0015     12   FORMAT(//35X,'INTERCEPT POINTS ARE'/20X,'Y1(T1)=',E15.8,4X,
             1   'Y2(T1)=',E15.8,4X,'Z1(T2)=',E15.8,4X,'Z2(T2)=',E15.8/)
0016     13   FORMAT(///30X,'TERMINATED RUN-TRIED TO DIVIDE BY ZERO')
0017     14   FORMAT(//20X,'PARTIALS OF J WRT T1 & T2, J1=',E15.8,5X,
             1   'J2=',E16.8/)
0018     15   FORMAT(4X,I5,4(5X,E16.8))
0019     16   FORMAT(//20X,'PARTIAL OF J WRT T1, J1=',E15.8//)
0020     17   FORMAT(//20X,'J11=',E15.8,5X,'J21=',E15.8)
0021     18   FORMAT(//20X,'J12=',E15.8,5X,'J22=',E15.8)
C    READ AND WRITE INPUT DATA
0022     100  WRITE (6,1)
0023          READ (5,INPUT1)
0024          WRITE(6,INPUT1)
0025     102  WRITE (6,2)
C    INITIALIZE PARAMETERS
0026          ITERAT=0
0027          LLL=0
0028          LLLL=0
0029          A7=1.
0030          C10=X41
0031          C20=X51
0032          C30=X60
```

```
0033              C40=X32
0034              C50=X42
0035              C60=X52
0036              DT1=T1/(N1-1)
0037              DT2=(T2-T1)/(N2-N1)
                C   COMPUTE NOMINAL INTERCEPT POINTS
0038              Y1DOT=VY*COS(Y30)
0039              Y2DOT=VY*SIN(Y30)
0040              Z1DOT=VZ*COS(Z30)
0041              Z2DOT=VZ*SIN(Z30)
0042           101  Y1T1=Y10+Y1DOT*T1
0043              Y2T1=Y20+Y2DOT*T1
0044              Z1T2=Z10+Z1DOT*T2
0045              Z2T2=Z20+Z2DOT*T2
                C   COMPUTE PIECEWISE LINEAR NOMINAL TRAJECTORY
0046              DEN=N1-1
0047              DX1T1=Y1T1-X10
0048              DX2T1=Y2T1-X20
0049              DO 110 I=1,N1
0050              COEF=(I-1)/DEN
0051              X1(I)=X10+DX1T1*COEF
0052              X2(I)=X20+DX2T1*COEF
0053              X3(I)=X30+(X31-X30)*COEF
0054           110  X6(I)=X60+(X61-X60)*COEF
0055              IF(N1.EQ.N2) GO TO 112
0056              DEN=N2-N1
0057              COEF1=-(X32-X31)/(T2-T1)
0058              DX1T2=Z1T2-Y1T1
0059              DX2T2=Z2T2-Y2T1
0060              DO 111 I=N1,N2
0061              COEF=(I-N1)/DEN
0062              X1(I)=X1(N1)+DX1T2*COEF
0063              X2(I)=X2(N1)+DX2T2*COEF
0064              X3(I)=X31+(X32-X31)*COEF
0065           111  X6(I)=X61+(X62-X61)*COEF
0066           112  WRITE(6,3) ITERAT,T1,T2
0067              IF(INFORM.EQ.1) GO TO 151
0068              READ(5,INPUT2)
0069              GO TO 151
0070           120  IF(ITERAT.GT.ITRTMX) GO TO 230
0071              IF(LLLL.GT.KSTOP) GO TO 230
                C   RECOMPUTE INTERCEPT POINTS
0072              DT1=T1/(N1-1)
0073              DT2=(T2-T1)/(N2-N1)
0074              Y1T1=Y10+Y1DOT*T1
0075              Y2T1=Y20+Y2DOT*T1
0076              Z1T2=Z10+Z1DOT*T2
0077              Z2T2=Z20+Z2DOT*T2
0078           122  WRITE(6,3) ITERAT,T1,T2
0079              WRITE(6,12)Y1T1,Y2T1,Z1T2,Z2T2
                C   REINITIALIZE PARAMETERS
0080              DO 121 I=1,N2
0081              SX3(I)=SIN(X3(I))
0082              CX3(I)=COS(X3(I))
0083              TRM1(I)=SX3(I)*X3(I)
0084           121  TRM2(I)=CX3(I)*X3(I)
0085              X1N(1)=X1(1)
0086              X2N(1)=X2(1)
0087              X3N(1)=X3(1)
0088              X4N=C10
0089              X5N=C20
0090              X6N(1)=C30
0091              J=0
0092              NI=1
0093              NF=N1-1
0094              DT=DT1
0095              COEF=DT/2.
0096              X4=X41
```

```
C
0097        X5=X51
C   INTEGRATE LINEAR DIFFERENTIAL EQUATIONS
0098     135 DO 131 I=NI,NF
0099        K=I+1
0100        MK=K
0101        IK=I
0102        IF(J.GE.4)   MK=N2-I
0103        IF(J.GE.4)   IK=N2+1-I
0104        DO 130 MM=I,K
0105        M=MM
0106        KK=M-I+1
0107        IF(J.GE.4)   M=N2+1-MM
0108        W1(KK)=-SX3(M)*X3N(M)
0109        W2(KK)=CX3(M)*X3N(M)
0110        W3(KK)=-X6N(M)
0111         W6(KK)=X4*W2(KK)-X5*W1(KK)+X4N*SX3(M)-X5N*CX3(M)
0112        IF(J.GE.1.AND.J.LE.5) GO TO 136
0113        W1(KK)=W1(KK)+TRM1(M)+CX3(M)
0114        W2(KK)=W2(KK)-TRM2(M)+SX3(M)
0115        W6(KK)=W6(KK)-X4*TRM2(M)-X5*TRM1(M)
0116     136 IF(MM.EQ.K) GO TO 137
0117        X3N(MK)=X3N(M)+DT*W3(KK)
0118     130 X6N(MK)=X6N(M)+DT*W6(KK)
0119     137 X1N(MK)=X1N(IK)+COEF*(W1(1)+W1(2))
0120        X2N(MK)=X2N(IK)+COEF*(W2(1)+W2(2))
0121        X3N(MK)=X3N(IK)+COEF*(W3(1)+W3(2))
0122     131 X6N(MK)=X6N(IK)+COEF*(W6(1)+W6(2))
C   SAVE TERM. COMP. OF TRANS. MAT. AND RESET IC'S
0123        IF(J.EQ.-1) GO TO 147
0124        IF(J.EQ.-2) GO TO 148
0125        IF(J.EQ.7) GO TO 140
0126        IF(J.EQ.6) GO TO 144
0127        IF(J.EQ.5) GO TO 145
0128        IF(J.EQ.4) GO TO 146
0129        IF(J.EQ.3) GO TO 141
0130        IF(J.EQ.2) GO TO 142
0131        IF(J.EQ.1) GO TO 143
0132        B1=-X1N(N1)+Y1T1
0133        B2=-X2N(N1)+Y2T1
0134        B5=-X3N(N1)
0135        B6=-X6N(N1)
0136        J=1
0137        X1N(1)=0.
0138        X2N(1)=0.
0139        X3N(1)=0.
0140        X4N=1.
0141        X5N=0.
0142        X6N(1)=0.
0143        GO TO 135
0144     143 A1=X1N(N1)
0145        A4=X2N(N1)
0146        A13=X3N(N1)
0147        A19=X6N(N1)
0148        J=2
0149        X4N=0.
0150        X5N=1.
0151        GO TO 135
0152     142 A2=X1N(N1)
0153        A5=X2N(N1)
0154        A14=X3N(N1)
0155        A20=X6N(N1)
0156        J=3
0157        X5N=0.
0158        X6N(1)=1.
0159        GO TO 135
```

```
0160          141  A3=X1N(N1)
0161               A6=X2N(N1)
0162               A15=X3N(N1)
0163               A21=X6N(N1)
0164               IF(N1.EQ.N2) GO TO 300
0165               J=4
0166               NT=1
0167               NF=N2-N1
0168               DT=-DT2
0169               COEF=DT/2.
0170               X1N(N2)=0.
0171               X2N(N2)=0.
0172               X3N(N2)=1.
0173               X4=X42
0174               X5=X52
0175               X6N(N2)=0.
0176               GO TO 135
0177          146  A7=X1N(N1)
0178               A10=X2N(N1)
0179               A16=-X3N(N1)
0180               A22=-X6N(N1)
0181               J=5
0182               X3N(N2)=0.
0183               X4N=1.
0184               GO TO 135
0185          145  A8=X1N(N1)
0186               A11=X2N(N1)
0187               A17=-X3N(N1)
0188               A23=-X6N(N1)
0189               J=6
0190               X4N=0.
0191               X5N=1.
0192               GO TO 135
0193          144  A9=X1N(N1)
0194               A12=X2N(N1)
0195               A18=-X3N(N1)
0196               A24=-X6N(N1)
0197               J=7
0198               X1N(N2)=Z1T2
0199               X2N(N2)=Z2T2
0200               X3N(N2)=C40
0201               X4N=C50
0202               X5N=C60
0203               GO TO 135
0204          140  B3=-X1N(N1)+Y1T1
0205               B4=-X2N(N1)+Y2T1
0206               B5=B5+X3N(N1)
0207               B6=B6+X6N(N1)
0208               J=0
0209               DT=DT1
0210               COEF=DT/2.
0211          300  IF(ABS(A1).LT.1.E-20.OR.ABS(A7).LT.1.E-20) GO TO 232
               C    SOLVE ITC'S FOR UNKNOWN INITIAL AND TERMINAL B. C.
0212               COEF1=A4/A1
0213               G1=A5-A2*COEF1
0214               G2=A6-A3*COEF1
0215               G3=B2-B1*COEF1
0216                IF(ABS(G1).LT.1.E-20) GO TO 232
0217               COEF2=A2/(A1*G1)
0218               G4=B1/A1-G3*COEF2
0219               G5=-A3/A1+G2*COEF2
0220               G6=G3/G1
0221               G7=-G2/G1
0222               IF(N2.EQ.N1) GO TO 301
0223               COEF1=A10/A7
0224               R1=A11-A8*COEF1
0225               R2=A12-A9*COEF1
```

```
C
0226          R3=B4-B3*COEF1
0227          IF(ABS(R1).LT.1.E-20) GO TO 232
0228          COEF2=A8/(A7*R1)
0229          R4=B3/A7-R3*COEF2
0230          R5=-A9/A7+R2*COEF2
0231          R6=R3/R1
0232          R7=-R2/R1
0233          D1=A13*G5+A14*G7+A15
0234          D2=A16*R5+A17*R7+A18
0235          D3=A19*G5+A20*G7+A21
0236          D4=A22*R5+A23*R7+A24
0237          D5=B5-A13*G4-A14*G6-A16*R4-A17*R6
0238          D6=B6-A19*G4-A20*G6-A22*R4-A23*R6
0239          DEN=D1*D4-D2*D3
0240          IF(ABS(DEN).LT.1.E-20) GO TO 232
0241          C3=(D4*D5-D2*D6)/DEN
0242          C6=(D1*D6-D3*D5)/DEN
0243          C4=R4+R5*C6
0244          C5=R6+R7*C6
0245          C1=G4+G5*C3
0246          C2=G6+G7*C3
0247      302 C10=C10+C1
0248          C20=C20+C2
0249          C30=C30+C3
0250          C40=C40+C4
0251          C50=C50+C5
0252          C60=C60+C6
0253          X1N(1)=X1(1)
0254          X2N(1)=X2(1)
0255          X3N(1)=X3(1)
0256          X4N=C10
0257          X5N=C20
0258          X6N(1)=C30
0259          DELX4=C50-C10
0260          DELX5=C60-C20
0261          WRITE(6,7)DELX4,DELX5
0262          NI=1
0263          NF=N1-1
0264          X4=X41
0265          X5=X51
0266          J=-1
0267          GO TO 135
0268      301 D1=A21+A19*G5+A20*G7
0269          D2=-A19*G4-A20*G6+B6
0270          IF(ABS(D1).LT.1.E-20) GO TO 232
0271          C3=D2/D1
0272          C1=G4+G5*C3
0273          C2=G6+G7*C3
0274          GO TO 302
0275      147 IF(N1.EQ.N2) GO TO 148
0276          X4=X42
0277          X5=X52
0278          X4N=C50
0279          X5N=C60
0280          DT=DT2
0281          COEF=DT/2.
0282          NI=N1
0283          NF=N2-1
0284          J=-2
0285          GO TO 135
0286      148 INT=1
0287          NI=1
0288          NF=N1
C   COMPUTE COST,J, AND ERROR MEASURE, SIGMA
```

```
0289        161  COST=0.
0290             SIG1=0.
0291             SIG2=0.
0292             SIG3=0.
0293             DO 149 I=NI,NF
0294             SIG1=SIG1+ABS(X1N(I)-X1(I))
0295             SIG2=SIG2+ABS(X2N(I)-X2(I))
0296             SIG3=SIG3+ABS(X3N(I)-X3(I))
0297             X1(I)=X1N(I)
0298             X2(I)=X2N(I)
0299             X3(I)=X3N(I)
0300             X6(I)=X6N(I)
0301        149  COST=COST+X6(I)*X6(I)
0302             IF(INT.EQ.2) GO TO 160
0303             COST1=(COST-(X6(1)*X6(1)+X6(N1)*X6(N1))/2.)*DT1/2.
0304             SIGMA1=(SIGK1*SIG1+SIGK2*SIG2+SIGK3*SIG3)*DT1
0305             INT=2
0306             NI=N1+1
0307             NF=N2
0308             GO TO 161
0309        160  COST=COST1+(COST+(X6(N1)*X6(N1)-X6(N2)*X6(N2))/2.)*DT2/2.
0310             SIGMA=((SIGK1*SIG1+SIGK2*SIG2+SIGK3*SIG3)*DT2+SIGMA1)/T2
0311             WRITE(6,8)COST,SIGMA
0312             X41=C10
0313             X51=C20
0314             X42=C50
0315             X52=C60
0316        151  INT=1
0317             NI=1
0318             NF=N1
0319             DT=DT1
0320             T0=0.
0321             X4=X41
0322             X5=X51
           C   PRINT OUT RESULTS OF ITERATION
0323             WRITE (6,4)
0324         156 DO 153 I=NI,NF,NWRITE
0325             T=DT*(I-1)+T0
0326         153 WRITE(6,5)I,T,X1(I),X2(I),X3(I),X4,X5,X6(I)
0327             IF(INT.EQ.2) GO TO 152
0328             INT=2
0329             NI=N1
0330             NF=N2
0331             DT=DT2
0332             T0=(N1-1)*(DT1-DT2)
0333             X4=X42
0334             X5=X52
0335             GO TO 156
0336         152 ITERAT=ITERAT+1
0337             LLLL=LLLL+1
0338             IF(ITERAT.EQ.1) GO TO 122
0339             IF(SIGMA.GT.ERROR1) GO TO 120
0340             LLLL=1
0341             WRITE (6,6)T1,T2
0342             IF(N1.EQ.N2) GO TO 310
           C   COMPUTE PARTIAL OF J WITH RESPECT TO T1 AND T2
0343             J1=-DELX4*(COS(X3(N1))-Y1DOT)-DELX5*(SIN(X3(N1))-Y2DOT)
0344             J2=X42*(COS(X3(N2))-Z1DOT)+X52*(SIN(X3(N2))-Z2DOT)
0345             WRITE(6,14)J1,J2
0346             IF(ABS(J1)+ABS(J2).LT.ERROR2) GO TO 220
0347             IF(LLL.EQ.2) GO TO 200
0348             IF(LLL.EQ.1) GO TO 210
0349             T1=T1+DDT1
0350             J10=J1
0351             J20=J2
0352             LLL=1
0353             GO TO 120
0354         210 J11=(J1-J10)/DDT1
```

```
      C
0355        J21=(J2-J20)/DDT1
0356        WRITE(6,17) J11,J21
0357        J10=J1
0358        J20=J2
0359        T2=T2+DDT2
0360        LLL=2
0361        GO TO 120
0362    200 J12=(J1-J10)/DDT2
0363        J22=(J2-J20)/DDT2
0364        WRITE(6,18) J12,J22
      C   COMPUTE INCREMENTS IN T1 AND T2
0365        DEN=-(J11*J22-J12*J21)
0366        DD1=(J1*J22-J2*J12)/DEN
0367        DD2=(J2*J11-J1*J21)/DEN
0368        IF(ABS(DD1)+ABS(DD2).LT.DTMIN) GO TO 220
0369        T1=T1+DD1
0370        T2=T2+DD2
0371        LLL=0
0372        GO TO 120
0373    310 J1=X41*(COS(X3(N1))-Y1DOT)+X51*(SIN(X3(N1))-Y2DOT)
0374        WRITE(6,16) J1
0375        IF(ABS(J1).LT.ERROR2) GO TO 220
0376        IF(LLL.EQ.1) GO TO 315
0377        T1=T1+DDT1
0378        T2=T1
0379        J10=J1
0380        LLL=1
0381        GO TO 120
0382    315 J11=(J1-J10)/DDT1
0383        DD1=-J1/J11
0384        IF(ABS(DD1).LT.DTMIN) GO TO 220
0385        T1=T1+DD1
0386        DDT1=DD1
0387        T2=T1
0388        J10=J1
0389        GO TO 120
0390    220 WRITE (6,9)COST,T1,T2
0391        GO TO 100
0392    230 WRITE(6,10)
0393        GO TO 100
0394    232 WRITE(6,13)
0395        GO TO 100
0396        END
```

APPENDIX F

SUBOPTIMAL CONTROL — PROGRAM FOR TWO-TARGET INTERCEPT

This appendix contains a listing of the computer program used in obtaining the numerical results for the suboptimal controller developed in Section 4.4. A flow chart for the program is given in Fig. 4.6. The program can be used to compute suboptimal trajectories to intercept two targets that turn at constant rates. The input data is read in using the NAMELIST format under the name INPUT. The input variables are defined at the top of the program listing.

The nominal intercept points are computed every NUPDAT*DT seconds by assuming that the targets will continue from their present positions at constant velocities on straight-line paths and that the missile follows a two-segment, straight-line intercept trajectory as indicated in Fig. 4.5. With the nominal intercept points, the optimal control, U, for the linear model and the turning rate control, X3DOT, are computed every DT seconds. Since the missile and target turning rates are assumed to be constant over each DT interval, the missile and target angles are determined exactly by rectangular integration. The missile coordinates are determined by second-order Runge-Kutta integration, while the target coordinates are determined by rectangular integration. A target is assumed to be intercepted when the running time is within DT seconds of the predicted nominal intercept time for that target. When both targets have been intercepted, or when the running time exceeds TTSTØP seconds, the run is terminated.

```
C
C          SUBOPTIMAL GUIDANCE AGAINST TWO MANEUVERING TARGETS
C
C             INPUT & OUTPUT ANGLES ARE IN RADIANS
C       X1,X2,X3,VX ARE MISSILE COORDS., ANGLE, AND VELOCITY
C       Y1,Y2,Y3,VY ARE TARGET 'Y'      ''
C       Z1,Z2,Z3,VZ ARE TARGET 'Z'      ''
C       DT=INTEGRATION TIME INCREMENT
C       NPRINT=NO. OF INCREMENTS BETWEEN PRINTOUTS
C       NUPDAT=NO. OF INCREMENTS BETWEEN UPDATES OF NOMINAL INTERCEPT
C       RATEMX=MAX TURNING RATE OF MISSILE
C       TTSTOP=MAX RUNNING TIME
C       BETAMX=MAX ALLOWABLE ANGLE BETWEEN X3 AND BASE LINE
C       Y3DOT,Z3DOT=CONSTANT TARGET TURNING RATES
C
C
0001          NAMELIST/INPUT/X1,X2,X3,Y1,Y2,Y3,Z1,Z2,Z3,VX,VY,VZ,RATEMX,
             1   DT,TTSTOP,NPRINT,NUPDAT,BETAMX,Y3DOT,Z3DOT,JORDER
0002        1   FORMAT ('1',40X,'INPUT DATA'//)
0003        2   FORMAT (///40X,'OUTPUT DATA'//)
0004        3   FORMAT (10X,'X1=',15X,'X2=',15X,'X3=',12X,'X3DOT=',16X,'U=',11X,
             1   'JORDER JPHASE NRUN'/10X,'Y1=',15X,'Y2=',15X,'Y3=',13X,
             2   'X1N1=',13X,'X2N1=',14X,'TN1='/10X,'Z1=',15X,'Z2=',15X,
             3   'Z3=',13X,'X1N2=',13X,'X2N2=',14X,'TN2='/8X,'PS1N=',13X,
             4   'PS2N=',13X,'BASE=',13X,'ETA1=',13X,'ETA2=',16X,'H='/)
0005        4   FORMAT (5(3X,E15.7),8X,I2,2(2X,I5)/3(6(3X,E15.7)/))
0006        5   FORMAT (///10X,'*** RUN TERMINATED-TRUN.GE. ',E13.5,'***'//)
0007        6   FORMAT (///10X,'***RUN TERMINATED-BOTH TARGETS INTERCEPTED***')
0008        7   FORMAT (7/25X,'***CONDITIONS AT INTERCEPT OF TARGET Y***'//)
0009        8   FORMAT (//25X,'***CONDITIONS AT INTERCEPT OF TARGET Z***'//)
0010        9   FORMAT(7/25X,'CONTROL EFFORT=',E16.8)
C     READ IN INITIAL CONDITIONS
0011      100 READ (5,INPUT)
0012          WRITE (6,1)
0013          WRITE (6,INPUT)
0014          WRITE (6,2)
0015          WRITE (6,3)
C     INITIALIZE PARAMETERS
0016          JPHASE=1
0017          ISWTCH=0
0018          NRUN=0
0019          NTEST1=0
0020          NTEST2=0
0021          BETMX2=(BETAMX+1.570796)/2.

0022          TRUN=0.0
0023          ITERM=0
0024          COST=0.
0025          COEF=DT*VX/2.
0026          COFY=DT*VY/2.
0027          COFZ=DT*VZ/2.
C     COMPUTE NOMINAL INTERCEPT POINTS
0028      200 IF(JPHASE.EQ.2) GO TO 250
0029          II=1
0030          IF(JORDER.EQ.2) GO TO 220
0031      201 PS=Y3
0032          V=VY
0033          D1=Y1-X1
0034          D2=Y2-X2
0035      210 GAM=ATAN2(D2,D1)
0036          C=PS-GAM
0037          SINC=SIN(C)
0038          RHO=VX/V
0039          B=ARSIN(SINC/RHO)
0040          PSO=B+GAM
0041          RO=SQRT(D1*D1+D2*D2)
```

```
0042                P=PS=PSO
                C   IF P IS SMALL MUST USE DIFFERENT EQ. FOR T1 TO
                C                 AVOID ROUND-OFF ERRORS
0043                IF((ABS(P).LT.1.0E-2).OR.(ABS(PI-P).LT.1.0E-2)) GO TO 211
0044                T1=RO*SINC/(VX*SIN(P))
0045                GO TO 212
0046            211 T1=RO/(VX-V*COS(P))
0047            212 IF(JPHASE.EQ.2) GO TO 255
0048                IF(II.EQ.2) GO TO 230
0049                T11=T1
0050                PS1N=PSO
0051                II=2
0052                IF(JORDER.EQ.2) GO TO 240
0053                V=VZ
0054                S=T1*V
0055                X=Z1+S*COS(Z3)
0056                Y=Z2+S*SIN(Z3)
0057                PS=Z3
0058                GO TO 231
0059            240 V=VY
0060                S=T1*V
0061                X=Y1+S*COS(Y3)

0062                Y=Y2+S*SIN(Y3)
0063                PS=Y3
0064            231 S=T1*VX
0065                X1N1=X1+S*COS(PS1N)
0066                X2N1=X2+S*SIN(PS1N)
0067                D1=X-X1N1
0068                D2=Y-X2N1
0069                GO TO 210
0070            220 PS=Z3
0071                V=VZ
0072                D1=Z1-X1
0073                D2=Z2-X2
0074                GO TO 210
0075            230 T2=T1
0076                PS2N=PSO
0077                T1=T11
0078                S=T2*VX
0079                X1N2=X1N1+S*COS(PS2N)
0080                X2N2=X2N1+S*SIN(PS2N)
0081                TN1=TRUN+T1
0082                TN2=TN1+T2
0083                GO TO 300
0084            250 IF(JORDER.EQ.2) GO TO 201
0085                GO TO 220
0086            255 T2=T1
0087                TN2=T2+TRUN
0088                PS2N=PSO
0089                S=T2*VX
0090                X1N2=X1+S*COS(PS2N)
0091                X2N2=X2+S*SIN(PS2N)
                C   COMPUTE CONTROL FOR LINEAR MODEL
0092            300 IF(JPHASE.EQ.2) GO TO 350
0093                BASE=(PS1N+PS2N)/2.
0094                BETA=X3=BASE
0095            310 X1R=X1N1-X1
0096                X2R=X2N1-X2
0097            311 R=SQRT(X1R*X1R+X2R*X2R)
0098                IF(JPHASE.EQ.2) GO TO 351
0099                PSO=PS1N  -BASE
0100                X1R=X1N1-X1N2
0101                X2R=X2N1-X2N2
0102                R2=SQRT(X1R*X1R+X2R*X2R)
0103                H=R2*SIN(PSO)
```

```
0104                ETA1=H-R*SIN(PSO)
0105                ETA2=VX*SIN(BETA)
0106                T1=TN1-TRUN
0107                T2=TN2-TN1
0108                T12=T1*T1
0109                U=-6.*T12*(3.*T1+2.*T2)*ETA1+2.*T1*T2*(T1+T2)*ETA2=
                   1    (T1+T2)*(T1+2.*T2)*H)/(T12*T2*(3.*T1+4.*T2))
0110             320 X3DOT=U/(VX*COS(BETA))
0111                IF(ABS(BETA).GT.BETAMX) GO TO 321
0112             323 IF(X3DOT.LE.RATEMX) GO TO 322
0113                X3DOT=RATEMX
0114                GO TO 400
0115             321 IF((BETA.GT.0..AND.U.LT.0..AND.X3DOT.LT.0.).OR.
                   1    (BETA.LT.0..AND.U.GT.0..AND.X3DOT.GT.0.)) GO TO 400
0116                X3DOT=(BETMX2-BETA)/DT
0117                GO TO 323
0118             322 IF(X3DOT.GE.-RATEMX) GO TO 400
0119                X3DOT=-RATEMX
0120                GO TO 400
0121             350 BETA=X3-PS2N
0122                X1R=X1N2-X1
0123                X2R=X2N2-X2
0124                GO TO 311
0125             351 THETA=PS2N   =ATAN2(X2R,X1R)
0126                ETA1=R*SIN(THETA)
0127                ETA2=VX*SIN(BETA)
0128                T2=TN2-TRUN
0129                T22=T2*T2
0130                U=-3.*(ETA1+T2*ETA2)/T22
0131                GO TO 320
                C    INTEGRATE MISSILE DYNAMICS
0132             400 X3OLD=X3
0133                X3=X3+DT*X3DOT
0134                X1=X1+COEF*(COS(X3)+COS(X3OLD))
0135                X2=X2+COEF*(SIN(X3)+SIN(X3OLD))
                C    INTEGRATE TARGET DYNAMICS
0136                Y3OLD=Y3
0137                Y3=Y3+DT*Y3DOT
0138                Y1=Y1+COFY*(COS(Y3)+COS(Y3OLD))
0139                Y2=Y2+COFY*(SIN(Y3)+SIN(Y3OLD))
0140                Z3OLD=Z3
0141                Z3=Z3+DT*Z3DOT
0142                Z1=Z1+COFZ*(COS(Z3)+COS(Z3OLD))

0143                Z2=Z2+COFZ*(SIN(Z3)+SIN(Z3OLD))
0144                COST=COST+X3DOT*X3DOT
0145                NRUN=NRUN+1
0146                TRUN=NRUN*DT
0147                NTEST1=NTEST1+1
0148                NTEST2=NTEST2+1
                C    CHECK IF TIME TO PRINT
0149                IF(NTEST1.GE.NPRINT) GO TO 140
0150                IF(JPHASE.EQ.1) GO TO 190
0151                GO TO 120
                C    CHECK IF TIME TO UPDATE NOMINAL INTERCEPT POINTS
0152             150 IF(NTEST2.LT.NUPDAT) GO TO 300
0153                NTEST2=0
0154                GO TO 200
0155             140 WRITE (6,4) X1,X2,X3,X3DOT,U,JORDER,JPHASE,NRUN,Y1,Y2,Y3,
                   1    X1N1,X2N1,TN1,Z1,Z2,Z3,X1N2,X2N2,TN2,PSIN,PS2N,BASE,
                   2    ETA1,ETA2,H
```

```
0156                IF(ITERM.EQ.1) GO TO 100
0157                IF(ISWTCH.EQ.0)  NTEST1=0
0158                IF(JPHASE.EQ.1) GO TO 190
0159                ISWTCH=0
0160           120  IF(TRUN-TN2.LT.-DT) GO TO 160
0161                WRITE (6,6)
0162                COST=COST*DT/2.
0163                WRITE(6,9) COST
0164                ITERM=1
0165                GO TO 140
                C   CHECK IF EXCEED MAX SPEC. RUN TIME
0166           160  IF(TRUN.GE.TTSTOP) GO TO 180
0167                GO TO 150
0168           180  WRITE (6,5) TTSTOP
0169                GO TO 100
0170           130  IF(JORDER.EQ.2) GO TO 170
0171                WRITE (6,7)
0172                GO TO 140
0173           170  WRITE (6,8)
0174                GO TO 140
                C   CHECK IF TIME TO SWITCH TARGETS
0175           190  IF(TRUN-TN1.LE.-DT) GO TO 160
0176                ISWTCH=1
0177                JPHASE=2
0178                GO TO 130
0179                END
```

BIBLIOGRAPHY

1. G. A. Bliss, Lectures on the Calculus of Variations, University of Chicago Press, Chicago, 1946.

2. N. I. Akhiezer, The Calculus of Variations, Blaisdell, New York, 1962.

3. I. M. Gelfand and S. V. Fomin, Calculus of Variations, Prentice-Hall, Englewood Cliffs, New Jersey, 1963.

4. F. B. Hildebrand, Methods of Applied Mathematics, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1963.

5. M. Athans and P. L. Falb, Optimal Control: An Introduction to the Theory and Its Applications, McGraw-Hill, New York, 1966.

6. L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko, The Mathematical Theory of Optimal Processes, (transl. by D. E. Brown), The Macmillan Company, New York, 1964.

7. S. E. Dreyfus, Dynamic Programming and the Calculus of Variations, Academic Press, New York, 1965.

8. A. E. Bryson, Jr. and Y. C. Ho, Optimization, Estimation, and Control, to be published, Blaisdell Publ. Co.

9. B. Paiewonsky, "Optimal Control: A Review of Theory and Practice", AIAA Journal, Vol. 3, No. 11, November, 1965, pp. 1985-2006.

10. M. Athans, "The Status of Optimal Control Theory and Applications for Deterministic Systems", IEEE Transactions on Automatic Control, Vol. AC-11, No. 3, July 1966, pp. 580-596.

11. L. D. Berkovitz, "On Control Problems with Bounded State Variables", Journal of Mathematical Analysis and Applications, Vol. 5, 1962, pp. 488-498.

12. S. E. Dreyfus, "Variational Problems with Inequality Constraints" Journal of Mathematical Analysis and Applications, Vol. 4, 1963, pp. 297-308.

13. L. D. Berkovitz and S. E. Dreyfus, "The Equivalence of Some Necessary Conditions for Optimal Control in Problems with Bounded State Variables", Journal of Mathematical Analysis and Application, Vol. 10, 1965, pp. 275-283.

14. S. S. L. Chang, "Optimal Control in Bounded Phase Space", Automatica, Vol. 1, No. 1, January 1963, pp. 55-67.

BIBLIOGRAPHY (Cont'd.)

15. A.E. Bryson, Jr., W.F. Denham, and S.E. Dreyfus, "Optimal Programming Problems with Inequality Constraints I: Necessary Conditions for Extremal Solutions", AIAA Journal, Vol. 1, No. 11, November 1963, pp. 2544-2550.

16. J.D. Mason, W.D. Dickerson, and D.B. Smith, "A Variational Method for Optimal Staging", AIAA Journal, Vol. 3, No. 11, November 1965, pp. 2007-2012.

17. H.S. Witsenhausen, "A Class of Hybrid-State Continuous-Time Dynamic Systems", IEEE Transactions on Automatic Control, Vol. AC-11, No. 2, April 1966, pp. 161-167.

18. J.E. Rodriquez, Optimization and Control of a Cyclic Process, S.M. Thesis, Dept. of Electrical Engineering, M.I.T., June 1961.

19. G. Leitman, ed., Optimization Techniques, Academic Press, New York, 1962.

20. A.V. Balakrishnan and L.W. Neustadt, eds., Computing Methods in Optimization Problems, Academic Press, New York, 1964.

21. H.J. Kelley, "Gradient Theory of Optimal Flight Paths", Journal of the American Rocket Society, Vol. 30, October 1960, pp. 947-953.

22. J.V. Breakwell, "The Optimization of Trajectories", Journal of the Society for Industrial and Applied Mechanics, Vol. 7, No. 2, June 1959, pp. 215-247.

23. A.E. Bryson and W.F. Denham, "A Steepest-Ascent Method for Solving Optimum Programming Problems", Journal of Applied Mechanics, Vol. 29, June 1962, pp. 247-257.

24. R. Rosenbaum, "Convergence Technique for the Steepest Descent Method of Trajectory Optimization", AIAA Journal, Vol. 1, No. 7, July 1963, pp. 1703-1705.

25. C.H. Knapp and P.A. Frost, "Determination of Optimum Control and Trajectories Using the Maximum Principle in Association with a Gradient Technique", IEEE Transactions on Automatic Control, Vol. AC-10, No. 2, April 1965, pp. 189-193.

26. W.F. Denham and A.E. Bryson, "Optimal Programming Problems with Inequality Constraints II: Solution by Steepest-Ascent", AIAA Journal, Vol. 2, No. 1, January 1964, pp. 25-34.

BIBLIOGRAPHY (Cont'd.)

27. D.S. Hague, "Solution of Multiple-Arc Problems by the Steepest Method", Recent Advances in Optimization Techniques, ed. A. Lavi and T. P. Vogl, John Wiley and Sons, Inc., 1966, pp. 489-517.

28. K. Okamura, "Some Mathematical Theory of the Penalty Method for Solving Optimum Control Problems", J. SIAM on Control, Vol. 2, pp. 317-331, 1964.

29. D. L. Russell, "Penalty Functions and Bounded Phase Coordinate Control", J. SIAM on Control, Vol. 2, pp. 409-422, 1964.

30. J.V. Breakwell, J.L. Speyer, and A.E. Bryson, "Optimization and Control of Nonlinear Systems Using the Second Variation", Journal of SIAM, Series A, Vol. 1, No. 2, 1963, pp. 193-223.

31. C.W. Merriam III, "An Algorithm for the Iterative Solution of a Class of Two-Point Boundary Value Problems", Journal of SIAM, Series A, Vol. 2, No. 1, 1964, pp. 1-10.

32. S.R. McReynolds and A.E. Bryson, Jr., "A Successive Sweep Method for Solving Optimal Programming Problems", 1965 Proc. JACC, Preprints, pp. 551-555.

33. R. McGill and P. Kenneth, "Solution of Variational Problems by Means of a Generalized Newton-Raphson Operator", AIAA Journal, Vol. 2, No. 10, October 1964. pp. 1761-1766.

34. R. McGill, "Optimal Control, Inequality State Constraints, and the Generalized Newton-Raphson Algorithm", J. SIAM on Control, Series A, Vol. 3, No. 2, 1965, pp. 291-298.

35. C.H. Schley, Jr. and I. Lee, "Optimal Control Computation by the Newton-Raphson Method and the Riccati Transformation", IEEE Transactions on Automatic Control, Vol. AC-12, No. 2, April 1967, pp. 139-144.

36. P. Kenneth and G.E. Taylor, "Solution of Variational Problems with Bounded Control Variables by Means of the Generalized Newton-Raphson Method", Recent Advances in Optimization Techniques, ed. A. Lavi and T.P. Vogl, John Wiley and Sons, Inc., 1966, pp. 471-487.

37. R. McGill and P. Kenneth, "A Convergence Theorem on the Iterative Solution of Nonlinear Two-Point Boundary Value Systems", presented at the XIVth IAF Congress, Paris, France, September 1963.

38. L.V. Kantorovich and G.P. Akilov, Functional Analysis in Normed Spaces, Macmillan, New York, 1964.

BIBLIOGRAPHY (Cont'd.)

39. R. Courant, "Variational Methods for the Solution of Problems of Equilibrium and Vibrations", Bull. Am. Math. Society 49, pp. 1-23, 1943.

40. R. Courant, "Calculus of Variations and Supplementary Notes and Exercises", 1945-1946. Revised and amended by J. Moser, New York University, Institute of Mathematical Sciences, New York (mimeographed lecture notes), 1956-1957.

41. R. Kalaba, "On Nonlinear Differential Equations, the Maximum Operation, and Monotone Convergence", J. Math. Mech., Vol. 8, pp. 519-574, July 1959.

42. R.E. Bellman, H.H. Kagiwada, and R.E. Kalaba, "Quasi-linearization, Boundary Value Problems and Linear Programming', IEEE Transactions on Automatic Control, Vol. AC-10, No. 2, April 1965, p. 199.

43. R. Bellman and R. Kalaba, "Dynamic Programming, Invariant Imbedding and Quasilinearization: Comparisons and Interconnections", Proceedings of Computing Methods in Optimization Problems, Academic Press, New York, 1964, pp. 135-145.

44. R.E. Bellman and R.E. Kalaba, Quasilinearization and Nonlinear Boundary-Value Problems, American Elsevier Publishing Company, New York, 1965.

45. J.B. Plant, "An Iterative Procedure for the Computation of Optimal Controls", Ph.D. dissertation, Dept. of Electrical Engineering, M.I.T., 1965.

46. R.E. Kopp, R. McGill, R. Moyer, and G. Pinkham, "Several Trajectory Optimization Techniques", Proceedings of the Computing Methods in Optimization Problems, Academic Press, New York, 1964.

47. F.D. Faulkner, "A Comparison Between Some Methods for Computing Optimum Paths in the Problem of Bolza", Proceedings of the Computing Methods in Optimization Problems, Academic Press, New York, 1964, pp. 147-157.

48. H.S. Witsenhausen, "Some Iterative Methods Using Partial Order for Solution of Nonlinear Boundary-Value Problems", Lincoln Laboratory Technical Note 1965-18, May 14, 1965.

49. F.H. Kishi and T.S. Bettwy, "Optimal and Sub-optimal Designs of Proportional Navigation Systems", Recent Advances in Optimization Techniques, ed. A. Lavi and T.P. Vogl, John Wiley and Sons, Inc., 1966, pp. 519-540.

BIBLIOGRAPHY (Cont'd.)

50. R. Isaacs, Differential Games, Wiley, New York, 1965.

51. Y. C. Ho, A. E. Bryson, Jr., and S. Baron, "Differential Games and Optimal Pursuit-Evasion Strategies", IEEE Transactions on Automatic Control, Vol. AC-10, No. 4, October 1965, pp. 385-389.

52. M. R. Hestenes, "Numerical Methods of Obtaining Solutions of Fixed End Point Problems in the Calculus of Variations", Rand Corp., Rept. RM-102, August 1949.

53. L. V. Kantorovich, "Functional Analysis and Applied Mathematics", Dokl. Akad. Nauk. SSSR(N. S.) 59, pp. 1237-1240, 1948.

54. W. Kaplan, Advanced Calculus, Addison-Wesley, Inc. Reading, Mass., 1959.

55. S. E. Dreyfus, "Control Problems with Linear Dynamics, Quadratic Criterion, and Linear Terminal Constraints", IEEE Transactions on Automatic Control, Vol. AC-12, No. 3, June, 1967, pp. 323-324.

56. R. Fletcher and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization," The Computer Journal, Vol. 6, No. 2, July, 1963, p. 163.