NONITERATIVE SOLUTION OF THE MINIMUM REGULATOR AND MINIMUM TIME PROBLEMS

For The Degree of Ph.D.
Michigan State University

GERALD E. BERNIER
1972

This is to certify that the

thesis entitled

NONITERATIVE SOLUTION
OF THE MINIMUM REGULATOR
AND MINIMUM TIME PROBLEMS

presented by

GERALD E. BERNIER

has been accepted towards fulfillment
of the requirements for
DOCTOR OF
PHILOSOPHY degree in ELECTRICAL ENGINEERING

Robert O. Barr Jr.
Major professor

Date 7-10-73

O-7639

ABSTRACT

NONITERATIVE SOLUTION
OF THE MINIMUM REGULATOR
AND MINIMUM TIME PROBLEMS

By

Gerald E. Bernier

This thesis considers the direct, noniterative computation of the solutions of a set of optimal control problems. The specific control problem considered is that of finding how close to the origin a given initial state can be driven in a fixed run time. This problem is studied for linear systems and for nonlinear systems which are linear in the control vector. The set of optimal final states for all possible run times must also be continuous in the state space.

This locus of optimal final states is shown to be the state trajectory of another system which is called the trajectory system. The differential equation of the trajectory system is developed in the dissertation. Computational solution requires that this differential equation be solved simultaneously with a constraint condition, which is also developed.

An algorithm is presented along with two different methods of solution. One method is shown to be generally superior. It is used in a special digital program to solve some example problems. The results of an example problem solved with an analog computer are also included.

NONITERATIVE SOLUTION OF THE MINIMUM REGULATOR AND MINIMUM TIME PROBLEMS

By

Gerald E. Bernier

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering and Systems Science

1972

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

One of the branches of control theory which has received a great deal of attention in the past twenty years is optimal control. In an optimal control problem, one starts with a system and a cost functional, J. The object of the optimal control problem is to apply inputs to the system in such a way as to minimize the cost functional J. In this dissertation, the optimal control problem of primary interest is to apply bounded inputs to the system so as to drive its state from a given initial state to the closest possible point to the origin within a fixed run time. The solution is considered to be the final state. An associated problem which is also considered is to apply bounded inputs to the system in such a manner as to drive the state from a given initial state to the origin in a minimum amount of time. The solution is considered to be that minimum run time. These are referred to as Problem 1 and Problem 2, respectively.

Probably the most significant event in the area of optimal control theory was the publication of the Maximum Principle by L. S. Pontryagin and his co-workers [12]. In using the Maximum Principle, a function called the Hamiltonian which depends upon the cost functional J, the derivative of the state vector, and a costate vector is defined. If a control or input vector is optimal, i.e., minimizes the cost functional, then it minimizes the Hamiltonian over the range of permissible control vectors, and the costate vector at the end time is transversal to the target set. The Maximum Principle describes a set of necessary conditions for optimal controls, and usually these necessary conditions do not provide sufficient information to calculate the optimal controls analytically. In general, there is no method for obtaining a closed

form solution to Problem 1 or Problem 2 — or any other realistic optimal control problem — so the computation of optimal controls is a very important branch of optimal control theory.

The methods for solving optimal control problems can be separated into three broad classes. The first refers to those special cases where it is possible to solve an optimal control problem analytically. Some examples of this are given in Athans and Falb [1], but these methods are not generally very useful because of the limited class of problems to which they apply.

Secondly, consider a system which has discrete states and discrete input levels which are defined at discrete points of time. In such a system, all possible inputs could be exhaustively considered and the results compared to find an optimum. This approach fails in most realistic problems because it requires too much time. The Dynamic Programming method [4] utilizes the Principle of Optimality to eliminate most of the permissible controls without first trying them, thus greatly reducing the time necessary for the search procedure. Even though it reduces the computation time, Dynamic Programming requires so much storage space that the procedure is not often feasible.

The last group of computational methods — which are of primary significance — are generally based on the Maximum Principle or the Calculus of Variations. These methods can be further subdivided. Considering the solution of Problem 1, for an example, the primitive method is to guess at a control function, run the system using that control function, and in some way improve the control function — i.e., find one that causes the final state to be closer to the origin. This process is referred to as iterating over the control function. Such a

method was introduced by Ho [9]. It should be obvious that iteration over an entire function would not normally be a very good method because of the large storage capacity that would be required. Most other computational methods have improved this primitive approach by finding single vectors or scalars to iterate over. For instance, one might iterate over the initial or final value of the optimal costate vector to solve Problem 1. This would be sufficient because knowledge of the initial value of the costate would permit running the whole problem in order to find the optimum final state or the control function which achieves it. Ho's method was improved by Fancher [6] who iterated over the final value of the state vector rather than the whole control function. A representative group of the other methods of this general class are those due to (alphabetically) Barr [2], Bryson [5], Gilbert [7], Neustadt [11] and Stratton [13].

The various methods in this third group are all iterative methods. This means that an initial guess of some quantity representative of the problem solution is made; then that and successive values of the given quantity are evaluated and improved until an optimum solution is obtained. Computational methods are called direct or indirect as well as iterative or noniterative. A direct method is one which is primarily concerned with the variable or variables which are considered to be the problem solution. This may not always be a meaningful designation. As an example, Ho's method is primarily concerned with the control function so it is an indirect method for solving Problem 1. But if the problem solution were defined to be the optimal control function, Ho's method would be termed direct. Normally, however, practical considerations force a specific quantity to be designated the "solution"

of the problem. Fancher's method is a direct, iterative technique for solving Problem 1.

This thesis is the result of research that was motivated by a dual consideration:

1) Iterative methods are often undesirable because of convergence problems and possible long computation times; and

2) A method that could be used on an analog computer alone could prove to be advantageous.

These are termed a dual consideration because iterative methods cannot be used with an analog computer alone — there must be a digital or a human interface. A computational procedure must also be direct if the problem solution is to be presented on an analog computer. So the research was aimed at developing a direct, noniterative computation procedure for solving Problem 1. Such a procedure introduces a different philosophy to the solution of optimal control problems; it is hoped that this philosophy will eventually lead to better methods of solving a wide range of optimal control problems. The solution method presented in this thesis has practical significance. Given an arbitrary initial state, there are an infinite number of meaningful cases of Problem 1 and there is the additional case of Problem 2. The procedure presented in the sequel makes it possible to solve all of these problems with a computation time which is of the same order of magnitude as that which is required by other methods to solve just one case of Problem 1.

After definitions, terminology and problem statements, some Lemmas are presented in Chapter 2. The computation procedure is developed for time invariant linear systems, time varying linear systems, and nonlinear systems in Chapter 3, Chapter 4, and Chapter 5 respectively. Then the results are summarized in Chapter 6.

2. PRELIMINARY CONSIDERATIONS

The purpose of this chapter is to define the problem which is solved in succeeding chapters and to present the background material which is pertinent to solving that problem. In preparation for this objective, section 2.1 is a presentation of the notation and terminology to be used throughout. The systems to be studied are defined in sections 2.2 and 2.3, and the control problem in section 2.4. The results of the Maximum Principle, as it pertains to the control problem at hand, are reviewed in section 2.5, and the normality condition is attached to the control problem in section 2.6. The optimal control vector and its switch times are discussed in section 2.7. Finally, two Lemmas which are needed in later chapters are proved in section 2.8.

2.1 NOTATION AND TERMINOLOGY

All vectors are column vectors or n x 1 matrices — where n is the dimension of the vector — and are denoted by underlined small-case Roman letters. Capital Roman letters are used to designate matrices, the dimension of which is specified unless clear from the context.

Superscripts are used to denote the components of a vector or a matrix: a scalar component of a vector is denoted by a single superscript (x^i is a component of $\underline{x}$), a scalar component of a matrix is designated by a double superscript (b^{ij} is a component of B), and a column of a matrix — being a vector — is denoted by an underlined small-case Roman letter with a single superscript ($\underline{b}^j$ is the j^{th} column or vector component of B). This allows the use of subscripts to distinguish between members of a set, such as a sequence. The transpose of a vector or a matrix is denoted by a superscript T. A superscript

asterisk, $\underline{x}^*$, is used to denote an optimum vector — a local optimum or a global optimum — where optimum is defined according to the particular control problem at hand.

The scalar, t, is used to denote time. If a particular vector, say $\underline{x}$, is a function of time, $\underline{x}(t)$ is its value at time t, and $\underline{x}(\cdot)$ is used to designate the whole function. The symbology $\underline{x}(\cdot)$ is associated with a specified time interval over which the function is defined. Differentiation of a variable with respect to time is normally written as that variable with a dot over it:

$$\dot{\underline{x}}(t) = \frac{d\underline{x}(t)}{dt} \quad .$$

The notation $\|\underline{x}\|$ is used for the Euclidean 2-norm of the vector

$$\|\underline{x}\| = \left[\sum_{i=1}^{n} \left(x^i\right)^2\right]^{1/2} \quad .$$

The inner product of two vectors is

$$\langle \underline{x}, \underline{y} \rangle = \sum_{i=1}^{n} x^i y^i \quad .$$

The scalar signum function, sgn (a), is defined by

$$\text{sgn (a)} = \begin{cases} -1, & a < 0 \\ 0, & a = 0 \\ 1, & a > 0 \end{cases}$$

and the vector signum function, $\underline{\text{sgn}}\ (\underline{x})$, is defined by

$$\underline{\text{sgn}}\ (\underline{x}) = \begin{bmatrix} \text{sgn}(x^1) \\ \vdots \\ \text{sgn}(x^n) \end{bmatrix}.$$

2.2 SYSTEM DEFINITION

Because of the nature of the material to be presented in this thesis, it is necessary to present that material as it applies to each of three different systems. Each of these three systems is a special case of the state equation.

$$\dot{\underline{x}} = \underline{f}(\underline{x},\underline{u}) \tag{2.2.1}$$

where $\frac{\partial \underline{f}}{\partial \underline{x}}$ and $\frac{\partial \underline{f}}{\partial \underline{u}}$ exist, and the m-dimensional control vector, $\underline{u}(\cdot)$, lies in the set

$$\Omega = \left\{ \underline{u} \,|\, u^i \leq 1,\ i = 1, \ldots, m \right\} \tag{2.2.2}$$

for all time. An admissible control is a measurable function whose range is the set Ω. The system state at $t = 0$ is designated $\underline{x}_0$.

The most general system to be discussed is described by the nonlinear state equation

$$\dot{\underline{x}} = \underline{f}(\underline{x}) + B(t)\underline{u} \tag{2.2.3}$$

where $\frac{\partial \underline{f}}{\partial \underline{x}}$ exists and $\underline{u}$ is admissible. A special case of this is the time-varying linear system given by

$$\dot{\underline{x}} = A(t)\underline{x} + B(t)\underline{u} \tag{2.2.4}$$

where $\underline{u}$ is admissible. Finally, the first system to be discussed is

the time-invariant linear system given by

$$\dot{\underline{x}} = A\underline{x} + B\underline{u} \qquad (2.2.5)$$

where $\underline{u}$ is admissible.

The material to be presented on optimal computing procedures is much more specific for linear systems and thus each system (2.2.5), (2.2.4), (2.2.3) is treated separately in Chapters 3, 4, 5 in increasing order of complexity.

2.3 THE REACHABLE SET AND CONTROLLABILITY

Given a system S, its initial state $\underline{x}_0$, a set Ω, and a run time T, the set of all possible final states is called the reachable set for $\underline{x}_0$ and T and is designated $R(\underline{x}_0,T)$ or just $R(T)$. Such a (S, Ω) is called completely controllable if for any initial state $\underline{x}_0$ and any final state $\underline{x}_f$, $\underline{x}_f \epsilon R(\underline{x}_0,T)$ for some finite $T \geq 0$. (S,Ω) is called completely null controllable if for any $\underline{x}_0$, $\underline{0} \epsilon R(\underline{x}_0,T)$ for some finite $T \geq 0$. Finally, for (S,Ω), the initial state $\underline{x}_0$ is said to be in the region of null controllability if $\underline{0} \epsilon R(\underline{x}_0,T)$ for some finite $T \geq 0$. In the rest of this dissertation, and unless otherwise specified, it is assumed that $\underline{x}_0$ is in the region of null controllability.

2.4 THE CONTROL PROBLEM

The general control problem to be considered in this dissertation is sometimes called the optimal regulator problem. The problem is to transfer the system state from $\underline{x}_0$ at t=0 to the point closest to the origin in the fixed run time $T \geq 0$ while using an admissible control. This final state is called $\underline{x}^*(T)$ and the control vector used to attain it is called $\underline{u}^*(\cdot)$. Such a point is called an optimal final state

and must be a member of the reachable set. Let T^* be the shortest run time required to reach the origin. Then as long as $T \leq T^*$, an optimal final state must lie on the boundary of the reachable set. This final state is called a local optimum if it has a neighborhood which contains no points in the reachable set which are closer to the origin; it is called a global optimum if no point of the reachable set lies closer to the origin — thus, a global optimum is also a local optimum.

Determination of $\underline{x}^*(T)$ is considered the primary problem and this thesis is directed toward that end. The dissertation focuses on determining $\underline{x}^*(T)$ for a variety of T's. For simplicity this is designated Problem 1. The related problem of finding $\underline{u}^*(\cdot)$ is shown to be solvable immediately in terms of $\underline{x}^*(T)$. The time optimal regulator problem, called Problem 2, is the problem of determining T^* — the minimum run time for which $\underline{x}^*(T) = \underline{0}$.

Now there is an infinite set of control problems in category 1, each one designated by its particular run time T. The solution of a particular Problem 1 is then labelled $\underline{x}_T^*(T)$ where the subscript denotes an element from the set of run times. If the set $\left\{\underline{x}_T^*(T)\right\}$ is known for all T, the solution T^* of Problem 2 can readily be found. The method to be presented for solving Problem 1 generates $\underline{x}_T^*(T)$ for all T, $0 \leq T \leq T^*$. For $T < T^*$, the solution of the control problem can be used to find $\underline{u}^*(T,\cdot)$, the optimal control vector for run time T. This is shown after the presentation of the Maximum Principle.

2.5 THE MAXIMUM PRINCIPLE

It is assumed that the reader is familiar with the Maximum Principle, at least to the extent which it is covered in an introductory text such as Athans and Falb [1] or Lee and Marcus [10]. This section is presented only for the purpose of applying it to the problem already defined.

For the control problem as stated and the system defined in (2.2.1), the cost functional is taken to be

$$J(\underline{u}) = 1/2 \, \|\underline{x}(T)\|^2 \quad . \tag{2.5.1}$$

This can be rewritten as follows:

$$J(\underline{u}) = 1/2 \, \|\underline{x}(0)\|^2 + 1/2 \int_0^T \frac{d}{dt} \|\underline{x}(t)\|^2 \, dt$$

$$= 1/2 \, \|\underline{x}(0)\|^2 + \int_0^T \dot{\underline{x}}^T(t) \, \underline{x}(t) \, dt \quad . \tag{2.5.2}$$

An equally good cost functional and the one which is used in the sequel is

$$J_1(\underline{u}) = \int_0^T \dot{\underline{x}}^T(t) \, \underline{x}(t) \, dt \quad . \tag{2.5.3}$$

since $1/2 \, \|\underline{x}_0\|^2$ is a constant, the solution obtained using $J_1(\underline{u})$ is the same as that obtained using $J(\underline{u})$, so the use of J_1 is, indeed, equivalent.

The Hamiltonian is formed as

$$H = \langle \underline{p} + \underline{x}, \dot{\underline{x}} \rangle \, , \tag{2.5.4}$$

where

$$\dot{\underline{p}}(t) = -\frac{\partial H}{\partial \underline{x}} = -\left[\dot{\underline{x}} + A^T(\underline{p} + \underline{x})\right] \quad . \tag{2.5.5}$$

In (2.5.5) A is $A(t) = \frac{\partial \underline{f}}{\partial \underline{x}}$ — this becomes A(t) or A in linear systems. An adjoint vector is defined by

$$\underline{y} = \underline{p} + \underline{x} \ , \tag{2.5.6}$$

so

$$\dot{\underline{y}} = \dot{\underline{p}} + \dot{\underline{x}} = -A^T\underline{y} \quad . \tag{2.5.7}$$

This allows the Hamiltonian to be rewritten as

$$H(\underline{x},\underline{y},\underline{u}) = \langle \underline{y},\dot{\underline{x}} \rangle \tag{2.5.8}$$

where the dependence of H on the control vector $\underline{u}$ is through $\dot{\underline{x}}$. Since the target set is all of R, the end-point transversality condition states that (Athans and Falb, page 288)

$$\underline{p}^*_T(T) = 0 \ , \tag{2.5.9}$$

so

$$\underline{y}^*_T(T) = \underline{x}^*_T(T) \quad . \tag{2.5.10}$$

The adjoint system is defined by (2.5.7) and (2.5.10).

In order for $\underline{u}^*$ to be an optimal control vector, it must minimize the Hamiltonian according to

$$H(\underline{x}^*,\underline{y}^*,\underline{u}^*) \leq H(\underline{x}^*,\underline{y}^*,\underline{u}) \tag{2.5.11}$$

for all admissible $\underline{u}$. If the system can be described by (2.2.3), this can be combined with (2.5.8) to yield

$$\underline{u}^*(T,t) = \underline{sgn}\left[-B^T(t)\ \underline{y}_T^*(t)\right] \qquad (2.5.12)$$

for $t\varepsilon[0,T]$.

2.6 NORMALITY AND THE OPTIMAL CONTROL VECTOR

The control problem defined above is said to be normal if each element of $B^T(t)\ \underline{y}_T^*(t)$ is zero only at a countable number of times in the interval [0,T]. This is a necessary requirement if (2.5.12) is to be a unique representation of the optimal control almost everywhere throughout that time interval. A control problem which does not meet the normality condition is said to be singular. Hereafter, all control problems of form (2.2.3) which are encountered are assumed to be normal. In a normal control problem, each optimal control u^i is either +1 or -1 almost everywhere. This is referred to as "bang-bang" control.

The symbol $\underline{u}^*(T,\cdot)$ is used to denote the control vector of a local optimum. A nonlinear or a singular linear problem may have more than one $\underline{u}^*(T,\cdot)$; however, in the case of a normal linear system, $\underline{u}^*(T,\cdot)$ is unique and no ambiguity exists (see Athans and Falb page 404). For the normal control problems considered here, each component of $\underline{u}^*(T,\cdot)$ is a bang-bang function.

The time t when the bang-bang function $u_i^*(T,t)$ changes from +1 to -1 or from -1 to +1 is called a <u>switch time</u>. One of these switch times occurs whenever an element of $B^T(t)\ \underline{y}_T^*(t)$ is zero. So the switch times are defined by the equation

$$B^T(t)\ \underline{y}_T^*(t) = 0 \qquad (2.6.1)$$

which is the equation of a moving hyperplane in the n-dimensional costate space. This hyperplane is called the switch hyperplane or just the switch plane. Some element of $\underline{u}^*(T,\cdot)$ switches when $\underline{y}_T^*(\cdot)$ intersects the switch plane. For some systems, there is a maximum number of switches that can occur as stated in the following theorem (see Athans and Falb, pages 402 and 403).

Theorem 2.6.1. Consider the normal control problem for the linear time-invariant system

$$\dot{\underline{x}} = A\underline{x} + B\underline{u} \quad , \tag{2.6.2}$$

where the eigenvalues of A, $\lambda_1, \lambda_2, \ldots, \lambda_n$, are all real. Let $u^j*(T,\cdot)$ be a component of the optimal control vector (if it exists). Let t_i be the switch times t of the bang-bang function $u^j*(T,t)$. Then the maximum value of the number i is (n-1). In other words, each on-off control $u^j*(T,\cdot)$ can switch at most (n-1) times.

2.7 DEPENDENCE OF $\underline{u}^*(T,t)$ ON T

In the previous sections, $\underline{u}^*(T,t)$ has been discussed as a function of t; it is also necessary to investigate $\underline{u}^*(T,t)$ as a function of T. The double-integrator plant, $\ddot{x} = u$, which is discussed in Appendix I, is referred to extensively in the following material to illustrate the concepts.

Attention is first focussed on a second-order normal linear system with a one-dimensional control and an A matrix having real eigenvalues — the specific example being $\ddot{x} = u$. This means that only one switch can occur.

To understand why it is necessary to examine the effect of run time, consider a system such that for T = 1 second the optimal final state is reached by letting u = -1 throughout the one second run. Consider further that for T = 1.01 seconds the optimal final state is reached if u = -1 for $0 \leq t < 1.009$ and u = +1 for the rest of T. An equally likely alternative is that u = +1 for $0 \leq t < 0.001$ and u = -1 for the rest of T. The interesting aspects of this situation are:

1) it is possible that for a sufficiently short run time T, no control switching occurs, and

2) when T is large enough so that a switch occurs, that switch could be either near the beginning or near the end of the run.

Consider the double-integrator plant with $\underline{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. If u = -1 is applied for one second, the final state is $\begin{bmatrix} 1.5 \\ 0 \end{bmatrix}$ and this is the optimal final state for T = 1 second. Also for any run time less than one second, the optimal final state is reached by applying u = -1 with no switching. Point 1 above has now been verified by an example. Now consider that T is greater than one second, say T = 1.2 seconds. In this case the optimal final state is reached if u = -1 from zero to something between 1.0 and 1.2 seconds, then u = +1 for the remainder of the 1.2 seconds. This example and several other cases of T > 1 are shown in Figure 2.7.1. The points on the dashed line are optimal final states for this particular $\underline{x}_0$. This is an example of the switch occuring near the end of the run time and is called Type I switching.

Taking again the double integrator plant, let $\underline{x}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. Now for T small enough, the optimal final state is reached by letting u = +1 with no switching. Let T_0 be the maximum value of T for which the optimal u does not switch and note that in this case it is somewhat less than 0.5 seconds (see Figure 2.7.2). Also, it should be

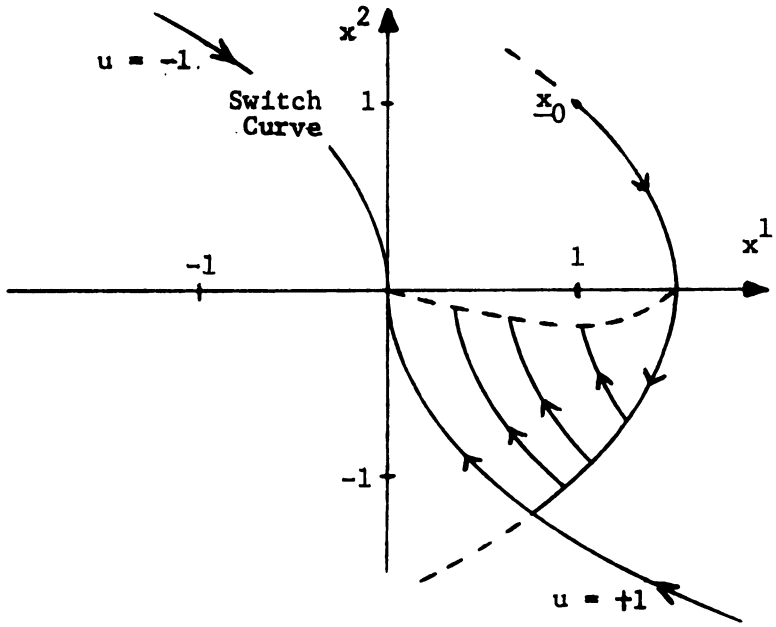

Figure 2.7.1. The Double Integrator Plant with $\underline{x}_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

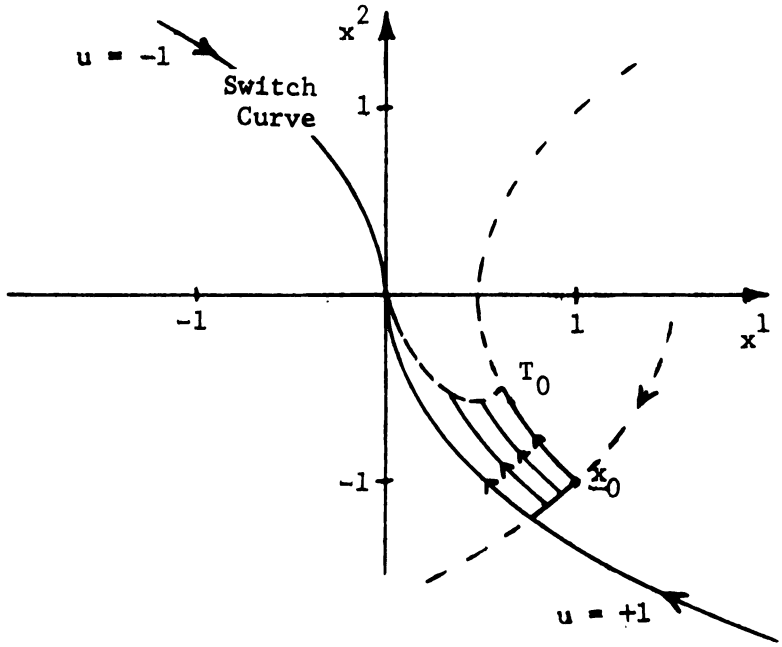

Figure 2.7.2. The Double Integrator Plant with $\underline{x}_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

observed that a precise value for T_0 is not obvious in this case as it was in the previous one. For $T > T_0$, the optimal final state is reached by letting $u = -1$ for a time and then switching to $u = +1$ for the rest of the run time. By taking T slightly greater than T_0 note that this is an example of the switch occuring near the beginning of the run time. This is called Type II switching.

Two special cases are now evident. The first is for $\underline{x}_0$ on the x^1 axis. In this case, the optimal control must be switched for any $T > 0$. For completeness, it is arbitrarily stated that a switch exists for $T = 0$ and $T_0 = 0$. The other special case occurs when $\underline{x}_0$ is an element of the switch curve. In that case, no switch is necessary because $\underline{x}_0$ lies on a trajectory through the origin and this trajectory can be followed for the whole run time. For completeness, it is arbitrarily stated that $T_0 = T^*$.

Whether Type I or Type II switching occurs for $T > T_0$ in the double integrator plant depends only on the location of $\underline{x}_0$ as shown in Figure 2.7.3. The marginal or transition cases are also shown and labeled according to the assumed values of T_0.

Consider a more general case such that for an arbitrary fixed $T \varepsilon [T_0, T^*]$ one and only one switch occurs in the function $u^*(T,\cdot)$. Let the time t when this switch occurs be denoted by $\tau = \tau(T)$. The notation suggests that the actual switch time τ depends on the specific run time T. When $T = T_0$, the definition of T_0 implies that the switch must occur at one end of the interval $[0,T_0]$; thus either $\tau(T_0) = T_0$ or $\tau(T_0) = 0$.

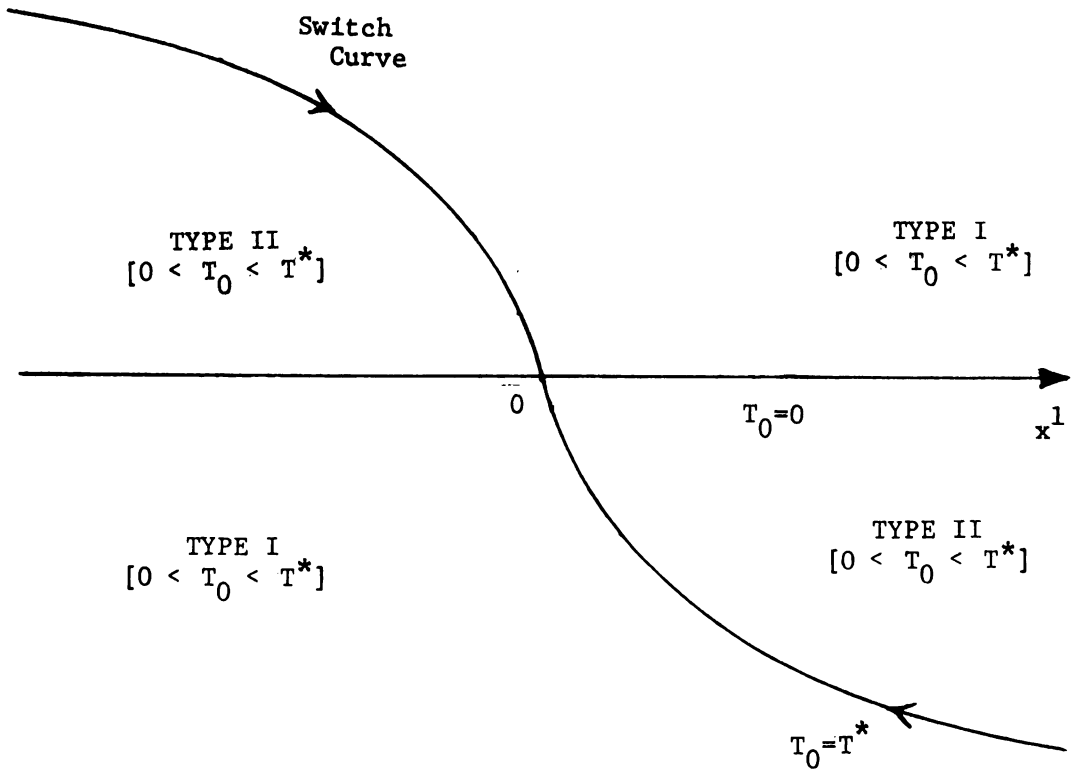

Figure 2.7.3. Switching Types According to the Location of $\underline{x}_0$ in the Double Integrator Plant

The above assertions can be shown by noting that $u^*(T,\cdot)$ switches when $\underline{y}_T^*(\cdot)$ crosses the switch plane. For $T < T_0$, $\underline{y}_T^*(\cdot)$ does not cross the switch plane, hence no switch occurs. The trajectory $\underline{y}_T^*(\cdot)$ is "attached" at one end to the set of optimal final states $\left\{\underline{x}_T^*(T) \,|\, 0 \leq T \leq T^*\right\}$, as shown by (2.5.10). So as T increases $\underline{y}_T^*(\cdot)$ can be pictured as a curve which is defined over longer time intervals moving through space — this movement specified by the position of its endpoint $\underline{y}_T^*(T)$. A typical example is shown in Figure 2.7.4 where the state and costate spaces are superimposed so $\underline{x}_T^*(T)$ and $\underline{y}_T^*(\cdot)$ can be shown simultaneously and where the switch hyperplane is assumed stationary for simplicity. Assuming the set of optimal final states is continuous — this assumption is discussed later — and knowing that $\underline{y}_{T_0}^*(\cdot)$ does not intersect

the switch plane, $\underline{y}^*_{T_0}(\cdot)$ must indeed intersect the switch plane at an endpoint. For the example shown in Figure 2.7.4 $\tau(T_0) = T_0$, but this is not necessarily the case as shown in Figure 2.7.5 where $\tau(T_0) = 0$.

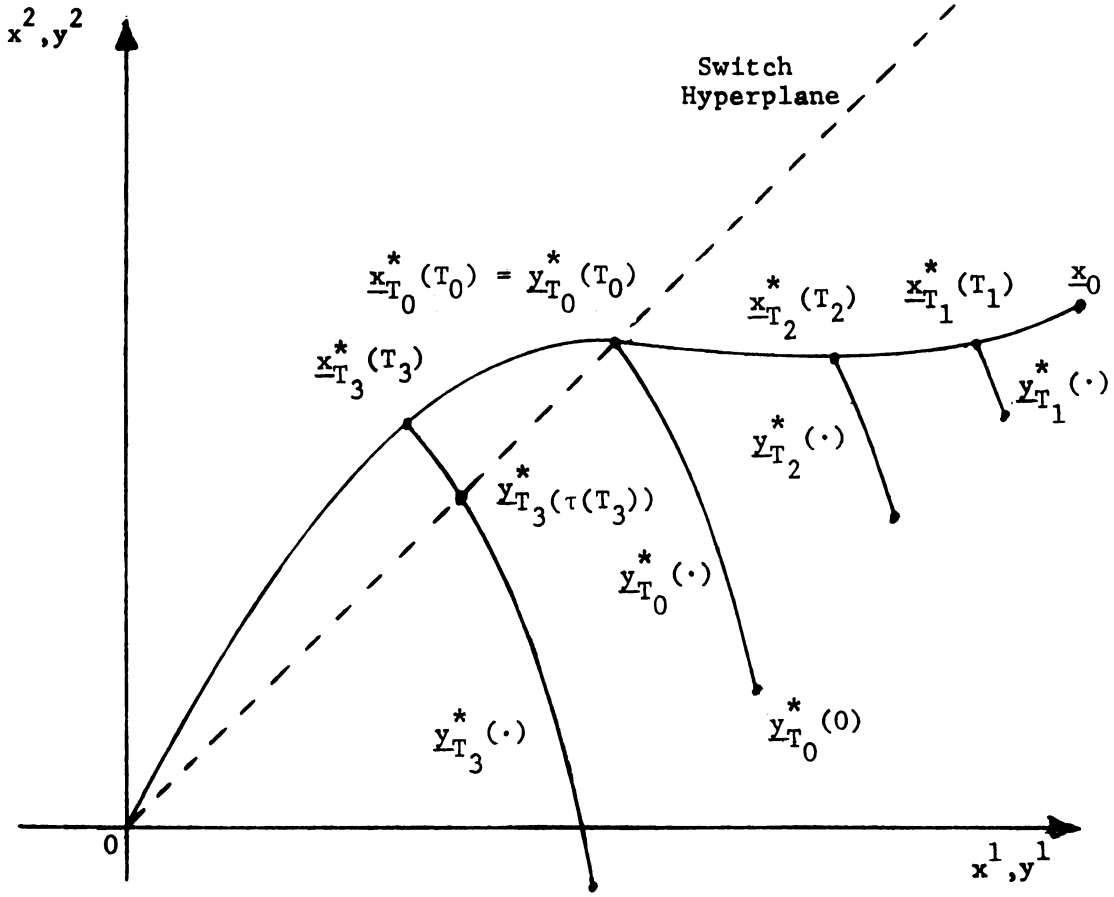

Figure 2.7.4. The Set of Costate Trajectories

As already stated, the situation depicted in Figure 2.7.4 where $\tau(T_0) = T_0$ is referred to as a Type I switch, and the other situation where $\tau(T_0) = 0$ is referred to as a Type II switch. In either case $\tau(T)$ is not defined for $T < T_0$, so a value is arbitrarily assigned: let $\tau(T) = \tau(T_0)$ for $0 \leq T < T_0$; note that for T in this range no switch occurs. This arbitrary assignment makes $\tau(T)$ continuous at T_0, and in no way affects the later results, because in neither case does

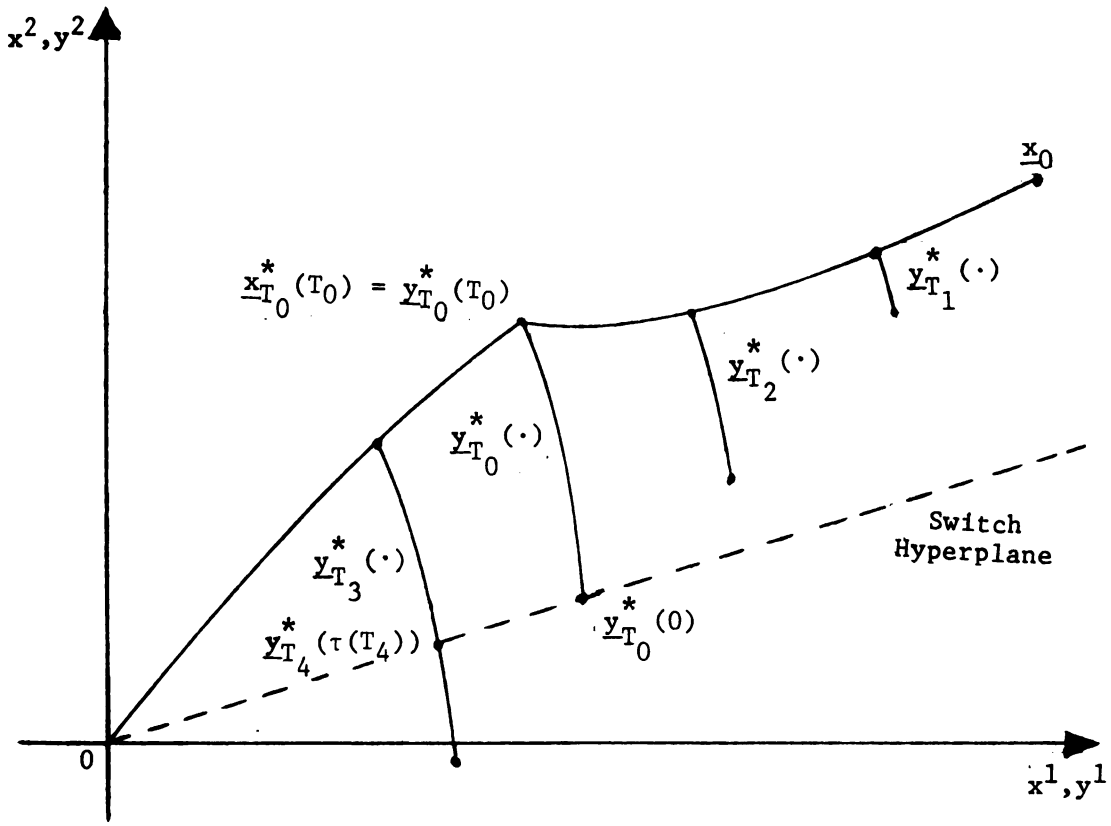

Figure 2.7.5. The Alternate Situation

$\tau(T)$ fall in the interval $(0,T)$ until $T \geq T_0$. Sketches of typical τ versus T plots are shown in Figures 2.7.6 and 2.7.7. In each case for $T_0 \leq T \leq T^*$, $\tau(T)$ must lie in the shaded trapezoidal region.

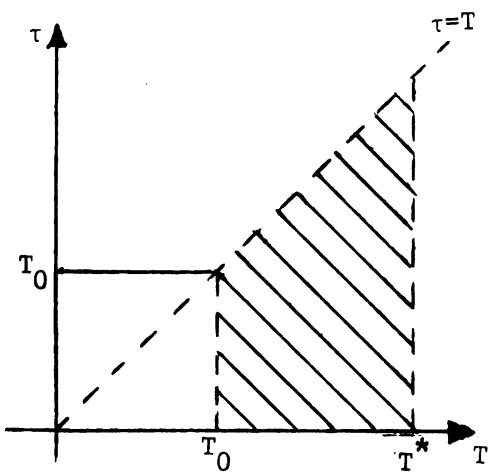

Figure 2.7.6. τ Versus T For a Type I Switch

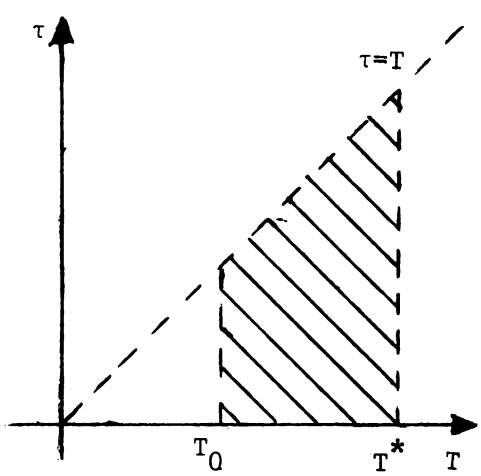

Figure 2.7.7. τ Versus T For a Type II Switch

Theorem 2.7.1. For a normal control problem such that the set of optimal final states $\left\{ \underline{x}_T^*(T) \,|\, 0 \leq T \leq T^* \right\}$ forms a continuous curve in state space, $\tau(\cdot)$ is a continuous function in any sub-interval of $[0,T^*]$ for which a switch occurs.

Proof: From (2.5.10) $\underline{y}_T^*(T) = \underline{x}_T^*(T)$, thus the set of optimal final costates also forms a continuous curve in costate space. From (2.5.7), the adjoint equation is a linear differential equation. The continuity of $\left\{ \underline{y}_T^*(T) \,|\, 0 \leq T \leq T^* \right\}$ means that for each $T_1 \varepsilon (0,T^*)$ and for each $\varepsilon > 0$, there is a $\delta > 0$ such that

$$\left\| \underline{y}_{T_1}^*(T_1) - \underline{y}_T^*(T) \right\| < \varepsilon \tag{2.7.1}$$

whenever $|T - T_1| < \delta$. The switch time $\tau(T)$ is given by the equation

$$\underline{b}^T \underline{y}_T^*(\tau(T)) = 0 \tag{2.7.2}$$

This means that if the costate equation is run in reverse time over an interval of time whose length is $(T - \tau(T))$ from an initial value equal to $\underline{y}_T^*(T)$, the costate at the end of that time interval will be an element of the switch hyperplane given by

$$\langle \underline{b}, \underline{y} \rangle = 0 \tag{2.7.3}$$

Since the costate equation is linear, trajectories with "nearly equal" initial conditions will lie "close" to each other. More precisely since $\underline{y}_T^*(T_1)$ lies in an ε-neighborhood of $\underline{y}_T^*(T)$ when the initial time T_1 lies in a δ-neighborhood of T, for each $\gamma > 0$ there exists an $\varepsilon > 0$ and thus a $\delta > 0$ such that

$$\left\| \underline{y}_{T_1}^*\Big(\tau(T) + |T - T_1|\Big) - \underline{y}_T^*\Big(\tau(T)\Big) \right\| < \gamma \tag{2.7.4}$$

whenever $|T-T_1| < \delta$. Therefore $\underline{y}^*_T(\tau(T) + |T-T_1|)$ lies in a γ-neighborhood of an element of the switch hyperplane. Furthermore, $\underline{y}^*_{T_1}(\tau(T_1))$ is an element of the switch hyperplane by definition. So it must also be true that for each $\omega > 0$ there exists a $\gamma > 0$ and thus a $\delta > 0$ such that

$$\left|\left(\tau(T) + |T - T_1|\right) - \tau(T_1)\right| < \omega \qquad (2.7.5)$$

whenever $|T - T_1| < \delta$. This means that

$$|\tau(T) - \tau(T_1)| < \omega + |T - T_1| = \rho \qquad (2.7.6)$$

To summarize for each $\rho > 0$ there must exist an $\omega > 0$ and thus a $\delta > 0$ such that $|\tau(T) - \tau(T_1)| < \rho$ whenever $|T - T_1| < \delta$. This has been shown for an arbitrary T_1 such that $\underline{y}^*_{T_1}(\cdot)$ intersects the switch hyperplane, so it must be true for all such run times.

Q.E.D.

Since it was assumed in the proof that $\underline{y}^*_{T_1}(\cdot)$ crossed the switch hyperplane, this proof shows that $\tau(T)$ is continuous on those intervals of T for which switching occurs. This means that it is possible that for short run times no control switching occurs, for longer run times switching does occur, and for yet longer run times switching does not occur again. But, over the interval for which switching does occur, $\tau(T)$ must be continuous. Also, if control switching occurs for the run time $T_1 < T^*$ but not for T_1+, then $\tau(T_1)$ must be either 0 or T_1. This follows from the fact that costate trajectories which start close together must remain close together throughout.

Even with the condition of continuity required in Theorem 2.7.1, it covers a wide range of problems; namely, all linear systems [3] as well as many nonlinear systems. In particular, consider a nonlinear system with the global optima such that $\left\{\underline{x}_T^*(T) \,|\, 0 \leq T \leq T^*\right\}$ is discontinuous. Even in this system it is quite possible that there are (possibly several) local optima each of which presents a continuous set of "optimal" final states. This point is discussed further in the section on nonlinear systems.

The following Lemma, inserted without proof, is a special case of Theorem 7.1 found on page 22 of Hestenes [8] where a proof is offered.

Lemma 2.7.2 Implicit Function Theorem

Given a function $f(T,\tau)$ which is continuous and has a continuous derivative with respect to τ on an open set S in T, τ-space and such that

$$f(T,\tau) = 0,$$

suppose

$$f(T_0,\tau_0) = 0, \qquad \frac{\partial f}{\partial \tau}(T_0,\tau_0) \neq 0$$

holds at a point (T_0,τ_0) in S. Then there are a continuous function $\tau(T)$ on a neighborhood T' of T_0 and a constant $\varepsilon > 0$ such that

$$\tau(T_0) = \tau_0, \qquad f(T,\tau(T)) = 0$$

and such that the relations

$$f(T,\tau) = 0, \qquad |\tau - \tau(T)| < \varepsilon \qquad (T \text{ in } T')$$

hold only in case $\tau = \tau(T)$. If the function f is of class $C^{(m)}$ on S, the function $\tau(T)$ is of class $C^{(m)}$ on T'.

This Lemma could have been used to prove Theorem 2.7.1, but the approach used there was intended to be more instructive. In any event, the Lemma is needed in the following.

Theorem 2.7.3. For any subinterval of $[0,T^*]$ over which $\left\{ \underline{x}_T^*(T) \,|\, 0 \leq t \leq T^* \right\}$ is differentiable and switching occurs, $\tau(T)$ is also differentiable.

Proof: In Lemma 2.7.2 let f be defined as

$$f(T,\tau) = \langle \underline{b}, \underline{y}_T^*(\tau) \rangle = b_1 y_T^{1*}(\tau) + \cdots + b_n y_T^{n*}(\tau) \quad .$$

Because $\left\{ \underline{x}_T^*(T) \right\}$ is differentiable, $\left\{ \underline{y}_T^*(T) \right\}$ is also. Therefore, when τ represents the switch time, $f(T,\tau) = 0$ and $\frac{\partial f}{\partial \tau}$ is continuous where S is all of T,τ. Also, as long as τ is defined — i.e., as long as a switch occurs — there is a set (T_0,τ_0) where

$$f(T_0,\tau_0) = 0, \qquad \frac{\partial f}{\partial \tau}(T_0,\tau_0) \neq 0 \quad .$$

Then, from the Lemma, there is a neighborhood T' about T_0 on which $\tau(T)$ is differentiable. Since there is such a T_0,τ_0 pair for each time T in the subinterval of interest, $\tau(T)$ is differentiable throughout that interval.

Q.E.D.

2.8 PRELIMINARY MATHEMATICS

The purpose of this section is to present a few lemmas which are needed in later proofs.

Lemma 2.8.1

$$\lim_{\delta T \to 0} \left\{ \frac{1}{\delta T} \left(X(T + \delta T) - X(T) \right) \right\} = A(T)\, X(T) \qquad (2.8.1)$$

where X(T) is a fundamental matrix defined by

$$\dot{X}(T) = A(T)\, X(T) \qquad (2.8.2)$$

with

$$X(0) = I \qquad (2.8.3)$$

Proof: This result follows trivially from the fact that the left hand side of (2.8.1) is the definition of $\dot{X}(T)$.

Q.E.D.

Lemma 2.8.2

$$\lim_{\delta T \to 0} \left[\frac{1}{\delta T} X(T + \delta T) \int_{T}^{T+\delta T} X(-t)\, B(t)\, dt \right] = B(T) \qquad (2.8.4)$$

where X(T) is defined above and B(t) is any continuous matrix.

Proof: Since X(t) and B(t) are continuous in t, the mean value theorem can be applied to yield:

$$\lim_{\delta T \to 0} \left[\frac{1}{\delta T} \int_{T}^{T+\delta T} X(T + \delta T - t)\, B(t)\, dt \right]$$

$$= \lim_{\delta T \to 0} \left[\frac{1}{\delta T} X(T + \delta T - \tau)\, B(\tau)\, (T + \delta T - T) \right] \qquad (2.8.5)$$

where $T \leq \tau \leq T + \delta T$.

Also,

$$\lim_{\delta T \to 0} \left[\frac{1}{\delta T} \int_{T}^{T+\delta T} X(T + \delta T - t)\, B(t)\, dt \right]$$

$$= \lim_{\delta T \to 0} X(T + \delta T - \tau)\, B(\tau)$$

$$= X(T - T)\, B(T) = X(0)\, B(T) = B(T) \qquad (2.8.6)$$

because, as $\delta T \to 0$, τ must approach T.

Q.E.D.

A special case of this is:

Lemma 2.8.3

$$\lim_{\delta T \to 0} \left[\frac{1}{\delta T}\, e^{A(T+\delta T)} \int_{T}^{T+\delta T} e^{-At} dt \right] = I \qquad (2.8.7)$$

Proof: Since e^{At} satisfies (2.8.2) and (2.8.3), this follows immediately from Lemma 2.8.2 with B(T) = I.

Q.E.D.

3. THE TIME-INVARIANT LINEAR SYSTEM

This chapter is concerned entirely with the time-invariant linear system as defined in Chapter 2. The reason for starting with a special case is two-fold: first, this problem is easier to handle mathematically and easier to understand physically than more general cases. Second, the results are more extensive for this case than in the more general cases. All of the results that appear in the more general cases do appear in this case. Thus, this chapter aids the understanding of later material.

The trajectory system is defined in section 3.1, and its differential equation is derived. A fact necessary for the solution of the trajectory system differential equation is presented in section 3.2. Three computational procedures are discussed in section 3.3, and the results of some example problems are presented in section 3.4. The results of the chapter are summarized in section 3.5.

3.1 THE TRAJECTORY SYSTEM

The state equation for the time-invariant linear system has already been introduced in Chapter 2:

$$\dot{\underline{x}} = A\underline{x} + B\underline{u} \qquad (2.2.5)$$

where $\underline{u}$ is admissible and the initial state $\underline{x}_0$ is in the region of null controllability.

The objective here is to develop a means of finding the set of optimal final states $\left\{ \underline{x}_T^*(T) \,\middle|\, 0 \leq T \leq T^* \right\}$, which is called the <u>solution trajectory</u>. This is done by developing a set of differential equations whose solution is the solution trajectory. Thus, this set of equations

is called the trajectory system. The independent variable of the trajectory system is T, the run time in the original system. It is necessary to pay particular attention to the differences between T and t in the following, because they both appear in the development of the trajectory system equations.

The first step in developing the trajectory system equations is to describe the location of a single point of the solution trajectory. Such a point is the final point of an optimal trajectory of the original system. An arbitrary point on that optimal trajectory, given at the time τ, satisfies the equation:

$$\underline{x}_T^*(\tau) = e^{A\tau}\left\{\underline{x}_0 + \int_0^{\tau} e^{-At}\, B\underline{u}^*(T,t)\, dt\right\} \qquad (3.1.1)$$

where the first argument of $\underline{u}^*$ denotes the run time and thus the particular problem, and the second argument denotes the particular point of the trajectory. It is necessary that $\tau \leq T$. The only point of (3.1.1) known to lie on the solution trajectory — besides $\underline{x}_0$, trivially — is the final point obtained by setting $\tau = T$. For convenience, the primed notation

$$\underline{x}'(T) = \underline{x}_T^*(T) \qquad (3.1.2)$$

is used to designate a point of the solution trajectory. Thus, $\underline{x}'(T)$ is the state variable of the trajectory system. Now,

$$\underline{x}'(T) = e^{AT}\left\{\underline{x}_0 + \int_0^{T} e^{-At}\, B\underline{u}^*(T,t)\, dt\right\} . \qquad (3.1.3)$$

Having obtained an equation for a point on the solution trajectory as a function of T, it is necessary to put that equation in the form most convenient for use in finding the whole solution trajectory. That form is a differential equation obtained by differentiating (3.1.3) with respect to T. This cannot be done by Leibnitz' rule because $\underline{u}^*(T,t)$ is not necessarily continuous, so the following approach is used.

The differential equation which is sought is given in (3.1.15). As the first step in the derivation of that differential equation $\underline{x}'$ is written for a slightly larger run time as

$$\underline{x}'(T + \delta T) = e^{A(T+\delta T)} \left\{ \underline{x}_0 + \int_0^{T+\delta T} e^{-At}\, B\underline{u}^*(T + \delta T, t)\, dt \right\} \quad (3.1.4)$$

Now, let

$$\delta\underline{x}'(T) = \underline{x}'(T + \delta T) - \underline{x}'(T) , \quad (3.1.5)$$

$$\delta\underline{u}^*(T,t) = \underline{u}^*(T + \delta T, t) - \underline{u}^*(T,t) , \quad (3.1.6)$$

and subtract (3.1.3) from (3.1.4) to get

$$\delta\underline{x}'(T) = \left[e^{A(T+\delta T)} - e^{AT} \right] \left\{ \underline{x}_0 + \int_0^{T} e^{-At}\, B\underline{u}^*(T,t)\, dt \right\}$$

$$+ e^{A(T+\delta T)} \int_T^{T+\delta T} e^{-At}\, B\underline{u}^*(T + \delta T, t)\, dt$$

$$+ e^{A(T+\delta T)} \int_0^{T} e^{-At}\, B\delta\underline{u}^*(T,t)\, dt$$

$$= \left[e^{A\delta T} - I \right] \underline{x}'(T) + e^{A(T+\delta T)} \int_T^{T+\delta T} e^{-At}\, B\underline{u}^*(T+\delta T, t)\, dt$$

$$+ e^{A(T+\delta T)} \int_0^T e^{-At}\, B\delta\underline{u}^*(T,t)\, dt \quad . \qquad (3.1.7)$$

If δT is chosen small enough so that no control switching occurs in the interval $(T, T+\delta T)$, the control vector in the second term becomes a constant — call it $\underline{u}^*_e(T)$ — and can be removed from the integral:

$$e^{A(T+\delta T)} \int_T^{T+\delta T} e^{-At}\, B\underline{u}^*(T+\delta T, t)\, dt$$

$$= e^{A(T+\delta T)} \left[\int_T^{T+\delta T} e^{-At} dt \right] B\underline{u}^*_e(T) \quad . \qquad (3.1.8)$$

From (2.5.10) and (2.5.12)

$$\underline{u}^*_e(T) = \underline{SGN} \left\{ -B^T \underline{x}^*_T(T) \right\} = \underline{SGN} \left\{ -B^T \underline{x}'(T) \right\} \quad . \qquad (3.1.9)$$

Substituting (3.1.8) into (3.1.7), dividing by δT, and taking the limit as δT vanishes yields

$$\lim_{\delta T \to 0} \left[\frac{\delta \underline{x}'(T)}{\delta T} \right] = \dot{\underline{x}}'(T) = \lim_{\delta T \to 0} \left\{ \left[\frac{e^{A\delta T} - I}{\delta T} \right] \underline{x}'(T) \right.$$

$$+ \frac{1}{\delta T} e^{A(T+\delta T)} \left[\int_T^{T+\delta T} e^{-At} dt \right] B\underline{u}^*_e(T)$$

$$\left. + \frac{1}{\delta T} e^{A(T+\delta T)} \int_0^T e^{-At}\, B\delta\underline{u}^*(T,t) dt \right\} . \qquad (3.1.10)$$

Equation (3.1.10) is the differential equation of the trajectory system, but each of the three terms on the right hand side must be simplified. The first term can be reduced by a polynominal expansion of $e^{A\delta T}$:

$$\lim_{\delta T\to 0}\left[\frac{e^{A\delta T}-I}{\delta T}\right] = \lim_{\delta T\to 0}\left[\frac{(I + A\delta T + \frac{A^2\delta T^2}{2!} + \cdots) - I}{\delta T}\right] = A \qquad (3.1.11)$$

By Lemma (2.8.3), the second term becomes

$$\lim_{\delta T\to 0}\left\{\frac{1}{\delta T}\, e^{A(T+\delta T)}\int_T^{T+\delta T} e^{-At}dt\right\} B\underline{u}^*(T) = B\underline{u}^*(T) \qquad (3.1.12)$$

To evaluate the final term in (3.1.10), first start by considering a scalar control u* with only one switch. This switch occurs at $\tau(T)$ in $u^*(T,\cdot)$ and at $\tau(T+\delta T)$ in $u^*(T+\delta T,\cdot)$. $\delta u^*(T,t)$ is zero except between these two switch times, where it has a magnitude of two. Thus, the term can be reduced using the above and the mean value theorem to:

$$\lim_{\delta T\to 0}\left[\frac{1}{\delta T}\, e^{A(T+\delta T)}\int_0^T e^{-At}\,\underline{b}\delta u^*(T,t)\,dt\right]$$

$$= \lim_{\delta T\to 0}\left[\frac{1}{\delta T}\, e^{A(T+\delta T)}\int_{(T)}^{(T+\delta T)} e^{-At}\,\underline{b}(\pm 2)\,dt\right]$$

$$= \lim_{\delta T\to 0}\left[\frac{1}{\delta T}\, e^{A(T+\delta T-t')}\,\underline{b}(\pm 2)(\tau(T+\delta T) - \tau(T))\right] \qquad (3.1.13)$$

where $\tau(T) \le t' \le \tau(T+\delta T)$ or $\tau(T+\delta T) \le t' \le \tau(T)$, and the sign of $\delta u^*(T,t)$ is not yet known. Letting k take on one of the values ± 1, (3.1.13) becomes

$$\lim_{\delta T \to 0} \left[2ke^{A(T+\delta T-t')} \underline{b} \frac{\tau(T+\delta T) - \tau(T)}{\delta T} \right]$$

$$= 2ke^{A(T-\tau(T))} \underline{b} \lim_{\delta T \to 0} \left[\frac{\tau(T+\delta T) - \tau(T)}{\delta T} \right]. \quad (3.1.14)$$

$\tau(\cdot)$ has been shown to be continuous in Theorem 2.7.1. On any interval for which it is also differentiable, (3.1.14) becomes $2k\dot{\tau}(T)e^{A(T-\tau(T))}\underline{b}$. If $u^*(T,\cdot)$ switches more than once, this becomes

$$2k_1\dot{t}_1(T)\ e^{A(T-t_1(T))}\ \underline{b} + 2k_2\dot{t}_2(T)\ e^{A(T-t_2(T))}\ \underline{b} + \cdots$$

Also, if $\underline{u}^*(T,\cdot)$ is of order $m \geq 1$ and each element switches once, it becomes

$$\sum_{i=1}^{m} 2k_i\dot{t}_i(T)\ e^{A(T-t_i(T))}\ \underline{b}_i$$

where $\underline{b}_i$ is the i^{th} column of B and $t_i(T)$ is the switch time of $u^{i*}(T,\cdot)$. In general, each element of $\underline{u}^*(T,\cdot)$ can switch any number of times, and each switch generates a separate term as above. Call the total number of switches r: r does not depend on m. Now, (3.1.10) reduces to

$$\dot{\underline{x}}'(T) = A\underline{x}'(T) + B\underline{u}_e^*(T) + \sum_{i=1}^{r} 2k_i\dot{t}_i(T)\ e^{A(T-t_i(T))}\ \underline{b}_i \quad (3.1.15)$$

This is the differential equation of the trajectory system.

As in Figures 2.7.3 and 2.7.4, each input has some run time such that for shorter run times it does not switch. For each of the terms in the summation in (3.1.15), let T_{i0} designate this threshold run

time. In (3.1.13) the j^{th} term of the summation is zero until $T \geq T_{j0}$. This fact is incorporated automatically by defining $t_i(T)$ as in section 2.7 thus making $\dot{t}_i(T) = 0$ for $T < T_{i0}$.

Equation (3.1.15) plus the initial condition given by $\underline{x}'(0) = \underline{x}_0$ defines the trajectory system. For clarity, the results of this section are summarized in the following theorem.

Theorem 3.1.1. Given the system (2.2.5)

$$\dot{\underline{x}} = A\underline{x} + B\underline{u}$$

where $\underline{u}$ is admissible and $\underline{x}(0) = \underline{x}_0$ is in the region of null controllability, the set of optimal final states of the various optimal regulator problems $\left\{ \underline{x}_T^*(T) \,\middle|\, 0 \leq T \leq T^* \right\}$ is coincident with the solution of the trajectory system differential equation given by

$$\dot{\underline{x}}'(T) = A\underline{x}'(T) + B\underline{u}_e^*(T) + \sum_{i=1}^{r} \dot{t}_i(T)\, \underline{v}_i(T) \qquad (3.1.17)$$

where

$$\underline{x}'(0) = \underline{x}_0 \ , \qquad (3.1.18)$$

$$\underline{u}_e^*(T) = \underline{SGN} \left\{ -B^T \underline{x}'(T) \right\} \qquad (3.1.9)$$

and

$$\underline{v}_i(T) = 2k_i e^{A(T-t_i(T))}\, \underline{b}_i \ . \qquad (3.1.16)$$

This theorem has been proven above. The main problem at this point is the difficulty in solving the trajectory system differential equation due to the terms of the summation.

3.2 EVALUATION OF $\dot{t}_i(T)$

Equation (3.1.15) indicates that the trajectory system differential equation can be expressed in terms of $t_i(T)$ and $\dot{t}_i(T)$. Before proceeding to methods for solving this system for the locus of optimal final states, a further fact which is useful for computing $\dot{t}_i(T)$ is presented in the following Theorem:

Theorem 3.2.1

$$\langle \underline{x}'(T), \underline{v}_i(T) \rangle = 0, \quad i = 1, \ldots, r \tag{3.2.1}$$

for all $T \geq T_{i0}$

Proof:

$$\underline{y}_T^*(t_i(T)) = e^{A^T(T-t_i(T))}\, \underline{y}_T^*(T) \tag{3.2.2}$$

from (2.5.7). And from (2.5.10)

$$\underline{y}_T^*(t_i(T)) = e^{A^T(T-t_i(T))}\, \underline{x}'(T) \quad . \tag{3.2.3}$$

But, by the definition of the switch time $t_i(T)$,

$$\langle \underline{b}_i, e^{A^T(T-t_i(T))}\, \underline{y}_T^*(T) \rangle = 0 \tag{3.2.4}$$

Thus,

$$\begin{aligned} 0 &= \langle \underline{b}_i, e^{A^T(T-t_i(T))}\, \underline{x}'(T) \rangle \\ &= \langle \underline{x}'(T), e^{A(T-t_i(T))}\, \underline{b}_i \rangle \end{aligned} \tag{3.2.5}$$

Now, multiplying both sides of (3.2.5) by $2k_i$ yields (3.2.1).

Q.E.D.

3.3 COMPUTATIONAL SOLUTION OF THE TRAJECTORY SYSTEM DIFFERENTIAL EQUATION

Three methods are presented in this section for finding the locus of optimal final states by solving the differential equation of the trajectory system. Method 1 is noniterative and well suited to use on a digital computer. Method 2 is noniterative and can be used on an analog computer as well as a digital computer. The last method is called Method 2A because it is an extension of Method 2. Its use is advantageous in certain limited situations which are discussed.

Method 1. The most straightforward method of solving (3.1.15) is by discretizing (3.1.15) and finding the points of the solution trajectory sequentially. Given $\underline{x}'(T)$ and each $t_i(T)$ for some $T\varepsilon[0,T^*]$ and an increment ΔT, $\underline{x}'(T + \Delta T)$ is given approximately by

$$\underline{x}'(T + \Delta T) = \underline{x}'(T) + \dot{\underline{x}}'(T)\ \Delta T \qquad (3.3.1)$$

where $\dot{\underline{x}}'(T)$ is given by (3.1.15) if the $\dot{t}_i(T)$'s are known. But the $\dot{t}_i(T)$'s can be found by choosing values such that when each $t_i(T + \Delta T)$ is computed by

$$t_i(T + \Delta T) = \dot{t}_i(T)\ \Delta T + t_i(T) \qquad (3.3.2)$$

(3.2.1) will be satisifed at $T + \Delta T$. This procedure does involve an iteration on the set of $\dot{t}_i(T)$'s, but the method is straightforward and even lends itself to hand computation with few enough switches. An example is given later in the chapter.

In order to obtain the solution by Method 1, it is necessary to have a means of computing a correction factor. That is, assume a problem with only one switch time $\tau(T)$. At run time T, $\underline{x}'(T)$, $\tau(T)$

and $\underline{v}(T)$ are assumed to be known. Further it should be true that $\langle \underline{x}'(T), \underline{v}(T) \rangle = 0$. The first step was taken at T = 0 or from T = 0 to T = ΔT. To do this an average value of $\dot{\tau}$, say $\dot{\tau}(\Delta T/2)$, would have been used. At the second step, either $\dot{\tau}(3\Delta T/2)$ could be computed and used, or it could be computed but

$$\dot{\tau}(T) = \frac{\dot{\tau}(\frac{\Delta T}{2}) + \dot{\tau}(\frac{3\Delta T}{2})}{2} \tag{3.3.3}$$

could be used for the next computation. The latter method is used here, so at run time T, it is also assumed that $\tau(T - \frac{\Delta T}{2})$ is known.

Now the value of $\underline{x}'(T + \Delta T)$ is found by guessing a value of $\dot{\tau}(T + \frac{\Delta T}{2}) = \dot{\tau}_1$ and then using

$$\begin{aligned} \underline{x}'(T + \Delta T) &\simeq \underline{x}'(T) + \dot{\underline{x}}'(T)\, \Delta T \\ &= x'(T) + \Delta T \left\{ A\underline{x}'(T) + b\underline{u}_e^*(T) + \dot{\tau}_1(T)\, \underline{v}(T) \right\} \end{aligned} \tag{3.3.4}$$

where

$$\dot{\tau}_1(T) = \frac{1}{2} (\dot{\tau}_1 + \dot{\tau}(T - \tfrac{\Delta T}{2})) \tag{3.3.5}$$

also

$$\underline{v}(T + \Delta T) = 2ke^{A[T + \Delta T - \tau(T) - \Delta T\dot{\tau}_1(T)]}\, \underline{b} \tag{3.3.6}$$

Now let the inner product of $\underline{x}'(T + \Delta T)$ and $\underline{v}(T + \Delta T)$ — which is a function of $\dot{\tau}_1$ only — be represented by the symbol $I(\dot{\tau}_1)$. If $\dot{\tau}_2$ can be found by using $\dot{\tau}_2 = \dot{\tau}_1 + \Delta\dot{\tau}$, it is true that

$$I(\dot{\tau}_2) = 0 \tag{3.3.7}$$

but

$$I(\dot{\tau}_2) = I(\dot{\tau}_1) + \frac{\partial I}{\partial \dot{\tau}_1} \Delta\dot{\tau} + \cdots \quad . \tag{3.3.8}$$

Using only the terms in (3.3.8),

$$\Delta\dot{\tau} \simeq - \frac{I(\dot{\tau}_1)}{\frac{\partial I}{\partial \dot{\tau}_1}} \tag{3.3.9}$$

where

$$\frac{\partial I}{\partial \dot{\tau}_1} = \left\langle \frac{\partial \underline{x}'(T + \Delta T)}{\partial \dot{\tau}_1} , \underline{v}(T + \Delta T) \right\rangle + \left\langle \underline{x}'(T + \Delta T), \frac{\partial \underline{v}(T + \Delta T)}{\partial \dot{\tau}_1} \right\rangle \tag{3.3.10}$$

From (3.3.4) and (3.3.6)

$$\frac{\partial \underline{x}'(T + \Delta T)}{\partial \dot{\tau}_1} = \frac{\Delta T}{2} \underline{v}(T) \quad , \tag{3.3.11}$$

and

$$\frac{\partial \underline{v}(T + \Delta T)}{\partial \dot{\tau}_1} = - \frac{\Delta T}{2} A\underline{v}(T + \Delta T) \quad . \tag{3.3.12}$$

Thus,

$$\frac{\partial I}{\partial \dot{\tau}_1} = \frac{\Delta T}{2} \left\{ \left\langle \underline{v}(T), \underline{v}(T + \Delta T) \right\rangle - \left\langle \underline{x}'(T + \Delta T), A\underline{v}(T + \Delta T) \right\rangle \right\} \tag{3.3.13}$$

By substituting (3.3.15) into (3.3.9) a correction factor can be obtained which leads to $\dot{\tau}_2(T + \frac{\Delta T}{2})$, and the iteration is continued until the inner product becomes as near zero as is desired.

An exception occurs where the first step is being made. In that case $\dot{\tau}(-\frac{\Delta T}{2})$ does not exist, so $\dot{\tau}(T) = \dot{\tau}(0)$ is computed, and later it is called $\dot{\tau}(\frac{\Delta T}{2})$. In this case, instead of (3.2.13) the following is used:

$$\frac{\partial I}{\partial \dot{t}} = \Delta T \left\{ \langle \underline{v}(T), \underline{v}(T + \Delta T) \rangle - \langle \underline{x}'(T + \Delta T) , A\underline{v}(T + \Delta T) \rangle \right\} \quad (3.3.13a)$$

For the more general case where there is more than one switch, values of $\dot{t}_i(T)$ are guessed for each i. Then an iteration is carried out over one $\dot{t}_i(T)$ at a time with the others held constant. This makes (3.3.4) become

$$\underline{x}'(T + \Delta T) = \underline{x}'(T) + \Delta T \left\{ A\underline{x}'(T) + \underline{b}u_e^*(T) + \dot{t}_k(T)\, \underline{v}_k(T) + \sum_{\substack{i=1 \\ i \neq k}}^{r} \dot{t}_i(T)\, \underline{v}_i(T) \right\} \quad (3.3.14)$$

Since the terms of the summation are constant while the iteration is being carried out over $\dot{t}_k(T)$, $\frac{\partial \underline{x}'(T + \Delta T)}{\partial t_k}$ is still given by (3.3.11). So the correction factor for each $\dot{t}_k(T)$ is found from (3.3.9) and (3.3.13).

Since the solution trajectory is being computed sequentially from T = 0 to T = T^*, it is necessary that there be some method for detecting a threshold switch time T_{i0} when it occurs. Also the value $\underline{v}_i(T_{i0})$ must be determined. Considering the latter first: from section 2.7,

$$t_i(T_{i0}) = \begin{cases} T_{i0} & \text{, type I switch} \\ 0 & \text{, type II switch} \end{cases} \quad (3.3.15)$$

This is used in (3.1.16) to find

$$\underline{v}_i(T_{i0}) = \begin{cases} 2k_i \underline{b}_i & \text{, Type I switch} \\ 2k_i e^{AT_{i0}} \underline{b}_i & \text{, Type II switch} \end{cases} \quad (3.3.16)$$

If the i^{th} switch is a Type I switch, then it first occurs at run time T_{i0} where $\underline{y}^*_{T_{i0}}(T_{i0})$ is an element of a switch hyperplane. Thus, $\underline{x}'(T_{i0})$ is also a member of that switch hyperplane, so T_{i0} can be detected for a Type I switch by observing when $\underline{u}^*_e(T)$ switches. Also, by noting which element of $u^*_e(T)$ switches, $\underline{v}_i(T_{i0})$ is found to be $2k_i\underline{b}_i$ where only k_i must be determined.

If instead the i^{th} switch is a Type II switch, then when it first occurs at T_{i0}, the starting point $\underline{y}^*_{T_{i0}}(0) = e^{AT_{i0}}\underline{y}^*_{T_{i0}}(T_{i0})$ is an element of a switch hyperplane. This T_{i0} is detected by realizing that $\langle \underline{x}'(T_{i0}), e^{AT_{i0}}\underline{b}_i \rangle = 0$. The second element of the inner product can be found by defining

$$\underline{z}_i(T) = e^{AT}\underline{b}_i \quad . \tag{3.3.17}$$

Now,

$$\dot{\underline{z}}_i(T) = A\underline{z}_i(T) \quad , \tag{3.3.18}$$

or

$$\underline{z}_i(T + \Delta T) = \underline{z}_i(T) + \Delta T(A\underline{z}_i(T)) \tag{3.3.19}$$

Thus the solution trajectory is computed from $\underline{x}_0$ at $T = 0$, and simultaneously (3.3.19) is solved from $2\underline{b}_i$ at $T = 0$, then T_{i0} is the shortest run time for which $\langle \underline{x}'(T), \underline{z}_i(T) \rangle = 0$. Also, when T_{i0} is reached, $\underline{v}_i(T_{i0}) = 2k_i e^{AT_{i0}}\underline{b}_i$ is known except for the signed factor k_i.

At the time $T = T_{i0}$, $\dot{t}_i(T)\ \underline{v}_i(T)$ is added to the already existing equation for $\underline{x}'(T)$. At that time k_i can be determined. The direction of the additional vector term is known without k_i, k_i merely designates the sense of that vector. Thus, at $T = T_{i0}$, k_i is chosen so that the additional vector — which is a part of $\dot{\underline{x}}'(T)$ — has the same general

sense as those already existing terms of $\dot{\underline{x}}'(T)$. In other words, k_i is chosen so that

$$\langle \dot{\underline{x}}'(T_{i0}-), \dot{t}_i(T_{i0})\, \underline{v}_i(T_{i0}) \rangle > 0 \qquad (3.3.20)$$

The steps of Method 1 are shown in the following algorithm:

Algorithm 3.3.1

1. At run time T, $\underline{x}'(T)$, $\underline{z}_j(T)$, j=1, ..., m, $\underline{v}_i(T)$, and $\dot{t}_i$ $(T - \frac{\Delta T}{2})$, i=1, ..., r, are known. Let k=1.

2. Let $\dot{t}_i(T + \frac{\Delta T}{2}) = \dot{t}_i(T - \frac{\Delta T}{2})$, i=1, ..., r. Find each $\underline{v}_i$ $(T + \Delta T)$ using (3.3.6) and then find $\underline{x}'(T + \Delta T)$ using (3.3.14). Find each $\underline{z}_j(T)$ using (3.3.19).

3. Evaluate $I_k = \langle \underline{v}_k(T + \Delta T), \underline{x}'(T + \Delta T) \rangle$.

4. If I_k is not sufficiently small, improve $\dot{t}_k(T + \frac{\Delta T}{2})$ using (3.3.9) and (3.3.13), and return to 2. When I_k is sufficiently small, increase k by 1 for k<r and let k=1 for k=r then return to 2. Repeat until each I_k is sufficiently small.

5. Determine whether a threshold switch time has been reached by checking for switches in $\underline{u}_e^*(\cdot)$ or a sign change in $\langle \underline{x}'(\cdot), \underline{z}_j(\cdot) \rangle$ from T to $(T + \Delta T)$. If no new switch, return to 1. In case of a switch set $\underline{v}(T_0)$ according to (3.3.16), determine k_i from (3.3.20), and return to 1.

Method 2. Another method for computing $\underline{v}_i(T)$ has been developed in order to avoid the iteration necessary in Method 1. This method is suitable for use on an analog computer. Differentiating both sides

of (3.1.16) with respect to T yields

$$\dot{\underline{v}}_i(T) = 2k_i(1 - \dot{t}_i(T))\, Ae^{A(T-t_i(T))}\underline{b}_i$$

$$= (1 - \dot{t}_i(T))\, A\underline{v}_i(T) \qquad (3.3.21)$$

where $\underline{v}_i(T_{i0})$ is determined according to (3.3.16). If k_i and $\dot{t}_i(T)$ are known and if it is known whether the i^{th} switch is a type I or a type II switch, $\underline{v}_i(T)$ can be found using (3.3.21) and (3.3.16). Theorem 3.2.1 may be used to simultaneously remove the need to find $\dot{t}_i(T)$ in (3.3.21) and to determine $\dot{t}_i(T)$ for use in (3.1.17). This is accomplished by solving (3.3.21) for $\underline{v}_i(T)$ where $\dot{t}_i(T)$ is chosen to be whatever value it must take on in order to satisfy (3.2.1). In this way $\dot{t}_i(T)$ is computed implicitly and can also be used in (3.1.17). Using an analog computer, the factor $(1 - \dot{t}_i(T))$ would be generated with a bootstrapping technique. This would become an iterative process on a digital computer. The constant k_i can be found using (3.3.20).

As long as $\dot{t}_i(T)$ is continuous, (3.1.16) shows that $\underline{v}_i(T)$ is also continuous — in fact, differentiable. Thus (3.1.17) shows that $\underline{x}'(T)$ is also differentiable where $\underline{u}_e^*(T)$ is continuous. Let the r switch times be arranged so that $T_{10} \leq T_{20} \leq \cdots \leq T_{r0}$. Each element of $\underline{u}_e^*(T)$ is piecewise constant, switching only at a threshold switch time T_{i0}. Thus for $0 \leq T < T_{10}$, $\underline{u}_e^*(T)$ is constant and each $\underline{v}_i(T) = 0$, so $\underline{x}'(T)$ is differentiable. For $T_{10} < T < T_{20}$, $\underline{u}_e^*(T)$ is also constant and as long as $\dot{t}_i(T)$ is continuous, $\underline{v}_i(T)$ and thus $\underline{x}'(T)$ are differentiable. This fact plus Theorem 2.7.3 show that $\dot{\underline{x}}'(T)$ is continuous if and only if $\dot{t}_i(T)$ is continuous. Therefore, except at the times T_{i0}, $\underline{x}'(T)$, $\underline{v}_i(T)$ and $t_i(T)$ are all differentiable. Furthermore, they are all continuous for $0 \leq T \leq T^*$.

As long as sufficient computer capacity is available, $\underline{x}'(T)$ can readily be found by solving the equations of Method 2 on an analog computer. The required analog capacity can be estimated by the number of integrators and multipliers required. Equation (3.1.17) requires n integrators and (3.3.21) requires (n x r) integrators. If T is desired, an analog integrator is needed. And r integrators are needed to find the $t_i(T)$ from $\dot{t}_i(T)$. Thus, the total number of analog integrators required is at least (n+1) x (r+1). Further, n multipliers are needed for calculating each of the inner products in (3.2.1) and (3.3.20). Thus, for example, in a second order system with one input switch, six integrators and four multipliers are needed.

But, it was also noted in the previous section that in order to locate the Type II switches it is necessary to solve (3.3.18) for each $\underline{v}_i(T)$. Thus, in addition to the above number of integrators, (m x n) more integrators are needed to search for additional Type II switches. It is only if the number of Type II switches is known in advance that these are not needed — each can be put to use as soon as the last Type II switch is found. An example of the use of an analog computer for the solution of the problem is given later in the chapter.

Method 2 can be programmed on a digital computer using Algorithm 3.3.1 with the following changes: instead of (3.3.6) use

$$\underline{v}(T + \Delta T) = \underline{v}(T) + \Delta T \left[(1 - \dot{\tau}(T))\, A\underline{v}(T) \right] , \qquad (3.3.21a)$$

instead of (3.3.12) use

$$\frac{\partial \underline{v}(T + \Delta T)}{2\dot{\tau}_1} = - \frac{\Delta T}{2} A\underline{v}(T) , \qquad (3.3.21b)$$

and instead of (3.3.13) use

$$\frac{\partial I}{\partial \dot{t}_1} = \frac{\Delta T}{2}\left\{\langle \underline{v}(T), \underline{v}(T + \Delta T)\rangle - \langle \underline{x}'(T + \Delta T), A\underline{v}(T)\rangle\right\} . \qquad (3.3.21c)$$

But this has no computational advantage over Method 1 in the case of a linear time-invariant system.

Method 2A. In the case of a second-order system, Method 2 can be simplified and the number of integrators reduced. This new method is called Method 2A. In such a system, each $\underline{v}_i(T)$ is normal to $\underline{x}'(T)$ and thus they are all parallel to one another. Define the vector $\underline{n}(T)$ to be a unit vector normal to $\underline{x}'(T)$. Thus,

$$\langle \underline{n}(T), \underline{n}(T)\rangle = 1 \qquad (3.3.22)$$

and

$$\langle \underline{n}(T), \underline{x}'(T)\rangle = 0 \qquad (3.3.23)$$

define $\underline{n}(T)$. One possible vector is

$$\underline{n}(T) = \frac{1}{|\underline{x}'(T)|}\begin{bmatrix} x^{2'}(T) \\ -x^{1'}(T) \end{bmatrix} , \qquad (3.3.24)$$

the other is its negative. Now define a set of scalars $\left\{c_i(T)\right\}$ so that

$$\underline{v}_i(T) = c_i(T)\, \underline{n}(T), \; i = 1, \ldots, r \; . \qquad (3.3.25)$$

For convenience, choose the polarity of $\underline{n}(T)$ so that

$$c_i(T_{10}) > 0 \quad . \tag{3.3.26}$$

Now from (3.2.1)

$$\langle \underline{x}'(T), \underline{v}_1(T) \rangle = 0 \tag{3.3.27}$$

so it follows — using (3.3.21) — that

$$\begin{aligned} 0 = \frac{d}{dT}\langle \underline{x}'(T), \underline{v}_1(T) \rangle &= \langle \dot{\underline{x}}'(T), \underline{v}_1(T) \rangle \\ &+ \langle \underline{x}'(T), \dot{\underline{v}}_1(T) \rangle = \langle \dot{\underline{x}}'(T), \underline{v}_1(T) \rangle \\ &+ \langle \underline{x}'(T), [1 - \dot{t}_1(T)] A\underline{v}_1(T) \rangle \\ &= \langle \dot{\underline{x}}'(T) + [1 - \dot{t}_1(T)] A^T \underline{x}'(T), \underline{v}_1(T) \rangle \end{aligned} \tag{3.3.28}$$

But, using (3.1.17), this becomes

$$\begin{aligned} 0 = \langle \underline{v}_1(T), A\underline{x}'(T) + B\underline{u}_e^*(T) + [1 - \dot{t}_1(T)] A^T \underline{x}'(T) \\ + \sum_{i=1}^{r} \dot{t}_1(T)\, \underline{v}_i(T) \rangle \quad . \end{aligned} \tag{3.3.29}$$

This reduces to

$$\begin{aligned} 0 = c_1(T) \Big\{ \langle \underline{n}(T), A\underline{x}'(T) + B\underline{u}_e^*(T) \\ + [1 - \dot{t}_1(T)] A^T \underline{x}'(T) \rangle + \sum_{i=1}^{r} \dot{t}_1(T)\, c_i(T) \Big\} \quad . \end{aligned} \tag{3.3.30}$$

Since $c_1(T) \neq 0$,

$$\begin{aligned} \sum_{i=1}^{r} \dot{t}_i(T)\, c_i(T) = -\langle \underline{n}(T), A\underline{x}'(T) + B\underline{u}_e^*(T) \\ + [1 - \dot{t}_1(T)] A^T \underline{x}'(T) \rangle \quad . \end{aligned} \tag{3.3.31}$$

But (3.1.17) can be rewritten as

$$\dot{\underline{x}}'(T) = A\underline{x}'(T) + B\underline{u}_e^*(T) + \left[\sum_{i=1}^{r} \dot{t}_1(T)\, c_i(T)\right] \underline{n}(T) \qquad (3.3.32)$$

and this can be combined with (3.3.31) to yield

$$\dot{\underline{x}}'(T) = \underline{q}(T) - \langle \underline{n}(T),\, \underline{q}(T) + [1 - \dot{t}_1(T)]\, A^T \underline{x}'(T) \rangle\, \underline{n}(T) \qquad (3.3.33)$$

where

$$\underline{q}(T) = A\underline{x}'(T) + B\underline{u}_e^*(T) \quad . \qquad (3.3.34)$$

If r is at least two, fewer integrators are needed because (3.3.21) needs to be solved only for one set — $\underline{v}_1(T)$ — and that only to determine $[1-\dot{t}_1(T)]$. But n additional multipliers are required to compute the inner product in (3.3.33). The additional multipliers would not be a problem on a digital computer. One potential difficulty is that if $\underline{v}_1(T)$ ever becomes zero, (3.3.31) does not follow from (3.3.30). Another problem is that the switch times can not be computed — except $t_1(T)$ — although this does result in a savings of an additional r integrators. Also, the switch times are not part of the solution of the problem defined in Chapter 2.

Theorem 2.6.1 gives the maximum number of switches for a system in which A has real eigenvalues. If A has one or more complex conjugate pairs of eigenvalues, a large number of switches could exist. Method 2A is useful as an efficient means of solving such a problem for a second-order system. Method 2A can not be extended to higher order systems because it depends on the fact that in a second-order system there is a unique normal to $\underline{x}'(T)$.

3.4 COMPUTATIONAL RESULTS

Method 1 and Method 2 have been used to solve some problems. The results are presented in this section. A computer program using Algorithm 3.1.1 to solve Problem 1 and Problem 2 for any system with a scalar control was written and is documented in Appendix II. It was used to solve Problem 1 and Problem 2 for the double-integrator plant by Method 1; the computed loci of optimal final states for four different initial conditions are shown in Figure 3.4.1. The solution to Problem 2, T^*, as a function of $\underline{x}_0$ is given in Athans and Falb — Equation (7-26), page 514. These values are compared with the computed results in Table 3.4.1. The computation times on an IBM 1800 were about 20 seconds. It should be noted that only a moderate accuracy was required from the program and no attempt was made to minimize either computation time or error. It might also be noted that the initial conditions chosen represent Type I switches, Type II switches, and a point on the switch line (hyperplane).

Method 2 was used to solve Problem 1 and Problem 2 for the double-integrator plant on an analog computer. The initial condition considered was $\underline{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, which is one of the initial conditions shown in Figure 3.4.1. The results of this analog solution are shown in Figure 3.4.2.

The program in Appendix II using Method 1 was also used to solve Problems 1 and 2 for the system:

$$\dot{\underline{x}}(t) = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} -1 \\ -2 \end{bmatrix} u(t) \qquad (3.4.1)$$

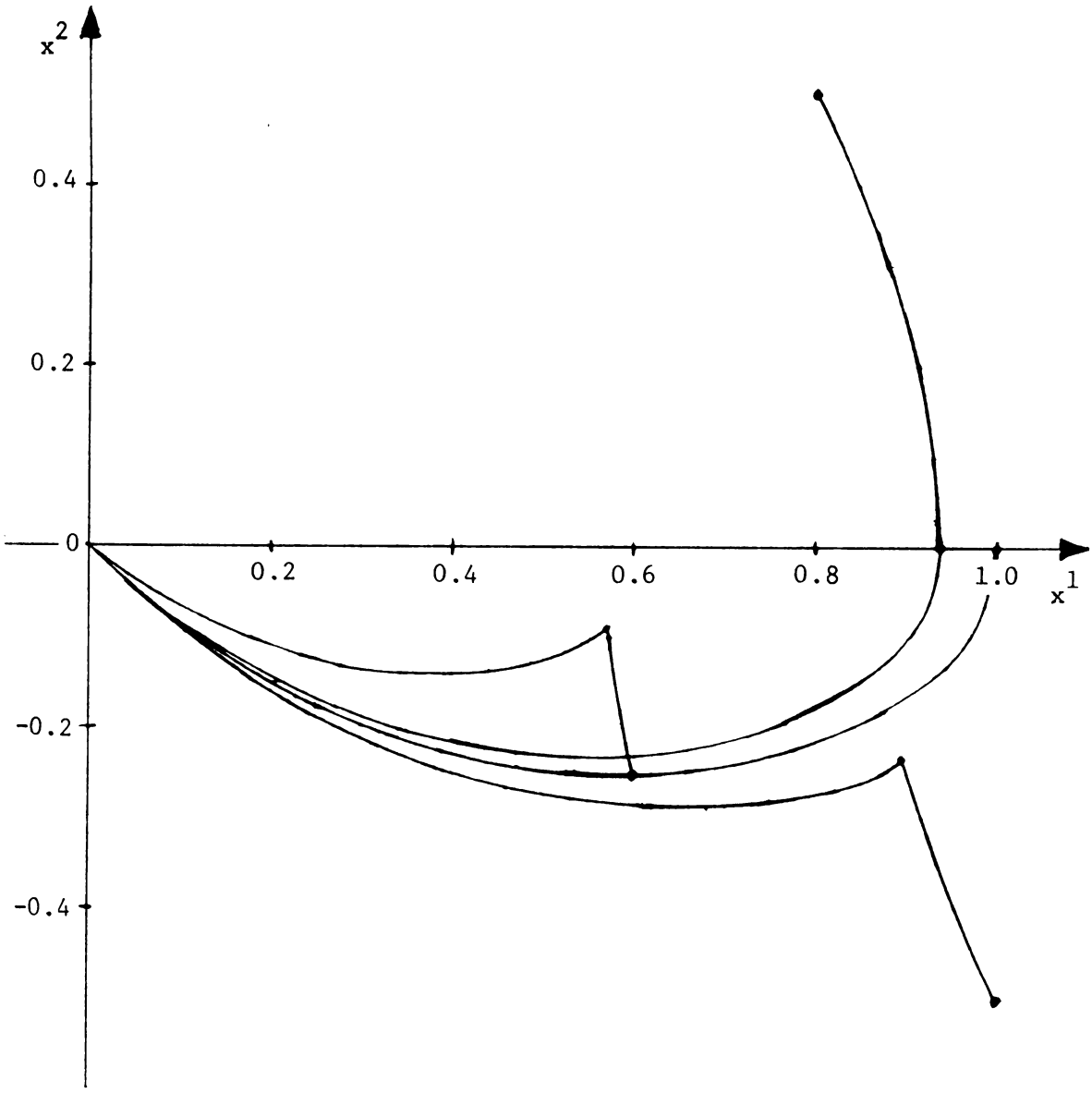

Figure 3.4.1. Solutions of Problem 1 for the Double-Integrator Plant

Table 3.4.1. Error in the Solutions to Problem 2 for the Double-Integrator Plant

$\underline{x}_0^T$	Actual T^*	Computed T^*	% Error
(1.00, 0.00)	2.000	2.019	0.95
(0.80, 0.50)	2.425	2.454	1.19
(0.60, -0.25)	1.340	1.329	0.82
(1.00, -0.50)	1.620	1.604	0.99

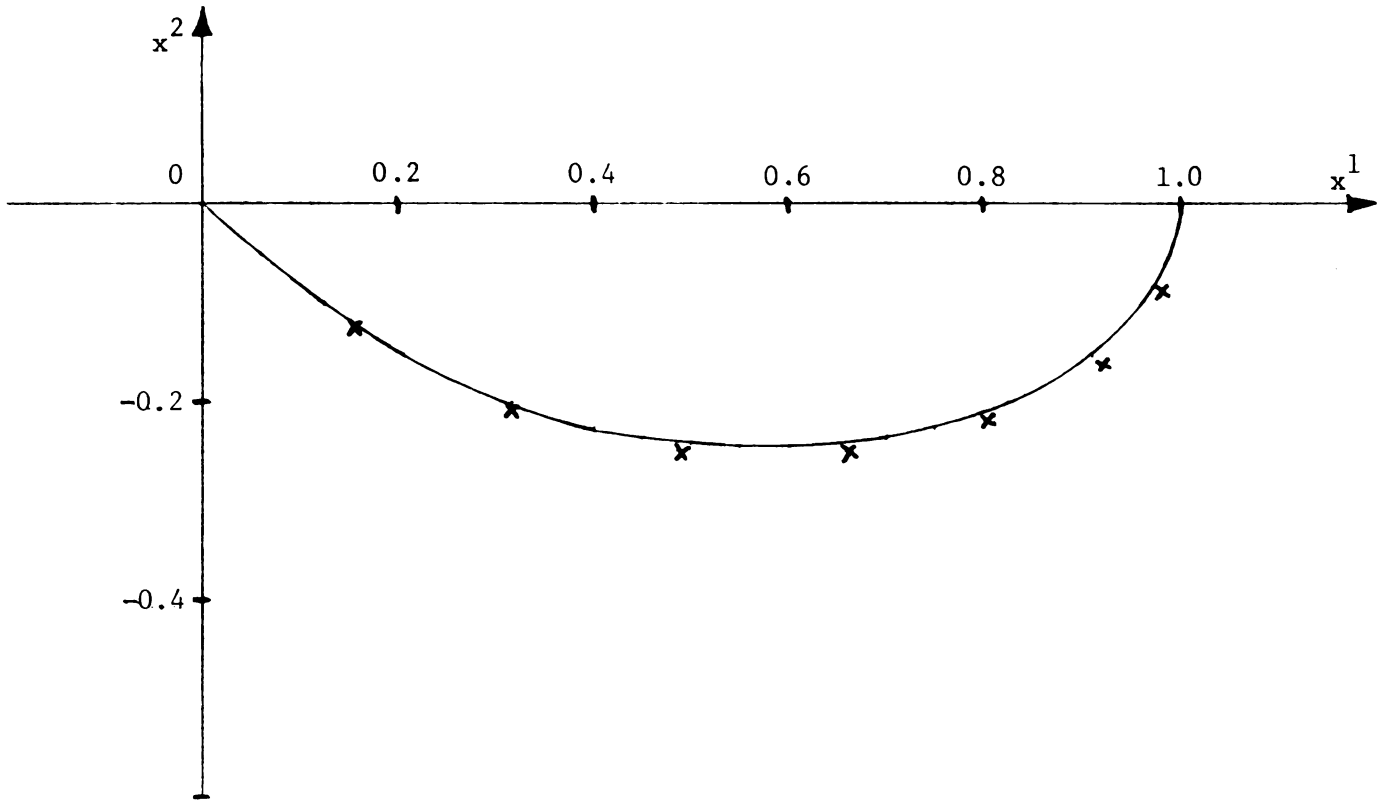

Figure 3.4.2. Analog Computer Solution of Double-integrator Plant with $\underline{x} = \begin{bmatrix} 1.0 \\ 0.0 \end{bmatrix}$

where

$$|u| \leq 1 \tag{3.4.2}$$

for several initial conditions. The results appear in Figure 3.4.3. This system is discussed in Athans and Falb pages 526-536, but no results useful for comparison are given there.

Method 1 was used to solve Problem 1 and Problem 2 for the system given by

$$\dot{\underline{x}}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} u(t) \tag{3.4.3}$$

where

$$|u| \leq 1 \tag{3.4.4}$$

and

$$\underline{x}_0 = \begin{bmatrix} 1.5 \\ 1.3 \\ 2.0 \end{bmatrix} . \tag{3.4.5}$$

This system is discussed in Athans and Falb, pages 536-551, but none of the results given there are useful for comparison. The solution trajectory computed for the problem above is shown in Figure 3.4.4.

3.5 SUMMARY

The trajectory system has been defined and a differential equation (3.1.15) describing it is derived. Three methods which use (3.1.15) along with (3.2.1) to find the locus of optimal final states for the problem defined in Theorem 3.1.1 are discussed in section 3.3. Method 1 and Method 2 are equally suitable for use on a digital com-

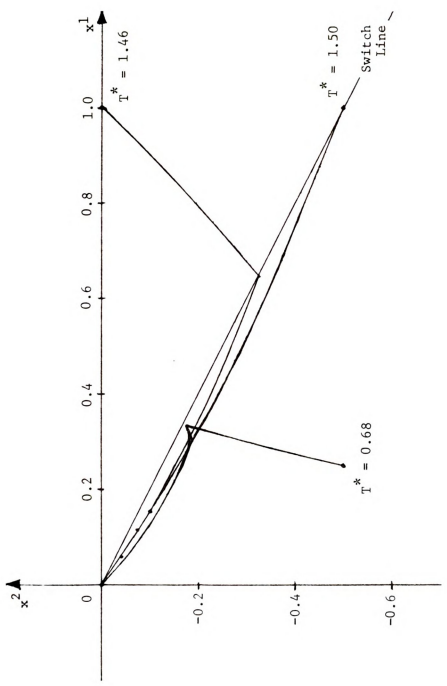

Figure 3.4.3. Solutions to Problems 1 and 2 for the System of Equation (3.4.1)

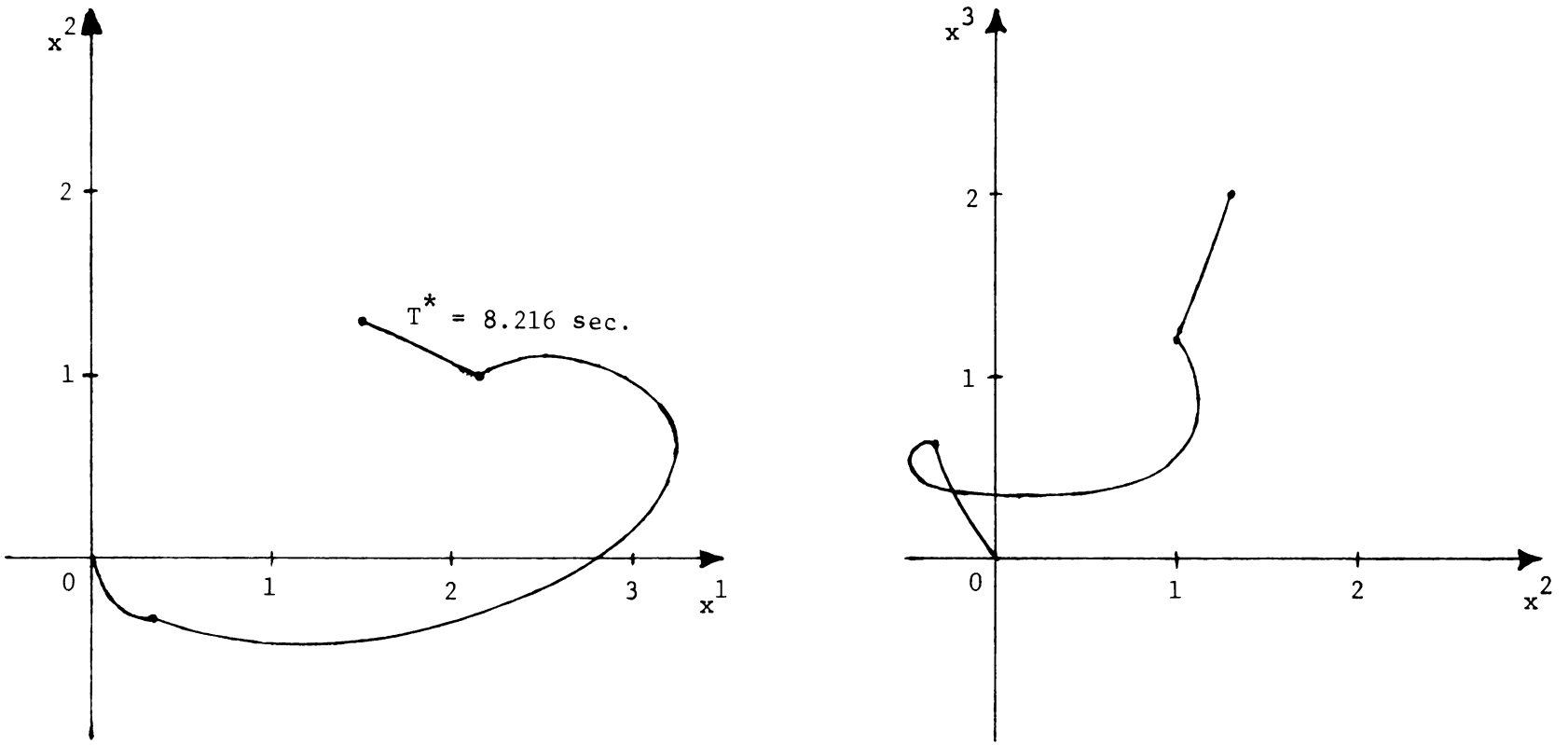

Figure 3.4.4. Solutions to Problems 1 and 2 for the Third Order System of Equation (3.4.3.)

computer via Algorithm 3.1.1, but Method 2 can be used on an analog computer as well. Method 2A is developed for second-order systems with several switches. The data resulting from solution of some example problems is presented in section 3.4.

4. THE GENERAL LINEAR SYSTEM

Chapter 3 is concerned with the time-invariant linear system. In this chapter, the results of Chapter 3 are extended to the case of a general or time-varying linear system. The trajectory system differential equation is developed in section 4.1, and the normality of $\underline{v}_i(\cdot)$ is attached in section 4.2. Section 4.3 discusses the extensions of Method 1 and Method 2 to the time-varying case. The solution of a simple problem in section 4.4 demonstrates the computational procedure developed. The results are summarized in section 4.5.

4.1 THE TRAJECTORY SYSTEM

From Chapter 2, the state equation for a general linear system can be given as

$$\dot{\underline{x}} = A(t)\underline{x} + B(t)\underline{u} \qquad (2.2.4)$$

where $\underline{u}$ is admissible and the initial state $\underline{x}_0$ is in the region of null controllability. Also, the matrices A(t) and B(t) are assumed continuous. The object of this section is to develop the trajectory system equations for this general linear system. The method used will be similar to that used in Chapter 3. The results are stated in the following theorem.

Theorem 4.1.1. For the linear system defined by

$$\dot{\underline{x}} = A(t)\underline{x} + B(t)\underline{u} \qquad (2.2.4)$$

where $\underline{u}$ is an admissible control and $\underline{x}(0) = \underline{x}_0$ is in the region of null controllability, the set of optimal final states of the various optimal regulator problems $\left\{\underline{x}_T^*(T) \,\middle|\, 0 \leq T \leq T^*\right\}$ is coincident with the

solution of the trajectory system given by:

$$\dot{\underline{x}}'(T) = A(T)\,\underline{x}'(T) + B(T)\,\underline{u}_e^*(T) + \sum_{i=1}^{r} 2k_i\,\dot{t}_i(T)\,X(T - t_i(T))\,\underline{b}_i(t_i(T)) \qquad (4.1.1)$$

where

$$\underline{x}'(0) = \underline{x}_0 \qquad (4.1.2)$$

and

$$\underline{u}_e^*(T) = \underline{SGN}\left\{-B^T(T)\,\underline{x}'(T)\right\} \qquad (4.1.3)$$

Proof: As in Chapter 3, for the run time T, the point on the optimal trajectory corresponding to the time τ is given by:

$$\underline{x}_T^*(\tau) = X(T)\left\{\underline{x}_0 + \int_0^{\tau} X(-t)\,B(t)\,\underline{u}^*(T,t)\,dt\right\} \qquad (4.1.4)$$

where $0 \leq \tau \leq T$ and $0 \leq T \leq T^*$. $X(T)$ is the fundamental matrix defined in 2.8. Also, the arguments of $\underline{u}^*$ are as in Chapter 3. The point of that optimal trajectory which is also on the solution trajectory is the final point:

$$\underline{x}'(T) = X(T)\left\{\underline{x}_0 + \int_0^{T} X(-t)\,B(t)\,\underline{u}^*(T,t)\,dt\right\}. \qquad (4.1.5)$$

Also, for a slightly longer run time

$$\underline{x}'(T + \delta T) = X(T + \delta T)\left\{\underline{x}_0 + \int_0^{T+\delta T} X(-t)\,B(t)\,\underline{u}^*(T + \delta T,t)\,dt\right\}. \qquad (4.1.6)$$

Subtracting (4.1.5) from (4.1.6) yields

$$\delta\underline{x}'(T) = \Big[X(T+\delta T) - X(T)\Big] \left\{\underline{x}_0 + \int_0^T X(-t)\, B(t)\, \underline{u}^*(T,t)dt\right\} + X(T+\delta T)\int_T^{T+\delta T} X(-t)\, B(t)\, \underline{u}^*(T+\delta T,t)dt + X(T+\delta T)\int_0^T X(-t)\, B(t)\, \delta\underline{u}^*(T,t)dt \qquad (4.1.7)$$

If δT is chosen small enough so that no control switching occurs in the interval $(T, T+\delta T)$, the control vector in the second term becomes a constant and that term can be rewritten as

$$X(T+\delta T)\int_T^{T+\delta T} X(-t)\, B(t)\, \underline{u}^*(T+\delta T, t)dt = X(T=\delta T)\left[\int_T^{T+\delta T} X(-t)\, B(t)dt\right]\underline{u}_e^*(T) \qquad (4.1.8)$$

where $\underline{u}_e^*(T)$ is as given in (4.1.3) by (2.5.10) and (2.5.12).

After substituting (4.1.8), dividing by δT, and taking the limit as δT vanishes, (4.1.7) becomes

$$\lim_{\delta T\to 0}\left[\frac{\delta\underline{x}'(T)}{\delta T}\right] = \dot{\underline{x}}'(T) = \lim_{\delta T\to 0}\left\{\left[\frac{X(T+\delta T) - X(T)}{\delta T}\right]X(-T)\,\underline{x}'(T) + \frac{1}{\delta T}X(T+\delta T)\left[\int_T^{T+\delta T} X(-t)\, B(t)dt\right]\underline{u}_e^*(T) + \frac{1}{\delta T}X(T+\delta T)\int_0^T X(-t)\, B(t)\,\delta\underline{u}^*(T,t)dt\right\} \qquad (4.1.9)$$

The first term of the right-hand side can be reduced by Lemma 2.8.1, the second term by Lemma 2.8.2, and the third term as follows — after section 3.1 — :

$$\lim_{\delta T \to 0} \left\{ \frac{1}{\delta T} X(T + \delta T) \int_0^T X(-t)\, B(t)\, \delta \underline{u}^*(T,t) dt \right\}$$

$$= \lim_{\delta T \to 0} \left\{ \frac{1}{\delta T} X(T + \delta T) \left[\sum_{i=1}^{r} \int_{t_i(T)}^{t_i(T+\delta T)} (2k_i)\, X(-t)\, \underline{b}_i(t) dt \right] \right\}$$

$$= \lim_{\delta T \to 0} \left\{ \sum_{i=1}^{r} \frac{2k_i}{\delta T} X(T + \delta T - \tau_i)\, \underline{b}_i(\tau_i) \left[t_i(T + \delta T) - t_i(T) \right] \right\}$$

$$= \sum_{i=1}^{r} 2k_i\, \dot{t}_i(T)\, X(T - t_i(T))\, \underline{b}_i(t_i(T)) \qquad (4.1.10)$$

where $t_i(T) \leq \tau_i \leq t_i(T + \delta T)$ or $t_i(T + \delta T) \leq \tau_i \leq t_i(T)$, i=1, ..., r and $k_i = \pm 1$. Now (4.1.9) reduces to (4.1.1). Equation (4.1.2) is obvious.

Q.E.D.

By the same argument posed in section 3.3, $\dot{\underline{x}}'(T)$ and $\dot{t}_i(T)$ are continuous except at the times T_{i0}.

4.2 SOLUTION OF THE TRAJECTORY SYSTEM EQUATION

The trajectory system differential equation (4.1.1) must be solved in order to find the locus of optimal final states. To simplify consideration of the terms of the summation, a set of vectors are again defined according to:

$$\underline{v}_i(T) = 2k_i\, X(T - t_i(T))\, \underline{b}_i(t_i(T)) \,, \qquad (4.2.1)$$

where $\underline{v}_i(T_{i0})$ is still given by (3.3.16). As in the case of the time-invariant linear system, there are two basic ways of computing $\underline{v}_i(T)$: either it can be computed directly from (4.2.1), or its differential equation can be solved concurrently with (4.1.1). The differential equation for $\underline{v}_i(T)$ can be found by differentiating both sides of (4.2.1) with respect to T to get:

$$\dot{\underline{v}}_i(T) = \left[1 - \dot{t}_i(T)\right] A(T - t_i(T))\, \underline{v}_i(T) + \dot{t}_i(T)\, X(T - t_i(T))\, \dot{\underline{b}}_i(t_i(T)) \qquad (4.2.2)$$

Whichever method is used for computing $\underline{v}_i(T)$, an important fact necessary for the solution of the trajectory system equations is given by:

Theorem 4.2.1. For the system defined in 4.1,

$$\langle \underline{x}'(T), \underline{v}_i(T) \rangle = 0, \quad i=1, \ldots, r \qquad (4.2.3)$$

for all $T \geq T_{i0}$.

Proof:

$$\underline{y}_T^*(t_i(T)) = X^T(T - t_i(T))\, \underline{y}_T^*(T) = X^T(T - t_i(T))\, \underline{x}'(T) \qquad (4.2.4)$$

from (2.5.7) and from (2.5.10). By the definition of the switch time,

$$\langle \underline{b}_i(t_i(T)), \underline{y}_T^*(t_i(T)) \rangle = 0 \qquad (4.2.5)$$

Thus

$$0 = \langle \underline{b}_i(t_i(T)), X^T(T - t_i(T))\, \underline{x}'(T) \rangle = \langle \underline{x}'(T), X(T - t_i(T))\, \underline{b}_i(t_i(T)) \rangle \qquad (4.2.6)$$

Now (4.2.3) is obtained by multiplying both sides of (4.2.6) by $2k_i$.

Q.E.D.

4.3 COMPUTATIONAL METHODS

As stated above, there are basically two ways to compute $\underline{v}_i(T)$. Each leads to a different computational procedure. The two procedures presented here are very similar to those presented in section 3.3.

Method 1. As in section 3.3, Method 1 involves the direction computation of $\underline{v}(T)$ from its defining equation (4.2.1). If $A(\cdot)$ and $B(\cdot)$ are known functions of time, then $X(\cdot)$ can be computed. Solution of (4.2.1) is then no more difficult with a digital computer than was its counterpart in the time-invariant linear system. But the overall method still involves finding $\dot{\tau}(T)$ by an iterative technique; so, a means for improving the estimate of $\dot{\tau}(T + \frac{\Delta T}{2})$ is needed.

The correction factor is formed in the same way as it was in section 3.3. But since

$$\underline{v}(T + \Delta T) = 2k_i \, X\left[T + \Delta T - \tau(T) - \Delta T \, \dot{\tau}_1(T)\right] \underline{b}_i\left[\tau(T) + \Delta T \, \dot{\tau}_1(T)\right], \tag{4.3.1}$$

there is a more complicated expression for

$$\begin{aligned}
\frac{\partial \underline{v}(T + \Delta T)}{\partial \dot{\tau}_1} &= 2k_i \Big\{ -\frac{\Delta T}{2} A\left[T + \Delta T - \tau(T) - \Delta T\dot{\tau}_1(T)\right] \\
&\qquad X\left[T + \Delta T - \tau(T) - \Delta T\dot{\tau}_1(T)\right] \underline{b}\left[\tau(T) + \Delta T\dot{\tau}_1(T)\right] \\
&\qquad + X\left[T + \Delta T - \tau(T) - \Delta T\dot{\tau}_1(T)\right] \dot{\underline{b}}\left[\tau(T) + \Delta T\dot{\tau}_1(T)\right] \Big\} \\
&= -\frac{\Delta T}{2} \Big\{ A\left[T + \Delta T - \tau(T) - \Delta T\dot{\tau}_1(T)\right] \underline{v}(T + \Delta T) \\
&\qquad - 2k_i X\left[T + \Delta T - \tau(T) - \Delta T\dot{\tau}_1(T)\right] \dot{\underline{b}}\left[\tau(T) + \Delta T\dot{\tau}_1(T)\right] \Big\}
\end{aligned} \tag{4.3.2}$$

If $\underline{b}$ is constant, (4.3.2) becomes the same as (3.3.13) except that $A(\cdot)$ must be evaluated at the argument shown. In that case, the correction factor is changed very little. For the more general time-varying $\underline{b}$

$$\frac{\partial I}{\partial t_1} = \frac{\Delta T}{2}\Big\{\langle \underline{v}(T), \underline{v}(T+\Delta T)\rangle - \langle \underline{x}'(T+\Delta T), A(T+\Delta T - \tau(T) - \Delta T\dot{\tau}_1(T))\ \underline{v}(T+\Delta T)\rangle + 2k_i \langle \underline{x}'(T+\Delta T), X(T+\Delta T - \tau(T) - \Delta T\ \dot{\tau}_1(T))\ \dot{\underline{b}}(\tau(T) + \Delta T\ \dot{\tau}_1(T))\rangle\Big\} \tag{4.3.3}$$

must be used to determine the correction factor. This requires that $\underline{b}(\cdot)$ be differentiable. The solution of the trajectory system differential equation by Method 1 is very similar to the procedure outlined in Algorithm 3.3.1. The similarities between the two versions of Method 1 are even more pronounced if B is a constant matrix.

Method 2. As in Chapter 3, Method 2 is a solution method in which (4.1.1), (4.2.1), and (4.2.2) are solved simultaneously — even on an analog computer. As in the case of Method 1, Method 2 is more easily adapted to the time-varying linear system if B is a constant matrix, because that removes the second term from the right-hand side of (4.2.2). But even if $B(\cdot)$ is not constant, it must be differentiable. Differentiability of $B(\cdot)$ is not necessary for the solution of (4.1.1), only to use (4.2.2) in solving (4.1.1). Evaluation of the first term of (4.2.2) requires evaluating $A(\cdot)$ at $(T - t_i(T))$. Doing this on an analog computer would probably be too involved, so even Method 2 may as well be considered a digital computer method. Again, this requires an iteration over $\dot{t}_i(T)$.

If B is a constant matrix, Method 2 can be used with Algorithm 3.1.1. In this case

$$\underline{v}(T + \Delta T) = \underline{v}(T) + \Delta T \left\{ \left[1 - \dot{\tau}(T) \right] A(T - \tau(T)) \, \underline{v}(T) \right\}. \quad (4.3.4)$$

Thus,

$$\frac{\partial \underline{v}(T + \Delta T)}{\partial \dot{\tau}_1} = - \frac{\Delta T}{2} A(T - \tau(T)) \, \underline{v}(T) \quad , \quad (4.3.5)$$

making

$$\frac{\partial I}{\partial \dot{\tau}_1} = \frac{\Delta T}{2} \left\{ \langle \underline{v}(T + \Delta T), \underline{v}(T) \rangle - \langle \underline{x}'(T + \Delta T), A(T - \tau(T)) \, \underline{v}(T) \rangle \right\} \quad (4.3.6)$$

Now Method 2 is seen to be far superior to Method 1 because the fundamental matrix does not appear anywhere in the process. Instead, only the known matrix A(·) must be evaluated in the system equations or in the correction factor equations. When B is a function of time, the second term on the right hand side of (4.2.2) shows that the fundamental matrix must be used in the problem solution, so the superiority of Method 2 is not extended to this case.

4.4 AN EXAMPLE PROBLEM

To illustrate the use of Method 2 in Algorithm 3.1.1 for a constant B matrix, the double-integrator plant was changed to the system:

$$\dot{\underline{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & t \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \quad (4.4.1)$$

where

$$|u| \leq 1 \quad (4.4.2)$$

and

$$\underline{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \quad (4.4.3)$$

This problem was solved using the program in Appendix II. The results are shown in Figure 4.4.1.

4.5 SUMMARY

The trajectory system is extended to the time-varying linear case (4.1.1). The solution of (4.1.1) and (4.2.3) by Method 1 and Method 2 are discussed in section 4.3. In particular it is shown that Method 2 is far superior to Method 1 when B is a constant matrix. A simple extension of the double-integrator plant is used in section 4.4 to demonstrate the use of Method 2 in Algorithm 3.3.1 when B is constant.

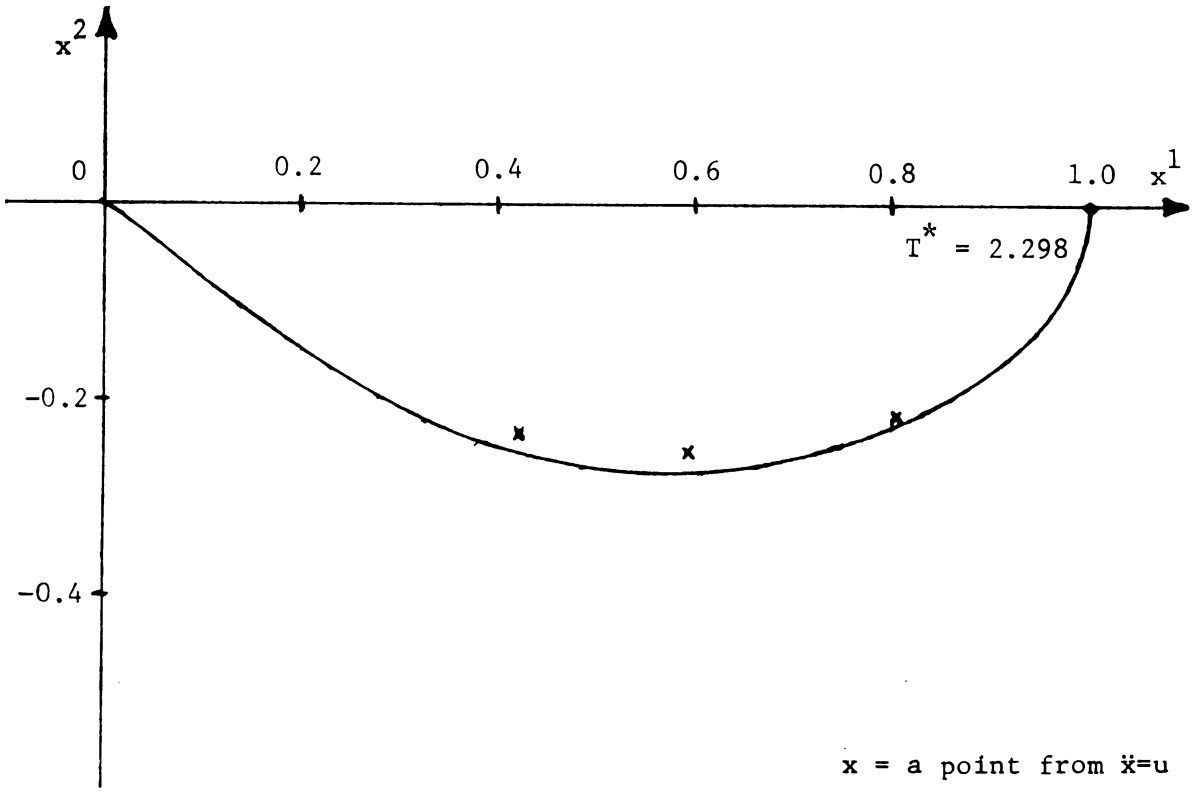

Figure 4.4.1. Solutions of Problems 1 and 2 for the System of Equation (4.4.1)

5. A CLASS OF NONLINEAR SYSTEMS

In this chapter the results of Chapters 3 and 4 are extended to a class of nonlinear systems. It should be noted at the outset that the results of this chapter have not been proven, they are merely a reasonable extension of the previous results. The problem encountered is that a proof of these results requires that the set of optimal final states be continuous. This is definitely not true for all nonlinear systems. In fact, all systems can be put into one of three classifications: 1) some systems — including all linear systems — have no local optima other than the global optimum and it is continuous; 2) some systems have a discontinuous global optimum which is made up of pieces of various local optima each of which is continuous; and 3) some systems have discontinuous local optima. For systems in the first group, the methods presented in this thesis work as they are presented. For systems in the second group, each local optimum can be found by the methods presented in this paper, but it is then necessary to construct the global optimum from these local optima. For systems in the third group, this thesis provides no method of solution. At this time, no procedure is available for determining a priori to which group a given nonlinear system belongs. The only approach with a nonlinear system is to assume it is in one of the first two groups. If it is in the third group, it is not known that it would be evident that the solution method had failed. In fact, this problem would be an interesting and probably a fruitful area for further research.

In section 5.1, the trajectory system differential equation is developed for a class of nonlinear systems. The convexity of the reachable set is discussed in section 5.2. The normality of $\underline{v}_i(\cdot)$ is

attached in section 5.3, and a solution method is discussed. A simple example is presented in section 5.4, and section 5.5 is a summary.

5.1 THE TRAJECTORY SYSTEM DIFFERENTIAL EQUATION

The state equation for the class of systems to be discussed here has been given in Chapter 2

$$\dot{\underline{x}} = \underline{f}(\underline{x}) + B(t)\,\underline{u} \tag{2.2.3}$$

where $\frac{\partial \underline{f}}{\partial \underline{x}}$ exists, $\underline{u}$ is admissible and the initial state $\underline{x}_0$ is in the region of null controllability. The matrix B(t) is assumed to be continuous. The trajectory system equations for this class of nonlinear systems is developed in this section subject to the assumption mentioned above.

Theorem 5.1.1. For the class of nonlinear systems defined by

$$\dot{\underline{x}} = \underline{f}(\underline{x}) + B(t)\,\underline{u} \tag{2.2.3}$$

where $\underline{u}$ is an admissible control, $\underline{x}(0) = \underline{x}_0$ is in the region of null controllability and the set of optimal final states of the various optimal regulator problems $\{\underline{x}_T^*(T) \mid 0 \leq T \leq T^*\}$ is continuous, this same set of optimal final states is coincident with the solution of the trajectory system given by:

$$\dot{\underline{x}}'(T) = \underline{f}(\underline{x}'(T)) + B(T)\,\underline{u}_e^*(T) + \sum_{i=1}^{r} 2k_i\,\dot{t}_1(T)\,X(T - t_i(T))\,\underline{b}_i(t_i(T)) \tag{5.1.1}$$

where

$$\underline{x}'(0) = \underline{x}_0 \tag{5.1.2}$$

and

$$\underline{u}_e^*(T) = \underline{SGN}\left\{-B^T(T)\ \underline{x}'(T)\right\} \quad . \tag{5.1.3}$$

Also $X(\cdot)$ is the fundamental matrix of the system linearized along an optimal trajectory — i.e., $X(\cdot)$ is the fundamental matrix associated with $A(t) = \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{\underline{x}_T^*(t)}$.

Proof: As in the preceding chapters it is first necessary to obtain $\delta\underline{x}'(T) = \underline{x}'(T + \delta T) - \underline{x}'(T)$. Finding $\underline{x}'(T + \delta T)$ is done by first extending $\underline{u}^*(T,T)$ over the interval $[T, T + \delta T]$ to obtain $\underline{x}_T^*(T + \delta T)$, and then applying $\delta\underline{u}^*(T,\cdot) = \underline{u}^*(T + \delta T,\cdot) - \underline{u}^*(T,\cdot)$ to obtain $\underline{x}'(T + \delta T)$. The first step is simply given by

$$\begin{aligned} \underline{x}_T^*(T + \delta T) &= \underline{x}_T^*(T) + \dot{\underline{x}}_T^*(T)\ \delta T + O(\delta T^2) \\ &= \underline{x}_T^*(T) + \left[\underline{f}(\underline{x}_T^*(T)) + B(T)\ \underline{u}_e^*(T)\right] \delta T + O(\delta T^2) \end{aligned} \tag{5.1.4}$$

where $\underline{u}_e^*(T)$ is as given in (5.1.3). Now the second step is accomplished by linearizing the system about the trajectory $\underline{x}_T^*(T)$ for $t\varepsilon[0, T + \delta T]$. The validity of this linearization depends on the assumed continuity of the set of optimal final states. From this linearization the following differential equation is obtained:

$$\delta\dot{\underline{x}}_T^*(t) = A(t)\ \delta\underline{x}_T^*(t) + B(t)\ \delta\underline{u}^*(T,t) \tag{5.1.5}$$

where

$$A(t) = \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{\underline{x}_T^*(t)} \quad . \tag{5.1.6}$$

Solving (5.1.5)

$$\delta \underline{x}_T^*(T + \delta T) = X(T + \delta T) \left\{ \underline{x}_T^*(0) + \int_0^{T+\delta T} X(-t)\, B(t)\, \delta \underline{u}^*(T,t)\, dt \right\} \qquad (5.1.7)$$

where X(t) is the fundamental matrix for A(t). Since $\delta \underline{x}_T^*(0) = 0$, it is easy to see that

$$\underline{x}'(T + \delta T) - \underline{x}'(T) = \left[\underline{f}(\underline{x}'(T)) + B(T)\, \underline{u}_e^*(T) \right] \delta T + O(\delta T^2) + X(T + \delta T) \int_0^{T+\delta T} X(-t)\, B(t)\, \delta \underline{u}^*(T,t) dt. \qquad (5.1.8)$$

If the interval δT is chosen small enough so that no control switching occurs in $[T, T + \delta T]$, then the integral is zero over that interval. Taking that into account, and dividing both sides of (5.1.8) by δT and taking the limit as δT vanishes,

$$\dot{\underline{x}}'(T) = \underline{f}(\underline{x}'(T)) + B(T)\, \underline{u}_e^*(T) + \sum_{i=1}^{r} 2k_i\, \dot{t}_1(T)\, X(T - t_i(T))\, \underline{b}_i(T_i(T)) \qquad (5.1.1)$$

where $k_i = \pm 1$. The limit in the last term is identical to (4.1.10).

Q.E.D.

5.2 NONCONVEX REACHABLE SETS

Note that the equations developed in the previous section are valid regardless of the convexity of the reachable set. To see why,

assume the reachable set can have two lobes as shown in Figure 5.2.1. Each lobe is a locally convex portion of the reachable set and each contains an extremum. These extrema are connected by the curves A and B. The trajectory system would follow one of these curves.

The fact that two curves are available means that at some point of the solution of the trajectory system — $\underline{x}_s$ — a choice of two possible switches must be made. One choice results in curve A, the other choice in curve B. Thus, all local optima can be found and the global optimum can be constructed from them. One important point to be made here is that the assumption in Theorem 4.1.1 is that each <u>local</u> optimum is continuous. As long as this is true, the global optimum need not be continuous.

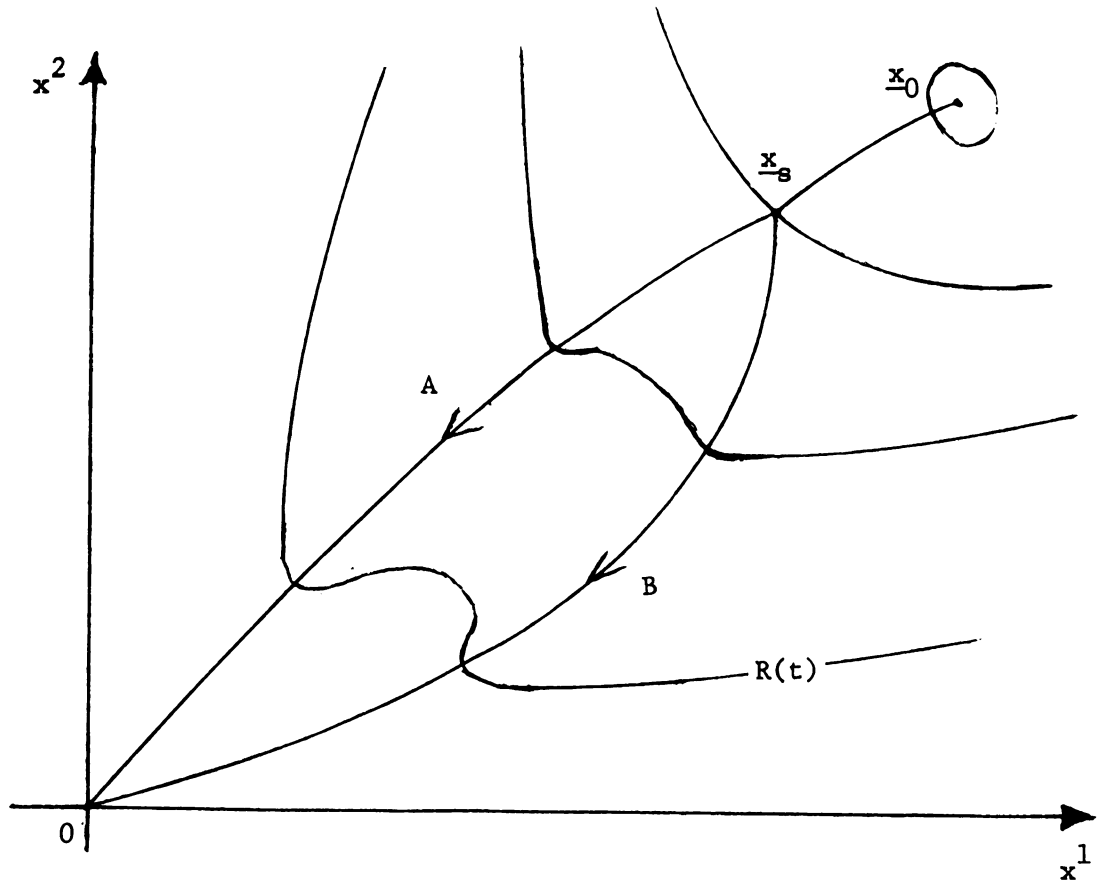

Figure 5.2.1. A Nonconvex Reachable Set

5.3 SOLUTION OF THE TRAJECTORY SYSTEM EQUATION

As in the linear systems, define the vectors

$$\underline{v}_i(T) = 2k_i\, X(T - t_i(T))\, \underline{b}_i(T_i(T)) , \tag{5.3.1}$$

i=1, ..., r. It is still true that each $\underline{v}_i(T)$ is normal to the trajectory system state vector:

Theorem 5.3.1.

$$\langle \underline{v}_i(T), \underline{x}'(T) \rangle = 0, \quad i=1, \ldots, r \tag{5.3.2}$$

for all $T \geq T_{i0}$.

Proof: From (2.5.7)

$$\dot{\underline{y}}_T^*(t) = -A^T(t)\, \underline{y}_T^*(t) \tag{5.3.3}$$

where the matrix A(t) is the same as that given in (5.1.6). Thus from (2.5.10)

$$\underline{y}_T^*(t_i(T)) = X^T(T - t_i(T))\, \underline{y}_T^*(T) \quad . \tag{5.3.4}$$

But, by the definition of the switch time $t_i(T)$,

$$\langle \underline{b}_i(t_i(T)), X^T(T - t_i(T))\, \underline{y}_T^*(T) \rangle = 0 \quad . \tag{5.3.5}$$

Thus,

$$\begin{aligned} 0 &= \langle \underline{b}_i(t_i(T)), X^T(T - t_i(T))\, \underline{x}'(T) \rangle \\ &= \langle \underline{x}'(T), X(T - t_i(T))\, \underline{b}_i(t_i(T)) \rangle \quad . \end{aligned} \tag{5.3.6}$$

Q.E.D.

All of the results to this point are identical in form to those of the time-varying linear system of Chapter 4; but it should be noted that, although $\underline{v}_i(T)$ in (5.3.1) looks identical to that in Chapter 4, its computation is far different. Specifically $X(T - t_i(T))$ must be evaluated along the original system trajectory instead of the trajectory system trajectory.

This problem can be circumvented as in Chapter 4 by letting B be a constant matrix and using solution Method 2. Differentiating (5.3.1) yields

$$\begin{aligned}\dot{\underline{v}}_i(T) &= 2k_i(1 - \dot{t}_i(T))\, A(T - t_i(T))\, X(T - t_i(T))\, \underline{b}_i \\ &= (1 - \dot{t}_i(T))\, A(T - t_i(T))\, \underline{v}_i(T) \qquad (5.3.7)\end{aligned}$$

In Algorithm 3.1.1,

$$\underline{v}_i(T + \Delta T) = \underline{v}_i(T) + \Delta T \left\{(1 - \dot{t}_i(T))\, A(T - t_i(T))\, \underline{v}_i(T)\right\} \qquad (5.3.8)$$

where

$$\frac{\partial \underline{v}_i(T + \Delta T)}{\partial \dot{t}_{il}} = -\frac{\Delta T}{2} A(T - t_i(T))\, \underline{v}_i(T) \qquad (5.3.9)$$

and

$$\frac{\partial I}{\partial \dot{t}_{il}} = \frac{\Delta T}{2}\Big\{\langle \underline{v}_i(T + \Delta T), \underline{v}_i(T)\rangle - \langle \underline{x}'(T + \Delta T), A(T - t_i(T))\, \underline{v}_i(T)\rangle\Big\} . \qquad (5.3.10)$$

It must be remembered that in the above equations $A(\cdot)$ is defined by (5.1.6) and must be evaluated accordingly. This is the difficult part of the solution.

5.4 AN EXAMPLE PROBLEM

In order to provide a simple example problem, the double-integrator plant is extended as follows:

$$\dot{\underline{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -|x^2| \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{5.4.1}$$

where

$$|u| \leq 1 \tag{5.4.2}$$

and

$$\underline{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{5.4.3}$$

This system is discussed in Athans and Falb pages 614-621. In this system

$$A(t) = \begin{bmatrix} 0 & 1 \\ 0 & -2|x_T^{2*}(t)| \end{bmatrix}, \tag{5.4.4}$$

and in (5.3.8) and (5.3.10)

$$A(T - t_i(T)) = \begin{bmatrix} 0 & 1 \\ 0 & -2|x_T^{2*}(T - t_i(T))| \end{bmatrix}. \tag{5.4.5}$$

For a given run time T, this is evaluated by running the original system backward from $\underline{x}_T^*(T) = \underline{x}'(T)$ by $t_i(T)$ seconds. At that point, $A(T - t_i(T))$ can be evaluated. Using a special subroutine to perform this calculation, the program in Appendix II was used to compute the locus of optimal final states which is shown in Figure 5.4.1.

5.5 SUMMARY

The trajectory system differential equation (5.1.1) has been developed and the normality of $\underline{v}_i(T)$ attached (5.3.2). This pair of

equations was solved by Method 2 for a simple extension of the double-integrator plant.

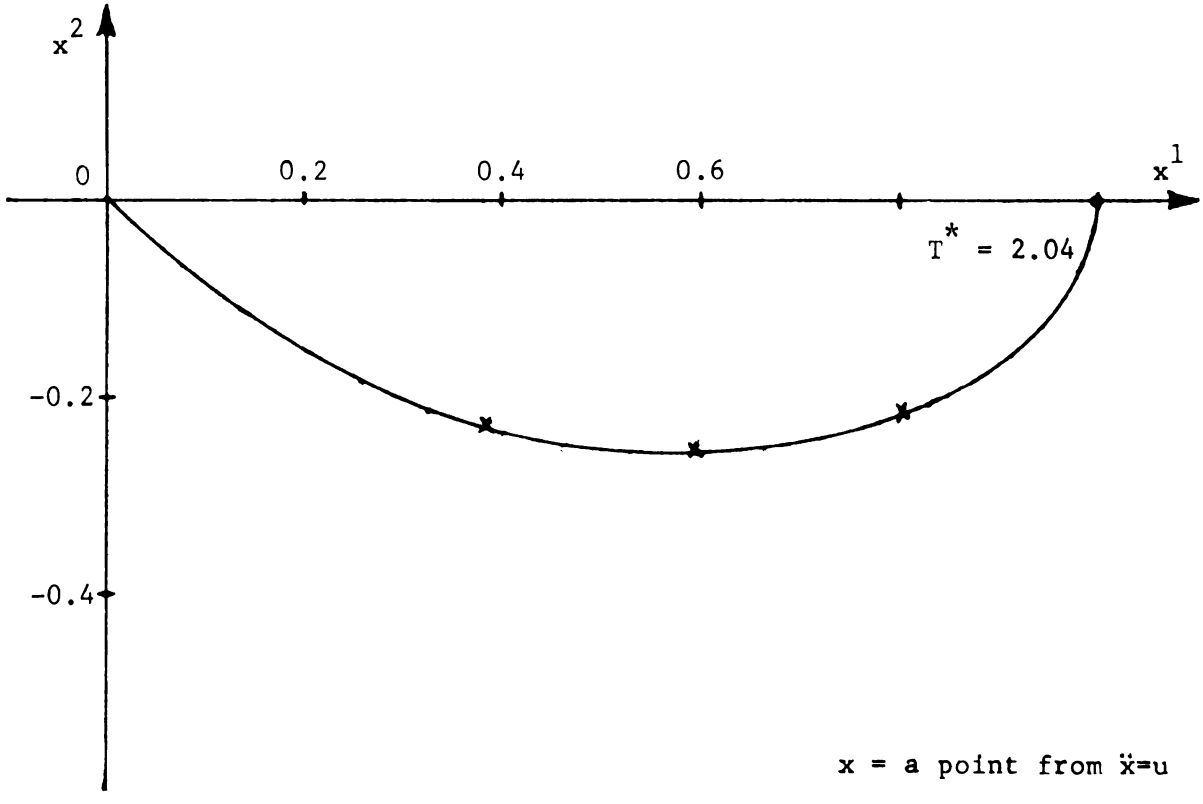

Figure 5.4.1. Solutions of Problems 1 and 2 for the System of Equation (5.4.1)

6. CONCLUSIONS AND EXTENSIONS

Section 6.1 summarizes the material presented in Chapters 2 through 5. The conclusions to be drawn from that material are included where they are pertinent. A possible extension of the trajectory system method to other control problems is treated briefly in section 6.2.

6.1 SUMMARY AND CONCLUSIONS

The particular control problems treated in this thesis are defined in Chapter 2. Briefly, Problem 1 is to determine how close to the origin a given initial state can be driven within a fixed run time; Problem 2 is to find the minimum run time necessary to drive a given initial state to the origin. The method presented in this dissertation solves all possible regulator problems for one arbitrary initial state.

In this dissertation, the above problems are treated for a class of linear and nonlinear systems. All of the systems in this class have two things in common: each element of the optimal control vector is a "bang-bang" function, and the locus of optimal final states is continuous for each local optimum. These properties are crucial to the computational procedure. Instead of solving the optimal control problem indirectly by solving the associated two-point boundary value problem which comes from the application of the Maximum Principle, this procedure solves the optimal control problem directly using the general form of the results which the Maximum Principle makes available.

Considering the set of control problems which are collectively labeled Problem 1, the specific optimal control function depends on the particular run time involved. The form of that dependence is investigated in section 2.7 and is one of the more important portions of this dissertation. In particular, each switch time is shown to be a function of the run time and each switch time $t_i(T)$ is shown to be differentiable whenever the locus of optimal final states is differentiable.

The locus of optimal final states can be computed by solving simultaneously the differential equations of the trajectory system (3.1.15), 4.1.1), (5.1.1) and the normality condition associated with the set of auxiliary inputs $\{\underline{v}_i(T)\}$ (3.2.1), (4.2.3), (5.3.2). This locus of optimal final states is the complete set of solutions of Problem 1 and readily results in the solution of Problem 2.

Method 1 and Method 2 presented in section 3.3 are two computational procedures which can be used in Algorithm 3.3.1 to solve the trajectory system equations. In Method 1, each $\underline{v}_i(T)$ is computed directly from its defining equation (3.1.16), (4.2.1), (5.3.1). In Method 2 each $\underline{v}_i(T)$ is generated from its differential equation instead. These two methods are shown to be computationally equivalent for most of the systems considered; however in a time-varying linear or a nonlinear system with a constant B matrix, Method 2 is far superior to Method 1. Both methods describe noniterative computation procedures, but an iteration is involved when using either procedure on a digital computer. Only Method 2 can be utilized on an analog computer.

A simple Fortran program which uses Algorithm 3.3.1 is presented in Appendix II. Either Method 1 or Method 2 can be used in this

program, but it only applies to systems with a scalar input that switches at most twice. This program was used to solve the example problems in sections 3.4, 4.4, 5.4. The use of an analog computer to solve an example of Problem 1 in section 3.4 shows the feasibility of analog solution.

6.2 GENERALIZATION AND FUTURE RESEARCH POSSIBILITIES

In this section the trajectory system approach to the solution of optimal control problems is generalized and then applied to the minimum fuel regulator problem. Consider an optimal control problem wherein the objective is to apply permissible inputs to a system so as to drive its state from a given initial state to the closest possible point to the origin in a fixed run time and with minimum cost. Assume the set of optimal final states resulting from all possible costs $0 \leq J \leq J^*$ — where J^* is the minimum cost necessary to reach the origin with an adequate run time — forms a continuous curve in the state space.

As in Chapter 3, consider the linear system

$$\dot{\underline{x}} = A\underline{x} + B\underline{u} \tag{6.2.1}$$

where $\underline{u}$ is admissible and $\underline{x}(0) = \underline{x}_0$. Let the fixed run time satisfy $T \leq T^*$. The equation for the optimal final state is

$$\underline{x}'(J) = \underline{x}_T^*(J) = e^{AT} \left\{ \underline{x}_0 + \int_0^T e^{-At} B\underline{u}^*(J,t)\, dt \right\} \tag{6.2.2}$$

where the notation is similar to that in the preceding chapters.

Now (6.2.2) must be differentiated with respect to J.

Consider the minimum fuel regulator problem where

$$J = F = \int_0^T \sum_{i=1}^{m} |u^i| \, dt \tag{6.2.3}$$

From the Maximum Principle, it can be shown that

$$\underline{u}^*(F,t) = - \underline{DEZ} \left\{ B' \underline{y}_F^*(t) \right\} \tag{6.2.4}$$

where

$$dez(a) = \begin{cases} -1, & a \leq -1 \\ 0, & -1 < a < 1 \\ 1, & a \geq 1 \end{cases} \tag{6.2.5}$$

and

$$\underline{DEZ}(\underline{a}) = \begin{bmatrix} dez(a_1) \\ \vdots \\ dez(a_n) \end{bmatrix} \tag{6.2.6}$$

Now the trajectory system differential equation is

$$\frac{d\underline{x}_T^*(F)}{dF} = \dot{\underline{x}}'(F) = \sum_{i=1}^{r} \dot{t}_i(F) \, k_i e^{A(T-t_i(F))} \underline{b}_i \tag{6.2.7}$$

But since it is also true that

$$\underline{y}_T^*(F) = \alpha(F) \, \underline{x}_T^*(F) \tag{6.2.8}$$

where $\alpha(F)$ is some unknown function, it can be shown that

$$|\langle \underline{x}'(F), \underline{v}_i(F) \rangle| = \frac{1}{\alpha(F)}, \quad i=1, \ldots, r \tag{6.2.9}$$

where

$$\underline{v}_i(F) = k_i e^{A(T-t_i(F))} \underline{b}_i \quad . \tag{6.2.10}$$

With no fuel consumed and a run time T,

$$\underline{x}(T) = e^{AT} \underline{x}_0 \quad , \tag{6.2.11}$$

so in the trajectory system

$$\underline{x}'(0) = e^{AT} \underline{x}_0 \quad . \tag{6.2.12}$$

Now the locus of optimal final states cannot be found because $\alpha(F)$ is not known. This type of extension of the trajectory system method would be a good area for further research.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Athans, M. and P. L. Falb, Optimal Control, McGraw-Hill Book Co., New York, (1966).

2. Barr, R. O., Computation of Optimal Controls by Quadratic Programming on Convex Reachable Sets, Ph.D. thesis, University of Michigan, (1966).

3. Barr, R. O., and E. G. Gilbert, "Some Efficient Algorithms for a Class of Abstract Optimization Problems Arising in Optimal Control," IEEE Trans. on Auto. Cont., AC-14, pp. 640-652, (1969).

4. Bellman, R., Dynamic Programming, Princeton University Press, Princeton, N. J., (1957).

5. Bryson, A. E., and W. F. Denham, "A Steepest-Ascent Method for Solving Optimum Programming Problems," J. Appl. Mech. Ser. E, V. 29, (1962), pp. 247-257.

6. Fancher, P. S., "Iterative Computation Procedures for an Optimum Control Problem," IEEE Trans. on Automatic Control, AC-10, pp. 346-348, (1965).

7. Gilbert, E. E., "An Iterative Procedure for Computing the Minimum of a Quadratic Form on a Convex Set," SIAM J. on Control, Ser. A, V. 4, No. 1, pp. 61-80, (1966).

8. Hestenes, M. R., Calculus of Variations and Optimal Control Theory, John Wiley and Sons, Inc., New York, (1967).

9. Ho, Y. E., "A Successive Approximation Technique for Optimal Control Systems Subject to Input Satruation," Trans. ASME, J. Basic Eng., Series D, V. 82, (1960), pp. 33-40.

10. Lee, E. B., and L. Markus, Foundations of Optimal Control Theory, John Wiley and Sons, Inc., New York (1967).

11. Neustadt, L. W., "Synthesizing Time Optimum Control Systems," J. Math. Anal. and Appl., V. 1, pp. 484-492, (1960).

12. Pontryagin, L. S., V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mischenko, The Mathematical Theory of Optimal Processes, John Wiley and Sons, Inc., New York, (1962).

13. Stratton, R. B., Computation of Optimal Controls for Nonlinear Systems Via Geometric Search, Ph.D. thesis, Michigan State University, (1969).

APPENDIX I

APPENDIX I

THE DOUBLE-INTEGRATOR PLANT

The double-integrator plant is so named because it is represented by the differential equation

$$\ddot{x} = u \tag{I-1}$$

This equation may arise, for instance, in describing the reaction of a unit mass when acted upon by a force or thrust u. Such a system with bounded input can be described by the state equations

$$\dot{\underline{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \underline{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{I-2}$$

where u is admissible.

For optimal time or optimal regulator control of this system $u = +1$ or $u = -1$ are the only two input values of interest. For $u = +1$, the state trajectories are parabolas opening along the positive x-axis; for $u = -1$, they are parabolas opening along the negative x-axis. Each of these two sets of trajectories has one member which intersects the origin. These two trajectories are shown in Figure I.1 with the direction of motion shown by the arrows.

The half of each curve for which the state is moving toward the origin is in solid line, while the other half is in dashed line. The two solid half trajectories together form the minimum time switch curve. Any initial state can be forced to the origin in minimum time by letting $u = \pm 1$ — whichever is correct for that initial state — until the switch curve is reached, and then switching u and following the switch curve in to the origin.

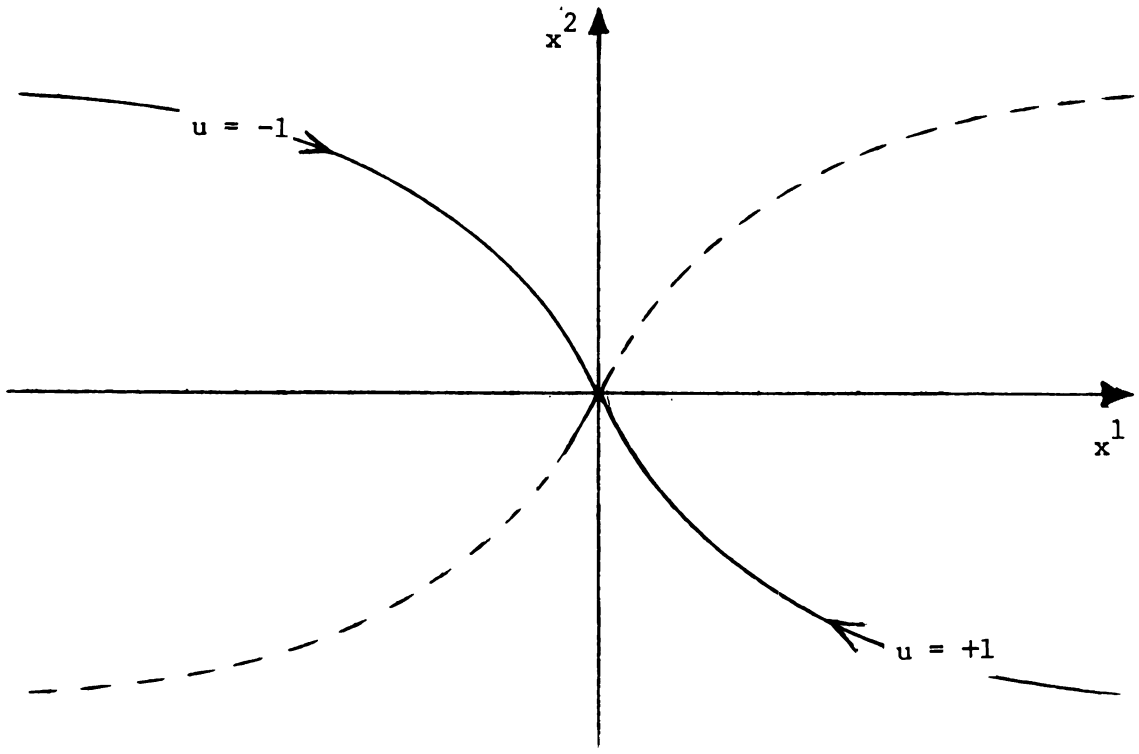

Figure I.1. Trajectories for the Double-Integrator Plant

The problem of forcing the system state as close as possible to the origin in a fixed run time is more complicated. To analyze this problem, it is only necessary to consider initial states above the switch curve in Figure I.1, because the trajectories for u = +1 and u = -1 are symmetrical about the origin. For the rest of this discussion initial states are assumed to be above the switch curve. For an initial state in the upper half-plane, the input is originally u = -1. For short enough run times, the control never switches. When there is sufficient run time to carry the state into the lower half-plane, switching does occur. The longer the run time available, the later the switch occurs and thus the farther the state penetrates into the lower half-plane on the u = -1 trajectory before switching occurs. The limit occurs at T^* the optimal time when switching occurs at the switch curve and the state reaches the origin. This is an example of

a Type I switch because switching first occurs at the end of the trajectory. The situation is shown in Figure I.2 where the switch curve and the set of optimal final states are shown as solid lines and the trajectories as dashed lines.

For an initial state in the lower half-plane, the input is $u = +1$ throughout for short run times and it is $u = -1$ originally and then $u = +1$ for longer run times. This is demonstrated in Figure I.3. Again the switch curve and the set of optimal final states are shown as solid lines while the trajectories are shown as dashed lines. This is an example of a Type II switch because switching first occurs at the beginning of the trajectory.

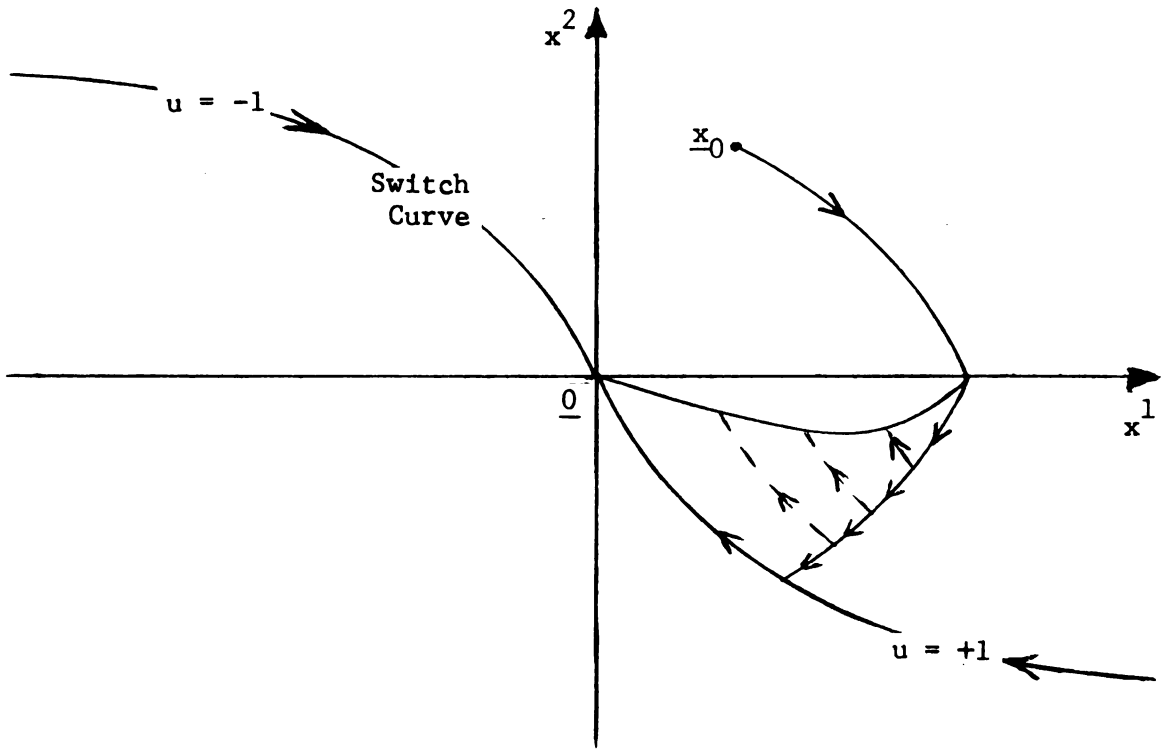

Figure I.2. Solutions of the Regulator Problem with a Type I Switch

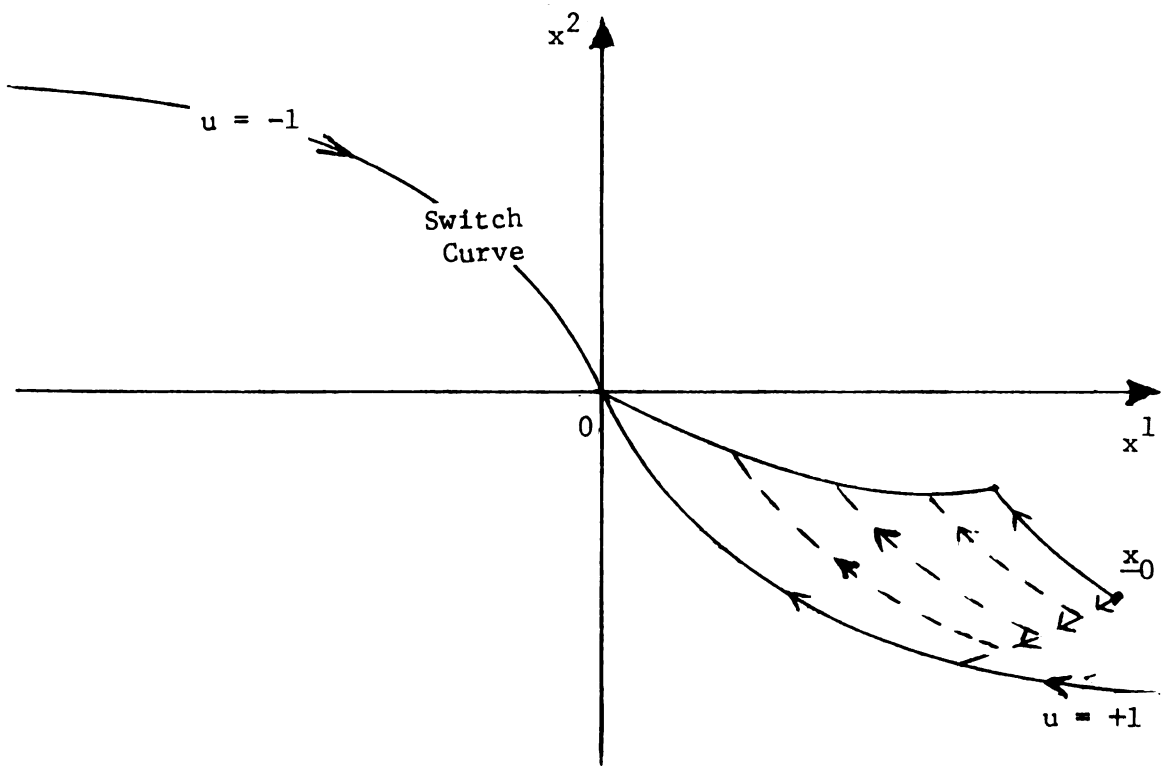

Figure I.3. Solutions of the Regulator Problem with a Type II Switch

APPENDIX II

APPENDIX II

The Fortran IV program presented at the end of this appendix uses Algorithm 3.3.1 and either Method 1 or Method 2 to solve both Problem 1 and Problem 2. The program given here can only be used for a system with a scalar input that switches at most twice. It would not be a very difficult matter to change the program so that it can handle any size control vector and any number of switches. It is also necessary that the B matrix be constant.

The name of the program is OPTML; the various subroutines are names PROB, REV, RENEW, SCALE, SWPLN, CHANG, END, DOT and SGN. The user supplies data cards and parts of PROB and REV. The first data card must have an integer number in column 1 which is the order of the system, N. The next N cards contain the elements of the vector $-2\underline{b}$; after that, each set of N cards is taken to be an initial state $\underline{x}_0$. These entries must each be in E format and right justified to column 15.

In PROB the user supplies the system equations. There are six groups of equations which are, in order: $\dot{\underline{x}} = A\underline{x} + \underline{b}y$, $\underline{z}(T + \Delta T) = \underline{z}(T) + \Delta T\, A\underline{z}$, $\underline{v}(T + \Delta T)$, Hyp $= -\underline{b}^T\underline{x}$, $\underline{w}$, and alt. Alt is an expression to be used in computing $y = u_e^*(T)$ when $\underline{x}(T)$ is an element of the switch hyperplane. The vector $\underline{w}$ is an element of the switch hyperplane. The vector $\underline{w}$ is used in computing the correction factor for $\dot{\tau}(T)$. Using Method 2, $\underline{v} = \underline{w} + \Delta T(1 - \dot{\tau})A\underline{w}$ and $\underline{w} = \frac{1}{s} A\underline{v}$. Using Method 1, $\underline{v} = 2e^{-As}\underline{b}$ and $\underline{w} = A\ \underline{vqn}$.

The subroutine REV is only necessary when the system is nonlinear. Using the vector A as the final value of the state and the costate, both sets of equations are run backwards in time an amount TINT. The

step size ΔT = DELT is available. At the end, the value of $\underline{x}$ is put in the location B. The example of PROB shown is for the double-integrator plant, and the example of REV is for the nonlinear system of the example in section 4.4.

Table II.1. Fortran IV Program

```
// FOR OPTML
*NONPROCESS PROGRAM
*IOCS(CARD,1443PRINTER,DISK)
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      ODIMENSION X(9),VO(9),XQ(9),V(9),VQ(9),DX(9),XQN(9),VQN(9),
     1                    Z(9),ZQ(9)
       COMMON                        T,DELT,TAU,TAUQ,DTAU,DTAUQ,N,AK
   51 FORMAT(1X,E15.8,E15.8,I2)
  101 FORMAT (1X,5X,1HT,12X,3HTAU,11X,4HX(1),10X,4HX(2),10X,4HX(3),
     1        10X,4HDTAU,/)
  102 FORMAT(1X,6(F9.5,5X))
  103 FORMAT( E15.8)
  104 FORMAT(I1)
  105 FORMAT (1X,29HTHE MINIMUM TIME SOLUTION IS ,F7.3,9H SECONDS.)
      READ (2,104) N
      IF(N) 17,17,199
  199 DO 201 ID=1,N
  201 READ (2,103) VO(ID)
   70 DO 200 ID=1,N
      Z(ID)=VO(ID)
      V(ID)=VO(ID)
  200 READ (2,103) X (ID)
      DONE=0.1
      DELT=0.05
      EPS=0.01
      IT=1
      I=0
      J=0
      K=0
      L=0
      NOT=1
      IEND=1
      AK=0.
```

Table II.1. (cont'd.)

```
      START=1.
      IF (DONE) 12,71,71
   71 T=0.
      TAU=0.
      DTAU=0.5
      WRITE(3,101)
    6 CONTINUE
      WRITE (3,102) T,TAU,X(1),X(2),X(3),XNORM
      NOT=NOT+1
      COMP=DOT(X,Z,N)
      DTAUQ=DTAU
      XNORM=SQRT(DOT(X,X,N))
      TEST=XNORM-DONE
      IF(TEST) 2, 2,1
    2 CONTINUE
      CALL END(X,IEND,    I,J,NEXT)
      IF(NEXT) 1,1,4
    4 WRITE (3,105) T
      GO TO 12
    1 CALL DATSW(5,MN)
      IF(MN-1) 12,12,18
   18 CONTINUE
      M=4
      CALL PROB (M,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP ,ALT)
      Y=SGN(HYP)
      IF(Y) 5,3,5
    3 CONTINUE
      CALL SWPLN       (IT,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP,ALT,START,NOT,J,
     1              L)
    5 CONTINUE
      CALL RENEW       (V,VQ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP,ALT,VQMAG,XQMAG,
     1         XQ)
      CHECK=DOT(XQN,VQN,N)
      IF(START) 39,14,39
   14 IF(CHECK)7,39,7
```

Table II.1. (cont'd.)

```
    7 STORE=SGN(CHECK) *CHECK
      TEST=STORE-EPS
      IF(TEST)39,39,11
   39 TEMP=DOT(XQ,ZQ,N)
      IF(J) 10,40,40
   40 NEXT=1
      CALL SCALE(TEMP,COMP,J,NEXT,START,I,IT,L, NOT,V,DX,VQ,ZQ)
      IF(NEXT)  1,10,31
   10 IF(I) 30,20,20
   20 CONTINUE
      M=4
      CALL PROB (M,V,DX,XQ,ZQ,Z,VQN,XQN,S,Y,HYPQ,ALT)
      NEXT=0
      CALL SCALE(HYPQ,Y,I,NEXT,START,J,IT,L, NOT,V,DX,VQ,ZQ)
      IF(NEXT) 1,30,31
   30 T=T+DELT
   31 TAU=TAUQ*AK*SGN(AK) + T*START
      R=(DTAU+DTAUQ)/2.
      DTAU=DTAUQ
      DO 747 ID=1,N
      X(ID)=XQ(ID)
      V(ID)=VQ(ID)
  747 Z(ID)=ZQ(ID)
      DONE=DELT*SQRT(DOT(DX,DX,N))
      GO TO 6
   11 CONTINUE
      CALL CHANG       (NOT,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP ,TEMP,VQMAG,
     1          XQMAG,ALT)
      GO TO 1
   12 CONTINUE
   13 CALL DATSW(10,MN)
      IF(MN-1) 70,17,70
   17 CONTINUE
      CALL EXIT
      END
```

Table II.1. (cont'd.)

```
// FOR
*NONPROCESS PROGRAM
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBRUUTINE PROB(I,V,DX,X,ZQ,Z,VQN,W  ,S,Y,HYP,ALT)
      DIMENSION     W(9),B(9), V(9),DX(9),Z(9),X(9),ZQ(9),VQN(9)
      COMMON                        T,DELT,TAU,TAUQ,DTAU,DTAUQ,N,AK
      B(1)=0.
      GO TO (1,2,3,4,5,6),I
    1 CONTINUE
      DX(1)=X(2)+V(1)*((DTAU+DTAUQ)/2.) *AK
      DX(2)= Y+V(2)*((DTAU+DTAUQ)/2.) *AK
      RETURN
    2 CONTINUE
      ZQ(1)=Z(1)+DELT*Z(2)
      ZQ(2)=Z(2)
      RETURN
    3 CONTINUE
      V(1)=W(1)+DELT*(1.-((DTAU+DTAUQ)/2.))*W(2)
      V(2)=W(2)
      RETURN
    4 CONTINUE
      HYP=-X(2)
      RETURN
    5 CONTINUE
      W(1)=V(2)/S
      W(2)=0.
      RETURN
    6 CONTINUE
      ALT= X(1)
      RETURN
      END
```

Table II.1. (cont'd.)

```
// FOR
*NONPROCESS PROGRAM
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE REV (A,B,TINT)
      DIMENSION A(9),B(9),X(9),P(9),XQ(9),PQ(9)
      COMMON                        T,DELT,TAU,TAUQ,DTAU,DTAUQ,N,AK
      DO 1 I=1,N
      X(I)=A(I)
    1 P(I)=A(I)
      S=0.
    2 CONTINUE
      IF(S-TINT) 3,5,5
    3 XQ(1)=X(1)-DELT*X(2)
      XQ(2)=X(2)-DELT*X(2)*X(2)*SGN(X(2))-DELT*SGN(P(2))
      PQ(1)=P(1)
      PQ(2)=P(2)-DELT*(-P(1)+2.*X(2)*SGN(X(2))*P(2))
      DO 4 I=1,N
      X(I)=XQ(I)
    4 P(I)=PQ(I)
      S=S+DELT
      GO TO 2
    5 DO 6 I=1,N
    6 B(I)=X(I)
      RETURN
      END
```

Table II.1. (cont'd.)

```
// FOR
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE RENEW(V,VQ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP,ALT,VQMAG,XQMAG,
     1        XQ)
      DIMENSION V(1),VQ(1),DX(1),X(1),ZQ(1),Z(1),VQN(1),XQN(1) ,XQ(1)
      COMMON                        T,DELT,TAU,TAUQ,DTAU,DTAUQ,N,AK
      M=1
      CALL PROB (M,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP ,ALT)
      DO 205 ID=1,N
  205 XQ(ID)=X(ID)+DELT*(DX(ID)       )
      M=2
      CALL PROB (M,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP ,ALT)
      XQMAG=SQRT(DOT(XQ,XQ,N))
      TAUQ=TAU+DELT*(DTAU+DTAUQ)/2.
      S=TAUQ-T-DELT
      M=3
      CALL PROB (M,VQ,DX,X,ZQ,Z,VQN,V   ,S,Y,HYP ,ALT)
      VQMAG=SQRT(DOT(VQ,VQ,N))
      DO 209 ID=1,N
      XQN(ID)=XQ(ID)/XQMAG
  209 VQN(ID)=VQ(ID)/VQMAG
      RETURN
      END
```

Table II.1. (cont'd.)

```
// FOR
*NONPROCESS PROGRAM
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE SCALE(A,B,J,NEXT,START,I,IT, L,NOT,V,DX,VQ,ZQ)
      DIMENSION V(1),DX(1),VQ(1),ZQ(1)
      COMMON                        T,DELT,TAU,TAUQ,DTAU,DTAUQ,N,AK
   50 FORMAT(1X,40HTHE SUBROUTINE HAS FOUND THE IMPOSSIBLE.)
   51 FORMAT(1X,E15.8,E15.8,I2)
  106 FORMAT(1X,/,22HCONTROL SWITCH NUMBER ,I1,16H HAS BEEN FOUND.,/)
      IF(J-1) 4,1,13
    1 IF(ABS(A)-.00001) 8,8,2
    2 C2=A
    3 BETA3=BETA1+(BETA2-BETA1)*ABS(C1/(C1-C2))
      BETA1=BETA2
      BETA2=BETA3
      C1=C2
      DELT=DELT*BETA2/BETA1
      J=1
      IT=2
      I=-1
      GO TO 14
    4 IF(A) 5,8,6
    5 IF(B) 15,52,7
    6 IF(B) 7,52,15
    7 BETA1=0.
      BETA2=1.
      C1=B
      C2=A
      GO TO 3
```

Table II.1. (cont'd.)

```
      8 START=0.
        AK=SGN(DOT(V,DX,N))
        NOT=-1
        DTAUQ=0.5
        L=L+1
        WRITE (3,106) L
        IF(NEXT) 52,11,9
      9 TAUQ=0.
        DO 10 K=1,N
     10 VQ(K)=ZQ(K)
        GO TO 12
     11 TAUQ=T+DELT
     12 T=T+DELT
        DELT=DELT*(1.-BETA2)/BETA2
        J=2
        GO TO 16
     13 J=0
        I=0
        T=T+DELT
        DELT=DELT/(1.-BETA2)
     16 NEXT=1
        RETURN
     14 NEXT=-1
        RETURN
     15 NEXT=0
        RETURN
     52 WRITE (3,50)
        WRITE (3,51) A,B,J
        RETURN
        END
```

Table II.1. (cont'd.)

```
// FOR
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE SWPLN(IT,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP,ALT,START,NOT,J,
     1             L)
      DIMENSION V(1),DX(1),X(1),ZQ(1),Z(1),VQN(1),XQN(1),W(9)
      COMMON                          T,DELT,TAU,TAUQ,DTAU,DTAUQ,N,AK
  106 FORMAT(1X,/,22HCONTROL SWITCH NUMBER ,I1,16H HAS BEEN FOUND.,/)
      M=6
      CALL PROB (M,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP ,ALT)
      Y=SGN(ALT)
      IF(Y)16, 5,16
   16 GO TO (15, 5),IT
   15 CONTINUE
      Y=-Y
      M=1
      CALL PROB (M,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP ,ALT)
      DO 214 ID=1,N
       W(ID)=X(ID)
  214 X(ID)= W(ID)-DELT*DX(ID)
      M=1
      CALL PROB (M,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP ,ALT)
      Y=-Y
      DO 216 ID=1,N
  216 X(ID)= W(ID)
      AK=SGN(DOT(V,DX,N))
      L=L+1
      WRITE (3,106) L
      START=0.
      NOT=0
      TAU=T
      DTAU=0.5
```

Table II.1. (cont'd.)

```
      DTAUQ=DTAU
      J=-1
      IT=2
    5 CONTINUE
      RETURN
      END
```

```
// FOR
*NONPROCESS PROGRAM
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE CHANG(NOT,V ,DX,X,ZQ,Z,VQN,XQN,S,Y,HYP ,TEMP,VQMAG,
     1          XQMAG,ALT)
      DIMENSION V(1),DX(1),X(1),ZQ(1),Z(1),VQN(1),XQN(1) ,W(9)
      COMMON                        T,DELT,TAU,TAUQ,DTAU,DTAUQ,N,AK
      M=5
      CALL PROB (M,V ,DX,X,ZQ,Z,VQN,W,VQMAG,Y,HYP,ALT)
      TEMP=DOT(XQN,W,N)-(DOT(V,VQN,N)/XQMAG)*AK
      COR=2.*(DOT(XQN,VQN,N)/(DELT*TEMP))
      IF(NOT) 60,60,61
   60 DTAUQ=DTAUQ+COR/2.
      DTAU=DTAUQ
      RETURN
   61 DTAUQ=DTAUQ+COR
      RETURN
      END
```

Table II.1. (cont'd.)

```
// FOR
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE END(X,IEND,    I,J,NEXT)
      DIMENSION X(1)
      COMMON                      T,DELT,TAU,TAUO,DTAU,DTAUO,N,AK
      TEMP=1.
      DO 212 ID=1,N
  212 TEMP=TEMP*X(ID)
      GO TO (4,8),IEND
    4 NAVE=SGN(TEMP)
      NX=SGN(X(1))
      I=-1
      J=-1
      DELT=DELT/10.
      IEND=2
   20 NEXT=0
      RETURN
    8 NEST=SGN(TEMP)
      IF (NEST-NAVE) 9,19,9
   19 MX=SGN(X(1))
      IF (MX-NX) 9,20,9
    9 DELT=DELT*10.
      I=0
      J=0
      IEND=1
      T   =T-DELT/10.
      NEXT=1
      RETURN
      END
```


Table II.1. (cont'd.)

```
// FOR
*NONPROCESS PROGRAM
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      FUNCTION DOT(A,B,N)
      DIMENSION A(1),B(1)
      DOT=0.
      DO 1 I=1,N
    1 DOT=DOT     +A(I)*B(I)
      RETURN
      END
```



```
// FOR
*NONPROCESS PROGRAM
*LIST SOURCE PROGRAM
*ONE WORD INTEGERS
      FUNCTION SGN(X)
      IF(X) 1,2,3
    1 SGN=-1.
      GO TO 4
    2 SGN=0.
      GO TO 4
    3 SGN=1.
    4 RETURN
      END
```