DETERMINATION OF BANG-BANG CONTROLS FOR LARGE NONLINEAR SYSTEMS

by

DAVID DONALD NIEMANN

B.S., Kansas State University, 1986

A THESIS

submitted in partial fulfillment of the

requirements for the degree

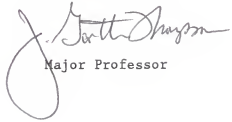
MASTER OF SCIENCE

MECHANICAL ENGINEERING

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

Approved by:



Major Professor

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ACKNOWLEDGEMENTS

I would like to thank the people and organizations which have contributed to this thesis. In particular, Dr. Warren White Jr. gave a great deal of time and contributed many ideas to this project. Dr. J. Garth Thompson, Dr. Chi-Lung Huang, and Dr. Ruth Dyer served on my committee. The Honor Society of Phi Kappa Phi, the Department of Mechanical Engineering at Kansas State University, and my parents provided financial support.

CHAPTER I

INTRODUCTION

Bang-Bang Control and the Time Minimization Problem

A control scheme in which control inputs are off, at maximum positive effort, or at maximum negative effort is termed a bang-bang control. Common examples of bang-bang controls are spacecraft maneuvering thrusters and residential heating systems. In these examples, bang-bang control is used to simplify the system hardware; however, the primary importance of bang-bang control lies in optimal control theory.

Before continuing, a few matters of notation need to be addressed. First, in this paper upper-case letters denote vector and matrix quantities. The dimensions of a particular matrix or vector are stated in the text when the matrix or vector is defined. Lower-case letters represent scalar quantities.

Second, at some points in the text the arguments of functions are dropped for notational convenience. In particular, the state and control vectors are always functions of time even though they are often denoted by a single letter.

Optimal control theory seeks to determine the control inputs and state trajectories for a physical system which minimize some performance measure. A particular optimal control problem is the minimum-time

transfer of a system from an initial state vector $X(t_0)$ to a desired final state vector $X(t_f)$. If the control vector U is constrained by upper and lower bounds, then a fundamental theorem due to Pontryagin establishes that the optimal control inputs for the minimum-time problem must be bang-bang unless the optimal trajectory contains singular intervals. Singular intervals are discussed in Chapter Two. Here it suffices to say that usually singular intervals will not exist, and the control trajectory will be strictly bang-bang. Pontryagin's minimum principle is discussed in standard texts on optimal control such as Kirk [1] and Bryson and Ho [2]. If one or more components of the control vector lies on a constraint boundary during the entire control interval, the system is called proper. If all the components of the control vector are bang-bang during the entire control interval, the system is called normal. In this paper, all systems will be assumed normal.

If the optimal control inputs are always on a constraint boundary, the time-optimal problem is reduced to finding the correct times where the controls switch from one boundary to another. This paper develops a method of iteratively adjusting the switching times to transfer the system from some initial state to a desired final state. The systems considered have the form

$$\dot{X} = G(X(t)) + F(X(t))U(t). \tag{1.1}$$

Here X is an n x 1 column vector of state variables, G(X(t)) is an n x 1 vector-valued function of the state variables, U(t) is an m x 1 vector of controls and F(X(t)) is an n x m matrix function of the states.

Previous Work

Theoretical Development

Much of the theoretical development of time-optimal control theory took place during the late 1950's. Papers by Bushaw [3], Bellman [4], LaSalle [5], and others established the basic result that the time-optimal control for a normal system is bang-bang and can be expressed as a function of the state and adjoint variables. Pontryagin [6] presented a set of necessary conditions for the minimization of functionals with constrained state and control variables. Desoer [7] demonstrated that the same basic results could be derived using variational calculus.

Other authors developed analytic solutions for the controls in terms of the state variables. These solutions involve the use of switching surfaces in the state space. The control vector switches each time the state trajectory intersects a switching surface. Oldenburger and Thompson [8] presented a general technique for determining switching surfaces in the state space of second-order, linear, time-invariant systems. The method was also demonstrated for third-order systems with one control input and for second-order systems with two control inputs. The switching surface approach is very desirable because the solution forms the basis for a time-optimal feedback controller. Unfortunately, it is not currently possible to find equations for the switching surfaces of large nonlinear systems such as robots. Instead, one must settle for determination of time-optimal trajectories between specified initial and final points in the state space.

Numerical Techniques

In general, an iterative solution must be used to determine the time-optimal state and control trajectories. There are three basic numerical techniques that have been developed: (1) minimization of a discretized problem, (2) iteration on the initial values of the adjoint equations, and (3) iteration on the switching times of the controls. Within each of these catagories a variety of minimization techniques have been used.

Many researchers have attempted to determine the time-optimal state and control trajectories by minimizing a discretized version of the problem. The techniques used vary widely and are often very problem specific. Shetty [9] has developed a finite-element approach. Time is discretized and the state and adjoint variables are treated as nodal unknowns at each time instant. The transversality equation is used as a boundary condition at the initial and final time. This method produces close agreement with results obtained by other techniques. However, the method is sensitive to the initial estimate on t_f. Subrahmanyam [10] has presented another general treatment of the discrete problem. In this technique, time is discretized and interpolation is used to approximate the state and control trajectories. A recursive formula is developed to calculate the time-optimal solution.

Several papers have been presented on time-optimal path planning for robot arms. These methods all involve constructing trial trajectories in the configuration space of the robot and iteratively adjusting these trajectories to minimize travel time. Shin and

McKay [11] presented a technique which converts the control bounds into bounds on the position, velocity, acceleration, and jerk of the system. Total travel time is minimized subject to these constraints.

Sahar and Hollerbach [12] developed a linear programming approach in which a grid search is conducted in joint space. Motion between grid points follows a straight line and is time optimal along that line. Dynamics scaling properties of the equations of motion greatly simplify the problem by limiting the grid search to configuration space instead of the entire state space. Even with this simplification, the technique is very computationally expensive. Rajan [13] used cubic splines and adjusted the spline parameters to minimize final time.

These approaches produce fast manipulator motion and easily incorporate path constraints; however, they suffer from a lack of theoretical underpinnings. The control trajectories obtained using these techniques are usually not bang-bang. This fact suggests that faster trajectories exist for these problems.

The second catagory of techniques for computing the time-optimal control involves iteration on the initial values of the adjoint variables. The standard optimal control formulation results in a two-point boundary-value problem involving the n state equations and n adjoint equations. However, the boundary-value problem associated with the time minimization problem is difficult to solve because no boundary values are known for the adjoint variables. Therefore, the initial values of the adjoint variables must be guessed and iteratively adjusted to obtain the correct solution. Knudsen [14] developed a Newton-Raphson iteration method for linear stationary systems. He examined the

relationship between the initial state values $X(t_0)$, the initial adjoint values $\Lambda(t_0)$, and the final time t_f. The problem is viewed as a mapping $X(t_0) = \Phi(\Lambda(t_0), t_f)$. Since $X(t_0)$ is known and $\Lambda(t_0)$ and t_f are unknown, the iterative procedure must produce the inverse mapping $\Phi^{-1}(\Lambda(t_0), t_f)$. Knudsen determined that this inverse mapping is one to one if the mapping does not fall onto a region of the switching hypersurface which is also an optimal trajectory. Lastman [15] has developed a technique applicable to both linear and nonlinear systems. Initial guesses are selected for $\Lambda(t_0)$ and t_f and the system is integrated forward in time. Newton's method is used to accurately determine the values of the switching times during each iteration. Lasdon, Mitter, and Waren [16] have taken a similar approach using the conjugate gradient method, and Lewing and Thorp [17] have used a second-variation technique.

These methods will converge quickly if the initial guesses for the adjoint values $\Lambda(t_0)$ and the final time t_f are close to the true solution. If the initial guesses are too far off, a unique relationship between these variables and the initial states $X(t_0)$ does not exist. Under these circumstances, the algthm will not converge.

The last catagory of techniques for the solution of time-optimal problems involves iteratively adjusting the switching times of the controls. In this approach, the time-optimal control is assumed to be bang-bang. This assumption simplifies the problem to determining the correct switching times for the controls. Larson [18] developed a method of sucessively adjusting the switching times using Newton-Raphson

iteration. The equations of motion are integrated using Picard's method of successive approximation. Yastreboff [19] presented a variation of this technique for linear systems with real eigenvalues. For these problems, the maximum number of switches is known to be n-1 for an nth order system. The controls are assumed constant during the interval between two switches, and an initial guess is placed on the switching times. The magnitude of the controls is determined so that the system will reach the desired final state. Finally, the switching times are adjusted to minimize the difference between the magnitudes of the controls during different intervals. When the controls' magnitudes during every interval become the same, the algorithm has converged to the time-optimal solution. Davison and Monro [20] developed a method of adjusting the switching times without calculating partial derivatives. A technique called Rosenbrock's method was used to minimize the error at the final state. The authors reported results which were accurate to at least five significant figures. More recently, Wen and Desrochers [21] have presented a technique for switching time optimization using the gradient method. Their technique forms the basis for the developments in this work, and will be dicussed in Chapter Two.

Switching-time optimization techniques are much less sensitive to initial guesses than are techniques which optimize the initial value of the adjoint variables. However, these methods can fail if the correct number of switching times and the correct initial controls are not used in the initial guess. If too few switching times are selected, the algorithm will not drive the final state error to zero. If too many switching times are used, the resulting solution may have zero cost but

it will not necessarily be time optimal. This condition is termed chattering.

Overview

This paper extends the work of Wen and Desrochers by developing an algorithm which can handle problems in which the initial values of the controls and the number of switches are not known. The technique developed combines switching-time iteration with the gradient projection technique used in dynamic programming. The result is a robust algorithm which can handle a wide variety of systems.

Chapter Two develops the theory of switching-time optimization using the gradient method. First, the necessary conditions for a bang-bang optimal control solution are derived. Second, the cost function is presented and the gradient equations in switching-time space are developed. Third, the numeric methods used to implement the switching-time iteration method are discussed. In particular, the Davidon-Fletcher-Powell method is presented and a double integrator example is used to illustrate the procedure. Finally, the limitations of this algorithm are discussed.

Chapter Three extends the switching-time iteration method developed in Chapter Two to systems in which the initial control vector and the number of switches required are not initially known. The problem is viewed as a constrained minimization in switching time space. Constraints are enforced using the gradient projection method. The double integrator example is used again to illustrate this procedure.

Chapter Four presents a method of converting a set of n second-order differential equations derived using Lagrange's equations into a set of 2n first-order differential equations. Using this conversion, the constrained switching-time iteration method can be applied to any system derived from a Lagrangian function.

Chapter Five presents a series of example problems which illustrate the constrained switching-time iteration method The examples cover a wide variety of systems, and offer a comparison between results obtained using this method and results obtained from other procedures.

Chapter Six presents the conclusions reached during this research, and provides recommendations for further work in this area.

CHAPTER II

THE SWITCHING-TIME ITERATION METHOD

Necessary Conditions for a Bang-Bang Optimal Control

This chapter develops a method for computing the bang-bang control which transfers a system from some initial state $X(t_o)$ to some final state $X(t_f)$. The type of system considered is given by equation (1.1), namely,

$$\dot{X}(t) = G(X(t)) + F(X(t))U(t) \tag{1.1}$$

where $\dot{X}(t)$ = n x 1 column vector of state derivatives,
$G(X(t))$ = n x 1 vector function of the states,
$F(X(t))$ = n x m matrix function of the states,
and $U(t)$ = m x 1 vector of the unknown controls.

This choice of state equations is not very restrictive. The only requirements are that the system is linear with respect to the control inputs and that the functions G and F do not depend explicitly on time. These restrictions are met by most dynamic systems. Before examining the switching-time iteration method, it is instructive to examine equation (1.1) in terms of Pontryagin's Minimum Principle. Necessary conditions for a bang-bang time-optimal control will be developed.

Given a system defined as in equation (1.1) the Hamiltonian for the system is defined as

$$H(X,\Lambda,U) = y(X(t),U(t)) + \Lambda^T(G(X(t)) + F(X(t))U(t)). \tag{2.1}$$

here $y(X(t),U(t))$ is the integrand of the scalar cost function, $J = \int_0^{t_f} y(X(\tau),U(\tau))d\tau$, which is to be minimized, and Λ is the n x 1 column vector of adjoint states. For the minimum-time problem, the integrand of the cost function is simply $y(X(t),U(t)) = 1$. The Hamiltonian is therefore

$$H(X,\Lambda,U) = 1 + \Lambda^T(G(X(t)) + F(X(t))U(t)). \tag{2.2}$$

The control vector U(t) is also constrained by the inequalities

$$lb_i \leq u_i \leq ub_i, \quad i = 1,\ldots,m. \tag{2.3}$$

where u_i is the ith component of U and lb_i and ub_i are the respective lower and upper bounds of u_i. Pontryagin's minimum principle provides a set of necessary conditions for the minimization of J subject to the constraints on the controls. These are

$$H(X^*,\Lambda^*,U^*) \leq H(X^*,\Lambda^*,U), \tag{2.4.a}$$

$$\dot{X}^* = \frac{\partial H}{\partial \Lambda}(X^*,\Lambda^*,U^*), \tag{2.4.b}$$

$$\dot{\Lambda}^* = -\frac{\partial H}{\partial X}(X^*,\Lambda^*,U^*), \tag{2.4.c}$$

and $$0 = y(X^*(t_f),U^*(t_f)) + \Lambda^{*T}(t_f)[G(X^*(t_f)) + F(X^*(t_f))U^*(t_f)]. \tag{2.4.d}$$

Here, an asterisk superscript on a function indicates that the function minimizes J. Additionally, if the final time is free and the Hamiltonian is not an explicit function of time, then

$$0 = y(X^*(t),U^*(t)) + \Lambda^{*T}(t)[G(X^*(t)) + F(X^*(t))U^*(t)]. \tag{2.4.e}$$

Equation (2.4.a) indicates that, out of the class of admissible control histories, the optimal control U^* will, at the optimal trajectory, cause the Hamiltonian to assume a minimum with respect to all admissible variations of the control U. In terms of the minimum-time problem expressed in equation (2.2), the condition (2.4.a) becomes

$$1 + \Lambda^{*T}(G(X^*(t)) + F(X^*(t))U^*(t)) \leq$$

$$1 + \Lambda^{*T}(G(X^*(t)) + F(X^*(t))U(t)). \tag{2.5}$$

Simplifying this equation by eliminating the terms not explicitly dependent on U gives

$$\Lambda^{*T}F(X^*(t))U^*(t)) \leq \Lambda^{*T}F(X^*(t))U(t). \tag{2.6}$$

Let f_{ij} denote the element of F located in row i and column j. Expanding the product $F(X^*(t))U^*(t)$ produces

$$\Lambda^{*T}\begin{bmatrix} f_{11}(X^*)u_1^* + f_{12}(X^*)u_2^* + \ldots + f_{1m}(X^*)u_m^* \\ \vdots \\ f_{n1}(X^*)u_1^* + f_{n2}(X^*)u_2^* + \ldots + f_{nm}(X^*)u_m^* \end{bmatrix} \leq$$

$$\Lambda^{*T}\begin{bmatrix} f_{11}(X^*)u_1 + f_{12}(X^*)u_2 + \ldots + f_{1m}(X^*)u_m \\ \vdots \\ f_{n1}(X^*)u_1 + f_{n2}(X^*)u_2 + \ldots + f_{nm}(X^*)u_m \end{bmatrix}. \tag{2.7}$$

Moving all terms to the left and letting λ_i denote the ith component of Λ, equation (2.7) becomes

$$\sum_{j=1}^{m}\sum_{i=1}^{n} \lambda_i^* f_{ij}(X^*)[u_j^* - u_j] \leq 0 \tag{2.8}$$

with the constraints $lb_i \leq u_i \leq ub_i$, $i = 1,\ldots,n$.

Because the control components are linearly independent, equation (2.8) implies that each component in the summation over j must independently satisfy the inequality. Therefore,

$$\sum_{i=1}^{n} \lambda_i^* f_{ij}(X^*)[u_j^* - u_j] \leq 0, \quad j = 1,\ldots,m. \tag{2.9}$$

Each u_j can be any function of time which satisfies the corresponding constraint in equations (2.3). For a particular u_j, consider a time interval (t_k, t_{k+1}) during which the corresponding quantity $\sum_{i=1}^{n} \lambda_i^* f_{ij}(X^*)$ does not change sign. There are three possible cases. These are

$$\text{Case 1:} \quad \sum_{i=1}^{n} \lambda_i^* f_{ij}(X^*) > 0 \tag{2.10.a}$$

$$\text{Case 2:} \quad \sum_{i=1}^{n} \lambda_i^* f_{ij}(X^*) < 0 \tag{2.10.b}$$

$$\text{Case 3:} \quad \sum_{i=1}^{n} \lambda_i^* f_{ij}(X^*) = 0 \tag{2.10.c}$$

If case one occurs, then (2.9) is satisfied only if $(u_j^* - u_j) \leq 0$. Therefore, u_j^* must equal lb_j during this interval. If case two occurs, then (2.9) is satisfied only if $(u_j^* - u_j) \geq 0$. Therefore, u_j^* must equal ub_j during this interval. If case three occurs then (2.9) is satisfied automatically and no information is obtained about the optimal control component u_j^*. When case three occurs, the problem is said to be singular.

Using the preceeding development, the definitions of normal and proper systems given in Chapter One can be restated. A system is said to be proper if at least one function $\sum_{i=1}^{n} \lambda_i^* f_{ij}(X^*)$ is not equal to zero during any finite interval in $[0,t_f]$. A system is said to be normal if every function $\sum_{i=1}^{n} \lambda_i^* f_{ij}(X^*)$ has a finite number of zeros on the interval $[0,t_f]$.

The conditions developed in this section provide a criteria for evaluating time-optimal problems to determine if the optimal control vector will be bang-bang. While these conditions may not be easy to determine for a general system, they do offer guidelines to test if the bang-bang assumption is valid for particular problems. Intuitively, one would expect that the singular case is more the exception than the rule and that in most cases the time-optimal control will indeed be bang-bang. Therefore, the basic assumption of switching-time iteration techniques, that every component of U(t) is on a constraint boundary except at a finite number of switching instants during the interval $[0,t_f]$, is usually valid.

The Cost and Gradient Functions

Wen and Deschrochers [21] have developed an algorithm for switching-time iteration based on gradient minimization. The controls are assumed to be bang-bang, and the initial locations of the switches are arbitrarily picked. Any prior knowledge about the form of the

solution is used when selecting the switch locations. Denote the desired final state vector as X_d. Under most circumstances, the initial guess will not bring the system to X_d. Therefore, a cost function is used to measure the distance between the actual final state $X(t_f)$ and the X_d. By determining the partial derivatives of the cost with respect to the switching times, the switching times can be adjusted to minimize the state error at the final time.

Denote the first switching time by t_1, the second switching time by t_2, and the ith switching time by t_i. The cost function is given by

$$C(t_1,t_2,\ldots,t_r,t_f) = \frac{1}{2}(X(t_f) - X_d)^T(X(t_f) - X_d). \qquad (2.11)$$

Note that the cost is a function of the final time t_f and the switching times, $t_1,t_2,\ldots,t_r$ where r is the number of assumed switching times. The final time and the r switching times can be considered a point in an (r+1)-dimensional space. The object of this algorithm is to locate a point in this switching-time space where $C(t_1,t_2,\ldots,t_r,t_f) = 0$. Given an initial guess on the switching times and the final time, a new set of switching times which reduces the cost can be found by moving in the negative gradient direction. The partial derivative of the cost function with respect to the ith switching time is given by

$$\frac{\partial C}{\partial t_i} = (X(t_f) - X_d)^T \frac{\partial X}{\partial t_i}(t_f); \qquad i = 1,\ldots,r. \qquad (2.12)$$

The problem now becomes one of finding each $\frac{\partial X}{\partial t_i}(t_f)$. We begin by integrating the equations of motion. This gives

$$X(t_f) = X(0) + \int_0^{t_f} G(X(\tau))d\tau + \int_0^{t_1} F(X(\tau))U_0 d\tau + \int_{t_1}^{t_2} F(X(\tau))U_1 d\tau$$

$$+ \ldots + \int_{t_r}^{t_f} F(X(\tau))U_r d\tau. \qquad (2.13)$$

Now consider the partial derviative of the state vector with respect to the ith switching time. For all $t < t_i$, $\frac{\partial X}{\partial t_i}(t) = 0$. For all $t > t_i$,

$$\frac{\partial X}{\partial t_i}(t) = F(X(t_i))(U_{i-1} - U_i) + \int_{t_i}^{t} \frac{\partial G}{\partial X}(X(\tau))\frac{\partial X}{\partial t_i}(s)d\tau$$

$$+ \int_{t_i}^{t_{i+1}} \sum_{j=1}^{n} \frac{\partial F}{\partial x_j}(X(\tau))\frac{\partial x_j}{\partial t_i}(\tau)U_i d\tau + \ldots + \int_{t_k}^{t} \sum_{j=1}^{n} \frac{\partial F}{\partial x_j}(X(\tau))\frac{\partial x_j}{\partial t_i}(\tau)U_k d\tau. \qquad (2.14)$$

where t_k is the last switching time occuring before time t. The n x m matrix $\frac{\partial F}{\partial x_j}$ represents the partial derivative of F with respect to the jth component of X. Taking the time derivative of this equation gives

$$\frac{\partial \dot{X}}{\partial t_i}(t) = \frac{\partial G}{\partial X}(X(t))\frac{\partial X}{\partial t_i}(t) + \sum_{j=1}^{n} \frac{\partial F}{\partial x_j}(X(t))\frac{\partial x_j}{\partial t_i}(t)U_k \qquad (2.15)$$

with initial condition $\frac{\partial X}{\partial t_i}(t_i) = F(X(t_i))(U_{i-1} - U_i)$.

Recall that for $t < t_i$, $\frac{\partial X}{\partial t_i}(t) = 0$. Therefore, $\frac{\partial \dot{X}}{\partial t_i}(t) = 0$ for all $t < t_i$. Because $\frac{\partial X}{\partial t_i}(t)$ is defined by an integral equation, it will be

continuous at every point in the interval $[0,t_f]$ except at the jump discontinuity occurring at t_i. A special case occurs for the partial derivatives of the states with respect to t_f. This partial derivative will be zero for all $t < t_f$. At t_f, the partial derivative is given by

$$\frac{\partial X}{\partial t_f}(t_f) = - F(X(t_f))U_i \tag{2.16}$$

The Gradient Search Algorithm

The basic gradient search algorithm is diagrammed in Figure 2.1. Define ns to be the number of switches during the control interval. Let T represent an (ns+1) vector composed of the individual switching times t_i, $i = 1,\ldots,ns$, and the final time t_f. This vector will be referred to as the switching-time vector. Let S represent an (ns+1) vector composed of integers s_i which specify the control component which switches at the corresponding t_i. This vector will be referred to as the control vector. The component s_f which corresponds to the final time is always assigned the value 0. For example, suppose a system has three control inputs. If control component u_1 switches at $t = 1$ and $t = 2$, u_2 switches at $t = 1$ and $t = 1.5$, u_3 switches at $t = 2.5$, and the final time is given by $t_f = 3$, then T and S are given by

$T = [\ 1\ \ 1\ \ 1.5\ \ 2\ \ 2.5\ \ 3\ ]^T$, and $S = [\ 1\ \ 2\ \ 2\ \ 1\ \ 3\ \ 0\ ]^T$.

The vectors T and S are used to numerically integrate equations (1.1) and (2.15). Equations (2.11) and (2.12) are used to calculate the cost

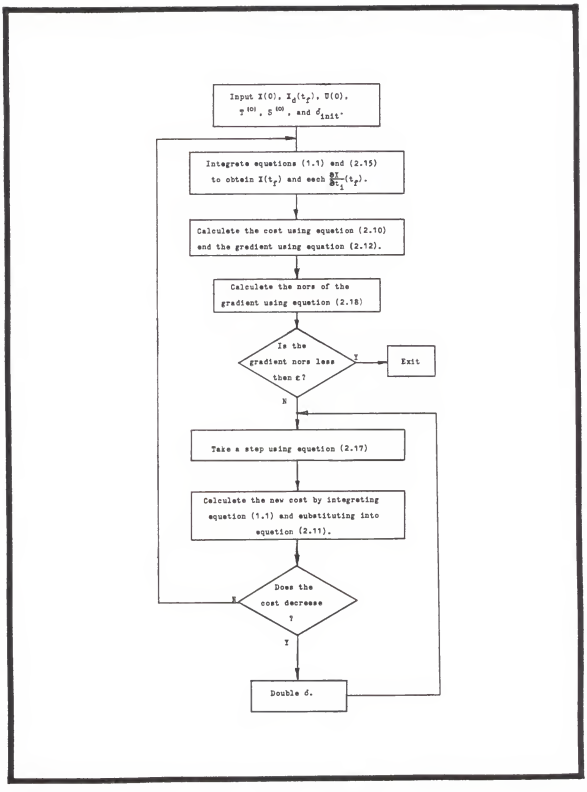


Figure 2.1 The basic gradient search algorithm

and gradient of the cost. The procedure begins by calculating the initial cost and the initial gradient of the cost. The update rule at the kth iteration is given by

$$T_{(k+1)} = T_{(k)} - \Theta \frac{\partial C}{\partial T}_{(k)} \tag{2.17}$$

In this equation, $\frac{\partial C}{\partial T}$ is an (ns+1) x 1 gradient vector, and Θ is an (ns+1) x (ns+1) matrix which determines the step size. The simplest form of Θ is given by $\Theta = \delta I$. Here δ is a scalar, and I is an (ns+1) x (ns+1) identity matrix. Algorithms which use this definition are called steepest-descent methods. Using the steepest-descent method, an improved switching-time vector is determined using a single-variable minimization of the cost as a function of δ. When a minimum is obtained, the switching-time vector is updated and a new gradient vector is calculated. The algorithm terminates when the norm of the gradient vector,

$$\left| \frac{\partial C}{\partial T} \right| = \left[\frac{\partial C}{\partial T}^T \frac{\partial C}{\partial T} \right]^{1/2}, \tag{2.18}$$

is less than some specified tolerance ϵ.

The algorithm of Figure 2.1 uses the steepest-descent method described in the last paragraph. The single-variable minimization is initialized with a small number for the initial guess on δ, and the cost is evaluated for this value of δ. If the cost decreases, δ is doubled, and the process is repeated. If the cost increases, the previous value of δ is used to determine $T_{(k+1)}$, a new gradient is calculated, and a new single-variable minimization begins.

However, the steepest-descent method caused the minimization algorithm to converge very slowly. Therefore, two changes were implemented to improve the convergence. First, the accuracy of the single-variable minimization was improved using the golden section method described by Siddall [22]. Second, a new definition of θ was implemented.

Other gradient minimization methods use different choices for θ which improve convergence. The basic idea is to modify the stepping direction in the single-variable minimization by using information about both the present gradient and the old gradients. The Davidon-Fletcher-Powell technique was selected for the switching-time iteration. This technique is described in Siddall [22]. In this procedure, θ is defined as

$$\theta = \delta_k H_k \tag{2.19}$$

δ_k is the scalar step size for the kth iteration, and H_k is an (ns+1) x (ns+1) matrix defined by

$$H_k = H_{k-1} - \frac{\delta_{k-1}\left(H_{k-1}\frac{\partial C}{\partial T}_{(k-1)}\right)\left(H_{k-1}\frac{\partial C}{\partial T}_{(k-1)}\right)^T}{\left(H_{k-1}\frac{\partial C}{\partial T}_{(k-1)}\right)^T\left(\frac{\partial C}{\partial T}_{(k)} - \frac{\partial C}{\partial T}_{(k-1)}\right)}$$

$$- \frac{H_{k-1}\left(\frac{\partial C}{\partial T}_{(k)} - \frac{\partial C}{\partial T}_{(k-1)}\right)\left(\frac{\partial C}{\partial T}_{(k)} - \frac{\partial C}{\partial T}_{(k-1)}\right)^T H_{k-1}}{\left(\frac{\partial C}{\partial T}_{(k)} - \frac{\partial C}{\partial T}_{(k-1)}\right)^T H_{k-1}\left(\frac{\partial C}{\partial T}_{(k)} - \frac{\partial C}{\partial T}_{(k-1)}\right)} \tag{2.20}$$

The derivation of this procedure is based on the concept of quadratic convergence. Given a cost function defined by $X^T CX$ where X is an n x 1

vector and C is an n x n matrix, an algorithm is said to converge quadratically if it is guaranteed to locate the minimum within n iterations. Note that the cost function used to mimimize the error at t_f is not quadratic; therefore, the algorithm is not guaranteed to converge in n steps. However, the results show that this choice of Θ does provide much faster convergence. A simple example will illustrate the procedure developed in this chapter.

Example 2.1 - Double Integrator

The equation of motion for a double integrator is $\frac{d^2x}{dt^2} = u$. Written in state-space form, this equation becomes

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u. \qquad (2.21)$$

The control is constrained by $-1 \leq u \leq 1$.

The problem is to find the state and control trajectories which will transfer the system from the initial conditions $x_1(0) = 1.0$, $x_2(0) = 1.0$ to the origin in minimum time using: (a) the steepest-descent method, and (b) the Davidon-Fletcher-Powell method.

Case (a)

Using the gradient method, Θ is given by $\Theta = \delta I$. The initial guess used was

$$T_{(0)} = [\ 10.\ 20.\ ]^T, \text{ and } S = [\ 1\ 0\ ]^T.$$

The switching-time vector converged in 512 iterations to

$$T_{(512)} = [\ 2.22474\ \ 3.44948\ ]^T.$$

Figure 2.2 illustrates how the switching-time vector varies with each iteration using the steepest-descent method.

Case (b)

Using the Davidon-Fletcher-Powell method, θ is given by equation (2.19). The initial guess used was the same as for case (a). The algorithm converged in twelve iterations to

$$T_{(12)} = [\ 2.22474\ \ 3.44948\ ]^T.$$

Figure 2.3 illustrates how the switching-time vector varies with each iteration using the Davidon-Fletcher-Powell method. Figure 2.4 illustrates the converged state and control trajectories.

The double integrator problem can be solved analytically. Oldenburger and Thompson [8] have derived the switching function for this problem. It is given by

$$x_1(t_s) = -\frac{1}{2}\, x_2(t_s)|x_2(t_s)|. \tag{2.23}$$

Here, t_s represents the switching time. The initial control is determined by

$$u_0 = \begin{cases} +1 \text{ if } x_1(t_s) - \frac{1}{2}\, x_2(t_s)|x_2(t_s)| < 0 \\ -1 \text{ if } x_1(t_s) - \frac{1}{2}\, x_2(t_s)|x_2(t_s)| > 0 \end{cases}. \tag{2.24}$$

For initial conditions $x_1(0) = 1$, $x_2(0) = 1$, and control constraints $-1 \le u \le 1$, u_0 equals -1. Integrating the state equations produces

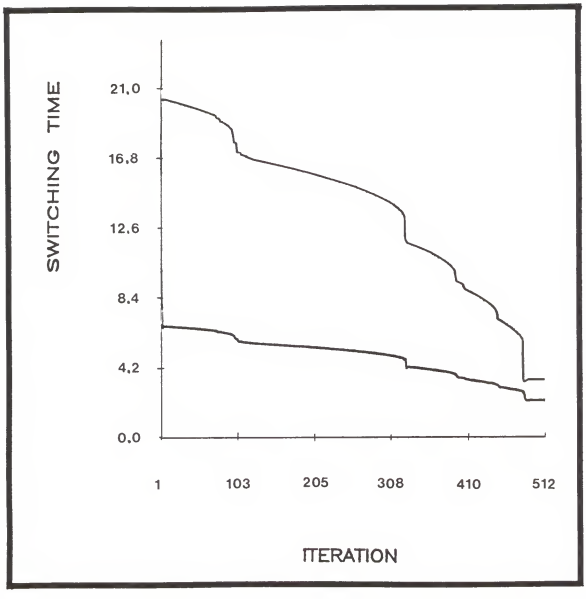


Figure 2.2 Switching-time vector at each iteration for the double integrator: steepest-descent method

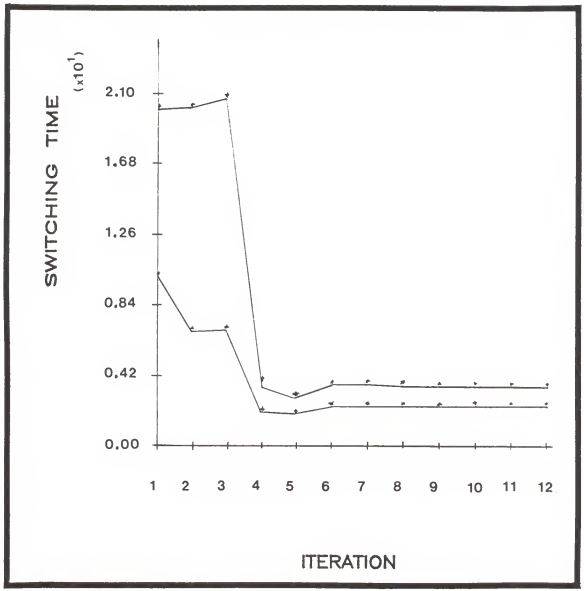


Figure 2.3 Switching-time vector at each iteration for the double integrator: Davidon-Fletcher-Powell Method

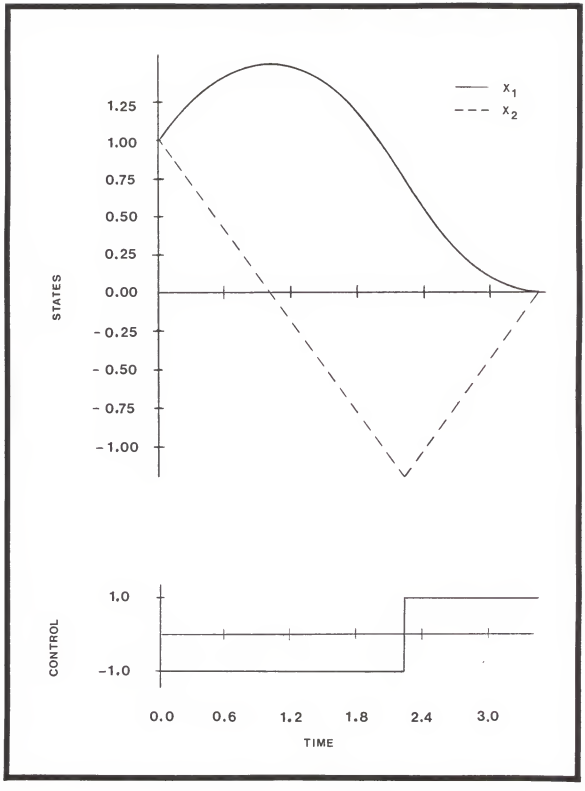


Figure 2.4 State and control trajectories for the double integrator

$$x_1(t) = 1 - t, \tag{2.25.a}$$

$$\text{and } x_2(t) = 1 - \frac{1}{2}t^2 + t \tag{2.25.b}$$

$$\text{for } 0 \le t \le t_s.$$

At the switching time, this state trajectory must intersect the switching curve. Substituting equations (2.25.a) and (2.25.b) into equation (2.23) produces

$$1 - \frac{1}{2}t_s^2 + t_s = \frac{1}{2}(1 - 2t_s + t_s^2). \tag{2.26}$$

Rearranging gives

$$0 = t_s^2 - 2t_s - \frac{1}{2}. \tag{2.27}$$

Solving for t_s produces

$$t_s = 2.2247449, \; -.2247449.$$

Eliminating the negative root, $t_s = 2.2247449$. The final time is found by integrating $x_2(t)$ from t_s to t_f.

$$x_2(t_f) = 0 = (1 - t_s) + \int_{t_s}^{t_f} 1 ds = (1 - t_s) + t_f - t_s \tag{2.28}$$

Solving for t_f produces

$$t_f = 2t_s - 1 = 3.44948.$$

These results confirm the numerical solution already obtained.

Limitations of the Switching-Time Iteration Method

In general, the algorithm developed in this chapter works well when the form of the solution can be determined in advance. Often, an initial guess can be found from physical reasoning, or from a linearized

version of the problem. However, problems occur when the initial guess is far from the correct solution. Experience gained working with the algorithm has shown that it will not converge if the initial quantities are picked incorrectly. Often switching times become negative or move past the final time. Wen and Desrochers [21] reported similar problems. They suggest that switching times which move out of the interval $[0,t_f]$ should be eliminated from the problem. However, they do not offer a systematic procedure for making these modifications, nor do they offer any theory concerning why these problems occur or how to correct them. Chapter Three presents an extension to the switching-time iteration method which eliminates these problems.

CHAPTER III

CONSTRAINED SWITCHING-TIME ITERATION

Constraining The Switching Times

This chapter presents a systematic approach to switching-time iteration when both the initial controls and the number of switches are unknown. The minimization technique developed in Chapter Two is reformulated by imposing constraint conditions on the switching-time space. These constraints are shown to be linear and simple in form. The problem becomes one of minimizing the distance between the actual and desired final states subject to the constraint conditions imposed upon the switching times. The minimization is accomplished using the gradient projection algorithm developed by Rosen [23] and detailed by Kirk [1]. While this technique adds complexity to the algorithm, it does not increase the number of variables which need to be minimized as does the Lagrangian multiplier method. In fact, Rosen has shown that the multiplier values can be obtained from the gradient projection algorithm.

In Chapter Two, the switching-time vector T and the control vector S were introduced. Each element of T contains a switching time with the first switch in t_1, the second switch in t_2, and so on. The last element of T contains the final time t_f. The corresponding elements of the vector S contain the integer values of the control which

is to be switched. A zero in the final element of S is used to designate the final time. The switching-time vector is updated using the algorithm developed in Chapter Two. As long as the initial guess on the switching-time vector is sufficiently close to the actual solution, the vector of initial conditions is correct, and the control vector S is correct, the changes in the switching times during each iteration will be small, and the algorithm will converge correctly. If this is not the case, the algorithm will produce answers which violate physical reality. Three situations can occur: the values of switching times can become negative, the values of switching times can become greater than the final time, or two switches can cross over each other. All these situations are characterized by the fact that the integration routine must integrate backward in time during some interval. Figure 3.1 illustrates an example of this problem.

Suppose that a system has one control input and that at some point in the iteration the T and S vectors are given by

$$T = [\ .1 \quad .3 \quad .4 \quad .6 \quad .7\ ]^T, \text{ and } S = [\ 1 \quad 1 \quad 1 \quad 1 \quad 0\ ]^T.$$

The routine will integrate along the path shown in Figure 3.1.a. Because the switching times are ordered correctly, the routine will produce correct values for the cost and the gradient. Now suppose that the single-variable minimization produces a new T given by

$$T = [\ -.1 \quad .4 \quad .2 \quad .65 \quad .6\ ]^T.$$

S does not change. If this new T is used in the integration routine, the integration will follow the path shown in Figure 3.1.b. Because the switching times are no longer arranged in ascending order, the routine

will integrate backward in time during the intervals where the arrows in Figure 3.1.b point to the left.

Initially, one may be tempted to simply sort the switching times into ascending order before integration. However, the sorting process destroys the geometric concept of stepping in the negative gradient direction, and the single variable minimization becomes a rather random process.

This problem can be rectified by redefining the switching time vector. Let nsl equal the number of switches of control one, ns2 equals the number of switches of control two, and nsm be the number of switches of control m. Now let the first nsl elements of T contain the switching times for control one stored in ascending order. The ns2 switching times for the second control are stored next, and the process is continued until the switches for the mth control are stored. The final time is stored last. The switch time and control vectors now appear as

$$T^T = [t_{11}, t_{12}, \ldots, t_{1nsl}, t_{21}, \ldots, t_{2ns2}, \ldots, t_{ml}, \ldots, t_{mnsm}, t_f]^T \quad (3.1.a)$$

$$\text{and } S^T = [1,1,\ldots,1,2,\ldots,2,\ldots,m,\ldots,m,0]^T. \quad (3.1.b)$$

For a particular control r, each switching time is constrained by $0 \le t_{ri-1} \le t_{ri} \le t_{ri+1} \le t_f$. These equations give nsl+1 constraints for the first control component, ns2+1 constraints for the second control component, and nsm+1 constraints for the mth control component. Written in matrix form, the constraint equations are given by

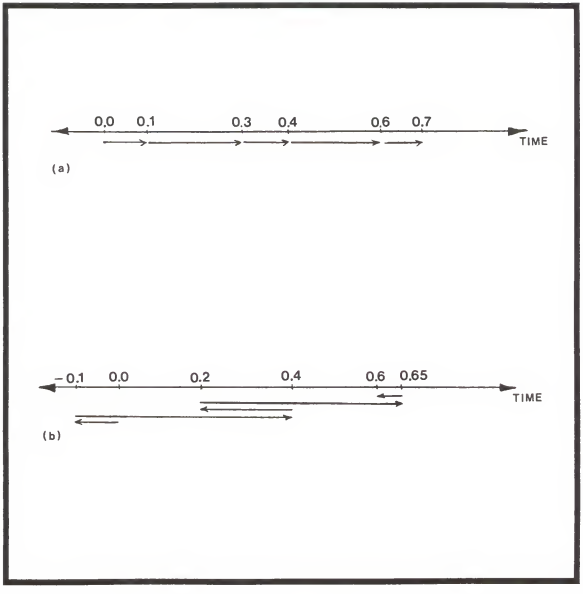


Figure 3.1 Correct and incorrect integration paths

$$\begin{bmatrix} -1 & 0 & 0 & & & \cdots & & 0 \\ 1 & -1 & 0 & 0 & & & & \\ 0 & 1 & -1 & 0 & 0 & & & \\ & & & \vdots & & \ddots & & \vdots \\ \cdots & 0 & 1 & -1 & 0 & & & \\ & 0 & 0 & 1 & 0 & \cdots & 0 & -1 \\ & & 0 & 0 & -1 & 0 & \cdots & 0 \\ & & 0 & 0 & 1 & -1 & \cdots & \\ & & & & \vdots & & & \\ & & & & 0 & 1 & -1 & 0 \\ & & & & & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{12} \\ \vdots \\ t_{1nsl} \\ t_{21} \\ t_{22} \\ \vdots \\ t_f \end{bmatrix} \leq 0 \qquad (3.2)$$

Normalizing the constraint equations produces

$$\begin{bmatrix} -1 & 0 & 0 & & & \cdots & & 0 \\ .707 & -.707 & 0 & 0 & & & & \\ 0 & .707 & -.707 & 0 & 0 & & & \\ & & & \vdots & & \ddots & & \vdots \\ \cdots & 0 & .707 & -.707 & 0 & & & \\ & 0 & 0 & .707 & 0 & \cdots & 0 & -.707 \\ & & 0 & 0 & -1 & 0 & \cdots & 0 \\ & & 0 & 0 & .707 & -.707 & \cdots & \\ & & & & \vdots & & & \\ & & & & 0 & .707 & -.707 & 0 \\ & & & \cdots & & 0 & 0 & .707 & -.707 \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{12} \\ \vdots \\ t_{1nsl} \\ t_{21} \\ t_{22} \\ \vdots \\ t_f \end{bmatrix} \leq 0 \qquad (3.3)$$

Each row of the constant matrix in equation (3.3) represents a unit vector normal to a hyperplane in the switching-time space. Each of these unit vectors points outward from the admissible region. Denoting this matrix as B^T, equation (3.3) can be written compactly as $B^T T \leq 0$.

The constraint surfaces for a system with one control and two switches are illustrated in Figure 3.2.

The minimization problem now has become one of finding the point in the switching-time space which minimizes the cost function subject to the constraints (3.3). This minimization can be accomplished using the gradient projection algorithm.

Minimization Using the Gradient Projection Method

There are many techniques available for the minimization of functions subject to linear constraints. The gradient projection method was chosen for this work because it fit in well with the algorithm developed in Chapter Two and did not increase the number of unknowns by the inclusion of Lagrangian multipliers. Kirk [1] presented a derivation of this procedure and gave an algorithm for the method. Rosen [23] offered a rigorous derivation and produced several theorems relating to the method.

The idea behind the gradient projection algorithm is that if a point lies on the intersection of k linearly independent constraint boundaries, then other points which have lower cost can be found by moving in the direction $-P\frac{\partial C}{\partial T}$ where P is an (ns+1) x (ns+1) projection matrix which projects the gradient vector into an (ns+1)-k dimensional subspace of the switching-time space. In geometric terms, one can think of stepping along the component of the negative gradient which is parallel to the constraint surface. The algorithm proceeds by stepping along the negative gradient projection until a minimum is encountered or a constraint boundary is reached. The gradient is recalculated at this

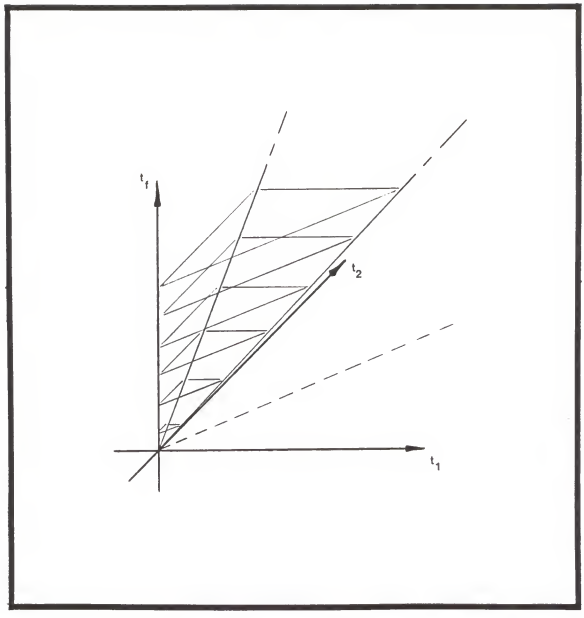


Figure 3.2 Constraint surfaces for a system with one control and two switches

new point and the gradient projection onto the new set of constraint boundaries is determined. The process continues until a constrained minimum is reached.

The Projection Matrix

At any particular step of the minimization, the trial point will only lie on some, if any, of the constraint boundaries. Constraint boundaries which contain the trial point are said to be active. Recall that each row of the matrix B^T consists of an outward facing unit vector normal to a constraint surface. Therefore each unit vector is represented by a column of B. Now define the (ns+1) x k matrix B_a which contains only the columns of B which correspond to active constraints. Suppose that at the ith iteration a point $T_{(i)}$ lies on k constraint boundaries. We wish to determine a new point $T_{(i+1)}$ which also lies on these constraint boundaries. Using the gradient projection to determine the stepping direction, the new point will be given by

$$T_{(i+1)} = T_{(i)} - \delta P \frac{\partial C}{\partial T} \tag{3.4}$$

where δ represents a scalar step magnitude. If this new point also lies on the same constraints, then it must satisfy the equation

$$B_a^T T_{(i+1)} = 0. \tag{3.5}$$

Substituting equation (3.4) into equation (3.5) gives

$$B_a^T (T_{(i)} - \delta P \frac{\partial C}{\partial T}) = 0. \tag{3.6}$$

The original assumption that $T_{(i)}$ lies on the k constraint boundaries implies that $B_a^T T_{(i)} = 0$. Therefore, equation 3.6 can be simplified to

$$B_a^T \delta P \frac{\partial C}{\partial T} = 0. \tag{3.7}$$

Rosen [23] has shown that if P is defined as

$$P = I - B_a (B_a^T B_a)^{-1} B_a^T, \tag{3.8}$$

then equation (3.7) will be identically satisfied. This fact can be checked by substituting equation (3.8) into equation (3.7) to obtain

$$\delta (B_a^T - B_a^T B_a (B_a^T B_a)^{-1} B_a^T) \frac{\partial C}{\partial T} \stackrel{?}{=} 0. \tag{3.9}$$

Simplifying the left hand side of the equation produces

$$\delta (B_a^T - B_a^T B_a (B_a^T B_a)^{-1} B_a^T) \frac{\partial C}{\partial T} = \delta (B_a^T - B_a^T) \frac{\partial C}{\partial T} = \delta (0) \frac{\partial C}{\partial T} = 0. \tag{3.10}$$

Therefore, using this definition of P, each successive $T_{(i)}$ is guaranteed to satisfy the constraint equations.

The Modified Minimization Algorithm

Figure 3.3 is a flowchart of the constrained mimimization algorithm. Several additions to the original algorithm of Figure 2.1 are required to incorporate gradient projection. The algorithm must be initialized with a feasible value for $T_{(0)}$. Because the elements of T are grouped by control component, the individual t_i must be sorted into ascending order before numerically integrating the state and gradient equations. After integration, the switching times and the corresponding gradient components are resorted into their original order.

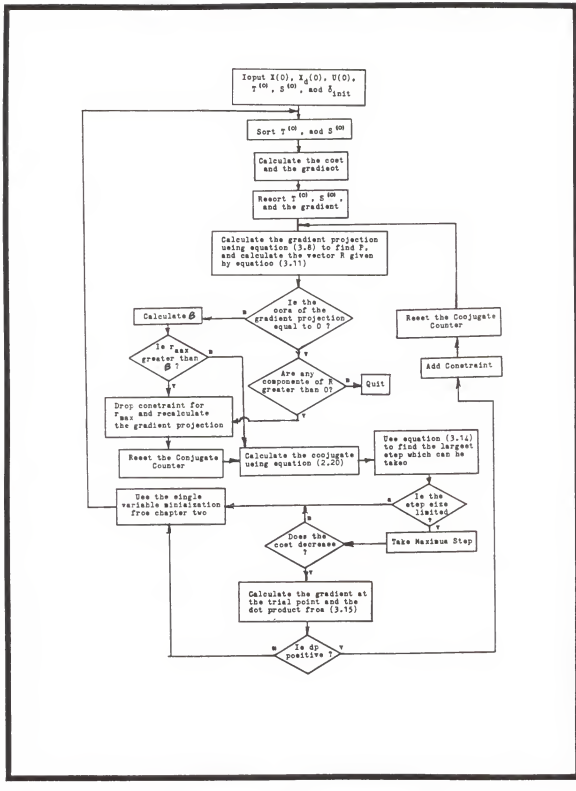


Figure 3.3 The constrained gradient search algorithm

During initialization, the program determines which constraint boundaries are active during the first iteration. This information is used to build B_a. The projection matrix is then calculated using equation (3.8), and the gradient projection is determined. Next, the algorithm checks to see if any constraints are unnecessary. Kirk [1] has outlined a procedure for making this test. The process begins by calculating a vector R given by

$$R = (B_a^T B_a)^{-1} B_a^T \left(\frac{\partial C}{\partial T}\right). \tag{3.11}$$

Note that if there are k active constraints, R is a k x 1 vector. This vector represents the portion of the gradient which is orthogonal to the intersection of the constraint surfaces. Each active constraint vector is one component of a usually nonorthogonal basis which spans this space. There are two cases to consider: (1) the norm of the gradient projection is zero, and (2) the norm of the gradient projection is nonzero.

If the norm of the gradient projection vector is zero and all the components of R are negative or zero, Rosen has proven that the point is a local constrained minimum. If some components of R are positive, the minimization can be continued by deactivating the constraint corresponding to the largest positive component of R.

If the norm of the gradient projection vector is not zero, dropping a constraint boundary may still be desirable. This situation occurs when the gradient vector points into the admissible region. In this case, stepping in the gradient vector direction will not violate the constraint; therefore, it can be deactivated. Kirk [1] has

described a test to determine if a constraint should be dropped when the gradient projection is not zero. The test consists of finding the quantity $\beta = \max\{\alpha_i\}$. The values α_i represent the sum of the absolute values of the elements of the ith row of the matrix $(B_a^T B_a)^{-1}$. If β is greater than or equal to the largest positive component of R, no constraint is deactivated. If β is less than the largest positive component of R, the corresponding constraint is deactivated. If a constraint is dropped, a new B_a must be formed, and a new projection matrix and gradient projection must be calculated.

If the active constraint boundaries have not changed since the last iteration, the conjugate gradient direction is calculated from the gradient projection. Note that the quantity $\frac{\partial C}{\partial T}_{(k)}$ used in equation (2.20) must be replaced by $P_k \frac{\partial C}{\partial T}_{(k)}$. If a boundary has been activated or deactivated since the last iteration, the counter for the conjugate routine is reset and the new gradient projection is used as the stepping direction. At this point, the routine is ready to begin the single variable minimization.

Because the space is constrained, the algorithm can only step until it reaches a constraint boundary. Therefore, the distance to each constraint must be calculated. At the kth iteration, the algorithm will step in the direction $-H_k P_k \frac{\partial C}{\partial T}_{(k)}$. Consider a line parallel to this vector which passes through the point $T_{(k)}$. Denote the position vector

of the point where this line intersects the ith inactive constraint by T_i^* and the distance from $T_{(k)}$ to T_i^* by δ_i^*. The vector T_i^* is given by

$$T_i^* = T_{(k)} - \delta_i^* H_k P_k \frac{\partial C}{\partial T}_{(k)} . \tag{3.12}$$

Now let N_i denote the normal vector for the ith constraint. Because every constraint surface passes through the origin, the dot product of T_i^* and N_i must equal zero. Therefore, dotting both sides of equation (3.12) with N_i gives

$$0 = N_i^T [T^{(k)} - \delta_i^* H_k P_k \frac{\partial C}{\partial T}_{(k)}]. \tag{3.13}$$

Rearranging this equation produces

$$\delta_i^* = N_i^T T_{(k)} \; / \; N_i^T H_k P_k \frac{\partial C}{\partial T}_{(k)} . \tag{3.14}$$

If the δ_i^* corresponding to each inactive constraint is negative, then the single-variable minimization is not constrained and the single-variable minimization technique used in Chapter Two is implemented. If some δ_i^* are positive, then the quantity δ_{max} is defined to be the minimum positive value of the δ_i^*. In this case, the routine takes a step of δ_{max} and checks to see if the cost is reduced. If the cost increases, the single-variable minimization technique used in Chapter Two is implemented. If the cost decreases, the partial derivative is calculated at this new point and the dot product of the stepping

direction with the new gradient is determined. This quantity is given by

$$dp = [H_k P_k \frac{\partial C}{\partial T}_{(k)}]^T \frac{\partial C}{\partial T}_{(k+1)} \tag{3.15}$$

If the dot product is negative, the minimum cost in the stepping direction is not at the constraint boundary, and a single-variable minimization must be conducted between 0 and δ_{max}. If the dot product is positive, the minimum cost in the stepping direction occurs at δ_{max}, and the constraint plane must be activated.

When the algorithm reaches this point in the minimization process, either the single-variable minimization has located a new trial point or a new constraint plane has been activated. The routine now loops back to the start and the process begins again. The routine converges when the norm of the gradient projection is less than some specified ϵ and all of the components of R are less than or equal to zero.

Example 3.1 - Double Integrator

The equations of motion for the double integrator are given in example 2.1. Again, the problem is to drive the system from the initial conditions $x_1(0) = 1$, $x_2(0) = 1$ to the origin in minimum time subject to the constraints $-1 \leq u \leq 1$. In example 2.1, the initial control was found to be $u(0) = -1$, and the solution converged to

$$T = [\ 2.22474 \quad 3.44948\ ]^T \text{ with } S = [\ 1 \quad 0\ ]^T.$$

In this example, the constrained minimization technique will be demonstrated when the initial guess is far from correct. Assume that the initial control is given by $u(0) = 1$, and that T and S are initially $T_{(0)} = [\ .3 \quad .6 \quad 1. \quad 1.5 \quad 2.0\ ]^T$, and $S = [\ 1 \quad 1 \quad 1 \quad 1 \quad 0\ ]^T$.

Using these initial guesses the switching-time vector converged in nine iterations to

$$T_{(9)} = [\ 0. \quad .84023 \quad .86103 \quad 2.26923 \quad 3.49687\ ]^T$$

Figure 3.4 shows the variation of switching-time vector with respect to each program iteration. Note that the sign of the initial control is effectively reversed by the switch at zero. Also the times of the second and third switches converge and almost cancel each other. The last switch corresponds to the true switching time. Although the solution obtained is not quit optimal, it is easy to see that the initial control should be changed to -1, and the first three switching times should be eliminated. Using this improved initial guess the optimal solution was obtained. Note that the difference between the converged solution and the true time optimal control is not due to the choice of the convergence tolerance ϵ. Rather, the error is due to chatter in the controls. Because the cost function just minimizes error at the final state, the chatter can only be eliminated by reducing the number of assumed switching times.

A more severe case of chatter occurs using the initial guess

$u(0) = 1$, $T_{(0)} = [\ 1. \quad 2. \quad 3. \quad 4. \quad 5.\ ]$, and $S = [\ 1 \quad 1 \quad 1 \quad 1 \quad 0\ ]$

Using these initial values the switching-time vector converged in eight iterations to

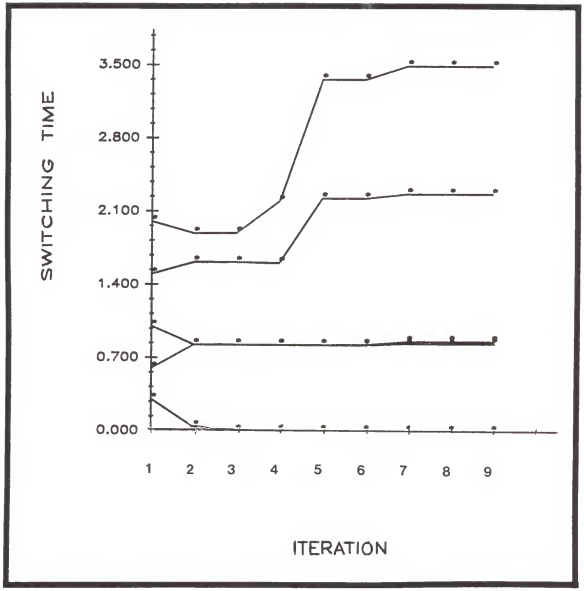


Figure 3.4 Switching-time vector at each iteration for the double integrator: constrained switching-time iteration

$$T_{(8)} = [\ .25533 \quad 2.52073 \quad 2.99064 \quad 3.61124 \quad 4.77202\ ]^T$$

The state vector does reach the origin, but the solution is obviously not time optimal. Figure 3.5 traces the components of the switching-time vector at each iteration for this case.

Presently, the only way to determine if a solution is truly time optimal is to integrate the adjoint equations using the state trajectories determined by the constrained switching-time minimization. If the zeros of equations (2.9) correspond to the minimized switching-time vector, the solution is time optimal. While this procedure is possible, it is not convenient for large systems, and a better technique would be desirable. Experience suggests that the final time will usually be very close to the minimum time if the initial guess on t_f is less than the true t_f. In practice, this algorithm has worked well for a variety of problems. Chapter Five contains several different examples and compares the answers found using the constrained switching-time iteration method to results obtained from other procedures.

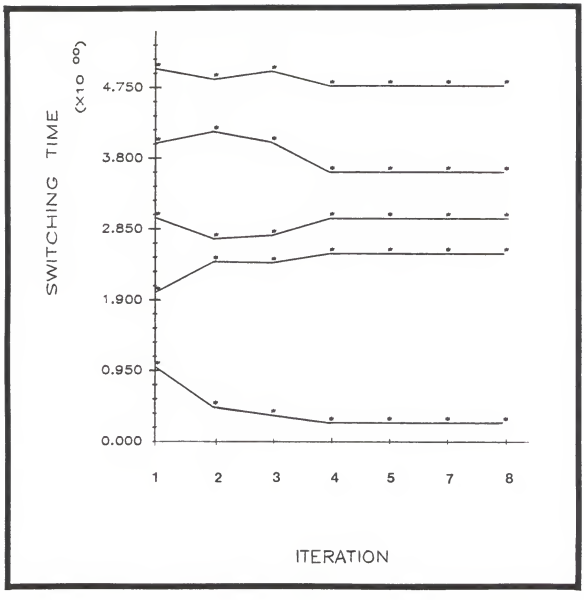


Figure 3.5 Switching-time vector at each iteration for the double integrator: chattering solution

CHAPTER IV

APPLICATION TO SYSTEMS DERIVED USING LAGRANGE'S EQUATIONS

Converting the Lagrangian Representation to a State Space Representation

The equations of motion for complicated dynamic systems are often derived using the Lagrangian formulation. For a system with n degrees of freedom, this approach leads to a system of n second-order equations. These n equations are referred to as Lagrange's equations. Lagrange's equations always have the form

$$\frac{d}{dt}[\, D(Q)\dot{Q} \,] - G(Q,\dot{Q}) = \Psi(t), \tag{4.1}$$

where

Q = an n x 1 generalized coordinate vector,

$\dot{Q}$ = an n x 1 generalized velocity vector,

$D(Q)$ = an n x n generalized mass matrix,

$G(Q,\dot{Q})$ = an n x 1 corriolis and potential force vector,

and $\Psi(t)$ = an n x 1 generalized force vector.

The equations in Chapters Two and Three were developed using a state space representation of the system. Therefore, the n second-order equations (4.1) must be converted into a set of 2n first-order equations.

The conversion is accomplished by defining the 2n x 1 state vector X to be

$$\begin{bmatrix} X \end{bmatrix} = \begin{bmatrix} Q \\ - - - \\ \dot{Q} \end{bmatrix} . \qquad (4.2)$$

The generalized force vector $\Psi(t)$ can also be redefined by the equation

$$\Psi(t) = F(Q,\dot{Q})U(t). \qquad (4.3)$$

Here F is an n x m matrix of gains, and U is an m x 1 vector of controls. In most cases, F will be a constant matrix. Rewriting equation (4.1) in terms of X, replacing $\Psi(t)$ with $F(Q,\dot{Q})U(t)$, and splitting the derivative term on the left-hand side of (4.1) produces

$$\begin{bmatrix} I & 0 \\ - - - & - - - \\ 0 & D \end{bmatrix} \dot{X} = \begin{bmatrix} 0 & I \\ - - - & - - - \\ 0 & -\dot{D} \end{bmatrix} X + \begin{bmatrix} 0 \\ - - - \\ G \end{bmatrix} + \begin{bmatrix} 0 \\ - - - \\ F \end{bmatrix} U . \qquad (4.4)$$

The conversion to a standard state space representation is completed by inverting the matrix on the left-hand side of equation (4.4). The resulting equation is

$$\begin{bmatrix} \dot{Q} \\ - - - \\ Q \end{bmatrix} = \begin{bmatrix} 0 & I \\ - - - & - - - \\ 0 & -D^{-1}\dot{D} \end{bmatrix} \begin{bmatrix} Q \\ - - - \\ \dot{Q} \end{bmatrix} + \begin{bmatrix} 0 \\ - - - \\ D^{-1}G \end{bmatrix} + \begin{bmatrix} 0 \\ - - - \\ D^{-1}F \end{bmatrix} U . \qquad (4.5)$$

Now define the matrices

$$G' = \begin{bmatrix} 0 & I \\ - - - & - - - \\ 0 & -D^{-1}\dot{D} \end{bmatrix} \begin{bmatrix} Q \\ - - - \\ \dot{Q} \end{bmatrix} + \begin{bmatrix} 0 \\ - - - \\ D^{-1}G \end{bmatrix} , \qquad (4.6)$$

and

$$F' = \begin{bmatrix} 0 \\ - - - \\ D^{-1}F \end{bmatrix} . \tag{4.7}$$

Equation (4.5) can now be rewritten as

$$\dot{X} = G' + F'U. \tag{4.8}$$

This equation has the same form as equation (1.1). Therefore, the equations derived in Chapter Two can be applied to (4.8), and the constrained switching-time alogorithm can be used to determine the time-optimal control. However, a few changes are required.

Computing the Inverse Generalized Mass Matrix and the Partial Derivatives of the Inverse Generalized Mass Matrix

Two modifications to the procedure outlined in Chapter Two must be made in order to implement constrained switching-time minimization using equation (4.8). First, equations (4.6) and (4.7) both contain the inverse of the generalized mass matrix. For very small problems such as example 5.4, the inverse can be computed analytically, but for large systems this matrix has to be inverted numerically at each time step. This change is fairly easy to implement in computer code, but it does slow the execution speed.

Second, the constrained switching time algorithm requires the calculation of the partial derivatives of G' and F' with respect to each element of the state vector X. This means that the partials of D^{-1} with respect to each state variable also have to be calculated. Direct

calculation of these partial derivatives is prohibitive for all but the smallest problems. Therefore a different technique is needed. Fortunately, a matrix identity can be used to simplify the task.

Beginning with the matrix D^{-1}, we write

$$D^{-1} = D^{-1} D D^{-1}. \tag{4.9}$$

Taking the partial derivative with respect to the ith element of X gives

$$\frac{\partial}{\partial x_i}(D^{-1}) = \frac{\partial}{\partial x_i}(D^{-1}) D D^{-1} + D^{-1} \frac{\partial}{\partial x_i}(D) D^{-1} + D^{-1} D \frac{\partial}{\partial x_i}(D^{-1}) \tag{4.10}$$

Because D is symmetic, D^{-1} and the partial derivatives of D and D^{-1} will also be symmetric. Using these facts and the fact that a symmetric matrix and its transpose are equal, the following simplifications can be made to (4.8)

$$\begin{aligned}
\frac{\partial}{\partial x_i}(D^{-1}) &= \frac{\partial}{\partial x_i}(D^{-1}) D D^{-1} + D^{-1} \frac{\partial}{\partial x_i}(D) D^{-1} + [D^{-1} D \frac{\partial}{\partial x_i}(D^{-1})]^T \\
&= \frac{\partial}{\partial x_i}(D^{-1}) D D^{-1} + D^{-1} \frac{\partial}{\partial x_i}(D) D^{-1} + [\frac{\partial}{\partial x_i}(D^{-1})]^T [D]^T [D^{-1}]^T \\
&= \frac{\partial}{\partial x_i}(D^{-1}) D D^{-1} + D^{-1} \frac{\partial}{\partial x_i}(D) D^{-1} + \frac{\partial}{\partial x_i}(D^{-1}) D D^{-1} \\
&= 2\frac{\partial}{\partial x_i}(D^{-1}) D D^{-1} + D^{-1} \frac{\partial}{\partial x_i}(D) D^{-1}.
\end{aligned} \tag{4.11}$$

Finally, moving the first term on the right side to the left and eliminating the product $D D^{-1}$, gives the relation

$$\frac{\partial}{\partial x_i}(D^{-1}) = -D^{-1} \frac{\partial}{\partial x_i}(D) D^{-1}. \tag{4.12}$$

This expression provides a method of calculating the partial derivative of D^{-1} without having to analytically calculate D^{-1} and then take the

partial derivative of each term. The matrix D^{-1} must be calculated numerically at each time step, but this imposes no additional computing overhead because the inverse matrix is already required for integration of the state equations. Computing the partial derivatives of D is a lengthy but manageable procedure which must be done by the user.

Computing the Partial Derivatives of the State Equations

Two cases occur when computing the partial derivatives of F' and G' with respect to the state variables. The first case corresponds to taking the partial derivative with respect to some generalized coordinate q_i. The second case corresponds to taking the partial derivative with respect to a generalized velocity $\dot{q}_i$.

Partial Derivatives With Respect to Elements of Q

The partial derivative of G' with respect to some q_i is given by

$$\frac{\partial G'}{\partial q_i} = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & -\frac{\partial}{\partial q_i}(D^{-1})\dot{D} - D^{-1}\frac{\partial}{\partial q_i}(\dot{D}) \end{array}\right] \left[\begin{array}{c} Q \\ \hline \dot{Q} \end{array}\right] + \left[\begin{array}{c} \frac{\partial \dot{Q}}{\partial q_i} \\ \hline -D^{-1}\dot{D}\frac{\partial \dot{Q}}{\partial q_i} \end{array}\right]$$

$$+\left[\begin{array}{c|c} 0 & \\ \hline \frac{\partial}{\partial q_i}(D^{-1})G & \end{array}\right] + \left[\begin{array}{c|c} 0 & \\ \hline D^{-1}\frac{\partial}{\partial q_i}(G) & \end{array}\right] . \qquad (4.13)$$

Substituting equation (4.12) into equation (4.13), and noting that the second term in equation (4.13) is a zero matrix produces

$$\frac{\partial G}{\partial q_i}' = \left[\begin{array}{c|c} 0 & I \\ \hline 0 & D^{-1}\frac{\partial}{\partial q_i}(D)D^{-1}\dot{D} - D^{-1}\frac{\partial}{\partial q_i}(\dot{D}) \end{array}\right] \left[\begin{array}{c} Q \\ \hline \dot{Q} \end{array}\right]$$

$$- \left[\begin{array}{c|c} 0 & \\ \hline D^{-1}\frac{\partial}{\partial q_i}(D)D^{-1}G & \end{array}\right] + \left[\begin{array}{c|c} 0 & \\ \hline D^{-1}\frac{\partial}{\partial q_i}(G) & \end{array}\right] . \qquad (4.14)$$

Expanding out $\dot{D}$ gives

$$\dot{D} = \sum_{j=1}^{n} \frac{\partial D}{\partial q_j}\dot{q}_j . \qquad (4.15)$$

The partial derivative of $\dot{D}$ with respect to some q_i is therefore given by

$$\frac{\partial}{\partial q_i}(\dot{D}) = \sum_{j=1}^{n} \frac{\partial^2 D}{\partial q_i \partial q_j}\dot{q}_j . \qquad (4.16)$$

In the minimization algorithm, equations (4.14), (4.15), and (4.16) are used to calculate the partial derivative of G' with respect to the generalized coordinates. Note that the user must provide expressions for each element in D, each element in the matrices $\frac{\partial D}{\partial q_i}$, and each element in the matrices $\frac{\partial^2 D}{\partial q_i \partial q_j}$, $i = 1, \ldots, n$, $j = 1, \ldots, n$.

Similarly, taking the partial derivative of F' with respect to some q_i and recalling equation (4.12) gives

$$\frac{\partial F'}{\partial q_i} = - \begin{bmatrix} 0 \\ \hline D^{-1}\frac{\partial}{\partial q_i}(D)D^{-1}F \end{bmatrix} + \begin{bmatrix} 0 \\ \hline D^{-1}\frac{\partial}{\partial q_i}(F) \end{bmatrix} . \qquad (4.17)$$

Partial Derivatives with Respect to Elements of $\dot{Q}$

The partial derivative of G' with respect to some $\dot{q}_i$ is given by

$$\frac{\partial G'}{\partial \dot{q}_i} = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & -\frac{\partial}{\partial \dot{q}_i}(D^{-1})\dot{D} - D^{-1}\frac{\partial}{\partial q_i}(\dot{D}) \end{array}\right] \left[\begin{array}{c} Q \\ \hline \dot{Q} \end{array}\right] + \left[\begin{array}{c} \frac{\partial \dot{Q}}{\partial q_i} \\ \hline -D^{-1}\dot{D}\frac{\partial \dot{Q}}{\partial \dot{q}_i} \end{array}\right]$$

$$+ \begin{bmatrix} 0 \\ \hline \frac{\partial}{\partial \dot{q}_i}(D^{-1})G \end{bmatrix} + \begin{bmatrix} 0 \\ \hline D^{-1}\frac{\partial}{\partial \dot{q}_i}(G) \end{bmatrix} . \tag{4.18}$$

Taking the partial derivative of $\dot{D}$ with respect to $\dot{q}_i$ produces

$$\frac{\partial \dot{D}}{\partial \dot{q}_i} = \frac{\partial D}{\partial q_i} . \tag{4.19}$$

Substituting equations (4.12) and (4.19) into equation (4.18) and recalling that D is only a function of variables in Q, equation (4.18) simplifies to

$$\frac{\partial G'}{\partial \dot{q}_i} = \begin{bmatrix} 0 & 0 \\ \hline 0 & -D^{-1}\frac{\partial D}{\partial q_i} \end{bmatrix} \begin{bmatrix} Q \\ \hline \dot{Q} \end{bmatrix} + \begin{bmatrix} \frac{\partial \dot{Q}}{\partial \dot{q}_i} \\ \hline -D^{-1}\dot{D}\frac{\partial \dot{Q}}{\partial \dot{q}_i} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ \hline D^{-1}\frac{\partial}{\partial \dot{q}_i}(G) \end{bmatrix} . \tag{4.20}$$

Following a similar procedure for the partial derivative of F' with respect to some $\dot{q}_i$ yields

$$\frac{\partial F'}{\partial q_i} = \begin{bmatrix} 0 \\ \hline D^{-1} \frac{\partial}{\partial \dot{q}_i} (F) \end{bmatrix} . \qquad (4.21)$$

Using equations (4.8), (4.14), (4.17), (4.20), and (4.21), the constrained switching-time iteration procedure can be applied to any set of differential equations obtained using Lagrange's equations.

CHAPTER V

PERFORMANCE OF THE CONSTRAINED SWITCHING-TIME ITERATION METHOD

Testing the Algorithm

Chapters Two and Three have developed an algorithm for determining the bang-bang control which will transfer a system from an intial state $X(0)$ to a final state $X(t_f)$. The double integrator problem has been used to illustrate the method. However, this example is a very simple single-input system, and may not reflect the performance of the algorithm for other problems. Therefore, the algorithm has been applied to four different examples.

The time-optimal control for an nth-order linear system with real roots is known to contain at most (n-1) switches. However, linear systems with complex roots can switch many times during the control interval. The performance of the constrained switching-time iteration method for a system with complex roots is examined using a harmonic oscillator. This system has a pair of complex poles which lie on the $j\omega$ axis. The solutions obtained are compared to analytic results.

Three problems are examined which test the algorithm on larger systems with multiple control inputs. Examples 5.2 and 5.4 are both fourth-order robotic systems with two control inputs. Example 5.3 is a sixth-order satellite system with three control inputs.

Example 5.1 - The Harmonic Ocsillator

The harmonic ocsillator is characterized by the equation $\frac{d^2y}{dt^2} + y = u(t)$. Written in state matrix form, this equation becomes

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \qquad (5.1)$$

The problem is to find the minimum time control which will drive this system (a) from an initial state $x_1(0) = 3$, $x_2(0) = 0.5$ to the origin, and (b) from the initial state $x_1(0) = 4$, $x_2(0) = 4$ to the origin. Both cases are subject to the constraint that $-1 \leq u \leq 1$. This problem is more difficult than the double integrator because the system has complex roots. When the roots are complex, the system may switch many times before it reaches the desired state. This fact makes it difficult to select a good initial guess on the number and location of the switching times. The correct solution for the harmonic oscillator problem can be determined analytically, but the initial guesses in this example were selected arbitrarily to examine how the algorithm would handle a poor initial control trajectory.

Case (a)

The initial control was selected to be $u(0) = 1$, and the initial switching-time and control vectors were selected to be

$T = [\ .5\ \ 1\ \ 2\ \ 3\ \ 4\ \ 4\ ]^T$, and $S = [\ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ ]^T$.

Using this initial guess, the algorithm converged to zero cost at

$T = [\ .624423 \quad .662762 \quad 2.672467 \quad 2.672467 \quad 3.844459 \quad 5.259266\ ]^T$.

A new guess was obtained by eliminating the two switching times which cancel each other. The modified guess was

$$T = [\ .5 \quad 1 \quad 4 \quad 4\ ]^T, \text{ and } S = [\ 1 \quad 1 \quad 1 \quad 0\ ]^T.$$

The switching-time vector converged to

$$T = [\ .725320 \quad .725320 \quad 3.883204 \quad 5.265403\ ]^T.$$

In this result, the first two switches cancel, effectively leaving one switch during the control interval. Recall that the initial control was assumed to be plus one. To test this assumption, the initial control was modified to u(0) = -1 and an additional switch was placed near the origin. The initial guess selected was

$$T = [\ .5 \quad 2.5 \quad 4.\ ]^T, \text{ and } S = [\ 1 \quad 1 \quad 0\ ]^T.$$

Figure 5.1 illustrates the components of the switching-time vector at each iteration for this initial guess. The algorithm converged in six iterations to

$$T = [\ 0.322683 \quad 3.566979 \quad 4.997713\ ]^T.$$

Therefore, the correct initial control is probably u(0) = -1 because the first switching time did not move to the origin and the final time decreased. The state and control trajectories for this solution are illustrated in Figure 5.2.

Case (b)

The initial guess selected for case (b) was u(0) = -1,

$$T = [\ 2 \quad 4 \quad 6 \quad 8\ ]^T, \text{ and } S = [\ 1 \quad 1 \quad 1 \quad 0\ ]^T.$$

The switching-time vector converged to

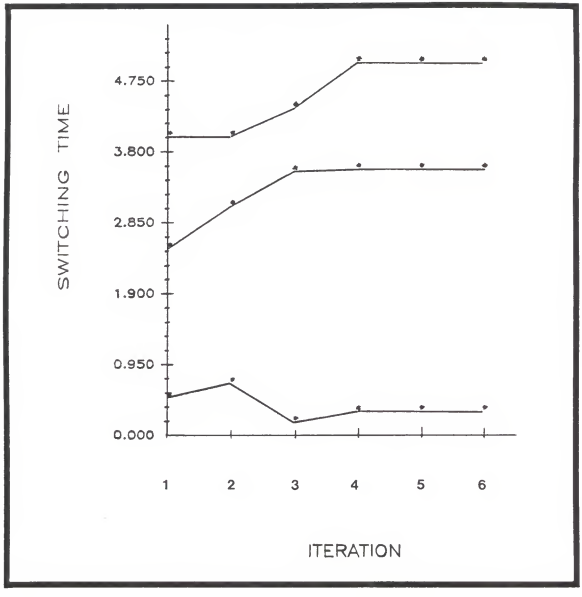


Figure 5.1 Switching-time vector at each iteration for the harmonic oscillator: case (a)

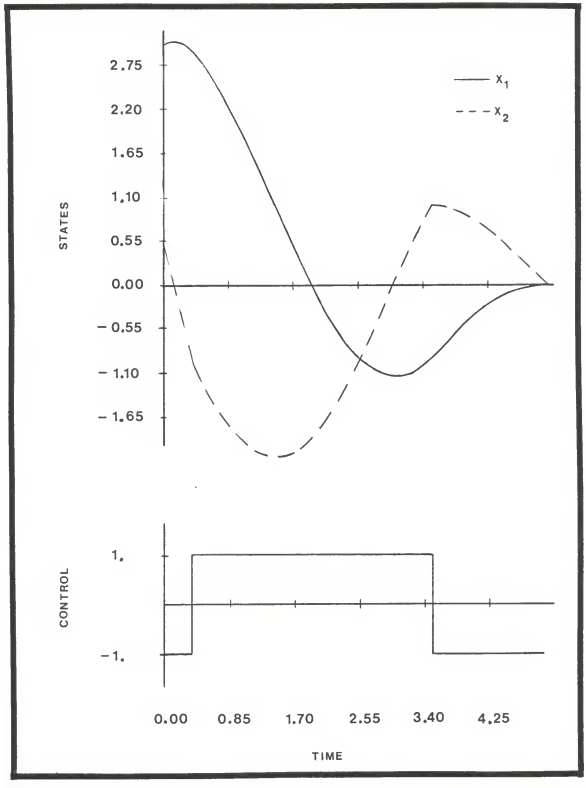


Figure 5.2 State and control trajectories for the harmonic oscillator: case (a)

$$T = [\ .911926 \quad 3.94743 \quad 7.03059 \quad 9.031150\ ]^T.$$

If the initial switching-time vector was modified to

$$T = [\ .5 \quad 2 \quad 6 \quad 8\ ]^T,$$

the algorithm converges to

$$T = [\ 1.011 \quad 3.358 \quad 7.330 \quad 9.566\ ]^T.$$

This result has a higher final time than the first answer. Therefore, the initial solution is probably close to the time-optimal control history. The third initial guess was obtained by slightly perturbing the first answer. The modified initial guess was $u(0) = -1$,

$$T = [\ .95 \quad 4.0 \quad 6.5 \quad 8.5\ ]^T, \text{ and } S = [\ 1 \quad 1 \quad 1 \quad 0\ ]^T.$$

The algorithm converged in six iterations to

$$T = [\ .81309 \quad 4.01257 \quad 7.0659 \quad 9.02061\ ]^T$$

The switching-time vector at each iteration for this initial guess is shown in Figure 5.3. The state and control trajectories for this solution are illustrated in Figure 5.4. Further perturbations of the solution did not result in any significant improvement in the final time.

The bang-bang solution for the harmonic oscillator can be determined analytically. Oldenburger and Thompson [8] illustrate the switching surface for this system. This surface consists of a series of semicircles along the x_1 axis. The semicircles lie below the axis when x_1 is positive and above the axis when x_1 is negative. These semicircles are described by the equation

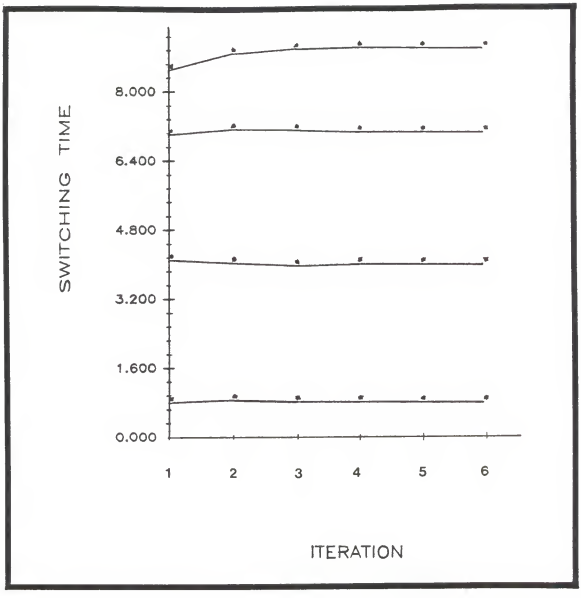


Figure 5.3 Switching-time vector at each iteration for the harmonic oscillator: case (b)

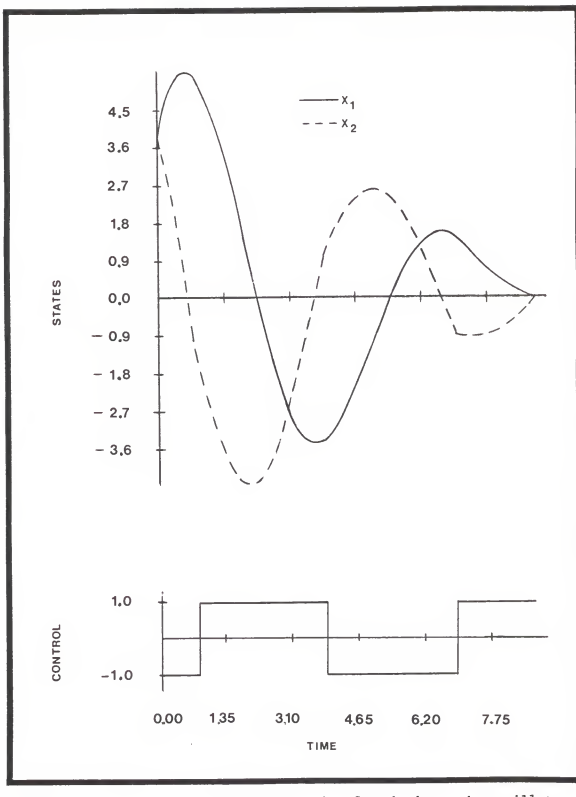


Figure 5.4 State and control trajectories for the harmonic oscillator: case (b)

$$x_2 = -\left|\frac{x_1}{x_1}\right|\left[\, 1 - (x_1 - b)^2\right]^{1/2} \tag{5.2}$$

where b is the odd integer which is closest in magnitude to $x_1(t_i)$. Here t_i represents the ith switching time. For example, if $0 < |\, x_1 \,| < 2$, then $b = 1$, and if $2 < |\, x_1 \,| < 4$, then $b = 1$. The optimal control is given by

$$u = \begin{cases} -1 \text{ if } x_2 > -\left|\dfrac{x_1}{x_1}\right|\left[\, 1 - (x_1 - b)^2\right]^{1/2} \\ +1 \text{ if } x_2 < -\left|\dfrac{x_1}{x_1}\right|\left[\, 1 - (x_1 - b)^2\right]^{1/2} \end{cases} \tag{5.3}$$

The initial control for both case (a) and case (b) is $u(0) = -1$. Following the same procedure as in Example 2.1, the state equations were integrated forward in time from the initial time to the first intersection with the switching curve. Using equation (5.2) the first switching time was calculated. The equations of motion were then integrated forward in time to the next intersection with the switching curve, and so on. Using this procedure, the correct switching-time vector for case (a) was found to be

$$T = [\ .3739 \quad 3.5155\ ]^T \text{ with } S = [\ 1 \quad 0\ ]^T.$$

For case (b), the correct switching-time vector is

$$T = [\ .83 \quad 3.97 \quad 7.11 \quad 8.93\ ]^T \text{ with } S = [\ 1 \quad 1 \quad 1 \quad 0\ ]^T.$$

The results for case (a) identically match the results obtained using the numerical algorithm. However, the results for case (b) are slightly different. A possible explanation is that the state trajectories are fairly insensitive to changes in the switching times

except at the final switch. This fact is illustrated in Figure 5.4. Even though the numerical solution is suboptimal, it still provides a good approximation to the true time-optimal control.

Example 5.2 - The R-Theta Manipulator

Shetty [9] has examined the time-optimal control of an r-theta manipulator. Figure 5.5 illustrates the r-theta manipulator. The equations of motion presented by Shetty for this system are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 x_4^2 \\ x_3 \\ -2x_2 x_4 / x_1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1/x_1^2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \qquad (5.4)$$

In equation (5.4), x_1 and x_3 correspond respectively to r and θ in Figure 5.5, and x_2 and x_4 correspond respectively to $\dot{r}$ and $\dot{\theta}$. Shetty has solved this problem using an iterative techniqe involving the initial values of the adjoint equation and a discrete-time finite-element method. The initial conditions used were $x_1(0) = 1$, $x_2(0) = 0$, $x_3(0) = 0$, and $x_4(0) = 0$. The desired final conditions were $x_1(t_f) = 1$, $x_2(t_f) = 0$, $x_3(t_f) = \pi/2$, and $x_4(t_f) = 0$. Using both methods, the final time obtained was $t_f = 1.9644$.

The initial guess selected for the constrained switching-time minimization was $u_1(0) = 1$, $u_2(0) = 1$,

$T = [\ 0\ \ .5\ \ 1\ \ 0\ \ .5\ \ 1\ \ 1\ ]^T$, and $S = [\ 1\ \ 1\ \ 1\ \ 2\ \ 2\ \ 2\ \ 0\ ]^T$.

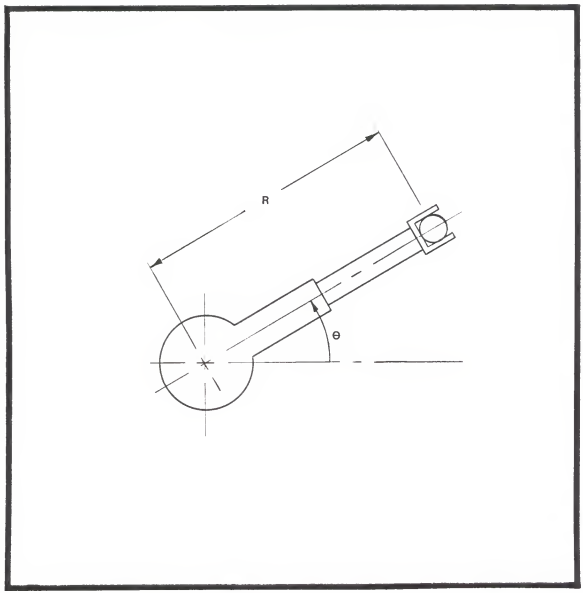


Figure 5.5 The r-theta manipulator

Figure 5.6 illustrates the switching-time vector at each iteration for these initial conditions. The algorithm converged in nineteen iterations to

$$T = [\ 0 \quad .8657122 \quad 1.098707 \quad .335025 \quad .335025 \quad .982207 \quad 1.964415\ ]^T,$$

$$\text{and } S = [\ 1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 2 \quad 0\ ]^T.$$

This is the first example with more than one control input. In Figure 5.6, note how the switching times for the same control are constrained while the switching times for different controls are free to cross over each other. Figure 5.7 illustrates the state and control trajectories for this problem.

Example 5.3 - Satellite

The equations of rotational motion for a satellite are given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_4 + s_2(s_1x_5 + c_1x_6)/c_2 \\ c_1x_5 - s_1x_6 \\ (s_1x_5 + c_1x_6)/c_2 \\ I_4x_5x_6 \\ I_5x_4x_6 \\ I_6x_4x_5 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$\begin{aligned} \text{where } I_4 &= -.280000, \\ I_5 &= .470588, \\ I_6 &= -.219512, \\ B_1 &= 9.6000 \times 10^{-3}, \\ B_2 &= 8.8235 \times 10^{-3}, \\ \text{and } B_3 &= 7.3171 \times 10^{-3}. \end{aligned} \tag{5.5}$$

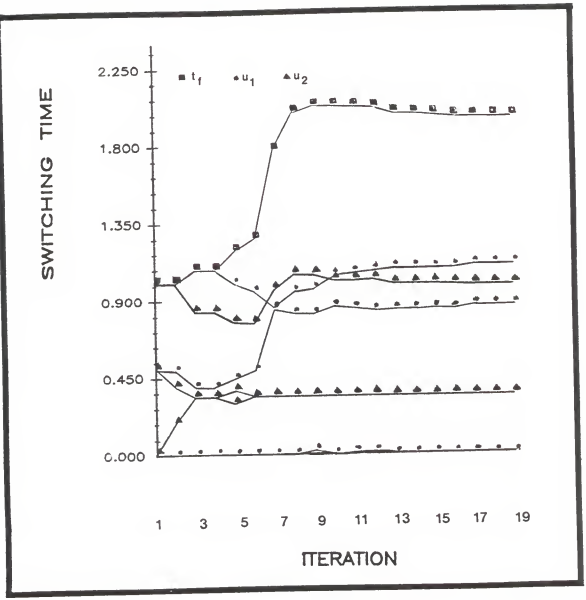


Figure 5.6 Switching-time vector at each iteration for the r-theta manipulator

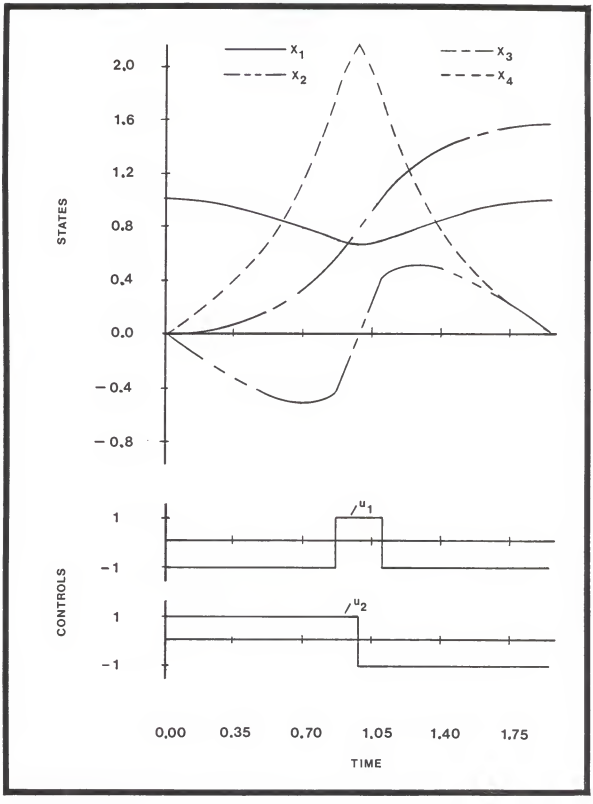


Figure 5.7 State and control trajectories for the r-theta manipulator

In this equation s_i and c_i respectively denote the sine and cosine of the ith component of X.

The problem is to drive the system to the origin in minimum time from (a) $x_1(0) = x_2(0) = x_3(0) = .1$, $x_4(0) = x_5(0) = x_6(0) = 0$, and (b) $x_1(0) = x_2(0) = x_3(0) = 1$, $x_4(0) = x_5(0) = x_6(0) = 0$.

Case (a)

The initial guess used was $u_1 = 1$, $u_2 = 1$, $u_3 = 1$, $T = [\ 0 \quad .5 \quad 1 \quad 0 \quad .5 \quad 1 \quad 0 \quad .5 \quad 1 \quad 1\ ]^T$, and

$$S = [\ 1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 2 \quad 3 \quad 3 \quad 3 \quad 0\ ]^T.$$

Figure 5.8 illustrates the switching times and final time at each iteration. The switching-time vector converge to

$$T = [\ 0 \quad 3.153295 \quad 6.791658 \quad 0 \quad 3.486751 \quad 7.052664 \quad 0 \quad 3.580353 \quad 7.222408 \quad 7.222408\ ]^T$$

Figure 5.9 illustrates the the positions and controls at the solution for case (a), and Figure 5.10 illustrates the velocities and controls for the same problem.

Case (b)

From the results of case (a), it appears that the correct initial controls should all be negative. Therefore, the initial controls for case (b) were selected to be $u_1 = -1$, $u_2 = -1$, and $u_3 = -1$. The initial guesses on the T and S vectors were

$$T = [\ 4 \quad 8 \quad 4 \quad 8 \quad 4 \quad 8 \quad 8\ ]^T, \text{ and}$$

$$S = [\ 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 3 \quad 0\ ]^T.$$

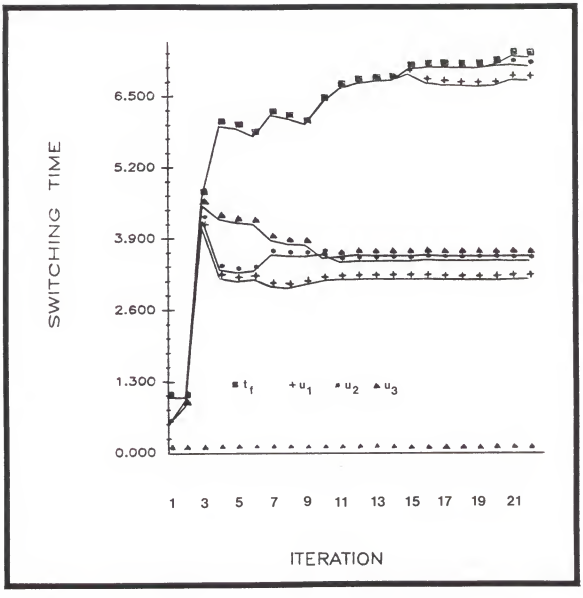


Figure 5.8 Switching-time vector at each iteration for the satellite: case (a)

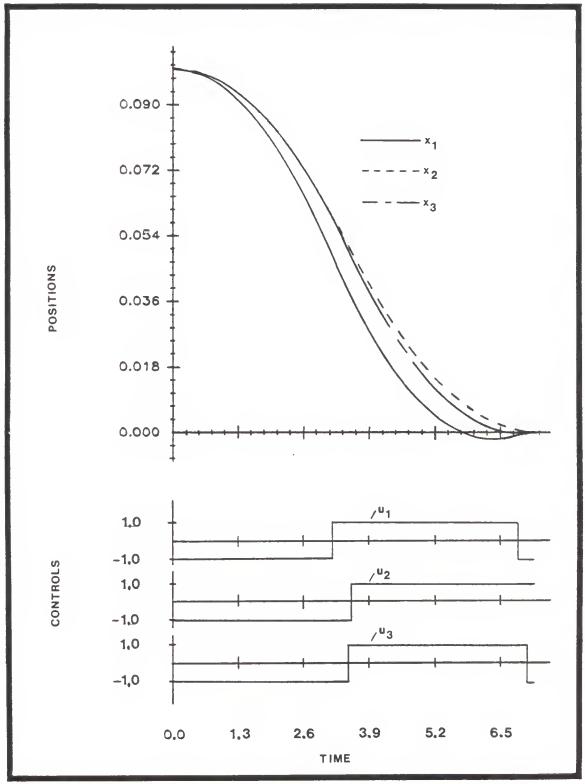


Figure 5.9 Position and control trajectories for the satellite: case (a)

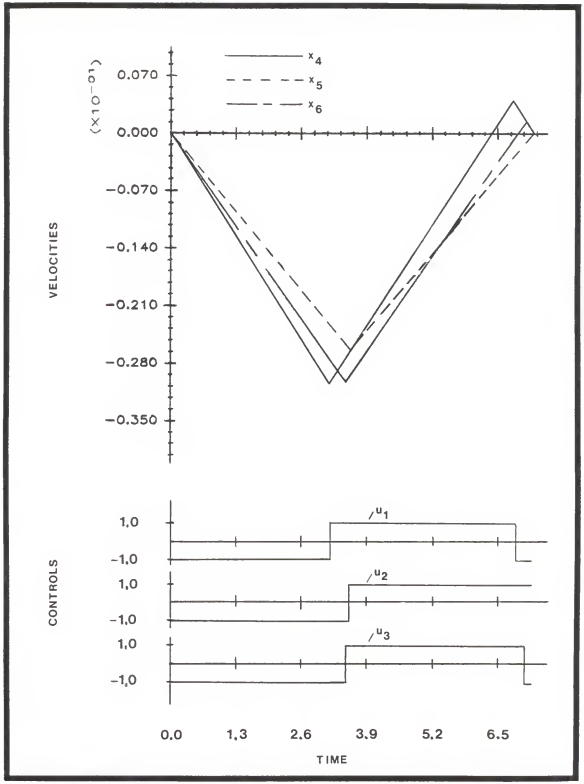


Figure 5.10 Velocity and control trajectories for the satellite: case (a)

The iteration history for this problem is illustrated in Figure 5.11. The solution converged to

$T = [\ 6.341065 \quad 18.610118 \quad 12.147811 \quad 23.274307 \quad 9.097048 \quad 20.953685 \quad 23.274317\ ]^T$.

The position coordinates and the controls are shown in Figure 5.12, and the velocity coordinates are shown in Figure 5.13.

Example 5.4 - Double Pendulum Manipulator

The last example in this chapter is a double pendulum manipulator. Figure 5.14 illustrates this system. The vector $\vec{g}$ is the gravitation vector with magnitude $g = 9.81\ m/s^2$. The variables l_1 and l_2 represent the link lengths of the arm. These parameters are given by $l_1 = l_2 = .5$ meters. Not shown in Figure 5.14 are the mass and inertia parameters. These are given by $m_1 = 50kg$, $m_2 = 30kg$, $I_1 = 5kg \cdot m^2$, and $I_2 = 3kg \cdot m^2$.

Sahar and Hollerbach [12] have examined the time-optimal control of this system. They adopted a linear programming approach in which the joint space is discretized into a grid. The program searchs the grid to find the minimum-time path between the initial and final states. An example they presented is the movement from the initial states $x_1(0) = 0$, $x_2(0) = 0$, $x_3(0) = 0$, and $x_4(0) = 0$ to the desired final states $x_1(t_f) = -\pi/3$, $x_2(t_f) = 0$, $x_3(t_f) = 2\pi/3$, and $x_4(t_f) = 0$. The solution they obtained is not bang-bang because the actuator torques are only on the constraint boundaries for part of the control interval. They reported a final time of $t_f = .525$ seconds.

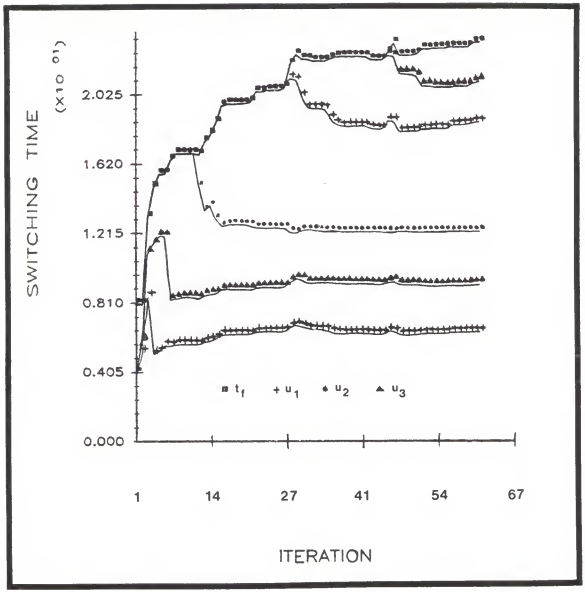


Figure 5.11 Switching-time vector at each iteration for the satellite: case (b)

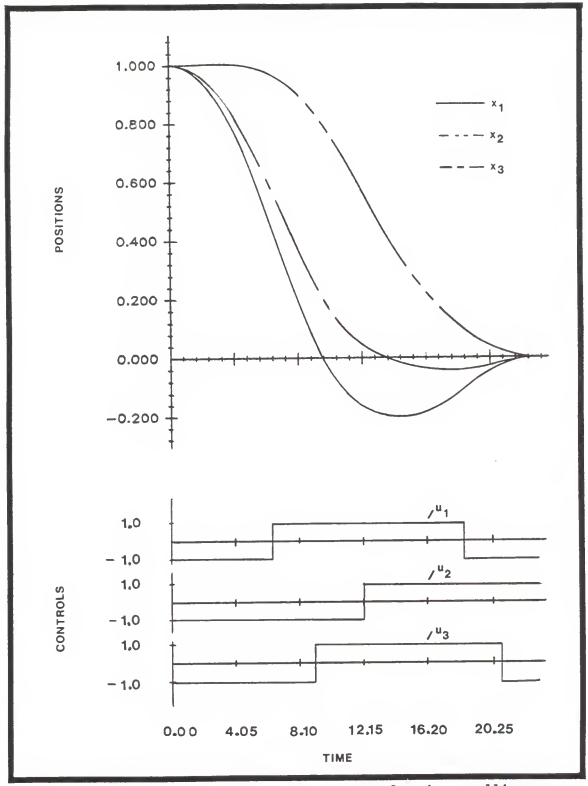


Figure 5.12 Position and control trajectories for the satellite: case (b)

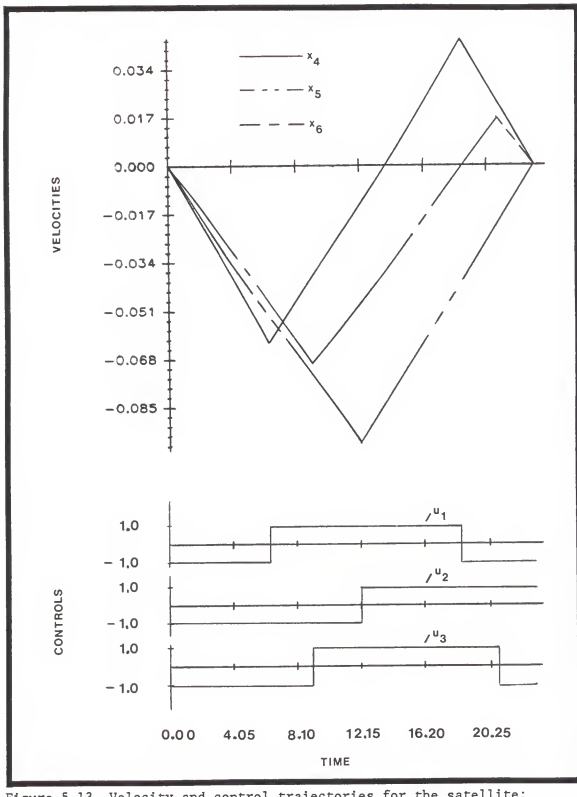


Figure 5.13 Velocity and control trajectories for the satellite: case (b)

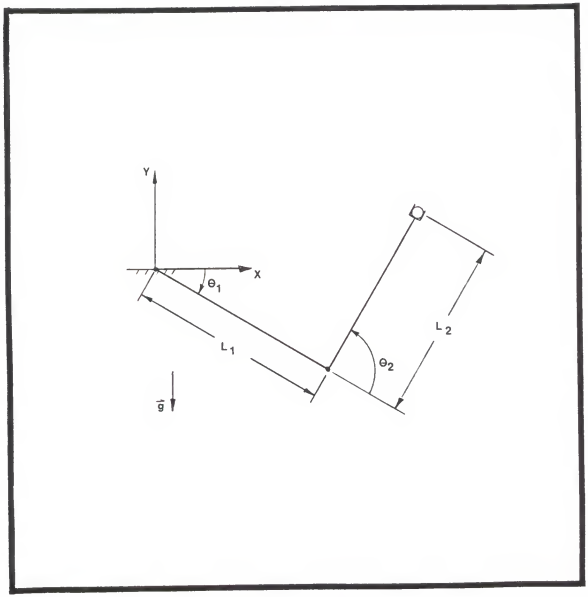


Figure 5.14 The double pendulum manipulator

The equations of motion for this system are

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & M_{11} & 0 & M_{12} \\ 0 & 0 & 1 & 0 \\ 0 & M_{12} & 0 & M_{22} \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 \\ H_1 \\ 0 \\ H_2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{5.6}$$

where

$$M_{11} = I_1 + I_2 + \tfrac{1}{4}(m_1 l_1^2 + m_2 l_2^2) + m_2 l_1^2 + m_2 l_1 l_2 c_2,$$

$$M_{12} = I_2 + \tfrac{1}{4} m_2 l_2^2 + \tfrac{1}{2} m_2 l_1 l_2 c_2,$$

$$M_{22} = I_2 + \tfrac{1}{4} m_2 l_2^2,$$

$$H_1 = -m_2 l_1 l_2 s_2 (\tfrac{1}{2} x_4^2 + x_2 x_4) + [\tfrac{1}{2} m_2 l_2 c_{12} + l_1 (\tfrac{1}{2} m_1 + m_2) c_1] g,$$

and $$H_2 = \tfrac{1}{2} m_2 l_2 l_1 s_2 x_2^2 + \tfrac{1}{2} m_2 l_2 c_{12} g.$$

In order to get the equations of motion into the standard state matrix form, the mass matrix on the left-hand side must be inverted. Carrying out the inversion, the equations of motion become

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \frac{1}{M_{11}M_{22} - M_{12}^2} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & M_{22} & 0 & -M_{12} \\ 0 & 0 & 1 & 0 \\ 0 & -M_{12} & 0 & M_{11} \end{bmatrix} \begin{bmatrix} 0 \\ H_1 \\ 0 \\ H_2 \end{bmatrix} \tag{5.7}$$

$$+ \frac{1}{M_{11}M_{22} - M_{12}^2} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & M_{22} & 0 & -M_{12} \\ 0 & 0 & 1 & 0 \\ 0 & -M_{12} & 0 & M_{11} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

This example was run for two different cases. In case (a), the manipulator was assumed to operate in the vertical plane, and the gravitational constant was set equal to 9.81. In case (b), the manipulator was assumed to operate in the horizontal plane, so the gravitational constant was set equal to 0.

Case (a)

The initial values of the controls were selected to be

$$u_1(0) = 350, \text{ and } u_2(0) = 100.$$

The initial vectors T and S were

$$T = [\ .1 \quad .2 \quad .1 \quad .2 \quad .3\ ]^T, \text{ and } S = [\ 1 \quad 1 \quad 2 \quad 2 \quad 0\ ]^T.$$

The switching-time vector converged to

$$T = [\ .00043 \quad .05456 \quad .31850 \quad .45317 \quad .46030\ ]^T.$$

The algorithm took 161 iterations to reach zero cost. Figure 5.15 illustrates the switching-time vector at each iteration for these initial conditions. Figure 5.16 illustrates the state and control trajectories at the solution. Next, the initial controls were changed to

$$u_1(0) = -350, \text{ and } u_2(0) = 100,$$

and the vectors T and S were changed to

$$T = [\ .05 \quad .32 \quad .45 \quad .46\ ]^T, \text{ and } S = [\ 1 \quad 2 \quad 2 \quad 0\ ]^T.$$

For these initial conditions, the switching-time vector converged to

$$T = [\ .054107 \quad .318956 \quad .452966 \quad .459974\ ]^T.$$

In this result, the last switching time and the final time are almost equal. Based on this observation, the final switch was eliminated from

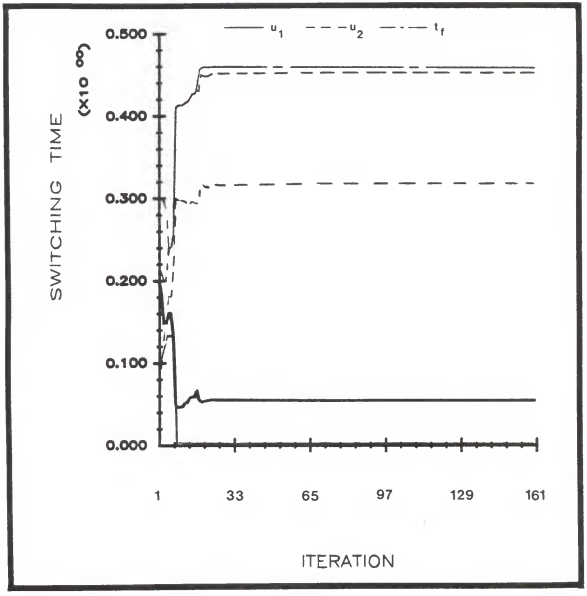


Figure 5.15 Switching-time vector at each iteration for the double pendulum manipulator: case (a)

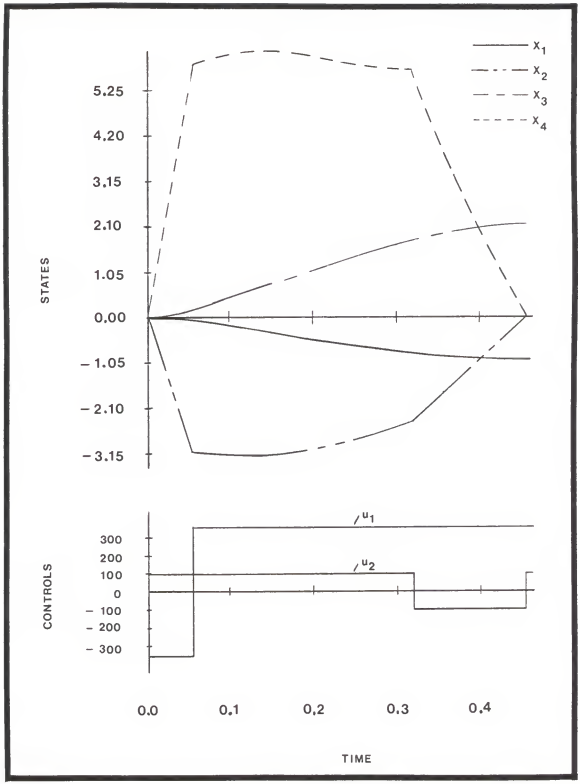


Figure 5.16 State and control trajectories for the double pendulum manipulator: case (a)

the next initial guess. However, when the last switching time was removed, the algorithm would not converge. Therefore, the last switch may be necessary.

Case (b)

The optimal control for this case was expected to be similar to the results obtained with gravity. Therefore, the initial controls were selected to be $u_1(0) = -350$, and $u_2(0) = 100$. The initial values of T and S were picked to be

$$T = [\ .1 \quad .15 \quad .2 \quad .3\ ]^T, \text{ and } S = [\ 1 \quad 2 \quad 2 \quad 0\ ]^T.$$

The algorithm converged in 18 iterations to

$$T = [\ .126 \quad .182 \quad .368 \quad .368\ ]^T$$

Figure 5.17 illustrates the switching-time vector at each iteration for this problem. Figure 5.18 shows the converged state and control trajectories.

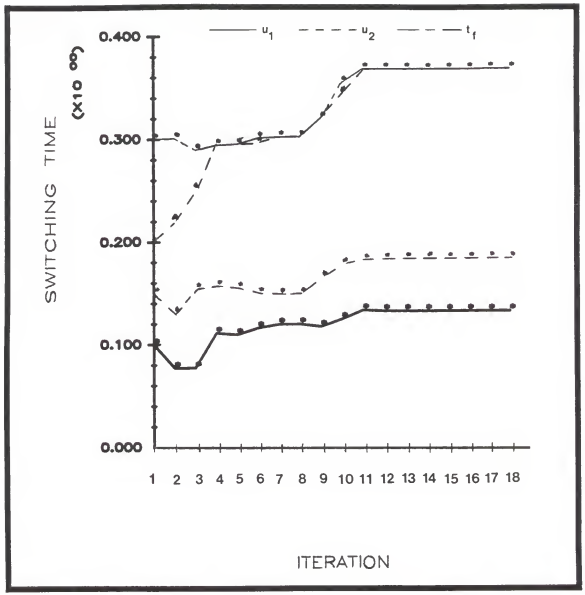


Figure 5.17 Switching-time vector at each iteration for the double pendulum manipulator: case (b)

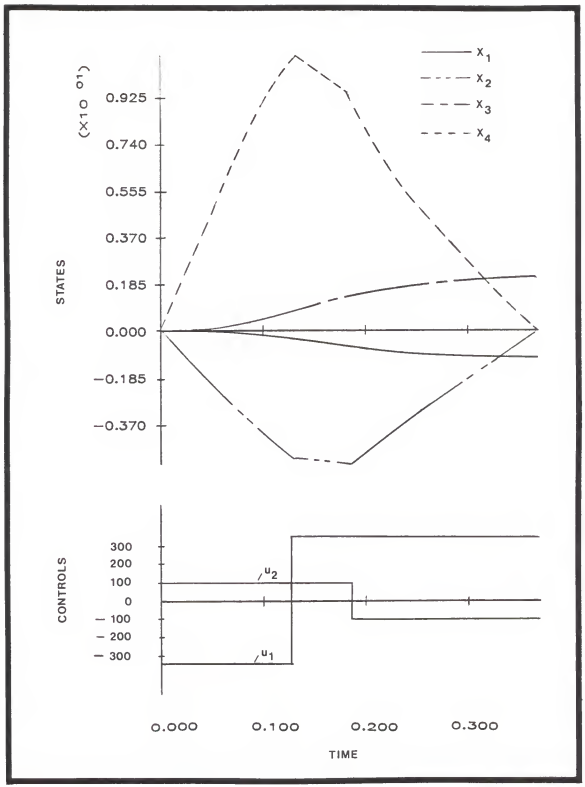


Figure 5.18 State and control trajectories for the double pendulum manipulator: case (b)

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

Conclusions Concerning the Constrained Switching-Time Iteration Method

The objective of this study has been to develop a reliable method of determining the bang-bang control sequence which will drive a system from some initial state to a desired final state. In this regard, the constrained switching-time iteration method has been very successful. In every example, the iteration process converged and drove the cost function to zero. Therefore, the objective of transferring the system from the initial state to the final state was met.

The inclusion of constraints on the switching times produced three desirable effects. First, the constraints eliminate the problems with switching times which move out of the control interval or cross over each other. The figures showing the switching-time vector at each iteration clearly illustrate how the constraints restrain the switching times. The result is a more robust algorithm. Second, because the constrained switching-time iteration method is more robust, the initial guesses on the number of switches, the initial control settings, and the initial switching times can be far from correct. The example problems show that the switching times will usually move close to their final values within a few iterations. Often, extra switches will move together and effectively cancel out. Third, the inclusion of

constraints clarifies the theory behind the gradient search algorithm by emphasizing the geometric concept of a switching-time space.

The computational speed of the algorithm is at least comparable with other iterative time-optimization methods. For example, the satellite problem and the double pendulum manipulator problem both took approximately 15 minutes of CPU time running on a Harris H-800 computer. In contrast, Sahar and Hollerbach [12] reported that their algorithm required up to several hours of computation.

Another appealing feature of the constrained switching time iteration method is its generality. Linear and nonlinear systems are treated identically. The equations developed in Chapter Four allow the technique to be applied to systems derived using Lagrange's equations. The only significant limitation is that the system equations cannot be explicit functions of time. However, the equations could probably be modified to remove this limitation.

The drawback of the constrained switching-time iteration method is that the algorithm does not actually minimize the final time. Chattering solutions can result if too many switches are initially used. Therefore the user must interactively modify the initial guesses in order to find the time-optimal solution. Experience gained working with the algorithm suggests that the difficulty in obtaining the time-optimal solution is very dependent on the state equations. For example, the double integrator and r-theta manipulator examples would produce the time-optimal trajectory even with poor initial guesses. On the other hand, the time-optimal control for the harmonic oscillator was very difficult to obtain even when the initial guesses were close to the

correct solution. One observation was that chatter was much less likely if the initial guess of the final time was picked less than the true final time.

Recommendations for Further Study

There are two major areas that merit further study. First, the cost function used in the constrained switching-time algorithm should be examined. Perhaps the cost could be modified to include some of the necessary conditions for a time-optimal control given by Pontryagin's minimum principle. Adding additional cost terms would help to eliminate the problem of chattering.

Second, it may be possible to use discrete versions of the differential equations presented in Chapter Two to develop a feedback control scheme. This type of system would allow for real-time control as opposed to the trajectory planning approach developed in this paper.

LITERATURE CITED

1. Kirk, D.E., Optimal Control Theory, Englewood Cliffs, NJ: Prentice-Hall Inc., 1970.

2. Bryson, A.E., and Ho, Y.C., Applied Optimal Control, John Wiley & Sons, 1975.

3. Bushaw, D.W., "Optimal Discontinuous Forcing Terms," In: Contributions to the Theory of Nonlinear Oscillations, Vol. IV, pp. 29-52, Princeton University Press, 1958.

4. Bellman, R., Glicksberg, I., and Gross, O., "On the 'Bang-Bang' Control Problem," Quarterly of Applied Mathmatics, Vol. 14, pp. 11-18, 1956.

5. LaSalle, J.P., "The Time Optimal Control Problem," In: Contributions to the Theory of Nonlinear Oscillations, Vol. V, pp. 1-24, Princton University Press, 1960.

6. Pontryagin, L.S., Boltyanski, V.G., Gamrelidze, R.V., and Mischenko, E.F., The Mathmatical Theory of Optimal Processes New York: Interscience, 1962.

7. Desoer, C.A., "The Bang-Bang Servo Problem Treated by Variational Techniques," Information and Control, Vol. 2, No. 4, pp. 333-348, 1959.

8. Oldenburger, R., and Thompson, G., "Introduction to Time Optimal Control of Sationary Linear Systems," Automatica, Vol. 1, pp. 177-205, 1963.

9. Shetty, A., "A Solution Technique for the Minimum-Time Control Problem of an R-Theta Manipulator," M.S. Thesis in Mechanical Engineering, Kansas State University, April, 1987.

10. Subrahmanyam, M.B.,"A Computational Method of the Solution of Time-Optimal Control Problems by Newton's Method," International Journal of Control, Vol. 44, No. 5, pp. 1233-1243, 1986.

11. Shin, K.G., and McKay, N.D., "Minimum-Time Trajectory Planning for Industrial Robots with General Torque Constraints," Proceedings of the 1986 IEEE International Conference on Robotics and Automation, pp. 412-417, April, 1986.

12. Sahar, G., and Hollerbach, J.M., "Planning of Minimum Time Trajectories for Robot Arms," Internal Journal of Robotics Research, Vol. 5, No. 3, pp. 90-100, Fall, 1986.

13. Rajan, V.T., "Minimum Time Trajectory Planning," Proceedings of the 1985 IEEE Conference on Robotics and Automation, pp. 759-764, April, 1985.

14. Knudsen, H.K., "An Iterative Procedure for Computing Time-Optimal Controls," IEEE Transactions on Automatic Control, Vol. AC-9, No. 1, pp. 23-30, January, 1964.

15. Lastman, G.L., "A Shooting Method for Solving Two-Point Boundary-Value Problems Arising from Non-Singular Bang-Bang Optimal Control Problems," International Journal of Control, Vol. 27, No. 4, pp. 513-524, 1978.

16. Lasdon, L.S., Mitter, S.K., and Waren, A.D., "The Conjugate Gradient Method for Optimal Control Problems," IEEE Transactions on Automatic Controls, Vol. AC-12, No. 2, pp. 132-138, 1967.

17. Lewine, R.N., and Thorp, J.S., "Computation of Time-Optimal Controls Using a Second-Variation Descent Search," IEEE Transactions on Automatic Control, Vol. AC-15, No. 3, pp. 358-362, June, 1970.

18. Larson, V.H., "Minimum Time Control by Time Interval Optimization," International Journal of Control, Vol. 7, No. 4, pp. 381-394, 1968.

19. Yastreboff, M., "Synthesis of Time-Optimal Control by Time Interval Adjustment," IEEE Transactions on Automatic Control, Vol. AC-14, No. 6, pp. 707-710, December, 1969.

20. Davison, E.J., and Monro, D.M., "A Computational Technique for Finding Time Optimal Controls of Nonlinear Time-Varying Systems," Proceeding of the Joint Automatic Controls Conference, pp. 270-280, 1969.

21. Wen, J.T., and Desrochers, A., "An Algorithm for Obtaining Bang-Bang Control Laws," ASMR Journal of Dynamic Systems, Measurement, and Control, Vol. 109, No. 2, pp. 171-175, June, 1987.

22. Siddall, J.N., Optimal Engineering Design, New York: Marcel Dekker, Inc., 1982.

23. Rosen, J.B., "The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints," SIAM Journal, Vol. 8, No. 1, pp. 181-217, March, 1960.

DETERMINATION OF BANG-BANG CONTROLS FOR LARGE NONLINEAR SYSTEMS

by

DAVID DONALD NIEMANN

B.S., Kansas State University, 1986

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

MECHANICAL ENGINEERING

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

ABSTRACT

This paper develops an iterative method of determining the bang-bang control sequence which will drive a system from some initial state to a desired final state. A cost function measures the error between the actual and desired states at the final time. The algorithm minimizes cost by iteratively adjusting the switching times using the gradient projection method. This procedure is viewed as a constrained minimization in switching-time space. The method is applicable to both linear and nonlinear systems with multiple control inputs. Under most circumstances, the resulting trajectories will be time optimal or near time optimal. Results are presented for five systems: a double integrator, a harmonic oscillator, an r-theta manipulator, a satellite, and a double pendulum manipulator. Equations are developed which allow this method to be used with any dynamic system derived using Lagrange's equations.